Íslenzka

MySQL gagnagrunnsbókin

um MySQL 5.0

Elías Ívarsson

ÍSLENZKA MYSQL GAGNAGRUNNSBÓKIN (MYSQL 5.0)

WWW.ISBOK.IS

ISBN 9979-9583-5-9

Íslenzka MySQL gagnagrunnsbókin

Um MySQL útgáfu 5.0.

Elías Ívarsson

Handritsútgáfa: 12. maí 2005

ISBN 9979-9583-5-9

Menntamálaráðuneytið hefur styrkt ritun og útgáfu þessarar bókar.

EFNISYFIRLIT

ÆFINGASKRÁ

1. Kafli. Inngangur

Þessi bók

Bók þessi er til orðin af þörf á íslenzkri kennslubók í SQL[1] gagnagrunnsfræði. Mikið er til af hugbúnaði sem byggir á þess konar tækni og sýnist sitt hverjum um hvaða búnaður er bestur. Slík umræða verður eftirlátin mér hæfara og vitrara fólki. Deila má t.d. um hvort biðlara/miðlara tól á borð við Oracle og SQL Server fylgi betur SQL staðlinum frekar en MySQL.

Ástæðan fyrir því að ég valdi MySQL sem viðfangsefni var sú að kerfið er vinsælt í vefsíðugerð auk annarra þátta: MySQL kostar ekkert fyrir námsfólk og þar sem búnaðurinn tilheyrir opnum lind-um (e. Open Source) er auðvelt bæði að nálgast hugbúnaðinn og upplýsingar um samsetningu hans.

Um mitt árið 2003 sótti ég um og fékk ég styrk frá Menntamálaráðuneytinu til verksins en vegna annarra verkefna dróst að ég hefði tíma til að ljúka verkinu það ár. Úr því rættist sumarið 2004 að ég hafði aðstæður til að sinna því af þeirri athygli sem verkið verðskuldar.

Aðaláhersla er lögð í bókinni á gagnagrunnsfræði og SQL málið. Því sterkari sem notandinn er í notkun þessara fræða, því betur á hann með að tileinka sér þau kerfi sem byggja á þeim. Fullyrði ég af reynslu minni í kennslu að því meir sem nemandinn tekst hlutlægt á við þessi fræði breikkar hann og dýpkar getu sína og færni án þess að vera háður einu vörumerki öðru fremur.

Bókina ber því fyrst og fremst að lesa sem gagnagrunnsbók með MySQL sem fararskjóta. Fyrsti kafli bókarinnar einblínir á gagnagrunns hugsunarhátt og þau hugtök sem þar að lúta. Því næst er stutt yfirferð sem stiklar á stóru með sýnidæmum sem allir gagnagrunns notendur og hönnuðir þurfa að skilja og kunna. Næstu kaflar þar á eftir er handbók og uppflettirit yfir þætti sem MySQL styður.

Þar sem MySQL nýtur mikilla vinsælda í gagnvirkri vefsíðugerð var freistandi að blanda saman umfjöllun um MySQL og PHP forritun í eina bók. Slíkar bækur eru algengar erlendis. Ég tók hins vegar þá afstöðu að einblína á gagnagrunns tæknina.

Alla bókin út í gegn hef ég reynt að nota íslensk hugtök eins og þau eru þekkt í Tölvuorðasafni Íslenskrar málnefndar. Alls staðar þar sem því er viðkomið hef ég sett tilsvarandi ensk hugtök í sviga aftan við notkun íslenska hugtaksins. Þetta er gert bæði til glöggvunar fyrir þá lesendur sem lesa sambærilegar bækur á ensku eða MySQL handbókina sem gefin er út á ensku.

Sumum kann að þykja blöndun enskra hugtaka óþægileg við lestur en ég trúi að það muni auðvelda notendum að tengja efni bókarinnar við aragrúa ítarefnis og bóka sem er fáanlegt á ensku máli, bæði um MySQL svo og gagnagrunnsfræði almennt. Hef ég einnig skellt neðanmáls athugasemdum víðsvegar í bókinni með beinum útskýringum á hugtökum (með tilvísun í ensku orðin) sem fletta má upp í Tölvuorðasafninu.

Það er gott að eiga Tölvuorðasafnið og fletta upp í. Bókin styður við vöxt íslenskrar tungu í tölvutækni. Einnig má fletta upp í orðasafninu á vefsetrinu http://ismal.hi.is/ob/birta.

Bið ég lesandann velvirðingar ef eitthvað má betur fara eða skilar sér illa. Ég vil hvetja lesandann til að senda tölvuskeyti á elias@isbok.is með athugasemdum, af hvaða tagi sem er. Athugasemdir sem verða til þess að bæta næstu útgáfa bókarinnar ættu að skila sér þangað. Sömuleiðis hvet ég lesandann til að fylgjast með á www.isbok.is varðandi ítarefni, villuleiðréttingar og annað.

---

[1] SQL er stytting enska hugtaksins Structured Query Language. Tölvuorðasafnið þýðir þetta þannig: [enska] **SQL** [íslenska] **SQL** [**skilgr.**] Heiti á fyrirspurnarmáli fyrir töflugagnasöfn. [**skýr.**] SQL er stytting á "Structured query language".

Umfjöllun bókarinnar miðast við MySQL 5.0. Ef einhverju er haldið fram varðandi eiginleika (e. Features) búnaðarins á það við um þessa útgáfu nema annað sé tekið fram. Allir kótar sem birtast í bókinni hafa verið prófaðir og framkvæmdir í Windows útgáfu MySQL og flestir í Linux útgáfunni. Á báðum verkvöngum (e. Platform) var notuð „MySQL 5.0 Alpha." Helstu undantekningar frá prófuðum kótadæmum eru kótar í kaflanum um innbyggð föll en flest dæmanna þar má einnig finna í handbók MySQL. Öll sýnidæmi eru hinsvegar mínar eigin niðurstöður og prófanir.

SQL staðallinn

SQL staðallinn er fyrst og fremst tungumál. Tungumálið er málskipun (e. Syntax) sem skilgreinir málfræði og rithátt sem forritari getur notað til vinnslu með gagnasöfn. Skipta má SQL málinu í tvo þætti: Annars vegar gagna-skilgreiningarmál og hins vegar gagna-meðhöndlunarmál.

Gagna-skilgreiningarmál, á ensku Data Definition Language eða DDL, eru notuð til að skilgreina gagnageymslur. Gagnageymslur fjalla um hvernig gagnagrunnurinn eða gagnasafnið skuli uppbyggt. Þessi uppbygging nær til skilgreiningar á töflum, gagnategundum innan taflanna og notkunarréttindi s.s. hver má lesa gögn, hver má skjóta inn gögnum og hver má breyta þeim.

Gagna-meðhöndlunarmál, á ensku Data Manipulation Language eða DML er málfræði til að meðhöndla upplýsingar. Þessi málfræði tiltekur hvernig gögn eru send inn í töflur og sóttar úr þeim, upplýsingunum breytt og hvernig þeim er eytt.

SQL staðallinn er því vel útfærð og hnitmiðuð málfræði sem nærri allir framleiðendur gagnasafns-umsjónarkerfa hafa orðið ásáttir um. Í dag eru það stofnanirnar ANSI (American National Standards Institute) og ISO (International Standards Organisation) sem gefa staðalinn út. Fjöldi framleiðanda hafa kosið að fylgja staðlinum en flestir ef ekki allir gera það á sinn eigin hátt. Þannig má læra SQL í hvaða SQL-samhæfðu kerfi sem er og á skjótum tíma tileinka sér næsta kerfi því þekkingin er yfirfæranleg.

SQL92, SQL99, SQL2, SQL3

Oft er talað um að SQL staðallinn sem gefinn var út árið 1989 sé SQL 1 og útgáfan frá 1992 sé SQL 2. Árið 1999 kom út SQL 3. Hvaða útgáfu miðlarinn styður er ávallt álitamál því flest gagnasafns-umsjónarkerfi styðja að nær öllu leyti við SQL 1, að mestu leyti við SQL 2 og mörg styðja við undirmengi úr SQL 3. Almennur notandi sér aldrei þessa staðla að fullu og fæstir (almennir notendur) hafa ástæðu til að kaupa staðalinn frá framangreindum stofnunum.

Þeir sem kaupa staðalinn eru framleiðendur gagnasafns-umsjónarkerfa sem velja að styðja við það sem þeim hentar í honum. Salatbar ekki satt? Hvað um það, þá styðja allir nægilega vel við SQL til að sá sem kann eða skilur almennar málreglur hans getur auðveldlega tileinkað sér kerfin.

Biðlari/miðlari

MySQL er hraðvirkur léttavigtar gagnagrunns miðlari[2] sem byggir á miðlara/biðlara hugmynd. Búnaðurinn verkar á þá leið, að venslaðir gagnagrunnar eru smíðaðir og uppsettir á MySQL þjóni. Þjónninn miðlar þeim síðan til notenda á öðrum vélum (eða á sömu vél).

Oft er á þessum vettvangi rætt um „three-tier," „two-tier" og „multi-tier" umhverfi. Er þá átt við þegar notandi er á einu sviði (e. Tier), gagnagrunnur á öðru og jafnvel annar grunnur eða t.d. vinnslugeta á því þriðja. Er umfjöllun um þessi „tier" málefni utan efnissviðs bókarinnar (e. Out of Scope).

[2] Enska hugtakið Server er á íslensku notað ýmist sem þjónn eða miðlari. Bæði orðin merkja hið sama en í bók þessari eru þau notuð bæði og hafa þá alltaf sömu merkingu.

MySQL þjónninn styður við forritunarskil fyrir forritunarmálin C, Perl, Java (með Java DataBase Connectivity API eða JDBC) og Python. Þá má tengjast þjóninum með aðstoð MyODBC tengingar sem er, ODBC tenging sérhönnuð fyrir MySQL en samhæfð ODBC staðlinum frá Microsoft.

Með blöndun þessara tæknitóla (e. Technologies) má smíða fullvaxnar biðlara/miðlara (e. Client/Server) lausnir á borð við gagnagrunnstengd vefsvæði án mikils kostnaðar (hlutfallslega). Sé þá verð hug-búnaðarins, og umhaldsvinna, borin saman við sambærileg (og jafnvel öflugri) kerfi sem eru vinsæl á markaðnum.

Þó MySQL kosti lítið, og í sumum tilfellum ekki neitt, þá er ekki allt fengið þar með. Þó nafn forritsins gefi annað til kynna þá styður það ekki fullkomlega við SQL gagnagrunnsstaðalinn í heild sinni, frekar en svo mörg önnur kerfi. Við sumar aðstæður gæti gagnagrunns hönnuðurinn þurft á að halda víðtækari lausnum auk þess sem sumar útfærslur krefjast meiri vinnu við kótun en önnur og dýrari kerfi bjóða.

MySQL gagnagrunns þjónninn hefur frá upphafi verið hannaður með það í huga að skila eins hraðvirkri vinnslu og framast er unnt og í samanburðar prófunum við ýmis umfangsmeiri kerfi kemur hann mjög vel út. Vegna þess að frá upphafi skipti hraði og áreiðanleiki meiru fyrir höfunda kerfisins frekar en hversu mikla SQL eiginleika (e. Features) breidd kerfið studdi, hefur þjónninn frá upphafi stutt SQL staðalinn aðeins að hluta en gert það vel. Fæstum finnst þetta koma að sök því framkvæma má nánast allt í MySQL umhverfinu sem SQL staðallinn krefst og vafamál er hvort nokkur miðlari styðji fullkomlega við SQL.

Lengi vel var til dæmis enginn stuðningur við framandlykla (e. Foreign key) í MySQL gagnagrunnum, eða þar til útgáfa 3.23.44 kom út og þá því aðeins að töflur í grunninum væru af tegundinni InnoDB. Fyrir suma gagnagrunnsforritara þykir stuðningur við framandlykla afar mikilvægur því hann getur létt vissa vinnu af forritaranum og auðveldað vissar tegundir af gagnagrunns hönnun. MySQL AB hefur reyndar lofað því að fullur stuðningur verði kominn við framandlykla í útgáfu 5.1.

Staðreyndin er sú að til að framfylgja þeim kröfum sem framandlyklar setja á gagnagrunninn í það og það sinnið, þarf miðlarinn að erfiða meira en annars. Þetta eykur biðtíma og álag við gagnavinnslu. Hraði og skilvirkni undir álagi er einmitt mjög mikilvægt þegar komið er í nettengda gagnagrunns vinnslu af því tagi sem MySQL er hannaður fyrir.

Algengt er að MySQL miðli grunnum sem innihalda milljónir færslna og oft þarf að miðla gögnum þessara grunna til hundruða notenda samtímis í rauntíma. Við slíkar aðstæður skiptir hraði og skilvirkni mun meira máli en hvort forritari þurfi að eyða hálfum vinnudegi í að tryggja ákvæði (e. Constraints) svo sem framandlykla í forritunar kóta sínum.

Margir forritarar álíta að framandlykla ákvæðin eins og „cascade on delete“ og „cascade on update“ megi framkvæma með forritunar-rökum (e. Programming Logic) Höfundar MySQL fylgja þeirri stefnu. Eftir því sem vinsældir MySQL hafa færst í aukanna hafa kröfur um víðtækari möguleika af þessu tagi og víðtækari stuðning við SQL staðalinn orðið háværari og með hverri útgáfu-uppfærslunni (e. Version upgrade) á fætur annarri hefur nýjum eiginleikum verið bætt inn.

Er það yfirlýst stefna MySQL AB að innan tíðar muni miðlarinn bjóða fullkominn stuðning við SQL staðalinn. Má geta þess t.d. að miðlarinn skilur nær allar SQL aðgerðir en þær sem ekki njóta stuðnings ennþá eru hunsaðar í vinnslu. Í umræðu af þessu tagi mætti þó íhuga að í upphafi var miðlarinn ekki hannaður sem söluvara heldur sem öflugur gagnagrunns þjónn sem gæti unnið samkvæmt þörfum höfundanna. Hins vegar sló búnaðurinn í gegn og í dag er talinn vinsælasti gagnagrunns hugbúnaðurinn sem gefinn er út undir merkjum opinna linda[3].

[3] Enska hugtakið „Open Source“ hefur ekki verið þýtt ennþá á íslensku þó oft sé talað um opinn hugbúnað þá myndi það merkja „Open Software“ sem er skylt hugtak en ekki hið sama. Höfundur hefur því kosið að nota hugtakið Opnar lindir.

Með MySQL útgáfu 5 var bætt við stuðningi við „vistaðar stefjur“ (e. Stored Procedures) sem leyfa hönnuðum að skrifa sínar eigin stefjur og kveikjur[4] (e. Triggers) sem geta brugðist við ákveðnum atvikum eða ástandi í grunnunum sem eiga þessar stefjur. Stuðningur við framandlykla var kominn í útgáfu 3.23 (að hluta) og stuðningur við undir-fyrirspurnir (e. Sub Selects) kom í útgáfu 4. Svo hröð virðist þróunin vera að það taki því varla að taka fram í svona bók hvað þjóninn skorti, svo hratt er nýjum eiginleikum bætt við. Notendur kynnast því hvort sem er fljótt hvað skortir í svona kerfum. Notendahandbókin sem fylgir MySQL, telur yfir 1200 síður í dag, og er eytt miklu púðri þar í að kenna notendum að sniðganga og gera hið besta úr því sem miðlarann skortir.

Einn styrkur MySQL felst í því að hver sem er getur farið á vefsetrið www.mysql.com og niðurhalað kerfinu. Þar má finna útgáfur fyrir öll þau helstu stýrikerfi sem eru notuð í tölvunotkun samtímans. Þannig getur hver sá sem óskar eftir öflugu gagnasafnsumsjónarkerfi í miðlara/biðlara umhverfi, s.s. hönnuðir gagnvirkra vefsíðuna, sótt sér ódýrt og í sumum tilfellum frítt kerfi sem er öruggt, hraðvirkt og býr yfir víðtækum eiginleikum er henta fyrir allflestar aðstæður. Þeir sem óska öflugri kerfa s.s. Oracle eða Microsoft SQL Server geta síðar keypt þessi kerfi og með tiltölulega lítilli kótun flutt grunna sína yfir. Í mörgum tilfellum er nóg fyrir þessa sömu aðila að bíða í nokkrar vikur, í mesta lagi fáeina mánuði, eftir næstu uppfærslu á MySQL.

Þegar ég kynnti mér MySQL fyrst var útgáfa 3.23 sú nýjasta og hraus mér hugur að lesa um allt það sem skorti. Innan tíðar, þegar ég hafði tileinkað sér notkun kerfisins saknaði ég einskis, auk þess sem þetta er allt að koma eða þegar komið.

Saga MySQL

Saga MySQL er að nokkru leyti tengd sögu mSQL gagnagrunnsþjónsins og má rekja til baka til ársins 1994. Á þeim tíma, þegar notendur þörfnuðust RDBMS[5] (e. Relational DataBase Management System) kerfa var fátt í boði. Helstu vörumerki þess tíma voru Oracle, Sybase og Informix. Þessi kerfi sem öll kostuðu og kosta enn skildinginn voru hönnuð til að miðla gríðarmiklu magni gagna og oft innan flókinna vensla (skyldleika).

Öll þessi kerfi kosta einnig mikið í vinnslu (e. Resource intensive). Bæði er að kerfin eru kostnaðarsöm í notendaleyfum auk þess að til þurfti öflugustu vélar undir þau og dugði þá ekki 100 Mhz 486 tölva (á þeim árum voru Pentium 60 Mhz rétt að koma inn á sjóndeildarhringinn) sem þó var dágóður kostur á þeim tíma. Helsti tölvukosturinn voru tölvur á borð við Digital Alpha RISC System, Sun SPARC eða sambærilegar vélar sem eingöngu voru notaðar með UNIX stýrikerfum oft sérhönnuðum fyrir hvern verkvang (e. Platform) fyrir sig.

Stór fyrirtæki og stærri háskólar áttu auðvelt með að eyða milljónum í notendaleyfi og vélbúnað en smærri aðilar og einstaklingar höfðu um fátt að velja nema þá helst veigaminni skjáborðsforrit (e. Desktop Programs). Þó eitthvað væri í boði af ódýrum miðlara/biðlara lausnum þá studdu þau síður við SQL staðalinn en buðu frekar eigin lausnir. Áberandi meðal þessara ódýrari kerfa var Postgres sem þróaðist frá sömu rótum og Ingres sem var einskonar dýr stóri bróðir. Bæði kerfin byggðu gagnagrunns vinnslu sína og samskipti notenda (eða notendaforrita) við gagnagrunna sína á QUEL málinu. Postgres studdist við undirmengi (e. Subset) þess máls sem nefnist PostQUEL.

Um sama leyti var nokkuð um að skjáborðsforrit á borð við Borland Paradox notuðu QBE málið sem er stytting hugtaksins „Query By Example.“ Ekki skyldi þó rugla því máli við SQL staðalinn sem er stöðluð

[4] Íslenska tölvuorðasafnið hefur ekki ennþá komið með tillögu á íslenskun enska gagnagrunns-hugtaksins „Trigger.“ Ég mun notast við orðið „Kveikja“ í sömu merkingu.

[5] Database Management System er þýtt sem gagnasafns-umsjónarkerfi. Relational er venslað.

málskipan fyrir samskipti á milli forrits annars vegar og gagnasafns-umsjónarkerfis hins vegar. QBE er myndrænt valmál sem auðveldar venjulegu fólki að veiða gögn úr gagnasafni. Síðar var QBE málið útfært betur í myndrænum fyrirspurnum í Microsoft Access sem þýðir þær yfir í SQL aðgerðir.

David Hughes

Á þessum tíma var David Hughes að vinna að þróun kerfisins „Minerva Network Management System." Starf hans var hluti undirbúningsnáms til meistaragráðu (Ph.D.) í upplýsingakerfum við háskóla í Ástralíu. Innan síns kerfis studdist hann við notkun gagnagrunns er geymdi upplýsingar um netkerfi sem Minerva átti að stjórna. Þar eð hann hafði ekki greiðan aðgang að neinu hinna öflugri kerfa sem á í boði voru á þeim tíma studdist hann við Postgres.

Hughes komst að þeirri niðurstöðu að Minerva þyrfti öflugan SQL stuðning svo ytri kerfi gætu tengst innri grunni þess og nálgast upplýsingar úr honum. SQL staðallinn myndi opna Minerva netupplýsingakerfið út á við en PostQUEL málið sem Postgres notaði hentaði ekki. Í dag styður Postgres reyndar SQL staðalinn. Reiptogið sem leiddi af þessari þarfagreiningu setti hann í erfiða aðstöðu því eina leiðin fyrir hann til að styðja við SQL staðalinn var að kaupa fokdýrar lausnir frá Oracle, Sybase eða Informix.

Millilendingin var að smíða SQL þýðanda (e. Translator) inn í Minervu sem gæti gripið (e. Intercept) öll SQL köll til kerfisins og þýtt þau yfir í PostQUEL sem Postgres vélin (e. Engine) í Minervu gæti þá svarað. Þessi lausn var kölluð miniSQL eða mSQL.

PostQUEL verður að RDBMS kerfi

Um sinn virtist framangreind lausn henta vel þar sem Minerva var notað til kerfisumhalds. Ekki skipti máli hvaða gagnasafns-umsjónarkerfi (DBMS) voru notuð til gagnavinnslu utan kerfisins, svo fremi að þau styddu við SQL. Frá sjónarhóli Minervu gat Postgres undirkerfið stutt við SQL köll því þar á milli lá mSQL til að þýða SQL aðgerðir (e. Statements) yfir í PostQUEL.

Því miður virtist þessi lausn fremur óhentug til langframa því eftir því sem Minerva kerfið var notað við stærri kerfi, og eftir því sem þróun kerfisins sjálfs miðaði áfram og varð umfangsmeiri, kom í ljós að Postgres og ýmis umfangsmeiri gagnasafns-umsjónarkerfi sinntu ekki nægilega hinu litla eiginleika-mengi (e. Feature Set) sem Minerva krafðist, sérstaklega ekki innan þeirra takmörkuðu vélbúnaðar linda (e. Hardware Resources) sem í boði voru.

Meðal annars krafðist Minerva þess að gagnagrunnur sinn gæti sinnt mörgum samhliða gagnagrunns-tengingum (e. Simultanious Database Connections). Postgres þjónninn gat því aðeins stutt þessa kröfu að keyrð væru mörg tilvik (e. Instance) hans í einu. Algengt er með þjóna í miðlara umhverfi að þeir geti sett í gang mörg ferli (e. Processes) sjálfs sín í senn. Þetta er svipað því að vefrápari, t.d. Internet Explorer, getur opnað fleiri glugga í einu til að sækja vefsíður, hver gluggi er þá sjálfstætt ferli eða tilvik sama forritsins.

Um þetta leiti kemur í ljós að hugsanlegir þátttakendur í Minerva verkefninu urðu undan að snúa þar sem þeirra kerfi gátu ekki átt samskipti við Postgres þjóninn og þessir sömu þátttakendur höfðu ekki ráð á að kaupa og setja upp hin öflugri kerfin.

Hughes var kominn í þá stöðu að hann þurfti að endurskoða afstöðu sína til Postgres. Hann var þegar kominn með mSQL inn í kerfið sem sinnti öllum fyrirspurnum sem Minerva sinnti. Allflestar voru þær af taginu INSERT, DELETE og SELECT setningar[6] (e. Statements). Postgres kerfið bjó þá þegar yfir miklu

[6] Statement er ýmist notað sem aðgerð, setning eða skipun. Tölvuorðasafnið mælir með orðinu aðgerð og oft er sú leið farin í bókinni. Ýmsar bækur nota hugtakið „segð" við svipaðar aðstæður en það orð er að hverfa. Sumar bækur nota orðið segð (setning) þar sem þessi bók notar orðið „yrðing" sem er enska orðið Expression samkvæmt Tölvuorðasafni.

stærra eiginleikamengi en mSQL og var full umsvifamikið fyrir Minervu sem krafðist einfaldara undirmengis. Í raun þurfti Hughes aðeins að bæta við mSQL þeim eiginleika að geta geymt og sótt gögn í eigið kerfi og væri hann þá kominn með sinn eigin gagnagrunns miðlara. Þessi þróun leiddi til þess hugbúnaðar sem mSQL hefur þróast til í dag.

Monty

Þó nöfn þessara tveggja kerfa séu svipuð þá væru mistök að líta á MySQL sem beint svar við eiginleika-skorti mSQL kerfisins eða aðra útgáfu af sama kerfi. Þetta eru ekki sömu kerfin þó saga þeirra sé tengd í upphafi. Upprunalegur höfundur MySQL, Michael „Monty“ Widenius, hafði unnið við gagnagrunnsþróun hjá sænska fyrirtækinu TcX frá árinu 1979 og meðal annars þróað þar gagnasafns-umsjónarkerfið UNIREG til notkunar innanhúss. UNIREG hefur síðan ’79 verið umskrifað oft og í mismunandi forritunarmálum. Kerfið getur í dag unnið með stór gagnasöfn[7].

Árið 1994 hóf TcX þróun á vefværum hugbúnaðarlausnum og notaði UNIREG gagnagrunna sem gagnatrog (e. Data Storage). UNIREG hins vegar krafðist mikillar vélbúnaðar yfirbyggingar til að geta miðlað kröftugum vefsíðum (e. Dynamic Webpages) á skilvirkan hátt. TcX leit þá til SQL staðalsins varðandi útfærslu en lenti á sama vegg og David Hughes hafði áður rekist á. mSQL (útgáfa 1.x) studdi t.d. ekki við atriðaskrár (e. Index) og var því enn hægvirkari og slappari en UNIREG.

Atriðaskrár eru undirtöflur á gagnagrunnstöflur. Ef við til dæmis lítum á nafnatöflu sem geymir lista yfir alla einstaklinga búsetta í Bandaríkjunum þyrftum við töflu fyrir u.þ.b. 400.000.000 færslur. Slík tafla þyrfti að geyma dálka með upplýsingum um kennitölur (í Bandaríkjunum væri það Social Security Number), nöfnum einstaklinga, gilt heimilisfang, fæðingardag, trúarflokk, kyn, og sitthvað fleira. Eins og gefur að skilja getur slík tafla orðið nokkuð stór og mjög tafsamt að leita í henni.

Atriðaskrá gæti þá verið undirtafla sem geymir hverja einustu kennitölu og númer þeirrar færslu, eða línu, þar sem hún finnst í aðaltöflunni. Þegar leitað væri eftir kennitölum mætti leita í undirtöflunni sem væri mjög viðbragðsfljót. Þá þyrfti aðeins að sækja upplýsingar úr þeim línunúmerum sem atriðaskráin vísar á. Gagnasafns-umsjónarkerfi geyma slíkar atriðaskrár þannig að notandi sér þær ekki. Hönnuður töflunnar skilgreinir að taflan skuli byggja atriðaskrá, þ.e. Index, á tiltekin svið töflunnar og kerfið sér sjálfkrafa um framangreinda hegðun.

Monty hafði samband við Hughes varðandi hvort sá síðarnefndi hefði hug á að tengja mSQL við B+ höndlarann sem UNIREG byggði á. B+ á hér við „B-tree“ gagnageymslu aðferð (e. Datastorage Method). Þetta er nálgun til geymslu gagna á diski og ósýnileg forriturum sem vinna beint með gagnasafns-umsjónarkerfið eða forritun biðlara. B-tree er ein möguleg (og vinsæl) lausn á geymslu gagna sem innviðir gagnagrunnskerfa nota þegar gögn eru vistuð á diskinn og langt, langt utan efnissviðs bókarinnar.

Hughes var á þessum tíma langt kominn með útfærslu sinna eigin hugmynda um uppbyggingu atriðaskráa fyrir mSQL 2. TcX tók því þá ákvörðun að endurhanna sinn eigin gagnagrunns miðlara fyrir eigin þarfir. Við þá þróun þótti TcX mönnum óþarfi að enduruppgötva hjólið. Þeir byggðu hið nýja kerfi að einhverju leyti á UNIREG þróunarreynslunni og miðuðu grunnþróun sína að miklu leyti við fjölda tækja og tóla fyrir mSQL sem í vaxandi mæli komu fram hjá þriðju-aðilum. Þannig urðu forritunar-skilin (e. Application Programming Interface eða API[8]) hjá TcX gagnagrunns miðlaranum samhæfð við mSQL forritunar-skilin.

---

[7] Gagnasafn er stór upplýsingabingur og getur verið óskipulagður, gagnagrunnur er hins vegar gagnasafn sem hefur verið skipulagt. Gagnasafns-umsjónarkerfi er hugbúnaður s.s. MS Access eða MySQL sem leyfir hönnuði að breyta gagnasafni í gagnagrunn. Einfalt?

[8] Tölvuorðasafnið hefur ekki gefið út tillögu á þýðingu þessa hugtaks svo ég nota hér hugtakið forritunar-skil til bráðabirgða.

Forritarar sem höfðu kótað gagnagrunnsvinnslu og tengingar við mSQL gátu því fært sín tól yfir á TcX lausnina með lágmarks fyrirhöfn.

Fyrsta gagnagrunns tól TcX var „Screen builder/report“ tól forritað í BASIC. Þetta er á þeim tíma þegar fínustu „State-Of-the-Art“ tölvur höfðu 4Mhz Z80 örgjörva með 16KB[9] vinnsluminni. Tólið var snemma fært (e. Ported) yfir á UNIX verkvang og þróað eitthvað áfram þar. Um miðjan níunda áratuginn heyrðust óánægjuraddir frá viðskiptavinum. Þó notendum líkaði vel við forritið þá vildu þeir eitthvað sem þeir könnuðust við. Svo TcX hóf leitina að vinsælu tískuyrði eða söluhugtaki (e. Buzzword).

Hönnuðirnir hjá TcX litu þá til SQL, tískuyrðis sem átti vaxandi vinsældum að fagna og gæti hentað sem framhlið (e. Front-end) á lágtækni stefjusöfnin (e. Low Level Libraries) sem þeir notuðu. Við fyrstu sýn virtist sem mSQL gæti hentað en við frekari eftirgrennslan og rannsóknir var ljóst að þörf var á að kóta SQL vél (SQL Engine) frá grunni. Hins vegar reyndust forritunarskilin hjá mSQL mjög hentug sem framhlið á hina nýju gagnagrunnsvél, eins og áður hefur komið fram, svo þeim var viðhaldið. Þetta gerði þeim kleift að nota ýmis gagnagrunns tól sem áður höfðu verið þróuð til samskipta við mSQL. Greinilega höfðu þeir farið þá leið að geyma grunna í mSQL vél og hannað forrit til samskipta við hana en skiptu svo út gagnagrunns-vélinni án þess að skipta út forritunum. Þetta hefur gert þeim kleyft að spara sér heilmikla vinnu. Eins og fyrir algjöra tilviljun stóðu þeir uppi með söluhæfan gagnagrunns miðlara.

Nú var MySQL gagnagrunns vélin orðin samhæf við vinnslu hjá miklu fleiri aðilum en bara TcX og þeirra viðskiptavinum því margir notuðu mSQL sem grunn og eigin tól (biðlara) til samskipta við hann, rétt eins og TcX hafði gert. Þessir aðilar gátu því fyrirhafnarlaust skipt út gagnagrunnsvélinni en haldið áfram notkun eigin tóla sem höfðu verið þróuð beinlínis til samskipta við mSQL. Því var tekin sú ákvörðun að gefa MySQL út í samræmi við hugmyndir Peter Deutsch hjá Aladdin Enterprises, sem hafði áður gefið út Ghostscript. Hugmyndin virkar þannig að útgefandi á allan höfundar- og útgáfurétt og leyfir öðrum að nota forritið svo fremi þeir selji það ekki sem hluta af sinni lausn.

Héðan er komið hugtakið opinn-hugbúnaður (e. Open Software); allir mega nota og dreifa en ekki selja. Opnar lindir (e. Open Source) er þegar uppruna-kóti (eða frumkóti) forritsins er einnig opinn og öðrum frjáls að aðlaga til eigin þarfa án þess að mega selja til þriðja aðila. Hugtakið frír-hugbúnaður (e. Free Software) er þegar eigandi hugbúnaðarins gefur hann á sama hátt og opin-hugbúnað en opnar ekki endilega lindirnar. Deilihugbúnaður (e. Shareware) er hugbúnaður sem dreift er frítt, án lindanna, og mælst er til að notendur greiði lágt gjald ef þeir kjósa að nota hann umfram tilskilin tímamörk.

Höfundar MySQL og Monty sjálfur segjast ekki vita hvaðan nafngiftin „My“ kemur en minnast má hér á að My er mjög algengt nafn meðal sænsku mælandi Finna. Monty er einmitt sænskur Finni og oft er nefnt í þessu samhengi að dóttir hans, sem fæddist um líkt leyti heitir My. Nýlega gaf MySQL AB út MaxDB sem er byggt á sama grunni og eins og fyrir tilviljun nefnist sonur Montys einmitt Max. Þess má geta að TcX grunnskrárnar og mikið af söfnunum (e. Libraries) hafa lengi haft formerkið „My“ og nafngiftin gæti rétt eins verið þaðan komin.

Í maí 1995 kemur MySQL 1.0 út en var aðeins dreift meðal fjögurra aðila en fáeinum mánuðum síðar í október 1995 kemur MySQL 3.11.1 út í tvíundar-skrá (e. Binary File) fyrir Solaris stýrikerfið frá Sun Microsystems. Mánuði síðar kom út Linux tvíundarskrá ásamt uppruna-kóta (e. Source Code) og ýmsum MySQL biðlaraforritum (e. Clients).

Þeir félagar Michael „Monty“ Widenius, David Axmark frá Detron HB ásamt Allan Larsson stofnuðu þá fyrirtækið MySQL AB sem enn í dag er aðaleigandi höfundar- og útgáfuréttar á MySQL kótanum. David

[9] Ég fylgi þeirri ritvenju að stórt b (B) tákni bæti í skammrituðum minnisstærðum, stundum geng ég yfir strikið og rita Bæti með stórum upphafsstaf þar sem því er við komið. Þetta er gert til aðgreiningar á litlu b sem táknar bita.

Axmark sem skrifaði fyrstu útgáfu skjölunarsafns (e. Documentation) grunnsins hvatti þá félaga til að gefa kótann út á Netinu undir merkjum samvinnu-hugbúnaðar (e. Co-operative software).

Samvinnu-hugbúnaður er í samsett hugtak fyrir opinn-hugbúnað og opnar-lindir. Forritið er sett í opna dreifingu og allir geta sótt það og prófað. Allir geta sömuleiðis sótt og lesið uppruna-kótann auk þess sem útgefendur taka tillit til þeirra tillagna og uppástungna sem koma til baka frá notendum. Þannig taka notendur þátt í að auka gæði búnaðarins.

Samvinnu-kótun af þessu tagi er viðskiptalíkan sem L. Peter Deutsch hjá Aladdin systems í Bandaríkjunum notaði við útgáfu á Ghostscript. Peter hefur efni á að setjast í helgan stein í dag. Hugmyndin gengur út á að kótinn fyrir hugbúnaðinn er gefinn út t.d. á Netinu. Hver sem vill getur lesið kótann, lagað að eigin þörfum, þýtt (eða túlkað) fyrir eigin verkvang og notað. Gera má ráð fyrir að unnendur þessa kóta/hugbúnaðar endurgefi (e. Feedback) breytingar sínar og lagfæringar, sem geti, hljóti þær samþykki eigenda, skilað sér áfram í endurbættum og uppfærðum útgáfum kótans.

Þegar þetta er ritað er MySQL útgáfa 5 nýkomin á sjóndeildarhringinn, sem þýðir að „Alpha“ útgáfan er komin í dreifingu á Netinu og víðsvegar í notkun. Jafnan þegar ný uppfærsla kemur út er sett Alpha útgáfa á Vefinn. Yfirleitt eru þetta nokkuð traustar útgáfur en að vissum tíma liðnum er sett Beta útgáfa sem er þá nokkuð reynd. Eftir nokkurn reynslutíma kemur svo lokaútgáfa. Hver sem vill getur sótt hugbúnaðinn beint af vefsvæði MySQL AB í Svíþjóð, eða gegnum einhvern tuga spegla (e. Mirrors) sem staðsettir eru víðsvegar í heiminum (m.a. á Íslandi). ´

Bæði er hægt að sækja frumkótann (uppruna-kótann) til að lesa, þýða og keyra eða þýddar tvíundar útgáfur sem setja má beint upp á einhverjum eftirtalinna verkvanga: Dec OSF, SGI Irix , OpenBSD, IBM AIX, Mac OS X, FreeBSD, Solaris, Windows, Linux AMD64, Linux x86, Linux IA64.

Verkvangar og prófanir

Grunnútgáfa MySQL virkaði eingöngu á Linux og Solaris og stærsti vandinn við að færa hann yfir á aðra verkvanga (e. Platform) var að þjónninn þurfti þrædd POSIX[10] söfn (e. Threaded POSIX libraries). Í janúar 1997 kom út breytt (e. Modified) útgáfa af MIT-pthreads með kótanum, sem í rauninni er endurbætt POSIX þráðaforritun.

Til þess að nota MySQL frá öðrum forritunarmálum þarf forritunarskil (API) og frá upphafi hefur miðlarinn verið gefinn út með C og Perl forritunarskilum. Síðan hafa komið út fleiri söfn (e. Libraries) sem öll utan Java safnsins byggja á C söfnum. Slíkt veitir forriturum greiðan aðgang að því að smíða eigin forrit og biðlara útgáfur sem nota MySQL grunna. Til dæmis eru PHP forritunarskilin sem vinsæl eru í gagna-grunnsmiðlun á Vefnum byggð á C safninu. Sá sem þekkir til notkunar PHP safnsins mun þekkja notkun þess ef sá sami vildi forrita aðgang að grunnum sinum í C málinu.

Þegar kerfið var orðið þokkalega heillegt kom upp þörfin á að afkastaprófa (e. Benchmarking) miðlarann og bera hann saman við önnur sambærileg kerfi. Leit var hafin að afkastaprófunum sem gætu hentað en í ljós kom að flestar slíkar prófanir s.s. TCP-afkastaprófanir[11] skoða afkastagetu SQL miðlara byggða eftir viðmiðuninni færslur-á-sekúndu (e. Transaction pr. Second) og þá sem ein lokatala.

Hönnuðir MySQL litu svo á að slík prófun væri þeim gagnslaus m.a. af þeim sökum að fáir notendur inna (e. Execute) forrit sín eftir sömu formerkjum og þessar prófanir svo nærri vonlaust þykir að dæma heildar virkni eftir þeim. Auk þess að MySQL bauð ekki upp á færslu (e. Transactions) möguleika á þessum tíma.

---

[10] POSIX er staðall fyrir þrædda forritun á UNIX verkvangi. lesa má um þetta efni á slóðinni: http://www.llnl.gov/computing/tutorials/pthreads/

[11] Um TCP benchmarking má lesa á slóðinni http://www.cs.odu.edu/~btu/benchmark/main.html

MySQL afkastaprófanirnar eru hannaðar til að sýna fram á hversu hraðvirkur SQL gagnagrunnsmiðlari er í algengum aðgerðum (e. Operation), s.s. að afla tengingar (e. Connection) við miðlarann, framkvæma einföld innskot[12] (e. Inserts) eða samtvinnun[13] (e. Join) tveggja taflna með lyklun (e. Key). Þetta leyfir þeim einnig að mæla álag (e. Load) á vefsíðu þegar aðgerðablöndunin (e. Mix of operations) er þekkt. Að sjálfsögðu krefjast slíkar mælingar þess að forritarinn skilji eðli forrita sinna þegar slíkar prófanir eru framkvæmdar.

Eftir því sem tímar liðu komu fram auknar kröfur á MySQL póstlistanum (e. Mailing list) varðandi MySQL eiginleika (e. Features) og hvernig þeir væru í samanburði við aðra miðlara. Aðalhönnuði miðlarans, Monty, leist illa á að grafa í gegnum gömul tilvísana söfn (e. Reference material) sem oft voru úrelt til að finna slíkar upplýsingar. Hann upphugsaði því forrit sem gæti sjálfkrafa (e. Automatic) fundið út hvaða getu gagnagrunnsmiðlarar hefðu upp á að bjóða. Honum datt í hug að slíkt forrit væri einnig öflug prófun á getu MySQL miðlarans þegar honum væru sendar margar afbrigðilegar fyrirspurnir (e. Abnormal Queries).

Til þess að framkvæma slíka prófun þurfti einhverja hugmynd um getu þeirra miðlara sem prófaðir voru en vinna við að fletta slíku upp og kóta er mjög seinleg. Því var búið til forrit sem gat fundið slíkt út sjálfkrafa. Við fyrstu prófanir tólsins kom ýmislegt óvænt í ljós, miðlararnir sem prófaðir voru hrundu (e. Crash) við mikið álag og var slíkt svo algengt að forritið var nefnt „crash-me.“ Svo ágengt er forritið að eini gagnagrunns miðlarinn sem hefur verið prófaður með því án þess að hrynja er Oracle. Við prófanir hafa að sjálfsögðu komið fram ótal lýs (e. Bugs, í. pöddur) í MySQL sem hafa allar verið lagaðar.

Forritið crash-me og afkastaprófanirnar eru útfærðar (e. Implemented) sem Perl DBI/DBD forskriftir (e. Scripts). Forrit þessi gera þúsundir fyrirspurna á valinn gagnagrunns miðlara til að finna út hvernig hann virkar við raunverulegar aðstæður. Við keyrslu koma þá upp helstu mörk (e. Limits) á vinnslu miðlarans. Einn kostur forritsins er sá að gagnagrunns hönnuður getur hannað SQL grunna sína á færanlegri (e. Portable) máta. Hann getur t.d. með aðstoð þeirra fundið út mörk þess miðlara sem myndi geyma grunnana.

Algengustu aðgerðategundir sem crash-me prófar eru INSERT, UPDATE, DELETE og SELECT, aðgerðir sem fylgt er eftir með að velja eða smíða töflur og atriðaskrár. Dæmi um þá gagnagrunns miðlara sem hafa verið prófaðir með tólinu eru: Adabas-D, Access, DB2, Impress, Informix, MS-SQL Server, MySQL, Oracle, PostgreSQL, Solid og Sybase.

Þær afkastaprófanir sem gerðar eru eiga að gefa nokkuð góða mynd af þeim hraða sem hver miðlari ber í það og það sinnið. Heildarniðurstöður á prófunum á milli miðlarategunda geta þó verið villandi og ber að varast að bera þær grimmilega saman. Sum próf eru keyrð mismunandi oft, með mismunandi skilyrðum og mismunandi fjölda lína (færslna). Einn SQL miðlari gæti verið mjög lélegur í sumum ómerkilegum (e. Unimportant) aðgerðum en afbragðs góður í öðrum sem gætu verið einmitt þær sem hönnuðurinn þarfnast.

Ágætt er að bera saman niðurstöður sama gagnagrunns miðlarans eftir verkvöngum (eða stýrikerfum) og útgáfunúmerum. Til dæmis hafa samanburðar-prófanir á MySQL miðlaranum sjálfum leitt í ljós ýmsar niðurstöður:

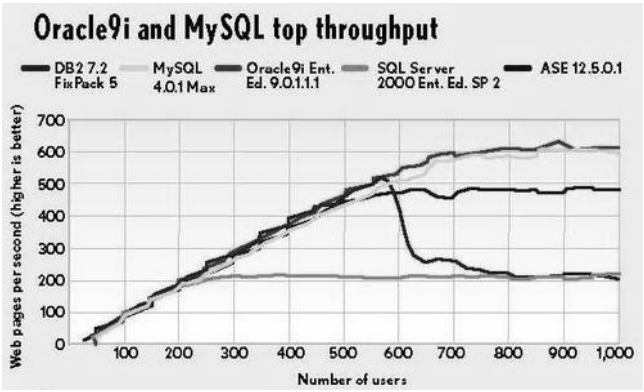
---

[12] Tölvuorðasafnið mælir með orðinu „innsetning“ fyrir enska orðið Insert. Ég slysast stundum að nota orðið innskot sem merkir hið sama.

[13] [enska] **join** [íslenska] **töflutenging** kv. [sh.] samtenging kv. [skilgr.] Aðgerð í töflualgebru þar sem búin er til ný tafla úr tveimur eða fleiri töflum sem hafa sameiginleg gildamengi fyrir einn eða fleiri dálka hverrar töflu. [skýr.] Aðgerðin felst í mengjamargföldun taflnanna ásamt því að taka saman þær línur úr upprunalegu töflunum sem hafa sömu gildi í sameiginlegu gildamengjunum. Ég slysast stundum til að nota orðið „tvinnun“ eða „samtvinnun“ fyrir þetta hugtak.

- Linux 2.2. er mun hraðvirkari en Linux 2.0 á fjölörgjörva tölvum. Þetta má meðal annars rekja til þess að Perl túlkurinn og MySQL miðlarinn ná að keyra á sitt hvorum örgjörvanum og SMP[14] kóti keyrist hraðar.
- Linux gefur 7% hraðvirkari vinnslu en Windows 98 og 49% hraðvirkari en Windows NT á samskonar tölvu.
- Windows 98 gefur 27% meiri hraða en Windows NT á samskonar tölvu.
- MySQL á Pentium II 400Mhz tölvu og Linux 2.2 stýrikerfi er miklum mun hraðvirkari en Sun Ultrasparc 400 Mhz tvíörgjörva tölva með Solaris 2.7 stýrikerfi. Þessar niðurstöður gætu orðið allt aðrar undir auknu álagi (e. Higher Load).
- Ef framkvæmd eru mörg innskot (e. Inserts) á Solaris tölvu má ná 22% hraðaaukningu með því að auka örgjörva aflið um 100%.
- Yfirbyggingin (e. Overhead) fyrir notkun MyODBC, og líklega fyrir flesta ODBC rekla er minnst 19%.
- Meðan prófanirnar voru framkvæmdar var auðvelt að vinna aðra vinnu á Linux vélinni á meðan en á NT var nánast ómögulegt að framkvæma neina aðra vinnu. Að opna kvaðningar-glugga (DOS glugga) tók 30 falt lengri tíma og við vélritun tók u.þ.b 10 sekúndur fyrir bókstafi að birtast á skjánum.

Heilmargt er að gerast í afkastaprófunum hjá MySQL og fylgjast má með hvað þar gerist á slóðinni http://dev.mysql.com/tech-resources/benchmarks/. Hér til gaman birtist mynd af samanburðargrafi frá eWeek sem sýnir niðurstöður úr prófunum.



Afkastaprófun á vegum eWeek. Myndin er afrituð af www.mysql.com.

Niðurstöður samanburðar prófana eru skemmtilegar fyrir notendur sem nota einfalda grunna t.d. fyrir vefsíður undir litlu álagi. Þær eru hins vegar nauðsynlegar forriturum sem starfa við þróun gagnasafns-umsjónarkerfa og þá til samanburðar. Samanburðar t.d. á getu eigin kerfis gagnvart getu samkeppnis kerfa auk lúsarleitunar og annarra kerfisþátta eins og þeir hjá MySQL AB hafa gert. Einnig henta slíkar prófanir vel fyrir kerfisstjóra sem e.t.v. reka blandað miðlara umhverfi eins og gert er hjá hýsingar fyrirtækjum t.d. gagnavöruhúsum.

Þess má að lokum geta að þeir hjá TcX nota gagnagrunna sem innihalda yfir 50 milljón færslur, og ekki má gleyma að miðlarinn var í upphafi þróaður til utanumhalds þeirra eigin grunna, sem skýrir að hann hefur

[14] SMP: Simultaneous Multi Processing

frá upphafi borið þess merki: eiginleikar eru eitthvað færri en SQL staðallinn krefst en rekstur er kostnaðarlítill og hraði (og væntanlega öryggi) er í hámarki.

Hraði

Við prófanir hefur komið í ljós að MySQL er afar hraðvirkur, hraðvirkari en nærri allar samkeppnishæfar lausnir. Ein af ástæðunum fyrir þessum mikla hraðamun er sú að miðlarinn styður ekki öll smáatriði SQL staðalsins. Sumir telja MySQL einnig til tekna að kótinn miðlarans er nær eingöngu gerður af einum og sama manninum sem jafnframt býr yfir margra ára reynslu í kótun. Þess vegna sé lítið af um-fremdar (e. Redundant) kóta í kerfinu. Megnið af algrímunum[15] sem kótinn er byggður á voru gerð á tímabili þegar tölvur voru frekar hægvirkar (miðað við hraðvirkari vélar nútímans) með takmörkuðu vinnsluminni. Helstu breytingarnar kótans í gegnum árin taka tillit til aukins vinnsluminnis með því að nýta sér hluta þess sem skyndiminni. Af þessum sökum voru algrímin í upphafi þróuð til að vera hraðvirk án þess að krefjast mikils af vélbúnaðinum.

Líta má á að kótinn hafi aðeins orðið skilvirkari með árunum. Miðað við þær vinnureglur sem hér koma fram þá mega notendur MySQL treysta því að nýir eiginleikar sem birtist í kerfinu eigi ekki að koma niður á hraða, því hraði og áreiðanleiki hafa jafnan verið aðal vörumerki miðlarans.

Þegar MySQL var fyrst hannaður, og allar götur síðan, hefur aðal krafan verið hraði og aftur hraði, miklu háværari krafa en krafan um fjölda eiginleika (e. Feature quantity). Þess vegna hefur verið sneitt hjá þróun dýrra[16] eiginleika sem eftir sem áður má setja upp með forrits-rökum. Í greinum eftir höfunda og í kóta dæmum hefur verið sýnt fram á að þetta stenst vel og m.a. er hægt að framkvæma færslur (e. Transactions) og ákvæði framandlykla algjörlega með forrits-rökum og nýta þar bæði þrædda vinnslu þjónsins og læsingu á töflum.

Á aðal vefsetri miðlarans (www.mysql.com) er að finna efni af þessu tagi auk langra lista yfir alla þá þætti sem miðlarinn styður. Það sem helsta athygli vekur við þann lestur er að þróun kerfisins hefur verið mjög hröð og í dag (útgáfa 5.0) eru nærri allir þættir, sem hefur skort hingað til, komnir á hlaðborðið (salat barinn). Bækur sem hafa tekið síðupláss í að telja upp hvað hægt er að gera og hvað ekki hægt að gera úreldast því hraðar.

Fyrir það að MySQL hönnuðirnir völdu í upphafi þá hluta SQL staðalsins sem þeim hentaði er ekki hægt að segja að forritið sé samhæft staðlinum í öllum atriðum. Þetta er þó álitamál því allar aðgerðir þjónsins fylgja staðlinum að einhverju leyti og í því helsta sem máli skiptir. Forritari sem kann skil á SQL getur því unnið með gagnagrunna í miðlaranum áreynslulítið. Auk þessa er það stefna MySQL AB að þjónninn verði í náinni framtíð al-samhæfður við ANSI SQL 92 staðalinn. Allt virðist líka stefna að þeim brunni að þetta muni standast.

Notendaleyfin

MySQL notendaleyfi eru gefin út miðað við sanngjörn skipti (e. Fair exchange) eða „Quid Pro Quo," sem útleggst sem „eitthvað fyrir eitthvað."

---

[15] [íslenska] **algrím** hk. [enska] **algorithm** [sh.] reiknisögn kv., algóritmi kk. [skilgr.] Endanleg röð skýrt afmarkaðra reglna til þess að leysa verkefni.

[16] Dýr (eða kostnaðarsamur) er hér notaður í þeirri merkingu að nota mikið vinnsluminni, þurfa mikinn örgjörvahraða eða gera kröfur til kraftmikillar vélar eða stýrikerfis.

Eins og fram hefur komið er MySQL gefið út með samskonar hætti og Aladdin Ghostscript, sem merkir að uppruna-kótinn (e. Source code) er opinber, miðlarinn og biðlaraforritin eru frí fyrir UNIX (og Linux). Auk þess eru eldri (úreltar) útgáfur algjörlega opnar með GPL leyfi.

Þetta merkir að sé MySQL miðlarinn notaður á UNIX/Linux verkvangi, hvort heldur í persónuþágu eða viðskipta kostar hann ekki neitt. Þó eru sett mörk á þessa skilgreiningu og óskað leyfisgreiðslu ef:

- MySQL miðlarinn er seldur til þriðja aðila eða sem hluti annars hugbúnaðar eða þjónustu.
- Notandi krefjist greiðslu fyrir t.d. vinnu við að setja upp og viðhalda MySQL þjóni fyrir viðskiptavini.
- MySQL miðlaranum er dreift á miðli í útgáfu (t.d. söluútgáfu eða á disk) og tekin er greiðsla fyrir miðilinn (þó ekki sé nema að hluta).
- MySQL er notaður á Win32 (Windows 95, 98, NT, 2000, XP, 2003) kerfi.

Við þær aðstæður að notendaleyfis er krafist þarf aðeins eitt leyfi fyrir þá vél sem keyrir miðlara forritið „mysqld“. Eitt leyfi dugar fyrir vélina þó hún hafi fleiri örgjörva og þó hún keyri mörg tilvik (e. Instance) þjónsins, einnig er eitt leyfi nóg án tillits til fjölda þeirra biðlara sem tengjast grunnum þjónsins.

Eftirfarandi heimspeki er lögð til grundvallar MySQL notendaleyfinu:

- SQL biðlara safnið (e. Client library) skal vera frítt svo það megi innifela í söluvöru án takmarkana.
- Fólk sem vill hafa frían aðgang að hugbúnaðinum, sem útgefendur hans hafa lagt mikla vinnu í, geti fengið hann að því tilskyldu að þeir séu ekki að hagnast beinlínis á honum m.a. með dreifingu.
- Fólk sem vill nota hugbúnaðinn í eigin kerfum, sem jafnframt séu til sölu í hagnaðarvon, getur fengið það en ættu að greiða fyrir afnotin.
- Venjuleg innanhúss notkun sé ókeypis. Hins vegar vilji einhver nota miðlarann í hagnaðarskyni, gæti sá sami aðstoðað við þróunarkostnaðinn með því að kaupa þjónustu samning, með þátttöku í skjölun, kóta-dæmum eða á einhvern annan hátt.
- Stefnan er sú að enginn ætti að þurfa að greiða fyrir hefðbundnar uppfærslur en þó gæti komið fyrir að greiðslu væri óskað fyrir meiriháttar uppfærslur með nýjum eiginleikum s.s. færslu stuðningi (e. Transaction support).
- Þannig ætti MySQL að vera góð fjárfesting miðað við aðrar lausnir.

Win32 verkvangurinn (Windows 95 og yngri Windows útgáfur) er stýrikerfi selt í hagnaðarskyni (e. Commercial system) og öll þróunar tól og forritunar umhverfi fyrir það eru dýr. Þess vegna sér MySQL AB sér ekki fært annað en að krefjast greiðslu fyrir notkun miðlarans í því umhverfi.

Vinnan á bak við Win32 útgáfuna er kostnaðarsöm og ekki væri hægt að réttlæta hana ef ekki kæmi eitthvað á móti. Til samanburðar má benda á að Linux verkvangurinn kostar ekki neitt og öll helstu þróunar tólin sem þarf eru til staðar eru ókeypis. Þess vegna krefst MySQL þróun á þeim verkvangi aðeins vinnu en ekki kostnaðarsamra tóla. Frekari upplýsingar má nálgast um þessa hluti hjá MySQL AB á slóðinni http://www.mysql.com/products/licensing/index.html

Hvað varðar verð á notendaleyfi, þ.e. við þær aðstæður þar sem þeirra er krafist er best að lesa skjalasafn útgefanda sem nálgast má á framangreindri slóð. Verðið fyrir eitt notendaleyfi var um árabil aðeins 200 USD (dollarar) en virðist hafa hækkað í 495 USD þegar þetta er ritað.

Hjá MySQL AB er boðin ágæt notendaþjónusta gegn vægu gjaldi, og er framsett í mismunandi verð-flokkum. Án efa hentar slík þjónusta vel fyrir þá sem reka MySQL í viðskiptatilgangi eða undir miklu álagi.

2. Kafli. Hugtakanotkun gagnagrunnsfræða

Hvað er gagnagrunnur

Við erum vön því að nota gagnagrunna daglega. Þegar flett er upp í símaskrá, eða skjal vistað á diski er notaður gagnagrunnur. Sé gerður minnis-, eða innkaupalisti er notaður gagnagrunnur. Gagnagrunnur (e. Database) er skipulagt safn upplýsinga sem geymt er eftir skilgreindu kerfi. Sjóðurinn er þá grunnurinn en upplýsingar sjóðsins eru gögnin í grunninum; gagnagrunnur. Bankareikningur er gagnagrunnur og líta má á allar upplýsingar sem geymdar eru eftir skipulögðu kerfi sem gagnagrunn, hvort sem hann er í tölvu eða ekki.

Gagnagrunnur sem ekki er skipulagður eftir neinu kerfi er því aðeins stór bingur upplýsinga. Slíkir upplýsingabingir eða upplýsingasjóðir nefnast Gagnasöfn (e. Collection of Data).

Bókhaldarar geyma til dæmis bókhald sitt í gatamöppu. Flest ef ekki öll bókasöfn geyma upplýsingar um bókaeign sína á sérstökum spjöldum, eða í spjaldskrá. Sjúkrasögur einstaklinga eru geymdar í þar til gerðum möppum. Þannig mætti lengi telja upp gagnagrunna sem geymdir eru á annars konar miðlum en í tölvu. Þeir eru þó allir skipulagðir eftir skilgreindu kerfi og má bæði sækja upplýsingar í þá og setja upplýsingar inn í þá eftir skipulögðum reglum.

Gagnagrunnur er væntanlega skipulagður með það fyrir augum að geyma þær upplýsingar varanlega sem í hann eru settar og sækja þær eftir skipulögðum reglum. Séu upplýsingarnar geymdar í áþreifanlegu formi t.d. í spjaldskrá er oft um að ræða eina aðferð til að sækja þær aftur.

Prentuð útgáfa Íslensku símaskrárinnar er gefin út í stafrófsröð eftir eiginnöfnum þeirra einstaklinga (og fyrirtækja) sem skráin nær yfir. Þessi framsetning gagnanna setur notendum þær skorður að finna símanúmer eftir nöfnum einstaklinga og fyrirtækja. Sé þess æskt að fletta upplýsingum eftir götuheitum verður að komast yfir aðra útgáfu grunnsins sem raðar gögnum eftir götuheitum og svæðum.

Orðabók er annað dæmi um prentaðan gagnagrunn. Hún er listi yfir orð í tilteknu tungumáli, ýmist með útskýringum á merkingu þeirra sem hugtaka, eða heitum samskonar hugtaka í öðrum málum. Þessir listar eru framsettir í stafrófsröð og sérstakur kafli undir þau orð sem hafa tiltekinn upphafsstaf. Nú mætti ímynda sér aðra framsetningu þar sem orðum væri raðað eftir tegund, t.d. nafnorð, lýsingarorð, eða eftir efnisflokk s.s. heimspeki, veðurfræði, siðfræði o.s.frv. Jafnvel mætti hugsa sér mismunandi sýnir (e. Views) á gögnin.

Greinilega setur það gagnagrunni nokkrar skorður ef hann er aðeins aðgengilegur í prentuðu formi, samanber símaskrána. Myndi það auka notagildi grunnsins ef hægt væri að fletta upp t.d. öllum númerum sem enda á *343* eða öllum eigendum símanúmera á *Aflagrund 40*. Slíkt er ekki framkvæmanlegt í prentaðri útgáfu en þarfir af slíku tagi hafa ýtt við þróun gagnagrunna í tölvutæku formi sem hafa slíkan sveigjanleika. Nú orðið er algengt að hægt sé að nálgast grunna af þessu tagi á Vefnum og eru vefir eins t.d. simaskra.is vinsælir. Sá upplýsingabrunnur er þó tiltölulega nýr af nálinni, margir eldri grunnar eru til og fæstir þeirra jafn aðgengilegir almenningi.

Gagnasafns-umsjónarkerfi (e. Database Software) er hugbúnaður sem bíður upp á aðferðir til að skipuleggja gagnasöfn í tölvum. Með öðrum orðum; forrit sem leyfa skipulagningu gagnasafna með aðstoð tölvu. Slík kerfi nefnast á ensku Database Management Systems (DBMS), hér þýtt sem gagnasafns-umsjónarkerfi, stundum uppnefnt gagnagrunns-kerfi. Hugtakið gagnagrunns-kerfi ætti þó frekar við þegar gagnagrunnurinn hefur verið hannaður og einhver forrit smíðuð til að meðhöndla hann. Þannig eru hvoru tveggja Microsoft Access og MySQL aðeins gagnasafns-umsjónarkerfi. Vefútáfa símaskrárinnar og vefsíðurnar sem stýra aðgangi að henni væru því gagnagrunns-kerfi.

Þróun gagnasafns-umsjónarkerfa hefur verið áberandi í þróun tölvunnar og segja má að þróun þessara tveggja viðfangsefna hafi haldist í hendur. Eftir því sem tölvan hefur eflst hefur einnig aukist getan til að vinna með umfangsmeiri og flóknari gagnagrunna. Eftir því sem notkun einkatölvunnar hefur færst í vöxt hefur aðgengi almennings að slíkum tólum, og grunnum þeirra jafnframt aukist. Lýðnetið (e. Internet) og aðgangur að Veraldarvefnum (e. World Wide Web) hefur komið hér nokkuð við sögu samanber aðgengi að símaskrám, notkun leitarvéla og framsetning ýmissa miðlægra grunna gegnum Netið. Leitarvélar eru einmitt dæmi um gríðarstóra grunna sem innihalda upplýsingar um innihald vefsíðna á Vefnum.

Hér þarf að gera greinarmun á Vefnum annars vegar og Netinu hins vegar. Þegar gagnagrunnur er gerður aðgengilegur á Neti merkir það aðeins að hægt er að tengjast honum frá öðrum nettengdum tölvum. Skiptir þá engu hvort netið sé víðnet á borð við Lýðnetið eða lokað staðarnet innan fyrirtækja.

Til dæmis getur þú opnað kvaðningu (e. Prompt) og gefið skipunina „Telnet grusk.isbok.is 3307“ og beðið eftir svari. Þú myndir þá fá svar frá MySQL miðlaranum á Linux vélinni hjá mér, og hefur það ekkert með Vefinn að gera. Telnet vinnur á TCP/IP samskiptum á Lýðnetinu óháð Vefnum sem slíkum.

Veraldarvefurinn (eða Vefurinn) byggir á þeirri aðferðafræði að nettengdar tölvur geti miðlað vefsíðum á Lýðnetinu með sérhæfðum miðlunar hugbúnaði sem nefnist Vefþjónn eða Vefmiðlari (e. Web Server). Hver sá sem hefur aðgang að Lýðnetinu og býr yfir Vefrápara[17] (e. Web Browser) getur rápað á og lesið (eða notað) þær vefsíður sem miðlað er. Þegar gagnagrunnur er aðgengilegur á Vefnum þýðir það að tölvan sem miðlar vefsíðunum hafi aðgang að gagnagrunns miðlara (e. Database Server) og geti miðlað þeim gögnum sem sá miðlari geymir.

Gagnagrunns hugbúnaður

Gagnagrunns umsjónarkerfi, eða gagnagrunns hugbúnaður nefnist á ensku „Database Management System,“ skammstafað DBMS. Slíkur hugbúnaður er þá ætlaður til að skilgreina og geyma gagnagrunna eftir einhverju kerfi og sníða mismunandi aðgengi að þeim gögnum sem grunnarnir geyma. Yfirleitt er ætlast til þess að gagnagrunns hugbúnaður leyfi hönnuði grunnsins að sníða hann eftir eigin kröfum eða kerfi, hvort heldur gagnagrunninn sjálfan eða aðgengi að gögnum hans.

Öll slík kerfi í biðlara/miðlara umhverfi bjóða forritunar skil (Application Programming Interface eða API) sem leyfa að gerð séu sérsniðin forrit er geti tengst við grunnana. Gott dæmi um slíkt er einmitt vefsíða sem vinsar upplýsingar úr gagnagrunni.

Til eru ýmsar tegundir slíkra gagnasafns-umsjónarkerfa en vinsælust eru kerfi fyrir venslaða gagnagrunna (e. Relational Databases). Tvö best þekktu dæmin fyrir einstaklings notkun og smærri fyrirtæki eru Microsoft Access og Borland Paradox. Þessi kerfi eru þó smápeð við hlið stóru gagnagrunns miðlaranna s.s. Oracle, Microsoft SQL Server, IBM DB2, PostgreSQL, SAP og MySQL, svo fáein vel þekkt nöfn séu nefnd.

Miðlara/biðlara vinnsla

Yfirleitt er gert ráð fyrir því þegar um gagnagrunns miðlara er að ræða að gagnagrunnurinn sé geymdur á „til þess gerðri tölvu“ (e. Dedicated Machine). Á vélinni sé keyrður sérstakur hugbúnaður, oftast sem púki (e. Daemon) sem stjórni öllum aðgangi að grunninum og þeim gögnum sem hann geymir. Slík uppsetning er jöfnun höndum nefnd miðlari eða þjónn (e. Server).

Miðlarinn sér um að miðla gögnum í grunninum til notendaforrita sem nefnast þá biðlarar (e. Clients) samkvæmt einhverri aðgangsstýringu (e. Access Control) og skammta notenda aðgengi (e. User Access).

[17] Ég hef kosið að fylgja leiðbeiningu tölvuorðasafns og nota orðið Rápari fyrir enska orðið Browser.

Mjög mikilvægt er þegar hér er komið að gefa aðgangsstýringu og notenda aðgengi góðan gaum. Þegar gagnagrunnur er hannaður er ekki aðeins ákvarðað hvaða gögn séu geymd og hvernig þau skuli geymd heldur er einnig ákvarðað hverjir geti séð hvaða gögn, hvernig þeir geti séð þau, og síðast en ekki síst hverjir mega eyða gögnum, bæta nýjum við eða dagrétta þau sem fyrir eru. Allt þetta er framkvæmt í miðlaranum en ýmist umbeðið eða skoðað gegnum biðlara.

Notkun símaskrárinnar á Vefnum er gott dæmi um slíkt kerfi; gagnagrunns þjónn geymir alla símaskrána og miðlar til einstaklinga sem sækja upplýsingar í grunninn með aðstoð nettengdrar tölvu. Vefsíðan gefur tvenns konar almennt aðgengi: Notendur geta flett óhindrað upp öllum þeim símanúmerum og aðilum sem símaskráin geymir en engin leið er fyrir hinn almenna notanda að bæta við, eyða eða dagrétta gögnin. Annarsstaðar á vefnum (undir „Mínar síður“ hjá simnet.is) getur notandi breytt því hvernig sínar eigin upplýsingar birtast í símaskránni en aðeins sínar eigin. Þriðja viðmótið er hér hugsanlegt, það er notandi sem hafi sérstakt leyfi til að breyta upplýsingum í grunninum en slíkt aðgengi væri væntanlega háð notendanafni og lykilorði.

Gagnagrunnsþjónar (e. Database Servers) miða ætíð að því að gögnin séu á netþjóni og biðlari sæki í aðgang að gögnunum gegnum nettengingu eftir atvikum (e. Respectively). Þá er gert ráð fyrir að gögnin fái þeir einir séð sem hafa til þess tiltekin leyfi eða aðgang. Yfirleitt er þannig gert ráð fyrir að mismunandi notendur með mismunandi aðgangsheimildir sjái misýtarleg gögn.

1. Æfing: Hvenær þurfum við gagnagrunn?

Hvað finnst þér vera viðeigandi svar við eftirfarandi þrem spurningu:

1. Hvenær þarf ég að tölvuvæða gagnasafnið mitt?
2. Hvenær þarf ég að nota einfalt sjáborðs-forrit á borð við Microsoft Access
3. Hvenær ætti ég að nota miðlara á borð við MySQL.

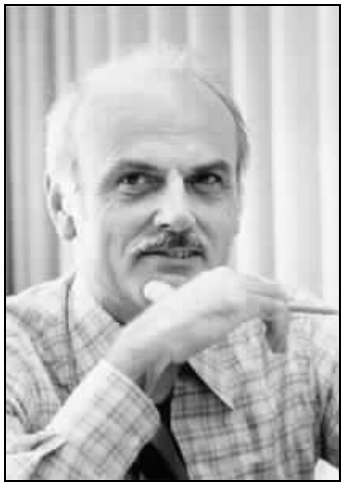
Opnaðu uppáhalds ritvinnsluforritið þitt og ritaðu þar spurningarnar ásamt svörunum. Þegar þú hefur lesið kaflann allan, eða eftir viku, ættir þú að svara þeim upp á nýtt til samanburðar.

Venslaðir gagnagrunnar

Ted Codd og vensla-algebra

Dr. E. F. Codd (f. 1923 d. 2003) var breskur stærðfræðingur sem vann að rannsóknum hjá IBM mest allan sinn starfsferil. Á sjöunda áratugnum hafði hann unnið allnokkuð við rannsóknir á gagnagrunnum og gagnagrunns notkun þess tíma. Um 1970 birti hann grein um venslaða gagnagrunna sem hefur gjörbylt allri hugsun varðandi gagnagrunns vinnslu síðan. Hann hafði komist að þeirri niðurstöðu að gagnagrunnar þess tíma og vinnsla með þau gögn sem þeir geymdu leiddi tíðum til skekkja (e. Errors) og umfremdar[18] (e. Redundancy). Árið 1953 flutti hann búferlum frá Bandaríkjunum til Kanada og bjó þar í um tíu ár, í mótmælum við baráttu McCarthy öldungadeildarþingmanns gegn Kommúnisma.

[18] Umfremd, á ensku „Redundancy,“ er þegar gögnum er ofaukið.



Ted Codd

Greinina nefndi hann „A Relational Model of Data for Large Shared Data Banks“ (í. Venslað gagnalíkan fyrir stóra samnýtta gagnabanka). Þar kynnti hann til sögunar gagnalíkan og aðferðir til gagnageymslu sem átti að leysa öll þau helstu vandamál sem á þeim tíma tröllreið allri gagnagrunns vinnslu. Grein hans má finna á slóðinni http://www.acm.org/classics/nov95/toc.html.

Á þessum tíma var mikið um Hierarchical Databases og Networked Databases. Umfjöllun um þau gagnalíkön er utan bókarsviðsins en vert væri að benda á að fletta hugtökunum upp í http://www.hyperdictionary.com/dictionary/ eða í http://computing-dictionary.thefreedictionary.com/.

Árið 1974 hóf IBM þróun á kerfi sem nefndist System/R sem átti að sanna kenningu Codds um venslaða gagnagrunna. Þessi þróun leiddi til fyrstu útgáfu á fyrirspurnamáli sem notað var til að vinna með gögn í gagnagrunnunum sem System/R vann á. Fyrirspurna málið var nefnt SEQUEL sem er skammstöfun á „Structured English Query Language.“ Síðar var málið endurnefnt sem „Structured Query Language“ (skammstafað sem SQL) af lagalegum ástæðum og síðar var það gert að opinberum staðli á vegum American National Standards Institute (ANSI). Árið 1978 voru útgáfur af System/R í notkun hjá ýmsum viðskiptavinum IBM. Árið 1979 var verkefnið lagt niður með þeirri niðurstöðu að venslaðir gagnagrunnar væru vænleg nálgun á gagnagrunns vinnslu.

Það var ekki fyrr en 13 árum eftir að grein Codds kom fyrst út að IBM tók þá ákvörðun að framleiða gagnasafns-umsjónarkerfi sem byggði á kenningum hans og nefndi DB2. Þá hafði Lawrence „Larry“ J. Ellison þegar framleitt Oracle gagnasafns-umsjónarkerfi sitt frá 1977 og öðlast forskot sem hann heldur enn. Larry Ellison hafði í fyrstu ætlað sínu kerfi að vera samhæft System/R kerfinu en IBM leyfði fremur takmarkaðan aðgang að System/R sem leiddi til þess að Ellison þróaði sitt kerfi fullvaxið.

Gagnasafns-umsjónarkerfi á þessum árum þóttu aðeins eiga erindi í stærri tölvukerfum og vert er að hafa í huga að þegar IBM hóf þróun á „IBM PC“samhæfðu einkatölvunni 1979-80 var almennt talið að enginn myndi hagnast á slíku ævintýri. Í dag 25 árum síðar má finna gagnasafns-umsjónarkerfi í einhverri mynd á allflestum tölvum og í notkun alls staðar þar sem tölvur eru notaðar í einhverri mynd.

Þegar leitað er að efni á Vefnum með leitarvél er flett upp í gagnagrunni. Þegar debet korti er rennt í gegnum posavél er tengst við gagnagrunn. Þegar pantaður er flugmiði, bókað hótelherbergi eða vörur pantaðar eru höfð samskipti við gagnagrunna.

Einföld gagnageymsla

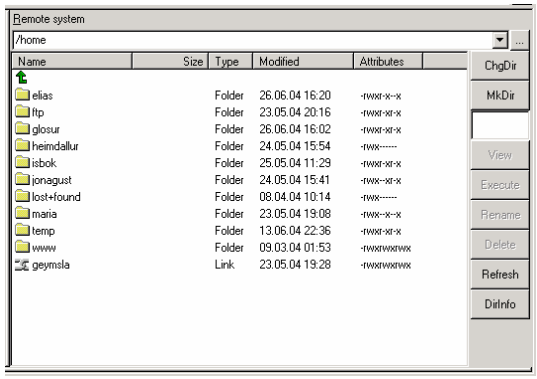
Hér á eftir er skráalisti úr „/home“ skráasafni (e. Directory) á tölvu með RedHat Linux stýrikerfi. Myndin sýnir lista yfir 11 skráasöfn og eina skrá í 12 línum. Allar línurnar innihalda lýsingu á viðkomandi skrá(eða skráasafni), stærð, dagsetning, hver sé eigandi, lesréttindi og inningarréttindi[19]. Fyrir allflesta tölvunotendur lítur slíkur listi út fyrir að vera nákvæmlega það sem hann er: listi yfir skrár sem innihalda gögn og skráasvæði sem geta innihaldið fleiri gögn og e.t.v fleiri skráasvæði.

Raunin er sú að hér er á ferðinni ein elsta tegund af geymslusvæðum í tölvum en einnig er þetta nokkuð vel skipulagður gagnagrunnur! Tölvunotendur eru svo vanir að vinna með slíkar upplýsingar að þeir meðhöndla þær yfirleitt sem áþreifanlega hluti frekar en skipulagðar upplýsingar og gögn.

```
Last login: Sun Jun 27 18:47:23 from xxx.yyy.z.z
 [elias@moltke home]$ ls -l
total 52
drwxr-x--x   14 elias    elias        4096 Jun 26 16:20 elias
drwxr-xr-x    2 root     root         4096 May 23 20:16 ftp
lrwxrwxrwx    1 elias    elias           8 May 23 19:28 geymsla -> /geymsla
drwxr-xr-x    2 elias    elias        4096 Jun 26 16:02 glosur
drwx------    4 heimdallur heimdallur 4096 May 24 15:54 heimdallur
drwx--xr-x   12 jonagust jonagust     4096 May 24 15:41 jonagust
drwx------    2 root     root        16384 Apr  8 10:14 lost+found
drwx--x--x    4 maria    maria        4096 May 23 19:08 maria
drwxrwxrwx    3 root     root         4096 Mar  9 01:53 www
[elias@moltke home]$
```

Skráasafn á Linux tölvu séð
með Bash skel í Linux.

Myndin hér á eftir eru sömu upplýsingar og á myndinni fyrir framan, eini munurinn er sá að nú er notað grafiskt viðmótsforrit (e. Graphical User Interface) til að skoða gögnin. Grafiska forritið hefur samband við stýrikerfi tölvunnar og fær sama lista og áður en unnið er úr gögnunum svo að þau birtast notandanum sem mynd með táknum. Notandinn fær síðan í hendurnar tól sem leyfa honum að vinna með gögnin t.d. mús og hnappa.



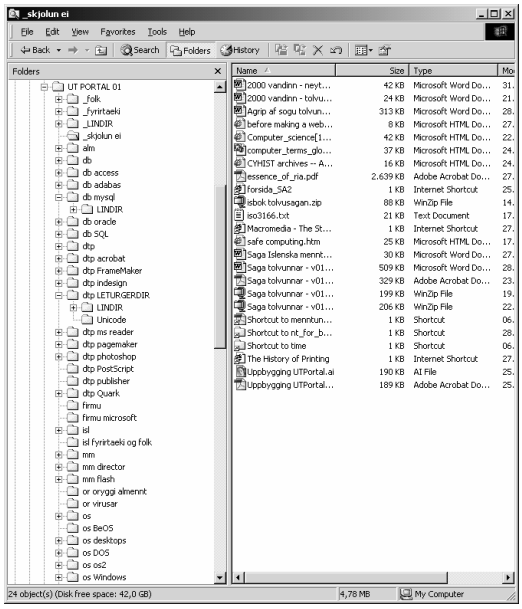
Skrásafn á Linux tölvu séð
með FTP forriti í Windows

Myndin hér á eftir er sama eðlis og áður nema gagnalistinn er á skráakerfi tölvu með Windows stýrikerfi. Mappa í Windows er samskonar fyrirbæri og skráasafn eða Directory á Linux.Áður en Windows kom til

[19] Sögnin að inna stendur fyrir ensku sögnina „to execute.“ Að inna aðgerð eða inna forrit, eftir atvikum.

sögunnar í PC heiminum var DOS stýrikerfið ráðandi sem einnig notaði hugtakið Directory. Bæði hugtökin eru notuð í þessari bók og eiga við sama fyrirbærið.

Ein mappa í trénu er opin svo hægra megin birtast öll þau skjöl sem í henni eru, hvert skjal er greinilega tagmerkt í stýrikerfinu fyrir stærð, heiti móðurforrits skjalsins og dagsetningu sköpunar eða breytingar skjalsins. Greinilega er hér á ferðinni vel skipulagt gagnasafn.



Notendahannað möpputré á Windows skráakerfi séð með Windows Explorer

Það sem gerir þennan tiltekna gagnagrunn athyglisverðan er að hann býður beinlínis upp á sum þau vandamál sem Codd glímdi við í grein sinni. Engin regla í gagnasafninu getur komið í veg fyrir að skjöl með sama heiti séu geymd á mörgum stöðum. Slíkt kemur oft fyrir þegar unnið er með gögn t.d. að fyrsta útgáfa skjals er geymd á einum stað, næst þegar skjalinu er breytt gæti það hafa verið vistað (í afritunarskyni) á annan stað o.s.frv. Við slíkar aðstæður er nærri því vonlaust að hafa yfirsýn yfir hvaða skjal er réttast eða hefur nýjastar upplýsingar. Væru skjölin mörg og sérstaklega ef margir notendur hefðu samtímis aðgang að þeim, væri útilokað að hafa yfirlit yfir hvort óþarfa gögn væru geymd í mismunandi skjölum eða upplýsingar margendurteknar, s.s. umfremdar- eða umframgögn (e. Redundancy).

Einfaldur gagnagrunnur

Til þess að átta sig á því hvers vegna hugmyndir Dr. Codd höfðu byltingarkennd áhrif á gagnagrunna er vert að athuga aðra möguleika á gagnasöfnum t.d. mætti athuga einfalda gagnaskrá sem geymir samskiptaupplýsingar einhverra aðila: nöfn, heimilisföng, netföng og slíkt. Einfaldast væri að álíta sem svo að skráakerfi stýrikerfisins myndi geyma skrána og ákveða mætti að hún væri afmörkuð (e. Delimited) textaskrá.

Tafla 1a

```
Nafn   Heimilisfang  Sími   NetfangVeffang
Grjóti Stefsson      Aflatröð 5    599-1234      grjoti@aflatradir.net
       www.aflatradir.net/grjoti
Jóndi BullssonBoðatröð 6    599-2341      jondi@bodatradir.net
       www.bodatradir.net/jondi
```

```
Gudda Karlsdóttir    Grandatröð    599-3412      gudda@grandatradir.net
       www.grandatradir.net/gudda
```

Þessi skrá er mjög einföld aðeins fimm dálkar og afmörkun dálkanna er með dálka-merki (e. Tab delimited). Einnig mætti afmarka dálkana með öðrum afmarkara (þ.e. lesstaki) s.s. semíkommu (;) og einnig mætti afmarka gildi dálkanna sjálfra með tvíhöggi, samanber:

Tafla 1b

```
“Nafn”;”Heimilisfang”;”Sími”;”Netfang”;”Veffang”
“Grjóti Stefsson”;”Aflatröð 5”;”599-
1234”;”grjoti@aflatradir.net”;”www.aflatradir.net/grjoti”
“Jóndi Bullsson”;”Boðatröð 6”;”599-
2341”;”jondi@bodatradir.net”;”www.bodatradir.net/jondi”
“Gudda Karlsdóttir”;”Grandatröð”;”599-
3412”;”gudda@grandatradir.net”;”www.grandatradir.net/gudda”
```

Þessi gögn eru e.t.v. ekki falleg á að líta eða fljótlesin enda ekki ætlast til þess. Tilgangur allra gagnagrunna, hvort heldur settir upp í einföldum textaskrám eða á annan máta er að tölvan geti lesið gögnin eftir einhverju kerfi. Fyrir þessa skrá mætti forrita fremur einfalt forrit sem a) læsi skrána og b) þáttaði (e. Parse) hana í línur, þar sem fyrsta línan hefði dálkaheiti, c) aðgreindi gildin og raðaði upp í línur og dálka eða raðaði upp á spjöld sem notandi grunnsins gæti síðan lesið, breytt, bætt inn eða eytt út og d) gæti skrifað í skrána hafi gögnin breyst eða hreinlega yfirritað hana. Einfalt ekki satt. Nú mætti bæta við forritið fáeinum notadrjúgum aðgerðum s.s. að raða nafnalistanum í stafrófsröð eða eftir heimilisföngum. Notandi gæti þá flett upp á notanda með því að leita eftir símanúmerum eða öðrum mismunandi skilyrðum.

Nú vandast málið ef sú þörf kemur upp að fleiri en einn notandi þurfi aðgang að skránni t.d. á Neti. Hvernig á að bregðast við ef tveir eða fleiri notendur eru að breyta upplýsingum og vista á sama sekúndubroti eða ef sömu línunni er breytt á sama tíma. Þetta mætti leysa þannig að forritið sem les skrána gæti læst aðgangi að henni á meðan hún er lesin í minni og aftur á meðan hún er skrifuð út, sem aftur getur hamlað vinnsluhraða ef t.d. 100 notendur væru að vinna með skrána á sama tíma. Augljóslega getur forritunar vinna vegna einfaldra gagna vaxið og vaxið.

Gott er að hafa hér í huga að 1970 þegar Codd er að birta hugleiðingar sínar um venslaða gagnagrunna með aðferðum „Vensla-algebru“ (e. Relational Algebra) – ekki má gleyma að Codd var fyrst og fremst stærðfræðingur – var öll tölvuvinnsla miðuð við tölvur af stórtölvu og millitölvu gerð og notendur tengdust tölvum sínum yfirleitt með einföldum-útstöðvum (e. Dumb terminals). Einstaklings tölvan sem hugtak þótti fáránleg og algengt var að notendur ynnu margir í einu með sömu gögnin.

Notendur Lýðnetsins eru bundnir að miklu leyti við samskonar vinnuhugmynd og þá var venja. Þegar notandi sækir og les vefsíðu, t.d. gagnagrunnstengdar upplýsingar á innankerfis (e. Online) pöntunarvef, er allt eins víst að hundruð og jafnvel þúsundir notenda séu að sækja sömu upplýsingarnar og hugsanlega á sama tíma. Aðal munurinn á þeim aðstæðum frá því sem var á blómatíma Codds, er að miðlarinn (eða stórtölvan) geta sent gögnin hrá til ráparans (e. Browser) sem getur þá tekið þátt í útlitsmótun upplýsinganna en slíkt er utan viðfangsefnis þessarar bókar.

Það sem þörf er á hér er öflugri lausn en skráakerfi stýrikerfisins geymsluvélarinnar. Lausnin gæti verið púki (e. Daemon) sem er vakandi í vinnsluminni geymslutölvunnar og stjórnar öllum aðgangi að skránni.

Púkinn gæti þá haft yfir að ráða stefjum og geymsluaðferðum sem þáttuðu gögnin fyrir notendaforritið og tækju við þeim aftur til geymslu. Slíkur púki gæti hugsanlega sinnt þörfum fleiri biðlara forrita samtímis án þess að þau yrðu sérstaklega vör við tafir vegna læsingu á skráaaðgangi.

Gögnin í gagnasafninu gætu þá litið út gagnvart notendaforritunum eitthvað svipað því sem birtist í eftirfarandi töflu. Hér mætti hugsanlega miðla gögnunum þannig til notenda að þeir geti aðeins átt við gildi í einni línu í senn þannig að séu fleiri notendur að vinna með gögnin samtímis, finni þeir ekkert fyrir því svo fremi þeir séu að vinna með ólíkar línur.

Tafla 1c

Nafn	Heimilisfang	Sími	Netfang	Veffang
Grjóti Stefsson	Aflatröð 5	599-1234	grjoti@aflatradir.net	www.aflatradir.net/grjoti
Jóndi Bullsson	Boðatröð 6	599-2341	jondi@bodatradir.net	www.bodatradir.net/jondi
Gudda Karlsdóttir	Grandatröð	599-3412	gudda@grandatradir.net	www.grandatradir.net/gudda

Ennfremur væri sætt ef púkinn gæfi biðlurunum leyfi til að sækja takmarkaðan hluta gagnanna án þess að forritarinn þyrfti að hafa sérstaklega fyrir þáttun og frekari forskriftun (e. Scripting). Til að mynda mætti sækja öll gögn í töflunni þar sem heimilisfang væri „Aflatröð“ og raða í hækkandi röð eftir símanúmerum, eða í stafrófsröð eftir gildunum í nafn o.s.frv. Slík tækni myndi sérstaklega vera mikilvæg ef gagnaskráin innihéldi t.d. 500.000 línur og notandi gagnanna þyrfti aðeins að vinna með þær 50 línur sem hugsanlegt er að framangreint skilyrði gæfi af sér.

Þetta er þó ekki nóg. Aðal ástæða þess að Dr. Codd vann að hugmyndafræði sinni um venslaða gagnagrunna var, eins og fram hefur komið, að gagnagrunnskerfi þess tíma buðu upp á skekkjur og umfremdir. Ennfremur rakst Codd á ýmis frábrigði[20] (e. Anomalies) sem gátu komið upp við ýmsar aðstæður í gagnavinnslu. Greinin sem hann ritaði árið 1970 voru í raun niðurstöður rannsókna hans sem áttu að sýna með aðferðum stærðfræðinnar hvernig forðast mætti frábrigði, koma í veg fyrir skekkjur (e. Errors) og útrýma umfremdum.

Í viðtölum við samstarfsmenn Codd frá þessum tíma hefur komið í ljós að fyrstu viðbrögð manna við niðurstöðum hans voru að hrista hausinn og segja sem svo „hver nema stærðfræðingur kemur fram með svona?“ Aðrir svo sem Larry Ellison lásu hinsvegar grein Codds og komust að raun um að lausn hans er afar einföld, ekki nóg með það hún svínvirkar. Svo rækilega virkar hún að Larry Ellison er einn ríkari manna samtímans og þú ert að lesa þessa bók. Ekki nóg með það heldur eru allar líkur til þess að þú notir daglega gagnasafns-umsjónarkerfi í einhverri mynd.

Frávik og fleiri þættir

Markmið Codds var að nálgast „áreiðanlegt upplýsinga líkan“ (e. Dependable Information Model). Allar geymslu aðferðir sem í boði voru buðu upp á vandamál sem hann kallaði frávik, á ensku Anomalies. Hann flokkaði þessi frávik í þrjá flokka: „dagréttinga frávik“ (e. Update Anomaly), „Eyðingar frávik“ (e. Delete Anomaly) og „Innskots frávik“ (e. Insert Anomaly).

Dagréttinga frávik

Gefum okkur að púkinn sem áður var talað um sé fyrir hendi og nú sé hægt að færa inn töflur með upplýsingum og að hann leyfi aðgengi margra notenda í senn að gögnunum. Næsta skref væri þá að ákveða, eða tilgreina, hvernig gögnin skuli geymd, þau sótt og dagréttuð.

Gefum okkur að við séum að störfum hjá fyrirtæki og að það geymi upplýsingar um önnur fyrirtæki sem það sé í viðskiptum við. Margir starfsmanna okkar þurfa aðgang að þessum upplýsingum, sem gætu litið þannig út:

[20] Frávik er þegar eitthvað bregður út af reglu eða brýtur upp skipulag sem gert var ráð fyrir. Óregla og frábrigði eru samheiti orðsins frávik. Nota ég öll hugtökin yfir sama fyrirbærið. Tölvuorðasafnið hefur ekki skilgreint enska orðið Anomaly sérstaklega en enskt samheiti þess er Exception sem þýtt er sem frábrigði í hlutbundinni forritun. Sú venja hefur skapast í umfjöllun um hlutbundna forritun verið notað orðið frávik yfir Exception en rétt væri að nota þar frábrigði samkvæmt Tölvuorðasafni. Skekkja og villa eru hins vegar er þýðingar á enska orðsins Error.

Tafla 2a

Nr	Fyrirtækisnafn	Fyrirt-heima	Svæði	Tengiliður	Titill Tengiliðar	Sími	Farsími	Netfang
1	Hugbrjótur ehf.	Sílíkongarðar 25	0220 Hafnarfirði	Grjóti Stefsson	Forstjóri	599-1234	999-1234	grjoti@aflatradir.net
2	Hugbrjótur ehf.	Sílíkongarðar 25	0220 Hafnarfirði	Fríða Fróðadóttir	Markaðsstjóri	599-1234	999-1234	grjoti@aflatradir.net
3	Ferlatækni ehf.	Sílíkongarðar 50	0210 Garðabær	Jóndi Bullsson	Forstjóri	599-2341	999-2341	jondi@bodatradir.net
4	Kerfagutl hf.	Miðstígum 6	0200 Reykjavík	Gudda Karlsdóttir	Forstjóri	599-3412	999-3412	gudda@grandatradir.net

Við fyrstu sýn virðist ekkert vandamál vera hér á ferðinni. Auðvelt er að fá upp allar færslur (e. Records) eða línur (e. Tuples) töflunnar og vinsa úr stakar færslur. Stakar færslur gætu verið t.d. „Allar færslur sem innihalda Hugbrjótur ehf. í sviðinu (e. Field) Fyrirtækisnafn.“ Slík fyrirspurn myndi gefa upplýsingarnar í línu 1 og 2, eftir atvikum (e. Respectively).

Segjum nú að fyrirtækið Hugbrjótur ehf. flytji á heimilisfangið „Gagnagarðar 66, 0112 Reykjavík.“ Þá þarf að dagrétta heimilisfang þess í töflunni. Eins og hún lítur út núna væri það tiltölulega auðvelt þar sem aðeins er um tvær færslur að ræða, en hvað ef tengiliðir fyrirtækisins væru orðnir 30? Vissulega mætti þá kalla upp allar færslurnar 30 og breyta gildunum í Fyrirt-heima og Svæði.

Hér kemur fyrsta „dagréttinga frávikið“, breyta þarf gildum fyrir eitt ákveðið fyrirtæki á fleiri stöðum en einum.

Einnig koma hér upp hugsanleg skekkju frávik. Segjum að ritarinn, sem falin er sú ábyrgð að dagrétta breytingarnar, misriti heimilisfangið í einni færslunni og riti „Gagnagarðat 66“ af misgáningi. Niðurstaðan yrði þá 29 réttar færslur og 1 röng. Segjum sem svo að við höfum ritað forrit sem vinsar út öll fyrirtæki eftir heimilisföngum og ritar hvert fyrirtæki á sérstaka síðu. Fyrirtækið Hugbrjótur myndi í þessu tilviki fá tvær aðgreindar síður, eina með 29 færslum og aðra með 1 færslu.

Segjum sem svo að fyrirtæki okkar þyrfti að skipta markaðssókn sinni þannig að „markaðsstjóri stærri[21] viðskiptavina“ fengi ábyrgð á samskiptum við öll fyrirtæki sem hafa tvo eða fleiri tengiliði en „markaðsstjóri minni viðskiptavina“ fengi ábyrgð á samskiptum við þau fyrirtæki sem hafa aðeins einn tengilið. Listaforritið okkar sendir markaðsstjórunum sjálfvirkt niðurstöðurnar en markaðsdeildirnar deila ekki húsnæði. Nú höfum við sent upplýsingar um sama fyrirtækið í tvær gagnstæðar áttir og höfum enn minni tækifærisgátt (e. Window of opportunity) til að laga skekkjuna. Skekkju frávikið vex, tvær markaðsdeildir hafa nú aðgang að sama viðskiptavininum í einu.

Vel mætti hugsa sér fleiri skekkjumörk. Segjum að ritarinn færi öll heimilisföngin rétt en næst þegar nýr tengiliður bætist í hóp viðskiptavina frá Hugbrjóti ehf. misritist nafn fyrirtækisins sem Hugbrjotur ehf. Sex mánuðum síðar breytir fyrirtækið um nafn og verður að Hugkvarnir ehf. Nú keyrum við lítið smáforrit á gagnagrunninn sem breytir öllum færslum með Fyrirtækisheitinu Hugbrjótur ehf. í Hugkvarnir ehf. Nú yrði eftir ein færsla með fyrirtækinu Hugbrjotur ehf. og í hinni ímynduðu markaðsskiptingu okkar er áframhaldandi skekkja. Sex mánuðum síðar hætta Hugkvarnir ehf. störfum, öllum færslum þess er eytt en Hugbrjotur ehf. er ennþá til í viðskiptafærslum okkar!

[21] Hér eru minni og stærri notað í þeirri merkingu að hver tengiliður eigi viðskipti við okkur og magn viðskipta sé stærðarvirki (e. Quantifyer) á stærð viðskiptavinar.

Besta leiðin til að greiða úr þessum flækjum og frávikum væri að skipta gögnunum upp í tvær töflur sem síðar tengjast saman á skyldleikum sínum. Þetta er einmitt niðurstaða Codds og efni þessarar bókar er að sýna hversu auðvelt þetta er allt saman.

Eyðingar frávik

Segjum nú sem svo að eyða þurfi Guddu Karlsdóttur úr grunninum, sem í sjálfu sér er einfalt. Línunni er eytt út. Því næst koma upp kröfur um að gera lista yfir öll fyrirtæki sem átt var viðskipti við á síðustu 12 mánuðum. Nú var línu 4 eytt þegar upplýsingarnar um Guddu voru fjarlægðar og þar með eru horfnar upplýsingarnar um Kerfagutl ehf.

Þetta eru „eyðingar frávik“ sem Codd nefndi á ensku „Deletion anomaly.“ Þegar eyða þarf gögnum er hætta á að önnur gögn eyðist með eða ekki sé hægt að eyða þeim sem eyða þarf vegna þess að önnur, tengd, gögn leyfi það ekki.

Sé upplýsingunum skipt upp í tvær töflur, fyrirtæki geymd sér og einstaklingar geymdir sér þá hefði verið nóg að eyða línunni með Guddu út úr einstaklings töflunni án þess að fyrirtækinu væri eytt sérstaklega.

Innskots frávik

Segjum nú sem svo að við þyrftum að bæta við nýjum einstakling í grunninn en vitum ekki hjá hvaða fyrirtæki hann er tengiliður. Þá væri óhægt um vik að skrá hann nema skrá fyrirtækið einnig. Hið sama gæti komið upp þegar skrá þarf inn nýtt fyrirtæki en við vitum ekki hver tengiliðurinn er.

Þessar aðstæður nefndi Codd „Innskots frávik“ eða „Insert Anomaly,“ þegar ekki er hægt að setja gögn inn í grunninn vegna þess að ónógar upplýsingar eru fyrir hendi eða skorður (e. Constraints) á innskotum krefjast þess.

Aftur er lausnin fólgin í því að skipta upplýsingunum upp í tvær töflu, fyrirtæki í sérstaka töflu og einstaklinga í aðra. Þá gætum við bætt nýjum einstaklingum inn sérstaklega og fyrirtækjum sérstaklega.

2. Æfing: Jólakortagrunnur

Þú færð það verkefni að útbúa jólakortagrunn fyrir fjölskylduna. Þú þarft að geyma upplýsingar um alla aðila sem a) þín fjölskylda sendir jólakort og b) alla þá sem hafa sent fjölskyldunni jólakort. Þú vilt vita hver sé tengiliður við þessa aðila t.d. gæti húsbóndinn á heimilinu verið tengiliður við „Jóa frænda“ sem býr fyrir Norðan, því þeir eru skyldir. Húsmóðirin er tengiliður við gamla vinkonu sína sem einnig er fjölskylduvinur. Grunnurinn þarf að geta birt frá hverjum komu jólakort í fyrra en fengu ekki kort á sama tíma. Merkja þarf við hverjir hafa ekki sent jólakort undanfarið.

Hvernig myndir þú leysa þessa kröfu?

1. Listaðu upp alla þætti sem hér er lýst án þess að hafa áhyggjur af töflunni. Endurtaktu listann allt að tvisvar sinnum, þú munt örugglega rekast á eitthvað sem þér yfirsást í fyrstu atrennu. Þessi atriði eru gildin sem þú þarft að geyma.
2. Rissaðu upp tvær til þrjár töflur, eftir þínum hentugleika, og raðaðu gildunum inn í töflur og hópaðu saman í viðeigandi línur. Þetta eru töflurnar með færslum. Endurtaktu þetta tvisvar.
3. Reyndu nú að greina hvaða gildi gera töflurnar skyldar innbyrðis. Hikaðu ekki við að endurhanna þær eina ferðina enn. Þú vilt ekki þurfa þess þegar þú byrjar að forrita[22] lausnina..

[22] Þó að þessi bók kenni ekki forritun þá muntu örugglega fá áhugann á slíku von bráðar og hver veit, kannski skrifa ég bráðum haug af skemmtilegum forritunarbókum.

Einindi

Algeng aðferð við að skipta upp töflum er sú að greina upplýsingar þeirra upp í einindi (e. Entity) og reyna eins og hægt er að greina atóms-gildi (e. Atomic values) þeirra eða frumgildi (e. Primary values) sem nákvæmast. Til þess að beita þessari aðferð er gott að líta fyrst á skilgreiningu[23] hugtaksins Einindi:

Einindi samkvæmt Tölvuorðasafninu er: „Sérhvert hlutrænt eða hugrænt fyrirbæri sem er til, hefur verið til eða gæti orðið til, þar með talin tengsl milli fyrirbæranna. [dæmi] Persóna, hlutur, atburður, hugmynd, ferli o.s.frv. [skýr.] Einindi er til, hvort sem gögn um það eru tiltæk eða ekki.“

Samkvæmt þessari skilgreiningu má því líta á fyrirtæki sem sérstakt *hugsanlegt* einindi. Maður er einindi, borð er einindi. Einindi sem til er í hlutveruleikanum er oft nefnt tilvik ímyndaðs einindis. Þannig ímyndum við okkur hvernig borð lítur út og hvaða eiginleika það hefur. Raunverulegt borð er þá hlutgert tilvik þeirrar hugmyndar. Hvorutveggja, tilvikið og hugmyndin, geta verið einindi.

Hugsanlegt fyrirtækis einindi hefur nafn, heimilisfang og samskipta gáttir s.s. síma, fax, veffang ofl. Einindið fyrirtæki er þannig almenn lýsing á fyrirtækjum sem hægt er að máta við raunveruleg fyrirtæki. Fyrirtækið Hugbrjótur ehf. er þá tilvik af einindinu fyrirtæki. Hugbrjótur ehf. er þannig einindi líka. Einskonar holdgerving hugmyndar sem lýst hefur verið.

Sé þessari hugsunar-aðferð beitt á Töflu 2a má sjá að hún geymir tvö skýr einindi, Fyrirtæki og Tengiliður. Fyrirtæki hefur heiti, staðsetningu og samskiptagáttir. Tengiliður hefur nafn, titil, heimilisfang og samskiptagáttir. Einnig má greina tvö önnur einindi þegar betur er að gáð en það eru Heimilisfang og Svæði.

Einindi er þannig hugtak (eða aðferð) til að lýsa einhverju sem getur átt sér sjálfstæða tilvist.

Þegar þessari aðferð er beitt á töflur koma upp á yfirborðið spurningar á borð við: „Hversu smámuna-samlega þarf ég að skipta töflunni upp í einindi?“ Tökum sem dæmi einindið Svæði. Svæði hefur aðeins tvö atóms-gildi, númer og heiti. Ágóðinn af að klippa Svæði út úr töflunni og smíða sérstaka töflu fyrir það er ótvíræður af sömu ástæðum „dagréttinga-frávika“ og áður hafa komið fram.

Nú myndi það spara vinnu við innslátt ef upplýsingar um póstsvæði væru fluttar í sérstaka töflu, í stað þess að slá inn „0210 Garðabæ“ (og stöku sinnum slá inn skekkjuna Garðabær) t.d. 100 sinnum fyrir hundrað einstaklinga eða fyrirtæki í Garðabæ, væri nóg að slá inn númerið sjálft, 0210[24]. Þegar flett væri upp á tilteknu fyrirtæki mætti nota póstnúmerið til að fletta upp heiti þess í viðkomandi töflu.

Tilvik eininda (e. Instance, Entity instance, Instance of Entities) er þegar tafla fær áþreifanlegar (e. Concrete) upplýsingar um einindi sem raunverulega er til í viðfangs-veruleikanum[25] (e. Universe of discourse) sem gagnagrunnurinn nær til. Viðfangs-veruleiki er einnig nefndur raunheimur. Gagna-

---

[23] Þess er gætt eins og kostur er í hugtakaskilgreiningum að notast við Tölvuorðasafn Íslenskrar málnefndar. Sakir skammleika (e. Brevity) er ekki hægt að gera öllum enskum hugtökum nægilega góð skil svo enskum hugtökum er dreift hér og þar í texta bókarinnar. Lesendur eru hvattir til að nota ágæta leitarvél Tölvuorðasafnsins á http://ismal.hi.is/ til að fletta þeim upp.

[24] Póstnúmer á íslandi eru númeruð frá 101 til 902 en í þessari bók hefur verið bætt tölunni 0 framan við póstnúmerin. Þetta er gert til tákns um að allar upplýsingar í gagnagrunnum bókarinnar eru skáldaðar og skrefi frá raunverulegum (almennum, íslenskum) gagnagrunnum. Búast má við af þessum sökum að ýmis póstnúmerasvæði í grunnum bókarinnar séu ekki til á íslandi. Sama afstaða er tekin t.d. til kennitalna, engin raunveruleg kennitala mun finnast í þessari bók.

[25] Margar gagnagrunnsbækur enskar, nefna viðfangs-veruleika eða Universe of Discourse þann veruleika sem við viljum geyma upplýsingar um. Viðfangs-veruleiki getur verið félagaskrár, þjóðskrár eða einkunnatöflur.

grunnurinn geymir með öðrum orðum upplýsingar um raunveruleg einindi sem til eru í raunheimi, eða viðfangs-veruleika. Samkvæmt framangreindu má líta á hugsanlegt fyrirtæki sem ákveðið einindi en þegar tiltekið fyrirtæki sem raunverulega er til er fært inn í töfluna má líta á þá færslu sem raunverulegt tilvik einindis. Fyrirtækið er til í raunheimi.

Smá innskot

Ég geri mér vel ljóst að byrjendum í svona umfjöllun líður ekkert voða vel þegar hingað er komið. Ég hef sjálfur verið í þeim sporum. Eitt sinn fékk ég viðamikla sýnikennslu í notkun gagnagrunna og var dálítið vankaður á eftir. Síðar þurfti ég að vinna með gögn og skipuleggja smærri gagnasöfn fyrir lítinn rekstur. Mjög fljótt varð ég var við, eftir því sem reynslan jókst, að þessi grunnur sem hér er framsettur styður við þessa reynslu og gerir hana ánægjulegri eftir því sem líður á. Í rauninni myndi ég hvetja byrjendur til að lesa kaflann hratt yfir, æfa sig í sýnidæmunum í næsta kafla og lesa þennan kafla aftur síðar.

Atómgildi og gildamengi

Atómgildi er gildi sem ekki verður brotið upp í einingar. Heiltalan 10 er atóm-gildi og mannsnafnið Adam er atómgildi (þó það sé samsett úr rittáknum). Mannsnafnið „Adam Alfreð Atlason“ er ekki atómgildi því það er samsett úr þrem stökum gildum. Atómgildi getur þó verið nokkuð teygjanlegt því við gætum þurft að líta svo á að fornafn-föðurnafn sé atóms-gildi eftir atvikum við hönnun tiltekins gagnasafns. Hérlendis eru einstaklingar yfirleitt skráðir með fullu nafni eða fornafn og föðurnafn saman en millinöfn sér. Erlendis er algengt að fornafn og föðurnafn/ættarnafn sé skrásett sitt í hvoru lagi.

Einindi er samsett úr atómgildum[26] (e. Atomic Values). Þetta þýðir t.d. fyrir einindið Fyrirtæki að ekki er hægt eða raunhæft að skipta nafni þess upp í einingar og ekki heldur númeri línunnar (eða færslunnar). Númer færslunnar fyrir Hugbrjótur ehf. er talan 1 sem er heiltala. Nafn fyrirtækisins í dálkinum (eða sviðinu) Fyrirtækisnafn er „Hugbrjótur ehf.“ sem er bókstafa-strengur. Orð og orðasambönd í tölvunar- og gagnagrunnsfræðum eru jafnan nefnd Strengir. Strengir eru röð rittákna sem lesin eru saman í einu lagi. Stakir stafir eru venjulega nefndir stafir, tákn eða rittákn. Þessi tvö svið má þannig skilgreina sem „Nr“ af taginu (e. Type) heiltala (e. Integer) og „Fyrirtækisnafn“ af taginu strengur (e. String).

Þegar einindum er skipt upp í atómgildi sín þá er hvert gildi sett innan ákveðins gildissviðs, t.d. hlýtur röð bókstafa að lenda innan gildissviðs strengja, tölur sem ekki innihalda brotatölur lenda innan gildissviðs heiltalna og tölur sem geta innihaldið brot innan gildissviðs rauntalna. Gildi sem gefa til kynna satt/ósatt, rétt/rangt, já/nei, af/á eru rökgildi á ensku nefnd „Boolean values“ eftir Ensk-Írska stærðfræðingnum Boole. Í gagnagrunnsfræðum eru slík gildi yfirleitt táknuð með 0 (ósatt) eða 1 (satt).

Gildistegundir í gagnagrunns fræðum eiga sér gildamengi[27] (e. Domain). Með gildamengi er átt við að t.d. heiltölur geta verið heilar jákvæðar eða neikvæðar tölur eða núll. Strengir hafa það gildamengi að geta verið röð rittákna sem rita má með eðlilegu ritmáli eða tómur strengur. Brotatölur (eða kommutölur) séu tölur sem geta haft aukastafi eða brot.

---

[26] Tölvuorðasafnið hefur ekki komið með tillögu fyrir gagnagrunns, og forritunarhugtakið, „Atomic value.“ Atómgildi, eða atómlegt-gildi verður því notað í bókinni.

[27] Þegar þetta er ritað hefur Tölvuorðasafnið ekki gefið út tillögu fyrir þýðingu á gagnagrunns hugtakinu Domain. Domain merkir á öðrum sviðum ýmist lén, umdæmi eða sérsvið. Hér hefur verið valin merkingin gildamengi sem merkir á ensku „set of values.“

Gildismengið tiltekur jafnan hversu langt það getur verið. Til dæmis getur mannsnafn verið í þjóðskrá Íslands allt að 31 staf og nafn einstaklings getur ekki verið ekkert (NULL). Mánaðafjöldi í almennu dagatali getur verið frá 1 og upp í 12 og getur ekki verið ekkert því alltaf er til mánuður[28].

Mannsnafn myndi falla undir gildamengið strengur og mánaðafjöldi undir heiltölu. Auk þess má tilgreina gildissvið mengisins. Streng má tilgreina að hann geti verið hámark 31 stafur og velja má að mánaðafjöldi sé innan gildissviðsins mjög-lítil-heiltala (e. Tiny-Integer eða TINYINT) sem í MySQL er á gildissviðinu -128 til 127.

Gildamengi er aðeins hugtak til að tilgreina hvers konar gildi séu gefin á einindi. Ef við t.d. skoðum einindið „Adam Atlason fæddur 1.1.1970." Þetta einindi á tvo eiginleika (e. Attributes), nafn og fæðingardag. Við getum geymt nafnið sem streng (gildamengi strengja) og fæðingardag sem dagsetningu. Gildamengi dagsetningarinnar sé þá Gregoríanskt dagatal? Með öðrum orðum þá er gildamengi aðeins reglur til að skilreina eðli þeirra eiginleika þ.e. eiginda, sem einindin hafa.

Einindið Hugbrjótur ehf. hefur þannig eigindin Nr, Nafn, Heimilisfang, Svæði, Aðalsíma, o.s.frv. Hvert eigindi sem slíkt dvelur innan ákveðins gildissviðs t.d. getur Nr. verið frá 0, 1, 2, og uppúr en ekki -1 og niðurúr, allt í heilum tölum.

3. Æfing: Gildamengi

Hvaða gildamengi myndir þú velja eigindum einindanna í jólakorta-grunninum? Bættu þeim við skilgreiningarnar úr þeirri æfingu.

Einindavensl, Töflur og Vensl

Hér erum við komin að kjarna nafngiftarinnar „Venslað gagnalíkan" (e. A Relational Model). Líkanið verkar þannig að öll vel skilgreind einindi eru sett í sínar eigin töflur eða vensl (e. Table, Relation).

Hverri töflu sé skipt upp í dálka (e. Column) sem nefnast svið (e. Field). Fyrsta lína töflunnar geymir þá heiti sviðanna en þessi lína er ekki meðhöndluð sem gögn heldur sem lýsing á hinu hugsanlega einindi. Hvert einindi sem til er í viðfangs veruleikanum er skráð inn í töfluna og kallast þá tilvik einindis.

Einindi sem er samsett úr upplýsingum annars konar eininda skipti þeim upplýsingum í aðrar töflur en hægt sé að tengja þau saman og fletta þeim upp eftir einhverju sameiginlegu gildi. Með öðrum orðum þá pörum við skyld einindi eftir skyldleikum sínum. Hið sameiginlega gildi er þá skyldleiki beggja einindanna. Þetta á við þegar atómsgildi í báðum töflum eru hin sömu. Ef fyrirtækið Hugbrjótur ehf. Inniheldur atómsgildið 0210 fyrir póstsvæði þá má fletta því upp í töflunni Póstsvæði og finna þar heiti póstsvæðis nr. 0210 sem er Garðabær.

Hér er kominn skyldleiki á milli fyrirtækis töflunnar og póstsvæða töfluna. Öll fyrirtæki búa á póstnúmeruðu svæði og öll póstnúmeruð svæði eiga sér heiti.

Tafla 2b

Töfluheiti: Fyrirtæki.

Nr	Fyrirtækisnafn	Fyrirt-heima	Svæði	Aðalsími	Netfang-fyrirtækis	Veffang fyrirtækis
1	Hugbrjótur ehf.	Sílíkongarðar 25	0210	599-1234	hugbrjotur@hugbrjotur.net	www.hugbrjotur.net
2	Ferlatækni ehf.	Sílíkongarðar 50	0210	599-2341	info@ferlataekni.net	www.ferlataekni.net

[28] Heimspeki er utan efnissviðs bókarinnar, en er mánuður yfirleitt til nema sem hugtak?

3	Kerfagutl hf.	Miðstígum 6	0200	599-3412	info@kerfagutl.net	www.kerfagutl.net

Hér hefur einindið fyrirtæki verið greint þannig að það hafi nafn, heimilisfang, póstsvæði, netfang og samskiptagáttir. Það má deila um það hvort við ættum að smíða sérstaka töflu fyrir símanúmer. Ef t.d. fyrirtækið hefur mörg síma- og faxnúmer, mismunandi vefslóðir og fjölda netfanga, gæti verið betra að geyma þau í tengdri töflu. Sakir skammleika (e. Brevity) verður slíkri greiningu sleppt hér. Spurningar af þessu tagi koma upp á öllum þrepum töflugreiningar og dæmi af þessu tagi er eitt hinna augljósari: Þegar einstaklingar t.d. hafa mörg símanúmer s.s. heimasími, vinnusími, vinnufarsími, einkafarsími, gamlifarsími, aukavinnusími, o.s.frv.

Tafla 3a

Töfluheiti: Einstaklingar.

Nr.	Fyrirt nr	Nafn	Menntun	Starfstitill	Heima	Svæði	Heimasími	Vinnusími	Farsími	Netfang
1	1	Grjóti Stefsson	MD. Tölvunarfræði	Forstjóri	Aflatröð 5	0110	199-1234	599-1234	999-1234	grjoti@aflatradir.net
2	2	Jóndi Bullsson	Ph D. Markaðsfræði	Forstjóri	Boðatröð 6	0210	199-2341	599-2341	999-2341	jondi@bodatradir.net
3	3	Gudda Karlsdóttir	Ph D. Markaðsfræði	Forstjóri	Grandatröð	0200	199-3412	599-3412	999-3412	gudda@grandatradir.net
4	1	Fríða Fróðadóttir	MD. Heimspeki	Markaðsstjóri	Eindartröð	0105	199-3412	599-3412	999-3412	gudda@grandatradir.net

Hér hafa einstaklingar verið settir í sér töflu og hver færsla hefur sitt eigið númer, rétt eins og öll fyrirtæki hafa sín færslunúmer. Sú spurning vaknar upp hvort betra væri (innan Íslenzks veruleika) að nota annars konar færslunúmer t.d. kennitölu? Sérstakur dálkur er í töflunni til að geyma færslunúmer þess fyrirtækis sem einstaklingurinn vinnur hjá.

4. Æfing: Greining

Hvað ef Grjóti Stefsson, sem er forstjóri hjá Hugbrjót ehf. gerist einnig forstjóri hjá nýju fyrirtæki sem einnig er í samskiptum við okkar fyrirtæki, hvernig ættum við að geyma slíkar upplýsingar?

Tafla 4

Töfluheiti: Póstsvæði.

Nr.	Nafn
0220	Hafnarfirði
0110	Reykjavík
0210	Garðabæ
0200	Kópavogi
0105	Reykjavík

Í báðum töflunum, Fyrirtæki og Einstaklingar er nú safn eininda sem öll hafa heimilisföng sem götuheiti og póstsvæði. Póstsvæði er samsett gildi úr póstnúmeri og heiti póstsvæðis. Götuheiti er samsett gildi úr heiti

götunnar og húsnúmeri, í einstaka tilfellum gæti jafnvel verið um húsheiti að ræða eins og algengt er á engilsaxneskum málsvæðum.

Hér hefur verið tekin sú afstaða að fyrst póstsvæði er samsett gildi að líta á það sem sérstakt einindi þar sem það hefur sjálfstæðra tilveru í raunveruleikanum. Þetta einindi á sitt sérstaka (einkvæma[29]) póstnúmer og heiti póstsvæðis. Hugsast getur að þetta póstsvæði eigi frekari upplýsingar s.s. aðsetur póststöðvar, upplýsingar um pósthólf o.s.frv. Þannig höfum við baktryggt okkur gagnvart frekari þarfagreiningu fyrir gagnagrunninn. Ef geyma þyrfti frekari upplýsingar t.d. um aðsetur póststöðvar þá þyrftum við aðeins að bæta þeim upplýsingum við töfluna Póstsvæði, frekar en að þurfa að troða þeim inn í allar færslur í Fyrirtæki og Einstaklingar.

Fyrirspurnir

Nú er komið að því að gera fyrirspurn sem sækir upplýsingar um Hugbrjótur ehf. og alla tengiliði sem við höfum við það fyrirtæki, ásamt póstsvæðum.

Á mæltu máli gætum við orðað fyrirspurnina þannig: „Gefðu mér allar upplýsingar um einindi í töflunum Einstaklingar og Fyrirtæki, þar sem númer fyrirtækis er 1 og fyrirtækis númer einstaklings sé líka 1, birtu upplýsingar um póstsvæði í niðurstöðunni.“ Á almennu máli mætti orða fyrirspurnina þannig: Byrtu mér alla einstaklinga hjá fyrirtæki númer 1 ásamt heimilisföngum bæði einstaklinga og fyrirtækis. Á gagnagrunnsmáli væri þetta orðað þannig:

```
select * from einstaklingar, fyrirtaeki, postsvaedi
  where fyrirtaeki.fyrirt_nr = 1
  and fyrirtaeki.fyrirt_nr = einstaklingar.fyrirt_nr
  and fyrirtaeki.postnumer = postsvaedi.postnr;
```

Gagnagrunns kerfið myndi skila eftirfarandi gögnum:

1	1	Grjóti Stefsson	MD. Tölvunarfræði	Forstjóri	Aflatröð 5	0110	199-1234	599-1234	999-1234
	grjoti@aflatradir.net	1	Hugbrjótur ehf.	Sílikongarðar 25	0210	599-1234	hugbrjotur@hugbrjotur.net	www.hugbrjotur.net	
4	1	Fríða Fróðadóttir	MD. Heimspeki	Forstjóri	Eindartröð	0105	199-3412	599-3412	999-3412
	gudda@grandatradir.net	1	Hugbrjótur ehf.	Sílikongarðar 25	0210	599-1234	hugbrjotur@hugbrjotur.net	www.hugbrjotur.net	
	0210	Garðabæ	Garðatorgi 5						

Einnig mætti setja sömu fyrirspurn upp þannig að færri dálkar kæmu fram í niðurstöðunum samanber eftirfarandi:

```
select e.einst_nr, e.fyrirt_nr, e.nafn, e.starfstitill,
  f.fyrirt_nr, f.fyrirt_nafn, f.fyrirt_heim, f.postnumer,
  f.adalsimi, p.postnr, p.heiti
  from einstaklingar as e, fyrirtaeki as f, postsvaedi as p
  where f.fyrirt_nr = 1
  and f.fyrirt_nr = e.fyrirt_nr
  and f.postnumer = p.postnr
  ;
```

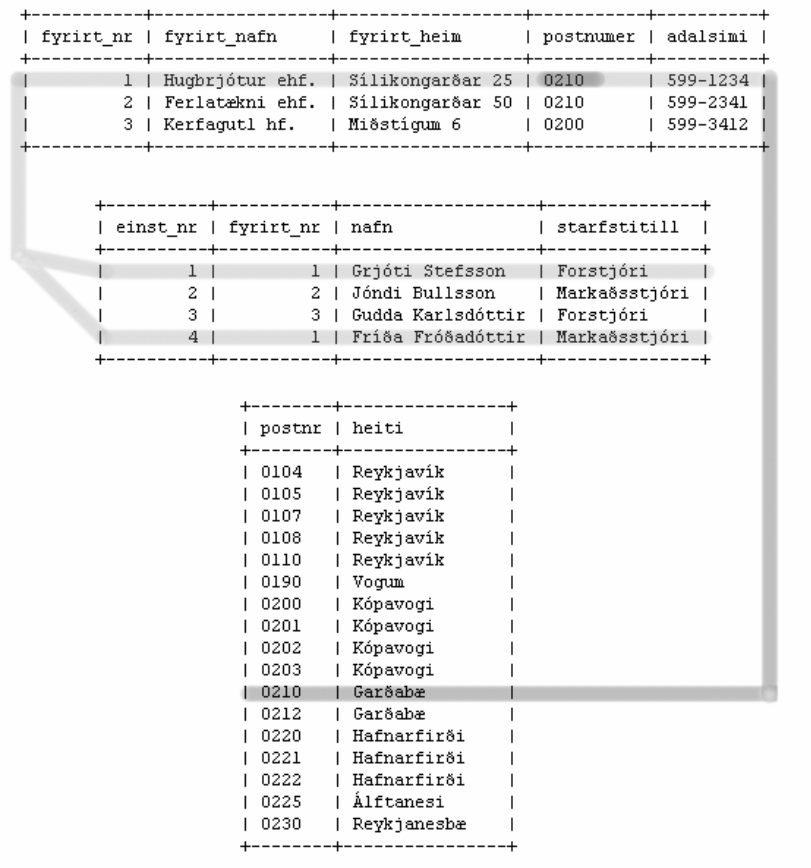
Niðurstöðurnar væru þá ögn læsilegri í framsetningu:

1	1	Grjóti Stefsson	Forstjóri	1	Hugbrjótur ehf.	Sílikongarðar 25	0210	599-1234	0210	Garðabæ

[29] Einkvæmt eða Unique stendur fyrir að aðeins eitt gildi sé fyrir hendi innan mengis (e. Set) annarra gilda. Til dæmis er kennitala íslendinga einkvæmt gildi í þjóðskránni því aðeins ein kennitala er til fyrir hvern og einn íslending.

4	1	Fríða Fróðadóttir	Markaðsstjóri	1	Hugbrjótur ehf.	Sílikongarðar 25	0210	599-1234	0210	Garðabæ

Myndin hér á eftir sýnir hvernig grunnurinn finnur línu í fyrirtækis töflunni þar sem fyrirtækis númerið er 1, tvær línur í einstaklings töflunni þar sem fyrirtækis númerið er líka 1, og póstnúmer og heiti póstsvæðis í póstsvæðis töflunni þar sem póstnúmerið er hið sama bæði í póstnúmeratöflunni og fyrirtækistöflunni.



fyrirt_nr	fyrirt_nafn	fyrirt_heim	postnumer	adalsimi
1	Hugbrjótur ehf.	Sílikongarðar 25	0210	599-1234
2	Ferlatækni ehf.	Sílikongarðar 50	0210	599-2341
3	Kerfagutl hf.	Miðstígum 6	0200	599-3412

einst_nr	fyrirt_nr	nafn	starfstitill
1	1	Grjóti Stefsson	Forstjóri
2	2	Jóndi Bullsson	Markaðsstjóri
3	3	Gudda Karlsdóttir	Forstjóri
4	1	Fríða Fróðadóttir	Markaðsstjóri

postnr	heiti
0104	Reykjavík
0105	Reykjavík
0107	Reykjavík
0108	Reykjavík
0110	Reykjavík
0190	Vogum
0200	Kópavogi
0201	Kópavogi
0202	Kópavogi
0203	Kópavogi
0210	Garðabæ
0212	Garðabæ
0220	Hafnarfirði
0221	Hafnarfirði
0222	Hafnarfirði
0225	Álftanesi
0230	Reykjanesbæ

Myndræn framsetning á fyrirspurnunum hér fyrir framan.

5. Æfing: Frávik

1. Hvaða frávik geta komið upp í jólakortagrunninum?
    a. Hvernig getur þú leyst þessi frávik?
    b. Hvers vegna?
2. Hverjir eiga að geta:
    a. Séð upplýsingar í grunninum?
    b. Dagrétt upplýsingarnar?

Skipulagning

Skipulagning (e. Normalization) í gagnasafni er samkvæmt Tölvuorðasafni íslenskrar málnefndar: „Það að breyta töflu í eina eða fleiri einfaldari töflur, lausar við umframdálka eða ósamkvæmni, í því skyni að halda vísunarheilleika“.

Vísunarheilleiki (e. Referential Integrity) er hugtak sem nær til þess að þegar gögnum, einindum, hafi verið skipt upp í töflur, að gögn í einni töflunni geti breyst um leið og tengdum eða skyldum gögnum sé breytt.

Við höfum þegar litið á hvernig frávik geta komið upp þegar gögn eru sett í töflur, einnig höfum við séð hvernig sneiða má hjá þessum frávikum með því að greina gögnin upp í þætti sem við nefnum einindi. Einnig höfum við séð hvernig einindin hafa eigindi (eiginleika) sem við skrásetjum sem atómleg gildi. Þetta ferli nefndi Dr. Codd einu nafni skipulagning eða „Normalization.“

Nauðsynlegt er að skilja eðli þessa ferlis þegar unnið er með venslaða gagnagrunna. Ferlið sem slíkt er þó aðeins viðmiðun, einskonar sniðmát að aðferð, við greiningu gagna sem síðan eru skipulögð sem töflur.

Fyrstu, annarrar og þriðju gráðu skipulagning

Skipulagning er sett upp í þrem stigum, Fyrsta stigs, Annars og Þriðja stigs skipulagning. Á ensku „1st Normal form,“ „2nd Normal form,“ og „3rd Normal form.“ Skipulagningin er þó ekki takmark í sjálfu sér heldur sniðmát sem ætlað er til aðstoðar við nálgun á gagnagrunni. Eiginlega er ekki til hinn endanlegi rétti gagnagrunnur heldur aðeins árangursríkur gagnagrunnur. Þannig mætti vel hugsa sér að sá gagnagrunnur sem hannaður var hér fyrir framan mætti hanna á annan máta. Látum það kyrrt liggja.

Ennfremur eru til fjögur stig til viðbótar sem nefnast „Boyce-Codd normal form,“ „4th Normal form,“ „5th Normal form“ og „Domain/Key Normal form.“ Þessi skipulags stig eru þó utan efnis sviðs bókarinnar (e. Out of Scope) og tilheyra frekar umræðu um gagnagrunna á æðra stigi á meðal vísindamanna sem starfa við þróun gagnagrunnsfræða. Í raunveruleikanum, við eiginlega grunnaþróun er hugtökum þessum lítið haldið á lofti sem slíkum en aðferðir þeirra, sem eru frekar einfaldar mikið notaðar.

Algengt er við hönnun gagnagrunna að fyrsta stigs skipulagning dugi en þegar svo er ekki þarf að taka hönnunina á annað stig. Sjaldan þarf að taka hönnunina alla leið á þriðja stig. Þegar þangað er komið eru grunnar yfirleitt nægilega vel hannaðir og nánast ekkert um frávik og/eða skekkjur.

Mikilvægt er að hafa í huga að takmark okkar við hönnun gagnasafns er að geta tekið tilteknar upplýsingar og sett þær skipulega inn í töflur. Við viljum geta sett gögn inn í töflurnar án þess að tapa hluta þeirra óvænt og við viljum ekki þurfa að endurhanna grunninn (töflurnar) þegar það ferli er komið af stað.

Það getur verið illgjörlegt eftir nokkurra vikna notkun á gagnasafni að taka gagnagrunns þjóninn niður (fjöldi notenda gæti þurft á aðgangi að honum að halda), endurgera töflurnar með breytingum og færa gögnin úr gömlu töflunum yfir í þær nýju. Slík vinna getur verið mun umfangsmeiri og erfiðari en sú vinna sem fer í að hanna grunninn í fyrsta sinn.

Einnig er það takmark okkar þegar kemur að áfram-þróun safnsins að geta bætt inn nýjum tegundum upplýsinga án þess að þurfa að endurhanna allan grunninn. Sérstaklega er þetta mikilvægt þegar við höfum þróað forrit sem meðhöndla þau gögn sem í grunninn fara.

Fyrsta stigs skipulag

Fyrsta stigs skipulagning er auðveld. Setja þarf gögnin upp í töfluformi og fylgja eftirtöldum viðmiðum:

Hver dálkur töflunnar innihaldi atómlegt gildi. Þannig sé aðeins eitt vel skilgreint gildi í hverjum reit dálksins. Þurfi að víkja frá þeirri reglu þá skuli gildið standa fyrir „óuppskipt eigindi“ í raunveruleikanum. Til dæmis er hefð fyrir því á íslandi að heimilisfang sé skráð sem Gata-númer, eða bæjarheiti[-númer] (sé um bændur að ræða). Samanber: Jón býr á „Aflagranda 99“ eða Jón Bóndi á „Hóli“ eða Jón bóndi á „Hóli

II.“ Yfirleitt er ekki litið svo á að skipta þurfi í sundur götuheitinu og númerinu. Í hinum engil-saxneska heimi er yfirleitt farið þannig: [heiti-húss,]-númer- gata. Samanber: John Johnson, Gatehouse, 32 Villows.

Hver dálkur hafi einkvæmt[30] heiti þannig að engir tveir dálkar í töflunni hafi sama nafn. Þetta er gert svo hægt sé að greina á milli dálka. Ef við t.d. skrásettum Fullt nafn og gælunafn einstaklinga í töflu þar sem báðir dálkarnir hétu „Nafn“ gætum við ekki í fyrirspurnum gert upp á milli dálkanna, slík fyrirspurn myndi engu skila eða skila báðum gildum.

Taflan skal hafa í það minnsta eitt einkvæmt gagnamengi (e. Unique set of data). Mengið má nota til að greina færslur hennar í sundur. Í flestum nafnatöflum á Íslandi er litið á kennitölur sem þetta mengi. Engir tveir einstaklingar í þjóðskrá Íslands hafa sömu kennitöluna. Mengi allra kennitalna á íslandi er þannig einkvæmt og ef við skrásetjum allar kennitölur í sérstakan dálk þegar við skráum einstaklinga getum við greint upp á milli allra einstaklinga sem heita „Jón Jónsson“ með því að bera saman kennitölur þeirra. Þessi regla er grundvöllur lykla (eða aðallykla) taflna. Aðallykill (e. Primary key) er stundum nefndur einkvæmur lykill (e. Unique key) á íslenzku.

Engar tvær raðir (línur) í töflunni mega vera nákvæmlega eins (e. Identical). Tökum sem dæmi að við eigum tvo vínskápa, vínskáp eitt í bókastofunni og vínskáp tvö í kvennastofunni. Í hvorum vínskáp eigum við eina flösku af „Jack Daniels“ viskíi. Töfluskráning myndi þá innihalda tvær línur með færslugildinu „Jack Daniels viskí.“ Sé fylgt reglunum hér að framan myndum við skrá þetta þannig: „Skápur 1 - Jack Daníels viskí “ og „Skápur 2 - „Jack Daniels viskí“.

Engin endurtekning á gagnahópum (e. Groups of data) er leyfð. Þetta merkir að þegar dálkur hefur síendurtekið sömu upplýsingarnar ættum við (þegar það á við) að skipta þeim upp. Ef við lítum aftur á dæmið um vínskápana hér að framan og vörpum þessari reglu á þá, myndum við fá tvær töflur. Vínlisti og Víntegundir. Í töflunni Víntegundir þyrftum við eina línu með eftirfarandi: „1, Jack Daníels viskíflaska.“ Í töflunni Vínlisti hefðum við áfram tvær færslur eins og áður; „Vínskápur, nr. 1“ og „Vínskápur, nr. 2.“ Talan 1 fyrir flöskuna gæti þá vísað á skáp nr. 1.

Skotveiðifélag, gagnagrunnur 1

Við fáum það verkefni að gera lítinn gagnagrunn sem skráir félagaskrá fyrir skotveiðifélag. Formaður félagsins lætur okkur fá upplýsingar um þau gögn sem í grunninn fara, sem eru eftirfarandi: Nafn meðlims, heimilisfang og árið sem hann gerðist meðlimur. Einnig vill félagið geta skráð byssueign meðlima. Við eigum að leysa verkefnið með lágmarksvinnu.

Við leitum einföldustu og fljótlegustu leiðar og komum upp með eftirfarandi lausn:

Nafn	Heimilisfang	Fyrsta ártíð	Byssa
Gikkur Gikksson	Skotgrund 50, 0999 Koley.	2002	Bruun einhleypa, Brzck tvíhleypa.
Skjótur Brunason	Dritgrund 55, 0888 Haglahrepp	2002	Volga riffill, Brzck einhleypa
Gjóla Hittin Miðsdóttir	Hrotalundi 33, 0777 Kúlugerði	2003	Bruun tvíhleypa, York veiðiriffill

Þessi tafla er ekki alveg í fyrsta stigs skipulagningu. Til dæmis vantar einkvæmt gildamengi sem auðkennir hvern einstakling frá öðrum sem gætu haft sama heiti. Við getum auðveldlega kippt því liðinn með því að bæta við dálki fyrir Kennitölur framan við dálkinn Nafn. Einnig sjáum við að Heimilisfang er illa skilgreint því hér er um samsett gildi að ræða sem er alls ekki atómlegt; gildið „Skotgrund 50, 0999 Koley“ er samsett úr gata-númer og póstnúmer-póstsvæði. Þar sem yfirleitt er litið á gata-númer sem atómlegt gildi á

[30] Einkvæmi stendur fyrir þegar aðeins eitt gildi kemur fyrir. Kennitala einstaklinga er einkvæmt gildi í þjóðskrá því aðeins eitt, og aðeins eitt, eintak er til af hverri kennitölu (og hverjum einstaklingi þar með) hverju sinni.

Íslandi myndum við sleppa með það í sér dálk en við hefðum átt að setja póstnúmer-póstsvæði í sérdálk og helst í sértöflu.

Lyklar

Samkvæmt Venslalíkani Dr. Codds skal hver tafla hafa lykil (e. Key) sem getur auðkennt hverja færslu sem einkvæma. Yfirleitt liggur í augum uppi hvaða gögn geta verið lyklar. Þegar taflan hefur verið skilgreind samkvæmt fyrsta stigs skipulagi og reglunni „**Taflan skal hafa í það minnsta eitt einkvæmt gagnamengi**“ verið fylgt, höfum við oftast nær fundið lykilinn.

Lykill gegnir því hlutverki að geta einn og óstuddur táknað einindið sem hann er hluti af. Í nafnatöflum á Íslandi höfum við kennitölur sem geta gegnt þessu hlutverki. Kennitalan getur komið í stað annarra upplýsinga um einstaklinginn sem á hana, þar sem það á við. Þegar við skrásetjum póstnúmer þá er þetta sömuleiðis auðvelt því hvert póstnúmer er einkvæmt og getur staðið eitt sér, t.d. hefur miðbæjarsvæðið í Reykjavík póstnúmer 101 en vesturbær hefur póstnúmer 107, þó heita bæði póstsvæðin „Reykjavík.“ Þess vegna getum við notað póstnúmerið sem lykilsvið á póstnúmera töflum og kennitölur á nafnatöflum. Ef þú segist búa á póstsvæði 300 vita flestir að átt er við Akranes og ef þú gefur upp kennitöluna þína má fletta upp auka-upplýsingum um þig; kennitalan getur táknað þig!

Þrjár tegundir lykla

Lykill: Lyklar af þessu tagi, sem eru hluti þeirra eininda sem við geymum og settir saman úr atómlegum gildum, eru ýmist nefndir Lyklar (e. Key), Aðallyklar (e. Super key) eða Frumlyklar (e. Primary key). Hér eftir verða þessi þrjú heiti, Lykill, Aðallykill og Frumlykill notað yfir sama fyrirbærið. Alltaf er litið á lykil sem einkvæman.

Hugsanlegur lykill: Þegar eðli einindanna eru þannig að ekki er ljóst hvort eitthvert gildi sé einkvæmt eða geti táknað allt einindið er fundinn „hugsanlegur lyklill“ (e. Candidate key). Lykill af þessu tagi er víkjandi hugtak í hönnunarferlinu því þegar við höfum lokið töfluhönnuninni ættum við ekki að hafa neina hugsanlega lykla heldur að hafa fundið og skilgreint alla einkvæma lykla. Hins vegar, á meðan hönnunarferlið stendur yfir getum við greint ýmsa hugsanlega lykla sem við höfum ekki tekið afstöðu til og haft þá í huga við hönnunina.

Samsettur lykill:. Í sumum tilfellum þurfum við hins vegar á samsettum lyklum (e. Composite key) að halda. Tökum sem dæmi fyrirtæki sem geymir upplýsingar um viðskiptavini í tveimur löndum, „Langtburtistan“ og „Nærlandi“, auk viðskiptavina á Íslandi. Tekin hefur verið sú afstaða að geyma í nafnatöflunni upplýsingar um póstnúmer og landnúmer og geyma öll póstsvæði í sömu töflunni.

Langtburtistan hefur landsnúmerið 601 og Nærland hefur 721, Ísland hefur 354. Í Langtburtistan eru tvö póstsvæði: „1 Langaborg“ og „2 Stuttaþorp“ meðan Nærland hefur póstsvæðin „1 Náborg“ og „2 Nýþorp“.

Taflan yfir viðskiptavini gæti litið svona út (að sumum dálkum slepptum):

Nr.	Heiti	Póstnúmer	Landnúmer
1	Fötugerðin ehf	101	354
2	Skjólugerðin ehf	103	354
3	Langfötugerðin inc.	1	601
4	Stuttfötugerðin inc.	2	601
5	Náfötugerðin ehe.	1	721
6	Nýfötugerðin ehe.	2	721

Í nafnatöflunni höfum við búið til dálkinn Nr. sem auðkennir númer hvers fyrirtækis í töflunni og er því aðallykill hennar. Hér hefur verið tekin sú afstaða að fyrirtæki erlendis hafi ekki samskonar kennitölur og íslensk fyrirtæki svo hugsanlegi lykillinn „nýtt sjálfkrafa númer fyrir hverja nýja færslu“ verið gerður að aðallykli frekar en kennitölur.

Taflan yfir landaheiti gæti orðið þannig:

Landnúmer	Landsheiti
354	Ísland
601	Langtburtistan
721	Nærland

Í landatöflunni er dálkurinn Landnúmer greinilega einkvæmt númer og verður notaður sem aðallykill.

Í póstnúmeratöflunni gætum við tekið sömu afstöðu og í nafnatöflunni og auðkennt hverja færslu með einhverju númeri sem við gefum hverri færslu. Þá þyrftum við ekki að skrásetja landnúmer og póstnúmer fyrir hvert fyrirtæki. Við myndum þá hanna dálkinn „Póstvísun“ sem gæti vísað á einhverja færslu í Póstnúmeratöflunni. Þannig gæti fyrirtækið „Fötugerðin ehf.“ fengið gildið 1 í dálkinn Póstvísun sem kæmi í stað landnúmers og póstnúmers.

Taflan gæti þá litið þannig út:

Sjálfkrafa nr.	Landnúmer	Póstnúmer	Póstsvæði
1	354	101	Reykjavík
2	354	101	Reykjavík
3	601	1	Langaborg
4	601	2	Stuttaþorp
5	721	1	Náborg
6	721	2	Nýþorp

Hinsvegar er þetta langsótt og við erum að geyma hér umfremdar-gögn (e. Redundant data). Við höldum upprunalegu hönnuninni á nafnatöflunni og sleppum dálkinum „Sjálfkrafa nr.“ í póstnúmeratöflunni. Í staðinn skilgreinum við töfluna þannig: Gildi sem sett er saman úr Landnúmer og Póstnúmer eru aldrei eins, hver lína er einkvæm. Þannig getum við lyklað töfluna á samsettan lykil úr dálkunum Landnúmer og Póstnúmer.

Landnúmer	Póstnúmer	Póstsvæði
354	101	Reykjavík
354	101	Reykjavík
601	1	Langaborg
601	2	Stuttaþorp
721	1	Náborg
721	2	Nýþorp

Málskipan lykla er þannig að við undirstrikum heiti þeirra í töflunum á greiningarstigi og teikningum. Ef vel er skoðað sést að nöfn dálkanna Nr., Landnúmer, og Sjálfkrafa nr. eru allir undirstrikaðir í dæmunum hér að framan. Í síðustu töflunni eru dálkarnir Landnúmer og Póstnúmer undirstrikaðir með brotnu striki til tákns um að þeir gegni saman hlutverki samsetts lykils.

Einkvæmir lyklar og skorður

Lyklar í töflufræðum hafa þá eiginleika að þeir skulu ætíð vera einkvæmir til þess að tryggja að ekki megi geyma sama lykilinn tvisvar í sömu töflunni. Gagnagrunns forrit eru þannig úr garði gerð að þegar lykill hefur verið skilgreindur fyrir töflu er þess gætt sjálfkrafa að hann sé einkvæmur.

Þannig notum við lykla sem skorður á heilindi þeirra eininda sem við setjum í töfluna. Með heilindum þá er átt við t.d. að ekki sé hægt að slá inn sama einstaklinginn tvisvar í nafnatöflu: upplýsingar um þann einstakling eru þá heil. Þetta á líka við um samsetta lykla.

Skotveiðifélag, grunnur 2

Nú líður eitt ár eða svo og einfalda lausnin okkar líkar vel. Stjórn skotveiðifélagsins lýðfyllir[31] (e. Populate) töfluna og notar til þess lítið forrit sem getur tengst gagnagrunninum yfir netið. Forritið leyfir stjórninni að raða upplýsingunum í stafrófsröð eftir nöfnum, heimilisföngum og eftir ártíðum. Félagsmenn eru orðnir 5.000 talsins.

Nú kemur nýr formaður félagsins sem vill skoða upplýsingarnar á annan máta. Hann vill geta skráð kennitölu félagsmanna og gælunöfn, skipta heimilisfanginu upp þannig að götuheiti og húsnúmer sé skráð saman og póstnúmer og póstsvæði saman. Þetta krefst þess að fyrstu töflunni sé breytt. Að auki vill hann skrá byssur þannig að auðvelt sé t.d. að skoða lista yfir allar einhleyptar haglabyssur sérstaklega eða riffla sérstaklega, sérstaka skrásetningu fyrir greiðslu félagsgjalda og aðseturstað til viðbótar lögheimili, þá vill hann að hægt sé að skoða sögu byssueignar því meðlimir skipta um byssur en stundum þarf félagið að leita álits þeirra á byssum sem þeir hafa áður átt.

Nú þurfum við að endurskoða grunninn frá upphafi, ein tillaga gæti litið svona út, lýðfyllt með upplýsingum Gikks Gikkssonar:

Tafla: Félagsmenn

Kt.	Nafn	Gælunafn	Aðsetur	Lögheimili
1234561239	Gikkur Gikksson	EKKERT	SAMA OG LÖGHEIMILI	1

Tafla: Heimili

Færslunr.	Húsnúmer-gata	Póstnúmer	Póstsvæði
1	Skotgrund 50	0999	Koley

Tafla: Byssueign

Færslunr.	Kt.	Byssa	Keypt	Seld
1	1234561239	Volga riffill	1/1/2002	ÓSELD
2	1234561239	Brzck einhleypa	1/1/2003	ÓSELD
3	1234561239	Colt skammbyssa 44	1/1/2003	1/5/2004

Tafla: Félagsgjald

Færslunr.	Kt.	Ár	GreittDags
1	1234561239	2002	1/2/2002

[31] Tölvuorðasafnið hefur ekki enn komið með tillögu fyrir enska gagnagrunns hugtakið „to populate“ sem notað er yfir þá aðgerð að setja gögn inn í töflur. Ég mun nota sögnina „að lýðfylla“ fyrir sama hugtak.

2	1234561239	2003	1/1/2004
3	1234561239	2004	ÓGREITT

Nú höfum við gagnagrunn sem lítur allt öðru vísi út en sá fyrsti. Látum liggja á milli hluta hvort hann er rétt hannaður eða ekki því endalaust má endurhanna gagnagrunna og endurmeta þá. Til dæmis mætti spyrja sig að því hvort við ættum að skrá kyn meðlima sem kk eða kvk og einnig mætti spyrja hvort við ættum að skrásetja millinöfn sér og jafnvel ættarnöfn. Venja er t.d. að skrá póstsvæði á Íslandi í sérstaka töflu en því er sleppt hér.

Fjarvera gildis eða NULL

Við sjáum að Gikkur Gikksson hefur ekkert gælunafn og frekar en að skrá þar textann „EKKERT“ gætum við haft sviðið tómt. Í gagnagrunns fræðunum er skráð sérstakt gildi; „NULL.“

NULL er hugtak sem táknar að ekkert gildi sé fyrir hendi eða fjarvera gildis. Þannig gætum við skráð Gælunafn er NULL og Aðsetur er NULL, sem myndi þá tákna að Gikkur Gikksson hefði ekkert gælunafn eða í það minnsta þekkjum við það ekki Við gætum jafnvel sett inn tóman streng (“”) til að tákna að hann hafi ekkert gælunafn og NULL til að tákna að við vitum ekki hvort hann hafi gælunafn.

Hið sama á við um aðsetursstað. Við getum notað gildisleysið NULL fyrir óseldu byssurnar hans tvær til að tákna að þær séu óseldar og í sömu merkingu fyrir ógreidda félagsgjaldið sem var ritað sem ÓGREITT.

Félagið samþykkir nýja gagnagrunninn. Við þurfum því að afrita allar upplýsingar úr gamla grunninum yfir í þann nýja. Þegar því er lokið og við höfum prófað að lýðfylla hann með nokkrum færslum fram og aftur, tökum við niður gamla grunninn og setjum þann nýja í staðinn. Auk þess þurfum við að skrifa frá grunni forritið sem áður var notað til að skrásetja og skoða upplýsingarnar í grunninum. Hér á eftir fer nýi grunnurinn með lítilsháttar breytingum:

Tafla: Félagsmenn

Kennitala	Nafn	Gælunafn	Aðsetur	Lögheimili
1234561239	Gikkur Gikksson	NULL	3	1
0123451119	Skjótur Brunason	NULL	NULL	2
0012342229	Gjóla Hittin Miðsdóttir	Gola	NULL	3

Tafla: Heimili

Færslunr.	Húsnúmer-gata	Póstnúmer
1	Skotgrund 50	0999
2	Dritgrund 55	0888
3	Hrotalundi 33	0777

Tafla: Póstnúmer

Póstnúmer	Póstsvæði
0999	Koley
0888	Haglahrepp
0777	Kúlugerði

Tafla: Byssueign

Færslunr.	Kt.	Byssa	Keypt	Seld

1	1234561239	Volga riffill	1/1/2002	NULL
2	1234561239	Brzck einhleypa	1/1/2003	NULL
3	1234561239	Colt skammbyssa 44	1/1/2003	1/5/2004
4	0123451119	Volga riffill	1/5/2002	NULL
5	0123451119	Brzck einhleypa	15/9/2002	NULL
6	0012342229	Bruun tvíhleypa	20/3/2003	NULL
7	0012342229	York veiðiriffill	31/8/2003	10/6/2004

Tafla: Félagsgjald

Kennit.	Ártal	GreittDags
1234561239	2002	1/2/2002
1234561239	2003	1/1/2004
1234561239	2004	NULL
0123451119	2002	NULL
0123451119	2003	NULL
0123451119	2004	NULL
0012342229	2003	1/2/2003
0012342229	2004	1/2/2004

Einkvæmis ákvæði lyklunar

Eins og fram hefur komið þarf lykill í töflu að standa fyrir einkvæmar færslur og þannig geti lykillinn staðið sjálfstæður sem táknrænt gildi fyrir önnur gildi færslunnar. Önnur gildi lýsi þannig þeim lykli sem táknar þau. Ég get því litið á nafn mitt og heimilisfang sem nánari lýsingu á kennitölunni minni í þjóðskrá. Einnig lítum við þannig á að gildin sem hnýtt eru við lykilinn eiga ekkert erindi í grunninn ef lyklinum er eytt en það var ein af skipulags reglum Dr. Codd.

Einkvæmt en ekki lykill

Stundum viljum við setja inn þau ákvæði að eitthvert sérstakt gildi í töflunni sé einkvæmt (e. Unique) án þess að vera sérstakur lykill. Taflan hér á eftir er einmitt dæmi um slíka töflu. Hér hefur verið skilgreind taflan Voðabyssur sem geymir upplýsingar um sérstakar byssur sem félagið á en felur traustsverðum félagsmönnum að geyma. Við sjáum að sviðið KT_ábyrgðarm er frumlykill töflunnar þannig að einn einstaklingur getur geymt eina, og aðeins eina, byssu í senn. Við gætum sett frekari skorður á töfluna þannig að t.d. sviðið Raðn sé einkvæmt án þess að það sé nauðsynlega lykilnúmer.

Tafla: Voðabyssur

KT_ábyrgðarm	*Raðn*	Heiti	Keypt	Seld	Tegund
0012342229	*sk-0003*	Holt	1/3/2004	NULL	Skammbyssa
0123451119	*sk-0002*	Keith & Messon	1/2/2004	NULL	Skammbyssa
1234561239	*sk-0001*	Muger	1/1/2004	NULL	Skammbyssa

Atriðaskrár

Í inngangi var lauslega minnst á atriðaskrár (e. Indexes) en þær tengjast lyklum. Atriðaskrá er sem fyrr segir aukatafla sem við meðhöndlum aldrei með beinum hætti. Þegar atriðaskrá (e. Index) er skilgreind þá er venjulega vísað á að skrásetja aðallykil í atriðaskránni. Sé t.d. nafnatafla yfir 300.000 færslur og kennitala væri lykilsvið, þá myndi atriðaskráin búa til tveggja dálka töflu sem geymir allar kennitölur og númer þeirra lína sem geyma þær.

Þegar leitað er að ákveðinni kennitölu í töflunni getur gagnasafns-umsjónarforritið leitað í atriðaskránni og fundið þannig færslunúmer lykilsins mjög hratt. Færslunúmerið vísar gjarnan á nákvæma staðsetningu færslunnar t.d. á diski tölvunnar og getur forritið þá sótt færsluna á mjög skömmum tíma án þess að þurfa að framkvæma yfirgripsmikla lesningu.

Atriðaskrár eru venjulega nefndar við stofnun töflunnar og útheimta litla aukakunnáttu í SQL. Atriðaskráin skrásetur þá sjálfkrafa alla lykla töflunnar og í hvaða færslum þeir finnast. Einnig má mynda atriðaskrár út frá tilgreindum dálkum (sviðum) töflunnar.

Annars stigs skipulag og ákvæði

Yfirleitt þurfum við ekki að skeyta um annars stigs skipulag nema við höfum töflur með samsettum lyklum. Lítum fyrst á hugtakið ákvæði (e. Dependency).

Gefum okkur að Skotveiðifélagið sem áður var nefnt hafi tvo skotæfingavelli, Leirárvöll og Sandárvöll, sem settir eru upp í eftirfarandi töflu:

Völlur	Aðstaða	Staðsetning	Umsjónarmaður
Leirárvöllur	Leirdúfupallur	201	Örn Valsson
Sandárvöllur	Skotmörk	220	Smyrill Þrastarson

Við sjáum að heiti vallarins er skilgreint sem lykill og gildið „Leirárvöllur“ getur því staðið sem táknrænt gildi fyrir restina þ.e. „Leirdúfupallur, 201 (Kópavogi?), Örn Valsson umsjónarmaður.“ Ef Leirárvelli yrði eytt þá þurfum við ekki lengur umsjónarmann og hann er því rekinn, honum má eyða út úr töflunni. Bíðum þó með að eyða færslunni:

Segjum nú sem svo að Leirárvöllur bætir við aðstöðu fyrir skotmörk, taflan myndi þá líta svona út:

Völlur	Aðstaða	Staðsetning	Umsjónarmaður
Leirárvöllur	Leirdúfupallur	201	Örn Valsson
Sandárvöllur	Skotmörk	220	Smyrill Þrastarson
Leirárvöllur	Skotmörk	201	Örn Valsson

Við höfum nú endurskilgreint töfluna þannig að dálkarnir Völlur og Aðstaða verða samsettir lyklar þ.e. sviðið Völlur getur ekki lengur verið einkvæmt eitt og sér heldur Völlur+Aðstaða. Nú sitjum við uppi með þá stöðu að nafn umsjónarmanns er tvítekið og ef Leirárvöllur myndi bæta við t.d. Handvopnasal þá yrði hann þrítekinn. Einnig gerist það að væri aðstöðunni Skotmörk á Sandárvelli eytt hyrfi Smyrill Þrastarson sem væri slæmt ef hann tæki við sem umsjónarmaður Skotmarka Leirárvallar. Við höfum hér frekar illa skilgreint gagnaákvæði.

Gagnaákvæði stendur fyrir að gögn séu ákveðin eftir öðrum gögnum eða séu háð tilvist þeirra. Við höfum óbeint gert tilvist umsjónarmanns þannig úr garði, að hún er ákveðin af tilvist vallar og aðstöðu. Ein aðal-ástæðan er reyndar sú að við höfum hér tvö einindi í sömu töflunni: Umsjónarmann og æfingavöll. Í rauninni hefðum við átt að leysa þetta strax á fyrsta stigi. Hvað um það.

Hvort er umsjónarmaðurinn umsjónarmaður fyrir völlinn eða aðstöðuna? Lausnin á þessu er sú sama og þegar við glímum við fyrsta stigs skipulag; skipta upp í töflur:

Tafla: Vellir

Völlur	Aðstaða	Staðsetning	Umsjón
Leirárvöllur	Leirdúfupallur	201	1
Sandárvöllur	Skotmörk	220	2
Leirárvöllur	Skotmörk	201	1

Tafla: Umsjónarmenn

Nr.	Nafn
1	Örn Valsson
2	Smyrill Þrastarson

Við höfum nú skipt gögnunum upp í viðeigandi einindi. Hver völlur er sjálfstætt hugtak og á heima í eigin töflu. Hver umsjónarmaður er líka sjálfstætt einindi og á heima í eigin töflu, sú tafla gæti líka geymt upplýsingar um fleiri starfsmenn sem geta gegnt hlutverkum í ýmsum öðrum upplýsingum sem við geymum, svosem ritari, formaður o.s.frv.

Við höfum lokið annars stigs skipulagi og greint í sundur þau gögn sem lýstu aðeins að hluta þeim lykli sem einkenndi þau.

Þriðja stigs skipulag

Við erum komin að þriðja stigs skipulagi: Ef einhverjir dálkar töflunnar lýsa öðrum dálkum hennar sem aftur lýsa lykli hennar. Þetta virkar dálítið langsótt en við megum ekki gleyma takmarkinu: Skipulag er aðferð sem nýtist okkur til að greina gagnagrunn í eitt skipti fyrir öll, svo ekki þurfi að endurhanna grunninn né forritin sem verka á hann þegar þau eru komin í notkun.

Hér á eftir fer taflan Umsjónarmenn með viðbættum þrem dálkum, símanúmer umsjónarmanns, maki og sími maka. Margt má setja út á þessa töfluhönnun. Hvað ef Örn Valsson skiptir um maka en við viljum eiga áfram nafn fyrri maka og símanúmer? Hvað ef Erla Tjaldsdóttir hefur tvo síma og við þurfum að vita þá báða? Hvað ef Örn Valsson hættir en við þurfum áfram að vita af Erlu? Hér eru alltof mörg frávik á ferðinni til að við getum sæst á þessa töflu. Fyrir utan þá staðreynd að dálkurinn „Sími Maka“ eru gögn sem lýsa einindinu „Maki“ sem aftur er eigindi á einindinu „Umsjónarmaður“.

Nr.	Nafn	Sími	Maki	Sími Maka
1	Örn Valsson	999-1234	Erla Tjaldsdóttir	999-4321
2	Smyrill Þrastarson	999-0123	Hrefna Fálkadóttir	999-3210

Lausnin á þeim vanda sem hér kemur upp er sú sama og upp hefur komið aftur og aftur: Skipta gögnunum upp í frekari töflur. Við þá skiptingu beitum við sömu aðferðum og áður, að finna og skilgreina einindi töflunnar og aðgreina þau eigindi sem lýsa þeim.

Hér höfum við greinilega tvö vel skilgreinanleg einindi: Umsjónarmennirnir Örn og Smyrill. Báðir hafa þeir fengið úthlutað númeri sem lykli og látum við hann standa. Starfsmaður 1 er þannig „Örn Valsson“ og hefur hann tvö eigindi (e. Attribute) eða eiginleika: Nafn og Sími. Við getum vissulega sagt sem svo að maki sé eiginleiki hans, en er það svo í raunveruleikanum? Er maki persóna rétt eins og umsjónarmaður er persóna? Hefur maki ekki eigindin Nafn og Sími rétt eins og umsjónarmaður? Eru einstaklingur og maki hans ekki bara skyld einindi sem má tengja saman? Svörin við þessum spurningum hvetja okkur til að skipta gögnunum á eftirfarandi hátt:

Tafla: Umsjónarmaður endurnefnd sem Einstaklingar_ekki_félagar

Nr.	Nafn	Sími	Maki	Titill
1	Örn Valsson	999-1234	3	Umsjónarmaður
2	Smyrill Þrastarson	999-0123	4	Umsjónarmaður
3	Hrefna Fálkadóttir	999-3210	1	NULL
4	Erla Tjaldsdóttir	999-4321	2	NULL
5	Rauðbrystingur Svansson	999-1122	7	Formaður
6	Álft Spóadóttir	999-2211	5	NULL
7	Lóa Gauksdóttir	999-3344	5	NULL

Hér höfum við rambað á nýja lausn sem hentar langtum betur. Í fyrsta lagi getum við skráð alla hugsanlega einstaklinga sem starfa fyrir félagið í eina vel skilgreinda töflu. Við þurfum því ekki lengur sérstaka töflu fyrir umsjónarmenn æfingavalla. Taflan Vellir hefur þegar verið skilgreind þannig að þar sé skráð hverjir í „Einstaklingar_ekki_félagar“ séu umsjónarmenn sínir. Að auki getum við nú skráð hverjir séu núverandi makar og einnig geymt upplýsingar um fyrrverandi maka. Í dálkinum Titill er skráður titill viðkomandi starfsmanns félagsins. Sé skráð NULL í dálkinn (ekkert gildi til eða þekkt) þá vitum við að viðkomandi er ekki starfsmaður heldur einhver sem tengist starfsmanni.

Einstaklingur nr. 5 sýnir þetta vel. „Rauðbrystingur Svansson“ er formaður félagsins og maki hans er einstaklingur nr. 7; „Lóa Gauksdóttir.“ Við getum hins vegar séð að „Álft Spóadóttir“ (Nr. 6) er ekki starfsmaður félagsins en að maki hennar sé Rauðbrystingur. Með samanburði þessara talna sjáum við að Álft er fyrrverandi maki Rauðbrystings og Lóa núverandi. Nú vaknar sú spurning hvort við ættum að geyma upplýsingar um fleiri einstaklinga t.d. börn starfsmanna, en það er kannski full langt gengið?

Það kann að virka dálítið tyrfið að sneiða gögnin niður á þennan hátt en það borgar sig alltaf þegar til lengri tíma er litið. Engin gagnagrunns hönnun stenst fyrstu lýðfyllingu (e. Population), eða sé vitnað í Moltke: „engin hernaðaráætlun stenst fyrstu snertingu við vígvöllinn.“

Sí og æ gerist það þegar við hönnum gagnagrunna að gögn eru sett inn í þá og hönnun skilar ekki tilgangi sínum. Hönnunin er þá endurskoðuð og aftur prófað að setja inn gögn. Verulega leiðinlegt er að setja gagnagrunn í almenna notkun og þurfa svo að endurhanna hann vegna einhverra smáatriða. Þetta er ein af ástæðunum fyrir því að við hönnum hvern gagnagrunn minnst þrisvar í stílabók áður en við svo mikið sem snertum lyklaborð tölvunnar.

6. Æfing: Jólakortagrunnur 2. útgáfa

Nú væri þjóðráð að endurhanna jólakorta grunninn að nýju:

1. Hverju var sleppt við fyrstu hönnun?
2. Finndu öll frávik og hannaðu grunninn samkvæmt reglunum um fyrsta, annars og þriðja stigs skipulag.

Töflur, skyldleikar og vensl

Taflan

Hingað til höfum við rætt um hvernig upplýsingar eru teknar og sniðnar til svo að þær passi inn í töflur. Hver tafla er mengi sambærilegra eininda sem til eru í viðfangs-veruleikanum. Við höfum séð hvernig við tökum hvert einindi fyrir sig og klippum niður í smærri einingar sem lýsa einindinu. Þessar smærri upplýsingar nefnum við eigindi (e. Attributes) eða eiginleika.

Hvert einindi á sína eigin röð (e. Row) eða línu (e. Tuple) sem einnig er nefnd færsla (e. Record). Þessi þrjú hugtök eru notuð jöfnum höndum í gagnagrunns umfjöllun en eiga öll við sama fyrirbærið. Hver eiginleiki einindis hefur sinn dálk (e. Column) sem einnig er nefnt svið (e. Field), eru þessi tvö hugtök einnig notuð jöfnum höndum og tákna hið sama. Skal hver slíkur dálkur geyma ákveðið atómlegt gildi sem sé endanlegt í sjálfu sér: Þó símanúmeri megi skipta upp í röð tölustafa þá er það röð stafanna sem táknar tiltekið símanúmer sem er hið eiginlega gildi; það er því atómsgildi í þeirri merkingu að því verður ekki skipt upp frekar.

Þegar einindum sem deila sömu eiginleikum er staflað saman mynda þau töflu (e. Table) eða vensl (e. Relation) eru þessi tvö hugtök notuð jöfnum höndum og tákna hið sama. Til að skilja hugtakið vensl í þessu sambandi mætti umorða það sem ættingi. Þannig er litið á töfluna sem stafla af einindum sem öll deila sömu eiginleikum. Einindin eru sömu tegundar (e. Type) eða sama tags. Taflan er tvívíð þannig að hún hefur breidd og hæð. Breiddin er fjöldi dálka og hæðin er fjöldi lína.

Töflubreidd er jafnan föst því við höfum ákveðinn fjölda dálka en hæðin er jafnan óákveðin því fjöldi færslna er sjaldnast þekktur fyrirfram. Stöðugt má búast við því að línum sé bætt inn eða þeim eytt og því er línufjöldinn aðeins sá fjöldi eininda sem við höfum skrásett hverju sinni. Litið er svo á að röð línanna sé ekki fyrirfram þekkt.

Við getum vissulega litið svo á að færslur í tiltekinni töflu séu til í þeirri röð sem þær eru færðar inn og við getum vissulega litið svo á að þeim sé raðað eftir lykli þeirra. Því er þó ekki þannig farið því við vitum ekki hvernig gagnagrunns þjónninn geymir færslurnar eða raðar þeim við geymslu. Við högnumst á þessu því við getum ákveðið birtingarröð færslnanna eftir því hvernig við lítum á gögn töflunnar. Lítum á fjórar mismunandi upprröðun einstaklinga í „Einstaklingar_ekki_félagar“ töflunni:

Röðun 1: Raðað eftir Nr.	**Röðun 2: Raðað í stafrófsröð**	**Röðun 3: Raðað eftir Sími.**	**Röðun 4: Raðað eftir Titill**
Örn Valsson	Álft Spóadóttir	Erla Tjaldsdóttir	Hrefna Fálkadóttir
Smyrill Þrastarson	Erla Tjaldsdóttir	Lóa Gauksdóttir	Erla Tjaldsdóttir
Hrefna Fálkadóttir	Hrefna Fálkadóttir	Hrefna Fálkadóttir	Álft Spóadóttir
Erla Tjaldsdóttir	Lóa Gauksdóttir	Álft Spóadóttir	Lóa Gauksdóttir
Rauðbrystingur Svansson	Rauðbrystingur Svansson	Örn Valsson	Rauðbrystingur Svansson
Álft Spóadóttir	Smyrill Þrastarson	Rauðbrystingur Svansson	Örn Valsson
Lóa Gauksdóttir	Örn Valsson	Smyrill Þrastarson	Smyrill Þrastarson

Hér birtist hluti af þeim krafti (e. Dynamics) sem fæst við að líta á töflurnar sem óraðaðar í upphafi; við getum litið á þau einindi sem þær geyma í hvaða röð sem er og einangrað okkur við hvaða gildi sem er. Í þessu dæmi voru t.d. sendar fjórar fyrirspurnir á töfluna:

a: Birtu öll gildi í dálkinum Nafn, fyrir allar færslur, raðað eftir Nr. (í hækkandi röð).

b: Birtu öll gildi í dálkinum Nafn, fyrir allar færslur, raðað eftir Nafn (í stafrófsröð).

c: Birtu öll gildi í dálkinum Nafn, fyrir allar færslur, raðað eftir Sími (í hækkandi röð).

d: Birtu öll gildi í dálkinum Nafn, fyrir allar færslur, raðað eftir Titill (stafrófsröð NULL gildi fyrst).

Við gætum útvíkkað þessa fyrirspurn þannig: Birtu öll gildi úr dálkunum Nafn og Sími þar sem Titill er ekki NULL, raðað í stafrófsröð. Niðurstaðan yrði þannig:

Nafn	Sími
Rauðbrystingur Svansson	999-1122

Smyrill Þrastarson	999-0123
Örn Valsson	999-1234

Ásýnd

Þannig getum við haldið áfram endalaust, að horfa á upplýsingar taflnanna rétt eins og okkur sýnist. Það sem gerir venslaða gagnagrunna hins vegar kraftmikla og sveigjanlega er að við getum tengt saman töflur með ólíkum upplýsingum og skoðað áfram nánast frá hvaða sjónarhóli sem við kærum okkur um.

Sumum byrjendum í gagnagrunns meðhöndlun finnst þetta e.t.v. dálítið óþægilegt. Þannig lagað sjáum við aldrei gögnin í heild sinni eins og þegar þau eru geymd, miðað við t.d. töflureikni eða ritvinnslu sem birta okkur ávallt gögn skjala sína í hrárri eða áþreifanlegri mynd.

Gagnagrunnur geymir gagnabing sinn á diski en við höfum aðeins ákveðna ásýnd (e. View) á þau. Síðan fer það eftir fyrirspurnum sem við gerum hversu mikið magn gagnanna (eða upplýsinganna) við sjáum og á hvaða máta við sjáum þau.

Þegar við skoðum til að mynda samtengd gögn svosem: „Jón Jónsson, Aflagrun 4, 229 Hafnarfirði," höfum við ekki hugmynd um að gildið „Hafnarfirði" gæti verið sótt í aðra töflu. Ásýndin birtir gögnin sem eina heild en grunnurinn geymir þau í tveim samtengdum töflum. Eftir því sem reynsla við notkun grunna eykst hjá notendum snúast óþægindin að jafnaði í öryggi þegar þeir læra að treysta og stjórna þessu fyrirbæri.

Hvaða gögn eru geymd

Hér fyrir framan skilgreindum við tvær töflur. Taflan Völlur sem lýsir æfingavöllum sem tiltekið skotveiðifélag á og rekur. Í kjölfarið skilgreindum við töfluna Einstaklingar_ekki_félagar (stytt í Einst-aklingar) sem lýsir einstaklingum sem starfa hjá félaginu og mökum þeirra. Sú tafla, þú giskaðir rétt, er enn ekki fullhönnuð en það má bíða í smá stund.

Eitt andartak skulum við rifja upp hvernig þessar töflur hófust: Skotveiðifélagið rekur tvo æfingavelli og hefur hvor völlur að lágmarki eitt æfingasvæði jafnvel fleiri. Hver völlur hefur umsjónarmann.

Ef við lítum á vellina getum við séð þá þannig í spjaldskrá:

Völlur: Leirárvöllur. Kópavogi.	Aðstaða: Leirdúfupallur, Skotmörk.
Umsjón: Örn Valsson, sími 999-1234, sími maka: 999-3210. Maki: Hrefna Fálkadóttir.	

Völlur: Sandárvöllur. Hafnarfirði.	Aðstaða: Skotmörk.
Umsjón: Smyrill Þrastarson, sími 999-0123, sími maka: 999-4321. Maki: Erla Tjaldsdóttir.	

Lykilatriði að gagnagrunns hönnun er að vita hvaða upplýsingar við viljum skoða og hvernig. Með öðrum orðum þá hönnum við gagnagrindurnar (e. Schema) með það í huga hvernig við viljum sjá gögnin síðar.

Hér höfum við sett upp tvö spjöld fyrir tvo æfingavelli. Við viljum vita nafn vallarins og hvaða aðstöðu hann býður uppá. Einnig viljum við vita hver umsjónarmaður vallarins er, símanúmer hans og til öryggis símanúmer maka, í þeim tilfellum sem ekki næst í umsjónarmann. Eðlilega viljum við vita hvað makinn heitir.

Hingað til hefur verið rætt ítarlega um hvernig við tökum upplýsingarnar og greinum þær eða skipuleggjum í töflur og hvaða aðferðum er beitt til þess. Við höfum greint að hver völlur hefur nafn, aðsetur, aðstöðu og umsjónarmann. Við höfum einnig greint að hver umsjónarmaður er í rauninni starfsmaður (eða einstaklingur) innan félagsins sem hefur síma, maka og titil.

Þannig er alltaf farið að: Fyrst eru sett upp spjöld með þeim upplýsingum sem við viljum skoða, á eins marga vegu og þurfa þykir. Þannig er greint hvernig notendur vilja skoða gögnin síðar. Því næst eru upplýsingarnar greindar niður í eiginleika og þeir hópaðir saman eftir þeim einindum sem eiga þá. Því næst eru töflur smíðaðar utan um einindin.

Skyldleikar vensla

Þegar einindum hefur verið skipt upp í mismunandi töflur svo sem hér hefur verið gert er búið að rífa einindin úr samhengi hvert við annað. Rifjum upp töfluna Völlur:

Tafla: Völlur

Völlur	Aðstaða	Staðsetning	Umsjón (umsjónarmaður)
Leirárvöllur	Leirdúfupallur	201	1
Sandárvöllur	Skotmörk	220	2
Leirárvöllur	Skotmörk	201	1

Hér sjáum við nafn, aðstöðu, staðsetningu og umsjón. Þetta eru eiginleikar vallarins. Í dálkinum Umsjón höfum við sett inn númer sem eitt og sér segir okkur ekki neitt. Ef einhver myndi líta á töfluna án þess að þekkja þarfalýsingu (e. Requirements) skotveiðifélagsins myndi dálkurinn virðast merkingarlaus. Ef við hins vegar breytum nafni dálksins Umsjón í „Umsjónarmaður“ skýrist merking talnanna þar. Við vitum nú að gildið Umsjónarmaður í færslunni fyrir Leirárvöll (Umsjón = 1) vísar á eða merkir starfsmann númer 1, sem væntanlega finnst í töflunni Einstaklingar.

Ef við hins vegar rifjum upp töfluna Einstaklingar sjáum við að hver starfsmaður hefur samskonar tölu í dálkinum Nr. Ef við tengjum töflurnar saman á þessum númerum getum við skoðað upplýsingar þeirra eins og væru þær í sömu töflunni. Töflurnar eða venslin eru nú skyld á eigindunum „númer umsjónarmanns“ í báðum töflunum. Þó dálkarnir hafi ekki sömu heitin þá hafa þeir sömu gildistegundina þ.e. gildið táknar hið sama og er innan sama gildamengis.

Tafla: Einstaklingar (makar ekki sýndir)

Nr.	Nafn	Sími	Maki	Titill
1	Örn Valsson	999-1234	3	Umsjónarmaður
2	Smyrill Þrastarson	999-0123	4	Umsjónarmaður
5	Rauðbrystingur Svansson	999-1122	7	Formaður

Það sem við gerum er að skilgreina ásýnd, þ.e. einhverja leið til að horfa á upplýsingar beggja taflna. Við skilgreinum ásýndina þannig að við sjáum aðeins þær færslur sem eru skyldar. Til að geta skoðað færslur sem eru skyldar þurfum við að skilgreina þennan skyldleika. Hér liggur beint við hver skyldleikinn er: Númer umsjónarmanns.

Að skilgreina ásýnd í gagnagrunns fræðunum er í rauninni svipað og að skilgreina fyrirspurn (e. Query) sem vinsar upplýsingar út úr töflu eða töflum og birtir. Mörg gagnasafns-umsjónarkerfi leyfa að niðurstöður úr slíkum fyrirspurnum séu vistaðar og eru þá raunverulegar ásýndir eða Views.

Þegar einindi eru sett í sitthvora töfluna þurfum við að gæta þess að vita hvað tengir þær saman, þ.e. hvað eiga þau sameiginlegt? Við vitum að hver völlur á umsjónarmann, númer umsjónarmanns er því einn af eiginleikum vallarins. Við vitum þ.a.l. að hver umsjónarmaður á sérstakt númer.

Við vitum því að Vellir og Umsjónarmenn eiga eitt sameiginlegt atriði og getum tengt þá saman með því. Fyrirspurnin sem við gerum á gagnagrunninn er þannig: Birtu allar færslur í Völlur og allar færslur í Einstaklingar, en aðeins þar sem gildið Umsjónarmaður og gildið Nr. er sama talan. Niðurstaðan er sýnd hér á eftir:

Ásýnd: Vellir og einstaklingar tengdir saman á skyldleika sínum:

Völlur	Aðstaða	Staðsetning	**Umsjónarmaður**	**Nr.**	Nafn	Sími	Maki	Titill
Leirárvöllur	Leirdúfupallur	201	**1**	**1**	Örn Valsson	999-1234	3	Umsjónarmaður
Sandárvöllur	Skotmörk	220	**2**	**2**	Smyrill Þrastarson	999-0123	4	Umsjónarmaður
Leirárvöllur	Skotmörk	201	**1**	**1**	Örn Valsson	999-1234	3	Umsjónarmaður

Þessi hugsun er lykilatriði vensla-líkansins. Við tökum upplýsingar og greinum þær niður í sjálfstæð atriði eða einindi, sem við setjum í töflur. Því næst finnum við hvaða eigindi þessi atriði eiga sameiginleg og tengjum töflurnar saman á þeim. Engin takmörk eru á því hversu margar töflur við getum sett upp og getum við tengt þær allar saman ef þetta er rétt[32] gert.

Æfing í venslum

Við skilgreindum töfluna Einstaklingar þannig að hver einstaklingur hefur eitt símanúmer. Nú kemur upp sú staða að einstaklingar hafa farsíma. Já, sú var tíðin að þetta var ekki séð fyrir í gagnagrunnum! Nú þyrftum við að breyta töflunni og bæta við sérstökum dálki fyrir þessar upplýsingar. Nú viljum við skrásetja vinnusíma þeirra sem eru í aukavinnu eða dagvinnu t.d. má búast við því að makar hafi dagvinnu. Einnig má búast við því að einhver hafi tvo farsíma, eða fleiri.

Við getum ekki endurskilgreint töflurnar í hvert sinn sem við þurfum að setja inn ný, hugsanleg, gildi. Eina leiðin til að gera þetta er að klippa símanúmerin út og setja í sérstakar töflur. Við tökum því símanúmerin algjörlega út úr töflunni Einstaklingar og setjum í sérstaka töflu. Sumum kann að þykja þetta yfirdrifið; fleiri töflur, fleiri töflur. Þó við höfum vanist þeirri hugsun í myndrænni hugsun að „minna sé meira“ þá á það ekki endilega við um töflufjölda. En smærri, niðurgreindari gögn geta verið betri gögn en það krefst fleiri troga[33].

Tafla: Símanúmer

Númer_einstaklings	Sími	Tegund númers
1	999-1234	Vinnusími
2	999-0123	Vinnusími
1	111-0123	NMT
1	222-0123	GSM

[32] Rétt er hér notað í merkingunni árangursríkur.

[33] Að taflan sé gagnatrog eða ílát undir gögn svipað og mjólkurtrog er ílát undir mjólk.

1	999-3210	Heimasími
1	222-0124	GSM
3	999-3210	Heimasími
3	222-0125	GSM

Hér höfum við hannað töflu sem geymir símanúmer og tegund númers. Freistandi væri að setja upp símanúmerið sem einkvæman lykil. Hins vegar getur komið upp sú staða að tveir einstaklingar deili sama símanúmeri. Þá þyrftum við að skilgreina flóknari tengingu við töfluna og við viljum forðast flækjur. Flóknara er verra, einfaldara er betra. Einfalt er hér notað í merkingunni; einföld vel skilgreind gögn gera tengingar einfaldar en það krefst oft fleiri taflna.

Við gætum t.d. lent í því að glata símanúmerum ef tveir einstaklingar deila sama númerinu. Þyrftum við þá að eyða símanúmeri þegar eyða þyrfti út einstaklingi, ef tveir eru tengdir á sama númerinu gætum við ekki eytt því. Afleiðing af slíku gæti verið að eftir fimm ára notkun ættum við átt 300 símanúmer í grunninum en enginn tengdist þeim; umfremdargögn.

Við skilgreinum þess vegna samsettan lykil: Númer einstaklings og Sími séu einkvæmur lykill því sú samsetning myndi aldrei koma fyrir tvisvar í töflunni enda ólögmætt í venslalíkani.

Við höfum skilgreint skyldleika þessarar töflu við töfluna Einstaklingar á númeri einstaklings í þeirri töflu. Rifjum aðeins upp töfluna Einstaklingar:

Tafla: Einstaklingar (endurhönnuð, aðeins umsjónarmenn sýndir og einn maki)

Nr.	Nafn	Maki	Titill
1	Örn Valsson	3	Umsjónarmaður
2	Smyrill Þrastarson	4	Umsjónarmaður
3	Hrefna Fálkadóttir	1	NULL

Nú viljum við sjá öll símanúmer sem tengjast starfsmanninum Örn Valsson. Við myndum þá orða hana þannig: birtu nafn starfsmanns í Einstaklingar sem hefur Nr. = 1 og öll símanúmer úr töflunni Símanúmer þar sem Númer_einstaklings er = 1. Niðurstaða fyrirspurnarinnar væri eftirfarandi ásýnd:

Nafn	Númer_einstaklings	Sími	Tegund númers
Örn Valsson	1	999-1234	Heimasími
Örn Valsson	1	111-0123	NMT
Örn Valsson	1	222-0123	GSM
Örn Valsson	1	333-0123	Vinnusími
Örn Valsson	1	222-0124	GSM

Ef við viljum líka sjá símanúmer maka myndum við umorða fyrirspurnina þannig: Birtu Nafn einstaklings úr Einstaklingar sem hefur Nr.=1 eða Maki=1, birtu einnig öll símanúmer þar sem Númer_einstaklings er = Nr.

Niðurstaðan gæti verið á þá leið:

Nafn	Númer_einstaklings	Maki nr.	Sími	Tegund númers
Örn Valsson	1	3	999-1234	Vinnusími
Örn Valsson	1	3	111-0123	NMT
Örn Valsson	1	3	222-0123	GSM

Örn Valsson	1	3	999-3210	Heimasími
Örn Valsson	1	3	222-0124	GSM
Hrefna Fálkadóttir	3	1	999-3210	Heimasími
Hrefna Fálkadóttir	3	1	222-0125	GSM

Venslategundir

Þrjár mismunandi tegundir vensla eru notaðar þegar töflur eru tengdar saman. Eins og við höfum séð, þegar við sníðum upplýsingar til og setjum inn í töflur þá þurfum við að tengja töflurnar saman til þess að geta séð venslaðar, skyldar, upplýsingar.

Í hvert sinn sem slík tenging er framkvæmd er það gert með notkun skyldra eða sameiginlegra eiginda á milli taflnanna. Þrjár tegundir slíkra tenginga er til, „Einn á einn“ (e. One-to-One), „Einn á marga“ (e. One-to-many) og „Margir á marga“ (e. Many-to-Many).

Einn á einn (One to One)

Töflutengingin „Einn á einn“ virkar þannig að í hvorri töflu má tiltekið eigindi (eiginleiki) vera skráð aðeins einu sinni þegar töflurnar eru tengdar saman.

Tökum sem dæmi að skotveiðifélagið eigi þrjár skammbyssur sem félagsmenn geta fengið til afnota til að æfa notkun slíkra voðatóla. Reglur félagsins kveða á um að ábyrgur meðlimur þess skuli gæta hverrar skammbyssu og aldrei fleiri en einnar í senn. Af þessum sökum þurfum við að fela þrem meðlimum það hlutverk að geyma byssurnar þ.e. þær eru þrjár.

Taflan yfir skammbyssur félagsins er þannig; kennitala þess sem geymir byssuna, númer byssunnar, heiti, hvenær hún var keypt, seld (ef svo kynni að vera), einnig er skilgreind tegund. Síðar kynni félagið að kaupa aðrar tegundir s.s. vélbyssur o.s.frv. og gæti þá þurft að gera fyrirspurn á borð við; birtu allar færslur þar sem Tegund=Vélbyssa.

Tafla: Voðabyssur

KT_ábyrgðarm	Raðn	Heiti	Keypt	Seld	Tegund
0012342229	sk-0003	Holt	1/3/2004	NULL	Skammbyssa
0123451119	sk-0002	Keith & Messon	1/2/2004	NULL	Skammbyssa
1234561239	sk-0001	Muger	1/1/2004	NULL	Skammbyssa

Tafla: Félagsmenn

Kennitala	Nafn	Gælunafn	Aðsetur	Lögheimili
1234561239	Gikkur Gikksson	NULL	3	1
0123451119	Skjótur Brunason	NULL	NULL	2
0012342229	Gjóla Hittin Miðsdóttir	Gola	NULL	3

Hér sjáum við að taflan Voðabyssur er skilgreind þannig að sviðið Raðn er einkvæmur lykill töflunnar þannig að ómögulegt væri að slá inn sama raðnúmerið tvisvar. Þannig getum við ekki skrásett sama vopnið tvisvar.

Þessi tafla var áður hönnuð undir „Einkvæmt en ekki lykill“ en þar var sviði KT_ábyrgðarm haft sem lykill og Raðn sem einkvæmt svið. Hér er því réttilega snúið við þ.e. Raðn er eðlilegt lykilsvið fyrir voðavopnið og eigindin sem lyklinum fylgja lýsa því nánar. Kennitalan er hins vegar ekki lýsing á voðavopninu heldur tenging við félagsmenn. Snúið?

Við skilgreinum sviðið KT_ábyrgðarm þannig að það sé einkvæmt (skáletrun) svo tryggt sé að væri keypt fjórða voðaverkfærið yrðum við að slá inn kennitölu fjórða félagsmannsins sem gæslumanns.

Einn á marga (One to Many)

Í dæmunum hér að framan höfum við notað ákveðna tegund tengingar sem nefna mætti sem „Einn á marga“ vensl (e. One-to-one relationship). Einn á marga virkar nákvæmlega eins og sýnt var í dæminu um skotveiðifélagið hér fyrir framan. Þar skilgreindum við töfluna Einstaklingar og getur hver einstaklingur verið skráður einu sinni og aðeins einu sinni í töfluna. Hver einstaklingur hefur lykil sem við nefndum Nr. Í töflunni Símanúmer skilgreindum við hins vegar að Númer_einstaklings getur komið oft fyrir og þá í tengslum við eitt sérstakt símanúmer.

Þannig má líta á að hver einstaklingur kemur aðeins einu sinni fyrir en getur haft mörg símanúmer. Táknum þetta með mynd:

Tafla: Einstaklingar

Nr.	Nafn	Maki	Titill
1	Örn Valsson	3	Umsjónarmaður
2	Smyrill Þrastarson	4	Umsjónarmaður
3	Hrefna Fálkadóttir	1	NULL

Tafla: Símanúmer

Númer_einstaklings	Sími	Tegund númers
1	999-1234	Vinnusími
2	999-0123	Vinnusími
1	111-0123	NMT
1	222-0123	GSM
1	999-3210	Heimasími
1	222-0124	GSM
3	999-3210	Heimasími
3	222-0125	GSM

Margir á marga (Many to Many)

Tengingin „Margir á marga“ virkar þannig að ein tafla geymir margar færslur af sama tagi og getur tengst mörgum öðrum færslum í annarri töflu.

Segjum sem svo að félagið skrásetji í sérstaka töflu þegar félagsmenn koma á einhvern æfingavallanna að æfa sig í skotfimi. Við þurfum þá að skrásetja eftirfarandi upplýsingar: félagsmaður, dagsetning, völlur og aðstaða. Við gætum þurft að skrásetja hvern félagsmann oft inn í æfingatöfluna og hvern völl (eða aðstöðu) sömuleiðis.

Taflan mun því hafa margar færslur sem geyma kennitölu, einskonar „Ein á margar“ tenging frá Félagsmenn á æfingatímana. Einnig mun taflan geyma margar færslur sem tilgreina æfingavöll og þá æfingaaðstöðu sem notuð er, sem er aftur „Ein á margar.“ Óbeint erum við þá komin með þá stöðu að allir félagsmenn geta æft sig oft á öllum völlum!

Taflan „Æfingatímar,“ ásamt tengingum gæti þá litið út eins og eftirfarandi mynd sýnir:

Tafla: Félagsmenn

Kennitala	Nafn	Gælunafn	Aðsetur	Lögheimili
1234561239	Gikkur Gikksson	NULL	3	1
0123451119	Skjótur Brunason	NULL	NULL	2
0012342229	Gjóla Hittin Miðsdóttir	Gola	NULL	3

Tafla „Æfingatímar

Kennit.	Dags	Völlur	Aðstaða
1234561239	1/7/2004	Leirárvöllur	Leirdúfupallur
1234561239	2/7/2004	Sandárvöllur	Skotmörk
1234561239	3/7/2004	Leirárvöllur	Skotmörk
0123451119	1/7/2004	Leirárvöllur	Leirdúfupallur
0123451119	2/7/2004	Sandárvöllur	Skotmörk
0123451119	3/7/2004	Leirárvöllur	Skotmörk

Tafla: Völlur

Völlur	Aðstaða	Staðsetning	Umsjónar maður
Leirárvöllur	Leirdúfupallur	201	1
Sandárvöllur	Skotmörk	220	2
Leirárvöllur	Skotmörk	201	1

Einnig mætti sýna „Margir á marga“ þannig: Allir meðlimir geta tekið þátt í öllum klúbbum og allir klúbbar geta talið alla meðlimi.

Nr.	Klúbbur	Athugasemdir
1	Leirdúfuklúbbur	Þriðjud ...
2	Skammbyssuklúbbur	Hittast á ...
3	Riffilklúbbur	Skjóta ...
4	Mauser klúbbur	Þýskir herriflar ...

Tafla: Skotklúbbur

Kennitala	Klúbbur
1234561239	1
0123451119	1
0012342229	1
1234561239	2
0123451119	2
0012342229	2
1234561239	3
0123451119	3
0012342229	3

Tafla: Klúbbþátttaka

Kennitala	Nafn	Gælunafn	Aðsetur	Lögheimili
1234561239	Gikkur Gikksson	NULL	3	1
0123451119	Skjótur Brunason	NULL	NULL	2
0012342229	Gjóla Hittin Miðsdóttir	Gola	NULL	3

Tafla: Félagsmenn

Framandlyklar og aðallyklar

Þegar tafla er skilgreind og lykluð þá reynum við ávallt að finna lykla sem lýsa hverju einindi töflunnar. Við vitum t.d. að póstnúmer á Íslandi eru ávallt einkvæm. Kópavogur hefur póstnúmer 200 og ekkert annað póstsvæði deilir sama póstnúmeri. Póstnúmerið er því einkvæmt lýsandi gildi póstsvæðis og skyldi notað sem aðallykill. Ímyndum okkur að á okkar landsvæði væru aðeins fjögur póstsvæði og við værum búin að setja þau upp í eftirfarandi töflu:

Tafla: Póstsvæði

Póstnúmer	Svæðisheiti
200	Kópavogur
300	Akranes
400	Ísafjörður
500	Staður

Framandlykill (e. Foreign key) er það gildi í töflu sem vísar á aðallykil í annarri töflu og myndar þannig bein töfluvensl (e. Relationship) eða skyldleika eru á milli taflnanna. Með beinum venslum er átt við að gagnagrunnurinn hefur skrásetta tengingu á milli taflnanna. Hingað til höfum við aðeins litið á tenginguna sem hugtak en mörg gagnasafns-umsjónarkerfi leyfa að töfluvensl[34] séu skráð sérstaklega með töflunum. Töfluvensl eru ávallt skrásett þannig að aðallykill einnar töflu tengist beinlínis við framandlykil í annarri töflu. Framandlykillinn er sjaldnast skráður sem einkvæmur lykill í hinni töflunni.

[34] Hugtakið Vensl merkir töflu. Hugtakið töfluvensl merkir tengingu eða skyldleika á milli tveggja vensla.

Ímyndum okkur að við geymum upplýsingar um viðskiptavini, við geymum Kennitölu, nafn, heimili og póstnúmer:

Tafla: Viðskiptavinir

Kt	Nafn	heima	Póstnúmer
1234560001	Njáll	Njálsborg	200
1234560002	Páll	Pálsborg	200
1234560003	Gunna	Gunnutröð	300

Hér sjáum við að taflan geymir engar upplýsingar um póstsvæði viðskiptavina enda óþarfi. Við eigum nöfn þessara póstsvæða í töflunni Póstsvæði. Við getum hvenær sem er fundið póstsvæðin með því að bera saman gildið póstnúmer í báðum töflunum. Þar sem Póstnúmer er einkvæmt lykilsvið í töflunni Póstsvæði þá er aðeins um eitt nafn að ræða hverju sinni. Hins vegar geta margir aðilar átt heima á sama póstsvæði eins og við sjáum í töflunni Viðskiptavinir.

Til þess að ná Svæðisheitum póstsvæða fram þegar við t.d. lítum á alla viðskiptavini í Kópavogi, myndum við gera eftirfarandi fyrirspurn: „Finna Nafn, heimilisfang, Póstnúmer í Viðskiptavinir og Svæðisheiti í Póstsvæði, þar sem póstnúmer er 200.“

Í þessari fyrirspurn lítum við svo á að Póstnúmer í Viðskiptavinir sé framandlykill á samskonar (einkvæmt) svið í Póstsvæði. Þegar við gerum fyrirspurnina tengjum við báðar töflurnar saman á milli aðal- og framandlykla.

Niðurstaða fyrirspurnarinnar væri þá:

Kt	Nafn	heima	Póstnúmer	Svæðisheiti
1234560001	Njáll	Njálsborg	200	Kópavogur
1234560002	Páll	Pálsborg	200	Kópavogur

Vísunar-heilleiki

Í flestum gagnagrunns umsjónarkerfum er hægt að skilgreina sérstaklega vísunar-heilleika[35] (e. Referential Integrity) og krefjast þess að kerfin gæti heilleika gagna í samtengdum töflum. Þessi möguleiki var lengi vel ekki til staðar í MySQL og var mikið kvartað yfir því. Slík tækni gerir ýmsar tengingar mun auðveldari fyrir forritarann á hönnunar stigi. Skiptar skoðanir eru um það hvort betra sé að láta gagnagrunns kerfið eða þau forrit sem verka á grunnana sjá um vísunar-heilleika.

Ef við notuðum vísunar-heilleika í dæminu hér fyrir framan þá myndum við skilgreina sviðið Póstnúmer í Viðskiptavina töflunni sérstaklega sem framandlykil á sviðið Póstnúmer í töflunni Póstnúmer. Gagnagrunns kerfið myndi þá tryggja að við gætum ekki sett inn póstnúmer sem ekki er til fyrir. Þannig gætum við ekki skráð inn viðskiptavin á póstnúmeri 600 því það er ekki til í póstsvæða töflunni.

Þegar framandlyklar eru skilgreindir má taka fram hvað gert er þegar heilindi þeirra eru brotin. Við getum t.a.m. skilgreint að sé einhverju breytt á aðallyklinum séu þær upplýsingar dagréttaðar á framandlyklinum. Skilgreina má Dagréttinga-skilyrði og Eyðingar-skilyrði.

Við gætum skilgreint framandlykilinn þannig að væri póstnúmeri 500 eytt úr póstsvæða töflunni þá myndu íbúar þess svæðis fá gildið NULL fyrir póstsvæði. Eða við gætum skilgreint að ef sama póstnúmer breyttist úr 500 í 509 að þær breytingar myndu fossa (e. Cascade) yfir á viðskiptavini.

[35] Referential integrity er á íslensku vísunarheilleiki samkv. Tölvuorðasafni: Sá eiginleiki samsafns af töflum að framandlyklar hafa þar engin gildi eða sömu gildi og aðallyklar í öðrum töflum.

Veik einindi

Veik einindi eru öll þau einindi sem ekki eiga skýrt auðkenndan lykil. Eins og fram hefur komið þá skal aðallykill í töflu geta staðið einn og óstuddur sem tákn fyrir einindi sín. Í töflunni Póstsvæði hér fyrir framan getur póstnúmer 400 staðið fyrir gildið „Ísafjörður." Lykillinn er því táknrænn og auðskilinn fyrir sitt póstsvæði. Eins getur kennitala staðið ein og óstudd sem merking fyrir einstaklinga.

Veikt einindi á alla tilvist sína undir öðru einindi og á engan skýrt auðgreindan lykil, heldur búum við lykilinn til. Veik einindi eru yfirleitt einhverjir hlutir eða hugtök sem ekki eiga erindi í gagnagrunninn nema sem fylgifiskar annarra eininda. Ef við t.d. skrásettum í sérstaka töflu öll samskipti við viðskiptavini okkar værum við með veikt einindi. Það skýrist þannig að í hvert skipti sem við eigum samskipti við tiltekinn viðskiptavin þá eru þau svo nátengd honum að væri honum eytt úr grunninum þá ætti að eyða samskiptasögu hans líka.

Lítum á hvernig samskiptataflan gæti litið út:

Tafla: Samskiptasaga

Auðkennisnúmer	Dagsetning	Kennitala	Glósur
1	2/5/2004	1234560001	Hann hringdi og hrósaði síðustu vörusendingu.
2	3/5/2004	1234560002	Hringdi og þurfti upplýsingar um ...
3	3/5/2004	1234560001	Hringdi og spurði um ...

Við sjáum að í hvert sinn sem samskipti við viðskiptavin fara fram myndast hugtaks-einindi (e. Conceptual Entity) sem lýsir samskiptunum. Þetta er veikt einindi vegna þess að það þarf annað einindi í töflunni Viðskiptavinir til að lýsa sér. Til dæmis í færslu 1 og 3 er það viðskiptavinurinn Njáll á Njálsborg í Kópavogi sem á viðkomandi samskipti. Ef við eyddum Njáli út úr grunninum þá væri ekkert lengur til staðar til að lýsa samskiptum númer 1 og 3.

Við gætum lyklað einindin í töflunni á samsettan lykil „Dagsetning+Kennitala" en þá myndum við takmarka viðskiptavini okkar við aðeins ein samskipti á dag. Því búum við til skáldaðan lykil eða auðkennisnúmer. Auðkennisnúmerið hækkar um einn fyrir hverja nýja færslu og skal vera einkvæmur. Nú er ekkert því til fyrirstöðu að skrá mörg samskipti á dag.

Nú væri tilvalið að skilgreina Kennitala í Samskiptasaga sem framandlykil á sviðið kennitala í Viðskiptavinir. Þannig gætum við t.d. skilgreint að ef viðskiptavini væri eytt þá myndi sú eyðing fossa yfir í Samskiptasaga og samskiptasögu hans eitt einnig þar (CASCADE ON DELETE). Sömuleiðis ef kennitala Njáls myndi breytast úr 1234560001 í 1234560010 þá myndi sú breyting fossa yfir í samskiptatöfluna (CASCADE ON UPDATE).

Auðkennisnúmer sem lyklar – sjálfkrafa teljarar

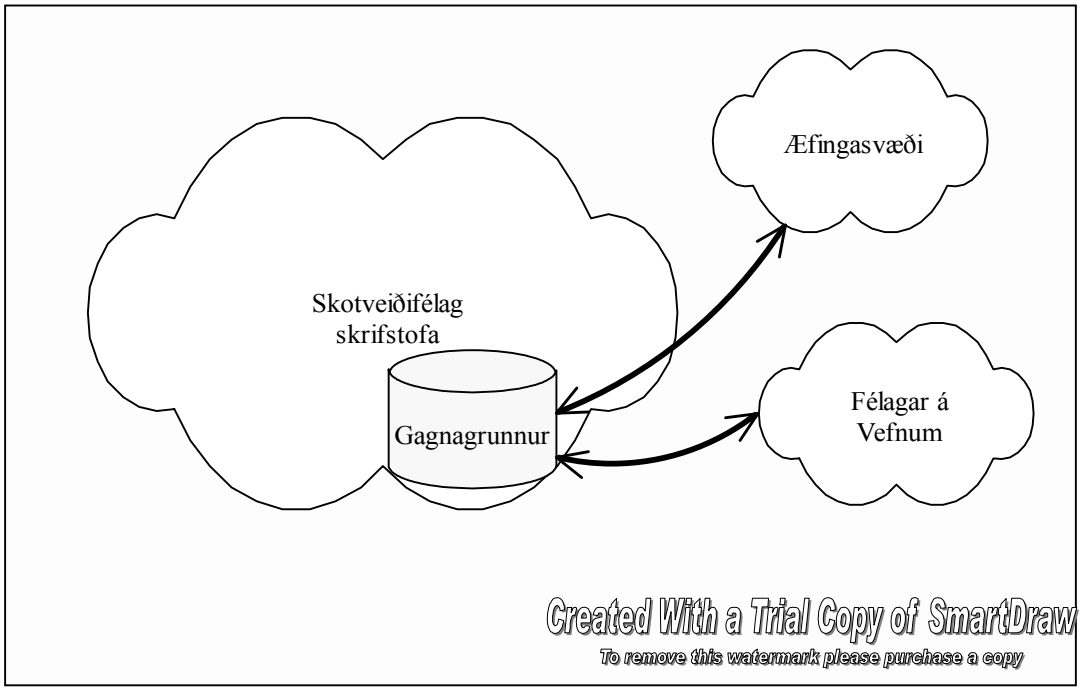
Algengt er að veik einindi fái sem lykil, tilbúin tölusett svið sem fá stighækkandi gildi eftir því sem færslum (línum) fjölgar í töflu. Undantekningalaust eru teljarar af þessu tagi innbyggðir í gagnagrunns kerfin og telja í heilum tölum. Í Microsoft Access er þessi eiginleiki skilgreindur sem „Autonumber" en í MySQL er þetta skilgreint sem AUTO_INCREMENT. Undantekningalaust eru þessir teljarar skilgreindir fyrir heiltölu dálka (e. Integer columns) og telja frá 1 og uppúr eins hátt og gildamengi dálksins leyfir.

Skotveiðifélag útgáfa 3

Það er ekki nóg að kunna á stjórntæki bifreiðar heldur þarf einnig að kunna umferðarreglurnar og skilja eðli bifreiðarinnar. Það er heldur ekki nóg að kunna að búa til gagnagrunna, töflur og fyrirspurnir í gagnasafns-forritinu sjálfu. Fyrst þarf að kunna að greina gögnin sem geyma skal og skipuleggja þau rétt. Að endingu

þarf að setja upp huglæga gagnagrind gagnanna. Best er að gera alla þessa vinnu á blaði í upphafi (eða töflureikni og/eða ritvinnslu). Mjög oft er hentugt að framsetja upplýsingar myndrænt. Framangreind útgáfa gagnagrunnsins fyrir skotveiðifélagið er myndræn samantekt á þeirra taflna sem komnar eru. Það að teikna upp töflurnar og teikna tengingar á milli þeirra er oft nóg fyrir einfalda grunna. Flóknari grunnar, við sjáum hjá skotveiðifélaginu að grunnar eru fljótir að stækka og verða flóknari, krefjast of annarra teikninga en bara töfluteikninga.

Yfirlitsteikning skotfélags



Myndin sýnir einfalda skissu að því hvernig notkunar-ferli á gagnagrunni skotveiðifélagsins myndi virka í notkun.

Gagnagrind (e. Schema) fyrir gagnagrunninn Skotveiðifélag.

Hér á eftir er listi allra þeirra taflna sem notaðar hafa verið hingað til. Mætti endurhanna þennan grunn frekar?

Tafla: Klúbbþátttaka

Kennitala	Klúbbur
1234561239	1
0123451119	1
0012342229	1
1234561239	2
0123451119	2
0012342229	2

1234561239	3
0123451119	3
0012342229	3

Tafla: Skotklúbbur

Nr.	Klúbbur	Athugasemdir
1	Leirdúfuklúbbur	Þriðjud ...
2	Skammbyssuklúbbur	Hittast á ...
3	Riffilklúbbur	Skjóta ...
4	Mauser klúbbur	Þýskir herriflar ...

Tafla: „Æfingatímar

Kennit.	Dags	Völlur	Aðstaða
1234561239	1/7/2004	Leirárvöllur	Leirdúfupallur
1234561239	2/7/2004	Sandárvöllur	Skotmörk
1234561239	3/7/2004	Leirárvöllur	Skotmörk
0123451119	1/7/2004	Leirárvöllur	Leirdúfupallur
0123451119	2/7/2004	Sandárvöllur	Skotmörk
0123451119	3/7/2004	Leirárvöllur	Skotmörk

Tafla: Voðabyssur

KT_ábyrgðarm	Raðn	Heiti	Keypt	Seld	Tegund
0012342229	sk-0003	Holt	1/3/2004	NULL	Skammbyssa
0123451119	sk-0002	Keith & Messon	1/2/2004	NULL	Skammbyssa
1234561239	sk-0001	Muger	1/1/2004	NULL	Skammbyssa

Tafla: Félagsmenn

Kennitala	Nafn	Gælunafn	Aðsetur	Lögheimili
1234561239	Gikkur Gikksson	NULL	3	1
0123451119	Skjótur Brunason	NULL	NULL	2
0012342229	Gjóla Hittin Miðsdóttir	Gola	NULL	3

Tafla: Símanúmer

Númer_einstaklings	Sími	Tegund númers
1	999-1234	Vinnusími
2	999-0123	Vinnusími

1	111-0123	NMT
1	222-0123	GSM
1	999-3210	Heimasími
1	222-0124	GSM
3	999-3210	Heimasími
3	222-0125	GSM

Tafla: Völlur

Völlur	Aðstaða	Staðsetning	Umsjón (umsjónarmaður)
Leirárvöllur	Leirdúfupallur	201	1
Sandárvöllur	Skotmörk	220	2
Leirárvöllur	Skotmörk	201	1

Tafla: Einstaklingar

Nr.	Nafn	Maki	Titill
1	Örn Valsson	3	Umsjónarmaður
2	Smyrill Þrastarson	4	Umsjónarmaður
3	Hrefna Fálkadóttir	1	NULL
4	Erla Tjaldsdóttir	2	NULL
5	Rauðbrystingur Svansson	7	Formaður
6	Álft Spóadóttir	5	NULL
7	Lóa Gauksdóttir	5	NULL

Tafla: Heimili

Færslunr.	Húsnúmer-gata	Póstnúmer
1	Skotgrund 50	0999
2	Dritgrund 55	0888
3	Hrotalundi 33	0777

Tafla: Póstnúmer

Póstnúmer	Póstsvæði
0999	Koley
0888	Haglahrepp
0777	Kúlugerði

Tafla: Byssueign

Færslunr.	Kt.	Byssa	Keypt	Seld
1	1234561239	Volga riffill	1/1/2002	NULL
2	1234561239	Brzck einhleypa	1/1/2003	NULL

3	1234561239	Colt skammbyssa 44	1/1/2003	1/5/2004
4	0123451119	Volga riffill	1/5/2002	NULL
5	0123451119	Brzck einhleypa	15/9/2002	NULL
6	0012342229	Bruun tvíhleypa	20/3/2003	NULL
7	0012342229	York veiðiriffill	31/8/2003	10/6/2004

Tafla: Félagsgjald

Kennit.	Ártal	GreittDags
1234561239	2002	1/2/2002
1234561239	2003	1/1/2004
1234561239	2004	NULL
0123451119	2002	NULL
0123451119	2003	NULL
0123451119	2004	NULL
0012342229	2003	1/2/2003
0012342229	2004	1/2/2004

Tafla: Árgjald (Ný)

Ár	Upphæð
2001	25000
2002	25000
2003	26500
2004	26500
2005	28000

Skotveiðifélag 3 – breyting

Í annarri útgáfu gagnagrunnsins Skotveiðifélag skilgreindum við töfluna félagsmenn þannig: Kennitala, Nafn, Gælunafn, Aðsetur og Lögheimili. Við skilgreindum töfluna Byssueign þannig: Færslunr., Kt., Byssa, Keypt og Seld.

Nú fær félagið bréf frá Ráðuneytinu með kröfu um að skrásetja skuli hverja byssu sérstaklega. Við eigum að skrásetja raðnúmer byssunnar (eða framleiðslunúmer), framleiðsluár og eiganda en jafnframt eru ýmsar upplýsingar sem okkur finnst viðeigandi. Þá skulum við vita hver á byssuna hverju sinni. Við viljum vita eftir sem áður sögu byssunnar þannig að við vitum hverjir hafa átt hverja byssu og þannig hverjir hafa reynslu af tiltekinni byssutegund. Þó t.d. Gikkur Gikksson eigi enga skammbyssu lengur þá getum við séð að hann átti Colt 44 skammbyssu í hálft annað ár, hann gæti því veitt tilsögn í notkun slíkra voðaverkfæra.

Tafla: Byssueign (ENDURSKOÐUÐ

Færslunr.	Kt.	Raðnúmer	Keypt	Seld
1	1234561239	RV0-O2-6789	1/1/2002	NULL
2	1234561239	HBRE-03-6789	1/1/2003	NULL

3	1234561239	SCO-03-6789	1/1/2003	1/5/2004
4	0123451119	RVO-02-6788	1/5/2002	NULL
5	0123451119	TBR-03-6789	15/9/2002	NULL
6	0012342229	TBU-03-6789	20/3/2003	NULL
7	0012342229	RYO-03-6789	31/8/2003	10/6/2004

Tafla: Byssuskrá: Nýja taflan lítur þannig út:

Raðnúmer	Árgerð	Heiti	Gerð	Vídd	Hl-lengd	Lásgerð	Mið	Ath.	Byssugerð
RV0-O2-6789	2002	Volga	Shoot	22	580 mm	Bolti			Riffill
EBR-03-6789	2003	Brzck	Plaff	12		Opn			Einhleypa
SCO-03-6789	2003	Colt 44	Army	44			Sigti		Skammbyssa
RVO-02-6788	2002	Volga	Shoot	22	580 mm	Bolti			Riffill
TBR-03-6789	2002	Brzck	Bang	12		Opn			Einhleypa
TBU-03-6789	2003	Bruun	Búmm	12		Opn			Tvíhleypa
RYO-03-6789	2003	York	Shoot	303	600mm	Bolti	Kíkir		Veiðiriffill

Eininda-vensla-teikning

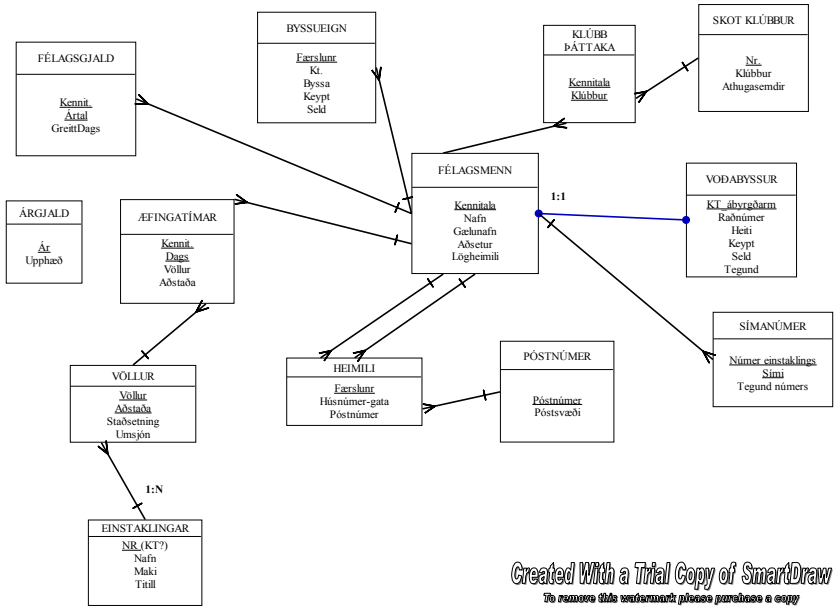
Sem fyrr segir þá er gott að gera skissur af hönnun gagnagrunna. Hér er á eftir sami grunnurinn teiknaður upp myndrænt. Tæknin sem notuð er, byggist á teiknireglum sem nefnast á ensku „Entity-Relationship-Diagram," og mætti nefna á íslenzku „Eininda-vensla-teikning." Hér eftir verður þetta fyrirbæri þó uppnefnt sem Einindateikning.

Ein ástæðan fyrir því að við teiknum grunna er sú að þá er auðveldara að tjá öðrum hvernig grunnurinn skuli hannaður. Ekki er bara um það að ræða að hönnuðurinn þurfi að skilja hvernig hans eigin grunnur skuli hannaður, einnig þarf hann að geta lesið hönnun annarra. Þá þarf hann að geta tjáð t.d. verkkaupum hvernig hann hefur skilið þarfagreininguna (e. Requirement Analysis), nú eða verkkaupinn þarf að geta skilið hönnuðinn.

Reglur um einindateikningar eru nokkuð skýrar innan gagnasafnsfræðanna. Aðeins er þó mismunandi eftir þeim forritum sem styðja slíkar tengingar hvernig reglurnar eru útfærðar. Til dæmis var ekki hægt að túlka samsetta lykla í eftirfarandi teikningum með brotastriki eins og ritreglur kveða á um. Forritið sem notað var, SmartDraw, bauð hins vegar upp á skáletrun í staðinn. Mörg forrit eru til af þessu tagi og líklega er þekktast Microsoft Visio. SmartDraw varð fyrir valinu hér því auðvelt er að nálgast prufu útgáfu þess hjá útgefanda og er forritið mjög ódýrt ef fjárfest væri í notendaleyfi.

Einindateikning 0 - Einindi sem töflur

Hér er hver tafla teiknuð upp sem kassi. Hver kassi hefur töfluheiti sitt ritað í hástöfum í efra hólfi, í neðra hólfi eru dálkaheitin rituð. Lykildálkar eru undirstrikaðir (með heilum strikum) og samsettir lyklar sömuleiðis. Allar tengingar utan ein eru „Ein á margar," táknað með beinu striki sem hefur þrífork öðru megin og þverstrik hinu megin (ein). Tengingin á milli FÉLAGSMENN og VOÐABYSSUR er „ein á eina" og er það táknað með punkti í báðum enda línunnar. Sú tenging hefur að auki rittáknið „1:1" til nánari útskýringar á því hvað við er átt. Tengingin á milli VÖLLUR og EINSTAKLINGAR hefur samskonar rithátt, „1:N" sem táknar hið sama og línan.



Einindateikning 1 – Gildi og vensl

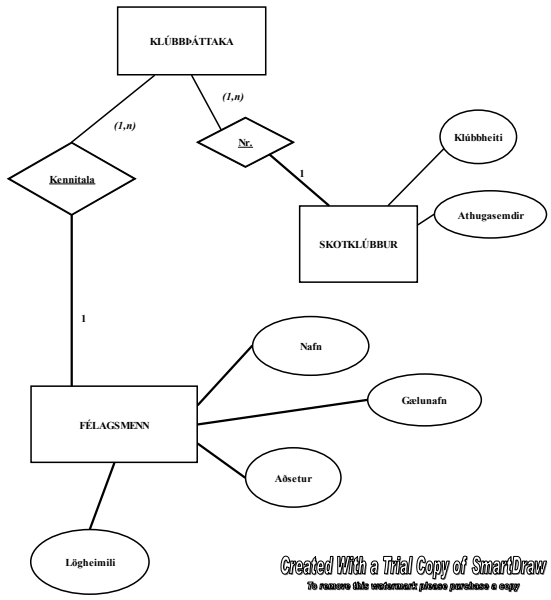
Hér er hvert einindi táknað sem ferkantaður kassi, með nafni þess ritað í hástöfum. Þau eigindi sem það er samsett úr, eru táknuð sem ýmist hringlaga eða tígullaga skapnaðir (e. Shapes) sem tengjast einindi sínu með línum.

Hringlaga skapnaðir tákna einföld eigindi sem tilheyra einindum sínum mjög ákveðið, samanber að hver skotklúbbur hefur sitt ákveðna klúbbheiti. Hver skotklúbbur er hins vegar tölusettur og hefur því gildið Nr. sem er þá lykilsvið í þeirri töflu og táknað með undirstriki.

Klúbbþátttaka geymir lista allra kennitalna sem þátt taka í klúbbum og svið þeirrar töflu eru því Kennitala og númer klúbbs. Þessi svið eru ekki lykilsvið ein sér en geta verið samsettur lykill.

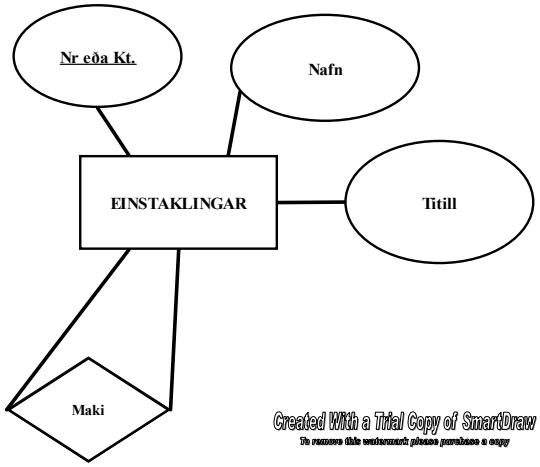
Félagsmenn hafa Kennitölu (undirstrikað), nafn, gælunafn, aðsetur og lögheimili. Þar sem lykilsviðið Kennitala tekur þátt í tengingu (töfluvenslum eða venslatengingu) við töfluna Klúbbþátttaka, er hún táknuð sem tígull frekar en hringur. Af sömu ástæðu er sviðið Nr. táknað sem tígull á milli Klúbbþátttaka og Skotklúbbur. Við þessar tvær töflutengingar er sett það ákvæði (e. Constraint) að Kennitala megi aðeins koma einu sinni fyrir í töflunni Félagsmenn en aldrei eða einu sinni eða oftar í Klúbbþátttaka. Hið sama er tekið fram fyrir tenginguna Nr.

Sú regla er ríkjandi að gildi sem mynda samsettan lykil hafi brotið undirstrik á nafni sínu. Hér hefur í staðinn verið sett sú venja að setja skáletrun ýmist i heiti gildis eða við tengingu þess við töfluna. Hér er skáletrun notuð til þess að tákna að Kennitala og Nr. mynda samsettan lykil í Klúbbþátttöku. Þetta var eingöngu vegna takmarka þess hugbúnaðar sem notaður var.



Einindateikning 2 – innri einindavensl

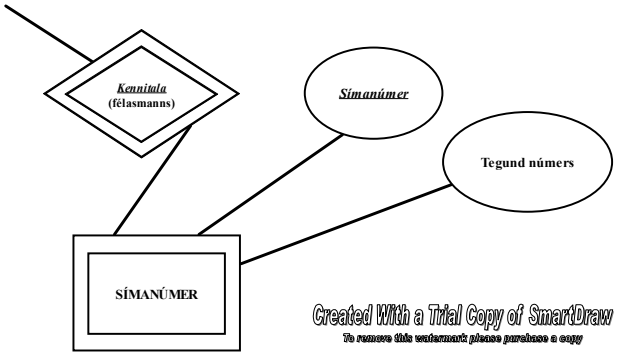
Hér er skilgreint með samskonar teikningu og áður að Einstaklingar (sem starfa hjá félaginu) eigi númer (sem mætti vera kennitala), nafn, titil og maka. Þar sem við höfum skilgreint að Maki sé númer annars einstaklings í sömu töflu, táknum við það gildi sem venslatengingu (töflutengingu) við sömu töflu. Einstaklingur 1 getur haft einstakling 2 sem maka, eftir atvikum.



Einindateikning 3 – veik einindi

Taflan Símanúmer er byggð á samsettum lykli kennitölu og símanúmers. Þetta var valið vegna þess að hvert símanúmer getur komið oft fyrir t.d. ef tveir félagsmenn búa saman og deila símanúmeri. Sömuleiðis getur félagsmaður haft mörg símanúmer og því þarf kennitalan að koma oft fyrir í töflunni.

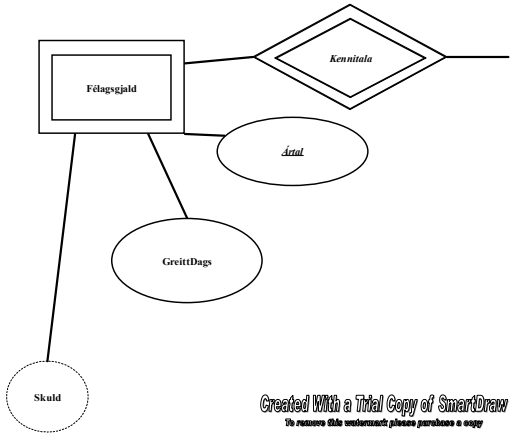
Hvert símanúmer fyrir sig er þó veikt-einindi vegna þess að það getur ekki staðið sjálfstætt – símanúmer tilheyrir t.d. eiganda sínum eða staðsetningu – þetta er að vísu teygjanlegt. Veik einindi eru öll þau einindi sem ekki eiga skýrt auðkenndan lykil heldur eiga tilvist sína komna undir öðrum einindum. Annað einindi útskýrir eðli veiks einindis. Veikt einindi er táknað með tvöfaldri línu utan um töfluna og venslatenging einnig (tígullinn).



Einindateikning 4 – Afleidd gildi

Hér er sýnt að taflan Félagsgjald er veikt einindi sem geymir Kennitölu greiðanda, ártalið sem hann á að greiða og dagsetningu greiðsludags. Væntanlega myndi gildið GreittDags hafa gildið NULL ef félagsmaður hefur ekki greitt gjaldið.

Hér er einnig sýnt gildið Skuld en það gildi er ekki geymt í töflunni heldur leitt af (e. Derived) öðrum gildum. Hér gefum við til kynna að gagnagrunnurinn skuli geta athugað að staðaldri hverjir hafi ekki greitt félagsgjaldið og reikna þannig út hugsanlega skuld. Afleidd gildi eru á ensku „Derived attributes."



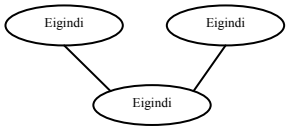
Yfirlit teiknitákna

Eininda-Vensla-Teikningar (Entity Relationship Diagrams (ERDs)), tákna rökræna uppsetningu gagnagrunns. Hér á eftir er yfirlit helstu tákna sem notuð eru við slíkar teikningar.

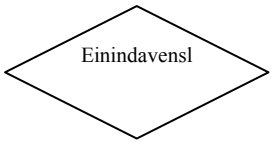
Einindi	Einindi (Entity): Einindi er hlutur eða hugtak sem til er í hlutveruleikanum (viðfangs veruleikanum) og geyma skal í töflu. Mengi samsvarandi eininda mynda töflu.
Einindi	Veikt einindi (Weak Entity): Veikt einindi er hlutur eða hugtak, geymt í eigin töflu en háð tilvist annars einindis. Mengi samsvarandi veikra eininda mynda töflu.
Eigindi	Eigindi (Attributes): Eigindi eru eiginleikar eða lýsandi gildi eininda. Eigindi mynda dálka í töflum.
Eigindi	Lykil-eigindi (Key attribute): Lykil eigindi (lykilsvið) eru einkvæm og greina á milli eininda í töflum. Til dæmis eru kennitölur fólks einkvæm gildi sem greina á milli fólks og því tilvaldar sem einkvæm lykilsvið.

Fjölgilda-eigindi (Multivalued attribute): Fjölgildi eru eigindi sem geta haft fleiri en eitt gildi, t.d. gæti einn einstaklingur haft tvö gildi fyrir dálkinn Menntun, samanber: „MD Markaðsfræði“, „PhD. Heimspeki.“ Eða Menntun: „Snyrtifræðingur,“ „Hárgreiðslumeistari.“

Afleidd eigindi (Derived attribute): Afleidd gildi eru þau sem ekki eru geymd heldur reiknuð út frá öðrum gildum: Pantanatafla geymir pantaðar einingar af vöru og hvað einingin kostar, óþarfi er að geyma verðgildi pöntunarinnar því hana má finna með formúlunni „pantaðar-einingar x einingaverð.“

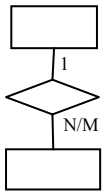


Samsett eigindi (Composite attribute): Samsett eigindi er gildi sem sett er saman úr öðrum gildum. Vel mætti hugsa sér að heimilisfang fólks væri geymt í dálkamenginu Húsnúmer og Gata. Íslensk hefð er þó sú að einn dálkur geymir hið samsetta gildi: húsnúmer-gata.



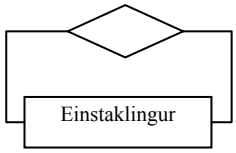
Vensl (Relationship): Vensl sýna hvernig tvö einindi deila sömu upplýsingum sín á milli.

Veik vensl (Weak relationship): Notað til að sýna hvernig veik einindi tengjast sterkum einindum.



Línufjöldi (Cardinality): Línufjöldi er notaður til þess að sýna með tölufjölda hversu mörg einindi í einni töflu geta tengst tilviki annars einindis (eða annarra) í annarri töflu.

Ákvæði, röðun (Constraint, Ordinality): Ákvæði eða röðun merkja hvort gildi sé t.d. nauðsynlegt eða valmöguleiki.



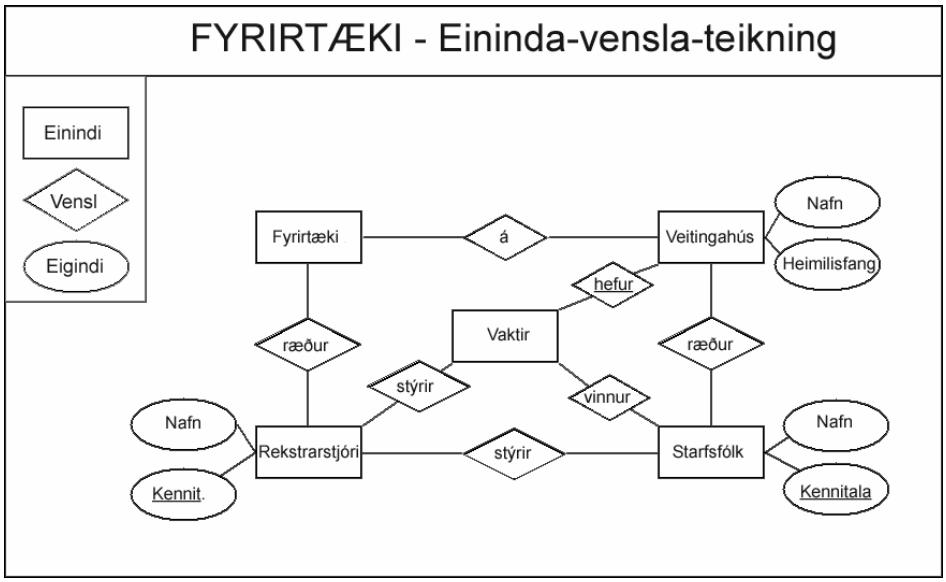
Endurkvæmt vensl (Recursive relationship): Endurkvæmt vensl er þegar gildi eins einindis er notað til að tengjast öðru einindi innan sömu töflu.

Eftirfarandi eru verkefni gera öll ráð fyrir að vera leyst í stílabók, ritvinnsluforriti eða töflureikni.

7. Æfing: Verkefni – einindateikning

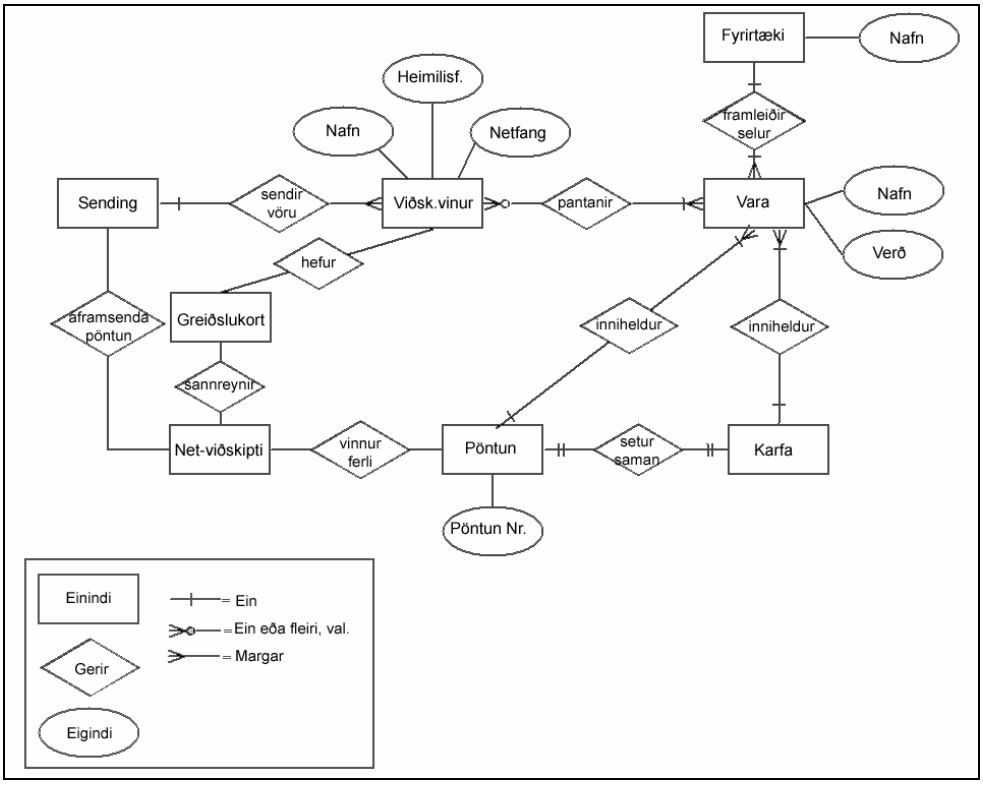
Fyrirtæki sem rekur keðju veitingahúsa biður þig að hanna gagnagrunn sem heldur utan rekstur allra veitingahúsanna. Þú færð í hendur eftirfarandi einindateikningu.

- Yfirfarðu teikninguna og bættu við þeim eigindum og venslum sem þér finnst vanta.
- Settu upp gagnagrind (e. Schema) þ.e. töflugrindur eftir teikningunni



Þekkt fyrirtæki sem stundar viðskipti á Vefnum hefur samband og biður þig að hanna gagnagrunn fyrir viðskipti sín. Þú færð í hendurnar eftirfarandi einindateikningu.

- Yfirfarðu teikninguna og gagnrýndu hana: bættu við þeim einindum og eigindum sem þér finnst vanta.
- Settu upp gagnagrind eftir teikningunni.



8. Æfing: Að vensla flata töflu

Tökum innkaupagrunn sem geymir lista yfir viðskiptavini og keyptar vörur, samanber:

dags	magn	vara	eint.verð	samtals	nafn	heima	póstsv	sími	kt	vsk.nr
1/1/2004	2 stk	Kúlupennar	256	512	Penna salan ehf	Ritstræti 5	101 Reykjavík	881-1111	123456-7899	12345
5/3/2004	5 stk	Srifblokkir	89	445	Ritfeng ur ehf	Öngstræti 6	103 Reykjavík	881-2222	654321-7899	54321

Þessi tafla er flöt þar sem geymdar eru skyldar og óskyldar upplýsingar á sama stað. Þetta er ekki æskilegt en gæti þó hugsast fyrir einhverjar aðstæður.

Nokkur vandamál birtast einmitt í töflunni:

- Greindu einindin sem koma fram í töflunni og skiptu þeim upp í töflur. Finndu sameiginleg eigindi sem töflurnar deila og gerðu ráð fyrir tengingum.
- Bættu við í.þ.m. fimm fyrirtækjum sem panta ritföng svosem; reglustikur, blýanta og heftara.

Skilgreindu (orðaðu) fyrirspurnirnar:

- Öll fyrirtæki sem hafa pantað kúlupenna. Dæmi um lausn: Velja allt úr fyrirtækjum og póstsvæðum þar sem vara = kúlupennar.
- Öll fyrirtæki sem hafa aðsetur á höfuðborgarsvæðinu.
- Öll fyrirtæki sem hafa gert viðskipti á fyrri helmingi ársins.

9. Æfing: Fréttagrunnur

Þú færð það verkefni að hanna fréttagrunn fyrir vefsetur.

1. Hvaða upplýsingar þarf að geyma? Settu svörin upp sem lista yfir upplýsingar (dálka/eigindi), því næst skaltu flokka upplýsingarnar í töflur og vensla þær.
2. Hvaða töflur þarf? Teiknaðu upp einindateikningu fyrir töflurnar í stíl við „Einindateikningu 0.“
3. Hvað þarf að koma fram á vefsíðunni og hvað ekki? Teiknaðu lausnina upp með samskonar aðferð og gert er í „Gagnagrind fyrir gagnagrunninn skotveiðifélag“
4. Hverjir þurfa aðgang að öllum gögnunum, og hvað fá net-lesendur að sjá? Hvernig myndir þú teikna það upp, eða lista það?

3. Kafli. SQL í notkun

Þessi kafli er ekki endanleg SQL yfirferð, heldur létt kynning (e. Tutorial) á því hvernig notandi ber sig að við fyrstu kynni af MySQL biðlaranum. Hvernig SQL aðgerðir eru framkvæmdar og sýnikennsla í algengustu hlutum sem framkvæmdir eru. Næstu kaflar á eftir gefa síðan hugmyndir um málfræði, gagnategundir og föll sem nota má í samskiptum við miðlarann.

SQL aðgerðaflokkar

Flokka má SQL aðgerðir gróflega í eftirfarandi fjóra flokka:

Að vinna með eða skoða gögn:

SELECT	Sækir upplýsingar í töflur.
INSERT	Skýtur upplýsingum inn í töflur.
UPDATE	Dagréttar upplýsingar í töflum.
DELETE	Eyðir upplýsingum úr töflum.
LOAD DATA	Les upplýsingar inn í töflur.
REPLACE	Skiptir út færslum í töflum.

Að vinna með gagnagrunna:

CREATE DATABASE	Smíðar nýjan gagnagrunn.
DROP DATABASE	Hendir gagnagrunni.
USE gagnagrunnur	Velur gagnagrunn til notkunar, eða gerir hann sjálfgildan.
SHOW DATABASES	Birtir þá gagnagrunna sem eru til (eða notandi fær að sjá).

Að vinna með töflur:

CREATE TABLE	Smíða nýja töflu.
ALTER TABLE	Breytir töflu.
DROP TABLE	Hendir töflu.
DESCRIBE	Birtir skilgreiningu töflunnar.
OPTIMIZE TABLE	Afslitrar (e. Defrags) töfluskrána.
LOCK TABLE	Læsir töflunni svo aðrir bíða á meðan aðgerð er framkvæmd.

Að vinna með notendur:

GRANT	Úthlutar réttindum á notendur.
REVOKE	Tekur réttindi af notendum.
DROP	Hendir notendum.
FLUSH PRIVILEGES	Sturtar notendaréttindum.

Ritvenjur

Lykilorð verða rituð í hástöfum í ritmáli en í lágstöfum í kótum[36]. Þannig verður SQL setningin „SELECT dálkur FROM tafla1;“ rituð með lykilorð í hástöfum í bókartexta. Sama setning væri þannig rituð í kóta: „`select dálkur from tafla1`“. SQL málið er hástafa-ónæmt en algengt er í dæmum og útskýringum að rita lykilorð málsins í hástöfum, til glöggvunar. Hins vegar er kvöl og pína að kóta þannig. Sum fyrirtæki setja reglur um hvernig slík kótun fer fram og þetta er reglan á SQL kótun hjá Ísbók. Sumir líta svo á að gagnagrunnar skuli alltaf vera nefndir í hástöfum og að töflu- og dálkaheiti skuli hafa stóran upphafsstaf. Regla ísbókar kveður á um að öll notendaskilgreind heiti í gagnagrunnum skuli vera í lágstöfum og samsett orð hafi undirstrik á milli orða.

Gæsalappir

Í Íslensku ritmáli eru gæsalappir ritaðar „þannig“ og ef þær eru aftast í setningu „þannig.“ Í ensku ritmáli er yfirleitt notað “þetta” (e. Double quotes, í. tvíhögg) en oft er ritað ‘þannig’ (e. Single quotes, í. Úrfellimerki). Þetta á við um almennt ritmál. Í bókinni verður það brotið upp að fylgja íslenskum reglum um gæsalappir, nema í kótadæmum þar sem enskar reglur eru notaðar eins og þær koma fyrir í handbókinni. Þegar nota þarf gæsalappir aftast í setningu utan um nöfn úr enska málinu verður punktur settur utan við gæsalöppina.

Í kótum í hefðbundnu SQL er þess krafist að úrfellingamerki sé notað utan um strengi og er þá ritað þannig: dálkur=’textastrengur innan úrfellimerkis’. MySQL leyfir hins vegar að tvíhögg sé notað í staðinn samanber: dálkur=”textastrengur innan tvíhöggs”. Einnig leyfir MySQL að bæði tvíhöggum og úrfellimerkjum sé blandað saman[37].

Algildistákn í SQL

Margir tölvunotendur hafa átt því að venjast að rittáknið „?“ sé algildistákn fyrir „eitt rittákn[38],“ og „*“ sé algildistákn fyrir „mörg rittákn, eða ekkert.“ Í SQL er undirstrik „_“ notað sem algildistákn eins rittákns og prósenta „%“ fyrir mörg eða engin.

Í fyrirspurnum er þess krafist að _ og % séu notuð í skilyrðingum sem leita að gögnum. Hins vegar má nota * og ? við ýmsar aðstæður svo sem val á töflum og dálkum, þ.e. aðstæður sem ganga nær stýrikerfinu. Notendur eru yfirleitt fljótir að átta sig á muninum.

Dæmi

Hefðbundið algildistákn sem velur alla dálka úr tafla1.

```
select * from tafla1;
```

SQL algildistákn notað til að finna alla í dálkinum nafn úr töflunni tafla1 sem innihalda „jón“ í nafni sínu.

```
select nafn from tafla1 where nafn like ‘%jón%’;
```

---

[36] Íslenska: kóti kk. [sh.] kótaþula kv. (í forritun) Bútur af forritstexta, skráður á forritunarmáli eða í formi sem smali, vistþýðandi eða annar þýðandi hefur skilað. Enska: code.

[37] Ef lesandinn er lærður í forritun þá finnst honum vafalaust ekkert mál að lesa svona upptalningar og útskýringar. Öðrum finnst þetta vafalaust vera broslegt en svona er lífið.

[38] Með hugtakinu Rittákn er átt við tölustafi frá 0..9, bókstafi frá a..z,þ,æ,ö í bæði hástöfum og lágstöfum, auk ýmissa annarra tákna s.s. `!”#$%&/()=_-~+’` (á ensku nefnd Punctuation Characters). Í DOS yrði ‘ og . og + ekki leyfð rittákn í slóðum.

SQL algildistákn notað til að finna alla í dálkinum „kennit“ úr „tafla1“ sem eru fæddir í maí 1972 á tuttugustu öld þar sem fæðingardagurinn byrjar á 2:

```
select kennit from tafla1 where kennit like '2_0572___9';
```

Skráa-afmarkari

Í tölvum er jafnan notaður skráasafns-afmarkari (e. Directory delmiter) til að afmarka heiti á skráasöfnum. Á Windows tölvum er notað öfugt-skástrik (e. Backslash) þ.e. „\“ en á Linux tölvum er notað venjulegt skástrik (e. Slash) þ.e. „/“.

Rætur MySQL þróunar liggja í Unix heiminum. Flótta-stafir eru jafnan ritaðir með \ framan við sig, samanber: \n og \t. Þá er skammritun ýmissa skipana í mysql biðlaranum ritaðar með \, samanber `\T`, `\.` og `\p`. Vegna þessa borgar sig, þegar bæði biðlaranum og miðlaranum er vísað á skrár sem skuli ýmist lesa inn eða skrifa í, að vísa í allar skrár (jafnvel í Windows) með skástriki, samanber:

```
tee c:/db-afrit/glósuskrá.sql;
source c:/db-prufur/runuskrá.sql
```

Runuskrá (e. Batch file) er skrá sem inniheldur eina eða fleiri SQL aðgerðir sem framkvæma má í runum þ.e. hverja á fætur annarri. Ef villa kemur upp í einni aðgerð þá stöðvast öll skráin svo rétt er að læra skipanirnar vel.

Kvaðning

Þegar unnið er í mysql biðlaranum birtist kvaðning[39] hans sem „`mysql>`“ og má rita inn setningar beint þar á eftir. samanber:

```
mysql> _
```

Skipun er þá rituð þannig:

```
mysql> show databases;
```

Bókin mun ekki birta kvaðninguna sjálfa í kótadæmum nema þar sem það þykir nauðsynlegt, samanber að framangreint breytist í:

```
show databases;
```

Aðeins aðgerðin er rituð í bókinni.

Sömuleiðis eru ýmsar skipanir í kvaðningunni sem skila aðeins einni svar-línu sem niðurstöðu. Yfirleitt er sú venja höfð á, ef aðgerðin heppnast, að slíkar línur eru ekki birtar í bókinni.

Aðgerðin:

```
create database firma_v1;
Query OK, 1 row affected (0.00 sec)
```

Myndi þá birtast þannig:

```
create database firma_v1;
```

Ensk heiti

Þegar notendaskilgreind nefni[40] eru smíðuð í forritun er almennt notaður enskur ritháttur. Í MySQL er þessi venja viðhöfð. Eina undantekningin sem ég þekki frá þessu er Java forritun sem leyfir að öll tungumál séu notuð.

---

[39] Enska: prompt. Íslenska: kvaðning kv. [sh.] hvatning kv. Sýnilegt eða heyranlegt boð, sent af forriti til þess að biðja um viðbrögð notanda. Kvaðning er oft sett fram á táknrænan hátt sem einn stafur.

[40] Nefni er hugtak sem í forritun táknar notendaskilgreint nafn á breytu, fasta eða skyldum þætti.

Þetta þýðir það að við skilgreinum ekki varanlega þætti svosem dálkaheiti, töfluheiti, gagnagrunnaheiti og skylda þætti með sérislenskum rittáknum. Dálkurinn „póstnúmer“ yrði því ritaður með enskum rithætti við skilgreiningu sína eða, „postnumer.“

Backus-Naur riháttur

Víða í málskipun SQL setninga[41] er notast við afbrigði af Backus-Naur rithætti. John Backus var fyrstur manna til að skilgreina sérstakt algrímsrit sem nota má til að tákna óhlutbundinn riháttt eða málskipan forritunarmála og algríma. Afbrigðið er táknað þannig:

HEITI_AÐGERÐAR notendaskilgreint nefni [BREYTA | ÖNNUR BREYTA].
HEITI_AÐGERÐAR nefni [, nefni, ...].

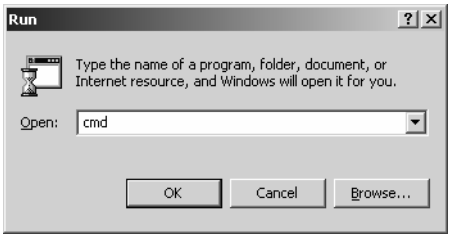
Hástafir eru lykilorð í SQL málskipan, lágstafir tákna notendaskilgreind nefni svo sem töflu eða dálkaheiti. Innan hornklofa er segð (e. Expression) sem má sleppa eftir atvikum og komma skilur á milli atriða sem koma mega oft fyrir. Efni sett innan slaufusviga má koma eins oft fyrir og þurfa þykir eða aldrei.

Eftirfarandi setningu má því lesa þannig: SELECT er nauðsynleg, lágmark er dálkurinn dálkheiti og einnig má rita dálkheiti2 og dálkheiti3. FROM er nauðsynleg og töfluheiti sömuleiðis. Að auki má tilgreina skilyrði og þá verður að rita WHERE.

SELECT dálkheiti [, dálkheiti2, dálkheiti3] FROM töfluheiti [WHERE a<1];

Skipanakvaðning

Forritið mysql er ræst úr kvaðningu á Windows tölvu. Gluggann má opna t.d. með Win+R (Windows lykill á lyklaborðinu) rita CMD og smella á OK. Einnig má músa[42] á Start – Run í sama tilgangi. Fyrir þá sem vilja fara lengri leið er músað á; Start – [All] Programs – Accessories – Command Prompt.



„Run“ glugginn í Windows sem leyfir að ræst sé skipana-kvaðning með skipuninni CMD.

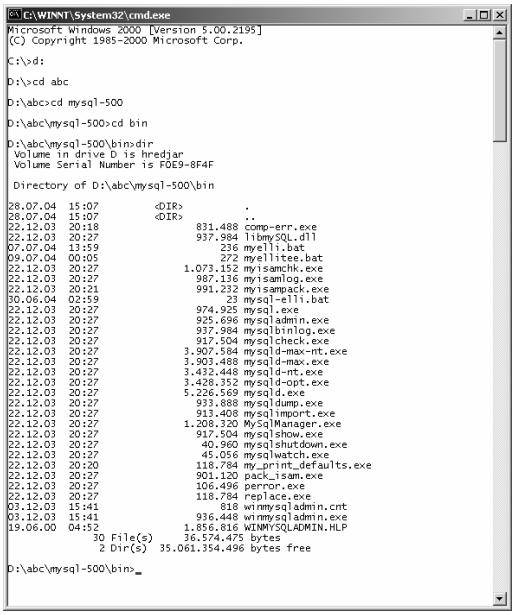
Auðvelt er að finna út hvaða DOS skipanir má nota í kvaðningu. Sé gefin skipunin „help“ og slegið inn Enter, birtist listi yfir allar DOS skipanir sem eru leyfðar í kvaðningunni. Sé ritað inn „`help skipun`“ + `Enter`, þar sem `skipun` er nafn skipunar, svo sem: „`help cd`“ birtast upplýsingar um notkun þeirrar skipunar.

Myndin hér á eftir sýnir hvar notandi er staddur í kvaðningu og fetar sig áfram yfir á D:\ drif tölvunnar, þar áfram inn í „bin“ skráasafnið og birtir þar lista yfir öll forrit sem fylgja MySQL. Kvaðningin hefur þá eiginleika að ekki væri hægt að ræsa nein þessara forrita nema fara fyrst inn í þessa möppu.

---

[41] Statement á ensku má ýmist þýða sem „setning“ eða „aðgerð.“ Yfirleitt nota ég orðið aðgerð en setning er jafnvægt orð.

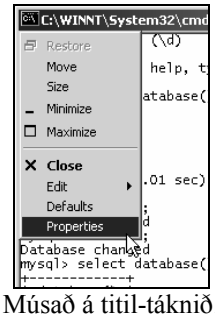
[42] Gísli Ólafur Pétursson notaði fyrstur sögnina „að músa“ í kennsluefni um tölvunotkun. Að músa merkir þá aðgerð að miða músarbendli á tákn eða skipun á skjánum og smella eða velja (smella og halda).



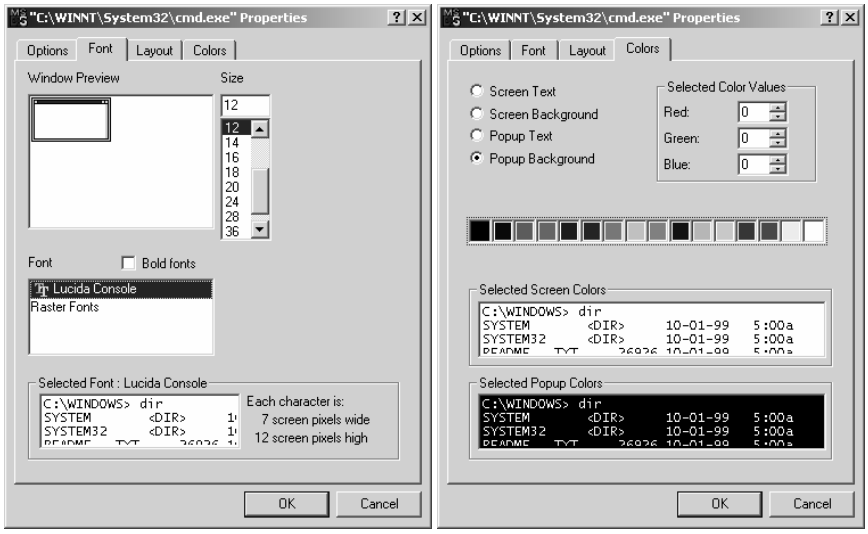
Innihald bin skráasafnsins.

Sé músað á titiltáknið lengst til vinstri á titillínu kvaðningar-gluggans má finna fáeina valmöguleika. Einn þeirra er Edit en þar undir má velja Copy, og Paste aðgerðir sem leyfa að texti sé afritaður eða límdur inn. Sömuleiðis má velja allan texta gluggans með Select All og með Mark má velja að hægt sé að blokka (velja með mús) texta (sjálfkrafa virkt í Windows 2000).

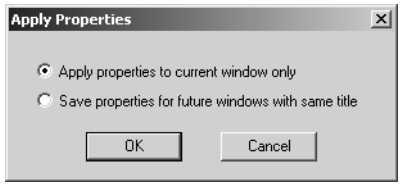
Sé músað á Properties má velja að bakgrunnur gluggans og textar séu í öðrum lit. Hér hefur t.d. verið valið að texti sé svartur á hvítum grunni. Þá borgar sig í Properties – Font að velja „True Type“ leturgerð fyrir gluggann svo hægt sé að rita íslenska stafi á eðlilegan hátt. Þegar músað er á OK er í boði að breytingarnar verið sjálfkrafa virkar upp frá því.



Músað á titil-táknið.



Font og Colors í Properties.



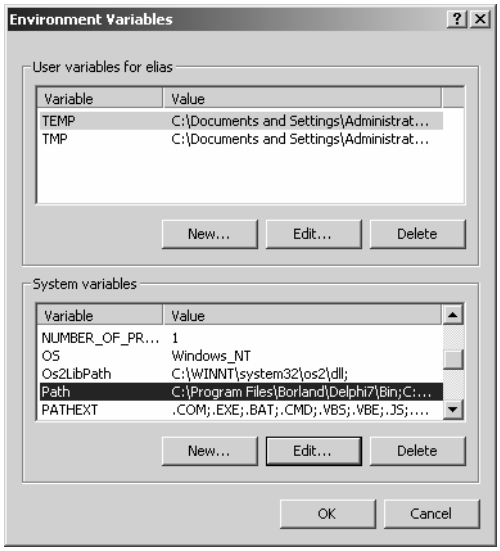
Staðfestar breytingar fyrir
a) í þetta sinnið eða b) upp frá þessu.

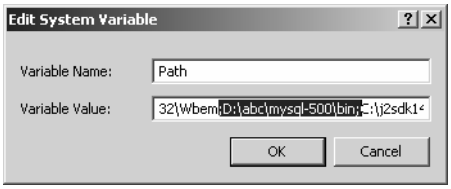
Slóð – Path

Til þess að geta ræst MySQL forritin úr kvaðningu, án tillits til þess í hvaða möppu notandinn er staddur, borgar sig að vísa á forritamöppu MySQL í PATH breytunni (í Windows). Ekki þarf að hafa fyrir þessu á Linux vél.

Á Windows XP er þetta gert þannig: Opnaðu „Control Panel“ og opnaðu þar System táknmyndina: á fremsta flipanum smellir þú á „Environment Variables“.

Á Windows 2000 er þetta gert þannig; Sama og í Windows XP nema hvað Environment Variables er á Advanced flipanum.





Myndirnar hér fyrir framan sýna hvar notandi opnar Environment Variables, fyrir „System Variables“ frekar en „User variables“ og breytir PATH breytunni. Mikilvægt er að breyta ekki þeim gildum sem fyrir eru í „Variable Value“, heldur að bæta við slóð í mysql\BIN möppuna. Slóð er bætt við með því að hafa semíkommu (;) framan við slóðina, og ef önnur slóð bætist aftan við hana, að sú slóð hafi einnig semí-kommu framan við sig.

Kerfisbreytan PATH, bæði á Linux og Windows tölvum, leyfir vísun í skráasöfn sem notandi vill geta ræst t.d. úr kvaðningu, án þess að þurfa að feta sig inn í skráasöfnin. Yfirleitt er óþarfi að stilla PATH í Linux á þennan hátt en það er til mikilla þæginda að gera það í Windows.

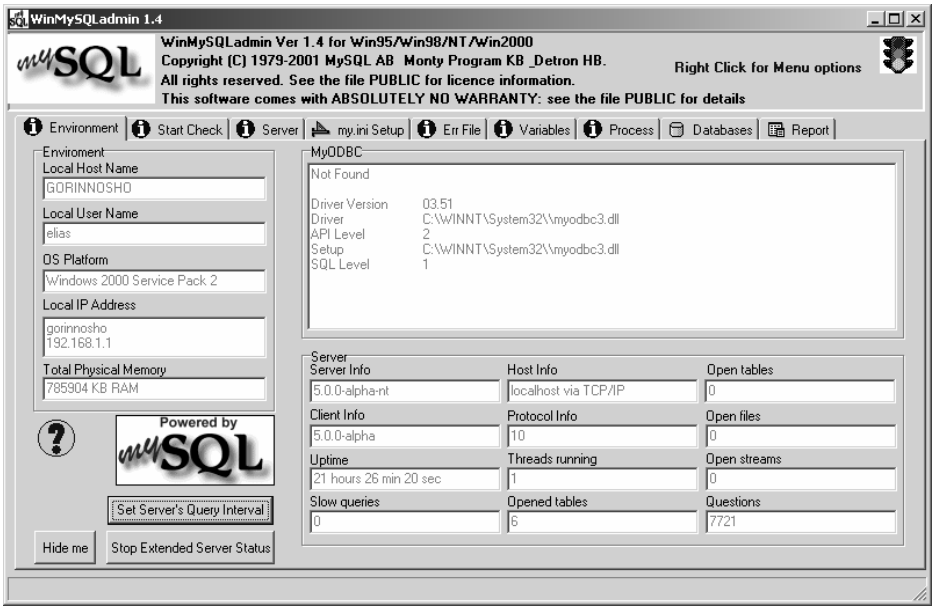
Er miðlarinn í gangi - winmyssqladmin

Þegar búið er að setja MySQL inn á Linux tölvu er miðlarinn yfirleitt ræstur sjálfkrafa, ef ekki strax þá við næstu ræsingu stýrikerfisins. Á Windows þarf hins vegar að setja hann af stað sérstaklega. Ein af þægilegri leiðum til þess er að finna forritið „winmysqladmin.exe“ í BIN möppunni og ræsa það. Bæði má músa á það í grafíska viðmóti Windows, eða slá inn nafn þess í kvaðningu.

Þegar forritið fer fyrst í gang biður það notandann um notendanafn og lykilorð og er búist við að það sé notendanafnið og lykilorðið sem kerfisstjóri miðlarans ætlar sjálfum sér. Hafi engin aðgangsstýring verið skilgreind mun þetta nafn hafa lítil áhrif. Hins vegar borgar sig, þegar aðgangsstýring er skilgreind að hafa sama nafn og lykilorð og hér er valið.

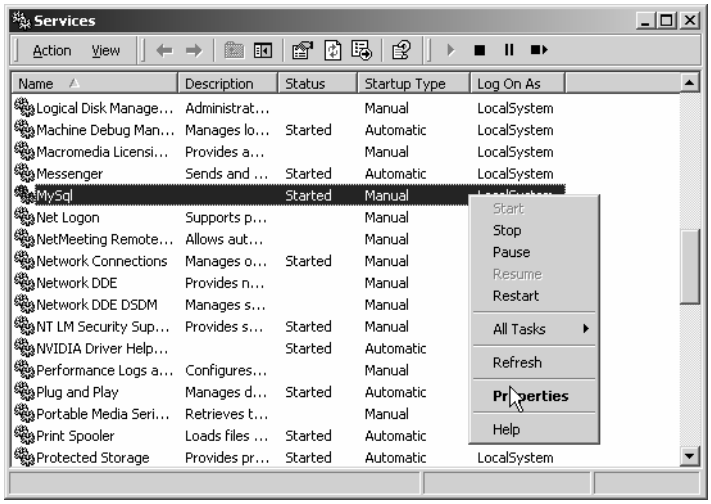
Fyrsta ræsing á winmysqladmin.

Þegar forritið er komið í gang, er miðlaraforritið mysqld sjálfkrafa sett í „Services“ kerfið í Windows. Eftir það mun miðlarinn ræstur í hvert sinn sem Windows er ræst. Það sem winmysqladmin gerir er að leyfa kerfisstjóranum að fylgjast með ástandi miðlarans. Hvað grunnar eru í gangi, hverjir eru tengdir, hvaða gildi hafa kerfisbreyturnar o.s.frv.



Grafíska forritið winmysqladmin í gangi.

Forrit þetta setur flýtivísi (e. Shortcut) í möppuna Startup út frá „Start-[All] Programs“ og má eyða því þaðan. Sumum kann þó að þykja þægilegt, á meðan þeir kynnast MySQL að leyfa því að fara sjálfkrafa í gang með Windows. Helsta notagildið sem ég hef af þessu forriti er að ritstýra (e. Edit) „my.ini“ skránni og fylgjast með gildum á þeim breytum (e. Variables) sem hafa áhrif á miðlarann.



Sé gefin skipunin Win+Run og slegið inn „services.msc“ má opna þennan glugga. Hér má gefa skipanir á borð við ræsa, stöðva og endurræsa.

mysqladmin

Í BIN möppunni er lítið og notadrjúgt forrit, „mysqladmin“ sem nota má frá kvaðningu til að fylgjast með miðlaranum og hafa áhrif á gang hans. Ein fyrsta aðgerðin sem gera ætti með þessu forriti er að setja lykilorð (e. Password) fyrir stjórnandann sem nefnist „root“.

Þetta er gert þannig:

```
mysqladmin password ****                # sé notandinn innritaður sem root.
mysqladmin -u root password ****        # notandinn tilgreinir að hann sé root.
mysqladmin -u root -p password ****     # root biður um kvaðningu lykilorðs.
mysqladmin -u root -p**** password **** # root gefur upp gamla lykilorðið strax.
```

Hér er forritið ræst úr kvaðningu ásamt stikanum[43] „password“ og þar á eftir eitthvert lykilorð. Eftir að þetta hefur verið gert, og ef root vill skipta um lykilorð þarf að nota mysqladmin með notendanafni og lykilorði.

Fleiri aðgerðir má framkvæmda, t.d. að endurræsa miðlarann með reload eða sturta niður ýmsum upplýsingum svo sem aðgangsstýringum (en þá eru þær endurnýjaðar í minni).

```
D:\abc\mysql-500>mysqladmin -u root -p***** reload
D:\abc\mysql-500>mysqladmin -u root -p***** flush-tables
D:\abc\mysql-500>mysqladmin -u root -p***** flush-hosts
D:\abc\mysql-500>mysqladmin -u root -p***** flush-privileges
D:\abc\mysql-500>mysqladmin -u root -p***** flush-logs
D:\abc\mysql-500>mysqladmin -u root -p***** flush-threads
```

[43] Orðið stiki er hér notað í sömu merkingu og færibreyta sem er enska orðið „parameter.“ Þegar forrit, og föll, eru vakin er algengt að sendar séu upplýsingar á þau með stikum. Á ensku mætti rita: program is invoked with with an argument, þ.e. forrit er vakið með stika. Forritin, og föllin, taka þá við stikanum sem færibreytu (e. Variable). Oft er ritað á ensku að senda „argument“ á fall sem taki við „parameter“ þ.e. senda frumgildi sem tekið er við sem færibreytu eða stika.

Dæmi

Hér reynir notandinn root að setja lykilorðið „gelli“. Miðlarinn neitar honum um breytinguna: í fyrsta lagi tilgreindi hann ekki gildan notanda, svo notandinn ODBC var tekinn í misgripum. Á Linux tölvu hefði hann verið skilinn sem það notendanafn sem hann skráir sig inn undir.

```
D:\abc\mysql-500\bin>mysqladmin password gelli
mysqladmin: connect to server at 'localhost' failed
error: 'Adgangur bannadur fyrir notanda: 'ODBC'@'localhost' (nota lykilord: NEI)'
```
[44]

Hér er tilgreint að sá sem vinnur verkið sé root og með „-p“ stikanum (e. Argument) segir hann forritinu að biðja sig um lykilorð. Hér gengur dæmið upp því hann er löglegur notandi og man gamla lykilorðið sitt.

```
D:\abc\mysql-500\bin>mysqladmin -u root -p password gelli
Enter password: *****
```

Hér setur hann gamla lykilorðið aftur inn í staðinn.

```
D:\abc\mysql-500\bin>mysqladmin -u root -p password gauros
Enter password: ******
```

Allsráðandi notandinn „root“

Þegar MySQL er fyrst settur inn í tölvu, og allar götur frá því, er skilgreindur einn notandi sem er allsráðandi á miðlaranum. Notendanafn hans er root, rétt eins og samskonar notandi í flestum Unix/Linux stýrikerfum. Þetta eru þó ekki sömu notendurnir.

Notandinn root hefur réttindi til að gera það sem honum sýnist í kerfinu. Venjulega er mælt með því að um leið og stjórnandi miðlarans hafi getu til, skilgreini hann annað notendanafn fyrir sjálfan sig sem hafi takmarkaðri réttindi. Ef stjórnandinn er löngum tengdur sem root eykur hann hættuna á að þrjótar (e. Malicious hackers) þefi upp aðgangsyrði hans og misnoti.

my.ini og my.cnf

Þegar miðlarinn fer í gang leitar hann að my.ini skránni í Windows. Sú skrá er yfirleitt geymd í C:\ rótinni eða í WINNT/Windows möppunni. Þessi skrá geymir allar stillingar sem þarf til að stilla bæði miðlarann og önnur forrit.

Í Linux tölvum er heiti skrárinnar my.cnf og getur hún leynst á fáeinum stöðum en aðal staðurinn er `/etc` skráasafnið og `./~` skráasafnið. Hið fyrrnefnda er þá fyrir víðværar stillingar en hið síðarnefnda er fyrir þann notanda sem ræsir t.d. biðlara.

Sökum tímaskorts og plássleysis get ég ekki gert þessari skrá fullkomin skil í bókinni. Síðar mun ég bæta úr því á heimasíðu bókarinnar og mun það skila sér í næstu útgáfu hennar. Þetta kemur þó ekki mjög illa að sök því byrjendur þurfa sjaldnast að stilla þessa skrá á fyrstu vikum og mánuðum. Einnig birti ég innihald my.ini skrárinnar í viðauka C eins og hún er stillt á Windows vélinni minni í dag. Í þeirri skrá má lesa heilmikið af ábendingum yfir það helsta sem notendur gætu þurft að nýta sér.

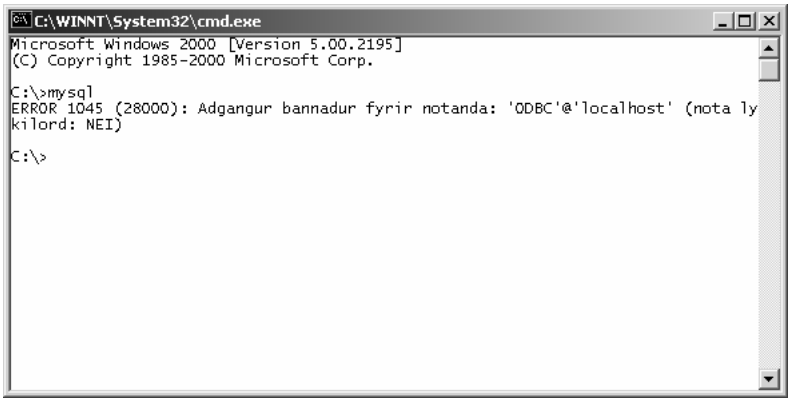
Biðlarinn mysql

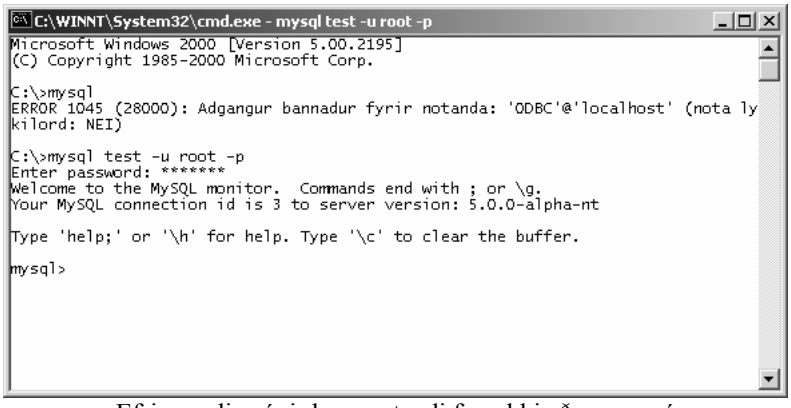
Til þess að nota MySQL þarf fyrst og fremst miðlarann, sem ræstur er með forritinu „mysqld“. Fleiri forrit fylgja vissulega og má finna þau í [MySQL-install-dir]\bin\ skráasafninu. Næst þarf biðlara til að tengjast

[44] Ég hef íslenzkað villuboðin sem miðlarinn gefur frá sér. Mun ég síðar setja þýðinguna á www.isbok.is með leiðbeiningum varðandi innsetningu. Þetta er ekki opinber íslenzkun, meira hugsuð til gamans og skilar sér óvart víðsvegar í kótadæmum bókarinnar.

þjóninum og framkvæma SQL aðgerðir. Algengt er að tengja önnur forrit svosem C eða C++ forrit eða vefsíður í formi PHP forskrifta við miðlarann.

Í tilraunum og námi er best að nota biðlaraforritið „mysql“ sem er í sömu möppu og mysqld. mysql leyfir öll samskipti við miðlarann með þeim SQL aðgerðum sem MySQL styður. Forritið er ræst úr skipanakvaðningu (e. Command Prompt), einnig nefnt texta-skel. Biðlarinn vinnur eins og kvaðning; beðið er eftir að notandinn slái inn skipun, gefin er til baka niðurstaða úr skipuninni og beðið er næstu skipunar.





Efri myndin sýnir hvar notandi fær ekki aðgang en á
á þeirri síðari fær hann aðgang og biðlarinn bíður frekari aðgerða.

Skoðað í kringum sig

Eftir að MySQL hefur verið settur inn á tölvu í fyrsta sinn getur hver sem er ræst mysql biðlarann án lykilorðs eða notendanafns. Eftirfarandi dæmi sýnir hvar mysql biðlarinn er ræstur án notendanafns og lykilorðs. Hér er búið að setja slóð fyrir „*mysql\bin*“ skráasafnið í PATH svo ræsa má hvaðan sem er:

```
C:\>mysql\bin\mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 5.0.0-alpha-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

Því næst er miðlarinn inntur eftir því hvaða gagnagrunnar séu til:

```
mysql> show databases;
+----------+
```

```
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.00 sec)
```

Nú er biðlarinn beðinn um að birta lista yfir allar innbyggðar skipanir sem hann styður, auk SQL aðgerða. Fyrst birtast vefslóðir sem nota má til að afla sér frekari upplýsinga á heimasíðu MySQL AB. Því næst birtast allar innbyggðar skipanir í þrem dálkum. Fremsti dálkurinn sýnir skipunina, mið dálkurinn sýnir skammritun (e. Shorthand) hennar sem nota má í staðinn. Aftasti dálkurinn birtir lýsingu á skipuninni.

```
mysql> help

For the complete MySQL Manual online visit:
   http://www.mysql.com/documentation

For info on technical support from MySQL developers visit:
   http://www.mysql.com/support

For info on MySQL books, utilities, consultants, etc. visit:
   http://www.mysql.com/portal

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
help    (\h)    Display this help.
?       (\?)    Synonym for `help'.
clear   (\c)    Clear command.
connect (\r)    Reconnect to the server. Optional arguments are db and host.
ego     (\G)    Send command to mysql server, display result vertically.
exit    (\q)    Exit mysql. Same as quit.
go      (\g)    Send command to mysql server.
notee   (\t)    Don't write into outfile.
print   (\p)    Print current command.
prompt  (\R)    Change your mysql prompt.
quit    (\q)    Quit mysql.
rehash  (\#)    Rebuild completion hash.
source  (\.)    Execute a SQL script file. Takes a file name as an argument.
status  (\s)    Get status information from the server.
tee     (\T)    Set outfile [to_outfile]. Append everything into given outfile.
use     (\u)    Use another database. Takes database name as argument.
delimiter       (\d)    Set query delimiter.

For server side help, type 'help all'
```

Þess má geta að í Linux útgáfunni má þreifa sig áfram eftir ábendingu neðstu línunnar, að ýtarlegri SQL aðgerðahjálp. Til að fá þessa hjálp yfir á Windows útgáfuna þarf þó að afrita innihald „help_*“ taflanna í mysql grunninum yfir í sama grunninn hinum megin.

Biðlarinn ræstur rétt

Gert er ráð fyrir því að notandinn root þurfi alltaf lykilorð til að fá aðgang að MySQL forritunum. Einnig er gert ráð fyrir því að biðlarinn sé ræstur á eftirfarandi sniði:

msyql *gagnagrunnsheiti* –u notandi –p*lykilorð* ENTER.

Þeir stikar sem hér eru sendir eru:
gagnagrunnsheiti; sá gagnagrunnur sem skal opna.

-u *notandi*; tilgreint með –u að notendanafn sé gefið upp.
-p*lykilorð*; tilgreint lykilorð með –p (alltaf áfast).

Ef ekkert lykilorð er tilgreint þá er beðið um það. Einnig má gefa upp fleiri stika svosem –P *númer* (stórt P) ef t.d. miðlarinn bíður eftir notendatengingum á öðru hliði en 3306 (sjálfgefna hliðið) má gefa upp nýtt hliðnúmer.

```
C:\>mysql test -u root -p
Enter password: *******
```

Hér tengist notandinn „root“ og tilgreinir að hann skuli beðinn um lykilorð. Kosturinn við að gefa upp lykilorð með –p stikanum er augljós, tengingin gerist hljóðlega. Sé hins vegar eingöngu tilgreint –p og biðlarinn látinn inna eftir lykilorðinu, þá er svar notandans dulkóðað áður en það er sent til miðlarans, sem á að auka öryggi gegn þrjótum.

Runuskrár og glósuskrár

Að öðrum skipunum ólöstuðum verða hér útskýrðar hvernig þrjár þessara skipana eru notaðar, „source“, „print“ og „tee“. Skipunin tee vísar biðlaranum á að glósa allar aðgerðir í sérstaka textaskrá. Þetta getur verið hentugt þegar verið er að prófa margar SQL aðgerðir sí og æ. Til dæmis þegar ég vinn í biðlaranum læt ég safna öllum aðgerðum mínum í glósuskjal, jafnvel allan daginn. Í lok dagsins eru glósurnar yfirfarnar og hent út því sem engu máli skiptir.

Biðlarinn látinn glósa allar aðgerðir í skjalið „20040716-glosur.txt“ á D: drifi tölvunnar.

```
mysql> tee d:/20040716-glosur.txt
Logging to file 'd:/20040716-glosur.txt'
```

Eins hefði mátt rita:

```
\T d:/20040716-glosur.txt
```

```
mysql> source c:/birta_grunna.sql
```

Biðlarinn látinn lesa inn runuskrána (e. Batch file) „birta_grunna.sql“ úr C: rótinni, úttakið við keyrslu hennar er:

```
Database changed
+---------------+
| Database      |
+---------------+
| firma_v1      |
| framandlyklar |
| kennsla       |
| malskipan     |
| matra_isbok   |
| mysql         |
| test          |
| world         |
+---------------+
8 rows in set (0.00 sec)
```

Hér sjáum við úttak úr einhverjum SQL aðgerðum í runuskránni, en við sjáum ekki aðgerðirnar sjálfar. Kótinn í runuskránni er mjög einfaldur, eða:

```
use test;
show databases;
```

Skipunin „print“ getur verið hentug en um leið villandi. Þó gefa megi upp skipunina með öðrum SQL setningum eða eftir að þær hafa verið framkvæmdar græðist ekkert á því.

```
select * from simi print;
+-----------+---------+
| simanumer | nafn    |
```

```
+----------+---------+
...
| 2234567   | Siggi   |
+----------+---------+
4 rows in set (0.00 sec)
```

Hér gerist það sem búast má við af SELECT aðgerð. Hins vegar ef sama skipun væri sett í runuskrá myndu koma villuboð. Sé hins vegar skammritun hennar sett í enda SQL aðgerðanna í runuskránni, samanber:

```
use test \p;
show databases \p;
```

Fæst allt annað úttak á skjáinn:

```
mysql> source c:/birta_grunna.sql
--------------
use test
--------------

Database changed
--------------
show databases
--------------

+---------------+
| Database      |
+---------------+
...klippt
| test          |
| world         |
+---------------+
8 rows in set (0.00 sec)
```

Hér fæst sama útskrift og áður þegar runuskráin var lesin inn, nema nú er einnig skrifuð út skipunin sem framkvæmdi aðgerðina.

Kosturinn við runuskrár er sá að þegar kóta þarf aðgerðir sem jafnvel spanna margar línur, getur verið betra að rita þær allar í runuskrá og lesa hana inn. Sé villa í kótanum þarf aðeins að laga eina villu. Hins vegar sé mysql biðlarinn notaður þarf að rita allar línurnar aftur.

Biðlara kvaðningin

Eftirfarandi kótabrot sýnir að reynt er að smíða töfluna „vesen“ en strax í fyrstu línu er villa þar sem gleymt er að opna sviga á eftir nafngiftinni.

```
mysql> use test;
Database changed
mysql> create table vesen
    -> nafn varchar(31),
    -> heima varchar(25),
    -> pnr varchar(3),
    -> fdagur date,
    -> kyn enum('kk','kvk')
    -> );
ERROR 1064 (42000): You have an error in your SQL syntax.  Check the manual that
 corresponds to your MySQL server version for the right syntax to use naerri 'nafn
varchar(31), ... hja linu 2
```

Hér sést að notaður er grunnurinn „test“, og smíðuð taflan vesen. Skilaboðin benda til þess að villan sé í línu 2, sem merkir að lína 1 er röng svo ekki er hægt að nálgast línu 2.

Nú þarf að rita allan kótann upp aftur. Við sjáum að þegar SQL aðgerð er framkvæmd má slá inn Enter í miðri línu samanber:

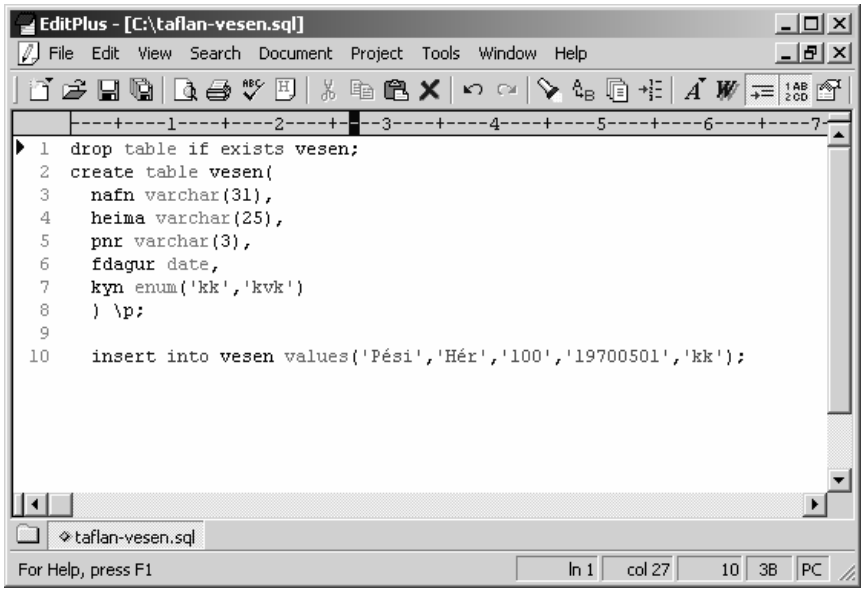
```
mysql> select
    -> *
    -> from
    -> vesen;
```

Biðlarinn tekur ekki við aðgerðinni fyrr en slegin er inn semíkomma, þá fyrst er hún innt. Einn kosturinn við þetta er að nota má örvalykla til að nálgast skipunina aftur, þannig má nálgast allar skipanir sem slegnar hafa verið inn nýlega, jafnvel þó biðlaranum sé lokað í millitíðinni.

Nú má hætta við SQL skipun í miðjum klíðum með skipuninni `clear` eða `\c`. Hér er SQL aðgerðin framkvæmd aftur en aldrei kláruð:

```
mysql> select * from vesen \c
```

Áður var minnst á að nota mætti runuskrár til að vinna margar SQL aðgerðir í einu. Myndin hér á eftir er þar sem forritið „EditPlus“ er notað til að rita skrána „taflan-vesen.sql“. Þar er gerð skipun sem hendir töflunni vesen, sé hún til fyrir. Því næst er taflan hönnuð og að lokum skotið inn einni færslu í hana. Ekkert væri því til fyrirstöðu að setja fleiri aðgerðir í skrána en hún er svo lesin inn með source (eða `\.`)



EditPlus notað við að rita SQL runuskrá.
Ending skrárinnar má vera hver sem er.

„mysql“ grunnurinn

Í „mysql“ grunninum eru skilgreindar töflur sem miðlarinn notar í aðgangsstýringar. Einnig eru þar töflur sem (á Linux í.þ.m.) má nota til að fletta upp nokkuð öflugri innankerfis (e. Online) hjálp.

Til að velja gagnagrunna er notuð skipunin „use gagnagrunnur“ og til að sjá töflur grunnsins er notuð skipunin „show tables“ sem birtir þá lista allra taflna í grunninum. Use skipunin birtir aðeins þá grunna sem notandinn hefur leyfi til að vinna með.

Til að fá upplýsingar um töflur er gefin skipunin „describe *töfluheiti*“. Sú skipun birtir alla dálka töflunnar og hvernig þeir eru skilgreindir. Þetta er ómetanlegt hjálpartæki til að meta hvernig taflan geymir upplýsingar og hvaða fyrirspurnir megi framkvæma á þeim.

```
use mysql;
```

```
show tables;
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv    |
| db              |
...klippt
| user            |
+-----------------+
11 rows in set (0.00 sec)

describe user;
+-----------------------+-----------------------------------+------+-----+-------
| Field                 | Type                              | Null | Key | Default
+-----------------------+-----------------------------------+------+-----+--------
| Host                  | varchar(60)                       |      | PRI |
| User                  | varchar(16)                       |      | PRI |
| Password              | varchar(41)                       |      |     |
| Select_priv           | enum('N','Y')                     |      |     | N
... klippt
| x509_subject          | blob                              |      |     |
| max_connections       | int(11) unsigned                  |      |     | 0
+-----------------------+-----------------------------------+------+-----+------
31 rows in set (0.03 sec)
```

Aðgangsstýring

Til að gefa notandanum „elias“ á tölvunni „localhost“ aðgang að öllum töflum mysql grunnsins notum við GRANT skipunina. Mikilvægt er að tilgreina að hann þurfi lykilorð til að tengjast. Ef hann skal geta tengst frá öðrum hýslum (e. Hosts) á „.is“ hluta Netsins, eða hvaða hýsli sem er þarf að gefa aðra GRANT skipun og jafnvel fleiri.

```
use mysql;

grant all privileges on mysql.* to 'elias'@'localhost'
identified by 'gaufast';

grant all privileges on mysql.* to 'elias'@'%.is'
identified by 'gaufast';

grant all privileges on mysql.* to 'elias'@'%'
identified by 'gaufast';


grant file on *.* to 'elias'@'localhost' identified by 'gaufast';


select User, Host, Password from user where User like ‘%elias%’;
+---------+-----------+-------------------+
| User    | Host      | Password          |
+---------+-----------+-------------------+
| elias   | %         | 64df47945e870204 |
| elias   | %.is      | 64df47945e870204 |
| elias   | localhost | 64df47945e870204 |
+---------+-----------+-------------------+
10 rows in set (0.00 sec)
```

Nú mætti eyða öllum þessum réttindum með REVOKE setningunni:

```
revoke all privileges on mysql.* from ‘elias’@’%’;
```

```
revoke all privileges on mysql.* from 'elias'@'%.is';
revoke all privileges on mysql.* from 'elias'@'localhost';
```

Eitt af því sem hér kemur í ljós er að mysql skilgreinir notanda á sniðinu „notandi-á-hýsli“ og því er elias@isbok.is ekki sami notandinn og elias@grusk.isbok.is eða elias@isbok.net.

Föll í SQL aðgerðum

Nota má öll, eða næstum öll innbyggð föll MySQL í SQL aðgerðum og án þess að tilgreina sérstaklega einhverjar töflur. Hér kíkjum við á föllin now() sem birtir dags- og tímasetningu, user() sem segir okkur notendanafn þess sem tengist og version() þ.e. útgáfu miðlarans sem er í notkun:

```
select now(), user(), version();
+---------------------+----------------+----------------+
| now()               | user()         | version()      |
+---------------------+----------------+----------------+
| 2004-07-16 13:22:37 | root@localhost | 5.0.0-alpha-nt |
+---------------------+----------------+----------------+
1 row in set (0.03 sec)
```

Fáum núgilda-dagsetningu og númer tengingarinnar sem biðlarinn kennist á (aðrir biðlarar á sama tíma myndu fá annað tengi- eða kenninúmer:

```
select current_date, connection_id();
+--------------+-----------------+
| current_date | connection_id() |
+--------------+-----------------+
| 2004-07-16   |              27 |
+--------------+-----------------+
1 row in set (0.00 sec)
```

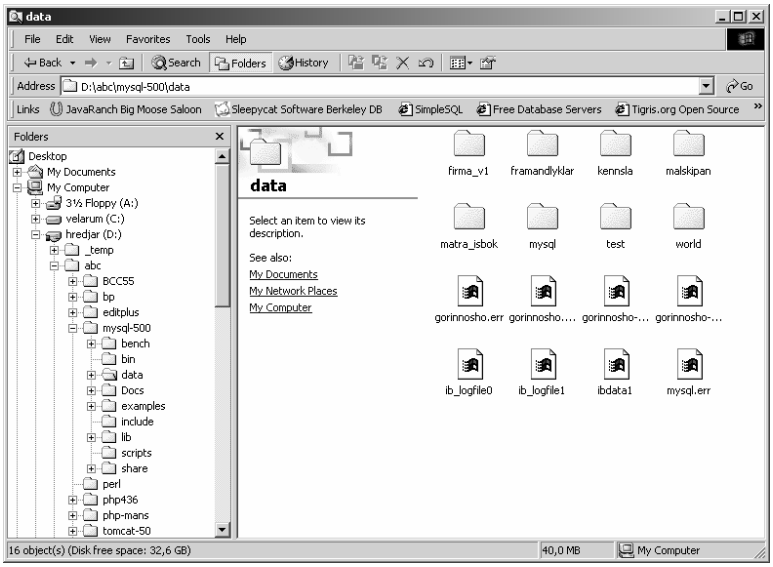
Sé MySQL biðlarinn ræstur án þess að tilgreindur sé sjálfgildur gagnagrunnur sem vinna skal á, myndi database() fallið skila NULL. Sé hins vegar búið að tilgreina gagnagrunn, annað hvort við ræsingu biðlarans eða með use setningu, skilar fallið nafni sjálfgilda grunnsins:

```
select database();
+------------+
| database() |
+------------+
| NULL       |
+------------+
1 row in set (0.00 sec)

use test;

select database();
+------------+
| database() |
+------------+
| test       |
+------------+
1 row in set (0.00 sec)
```

Snöggsoðið SQL

Að vinna með grunna

Skipunin „CREATE DATABASE gagnagrunnsheiti“ býr til tóman nýjan gagnagrunn. Gagnagrunnar tilheyra skráasöfnum (í Windows nefnt Folders eða möppur) undir „data“ skráasafninu þar sem MySQL var innsettur. Staðsetningin á Linux tölvum (Redhat) er „/var/lib/mysql“.



DATA skráasafnið á Windows vél minni. Hér sjást grunnarnir firma_v1, framandlyklar, kennsla, malskipan, matra_isbok, mysql, test og world. Einnig sjást dagbækur (e. Logs).

Með SHOW DATABASES má sjá hvaða grunnar eru til, og með „DROP DATABASE gagnagrunnsheiti“ má eyða hvaða gagnagrunni sem er. Smíðum nú grunninn firma1 og eyðum honum síðan:

```
create database firma1;

show databases;
+----------+
| Database |
+----------+
| firma1   |
| mysql    |
| test     |
+----------+
3 rows in set (0.00 sec)

drop database firma1;
Query OK, 1 row affected (0.00 sec)

show databases;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
3 rows in set (0.00 sec)
```

Nú reynum við aftur en rekumst á það að MySQL leyfir ekki bandstrik í heiti á grunnum.

```
create database firma-v1;
ERROR 1064 (42000): You have an error in your SQL syntax.  Check the manual that
corresponds to your MySQL server version for the right syntax to use near '-v1' at
line 1
```

Reynum aftur en nú höfum við undirstrik í heiti gagnagrunnsins:

```
create database firma_v1;

use firma_v1;
```

Unnið með töflur

Birtum nú allar töflur og ritum athugasemd, þó hún eigi e.t.v. ekki við hér þá er slíkt mjög hentugt í runuskrám.

```
mysql> # Hvaða töflur eigum við
mysql> show tables;
Empty set (0.00 sec)
```

Smíðum nú töfluna „skynditafla“, birtum allar töflur í grunninum og fáum að lokum hönnunarlýsingu nýju töflunnar:

```
mysql> create table skynditafla(
    -> nafn varchar(31)
    -> );

show tables;
+--------------------+
| Tables_in_firma_v1 |
+--------------------+
| skynditafla        |
+--------------------+
1 row in set (0.00 sec)

describe skynditafla;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| nafn  | varchar(31) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
1 row in set (0.00 sec)
```

Töflusmíðin hefur heppnast prýðilega, í henni er einn dálkur sem geymir texta að hámarkslengd 31 stafs. Prófum að lýðfylla (e. Populate) töfluna með einni færslu og því næst að birta allar færslur hennar. Að lokum hendum við töflunni.

```
insert into skynditafla values('Erdanu');

select * from skynditafla;
+--------+
| nafn   |
+--------+
| Erdanu |
+--------+
1 row in set (0.00 sec)
```

```
drop table skynditafla;
```

Ef taflan er ekki til kæmu villuboð þegar DROP er framkvæmt og því geta lykilorðin „IF EXISTS“ verið hentug, sérstaklega í runuskrám. Taktu eftir að MySQL spyr ekki hvort ég sé viss um að henda töflunni henni er fleygt og það er verður ekki afturkallað.

```
drop table if exists skynditafla;
```

```
drop table skynditafla;
ERROR 1051 (42S02): Unknown table 'skynditafla'
```

Smíðum nýja töflu nema í þetta sinn viljum við bæta við dálki sem setur sjálfkrafa nýtt númer fyrir hverja nýja færslu. Því skilgreinum við dálkinn sem „AUTO_INCREMENT“ en sú breyta krefst þess að dálkurinn sé jafnframt aðallykill og því skilgreinum við „PRIMARY KEY“. AUTO_INCREMENT verkar bara á heiltöludálka og best er að hafa þá UNSIGNED til að fá hámarksnýtni þeirra.

Sjálfvirka talningin byrjar í 1 og telur upp á við, allt jákvæðar tölur. SIGNED dálkur er eyðsla á minnisplássi því hann geymir einnig neikvæðar tölur en sé hann stilltur sem UNSIGNED hækkar gildissvið hans í jákvæða hlutanum um helming þ.e. það gildissvið neikvæða hlutans sem klippt var af.

Heiltölutegundin MEDIUMINT hefur til dæmis gildissvið frá -8388608 og upp í 8388607 sé hún SIGNED. Sé slíkur dálkur geymdur sem UNSIGNED færist neikvæða gildið upp í núll en tegundin spannar áfram sama gildissvið og verður því frá 0 og upp í 16777215.

Skilgreinum nú að við geymum dagsetningu með tímastimpli. Þar sem við tilgreinum ekki fyrir neinn dálk að hann geti verið NULL þá er það sjálfkrafa leyft en dálkurinn „numer“ mun aldrei vera NULL því AUTO_INCREMENT sér til þess. Sömuleiðis mun dálkurinn „dags“ heldur aldrei vera NULL því TIMESTAMP setur alltaf réttan tímastimpil í sinn dálk, nema þeir séu tveir saman þá fengi sá aftari sjálfgefna gildið ‘0000-00-00 00:00:00’.

```
mysql> create table jabba(
    -> numer int unsigned auto_increment primary key,
    -> nafn varchar(31),
    -> dags timestamp
    -> );
```

```
describe jabba;
+-------+------------------+------+-----+---------+----------------+
| Field | Type             | Null | Key | Default | Extra          |
+-------+------------------+------+-----+---------+----------------+
| numer | int(10) unsigned |      | PRI | NULL    | auto_increment |
| nafn  | varchar(31)      | YES  |     | NULL    |                |
| dags  | timestamp        | YES  |     | NULL    |                |
+-------+------------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

Skjótum nú inn tveim færslum. Við getum leyft okkur að senda NULL bæði fyrir númerið og dagsetninguna af framangreindum ástæðum:

```
insert into jabba values(null,'Jógúrt',null);
```

```
select * from jabba;
+-------+--------+---------------------+
| numer | nafn   | dags                |
+-------+--------+---------------------+
|     1 | Skyr   | 2004-07-16 14:10:55 |
|     2 | Jógúrt | 2004-07-16 14:11:09 |
```

```
+-------+--------+---------------------+
2 rows in set (0.00 sec)
```

Nú skulum við endurnefna jabba í bubbi með RENAME TABLE aðgerð. Gögnin verða áfram til staðar í töflunni.

```
rename table jabba to bubbi;
Query OK, 0 rows affected (0.02 sec)
```

Bætum nú við einum dálki aftast í töflunni sem getur geymt athugasemdir, allt að 255 stafi. Birtum því næst skilgreiningu töflunnar:

```
alter table bubbi add column athugasemd char(255);
```

```
describe bubbi;
+------------+------------------+------+-----+---------+----------------+
| Field      | Type             | Null | Key | Default | Extra          |
+------------+------------------+------+-----+---------+----------------+
| numer      | int(10) unsigned |      | PRI | NULL    | auto_increment |
| nafn       | varchar(31)      | YES  |     | NULL    |                |
| dags       | timestamp        | YES  |     | NULL    |                |
| athugasemd | varchar(255)     | YES  |     | NULL    |                |
+------------+------------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Skjótum nú einni færslu í töfluna og birtum því næst allar færslur:

```
insert into bubbi values(null,'Súrmjólk',null,'Með púðursykri');
```

```
select * from bubbi;
+-------+----------+---------------------+-----------------+
| numer | nafn     | dags                | athugasemd      |
+-------+----------+---------------------+-----------------+
|     1 | Skyr     | 2004-07-16 14:10:55 | NULL            |
|     2 | Jógúrt   | 2004-07-16 14:11:09 | NULL            |
|     3 | Súrmjólk | 2004-07-16 14:53:21 | Með púðursykri  |
+-------+----------+---------------------+-----------------+
3 rows in set (0.00 sec)
```

Gagnagrunnurinn Firma (útgáfa 1)

Við notum hér áfram grunninn „firma_v1“ sem byggður er á umfjöllun í kafla 1.

Þegar eftirfarandi kótar voru hannaðir og prófaðir, voru þeir allir ritaðir í einni runuskrá. Skráin var vistuð sem `d:/firma_1_samantekt.sql` og lesin inn með `source` skipun.

CREATE, ALTER og LOAD DATA

Hönnum skrána „postnumer“ sem mun geyma póstnúmer og póstsvæði.

```
drop table if exists postnumer;
create table postnumer(
       postnr varchar(4) not null primary key,
       postsvaedi varchar(25) not null
);
```

Sjáum okkur um hönd, endurnefnum „postsvaedi“ sem „heiti“ og bætum við dálkinum „afgreidslust“. Því næst lesum við inn textaskrá sem við bjuggum til í Excel og vistuðum sem postnumer.csv. (.CSV er staðall sem Excel þekkir fyrir textaskrár). Póstnúmeraskráin er afrituð frá www.posturinn.is.

```
alter table postnumer change postsvaedi heiti varchar(25) not null;
alter table postnumer add column afgreidslust varchar(50) null;

load data local infile
  'D:/postnumer.csv'
  into table postnumer
  fields terminated by ';'    # gæti líka verið \t fyrir Tab Delimited
  lines terminated by '\r\n'  # venjuleg línuskipting í Dos/Windows
  ignore 1 lines;             # Sleppa fyrstu línunni sem eru dálkaheiti
```

Hönnum nú töfluna „einstaklingar“ þó við höfum dálkinn „einst_nr“ sem aðallykill þá væri réttara að nota kennitölu í staðinn.

```
drop table if exists einstaklingar;

create table einstaklingar(
       einst_nr int unsigned not null primary key,
       fyrirt_nr int unsigned null,
       nafn varchar(31) not null,
       menntun varchar(50) null,
       starfstitill varchar(25) null,
       heima varchar(25) null,
       postnr varchar(4) null,
       simi_he varchar(8) null,
       simi_vi varchar(8) null,
       simi_gs varchar(8) null,
       netfang varchar(50) null
);
```

Þrjár villur munu nú koma upp:

Eftirfarandi aðgerð misheppnast (e. Error) vegna þess að strengja-afmarkari (úrfellingamerki) var ekki réttur (aftan við NULL gildið).

```
insert into einstaklingar
 (einst_nr, fyrirt_nr, nafn,menntun, starfstitill, heima, postnr) values
 (null',1,'Grjóti Stefsson','MD. Tölvunarfræði', 'Forstjóri','Aflatröð 5','0110');
```

Eftirfarandi aðgerð misheppnast vegna þess að við skilgreindum „einst_nr“ sem NOT NULL dálk. Hefði hann verið AUTO_INCREMENT þá hefði þetta verið í lagi:

```
insert into einstaklingar
 (einst_nr, fyrirt_nr, nafn,menntun, starfstitill, heima, postnr) values
 (null,1,'Grjóti Stefsson','MD. Tölvunarfræði', 'Forstjóri','Aflatröð 5','0110');
```

Eftirfarandi aðgerð misheppnast vegna þess að við skilgreindum „einst_nr“ sem einkvæmt lykilsvið með PRIMARY KEY og því getum við ekki sett „0“ tvisvar inn í dálkinn:

```
insert into einstaklingar
(einst_nr, fyrirt_nr, nafn,menntun, starfstitill, heima, postnr) values
(0,1,'Grjóti Stefsson','MD. Tölvunarfræði', 'Forstjóri','Aflatröð 5','0110')
(0,2,'Jóndi Bullsson','Ph D. Markaðsfræði', 'Forstjóri','Boðatröð 6','0210')
;
```

Við tökum nú þá ákvörðun að breyta „einst_nr“ þannig að dálkurinn verði AUTO_INCREMENT og mun það leysa öll vandamálin sem komu upp hér að framan. Hins vegar munum við ekki fá að framkvæma

breytinguna. Dálkurinn er einkvæmt lykilsvið nú þegar og MySQL birtir villuboð vegna þess að einhver gildi myndu tvítakast.

```
alter table einstaklingar
  change einst_nr einst_nr int unsigned auto_increment primary key;
```

Svo við hendum lykilsviðinu algjörlega og bætum því svo við aftur:

```
alter table einstaklingar drop einst_nr;

alter table einstaklingar
  add einst_nr int unsigned auto_increment primary key first;
```

INSERT - Innskot

Nú getum við lýðfyllt töfluna og þó við sendum NULL á AUTO_INCREMENT dálkinn skeytir MySQL því engu. Fyrst skulum við tæma töfluna:

```
delete from einstaklingar;

insert into einstaklingar values
(null,1,'Grjóti Stefsson','MD. Tölvunarfræði','Forstjóri','Aflatröð 5', '0110',
  '199-1234', '599-1234','999-1234','grjoti@aflatradir.net'),
(null,2,'Jóndi Bullsson','Ph D. Markaðsfræði','Markaðsstjóri','Boðatröð 6',
  '0210', '199-2341','599-2341','999-2341','jondi@bodatradir.net'),
(null,3,'Gudda Karlsdóttir','Ph D. Markaðsfræði', 'Forstjóri', 'Grandatröð',
  '0200', '199-3412','599-3412','999-3412', 'gudda@grandatradir.net'),
(null,1,'Fríða Fróðadóttir','MD. Heimspeki','Forstjóri','Eindartröð','0105','199-
  3412', '599-3412','999-3412','gudda@grandatradir.net')
;
```

Skoðum nú valda dálka í töflunni:

```
select einst_nr, nafn, menntun, heima from einstaklingar;
+----------+------------------+--------------------+------------+
| einst_nr | nafn             | menntun            | heima      |
+----------+------------------+--------------------+------------+
|        1 | Grjóti Stefsson  | MD. Tölvunarfræði  | Aflatröð 5 |
|        2 | Jóndi Bullsson   | Ph D. Markaðsfræði | Boðatröð 6 |
|        3 | Gudda Karlsdóttir | Ph D. Markaðsfræði | Grandatröð |
|        4 | Fríða Fróðadóttir | MD. Heimspeki      | Eindartröð |
+----------+------------------+--------------------+------------+
4 rows in set (0.01 sec)
```

Því miður er ekki hægt að skjóta inn nýjum færslum í margar töflur í senn. Fúlt því hægt er að velja upplýsingar úr mörgum töflum í senn með SELECT og hægt er að eyða færslum úr mörgum töflum í einu með DELETE. Hvað um það, eftirfarandi aðgerð mun ekki ganga upp því aðeins má lýðfylla eina töflu í senn:

```
insert into fyrirtaeki(fyrirt_nafn,fyrirt_heim,postnumer), postnumer(postnr,heiti)
values('Leppar','Grýluboð','010','010','Esjuhellar');
ERROR 1064 (42000): You have an error in your SQL syntax.  ...
naerri ' postnumer(postnr,heiti) values
... klippt
```

Að skjóta færslunum inn sitt í hvoru lagi gengur hins vegar ágætlega:

```
insert into postnumer(postnr, heiti) values('010', 'Esjuhellar');
```

```
insert into fyrirtaeki(fyrirt_nafn, fyrirt_heim, postnumer)
values('Leppar', 'Grýluboð', '010');
```

Nú má hins vegar skoða færsluna, samtvinnaða úr tveim töflum.

```
select * from fyrirtaeki, postnumer
where postnumer.postnr = '010'
and postnumer.postnr=fyrirtaeki.postnumer;
+----------+-------------+-------------+-----------+----------+---------  ...
| fyrirt_nr | fyrirt_nafn | fyrirt_heim | postnumer | adalsimi | netfang   ...
+----------+-------------+-------------+-----------+----------+---------  ...
|         4 | Leppar      | Grylubod    | 010       | NULL     | NULL      ...
+----------+-------------+-------------+-----------+----------+---------  ...
1 row in set (0.00 sec)
```

10. Æfing: Fleiri töflur í firma_v1

Næstu fyrirspurnir birta fyrstu færsluna í þrem töflum sem hanna þarf fyrir gagnagrunninn „firma_v1“. Hannaðu og lýðfylltu þessar töflur. Berðu þær saman við umfjöllunina í 1. kafla.

```
select * from fyrirtaeki limit 1;
+----------+-----------------+------------------+-----------+----------+
| fyrirt_nr | fyrirt_nafn     | fyrirt_heim      | postnumer | adalsimi |
+----------+-----------------+------------------+-----------+----------+
|         1 | Hugbrjótur ehf. | Sílikongarðar 25 | 0210      | 599-1234 |
+----------+-----------------+------------------+-----------+----------+
                                  ---------------------------+--------------------+
                                   netfang                   | veffang            |
                                  ---------------------------+--------------------+
                                   hugbrjotur@hugbrjotur.net | www.hugbrjotur.net |
                                  ---------------------------+--------------------+
                                                1 row in set (0.00 sec)
```

```
select * from launaflokkur limit 1;
+----------+--------------+------------------+---------------+-------------+
| flokk_nr | taxti_manudur | viku_vinnustundir | yfirvinnu_alag | athugas ... |
+----------+--------------+------------------+---------------+-------------+
|        1 |        160000 |               40 |          0.35 | Fyrsti ...  |
+----------+--------------+------------------+---------------+-------------+
1 row in set (0.00 sec)
```

```
select * from starfsmadur limit 1;
+------------+----------------+------------+-------------+------------+
| id_kt      | nafn           | deild      | launaflokkur | hof_storf  |
+------------+----------------+------------+-------------+------------+
| 0202802229 | Jóndi Jóndason | Tölvudeild |           5 | 2004-07-16 |
+------------+----------------+------------+-------------+------------+

                                    -------------+-----------+------------+
                                     lauk_storfum | titill    | launa_alag |
                                    -------------+-----------+------------+
                                     NULL         | Forritari |          0 |
                                    -------------+-----------+------------+
                                                   1 row in set (0.00 sec)
```

SELECT – Velja

Þegar við höfum greint upplýsingar upp í mismunandi, skyldar, töflur og þurfum að skoða þær í samhengi tilgreinum við einfaldlega töflurnar t.d. í SELECT aðgerð. Eftirfarandi aðgerð sækir alla starfsmenn og alla launaflokka. Þau mistök eru þó gerð, að MySQL veit ekki hvernig skal tengja upplýsingarnar. Til dæmis geymist gildi í dálkinum „launaflokkur“ en MySQL veit ekki að það á við númerið í dálkinum „flokk_nr“.

MySQL sækir því allar færslur í báðum töflum og fyrir hverja færslu í töflunni starfsmadur er birt hver einasta færsla í launaflokkur. Alls eru birtar 34 færslur, sem segir okkur aðeins að við höfum margar færslur en ekkert um upplýsingarnar sjálfar.

```
select id_kt, nafn, launaflokkur as flokkur,
taxti_manudur as taxti, yfir vinnu_alag as alag
from starfsmadur, launaflokkur;
+------------+------------------+---------+--------+------+
| id_kt      | nafn             | flokkur | taxti  | alag |
+------------+------------------+---------+--------+------+
| 0202802229 | Jóndi Jóndason   |       5 | 160000 | 0.35 |
| 0303812229 | Gudda Leppadóttir |       5 | 160000 | 0.35 |
| 0202802229 | Jóndi Jóndason   |       5 | 165000 | 0.35 |
...klippt
| 0303812229 | Gudda Leppadóttir |       5 | 400000 |  0.5 |
| 0202802229 | Jóndi Jóndason   |       5 | 450000 |  0.5 |
| 0303812229 | Gudda Leppadóttir |       5 | 450000 |  0.5 |
+------------+------------------+---------+--------+------+
34 rows in set (0.00 sec)
```

Við verðum því að tengja töflurnar saman í fyrirspurninni. Þetta er gert í WHERE hluta og við einfaldlega biðjum MySQL að tengja þær þannig: „birta og para saman allar færslur þar sem sama gildið er í launaflokkur og flokk_nr“:

```
select id_kt, nafn, launaflokkur as flokkur,
taxti_manudur as taxti, yfirvinnu_alag as alag
from starfsmadur, launaflokkur
where launaflokkur = flokk_nr;
+------------+------------------+---------+--------+------+
| id_kt      | nafn             | flokkur | taxti  | alag |
+------------+------------------+---------+--------+------+
| 0202802229 | Jóndi Jóndason   |       5 | 180000 |  0.4 |
| 0303812229 | Gudda Leppadóttir |       5 | 180000 |  0.4 |
+------------+------------------+---------+--------+------+
2 rows in set (0.00 sec) >
```

Ef við vildum nú sjá öll fyrirtækin í grunninum, ásamt nafni einhvers einstaklings sem vinnur hjá þeim, ásamt póststöð þá gætum við gert fyrirspurn á töflurnar „einstaklingar,“ „fyrirtaeki“ og „postnumer.“ Ef við gerum það án þess að tengja upplýsingarnar saman fáum við fáránlegar niðurstöður eins og áður

Rétt aðferð við að tengja upplýsingarnar saman væri því: birta sömu dálka og áður en samtengja starfsmenn við fyrirtæki og fyrirtæki við póstnúmer. Ekki ætti að skipta máli í hvaða röð töflurnar eru nefndar en við sumar samtengingar skiptir það máli:

```
mysql> select nafn as tengill, fyrirt_nafn as firma,
    -> fyrirt_heim as gata, postnumer as pnr, postnumer.heiti as poststod
    -> from einstaklingar, fyrirtaeki, postnumer
    -> where fyrirtaeki.fyrirt_nr=einstaklingar.fyrirt_nr
    -> and fyrirtaeki.postnumer=postnumer.postnr
    -> order by tengill;
+------------------+-----------------+-----------------+------+---------------+
```

```
| tengill           | firma           | gata              | pnr  | poststod      |
+-------------------+-----------------+-------------------+------+---------------+
| Fríða Fróðadóttir | Hugbrjótur ehf. | Sílikongarðar 25  | 0210 | Garðabæ       |
| Grjóti Stefsson   | Hugbrjótur ehf. | Sílikongarðar 25  | 0210 | Garðabæ       |
| Gudda Karlsdóttir | Kerfagutl hf.   | Miðstígum 6       | 0101 | Reykjavík - 1 |
| Jóndi Bullsson    | Ferlatækni ehf. | Sílikongarðar 50  | 0210 | Garðabæ       |
+-------------------+-----------------+-------------------+------+---------------+
4 rows in set (0.01 sec)
```

Tölfræði

Eftirfarandi aðgerð telur hverja einustu færslu í öllum töflunum þrem. Niðurstöðu hverrar talningar er haldið til haga í breytunum „`@a1..3`“ og þær margfaldaðar saman í lokin. Gættu vel að lykilorðinu DISTINCT sem notað er með hverju tilfelli fallsins COUNT() en án þess hefði komið sama gildið í öllum dálkum þ.e. 1764.

Aðalatriðið hér er þó ekki svöl SELECT aðgerð heldur að þegar tvær töflur eða fleiri eru samtvinnaðar (e. Joined) í niðurstöðum án þess að vera sérstaklega tengdar saman, verða niðurstöðurnar fáránlegar og fjöldi færslna verður margfeldi af fjölda færslna þeirra! Eftirfarandi kóti er til að sýna þetta en einnig kynning á notkunargildi COUNT() fallsins:

```
select @a1:=count(distinct einst_nr) as einstaklingar,
  @a2:=count(distinct fyrirtaeki.fyrirt_nr) as firmu,
  @a3:=count(distinct postnumer.postnr) as fj_pnr,
  (@a1*@a2*@a3)
  from einstaklingar, fyrirtaeki, postnumer group by '';
+---------------+-------+--------+---------------+
| einstaklingar | firmu | fj_pnr | (@a1*@a2*@a3) |
+---------------+-------+--------+---------------+
|             4 |     3 |    147 |          1764 |
+---------------+-------+--------+---------------+
1 row in set (0.01 sec)
```

Sömu fyrirsögn má vissulega gera þannig, sem gefur svipaðar niðurstöður og áður:

```
select count(distinct einst_nr) as einstaklingar,
  count(distinct fyrirtaeki.fyrirt_nr) as firmu,
  count(distinct postnumer.postnr) as fj_pnr
  from einstaklingar, fyrirtaeki, postnumer;
+---------------+-------+--------+
| einstaklingar | firmu | fj_pnr |
+---------------+-------+--------+
|             4 |     3 |    147 |
+---------------+-------+--------+
1 row in set (0.01 sec)
```

Ef reynt væri, svipað og gert var í launaflokka fyrirspurninni, að sækja téða dálka í þessar töflur, t.d. þannig:

```
select nafn as tengill, fyrirt_nafn as firma, fyrirt_heima as gata,
  postnumer as pnr, postnumer.heiti as poststod
  from einstaklingar, fyrirtaeki, postnumer;
```

Þá kæmu þessar niðurstöður:

```
...klippt
1764 rows in set (0.03 sec)
```

DELETE - Eyða

Ekkert er eins auðvelt í SQL eins og að eyða gögnum úr töflu: Eftirfarandi aðgerð myndi tæma töfluna fyrirtaeki, án viðvörunar. Nema einhver sjái að sér og skammritaði „clear“ (\c) í lok setningarinnar svo hætt væri við hana. Allar breytingar á gögnum í gagnagrunnum eru framkvæmdar samstundis og þýðir ekkert að láta sig dreyma um „undo“ skipun því hún er ekki til.

```
select * from fyrirtaeki\c
```

Áður en eyðingar-aðgerð er framkvæmd er best að skoða vel hvaða gögn eru til staðar í töflunni. Þá er bætur hægt að átta sig á því hvernig skuli skilgreina hverju eytt verði. Annars má búast við alls konar mistökum.

```
select fyrirt_nr, fyrirt_nafn, fyrirt_heim, postnumer from fyrirtaeki;
+-----------+-----------------+------------------+-----------+
| fyrirt_nr | fyrirt_nafn     | fyrirt_heim      | postnumer |
+-----------+-----------------+------------------+-----------+
|         1 | Hugbrjótur ehf. | Sílikongarðar 25 | 0210      |
|         2 | Ferlatækni ehf. | Sílikongarðar 50 | 0210      |
|         3 | Kerfagutl hf.   | Miðstígum 6      | 0101      |
|         4 | Leppar          | Grýluboð         | 010       |
|         6 | Jólasveinar     | Grýluboð         | 010       |
+-----------+-----------------+------------------+-----------+
5 rows in set (0.00 sec)
```

Eyðum nú fyrirtæki númer 6, engin vandkvæði. Því er eytt.

```
delete from fyrirtaeki where fyrirt_nr=6;
Query OK, 1 row affected (0.00 sec)
```

Eyðum nú öllum fyrirtækjum sem hafa „e“ sem annan staf . Engin vandkvæði hér heldur, þrem fyrirtækjum er eytt, en ég ætlaði bara að eyða „Leppar“. Er til afrit?

```
delete from fyrirtaeki where  fyrirt_nafn like '_e%';
Query OK, 3 rows affected (0.01 sec)
```

Aðgerðin DELETE leyfir að eytt sé úr mörgum töflum í einu. Næsta dæmi eyðum við öllum fyrirtækjum sem eru númer 1 úr fyrirtækja töflunni og póstnúmeri ‘010’ úr póstnúmera töflunni. Samtals verður tveim færslum eytt:

```
delete fyrirtaeki.*, postnumer.*
from fyrirtaeki, postnumer
where fyrirt_nr=1 and postnr='010';
Query OK, 2 rows affected (0.02 sec)
```

11. Æfing: Bekkjarfélag

Gerðu annaðhvort:

1. Hannaðu gagnagrunn sem geymir upplýsingar allra í bekknum þínum (sértu í skóla) og láttu gagnagrunninn geyma námssögu samnemenda þinna.
2. Hannaðu samskonar grunn og í a lið en yfir vini þína.

12. Æfing: Jólakortagrunnur

Rifjaðu upp hönnun þína á jólakortagrunni sem þú hannaðir í síðasta kafla.

1. Smíðaðu grunninn í mysql. Smíðaðu og lýðfylltu allar töflur með runuskrám.
2. Hannaðu nokkrar venslatengdar fyrirspurnir sem birta alla jólakortasöguna, svosem alla sem fjölskyldu þinni jólakort í fyrra annars vegar og hins vegar alla sem þú sendir jólakort til en ekki höfðu sent þér.

13. Æfing: Skotveiðifélagið heimsótt á ný

Líttu nú á hönnunina úr síðasta kafla fyrir gagnagrunn skotveiðifélagsins.

1. Smíðaðu gagnagrunninn og allar töflurnar.
2. Lýðfylltu þær og æfðu nokkrar SELECT aðgerðir á þeim og nokkrar DELETE aðgerðir.
3. Eyddu nú öllum töflunum en ekki grunninum sjálfum.
4. Búðu nú til runuskrá fyrir hverja töflu. Láttu skrána smíða töfluna, lýðfylla hana og birta allar færslur í lokin. Notaðu `\p` í runuskránni svo mysql prenti út allt sem gert er. Að lokum læturðu biðlarann lesa runuskrárnar inn.
5. Í lokin skaltu skilgrein í það minnsta þrjá notendur sem hafa leyfi til að framkvæma SELECT, INSERT, UPDATE og DELETE á töflur grunnsins.

4. Kafli. Yfirlit yfir MySQL forritin

MySQL dreifingin inniheldur eftirfarandi forrit (eða forskriftir):

MySQL miðlarinn og ræsi-forskriftir (e. Startup scripts):

- mysqld er MySQL miðlarinn.
- mysqld_safe, mysql.server, and mysqld_multi eru ræsi forskriftir.
- mysql_install_db frumstillir gagnagrunns skráasafnið og fyrstu grunnana.

Biðlara-forrit sem hafa samskipti við miðlarann:

- mysql er biðlari fyrir texta-kvaðningu sem leyfir allar SQL aðgerðir miðlarans, bæði í innslætti eða í runuskrám.
- mysqlcc (MySQL Control Center) er forrit með grafísku notendaviðmóti til að framkvæma sömu hluti og „mysql“.
- mysqladmin er forrit til að stjórna miðlaranum.
- mysqlcheck framkvæmir athuganir og lagfæringar á töflum.
- mysqldump og mysqlhotcopy taka afrit af grunnum.
- mysqlimport flytur inn gagnaskrár.
- mysqlshow birtir ástandsupplýsingar um grunna og töflur.

Ýmis tól og tæki, óháð miðlaranum:

- myisamchk, sjá hér ofar.
- myisampack pakkar töflum og gerir þær aðeins-lesanlegar (e.read-only).
- mysqlbinlog tæki til að lesa úr tvíundar-dagbókum (e. Binary logs).
- perror birtir merkingu á villukótum.

Lagfæringar á aðgangsstýringa-töflunum

- mysql_prepare_privilege_tables.sql og mysql_prepare_privilege_tables_for_5.sql eru runuskrár (nafn fer eftir útgáfu) sem lagfæra skilgreiningar á notendastýringa-töflum.

Flest þessi forrit fylgja MySQL og eru ýmist staðsett í „bin“ möppunni eða „scripts“ eftir verk-vöngum (e. Platform).

Að ræsa MySQL forritin

```
mysql test
mysqladmin extended-status variables
mysqlshow --help
mysqldump --user=root personnel
```

14. Æfing: Afrit og ástand

Öll forritin sem nefnd eru hér að framan má ræsa með stikanum „--help“. Framkvæmdu eftirfarandi:

1. Finndu út hvernig þú getur skoðað ástand gagnagrunna (mysqlshow).
2. Finndu út hvernig þú tekur afrit af mysql grunninum (mysqldump).
3. Finndu út hvernig þú getur endurræst miðlarann (mysqladmin).

15. Æfing: Póst-, og götuskrá

Á vefsvæðinu www.posturinn.is má finna textaskrár með öllum póstnúmerum á Íslandi og öllum götum innan póstsvæðanna.

1. Finndu þessar skrár og niðurhalaðu.
2. Útbúðu töflur í MySQL sem geta geymt upplýsingarnar.
3. Notaðu mysqlimport til að afrita upplýsingar skránna inn í töflurnar.
4. Gerðu nú fyrirspurnir sem birta allar götur á landinu, í póstnúmeraröð þar sem nafn hvers póstsvæðis birtist með hverri götu.

5. Kafli. Málritun

Eftirfarandi eru útskýringar á hvernig MySQL meðhöndlar SQL rithátt. Ritháttarreglur ná til atriða s.s: Lesgildi (e. Literals) fyrir strengi og tölur, notendaskilgreind heiti (nefni) fyrir töflur og dálka, notendaskilgreind heiti fyrir kerfis- og notendabreytur, athugasemdir í kóta og frátekin heiti í MySQL og SQL.

Lesgildi

Lesgildi taka til strengja, talna, sextándukerfis talna, rökgilda (e. Boolean values) og NULL gildið. Með öðrum orðum, þau gildi sem slá má inn eða taka bókstaflega.

Algildisstafir

`%` Algildistákn[45], margir stafir. Merkir alla stafi endurtekna eins oft og þurfa þykir, sama merking og * í Windows skráakerfinu.

`_` Algildistákn, einn stafur. Merkir alla stafi, eitt tilfelli, sama merking og „?“ í Windows skráakerfinu.

`‘’, “”` Úrfellingamerki (‘) og tvíhögg (“) eru notuð sem strengja-afmarkarar (e. String delimiters). Þannig er ritháttúrinn: “`fyrri strengurinn`” og `‘hinn strengurinn’` notaður til að afmarka röð bókstafa eða rittákna sem streng. Sama er notað utan um stafatákn (e. Character). Ritháttúrinn: `‘a’` og “`a`” er því afmörkun á bókstafnum a og A. Í kóta kemur þetta fram þannig:

```
insert into tafla (nafndálkur) values(“Jói Jóason”);
```

Þess ber að gæta að ekki er rétt að blanda saman þessum táknum á sama strenginn, þannig væri: `‘strengurinn minn”` ólöglegt. Hins vegar má nota úrfellingamerki innan tvíhöggs markaðra strengja og öfugt!

`Hástafanæmi.` SQL er ekki hástafa-ónæmt, því merkja skipanirnar „`SELECT * FROM TAFLA`“ og „`select * from tafla,`“ hið sama og eru báðar löglegar.

`Línuending`: SQL krefst þess að allar skipanir endi á semíkommu (`;`). Þannig hlýtur setningin „`select * from tafla;`“ að vera því aðeins lögleg ef hún endar svo.

Strengir

Strengur er röð stafa innan úrfellingamerkja (‘’) eða innan tvíhöggs (“”), samanber:

```
‘þetta er strengur’
“þetta er strengur”
```

Stilla má miðlarann á ANSI_QUOTES ham en þá eru strengir eingöngu túlkaðir innan úrfellingamerkis en strengur innan tvíhöggs yrði þá skilinn sem heiti t.d. á dálk eða töflu.

Lausnarrunur (e. Escape characters)

Í strengjum má nota lausnarrunur (flótta stafi[46]) til að miðla sérstökum merkingum og eru þeir formerktir með öfugu skástriki (\). Eftirfarandi tafla listar upp þá flótta stafi sem MySQL þekkir:

[45] Wildcard eða arbitrary character. Algildisstafur er stafur sem stendur fyrir hvaða staf sem er eða stafastreng. Oft er ? eða # notað í staðinn fyrir einn staf en * í staðinn fyrir stafastreng. NN* getur t.d. staðið fyrir öll skráarnöfn sem hefjast á NN.

\0	Stafurinn ASCII 0 (NUL).
\'	Úrfellingamerkið sem stafur ‘’.
\"	Tvíhögg sem rittákn ‘"’ .
\b	Hopstafur (e. Backspace).
\n	Línuskiptingar stafur (e. Linefeed) innan strengs.
\r	Vendistafur (e. Carriage return).
\t	Dálkastafur (e. Tab).
\z	ASCII 26 (Control-Z). ASCII 26 merkir skráarendingu (END-OF-FILE) á Windows stýrikerfinu. Sem getur orðið til vandræða þegar skrár eru lesnar inn.
\\	Öfugt skástrik ‘\’.
\%	Prósenturittákn ‘%’.
_	Undirstrik ‘_’.

Allir flótta stafir eru hástafanæmir þannig að ‘\b’ er flótta stafur en `\B' er bara B. Flótta stafirnir ‘\%’ og ‘_’ eru notaðir í mynstra-samanburði á strengjum sem „%“ og „_“ en í flestum öðrum tilfellum sem „\%“ og „_“.

Gæsalappir í strengjum

Setja má úrfellingamerki og tvíhögg í strengi á eftirfarandi hátt:

- Úrfellingamerki í streng innan úrfellingamerkis:
  ```
  ‘texti með ‘’enskri gæsalöpp’’ utanum aukastreng’
  ’texti með \’enskri gæsalöpp\’ utanum aukastreng’
  ```
- Tvíhögg í streng innan tvíhöggs:
  ```
  ”texti með “”enskri gæsalöpp”” utanum aukastreng”
  ”texti með \”enskri gæsalöpp\” utanum aukastreng”
  ```
- Tvíhögg í streng innan úrfellingamerkis:
  ```
  ‘texti með “enskri gæsalöpp” utanum aukastreng’
  ```
- Úrfellingamerki í streng innan tvíhöggs:
  ```
  ”texti með ‘enskri gæsalöpp’ utanum aukastreng”
  ```

Við aðstæður þar sem setja þarf texta inn í BLOB dálk og sambærilega tvíundar-geymslu, þarf að túlka eftirfarandi stafi með flótta stöfum (lausnarrunu):

NUL Núll bæti (ASCII 0) túlkað sem \0

\ Öfugt skástrik (ASCII 92) túlkað sem \\.

' Úrfellingamerki (ASCII 39) túlkað sem \'.

" Tvíhögg (ASCII 34) túlkað sem \".

Þegar ytri, sérhönnuð forrit (eða vefsíður) eru notuð til þess að senda gildi inn í slíka dálka þarf að nota föll í viðkomandi forritunarmálum til að gæta að þessi gildi fari rétt inn. Í PHP forritunarmálinu má t.d. nota fallið `addslashes(texti)` til að setja þessi gildi sem lausnarrunu.

---

[46] Escape sequence: Lausnarruna er stafastrengur, notaður við kótaskipti. Fyrsti stafur í strengnum er oftast lausnarstafur. Þetta er rétt merking þess hugtaks sem ég hef nefni sums staðar „flótta-staf.“

Tölur

Heiltölur eru röð talna án brotastafs. Rauntölur nota punkt (.) sem kommu sem skilur á milli tugakerfis-hlutans og brotsins. Báðar tegundir geta tekið mínusmerki til að tákna neikvætt gildi þ.e. geta verið formerktar.

Heiltölur:

```
250
0
-127
```

Rauntölur:

```
250.6
25e+10
0.0
-127.34567
```

Sextándukerfis talnagildi

Rita má sextándukerfis tölur sem strengi en þau eru túlkuð sem heiltölur af 64 bita lengd, nota má fallið HEX() til að finna út hvernig tiltekinn strengur er ritaður í sextándukerfi:

```
select 0xA1736C656E7A6B61;  Útkoma: íslenzka
select hex('íslenzka');     Útkoma: A1736C656E7A6B61
select hex(12);             Útkoma: C
```

Gildi af röktegund - Boolean

Gefa má fastann TRUE sem ákvarðast sem 1 (satt) og fastann FALSE sem ákvarðast sem 0 (ósatt). Þessa fasta má rita bæði í lág- og hástöfum.

```
SELECT TRUE, true, FALSE, false;       Útkoma: 1, 1, 0, 0
```

Sérstaka gildið NULL

NULL gildið merkir „engin gögn“ og má rita bæði í há- og lágstöfum. NULL hefur allt aðra merkingu en 0 fyrir tölur eða “” (tómur strengur) fyrir strengja tegundir.

Þegar gögn eru hlaðin inn með LOAD DATA INFILE eða SELECT … INTO OUTFILE er NULL gildið táknað sem \N.

Mörgum byrjendum hefur fundist NULL gildið nokkuð ruglingslegt og túlka það gjarnan sem 0 eða tóman streng. NULL hins vegar hefur allt aðra merkingu þ.e. fjarvera gildis. Þannig eru eftirfarandi SQL aðgerðir gjörólíkar.

```
INSERT INTO my_table (simi) VALUES (NULL);
INSERT INTO my_table (simi) VALUES ('');
```

Báðar aðgerðirnar setja nýtt gildi inn í dálkinn „simi“ en sú fyrri setur NULL gildi sem gæti táknað „hefur símanúmer en við þekkjum það ekki,“sú síðari hins vegar setur tóman streng sem gæti táknað „hefur ekki síma.“

Til að meðhöndla NULL gildið eru notaðir virkjarnir IS NULL og NOT NULL auk fallsins IFNULL(). NULL gildið skilar aldrei sönnu (e. True) í samanburði og yrðing sem inniheldur NULL skilar alltaf NULL í SQL, þó eru til einstaka staðir í t.d. föllum MySQL sem geta skilað öðrum niðurstöðum. Allar eftirfarandi yrðingar skila NULL.

```
SELECT NULL, 1+NULL, CONCAT('Invisible',NULL);
```

Til að finna út dálka sem hafa NULL þýðir ekkert að nota samanburð af taginu „yrðing = NULL,“ því hún skilar áreiðanlega NULL. Nota verður IS NULL eða NOT NULL virkjana samanber eftirfarandi dæmi sýnir:

NULL dæmi

```
create table simi(
  simanumer varchar(7),
  nafn varchar(31)
);

insert into simi values
  ('','Herbert'),(NULL,'Jonas'),('1234567','Anna'),('2234567','Siggi');

select * from simi;
+-----------+---------+
| simanumer | nafn    |
+-----------+---------+
|           | Herbert |
| NULL      | Jonas   |
| 1234567   | Anna    |
| 2234567   | Siggi   |
+-----------+---------+
4 rows in set (0.01 sec)

select * from simi where simanumer is NOT NULL;
+-----------+---------+
| simanumer | nafn    |
+-----------+---------+
|           | Herbert |
| 1234567   | Anna    |
| 2234567   | Siggi   |
+-----------+---------+
3 rows in set (0.00 sec)

select * from simi where simanumer IS NULL;
+-----------+-------+
| simanumer | nafn  |
+-----------+-------+
| NULL      | Jonas |
+-----------+-------+
1 row in set (0.00 sec)
```

Heiti á nefnum

Heiti á gagnagrunnum, töflum, atriðaskrám (e. Index) dálkum og uppnefnum eru nefni[47]. Eftirfarandi eru reglur um þau en fyrst er tafla sem sýnir hámarkslengd þessara heita:

Nefni	Hámarkslengd í bætum/stöfum	Leyfðir stafir

[47] I**dentifier**: **Nefni** er lesstak sem er heiti á máleiningu. Heiti á breytum, fylkjum, færslum, merkjum og stefjum. Nefni hefst venjulega á bókstaf. Síðan geta komið bókstafir, tölustafir eða aðrir stafir.

Gagnagrunnur (Database)	64	Allir stafir sem leyfðir eru í heitum mappa/skráasafna nema „/“, „\“, eða „.“.
Tafla (Table)	64	Allir stafir sem leyfðir eru í skráaheitum nema nema „/“, „\“, eða „.“.
Dálkur (Column)	64	Allir stafir.
Atriðisskrá (Index)	64	Allir stafir.
Uppnefni (Alias)	255	Allir stafir.

Ekkert skilgreint heiti á nefnum má innihalda ASCII 0 eða bæti af gildinu 255. Þá er óleyfilegt að heiti endi á orðabili. Öll nefni í MySQL eru vistuð sem Unicode strengir á sniðinu UTF8, bæði hvað varðar töfluskrár (*.frm) og upplýsingar um aðgangsstýringar í „grant“ töflum mysql gagnagrunnsins.

Þar sem Unicode strengir eru fjölbæta-stafir (hver stafur samkvæmt skilgreiningu er tvö bæti þó UTF8 geti breytt út af því) þarf að gæta vel að nafnalengd, því hámarkslengd nefna er mæld í bætum. Þannig getur 32 stafa heiti í Unicode tekið 64 bæta minnispláss.

Best er að hafa öll nefni skilgreind með ensku stafamengi.

Sérgreinir nefna

MySQL leyfir að heiti séu samsett úr einum eða fleiri nefnum og eru þá nefnin aðgreind með punkti. Sérstakir hlutar þessara nefna gegna hlutverki sérgreina[48] (e. Qualifier) sem auðkenna nánar merkingu nafnsins (e. Semantics of name) og þar með endanlega merkingu síðasta hluta nafnsins.

Eftirfarandi tafla sýnir hvernig nafngiftir með þessu sniði eru ritaðar:

Dálkatilvísun	**Merking**
`heiti_dálks`	Dálkurinn heiti_dálks frá hvaða töflu svo sem vísað er til í fyrirspurn.
`heiti_töflu.heiti_dálks`	Dálkurinn heiti_dálks í töflunni heiti_töflu innan sjálfgefins (núgildandi) gagnagrunns.
`heiti_ggr.heiti_t.heiti_d`	Dálkurinn heiti_d í töflunni heiti_t sem er í gagnagrunninum heiti_ggr.

Yfirleitt dugar fyrsti ritháttur inn „heiti_dálks“ til að vísa á dálka nema þegar heiti þeirra verður tvírætt (e. Ambigous). Tvíræðni á sér stað þegar fyrirspurn leitar fanga í tveim töflum og báðar töflurnar hafa dálk með sama heiti: Í slíkum tilvikum þarf að grípa til annars ritháttarins. Þriðji og síðasti ritháttur inn á við þegar leitað er fanga í öðrum gagnagrunni á miðlaranum en MySQL leyfir að gagnagrunnar sjái töflur og dálka hvor hjá öðrum svo fremi að notandinn hafi réttindi til þess:

Í eftirfarandi dæmi er gerð fyrirspurn á tvær töflur. Önnur taflan er í gagnagrunninum „test“ og hin er í gagnagrunninum „test2.“ Fyrirspurnin er gerð í test2 grunninum og eru töflurnar eins hannaðar í báðum grunnunum.

```
select a.fornafn, a.fodurnafn, b.fornafn, b.fodurnafn
  from test.nafnaskra as a, nafnaskra as b;
```

[48] Qualifier. Sérgreinir er merki sem bætt er við gildi eða nefni til þess að tilgreina svæði eða undirgildi. Ef Y er svæði í færslu X er það tilgreint með X.Y þar sem Y er notað sem sérgreinir.

```
+---------+-----------+---------+-----------+
| fornafn | fodurnafn | fornafn | fodurnafn |
+---------+-----------+---------+-----------+
| Pall    | Palsson   | Jonas   | Jonasson  |
+---------+-----------+---------+-----------+
1 row in set (0.00 sec)
```

Dæmi um nefni

Skoðum notendur sem þekktir eru í töflunni „user“ í gagnagrunninum „mysql“.

```
select user from mysql.user;
+-------+
| user  |
+-------+
| %     |
| ODBC  |
| elias |
| %     |
| elias |
| root  |
+-------+
6 rows in set (0.03 sec)
```

Tilgreinum að við notum gagnagrunninn „test“ og búum þar til töfluna „user“. Lýðfyllum töfluna með fáeinum færslum og birtum þær síðan:

```
use test;
create table user(
  User varchar(31)
);

insert into User values
 ('Grettir'),('Gunnar'),('Ormur'),('Njáll'),('Kolumkilli');

select User from user;
+------------+
| User       |
+------------+
| Grettir    |
| Gunnar     |
| Ormur      |
| Njáll      |
| Kolumkilli |
+------------+
5 rows in set (0.00 sec)
```

Freistum þess nú að birta alla notendur í „mysql.user“ og í „test.user“ en fáum villuboð því nöfn þeirra eru óskýr (e. Ambiguous).

```
select user.User, mysql.user.User from test.user, mysql.user;
ERROR 1052 (23000): Column: 'user.User' in field list is ambiguous
```

Ef við hins vegar tilgreinum gagnagrunn, töfluheiti og dálkheiti sérstaklega er ekkert vandamál.

```
select test.user.User, mysql.user.User from test.user, mysql.user;
+------------+-------+
| User       | User  |
+------------+-------+
| Grettir    | root  |
| Gunnar     | root  |
... klippt
| Ormur      | elias |
| Njáll      | elias |
```

```
| Kolumkilli | elias |
+------------+-------+
30 rows in set (0.01 sec)
```

Líklega væri einfaldara að taka niðurstöður úr báðum töflum sem um einn dálk væri að ræða:

```
select test.user.User from test.user
union
select mysql.user.User from mysql.user;
+------------+
| User       |
+------------+
| Grettir    |
| Gunnar     |
| Ormur      |
| Njáll      |
| Kolumkilli |
| %          |
| ODBC       |
| elias      |
| root       |
+------------+
9 rows in set (0.04 sec)
```

Gættu að því að í dæminu hér að framan er verið að leika sér dálítið, m.a. til að sýna óbeint að sækja má upplýsingar úr töflum í aðskildum gagnagrunnum.

Hástafanæmi í nefnum

SQL málskipan krefst þess að nefni séu hástafa-ónæm. Þess vegna ættu allar þrjár eftirfarandi SQL aðgerðir að vera jafnvægar og eru það:

```
select user();
SELECT USER();
Select User();
```

Gagnagrunnar í MySQL eru geymdir sem skráasöfn (möppur) í skráakerfi þess stýrikerfis sem miðlarinn vinnur á hverju sinni, sömuleiðis eru töflur geymdar sem skrár í þessum skráasöfnum. Þess vegna, þó MySQL gæti þess að vera ónæmur á hástafa-, lágstafanæmi þá hefur stýrikerfið óneytanlega áhrif þar á.

Sá sem notar miðlarann þarf því að vera meðvitaður um skráakerfis reglur stýrikerfisins hverju sinni. Linux til að mynda er hástafa-næmt stýrikerfi og taflan „User“ er þar ekki sama tafla og „user“ í sama gagnagrunni (skráasafni). Windows er hinsvegar stafa-ónæmt og í sama gagnagrunni geta ekki verið tvær töflur með nafninu „user“ því „User“ væri sama heiti.

Eftirfarandi dæmi sýnir þetta vel:

Hér er tengst gagnagrunninum „test“ í MySQL á Windows vél, úr kvaðningu á Windows, og birtar töflurnar þar:

```
C:\>mysql test -h localhost -u root -p
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 5.0.0-alpha-nt-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

show tables;
+----------------+
| Tables_in_test |
+----------------+
| articles       |
... klippt
```

```
| user           |
+----------------+
15 rows in set (0.01 sec)
```

Við sjáum að ein taflan heitir „user“ og við biðjum um lýsingu á töflunni en ritum „User“. Þetta er löglegt á Windows.

```
describe User;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| User  | varchar(31) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
1 row in set (0.03 sec)
```

Næst er tengst gagnagrunninum „test“ í MySQL á Linux vél, úr sömu dos-kvaðningu og birtar töflurnar þar:

```
C:\>mysql test -h moltke -P 3306 -u elias -p
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 55 to server version: 5.0.0-alpha-standard-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

show tables;
+----------------+
| Tables_in_test |
+----------------+
| Apar           |
... klippt
| testannad      |
+----------------+
10 rows in set (0.00 sec)
```

Við biðjum um lýsingu á töflunni „Apar“ en misritum sem „apar“ sem er löglegt á Linux en taflan apar er ekki til þó Apar séu á staðnum:

```
describe apar;
ERROR 1146 (42S02): Table 'test.apar' doesn't exist
```

Umritum fyrirspurnina *rétt* og náum árangri.

```
describe Apar;
+--------+---------------------+------+-----+---------+-------+
| Field  | Type                | Null | Key | Default | Extra |
+--------+---------------------+------+-----+---------+-------+
| tegund | varchar(20)         | YES  |     | NULL    |       |
| kyn    | tinyint(3) unsigned |      |     | 0       |       |
| nafn   | varchar(25)         | YES  |     | NULL    |       |
+--------+---------------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

Eins og sést á dæminu hér á undan þá skiptir hástafa-næmi nokkru máli þó MySQL miðlaranum sé sama. Notandi sem meðhöndlar MySQL gagnagrunna í Linux umhverfi þarf t.d. að gæta að því að sé leitað upplýsinga um notendur í töflunni „mysql.user“ þarf hann að rita fyrirspurnina sem „select user from user“ þegar hann vísar í notenda dálkinn, fyrirspurnin „select user from User“ væri þó lögleg á Windows þó hún myndi ekki virka rétt á Linux.

Heiti á dálkum, atriðiskrám og uppnefnum er stafa-ónæmt í MySQL á öllum verkvöngum[49], framangreind hástafa-næmi á fyrst og fremst við heiti á gagnagrunnum og töflum.

[49] Platform: Verkvangur.

Notenda-breytur

Notenda breytur eru háðar virku tengingunni þegar þær eru skilgreindar. Um leið og tengingin er rofin hverfur breytan. Þær eru ritaðar með @ merki áföstu nefni sínu. Notendabreytur má rita með öllum almennum stöfum (best að nota enska stafrófið), punkti (.), úrfellingamerki (‘), undirstriki (_) og dollaramerki ($). Þær eru ekki hástafa-næmar frá og með MySQL útgáfu 5 en voru það í eldri útgáfum og skulu alltaf hefjast með á-merkinu „@“.

Skilgreina má notendabreytur með bæði „=“ og „:=“ við gildistökuna. Notendabreyta sem ekki hefur neitt gildi, hefur NULL gildið. Einnig má gefa notendabreytum ný gildi án þess að nota SET skipun en þá verður að nota gildistöku virkjann „:=“.

```
SET @var_name = expr [, @var_name = expr] ...
```

Hér er búin til notendabreytan „@var_name“ sem inniheldur gildi yrðingarinnar „expr“. Eins og sjá má er hægt að skilgreina margar breytur í einu sé rituð komma á milli. Nota má breytur af þessu tagi til að geyma ýmis gildi en þau eru ávallt skilin sem strengur.

Eftirfarandi dæmi sýnir hvar búnar eru til tvær breytur, @t1 og @t2 og þeim gefin gildi, síðan eru gildi þeirra lögð saman í SELECT aðgerð:

```
set @t1 = 5;
set @t2 = 6;
select @t1+@t2;
+---------+
| @t1+@t2 |
+---------+
|      11 |
+---------+
```

Næsta aðgerð sýnir hvar gildinu á @t2 er breytt áður en það er lagt við @t1.

```
select @t1+(@t2:=3);
+--------------+
| @t1+(@t2:=3) |
+--------------+
|            8 |
+--------------+
1 row in set (0.00 sec)
```

Eitt af notagildum þessara breyta er að geyma í þeim niðurstöður úr fyrirspurnum. Niðurstaðan sé þá eitt gildi úr einum dálki, samanber eftirfarandi dæmi:

Lýsum töflunni „sulta.“

```
describe sulta;
+--------+------------------+------+-----+---------+----------------+
| Field  | Type             | Null | Key | Default | Extra          |
+--------+------------------+------+-----+---------+----------------+
| num    | int(10) unsigned |      | PRI | NULL    | auto_increment |
| titill | varchar(255)     | YES  | MUL | NULL    |                |
| efni   | text             | YES  |     | NULL    |                |
+--------+------------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Skilgreinum breytuna „@nidurst“ beint úr fyrirspurn sem sameinar (með fallinu CONCAT_WS()) tvo dálka úr töflunni í einn:

```
set @nidurst = (select concat_ws(':',titill,efni)
from sulta where num = 3);
```

```
select @nidurst;
+---------------------------------+
| @nidurst                        |
+---------------------------------+
| Gagnagrunnar:mysql oracle access |
+---------------------------------+
1 row in set (0.00 sec)
```

Kerfis-breytur

Miðlarinn, forritið mysqld sér um að skilgreina fjöldann allan af víðværum (e. Global) breytum sem öll MySQL forritin hafa aðgang að, auk miðlarans sjálfs. Þegar miðlarinn fer í gang skilgreinir hann og frumstillir allar víðværar breytur, sumar þeirra fá innbyggð sjálfgildi (e. Default values) en aðrar fá gildi eftir því sem tilgreint er í skránni „my.cnf“ eða „my.ini“ eftir atvikum.

Notandi getur endursett gildi víðværrar breytu með skipuninni „SET GLOBAL breytu_heiti“ en þarf til þess aðgangsstýringuna SUPER. Einnig sér miðlarinn um allt utanumhald á staðværum breytum sem háðar eru tengitíð (e. Session), eða tengilotu, þess notanda sem tengdur er hverju sinni. Skoða má margar þessara breyta frá mysql biðlaranum og þaðan má einnig endursetja gildi þeirra, samanber:

```
set global max_connections = 150;
Query OK, 0 rows affected (0.00 sec)

show global variables like '%conn%';
+--------------------------+-------------------+
| Variable_name            | Value             |
+--------------------------+-------------------+
| character_set_connection | latin1            |
| collation_connection     | latin1_swedish_ci |
| connect_timeout          | 5                 |
| init_connect             |                   |
| max_connections          | 150               |
| max_connect_errors       | 10                |
| max_user_connections     | 0                 |
+--------------------------+-------------------+
7 rows in set (0.00 sec)
```

Engin sérstök aðgangsréttindi þarf til að breyta gildum á staðværum breytum en þau gildi eiga aðeins þann líftíma sem tengitíð biðlarans varir. Auk þess ef notandi skilgreinir staðvær (e. Local) gildi einhverrar breytu hefur það engin áhrif á gildi sömu breytu í tengingu annars notanda sem gæti verið tengdur á sama tíma.

Athugasemdir

Nota má þrenns konar athugasemdir í SQL kóta, samanber:

- Frá „#“ og út línuna
- Frá „-- “ og út línuna. Um er að ræða „mínus-mínus-orðabil“ þar sem orðabilið er nauðsynlegur hluti athugasemdarinnar eða þá flótta stafurinn „\n“ (newline).
- Frá „/*“ og alveg til „*/“ án tillits til línunnar, þ.a.l. má athugasemdin spanna margar línur.

Eftirfarandi kóti smíðar töfluna „fjolsk“ og notar allar þrjár tegundir athugasemda:

```
create table fjolsk( # Þessi tafla geymir nöfn fjölskyldumeðlima
    -> nafn varchar(31) not null, -- Hámarkslengd nafns er 31 stafur
```

```
    -> f_ar date not null, /* við verðum að skilgreina
    -> og geyma fæðingarár en við megum
    -> sleppa því að geyma giftingarár */
    -> maki varchar(31) null
    -> );
Query OK, 0 rows affected (0.08 sec)
```

Frátekin heiti í MySQL

MySQL leyfir að ýmis notendaskilgreind heiti (nefni) s.s. nöfn á gagnagrunnum, töflum og dálkum séu nefnd eftir ýmsum innbyggðum nefnum svosem TIMESTAMP eða GROUP. Þetta getur þó verið gríðarlega varasamt og ber að forðast. Til dæmis er vel hægt að skilgreina dálk með heitinu ABS þó ABS() sé eitt af innbyggðum föllum og af þessum sökum er ekki leyft þegar föll eru notuð að hafa bil á milli heiti fallsins og svigans þar á eftir.

Eftirfarandi dæmi sýnir tvær töflur, önnur með dálkinum timestamp af tegundinni TIMESTAMP og hin síðari með tveim dálkum er heita abs og DATE, hvorttveggja nafngiftir sem mætti forðast, ekki tókst að búa til dálkinn INT eða INTEGER.

```
create table klikk(
  timestamp timestamp
);

insert into klikk values(now());

select * from klikk;
+---------------------+
| timestamp           |
+---------------------+
| 2004-07-27 17:28:08 |
+---------------------+
1 row in set (0.00 sec)

create table mjogklikk(
  abs char(10),
  date date,
  heiltala int
);

insert into mjogklikk values('sulta',now(),1);

select * from mjogklikk;
+-------+------------+----------+
| abs   | date       | heiltala |
+-------+------------+----------+
| sulta | 2004-07-27 |        1 |
+-------+------------+----------+
1 row in set (0.00 sec)
```

Eftirfarandi tafla birtir öll heiti sem eru sérstaklega (e. Explicitly) frátekin í MySQL og ekki er hægt að nota sem notendaskilgreind nefni. Forðast ætti á öllum stundum að nota heiti sem forritunarmál eða gagnagrunnsforrit nota til að skilgreina tegundir, lykilorð eða föll.

ADD	ALL	ALTER	ANALYZE
AND	AS	ASC	ASENSITIVE
AUTO_INCREMENT	BDB	BEFORE	BERKELEYDB
BETWEEN	BIGINT	BINARY	BLOB

BOTH	BY	CALL	CASCADE
CASE	CHANGE	CHAR	CHARACTER
CHECK	COLLATE	COLUMN	COLUMNS
CONDITION	CONNECTION	CONSTRAINT	CONTINUE
CREATE	CROSS	CURRENT_DATE	CURRENT_TIME
CURRENT_TIMESTAMP	CURSOR	DATABASE	DATABASES
DAY_HOUR	DAY_MICROSECOND	DAY_MINUTE	DAY_SECOND
DEC	DECIMAL	DECLARE	DEFAULT
DELAYED	DELETE	DESC	DESCRIBE
DETERMINISTIC	DISTINCT	DISTINCTROW	
DIV	DOUBLE	DROP	ELSE
ELSEIF	ENCLOSED	ESCAPED	EXISTS
EXIT	EXPLAIN	FALSE	FETCH
FIELDS	FLOAT	FOR	FORCE
FOREIGN	FOUND	FRAC_SECOND	FROM
FULLTEXT	GRANT	GROUP	HAVING
HIGH_PRIORITY	HOUR_MICROSECOND	HOUR_MINUTE	HOUR_SECOND
IF	IGNORE	IN	INDEX
INFILE	INNER	INNODB	INOUT
INSENSITIVE	INSERT	INT	INTEGER
INTERVAL	INTO	IO_THREAD	IS
ITERATE	JOIN	KEY	KEYS
KILL	LEADING	LEAVE	LEFT
LIKE	LIMIT	LINES	LOAD
LOCALTIME	LOCALTIMESTAMP	LOCK	LONG
LONGBLOB	LONGTEXT	LOOP	LOW_PRIORITY
MASTER_SERVER_ID	MATCH	MEDIUMBLOB	MEDIUMINT
MEDIUMTEXT	MIDDLEINT	MINUTE_MICROSECOND	MINUTE_SECOND
MOD	NATURAL	NOT	NO_WRITE_TO_BINLOG
NULL	NUMERIC	ON	OPTIMIZE
OPTION	OPTIONALLY	OR	ORDER
OUT	OUTER	OUTFILE	PRECISION
PRIMARY	PRIVILEGES	PROCEDURE	PURGE

READ	REAL	REFERENCES	REGEXP
RENAME	REPEAT	REPLACE	REQUIRE
RESTRICT	RETURN	REVOKE	RIGHT
RLIKE	SECOND_MICROSECOND	SELECT	SENSITIVE
SEPARATOR	SET	SHOW	SMALLINT
SOME	SONAME	SPATIAL	SPECIFIC
SQL	SQLEXCEPTION	SQLSTATE	SQLWARNING
SQL_BIG_RESULT	SQL_CALC_FOUND_ROWS	SQL_SMALL_RESULT	SQL_TSI_DAY
SQL_TSI_FRAC_SECOND	SQL_TSI_HOUR	SQL_TSI_MINUTE	SQL_TSI_MONTH
SQL_TSI_QUARTER	SQL_TSI_SECOND	SQL_TSI_WEEK	SQL_TSI_YEAR
SSL	STARTING	STRAIGHT_JOIN	STRIPED
TABLE	TABLES	TERMINATED	THEN
TIMESTAMPADD	TIMESTAMPDIFF	TINYBLOB	TINYINT
TINYTEXT	TO	TRAILING	TRUE
UNDO	UNION	UNIQUE	UNLOCK
UNSIGNED	UPDATE	USAGE	USE
USER_RESOURCES	USING	UTC_DATE	UTC_TIME
UTC_TIMESTAMP	VALUES	VARBINARY	VARCHAR
VARCHARACTER	VARYING	WHEN	WHERE
WHILE	WITH	WRITE	XOR
YEAR_MONTH	ZEROFILL		

Eftirfarandi lykilorð eru leyfð sem notendaskilgreind heiti því mörg þeirra eru eðlileg orð í ensku máli, en sem fyrr segir ætti að forðast slík heiti:

ACTION, BIT, DATE, ENUM, NO, TEXT, TIME, TIMESTAMP

6. Kafli. Dálkategundir

MySQL býður uppá margar dálkategundir sem skiptast í eftirfarandi þrjá flokka: tölur, dagssetningar og tími, og strengi. Hér verða allar þessar tegundir taldar upp og stutt skýring gefin á þeim. Þeir sem þarfnast ýtarlegri upplýsinga ættu að kynna sér á handbókina sem fylgir miðlaranum.

Fáeinar útskýringar á dálkategundunum nota eftirfarandi ritvenju:

M Gefur til kynna hámarks birtingarlengd tölunnar. Hámarkslengd af þessu tagi er þó 255.

D Á við hlaupakommutölur[50] (e. floating-point) og fast-kommutög[51] (e. fixed-point types) og gefur til kynna sætafjölda aftan við kommu (aftan við brotamerki)[52]. Hámarksgildi er 30 en ætti ekki að vera hærra en M-2 (hámarks-birtingarlengd mínus 2).

[] Hornklofar gefa til kynna tag-hluta sem er val en ekki nauðsynlegur.

Vísindamálskipan. Í vísindum eru rauntölur, sérstaklega þegar þær hækka í nærri ólæsileg gildi, oft ritaðar á vísindamálskipan sem virkar þannig: 1.000.000 verður 10E+6. Þessi ritun útleggst þannig, talan 10 í veldinu sex. E er skammritun enska orðsins „Exponent“ sem merkir veldi.

Talnategundir

Eftirfarandi er yfirlit talna dálka í MySQL.

Ef skilgreind er ZEROFILL fyrir talnadálk setur MySQL sjálfkrafa inn skilgreininguna UNSIGNED fyrir dálkinn. Þegar frádráttur er viðhafður á heiltöludálka sem hafa UNSIGNED er niðurstaðan sjálfkrafa UNISIGNED.

Tegund	Skýring
`TINYINT[(M)] [UNSIGNED] [ZEROFILL]`	Mjög lítil heiltala[53]. Formerkt[54] svið á gildissviðinu 128 til 127. Óformerkt gildissvið er 0 til 255. Minnispláss er 1 Bæti.

`BIT`
`BOOL`
`BOOLEAN`
	Þessi eru samheiti fyrir TINYINT.

[50] Floating-point type: Hlaupakommutag er rauntölutag þar sem hvert tilvik er sett fram í hlaupakommukerfi.

[51] Fixed-point type, implied decimal type: Fastakommutag er rauntölutag þar sem hvert tilvik er sett fram í fastakommukerfi.

[52] Hlaupakommukerfi: Talnaritunarkerfi þar sem sérhver rauntala er sett fram með tveimur tölutáknum. þ.e. tölukjarna og veldisvísi, þannig að rauntalan sé margfeldi tölukjarnans og hlaupakommustofnsins í því veldi sem veldisvísirinn segir til um. Í hlaupakommukerfi má setja sömu tölu fram á marga vegu með því að færa brotskilin og breyta veldisvísinum til samræmis. Enska: floating-point representation system.

[53] Integer, integer number: Heil tala, heiltala. Ein talnanna núll, einn, mínus einn, tveir, mínus tveir, ...

[54] Sign character: Formerkisstafur er stafur sem er í stöðu formerkis í tölutákni og segir til um hvort talan sem tölutáknið stendur fyrir er jákvæð eða neikvæð. Formerkin + og - eru sett fram með formerkisbitunum 0 eða 1 í tilteknu sæti í tölutákninu þegar tölur eru geymdar í tölvu.

`SMALLINT[(M)] [UNSIGNED] [ZEROFILL]`	Lítil heiltala. Formerkt gildissvið er -32768 til 32767. Óformerkt gildissvið er 0 til 65535. Minnispláss eru 2 Bæti.
`MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]`	Miðlungsstór heiltala. Formerkt gildissvið er á bilinu -8388608 til 8388607. Óformerkt gildissvið er á bilinu 0 til 16777215. Minnispláss eru 3 Bæti.
`INT[(M)] [UNSIGNED] [ZEROFILL]`	Venjuleg heiltölustærð. Formerkt gildissvið er á bilinu -2147483648 til 2147483647. Óformerkta gildissviðið er á bilinu 0 til 4294967295. Minnispláss eru 4 Bæti.
`INTEGER[(M)] [UNSIGNED] [ZEROFILL]`	Þetta er samheiti fyrir INT.
`BIGINT[(M)] [UNSIGNED] [ZEROFILL]`	Stór heiltala. Formerkt gildissvið er -9223372036854775808 til 9223372036854775807. Óformerkta gildissviðið er 0 til 18446744073709551615. Minnispláss eru 8 Bæti. Allur útreikningur fer fram með formerktum BIGINT og DOUBLE gildum og því ætti að forðast að nota óformerkt BIGINT gildi stærri en 9223372036854775807 (63 bitar) nema með bita-föllum (Bit Functions). annars má búast við því að sumar niðurstöður verði rangar vegna skekkju í námundun þegar BIGINT er breytt í DOUBLE. Alltaf er hægt að geyma rétt heiltölugildi í BIGINT dálki með strengjagildi en þá er engum útreikningum beitt. MAX og MIN föllin geta meðhöndlað þessar tölur og hægt er að reikna með plús, mínus og margföldun sé niðurstaðan heiltala en sé hún hærri en 9223372036854775807 má búast við skekkjum.
`FLOAT(p) [UNSIGNED] [ZEROFILL]`	Hlaupakommu tegund þar sem p stendur fyrir nákvæmni í aukastöfum. Aukastafir geta verið frá 0 til 24 hjá einnar-nákvæmnis hlaupatölu og frá 25 til 53 fyrir tví-nákvæmnis hlaupatölu. Þessar tegundir eru líkar FLOAT og DOUBLE tegundunum, nema birtingastærð (M) og fjöldi aukastafa (D) er óskilgreind. Einhverjir útreikningar geta gefið óvæntar niðurstöður því útreikningar fara fram með tví-nákvæmni (e. Double precision). Einnar-nákvæmnis tala tekur 4 Bæti en tví-nákvæmnis tala tekur 8 Bæti.
`FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]`	Lítil (einnar-nákvæmnis) hlaupakommu tala. Leyfileg gildi eru á sviðinu -3.402823466E+38 til -1.175494351E-38. Ef UNSIGNED er tilgreint eru neikvæð gildi bönnuð. M er birtingar breidd gildanna og D er fjöldi aukastafa. Minnispláss 4 Bæti.

`DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]`	Venjuleg (tví-nákvæmnis) hlaupakommu tala. Leyfð gildi frá -1.7976931348623157E+308 til -2.2250738585072014E-308, 0, og 1.7976931348623157E+308 til 1.7976931348623157E+308. Ef UNSIGNED er tilgreint eru neikvæð gildi bönnuð. M er birtingar breidd gildanna og D er fjöldi aukastafa. Minnispláss er 8 Bæti.
`DOUBLE PRECISION[(M,D)] [UNSIGNED] [ZEROFILL]` `REAL[(M,D)] [UNSIGNED] [ZEROFILL]`	Þessi eru samheiti fyrir DOUBLE tegundina. Frábrigði (e. Exception) frá þessari reglu er ef SQL hamur miðlarans er stilltur á REAL_AS_FLOAT þá virkar REAL sem samheiti fyrir FLOAT.
`DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]`	Ópakkað fastakommutags[55] talnagildi sem hagar sér líkt og CHAR dálkur. Ópakkað merkir að talan er geymd sem strengur þar sem einn bókstafur er notaður fyrir hvern tölustaf. M er notað til að tákna fjölda tölustafa og D táknar fjölda kommutalna (brotastafa). Tugakerfis komman (punktur í SQL) er ekki talinn með í M, né formerkið (‘-‘) sé um neikvæða tölu að ræða. Hámarks gildi er hið sama og í DOUBLE en hið eiginlega gildissvið tagsins er það sem er skorðað (e. Constrained) af M og D. Ef UNSIGNED er skilgreint verða neikvæð gildi óleyfileg. Ef D er sleppt er það sjálfgefið (e. Default) sem 0 og sé M sleppt er sjálfgefið 10.

`DEC[(M[,D])] [UNSIGNED] [ZEROFILL]`
`NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL]`
`FIXED[(M[,D])] [UNSIGNED] [ZEROFILL]`
	Þessi eru samheiti fyrir DECIMAL. Lykilorðið FIXED er samheiti sem bætt var við fyrir samræmingu við aðra miðlara.

Birtingabreidd

MySQL leyfir að skilgreind sé birtingabreidd heiltalna, samanber INT(5) tilgreinir að birtingabreidd tölunnar sé aðeins fimm tölustafir. Þetta þrengir þó ekki gildissvið dálksins heldur tilgreinir hversu marga stafi skuli nota til að birta töluna sem dálkurinn geymir. Sé lykilorðið ZEROFILL notað er tölugildið fóðrað (e. Padded) með núllum vinstramegin við töluna sé hún lægri en birtingabreiddin. Slík dálkaskilgreining getur verið vafasöm í notkun með ýmsum JOIN tilbrigðum sem búa til skammtöflur (e. Temporary tables).

Dæmi um notkun:

```
create table tolur(
  tala1 int(4),
  tala2 int(4) zerofill
 );

insert into tolur
  values(1,20),(2,400),(10,2000),(100,5050),(250,12345);
```

[55] F**ixed-point type**, implied decimal type: F**astakommutag** er rauntölutag þar sem hvert tilvik er sett fram í fastakommukerfi.

```
select * from tolur;
+-------+-------+
| tala1 | tala2 |
+-------+-------+
|     1 |  0020 |
|     2 |  0400 |
|    10 |  2000 |
|   100 |  5050 |
|   250 | 12345 |
+-------+-------+
5 rows in set (0.00 sec)
```

Formerki

Allar heiltölutegundir má skilgreina með (óstöðluðu) UNSIGNED breytunni við stofnun. Þetta er gert ef óskað er að breytan hafi hærra efra-gildi (jákvæð) og geti ekki geymt neikvæð. Fastakommutölur (e. Fixed-point numbers) geta einnig verið UNSIGNED en efri-gildi þeirra eru þó óbreytt. Ef tilgreint er ZEROFILL fyrir talnadálk setur MySQL sjálfkrafa breytuna UNSIGNED á dálkinn.

Ef sett er hærra gildi inn í talnadálk en gildissvið hans leyfir er gildið rúnnað niður í hæsta gildið. Ef skilgreindur er dálkur af tegundinni MEDIUMINT t.d. sem tekur gildissviðið -127 til 128 og sett inn gildið 200 rúnnast það niður í 128.

Hér er bætt við dálkunum „heil_jakv“ sem UNSIGNED af tegund INT og „raun_jakv“ af tegundinni FLOAT sem UNSIGNED í töfluna „tolur.“ Því næst er bætt við nokkrum gildum í fyrstu fimm línur töflunnar og sett inn hærra gildi í hvert sinn. Þegar fyrirspurn er gerð á töfluna kemur í ljós að gildin sem sett voru í hana tóku að rúnnast á heiltöludálkinn en flotkommudálkurinn skráði gildi sín með vísindamálskipan.

```
alter table tolur
  add column heil_jakv int unsigned,
  add column raun_jakv float unsigned;

describe tolur;
+-----------+--------------------------+------+-----+---------+-------+
| Field     | Type                     | Null | Key | Default | Extra |
+-----------+--------------------------+------+-----+---------+-------+
| tala1     | int(4)                   | YES  |     | NULL    |       |
| tala2     | int(4) unsigned zerofill | YES  |     | NULL    |       |
| heil_jakv | int(10) unsigned         | YES  |     | NULL    |       |
| raun_jakv | float unsigned           | YES  |     | NULL    |       |
+-----------+--------------------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

update tolur set heil_jakv = 5000, raun_jakv = 0.3 where tala1 =1;
update tolur set heil_jakv = 60000, raun_jakv = 5001.3 where tala1 = 2;
update tolur set heil_jakv = 6000500000, raun_jakv = 500100.53 where tala1 = 10;
update tolur set heil_jakv = 6000500000100, raun_jakv = 500100100.53 where tala1 = 100;
update tolur set heil_jakv = 6000500000100500, raun_jakv = 500100100500.536 where tala1 = 250;

select * from tolur;
+-------+-------+------------+------------+
| tala1 | tala2 | heil_jakv  | raun_jakv  |
+-------+-------+------------+------------+
|     1 |  0020 |       5000 |        0.3 |
|     2 |  0400 |      60000 |     5001.3 |
|    10 |  2000 | 4294967295 |     500101 |
|   100 |  5050 | 4294967295 | 5.001e+008 |
|   250 | 12345 | 4294967295 | 5.001e+011 |
```

```
+-------+-------+------------+------------+
5 rows in set (0.00 sec)
```

DECIMAL og NUMERIC tegundirnar eru meðhöndlaðar sem sama tagið í MySQL og eru helst notaðar þar sem geyma þarf mjög há gildi með mikilli nákvæmni. Þegar einhver þessara tegunda er skilgreind má (og ætti að) skilgreina nákvæmni (e. Precision) og kvarða (e. Scale), samanber:

```
laun decimal(10,1)
```

Hér er tilgreint að talan geymi 10 sæti með 1 tölu aftan við kommu. þannig mun dálkurinn laun geta geymt 10 stafa tölu framan við kommuna og 1 aftan við. Ef sett er inn 0 fyrir kvarða er ekki tekið frá neitt sæti fyrir brot. Tölurnar eru geymdar sem strengir og ef sett er inn neikvæð tala er tekið sérstakt sæti í formerkið. Dæmi:

```
create table laun(
  upphaed decimal(5,1)
);

insert into laun values(1234.1);
insert into laun values(21234.11);
insert into laun values(-234.16);
insert into laun values(-2234.16);
insert into laun values(-32234.16);

show warnings;
+---------+------+-----------------------------------------------------------+
| Level   | Code | Message                                                   |
+---------+------+-----------------------------------------------------------+
| Warning | 1263 | Data truncated, out of range for column 'upphaed' at row 1 |
+---------+------+-----------------------------------------------------------+
1 row in set (0.00 sec)

insert into laun values(123456.6);

select * from laun;
+---------+
| upphaed |
+---------+
|  1234.1 |
| 21234.1 |
|  -234.2 |
| -2234.2 |
| -9999.9 |
| 99999.9 |
+---------+
6 rows in set (0.00 sec)
```

SQL staðallinn krefst þess að dálkurinn „upphaed“ (samkvæmt framangreindri skilgreiningu) geti geymt hvaða gildi sem er með fimm tölum og einu broti. Í þessu tilfelli ætti dálkurinn að geyma að hámarki 9999.9 og að lágmarki -9999.9 þar sem fimmta töluplássið sé notað undir formerkið (+/-). MySQL leyfir hins vegar að jákvæða formerkið geti geymt tölustaf og því er gildissviðið hér að hámarki 99999,9 og lágmarki -9999,9.

Eins og sést á framangreindu dæmi þá námundar MySQL talnagildi sem eru hærri en dálkurinn leyfir. Þó gildissvið DECIMAL og NUMERIC dálkanna sé eiginlega hið sama og hjá DOUBLE þá getur MySQL geymt ívið stærri tölur þ.s. þær eru geymdar sem strengir.

Rauntölur

Rauntölur í MySQL eru tvennskonar, FLOAT sem tekur 4 Bæti og DOUBLE sem tekur 8 Bæti. DECIMAL og NUMERIC tegundirnar eru skyldar og taka jafn mikið pláss eftir aðstæðum. Ef reynt er að

geyma hærra gildi í dálkum af þessum tegundum er gildið klippt niður í hæstu mögulegu stærð dálksins. Hægt er að stilla rauntölur þannig að tilgreina má fjölda tölustafa framan við kommu og fjölda brotatalna aftan við kommu.

Tíðategundir

Yfirlit tíðategunda sem geta geymt dagsetningar, tímasetningar og bæði, er eftirfarandi:

Tegund	Skýring
DATE	Dagsetning á gildissviðinu frá '1000-01-01' til '9999-12-31'. MySQL birtir dagsetningar á í röðinni „ár-mánuður-dagur" á sniðinu: 'áááá-mm-dd' en leyfir að gildi séu gefin ýmist sem tölur eða strengir.
DATETIME	Samsetning dag- og tímasetningar á gildissviðinu frá '1000-01-01 00:00:00' til '9999-12-31 23:59:59'. MySQL birtir tíðasetningar í röðinni „ár-mánuður-dagur klukkustund-mínúta-sekúnda" á sniðinu 'áááá-mm-dd kk:mm:ss' en leyfir að gildi séu gefin upp sem ýmist tölur eða strengir.
TIMESTAMP[(M)]	Tímastimpill ættaður úr UNIX kerfi á gildissviðinu frá '1970-01-01 00:00:00' til hluta ársins 2037. Þessi tegund er hentug til að skrásetja sjálfkrafa dagsetningu þegar línum er bætt við töfluna með INSERT eða þær dagréttar með UPDATE. Fremsti TIMESTAMP dálkur í töflu er settur á yngstu dags- og tímasetningu miðað við síðustu aðgerð á töfluna ef ekkert gildi er sent inn í dálkinn. Einnig má setja hvaða TIMESTAMP dálk sem er á núgilda dags- og tímasetningu með því að gefa honum gildið NULL. TIMESTAMP gildi er skilað út sem strengur á sniðinu 'áááá-mm-dd kk:mm:ss'. Ef þess er óskað að gildinu sé skilað sem tölu skyldi bæta +0 við gildi dálksins. Senda má inn ný gildi bæði sem tölur og strengi. Minnispláss tegundarinnar er 4 Bæti. Í eldri útgáfum var M stikinn notaður til að hafa áhrif á hvernig gildið er birt en ekki á minnispláss (eiginlega lengd gildisins). Sem dæmi má tilgreina 8 til að tilgreina að einungis séu birtir fyrstu átta stafir gildisins. Sé M sett sem 8 eða 14 er gildið geymt sem tala.
TIME	Tímasetning á gildissviðinu frá '-838:59:59' til '838:59:59' sem MySQL birtir í röðinni „klukkustund-mínúta-sekúnda" á sniðinu 'KK:MM:SS'. Setja má inn gildin bæði sem tölur og strengi.
YEAR[(2|4)]	Ártal ýmist í tveim eða fjórum tölustöfum, sjálfgefið eru fjórir stafir. Leyfileg gildi eru á gildissviðinu frá 1901 til 2155 auk 0000. Í tveggja tölustafa formi má setja leyfileg gildi frá 70 til 69, sem táknar ártölin 1970 – 2069. MySQL birtir YEAR gildi á fjögurra talna sniði og setja má inn gildi bæði sem tölur og strengi.

Nánar um tíðir

Allar tíðategundir leyfa auk gildissviðs síns núll gildi sem notað er fyrir gildi sem MySQL skilur ekki eða á einhvern hátt eru óleyfileg fyrir dálkinn. Þannig má til dæmis skrá fæðingardag manns, í töflu sem krefst þess að fæðingardagur sé skráður, sem ‘1971-00-00‘ ef fæðingarár er þekkt en ekki mánuður og dagur. Þess ber að geta að séu núll gildi skrásett inn í tíðategundir geta tíðaföll s.s. DATE_SUB() og DATE_ADD gefið skakkar niðurstöður.

MySQL leyfir einnig að settar séu ógildar dagsetningar svosem ‘2004-11-31‘ sem er ekki til í okkar vestræna dagatali. MySQL athugar dagsetningar sem sendar eru inn þannig að fyrst er athugað hvort mánuðir séu á bilinu 1 til 12 og dagsetningar á bilinu 1 til 31. Þetta neyðir forritara sem senda inn þessi gildi til að sannreyna hvort þau séu lögleg eða ekki.

Eftirfarandi viðmiðanir þarf að hafa í huga þegar tíðir eru skráðar:

- MySQL birtir tíðir á sérstöku sniði en reynir að ráða í þau gildi sem send eru inn eftir atvikum. Þannig getur MySQL lesið úr tölunni 20041130 og strengnum 2004-11-30 sem sömu dagsetningu.
- Sé ártal gefið upp sem tveggja stafa tala er hún óskýr en MySQL les úr tveggja stafa gildunum 00 til 69 sem 2000 til 2069 og úr 70 til 99 sem 1970 til 1999.
- Þó MySQL reyni að ráða í innsend gildi og lesa rétt úr þeim skulu þau alltaf vera í röðinni ár-mánuður-dagur. Þetta er mikilvæg staðreynd vegna landa sem eins og t.d. Bandaríkin og fleiri sem nota aðrar samsetningar.
- Sé dags- eða tímasetning notuð í samhengi við tölur þá eru þær lesnar sem tölur, og öfugt.
- Sé dagsetning eða tímasetning send inn utan gildissviðs þá setur þjónninn núll gildi viðkomandi dálks í staðinn. Þetta á þó ekki við um TIME tegundina sem klippir gildið í hæsta leyfilega gildi sitt. Eftirfarandi tafla sýnir núll gildin eins og þau eru skilgreint fyrir hverja tegund:

Dálkategund	**„núll“ gildi**
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIMESTAMP	00000000000000
TIME	'00:00:00'
YEAR	0000

- Núll gildin eru sérstök í MySQL en setja má þau og sækja sérstaklega. Einnig má nota 0 og ‘0‘ á allar tegundirnar, sem er þægilegra.
- Núll daga- eða tímagildi sem notuð eru í gegnum ODBC tengingu (MyODBC) er sjálfkrafa umbreytt í NULL því ODBC leyfir ekki núll gildið.

Tegundirnar DATETIME, DATE, og TIMESTAMP

Tegundirnar DATETIME, DATE og TIMESTAMP eru skyldar og margt í meðhöndlun þeirra svipar til hvorrar annarrar. Setja má gildi í DATETIME, DATE, og TIMESTAMP á eftirfarandi vegu:

- Sem strengur á sniðinu 'YYYY-MM-DD HH:MM:SS' eða 'YY-MM-DD HH:MM:SS'. Afslappaðri málskipan er leyfð þannig að hvaða punkt-stafi sem er má nota sem afmarkara innan strengs, samanber: '98-12-31 11:30:45', '98.12.31 11+30+45', '98/12/31 11*30*45', og '98@12@31 11^30^45' eru allar samstæðar..

- Strengur án afmarkara á sniðinu 'YYYYMMDDHHMMSS' eða 'YYMMDDHHMMSS' svo fremi að dagsetningin sé ekki fáránleg.
- Tíðirnar '19970523091528' og '970523091528' eru túlkaðar sem '1997-05-23 09:15:28',
- Tíðin '971122129015' er ómerk því mínútan er fáránleg og er því túlkuð sem '0000-00-00 00:00:00'
- Tala á sniðinu YYYYMMDDHHMMSS eða YYMMDDHHMMSS svo fremi að talan sé ekki fáránleg. Tíðirnar 19830905132800 og 830905132800 eru túlkaðar sem '1983-09-05 13:28:00'.
- Sem tala á sniðinu YYYYMMDDHHMMSS eða YYMMDDHHMMSS, svo fremi að talan sé ekki fáránleg.
- Sem niðurstaða úr falli sem skilar gildi sem DATETIME, DATE eða TIMESTAMP getur tekið við. Dæmi eru föllin NOW() og CURRENT_DATE.

Sem fyrr segir, ef gildið er óskiljanlegt sem tíð, þá er sérstakt núll gildi geymt í dálkinum.

Þess er ekki krafist að setja inn tveggja stafa gildi fyrir stök sæti s.s. mánuði eða mínútur þegar gildið er sent inn sem strengur með afmarkara. Þannig er ‘2004-9-25‘ túlkað sem sama gildið og ‘2004-09-25‘, einnig er gildið ‘2004-10-25 01:2:0‘ túlkað sem sama gildi og ‘2004-10-25 01:02:00‘

Sé tíð send inn sem tala skiptir fjöldi tölustafa máli og mælt er með 6, 8, 12 eða 14 tölustöfum. Sé fjöldi tölustafa t.d. 8 eða 14 er sjálfkrafa reiknað með að fyrstu fjórir stafirnir séu ártal en ef tölustafafjöldinn er 6 eða 12 er gert ráð fyrir að fyrstu tveir stafirnir séu ártal. Aðgát skal höfð þegar tala er notuð því niðurstöðurnar geta auðveldlega gefið núll gildi eða aðra ranga niðurstöður.

Sé strengur án afmarkara notaður til að senda inn tíðir er fjöldi stafa notaður til að finna út hvaða stafir séu árið, fyrstu tveir eða fyrstu fjórir. Sé strengurinn 8 eða 14 stafir að lengd er t.d. gert ráð fyrir að fyrstu fjórar tölurnar séu ártalið annars fyrstu 2. Strengurinn er alltaf lesinn frá vinstri til hægri. Aldrei ætti að nota streng sem er styttri en sex stafir ef forðast á núll gildi, strengurinn ‘0404‘ táknar ekki „Apríl 2004“ eins og ætla mætti því daginn vantar, og geymt er núll í dálknum, hins vegar er ‘040400‘ löglegt sem „0 Apríl 2004“ sem gefur til kynna að dagurinn sé skráður þó hann sé óþekktur.

Eftirfarandi viðmið skyldi viðhafa þegar gildi úr mismunandi tíðaföllum eru afrituð á milli þeirra:

- Sé DATE gildi sett á DATETIME eða TIMESTAMP dálka er tímahluti þeirra settur á ’00:00:00’ því DATE hefur ekkert tímagildi.
- Sé DATETIME eða TIMESTAMP gildi afritað yfir á DATE er tíminn klipptur af því DATE getur ekki geymt hann.
- Muna þarf að TIMESTAMP dálkurinn hefur gildissvið frá 1970 til 2037 svo tíð sem afrituð er á TIMESTAMP er settur á núll sé gildið utan gildissviðs.

Gæta þarf vel að eftirfarandi þegar tíðir eru skráðar:

- Þó MySQL geti tekið við tíðum á afslappaðan máta getur það blekkt augað. Sé sent inn gildið ‘20:12:30‘ á DATETIME dálk þá er gildið túlkað sem 2020-12-30. Þó tvípunkturinn sé sendur inn á rithætti tímasetninga þá er hann í dagsetningar hluta. Þetta gildi mætti senda inn á sniðinu ‘2004-00-00 20:12:30‘ og væri þá löglegt. Gildið ‘20:13:30‘ væri sett sem núll gildi því þrettándi mánuðurinn er ekki til.
- MySQL miðlarinn sannreynir ekki hvort dagsetningar séu til í dagatalinu eða ekki. Sé árið á bilinu 1000-9999, mánuðir á bilinu 00 til 12 og dagar á bilinu 00 til 31 tekur hann við þeim og geymir. Gildi sem að einhverju leyti fara út fyrir þessi svið eru túlkuð sem núll gildi, jafnframt leyfir þetta að skráð sé óeðlileg dagsetning s.s. 30 febrúar og verður notendaforritið að sannreyna rétt dagatal.

- Gæta þarf að því að MySQL túlkar tveggja stafa ártöl þannig: 00-69 sem 2000-2069 og 70-99 sem 1970-1999.
- Gildi úr TIMESTAMP dálkum er birt á DATETIME sniði og eldri skilgreiningar dálksins sem leyfðu að skilgreina mætti birtingarbreidd eru fallnar úr gildi.

Tegundin TIME

MySQL sækir og birtir TIME gildi á sniðinu ‘HH:MM:SS‘ og ‘HHH:MM:SS‘ þegar klukkustundir eru mjög háar. Gildissvið tegundarinnar er á bilinu ‘-838:59:59’ til ‘838:59:59‘. Ástæðan fyrir því að klukkustundir geta orðið hærri en 23 er sú að þegar reikna þarf mismun á tímasetningu getur niðurstaðan, sé t.d. reiknað hversu langur tími hefur liðið í klukkustundum á milli fáeinna daga, orðið mjög há.

Setja má TIME á eftirfarandi sniði:

- Á sniðinu 'D HH:MM:SS.fraction' og má einnig nota eftirfarnandi afslappaðar málskipanir: 'HH:MM:SS.fraction', 'HH:MM:SS', 'HH:MM', 'D HH:MM:SS', 'D HH:MM', 'D HH', or 'SS'. Hér stendur D fyrir daga sem geta verið frá 0 til 34 og MySQL geymir ekki ennþá „fraction“ hlutann.
- Sem streng án afmarkara á sniðinu 'HHMMSS' svo fremi að tíminn sé ekki fáránlegur. Sem dæmi er '101112' skilið sem '10:11:12', en '109712' er ólögleg þ.e. mínútan er fáránleg og breytist í núll gildið '00:00:00'.
- Sem tölu á sniðinu HHMMSS svo fremi að hún sé ekki fáránleg. Einnig eru eftirtalin snið leyfileg: SS, MMSS, HHMMSS, HHMMSS.fraction.
- Sem niðurstaða úr falli sem skilar tímagildi s.s. CURRENT_TIME.

Varast skyldi að setja stutt tímagildi á TIME dálka því tímagildið er þá lesið hægra megin frá og túlkað sem liðinn tími þar sem gildið lengst til hægri er lesið í sekúndum. Gildin '1112' og 1112 gætu verið slegin inn í merkingunni '11:12:00' (12 mínútur yfir 11) en MySQL túlkar þau sem '00:11:12' (11 mínútur og 12 sekúndur). Þannig eru '12' og 12 túlkaðar sem '00:00:12'. Tímagildi með tvípunktum eru hinsvegar meðhöndluð frá vinstri til hægri sem tímagildi þar sem '11:12' merkir '11:12:00' en ekki '00:11:12'.

Gildi sem liggja utan gildissviðs TIME dálksins eru klippt niður í hæsta gildi sem dálkurinn leyfir. Gildin '-850:00:00' og '850:00:00' er breytt í '-838:59:59' og '838:59:59'.

Ólögleg gildi breytast í '00:00:00'. Vel að merkja þá er gildið '00:00:00' löglegt í sjálfu sér og ómögulegt er að þekkja muninn.

Tegundin YEAR

YEAR tegundin notar eitt bæti til að túlka ártöl og eru gildin sótt og birt á sniðinu YYYY. Gildissviðið er 1901-2155. Ólögleg gildi eru geymd sem núll gildi. Nota má eftirtalin snið til að setja gildi á slíka dálka:

- Fjögurra tölustafa streng á sniðinu '1901' til '2155'.
- Fjögurra tölustafa tölu á sviðinu 1901 til 2155.
- Sem tveggja tölustafa streng á sviðinu '00' til '99'. Sjá annarsstaðar um slíkt gildissvið.
- Sem tveggja tölustafa tölu samkvæmt sömu reglu og tveggja stafa strengur, utan eftirfarandi: Tilgangslaust er að senda inn töluna 0 og vænta þess að hún sé túlkuð sem 2000 heldur verður að senda hana inn sem ‘0‘ annars er hún túlkuð sem núll gildi.
- Sem niðurstaða úr falli sem skilar tíðagildi s.s. fallið NOW().

Tíðaföll og árþúsundið

Þó MySQL leyfi að ártöl séu send inn á tveggja stafa sniði er ákjósanlegra að nota fjögurra stafa snið frekar en að treysta á hvernig miðlarinn les úr þeim. Þó ORDER BY geti raðað rétt TIMESTAMP og YEAR

tegundum þá eru aðrir þættir s.s. föllin MIN og MAX sem breyta gildunum í tölur þegar unnið er með þau. Séu ártöl geymd á fjögurra stafa sniði þá er hinsvegar hægt að treysta á rétta meðhöndlun þeirra.

Strengja tegundir

Hljóðlátar tegundabreytingar

Í sumum tilfellum breytast tegundir sem skilgreindar eru við CREATE TABLE og ALTER TABLE skipanir yfir í aðrar tegundir en beðið var um.

Margir strengjadálkanna geta tekið breytuna CHARACTER SET (í. Stafamengi) og þá að auki COLLATE (e. Safnhættir[56]). Þetta á við um CHAR, VARCHAR, ENUM og SET.

Dæmi um notkun stafamengja:

```
show character set;
+----------+-----------------------------+---------------------+--------+
| Charset  | Description                 | Default collation   | Maxlen |
+----------+-----------------------------+---------------------+--------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |      2 |
| dec8     | DEC West European           | dec8_swedish_ci     |      1 |
... klippt
| binary   |                             | binary              |      1 |
| geostd8  | GEOSTD8 Georgian            | geostd8_general_ci  |      1 |
+----------+-----------------------------+---------------------+--------+
34 rows in set (0.00 sec)

show collation like 'latin%';
+--------------------+---------+----+---------+----------+---------+
| Collation          | Charset | Id | Default | Compiled | Sortlen |
+--------------------+---------+----+---------+----------+---------+
| latin1_german1_ci  | latin1  |  5 |         |          |       0 |
| latin1_swedish_ci  | latin1  |  8 | Yes     | Yes      |       1 |
... klippt
| latin7_bin         | latin7  | 79 |         |          |       0 |
+--------------------+---------+----+---------+----------+---------+
18 rows in set (0.01 sec)
```

Smíðum töfluna stafir sem getur geymt texta á Unicode sniði í undirmengi þess UTF8.

```
create table stafir(
  st1 char(20) character set utf8,
  st2 char(20) character set latin1 collate latin1_general_ci
 );

insert into stafir values('Halló', 'Halló');

select * from stafir;
+-------+-------+
| st1   | st2   |
+-------+-------+
| Halló | Halló |
+-------+-------+
1 row in set (0.00 sec)
```

[56] Ekki er búið að gefa út viðeigandi þýðingu fyrir ensku gagnagrunns hugtökin Collate og Collation. Nota ég ýmist orðin safnháttur, röðunarháttur eða röðunarsafn fyrir þessi hugtök.

Fyrst eru framkvæmdar tvær fyrirspurnir sem birta öll stafamengi og röðunarsöfn sem MySQL þjónninn hefur uppsett. Því næst er smíðuð taflan „stafir“ með tveim stafadálkum uppsettum í sitthvoru stafamenginu.

CHARACTER SET, eða stafamengið, skilgreinir fyrir töfluna hvernig bókstafir eru geymdir í henni og þ.a.l. hvort stafir eins og sérískensk rittákn komist til skila. COLLATE breytan er viðauki við stafamengið sem skilgreinir hvernig stafir eru bornir saman. Til dæmis er „a“ ekki sama og „A“ ef stafir eru bornir saman eftir tölugildi sínu. Einnig skiptir máli hvort röðunarsafnið kannist við stafi eins og þ, æ og ö því við viljum t.d. að þeir birtist aftast þegar raðað er í stafrófsröð. Þá eru stafirnir a og A ekki jafnir samanber eftirfarandi MySQL fyrirspurn því tölvan þekkir öll rittákn samkvæmt númeragildi:

```
select char(65), char(97);
+----------+----------+
| char(65) | char(97) |
+----------+----------+
| A        | a        |
+----------+----------+
1 row in set (0.00 sec)
```

COLLATION eða röðunarsöfn eru aukaskrár sem skilgreina hvernig skrár skuli bornar saman og í dæminu hér fyrir framan er dálkurinn st1 án sérstaklega tilgreindrar röðunar svo miðlarinn notar þá röðun sem hann telur sjálfgilda en dálkurinn st2 er sérstaklega skilgreindur í venjulegri latin1 röðun án hástafnæmis. Þau röðunarsöfn sem heita „nafn_ci“ og „nafn_cs“ eru ýmist ónæm á hástafi við röðun (e. Case insensitive) eða næm á hástafi (e. Case sensitive). Í dæminu hér fyrir framan er röðunin ónæm á hástafi.

Lengd á stafadálkum er í mæld í stafafjölda en áður var það gert í Bætum.

Yfirlit strengjategunda

Tegund	Skýring
`[NATIONAL] CHAR(M) [BINARY | ASCII | UNICODE]`	Strengur af fastri lengd sem er alltaf fóðraður hægra megin frá með orðabilum svo lengdin sé ávallt nýtt. M stendur fyrir lengd dálksins og má vera frá 0 til og með 255. Slef-bil (e. Trailing spaces) eru fjarlægð þegar CHAR gildi eru sótt. CHAR dálkur með lengd sem fer yfir 255 er breytt í smæstu TEXT tegundina sem geymt getur þá lengd sem tilgreind er. CHAR(500) er þannig breytt í TEXT og CHAR(200000) er breytt í MEDIUMTEXT. CHAR er í rauninni stytting á CHARACTER. NATIONAL CHAR (stytt sem NCHAR) er SQL staðall fyrir skilgreiningu á CHAR dálki sem notar sjálfgefið stafamengi miðlarans. Breytan BINARY gerir röðun og samanburð dálksins hástafanæma (e. Case sensitive), ASCII skilgreinir dálkinn sem stafamengið latin1 og UNICODE skilgreinir hann sem ucs2. Setja má lengdina sem núll, CHAR(0). Þá tekur dálkurinn aðeins einn bita í minnispláss og getur aðeins geymt gildin NULL eða ‘’ (tóman streng).
`CHAR`	Samheiti fyrir CHAR(1).
`[NATIONAL] VARCHAR(M) [BINARY]`	Strengur af breytilegri stærð þar sem M er hámarks lengd dálksins.

	Gildissvið M má vera frá 0 til 255. Slef-bil eru fjarlægð þegar VARCHAR gildi eru geymd. Sé VARCHAR dálkur skilgreindur með hærra gildi en 255 er honum breytt á sama hátt og CHAR, í TEXT, MEDIUMTEXT o.s.frv. sem getur haft áhrif á slef-bil. BINARY gerir röðun og samanburð hástafanæman.
`TINYBLOB` `TINYTEXT`	BLOB (Binary Large Object) eða TEXT dálkur með hámarkslengd 255 (2^8 - 1) stafa (255 Bæti).
`BLOB` `TEXT`	BLOB eða TEXT dálkur af hámarkslengd 65.535 (2^16 -1) stafa (65,5 Kílóbæti).
`MEDIUMBLOB` `MEDIUMTEXT`	BLOB eða TEXT dálkur með hámarkslengd 16.777.215 (2^24 - 1) stafa (16,7 MB).
`LONGBLOB` `LONGTEXT`	BLOB eða TEXT dálkur með hámarkslengd 4.294.967.295 (2^32 - 1) stafa (4 GB). Allt að útgáfu 3.23 af MySQL voru samskiptareglur (e. Protocol) fyrir biðlarann/miðlarann háðar 16 MB samskipta-pakka stærð (e. Communication package), svo var einnig um línustærð í MyISAM töflum. Frá og með útgáfu 4.0 hefur hámarksstærð fyrir LONGBLOB og LONGTEXT verið stillt á hámarksstærð í samskiptapakkanum og laust minni.
`ENUM('value1','value2',...)`	Upptalningartag[57] (e. Enumeration type). Strengur sem getur aðeins geymt eitt gildi sem valið er frá lista af skilgreindum gildum: ‘gildi1‘, ‘gildi2‘, …, NULL eða ‘’ (skekkju gildið). ENUM dálkur getur haft mest 65.535 aðgreind gildi í skilgreiningu sinni en þau eru geymd hið innra sem heiltölur.
`SET('value1','value2',...)`	Mengi (e. Set). Strengur sem getur haft ekkert eða fleiri gildi þar sem hvert gildi sé valið frá skilgreindum gildalista; ‘gildi1’, ‘gildi2’, … Mengjadálkur getur haft mest 64 meðlimi og er hver meðlimur geymdur sem heiltala.

[57] Enumeration type, enumerated type: Upptalningartag er stakrænt tag þar sem stök tagsins eru talin upp í skilgreiningu þess.

Nánar um tegundirnar CHAR og VARCHAR

CHAR og VARCHAR tegundirnar eru svipaðar en greinir þó á í vissum atriðum. VARCHAR gildi taka t.d. aðeins það geymslupláss sem þarf undir þau gildi sem þeim eru gefin. CHAR hins vegar fóðrar gildið með auka orðabilum til að fylla upp í svið sitt. Sé VARCHAR dálki gefið gildi með auka orðabilum (e. Trailing spaces þýð. slef-bil) í endana þá eru þau fjarlægð við geymslu. VARCHAR umbreytir þó ekki stöfum þegar þeir eru sendir inn. Ef þörf er á að geyma slef-bil í dálki er betra að nota BLOB eða TEXT, einnig ef þörf er á að geymt sé lykilorð eða önnur stafatákn þar sem forðast þarf alla möguleika á eðlisbreytingu sviðsins eða eðlisbreytingu stafa. Ef lengri textastrengur er settur inn í CHAR eða VARCHAR dálk en lengd hans tiltekur eru umfram stafirnir klipptir af.

Taflan hér á eftir listar mismunandi geymslupláss sem CHAR og VARCHAR dálkarnir taka. CHAR tekur alltaf það pláss sem hann er skilgreindur í við stofnun, VARCHAR hins vegar tekur pláss undir stafina sem hann geymir og eitt aukabæti til að skrásetja lengd sína. Gildin sem sótt eru í töfluna eru þó hin sömu því miðlarinn fjarlægir slef-bil þegar strengjagildi eru sótt.

Gildi	CHAR(4)	**Geymslupláss**	VARCHAR(4)	**Geymslupláss**
''	' '	4 Bæti	''	1 Bæti
'ab'	'ab '	4 Bæti	'ab'	3 Bæti
'abcd'	'abcd'	4 Bæti	'abcd'	5 Bæti
'abcdefgh'	'abcd'	4 Bæti	'abcd'	5 Bæti

CHAR og VARCHAR dálkarnir nota röðun og samanburð á dálka sína samkvæmt því röðunarsafni (e. Collation) sem tilgreint er fyrir töfluna hverju sinni. Sé ekkert slíkt röðunarsafn tilgreint er notað sjálfgefið röðunarsafn miðlarans. Sé dálkurinn skilgreindur með breytunni BINARY er öll röðun og samanburður á dálkinum framkvæmdur með hástafanæmi.

Nánar um tegundirnar BLOB og TEXT

BLOB stendur fyrir „Binary Large Object“ og er til í fjórum útgáfum: TINYBLOB, BLOB, MEDIUMBLOB og LONGBLOB. Eini munurinn á þeim felst í lengdarskilgreiningu þeirra, þ.e. hámarks geymslugetu í bætum. TEXT tegundirnar eru einnig fjórar; TINYTEXT, TEXT, MEDIUMTEXT og LONGTEXT. Þessar tegundir samsvara BLOB tegundunum og hafa sömu lengdarmöguleika og geymslupláss.

BLOB tegundir eru meðhöndlaðar við t.d. röðun og samanburð sem tvíundar-strengir (e. Binary strings) og eru því hástafanæmir við slíkar aðgerðir. TEXT tegundirnar geta hins vegar framkvæmt röðun og samanburð eftir því stafamengi og röðunarsafni sem tilgreina má fyrir dálkinn.

Sé sett hærra gildi á dálkinn en hann er skilgreindur fyrir er klippt af því við geymslu en engin umbreyting á texta á sér stað. Meðhöndla má TEXT dálka rétt eins og væru þeir VARCHAR nema miklu stærri.

Í ODBC er BLOB meðhöndlað sem LONGVARBINARY og MEDIUMTEXT sem LONGVARCHAR.

Ef nota þarf GROUP BY eða ORDER BY á BLOB eða TEXT dálka þarf að setja þá í fasta lengd fyrst en þetta má gera með SUBSTRING fallinu. Þannig eru tekinn t.d. fyrstu 20 bætin úr hverju gildi og verður til föst lengd 20 stafa gilda fyrir hvert gildi í töflunni. Þá má raða eftir þessum gildum. Ef þetta er ekki gert raðar miðlarinn sjálfrafa eftir fyrstu 1024 bætunum í strengnum, sem getur verið ónákvæmt. Dæmi:

```
SELECT comment FROM tbl_name,SUBSTRING(comment,20) AS substr
  ORDER BY substr;
SELECT id,SUBSTRING(blob_col,1,100) AS b
  FROM tbl_name GROUP BY b;
```

Þó hámarks geymslupláss BLOB og TEXT tegunda sé takmarkað af tegund dálksins geta þó verið takmörk á því hvað stuðpúðar (e. Buffers) í kerfinu leyfa svo og samskipta staðlar. Ef geyma þarf meira en 16MB á línu ætti að gera tilraunir og rannsóknir á því hvaða takmarkanir séu í gildi innan kerfisins s.s. stýrikerfisins.

Um ENUM tegundina

Enska heitið Enumeration hefur ýmist verið þýtt sem Upptalningartag, Upptalningar-tegund, Runuröðuð-tegund eða Notenda-skilgreind raðtegund. Öll þessi hugtök eiga við en hér verður miðað við Tölvuorðasafn Íslenskrar málnefndar sem notar hugtakið Upptalningartag.

ENUM tegundin er strengur sem leyfir að valið sé gildi úr fyrirfram gerðum strengja lista sem skilgreindur er við stofnun dálksins í töflu. Dálkurinn getur einnig geymt NULL gildið og tóman streng (‘’):

- Ef sett er inn gildi í ENUM dálk og gildið finnst ekki í strengjalistanum setur miðlarinn tóman streng í stað gildis og táknar hann þá skekkju eða skekkjugildi. Athuga má hvort ENUM dálkar innihalda þetta skekkjugildi og gera þá viðeigandi ráðstafanir. Tómi strengurinn hefur tölugildið 0 (núll).
- Sé ENUM dálkur skilgreindur með NULL breytunni getur hann geymt NULL gildisleysið og verður NULL þá sjálfgildi. Sé dálkurinn skilgreindur með NOT NULL breytunni verður fyrsti strengur í strengjalistanum sjálfgildi fyrir dálkinn.

Búum til töflu með upptalningar-dálki og lýsum henni því næst:

```
create table upptalningar(
  nr int unsigned auto_increment primary key,
  neytandi char(31) null default 'othekktur',
  n1 enum ('Jamm', 'Sulta') not null,
  n2 enum ('Fransbraud','Rugbraud') null
);

describe upptalningar;
+----------+-------------------------------+------+-----+---------- ...
| Field    | Type                          | Null | Key | Default   ...
+----------+-------------------------------+------+-----+---------- ...
| nr       | int(10) unsigned              |      | PRI | NULL      ...
| neytandi | char(31)                      | YES  |     | othekktur ...
| n1       | enum('Jamm','Sulta')          |      |     | Jamm      ...
| n2       | enum('Fransbraud','Rugbraud') | YES  |     | NULL      ...
+----------+-------------------------------+------+-----+---------- ...
4 rows in set (0.01 sec)
```

Því næst skulum við skjóta inn þrem færslum og gera svo almenna fyrirspurn:

```
insert into upptalningar(neytandi) values('Elli Pelli');
insert into upptalningar(neytandi,n1,n2)
  values('Anna panna','Alpabraud','Rugbraud');
insert into upptalningar (n1,n2) values('Sulta','Fransbraud');

select * from upptalningar;
+----+------------+-------+------------+
| nr | neytandi   | n1    | n2         |
+----+------------+-------+------------+
|  1 | Elli Pelli | Jamm  | NULL       |
```

```
|  2 | Anna panna |       | Rugbraud   |
|  3 | othekktur  | Sulta | Fransbraud |
+----+------------+-------+------------+
3 rows in set (0.00 sec)
```

Finnum nú út hvort einhver dálkur hafi fengið ólöglegt gildi og því settur á tóman streng:

```
select * from upptalningar where n1 = '';
+----+------------+----+----------+
| nr | neytandi   | n1 | n2       |
+----+------------+----+----------+
|  2 | Anna panna |    | Rugbraud |
+----+------------+----+----------+
1 row in set (0.00 sec)
```

Hvert einasta upptalningar-gildi hefur sérstakan sætisvísi (e. Index):

- Leyfð gildi fyrir dálkinn eru tölusett frá sætisvísi 1.
- Skekkjugildið 0 hefur sætisvísinn 0.
- Sætisnúmerið fyrir NULL er NULL.

Þetta merkir m.a. að skoða má gildi listans út frá sætisnúmerum. Sé gefin skipunin:

```
SELECT * FROM tbl_name WHERE enum_col=0;
```

er skilað þeim gildum sem hafa skekkjugildið.

Skoðum ENUM dálk sem er skilgreindur sem ENUM('Einn', 'Tveir') og getur haft hvert og eitt af eftirfarandi gildum.

Gildi	**Sætisvísir**
NULL	NULL
''	0
'Einn'	1
'Tveir'	2

Upptalningar dálkur getur haft að hámarki 65.535 þætti (e. Element) eða sæti. Einn kosta slíkra dálka fyrir utan að leyfa aðeins eitt, og aðeins eitt, þeirra gilda sem skilgreind eru í senn fyrir hverja línu, er að dálkurinn tekur afar lítið geymslupláss. Ef við hefðum 50 færslur af framangreindum dálki, og hver færsla hefði gildið 'Tveir' (5 stafir hver og tæki því 5 bæti hver) myndu þær taka 50 sinnum 5 bæti, 250 bæti alls, í geymslu. Hins vegar er upptalningin geymd aðeins einu sinni og fyrir hvert gildi aðeins skráð sætisvísirinn sem tekur aldrei meira en 1 bæti í senn. Þannig yrðu þessar 50 færslur aðeins 50 Bæti. Segjum nú að við þyrftum að geyma 5 milljón færslur?

Þar sem gildin eru í rauninni geymd sem sætisvísar, þarf aðeins að bæta við útreikningi á dálkaheitin t.d. „dálkaheiti+0“ sem sækir sætisvísana.

Dæmi um notkun sætisvísa:

```
select neytandi, n1+0,n1, n2+0, n2 from upptalningar;
+------------+------+-------+------+------------+
| neytandi   | n1+0 | n1    | n2+0 | n2         |
+------------+------+-------+------+------------+
| Elli Pelli |    1 | Jamm  | NULL | NULL       |
| Anna panna |    0 |       |    2 | Rugbraud   |
| othekktur  |    2 | Sulta |    1 | Fransbraud |
+------------+------+-------+------+------------+
3 rows in set (0.00 sec)
```

```
select * from upptalningar where n2+0 = 0 or n1+0 = 0;
+----+------------+----+----------+
| nr | neytandi   | n1 | n2       |
+----+------------+----+----------+
|  2 | Anna panna |    | Rugbraud |
+----+------------+----+----------+
1 row in set (0.00 sec)
```

Slef-bil eru sjálfkrafa fjarlægð úr strengjalistanum þegar taflan er smíðuð. Þá er gildislistinn ekki hástafanæmur þegar ENUM dálkur geymir innsend gildi en þau eru alltaf birt eins og þau voru skilgreind við stofnun töflunnar.

Ef geymd er tala í ENUM dálki er talan meðhöndluð sem sætisvísir, nema gildin séu send inn með LOAD DATA sem ávallt meðhöndlar innsend gildi sem strengi, samanber:

```
insert into upptalningar(n1) values(1);
```

```
select n1+0,n1 from upptalningar;
+------+-------+
| n1+0 | n1    |
+------+-------+
|    1 | Jamm  |
|    0 |       |
|    2 | Sulta |
|    1 | Jamm  |
+------+-------+
4 rows in set (0.00 sec)
```

Best er að forðast að smíða strengjalista með tölustöfum samanber eftirfarandi lista sem geymir strengina '0', '1', og '2', sem geymast á sætisvísunum 1, 2, og 3. Þetta virkar villandi bæði fyrir miðlarann og forritarann. Einnig þarf að hafa í huga að röðun á ENUM dálki fer fram eftir sætisvísum hans, þannig er upptalningardálkur sem skilgreindur er þannig ENUM('b', 'a') með ‘a’ á eftir ‘b’ í röðun því ‘a’ er í sæti 2 en ‘b’ í sæti 1. NULL gildi raðast á undan öllum gildunum.

Steypa[58] (e. Cast) má röðun ENUM dálka í almenna orðabóka röðun (e. Lexical sorting), þannig í fyrirspurnum: GROUP BY CAST(col AS VARCHAR) eða GROUP BY CONCAT(col). Þetta steypir dálkinum yfir í almennan texta.

Um tegundina SET

Tegundin SET er strengur sem getur haft núll eða fleiri gildi þar sem hvert gildi er valið úr skilgreindu strengjamengi þegar taflan er smíðuð. Dálkurinn getur haft öll þau gildi sem skilgreind eru eða ekki neitt þeirra.

Þegar dálkurinn er búinn til er mengið skilgreint þannig t.d.: SET(‘einn’,’tveir’) NOT NULL, hér getur dálkurinn innihaldið gildin ‘einn’, eða ‘tveir’ eða ‘einn, tveir’. Þar sem komma er notuð til að skilja á milli gilda geta þau sjálf ekki innihaldið kommu. Sé dálkurinn skilgreindur sem SET(‘einn’,’tveir’) NULL, má gefa honum gildið NULL og einnig tóman streng (‘’).

[58] Enska sögnin „to Cast“ er notuð í forritun fyrir það að steypa einni tegund yfir í aðra. Sum íslensk rit nota hér ranglega sögnina „að varpa“ sem er þýðing ensku sagnarinnar „to map.“ Ég nota sögnina „að steypa“ í þeirri merkingu að þegar einu gildi er steypt á annað þá er viðtöku tegundin notuð sem steypumót. Sum staðar nota ég „að kasta“ en einnig mætti segja „að móta.“

Mengisdálkur getur haft flest 64 mismunandi meðlimi og slef-bil eru sjálfkrafa eydd út þegar dálkurinn er skilgreindur.

Meðlimir mengja eru geymdir í tölusamhengi (e. Numeric context) eftir tvíundarkerfi, þannig að fremsti meðlimurinn á lægsta bita í tölunni og aftasti meðlimur á hæsta bitann. Þetta virkar m.a. þannig að sé gildi dálksins sótt í talnasamhengi (svipað og gert er með ENUM) kemur bitasamsetning tölunnar fram eftir tugakerfi.

Þetta virkar þannig á mengisdálkinn „SET('a','b','c','d')“ að:

SET **meðlimur**	**Tugakerfis-gildi**	**Tvíundar-gildi**
'a'	1	0001
'b'	2	0010
'c'	4	0100
'd'	8	1000

Ef t.d. dálkurinn fengi gildið 9, sem er tvíundargildið 1001 yrði fremsti og aftasti meðlimur mengisins fyrir valinu og gildið yrði því 'a,d'.

Ekki skiptir máli í hvaða röð gildin eru send inn á dálkinn, né heldur hversu oft eitthvert gildi er sett fyrir tiltekinn dálk. Þeim er alltaf raðað í þeirri röð sem þau voru skilgreind fyrir töfluna og hver meðlimur er skráður aðeins einu sinni fyrir hvern dálk. Ef SET dálki er sent gildi sem ekki er til í dálkinum er það sniðgengið. Setja má DEFAULT gildi sem sé sjálfkrafa valið sé ekkert gildi sent inn

Ef gleymst hefur hvaða gildi eru leyfð á dálk, má gefa skipunina SHOW COLUMNS FROM tbl_name LIKE set_col.

Dæmi um notkun á mengjadálkum, fyrst smíðum við töflu og heimtum svo lýsingu:

```
create table mengi(
  nr int unsigned auto_increment primary key,
  m1 set('Mjolk','Sykur','Kaffi') not null,
  m2 set('Kex','Snudur')
);


describe mengi;
+-------+-----------------------------+------+----- ...
| Field | Type                        | Null | Key  ...
+-------+-----------------------------+------+----- ...
| nr    | int(10) unsigned            |      | PRI  ...
| m1    | set('Mjolk','Sykur','Kaffi') |      |      ...
| m2    | set('Kex','Snudur')         | YES  |      ...
+-------+-----------------------------+------+----- ...
3 rows in set (0.00 sec)
```

Skjótum nú inn tveim færslum og gerum því næst almenna fyrirspurn á töfluna:

```
insert into mengi values(null,'Mjolk','Kex');
insert into mengi values(null,'Sykur,Mjolk','Kex');

select * from mengi;
+----+-------------+------+
| nr | m1          | m2   |
+----+-------------+------+
|  1 | Mjolk       | Kex  |
|  2 | Mjolk,Sykur | Kex  |
```

```
+----+-------------+------+
2 rows in set (0.00 sec)
```

Finnum nú út talnagildi þeirra gilda sem send voru inn: Við hefðum allt eins getað sent þessar tölur inn:

```
select m1+0, m1 from mengi;
+------+-------------+
| m1+0 | m1          |
+------+-------------+
|    1 | Mjolk       |
|    3 | Mjolk,Sykur |
+------+-------------+
2 rows in set (0.00 sec)
```

Notum nú innbyggt fall sem getur fundið færslur þar sem mengjadálkur inniheldur tiltekið gildi en taka þarf fram hvaða dálkur það er:

```
select * from mengi where find_in_set('mjolk',m1);
+----+-------------+------+
| nr | m1          | m2   |
+----+-------------+------+
|  1 | Mjolk       | Kex  |
|  2 | Mjolk,Sykur | Kex  |
+----+-------------+------+
2 rows in set (0.03 sec)
```

Notum nú almenna SQL fyrirspurn sem leitar að tilteknu gildi eftir algildismynstri, þ.e. öll gildi sem innihalda textann „jol“ í sér.

```
select * from mengi where m1 like '%jol%';
+----+-------------+------+
| nr | m1          | m2   |
+----+-------------+------+
|  1 | Mjolk       | Kex  |
|  2 | Mjolk,Sykur | Kex  |
+----+-------------+------+
2 rows in set (0.00 sec)
```

Ekki má senda „ranga gildið“ á mengis dálk:

```
insert into mengi(m1) values('');

mysql> select * from mengi;
+----+-------------+------+
| nr | m1          | m2   |
+----+-------------+------+
|  1 | Mjolk       | Kex  |
|  2 | Mjolk,Sykur | Kex  |
|  3 |             | NULL |
+----+-------------+------+
3 rows in set (0.00 sec)
```

Þó fer það eftir upprunalegri skilgreiningu dálksins hvernig hann bregst við þegar hann fær ólöglega gildið:

```
insert into mengi values(null,'','');
insert into mengi values(null,'',null);

select * from mengi;
+----+-------------+------+
| nr | m1          | m2   |
+----+-------------+------+
|  1 | Mjolk       | Kex  |
|  2 | Mjolk,Sykur | Kex  |
```

```
|  3 |             | NULL |
|  4 |             |      |
|  5 |             | NULL |
+----+-------------+------+
5 rows in set (0.00 sec)
```

Óleyfileg gildi fá „ólöglega gildið:“

```
insert into mengi(m1) values('Te');
```

```
mysql> select * from mengi;
+----+-------------+------+
| nr | m1          | m2   |
+----+-------------+------+
|  1 | Mjolk       | Kex  |
|  2 | Mjolk,Sykur | Kex  |
|  3 |             | NULL |
|  4 |             |      |
|  5 |             | NULL |
|  6 |             | NULL |
+----+-------------+------+
6 rows in set (0.00 sec)
```

Skorður á stærð dálkategunda

Dálkategundir hafa mismunandi stærðarmörk hvað varðar geymslupláss í bætum. Sumar þessara skorða hafa komið fram áður en hér eru allar stærðarkröfur listaðar. MyISAM töflutegundin leyfir að lína taki hámark 65.534 Bæti og hver BLOB eða TEXT dálkur taka t.d. 5 til 9 bæti af þessari skilgreiningu.

Ef MyISAM eða ISAM töflutegund inniheldur dálka af breytilegri lengd s.s. VARCHAR þá verður línan einnig þannig.

Dálkategund	**Geymslupláss**
`TINYINT`	1 bæti
`SMALLINT`	2 bæti
`MEDIUMINT`	3 bæti
`INT, INTEGER`	4 bæti
`BIGINT`	8 bæti
`FLOAT(p)`	4 bæti ef 0 <= p <= 24, 8 bæti ef 25 <= p <= 53
`FLOAT`	4 bæti
`DOUBLE [PRECISION], teg REAL`	8 bæti
`DECIMAL(M,D), NUMERIC(M,D)`	M+2 bæti ef D > 0, M+1 bæti ef D = 0 (D+2, ef M < D)
`DATE`	3 bæti

`DATETIME`	8 bæti
`TIMESTAMP`	4 bæti
`TIME`	3 bæti
`YEAR`	1 bæti
`CHAR(M)`	M bæti, 0 <= M <= 255
`VARCHAR(M)`	L+1 bæti, þar sem L <= M og 0 <= M <= 255
`TINYBLOB, TINYTEXT`	L+1 bæti, þar sem L < 2^8
`BLOB, TEXT`	L+2 bæti, þar sem L < 2^16
`MEDIUMBLOB, MEDIUMTEXT`	L+3 bæti, þar sem L < 2^24
`LONGBLOB, LONGTEXT`	L+4 bæti, þar sem L < 2^32
`ENUM('gildi1','gildi2',...)`	1 eða 2 bæti, fer eftir hámarksfjölda upptalningar-gilda (65.535 hámark).
`SET('gildi1','gildi2',...)`	1, 2, 3, 4, eða 8 bæti, fer eftir fjölda mengis-meðlima (hámark 64 meðlimir).

VARCHAR þarf eitt bæti í að skrásetja lengd gildisins, að öðru leyti fer plássið eftir lengdinni. Einnig þurfa BLOB og TEXT tegundir 1, 2, 3, og 4 bæti til að skrá lengd sína eftir atvikum.

Að velja rétta tegund á dálka

Til að auka hagræðingu sem mest er best að nota sem nákvæmasta skilgreiningu á dálkum. Ef nauðsynlegt er t.d. að skrásetja heiltölur á bilinu 1 til 99999 er MEDIUMINT UNSIGNED besti kosturinn.

Ekki er nein tegund sem geymir, að svo stöddu, nákvæm gildi fyrir gjaldmiðla og er mælt með notkun DECIMAL tegundarinnar fyrir slík gildi. Útreikningar á DECIMAL gildum fara fram í DOUBLE tegund og geta verið nokkuð nákvæmar. Vandinn felst í því ef gildið er ofurhátt og er bent á það í handbók að breyta megi slíkum tölum í BIGINT ef þurfi að framkvæma útreikninga sem geti orðið afar háir en þeim má alltaf breyta til baka í rauntölur.

7. Kafli. Málskipan SQL aðgerða

Viðauki A. inniheldur yfirlit allra SQL aðgerða sem MySQL styður. Í þessum kafla eru ekki allar málskipanir sýndar eða útskýrðar, heldur miðað við þær sem mest eru notaðar. Einnig er víða búið að skera af hinni eiginlegu málskipan í sama tilgangi: Hér verða aðeins taldar upp þær aðgerðir sem mest eru notaðar og með snöggsoðnara sniði. Þeim verður þó lýst og þær útskýrðar með dæmum.

SQL staðallinn skiptist í tvennt. Annars vegar er málskipan fyrir gagnameðhöndlun á ensku „Data Manipulation Language," skammstafað DML. Hins vegar er málskipan fyrir gagnaskilgreiningar, á ensku „Data Definition Language," skammstafað sem DDL.

Gagnameðhöndlun (DML)

DELETE

```
DELETE FROM töflu_heiti
       [WHERE where_definition]
       [ORDER BY ...]
       [LIMIT row_count]
```

DELETE eyðir línum út úr `töflu_heiti` sem fullnægja því skilyrði sem sett er í `where_definition` og skilar hversu mörgum færslum var eytt.

Ef DELETE skipun er gefin án þess að skilgreina neina where_definition er öllum færslum eytt úr töflunni. Sama árangri má ná með því að gera TRUNCATE TABLE sem einnig eyðir öllum færslum úr töflu:

```
select * from t4;
+-------+-------+
| dalk1 | dalk2 |
+-------+-------+
|     5 |     6 |
|     2 |     6 |
|    34 |  NULL |
+-------+-------+
3 rows in set (0.00 sec)

delete from t4;
Query OK, 3 rows affected (0.00 sec)

select * from t4;
Empty set (0.00 sec)
```

Hægt er að nota LIMIT og ORDER BY þannig að töflunni sé raðað í ákveðna röð og aðeins sé eytt t.d. 2 færslum frekar en öllum hugsanlegum. Eftirfarandi dæmi skilgreinir eyðingu á öllum færslum í töflunni t1 þar sem dálkur 2 hefur hærra gildi en 5, þetta gæti eytt öllum færslum hennar en við skilgreinum að aðeins megi eyða einni færslu:

```
select * from t1;
+-------+-------+
| dalk1 | dalk2 |
+-------+-------+
|     5 |     6 |
|     2 |     6 |
|    34 |  NULL |
+-------+-------+
3 rows in set (0.00 sec)

delete from t1 where dalk2 > 5 order by dalk1 limit 1;
```

```
Query OK, 1 row affected (0.02 sec)

select * from t1;
+-------+-------+
| dalk1 | dalk2 |
+-------+-------+
|     5 |     6 |
|    34 |  NULL |
+-------+-------+
2 rows in set (0.00 sec)
```

Hægt er að eyða úr mörgum töflum í einu ef því er að skipta.

INSERT

```
INSERT
    [INTO] töflu_heiti [(col_name,...)]
    VALUES ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

Eða:

```
INSERT  [IGNORE]
    [INTO] töflu_heiti
    SET col_name={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

Eða:

```
INSERT
     töflu_heiti [(col_name,...)]
    SELECT ...
```

Færir nýjar línur inn í töflu, sem þegar er til, INSERT..VALUES og INSERT..SET er ætluð til að tilgreina sérstök gildi en INSERT..SELECT setur inn þau gildi sem finnast í annarri töflu.

Ef ekki er ætlunin að setja inn gildi í alla dálka töflunnar þarf að tilgreina þá sérstaklega en að öðrum kosti þarf aðeins að tilgreina gildin sem send eru inn. Senda má margar færslur inn í einu. Eftirfarandi dæmi notar fjórar útfærslur á fyrstu tveim málskipununum hér að framan.

```
show columns from sjoppur;
+---------+------------------+------+-----+---------+----------------+
| Field   | Type             | Null | Key | Default | Extra          |
+---------+------------------+------+-----+---------+----------------+
| numer   | int(10) unsigned |      | PRI | NULL    | auto_increment |
| kennit  | varchar(10)      |      |     |         |                |
| nafn    | varchar(31)      | YES  |     | NULL    |                |
| heima   | varchar(25)      | YES  |     | NULL    |                |
| pnr     | char(3)          | YES  |     | NULL    |                |
| eigandi | varchar(10)      | YES  | MUL | NULL    |                |
+---------+------------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

insert into sjoppur
   values(null,'1234560004','Bitinn','Tröðum','305','1234560002');
insert into sjoppur(nafn,heima,pnr) values('Kot','Tröðum','305');
insert into sjoppur(nafn,heima) values
     ('Aurinn','Stræti'),('Bjólan','Stígum'),('Dollan','Vegum');

insert into sjoppur
   set eigandi='1234560003',
   nafn = 'Gjugg',
   kennit = '1234560006';

select * from sjoppur;
```

```
+-------+------------+-------------+------------+------+------------+
| numer | kennit     | nafn        | heima      | pnr  | eigandi    |
+-------+------------+-------------+------------+------+------------+
... klippt
|     8 |            | Kot         | Trödum     | 305  | NULL       |
|     9 |            | Aurinn      | Stræti     | NULL | NULL       |
|    10 |            | Bjólan      | Stígum     | NULL | NULL       |
|    11 |            | Dollan      | Vegum      | NULL | NULL       |
|    12 | 1234560006 | Gjugg       | NULL       | NULL | 1234560003 |
+-------+------------+-------------+------------+------+------------+
11 rows in set (0.00 sec)
```

Setja má inn í töflu þau gildi sem undir-fyrirspurn sækir frá annarri töflu með INSERT..SELECT aðgerð. Eftirfarandi dæmi sýnir hvernig það er gert:

```
create temporary table tmpsjoppur like sjoppur;

insert into tmpsjoppur select * from sjoppur;

insert into tmpsjoppur(kennit,nafn) select kennit,nafn from sjoppur;

select * from tmpsjoppur;
+-------+------------+-------------+------------+------+------------+
| numer | kennit     | nafn        | heima      | pnr  | eigandi    |
+-------+------------+-------------+------------+------+------------+
... klippt
|    21 |            | Bjólan      | NULL       | NULL | NULL       |
|    22 |            | Dollan      | NULL       | NULL | NULL       |
|    23 | 1234560006 | Gjugg       | NULL       | NULL | NULL       |
+-------+------------+-------------+------------+------+------------+
22 rows in set (0.00 sec)
```

Með því að gefa lykilorðið IGNORE, þ.e. INSERT IGNORE ..., getur MySQL sneitt framhjá aðallyklum (sem eru væntanlega einkvæmir). Þetta er hentugt ef setja þarf inn mörg gildi í einu því að öðrum kosti myndi INSERT aðgerðin stöðvast ef upp kæmi villa.

Einnig má nota málskipunina ON DUPLICATE KEY UPDATE til að dagrétta eigindi aðallykils frekar en að hætta við færsluna vegna þess að aðallykillinn sé til fyrir.

Dæmi með IGNORE

Búin til tafla sem notar kennitölu sem aðallykil:

```
create table nafn(
    kt  varchar(10) not null primary key,
    nafn varchar(31),
    f_dag date default '19750000');
```

Nú reynum við að afrita færslur úr sjoppur yfir í nafn en fáum bara sumar færslurnar vegna þess að „primary key“ svið leyfir ekki tvítekningu á kennitölum. Í töflunni sjoppur eru nokkrar kennitölur sem hafa sama gildið:

```
select distinct kennit, nafn from sjoppur;
+------------+-------------+
| kennit     | nafn        |
+------------+-------------+
| 1234567890 | Pallasjoppa |
| 1234567891 | Jóasjoppa   |
| 1234560002 | Jói Jóa     |
| 1234560003 | Sigga Sigg  |
| 1234560001 | Palli Palla |
```

```
| 1234560004 | Bitinn      |
|            | Kot         |
|            | Aurinn      |
|            | Bjólan      |
|            | Dollan      |
| 1234560006 | Gjugg       |
+------------+-------------+
11 rows in set (0.00 sec)

insert ignore into nafn(kt, nafn)
     select distinct kennit, nafn from sjoppur;


select * from nafn;
+------------+-------------+------------+
| kt         | nafn        | f_dag      |
+------------+-------------+------------+
| 1234567890 | Pallasjoppa | 1975-00-00 |
| 1234567891 | Jóasjoppa   | 1975-00-00 |
| 1234560002 | Jói Jóa     | 1975-00-00 |
| 1234560003 | Sigga Sigg  | 1975-00-00 |
| 1234560001 | Palli Palla | 1975-00-00 |
| 1234560004 | Bitinn      | 1975-00-00 |
|            | Kot         | 1975-00-00 |
| 1234560006 | Gjugg       | 1975-00-00 |
+------------+-------------+------------+
8 rows in set (0.00 sec)
```

Dæmi með ON DUPLICATE KEY

Við getum skilgreint að við innskot sé tiltekið gildi í töflunni dagréttað sé það til fyrir. Eftirfarandi dæmi reynir að setja inn nafn og kennitölu einstaklings. Þar sem kennitalan gæti verið til fyrir í töflunni og það svið er aðallykill (einkvæmt) þá tiltökum við að nafn kennitölunnar sé dagréttað sé hún til fyrir:

```
insert into nafn(kt, nafn) values('1234560006','Pelias')
    on duplicate key update nafn='Pelias';

select * from nafn;
+------------+-------------+------------+
| kt         | nafn        | f_dag      |
+------------+-------------+------------+
... klippt
| 1234560005 | Gjugg       | 1975-00-00 |
| 1234560006 | Pelias      | 1975-00-00 |
+------------+-------------+------------+
9 rows in set (0.00 sec)
```

Prófum nú að breyta nafni þessarar kennitölu:

```
insert into nafn(kt, nafn) values('1234560006','Gaur')
    on duplicate key update nafn='Gaur';

select * from nafn;
+------------+-------------+------------+
| kt         | nafn        | f_dag      |
+------------+-------------+------------+
... klippt
| 1234560005 | Gjugg       | 1975-00-00 |
| 1234560006 | Gaur        | 1975-00-00 |
+------------+-------------+------------+
9 rows in set (0.00 sec)
```

LOAD DATA INFILE og SELECT INTO OUTFILE

```
LOAD DATA  [LOCAL] INFILE 'file_name.txt'    SELECT select_expr,...
    INTO TABLE töflu_heiti                     [INTO OUTFILE 'file_name'
    [FIELDS                                    export_options]
        [TERMINATED BY '\t']                   [FROM table_references
        [[OPTIONALLY] ENCLOSED BY '']          [WHERE where_definition]
            ]
    [LINES
        [STARTING BY '']
        [TERMINATED BY '\n']
    ]
    [IGNORE number LINES]
    [(col_name,...)]
```

LOAD DATA INFILE les línur úr textaskrá inn í tiltekna töflu. Sömuleiðis má nota forritið mysqlimport til að ná sama árangri. Textaskráin ætti ekki að hafa neinar SQL aðgerðir heldur vera gögn eins og þau koma fyrir í töflu.

Þessi aðgerð er hentug til að taka afrit af gögnum, færa þau á milli miðlara/tölva eða að lesa inn gögn sem áður voru á sniði töflureiknis eða ritvinnslu. Yfirleitt þegar gögn eru færð á milli með textaskrám er algengt að dálkar séu aðskildir með ýmist dálkainnslætti (Tab á lyklaborðinu) eða semíkommu (;). Sömuleiðis kemur fyrir að gildin eru sett innan tvíhöggs (“) eða úrfellingarmerkis (‘).

Ef FIELDS hlutinn er ekki skilgreindur lítur MySQL á eftirfarandi sem sjálfgefið:

```
FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'
```

Lykilorðið LOCAL skilgreinir að mysql-biðlarinn (eða hvaða biðlari sem notar þessa aðgerð) þarf að lesa skrána og senda hana yfir gagnatenginguna til miðlarans. Sé LOCAL hins vegar sleppt þarf skráin að vera til á miðlara-tölvunni og hann les hana sjálfur.

Eftirfarandi dæmi afritar allar línur í töflunni „nafn“ í textaskrá, síðan er öllum færslum eytt úr töflunni og að lokum lesnar inn aftur úr textaskránni:

```
select * into outfile 'd:/nafnat.sql'
   fields terminated by ';' enclosed by '"'
   from nafn;
Query OK, 9 rows affected (0.00 sec)

delete from nafn;

select * from nafn;
Empty set (0.00 sec)

load data infile 'd:/nafnat.sql' into table nafn
   fields terminated by ';' enclosed by '"';
Query OK, 9 rows affected (0.01 sec)

select * from nafn;
+------------+-------------+------------+
| kt         | nafn        | f_dag      |
+------------+-------------+------------+
| 1234567890 | Pallasjoppa | 1975-00-00 |
| 1234567891 | Jóasjoppa   | 1975-00-00 |
... klippt
| 1234560005 | Gjugg       | 1975-00-00 |
```

```
| 1234560006 | Gaur        | 1975-00-00 |
+------------+-------------+------------+
9 rows in set (0.00 sec)
```

Sé lykilorðið OPTIONALLY notað á undan ENCLOSED BY eru aðeins textagildi sem verða rituð/lesin innan tvíhöggs en að öðrum kosti eru öll gildin sett innan tvíhöggs/úrfellingamerkis sé ENCLOSED BY '*staf*' notað.

Svona líta tvær línanna út í textaskránni sem unnið var með hér fyrir framan.

```
"1234560003";"Sigga Sigg";"1975-00-00"
"1234560001";"Palli Palla";"1975-00-00"
```

Sé enginn dálkalisti skilgreindur, samanber dæmin hér á undan er gert ráð fyrir að allir dálkar töflunnar eigi samsvarandi dálka í textaskránni. Sé svo ekki, t.d. að textaskráin innihaldi eingöngu kennitölur mætti gera eftirfarandi:

```
load data infile 'd:/nafnat.sql' into table nafn
    fields terminated by ';' enclosed by '"' (kt);
```

Hér er gert ráð fyrir að textaskráin innihaldi eingöngu kennitölu dálk og við færum allar þær færslur inn í dálkinn. Ef fremsti dálkurinn í textaskránni innihéldi nöfn og annar dálkurinn kennitölur hefði þurft að færa inn:

```
fields terminated by ';' enclosed by '"' (nafn, kt);
```

REPLACE

REPLACE virkar nákvæmlega eins og INSERT nema ef reynt er að færa inn ný gildi á færslur sem hafa sömu gildi á aðallykli (PRIMARY KEY) eða undir einkvæmis skilgreiningu (UNIQUE) er gömlu færslunum eytt úr töflunni og hinar nýju settar í staðinn.

SELECT

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
    select_expr,...
    [FROM table_references
      [WHERE where_definition]
      [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_definition]
      [ORDER BY {col_name | expr | position}
        [ASC | DESC] ,...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Eða:

```
SELECT
    select_expr,...
    [FROM table_references
      [WHERE where_definition]
```

Aðgerðin SELECT er notuð til að sækja og birta upplýsingar úr töflum eða yrðingum og föllum hjá miðlaranum. Allra einföldustu SELCECT aðgerðir geta verið eins og eftirfarandi, þar sem sóttar eru niðurstöður úr keyrslu á fjórum föllum:

```
select now(), user(), database(), version();
+---------------------+----------------+------------+----------------+
| now()               | user()         | database() | version()      |
+---------------------+----------------+------------+----------------+
| 2004-07-29 13:15:40 | root@localhost | malskipan  | 5.0.0-alpha-nt |
+---------------------+----------------+------------+----------------+
```

```
1 row in set (0.03 sec)
```

Í dæminu hér að framan þurfti ekki að tilgreina hvaðan upplýsingarnar koma þ.e. ekki var sótt í neinar töflur, sem þó er yfirleitt venjan.

Eftirfarandi dæmi eru þrjár SQL aðgerðir sem sækja nafn og heimilisfang þeirra sem skráðir eru í töfluna sjoppur. Fyrsta aðgerðin sækir allar færslur en lykilorðinu ALL má sleppa.

Næsta aðgerð sækir aðeins þær línur/færslur sem hafa einstök gildi og þriðja aðgerðin sækir alla dálka, allar línur úr sömu töflu:

```
select distinct nafn, heima from sjoppur;
+-------------+------------+
| nafn        | heima      |
+-------------+------------+
| Pallasjoppa | Sigurtröd  |
| Jóasjoppa   | Jodtröd    |
| Jói Jóa     | Kindabraut |
| Sigga Sigg  | Hérabraut  |
| Palli Palla | Bolabraut  |
| Bitinn      | Trödum     |
| Kot         | Trödum     |
| Aurinn      | Stræti     |
| Bjólan      | Stígum     |
| Dollan      | Vegum      |
| Gjugg       | NULL       |
+-------------+------------+
11 rows in set (0.00 sec)
```

```
select * from sjoppur;
+-------+------------+-------------+------------+------+------------+
| numer | kennit     | nafn        | heima      | pnr  | eigandi    |
+-------+------------+-------------+------------+------+------------+
|     1 | 1234567890 | Pallasjoppa | Sigurtröd  | 100  | 1234560001 |
|     2 | 1234567891 | Jóasjoppa   | Jodtröd    | 205  | 1234560002 |
|     4 | 1234560002 | Jói Jóa     | Kindabraut | 206  | NULL       |
|     5 | 1234560003 | Sigga Sigg  | Hérabraut  | 306  | NULL       |
|     6 | 1234560001 | Palli Palla | Bolabraut  | 100  | NULL       |
|     7 | 1234560004 | Bitinn      | Trödum     | 305  | 1234560002 |
|     8 |            | Kot         | Trödum     | 305  | NULL       |
|     9 |            | Aurinn      | Stræti     | NULL | NULL       |
|    10 |            | Bjólan      | Stígum     | NULL | NULL       |
|    11 |            | Dollan      | Vegum      | NULL | NULL       |
|    12 | 1234560006 | Gjugg       | NULL       | NULL | 1234560003 |
|    13 |            | Bitinn      | Trödum     | NULL | NULL       |
+-------+------------+-------------+------------+------+------------+
12 rows in set (0.00 sec)
```

Næsta dæmi sækir aðeins þær færslur þar sem kennitalan hefur gildi á sniðinu ‘1234...’ og því aðeins að póstnúmerið sé ekki NULL.

```
select kennit, nafn, heima, pnr from sjoppur
    where kennit like '1234%' and pnr is not null;
+------------+-------------+------------+------+
| kennit     | nafn        | heima      | pnr  |
+------------+-------------+------------+------+
| 1234567890 | Pallasjoppa | Sigurtröd  | 100  |
| 1234567891 | Jóasjoppa   | Jodtröd    | 205  |
| 1234560002 | Jói Jóa     | Kindabraut | 206  |
| 1234560003 | Sigga Sigg  | Hérabraut  | 306  |
| 1234560001 | Palli Palla | Bolabraut  | 100  |
| 1234560004 | Bitinn      | Trödum     | 305  |
+------------+-------------+------------+------+
6 rows in set (0.00 sec)
```

Nota má alls konar samanburðarhætti í WHERE klausunni. Dæmið hér fyrir framan notaði virkjann LIKE sem er notaður til að bera saman strengi eftir ákveðnu mynstri, t.d. hefði mátt skilgreina „kennit LIKE '_23%" sem hefði þá sótt allar kennitölur sem hefðu haft 23 sem annan og þriðja staf, hvaða staf sem er í fyrsta sæti og allar hugsanlegar endingar.

Sömuleiðis má nota IS NULL og IS NOT NULL, samanber:

```
select * from sjoppur where eigandi IS NULL;
+-------+------------+-------------+------------+------+---------+
| numer | kennit     | nafn        | heima      | pnr  | eigandi |
+-------+------------+-------------+------------+------+---------+
|     4 | 1234560002 | Jói Jóa     | Kindabraut | 206  | NULL    |
|     5 | 1234560003 | Sigga Sigg  | Hérabraut  | 306  | NULL    |
|     6 | 1234560001 | Palli Palla | Bolabraut  | 100  | NULL    |
|     8 |            | Kot         | Trödum     | 305  | NULL    |
|     9 |            | Aurinn      | Stræti     | NULL | NULL    |
|    10 |            | Bjólan      | Stígum     | NULL | NULL    |
|    11 |            | Dollan      | Vegum      | NULL | NULL    |
|    13 |            | Bitinn      | Trödum     | NULL | NULL    |
+-------+------------+-------------+------------+------+---------+
8 rows in set (0.00 sec)

select * from sjoppur where eigandi IS NULL and pnr IS NOT NULL;
+-------+------------+-------------+------------+------+---------+
| numer | kennit     | nafn        | heima      | pnr  | eigandi |
+-------+------------+-------------+------------+------+---------+
|     4 | 1234560002 | Jói Jóa     | Kindabraut | 206  | NULL    |
|     5 | 1234560003 | Sigga Sigg  | Hérabraut  | 306  | NULL    |
|     6 | 1234560001 | Palli Palla | Bolabraut  | 100  | NULL    |
|     8 |            | Kot         | Trödum     | 305  | NULL    |
+-------+------------+-------------+------------+------+---------+
4 rows in set (0.00 sec)
```

Nota má fleiri tegundir af samanburðar virkjum í WHERE, og raunar allsstaðar þar sem nota má yrðingar svosem hér er gert:

```
select numer, nafn, pnr, eigandi from sjoppur
   where numer > 4 and numer <= 8 and eigandi IS NULL or pnr > 300;
+-------+-------------+------+------------+
| numer | nafn        | pnr  | eigandi    |
+-------+-------------+------+------------+
|     5 | Sigga Sigg  | 306  | NULL       |
|     6 | Palli Palla | 100  | NULL       |
|     7 | Bitinn      | 305  | 1234560002 |
|     8 | Kot         | 305  | NULL       |
+-------+-------------+------+------------+
4 rows in set (0.02 sec)
```

Í dæminu hér fyrir framan viljum við allar færslur þar sem númer er á bilinu 5 til 8, eigandi hefur verið skráður inn, eða alla sem eru skráðir á póstnúmer 301 og hærra. Gætið vel að notkun á AND og OR í þessu dæmi.

Eftirfarandi dæmi sýnir hvernig nota má ORDER BY til að raða niðurstöðunum:

```
select numer, nafn, pnr, eigandi from sjoppur
   where numer > 4 and numer <= 8 and eigandi IS NULL or pnr > 300
   order by nafn;
+-------+-------------+------+------------+
| numer | nafn        | pnr  | eigandi    |
+-------+-------------+------+------------+
|     7 | Bitinn      | 305  | 1234560002 |
|     8 | Kot         | 305  | NULL       |
|     6 | Palli Palla | 100  | NULL       |
|     5 | Sigga Sigg  | 306  | NULL       |
```

```
+-------+------------+------+------------+
4 rows in set (0.00 sec)
```

Eftirfarandi dæmi sýnir hvernig niðurstöðurnar hafa verið hópaðar eftir póstnúmerum, en það þýðir líka að aðeins eitt póstnúmer er birt fyrir hverja færslu. Þessi klausa hentar betur sé hún notuð með samsteypu[59] föllum á borð við SUM(), AVG(), MAX() og MIN():

```
select numer, count(nafn), pnr from sjoppur
    where numer > 4 and numer <= 8 and eigandi IS NULL or pnr > 300
    group by pnr;
+-------+-------------+------+
| numer | count(nafn) | pnr  |
+-------+-------------+------+
|     6 |           1 | 100  |
|     7 |           2 | 305  |
|     5 |           1 | 306  |
+-------+-------------+------+
3 rows in set (0.01 sec)

select numer, count(nafn), pnr from sjoppur
    group by pnr;
+-------+-------------+------+
| numer | count(nafn) | pnr  |
+-------+-------------+------+
|     9 |           5 | NULL |
|     1 |           2 | 100  |
|     2 |           1 | 205  |
|     4 |           1 | 206  |
|     7 |           2 | 305  |
|     5 |           1 | 306  |
+-------+-------------+------+
6 rows in set (0.00 sec)
```

Samsetning taflna með SELECT

Nota má margar töflur í einu þegar SELECT er framkvæmt og eru þær þá listaðar aftan við FROM með kommu á milli sín. Ef dálkar í tveim töflum heita sama nafni þarf að aðgreina þá með því að rita *töflunafn.dálkanafn* í aðgerðinni:

Byrjum á að skoða allar færslur í sjoppur:

```
select nafn, pnr from sjoppur;
+-------------+------+
| nafn        | pnr  |
+-------------+------+
| Pallasjoppa | 100  |
| Jóasjoppa   | 205  |
| Jói Jóa     | 206  |
| Sigga Sigg  | 306  |
| Palli Palla | 100  |
| Bitinn      | 305  |
| Kot         | 305  |
| Aurinn      | NULL |
| Bjólan      | NULL |
| Dollan      | NULL |
| Gjugg       | NULL |
| Bitinn      | NULL |
+-------------+------+
```

[59] Samsteypa: Aggregate

```
12 rows in set (0.00 sec)
```

Því næst skoðum við allar færslur úr sjoppur, nema þær sem hafa NULL í póstnúmeri, auk þess viljum við birta öll póstsvæðaheiti úr töflunni postnr, þar sem *sjoppur.pnr* er hið sama og *postnr.numer*. Við verðum að greina á milli postnr.numer og sjoppur.numer því það eru alls ekki samskonar dálkar og hefðum því fengið villu.

```
select nafn, pnr, heiti from sjoppur, postnr
   where sjoppur.pnr is not null and pnr = postnr.numer;
+-------------+------+-----------+
| nafn        | pnr  | heiti     |
+-------------+------+-----------+
| Pallasjoppa | 100  | Reykjabær |
| Palli Palla | 100  | Reykjabær |
| Jóasjoppa   | 205  | Kópabær   |
| Jói Jóa     | 206  | Kópahæd   |
| Sigga Sigg  | 306  | Akraþorp  |
| Bitinn      | 305  | Akraþorp  |
| Kot         | 305  | Akraþorp  |
+-------------+------+-----------+
7 rows in set (0.00 sec)
```

Að lokum röðum við úttakinu eftir númeri sjoppanna en áður var raðað eftir póstnúmerum:

```
select nafn, pnr, heiti from sjoppur, postnr
     where sjoppur.pnr is not null and pnr = postnr.numer
     order by sjoppur.numer;
+-------------+------+-----------+
| nafn        | pnr  | heiti     |
+-------------+------+-----------+
| Pallasjoppa | 100  | Reykjabær |
| Jóasjoppa   | 205  | Kópabær   |
| Jói Jóa     | 206  | Kópahæd   |
| Sigga Sigg  | 306  | Akraþorp  |
| Palli Palla | 100  | Reykjabær |
| Bitinn      | 305  | Akraþorp  |
| Kot         | 305  | Akraþorp  |
+-------------+------+-----------+
7 rows in set (0.00 sec)
```

Viðurnefni (Alias) fyrir dálka og töflur

Í öllum SQL yrðingum (e. Expressions) má gefa dálkum og töflum viðurnefni (e. Alias) eða uppnefni. Þetta má framkvæma á tvo vegu eins og eftirfarandi tvær aðgerðir sýna, sem endurnefna alla dálka og töflur úr aðgerðunum hér fyrir framan:

```
select nafn as n, pnr as p, heiti as h from sjoppur as sj, postnr as po
   where sj.pnr is not null and pnr = po.numer
   order by sj.numer;
select nafn n, pnr p, heiti h from sjoppur sj, postnr po
   where sj.pnr is not null and pnr = po.numer
   order by sj.numer;
```

Málskipan fyrir JOIN samsetningar

```
töflu_tilvísun, töflu_tilvísun
töflu_tilvísun [INNER | CROSS] JOIN töflu_tilvísun [join_skilyrði]
töflu_tilvísun STRAIGHT_JOIN töflu_tilvísun
töflu_tilvísun LEFT [OUTER] JOIN töflu_tilvísun [join_skilyrði]
töflu_tilvísun NATURAL [LEFT [OUTER]] JOIN töflu_tilvísun
{ OJ töflu_tilvísun LEFT OUTER JOIN töflu_tilvísun
    ON conditional_expr }
```

```
töflu_tilvísun RIGHT [OUTER] JOIN töflu_tilvísun [join_skilyrði]
töflu_tilvísun NATURAL [RIGHT [OUTER]] JOIN töflu_tilvísun
```

Þar sem `töflu_tilvísun` skilgreinist sem:

```
töflu_heiti [[AS] alias]
    [[USE INDEX (key_list)]
      | [IGNORE INDEX (key_list)]
      | [FORCE INDEX (key_list)]]
```

Þar sem `join_skilyrði` skilgreinist sem:

```
ON conditional_expr | USING (column_list)
```

Almennt séð ef notað er lykilorðið JOIN ætti ekki að hafa neitt skilyrði í ON hlutanum sem hefur takmarkandi áhrif á niðurstöðurnar. Sumt af því sem listað er í málskipaninni s.s. LEFT OUTER JOIN og OUTER JOIN er haft fyrir samhæfni við ODBC staðalinn. Nota má viðurnefni á töflur líkt og annars staðar í SELECT.:

```
töflu_heiti AS viður_nefni eða töflu_heiti viður_nefni join_condition
```

Í ON hlutanum má nota allar tegundir skilyrðinga sem annars eru löglegar í WHERE hluta.

Í sjálfu sér er JOIN skilyrðing mun einfaldari í notkun en virðist við fyrstu sýn. Þegar sóttar eru upplýsingar í töflu á borð við eftirfarandi dæmi, eru sóttar allar færslur í báðum töflum (eða fleiri) sem fullnægja einhverju skilyrði til dæmis að para saman allar færslur sem hafa sama póstnúmer:

```
select * from sjoppur, postnr
  where sjoppur.pnr = postnr.pnr;
+-------+------------+-------------+------------+------+------------+-----+----------
| numer | kennit     | nafn        | heima      | pnr  | eigandi    | pnr | heiti
+-------+------------+-------------+------------+------+------------+-----+---------
|     1 | 1234567890 | Pallasjoppa | Sigurtröð  | 100  | 1234560001 | 100 | Reykjabær
|     6 | 1234560001 | Palli Palla | Bolabraut  | 100  | NULL       | 100 | Reykjabær
|     2 | 1234567891 | Jóasjoppa   | Joðtröð    | 205  | 1234560002 | 205 | Kópabær
... klippt
```

Hér eru sóttar allar færslur í töfluna sjoppur og við hverja færslu er parað við færslu úr póstnúmera töflunni sem hefur sama póstnúmer. Þetta er bæði einfaldasta JOIN aðferðin og sú langmest notaða.

LEFT JOIN virkar þannig að allar færslur úr töflunni vinstra megin birtast, þó tengi-skilyrðið á milli taflanna sé ekki fullnægjandi í þeirri hægri. Ef t.d. einhver í sjoppu töflunni hér á undan byggi á póstnúmeri 555 og það væri ekki til staðar í póstnúmera töflunni þá hefði sá aðili ekki birst í niðurstöðunum hér fyrir framan. Hins vegar mætti skilgreina sérstaklega LEFT JOIN og þá hefði hann birst með NULL sem póstsvæði. RIGHT JOIN virkar á sama hátt en frá hægri hlið.

Dæmin hér á eftir nota tvær töflur, tafla1 sem geymir upplýsingar um álegg og tafla2 sem geymir upplýsingar um brauðmeti, reynt verður að para saman brauðmeti og álegg eftir númerum þeirra í báðum töflum:

Fyrst skulum við skoða hvað er í báðum töflum, sem hvor um sig inniheldur aðeins þrjár færslur, gættu vel að númerum færslnanna:

```
select * from tafla1;
+------+--------+
| num  | txt    |
+------+--------+
|    5 | sulta  |
|    7 | ostur  |
|    6 | skinka |
+------+--------+
3 rows in set (0.00 sec)
```

```
select * from tafla2;
+------+-----------+
| num  | texti     |
+------+-----------+
|    5 | kex       |
|    6 | braud     |
|    8 | flatbraud |
+------+-----------+
3 rows in set (0.00 sec)
```

Næsta fyrirspurn samtvinnar (e. Join) töflurnar án þess að skilyrða það frekar, því mun vinstri taflan birta allar sínar færslur þar sem hver færsla er pöruð einu sinni á hverja færslu í hægri töflunni. Ef hvor tafla hefði t.d. 9 færslur hefðum við fengið 9^9 færslur alls.

```
select * from tafla1, tafla2
  order by tafla1.num;
+------+--------+------+-----------+
| num  | txt    | num  | texti     |
+------+--------+------+-----------+
|    5 | sulta  |    5 | kex       |
|    5 | sulta  |    6 | braud     |
|    5 | sulta  |    8 | flatbraud |
|    6 | skinka |    8 | flatbraud |
|    6 | skinka |    5 | kex       |
|    6 | skinka |    6 | braud     |
|    7 | ostur  |    5 | kex       |
|    7 | ostur  |    6 | braud     |
|    7 | ostur  |    8 | flatbraud |
+------+--------+------+-----------+
9 rows in set (0.00 sec)
```

Næsta fyrirspurn leitast við að para saman áleggin eftir númerum sínum, þannig verður sulta pöruð á kex þar eð báðar færslurnar eru númer 5. Núna vantar ost númer 7 og flatbraud númer 8.

```
select * from tafla1, tafla2
  where tafla1.num=tafla2.num;
+------+--------+------+-------+
| num  | txt    | num  | texti |
+------+--------+------+-------+
|    5 | sulta  |    5 | kex   |
|    6 | skinka |    6 | braud |
+------+--------+------+-------+
2 rows in set (0.00 sec)
```

Með því að nota LEFT JOIN krefjumst við þess að allar færslur í vinstri töflunni skuli birtast þó ekki sé til færsla á hana í þeirri hægri, svo við fáum NULL í pörun. Næsta dæmi á eftir birtir samskonar niðurstöður nema þar er dæminu snúið við með RIGHT JOIN.

```
select * from tafla1 left join tafla2 using(num)
  order by tafla1.num;
+------+--------+------+-------+
| num  | txt    | num  | texti |
+------+--------+------+-------+
|    5 | sulta  |    5 | kex   |
|    6 | skinka |    6 | braud |
|    7 | ostur  | NULL | NULL  |
+------+--------+------+-------+
3 rows in set (0.00 sec)
```

```
select * from tafla1 right join tafla2 using(num)
  order by tafla2.num;
+------+--------+------+-----------+
| num  | txt    | num  | texti     |
+------+--------+------+-----------+
|    5 | sulta  |    5 | kex       |
|    6 | skinka |    6 | braud     |
| NULL | NULL   |    8 | flatbraud |
+------+--------+------+-----------+
3 rows in set (0.00 sec)
```

Fáein dæmi um samtvinnun:

```
select * from sjoppur, postnr where sjoppur.pnr=postnr.pnr;
select * from sjoppur left join postnr on sjoppur.pnr=postnr.pnr;
select * from sjoppur left join postnr using(pnr);
select * from sjoppur left join postnr using(pnr);
select * from sjoppur right join postnr using(pnr);
select * from sjoppur straight join postnr using(pnr);
select * from sjoppur, postnr where sjoppur.pnr = postnr.pnr;
```

UNION

```
SELECT ...
UNION [ALL | DISTINCT]
SELECT ...
  [UNION [ALL | DISTINCT]
   SELECT ...]
```

Lykilorðið UNION leyfir að sameinaðar séu færslur úr mismunandi SELECT aðgerðum í eina niðurstöðu. Mikilvægt er þegar þessi aðferð er notuð að dálkar úr báðum/öllum SELECT aðgerðum séu sömu tegundar og listaðir í sömu röð. Dálkaheiti úr niðurstöðum fyrstu SELECT aðgerðarinnar eru notaðir sem dálkaheiti og síðasta SELECT aðgerðin getur notað INTO OUTFILE. Ef þess er óskað að raða niðurstöðunum þarf að nota sviga utan um SELECT aðgerðirnar.

Eftirfarandi dæmi sameinar fornöfn og millinöfn úr nafnaskrá í einn dálk sem nota mætti til að gera samantekt allra nafna sem til eru:

```
select * from nafnaskra;
+---------+----------+---------------+
| fornafn | millinafn | fodurnafn     |
+---------+----------+---------------+
| Emilía  | Vilborg   | Gestsdóttir   |
| Gerdur  | Birta     | Stefánsdóttir |
| Emilía  | Gudrún    | Jónsdóttir    |
| Gerdur  | Gudrún    | Jónsdóttir    |
+---------+----------+---------------+
4 rows in set (0.01 sec)

select fornafn from nafnaskra union select millinafn from nafnaskra;
+---------+
| fornafn |
+---------+
| Emilía  |
| Gerdur  |
| Vilborg |
| Birta   |
| Gudrún  |
+---------+
5 rows in set (0.00 sec)
```

```
(select all fornafn from nafnaskra)
union
(select millinafn from nafnaskra)
order by fornafn;
+---------+
| fornafn |
+---------+
| Birta   |
| Emilía  |
| Gerdur  |
| Gudrún  |
| Vilborg |
+---------+
5 rows in set (0.01 sec)
```

Málskipan fyrir undir-fyrirspurnir

Undir fyrirspurn er fyrirspurn sem er hreiðruð innan í annarri. Krafan er yfirleitt sú að undir-fyrirspurnin skili aðeins einu gildi í einum dálki. Þó er framkvæmanlegt við sérstakar aðstæður að fá niðurstöðu með einni línu í mörgum dálkum.

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

Í þessu dæmi er sagt að sú hreiðraða sé innri fyrirspurn og aðal fyrirspurnin sé ytri fyrirspurn. Í sjálfu sér er ekkert því til fyrirstöðu að hreiðra margar aðgerðir hverja inni í annarri.

Einn aðal kosturinn við innri-fyrirspurnir er sá að hægt er að framkvæma í einni fyrirspurn einhverja aðgerð sem annars hefði þurft flókna samtvinnun (e. Join) eða fleiri fyrirspurnir hverja á fætur annarri. Þegar SQL staðallinn var hannaður er sagt að þessi möguleiki hafi átt þátt í nafnhlutanum *structured* í heitinu Structured Query Language.

Eftirfarandi dæmi sýnir hvernig hreiðra má margar innri-fyrirspurnir saman, samkvæmt SQL staðlinum sem MySQL fylgir orðrétt í þessu tilfelli.

```
DELETE FROM t1
WHERE s11 > ANY
 (SELECT COUNT(*)FROM t2
 WHERE NOT EXISTS
  (SELECT * FROM t3
   WHERE ROW(5*t2.s1,77)=
    (SELECT 50,11*s1 FROM t4 UNION SELECT 50,77 FROM
     (SELECT * FROM t5) AS t5)));
```

Samanburður með undir-fyrirspurnum

Dæmi:

```
... 'a' = (SELECT column1 FROM t1)
```

Yfirleitt er búist við að innri-fyrirspurn birtist hægra megin við samanburðar virkjann, þó eru til gagnagrunns forrit sem krefjast þess ekki.

Eftirfarandi dæmi er ekki hægt að framkvæma með samtvinnun (e. Join):

```
SELECT column1 FROM t1
       WHERE column1 = (SELECT MAX(column2) FROM t2);

SELECT * FROM t1
       WHERE 2 = (SELECT COUNT(column1) FROM t1);
```

Undir-fyrirspurnir með ANY, IN, og SOME

Málskipan

```
þolandi[60] samanburðarvirki ANY (undir-fyrirspurn)
þolandi IN (undir-fyrirspurn)
þolandi samanburðarvirki SOME (undir-fyrirspurn)
```

Lykilorðin ANY og IN eru jafnvæg og túlkast eins. Sé ANY notað aftan við samanburðar virkjann (lögleg staðsetning) þá er skilað SATT ef samanburðar-gildið er TRUE fyrir eitthvert af þeim gildum sem innri-fyrirspurnin skilar. SOME hefur sömu merkingu:

Skila s1 þar sem s1 er stærri en einhver gildanna í ...

```
SELECT s1 FROM t1 WHERE s1 > ANY (SELECT s1 FROM t2);
```

Skila s1 þar sem s1 er jafnt einhverjum gilanna í ..., eða skila s1 þar sem s1 er í menginu ...

```
SELECT s1 FROM t1 WHERE s1 = ANY (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 IN    (SELECT s1 FROM t2);
```

Undir-fyrirspurnir með ALL

Málskipan:

```
þolandi samanburðar_virki ALL (undir-fyrirspurn)
```

Skilar SATT ef samanburðurinn er sannur (e. True) fyrir allar niðurstöður innri-fyrirspurnarinnar.

Dæmi um jafnvæga undir-fyrirspurn með `<> ALL` sem er samheiti fyrir NOT IN.

```
SELECT s1 FROM t1 WHERE s1 <> ALL (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 NOT IN (SELECT s1 FROM t2);
```

EXISTS og NOT EXISTS

Ef undir-fyrirspurn skilar yfir höfuð einhverjum gildum getum við notað EXISTS og NOT EXISTS til samanburðar. Yfirleitt er gert ráð fyrir því að innri-fyrirspurnin framkvæmi SELECT * ...

Byrjum á að skoða innihaldið í töflunum postnr og nafnaskra:

```
select * from postnr;
+-----+-----------+
| pnr | heiti     |
+-----+-----------+
| 100 | Reykjabær |
| 205 | Kópabær   |
| 206 | Kópahæd   |
| 306 | Akraþorp  |
| 305 | Akraþorp  |
+-----+-----------+
5 rows in set (0.00 sec)

select * from nafnaskra;
+-------+---------+-----------+---------------+------+
| numer | fornafn | millinafn | fodurnafn     | pnr  |
+-------+---------+-----------+---------------+------+
|     1 | Emilía  | Vilborg   | Gestsdóttir   | 100  |
|     2 | Gerdur  | Birta     | Stefánsdóttir | 205  |
|     3 | Emilía  | Gudrún    | Jónsdóttir    | 333  |
|     4 | Gerdur  | Gudrún    | Jónsdóttir    | 444  |
+-------+---------+-----------+---------------+------+
4 rows in set (0.00 sec)
```

Nú viljum við skoða allar einstakar færslur í nafnaskra en því aðeins að undirfyrirspurnin skilar okkur einhverjum niðurstöðum:

[60] Þolandi: Operand

```
select distinct * from nafnaskra
   where exists (select * from postnr where postnr.pnr = nafnaskra.pnr);
+-------+---------+-----------+--------------+------+
| numer | fornafn | millinafn | fodurnafn    | pnr  |
+-------+---------+-----------+--------------+------+
|     1 | Emilía  | Vilborg   | Gestsdóttir  | 100  |
|     2 | Gerdur  | Birta     | Stefánsdóttir | 205  |
+-------+---------+-----------+--------------+------+
2 rows in set (0.00 sec)
```

Nú viljum skoða allar einstakar færslur í nafnaskra en sem ekki finnast í niðurstöðum undir-fyrirspurnar:

```
select distinct * from nafnaskra
   where not exists (select * from postnr
     where postnr.pnr = nafnaskra.pnr);
+-------+---------+-----------+------------+------+
| numer | fornafn | millinafn | fodurnafn  | pnr  |
+-------+---------+-----------+------------+------+
|     3 | Emilía  | Gudrún    | Jónsdóttir | 333  |
|     4 | Gerdur  | Gudrún    | Jónsdóttir | 444  |
+-------+---------+-----------+------------+------+
2 rows in set (0.00 sec)?''
```

Undir-fyrirspurnir í FROM hluta

Löglegt er að setja innri-fyrirspurnir í FROM hluta SELECT aðgerða en þeim verður að gefa viðurnefni því allir hlutir í FROM hluta verða að bera heiti.

```
select * from (select fornafn, fodurnafn from nafnaskra) as nofn;
```

Þessi tækni gerir kleift að vinna lausnir eins og eftirfarandi dæmi sýnir. Fyrst skoðum við töfluna fyrir morgunverðar álegg, sem er í mismunandi verðflokkum:

```
select * from tafla1;
+------+--------+-------+
| num  | txt    | verd  |
+------+--------+-------+
|    5 | sulta  |  56.7 |
|    6 | ostur  |    99 |
|    7 | skinka | 109.5 |
|    8 | sulta  |    59 |
|    9 | ostur  |  89.9 |
|   10 | skinka |   129 |
+------+--------+-------+
6 rows in set (0.00 sec)
```

Því næst tökum við meðaltal af verðinu eftir flokkum:

```
select avg(verd) from tafla1 group by txt;
+-----------------+
| avg(verd)       |
+-----------------+
| 94.450000762939 |
|          119.25 |
|  57.85000038147 |
+-----------------+
3 rows in set (0.01 sec)
```

Byggjum að lokum á síðasta dæmi, skellum því í FROM hluta og leggjum saman allt saman:

```
select sum(summa) from (select avg(verd) as summa
  from tafla1 group by txt) as tata;
+-----------------+
| sum(summa)      |
```

```
+-----------------+
| 271.55000114441 |
+-----------------+
1 row in set (0.00 sec)
```

TRUNCATE

TRUNCATE TABLE tæmir töflu, algjörlega.

```
TRUNCATE TABLE töflu_heiti
```

UPDATE

Aðgerðin UPDATE dagréttar[61] (e. Updates) gögn sem þegar eru til í töflunni, skilgreina má að dagrétt sé í mörgum töflum samtímis. SET klausan skilgreinir í hvaða dálka skuli dagrétta og WHERE klausan skilgreinir þau skilyrði sem geta tilgreint í hvaða línum skuli dagrétt. Sé ekkert sett í WHERE, eru allar línur dagréttaðar.

Sé lykilorðið IGNORE látið fylgja með, er haldið áfram að dagrétta þó ein lína t.d. innihaldi tvítekið lykilgildi (e. Duplicate key value). Með ORDER BY má tilgreina þá röð sem framkvæmdin er unnin eftir og LIMIT getur tilgreint hversu margar línur dagréttast.

```
UPDATE [IGNORE] töflu_heiti [, töflu_heiti ...]
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_definition]
    [ORDER BY ... ]
    [LIMIT línu_fjöldi]
```

Byrjum á að smíða töfluna tafla11 úr innihaldi töflunnar tafla1, skoðum svo innihaldið:

```
create table tafla11 (select * from tafla1);
```

```
select * from tafla11;
+------+--------+-------+
| num  | txt    | verd  |
+------+--------+-------+
|    5 | sulta  |  56.7 |
|    6 | ostur  |    99 |
|    7 | skinka | 109.5 |
|    8 | sulta  |    59 |
|    9 | ostur  |  89.9 |
|   10 | skinka |   129 |
+------+--------+-------+
6 rows in set (0.00 sec)
```

Hér er skilgreint að dálkurinn num sé dagréttaður á gildið 1. Þar sem engin skilyrði eru tilgreind mun aðgerðin verka á öll gildi dálksins:

```
update tafla11 set num=1;
```

```
select * from tafla11;
+------+--------+-------+
| num  | txt    | verd  |
+------+--------+-------+
```

[61] Tölvuorðasafnið hefur mælt með sögninni að dagrétta sem merkingu á ensku sögninni „to update.“ Nýlega hefur verið mælt með sögninni að uppnýja í sömu merkingu. Sumir nota ranglega sögnina að uppfæra sem Tölvuorðasafnið bendir á sem þýðingu ensku sagnarinnar „to upgrade.“ Nánar um þetta á slóðinni http://www.ismal.hi.is/to/Vikuord.htm eða http://www.sky.is/?q=ordamore&id=54.

```
|    1 | sulta  |  56.7 |
|    1 | ostur  |    99 |
|    1 | skinka | 109.5 |
|    1 | sulta  |    59 |
|    1 | ostur  |  89.9 |
|    1 | skinka |   129 |
+------+--------+-------+
6 rows in set (0.00 sec)
```

Hér er tilgreint að gildið í dálkinum num skuli sett á 2 í þeirri línu sem verd er hærra en 60. Þar sem við tilgreinum að fyrst skuli raða dálkinum og aðeins dagrétta eina línu þá gætir áhrifanna samanber:

```
update tafla11 set num=2 where verd > 60 order by verd limit 1;

select * from tafla11 order by verd;
+------+--------+-------+
| num  | txt    | verd  |
+------+--------+-------+
|    1 | sulta  |  56.7 |
|    1 | sulta  |    59 |
|    2 | ostur  |  89.9 |
|    1 | ostur  |    99 |
|    1 | skinka | 109.5 |
|    1 | skinka |   129 |
+------+--------+-------+
6 rows in set (0.01 sec)
```

Hér er dagréttað í tveim töflum og verður að gæta þess að nefna dálka sem koma fyrir í öllum töflum sem tilgreindar eru. Hér hefði mátt bæta við OR skilyrði til að tilgreina hvar í tafla22 hefði átt að dagrétta:

```
update tafla11, tafla22 set txt='kjöt', texti='fiskur'
   where tafla11.num >=6;

select * from tafla11;
+------+-------+-------+
| num  | txt   | verd  |
+------+-------+-------+
|    5 | sulta |  56.7 |
|    6 | kjöt  |    99 |
|    7 | kjöt  | 109.5 |
|    8 | kjöt  |    59 |
|    9 | kjöt  |  89.9 |
|   10 | kjöt  |   129 |
+------+-------+-------+
6 rows in set (0.00 sec)

select * from tafla22;
+------+--------+
| num  | texti  |
+------+--------+
|    5 | fiskur |
|    6 | fiskur |
|    8 | fiskur |
+------+--------+
3 rows in set (0.00 sec)
```

Að lokum eitt dæmi sem sýnir hvernig draga má út ákveðið gildi, breyta því og senda það inn aftur:

```
update tafla22 set num = num+1 where num=8;

select * from tafla22;
+------+--------+
| num  | texti  |
```

```
+------+--------+
|    5 | fiskur |
|    6 | fiskur |
|    9 | fiskur |
+------+--------+
3 rows in set (0.00 sec)
```

Gagna skilgreiningar (DDL)

ALTER DATABASE

```
ALTER DATABASE db_name
    alter_specification [, alter_specification] ...

alter_specification:
    [DEFAULT] CHARACTER SET charset_name
  | [DEFAULT] COLLATE collation_name
```

Með ALTER DATABASE má breyta máli gagnagrunnsins þ.e. hvert sé sjálfgefið stafamengi grunnsins og hver sé COLLATION fyrir hann.

ALTER TABLE

```
ALTER  TABLE töflu_heiti
    alter_specification [, alter_specification] ...

alter_specification:
    ADD [COLUMN] column_definition [FIRST | AFTER col_name ]
  | ADD [COLUMN] (column_definition,...)
  | ADD INDEX [index_name] [index_type] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [index_name] [index_type] (index_col_name,...)
  | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
  | CHANGE [COLUMN] old_col_name column_definition
        [FIRST|AFTER col_name]
  | MODIFY [COLUMN] column_definition [FIRST | AFTER col_name]
  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP INDEX index_name
  | RENAME [TO] new_töflu_heiti
```

Að öðrum mikilvægum setningum ólöstuðum þá er mikilvægt að geta breytt töflu eftir að hún hefur verið smíðuð, sem er nokkuð algeng nauðsyn. ALTER TABLE leyfir einmitt að einstökum dálkum töflunnar sé eytt, nýjum bætt við og röð þeirra breytt:

Málskipanin eins og hún er listuð hér að framan er talsvert niðurskorin því hér verður einblínt á þær aðgerðir sem eru algengastar, fyrir fleiri möguleika er bent á Viðauka A.

Tökum dæmi af töflunni sjoppur sem lýst er hér á eftir:

```
describe sjoppur;
+---------+------------------+------+-----+---------+----------------+
| Field   | Type             | Null | Key | Default | Extra          |
+---------+------------------+------+-----+---------+----------------+
| numer   | int(10) unsigned |      | PRI | NULL    | auto_increment |
| kennit  | varchar(10)      |      |     |         |                |
| nafn    | varchar(31)      | YES  |     | NULL    |                |
| heima   | varchar(25)      | YES  |     | NULL    |                |
| pnr     | char(3)          | YES  |     | NULL    |                |
| eigandi | varchar(10)      | YES  | MUL | NULL    |                |
```

```
+---------+-----------------+------+-----+---------+----------------+
6 rows in set (0.05 sec)
```

Nú hendum við dálkinum „eigandi“ og öll gildin í þeim dálki hverfa.

```
alter table sjoppur drop column eigandi;
```

Nú bætum við tveim nýjum dálkum við töfluna, „mynd“ sem gæti geymt mynd af sjoppunni og „eig“ sem í rauninni er sami dálkur og eigandi var áður nema tómur. Strax á eftir lengjum við geymslugetuna á „heima“ um 5 stafi, taktu eftir að við þurfum að tvítaka nafn dálksins.

```
alter table sjoppur add (mynd blob null, eig varchar (10) null);
alter table sjoppur change heima heima varchar(30) not null;
```

Lýsum nú töflunni aftur

```
describe sjoppur;
+--------+------------------+------+-----+---------+----------------+
| Field  | Type             | Null | Key | Default | Extra          |
+--------+------------------+------+-----+---------+----------------+
| numer  | int(10) unsigned |      | PRI | NULL    | auto_increment |
| kennit | varchar(10)      |      |     |         |                |
| nafn   | varchar(31)      | YES  |     | NULL    |                |
| heima  | varchar(25)      | YES  |     | NULL    |                |
| pnr    | char(3)          | YES  |     | NULL    |                |
| mynd   | blob             | YES  |     | NULL    |                |
| eig    | varchar(10)      | YES  |     | NULL    |                |
+--------+------------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

Nú reynum við að gera „kennit“ dálkinn einkvæman en rekumst á að það er ekki hægt, því eitthvað verður af endurteknum gildum í dálkinum:

```
alter table sjoppur add unique(kennit);
ERROR 1062 (23000): Tvikvaem innsetning '' a lykli 2[62]
```

Þegar við skoðum dálkinn sjáum við fullt af tómum strengjum (‘’) í kennit.

```
select kennit, nafn from sjoppur order by kennit;
+------------+-------------+
| kennit     | nafn        |
+------------+-------------+
|            | Bitinn      |
|            | Dollan      |
|            | Bjólan      |
|            | Aurinn      |
|            | Kot         |
| 1234560001 | Palli Palla |
| 1234560002 | Jói Jóa     |
| 1234560003 | Sigga Sigg  |
| 1234560004 | Bitinn      |
| 1234560006 | Gjugg       |
| 1234567890 | Pallasjoppa |
```

[62] Höfundur rakst á að forritið comp_err sem fylgir með MySQL leyfir að skráin [tungum.mappa]/errmsg.txt sé endurþýdd sem [tungum.mappa]/errmsg.sys svo það varð að prófa að setja upp möppuna [icelandic]/errmsg.sys fyrir miðlarann, sem birtist í einstaka úttaki hálfmótað. Á www.isbok.is verður síðar sett fullmótuð skrá með leiðbeiningum um notkun.

```
| 1234567891 | Jóasjoppa   |
+------------+-------------+
12 rows in set (0.00 sec)
```

Nú smíðum við notenda-breytu með strengjagildi sem líkir eftir því kennitölu mynstri sem við erum að nota.

```
set @kk = '1234560011';
```

Nú notum við breytuna og MySQL getuna til að skipta kraftmikið (e. Dynamic) á milli talna og strengja, til að innskrá nýjar einkvæmar kennitölur. Áhrifin verða þau að öftustu tölurnar í strengnum hækka úr 11 í 12, úr 12 í 13 eftir atvikum:

```
update sjoppur set kennit = (@kk:=@kk+1) where kennit='';

select kennit, nafn from sjoppur order by kennit;
+------------+-------------+
| kennit     | nafn        |
+------------+-------------+
| 1234560001 | Palli Palla |
| 1234560002 | Jói Jóa     |
| 1234560003 | Sigga Sigg  |
| 1234560004 | Bitinn      |
| 1234560006 | Gjugg       |
| 1234560012 | Kot         |
| 1234560013 | Aurinn      |
| 1234560014 | Bjólan      |
| 1234560015 | Dollan      |
| 1234560016 | Bitinn      |
| 1234567890 | Pallasjoppa |
| 1234567891 | Jóasjoppa   |
+------------+-------------+
12 rows in set (0.00 sec)
```

Nú tekst okkur auðveldlega að skilgreina „kennit“ sem einkvæman dálk.

```
alter table sjoppur add unique(kennit);
```

Að lokum framkvæmum við breytingu á „eigandi“ sem líklega hefði verið skynsamlegri kostur:

```
alter table sjoppur change eig eigandi varchar(10);
```

Að lokum kíkjum við á hvernig dálkur með sjálfkrafa tölusetningu bætist við töflu:

```
describe tafla1;
+-------+----------------+------+-----+---------+-------+
| Field | Type           | Null | Key | Default | Extra |
+-------+----------------+------+-----+---------+-------+
| num   | int(11)        | YES  |     | NULL    |       |
| txt   | char(10)       | YES  |     | NULL    |       |
| verd  | float unsigned | YES  |     | NULL    |       |
+-------+----------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Taflan hefur þrjá dálka en við viljum bæta við dálki fremst sem sjái um sjálfkrafa tölusetningu hverrar færslu og sé aðallykill:

```
alter table tafla1
    add column sjalfnumer int unsigned auto_increment primary key first;
```

Þegar við skoðum allar færslur töflunnar sjáum við að dálkurinn fékk sjálfkrafa öll nauðsynleg frumgildi um leið og hann var stofnaður.

```
select * from tafla1;
+------------+------+--------+-------+
| sjalfnumer | num  | txt    | verd  |
+------------+------+--------+-------+
|          1 |    5 | sulta  |  56.7 |
|          2 |    6 | ostur  |    99 |
|          3 |    7 | skinka | 109.5 |
...klippt
+------------+------+--------+-------+
```

CREATE DATABASE

```
CREATE DATABASE [IF NOT EXISTS] db_name
    [create_specification [, create_specification] ...]

create_specification:
    [DEFAULT] CHARACTER SET charset_name
  | [DEFAULT] COLLATE collation_name
```

Aðgerðin CREATE DATABASE smíðar nýjan gagnagrunn og má tiltaka í hvaða stafamengi hann skuli vera og hvaða COLLATION. Þessi aðgerð smíðar nýja möppu með nafni hins nýja gagnagrunns undir „data“ möppunni sem MySQL notar undir gagnagrunna.

CREATE INDEX

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name [index_type]
    ON töflu_heiti (index_col_name,...)

index_col_name:
    col_name [(length)] [ASC | DESC]
```

Venjulega er atriðaskrá (e. Index) búin til um leið og taflan en með þessari aðgerð má bæta við nýjum atriðaskrám fyrir töflur og má tiltaka að atriðaskráin spanni marga dálka töflunnar. Atriðaskrár geta gert leitir innan taflna mun hraðvirkari en ella, sérstaklega þegar töflurnar verða stórar.

CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] töflu_heiti
    [(smíða_skilgreining,...)]
```

Eða:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] töflu_heiti
    [[(] LIKE eldri_tafla [)]] | [lögleg_select_aðgerð];
```

Málskipanin eins og hún birtist hér hefur verið skorin niður í allra algengustu notkun CREATE TABLE, full málskipan er listuð í Viðauka A.

Aðal málskipan fyrir CREATE TABLE er þannig að skilgreina má alla dálka hennar við smíði töflunnar, TEMPORARY tilgreinir að taflan sé aðeins til á meðan biðlarinn sem smíðar hana er tengdur, IF NOT EXISTS er hentugt ef töflugerðin er framkvæmd í textaskrá sem lesin er inn og sé taflan til fyrir þá er ekki hætt við alla aðgerðina heldur er afgangur aðgerðanna samt framkvæmdur:

Smíða má töflu þannig að tilgreint er að hún líkist annarri sem til er fyrir, eða að hún sé smíðuð utan um niðurstöður úr SELECT aðgerð:

Smíða_skilgreining er þannig:

```
    dálka_skilgreining
  | PRIMARY KEY (index_dálka_heiti,...)
```

```
  | INDEX [index_name] (index_ dálka_heiti,...)
  | UNIQUE (index_ dálka_heiti,...)
```

Dálka_skilgreining er þannig:

```
    dálka_heiti tegund [NOT NULL | NULL] [DEFAULT default_value]
        [AUTO_INCREMENT] [[PRIMARY] KEY] [COMMENT 'string']
```

Tegund getur verið:

```
  | INT[(length)] [UNSIGNED] [ZEROFILL]
      // TINYINT SMALLINT MEDIUMINT BIGINT eru skilgreindar eins
  | DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
     // REAL FLOAT DECIMAL NUMERIC eru skilgreindar eins
  | DATE
  | TIME
  | TIMESTAMP
  | DATETIME
  | CHAR(length) [BINARY | ASCII | UNICODE]
  | VARCHAR(length) [BINARY]
  | TINYBLOB
  | BLOB
  | MEDIUMBLOB
  | LONGBLOB
  | TINYTEXT
  | TEXT
  | MEDIUMTEXT
  | LONGTEXT
  | ENUM(value1,value2,value3,...)
  | SET(value1,value2,value3,...)
```

Best er að útskýra töfluskilgreiningar með dæmum: Fyrsta dæmið býr til töfluna „sjoppur“ sem er af töflutegundinni MyISAM í stafamenginu latin1, sem hvorutveggja er sjálfgefið í staðlaðri uppsetningu MySQL.

Fyrsti dálkurinn „numer“ inniheldur heiltölur (10 stafa birtingarbreidd) og er óformerktur svo hann innihaldi engin neikvæð gild og tvöfaldar það hámarks gildið í jákvæðum tölum, AUTO_INCREMENT skilgreinir að í hvert sinn sem ný færsla bætist við töfluna hækkar gildið í numer um einn. Dálkurinn „kennit“ er 10 stafir að lengd og má ekki vera NULL því enginn sjoppa getur verið án kennitölu, „nafn“ getur hins vegar verið NULL ef við vitum ekki hvað sjoppan heitir en „heima“ getur ekki verið NULL því sjoppan hlýtur að eiga heima einhversstaðar. Dálkurinn „pnr“ er CHAR því póstnúmer er alltaf þrír stafir, aldrei minna og aldrei meira. Við getum geymt mynd af sjoppunni og við getum skráð kennitölu eiganda í „eigandi“ en við höfum það NULL því e.t.v. er eigandi óþekktur.

Að lokum er skilgreint að dálkurinn „numer“ sé aðallykill en AUTO_INCREMENT dálkar verða að vera aðallyklar og við skilgreinum að „kennit“ sé einkvæm því tvær sjoppur geta ekki haft sömu kennitöluna.

Taktu vel eftir að komma (,) skilur á milli dálkaskilgreininga og að semíkomma (;) endar aðgerðina.

```
CREATE TABLE sjoppur (
  numer int(10) unsigned NOT NULL auto_increment,
  kennit varchar(10) NOT NULL default '',
  nafn varchar(31) default NULL,
  heima varchar(30) NOT NULL default '',
  pnr char(3) default NULL,
  mynd blob,
  eigandi varchar(10) default NULL,
  PRIMARY KEY  (numer),
  UNIQUE KEY kennit (kennit)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Næsta tafla er skilgreind með athugasemd (e. Comment) fyrir dálkinn sinn, því næst tökum við afrit af töflunni með „mysqldump“ forritinu (úr skipana kvaðningu[63]). Þó athugasemdir við dálkana sjáist ekki í mysql biðlaranum þá fylgja þær töfluskilgreiningunni hjá miðlaranum.

Tafla smíðuð í mysql:

```
create table henda(
  numer int comment 'heiltala'
);
```

Afrit tekið af töflunni með mysqldump í kvaðningar glugga:

```
C:\>mysqldump -u root -p malskipan henda > d:/henda.sql
Enter password: *******
```

Hér sést að athugasemdin fylgir dálkinum í töflu-afritinu:

```
... klippt
-- Table structure for table `henda`
--

DROP TABLE IF EXISTS henda;
CREATE TABLE henda (
  numer int(11) default NULL COMMENT 'heiltala'
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

 ... klippt
```

Skráin „henda.sql“ mætti nota með „mysqlimport“ til að endursmíða og lýðfylla töfluna.

Að lokum eru smíðaðar tvær töflur, önnur eftir eldri töflu og skal hún hverfa þegar biðlarinn lokast, hin utan um niðurstöður úr SELECT aðgerð.

Byrjum á að skoða töfluna „nafn“ og smíðum svo aðra eins töflu:

```
select * from nafn;
+------------+-------------+------------+
| kt         | nafn        | f_dag      |
+------------+-------------+------------+
| 1234567890 | Pallasjoppa | 1975-00-00 |
| 1234567891 | Jóasjoppa   | 1975-00-00 |
| 1234560002 | Jói Jóa     | 1975-00-00 |
| 1234560003 | Sigga Sigg  | 1975-00-00 |
| 1234560001 | Palli Palla | 1975-00-00 |
| 1234560004 | Bitinn      | 1975-00-00 |
| 1234560005 | Gjugg       | 1975-00-00 |
| 1234560006 | Gaur        | 1975-00-00 |
+------------+-------------+------------+
8 rows in set (0.00 sec)

create temporary table bid_nafn like nafn;

select * from bid_nafn;
Empty set (0.00 sec)
```

[63] Skipana kvaðning eða kvaðning er það sama og áður var nefnt DOS PROMPT á PC/Intel tölvum með DOS stýrikerfi. Texta kvaðning, eða texta skel, er algeng enn í dag á LINUX tölvum frekar en grafíska skelin X sem minnir um margt á grafísk umhverfi í Windows og Macintosh.

Þó taflan líti út eins og „nafn“ þá er hún sköpuð tóm. Næsta aðgerð skýtur inn fyrstu þrem færslunum úr „nafn“ yfir í „bid_nafn“

```
insert into bid_nafn select * from nafn limit 3;
```

```
select * from bid_nafn;
+------------+-------------+------------+
| kt         | nafn        | f_dag      |
+------------+-------------+------------+
| 1234567890 | Pallasjoppa | 1975-00-00 |
| 1234567891 | Jóasjoppa   | 1975-00-00 |
| 1234560002 | Jói Jóa     | 1975-00-00 |
+------------+-------------+------------+
3 rows in set (0.00 sec)
```

Nú skulum við líta á töfluna „nafnaskra“ sem inniheldur fjórar færslur.

```
select * from nafnaskra;
+-------+---------+-----------+---------------+------+
| numer | fornafn | millinafn | fodurnafn     | pnr  |
+-------+---------+-----------+---------------+------+
|     1 | Emilía  | Vilborg   | Gestsdóttir   | 100  |
|     2 | Gerdur  | Birta     | Stefánsdóttir | 205  |
|     3 | Emilía  | Gudrún    | Jónsdóttir    | 333  |
|     4 | Gerdur  | Gudrún    | Jónsdóttir    | 444  |
+-------+---------+-----------+---------------+------+
4 rows in set (0.04 sec)
```

Því næst gerum við fyrirspurn sem sækir allar færslur úr dálkunum „fornafn“ og „millinafn“, smíðum jafnframt nýja töflu utan um gildin (við hefðum getað haft hana TEMPORARY).

Þegar við gerum fyrirspurn á hina nýju töflu sjáum við að hún hefur aðeins tvo dálka, í samræmi við SELECT aðgerðina sem notuð var til að smíða töfluna.

```
create table ny_nafnaskra select fornafn, millinafn from nafnaskra;
```

```
select * from ny_nafnaskra;
+---------+-----------+
| fornafn | millinafn |
+---------+-----------+
| Emilía  | Vilborg   |
| Gerdur  | Birta     |
| Emilía  | Gudrún    |
| Gerdur  | Gudrún    |
+---------+-----------+
4 rows in set (0.00 sec)
```

DROP DATABASE

```
DROP DATABASE [IF EXISTS] gagnagrunns_nafn
```

Þessi aðgerð hendir öllum töflum gagnagrunnsins og eyðir möppunni sem hann geymdist í. Þessi skipun er óafturkallanleg, og því væri ekki úr vegi að nota „mysqldump“ til að taka öryggisafrit af honum fyrst.

DROP INDEX

```
DROP INDEX index_heiti ON töflu_heiti
```

Hendir atriðaskrá úr töflunni „töflu_heiti“.

DROP TABLE

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    töflu_heiti [, töflu_heiti] ...
```

Aðgerðin DROP TABLE hendir (fellir) töflu og eyðir henni. Þessi aðgerð er ekki afturkölluð. Lykilorðið TEMPORARY eyðir eingöngu töflum sem eru TEMPORARY og IF EXISTS er hentugt í runu skrám (e. Batch files) sem eru lesnar inn því sé þessi möguleiki notaður nær runu skráin að halda áfram keyrslu við næstu SQL aðgerðir þó taflan væri til fyrir. Eyða má mörgum töflum ef því er að skipta.

RENAME TABLE

```
RENAME TABLE töflu_heiti TO nýtt_töflu_heiti
    [, töflu_heiti2 TO nýtt_töflu_heiti2] ...
```

Með þessari aðgerð má skipta um nafn á töflu sem fyrir er án þess að glata gögnum hennar. Þetta má einnig gera með „ALTER TABLE töfluheiti RENAME TO ...“.

Aðgerðin vinnur frá hægri til vinstri, þannig má vel framkvæma eftirfarandi aðgerð án hættu á gagnatapi, svo fremi að taflan „tmp_tafla“ sé til. Ef MySQL fær villuboð á meðan á endurnefningu stendur veltur aðgerðin til baka.

```
RENAME TABLE gamla_tafla TO tmp_tafla,
             new_table TO gamla_tafla,
             tmp_tafla TO new_table;
```

Tóla- og tækjaaðgerðir

DESCRIBE

```
{DESCRIBE | DESC} töflu_heiti [col_name | wild]
```

Aðgerðin DESCRIBE birtir upplýsingar um dálka í tilgreindri töflu. Aðgerðin er ORACLE samhæfni og í raun samheiti aðgerðarinnar SHOW COLUMNS FROM.

Allar þessar þrjár aðgerðir skila sömu niðurstöðu:

```
describe sjoppur;
desc sjoppur;
show columns from sjoppur;
+---------+------------------+------+-----+---------+----------------+
| Field   | Type             | Null | Key | Default | Extra          |
+---------+------------------+------+-----+---------+----------------+
| numer   | int(10) unsigned |      | PRI | NULL    | auto_increment |
| kennit  | varchar(10)      |      | UNI |         |                |
| nafn    | varchar(31)      | YES  |     | NULL    |                |
| heima   | varchar(30)      |      |     |         |                |
| pnr     | char(3)          | YES  |     | NULL    |                |
| mynd    | blob             | YES  |     | NULL    |                |
| eigandi | varchar(10)      | YES  |     | NULL    |                |
+---------+------------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

USE

```
USE gagnagrunns_heiti
```

Þessi aðgerð tiltekur fyrir MySQL hvaða gagnagrunn biðlarinn ætlar sér að nota sem sjálfgefinn grunn. Grunnurinn verður sjálfgefinn þar til tengingin rofnar eða annar grunnur er valinn. Skipunin er fyrir Sybase samhæfni.

Eftirfarandi dæmi sýnir að sækja má upplýsingar í aðra grunna (á miðlaranum ef notandinn hefur aðgang að þeim). Virki gagnagrunnurinn er „malskipan“ og því sjást allar töflur í þeim gagnagrunni án þess að taka þurfi það fram sérstaklega.

```
use malskipan;
Database changed

select nafn from sjoppur limit 2;
+-------------+
| nafn        |
+-------------+
| Pallasjoppa |
| Jóasjoppa   |
+-------------+
2 rows in set (0.00 sec)
```

Ný skoðum við innihald úr töflu í öðrum gagnagrunni:

```
select * from test.heitidaga_is as hh
  order by nr desc limit 1;
+----+------------+
| nr | heiti      |
+----+------------+
|  6 | Sunnudagur |
+----+------------+
1 row in set (0.00 sec)
```

Færslur og læsingar

START TRANSACTION, COMMIT og ROLLBACK

Sjálfgefið er að „autocommit“ hamur sé virkur í MySQL, sem merkir að um leið og t.d. dagrétting er framkvæmd á töflu tekur hún gildi og er vistuð á diski. Sé töflutegundin hins vegar „InnoDB“ eða „BDB“ má gera þennan ham óvirkan með:

```
SET AUTOCOMMIT=0;
```

Sé autocommit orðið óvirkt, verður að nota COMMIT eða ROLLBACK til að klára SQL aðgerðina/aðgerðirnar. COMMIT myndi þá staðfesta færsluna en ROLLBACK myndi velta til baka. Sé notuð aðgerðin START TRANSACTION verður autocommit óvirkt þar til annað hvort COMMIT eða ROLLBACK er framkvæmt.:

```
START TRANSACTION;
SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summary=@A WHERE type=1;
COMMIT;
```

Aðgerðir sem ekki er hægt að velta til baka

Ekki er hægt að velta til baka með ROLLBACK þeim aðgerðum sem skilgreina gögn, s.s. CREATE og DROP fyrir gagnagrunna eða CREATE, DROP og ALTER fyrir töflur. Því þyrfti að hanna allar færslu-aðgerðir þannig að gagnaskilgreiningar séu færðar sérstaklega.

Aðgerðir sem óbeint framkvæma COMMIT

Eftirfarandi aðgerðir stöðva færslu (e. Transaction) líkt og ef COMMIT væri framkvæmt, svo varist að nota þær innan færslu.

ALTER TABLE	BEGIN	CREATE INDEX
DROP DATABASE	DROP INDEX	DROP TABLE
LOAD MASTER DATA	LOCK TABLES	RENAME TABLE
SET AUTOCOMMIT=1	START TRANSACTION	TRUNCATE TABLE

Aðgerðin UNLOCK TABLES stöðvar einnig færslu ef einhverjar töflur eru læstar .

LOCK TABLES og UNLOCK TABLES

```
LOCK TABLES
    töflu_heiti [AS alias] {READ [LOCAL] | [LOW_PRIORITY] WRITE}
    [, töflu_heiti [AS alias] {READ [LOCAL] | [LOW_PRIORITY] WRITE}]
...
UNLOCK TABLES
```

Þessi aðgerð læsir þeim töflum sem valdar eru, fyrir þráðinn sem biðlarinn notar. Enginn annar þráður getur lesið úr eða skrifað í töflurnar á meðan og verða því að bíða þar til UNLOCK er framkvæmt, nýjum töflum er læst eða tengingin rofnar.

Tilgreina þarf annaðhvort READ eða WRITE eftir því hver tilgangurinn er með læsingunni. Sé töflunni gefið viðurnefni (e. Alias) verður að nota það í þeim aðgerðum sem vísa á töfluna:

```
lock table nafnaskra read;

select * from nafnaskra limit 1;
+-------+---------+-----------+-------------+------+
| numer | fornafn | millinafn | fodurnafn   | pnr  |
+-------+---------+-----------+-------------+------+
|     1 | Emilía  | Vilborg   | Gestsdóttir | 100  |
+-------+---------+-----------+-------------+------+
1 row in set (0.00 sec)

unlock table;

lock table nafnaskra as nn read;
```

Hér koma villuboð því vísa verður í töfluna sem „nn“

```
select * from nafnaskra limit 1;
ERROR 1100 (HY000): Tafla 'nafnaskra' var ekki laest med LOCK TABLES

select * from nafnaskra as nn limit 1;
+-------+---------+-----------+-------------+------+
| numer | fornafn | millinafn | fodurnafn   | pnr  |
+-------+---------+-----------+-------------+------+
|     1 | Emilía  | Vilborg   | Gestsdóttir | 100  |
+-------+---------+-----------+-------------+------+
1 row in set (0.00 sec)
```

Næstu tvær aðgerðir sýna að sú fyrri beið í 13 sekúndur áður en aðgerðin kláraðist því annar biðlari hafði læst töflunni (í öðrum glugga) með „LOCK TABLE *nafnaskra* WRITE.“ Þó notað hefði verið READ hefði bið einnig átt sér stað.

```
insert into nafnaskra (fornafn,millinafn) values('sigga','vigga');
Query OK, 1 row affected (13.40 sec)

insert into nafnaskra (fornafn,millinafn) values('jói','spói');
Query OK, 1 row affected (0.00 sec)
```

Gagnagrunns stjórnun

Notenda stjórnun

DROP USER

```
DROP USER notandi
```

Þessi aðgerð eyðir út notanda úr grant töflum MySQL, svo fremi að notandinn hafi engin réttindi (e. Privileges). Nefna þarf notandann á sniðinu „notandi@hýsiltölva/lén“ svipað og í GRANT og REVOKE aðgerðum.

Til að geta framkvæmt aðgerðina með árangri þarf að vinna á eftirfarandi máta:

1. Nota SHOW GRANTS til að sjá hvaða réttindi notandinn hefur.
2. Nota REVOKE til að fjarlægja þau réttindi sem birtast fyrir notandann.
3. Eyða notandanum með DROP USER aðgerð.

GRANT og REVOKE

```
GRANT réttindi [(dálka_listi)] [, réttindi [(dálka_listi)]] ...
    ON {töflu_heiti | * | *.* | gagnagrunns_heiti.*}
    TO notandi [IDENTIFIED BY [PASSWORD] 'lykilorð']
        [, notandi [IDENTIFIED BY [PASSWORD] 'lykilorð']] ...
    [WITH [GRANT OPTION]

REVOKE réttindi [(dálka_listi)] [, réttindi [(dálka_listi)]] ...
    ON {töflu_heiti | * | *.* | gagnagrunns_heiti.*}
    FROM notandi [, notandi] ...

REVOKE ALL PRIVILEGES, GRANT OPTION FROM notandi [, user] ...
```

Aðgerðirnar GRANT og REVOKE gera kerfisstjóranum kleyft að úthluta og stjórna réttindum á notkun gagnagrunna sem miðlarinn geymir. Öll réttindin eru geymd í „mysql“ gagnagrunninum í „data“ möppunni.

Einnig má stýra réttindum og/eða skoða þau með hefðbundnu SQL aðgerðunum SELECT, INSERT, UPDATE og DELETE. Töflurnar sem aðgangsstýringar geymast í eru „user, host og db.“ Einnig eru viðbótar stýringar geymdar í töflunum; „tables_priv og columns_priv.“

Notendastýring krefst þess að vera gefin á sniðinu notanda-á-léni þ.e. notandi@lén.

Þannig er aðgerðin GRANT ALL ON test.* to elli@isbok.is, notuð til að úthluta notandanum elli sem þekktur er á hýslinum/léninu isbok.is öll almenn réttindi á test gagnagrunninum. Nota má „REVOKE ALL ON test.* FROM elli@isbok.is“ til að taka réttindin af honum aftur.

Sjáum þetta í kóta:

Dæmi a: notandi án lykilorðs

Skilgreinum notanda fyrir gagnagrunninn „test“ án lykilorðs (sem við ættum aldrei að gera):

```
grant all on test.* to elli@isbok.is;
Query OK, 0 rows affected (0.02 sec)
```

Skoðum nú allar aðgangsstýringar fyrir framangreindan notanda:

```
show grants for elli@isbok.is ;
+-----------------------------------------------------+
| Grants for elli@isbok.is                            |
+-----------------------------------------------------+
| GRANT USAGE ON *.* TO 'elli'@'isbok.is'             |
| GRANT ALL PRIVILEGES ON `test`.* TO 'elli'@'isbok.is' |
+-----------------------------------------------------+
2 rows in set (0.00 sec)
```

Skoðum alla skilgreinda notendur sem hafa „li“ í nafni sínu:

```
select User, Host from mysql.user where user like '%li%';
+-------+-----------+
| User  | Host      |
+-------+-----------+
| elias | *         |
| elli  | isbok.is  |
| elias | localhost |
+-------+-----------+
3 rows in set (0.00 sec)
```

Tökum nú öll réttindi af notandanum og eyðum honum svo úr notendagrunninum:

```
revoke all on test.* from elli@isbok.is;

drop user elli@isbok.is;

select User, Host from mysql.user where user like '%li%';
+-------+-----------+
| User  | Host      |
+-------+-----------+
| elias | *         |
| elias | localhost |
+-------+-----------+
2 rows in set (0.00 sec)
```

Dæmi b: notandi með lykilorði

Skilgreinum nú nýjan notanda fyrir gagnagrunninn „malskipan“ og að hann þurfi tilgreint lykilorð til þess að tengjast:

```
grant all privileges on malskipan.*
    to 'tomas'@'hvergi.is' identified by 'horfinn';
```

Birtum nú allar aðgangsstýringar notandans, tökum leyfin af honum og eyðum honum síðan:

```
show grants for tomas@hvergi.is;
+-------------------------------------------------------------------+
| Grants for tomas@hvergi.is
+-------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'tomas'@'hvergi.is' IDENTIFIED BY
|       PASSWORD '63f336a60fbb4aef'
| GRANT ALL PRIVILEGES ON `malskipan`.* TO 'tomas'@'hvergi.is'
+-------------------------------------------------------------------+
2 rows in set (0.02 sec)

revoke all privileges on malskipan.* from 'tomas'@'hvergi.is';

drop user tomas@hvergi.is;
```

Úthluta má réttindum á fjórum sviðum:

Global level (víðvært svið): Víðvær-réttindi ná til allra grunna á miðlaranum og eru geymd í töflunni mysql.user. Ritháttur: GRANT ALL ON *.* og REVOKE ALL ON *.*.

Database level (gagnagrunns svið): Gagnagrunns-réttindi ná til þess gagnagrunns sem nefndur er og geymast í töflunum mysql.db og mysql.host. Ritháttur: GRANT ALL ON gagnagrunns_heiti.* og REVOKE ALL ON gagnagrunns_heiti.*.

Table level (töflu svið): Töflu-réttindi ná til þeirra taflna sem nefndar eru og geymast í töflunni mysql.tables_priv. Ritháttur: GRANT ALL ON gagnagrunns_heiti.töflu_heiti and REVOKE ALL ON gagnagrunns_heiti.töflu_heiti.

Column level (dálka svið): Dálkaréttindi ná til stakra dálka í nefndri töflu og geymast í mysql.columns_priv. Í REVOKE aðgerð þarf að tiltaka þá dálka sem gefin voru réttindi á.

Auðvelt er að fjarlægja öll réttindi frá notanda á einu bretti með eftirfarandi aðgerð:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```

Eftirfarandi tafla listar þau réttindi sem má úthluta:

Réttindi	**Merking**
`ALL [PRIVILEGES]`	Setur öll einföld réttindi nema GRANT OPTION
`ALTER`	Leyfir notkun á ALTER TABLE
`CREATE`	Leyfir notkun á CREATE TABLE
`CREATE TEMPORARY TABLES`	Leyfir notkun á CREATE TEMPORARY TABLE
`DELETE`	Leyfir notkun á DELETE
`DROP`	Leyfir notkun á DROP TABLE
`EXECUTE`	Leyfir notanda að inna „vistaðar stefjur“ (stored procedures)
`FILE`	Leyfir SELECT ... INTO OUTFILE og LOAD DATA INFILE
`INDEX`	Leyfir notkun á CREATE INDEX og DROP INDEX
`INSERT`	Leyfir notkun á INSERT
`LOCK TABLES`	Leyfir LOCK TABLES á töflur sem notandi hefur SELECT réttindi á
`PROCESS`	Leyfir notkun á SHOW FULL PROCESSLIST
`REFERENCES`	Óútfært
`RELOAD`	Leyfir notkun á FLUSH

`REPLICATION CLIENT`	Notandi getur spurt hvar þræl-miðlari og meistara-miðlari eru
`REPLICATION SLAVE`	Nauðsynlegt fyrir þræla sem afrita tvíundar-dagbók frá meistara
`SELECT`	Leyfir notkun á SELECT
`SHOW DATABASES`	Leyfir notkun á SHOW DATABASES
`SHUTDOWN`	Leyfir notkun á `mysqladmin shutdown`
`SUPER`	Leyfir notkun á CHANGE MASTER, KILL, PURGE MASTER LOGS, og SET GLOBAL, auk þess getur skipunin `mysqladmin debug` tengst þó `max_connections` sé náð.
`UPDATE`	Leyfir notkun á UPDATE
`USAGE`	Samheiti fyrir „engin réttindi“ þ.e. “no privileges”
`GRANT OPTION`	Leyfir að notandi úthluti réttindum. Notist varlega.

Gott er að úthluta USAGE til að skilgreina notanda án réttinda og aðeins er hægt að eyða notendum með DROP USER sem hefur USAGE réttindi eingöngu.

Varúð. Nota má ritháttinn ON * þegar réttindum er úthlutað og taka þá gildi á þann gagnagrunn sem er sjálfgefinn/virkur í það sinnið, sé enginn grunnur virkur yrðu réttindin virk á öllum grunnum, þ.e. víðvær.

Úthluta má réttindum á gagnagrunna sem ekki eru til sem getur auðveldað flutning gagnagrunna og notenda á milli miðlara-véla. Auk þess ef gagnagrunnur, eða tafla í honum, hverfur þá eru réttindi notendanna ennþá virk.

Algildistákn í notendastýringum

í GRANT og REVOKE eru notaðir SQL algildisstafirnir (e. Wildcard letters) „_“ og „%“

Þannig myndi eftirfarandi aðgerð „GRANT ALL ON minn_grunnur.* to ...“ veita notandanum réttindi á minnagrunnur, minnbgrunnur, minndgrunnur, o.s.frv.

Ef nota þarf stafi svosem „_“ í nafni má flýja frá algildisstafnum með flóttarunu samanber: „_“ Þannig mætti gera „GRANT ALL ON minn_grunnur.* to ...“ og þá eru réttindin örugglega skorðuð við „minn_grunnur“. Einfalt og öruggt.

Af sömu sökum myndi aðgerðin „GRANT ALL ON test.* TO jonsi@’%.is’“ úthluta notandanum jonsi notkunarréttindum á test grunnin, svo fremi að hann tengist af „.is“ léni.´

Sem fyrr segir þá verður að úthluta réttindum á sniðinu notandi-á-léni. Sé ákveðið að veita réttindi ótilgreindum notanda mætti nota algildisstafi en rétt er að nota ekki algildisstafi fyrir notendanöfn, það er varasamt af öryggisástæðum og virkar ekki alltaf í MySQL.

Hins vegar má nota algildisstafi mjög auðveldlega fyrir hýsla/lén en þá er rétt að nota úrfellingarmerki utan um notendanöfnin sér og hýsil-heitin sér, samanber:

```
GRANT ... TO ‘gestur’@’hvergi.is’... ,
GRANT ... TO ‘gestur’@’%.is’ ...,
GRANT ... TO ‘gestur’@’%.hvergi.is’ ... ,
GRANT ... TO ‘gestur’@’212.30.156.%’ ...,
GRANT ... TO ‘gestur’@’localhost’ ...,
```

Í fyrstu línunni fær notandinn ‘gestur’ aðgang ef hann tengist frá ‘hvergi.is’ en ekki ef hann tengist frá ‘rannsokn.hvergi.is’. Í næstu línu fær hann aðgang ef hann tengist frá ‘.is’ léni, í þriðju línu fengi hann aðgang ef hann tengdist frá ‘rannsokn.hvergi.is’ og frá ‘hvergi.is’, í línunni þar á eftir fengi hann aðgang ef hann tengdist frá hvaða vél sem væri á IP tölunni ‘212.30.156.xx’ og í síðustu línunni ef hann tengist af ‘localhost’ sem er sama vél og miðlarinn er staðsettur á.

Óþekkti notandinn

Þegar MySQL er fyrst sett upp eru skilgreindar aðgangsstýringar fyrir óþekkta notandann (‘’, tómur strengur) á ‘localhost’. Varasamt er að skilgreina óþekkta notendur frá öðrum lénum því þeir myndu sjálfkrafa auðkennast sem þessi notandi og fá hans réttindi.

Flestir varkárir kerfisstjórar eyða algjörlega þessum notanda. Hægt er að athuga með eftirfarandi skipun, hvort hann er til eða ekki og þá gera viðeigandi ráðstafanir:

```
SELECT Host, User FROM mysql.user WHERE User='';
```

Þá mætti eyða honum með, auk þess sem áður er fjallað um undir DROP USER, með:

```
DELETE FROM mysql.user WHERE Host='localhost' AND User='';
FLUSH PRIVILEGES;
```

Aðgerðin FLUSH PRIVILEGES er nauðsynleg til að gera aðgangsstýringar virkar ef og aðeins ef notaðar eru almennar SQL aðgerðir til að meðhöndla þær. Ef FLUSH er ekki notað gerast þær virkar næst þegar miðlarinn er endurræstur.

Dæmi c: Notanda hleypt inn utan úr bæ

Hér sést hvar notandi reynir að tengjast utan úr bæ við „test“ grunninn á vél höfundar. Hann verður að tengjast í gegnum hlið 3307 því hið hefðbundna hlið 3306 er upptekið fyrir annan miðlara á sama léni.

Notandinn gefur ekki upp notendanafn svo MySQL sér hann sem notandann ODBC á því léni sem ADSL þjónusta hans hefur úthlutað honum.

```
C:\mysql\bin>mysql -h grusk.isbok.is -P 3307 test
ERROR 1045 (28000): Adgangur bannadur fyrir notanda: 'ODBC'@'adslxx-x-
xxx.du.simnet.is' (nota lykilord: NEI)
```

Notandinn ODBC hefur verið skilgreindur á vél höfundar með ákveðnu lykilorði, og hann rifjar það nú upp og reynir aftur. Hann nær tenginu og framkvæmir þrjár einfaldar aðgerðir:

```
C:\mysql\bin>mysql -h grusk.isbok.is -P 3307 -p
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
... klippt
mysql> show tables;
ERROR 1046 (3D000): Enginn gagnagrunnur valinn

mysql> use test
Database changed

mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| articles       |
... klippt
| sulta          |
+----------------+
21 rows in set (0.02 sec)
```

```
mysql> select * from glosur;
+-----+-----------+------------------------------+
| num | fyrirsogn | bukur                        |
+-----+-----------+------------------------------+
|   1 | Matur     | Pottur og diskar             |
...klippt
+-----+-----------+------------------------------+
4 rows in set (0.01 sec)

mysql>quit
Bye
```

Notandinn reynir nú aftur með tveim öðrum notendanöfnum:

```
C:\mysql\bin>mysql -h grusk.isbok.is -P 3307 -p -u adam
Enter password: *******
ERROR 1045 (28000): Adgangur bannadur fyrir notanda: 'adam'@'adslxx-x-
xxx.du.simnet.is' (nota lykilord: Ja)

C:\mysql\bin>mysql -h grusk.isbok.is -P 3307 -p -u eva
Enter password: *******
ERROR 1045 (28000): Adgangur bannadur fyrir notanda: 'eva'@'adslxx-x-
xxx.du.simnet.is' (nota lykilord: Ja)
```

Nú tekur kerfisstjórinn sig til á „grusk.isbok.is“ og skilgreinir tvo nýja notendur, adam annars vegar og eva hins vegar. Hann leyfir „evu“ að tengjast frá öllum vélum á léninu „du.simnet.is“ en adam má komast frá öllum vélum á léninu „simnet.is“:

```
grant select, insert, update on test.* to
  'eva'@'%.du.simnet.is' identified by 'paradis';
Query OK, 0 rows affected (0.00 sec)

grant select, insert, update on test.* to
  'adam'@'%.simnet.is' identified by 'paradis';
Query OK, 0 rows affected (0.00 sec)
```

Skoðum aðeins innihaldið í „user“ töflunni:

```
select user, host from mysql.user;
+-------+----------------+
| user  | host           |
+-------+----------------+
| root  | localhost      |
| ODBC  | %.du.simnet.is |
| elias | %              |
| %     | %.du.simnet.is |
| %     | localhost      |
| elias | localhost      |
| eva   | %.du.simnet.is |
| adam  | %.simnet.is    |
+-------+----------------+
7 rows in set (0.00 sec)
```

Staðreyndin er sú að þegar miðlarinn sér notendurna þá er þeim raðað eftir „host“ dálkinum og sá notandi sem fyrstur finnst á tilteknu léni, án tillit til notendanafns, fær aðgang, í þessu tilfelli munu notendurnir eva, og adam hugsanlega komast inn á réttindum % lénsins.

```
select user, host from mysql.user order by host;
+----------+----------------+
| user     | host           |
+----------+----------------+
| ODBC     | %.du.simnet.is |
| %        | %.du.simnet.is |
| eva      | %.du.simnet.is |
| adam     | %.simnet.is    |
| elias    | %              |
| root     | localhost      |
| %        | localhost      |
| elias    | localhost      |
+----------+----------------+
8 rows in set (0.00 sec)

mysql>quit
bye
```

Nú reynir notandinn að tengjast aftur, utan úr bæ með þessum nýju notendanöfnum:

```
C:\mysql\bin>mysql -h grusk.isbok.is -P 3307 -p -u eva
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
... klippt

mysql> quit

C:\mysql\bin>mysql -h grusk.isbok.is -P 3307 -p -u adam
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g
... klippt
```

SET PASSWORD

```
SET PASSWORD = PASSWORD('some password')
SET PASSWORD FOR user = PASSWORD('some password')
SET PASSWORD FOR user = OLD_PASSWORD('some password')
```

Með þessari aðgerð má skilgreina lykilorð fyrir annaðhvort þann notanda sem er tengdur með fyrri línunni, eða tilgreindan notanda með þeirri síðari. Sömu ritháttarreglur gilda um notendanafnið og í GRANT/REVOKE aðgerðum.

Ýmsir biðlarar, s.s. MyODBC, PHP og fleiri nota ennþá lykilorð á sama formi og MySQL notaði fram til útgáfu 4 og breyttist með útgáfu 4.1. Þegar notendur eru skilgreindir fyrir þessa notendur þarf að setja lykilorðin með OLD_PASSWORD() fallinu.

```
SET PASSWORD FOR 'gestur'@'%.hvergi.is' = PASSWORD('leyndo');
```

Eða:

```
UPDATE mysql.user SET Password=PASSWORD('leyndo')
     WHERE User='gestur' AND Host='%.hvergi.is';
FLUSH PRIVILEGES;
```

Töflu umhald

ANALYZE TABLE

```
ANALYZE TABLE töflu_heiti [, töflu_heiti] ...
```

Þessi aðgerð athugar og vistar lykla töflunnar, hún er læst fyrir öðrum á meðan. Þetta verkar á MyISAM , BDB og InnoDB töflur.

BACKUP TABLE

```
BACKUP TABLE töflu_heiti [, töflu_heiti] ... TO
/path/to/backup/directory'
```

Tekur afrit af töflu. Þessi aðgerð er úrelt (e. Deprecated) og beðið er nýrrar útgáfu.

CHECK TABLE

```
CHECK TABLE töflu_heiti [, töflu_heiti] ...
```

Aðgerðin verkar aðeins á MyISAM og InnoDB töflur. Aðgerðin villuleitar töflu og birtir niðurstöður um ástand hennar. Lyklar eru dagréttaðir.

CHECKSUM TABLE

```
CHECKSUM TABLE töflu_heiti [, töflu_heiti] ...
```

Skilar prófsummu töflu.

OPTIMIZE TABLE Syntax

```
OPTIMIZE TABLE töflu_heiti [, töflu_heiti] ...
```

Þessi aðgerð yfirfer gagnaskrásetningu töflunnar á diski afslitrar hana og er hún læst fyrir öðrum á meðan.. Öll gögn eru geymd í samtengdum lista (e. Linked list). Þegar unnið er mikið með innfærslur, eyðingar og dagréttingar á gögnum getur tafla slitrast[64] (e. Fragment). Þetta getur haft mikla þýðingu fyrir hraða í fyrirspurnum og pláss á diski séu gögn geymd t.d. í VARCHAR, BLOB og TEXT dálkum.

REPAIR TABLE

```
REPAIR TABLE töflu_heiti [, töflu_heiti] ...
```

Aðgerðin freistar þess að laga töflu sem gæti hafa spillst (e. Corrupt) og verkar aðeins á MyISAM töflur. Aðgerðin er yfirleitt óþörf nema þegar slys gerast að þá getur hún náð öllum gögnum til baka.

RESTORE TABLE

```
RESTORE TABLE töflu_heiti [, töflu_heiti] ... FROM '/path/to/backup/directory'
```

Endurvekur töflu sem var afrituð með BACKUP TABLE. Sé taflan til fyrir verður hún ekki yfirrituð. Verkar aðeins á MyISAM töflur.

SET

Aðgerðin SET leyfir notanda að setja breytu-gildi (e. Variable value) og valmöguleika (e. Options).

```
SET variable_assignment [, variable_assignment] ...

variable_assignment:
      user_var_name = expr
    | [GLOBAL | SESSION] system_var_name = expr
    | @@[global. | session.]system_var_name = expr
```

[64] Orðanefnd mælir með hinu ágæta orði Slitrun fyrir „Fragmentation“ og Heilun fyrir „Defragmentation“ (http://www.ismal.hi.is/to/Vikuord.htm). Höfundur kýs að nota „afslitrun“ fyrir Defragmentation þar til hann verður „skammaður ofan af því.“

Setja má breytur á eftirfarandi máta:

```
SET max_join_size=DEFAULT;
SET @@session.max_join_size=@@global.max_join_size;
```

Birta má lista fyrir breytur sem eru virkar með skipun á borð við:

```
SHOW VARIABLES LIKE 'max_join_size';
SHOW GLOBAL VARIABLES LIKE 'max_join_size';
```

SHOW

Aðgerðin SHOW ... leyfir notanda að skoða heilmargt svosem hvaða stafamengi eru til, hvernig var tafla smíðuð, hvernig litur taflan út o.s.frv.

Málskipan og möguleikar fyrir SHOW eru eftirfarandi:

```
SHOW [FULL] COLUMNS FROM töflu_heiti [FROM db_name] [LIKE 'pattern']
SHOW CREATE DATABASE db_name
SHOW CREATE TABLE töflu_heiti
SHOW DATABASES [LIKE 'pattern']
SHOW [STORAGE] ENGINES
SHOW ERRORS [LIMIT [offset,] row_count]
SHOW GRANTS FOR user
SHOW INDEX FROM töflu_heiti [FROM db_name]
SHOW INNODB STATUS
SHOW [BDB] LOGS
SHOW PRIVILEGES
SHOW [FULL] PROCESSLIST
SHOW STATUS [LIKE 'pattern']
SHOW TABLE STATUS [FROM db_name] [LIKE 'pattern']
SHOW [OPEN] TABLES [FROM db_name] [LIKE 'pattern']
SHOW [GLOBAL | SESSION] VARIABLES [LIKE 'pattern']
SHOW WARNINGS [LIMIT [offset,] row_count]
```

Hér á eftir fara nokkur dæmi um algengustu notkun á SHOW

Hvaða innri ferli eru virk hjá miðlaranum:

```
show processlist;
+----+-------+----------------+-------+---------+------+-----------------
| Id | User  | Host           | db    | Command | Time | State
+----+-------+----------------+-------+---------+------+-----------------
|  1 | root  | localhost:2534 | mysql | Query   |    0 | NULL
| 12 | elias | localhost:2752 | NULL  | Connect |   97 | Reading from net
| 15 | %     | localhost:2755 | NULL  | Sleep   |   11 |
+----+-------+----------------+-------+---------+------+-----------------
3 rows in set (0.00 sec)
```

Hvaða breytur hafa hvaða gildi:

```
show variables like '%connect%';
+--------------------------+-------------------+
| Variable_name            | Value             |
+--------------------------+-------------------+
| character_set_connection | latin1            |
| collation_connection     | latin1_swedish_ci |
| connect_timeout          | 5                 |
| init_connect             |                   |
| max_connections          | 150               |
| max_connect_errors       | 10                |
| max_user_connections     | 0                 |
+--------------------------+-------------------+
```

```
7 rows in set (0.00 sec)
```

Birtu aðferðina sem smíðaði gagnagrunninn `gagnagrunnur`:

```
show create database malskipan;
+-----------+-----------------------------------------------------------------
| Database  | Create Database
+-----------+-----------------------------------------------------------------
| malskipan | CREATE DATABASE `malskipan` /*!40100 DEFAULT CHARACTER SET ...
+-----------+-----------------------------------------------------------------
1 row in set (0.01 sec)
```

Birtu lista yfir töflur í virkum gagnagrunni:

```
show tables;
+---------------------+
| Tables_in_malskipan |
+---------------------+
| henda               |
...klippt
| tafla22             |
+---------------------+
14 rows in set (0.02 sec)
```

Birtu aðferðina sem smíðaði töflu:

```
show create table postnr;
+--------+-------------------------------------+
| Table  | Create Table                        |
+--------+-------------------------------------+
| postnr | CREATE TABLE `postnr` (             |
|  `pnr` char(3) NOT NULL default '',          |
|  `heiti` varchar(25) NOT NULL default '',    |
|  PRIMARY KEY  (`pnr`)                        |
| ) ENGINE=MyISAM DEFAULT CHARSET=latin1       |
+--------+-------------------------------------+
1 row in set (0.03 sec)
```

Birtu yfirlit yfir tegundir af aðgangsstýringum:

```
show privileges;
+-----------+-------------------------+-----------------------------------------------------+
| Privilege  | Context                 | Comment                                             |
+-----------+-------------------------+-----------------------------------------------------+
| Select     | Tables                  | To retrieve rows from table                         |
| Insert     | Tables                  | To insert data into tables                          |
...klippt
| Process    | Server Admin            | To view the plain text of currently executing queries |
| File       | File access on server   | To read and write files on the server               |
+-----------+-------------------------+-----------------------------------------------------+
14 rows in set (0.00 sec)
```

Birtu athugasemdalista yfir um nýliðnar SQL aðgerðir:

```
show warnings;
Empty set (0.01 sec)
```

```
show databases;
+---------------+
| Database      |
+---------------+
| malskipan     |
| mysql         |
| test          |
+---------------+
```

```
3 rows in set (0.00 sec)
```

Birtu aðgangsstýringar fyrir „notandi-á-hýsli.“

```
show grants for '%'@'localhost';
+-----------------------------------------------------------------------------+
| Grants for %@localhost                                                      |
+-----------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO '%'@'localhost' IDENTIFIED BY PASSWORD '64df4794'     |
| GRANT ALL PRIVILEGES ON `isb_vidsk`.* TO '%'@'localhost' WITH GRANT OPTION |
+-----------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

Birtu lista yfir stafmengin sem miðlarinn styður:

```
show character set;
+----------+-----------------------------+---------------------+--------+
| Charset  | Description                 | Default collation   | Maxlen |
+----------+-----------------------------+---------------------+--------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |      2 |
...klippt
| geostd8  | GEOSTD8 Georgian            | geostd8_general_ci  |      1 |
+----------+-----------------------------+---------------------+--------+
34 rows in set (0.00 sec)
```

Finnum stafamengi sem hefst á latin*:d

```
show character set like 'latin%';
+---------+-----------------------------+-------------------+--------+
| Charset | Description                 | Default collation | Maxlen |
+---------+-----------------------------+-------------------+--------+
| latin1  | ISO 8859-1 West European    | latin1_swedish_ci |      1 |
| latin2  | ISO 8859-2 Central European | latin2_general_ci |      1 |
| latin5  | ISO 8859-9 Turkish          | latin5_turkish_ci |      1 |
| latin7  | ISO 8859-13 Baltic          | latin7_general_ci |      1 |
+---------+-----------------------------+-------------------+--------+
4 rows in set (0.00 sec)
```

Birtu lista yfir röðunarsöfn miðlarans:

```
show collation;
+----------------------+----------+----+---------+----------+---------+
| Collation            | Charset  | Id | Default | Compiled | Sortlen |
+----------------------+----------+----+---------+----------+---------+
| big5_chinese_ci      | big5     |  1 | Yes     | Yes      |       1 |
...klippt
| geostd8_general_ci   | geostd8  | 92 | Yes     |          |       0 |
| geostd8_bin          | geostd8  | 93 |         |          |       0 |
+----------------------+----------+----+---------+----------+---------+
82 rows in set (0.00 sec)
```

Finnum þau röðunarsöfn sem innihalda stafina „utf“ í heiti sínu:

```
show collation like '%utf%';
+-----------------+---------+----+---------+----------+---------+
| Collation       | Charset | Id | Default | Compiled | Sortlen |
+-----------------+---------+----+---------+----------+---------+
| utf8_general_ci | utf8    | 33 | Yes     | Yes      |       1 |
| utf8_bin        | utf8    | 83 |         | Yes      |       1 |
+-----------------+---------+----+---------+----------+---------+
2 rows in set (0.00 sec)
```

Ýmis stjórnun

FLUSH

```
FLUSH flush_option [, flush_option] ...
```

Aðgerðin FLUSH er notuð til að hreinsa ýmis innri biðminni í MySQL og taka til. Þegar gert er FLUSH PRIVILEGES eru hreinsuð úr biðminni miðlarans allar notendastýringar og neyðir hann til að lesa inn gildin í notendatöflunum. Til að gera FLUSH þarf notandinn að hafa RELOAD réttindi.

Valmöguleikinn „`flush_option`“ getur verið:

HOSTS: Hreinsar út allar upplýsingar um lén eða hýsla sem geta tengst gagnagrunni. Þetta er nauðsynlegt ef einhver vél sem tengist reglulega hefur skipt um IP tölu eða léns-heiti.

LOGS: Lokar öllum dagbókum (e. Logs) og opnar þær aftur. Fyrir tvíundar-dagbókina (e. Binary Log) hefur þetta þau áhrif að ný skrá er búin til.

PRIVILEGES: Sturtar niður öllum réttindum og endurhleður þeim í minni. Gerist sjálfkrafa sé notað GRANT eða REVOKE til að skilgreina notendur. Þarf að framkvæma ef notaðar eru INSERT, UPDATE eða DELETE aðgerðir við aðgangsstjórnun.

QUERY CACHE: Afslitrar „fyrirspurna biðminnið“ (Query Cache) svo það nýti vinnsluminnið betur.

STATUS: Setur flesta ástands-mæla (status variables) á núll.

{**TABLE** | **TABLES**} [töflu_heiti [, töflu_heiti] ...]: Lokar öllum nefndum töflum og opnar þær aftur.

TABLES WITH READ LOCK: Lokar öllum opnum töflum og læsir (í öllum grunnum) með les-læsingu þar til framkvæmt er UNLOCK TABLES. Hentugt við afritunartöku því biðlarar geta þá beðið á meðan afritun fer fram, sem þarf ekki að taka nema örfáar sekúndur.

USER_RESOURCES: Endursetur mæla fyrir notendur sem eru háðir tengifjölda og fyrirspurnafjölda þannig að þeir geta hafist handa að nýju.

Sumar FLUSH aðgerðir má framkvæma með „mysqladmin“

KILL

```
KILL [CONNECTION | QUERY] thread_id
```

Hver einasta tenging við miðlarann fer fram í sjálfstæðum þráðum sem drepa má með þessari aðgerð. Sá sem hefur PROCESS réttindi getur séð alla þræði og sé sem hefur SUPER réttindi getur drepið alla þræði.

RESET

```
RESET reset_option [, reset_option] ...
```

Þessi aðgerð er öflug FLUSH útgáfa og endurstillir ýmislegt ástand miðlarans. Notandinn verður að hafa RELOAD réttindi.

Valmöguleikinn „`reset_option`“ getur verið:

MASTER: Eyðir öllum tvíundar-dagbókum í atriðaskrá og setur upp nýja tóma.

QUERY CACHE: Eyðir öllum fyrirspurnum í biðminni miðlarans. Fyrirspurna-biðminni er eiginleiki MySQL til að muna allar fyrirspurnir á gögn sem ekki hafa breyst þannig að næst þegar þær séu framkvæmdar þurfi þess ekki heldur megi birta niðurstöður beint..

SLAVE: Lætur miðlara-þræl gleyma hvert dagréttunar-ástand hans er samkvæmt aðal tvíundardagbók.

8. Kafli. MySQL töflutegundir

MySQL styður nokkrar tegundir af töflum, eða geymslu maskínum, bæði færslu-hæfar (e. Transaction-safe) og færslu-óhæfar. Upprunalega töflutegundin var ISAM tegund en hún er nú úrelt (e. Deprecated) og fjarlægð frá og með útgáfu 5.0.

Sjálfgefna töflutegundin er MyISAM sem er mjög hraðvirk en færslu-óhæf. Skyld tegund er HEAP (eða MEMORY) sem er aðeins til í vinnsluminni miðlarans og á meðan tenging varir. Þá er til MERGE tegund sem er notuð til að líta á margar töflur (sem líta eins út) sem eina.

Færslu-hæfar tegundir eru InnoDB og BDB töflur og hægt er að þýða MySQL (úr uppruna-kóta e. Source code) með stuðningi við NDBCluster töflutegund sem getur skipt einni töflu yfir á margar tölvur í senn.

Þegar tafla er smíðuð má tilgreina sérstaklega hvaða töflutegund hún skuli nota samanber:

```
CREATE TABLE t (i INT) ENGINE = INNODB;
CREATE TABLE t (i INT) TYPE = MEMORY;
CREATE TABLE t (i INT); (hér er sjálfgefið MyISAM)
```

Lykilorðið ENGINE er hið meðmælta form, lykilorðið TYPE er eldra og búast má við að það úreldist með tímanum. Hægt er að setja kerfisbreytuna „`table_type=töflutegund`“ til að skilgreina að önnur en MyISAM tegundin sé notuð sjálfkrafa fyrir nýjar töflur.

Skipta má um tegund með ALTER TABLE aðgerð samanber:

```
ALTER TABLE t ENGINE = MYISAM;
ALTER TABLE t TYPE = BDB;
```

Ef sú töflutegund sem beðið er um er ekki til fyrir einhverjar sakir þá er sjálfkrafa valið MyISAM þegar tilgreind er töflutegund, þetta er sérlega hentugt þegar t.d. gagnagrunnar eru færðir á milli tölva með MYSQLDUMP og MYSQLIMPORT en þær aðgerðir tilgreina ávallt töflutegundina sem nota þarf.

Hver gagnagrunnur getur notað blöndu af þeim töflutegundum sem til eru og tafla (þ.e. skilgreining dálka hennar) er ávallt geymd í skrá með endingunni „`.frm`“. Til eru fleiri tegundir skráa sem notaðar eru eftir töflutegundum.

Notkun færslu-hæfra taflna hefur þá helstu kosti að auðvelt er að endurvekja gögn í þeim ef þær hrynja, bæði má endurvekja sjálfkrafa og frá dagbókum (e. Logs). Þá leyfa þær að fjöl-aðgerðir (e. Multiple statements) þannig að þær séu undirbúnar og framkvæmdar allar í einu með COMMIT (í. Staðfesta) eða þeim „rúllað afturábak“ með ROLLBACK svo að engin þeirra framkvæmist. Einn kostur þeirra er að séu margar töflur í gagnagrunninum er hægt að auka samkvæmni þeirra án sérstakrar viðbótarvinnu.

Færslu-óhæfar töflur hins vegar hafa ýmsa kosti m.a. eru þær hraðvirkari í vinnslu, taka minna diskpláss og tiltölulega (e. Respectively) lítið minnispláss. Kosið er að nota færslu-óhæfar töflur við þær aðstæður þar sem ekki þarf að gæta þess að færslur í aðskildum gagnagrunnum þurfi að vera í samræmi hver við aðra sakir framangreindra kosta.

Töflutegundin MyISAM

MyISAM töflutegundin er sú sjálfgefna fyrir nýjar töflur ef engin „`ENGINE=`“ valmöguleiki er notaður við smíði hennar. Yfirleitt dugar þessi tegund mjög vel fyrir töflur og er öll gagnavinnsla þeirra mjög hraðvirk.

MyISAM töflur eru geymdar í þrem skrám með endingunum „.frm“ (töfluskilgreining), „.MYI“ (atriða-skrár e. Index) og „.MYD“ (gögn töflunnar). Taflan „hestar“ myndi þannig vera geymd í „hestar.frm,“ „hestar.MYI“ og „hestar.MYD.“

Þegar þess þarf má pakka töflum svo þær taki minna pláss með „myisampack“ forritinu en þá verða þær aðeins læsilegar (e. Read only). Laga má töflur sem hafa bilað með forritinu „myisamchk“.

Töflutegundin MERGE

MERGE tafla er tafla sem birtir margar töflur sem eina. Töflurnar sem liggja á bak við MERGE töflu verða að vera eins og í alla staði af sams konar skilgreiningu, vera MyISAM og hafa samskonar atriðiðaskrár. Þær þurfa þó ekki að vera í sama gagnagrunni.

Hægt er að pakka MERGE töflum með „myisampack“ og framkvæma á þeim aðgerðirnar SELECT, DELETE, UPDATE og INSERT, svo fremi að notandinn hafi aðgang að þeim samkvæmt „grant“ skilgreiningum.

MERGE töflur eru geymdar í „.frm“ og „.MRG“ skrám á diski. Þegar þær eru skilgreindar verður að nota UNION klausu til að tilgreina þær töflur sem liggja á bak við. Þá má bæta við valmöguleikanum INSERT_METHOD í skilgreininguna sem getur haft LAST, FIRST eða NO sem stika og tilgreinir hvort leyft sé að setja gögn í samsettu töfluna og þá hvort sett sé í fremstu eða öftustu undir-töfluna. Sé INSERT_METHOD sleppt munu gagna-innsetningar ekki virka.

Smíðum tvær töflur sem líta eins út og setjum tvær færslur í hvora töflu:

```
create table e1(
  a int unsigned auto_increment primary key,
  glosur varchar(35)
);

create table e2(
  a int unsigned auto_increment primary key,
  glosur varchar(35)
);

insert into e1(glosur) values('jól'),('Páskar');
insert into e2(glosur) values('Fiskur'),('Kjöt');
```

Smíðum nú töflu sem sameinar fyrri töflurnar tvær. Skilgreinum að síðari taflan skuli taka við öllum innskotum:

```
create table e12(
  a int unsigned auto_increment primary key,
  glosur varchar(35),
  index(a)
  ) engine=MERGE union(e1,e2) insert_method=last;
```

Skoðum nú allar færslur í sameinuðu töflunni, skjótum inn einni færslu og skoðum svo allar færslur aftur:

```
select * from e12;
+---+--------+
| a | glosur |
+---+--------+
| 1 | jól    |
| 2 | Páskar |
| 1 | Fiskur |
| 2 | Kjöt   |
+---+--------+
4 rows in set (0.02 sec)

insert into e12(glosur) values('Hvítasunna');

select * from e12;
+---+------------+
| a | glosur     |
+---+------------+
| 1 | jól        |
| 2 | Páskar     |
| 1 | Fiskur     |
```

```
| 2 | Kjöt       |
| 3 | Hvítasunna |
+---+------------+
5 rows in set (0.00 sec)
```

Áð lokum skulum við skoða hvert síðasta innskotið fór:

```
select * from e1;
+---+--------+
| a | glosur |
+---+--------+
| 1 | jól    |
| 2 | Páskar |
+---+--------+
2 rows in set (0.00 sec)

select * from e2;
+---+------------+
| a | glosur     |
+---+------------+
| 1 | Fiskur     |
| 2 | Kjöt       |
| 3 | Hvítasunna |
+---+------------+
3 rows in set (0.01 sec)
```

Helstu kostirnir við samsettar töflur er að ef notaðar væru mismunandi töflur undir dagbókarfærslur, þar sem allar töflurnar litu eins út en hver um sig væri fyrir sinn mánuð í senn, mætti nota MERGE töflu til að skoða öll gögn þeirra í einu. Þá mætti vel hugsa sér að skipta mætti mjög stórum töflum upp í margar, sem gæti aukið hraða í notkun þeirra, og nota MERGE töflu undir samantektir. Við slíkar aðstæður gætu ýmsar hnitmiðaðar leitir orðið mun skilvirkari.

Ókostur þessara taflna er að allar undirliggjandi töflur þurfa að vera eins hannaðar og verða að vera af MyISAM tegund. Auk þess krefjast þær aukins álags á vélina. Til dæmis ef 10 notendur eru að skoða sömu MERGE töfluna, sem sameinar 10 töflur þarf (10*10)+10 skráa-lýsingar fyrir hana. Tíu fyrir hvern biðlara sem er tengdur, tíu fyrir hverja töflu og tíu fyrir hverja atriðisskrá.

Töflutegundin MEMORY (áður HEAP)

Töflutegundin MEMORY er geymd í vinnsluminni tölvunnar. Reyndar er skilgreining hennar geymd á diski í „.frm“ skrá en engin gögn eru geymd. Þannig að þegar miðlarinn slekkur á sér (e. Shuts down) hverfa gögn hennar. Taflan er þó til áfram þó biðlarinn sem smíðaði hana lokist, öfugt við „temporary“ töflur sem eru eingöngu til meðan biðlarinn sem bjó þær til er tengdur.

```
create table minni(
  num int auto_increment primary key,
  glosur varchar(255)
) engine memory;

insert into minni(glosur) values('Fílar'),('Gíraffar');
```

Eyða má þessum töflum með DROP TABLE aðgerð.

Ekki skyldi nota ISAM töflutegundina þar sem hún er úrelt og hverfur í MySQL útgáfu 5.0. Hægt er að breyta gömlum ISAM töflum í MyISAM með eftirfarandi aðgerð:

```
ALTER TABLE tbl_name TYPE = MYISAM;
```

Töflutegundin InnoDB

InnoDB töflutegundin styður færslu-hæfni og einnig framandlykla og tilvísana-heilindi (e. Referential Integrity).

Framandlyklar eru notaðir þannig að ein tafla skilgreinir tiltekinn dálk sem framandlykil á aðallykil annarrar töflu. Þannig gætum við litið á töflu yfir póstnúmer sem hefur tvo dálka, póstnúmer og póstsvæði. Í þeirri töflu væri póstnúmerið skilgreint sem aðallykill og væri aðeins hægt að skrásetja eitt póstnúmer í senn.

Nú mætti skilgreina nafnatöflu sem geymir upplýsingar um einstaklinga og hefur eftirfarandi dálka: Kennitala sem aðallykill, nafn, heimilisfang og póstnúmer. Dálkinn póstnúmer mætti nú skilgreina sem framandlykil á póstnúmera töfluna.

Framandlykilinn gæti nú fylgst með tilvísana-heilleika sem virkar þannig:

1. Ef breytt er póstnúmeri í póstnúmera töflunni úr t.d. 101 í 200 þá myndi póstnúmerið í nafnatöflunni verða dagréttað (e. Update) svo það yrði líka 200 þar.
2. Ef póstnúmeri væri eytt út (úr póstnúmera töflunni) þá annaðhvort myndum við eyða öllum nöfnum (í nafnatöflunni) sem eiga sama póstnúmer eða setjum t.d. póstnúmerin þar á NULL.
3. Ekki væri hægt að setja inn í nafnatöfluna nein póstnúmer sem ekki væru til í póstnúmera töflunni.

Framandlykla má skilgreina sem mjög einfalda en oftast nær viljum við þó tiltaka að eitthvað gerist sé heilleika þeirra ógnað. Sé til dæmis eytt út póstnúmeri í raunveruleikanum þá stenst varla að einstaklingur í nafnatöflunni búi þar lengur því það er ekki lengur til. Óþarfi er þó að eyða einstaklingnum því hann er líklega ennþá til. Þess vegna væri eðlilegt að skilgreina eftirfarandi: sé póstnúmeri eytt þá setjum við sama póstnúmer á NULL í nafnatöflunni en sé skipt um númer eða nafn á póstnúmeri afritum við nýja póstnúmerið inn í nafnatöfluna á rétta staði.

Eftirfarandi er málskipan framandlykla:

```
[CONSTRAINT symbol] FOREIGN KEY [id] (index_col_name, ...)
    REFERENCES tbl_name (index_col_name, ...)
    [ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT}]
    [ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT}]
```

Aðal atriðið varðandi notkun framandlykla í InnoDB töflum er að bæði sá dálkur sem á að vera framandlykill verður að vera fremsti dálkur töflunnar og geta geymt NULL. Einnig þarf hann að vera samsettur lykill, þannig að t.d. framandlykillinn og aðallykillinn í þeirri töflu séu einkvæmir þegar báðir koma fyrir saman. Í dæminu hér gætum við ákveðið að engin kennitala (einstaklingur) getur búið á tveim póstnúmerum og því skilgreint: UNIQUE(póstnúmer, kennitala), póstnúmer getur nú verið framandlykill svo fremi að hann sé fremst í töflunni og geti geymt NULL gildi.

Dæmi um notkun framandlykla

Eftirfarandi dæmi setur upp póstnúmeratöflu sem geymir einkvæmt póstnúmer og er póstnúmerið aðallykill. Því næst er skilgreind nafnatafla sem geymir einstaklinga og þar er aðallykillinn kennitala og samsettur lykill póstnúmera og kennitala.

Smíðum sérstakan grunn fyrir æfinguna:

```
create database framandlyklar;
use framandlyklar;
```

Smíðum töflu utan um póstnúmer og tilgreinum að póstnúmerið sjálft sé aðallykill:

```
create table postnr(
   pnr varchar(3) not null primary key,
```

```
    pnafn varchar(25) not null
    ) engine innodb;
```

Nú er smíðuð nafnatafla. Sviðið pnr sem geymir póstnúmer verður framandlykill á aðallykilinn í póstnúmera töflunni. Til þess að geta það verðum við að tilgreina að sviðið sé fremst í töflunni. Einnig þurfum við að skilgreina að það sé að einhverju leyti einkvæmt. Þetta má gera þannig að skilgreina það sem hluta samsetts lykils (e. Unique). Framandlykill getur aldrei verið án gildis í InnoDB töflum.

```
create table nofn(
    pnr varchar(3) not null,  # Rangt þarf að vera NULL
    kennit varchar(10) not null primary key,
    nafn varchar(31) not null,
    heima varchar(25) null,
    unique(pnr, kennit),
    foreign key (pnr)
       references postnr (pnr) on update cascade on delete set null
    )engine innodb;

ERROR 1005 (HY000): Can't create table '.\framandlyklar\nofn.frm' (error nr
: 150)
```

Villunúmer 150 merkir að framandlyklar og/eða skorður á þá séu rangar, breytum því pnr í NULL og allt gengur upp:

```
create table nofn(
    pnr varchar(3) null,  # framandlykil
    kennit varchar(10) not null primary key,
    nafn varchar(31) not null,
    heima varchar(25) null,
    unique(pnr, kennit),  # samsettur lykill
    foreign key (pnr)
        references postnr (pnr) on update cascade on delete set null
    )engine innodb;
```

Skjótum inn tveim póstnúmerum:

```
insert into postnr values('101','Reykjavík');
insert into postnr values('200','Kópavogur');
```

Skjótum inn tveimur einstaklingum:

```
insert into nofn values('101','1234567890','Jóli','Hérabraut');
insert into nofn values('200','1234567891','Moli','Kýrbraut');
```

Skoðum því næst gildin í báðum töflum:

```
select * from postnr;
+-----+-----------+
| pnr | pnafn     |
+-----+-----------+
| 101 | Reykjavík |
| 200 | Kópavogur |
+-----+-----------+
2 rows in set (0.00 sec)

select * from nofn;
+------+------------+------+-----------+
| pnr  | kennit     | nafn | heima     |
+------+------------+------+-----------+
| 101  | 1234567890 | Jóli | Hérabraut |
| 200  | 1234567891 | Moli | Kyrbraut  |
+------+------------+------+-----------+
2 rows in set (0.00 sec)
```

Nú skulum við breyta póstnúmeri 101 í að vera „300 Akranes“:

```
update postnr set pnr='300', pnafn='Akranes' where pnr='101';

select * from postnr;
+-----+-----------+
| pnr | pnafn     |
+-----+-----------+
| 200 | Kópavogur |
| 300 | Akranes   |
+-----+-----------+
2 rows in set (0.00 sec)
```

Þegar við skoðum nú öll nöfn þá sjáum við að pnr fyrir Jóli dagréttast hljóðlega úr 101 í 300.

```
select * from nofn;
+------+------------+------+-----------+
| pnr  | kennit     | nafn | heima     |
+------+------------+------+-----------+
| 300  | 1234567890 | Jóli | Hérabraut |
| 200  | 1234567891 | Moli | Kyrbraut  |
+------+------------+------+-----------+
2 rows in set (0.00 sec)
```

Ef við nú reynum að skjóta inn einstaklingi sem býr á póstnúmeri 400 fáum við villuboð. Þar sem póstnúmerið er ekki til í póstnúmera skránni þá getum við ekki skráð neinn á því númeri.

```
insert into nofn values('400','123456789','Boli','Hvalstíg');
ERROR 1216 (23000): Cannot add or update a child row: a foreign key constraint fails
```

Yfirlit töflutegunda

Töfluvél/töflutegund	Lýsing
BDB	Töflur sem styðja færslur og læsingar.
BerkeleyDB	Viðurnefni fyrir BDB.
HEAP	Gögn í þessari töflu eru geymd í vinnsluminni.
ISAM	Upprunalega (úrelta) töfluvélin í MySQL.
InnoDB	Töflur sem styðja færslur, læsingar og framandlykla.
MEMORY	Viðurnefni fyrir HEAP. Mælt er með þessari notkun orðsins.
MERGE	Leyfir að MyISAM töflum sé safnað saman sem ein væri.
MRG_MyISAM	Viðurnefni á MERGE.
MyISAM	Töfluvél sem geymir tvíundargögn. Aðalvélin í MySQL og kemur í stað ISAM tegundarinnar.

9. Kafli. Föll og virkjar

Segðir[65] má nota víðsvegar í SQL setningum s.s. í ORDER BY og HAVING klausum í SELECT aðgerðum, í WHERE klausum á SELECT, DELETE og UPDATE setningum eða í SET setningum. Segðir má rita með lesgildum, dálkagildum, NULL gildum, föllum og virkjum. Þessi kafli birtir yfirlit yfir þau föll, segðir og virkja sem MySQL styður.

Segð sem inniheldur NULL virkja eða gildi skilar alltaf NULL gildi nema það sé sérstaklega tekið fram.

MySQL leyfir ekki að hvítabil (e. White space, þ.e. orðabil) sé á milli heiti falls og opnun svigans, nema „`--sql-mode=IGNORE_SPACE`“ sé sett við ræsingu miðlarans. Samanber: fall() er löglegt en fall () er ólöglegt nema með þessari breytu. Þetta hjálpar miðlaranum að greina á milli sviga sem innihalda dálkatilvísanir og falla. Hvítabil eru leyfð utanum stika[66] (e. Parameter) sem sendir eru á föll.

Sakir stuttleika (e. Brevity) er sú málvenja tekin að láni í handbók miðlarans að breyta úttaki mysql biðlarans á þann hátt sem hér er sýnt:

```
SELECT MOD(29,9);                                   +-----------+
                                                    | mod(29,9) |
                                                    +-----------+
                                                    |         2 |
                                                    +-----------+
                                                    1 rows in set (0.00 sec)

SELECT MOD(29,9);                                    --> 2
```

Virkjar

(...) Svigar

Nota skal sviga ef stýra þarf ákvörðunar[67] forgangi segðar, samanber:

```
SELECT 1+2*3;                                        --> 7
SELECT (1+2)*3;                                      --> 9
```

[65] Expression: Segð er máleining sem er sett saman úr aðgerðum og þolendum, sem aðgerðirnar verka á, og getur gefið eina eða fleiri útkomur. Þolendur geta verið lesgildi, nefni eða köll á föll.

[66] Parameter: Stiki (í forritunarmálum) er máleining, ætluð til að hleypa gagnahlutum eða gagnagildum á milli forritseininga.
Sú málvenja ríkir í bókum ÍSBÓKAR að stiki sé sú færibreyta sem send er á fall eða stefju (e. Procedure) en færibreyta sé sú sem tekur við stikanum, þ.e. færibreyta er skilgreind í undirskrift (e. Signature) falls eða stefju. Í sjálfu sér mætti kalla þetta báðum samheitum sínum þvers og kruss. Sú málvenja ríkir í enskum tölvubókmenntum að senda frumgildi á föll og stefjur (e. pass an argument to a function or procedure or method). Sem stendur er málvenja okkar sú að „senda stika“ í þessum tilfellum.

[67] Evaluate: Ákvarða gildi (ráða), er að reikna út gildi á tiltekinni segð með því að gefa breytum gildi og reikna út gildi á hverjum lið segðarinnar þar til útkoman er aðeins eitt gildi. Enska: Evaluation, Íslenska: gildisákvörðun, ráðning kv. Það að ákvarða gildi segðar.

Samanburðar virkjar

Samanburðar aðgerðir skila gildinu 1 fyrir satt (e. True) og 0 fyrir rangt (e. False) eða NULL. Samanburður verkar bæði á tölur og strengi og er tölum breytt í strengi og strengjum breytt í tölur eftir því sem með þarf.

MySQL fylgir eftirfarandi reglum við samanburð:

- Ef annar eða báðir stikarnir eru NULL er niðurstaða samanburðar ávallt NULL, nema þegar notaður er <=> virkinn sem er NULL-öruggur.
- Ef báðir stikar í samanburði eru strengir eru þeir bornir saman sem slíkir.
- Ef báðir stikar eru heiltölur eru þær bornar saman sem slíkar.
- Sextándukerfis talnagildi eru meðhöndluð sem tvíundar strengir ef borin saman við tölur.
- Ef annar stikinn er TIMESTAMP eða DATETIME dálkur og hinn stikinn er fasti (e. Constant) er fastanum breytt í tímastimpil (Timestamp) áður en samanburðurinn er framkvæmdur. Þetta er ekki framkvæmt á stika í IN() og væri öruggara að nota löglega tíðastrengi við samanburð.
- Eftir öðrum atvikum eru stikar bornir saman sem hlaupakommutölur (rauntölur).

Sjálfgefið er að strengja samanburður sé hástafa-ónæmur (e Case insensitive) og noti það stafamengi sem ráðandi er á miðlaranum hverju sinni (algengast er þá latin1 stafamengið: ISO-8859-1).

Eftirfarandi dæmi sýnir hvernig strengir breytast í tölur eftir atvikum:

```
SELECT 1 > '6x';                                    --> 0
SELECT 7 > '6x';                                    --> 1
SELECT 0 > 'x6';                                    --> 0
SELECT 0 = 'x6';                                    --> 1
```

= jafnt-og:

```
SELECT 1 = 0;                                       --> 0
SELECT '0' = 0;                                     --> 1
SELECT '0.01' = 0;                                  --> 0
SELECT '.01' = 0.01;                                --> 1
```

<=> NULL-öruggt jafnt-og

Samanburður á sama hátt og með samasem nema skilað er 1 frekar en NULL ef báðir stikar eru NULL, og 0 frekar en NULL ef annar stikinn er NULL.

```
SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;          --> 1, 1, 0
SELECT 1 = 1, NULL = NULL, 1 = NULL;                --> 1, NULL, NULL
```

<>
!= Ekki jafnt-og:

```
SELECT '.01' <> '0.01';                             --> 1
SELECT 'zapp' <> 'zappp';                           --> 1
```

<= Minna en eða jafnt-og

```
SELECT 0.1 <= 2;                                    --> 1
```

< Minna en:

```
SELECT 2 < 2;                                    --> 0
```

>= Stærra en eða jafnt-og:

```
SELECT 2 >= 2;                                   --> 1
```

> Stærra en:

```
SELECT 2 > 2;                                    --> 0
```

IS NULL
IS NOT NULL Hvort gildi sé, eða sé ekki NULL

```
SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;                    --> 0, 0, 1
SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;        --> 1, 1, 0
```

Til þess að vinna vel með ODBC forritum styður MySQL eftirfarandi við IS NULL notkun:

- Finna má út nýjasta AUTO_INCREMENT gildið, með því að gefa eftirfarandi skipun strax og það hefur verið framleitt:
  SELECT * FROM töfluheiti WHERE sjálf_númeraður_dálkur IS NULL
- Fyrir dálka af DATE og DATETIME tegundum sem eru skilgreindir NOT NULL má finna núll-gildið ('0000-00-00') með skipuninni:
  SELECT * FROM töfluheiti WHERE tíða_dálkur IS NULL

segð BETWEEN lágmark AND hámark

Ef segð er stærri-en eða jöfn lágmarki og segð er minni-en eða jöfn hámarki skilar BETWEEN 1 annars er skilað 0.

```
SELECT 1 BETWEEN 2 AND 3;                        --> 0
SELECT 'b' BETWEEN 'a' AND 'c';                  --> 1
SELECT 2 BETWEEN 2 AND '3';                      --> 1
```

segð NOT BETWEEN lágmark AND hámark

Sama og NOT (segð BETWEEN lágmark AND hámark).

COALESCE(value,...)

Skilar fyrsta ekki-NULL gildi í listanum

```
SELECT COALESCE(NULL,1);                         --> 1
SELECT COALESCE(NULL,NULL,NULL);                 --> NULL
```

GREATEST(value1,value2,...)

Skilar til baka þeim stika sem hefur hæsta gildið.

```
SELECT GREATEST(2,0);                            --> 2
SELECT GREATEST('B','A','C');                    --> 'C'
```

segð IN (value,...)

Skilar 1 ef segð er nokkurt gildanna í IN listanum, annars er skilað 0. Fjöldi gilda í IN listanum er aðeins takmarkaður af „max_allowed_packet“ gildinu.

```
SELECT 2 IN (0,3,5,'wefwf');                    --> 0
SELECT 'wefwf' IN (0,3,5,'wefwf');              --> 1
```

segð NOT IN (value,...)

Hið sama og NOT (segð IN (value,...)).

ISNULL(segð)

Ef segð er NULL, skilar ISNULL() 1 annars er skilað 0. Athugið að samanburður á NULL gildum skilar ávallt röngu (false).

```
SELECT ISNULL(1+1);                             --> 0
SELECT ISNULL(1/0);                             --> 1
```

INTERVAL(N,N1,N2,N3,...)

Skilar 0 ef N < N1, 1 ef N < N2 og s.frv. eða -1 ef N er NULL. Allir stikar eru meðhöndlaðir sem heiltölur og krafist er að N1 < N2 < N3 < … < Nn til að virka rétt.

```
SELECT INTERVAL(23, 1, 15, 17, 30, 44, 200);    --> 3 (minna en 30)
SELECT INTERVAL(10, 1, 10, 100, 1000);          --> 2 (minna en 100)
SELECT INTERVAL(22, 23, 30, 44, 200);           --> 0 (minna en 23)
```

LEAST(value1,value2,...)

Séu gefnir tveir eða fleiri stikar er þeim gildis-lægsta skilað. Stikarnir eru bornir saman samkvæmt eftir-farandi: Sé niðurstaðan notuð sem heiltala eða allir stikarnir séu heiltölur vinnur fallið á heiltölum. Sé niðurstaðan eða allir stikarnir sem rauntölur er unnið á rauntölum. Sé einhver stikanna hástafanæmur strengur eru allir stikarnir meðhöndlaðir þannig. Að öðrum kosti er unnið á óstafanæmum strengjum.

```
SELECT LEAST(2,0);                              --> 0
SELECT LEAST('B','A','C');                      --> 'A'
SELECT CAST(LEAST(3600, 9223372036854775808.0) as       --> -9223372036854775808
SIGNED);
```

Rökvirkjar

Í SQL eru allir rökvirkjar ákvarðaðir sem TRUE (satt), FALSE (rangt) eða NULL (óþekkt). Í MySQL er þetta útfært þannig; 1 (TRUE), 0 (FALSE) og NULL. Flestir gagnagrunnsmiðlarar vinna þannig þó til séu þeir sem skila sem TRUE öllum gildum sem eru ekki núll.

NOT
! Röklegt EKKI

Ákvarðast sem 1 ef þolandinn[68] er 0 og sem 0 ef þolandinn er allt annað en 0 nema NOT NULL ákvarðast sem NULL.

```
SELECT NOT 10;                    --> 0
SELECT NOT 0;                     --> 1
SELECT NOT NULL;                  --> NULL
SELECT ! (1+1);                   --> 0
SELECT ! 1+1;                     --> 1
```

AND
&& Röklegt AND

Ákvarðast sem 1 ef allir virkjar eru ekki-núll og ekki NULL, sem 0 ef einn eða fleiri þolenda er 0, annars er NULL skilað.

```
SELECT 1 && 1;                    --> 1
SELECT 1 && 0;                    --> 0
SELECT 1 && NULL;                 --> NULL
SELECT 0 && NULL;                 --> 0
SELECT NULL && 0;                 --> 0
```

OR
|| Röklegt OR

Ákvarðast sem 1 ef einhver þolenda er ekki-núll, NULL ef einhver þolenda er NULL annars er 0 skilað..

```
SELECT 1 || 1;                    --> 1
SELECT 1 || 0;                    --> 1
SELECT 0 || 0;                    --> 0
SELECT 0 || NULL;                 --> NULL
SELECT 1 || NULL;                 --> 1
```

XOR Röklegt XOR

Skilar NULL ef annar hvor þolenda er NULL. Sé enginn þolenda NULL er ákvarðað 1 ef oddatala fjölda þolenda er ekki-núll, annars er 0 skilað. Segja mætti á mæltu máli, „annar en alls ekki báðir.“

```
SELECT 1 XOR 1;                   --> 0
SELECT 1 XOR 0;                   --> 1
SELECT 1 XOR NULL;                --> NULL
```

[68] Operand: Þolandi er stærð sem aðgerð er framkvæmd á.

```
SELECT 1 XOR 1 XOR 1;                                    --> 1
```

Hástafanæmir virkjar

BINARY Tvíundarvirki

Tvíundarvirkinn umbreytir streng yfir í tvíundar-streng, sem neyðir samanburð á honum í hástafanæmi jafnvel þótt dálkurinn sem hann kemur úr sé hástafa-ónæmur. BINARY og BLOB tegundir eru t.d. hástafanæmar. Varist að nota þennan virkja á strengja dálka sem eru hluti af atriðaskrá (e. Index).

Ef þarf að bera strengi úr tvíundardálkum, s.s. BLOB, saman við strengi í hástafa-ónæmi má umbreyta þeim með CONVERT() fallinu. Einnig má nota CONVERT() fallið til að bera strengi saman eftir mismunandi stafamengjum.

```
SELECT 'a' = 'A';                                        --> 1

SELECT BINARY 'a' = 'A';                                 --> 0

SELECT 'A' LIKE CONVERT(blob_col USING latin1) FROM
tbl_name;
```

Föll fyrir flýðistýringu

CASE value WHEN [compare-value] THEN result [WHEN [compare-value] THEN result ...] [ELSE result] END

CASE WHEN [condition] THEN result [WHEN [condition] THEN result ...] [ELSE result] END

Fyrri útgáfan skilar „result“ þegar „`value=compare-value`.“ Síðari útgáfan skilar niðurstöðu fyrir fyrsta skilyrðið sem er satt (TRUE). Ef ekkert gildi finnst sem ákvarðast sem satt er gildinu aftan við ELSE skilað. Tegund skilagildis (INTEGER, DOULBE eða STRING) er sú sama og tegund gildisins sem ritað er á eftir fyrsta THEN gildinu..

```
SELECT CASE 1 WHEN 1 THEN 'one' WHEN 2 THEN 'two' ELSE      --> 'one'
'more' END;

SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;          --> 'true'

SELECT CASE BINARY 'B' WHEN 'a' THEN 1 WHEN 'b' THEN 2      --> NULL
END;
```

DÆMI

```
select * from hlaupid;
+-------+------------+---------------------+-------------+----------------+
| numer | hlaupari   | dags                | lengdmetrar | leid           |
+-------+------------+---------------------+-------------+----------------+
|     1 | 0101770001 | 2004-07-21 00:00:00 |         500 | Krýsuvíkurleið |
|     2 | 1505780002 | 2004-07-21 00:00:00 |         455 | Krýsuvíkurleið |
|     3 | 3010760003 | 2004-07-21 00:00:00 |         680 | Krýsuvíkurleið |
|     4 | 0101770001 | 2003-07-22 13:45:40 |        1500 | Þingvallaleið  |
+-------+------------+---------------------+-------------+----------------+
4 rows in set (0.03 sec)

select case lengdmetrar when 500 then 1 when 1500 then 2 else 3 end, lengdmetrar
 from hlaupid;
```

```
+-------------------------------------------------------------+-------------+
| case lengdmetrar when 500 then 1 when 1500 then 2 else 3 end | lengdmetrar |
+-------------------------------------------------------------+-------------+
|                                                           1 |         500 |
|                                                           3 |         455 |
|                                                           3 |         680 |
|                                                           2 |        1500 |
+-------------------------------------------------------------+-------------+
4 rows in set (0.00 sec)
```

IF(expr1,expr2,expr3)

Ef expr1 er sönn (ekki núll og ekki NULL) þá skilar fallið expr2, annars er skilað expr3. Skilagildið fer eftir samhengi við þau gildi sem notuð eru: Ef annaðhvort expr2 eða expr3 er strengur er streng skilað, sé annaðhvort rauntala er rauntölu skilað og sé annaðhvort heiltala er heiltölu skilað. Samanburðurinn þ.e. expr1 skyldi ávallt vera heiltala og öruggast að umbreyta öðrum tegundum. Sé expr2 eða expr3 hástafanæmir strengir þá er samanburður framkvæmdur í því samhengi..

```
SELECT IF(1>2,2,3);                                    --> 3

SELECT IF(1<2,'yes','no');                             --> 'yes'

SELECT IF(STRCMP('test','test1'),'no','yes');          --> 'no'

SELECT IF(0.1,1,0);                                    --> 0

SELECT IF(0.1<>0,1,0);                                 --> 1
```

DÆMI

```
select if(lengdmetrar>1000,lengdmetrar,0) from hlaupid;
+-----------------------------------+
| if(lengdmetrar>1000,lengdmetrar,0) |
+-----------------------------------+
|                                 0 |
|                                 0 |
|                                 0 |
|                              1500 |
|                              1250 |
|                              1360 |
|                              5960 |
+-----------------------------------+
7 rows in set (0.02 sec)

select if(lengdmetrar>1000,lengdmetrar,'of stutt') from hlaupid;
+-------------------------------------------+
| if(lengdmetrar>1000,lengdmetrar,'of stutt') |
+-------------------------------------------+
| of stutt                                  |
| of stutt                                  |
| of stutt                                  |
| 1500                                      |
| 1250                                      |
| 1360                                      |
| 5960                                      |
+-------------------------------------------+
7 rows in set (0.00 sec)
```

IFNULL(expr1,expr2)

Sé expr1 ekki NULL er skilað expr1 annars er skilað expr2. Tegund skilagildis fer eftir samhengi við þau gildi sem notuð eru.

```
SELECT IFNULL(1,0);                    --> 1
SELECT IFNULL(NULL,10);                --> 10
SELECT IFNULL(1/0,'yes');              --> 'yes'
```

NULLIF(expr1,expr2)

Skilar NULL ef expr1=expr2, annars er skilað expr1.

```
SELECT NULLIF(1,1);                    --> NULL
SELECT NULLIF(1,2);                    --> 1
```

Strengjaföll

Strengjaföll skila NULL ef skilagildið er lengra (í bætum) en „`max_allowed_packet`“ kerfisbreytan. Föll sem vinna á staðsetningu stafa innan strengja er fremsti sætisvísirinn númer 1.

ASCII(str)

Skilar tölugildi (í ASCII kerfi) þess stafs sem er lengst til vinstri í str. Skilar 0 ef str er tómur strengur og NULL sé str NULL. Virkar aðeins fyrir tölugildin 0 til og með 255. Sjá einnig ORD() fallið.

```
SELECT ASCII('2');                     --> 50
SELECT ASCII(2);                       --> 50
SELECT ASCII('dx');                    --> 100
```

BIN(N)

Skilar strengútgáfu tvíundargildis N stikans sem sé BIGINT tala.

```
SELECT BIN(12);                        --> '1100'
```

BIT_LENGTH(str)

Skilar lengd str í bitum.

```
SELECT BIT_LENGTH('text');             --> 32
```

CHAR(N,...)

Túlkar innsenda stika sem heiltölur og skilar til baka streng samsettum úr stöfum samkvæmt þeim. NULL gildum er sleppt.

```
SELECT CHAR(77,121,83,81,'76');        --> 'MySQL'
SELECT CHAR(77,77.3,'77.3');           --> 'MMM'
```

CHAR_LENGTH(str)

Skilar til baka lengd strengjarins str, mældum í stöfum. Fjölbæta stafir s.s. UNICODE eru taldir sem einn stafur. LENGTH() fallið myndi hins vegar telja þá sem 2 stafi.

CHARACTER_LENGTH(str)

Er samheiti fyrir CHAR_LENGTH().

COMPRESS(string_to_compress)

Þjappar streng. Fallið krefst þess að MySQL hafi verið þýtt með þjöppunar safni s.s. zlib annars yrði skilagildið NULL.

```
SELECT LENGTH(COMPRESS(REPEAT('a',1000)));              --> 21
SELECT LENGTH(COMPRESS(''));                            --> 0
```

CONCAT(str1,str2,...)

Skilar streng sem er samsettur úr innsendum strengja-stikum. Skilar NULL ef einhver stikanna sé NULL. Tölum er breytt í strengi og senda má marga stika.

```
SELECT CONCAT('My', 'S', 'QL');                         --> 'MySQL'
SELECT CONCAT('My', NULL, 'QL');                        --> NULL
SELECT CONCAT(14.3);                                    --> '14.3'
```

CONCAT_WS(separator,str1,str2,...)

Er sama og CONCAT() nema með skiltákni[69] sem mun þá vera skeytt inn í samsetta strenginn á samsetningarmörkum innsendra strengja-stika.

```
SELECT CONCAT_WS(',','First name','Last Name');      --> 'First name,Last Name'
SELECT CONCAT_WS(',','First name',NULL,'Last         --> 'First name,Last Name'
Name');
```

CONV(N,from_base,to_base)

Breytir tölum á milli talnakerfa (e. Base). Skilar út streng sem táknar töluna N sem breytt er úr from_base yfir í to_base. Skilar NULL ef einhver innsendra stika er NULL en N er alltaf litið á sem heiltölu. Lágmarks talnakerfi er 2 og hámark er 36. Sé N neikvæð tala er heiltalan með formerki (e. Signed) annars án formerkis.

```
SELECT CONV('a',16,2);                                  --> '1010'
SELECT CONV('6E',18,8);                                 --> '172'
```

[69] Separator: Skili eða skiltákn (í forritunarmálum) er afmarkari sem kemur í veg fyrir að litið sé á grannstæð lesstök eða málskipanareiningar sem eina heild. Bilstafur eða sniðstafur. Einnig afmarkari.

Dæmi með conv()

```
select conv('A',16,10), conv('A',16,2), conv(1010,2,10), conv(1010,2,16);
+----------------+---------------+----------------+----------------+
| conv('A',16,10) | conv('A',16,2) | conv(1010,2,10) | conv(1010,2,16) |
+----------------+---------------+----------------+----------------+
| 10              | 1010          | 10              | A              |
+----------------+---------------+----------------+----------------+
1 row in set (0.01 sec)
```

ELT(N,str1,str2,str3,...)

Skilar str1 ef N=1, str2 ef N=2, o.s.frv. Skilar NULL ef N er minna en 1 eða stærri en fjöldi stika.

```
SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo');              --> 'ej'
SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo');              --> 'foo'
```

EXPORT_SET(bits,on,off,[separator,[number_of_bits]])

Skilar streng þar sem hver biti sem er AF í bits fáum við Á, og AF fyrir hvern bita sem er ekki Á.

```
SELECT EXPORT_SET(5,'Y','N',',',4)                      --> Y,N,Y,N
```

FIELD(str,str1,str2,str3,...)

Skilar sætisvísi str í listanum str1, str2, str3, … Skilar 0 ef str finnst ekki í listanum.

```
SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo');          --> 2
SELECT FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo');          --> 0
```

FIND_IN_SET(str,strlist)

Skilar gildinu 1 til N ef strengurinn str finnst í strengjalistanum strlist sem sé samsettur úr N undir-strengjum, þar sem hver undirstrengur er aðskilinn með kommu (,) rétt eins og í SET tegundinni. strlist má því vera tilvísun á SET dálk. Skilar 0 ef str er ekki í listanum eða listinn er tómur strengur. Skilar NULL ef annarhvor stikanna er NULL. Varist að str innihaldi kommu.

```
SELECT FIND_IN_SET('b','a,b,c,d');                      --> 2
```

HEX(N_or_S)

Stikinn má vera annaðhvort strengur eða tala. Skilað er streng sem táknar stikann í sextándukerfis tölu. Litið er á stikann sem BIGINT gildi.

```
SELECT HEX(255);                                        --> 'FF'
SELECT 0x616263;                                        --> 'abc'
SELECT HEX('abc');                                      --> 616263
```

INSERT(str,pos,len,newstr)

Skilar strengnum str, þar sem undirstreng frá pos að len er skipt út fyrir newstr.

```
SELECT INSERT('Quadratic', 3, 4, 'What');               --> 'QuWhattic'
```

Dæmi með insert()

```
select insert('jólinn',4,0,'a sve');
```

```
+-----------------------------+
| insert('jólinn',4,0,'a sve') |
+-----------------------------+
| jóla sveinn                 |
+-----------------------------+
1 row in set (0.00 sec)
```

INSTR(str,substr)

Skilar staðsetningu fyrsta tilfellis af undirstrengnum substr í str.

```
SELECT INSTR('foobarbar', 'bar');                        --> 4

SELECT INSTR('xbar', 'foobar');                          --> 0
```

LCASE(str)

Er samheiti fallsins LOWER().

LEFT(str,len)

Skilar stöfunum lengst til vinstri í str, jafnmörgum og len segir til um.

```
SELECT LEFT('foobarbar', 5);                             --> 'fooba'
```

LENGTH(str)

Skilar lengd strengjarins str, mælt í bætum. Venjulegir (1 Bætis strengir) telja því hvern staf sem einn en UNICODE stafir telja 2 Bæti. Sjá CHAR_LENGTH()

```
SELECT LENGTH('text');                                   --> 4
```

Dæmi með length()

Búið er að smíða töfluna svo við notum SHOW til að skoða hvernig mætti smíða hana aftur.

```
show create table hlauparar;
+-----------+-------------------------------------------------+
| Table     | Create Table hlauparar                          |
+-----------+-------------------------------------------------+
| CREATE TABLE `hlauparar` (                                  |
|   `kt` varchar(10) collate ucs2_bin NOT NULL default '',    |
|   `nafn` varchar(31) collate ucs2_bin NOT NULL default '',  |
|   `kilo` float unsigned default NULL,                       |
|  PRIMARY KEY  (`kt`)                                        |
|) ENGINE=MyISAM DEFAULT CHARSET=ucs2 COLLATE=ucs2_bin       |
+-----------+-------------------------------------------------+
1 row in set (0.01 sec)
```

Taktu eftir að taflan er smíðuð í UCS2 stafamenginu sem styður Unicode.

```
select length(nafn),nafn, char_length(nafn) from hlauparar;
+--------------+-------------------+-------------------+
| length(nafn) | nafn              | char_length(nafn) |
+--------------+-------------------+-------------------+
|           22 | Jói Jónsson       |                11 |
|           34 | Gudda Gaujadóttir |                17 |
|           26 | Alli Baugsson     |                13 |
+--------------+-------------------+-------------------+
rows in set (0.00 sec)
```

LOAD_FILE(file_name)

Les inn úr skrá og skilar innihaldi hennar sem streng (sem hentar til geymslu í BLOB dálkum). Skráin þarf að vera á þjóns-tölvunni, gefa verður upp slóð að skránni og notandi biðlarans verður að hafa FILE réttindi. Skráin verður að vera læsileg af öllum (Linux/Unix skráakerfis aðgangsstýring) og smærri en `max_allowed_packet` í bætum. Ef skráin er ekki lesanleg er skilað NULL.

Búast má við að nýuppsettur þjónn hafi „max_allowed_packet“ sett á rúmlega 1 MB. Biðlarinn getur breytt því fyrir gildandi tengingu með „set max_allowed_packet=10000000“ til að hækka gildið í 10MB. Stilla þarf þetta „MY.INI“ eða „MY.CNF“varanlega.

```
UPDATE tbl_name
           SET blob_column=LOAD_FILE('/tmp/picture')
           WHERE id=1;


Update myndatafla
  set mynd = load_file('c:/myndir/mynd.gif'), titill = 'mynd.gif';
```

LOCATE(substr,str)
LOCATE(substr,str,pos)

Fyrri málskipanin skilar staðsetningu fyrsta tilfellis af undirstrengnum substr í strengnum str. Sú síðari skilar fyrsta tilfelli undirstrengsins frá og með pos. Skilað er 0 ef undirstrengurinn finnst ekki.

```
SELECT LOCATE('bar', 'foobarbar');                    --> 4
SELECT LOCATE('xbar', 'foobar');                      --> 0
SELECT LOCATE('bar', 'foobarbar',5);                  --> 7
```

LOWER(str)

Skilar strengnum str þar sem öllum stöfum hefur verið breytt í lágstafi.

```
SELECT LOWER('QUADRATICALLY');                        --> 'quadratically'
```

LPAD(str,len,padstr)

Skilar strengnum str, fóðruðum vinstra megin frá með strengnum padstr í allt að len stafi. Sé str lengri en len, er honum skilað klipptum í lengdina len.

```
SELECT LPAD('hi',4,'??');                             --> '??hi'
SELECT LPAD('hi',1,'??');                             --> 'h'
```

LTRIM(str)

Skilar strengnum str þar sem orðabilum hefur verið eytt framan af (vinstra megin frá) í strengnum.

```
SELECT LTRIM('  barbar');                             --> 'barbar'
```

MAKE_SET(bits,str1,str2,...)

Skilar mengi, streng samsettum úr undirstrengjunum, aðskildum með kommu (,). Aðeins þeim strengjum er skilað sem samsvara tilgreindum bitafjölda í bits. str1 er biti 0, str2 biti 1, …

```
SELECT MAKE_SET(1,'a','b','c');                       --> 'a'
SELECT MAKE_SET(1 | 4,'hello','nice','world');        --> 'hello,world'
```

```
SELECT MAKE_SET(1 | 4,'hello','nice',NULL,'world');      --> 'hello'
```

MID(str,pos,len)

Samheiti fallsins SUBSTRING(str,pos,len).

OCT(N)

Skilar streng sem stendur fyrir áttundukerfis gildið af N, þar sem N er BIGINT. Samsvarandi og CONV(N,10,8). Skilar NULL ef N er NULL.

```
SELECT OCT(12);                                         --> '14'
```

OCTET_LENGTH(str)

Samheiti fallsins LENGTH().

ORD(str)

Sé stafurinn lengst til vinstri í str fjölbæta stafur (t.d. Unicode) er skilað talnakóta þess stafs, reiknað þannig: kóti fyrsta stafs * 256, kóti annars * 256 ^ 2, o.s.frv. Sé stafurinn lengst til vinstri hefðbundinn stafur skilar fallið sama gildi og ASCII() fallið.

```
SELECT ORD('2');                                        --> 50
```

POSITION(substr IN str)

Samheiti fyrir LOCATE(substr,str).

QUOTE(str)

Skeytir flóttarunu (e. Escape characters) inn í strenginn str, s.s. innihaldi strengurinn úrfellingamerkið ‘ er sett öfugt skástrik framan við það; \’. Einnig framan við ASCII NUL og Control-z. Lykilorðinu NULL er skilað sem „NULL“. Þetta fall er nauðsynlegt til að skeyta sumum strengjum s.s. vefsíðu innslætti inn í gagnagrunn.

```
SELECT QUOTE('Don\'t');                                 --> 'Don\'t!'
SELECT QUOTE(NULL);                                     --> NULL
```

REPEAT(str,count)

Skilar strengnum str endurteknum count sinnum. Sé count <= 0 er tómum streng skilað og sé str eða count NULL er NULL skilað.

```
SELECT REPEAT('MySQL', 3);                              --> 'MySQLMySQLMySQL'
```

REPLACE(str,from_str,to_str)

Skilar strengnum str þar sem öllum tilfellum from_str er skipt út í stað to_str.

```
SELECT REPLACE('www.mysql.com', 'w', 'Ww');              --> 'WwWwWw.mysql.com'
```

REVERSE(str)

Skilar strengnum str þar sem röð stafanna er snúið við.

```
SELECT REVERSE('abc');                                  --> 'cba'
```

RIGHT(str,len)

Skilar len mörgum stöfum hægra megin úr str.

```
SELECT RIGHT('foobarbar', 4);                    --> 'rbar'
```

RPAD(str,len,padstr)

Skilar strengnum str, fóðruðum hægra megin frá, með padstr, í allt að len stafi. Sé str lengri en len er hann klipptur niður í len.

```
SELECT RPAD('hi',5,'?');                         --> 'hi???'
SELECT RPAD('hi',1,'?');                         --> 'h'
```

RTRIM(str)

Skilar strengnum str þar sem slef-bil eru fjarlægð hægra megin úr str.

```
SELECT RTRIM('barbar   ');                       --> 'barbar'
```

SOUNDEX(str)

Skilar hljóðlíkum (e. Soundex) streng miðað við str. Tveir strengir sem hljóma líkir ættu að hafa sama hljóðlíka strenginn. Allir stafir sem ekki eru í stafrófinu eru sniðgengnir og allir bókstafir sem ekki eru í því enska eru meðhöndlaðir sem sérhljóðar.

```
SELECT SOUNDEX('Hello');                         --> 'H400'
SELECT SOUNDEX('Quadratically');                 --> 'Q36324'
```

expr1 SOUNDS LIKE expr2

Hið sama og SOUNDEX(expr1) = SOUNDEX(expr2).

SPACE(N)

Skilar streng samsettum úr N mörgum orðabilum.

```
SELECT SPACE(6);                                 --> '      '
```

SUBSTRING(str,pos)
SUBSTRING(str FROM pos)
SUBSTRING(str,pos,len)
SUBSTRING(str FROM pos FOR len)

Skilar undirstreng úr str frá og með staðsetningunni pos. Sé len sent inn er skilað len staf löngum undirstreng. FROM og FOR er stöðluð SQL málskipan.

```
SELECT SUBSTRING('Quadratically',5);             --> 'ratically'
SELECT SUBSTRING('foobarbar' FROM 4);            --> 'barbar'
SELECT SUBSTRING('Quadratically',5,6);           --> 'ratica'
```

SUBSTRING_INDEX(str,delim, count)

Skilar undirstreng úr strengnum str áður en count fjölda skilstafsins delim er náð. Sé count jákvæð tala er öllu vinstra megin við síðasta skilstafinn skilað talið vinstra megin frá, sé count neikvæð er öllu hægra megin við hann skilað talið frá hægri.

```
SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);        --> 'www.mysql'
SELECT SUBSTRING_INDEX('www.mysql.com', '.', -2);       --> 'mysql.com'
```

TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)

Skilar strengnum str þar sem öll remstr forskeyti og aftanvið-skeyti eru fjarlægð. Sé enginn af BOTH, LEADING, eða TRAILING sendir inn er BOTH sjálfgefið. Sé remstr ekki tilgreindur eru orðabil fjarlægð.

```
SELECT TRIM('  bar   ');                                --> 'bar'
SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');              --> 'barxxx'
SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');                 --> 'bar'
SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz');             --> 'barx'
```

UCASE(str)

Samheiti fallsins UPPER().

UNCOMPRESS(string_to_uncompress)

Afþjappar streng sem þjappaður var með COMPRESS(). MySQL þarf að vera þýddur með zlib safni.

```
SELECT UNCOMPRESS(COMPRESS('any string'));              --> 'any string'
SELECT UNCOMPRESS('any string');                        --> NULL
```

UNCOMPRESSED_LENGTH(compressed_string)

Skilar lengd óþjappaðs strengs áður en hann er þjappaður.

```
SELECT UNCOMPRESSED_LENGTH(COMPRESS(REPEAT('a',30)));           --> 30
```

UNHEX(str)

Virkar öfugt við fallið HEX(string), þ.e. túlkar allar samstæður af sextándutalnastöfum sem tölur og finnur bókstafi samkvæmt þeim talnakótum.

```
SELECT UNHEX('4D7953514C');                             --> 'MySQL'
SELECT 0x4D7953514C;                                    --> 'MySQL'
```

UPPER(str)

Skilar strengnum str þar sem öllum stöfum er breytt í hástafi.

```
SELECT UPPER('Heyrðu');                                 --> 'HEYRÐU'
```

Strengjasamanburður

MySQL umbreytir tölum sjálfkrafa í strengi og öfugt eftir þörfum. Þurfi að breyta tölu í streng sérstaklega, skyldi nota föllin CAST() eða CONCAT(). Sé strengjafalli gefinn tvíundarstrengur (e. binary string) er skila-strengurinn einnig þannig.

```
SELECT 1+'1';                                   --> 2
SELECT CONCAT(2,' test');                       --> '2 test'
SELECT 38.8, CAST(38.8 AS CHAR);                --> 38.8, '38.8'
```

expr LIKE pat [ESCAPE 'escape-char']

Mynstursamanburður með einföldum samanburði með SQL „reglulegri yrðingu“ (e. Regular expression). Skilar 1 (TRUE) eða 0 (FALSE). Ef expr eða pat er NULL, er skilað NULL.

Með lykilorðinu LIKE má nota eftirfarandi algildisstafi í mynstrinu: „_“ fyrir einn bókstaf og „%“ fyrir alla stafi. Sé þess óskað að finna algildisstafina sjálfa í mynstrinu er sett flóttatáknið „\“ framan við stafinn. Sé þess óskað að nota annarskonar flóttatákn má skilgreina það með ESCAPE hluta. Flóttatákns-dæmi: Vegna C grunns MySQL þarf að flótta-setja suma kóta þannig: ‘\n’ verður ‘\\n’. ‘\’, verður þannig ‘\\\’.

Algildisstafur	**Lýsing**
%	Finnur alla stafi, hvaða stafi sem er jafnvel núll stafi.
_	Finnur einn staf, hvaða staf sem er.
\%	Finnur eitt tilfelli af „%“ bókstafnum.
_	Finnur eitt tilfelli af „_“ bókstafnum.

Dæmi:

```
SELECT 'Grettir!' LIKE 'Grettir_';              --> 1
SELECT 'Grettir!' LIKE '%G%e%';                 --> 1
SELECT 'Grettir!' LIKE 'Grettir\_';             --> 0
SELECT 'Grettir_' LIKE 'Grettir\_';             --> 1
SELECT 'Grettir_' LIKE 'Grettir|_' ESCAPE '|';  --> 1
SELECT 'abc' LIKE 'ABC';                        --> 1
SELECT 'abc' LIKE BINARY 'ABC';                 --> 0
SELECT 10 LIKE '1%';                            --> 1
```

expr NOT LIKE pat [ESCAPE 'escape-char']

Þetta er hið sama og NOT (expr LIKE pat [ESCAPE 'escape-char']).

expr NOT REGEXP pat
expr NOT RLIKE pat

Hið sama og NOT (expr REGEXP pat).

expr REGEXP pat
expr RLIKE pat

Framkvæmir mynstur samanburð á strengja yrðingunni expr á móti mynstrinu í pat. Mynstrið getur verið ýtarleg regluleg yrðing (e. Extended regular expression). Málskipan reglulegra yrðinga er rædd ýtarlega í handbók MySQL.

Skilar 1 ef yrðingin expr er samstæð við mynstrið pat annars 0. Ef expr eða pat er NULL er skilað NULL. Við sumar aðstæður þarf að nota flóttatákn (e. Escape characters) í mynstrinu.

```
SELECT 'Grettir!' REGEXP 'g%e%%';                    --> 0
SELECT 'Grettir!' REGEXP '.*';                       --> 1
SELECT 'new*\n*line' REGEXP 'new\\*.\\*line';        --> 1
SELECT 'a' REGEXP 'A', 'a' REGEXP BINARY 'A';        --> 1  0
SELECT 'a' REGEXP '^[a-d]';                          --> 1
```

STRCMP(expr1,expr2)

Skilar 0 ef strengirnir eru eins, -1 ef fremri stikinn er styttri en sá síðari, annars er skilað 1. Samanburður fer eftir gildandi stafamengi.

```
SELECT STRCMP('text', 'text2');                      --> -1
SELECT STRCMP('text', 'text');                       --> 0
```

Talnaföll

Reiknivirkjar

Hefðbundnir reiknivirkjar eru til staðar. Sé reiknað með -, + eða * er niðurstaðan reiknuð sem BIGINT (64 bita) nákvæmni séu báðir þolendur heiltölur. Sé annar þolandi óformerkt heiltala og hin formerkt er niðurstaðan óformerkt.

+ Samlagning

```
SELECT 3+5;                                          --> 8
```

- Frádráttur

```
SELECT 3-5;                                          --> -2
```

- Einstæður mínus

Snýr formerki tölunnar sem hann er ritaður við úr jákvæðu í neikvætt. Sé virkinn notaður á BIGINT, er skilagildið BIGINT. Forðist notkun við aðstæður sem geta skilað 2^63!

```
SELECT - 2;                                          --> -2
```

* Margföldun

Gætið að dæminu: síðasta niðurstaðan er röng því hún er hærri en 64 bita gildissvið BIGINT þolir.

```
SELECT 3*5;                                  --> 15
```

```
SELECT 18014398509481984 *                  --> 3245185536584267267831560205762560
18014398509481984.0;

SELECT 18014398509481984 *                  --> 0
18014398509481984;
```

/ Deiling

Deiling með núlli skilar NULL niðurstöðu. Deiling framkvæmist því aðeins í BIGINT að niðurstöðu sé umbreytt í heiltölu.

```
SELECT 3/5;                                 --> 0.60

SELECT 102/(1-1);                           --> NULL
```

DIV Heiltöludeiling

Skilar niðurstöðu úr deilingu á heiltölum sem heiltölu. Notið MOD til að finna afganginn. Líkt FLOOR() fallinu en öruggara á BIGINT gildi.

```
SELECT 5 DIV 2;                             --> 2
```

Stærðfræðiföll

Öll stærðfræðiföll skila NULL ef skekkja kemur upp.

ABS(X)

Skilar algjöru gildi X.

```
SELECT ABS(2);                              --> 2

SELECT ABS(-32);                            --> 32
```

ACOS(X)

Skilar kósínus af X. Skilar NULL ef X er liggur ekki á gildissviðinu -1 til 1.

```
SELECT ACOS(1.0001);                        --> NULL

SELECT ACOS(0);                             --> 1.570796
```

ASIN(X)

Skilar arkarsínus af X. Skilar NULL ef X er ekki á gildissviðinu -1 til 1.

```
SELECT ASIN(0.2);                           --> 0.201358

SELECT ASIN('foo');                         --> 0.000000
```

ATAN(X)

Skilar arkartangans af X.

```
SELECT ATAN(2);                             --> 1.107149

SELECT ATAN(-2);                            --> -1.107149
```

ATAN(Y,X)
ATAN2(Y,X)

Skilar arkartangans af tveim breytum, X og Y. Svipað og reikna arkartangans af Y / X

```
SELECT ATAN(-2,2);                    --> -0.785398
SELECT ATAN2(PI(),0);                 --> 1.570796
```

CEILING(X)
CEIL(X)

Skilar lægsta heiltölugildinu sem ekki sé lægra en X. Skilagildinu er umbreytt í BIGINT.

```
SELECT CEILING(1.23);                 --> 2
SELECT CEIL(-1.23);                   --> -1
```

COS(X)

Skilar kósínus af X þar sem X er gefið upp í radíönum.

```
SELECT COS(PI());                     --> -1.000000
```

COT(X)

Skilar kótangens af X.

```
SELECT COT(12);                       --> -1.57267341
SELECT COT(0);                        --> NULL
```

CRC32(expr)

Reiknar hringferils-umfremd[70] (e. Cyclic Redundancy) og skilar 32 bita óformerktu gildi. Skilar NULL ef stikinn er NULL. Stikinn er ávallt meðtekinn sem strengur.

```
SELECT CRC32('MySQL');                --> 3259397556
```

DEGREES(X)

Skilar stikanum X, breytt úr radíönum í gráður.

```
SELECT DEGREES(PI());                 --> 180.000000
```

EXP(X)

Skilar gildinu e (náttúrulegur lógaritmi) í veldinu af X.

```
SELECT EXP(2);                        --> 7.389056
SELECT EXP(-2);                       --> 0.135335
```

FLOOR(X)

Skilar hæsta heiltölugildinu sem ekki sé hærra en X. Skilagildið er í BIGINT.

[70] http://www2.rad.com/networks/1994/err_con/crc.htm

```
SELECT FLOOR(1.23);                    --> 1
SELECT FLOOR(-1.23);                   --> -2
```

LN(X)

Skilar náttúrulegum lógaritma af X. Samheiti fallsins LOG(X).

```
SELECT LN(2);                          --> 0.693147
SELECT LN(-2);                         --> NULL
```

LOG(X)
LOG(B,X)

Útgáfan með einum stika skilar náttúrulegum lógaritma af X. Séu tveir stikar sendir er skilað náttúrulegum lógaritma af X út frá breytilegum grunni B. LOG(B,X) er jafngildi LOG(X)/LOG(B).

```
SELECT LOG(2);                         --> 0.693147
SELECT LOG(-2);                        --> NULL
SELECT LOG(2,65536);                   --> 16.000000
SELECT LOG(1,100);                     --> NULL
```

LOG2(X)

Skilar lógaritma af X í tvíundargildi. Hentugt til að finna út hve marga bita þarf til að geyma tölu. Jafngild LOG(X)/LOG(2).

```
SELECT LOG2(65536);                    --> 16.000000
SELECT LOG2(-100);                     --> NULL
```

LOG10(X)

Skilar náttúrulegum lógaritma af X í tugagildi.

```
SELECT LOG10(2);                       --> 0.301030
SELECT LOG10(-100);                    --> NULL
```

MOD(N,M)
N % M
N MOD M

Afgangur heiltöludeilingar, svipað % deilivirkjanum í C og Java. Skilar afganginum af N deilt með M.

```
SELECT MOD(234, 10);                   --> 4
SELECT 253 % 7;                        --> 1
SELECT MOD(29,9);                      --> 2
SELECT 29 MOD 9;                       --> 2
```

PI()

Skilar PÍ. Fjöldi birtra aukastafa er fimm, innri nákvæmni er þó í tvöföldu-kommutölutagi.

```
SELECT PI();                                    --> 3.141593
SELECT PI()+0.000000000000000000;               --> 3.141592653589793116
```

POW(X,Y)
POWER(X,Y)

Skilar X í Y veldinu.

```
SELECT POW(2,2);                                --> 4.000000
SELECT POW(2,-2);                               --> 0.250000
```

RADIANS(X)

Skilar X, breytt úr gráðum í radíana.

```
SELECT RADIANS(90);                             --> 1.570796
```

RAND()
RAND(N)

Skilar slembitölu af kommutölutagi á gildissviðinu 0 til 1.0. Sé heiltölustikinn N tilgreindur er hann notaður sem sæðistala sem hentar fyrir aðstæður sem þarf endurtekningu á sama gildinu. Ekki er hægt að nota í ORDER BY dálki sem inniheldur slembitölur.

Í úttaki SELECT aðgerðar má raða eftir slembitölu, samanber:

```
SELECT * FROM tbl_name
ORDER BY RAND();
```

Einnig má takmarka fjölda niðurstaðna með LIMIT. Hentugt til slembiúttaks takmarkaðs línufjölda úr töflu.

```
SELECT * FROM table1, table2
WHERE a=b AND c<d ORDER BY RAND() LIMIT 1000;
```

```
SELECT RAND();                                  --> 0.9233482386203
SELECT RAND(20);                                --> 0.15888261251047
SELECT RAND(20);                                --> 0.15888261251047
SELECT RAND();                                  --> 0.63553050033332
```

ROUND(X)
ROUND(X,D)

Skilar stikanum X námunduðum að næstu heiltölu. Séu tveir stikar sendir er X námundað með D aukastöfum. Sé D neikvæð tala er heiltöluhlutinn núllaður. Vegna grundvöllunar MySQL á C safninu er mælt með notkun TRUNCATE() og FLOOR() við sumar aðstæður.

```
SELECT ROUND(-1.23);                            --> -1
SELECT ROUND(-1.58);                            --> -2
SELECT ROUND(1.58);                             --> 2
SELECT ROUND(1.298, 1);                         --> 1.3
SELECT ROUND(1.298, 0);                         --> 1
```

```
SELECT ROUND(23.298, -1);                                    --> 20
```

SIGN(X)

Skilar formerki stikans sem -1, 0, eða 1, eftir því hvort X sé neikvæð, núll eða jákvæð.

```
SELECT SIGN(-32);                                            --> -1
SELECT SIGN(234);                                            --> 1
```

SIN(X)

Skilar sínus af X þar sem X er í radíönum.

```
SELECT SIN(PI());                                            --> 0.000000
```

SQRT(X)

Skilar óneikvæðri kvaðratrót af X.

```
SELECT SQRT(4);                                              --> 2.000000
SELECT SQRT(20);                                             --> 4.472136
```

TAN(X)

Skilar tangens af X, þar sem X er í radíönum.

```
SELECT TAN(PI()+1);                                          --> 1.557408
```

TRUNCATE(X,D)

Skilar tölunni X, klippt í D marga aukastafi. Sé D gefið sem núll mun skilagildið vera án kommu og brotahluta (aukastafa). Sé D neikvæð er heiltöluhlutinn núllaður. Allar tölur eru rúnnaðar í áttina að núll.

```
SELECT TRUNCATE(1.223,1);                                    --> 1.2
SELECT TRUNCATE(1.999,0);                                    --> 1
SELECT TRUNCATE(-1.999,1);                                   --> -1.9
SELECT TRUNCATE(122,-2);                                     --> 100
```

Tíðaföll

Hér eru útskýrð tíðaföll sem vinna á dags- og tímasetningum. Hér er dæmi um fyrirspurn sem skilar öllum færslum þar sem dálkurinn `date_col` hefur gildi innan síðustu 30 daga, einnig er skilað dagsetningum í framtíðinni:

```
SELECT something FROM tbl_name
WHERE DATE_SUB(CURDATE(),INTERVAL 30 DAY) <= date_col;
```

Föll sem vænta dagsetninga taka einnig við tímasetningum en sniðgengur þær, föll sem vænta tímasetninga taka sömuleiðis við dagsetningum en sniðganga þær.

Föll sem skila núgildum (e. Current) degi eða tíma eru aðeins metin (eða innt)einu sinni í fyrirspurn, jafnvel þó fyrirspurnin þurfi að skila mörgum færslum.

Skilagildi tíðafalla sem skila dagsetningum skila þeim fullrituðum. Sé dagsetning með núll-gildi eða ólöglegum degi s.s. '2001-11-00' má vænta 0 sem skilagildi.

ADDDATE(date,INTERVAL expr type)
ADDDATE(expr,days)

Þegar INTERVAL breytan er send sem annar stiki er ADDDATE() samheiti fyrir DATE_ADD(). Skylt fall, SUBDATE() er samheiti fyrir DATE_SUB(). Sjá lýsingu þessara falla.

```
SELECT ADDDATE('1998-01-02', INTERVAL 31 DAY);          --> '1998-02-02'
SELECT ADDDATE('1998-01-02', 31);                       --> '1998-02-02'
```

ADDTIME(expr,expr2)

Stikinn expr2 er bætt við expr og niðurstöðunni skilað. Stikinn expr sé dagsetning og expr2 sé tímasegð.

```
SELECT ADDTIME('1997-12-31 23:59:59.999999','1          --> '1998-01-02
1:1:1.000002');                                        01:01:01.000001'
SELECT ADDTIME('01:00:00.999999', '02:00:00.999998');   --> '03:00:01.999997'
```

CURDATE()

Skilar dagréttum (e. Updated) tíma sem gildi í sniðinu 'YYYY-MM-DD' eða YYYYMMDD eftir því hvort fallið sé notað í strengja- eða talna-samhengi.

```
SELECT CURDATE();                                       --> '1997-12-15'
SELECT CURDATE() + 0;                                   --> 19971215
```

CURRENT_DATE
CURRENT_DATE()

Samheiti fallsins CURDATE().

CURTIME()

Skilar núgildum tíma sem gildi í sniðinu 'HH:MM:SS' eða HHMMSS eftir því hvort fallið er notað í strengja- eða talna-samhengi.

```
SELECT CURTIME();                                       --> '23:50:26'
SELECT CURTIME() + 0;                                   --> 235026
```

CURRENT_TIME
CURRENT_TIME()

Samheiti fyrir fallið CURTIME().

CURRENT_TIMESTAMP
CURRENT_TIMESTAMP()

Samheiti fyrir fallið NOW().

DATE(expr)

Dregur út og skilar dagsetningunni úr dags-segðinni eða dags-tíma-segðinni expr.

```
SELECT DATE('2003-12-31 01:02:03');                     --> '2003-12-31'
```

DATEDIFF(expr,expr2)

Skilar fjölda daga á milli upphafs dagsins expr og loka dagsins expr2. Báðir stikarnir séu dags- eða tímagildi og aðeins dagsetninga hlutinn er notaður.

```
SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30');     --> 1
SELECT DATEDIFF('1997-11-30 23:59:59','1997-12-31');     --> -31
```

DATE_ADD(date,INTERVAL expr type)
DATE_SUB(date,INTERVAL expr type)

Þessi föll framkvæma útreikning á dögum. Stikinn date sé DATETIME eða DATE tegund sem tilgreinir byrjunardag. Stikinn expr er segð sem tilgreinir millibil eða þrep sem tekin eru til að bæta við dögum eða draga frá.

Stikinn expr er strengur sem má vera formerktur með mínus „-“ til frádráttar. Breytan type er lykilorð sem tilgreinir þá tegund þrepa sem notuð skulu við frádrátt/samlagningu. Lykilorðin eru ekki hástafanæm. Eftirfarandi tafla lýsir tegundum lykilorða sem gild eru fyrir type og expr.

type **lykilorð**	**Leyft** expr **snið**
MICROSECOND	MICROSECONDS
SECOND	SECONDS
MINUTE	MINUTES
HOUR	HOURS
DAY	DAYS
WEEK	WEEKS
MONTH	MONTHS
QUARTER	QUARTERS
YEAR	YEARS
SECOND_MICROSECOND	'SECONDS.MICROSECONDS'
MINUTE_MICROSECOND	'MINUTES.MICROSECONDS'
MINUTE_SECOND	'MINUTES:SECONDS'
HOUR_MICROSECOND	'HOURS.MICROSECONDS'
HOUR_SECOND	'HOURS:MINUTES:SECONDS'
HOUR_MINUTE	'HOURS:MINUTES'
DAY_MICROSECOND	'DAYS.MICROSECONDS'
DAY_SECOND	'DAYS HOURS:MINUTES:SECONDS'
DAY_MINUTE	'DAYS HOURS:MINUTES'
DAY_HOUR	'DAYS HOURS'
YEAR_MONTH	'YEARS-MONTHS'

MySQL leyfir ýmsan rithátt á expr stikanum og taflan er því aðeins viðmiðun. Sé date stikinn aðeins með DATE gildi er skilagildið DATE gildi annars DATETIME.

```
SELECT INTERVAL 1 DAY + '1997-12-31';                    --> '1998-01-01'

SELECT '1998-01-01' - INTERVAL 1 SECOND;                 --> '1997-12-31 23:59:59'

SELECT DATE_ADD('1997-12-31 23:59:59',                   --> '1998-01-01 00:01:00'
  INTERVAL '1:1' MINUTE_SECOND);

SELECT DATE_ADD('1998-01-01 00:00:00',                   --> '1997-12-30 14:00:00'
  INTERVAL '-1 10' DAY_HOUR);

SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);          --> '1997-12-02'
```

Ef þrepagildið sem er tilgreint er of stutt og vanti hluta þess reynir MySQL að giska á fullritun þess. Ef t.d. er tilgreint DAY_SECOND í type, þá er þess vænst að expr innihaldi daga, klukkustundir, mínútur og sekúndur. Ef þá er ritað '1:10' gerir miðlarinn ráð fyrir að í raun sé spurt um '1:10' MINUTE_SECOND. Ef notuð eru illa skilgreind dagagildi er skilagildið NULL. Ef bætt er við MONTH, YEAR_MONTH, eða YEAR þannig að niðurstaðan sé dagsgildi hærra en hæsta mögulega gildi er niðurstaðan hæsta mögulega gildi tegundarinnar.

```
SELECT DATE_ADD('1999-01-01', INTERVAL 1 DAY);           --> '1999-01-02'

SELECT DATE_ADD('1999-01-01', INTERVAL 1 HOUR);          --> '1999-01-01 01:00:00'

SELECT DATE_ADD('1998-01-30', INTERVAL 1 MONTH);         --> '1998-02-28'
```

DATE_FORMAT(date,format)

Sníður date gildið til í samræmi við format strenginn. Eftirfarandi lykla má nota í format strengnum.

Lykill	**Lýsing**
%a	Stytt vikudagsheiti (Sun..Sat)
%b	Stytt mánaðarheiti (Jan..Dec)
%c	Númer mánaðar í tölugildi (0..12)
%D	Dagur mánaðar í tölugildi með enskum rithætti (0th, 1st, 2nd, 3rd, ...)
%d	Tölugildi mánaðardags. (00..31)
%e	Tölugildi mánaðardags (0..31)
%f	Míkrósekúndur (000000..999999)
%H	Klukkustund (00..23)
%h	Klukkustund (01..12)
%I	Klukkustund (01..12)
%i	Tölugildi mínútna (00..59)
%j	Dagur ársins, tölugildi (001..366)
%k	Klukkustund (0..23)
%l	Klukkustund (1..12)
%M	Heiti mánaðar (January..December)
%m	Tölugildi mánaðar (00..12)
%p	AM eða PM

%r	Tími, 12-stunda snið (hh:mm:ss með AM eða PM)
%S	Sekúndur (00..59)
%s	Sekúndur (00..59)
%T	Tími, 24 stunda snið (hh:mm:ss)
%U	Vika (00..53), með sunnudegi sem fyrsta vikudegi
%u	Vika (00..53), mánudegi sem fyrst vikudegi
%V	Vika (01..53), með sunnudegi sem fyrsta vikudegi, notað með %X
%v	Vika (01..53), þar sem mánudagur er fyrsti vikudagur, notað með %x
%W	Heiti vikudags (Sunday..Saturday)
%w	Vikudagur (0=sunnudagur .. 6=laugardagur)
%X	Ártal viku með sunnudegi sem fyrsta vikudegi, tala í fjórum tölustöfum. Notað með %V
%x	Ártal viku með mánudegi sem fyrsta vikudegi, tala í fjórum tölustöfum. Notað með %v
%Y	Ártal, tala með fjórum tölustöfum
%y	Ártal, tala með tveim tölustöfum
%%	Lesgildið '%'

Allir aðrir stafir eru afritaðir án túlkunar. Stafurinn '%' er nauðsynlegur framan við lyklana. Ástæðan fyrir því að gildissvið mánaða og daga er talið frá núlli er sú að MySQL leyfir ófullritaða daga s.s. '2004-00-00'

```
SELECT DATE_FORMAT('1997-10-04 22:23:00',        --> 'Saturday October 1997'
  '%W %M %Y');

SELECT DATE_FORMAT('1997-10-04 22:23:00',        --> '22:23:00'
  '%H:%i:%s');

SELECT DATE_FORMAT('1997-10-04 22:23:00',        --> '4th 97 Sat 04 10 Oct 277'
  '%D %y %a %d %m %b %j');

SELECT DATE_FORMAT('1999-01-01',                 --> '1998 52'
  '%X %V');
```

DAY(date)

Samheiti fallsins DAYOFMONTH().

DAYNAME(date)

Skilar heiti vikudagsins date.

```
SELECT DAYNAME('1998-02-05');                    --> 'Thursday'
```

DAYOFMONTH(date)

Skilar tölugildi dags í mánuðinum date, á gildissviðinu 1 til 31.

```
SELECT DAYOFMONTH('1998-02-03');                 --> 3
```

DAYOFWEEK(date)

Skilar sætisvísi vikudagsins date þar sem 1=sunnudagur, 2=mánudagur, .., 7=laugardagur. Þetta er ODBC samhæfing.

```
SELECT DAYOFWEEK('1998-02-03');                  --> 3
```

DAYOFYEAR(date)

Skilar sætisnúmeri dagsins í date innan árs á gildissviðinu 1 til 366.

```
SELECT DAYOFYEAR('1998-02-03');                    --> 34
```

EXTRACT(type FROM date)

Virkar svipað og DATE_ADD() eða DATE_SUB(), en dregur út hluta dagsins date frekar en að framkvæma útreikninga.

```
SELECT EXTRACT(YEAR FROM '1999-07-02');                    --> 1999
SELECT EXTRACT(YEAR_MONTH FROM '1999-07-02 01:02:03');     --> 199907

SELECT EXTRACT(DAY_MINUTE FROM '1999-07-02 01:02:03');     --> 20102
```

FROM_DAYS(N)

Sé gefin upp tala er skilað DATE gildi samkvæmt tölugildinu. Fallið getur ekki gert ráð fyrir dögunum sem hurfu þegar Gregoríska dagatalið var tekið í notkun árið 1582.

```
SELECT FROM_DAYS(729669);                          --> '1997-10-07'
```

FROM_UNIXTIME(unix_timestamp)
FROM_UNIXTIME(unix_timestamp,format)

Skilar UNIX tímastimpli stikans unix_timestamp sem gildi á sniðinu 'YYYY-MM-DD HH:MM:SS' eða YYYYMMDDHHMMSS eftir samhengi talna eða strengja. Sé stikinn format sendur inn er skilagildið sniðið í samræmi við hann. Sníða má format á sama hátt og fyrir DATE_FORMAT() fallið.

```
SELECT FROM_UNIXTIME(875996580);              --> '1997-10-04 22:23:00'
SELECT FROM_UNIXTIME(875996580) + 0;          --> 19971004222300
SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(),        --> '2003 6th August 06:22:58 2003'
  '%Y %D %M %h:%i:%s %x');
```

GET_FORMAT(DATE|TIME|TIMESTAMP, 'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL')

Skilar virku sniði fyrir tegundina sem er fremri stikinn eftir æskilegum rithætti aftari stikans. Niðurstaðan getur verið ein af fimmtán mögulegum (ISO snið er ISO 9075, ekki ISO 8601):

Kallað á fallið	**Skilagildi**
GET_FORMAT(DATE,'USA')	'%m.%d.%Y'
GET_FORMAT(DATE,'JIS')	'%Y-%m-%d'
GET_FORMAT(DATE,'ISO')	'%Y-%m-%d'
GET_FORMAT(DATE,'EUR')	'%d.%m.%Y'
GET_FORMAT(DATE,'INTERNAL')	'%Y%m%d'
GET_FORMAT(TIMESTAMP,'USA')	'%Y-%m-%d-%H.%i.%s'
GET_FORMAT(TIMESTAMP,'JIS')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(TIMESTAMP,'ISO')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(TIMESTAMP,'EUR')	'%Y-%m-%d-%H.%i.%s'
GET_FORMAT(TIMESTAMP,'INTERNAL')	'%Y%m%d%H%i%s'
GET_FORMAT(TIME,'USA')	'%h:%i:%s %p'

```
GET_FORMAT(TIME,'JIS')                    '%H:%i:%s'
GET_FORMAT(TIME,'ISO')                    '%H:%i:%s'
GET_FORMAT(TIME,'EUR')                    '%H.%i.%S'
GET_FORMAT(TIME,'INTERNAL')               '%H%i%s'
```

Dæmi:

```
SELECT DATE_FORMAT('2003-10-03', GET_FORMAT(DATE,'EUR'));        --> '03.10.2003'
SELECT STR_TO_DATE('10.31.2003', GET_FORMAT(DATE,'USA'));        --> 2003-10-31
```

HOUR(time)

Skilar klukkustundinni í time. Gildissviðið er frá 0 til 23 fyrir „tími dags“en getur verið mun hærra þ.e.s. TIME hefur hátt gildissvið fyrir klukkustundir.

```
SELECT HOUR('10:05:03');                                  --> 10
SELECT HOUR('272:59:59');                                 --> 272
```

LAST_DAY(date)

Tekur við DATE eða DATETIME gildi og skilar síðasta degi mánaðar í þeim mánuði. Skilar NULL ef gildið er ómerkt.

```
SELECT LAST_DAY('2003-02-05');                            --> '2003-02-28'
SELECT LAST_DAY('2004-02-05');                            --> '2004-02-29'
SELECT LAST_DAY('2004-01-01 01:01:01');                   --> '2004-01-31'
SELECT LAST_DAY('2003-03-32');                            --> NULL
```

LOCALTIME
LOCALTIME()

Samheiti fallsins NOW().

LOCALTIMESTAMP
LOCALTIMESTAMP()

Samheiti fallsins NOW().

MAKEDATE(year,dayofyear)

Skilar dagsetningu innan ársins year í mánuði og degi samkvæmt dayofyear sem verður að vera hærri en núll annars er skilað NULL.

```
SELECT MAKEDATE(2001,31), MAKEDATE(2001,32);      --> '2001-01-31', '2001-02-01'
SELECT MAKEDATE(2001,365), MAKEDATE(2004,365);    --> '2001-12-31', '2004-12-30'
SELECT MAKEDATE(2001,0);                          --> NULL
```

MAKETIME(hour,minute,second)

Skilar tímagildi reiknuðu út frá hour, minute, og second stikunum.

```
SELECT MAKETIME(12,15,30);                                --> '12:15:30'
```

MICROSECOND(expr)

Skilar míkrósekúndu út frá yrðingunni expr sem sé þá TIME eða DATETIME yrðing. Gildissvið skilagildis er frá 0 til 999999.

```
SELECT MICROSECOND('12:00:00.123456');                  --> 123456
SELECT MICROSECOND('1997-12-31 23:59:59.000010');       --> 10
```

MINUTE(time)

Skilar mínútunni í time á gildissviðinu 0 til 59.

```
SELECT MINUTE('98-02-03 10:05:03');                     --> 5
```

MONTH(date)

Skilar mánuðinum í date á gildissviðinu 1 til 12.

```
SELECT MONTH('1998-02-03');                             --> 2
```

MONTHNAME(date)

Skilar mánaðarheiti (á ensku) mánaðarins í date.

```
SELECT MONTHNAME('1998-02-05');                         --> 'February'
```

NOW()

Skilar núgildri (dagréttri) dags- og tímasetningu á sniðinu 'YYYY-MM-DD HH:MM:SS' eða YYYYMMDDHHMMSS eftir því hvort notkun sé í strengja- eða talnasamhengi.

```
SELECT NOW();                                           --> '1997-12-15 23:50:26'
SELECT NOW() + 0;                                       --> 19971215235026
```

PERIOD_ADD(P,N)

Bætir N mánuðum við tímabilið P á sniðinu YYMM or YYYYMM. Skilar gildi í sniðinu YYYYMM. Tímabilið P er ekki tíðagildi.

```
SELECT PERIOD_ADD(9801,2);                              --> 199803
```

PERIOD_DIFF(P1,P2)

Skilar mánaðafjölda á milli tíðanna P1 og P2 sem skyldu báðir vera á sniðinu YYMM eða YYYYMM en eru ekki tíðagildi.

```
SELECT PERIOD_DIFF(9802,199703);                        --> 11
```

QUARTER(date)

Skilar fjórðungi ársins í date á gildissviðinu 1 til 4.

```
SELECT QUARTER('98-04-01');                             --> 2
```

SECOND(time)

Skilar sekúndunni í time á gildissviðinu 0 til 59.

```
SELECT SECOND('10:05:03');                                --> 3
```

SEC_TO_TIME(seconds)

Skilar stikanum seconds sem klukkustundir, mínútur og sekúndur, á sniðinu 'HH:MM:SS' eða HHMMSS eftir samhengi við strengja- eða talnagildis notkun.

```
SELECT SEC_TO_TIME(2378);                                 --> '00:39:38'

SELECT SEC_TO_TIME(2378) + 0;                             --> 3938
```

STR_TO_DATE(str,format)

Virkar öfugt við DATE_FORMAT() fallið. Sendur er stikinn str með gildi og æskilegu sniði í stikanum format. Skilað er DATETIME gildi. Gildið í str sé ritað í samræmi við sniðið í format. Sjá fallið DATE_FORMAT() varðandi mögulega lykla í format. Ef str inniheldur ólöglega tíð er skilagildið NULL.

```
SELECT STR_TO_DATE('03.10.2003 09.20',                --> '2003-10-03 09:20:00'
  '%d.%m.%Y %H.%i');

SELECT STR_TO_DATE('10arp', '%carp');                 --> '0000-10-00 00:00:00'

SELECT STR_TO_DATE('2003-15-10 00:00:00',             --> NULL
  '%Y-%m-%d %H:%i:%s');
```

SUBDATE(date,INTERVAL expr type)
SUBDATE(expr,days)

Sé fallið vakið með INTERVAL lykilorðinu er það samheiti fallsins DATE_SUB(). Sjá einnig umfjöllun um DATE_ADD().

```
SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);           --> '1997-12-02'

SELECT SUBDATE('1998-01-02', INTERVAL 31 DAY);            --> '1997-12-02'

SELECT SUBDATE('1998-01-02 12:00:00', 31);                --> '1997-12-02 12:00:00'
```

SUBTIME(expr,expr2)

Dregur út expr2 úr yrðingunni expr og skilar niðurstöðunni. Stikinn expr sé DATE eða DATETIME og expr2 sé tímasetning.

```
SELECT SUBTIME('1997-12-31 23:59:59.999999',          --> '1997-12-30
  '1 1:1:1.000002');                                 22:58:58.999997'

SELECT SUBTIME('01:00:00.999999',                      --> '-00:59:59.999999'
  '02:00:00.999998');
```

SYSDATE()

Samheiti fallsins NOW().

TIME(expr)

Dregur út tímahlutann úr stikanum expr sem annaðhvort sé TIME eða DATETIME yrðing.

```
SELECT TIME('2003-12-31 01:02:03');                       --> '01:02:03'
```

TIMEDIFF(expr,expr2)

Skilar út tímanum á milli byrjunartíma (expr) og lokatíma (expr2). Báðir stikar séu af sömu tegund og mega vera TIME eða DATETIME yrðingar.

```
SELECT TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001');      --> '-00:00:00.000001'
```

TIMESTAMP(expr)
TIMESTAMP(expr,expr2)

Skilar stikanum expr sem DATETIME gildi. Séu tveir stikar sendir er þeim síðari bætt við þann fyrri.

```
SELECT TIMESTAMP('2003-12-31');                          --> '2003-12-31 00:00:00'
SELECT TIMESTAMP('2003-12-31 12:00:00','12:00:00');      --> '2004-01-01 00:00:00'
```

TIMESTAMPADD(interval,int_expr,datetime_expr)

Bætir heiltölugildinu int_expr við DATE eða DATETIME yrðinguna í datetime_expr. Þrepið (interval) fyrir int_expr getur verið einhver af FRAC_SECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, eða YEAR. Bæta má lykilorðinu SQL_TSI_ við þrepið s.s. SQL_TSI_DAY.

```
SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02');          --> '2003-01-02 00:01:00'
SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');            --> '2003-01-09'
```

TIMESTAMPDIFF(interval,datetime_expr1,datetime_expr2)

Skilar mismuninum á date og/eða datetime segðunum í heiltölu. Stikinn interval skilgreinir þá tímaeiningu sem notuð er á skilagildið. Lögleg gildi fyrir interval eru útskýrð í umfjöllun um TIMESTAMPADD() fallið.

```
SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');        --> 3
SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01');         --> -1
```

TIME_FORMAT(time,format)

Virkar svipað og DATE_FORMAT() nema format strengurinn má aðeins innihalda snið-lykla fyrir klukkustundir, mínútur og sekúndur, að öðrum kosti er NULL eða 0 skilað.

```
SELECT TIME_FORMAT('100:00:00', '%H %k %h %I %l');       --> '100 100 04 04 4'
```

TIME_TO_SEC(time)

Skilar stikanum time breytt í sekúndur.

```
SELECT TIME_TO_SEC('22:23:00');                          --> 80580
SELECT TIME_TO_SEC('00:39:38');                          --> 2378
```

TO_DAYS(date)

Sé uppgefin dagsetningin date er skilað tölugildi/númeri dags frá árinu 0. Glatar þeim dögum sem týndust þegar breytt var úr Júlíanska yfir í Gregoríska dagatalið og virkar því óskilgreint fyrir þann tíma. Varist að nota með tveggja tölustafa ártölum.

```
SELECT TO_DAYS(950501);                                  --> 728779
```

```
SELECT TO_DAYS('1997-10-07');                                    --> 729669
SELECT TO_DAYS('1997-10-07'), TO_DAYS('97-10-07');       --> 729669, 729669
```

UNIX_TIMESTAMP()
UNIX_TIMESTAMP(date)

Skilar UNIX tímastimpli (sekúndur frá '1970-01-01 00:00:00' GMT) sem óformerkt heiltala. Sé stikinn date sendur inn táknar tölugildið þann dag. Stikinn date má vera DATE eða DATETIME strengur, TIMESTAMP gildi eða tala á sniðinu YYMMDD eða YYYYMMDD. Sé date stikinn TIMESTAMP er honum skilað til baka.

```
SELECT UNIX_TIMESTAMP();                                    --> 882226357
SELECT UNIX_TIMESTAMP('1997-10-04 22:23:00');            --> 875996580
```

UTC_DATE
UTC_DATE()

Skilar núgildri UTC dagsetningu á sniðinu 'YYYY-MM-DD' eða YYYYMMDD eftir strengja- eða talna-samhengi.

```
SELECT UTC_DATE(), UTC_DATE() + 0;                    --> '2003-08-14', 20030814
```

UTC_TIME
UTC_TIME()

Skilar núgildri UTC tímasetningu sem gildi á sniðinu 'HH:MM:SS' eða HHMMSS eftir strengja eða talnasamhengi.

```
SELECT UTC_TIME(), UTC_TIME() + 0;                          --> '18:07:53', 180753
```

UTC_TIMESTAMP
UTC_TIMESTAMP()

Skilar núgildri dags- og tímasetningu sem gildi á sniðinu 'YYYY-MM-DD HH:MM:SS' eða YYYYMMDDHHMMSS eftir strengja eða talnasamhengi.

```
SELECT UTC_TIMESTAMP(), UTC_TIMESTAMP() + 0;        --> '2003-08-14 18:08:04',
                                                    20030814180804
```

WEEK(date[,mode])

Skilar vikunúmeri vikunnar í date. Sé stikinn mode tilgreindur má velja hvort mánudagur eða sunnudagur séu fyrstu dagar vikunnar og einnig hvort skilagildið sé á gildissviðinu 0 – 53 eða 1 til 52. Sé mode sleppt er sjálfgildið í kerfisbreytunni `default_week_format` notuð. Ef date er í síðustu viku ársins á undan skilar fallið 0 ef ekki er notað 2, 3, 6 eða 7 í mode. Að auki má nota fallið YEARWEEK().

mode stikinn	**Merking**
0	Vikan hefst á sunnudegi, talið frá 0 til 53, vika 1 er sú fyrsta sem hefst á þessu ári.
1	Vikan hefst á mánudegi, talið frá 0 til 53, vika 1 er sú fyrsta sem hefur fleiri en þrjá daga í

	þessu ári.
2	Vikan hefst á sunnudegi, talið frá 1 til 53, vika 1 er sú fyrsta sem hefst á þessu ári.
3	Vikan hefst á mánudegi, talið frá 1 til 53, vika 1 er sú fyrsta sem hefur fleiri en þrjá daga í þessu ári.
4	Vikan hefst á sunnudegi, talið frá 0 til 53, vika 1 er sú fyrsta sem hefur fleiri en þrjá daga í þessu ári.
5	Vikan hefst á mánudegi, talið frá 0 til 53, vika 1 er sú fyrsta sem hefst á þessu ári.
6	Vikan hefst á sunnudegi, talið frá 1 til 53, vika 1 er sú fyrsta sem hefur fleiri en þrjá daga í þessu ári.
7	Vikan hefst á mánudegi, talið frá 1 til 53, vika 1 er fyrsta vikan sem hefst á þessu ári.

Dæmi:

```
SELECT WEEK('1998-02-20');                          --> 7
SELECT WEEK('1998-02-20',0);                        --> 7
SELECT WEEK('1998-02-20',1);                        --> 8
SELECT WEEK('1998-12-31',1);                        --> 53
SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0);    --> 2000, 0
SELECT WEEK('2000-01-01',2);                        --> 52
```

WEEKDAY(date)

Skilar sætisvísi vikudagsins, talið frá núlli (0 = mánudagur, 1 = þriðjudagur , ... 6 = sunnudagur).

```
SELECT WEEKDAY('1998-02-03 22:23:00');              --> 1
SELECT WEEKDAY('1997-11-05');                       --> 2
```

WEEKOFYEAR(date)

Skilar tölusæti viku innan dagatalsins sem tala á sviðinu 1 til 53.

```
SELECT WEEKOFYEAR('1998-02-20');                    --> 8
```

YEAR(date)

Skilar árinu í date á talnabilinu 1000 til 9999.

```
SELECT YEAR('98-02-03');                            --> 1998
```

YEARWEEK(date)
YEARWEEK(date,start)

Skilar ártali og vikunúmeri samkvæmt date. Stikinn start virkar á sama hátt og mode í WEEK().

```
SELECT YEARWEEK('1987-01-01');                      --> 198653
```

Umbreytiföll

CAST(expr AS type)
CONVERT(expr,type)
CONVERT(expr USING transcoding_name)

Föllin CAST() og CONVERT() geta tekið við gildi af einni tegund og framleitt gildi af annarri tegund. CAST() og CONVERT(... USING ...) eru staðlaðar SQL málskipanir. Convert() án USING er ODBC málskipan. Sé CAST() notað á DATE, DATETIME, eða TIME dálka eru þeir aðeins skilgreindir sem sú tegund en gildum þeirra ekki breytt, nema þegar því er skilað til notandans í fyrirspurn.

Tegundir geta verið eftirfarandi:

- BINARY
- CHAR
- DATE
- DATETIME
- SIGNED [INTEGER]
- TIME
- UNSIGNED [INTEGER]

```
SELECT CONVERT('abc' USING utf8);                    Breytir abc í utf8
                                                     stafamengið.

CREATE TABLE new_table                               Smíðar nýja töflu með
  SELECT CAST('2000-01-01' AS DATE);                 tíðadálk.

SELECT enum_col FROM tbl_name ORDER BY               Raðar ENUM dálki eftir orða-
CAST(enum_col AS CHAR);                              bókarröðun, sem annars fer
                                                     fram eftir sætisvísum
                                                     (töluröð.

SELECT CAST(NOW() AS DATE);                          --> 2003-05-26

SELECT CAST(1-2 AS UNSIGNED)                         -> 18446744073709551615

SELECT CAST(CAST(1-2 AS UNSIGNED) AS SIGNED);        --> -1
```

Ýmis föll

Bitaföll

MySQL notar 64 bita BIGINT útreikning í allri bita-meðhöndlun. Niðurstaðan úr öllum eftirfarandi dæmum er 64 bita.

| Bitwise OR:

```
SELECT 29 | 15;                                       -> 31
```

& Bitwise AND:

```
SELECT 29 & 15;                                       -> 13
```

^ Bitwise XOR

```
SELECT 1 ^ 1;                    -> 0
SELECT 1 ^ 0;                    -> 1
SELECT 11 ^ 3;                   -> 8
```

<< Ýtir BIGINT tölugildi til vinstri

```
SELECT 1 << 2;                   -> 4
```

>> Ýtir BIGINT tölugildi til hægri

```
SELECT 4 >> 2;                   -> 1
```

~ Snýr öllum bitum við

```
SELECT 5 & ~1;                   -> 4
```

BIT_COUNT(N)

Skilar fjölda þeirra bita sem eru Á í stikanum N

```
SELECT BIT_COUNT(29);            -> 4
```

Dulkóðunarföll

Föll sem dulkóða og afkóða gildi. Sé ætlunin að geyma slík gildi er betra að nota BLOB frekar en CHAR eða VARCHAR dálka sem t.d. geta eytt slef-bilum.

AES_ENCRYPT(str,key_str)
AES_DECRYPT(crypt_str,key_str)

Leyfir dulkóðun og afkóðun með lykli af algríms-staðlinum AES (Advanced Encryption Standard) sem áður var nefndur „Rijndael." Dulkóðun fer fram með 128-bita lykli.

Nota má AES föll til að geyma dulkóðuð gögn þannig:

```
INSERT INTO t VALUES (1,AES_ENCRYPT('text','password'));
```

Geyma má lykilorð miðlaramegin þannig (svo ekki þurfi að senda þau oftar en einu sinni):

```
SELECT @password:='my password';
INSERT INTO t VALUES (1,AES_ENCRYPT('text',@password));
```

DECODE(crypt_str,pass_str)

Afkóðar dulkóðaða strenginn crypt_str með pass_str sem lykilorð. Stikinn crypt_str skyldi vera skila-strengur frá ENCODE().

ENCODE(str,pass_str)

Dulkóðar strenginn str með pass_str sem lykilorð. Notið DECODE() til að afkóða. Ef geyma skal gildið ætti að nota BLOB tegund.

DES_DECRYPT(crypt_str[,key_str])

Afkóðar streng sem var dulkóðaður með DES_ENCRYPT(). Komi upp skekkja er skilað NULL. Þetta fall virkar því aðeins að MySQL hafi verið stilltur fyrir SSL stuðning.

DES_ENCRYPT(str[,(key_num|key_str)])

Dulkóðar streng í samræmi við uppgefinn lykil key_num eða key_str, samkvæmt „Triple-DES“ algríminu. Komi upp skekkja er skilað NULL. Þetta fall virkar því aðeins að MySQL hafi verið stilltur fyrir SSL stuðning.

Dæmi:

```
SELECT customer_address FROM customer_table WHERE
       crypted_credit_card =
DES_ENCRYPT('credit_card_number');
```

ENCRYPT(str[,salt])

Dulkóðar str með crypt() kerfisfallinu í Unix. Stikinn salt ætti að vera strengur með tveim stöfum en má vera lengri. Ef crypt() er ekki til staðar í stýrikerfinu er skilað NULL og er því mælt með MD5() eða SHA1() í staðinn.

```
SELECT ENCRYPT('hello');                                    -> 'VxuFAJXVARROc'
```

MD5(str)

Reiknar 128 bita MD5 prófsummu[71] fyrir strenginn str og skilar sem 32 sextándukerfis tölustöfum eða NULL ef stikinn er NULL. MD5 er „Message-Digest“ algrímið.

```
SELECT MD5('testing');                  -> 'ae2b1fca515949e5d54fb22b8ed95575'
```

OLD_PASSWORD(str)

Skilar dulkóðuðu lykilorði samkvæmt „gamla sið“ í MySQL. Nauðsynlegt þegar setja þarf lykilorð í samskiptum við s.s. PHP, ODBC ofl, sem enn nota gömlu lykilorðin í MySQL. Er hið sama og gamla útgáfan af PASSWORD() fyrir útgáfur eldri en 4.1.

PASSWORD(str)

Reiknar út og skilar lykilorðs-streng sem sleginn er inn sem venjulegt lykilorð í str eða NULL ef stikinn er NULL. Þetta fall er notað í MySQL til að geyma lykilorð í dálkinum Password í mysql.user töflunni.

PASSWORD() dulkóðun virkar aðeins í aðra áttina og ekki er hægt að reikna dulkóðaða strenginn til baka. Fallið má nota í mysql biðlaranum en í sérhönnuðum forritum er betra að nota MD5() eða SHA1() í staðinn.

```
SELECT PASSWORD('badpwd');                                  --> '7f84554057dd964b'
```

SHA1(str)
SHA(str)

Reiknar út 160 bita prófsummu í SHA1 samkvæmt algríminu RFC 3174. Skilagildið er 40 stafa strengur í sextándukerfis tölum. Mælt er með fallinu til notkunar í sérhönnuðum forritum.

```
SELECT SHA1('abc');         --> 'a9993e364706816aba3e25717850c26c9cd0d89d'
```

---

[71] Checksum: Prófsumma er summa sem fæst með því að leggja saman gildi í tilteknu svæði meðal færslna og er notuð til þess að fylgjast með því hvort gögn eru rétt. Samanlagður skattfrádráttur hóps einstaklinga, reiknaður í launaforriti getur verið prófsumma. Gögnin eru annaðhvort töluleg eða stafastrengir sem litið er á sem töluleg gögn svo að reikna megi prófsummuna.

Upplýsingaföll

BENCHMARK(count,expr)

Innir yrðinguna expr count sinnum og finnur út hversu hratt MySQL innir yrðinguna. Skilagildið er ávallt núll. Notkun fer fram í mysql biðlaranum sem gefur skýrslu um hversu langan tíma inningin tók:

```
SELECT BENCHMARK(1000000,ENCODE('hello','goodbye'));
+--------------------------------------------+
| BENCHMARK(1000000,ENCODE('hello','goodbye')) |
+--------------------------------------------+
|                                          0 |
+--------------------------------------------+
1 row in set (4.74 sec)
```

CHARSET(str)

Skilar heiti stafamengisins sem strengurinn str er ritaður í.

```
SELECT CHARSET('abc');                                  --> 'latin1'
SELECT CHARSET(CONVERT('abc' USING utf8));              --> 'utf8'
```

COERCIBILITY(str)

Skilar þvingunargildi (e. Coercion) strengsins str fyrir samröðunarreglur (e. Collate).

```
SELECT COERCIBILITY('abc' COLLATE latin1_swedish_ci);   --> 0
SELECT COERCIBILITY('abc');                             --> 3
```

Skilagildin eru eftirfarandi:

Þvingun	**Merking**
0	Sérstök (e. Explicit) samröðun.
1	Engin samröðun.
2	Óbein (e. implicit) samröðun.
3	Þvingað.

COLLATION(str)

Skilar samröðunarreglum fyrir stafamengið sem str tilheyrir.

```
SELECT COLLATION('abc');                                --> 'latin1_swedish_ci'
SELECT COLLATION(_utf8'abc');                           --> 'utf8_general_ci'
```

CONNECTION_ID()

Skilar skilríkjum (e. ID, Identification) fyrir tenginguna. Hver einasta tenging við miðlarann hefur sitt eigið skilríki.

```
SELECT CONNECTION_ID();                                 --> 23786
```

CURRENT_USER()

Skilar notendanafni og hýsli (e. Host) núgilds notanda (e. Current user)samkvæmt því hvernig miðlarinn sér hann miðað við þær aðgangsstýringar sem notandinn hlítir.

```
SELECT USER();                            --> 'gestur@localhost'

SELECT * FROM mysql.user;                ERROR 1044: Access denied for user
                                         '@localhost' to database 'mysql'

SELECT CURRENT_USER();                    --> '@localhost'
```

DATABASE()

Skilar núgildum, völdum, gagnagrunni. Ef enginn sjálfgildur gagnagrunnur er virkur er skilað NULL.

```
SELECT DATABASE();                                          --> 'test'
```

FOUND_ROWS()

SELECT aðgerð getur innihaldið LIMIT hluta sem takmarkar fjölda færslna sem miðlarinn skilar til baka. Sé þess óskað að finna út hversu mörgum færslum væri skilað hefði LIMIT hlutanum verið sleppt má inna FOUND_ROWS() án þess að þurfa að inna fyrirspurnina aftur.

Til þess að þetta virki þarf að inna fyrri fyrirspurnina með SQL_CALC_FOUND_ROWS lykilorðinu í fyrra sinnið og síðan inna FOUND_ROWS():

```
SELECT SQL_CALC_FOUND_ROWS * FROM tbl_name
    -> WHERE id > 100 LIMIT 10;
SELECT FOUND_ROWS();
```

Síðari fyrirspurnin skilar til baka hversu mörgum færslum miðlarinn hefði skilað hefði sú fyrri verið rituð án LIMIT hlutans. Án lykilorðsins SQL_CALC_FOUND_ROWS getur niðurstaðan orðið önnur og MySQL getur jafnvel reiknað út allar færslur í töflunni. Þetta er fljótlegri leið en að inna fyrirspurn aftur án LIMIT hlutans. Lykilorðið SQL_CALC_FOUND_ROWS verður að vera listað í fyrsta hluta SELECT setningarinnar.

LAST_INSERT_ID()
LAST_INSERT_ID(expr)

Skilar síðasta sjálfvirkt framleidda gildinu sem sett var í dálk með AUTO_INCREMENT skilgreiningu. Gildið er geymt í miðlaranum og númerað samkvæmt tengingu biðlarans, ef t.d. annar biðlari sendir inn gildi á sömu töflu í millitíðinni frá því að biðlari 1 sendir inn færslu og framkvæmir LAST_INSERT_ID() fyrirspurn táknar talan síðasta sjálfvirkt-framleidda gildi sem biðlari 1 framleiddi án tillits til þess sem biðlari 2 framleiddi.

```
SELECT LAST_INSERT_ID();                                    --> 195
```

Ef send eru mörg gildi í einu, þ.e. í einni INSERT aðgerð er LAST_INSERT_ID() skilagildið, tala fyrir fyrsta gildið sem sent var inn.

Gera mætti eftirfarandi töflu:

```
CREATE TABLE rodun (nr INT NOT NULL);
INSERT INTO rodun VALUES (0);
```

Nota mætti töfluna á eftirfarandi máta:

```
UPDATE rodun SET nr=LAST_INSERT_ID(nr+1);
SELECT LAST_INSERT_ID();
```

SESSION_USER()
Samheiti fyrir USER().

SYSTEM_USER()
Samheiti fyrir USER().

USER()
Skilar núgildum notanda í sniðinu „notandi á hýsli“, samkvæmt því notendanafni sem notað var til að tengjast miðlaranum og hýsilnafni, lénheiti, hýsiltölvunnar sem biðlarinn tengist frá. Strengurinn sem skilað er til baka er í stafamenginu utf8.

```
SELECT USER();                                  --> 'gestur@localhost'
SELECT SUBSTRING_INDEX(USER(),'@',1);           --> 'gestur'
SELECT SUBSTRING_INDEX(USER(),_utf8'@',1);      --> 'gestur'
```

VERSION()
Skilar streng með upplýsingum um útgáfu miðlarans sem er í notkun. Ef strengurinn endar á „-log“ merkir það að dagbókarritun (e. Logging) er virk.

```
SELECT VERSION();                               --> '4.1.2-alpha-log'
```

FORMAT(X,D)
Sníður töluna X í samræmi við sniðið '#,###,###.##', námundað í D aukastafi. Ef D er núll er skilað tölu án aukastafa.

```
SELECT FORMAT(12332.123456, 4);                 --> '12,332.1235'
SELECT FORMAT(12332.1,4);                       --> '12,332.1000'
SELECT FORMAT(12332.2,0);                       --> '12,332'
```

GET_LOCK(str,timeout)
Reynir að ná læsingu með nafninu str innan timeout sekúndna. Skilað er 1 ef læsing er árangursrík, 0 ef hún náðist ekki innan timeout tímabilsins eða NULL ef skekkja kom upp s.s. vegna þess að vinnsluminnið þraut eða þráðurinn var drepinn með mysqladmin kill.

Ef læsing hefur náðst losnar um hana þegar gefið er RELEASE_LOCK(), ný GET_LOCK() inning framkvæmd eða tengingin rofnar. Sérhönnuð forrit geta nýtt sér fallið til að læsa aðgangi að tilteknum færslum, sem eru þá ekki aðgengilegar öðrum forritum á meðan.

```
SELECT GET_LOCK('lock1',10);                    --> 1
SELECT IS_FREE_LOCK('lock2');                   --> 1
SELECT GET_LOCK('lock2',10);                    --> 1
SELECT RELEASE_LOCK('lock2');                   --> 1
SELECT RELEASE_LOCK('lock1');                   --> NULL
```

Síðasta RELEASE_LOCK() inningin hér fyrir framan skilar NULL því önnur GET_LOCK() inningin hafði þegar losað um „lock1“

INET_ATON(expr)

Sé hefðbundinn IP talnastrengur gefinn er skilað heiltölu sem sendur fyrir tölurnar í strengnum. Slóðin getur verið 4 bæta eða 8 bæta slóð. INET_ATON() skilur einnig styttar útgáfur IP talna.

```
SELECT INET_ATON('209.207.224.40');                    --> 3520061480
SELECT INET_ATON('127.0.0.1'), INET_ATON('127.1');     --> 2130706433, 2130706433
```

INET_NTOA(expr)

Sé gefin heiltala (4 eða 8 bæti) er skilað streng með hefðbundinni netslóð á IP talnasniði.

```
SELECT INET_NTOA(3520061480);                          --> '209.207.224.40'
```

IS_FREE_LOCK(str)

Athugar hvort lás með nafni str sé laus til notkunar, þ.e. ekki læstur. Skilar 1 ef lásinn er laus, 0 ef hann er læstur eða í notkun og NULL ef skekkja kemur upp s.s. rangur stiki.

IS_USED_LOCK(str)

Athugar hvort lás með nafninu í str sé í notkun, þ.e. læstur og sé svo er tengi-skilríki (e. Connection identifier) biðlarans sem hefur lásinn skilað til baka. Sé str ekki læstur er skilað NULL.

MASTER_POS_WAIT(log_name,log_pos[,timeout])

Fall sem stjórnar samstillingu meistara og þræls. Þegar fallið er innt bíður það (e. Blocks) þar til þrællinn hefur sótt allar dagréttingar í dagbók meistarans.

Skilagildið er fjöldi dagbókarfærslna sem fallið þurfti að bíða eftir á meðan sóttar voru dagbókarfærslur upp að staðsetningunni log_pos. Skilað er NULL ef þræls-þráðurinn er ekki ræstur eða upplýsingar í meistaranum ekki til staðar. Skilað er -1 ef tíminn timeout rennur út. Ef þrællinn stöðvast á meðan MASTER_POS_WAIT() bíður stöðvast vinnsla fallsins. Ef timeout stikinn er gefinn verður hann að vera hærri en núll. Núll eða neikvætt timeout er sama og það sé ekki gefið og verður því ekkert timeout.

RELEASE_LOCK(str)

Opnar læsingu að nafni str sem áður var sótt með GET_LOCK(). Skilar 1 sé lásinn opnaður og 0 ef lásinn var aldrei til í vinnsluþræðinum.

UUID()

Skilar „Universal Unique Identifier“ (UUID) smíðuðum samkvæmt „DCE 1.1: Remote Procedure Call“ CAE (Common Applications Environment) Specifications gefin út af „The Open Group in October 1997“ (Document Number C706).

UUID tala er hönnuð þannig að tvö köll í fallið UUID() muni ávallt gefa mismunandi gildi, jafnvel þó köllin fari fram samtímis á tveimur tölvum á sitthvorum staðnum í tíma og rúmi.Gildið er 128-bita tala á sniðinu aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee.

```
SELECT UUID();                    --> '6ccd780c-baba-1026-9564-0040f4311e29'
```

Föll og breytivirkjar fyrir hópklausur (GROUP BY)

Notkun á hópun í „Group by“

Ef notuð er samsteypu föll (e. Group function) án þess að nota GROUP BY klausu er það jafngilt því að hópa á öllum línum fyrirspurnarinnar:

Veljum fyrst allar færslur úr töflunni „hlaupid“ fyrir dálkana „hlaupari“ og „lengdmetrar:“

```
select hlaupari, lengdmetrar from hlaupid;
+------------+-------------+
| hlaupari   | lengdmetrar |
+------------+-------------+
| 0101770001 |         500 |
| 1505780002 |         455 |
| 3010760003 |         680 |
| 0101770001 |        1500 |
| 1505780002 |        1250 |
| 3010760003 |        1360 |
| 3010760003 |        5960 |
+------------+-------------+
7 rows in set (0.00 sec)
```

Næst veljum við sömu upplýsingar og áður en með klausunni „GROUP BY hlaupari“ fáum við að sjá stök gildi í dálkinum „hlaupari“ hópaðar saman, færslur úr „lengdmetrar“ eru hópaðar eftir þeim kennitölum sem eiga þær en í sinni einföldustu mynd eins og hér sést er tekin ein færsla úr lengdmetrar á móti hverri færslu af hlaupari.

Þessi notkun á GROUP BY er röng vegna þess að ekkert samsteypufall er notað, tilgangurinn með GROUP BY er sá að hópa eins og hér er gert öllum færslum töflunnar eftir einum dálk (eða fleiri) en sjálfkrafa verða þá ýmsar færslur ósýnilegar nema þær séu teknar saman með samsteypufalli.

```
select hlaupari, lengdmetrar from hlaupid group by hlaupari;
+------------+-------------+
| hlaupari   | lengdmetrar |
+------------+-------------+
| 0101770001 |         500 |
| 1505780002 |         455 |
| 3010760003 |         680 |
+------------+-------------+
3 rows in set (0.00 sec)
```

Næst fáum við rétta niðurstöðu. Með því að bæta við notkun á samsteypufalli s.s. SUM á lengdmetrar þá fáum við samlagningu á öllum færslum í lengdmetrar sem tilheyra stöku gildi í hlaupari, en til þess þurfum við að hópa eftir þeim dálki:

```
select hlaupari, sum(lengdmetrar) from hlaupid group by hlaupari;
+------------+------------------+
| hlaupari   | sum(lengdmetrar) |
+------------+------------------+
| 0101770001 |             2000 |
| 1505780002 |             1705 |
| 3010760003 |             8000 |
+------------+------------------+
3 rows in set (0.00 sec)
```

Sú hegðun sem hér er útskýrð á við um öll samsteypuföll.

AVG(expr)

Skilar meðaltali yrðingarinnar expr.

```
SELECT student_name, AVG(test_score)
```

```
 FROM student
 GROUP BY student_name;
```

BIT_AND(expr)

Skilar „bitwise AND“ allra bitanna í expr. Útreikningurinn fer fram með 64 bita BIGINT.

BIT_OR(expr)

Skilar „bitwise OR“ allra bitanna í expr. Útreikningurinn fer fram með 64 bita BIGINT. Skilað er 0 ef engar línur voru mátaðar við fallið.

BIT_XOR(expr)

Skilar „bitwise XOR“ allra bitanna í expr. Útreikningurinn fer fram með 64 bita BIGINT. Skilað er 0 ef engar línur voru mátaðar við fallið.

COUNT(expr)

Skilar count fjölda ekki-NULL gilda í þeim línum sem SELECT aðgerðin sótti. COUNT(*) hegðar sér þannig að fjölda allra lína er skilað, hvort sem þær innihéldu NULL eða ekki. Þetta á aðeins við um MyISAM og ISAM töflur.

```
SELECT student.student_name,COUNT(*)
  FROM student,course
  WHERE student.student_id=course.student_id
  GROUP BY student_name;
```

COUNT(DISTINCT expr,[expr...])

Skilar fjölda lína sem SELECT aðgerðin sækir sem hafa einstök (e. Distinct) gildi og ekkert þeirra sé ekki-NULL.

```
SELECT COUNT(DISTINCT results) FROM student;
```

GROUP_CONCAT(expr)

Skilar streng sem inniheldur samsett gildi sem finnast í hóp. Fjarlægja má endurtekin gildi með notkun DISTINCT og raða má niðurstöðum með ORDER BY klausu.

Full málskipan er þannig:

```
GROUP_CONCAT([DISTINCT] expr [,expr ...]
             [ORDER BY {unsigned_integer | col_name | expr}
                 [ASC | DESC] [,col ...]]
             [SEPARATOR str_val])
```

```
SELECT student_name,
  GROUP_CONCAT(test_score)
  FROM student
  GROUP BY student_name;

SELECT student_name,
  GROUP_CONCAT(DISTINCT test_score
  ORDER BY test_score DESC SEPARATOR ' ')
  FROM student
  GROUP BY student_name;
```

MIN(expr)
MAX(expr)

Skilar hæsta eða lægsta gildi yrðingarinnar expr sem má bæði vera tala og strengur. Í þessum föllum, og í öðrum samsteypu-föllum[72] (e. Aggregate functions) eru tegundirnar ENUM og SET bornar saman eftir strengjagildum sínum en ekki eftir sætisvísum. ORDER BY notar hins vegar sætisvísana.

```
SELECT student_name, MIN(test_score),
MAX(test_score)
FROM student
GROUP BY student_name;
```

STD(expr)
STDDEV(expr)

Skilar staðlaðri misvísun (e. Deviation) yrðingarinnar expr eða kvaðratrót af VARIANCE(). Þetta fall er hannað fyrir samhæfni við Oracle.

SUM(expr)

Skilar samtals-summu yrðingar innar expr. Ef engar línur skila sér er skilað NULL.

VARIANCE(expr)

Skilar staðlaðri dreifni[73] (e. Variance) yrðingarinnar expr.

Dæmi um notkun samsteypufalla

Notum gagnagrunninn „test“ og smíðum tvær töflur. Taflan „hlauparar“ geymir upplýsingar um hlaupara s.s. kennitala, nafn hlauparans og þyngd hans. Taflan „hlaupid“ geymir upplýsingar um hver hleypur (kennitala hlaupara), dagsetning hlaups, lengd hlaupsins í metrum og hvaða leið var hlaupið. Gætum þess að henda töflunum séu þær til fyrir, einnig látum við þær geyma allan texta í Unicode.

Í eftirfarandi dæmi voru allar sql aðgerðir settar í sér skrá og mysql biðlarinn látinn lesa skrána með skipuninni: „`source d:/20040721-hlauparar.sql`“

```
use test;
drop table if exists hlauparar;
create table hlauparar(
  kt varchar(10) primary key,
  nafn varchar(31) not null,
  kilo float unsigned null
 ) character set ucs2 collate ucs2_bin;

drop table if exists hlaupid;
create table hlaupid(
    numer int unsigned auto_increment primary key,
    hlaupari varchar(10) not null,
```

[72] Aggregate: Samsteypa er skipulagt samsafn þátta, þar sem þættirnir geta haft sams konar eða ólíka gagnaskipan, og gagnaskipan safnsins sjálfs getur einnig verið hluti af samsvarandi samsettu tagi.

[73] Variance, population variance (of a random variable or a probability distribution). Enska: The expectation of the square of the centred random variable. #CAs$ = V(X) = E((X-E(X)#SC2$). Íslenska: Dreifni (um hendingu eða dreifingu) er meðalgildi af öðru veldi mismunar á hendingu og meðalgildi hennar. Dreifni er mælikvarði á dreifingu hendingar um meðalgildi sitt. Dreifni er staðalfrávik í öðru veldi. Samheiti: tvíveldisfrávik.

```
    dags datetime not null,
    lengdmetrar int unsigned not null,
    leid varchar(50)
    ) character set ucs2 collate ucs2_bin;
```

Setjum þrjá hlaupara inn í hlauparar og látum þá alla hlaupa þrjú hlaup utan einn sem hefur hlaupið meira. Fyrstu þrjú hlaupin í töflunni „hlaupid“ eru hlaupin núna, næstu þrjú eru hlaupin fyrir 365 dögum nema eitt aukahlaup sem var hlaupið fyrir 65 dögum.

```
  insert into hlauparar values
    ('0101770001','Jói Jónsson', 95.5),
    ('1505780002','Gudda Gaujadóttir', 55 ),
    ('3010760003','Alli Baugsson', 92);

  insert into hlaupid values
    (null,'0101770001',current_date,500,'Krýsuvíkurleið'),
    (null,'1505780002',current_date,455,'Krýsuvíkurleið'),
    (null,'3010760003',current_date,680,'Krýsuvíkurleið'),
    (null,'0101770001',date_sub(now(), interval 365 day),1500,'Þingvallaleið'),
    (null,'1505780002',date_sub(now(), interval 365 day),1250,'Þingvallaleið'),
    (null,'3010760003',date_sub(now(), interval 365 day),1360,'þingvallaleið'),
    (null,'3010760003',date_sub(now(), interval 65 day),5960,'Mývatnsleid');
```

Áður en lengra er haldið athugum við hvernig upplýsingarnar líta út í grunninum:

```
select * from hlauparar;
+------------+-------------------+------+
| kt         | nafn              | kilo |
+------------+-------------------+------+
| 0101770001 | Jói Jónsson       | 95.5 |
| 1505780002 | Gudda Gaujadóttir |   55 |
| 3010760003 | Alli Baugsson     |   92 |
+------------+-------------------+------+
3 rows in set (0.01 sec)

select * from hlaupid;
+-------+------------+---------------------+-------------+----------------+
| numer | hlaupari   | dags                | lengdmetrar | leid           |
+-------+------------+---------------------+-------------+----------------+
|     1 | 0101770001 | 2004-07-21 00:00:00 |         500 | Krýsuvíkurleið |
|     2 | 1505780002 | 2004-07-21 00:00:00 |         455 | Krýsuvíkurleið |
|     3 | 3010760003 | 2004-07-21 00:00:00 |         680 | Krýsuvíkurleið |
|     4 | 0101770001 | 2003-07-22 13:45:40 |        1500 | Þingvallaleið  |
|     5 | 1505780002 | 2003-07-22 13:45:40 |        1250 | Þingvallaleið  |
|     6 | 3010760003 | 2003-07-22 13:45:40 |        1360 | þingvallaleið  |
|     7 | 3010760003 | 2004-05-17 13:45:40 |        5960 | Mývatnsleid    |
+-------+------------+---------------------+-------------+----------------+
```

Birtum upplýsingar um öll hlaup og freistum þess að tengja töflurnar saman á kennitölu hlaupara. Þannig fáum við einnig út hver hljóp svo og þyngd hlauparans. Dálkarnir eru full margir fyrir útskrift á síðuna svo letrið hefur verið minnkað í útskriftinni.

```
select * from hlauparar, hlaupid
   where hlauparar.kt = hlaupid.hlaupari;
+------------+-------------------+------+-------+------------+---------------------+-------------+----------------+
| kt         | nafn              | kilo | numer | hlaupari   | dags                | lengdmetrar | leid           |
+------------+-------------------+------+-------+------------+---------------------+-------------+----------------+
| 0101770001 | Jói Jónsson       | 95.5 |     1 | 0101770001 | 2004-07-21 00:00:00 |         500 | Krýsuvíkurleið |
| 1505780002 | Gudda Gaujadóttir |   55 |     2 | 1505780002 | 2004-07-21 00:00:00 |         455 | Krýsuvíkurleið |
| 3010760003 | Alli Baugsson     |   92 |     3 | 3010760003 | 2004-07-21 00:00:00 |         680 | Krýsuvíkurleið |
| 0101770001 | Jói Jónsson       | 95.5 |     4 | 0101770001 | 2003-07-22 13:45:40 |        1500 | Þingvallaleið  |
| 1505780002 | Gudda Gaujadóttir |   55 |     5 | 1505780002 | 2003-07-22 13:45:40 |        1250 | Þingvallaleið  |
| 3010760003 | Alli Baugsson     |   92 |     6 | 3010760003 | 2003-07-22 13:45:40 |        1360 | þingvallaleið  |
| 3010760003 | Alli Baugsson     |   92 |     7 | 3010760003 | 2004-05-17 13:45:40 |        5960 | Mývatnsleid    |
+------------+-------------------+------+-------+------------+---------------------+-------------+----------------+
7 rows in set (0.01 sec) * klippt var af dags dálki svo taflan kæmis fyrir í næstu niðurstöðum
```

Birtum sömu upplýsingar aftur en í þetta sinnið viljum við raða útskriftinni eftir kennitölu hlauparans.

```
select * from hlauparar, hlaupid
  where hlauparar.kt = hlaupid.hlaupari
  order by hlauparar.kt;
```

```
+------------+------------------+------+-------+------------+------------+------------+----------------+
| kt         | nafn             | kilo | numer | hlaupari   | dags       | lengdmetrar | leid           |
+------------+------------------+------+-------+------------+------------+------------+----------------+
| 0101770001 | Jói Jónsson      | 95.5 |     1 | 0101770001 | 2004-07-21 |         500 | Krýsuvíkurleið |
| 0101770001 | Jói Jónsson      | 95.5 |     4 | 0101770001 | 2003-07-22 |        1500 | Þingvallaleið  |
| 1505780002 | Gudda Gaujadóttir |   55 |     5 | 1505780002 | 2003-07-22 |        1250 | Þingvallaleið  |
| 1505780002 | Gudda Gaujadóttir |   55 |     2 | 1505780002 | 2004-07-21 |         455 | Krýsuvíkurleið |
| 3010760003 | Alli Baugsson    |   92 |     7 | 3010760003 | 2004-05-17 |        5960 | Mývatnsleid    |
| 3010760003 | Alli Baugsson    |   92 |     6 | 3010760003 | 2003-07-22 |        1360 | þingvallaleið  |
| 3010760003 | Alli Baugsson    |   92 |     3 | 3010760003 | 2004-07-21 |         680 | Krýsuvíkurleið |
+------------+------------------+------+-------+------------+------------+------------+----------------+
7 rows in set (0.00 sec)
```

Til að fækka dálkum í útskrift þarf ekki að birta kennitöluna tvisvar svo við tilgreinum dálkana sérstaklega. Auk þess höfum við lítinn áhuga á hverju hlaupi fyrir sig, við viljum sjá hversu mikið hver hlaupari hefur hlaupið í metrum.

```
select kt, nafn, lengdmetrar
    from hlauparar, hlaupid
    where hlauparar.kt = hlaupid.hlaupari
    order by hlauparar.kt;
```

```
+------------+------------------+-------------+
| kt         | nafn             | lengdmetrar |
+------------+------------------+-------------+
| 0101770001 | Jói Jónsson      |         500 |
| 0101770001 | Jói Jónsson      |        1500 |
| 1505780002 | Gudda Gaujadóttir |         455 |
| 1505780002 | Gudda Gaujadóttir |        1250 |
| 3010760003 | Alli Baugsson    |        5960 |
| 3010760003 | Alli Baugsson    |         680 |
| 3010760003 | Alli Baugsson    |        1360 |
+------------+------------------+-------------+
7 rows in set (0.00 sec)
```

Nú veljum við að leggja saman heildarvegalengd hvers hlaupara með SUM og hópum hlauparana saman með GROUP BY svo „lengdmetrar“ fyrir hvern hlaupara er lagt saman sérstaklega, einnig viljum við raða útskriftinni eftir kennitölu hlauparans.

```
select kt, nafn, sum(lengdmetrar)
    from hlauparar, hlaupid
    where hlauparar.kt = hlaupid.hlaupari
    group by hlauparar.kt
    order by hlauparar.kt;
```

```
+------------+------------------+------------------+
| kt         | nafn             | sum(lengdmetrar) |
+------------+------------------+------------------+
| 0101770001 | Jói Jónsson      |             2000 |
| 1505780002 | Gudda Gaujadóttir |             1705 |
| 3010760003 | Alli Baugsson    |             8000 |
+------------+------------------+------------------+
3 rows in set (0.00 sec)
```

Nú veljum við að telja hversu oft hver hlaupari hefur hlaupið með því að telja hversu oft hann hefur skráðan hlaupdag með COUNT, ennþá leggjum við saman heildarvegalengd hans en veljum einnig að sjá meðaltal hlaupvegalengdar með AVG (stytting á Average).

```
select kt, nafn, count(dags), sum(lengdmetrar), avg(lengdmetrar)
    from hlauparar, hlaupid
    where hlauparar.kt = hlaupid.hlaupari
    group by hlauparar.kt
    order by hlauparar.kt;
```

```
+------------+------------------+-------------+------------------+------------------+
```

```
| kt         | nafn              | count(dags) | sum(lengdmetrar) | avg(lengdmetrar) |
+------------+-------------------+-------------+------------------+------------------+
| 0101770001 | Jói Jónsson       |           2 |             2000 |        1000.0000 |
| 1505780002 | Gudda Gaujadóttir |           2 |             1705 |         852.5000 |
| 3010760003 | Alli Baugsson     |           3 |             8000 |        2666.6667 |
+------------+-------------------+-------------+------------------+------------------+
3 rows in set (0.00 sec)
```

Að lokum veljum við að skoða hlaupfjölda, stystu og lengstu vegalengd hvers hlaupara auk þess sem við námundum meðaltalsvegalengdina niður í einn aukastaf en fyrirspurnin fyrir framan sýndi fjóra aukastafi.

```
select nafn, count(dags), min(lengdmetrar),
   max(lengdmetrar), round(avg(lengdmetrar),1)
   from hlauparar, hlaupid
   where hlauparar.kt = hlaupid.hlaupari
   group by hlauparar.kt
   order by hlauparar.kt;
+--------------+------------+-----------------+-----------------+--------------------------+
| nafn         | count(dags) | min(lengdmetrar) | max(lengdmetrar) | round(avg(lengdmetrar),1) |
+--------------+------------+-----------------+-----------------+--------------------------+
| Jói Jónsson  |          2 |             500 |            1500 |                   1000.0 |
| Gudda Gaujad. |          2 |             455 |            1250 |                    852.5 |
| Alli Baugsson |          3 |             680 |            5960 |                   2666.7 |
+--------------+------------+-----------------+-----------------+--------------------------+
rows in set (0.01 sec)
```

Endum á sömu fyrirspurn en styttum nöfn dálkanna og námundum alla aukastafi af meðaltalinu.

```
select nafn, count(dags) as fj_hlaupa,
   min(lengdmetrar) as stydsta_hl, max(lengdmetrar) as lengsta_hl,
   round(avg(lengdmetrar),0) as medaltal
   from hlauparar, hlaupid
   where hlauparar.kt=hlaupid.hlaupari
   group by hlauparar.kt
   order by hlauparar.kt;
+------------------+-----------+------------+------------+----------+
| nafn             | fj_hlaupa | stydsta_hl | lengsta_hl | medaltal |
+------------------+-----------+------------+------------+----------+
| Jói Jónsson      |         2 |        500 |       1500 |     1000 |
| Gudda Gaujadóttir |         2 |        455 |       1250 |      852 |
| Alli Baugsson    |         3 |        680 |       5960 |     2667 |
+------------------+-----------+------------+------------+----------+
3 rows in set (0.00 sec)
```

Breytt hegðun á hópun

Þegar notuð eru samsteypuföll og þá væntanlega GROUP BY, sem verður að nota með þeim, má nota breytuna ROLLUP sem framkvæmir viðbótar samantekt á þeim dálkum sem notaðir eru á hópun. ROLLUP virkar þannig að sé t.d. gerð samantekt á sölu mánaðar eftir sölumönnum mætti nota ROLLUP sem bætir við heildarsummu allra sölumannanna. Þegar ROLLUP er notað þýðir ekkert að nota ORDER BY til að raða úttakinu.

Smíðum eftirfarandi töflu sem geymir sölusögu hjá sölumönnum fyrirtækis þannig; mánuður sem sölur fara fram, nafn sölumanns, varan sem hann seldi (getur verið ýmsar tegundir) og heildarmagn þeirrar vöru sem sölumaðurinn selur:

```
CREATE TABLE t1 (
  manudur` date default NULL,
  nafn char(25) default NULL,
  selvoru char(25) default NULL,
  magn int(10) unsigned default NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Skrifum út lýsingu á töflunni:

```
describe t1;
+---------+------------------+------+-----+---------+-------+
| Field   | Type             | Null | Key | Default | Extra |
+---------+------------------+------+-----+---------+-------+
| manudur | date             | YES  |     | NULL    |       |
| nafn    | char(25)         | YES  |     | NULL    |       |
| selvoru | char(25)         | YES  |     | NULL    |       |
| magn    | int(10) unsigned | YES  |     | NULL    |       |
+---------+------------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Lýðfyllum töfluna með fáeinum færslum. Við sjáum að allir sölumenn hafa selt sykur. Tökum eftir því að við höfum enga sérstaka dagsetningar á sölum í maí því búið er að taka saman einstakar sölur. Í júlí hafa allir sölumenn hins vegar selt í dag.

```
insert into t1 values
  (20040500,'joi','sykur',50),
  (20040500,'godi','sykur',60),
  (20040500,'Ufi','sykur',70),
  (20040700,'joi','salt',20),
  (current_date,'joi','sykur',25),
  (current_date,'godi','sykur',55);
```

Birtum allar færslur í töflunni og röðum eftir mánuðum:

```
select * from t1 order by manudur;
+------------+------+---------+------+
| manudur    | nafn | selvoru | magn |
+------------+------+---------+------+
| 2004-05-00 | joi  | sykur   |   50 |
| 2004-05-00 | godi | sykur   |   60 |
| 2004-05-00 | Ufi  | sykur   |   70 |
| 2004-07-00 | joi  | salt    |   20 |
| 2004-07-25 | joi  | sykur   |   25 |
| 2004-07-25 | godi | sykur   |   55 |
+------------+------+---------+------+
6 rows in set (0.00 sec)
```

Tökum nú saman heildarsölu í hverjum mánuði. Við sjáum að við fáum ekki algjöra heildarsölu hvers mánaðar því í maí er sykur lagður saman sérstaklega og salt sérstaklega.

```
select sum(magn), manudur,selvoru from t1 group by manudur;
+-----------+------------+---------+
| sum(magn) | manudur    | selvoru |
+-----------+------------+---------+
|       180 | 2004-05-00 | sykur   |
|        20 | 2004-07-00 | salt    |
|        80 | 2004-07-25 | sykur   |
+-----------+------------+---------+
3 rows in set (0.00 sec)
```

Við reynum sömu fyrirspurn aftur en sleppum dálkinum „selvoru“ ef ske kynni að hann sé ástæðan fyrir ósamræminu fyrir framan, en ennþá fáum við ekki summu hvers mánaðar staka.

```
select sum(magn), manudur from t1 group by manudur;
+-----------+------------+
| sum(magn) | manudur    |
+-----------+------------+
|       180 | 2004-05-00 |
|        20 | 2004-07-00 |
|        80 | 2004-07-25 |
+-----------+------------+
rows in set (0.00 sec)
```

Nú reynum við aftur við summuna en freistum þess að draga dagsetninguna út sem mánaðarheiti eða númer mánaðar. Ennþá erum við ekkert nær takmarkinu.

```
select sum(magn), monthname(manudur) as manheiti,
month(manudur) as man_nr
from t1 group by manudur;
+-----------+----------+--------+
| sum(magn) | manheiti | man_nr |
+-----------+----------+--------+
|       180 | May      |      5 |
|        20 | July     |      7 |
|        80 | July     |      7 |
+-----------+----------+--------+
3 rows in set (0.00 sec)
```

Nú prófum við að hópa niðurstöðurnar með „GROUP BY“ eftir númeragildi mánaðanna með fallinu MONTH(). Viti menn niðurstöðurnar eru þær sem við óskum, heildarsala hvers mánaðar, hópuð eftir mánuðum.

```
select sum(magn), monthname(manudur) as manheiti,
month(manudur) as man_nr from t1
group by month(manudur);
+-----------+----------+--------+
| sum(magn) | manheiti | man_nr |
+-----------+----------+--------+
|       180 | May      |      5 |
|       100 | July     |      7 |
+-----------+----------+--------+
rows in set (0.00 sec)
```

Svo við rúllum þessu upp öllu saman, birtum heildarsölu hvers mánaðar hópaðri eftir mánuðum og sölumönnum, birtum heiti mánaðarins og látum MySQL birta heildarniðurstöður hvers mánaðar. Við sjáum þannig að salan í maí var 180, salan í júlí var 100 en heildarsala beggja mánaðanna voru 280 (milljónir?).

```
select sum(magn), nafn, selvoru, monthname(manudur)
from t1
group by month(manudur),nafn with rollup;
+-----------+------+---------+--------------------+
| sum(magn) | nafn | selvoru | monthname(manudur) |
+-----------+------+---------+--------------------+
|        60 | godi | sykur   | May                |
|        50 | joi  | sykur   | May                |
|        70 | Ufi  | sykur   | May                |
|       180 | NULL | sykur   | May                |
|        55 | godi | sykur   | July               |
|        45 | joi  | salt    | July               |
|       100 | NULL | salt    | July               |
|       280 | NULL | salt    | July               |
+-----------+------+---------+--------------------+
8 rows in set (0.00 sec)
```

Falin svið í hópun

MySQL leyfir að hópað sé eftir dálkum þó þeir séu að öðru leyti ósýnilegir í niðurstöðum. Eftirfarandi dæmi sýnir hvernig við tökum saman magn seldrar vöru, heildarupphæð og tegund vöru. Við hópum niðurstöðurnar eftir tegund vöru (sem er sýnileg) en einnig eftir mánuðinum sem varan var seld í sem er ósýnileg í niðurstöðunum.

```
select selvoru, sum(magn), sum(upph) from t1,t2
  where t1.manudur=t2.manudur and t1.nafn=t2.solumadur
  group by selvoru, month(t1.manudur);
+---------+-----------+-----------+
| selvoru | sum(magn) | sum(upph) |
```

```
+---------+-----------+-----------+
| salt    |        20 |     30000 |
| sykur   |       180 |    185000 |
| sykur   |        80 |     59000 |
+---------+-----------+-----------+
rows in set (0.00 sec)
```

Sölutaflan t2

```
drop table t2;
create table t2(
  solumadur char(25) not null,
  kaupandi char(25) not null,
  manudur date not null,
  primary key (solumadur,kaupandi,manudur)
);

create table t2(
  solumadur char(25) not null,
  kaupandi char(25) not null,
  manudur date not null,
  primary key (solumadur,kaupandi,manudur)
);

alter table t2 add column upph bigint unsigned not null default 0;
insert into t2 values('joi','Skuti',20040500);
update t2 set upph=50000
  where solumadur='joi' and kaupandi='Skuti' and manudur=20040500;

update t1 set manudur = 20040700 where manudur = 20040725;
insert into t2 values('godi','Hali',20040500,60000);
insert into t2 values('Ufi','Moli',20040500,75000);
insert into t2 values('joi','Skuti',20040700,30000);
insert into t2 values('godi','Hali',20040700,29000);
insert into t2 values('Ufi','Moli',20040700,15000);
```

Föll fyrir textaleit

MATCH (col1,col2,...) AGAINST (expr [IN BOOLEAN MODE | WITH QUERY EXPANSION])

Fall sem leyfir að leitað sé mjög ýtarlega að texta sem kemur fyrir í textadálkum. Til þess að þetta sé hægt þarf að smíða töfluna með FULLTEXT breytu þar sem tilgreint er hvaða dálkar skuli notaðir. Þetta virkar aðeins á MyISAM töflur og virkar á CHAR, VARCHAR, og TEXT..

Breytan „IN BOOLEAN MODE“ leyfir að leitarstrengurinn expr innihaldi Boolan rökskilyrði líkt og öflugri leitarvélar á Vefnum gera. Eftirfarandi virkja má nota við leitarskilyrði:

+	Plús áfastur orðinu krefst þess að það finnist í niðurstöðum.
-	Mínus áfastur orðinu krefst þess að það finnist ekki í niðurstöðunum
(enginn)	Enginn virki er sjálfgefið og leitað er orðsins í öllum færslum.
> <	Stærri-en og minni-en áfastir orðum tilgreina aukið eða minnkað mikilvægi orðanna í niðurstöðum.
( )	Svigi utan um leitarskilyrði leitar fyrst að því.
~	Tilda áföst leitarorði tilgreinir það sem ómikilvægt en hefur það með, mínus hefði sleppt því en tildan leyfir því að vera með

* Stjarna er hefðbundinn algildisstafur.

" Texti sem settur er innan “tvíhöggs” er notaður til að tilgreina orðasamband.

Breytan WITH QUERY EXPANSION virkar þannig að leitað er tvisvar að gefnum leitarstreng, mælt er með að notað sé sem styst leitarskilyrði. Þannig sé leitað að orðinu ‘database’ en óbeint er óskað niðurstaðna sem innihalda orðið ‘MySQL’ eða ‘Oracle’, myndi þessi breyta virka þannig að allar færslur sem innihalda database birta niðurstöður s.s. Oracle ef það orð kæmi fyrir í sama texta og orðið database.

Dæmi a

Smíðum og lýðfyllum töflu sem getur geymt mikið magn af texta ásamt fyrirsögnum.

```
create table sulta(
  num int unsigned auto_increment primary key,
  titill varchar(255),
  efni text,
  fulltext(titill, efni)
) ;

insert into sulta (titill, efni) values
  ('Gagnagrunnar','Gagnagrunnar og gagnageymslur ...'),
  ('Forritun','Pascal og java og php og visual basic'),
  ('Gagnagrunnar','mysql oracle access'),
  ('Forritun','stefjur og breytur og slaufur og ...'),
  ('Forritun', 'java forritun er ...') ;
```

Tvær fyrirspurnir, sú fyrri biður um allar færslur sem innihalda orðið „gagnagrunnar“ og sú síðari um allar færslur sem innihalda orðið „java.“

```
select * from sulta
  where match(titill, efni) against('gagnagrunnar')

+-----+--------------+-----------------------------------+
| num | titill       | efni                              |
+-----+--------------+-----------------------------------+
|   3 | Gagnagrunnar | mysql oracle access               |
|   1 | Gagnagrunnar | Gagnagrunnar og gagnageymslur ... |
+-----+--------------+-----------------------------------+
2 rows in set (0.01 sec)

select * from sulta
  where match(titill, efni) against('java')

+-----+----------+----------------------------------------+
| num | titill   | efni                                   |
+-----+----------+----------------------------------------+
|   2 | Forritun | Pascal og java og php og visual basic  |
|   5 | Forritun | java forritun er ...                   |
+-----+----------+----------------------------------------+
2 rows in set (0.00 sec)
```

Nú biðjum við um allar færslur sem innihalda orðið „mysql“ en við viljum líka skylt efni.

```
select * from sulta
  where match(titill, efni) against('mysql' WITH QUERY EXPANSION)

+-----+--------------+-----------------------------------+
| num | titill       | efni                              |
+-----+--------------+-----------------------------------+
|   3 | Gagnagrunnar | mysql oracle access               |
|   1 | Gagnagrunnar | Gagnagrunnar og gagnageymslur ... |
+-----+--------------+-----------------------------------+
2 rows in set (0.01 sec)
```

Þá er að biðja um allar færslur sem innihalda orðið „pascal“ ásamt skyldu efni.

```
select * from sulta
  where match(titill, efni) against('pascal' WITH QUERY EXPANSION)

+-----+----------+---------------------------------------+
| num | titill   | efni                                  |
+-----+----------+---------------------------------------+
|   2 | Forritun | Pascal og java og php og visual basic |
|   5 | Forritun | java forritun er ...                  |
+-----+----------+---------------------------------------+
2 rows in set (0.00 sec)
```

Nú biðjum við um allar færslur sem innihalda orðin „java“ og „breytur.“

```
select * from sulta
  where match(titill, efni) against('java breytur')

+-----+----------+---------------------------------------+
| num | titill   | efni                                  |
+-----+----------+---------------------------------------+
|   4 | Forritun | stefjur og breytur og slaufur og ...  |
|   2 | Forritun | Pascal og java og php og visual basic |
|   5 | Forritun | java forritun er ...                  |
+-----+----------+---------------------------------------+
3 rows in set (0.00 sec)
```

Framkvæmum sömu fyrirspurn og hér fyrir framan nema gerum það í „Boolean mode“, taktu eftir hvernig röðunin breytist á niðurstöðunum, sem að öðru leyti eru hinar sömu.

```
select * from sulta
  where match(titill, efni) against('java breytur' in boolean mode)

+-----+----------+---------------------------------------+
| num | titill   | efni                                  |
+-----+----------+---------------------------------------+
|   2 | Forritun | Pascal og java og php og visual basic |
|   4 | Forritun | stefjur og breytur og slaufur og ...  |
|   5 | Forritun | java forritun er ...                  |
+-----+----------+---------------------------------------+
3 rows in set (0.01 sec)
```

Dæmi b

Smíðum töflu sem getur geymt helling af efni á sniðinu fyrirsögn og textabúkur. Textabúkur getur geymt heilmikið af efni en fyrirsögn getur lengst verið 255 stafir. Við skilgreinum töfluna sem UTF8 svo hún geti geymt Unicode rittákn, því miður er ekki ennþá hægt að nota UCS2 sem er öflugri Unicode útfærsla en UTF8.

Þetta dæmi er gott að setja í runu skrá (t.d. sem runa.sql á C:\ drifinu) og láta mysql biðlarann lesa inn með skipuninni „`source c:/runa.sql`“.

Smíðum töfluna en eyðum henni fyrst ef ske kynni að hún sé til fyrir.

```
drop table if exists glosur;
create table glosur(
  num int unsigned auto_increment primary key,
  fyrirsogn varchar(255) not null,
  bukur mediumtext not null,
  fulltext(fyrirsogn,bukur)
) character set utf8;
```

Bráðabirgða lýðfylling og skoðun á innihaldi töflunnar strax á eftir.

```
insert into glosur values
  (null,'Veiðar','Sjóstangaveiði er svöl'),
  (null,'Veiðar','Skytterí er skemmtilegra'),
  (null,'Forritun','Forritun er þreytandi en skemmtileg'),
  (null,'Matreiðsla','Matreiðsla er auðveld, skemmtileg og gefandi');

select * from glosur;
+-----+-----------+-----------------------------------------+
| num | fyrirsogn  | bukur                                   |
+-----+-----------+-----------------------------------------+
|   1 | Veiðar     | Sjóstangaveiði er svöl                  |
|   2 | Veiðar     | Skytterí er skemmtilegra                |
|   3 | Forritun   | Forritun er þreytandi en skemmtileg     |
|   4 | Matreiðsla | Matreiðsla er auðveld, skemmtileg og gefandi |
+-----+-----------+-----------------------------------------+
4 rows in set (0.00 sec)
```

Leitum að öllum færslum sem innihalda orðið „Forritun“ og strax á eftir allar færslur sem innihalda orðið „veiðar.“

```
select fyrirsogn, bukur
from glosur
where match(fyrirsogn,bukur) against('Forritun' with query expansion);
+-----------+-----------------------------------+
| fyrirsogn | bukur                             |
+-----------+-----------------------------------+
| Forritun  | Forritun er þreytandi en skemmtileg |
+-----------+-----------------------------------+
1 row in set (0.00 sec)
```

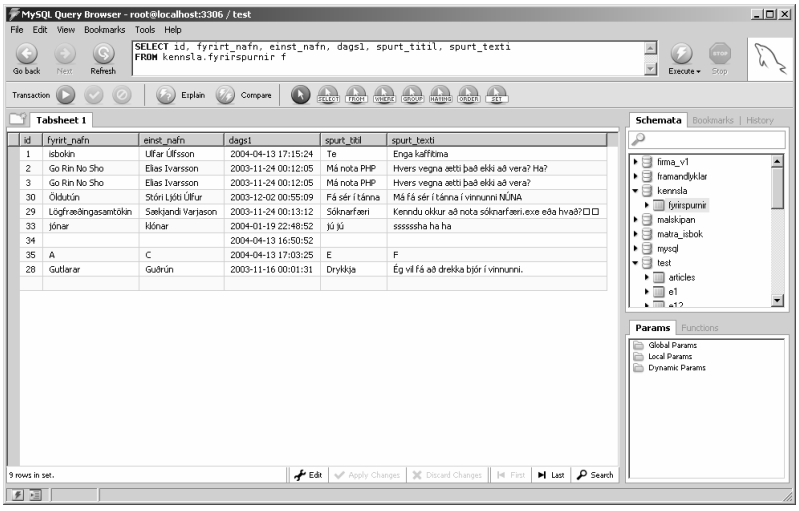
Finnum allar færslur sem innihalda textann „veiðar“ en viljum þær ekki ef þær innihalda orðið „skytterí.“

```
select fyrirsogn, bukur from glosur
  where match(fyrirsogn,bukur) against('+veiðar -skytterí' in boolean mode);
+-----------+-----------------------+
| fyrirsogn | bukur                 |
+-----------+-----------------------+
| Veiðar    | Sjóstangaveiði er svöl |
+-----------+-----------------------+
1 row in set (0.00 sec)
```

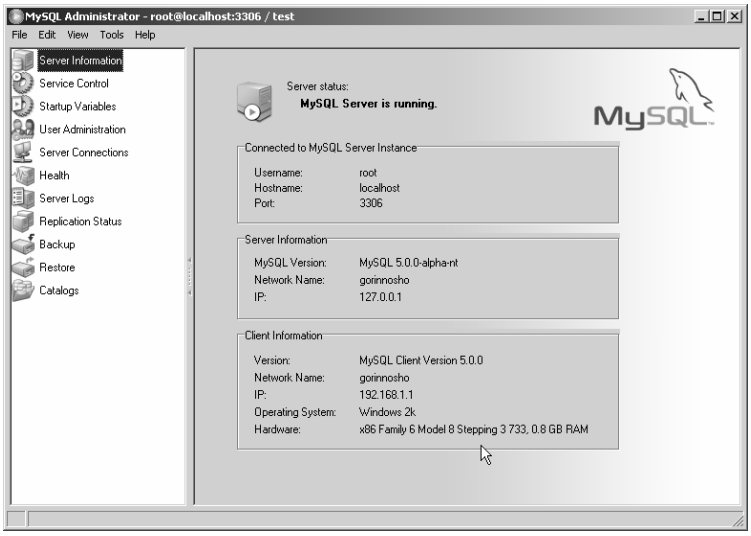
10. Kafli. Grafísk tól og uppsetning

Grafísk tól

MySQL Query Browser er nýlegt tól frá MySQL AB sem enn er á þróunarstigi en má þó sækja og prófa. Með tólinu má sjá þá grunna sem miðlarinn hýsir. Forritið býður upp á glugga þar sem rita má allar SQL aðgerðir sem venjulega eru ritaðar í kvaðningu mysql biðlarans og niðurstöður eru birtar í töfluformi. Tólið man þær fyrirspurnir sem hafa áður verið framkvæmdar og má flakka á milli þeirra á mismunandi vegu.

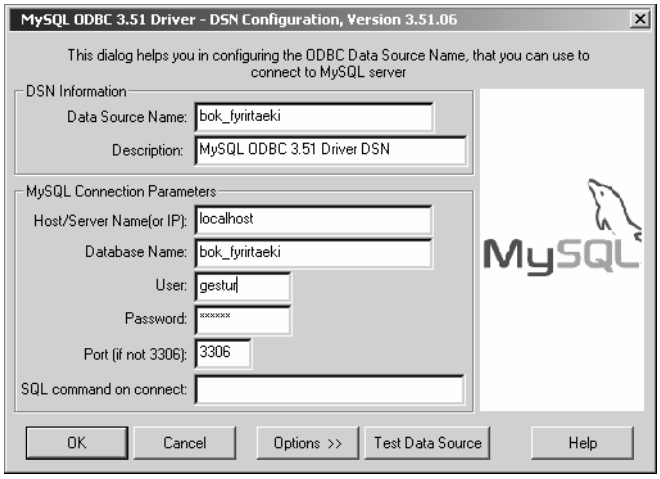


MySQL Administrator er grafískt tól sem, eins og Query Browser, er nýkominn út. Með forritinu má fá yfirsýn yfir ástand miðlarans með mun aðgengilegra sniði en í gamla tólinu winmysqladmin. með tólinu má sömuleiðis framkvæma alla stjórn miðlarans.



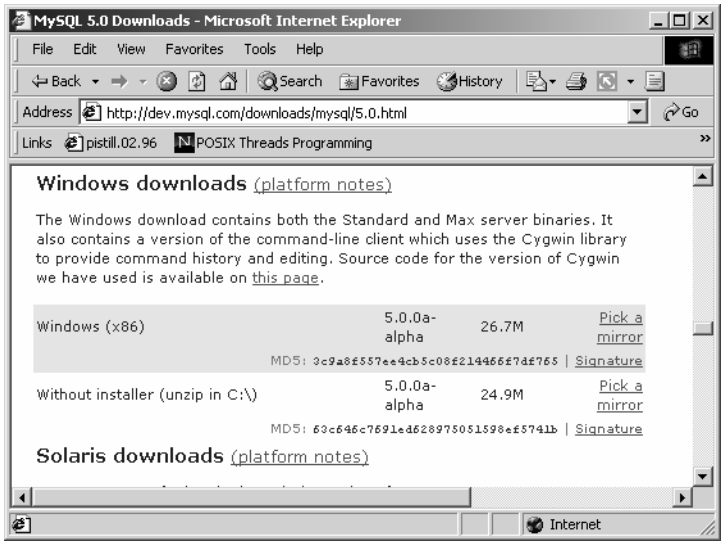
MyODBC er forrit sem leyfir skilgreiningar á ODBC tenginum við MySQL miðlarann. Þetta tól er nauðsynlegt fyrir ýmsar aðstæður. Þeir sem t.d. þurfa að tengja ASP vefsíður við MySQL grunna þurfa yfirleitt að tengja með ODBC sniði og þurfa þá að hafa þetta tól uppsett. Ýmis önnur forritunartól þarfnast þess sömuleiðis.

Myndin sýnir hvar skilgreind er tengingin „ibok_fyrirtaeki“ gagnagrunninn ibok_fyrirtaeki á miðlara á sömu vél og tengingin er gerð. Notandinn er gestur og lykilorðið er „giskaðu“. TCP/IP hliðið er 3306, sem er venja í sjálfgefnum uppsetningum miðlarans. Hliðinu verður að vera hið sama og er skilgreint á miðlaranum sjálfum. Notandinn gestur þarf sömuleiðis að hafa aðgangsleyfi á grunninn.



Uppsetning MySQL á tölvu

Á Windows verkvangi eru yfirleitt í boði tvennskonar pakkar fyrir hverja útgáfu, báðir í .ZIP skrá. Önnur skráin inniheldur setup skrá sem má nota til innsetningar rétt eins og öll önnur forrit. Sjálfgefið er að miðlarinn, og allt sem honum tilheyrir, setji sig í möppuna „c:\mysql“ en því má breyta í uppsetningu. Hin skráin er mysql uppsetningin eins og hún myndi líta út nýuppsett.



Myndin er af niðurhalssíðunni fyrir útgáfu 5.0 og sjást stiklurnar sem bjóða niðurhal á framangreindum skrám.

Á Linux tölvu er sömuleiðis auðvelt að setja miðlarann upp, en það er yfirleitt gert úr kvaðningu, þó vissulega megi gera það í X umhverfinu. Í boði er að niðurhala mjög mörgum skrám. Það þýðir þó ekki að fleira sé í boði fyrir Linux heldur að hægt er að sækja útgáfuna í smærri einingum. Nauðsynlegar einingar eru client og server pakkinn. Allir pakkarnir niðurhalast sem „rpm “ skrár, þó má sækja uppruna-kóta í staðinn en þá þarf að hafa meira fyrir að koma því inn.

Eftirfarandi listi eru þær skrár sem í boði eru. Endingin .rpm stendur fyrir að pakkinn sé framleiddur fyrir RedHat Package Manager en það er forrit sem auðveldar innsetningu Linux forrita.

```
MySQL-bench-5.0.0-0.i386.rpm
MySQL-client-5.0.0-0.i386.rpm
MySQL-devel-5.0.0-0.i386.rpm
MySQL-embedded-5.0.0-0.i386.rpm
MySQL-server-5.0.0-0.i386.rpm
MySQL-shared-5.0.0-0.i386.rpm
```

Þegar ég uppfæri MySQL á Linux vélinni þá byrja ég yfirleitt á að taka gömlu útgáfuna í burtu. Þetta má gera með skipuninni:

```
kvaðn# rpm –e mysql
kvaðn# rpm –e ––nodeps mysql
```

Stikinn –e stendur fyrir „erase“ til að eyða uppsetningunni og --nodeps er svo aðgerðin takist óháð því hvort einhver önnur forrit reiði sig á MySQL. Ég mæli með því að aðgerðin sé fyrst framkvæmd án nodeps stikans því þá birtast aðvaranir um hvaða forrit krefjist MySQL.

Þegar ég tók út síðustu útgáfu (4.1) af Linux vélinni þurfti ég nodeps til að koma honum út. Því næst framkvæmdi ég eftirfarandi aðgerð:

```
kvaðning [niðurhals-skráasafnið]# rpm –i MySQL*.rpm
```

Þessi aðgerð dugði til að skipa rpm forritinu að innsetja alla pakkana. Stikinn –i stendur fyrir „install.“ Pakkahöndlarinn rpm býður reyndar upp á stikann –u fyrir „upgrade“ og þá má reyna að uppfæra pakkana yfir eldri útgáfu, án þess þá að eyða þeirri eldri út fyrst.

Þar sem ég er vanur að eyða út fyrst og þá með nodeps stikanum þarf ég yfirleitt að endur-innsetja PHP túlkinn á vélinni og einstaka sinnum að dagrétta Apache uppsetninguna. Því miður er umfjöllun um slíkt

utan efnissviðs bókarinnar. Bendi ég á ágætist umfjöllun um þessa þætti á www.php.net og www.apache.org.

Yfirleitt þarf ekkert frekar til en fyrrgreinda aðgerð, miðlarinn fer er ræstur sjálfkrafa og skilgreinir sjálfur ef þörf krefur notandann mysql sem verður eigandi vinnsluferlis hans. Ekki eyða þeim notanda út. Að öðru leyti er notkun MySQL keimlík á öllum verkvöngum.

16. Æfing: Bókasafn Ísbókar

Tæknibókasafn Íslenzkra tölvubóka (TÍT) hefur u.þ.b. 16.000 meðlimi (eða lánþega), 100.000 titla og 250.000 bækur (e. Volumes) eða að meðaltali 2.5 eintak pr. titil. Rúmlega 10% bóka eru í útláni hverju sinni. Starfsfólk bókasafnsins gæta þess að eintök séu ávallt fyrir hendi þegar meðlimir óska að taka þær að láni. Því þarf starfsfólk að vita hversu mörg eintök hverrar bókar séu í útláni á hverjum tíma eða inni.

Bókaskrá safnsins er aðgengileg innankerfis (e. On-line) sem listar bækur eftir höfundi, titli og efnisflokki. Fyrir hvern titil sem til er í safninu er einnig til bókarlýsing sem tekur allt frá einni línu til margra síðna. Bókasafnsfræðingar og annað starfsfólk þarf að hafa aðgang að þessum lýsingum þegar meðlimir safnsins leita upplýsinga um bók. Starfsfólk safnsins skiptist þannig: Yfirbókavörður, Deildar-bókaverðir, aðstoðarfólk innan deilda, skjölunar-verðir, afgreiðslufólk og almennir bókaverðir.

Meðlimir geta haft bók að láni í 21 dag og mest 5 bækur í senn. Venjulega er bókum skilað innan 3ja til 4ra vikna. Flestir meðlimir vita að þeir hafa skilafrest í eina viku eftir að útlánstíma lýkur áður en þeim er send tilkynning, yfirleitt er bókum skilað áður en þessi frestur rennur út. Að meðaltali þarf að minna um það bil 5% meðlima á að skila bókum með tilkynningu og u.þ.b. 95% þeirra skila eftir það, afgangurinn skilar sér sjaldnast. Virkustu meðlimir safnsins eru skilgreindir þannig að þeir taki bækur að útláni eigi sjaldnar en 10 sinnum á ári. 1% meðlima tekur út 15% útlána og 10% meðlima taka út 40% útlána, um 20% meðlima eru óvirkir þannig að þó þeir séu meðlimir að öllu jöfnu taka þeir nánast aldrei út bækur.

Til þess að gerast meðlimir þurfa umsækjendur að fylla út eyðublað þar sem fram komi kennitala, fullt nafn, lögheimili, aðsetur sé þess þörf, heimasími, vinnusími og gsm sími, auk þess að skila þarf inn passamynd. Bókavörður gefur þá út skírteini með raðnúmeri, kennitölu og mynd af meðlimi sem lesa má með kortalesara (eldra kerfi notaði strikamerkis lesara og mörg þeirra korta eru óútrunnin) og er kortið gilt í fjögur ár. Mánuði áður en kortið rennur út er meðlimi send um það tilkynning og boðin endurnýjun. Starfsfólk Íslenzkra tölvubóka; bókahöfundar, starfsfólk bókasafns, kennarar, forritarar, o.s.frv. eru sjálfkrafa meðlimir. Þegar nýr starfsmaður er skráður í starfsmannagrunn er hann sjálfkrafa skráður í meðlimaskrá bókasafnsins og skírteini hans sent í innanhúss pósti. Starfsfólk getur tekið út bækur í allt að tvo mánuði og hafa mánuð í skilafrest að lánstíma loknum, áminningar um bókaskil eru send í innanhúss pósti.

Ekki eru lánaðar uppflettibækur, sjaldgæfar bækur né kort. Bókaverðirnir bera ábyrgð á að flokka bækurnar í lánshæfar bækur og ólánshæfar. Auk þess viðhalda bókaverðir lista yfir bækur sem þeir vildu gjarnan komast yfir en geta tæpast; t.d. sjaldgæfar bækur, „out-of-print“ bækur, eða bækur sem hafa týnst eða eyðilagst og hafa ekki verið endurnýjaðar. Bókaverðir þurfa kerfi sem lista þær bækur sem ekki má lána út auk bóka sem þeir hafa hug á að komast yfir. Sumar bækur gætu haft sama titil, því verður að finna og nota aðra leið til að bera kennsl á hverja bók (eða bókartitil). Hver einasta bók er þekkt eftir „International Standard Book Number (ISBN)“ númeri sínu, sem er einkvæmt númer sem hver útgefinn bókartitill fær úthlutað. Tvær bækur með sama titli geta haft mismunandi ISBN númeragildi ef þær eru á sitt hvoru tungumálinu eða mismunandi kilju-útgáfu (pappírskilja eða harðkilja t.d.). Mismunandi útgáfur sama titils hafa mismunandi ISBN númer s.s. „Íslenzka MySQL bókin“ útgáfa 1 og útgáfa 2.

Óskað er gagnagrunns kerfis, hannað þannig að geymdar séu upplýsingar varðandi meðlimi, bækurnar, efnisskrána (bókaskrána) og útlán. Hafðu í huga að ekki er til ein rétt útfærsla á lausn af þessu tagi, ef þú ynnir verkefnið aftur t.d. að sex mánuðum liðnum kemurðu líklega með aðra hugmynd að lausn. Hafðu líka

annað í huga; því oftar sem lýsing af þessu tagi er rissuð upp og hún leyst, því öflugri verða þær lausnir sem þú vinnur, auk þess sem þær vinnast léttari í hvert sinn.

Verklýsing:

1. Byrjaðu á að greina niður eftirfarandi: Þátttakendur (hugsanleg einindi, t.d. lántakar og bækur), viðburðir (t.d. útlán er viðburður, tilkynning um útrunninn lánstíma er viðburður), þau gildi (eigindi) sem geymd eru (ISBN númer, kennitala, bókartitlar). Listaðu upp öll þessi atriði í stafrófsröð á sniðinu: atriði – eðli (bók – einindi, ISBN – gildi, ...).
2. Teiknaðu upp myndræna útgáfu þessarar greiningar þannig að þú flokkar einindin og gildi þeirra niður t.d. teiknarðu ramma fyrir lánþega bókasafnsins (lánþega einindið) og listar í þeim ramma öll gildin sem hann hefur.
3. Teiknaðu sömu atriði og áður en nú skaltu nota ERD (Entity-Relationship-Diagram) til að tákna öll framangreind atriði.
4. Nú væri tímabært að smíða grunninn, töflurnar og lýðfylla, en sannreyndu hönnunina fyrst.

11. Kafli. Viðaukar

Viðauki a: Yfirlit SQL málskipana

Málskipanir SQL aðgerðaí MySQL. Þessi kafli er yfirlit yfir málskipanir þeirra SQL aðgerða[74] sem MySQL útgáfa 5.0 Alpha styður. Kaflinn er ætlaður sem uppflettiefni yfir málskipanir allra SQL aðgerða miðlarans.

Aðgerðir sem meðhöndla gögn

DELETE

Ein tafla

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
       [WHERE where_definition]
       [ORDER BY ...]
       [LIMIT row_count]
```

Margar töflur

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
       tbl_name[.*] [, tbl_name[.*] ...]
       FROM table_references
       [WHERE where_definition]
```

eða:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
       FROM tbl_name[.*] [, tbl_name[.*] ...]
       USING table_references
       [WHERE where_definition]
```

DO

```
DO expr [, expr] ...
```

HANDLER

```
HANDLER tbl_name OPEN [ AS alias ]
HANDLER tbl_name READ index_name { = | >= | <= | < } (value1,value2,...)
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ index_name { FIRST | NEXT | PREV | LAST }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ { FIRST | NEXT }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name CLOSE
```

INSERT

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    VALUES ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

Eða:

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
```

[74] SQL Statement er ýmist þýtt sem aðgerð, setning eða skipun og þá í sömu merkingu.

```
    [INTO] tbl_name
    SET col_name={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

Eða:

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
```

INSERT ... SELECT

```
INSERT [LOW_PRIORITY] [IGNORE] [INTO] tbl_name [(column_list)]
    SELECT ...
```

INSERT DELAYED

```
INSERT DELAYED ...
```

LOAD DATA INFILE

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name.txt'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [FIELDS
        [TERMINATED BY '\t']
        [[OPTIONALLY] ENCLOSED BY '']
        [ESCAPED BY '\\' ]
    ]
    [LINES
        [STARTING BY '']
        [TERMINATED BY '\n']
    ]
    [IGNORE number LINES]
    [(col_name,...)]
```

REPLACE

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [(col_name,...)]
    VALUES ({expr | DEFAULT},...),(...),...
```

Eða:

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name
    SET col_name={expr | DEFAULT}, ...
```

Eða:

```
REPLACE [LOW_PRIORITY | DELAYED]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
```

SELECT

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr,...
    [INTO OUTFILE 'file_name' export_options
      | INTO DUMPFILE 'file_name']
```

```
    [FROM table_references
      [WHERE where_definition]
      [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
      [HAVING where_definition]
      [ORDER BY {col_name | expr | position}
        [ASC | DESC] ,...]
      [LIMIT {[offset,] row_count | row_count OFFSET offset}]
      [PROCEDURE procedure_name(argument_list)]
      [FOR UPDATE | LOCK IN SHARE MODE]]
```

JOIN

MySQL styður „JOIN“ málskipun fyrir töflu_tilvísun sem hluta af SELECT skipun og fjöl-töflu DELETE og UPDATE skipana:

```
table_reference, table_reference
table_reference [INNER | CROSS] JOIN table_reference [join_condition]
table_reference STRAIGHT_JOIN table_reference
table_reference LEFT [OUTER] JOIN table_reference [join_condition]
table_reference NATURAL [LEFT [OUTER]] JOIN table_reference
{ OJ table_reference LEFT OUTER JOIN table_reference
    ON conditional_expr }
table_reference RIGHT [OUTER] JOIN table_reference [join_condition]
table_reference NATURAL [RIGHT [OUTER]] JOIN table_reference
töflu_tilvísun er skilgreind sem:
tbl_name [[AS] alias]
    [[USE INDEX (key_list)]
      | [IGNORE INDEX (key_list)]
      | [FORCE INDEX (key_list)]]
join_skilyrði er skilgreint sem:
ON conditional_expr | USING (column_list)
```

UNION

```
SELECT ...
UNION [ALL | DISTINCT]
SELECT ...
  [UNION [ALL | DISTINCT]
   SELECT ...]
```

Undir-fyrirspurnir (e. Subqueries)

Undir-fyrirspurnir skila helst einum dálki.

Dæmi:

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

Dæmi:

```
DELETE FROM t1
WHERE s11 > ANY
 (SELECT COUNT(*) /* no hint */ FROM t2
 WHERE NOT EXISTS
  (SELECT * FROM t3
   WHERE ROW(5*t2.s1,77)=
    (SELECT 50,11*s1 FROM t4 UNION SELECT 50,77 FROM
     (SELECT * FROM t5) AS t5)));
```

Undir-fyrirspurn sem kverðu[75] virki

Dæmi:

```
CREATE TABLE t1 (s1 INT, s2 CHAR(5) NOT NULL);
SELECT (SELECT s2 FROM t1);
```

Dæmi:

```
CREATE TABLE t1 (s1 INT);
INSERT INTO t1 VALUES (1);
CREATE TABLE t2 (s1 INT);
INSERT INTO t2 VALUES (2);
SELECT (SELECT s1 FROM t2) FROM t1;
SELECT UPPER((SELECT s1 FROM t1)) FROM t2;
```

Samanburður með undir-fyrirspurn

Algengasta snið:

```
non_subquery_operand comparison_operator (subquery)
```

Þar sem samanburðar virki er einhver af:

```
=  >  <  >=  <=  <>
```

Dæmi:

```
... 'a' = (SELECT column1 FROM t1)
```

Undir-fyrirspurnir með ANY, IN, og SOME

```
operand comparison_operator ANY (subquery)
operand IN (subquery)
operand comparison_operator SOME (subquery)
```

Dæmi:

```
SELECT s1 FROM t1 WHERE s1 > ANY (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 IN    (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 <> ANY  (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 <> SOME (SELECT s1 FROM t2);
```

Undir-fyrirspurnir með ALL

```
operand comparison_operator ALL (subquery)
```

Dæmi:

```
SELECT s1 FROM t1 WHERE s1 > ALL (SELECT s1 FROM t2);
```

Samsvarandi undir-fyrirspurnir[76]

Samsvarandi undir-fyrirspurn er sú sem hefur töflu-tilvísun á sömu töflu bæði í innri og ytri fyrirspurninni.

Dæmi:

```
SELECT * FROM t1 WHERE column1 = ANY
       (SELECT column1 FROM t2 WHERE t2.column2 = t1.column2);
```

Svæðisregla: MySQL metur fyrirspurnina að innan og út samanber:

```
SELECT column1 FROM t1 AS x
  WHERE x.column1 = (SELECT column1 FROM t2 AS x
    WHERE x.column1 = (SELECT column1 FROM t3
```

---

[75] Kverði er á ensku „Scalar.“ Stærð sem ákvarðast af einu gildi.

[76] Correlate: Samsvari er annað af tvennu, sem er skylt eða samsvarar hvað öðru (Orðabók: uppeldisfræði).

```
      WHERE x.column2 = t3.column1));
```

EXISTS og NOT EXISTS

Ef undir-fyrirspurn skilar einhverjum gildum þá skilar `EXISTS subquery` fyrirspurn `TRUE`, og `NOT EXISTS subquery` skilar `FALSE`.

Dæmi:

```
SELECT column1 FROM t1 WHERE EXISTS (SELECT * FROM t2);
SELECT DISTINCT store_type FROM Stores
  WHERE NOT EXISTS (SELECT * FROM Cities_Stores
                    WHERE Cities_Stores.store_type = Stores.store_type);
```

Línu undir-fyrirspurnir

Hingað til hafa allar undir-fyrirspurnir skilað *einum dálk* en hægt er að nota línu undir-fyrirspurnir sem skila einni línu en með mörgum dálkum.

Dæmi:

```
SELECT * FROM t1 WHERE (1,2) = (SELECT column1, column2 FROM t2);
SELECT * FROM t1 WHERE ROW(1,2) = (SELECT column1, column2 FROM t2);
SELECT * FROM t1 WHERE (column1,column2) = (1,1);
SELECT * FROM t1 WHERE column1 = 1 AND column2 = 1;
SELECT column1,column2,column3
       FROM t1
       WHERE (column1,column2,column3) IN
             (SELECT column1,column2,column3 FROM t2);
```

Fyrirspurnir skila `TRUE` ef taflan sem spurð er hefur línu þar sem `column1 = 1` and `column2 = 2`. Segðin `(1,2)` og `ROW(1,2)` eru stundum nefndar *línusmiðir* (e. *row constructors*).

Undir-fyrirspurnir í FROM klausu

```
SELECT ... FROM (subquery) AS name ...
```

Umritun á undir-fyrirspurnum sem tvinnun (e. Joins) fyrir eldri MySQL útgáfur

Undir-fyrirspurnina:

```
SELECT * FROM t1 WHERE id IN (SELECT id FROM t2);
```

má rita þannig:

```
SELECT DISTINCT t1.* FROM t1,t2 WHERE t1.id=t2.id;
```

Fyrirspurnirnar:

```
SELECT * FROM t1 WHERE id NOT IN (SELECT id FROM t2);
SELECT * FROM t1 WHERE NOT EXISTS (SELECT id FROM t2 WHERE t1.id=t2.id);
```

má rita þannig:

```
SELECT table1.* FROM table1 LEFT JOIN table2 ON table1.id=table2.id
                                     WHERE table2.id IS NULL;
```

TRUNCATE

```
TRUNCATE TABLE tbl_name
```

UPDATE

Ein tafla:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1=expr1 [, col_name2=expr2 ...]
```

```
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT row_count]
```

Margar töflur:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name [, tbl_name ...]
    SET col_name1=expr1 [, col_name2=expr2 ...]
    [WHERE where_definition]
```

Aðgerðir fyrir gagnaskilgreiningar

ALTER DATABASE

```
ALTER DATABASE db_name
    alter_specification [, alter_specification] ...

alter_specification:
    [DEFAULT] CHARACTER SET charset_name
  | [DEFAULT] COLLATE collation_name
```

ALTER TABLE

```
ALTER [IGNORE] TABLE tbl_name
    alter_specification [, alter_specification] ...

alter_specification:
    ADD [COLUMN] column_definition [FIRST | AFTER col_name ]
  | ADD [COLUMN] (column_definition,...)
  | ADD INDEX [index_name] [index_type] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        PRIMARY KEY [index_type] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        UNIQUE [index_name] [index_type] (index_col_name,...)
  | ADD [FULLTEXT|SPATIAL] [index_name] (index_col_name,...)
  | ADD [CONSTRAINT [symbol]]
        FOREIGN KEY [index_name] (index_col_name,...)
        [reference_definition]
  | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
  | CHANGE [COLUMN] old_col_name column_definition
        [FIRST|AFTER col_name]
  | MODIFY [COLUMN] column_definition [FIRST | AFTER col_name]
  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP INDEX index_name
  | DROP FOREIGN KEY fk_symbol
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] new_tbl_name
  | ORDER BY col_name
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | [DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | table_options
```

CREATE DATABASE

```
CREATE DATABASE [IF NOT EXISTS] db_name
    [create_specification [, create_specification] ...]

create_specification:
```

```
    [DEFAULT] CHARACTER SET charset_name
  | [DEFAULT] COLLATE collation_name
```

CREATE INDEX

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name [index_type]
    ON tbl_name (index_col_name,...)

index_col_name:
    col_name [(length)] [ASC | DESC]
```

CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)]
    [table_options] [select_statement]
```

Eða:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [() LIKE old_tbl_name ()];

create_definition:
    column_definition
  | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
  | KEY [index_name] [index_type] (index_col_name,...)
  | INDEX [index_name] [index_type] (index_col_name,...)
  | [CONSTRAINT [symbol]] UNIQUE [INDEX]
        [index_name] [index_type] (index_col_name,...)
  | [FULLTEXT|SPATIAL] [INDEX] [index_name] (index_col_name,...)
  | [CONSTRAINT [symbol]] FOREIGN KEY
        [index_name] (index_col_name,...) [reference_definition]
  | CHECK (expr)

column_definition:
    col_name type [NOT NULL | NULL] [DEFAULT default_value]
        [AUTO_INCREMENT] [[PRIMARY] KEY] [COMMENT 'string']
        [reference_definition]

type:
    TINYINT[(length)] [UNSIGNED] [ZEROFILL]
  | SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
  | MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
  | INT[(length)] [UNSIGNED] [ZEROFILL]
  | INTEGER[(length)] [UNSIGNED] [ZEROFILL]
  | BIGINT[(length)] [UNSIGNED] [ZEROFILL]
  | REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
  | DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]
  | NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL]
  | DATE
  | TIME
  | TIMESTAMP
  | DATETIME
  | CHAR(length) [BINARY | ASCII | UNICODE]
  | VARCHAR(length) [BINARY]
  | TINYBLOB
  | BLOB
  | MEDIUMBLOB
  | LONGBLOB
  | TINYTEXT
  | TEXT
```

```
  | MEDIUMTEXT
  | LONGTEXT
  | ENUM(value1,value2,value3,...)
  | SET(value1,value2,value3,...)
  | spatial_type

index_col_name:
    col_name [(length)] [ASC | DESC]

reference_definition:
    REFERENCES tbl_name [(index_col_name,...)]
               [MATCH FULL | MATCH PARTIAL]
               [ON DELETE reference_option]
               [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

table_options: table_option [table_option] ...

table_option:
    {ENGINE|TYPE} = {BDB|HEAP|ISAM|InnoDB|MERGE|MRG_MYISAM|MYISAM}
  | AUTO_INCREMENT = value
  | AVG_ROW_LENGTH = value
  | CHECKSUM = {0 | 1}
  | COMMENT = 'string'
  | MAX_ROWS = value
  | MIN_ROWS = value
  | PACK_KEYS = {0 | 1 | DEFAULT}
  | PASSWORD = 'string'
  | DELAY_KEY_WRITE = {0 | 1}
  | ROW_FORMAT = { DEFAULT | DYNAMIC | FIXED | COMPRESSED }
  | RAID_TYPE = { 1 | STRIPED | RAID0 }
        RAID_CHUNKS = value
        RAID_CHUNKSIZE = value
  | UNION = (tbl_name[,tbl_name]...)
  | INSERT_METHOD = { NO | FIRST | LAST }
  | DATA DIRECTORY = 'absolute path to directory'
  | INDEX DIRECTORY = 'absolute path to directory'
  | [DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]

select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select statement)
```

Þöglar breytingar á dálkaskilgreiningum

Í sumum tilfellum breytast tegundir sem skilgreindar eru við CREATE TABLE og ALTER TABLE skipanir yfir í aðrar tegundir en beðið var um. Helstu breytingar eru sem hér segir:

- VARCHAR dálkar með lengd undir fjórum er breytt í CHAR.
- Ef töfludálkur hefur breytilega lengd, verður línan einnig af breytilegri lengd. Ef tafla notar VARCHAR, TEXT eða BLOB er öllum CHAR dálkum (lengri en þrír stafir) breytt í VARCHAR.
- CAR og VARCHAR dálkum sem skilgreindir eru lengri en 255 er breytt í minnstu TEXT tegundina sem geymt getur hina skilgreindu lengd. Dæmi: VARCHAR(500) verður MEDIUMTEXT.
- Skilgreiningar á TIMESTAMP birtingar-stærðum (e. Display length) eru sniðgengnar frá og með MySQL 4.1.

- Ekki er hægt að setja TIMESTAMP dálk á NULL, sé það reynt er hann settur á dagsetningu þess (gildandi) dags. Þess vegna er tilgangslaust að skilgreina TIMESTAMP dálk sem NULL eða NOT NULL.
- Dálkar sem eru hluti af frumlykli eru settir sem NOT NULL jafnvel þó þeir séu ekki skilgreindir þannig.
- Slef-orðabil (e. Trailing spaces) eru sjálfkrafa fjarlægð úr ENUM og SET gildum þegar taflan með slíkum dálkum er smíðuð.
- MySQL varpar (e. Maps) sumum dálkategundum annarra SQL þjóna yfir í MySQL tegundir, samanber:

Aðrir þjónar	MySQL tegund
BINARY(M)	CHAR(M) BINARY
CHAR VARYING(M)	VARCHAR(M)
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT
INT2	SMALLINT
INT3	MEDIUMINT
INT4	INT
INT8	BIGINT
LONG VARBINARY	MEDIUMBLOB
LONG VARCHAR	MEDIUMTEXT
LONG	MEDIUMTEXT (MySQL 4.1.0 on)
MIDDLEINT	MEDIUMINT
VARBINARY(M)	VARCHAR(M) BINARY

- Ef notuð er USING klausa til að skilgreina ólöglega atriðisskrá (Index) fyrir valda töflutegund en önnur tegund atriðisskrár er í boði án þess að breyta niðurstöðum fyrirspurna, er sú tegund valin.

DROP DATABASE

```
DROP DATABASE [IF EXISTS] db_name
```

Eyðir skrám með þessum endingum: `.BAK` , `.DAT` , `.HSH` , `.ISD, .ISM` , `.ISM` , `.MRG` , `.MYD`, `.MYI` , `.db` , `.frm`

DROP INDEX

```
DROP INDEX index_name ON tbl_name
```

DROP TABLE

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

RENAME TABLE

```
RENAME TABLE tbl_name TO new_tbl_name
    [, tbl_name2 TO new_tbl_name2] ...
```

MySQL Verkfæra skipanir

DESCRIBE

Upplýsingar um dálka

```
{DESCRIBE | DESC} tbl_name [col_name | wild]
```

USE

```
USE db_name
```

MySQL færslu og læsinga skipanir

START TRANSACTION, COMMIT, og ROLLBACK

MySQL keyrir sjálfkrafa í „autocommit“ ham, sem þýðir að um leið og uppfærslu skipun er gefin er uppfærslan geymd á disk. Séu „færslu-færar“ töflur notaðar (`InnoDB`, `BDB`), má gera autocommit haminn óvirkan með:

```
SET AUTOCOMMIT=0;
```

Ef ætlunin er að gera autocommit ham óvirkan fyrir aðeins eina seríu (röð) af skipunum má gefa setninguna START TRANSACTION:

```
START TRANSACTION;
SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summary=@A WHERE type=1;
COMMIT;
```

Skipanir sem ekki er mögulegt að „velta til baka“

Sumum skipunum s.s. í gagna-skilgreiningar málinu (DDL) er ekki hægt að velta til baka (e. Rollback), s.s. þeim sem smíða eða henda gagnagrunnum og þeim sem smíða, henda eða breyta töflum.

Forðast skyldi að nota slíkar skipanir í færslu (e. Transaction). Ef ein færsla framkvæmir skipun sem gerir aðra færslu ófæra um að staðfesta (e. Commit) þá er ekki hægt að velta-til-baka með ROLLBACK.

Skipanir sem sem óbeint framkvæma staðfestingu

Eftirtaldar skipanir enda færslur óbeint, rétt eins og framkvæmt væri COMMIT rétt á undan framkvæmd þeirra.

`ALTER TABLE` , `BEGIN` , `CREATE INDEX`, `DROP DATABASE`, `DROP INDEX` , `DROP TABLE` , `LOAD MASTER DATA`, `LOCK TABLES`, `RENAME TABLE`, `SET AUTOCOMMIT=1`, `START TRANSACTION` , `TRUNCATE TABLE`.

SAVEPOINT og ROLLBACK TO SAVEPOINT

```
SAVEPOINT identifier
ROLLBACK TO SAVEPOINT identifier
```

LOCK TABLES og UNLOCK TABLES

```
LOCK TABLES
    tbl_name [AS alias] {READ [LOCAL] | [LOW_PRIORITY] WRITE}
    [, tbl_name [AS alias] {READ [LOCAL] | [LOW_PRIORITY] WRITE}] ...
UNLOCK TABLES
```

SET TRANSACTION

```
SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
{ READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }
```

Skipanir fyrir gagnagrunns stjórn

Skipanir fyrir notanda meðhöndlun

DROP USER

```
DROP USER user
```

GRANT og REVOKE

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON {tbl_name | * | *.* | db_name.*}
    TO user [IDENTIFIED BY [PASSWORD] 'password']
        [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
    [REQUIRE
        NONE |
        [{SSL| X509}]
        [CIPHER 'cipher' [AND]]
        [ISSUER 'issuer' [AND]]
        [SUBJECT 'subject']]
    [WITH [GRANT OPTION | MAX_QUERIES_PER_HOUR count |
                          MAX_UPDATES_PER_HOUR count |
                          MAX_CONNECTIONS_PER_HOUR count]]
REVOKE priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON {tbl_name | * | *.* | db_name.*}
    FROM user [, user] ...

REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```

Fyrir allar `GRANT` og `REVOKE` skipanir, getur `priv_type` verið hver sem er af:

Réttindi	Merking
`ALL [PRIVILEGES]`	Setur öll einföld réttindi nema `GRANT OPTION`.
`ALTER`	Leyfir notkun á `ALTER TABLE`.
`CREATE`	Leyfir notkun á `CREATE TABLE`.
`CREATE TEMPORARY TABLES`	Leyfir notkun á `CREATE TEMPORARY TABLE`.
`DELETE`	Leyfir notkun á `DELETE`.
`DROP`	Leyfir notkun á `DROP TABLE`.
`EXECUTE`	Leyfir notanda að inna geymdar-stefjur (e. Stored procedures).
`FILE`	Leyfir notkun á `SELECT ... INTO OUTFILE` og `LOAD DATA INFILE`.
`INDEX`	Leyfir notkun á `CREATE INDEX` og `DROP INDEX`.

INSERT	Leyfir notkun á INSERT.
LOCK TABLES	Leyfir notkun á LOCK TABLES.
PROCESS	Leyfir notkun á SHOW FULL PROCESSLIST.
REFERENCES	Óútfært.
RELOAD	Leyfir notkun á FLUSH.
REPLICATION CLIENT	Leyfir notanda að spyrja hvar meistara (e. Master) miðlarar eru.
REPLICATION SLAVE	Notað af þrælum (e. Slaves) til lesturs atburðadagbókar frá meistara.
SELECT	Leyfir notkun á SELECT.
SHOW DATABASES	SHOW DATABASES sýnir alla gagnagrunna.
SHUTDOWN	Leyfir notkun á mysqladmin shutdown.
SUPER	Leyfir notkun á CHANGE MASTER, KILL, PURGE MASTER LOGS, og SET GLOBAL skipunum, mysqladmin debug skipuninni; leyfir notanda að tengjast (einu sinni) þó hámarksfjölda tenginga (max_connections) sé náð.
UPDATE	Leyfir notkun á UPDATE.
USAGE	Samheiti fyrir „no privileges“ þ.e. engin réttindi.
GRANT OPTION	Leyfir notanda að úthluta réttindum.

SET PASSWORD

```
SET PASSWORD = PASSWORD('some password')
SET PASSWORD FOR user = PASSWORD('some password')
```

Skipanir fyrir töfluviðhald

ANALYZE TABLE

```
ANALYZE [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name [, tbl_name] ...
```

BACKUP TABLE

```
BACKUP TABLE tbl_name [, tbl_name] ... TO '/path/to/backup/directory'
```

CHECK TABLE

```
CHECK TABLE tbl_name [, tbl_name] ... [option] ...

option = {QUICK | FAST | MEDIUM | EXTENDED | CHANGED}
```

Tegund	Merking
QUICK	Ekki skanna línur til að finna rangar tengingar.
FAST	Athuga aðeins töflur sem ekki var lokað almennilega.
CHANGED	Athuga aðeins töflur sem hafa breyst eða ekki var lokað almennilega.
MEDIUM	Skanna línur til að sannreyna að eyddar tengingar séu í lagi. Reiknar einnig prófsummu[77] (e. checksum) fyrir línurnar og sannreynir við útreiknaða prófsummu út frá lyklunum.
EXTENDED	Flettir upp öllum lyklum fyrir hverja línu. Þetta tryggir að taflan er 100% sjálfkvæm (e. Consistent) en tekur langan tíma.

CHECKSUM TABLE

```
CHECKSUM TABLE tbl_name [, tbl_name] ... [ QUICK | EXTENDED ]
```

OPTIMIZE TABLE

```
OPTIMIZE [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name [, tbl_name] ...
```

REPAIR TABLE

```
REPAIR [LOCAL | NO_WRITE_TO_BINLOG] TABLE
    tbl_name [, tbl_name] ... [QUICK] [EXTENDED] [USE_FRM]
```

RESTORE TABLE

```
RESTORE TABLE tbl_name [, tbl_name] ... FROM '/path/to/backup/directory'
```

SET og SHOW

SET leyfir að skilgreindar séu breytur og valmöguleikar.

SHOW er til í ýmsum útgáfum sem gefa upplýsingar um gagnagrunna, töflur, dálka og stöðu mála hjá þjóninum, samanber:

```
SHOW [FULL] COLUMNS FROM tbl_name [FROM db_name] [LIKE 'pattern']
SHOW CREATE DATABASE db_name
SHOW CREATE TABLE tbl_name
SHOW DATABASES [LIKE 'pattern']
SHOW [STORAGE] ENGINES
SHOW ERRORS [LIMIT [offset,] row_count]
SHOW GRANTS FOR user
SHOW INDEX FROM tbl_name [FROM db_name]
SHOW INNODB STATUS
SHOW [BDB] LOGS
SHOW PRIVILEGES
SHOW [FULL] PROCESSLIST
```

[77] C**hecksum**: P**rófsumma** er summa sem fæst með því að leggja saman gildi í tilteknu svæði meðal færslna og er notuð til þess að fylgjast með því hvort gögn eru rétt. Samanlagður skattfrádráttur hóps einstaklinga, reiknaður í launaforriti. Gögnin eru annaðhvort töluleg eða stafastrengir sem litið er á sem töluleg gögn svo að reikna megi prófsummuna.

```
SHOW STATUS [LIKE 'pattern']
SHOW TABLE STATUS [FROM db_name] [LIKE 'pattern']
SHOW [OPEN] TABLES [FROM db_name] [LIKE 'pattern']
SHOW [GLOBAL | SESSION] VARIABLES [LIKE 'pattern']
SHOW WARNINGS [LIMIT [offset,] row_count]
```

SET

```
SET variable_assignment [, variable_assignment] ...

variable_assignment:
      user_var_name = expr
    | [GLOBAL | SESSION] system_var_name = expr
    | @@[global. | session.]system_var_name = expr
```

SHOW CHARACTER SET

```
SHOW CHARACTER SET [LIKE 'pattern']
```

SHOW COLLATION

```
SHOW COLLATION [LIKE 'pattern']
```

SHOW COLUMNS

```
SHOW [FULL] COLUMNS FROM tbl_name [FROM db_name] [LIKE 'pattern']
```

SHOW CREATE DATABASE

```
SHOW CREATE DATABASE db_name
```

SHOW CREATE TABLE

```
SHOW CREATE TABLE tbl_name
```

SHOW DATABASES

```
SHOW DATABASES [LIKE 'pattern']
```

SHOW ENGINES

```
SHOW [STORAGE] ENGINES
```

SHOW ERRORS

```
SHOW ERRORS [LIMIT [offset,] row_count]
SHOW COUNT(*) ERRORS
```

SHOW GRANTS

```
SHOW GRANTS FOR user
```

SHOW INDEX

```
SHOW INDEX FROM tbl_name [FROM db_name]
```

SHOW INNODB STATUS

```
SHOW INNODB STATUS
```

SHOW LOGS
```
SHOW [BDB] LOGS
```

SHOW PRIVILEGES
```
SHOW PRIVILEGES
```

SHOW PROCESSLIST
```
SHOW [FULL] PROCESSLIST
```

SHOW STATUS
```
SHOW STATUS [LIKE 'pattern']
```

SHOW TABLE STATUS
```
SHOW TABLE STATUS [FROM db_name] [LIKE 'pattern']
```

SHOW TABLES
```
SHOW [OPEN] TABLES [FROM db_name] [LIKE 'pattern']
```

SHOW VARIABLES
```
SHOW [GLOBAL | SESSION] VARIABLES [LIKE 'pattern']
```

SHOW WARNINGS
```
SHOW WARNINGS [LIMIT [offset,] row_count]
SHOW COUNT(*) WARNINGS
```

Kerfisumsjóna aðgerðir

CACHE INDEX
```
CACHE INDEX
  tbl_index_list [, tbl_index_list] ...
  IN key_cache_name

tbl_index_list:
  tbl_name [[INDEX] (index_name[, index_name] ...)]
```

FLUSH
```
FLUSH [LOCAL | NO_WRITE_TO_BINLOG] flush_option [, flush_option] ...
```
FLUSH skipanir hreinsa ýmis innri biðminni sem MySQL notar. Til að nota FLUSH þarf notandinn að hafa RELOAD réttindi.

`flush_option` getur verið hver sem er af:
```
HOSTS
DES_KEY_FILE
LOGS
PRIVILEGES
QUERY CACHE
STATUS
{TABLE | TABLES} [tbl_name [, tbl_name] ...]
TABLES WITH READ LOCK
```

```
USER_RESOURCES
```

KILL

```
KILL [CONNECTION | QUERY] thread_id
```

LOAD INDEX INTO CACHE

```
LOAD INDEX INTO CACHE
  tbl_index_list [, tbl_index_list] ...

tbl_index_list:
  tbl_name
    [[INDEX] (index_name[, index_name] ...)]
    [IGNORE LEAVES]
```

RESET

```
RESET reset_option [, reset_option] ...
reset_option getur verið hver sem er af:
```

MASTER	Eyðir öllum tvíundar-dagbókum (e. Binary logs) í atriðisskránni, endursetur tvíundar-dabækurnar sem tómar og smíðar nýja tvíundar-dabækur (e. Binary log files).
QUERY CACHE	Fjarlægir allar niðurstöður fyrirspurna úr fyrirspurna-biðminni.
SLAVE	Fær þræl til að gleyma „replication“ stöðu sinni í tvíundar-dagbókum.

Skipanir fyrir „Replication“

SQL skipanir sem stjórna meistara-miðlurum

PURGE MASTER LOGS

```
PURGE {MASTER | BINARY} LOGS TO 'log_name'
PURGE {MASTER | BINARY} LOGS BEFORE 'date'
```

RESET MASTER

```
RESET MASTER
```

SET SQL_LOG_BIN

```
SET SQL_LOG_BIN = {0|1}
```

SHOW BINLOG EVENTS

```
SHOW BINLOG EVENTS
   [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]
```

SHOW MASTER LOGS

```
SHOW MASTER LOGS
```

SHOW MASTER STATUS

```
SHOW MASTER STATUS
```

SHOW SLAVE HOSTS

```
SHOW SLAVE HOSTS
```

SQL aðgerðir sem stjórna þræla-miðlurum

CHANGE MASTER TO

```
CHANGE MASTER TO master_def [, master_def] ...

master_def:
      MASTER_HOST = 'host_name'
    | MASTER_USER = 'user_name'
    | MASTER_PASSWORD = 'password'
    | MASTER_PORT = port_num
    | MASTER_CONNECT_RETRY = count
    | MASTER_LOG_FILE = 'master_log_name'
    | MASTER_LOG_POS = master_log_pos
    | RELAY_LOG_FILE = 'relay_log_name'
    | RELAY_LOG_POS = relay_log_pos
    | MASTER_SSL = {0|1}
    | MASTER_SSL_CA = 'ca_file_name'
    | MASTER_SSL_CAPATH = 'ca_directory_name'
    | MASTER_SSL_CERT = 'cert_file_name'
    | MASTER_SSL_KEY = 'key_file_name'
    | MASTER_SSL_CIPHER = 'cipher_list'
```

LOAD DATA FROM MASTER

```
LOAD DATA FROM MASTER
```

LOAD TABLE tbl_name FROM MASTER

```
LOAD TABLE tbl_name FROM MASTER
```

MASTER_POS_WAIT()

```
SELECT MASTER_POS_WAIT('master_log_file', master_log_pos)
```

RESET SLAVE

```
RESET SLAVE
```

SET GLOBAL SQL_SLAVE_SKIP_COUNTER

```
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = n
```

SHOW SLAVE STATUS

```
SHOW SLAVE STATUS
```

START SLAVE

```
START SLAVE [thread_type [, thread_type] ... ]
START SLAVE [SQL_THREAD] UNTIL
    MASTER_LOG_FILE = 'log_name', MASTER_LOG_POS = log_pos
START SLAVE [SQL_THREAD] UNTIL
    RELAY_LOG_FILE = 'log_name', RELAY_LOG_POS = log_pos

thread_type: IO_THREAD | SQL_THREAD
```

STOP SLAVE

```
STOP SLAVE [thread_type [, thread_type] ... ]

thread_type: IO_THREAD | SQL_THREAD
```

Viðauki b: Hugtakaskrá

Enska	Íslenska
Aggregate	Samsetja, Samfelling
Attribute	Eigindi, sh. Eiginleiki
Candidate keys	Líklegir lyklar
Column	Dálkur
Data collection, Collection of Data	Gagnasafn
Data	Gögn
Database Management System, DBMS	Gagnasafns umsjónarkerfi
Database Schema	Gagnagrunns grind
Database	Gagnagrunnur
DDL, Data Definition Language	Gagna-skilgreiningarmál
DML, Data Manipulation Language	Gagna-meðhöndlunarmál
Domain	Gildamengi
Entity	Einindi
Field	Svið
Foreign key	Framandlykill
Function	Fall
Information	Upplýsingar
JOIN	Samtengja?
Keys	Lyklar
Metadata	Duldargögn
Primary key	Einkvæmur lykill
Primary Keys	Aðallyklar
QBE, Query By Example	Fyrirspurn-eftir-dæmi
Query	Fyrirspurn
Record	Færsla
Referential integrity	Vísunar heilleiki, tilvísunar heilleiki.
Relation	Tafla
Relationship	Skyldleiki, Vensl
Results	Niðurstöður
Result value	Skilagildi
Row	Röð, Lína
Schema, sh. Database schema, table schema	Gagnagrind, Gagnagrunnsgrind, töflugrind.

SQL, Structured Query Language	Heiti á fyrirspurnarmáli fyrir töflugagnasöfn.
Super Key	Yfirlykill, aðallykill
Table	Tafla, Vensl
Tuple	Lína
Unique key	Einkvæmur lykill
Unique primary key	Einkvæmur aðallykill
Unique	Einkvæmt
Variable	Breyta
View	Sýnd, Ásýnd

Viðauki c: my.ini eða my.cnf

Eftirfarandi er innihald my.ini skrárinnar af Windows vélinni hjá mér þegar bókin var skrifuð. Sama skrá á Linux nefnist my.cnf. Allar línur sem byrja á hassmerki (#) eru „athugaðar út“ (e. Commented out) þannig að þær virka sem athugasemdir en ekki raunverulegar skipanir. Tilgangur skrárinnar er sá að setja gildi fyrir víðværar breytur MySQL forritanna. Það forrit sem skal stilla er nefnt innan hornklofa, breytan er rituð á sniðinu `nafn=gildi`.

```
#This File was made using the WinMySQLAdmin 1.4 Tool
#16.6.2004 08:09:05

#Uncomment or Add only the keys that you know how works.
#Read the MySQL Manual for instructions

[mysqld]
basedir=D:/abc/mysql-500
datadir=D:/abc/mysql-500/data
language=D:/abc/mysql-500/share/icelandic
old-passwords
max-connections=150
# hér á eftir eru innoDB breytur samkv handbók. Best að eyða
# gagnaskránum á meðan þjónninn er niðri, við endurnýjun
innodb_data_home_dir=d:/abc/mysql-500/data
innodb_data_file_path = ibdata1:20M:autoextend
set-variable = innodb_buffer_pool_size=50M
set-variable = innodb_additional_mem_pool_size=10M
set-variable = innodb_log_file_size=10M
set-variable = innodb_log_buffer_size=8M
innodb_flush_log_at_trx_commit=1
log_error=X:/afrit-logs/mysql5-errors
# séu næstu tvær skrár afmerktar hefst dagbókarskráning
# ræsingu miðlarans við næstu
#log_bin=x:/afrit-logs/gorinnosho-bin-log
#log=x:/afrit-logs/almenn-dagbok.log
#binlog-ignore-db=mysql

# Eftirtalinn notandi mætti vera sá sem á að sjá alla grunna
# nota mysqladmin til að setja pwd
[WinMySQLadmin]
user=rotin
password=giskadu
Server=D:/abc/mysql-500/bin/mysqld-nt.exe
QueryInterval=10

# Setja stillingar fyrir biðlarann
```

```
[mysql]
max_allowed_packet=10000000
#tee=d:/glosur_nuna.sql
```

Viðauki d: Tólf gagnagrunnsreglur Dr. Codds

Árið 1985 gaf Ted Codd út lista 12 reglna sem taka saman einskonar heildarviðmiðun fyrir æskilega venslaða gagnagrunns útgáfu. Þessar reglur eru hafðar í huga við þróun flestra gagnasafns-umsjónarkerfa. Viðmiðanir eru e.t.v. betra hugtak frekar en reglur því ekkert kerfi fylgir þessum reglum til fullnustu en öll miða þó við þær.

Þess vegna er sjaldnast ritað ýtarlega um þessar reglur í umfjöllun um gagnasöfn því þær eru fremur óhlutlægar (e. Abstracted). Þó var til skamms tíma venja margra að bera mismunandi kerfi saman eftir því hversu vel þau fylgdu þessum viðmiðunum.

Ég birti þessar reglur hér í þeim tilgangi að styðja við umfjöllun mína um gagnagrunns hugtök. Sumar þessara viðmiðana eru erfiðar í útfærslu, aðrar frekar sjálfsagðar. Allir sem starfa við þróun gagnagrunna geta þó haft gagn af að þekkja þær. Ég hef þó ekki sett þessar reglur hér fram í upprunalegri mynd.

Regla 0: Grunnreglan: Kerfið skal geta tengt (venslað) upplýsingar, vera gagnagrunnskerfi og geta stjórnað (skilgreint og meðhöndlað) gagnagrunna. Kerfið skuli nota eingöngu (e. Exclusively) vensla eiginleika sína í þessum tilgangi.

Regla 1: Upplýsingareglan (e. Information): Allar upplýsingar í gagnagrunni skulu táknaðar á einn, og aðeins einn veg, sem gildi í dálkum og línum taflna.

Regla 2: Tryggt aðgengi (e. Guaranteed access): Öll gögn skulu aðgengileg án tvíræðni. Þetta er önnur lýsing á aðallykli: Hvert gildi í grunninum skal skiljanlegt og aðgengilegt í gegnum nafn töflunnar sem hýsir það, dálksins sem geymir það og lykils þeirrar línu sem á það[78].

Regla 3: Kerfisbundin meðhöndlun NULL gilda (e. Systematic treatment of null values): Gagnasafns-umsjónarkerfi verður að leyfa hverju sviði að vera tómt þ.e. NULL. Nánar tiltekið (e. Specifically) verður það að tákna „fjarveru gildis“ eða „ónothæft gildi“ sem sé skýrt aðgreinanlegt frá öðrum gildum svosem tölum (og núll tölum) eða strengjum (og tómum strengjum). Slík gildi skal vera hægt að meðhöndla kerfisbundið í kerfinu.

Regla 4: Virk innankerfis gagnaskrá byggð á venslalíkaninu (e. Active online catalog based on the relational model): Innan kerfisins og aðgengilegt notendum þess skal vera lýsing á gagnaskránum (e. Catalogs) eða gagna-, töflugrindunum (e. Schemas). Þessa lýsingu skulu notendur geta skoðað með sama fyrirspurnamáli og þeir nota til aðgengis að gögnunum sem grunnurinn geymir.

Regla 5: Fullvaxið og ítarlegt gagnatungumál: (Comprehensive data sublanguage rule): Kerfið skal styðja í.þ.m. eitt tungumál sem byggist á venslalíkani, hefur línulega málskipan (e. Linear syntax), nota má gagnvirkt (e. Interactive) og í kótuðum forritum sem eiga samskipti við gagnasafns-umsjónarkerfið. Gagnatungumálið skal geta skilgreint aðgerðir (e. Operations), ásýnd (e. Views) gagna, gagnasókn (e. Retrieval), dagréttingar, öryggis og ákvæða skilgreiningar og færslumeðhöndlun (e. Transaction controls).

[78] Þetta er hið sama og á ensku kallast „a transparent meaning of“. Nafn töflunnar, dálksins og lykilsins er gegnsætt í þeim skilningi að gögnin sem sótt eru sjást þar í gegn. Kennitala einstaklings er þannig gegnsæ: upplýsingakerfi geta séð einstaklinginn í gegnum kennitöluna. Hugtakanotkun af þessu tagi er algeng í hinum engilsaxneska heimi.

Regla 6: Ásýnda dagréttingar (e. View updating): Allar ásýndir sem fræðilega mætti dagrétta skal mega dagrétta í kerfinu.

Regla 7: Æðra stigs innskot, eyðing og dagrétting (e. High-level insert, delete and update): Kerfið skal styðja við aðgerðir sem geta framkvæmt innskot, eyðingu og dagréttingar á gögnum þar sem hver aðgerð getur farið fram ein í einu.

Regla 8: Raunlægt gagnasjálfstæði (e. Physical data independence): Raunlæg (e. Physical) geymsla gagnanna skal vera ósýnileg notandanum svo notendur hafi aðgang að þeim eftir þörfum en ekki geymsluhætti.

Regla 9: Röklægt gagnasjálfstæði (e. Logical data independence): Hafi notandi skilgreint ásýnd á gögn ætti röklæg (e. Logical) uppbygging gagnanna (t.d. taflna, dálka og lína) ekki að breyta ásýndinni.

Regla 10: Heilleika sjálfstæði (e. Integrity independence): Ákvæði um heilleika gagna skyldu geymd áháð þeim forritum sem sækja í gögnin og geymd í gagnaskránni (e. Catalog). Óhætt þarf að vera að breyta ákvæðum án þess að breyta þurfi forritunum.

Regla 11: Dreifðar-sjálfstæði (e. Distribution independence): Ef dreifa þarf hluta gagnagrunnsins á mismunandi staði (t.d. jafna á netþjóna) þá skyldi það vera hulið notendum grunnsins og forrit sem sækja í gögnin ættu að vinna með sama árangri og þegar dreifð útgáfa grunnsins var fyrst kynnt og þegar gögnum er dreift innan kerfisins.

Regla 12: Ekkert niðurrif (e. Nonsubversion): Bjóði kerfið upp á lægri línu skil (e. Low-level record interface) s.s. þegar forrit geta fengið beint aðgengi á töflur og línur, skal kerfið ekki leyfa að öryggis, vensla eða heilleika ákvæði séu sniðgengin.

Viðauki e: Ýtarlegra efnisyfirlit