2. Andmebaasid ja andmemudelid

2.1. Andmebaasi "füüsiline kuju"

andmebaas on fail või kogum faile

Arvuti operatsioonisüsteemi mõttes on andmebaas fail või kogum faile. See kumb ta on, sõltub konkreetsest andmebaasisüsteemist. Ei saa öelda, et üks neist kahest variandist on levinum kui teine. Siiski on see variant, kus andmebaas koosneb mitmetest failidest, mitmekesisem, kuna erinevad andmebaasisüsteemid jagavad andmeid failidesse erinevalt. Samas on ka selliseid andmebaasisüsteeme, mis salvestavad mitu erinevat andmebaasi samasse faili või siis haldavad paljusid faile, kus on andmebaasid, kuid üks baas ei ole mitte ühes failis vaid andmebaase hoitakse nendes failides selliselt, et samas failis võib olla mitu andmebaasi ja üks andmebaas võib olla jagatud mitmete failide vahel.

arvuti operatsioonisüsteemi jaoks on andmebaasid lihtsalt failid

Arvuti operatsiooni süsteem näeb andmebaase kui faile ja ei erista neid faile teistest failidest mitte millegi poolest. Need on tavalised failid, mida võib kopeerida, kustutada jms. Tegelikkuses ei tohiks neid faile siiski käsitleda operatsioonisüsteemi vahenditega vaid andmebaasijuhtimissüsteemi vahenditega. See välistab suuremad eksimused.

Andmebaasi failid ei pea asuma ühes arvutis. Seda printsiipi saame me rakendada siiski ainult nende andmebaaside puhul, mille talletamiseks kasutatakse mitmeid faile. Vaatame nüüd seda kõike erinevate andmebaasisüsteemide näitel.

MySQL / Linux

MySQL andmebaas on kogum faile, mis asuvad samas ketta kataloogis (directory's). Iga tabel on eraldi fail. Lisaks sellel on selles kataloogis veel ka süsteemsed tabelid. Süsteemsetes tabelites hoitakse kasutajate nimesid, tabelite vaheliste seoste kirjeldusi jms. Lisaks sellel on selles teatmikus veel ka failid, kus hoitakse indeksite andmeid. Kasutajateks saavad olla Linuxi kasutajateks registreeritud isikud, kellele kirjeldatakse lisaks LInux' õigustele veel ka andmebaasi kasutajaõigused.

MS Visual FoxPro (DBF-formaat)

Ligikaudu sama on andmebaasistruktuur ka MS Visual FoxPro-l (DBF-formaat). Siin on iga tabel talletatud eraldi faili. Lisaks sellele on eraldi failides ka tabelite indeksid.

Unisys SQLBase

Unify corp andmebaasisüsteem SQLBase hoiab oma andmebaase tavajuhtumil ühes failis. Andmebaasi loomisel tehakse kettale "tühi" andmebaas, kuhu saab hakata looma tabeleid. Kui Linuxi MySql-I oli kasutajate baas i lauseks operatsioonisüsteemi kasutajad, siis SQLBase's luuakse igale andmebaasile omad kasutajad ja defineeritakse need igas baasis eraldi. See tähendab seda, et kui kahes baasis on sama kasutajanimi, siis seos nende vahel on määratud ainult sellega, et neid kasutajanimesid võib toimingute tegemiseks kasutada sama inimene. Aga neid kasutajanimesid võivad kasutada täiesti erinevad inimesed.

SQLBase on võimeline sama andmebaasi jagama ka osadeks ja neid osi talletame erinevatel ketastel *(partitioning)*.

MS Access

MS Access on andmebaasisüsteem, mis oma pisikesi andmebaase hoiab tingimusteta ühes failis. Kõik kasutajad on ainult ühe andmebaasi kesksed.

ORACLE, SyBase ja MS SQLServer

Suured andmebaasisüsteemid kasutavad eelnevalt kirjeldatust hoopis erinevat ketasmälu kasutamiseks reserveerimisemeetodit. Siin hakkab andmebaaside loomine pihta ketastel mälu eraldamiseks. Selleks kirjutatakse ketta pinna reserveerimiseks ketastele failid (operatsioonisüsteemi mõttes) ja registreeritakse need andmebaasisüsteemi mootoris kui "piirkonnad kuhu saab andmebaase teha". Need piirkonnad võivad olla erinevatel ketastel ja isegi erinevates arvutites Andmebaasi mootor käsitleb neid kui ühte sidusat piirkonda. Kui juba reserveeritud mälust jääb väheks saab olemasolevate piirkondadega igal hetkel liita uusi reserveeringuid. Seda muidugi juhul kui vaba ketta pinda on olemas.

Andmebaasi loomisel saab ette öelda millistesse piirkondadesse ja millises mahus andmebaas luuakse. Vajadusel saab hiljem, kui esialgselt eraldatud ruum otsa saab, andmebaasile uusi ketta piirkondi juurde lisada. Pärast andmebaasile mälu eraldamist käsitleb andmebaasi mootor seda kui ühtsed ja sidusat piirkonda ja paigutab loodavad tabelid sinna selliselt, et kasutaja ei saa arugi, et andmebaas paikneb füüsiliselt erinevatel ketastel.

Sellise skeemi järgi üles ehitatud andmebaasides luuakse ühised kasutajad üle kõigi sama andmebaasi mootoriga hallatavate andmebaaside ja kirjeldatakse neile õigused erinevate andmebaaside jaoks selles serveris.

2.2. Andmebaas kui tabelite kogum. Tabelid. Veerud. Andmetüübid

andmebaas kui tabelite kogum

Kõik andmebaasis olevad andmed talletatakse rist-tabelitena. Struktuuriliselt ongi andmebaas rist-tabelite kogum. Olgu meil vaja näiteks moodustada andmebaasi loend isikutest, kellega me peame läbirääkimisi tööle asumiseks. Esimese tööna tuleb meil välja mõelda see, milliseid andmeid me andmebaasis registreerida soovime. Nendeks andmeteks võiksid olla:

tabeli struktuur (tabel ISIK)

Veeru nimetus	Veeru semantika
eesnimi	Isiku eesnimi või mõnel juhul ka eesnimed
perenimi	Isiku perekonnanimi
isikukood	Isikukood, mis viitab konkreetsele inimesele ja võimaldab eristada sama nimelisi inimesi
kodakondsus	isikukood ei oma tähendust ilma kodakondsuseta, sest erinevates riikides võib olla isikuid, kellel on sama isikukood
sugu	Isiku sugu, mis mõnedes infosüsteemides (näiteks meditsiinisüsteemides, ei pruugi olla sugugi ainult mees ja naine. Sellest võivad sõltuda näiteks see, kui palju vastava soo riietusruumi kappe me vajame. Sugu sisaldub küll EESTI isikukoodis ja ka ilmselt mõnede välisriikide isikukoodides aga me ei tea, kus see seal täpselt iga riigi puhul asub ja kõigis riikides ei pruugi isikukoodi ollagi - seega peame seda eraldi veerus dubleerima. Töö lihtsustamiseks võime me Eesti isikukoodide puhul lasta programmil selle välja automaatselt täita.
sünnipäev	Isiku sünnikuupäev. Siin on sama asi, mis soo puhulgi – see sisaldub Eesti isikukoodis aga kuna tööle võivad tahta ka välismaalased, kellede isikukoodi struktuuri me ei tea, siis peame seda dubleerima. Töö lihtsustamiseks võime me Eesti isikukoodide puhul lasta programmil selle välja automaatselt täita
postiaadress	Aadress kirja teel suhtlemiseks.
E-posti aadress	Aadress E-posti teel suhtlemiseks.

telefon	Telefoni number suhtlemiseks telefoni teel.
haridustase	Isiku haridustase: 1 - põhiharidus, 2-gümnaasiumi haridus, 3-keskeri haridus, 3 -inseneri haridus, 4-bakalaureus, 5-magister, 6-doktor, 0-määramata
omandatud eriala	Omandatud eriala kood.

Nüüd tuleb luua andmebaas ja sellesse üks tabel ISIK. Selleks kasutame SQL-keelt:

NÄIDE:

```
CREATE DATABASE VARBAMINE;          /* andmebaasi nimes pole soovitav täppidega tähti kasutada */
CREATE TABLE ISIK (                 /* loome tabeli ISIK */
EESNIMI VARCHAR(25) NOT NULL,       /* eesnimi on kohustuslik */
PERENIMI VARCHAR(25) NOT NULL,      /* perenimi on kohustuslik */
ISIKUKOOD VARCHAR(20),              /* isikukood võib ka puududa või olla teistes riikides pikem kui 11 kohta */
KODAKONDSUS VARCHAR(60),            /* veergude kirjelduste vahel on eraldajaks koma */
SUGU CHAR(1) NOT NULL,
SYNNIPAEV DATE,                     /* veergude nimedes ei ole soovitav täppidega tähti kasutada */
AADRESS VARCHAR(100),
E_POST VARCHAR(50),                 /* veergude nimedes märki 'miinus' ei või kasutada. need asendatakse alakriipsudega */
TELEFON VARCHAR(20),                /* telefoninumber on alati tekst-tüüpi mitte täisarv-tüüpi, kuna temaga ei arvutata */
HARIDUSTASE CHAR(1) NOT NULL,       /* siia sisestatakse haridustaseme kood; ka see on tekst-tüüpi kuna temaga ei arvutata */
OMANDATUD_ERIALA VARCHAR(60)        /* välja nimedes tühikuid ei tohi olla - need asendatakse alakriipsudega */
);                                  /* SQL-lause lõppeb semi-kooloniga */
```

Pärast nende käskude täitmist on loodud andmebaas VARBAMINE (värbamine) ja sinna sisse loodud tabel ISIK. Tabelis on 11 veergu. Veergude nimed (ja ühtlasi ka pealkirjad) on näha ülal toodud CREATE lauses. See tabel on esialgu tühi.

andmetüübid tabeli kirjeldamisel

Vaatame nüüd andmetüüpe. Igal tabeli veerul peab olema määratud andmetüüp. Loodud tabelis on kasutatud andmetüüpe VARCHAR, CHAR ja DATE. VARCHAR ja CHAR on mõlemad teksti andmetüübid, mille lubatav suurim pikkus on 254 positsiooni. Number, mis on sulgudes andmetüübi võtmesõna järel (Näit. VARCHAR(100) või CHAR(10)) näitabki maksimaalset sümbolite arvu, mida on võimalik sellesse veergu talletada. Näiteks eesnimi saab olla maksimaalselt 25 sümbolit pikk. Pikemaid nimesid sellesse veergu kirjutada ei saa. Kui nimi on pikem, siis tuleb piirduda esimese 25 sümboliga või teha valik, millised nime osad sinna kirjutada, mingi muu kriteeriumi alusel. On see veerg defineeritud liiga lühike? Seda peab otsustama projekteerija. Peab küll ütlema, et veerge saab hiljem teha pikemaks. Samas ei saa veerge teha lühemaks. Nende kahe andmetüübi puhul on 254 maksimaalne pikkus.

VARCHAR vs. CHAR

Mis on nende kahe andmetüübi vahe? CHAR on ajalooliselt vanem ja VARCHAR uuem. Vahe on selles, kuidas neile andmebaasis mälu eraldatakse. VARCHAR "kulutab" mälu ökonoomsemalt kui CHAR. Nimelt eraldatakse andmebaasis VARCHAR tüüpi veergudes mälu parasjagu nii palju kui mitu sümbolit konkreetsesse tabeli välja on kirjutatud. CHAR reserveerib aga alati maksimaalse sümbolite hulga - just nii palju kui definitsioonis on kirjas. Näiteks, kui me oleks veeru EESNIMI defineerinud kui CHAR(25) siis oleks nime "Jaan Jalgratas" ja "Ly Kao" andmebaasi kirjutamisel mõlemal puhul reserveeritud 25 baiti (Unicode puhul muidugi 2 korda rohkem). Andmetüübi VARCHAR(25) korral reserveeritakse aga esimese korral 14 baiti ja teisel puhul ainult 6 baiti.

LONG VARCHAR

LONG VARCHAR on "lõpmatu" pikkusega teksti väli. "Lõpmatus" on tegelikult iga andmebaasisüsteemi korral siiski kirjeldatud kui mingi lõplik väga suur baitide hulk. Seda välja kasutatakse tavaliselt vabatekstiliste kommentaaride, kirjelduste jms. sisestamiseks andmebaasi. Sellesse välja saab kirjutada ka mistahes dokumente - DOC, XLS, AVI, HTML jms.

BINARY, BLOB, BITMAP, ...

Lisaks LONG VARCHAR andmetüübile on mitmetes andmebaasisüsteemides kasutusel veel ka andmetüübid BINARY, BLOB, BITMAP, mis on põhimõtteliselt LONG VARCAHR sünonüümideks, kuid

eksisteerivad tema kõrval ja on ettenähtud just dokumentide salvestamiseks andmebaasi. On võimalikud ka muud konkreetsete andmebaaside spetsiifikast tingitud analoogilised andmetüübid.

DATE

DATE on kuupäeva tüüp. Andmebaasis hoitakse kuupäeva järjenumbrina mingist "null kuupäevast" alates. Igal andmebaasisüsteemil on see "null-kuupäev" erinev. Aga tegelikult pole see üldse tähtis milline on see null-kuupäev. Oluline on teada seda, et tänu sellele on kuupäevad järjestatud ja ühest kuupäevast teise lahutamisel ja sellele kuupäevade vahel "ühe" liitmisel saame me teada nende kuupäevade vahele jääva päevade arvu. Kuupäevale mingi täisarvu liitmine annab meile selle täisarvu võrra hilisema kuupäeva. Täisarvu lahutamisel kuupäevast saame aga vastava päevade arvu võrra varasema kuupäeva. Tänu sellele, et kuupäeva andmebaasi talletusvorm on number, ei ole meil mingit takistust teisendada seda esitusel üks kõik millise riigi esitusvormi (DD.MM.YY, DD.MM.YYYY, MM/DD/YY vms.) Näiteks Windows' rakendused teisendavad kuupäeva tavaliselt ise Windows' vaikimisi kuupäeva formaati. Programmide kirjutamisel on aga võimalikud teisendused mistahes vajalikku formaati.

DATETIME ja TIMESTAMP

DATETIME ja TIMESTAMP mõlemad kirjeldavad andmetüüpi, kus samasse veergu saab kirjutada kuupäeva ja kellaaja. Nendel andmetüüpidel pole mingit vahet - nad on sünonüümid. Osades andmebaasisüsteemides kasutatakse ühte osades teist võtmesõna. On ka selliseid andmebaasisüsteeme, mis tunnistavad mõlemat. Nagu kuupäev-tüüpi (DATE) andmeid, hoitakse ka kuupäev/kellaaeg-tüüpi andmeid andmebaasis järjenumbritena - selle arvu täisosa näitab päevade arvu "null kuupäevast" ja murdosa millisekundite arvu kesköost. Andmete väljastusel on seda võimalik teisendada mistahes formaati. Windows' rakendused teisendavad selle tavaliselt Window' vaikimisi kuupäeva ja kellaaja formaati.

INTEGER, INT, SMALLINT, TINYINT

INTEGER on täisarvu fortmaat. INT on INTEGR' sünonüüm. INTEGER tähistab praegusel ajal enamasti 4-baidi pikkust täisarvu. st. seda, et minimaalne arv, mida saab baasi talletada on -4294967295 ja

maksimaalne 4294967295 (+/- 2^{32}-1). Siiski tuleb konkreetse andmebaasisüsteemi kasutamisel üle kontrollida - kui pikk see täisarv ikkagi on. INT või INTEGER andmetüüp on olemas igas andmebaasisüsteemis. Lisaks sellele võivad seal olla veel täisarvu andmetüübid SMALLINT ja TINYINT, mis on lühemad kui INTEGEF (INT), kuid siin mingit reeglit enam pole ja iga andmebaasisüsteemi korral tuleb juhendist lugeda, kui pikad nad tegelikult on. Siiski võib öelda, et suure tõenäosusega on SMALLINT kahe-baidine täisarv ja TINYINT ühe-baidine täisarv.

DECIMAL

DECIMAL on fikseeritud punkti (koma) kohaga kümnendarv. DECIMAL andmetüübi kirjeldamise korral näidatakse ära arvu pikkus (ilma märgita ja punktita) ja kümnendkohtade arv. Näiteks kui meil on vaja andmebaasi panna arvu, mille maksimaalne suurus on 999.99 ja minimaalne suurus -999,99, siis kirjeldame andmebaasi tabelisse veeru, mille tüüp on DECIMAL(5,2). Nagu öeldud miinus-märki arvu ees ja punkti (koma) arvu sees pikkuste arvutamisel arvesse tavaliselt ei võta. Siiski on olemas väikseid andmebaase, mis vahel käituvad teisiti. Seega tavaline reegel - kontrolli enne tööle hakkamist juhendist üle.

NOT NULL

Kui vaadata ülal toodu CREATE- lauset tabeli isik loomiseks siis avastame seal mitmel real fraasi "NOT NULL". See fraas määrab veerus, mis ei saa andmete sisestamisel jääda tühjaks st. ei ole võimalik luua tabelisse rida, kus selle fraasiga tähistatud veerus oleks väärtus määramata.

veel üks tabel (tabel TÖÖKOHT)

Ees pool sai tõdetud, et andmebaas on tabelite kogum. Värbamise ülesande lahendamiseks on meil vaja veel vähemalt ühte tabelit, kus meil on ära kirjeldatud vakantsed töökohad ja kui toimub inimese tööle võtmine, siis saab sinna märkida, kes millisele töökohale võeti, m,illisse palgaga millise koormusega ja mis ajast alates. Selles tabeli võiks olla järgmised veerud:

Veeru nimetus	**Veeru semantika**
Üksuse kood	Selle üksuse kood, kuhu värbamine toimub

tabeli struktuur (tabel TÖÖKOHT)

Üksuse nimetus	Selle üksuse nimetus, kuhu värbamine toimub
Ametikoha nimetus	Ametikoha nimetus, kuhu värbamine toimub
Eeldatav palk	Palk, millega töötajat otsitakse
Töötaja nimi	Töötaja, kes valiti töökohale välja; nii kaua kuni me töötajat pole välja valinud, on see väli tühi.
Isikukood	Töötaja isikukood; nii kaua kuni me töötajat pole välja valinud, on see väli tühi.
Tööle tuleku aeg	Kuupäev, millal töötaja tööd alustab; nii kaua kuni me töötajat pole välja valinud, on see väli tühi.
Koormus	Koormus, millega töötaja tööle võetakse; nii kaua kuni me töötajat pole välja valinud, on see väli tühi.
Soovitud palk	Palk, mida töötaja soovis; nii kaua kuni me töötajat pole välja valinud, on see väli tühi.
Kokku lepitud palk	Palk, milles kokku lepiti; nii kaua kuni me töötajat pole välja valinud, on see väli tühi.
Kommentaar	Vabatekstiline kommentaar.

Loome nüüd ka selle tabeli:

NÄIDE:

```
CREATE TABLE TOOKOHT (          /* loome tabeli TOOKOHT; tabeli nimes pole soovitav täpitähti kasutada */
YKSUS_KOOD VARCHAR(100) NOT NULL,   /* üksuse kood */
YKSUS_NIMI VARCHAR(100) NOT NULL,   /* üksuse nimetus */
AMETIKOHA_NIMI VARCHAR(60) NOT NULL,
EELDATAV PALK INTEGER NOT NULL,     /* kuna palk antakse täis-kroonides siis ei kasuta me siin DECIMAL-tüüpi vaid täisarvu tüüpi INTEGER*/
TOOTAJA VARCHAR(60),
ISIKUKOOD CHAR(20),
TOOLE_TULEKU_AEG DATE,
KOORMUS DECIMAL(4,2),           /* Arvu kogupikkus 4 kohta, pärast koma kaks kohta */
SOOVITUD_PALK INTEGER,
KOKKU_LEPITUD_PALK INTEGER,
KOMMENTAAR LONG VARCHAR         /* siia lubame kirjutada kommentaariks "kui tahes pika" teksti */
```

);

NULL vs. NOT NULL

Siin ei saa me NOT NULL fraasi kirjutada ühelegi veerule, mis väärtustatakse pärast värbamise läbi viimist (veerud alates veerust TOOTAJA). Seda selle pärast, et vakantsi kirjeldamisel me veel ei tea, kelle me konkreetsele ametikohale värbame. Samas vakantsi kirjeldamisel peame me igal juhul kohe väärtustama üksuse nime, ametikoha nime ja eeldatava palga. Ilma nende andmeteta ei ole vakants kirjeldatud.

Andmebaasisüsteem kontrollib nende reeglite täitmist ja takistab (annab veateate) selliste kirjete tabelisse lisamist, millele NOT NULL fraasiga veerud on väärtustamata.

seosed tabelite vahel

Otse kui "märkamatult" tekkis meil kahe tabeli vahel ka seos - isiku nime ja isikukoodi kaudu on loodud kaks tabelit seotud. Kui me vaatame isikut tabelis ISIK, siis me saame isikukoodi ja/või töötaja nime alusel tabelist VAKANTS teada, millisele ametikohale (ja kas üldse) konkreetne töötaja platseerus. Kui me leiame aga tabelist VAKANTS töötaja, siis isikukoodi ja nime alusel saame me tabelist ISIK teada tema kontaktandmed, haridustaseme ja omandatud eriala.

2.3. Andmemudel. Andmete modelleerimine

andmete modelleerimise tähtsus

Jaotises 2.2 tegime me lihtsa näite, kuidas kahe tabeliga pisikese andmebaasiga toetada värbamise protsessi. Tegemist oli väga lihtsustatud lahendusega, mille "projekteerimisel" me eriti vaeva ei näinud. Tegelikkuses, reaalselt kasutusse minevate süsteemide korral, on aga ka kõige lihtsamatena näivate ülesannete juures vaja teha läbi antud ülesande raskuskategooriale vastav modelleerimisprotsess. Seejuures ei tohi ühtegi ülesannet alahinnata ja loobuda esmapilgul "lihtsana" tunduva ülesande andmemudeli läbi mõtlemisest. Andmemudelis tehtud vead kanduvad üle andmemudeli peale ehitatavasse rakendusse ja iga viga, mida parandatakse andmemudelis, genereerib hulgaliselt parandusi rakenduskihis. Seega on andmemudelis tehtud vead kõige kallimad vead ja peaksid eriti ärikaalutlustel tehtud tarkvara tootmise olema välditud nii palju kui võimalik.

Lisaks sellele käib räpakalt ja "üle-pea-kaela" koostatud andmemudelitega kaasas veel üks oht. Kui rakenduse kirjutajad avastavad, et neil ei õnnestu antud andmemudeli peale kirjutada sellist programmi, mida on vaja, siis nad "leiavad lähenemise" kuidas see ikkagi ära teha. See tähendab, et tehakse andmebaasi "abi-kirjeid", lisatakse tunnuseid jms. ning tehakse IKKAGI asi ära. Seejuures arvatakse, et asja huvides on praegu vaja kiiresti edasi minna ja teha ajutine lahendus - küll hiljem tehakse asi korda. Selle tulemusena saadakse programmikood, mida hiljem on keeruline muuta. Pealegi saab sellest ajutisest lahendusest igavene lahendus. Seda arendatakse edasi, kuni saabub hetk, kus muudatusi ei ole enam lihtsalt võimalik teha, sest programm "käriseb" iga kandi pealt. Oleks vaja see viga lõpuks ära parandada. Selleks ajaks on aga too kunagine "pisi-viga" genereerinud süsteemi sellise koguse koodi, et parandamiseks tuleb suur osa lähtetekstist ümber kirjutada.

pisi-viga andmemudelis, mis aja möödudes muutub "hiiglaseks"

Oluliselt lihtsam ja odavam on andmemudel kohe projekteerida korralikult, kui kunagi hiljem hakata parandama kõike seda, mis oleks võinud olla olemata. Igale andmebaasi loomisele peab eelnema andmete modelleerimine.

andmete modelleerimise kohustuslikkus

2.3.1. Abielu andmemudel

Vaatame ühte üsna lihtsat andmemudelit ja seda, kuidas sellesse probleemi süvenemine aitab viia ta sellisel kujule, millega tulevikus suure tõenäosusega pole vaja enam tegeleda st. me esitame ta kohe parimal ja paindlikumal viisil. Selleks mudeliks on abielusuhete registreerimise mudel.

Meie kultuuriruumis on abielu klassikalises mõttes liit (ühe) mehe ja (ühe) naise vahel. Ühel mehel on korraga üks naine ja ühel naisel korraga üks mees. Vähemasti seaduse silma ees on asi nii. Samuti on hulk naisi ja mehi, kelle pole seaduslikku abikaasat - nad on kas liiga noored või pole nad leidnud kedagi enda jaoks sobivat. Osa on ka sellised kes on lesestunud. Samal ajal võivad inimesed

ülesande kirjeldus

olla abielus ka mitu korda - pärast lahutust või abikaasa surma võivad nad uuesti abielluda.

Vaatame, milline see mudel välja näeb:

Iga mees võib olla abielus kas üks ( ), mitu ( ) või ei ühtegi ( ) korda ( ). Sama pädeb naise kohta. Samas igas abielus saab olla üks ( ) mees ja üks ( ) naine. Pole olemas abielu, milles poleks osaline üks mees ja üks naine. Seoseid, mis on tabelite MEES ja ABIELUS OLEMINE ning NAINE ja ABIELUS OLEMINE

üks-mitmene suhe

Milliseid andmeid talletataks siin mudelis ühes või teises tabelis. Vaatame andmete minimaalset komplekti. Tabelites MEES ja NAINE on isikukood, kodakondsus, eesnimi (eesnimed), perekonnanimi, sünniaeg ja surmaaeg (kes on surnud, seda ei saa enam ühegi uue abieluga liita). Tabelis ABIELUS OLEMINE on vaja ära näidata abielu osalised (mees ja naine) ning abielu algus-kuupäev ja abielu lõpp-kuupäev ning abielutunnistuse number ja välja andnud asutus. Loomulikult jääb abielu sõlmimisel lõppkuupäev määramata (tegemist on tähtajatu lepinguga, üles ütlemisega kuuajase etteteatamisega :-). Uue abielu kirjutamisel baasi tuleb kontrollida, et kumbki abielu pooltest ei oma selle hetkel kehtivat abielu st. tabelis ABIELUS OLEMINE ei leidu kumbagi abielluja jaoks sellist kirjet, kus algus-kuupäev oleks minevikus ja lõpp-kuupäev määramata või tulevikus st. et abielu on juriidiliselt lõpetatud näiteks alates homsest. Sellisel juhul saab uue abielu sõlmida ka alles alates homsest.

ülesande 1. täpsustus

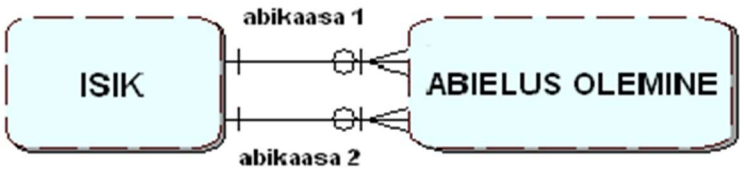
Kõik tundub korras olevat, aga arvestades praegust poliitilist olukorda, oleks väga vastutustundetu jätta arvestamata see, et varsti on päevakorras samasooliste abielud. Praegune skeem ei võimalda aga kahe sama soolise abielu registreerimist - abielus saavad olla ainult erinevast soost inimesed. Kuidas asja parandada? Asja parandamiseks tuleb mehed ja naised viia kokku ühte tabelisse. Need kaks tabelist on nii

või teisiti sama semantikaga (inimesed, abielu osapooled) ja nad on lahutatud jõuga soolise tunnuse alusel. (diskrimineerimine? kindlasti! Aga kelle?)

sama semantikaga tabelite liitmine

Tulemuseks saame tabeli ISIK, mis on kahe seosega seotud tabeliga ABIELUS OLEMINE. Tegelikult ei tohikski ilma väga hea põhjuseta lahutada ühtse semantikaga tabelit mingi tunnuse alusel kaheks või ka mitmeks erinevaks tabeliks. See muudab mudeli oluliselt raskepärasemaks, struktuurilt keerukamaks (suureneb seoste arv) ja ka raskemini käsitletavaks (programmid muutuvad keerukamaks ja ka kasutajatel tuleb töömahtu juurde). Igas andmemudelis peaks olema ainult üks tabel ISIK või YKSUS või AMET. Ei ole mingit mõtet jagada inimesi ameti, soo või mingi muu tunnuse järgi erinevatesse tabelitesse. Mõelda tuleb sellele, et erinevates rollides võivad erineda samad inimesed - olenemata sellest, millisesse tabelisse ja millise tunnuse alusel nad jagatud on. See tähendab aga seost kõikidesse nendesse tabelitesse. Seega suureneb suhete arv vähemasti nii mitu korda, kui mitmeks ühtse semantikaga tabel jagatud on

Pärast muudatust näeb andmemudel välja järgmine:



Kui eelmises mudelis oli tabelite MEES ja ABIELUS OLEMINE vahel oleva suhte semantika "abielumees" ja tabelite NAINE ning ABIELUS OLEMINE vahelise suhte semantika "abielunaine", siis antud skeemis on tegemist "võrdsete abikaasadega", kellede sugu pole abielu sõlmimisel enam üldse oluline. Küll aga võib olla sugu oluline statistika jaoks - soovitakse näiteks teada, kui palju on erisooliste paare, samasooliste

paare, homode paare, lesbide paare, samasooliste paarides olevaid homosid, kes on varem olnud hetero-suhtes jne. Seepärast tuleb tabelisse ISIK lisada tunnus "sugu". Kõik protseduurid jäävad samaks, muutub ainult see, et abielluvaid inimesi valitakse nüüd, sõltumata soost, samast tabelist. Nagu nägite lisandub muidugi teatav hulk statistikat. Seejuures jääb alles ilmselt ka kogu varasem statistiline aruandlus.

ülesande 2. täpsustus

Tundub, et ongi valmis, aga mõtleme nüüd natuke laiemalt - äkki tahaks oma infosüsteemi müüa ka mujal maailmas, seal on aga mõnel pool lubatud mitme naise või mehe pidamine - haaremid, mormoonid jms. Meie skeem seda otseselt ei võimalda. Võiksime küll registreerida ühele mehele mitu samaaegset abielu erinevate naistega või ühele naisele mitu samaaegset abielu mitme erineva mehega, kuid see on see sama "mudeli sobitamine", millest üleval pool juttu oli. Seega peaksime mudeliga midagi ette võtma. Mida siis?

oluline muudatus mudelis

Mis on see muudatus, mis meil haaremi lisamisega ülesandepüstitusse tekkis? Kui enne oli abieluakt selline, kus liitusid kas inimest mingil kuupäeva ja elasid koos mingi kuupäevani, siis nüüd tekkib olukord, kus abielusse võib liituda inimesi selle kestel ja ka ära minna, aga nii kaua kui haaremi omanik on elus kestab ka kogu abielu. Seega on abielu kui selline muutunud "üheks abielu osapooleks" millega liidetakse teisi osapooli so. isikuid. Ja nii kaua, kui abieluga on seotud selle abielu valdaja (kas mees kellel on palju naisi või naine kellel on palju mehi) ja vähemakt üks teise sugupoole esindaja, on jõus ka abielu. Samasse abiellu liidetakse haaremi kasvamisel uusi isikuid ja kui nad surevad, siis lahkuvad nad haaremi koosseisust.

Uus mudel on järgmine:

Tabeli ISIK sisu jääb täpselt samaks. Tabelis ABIELU on andmed abielutunnistuse numbri, välja andmise kuupäeva, väljaandja, abielu alguse kuupäeva ja abielu lõpu kuupäeva kohta. Tabeli

ABIELUS olemise sisu muutus selliselt, et kui enne sidus iga selle tabeli kirje kahte inimest, siis nüüd kirjeldab ta ühe inimese seose abieluga. Selles tabelis on andmed selle kohta, millise abieluga seost kirjeldatakse (seos tabeliga ABIELU, näiteks abielulepingu numbri abil), kelle abielus olemist kirjeldatakse (seos tabelisse ISIK, näiteks isikukoodi abil) ja isiku abieluga liitumise kuupäeva ning abielust lahkumise kuupäeva kohta

Tähele tuleb siin mudelis panna seda, et seosel tabelite ABIELU ja ABIELUS OLEMINE vahel, ei ole seose mitme-poolses otsas "mullikest". See tähendab seda, et kui on registreeritud abielu, peab mõni isik sellega ka seotud olema. Antud juhul vähemalt kaks inimest. See pannakse paika tarkvaraga, mis antud mudeli peale ehitatakse.

mudeli piisavuse kontroll

Kas nüüd oleme jõudnud täiusliku mudelini, mis kirjeldab mistahes struktuuriga abielu? Selle kontrollimiseks tuleb koostada loend, kõigist võimalikest suhte kombinatsioonidest erinevate isikute ja isikute gruppide vahel:

- üks mees - üks naine
- üks mees - üks mees so. kaks meest - ei ühtegi naist
- üks naine - üks naine so. kaks naist - ei ühtegi meest
- üks mees - mitu naist
- üks naine - mitu meest
- mitu meest (rohkem kui 2) - ei ühtegi naist
- mitu naist (rohkem kui 2) - ei ühtegi meest
- mitu meest - mitu naist

Rohkem kombinatsioone ei ole võimalik koostada. Kui nüüd vaadata neid kombinatsioone, siis taanduvad need kõik mudelile "N meest - M naist" kus N>=0 ja M>=0 ja M+N>=2. See mudel omakorda on võimalik taandada mudelile "N inimest", kus N>=2. Kõrvutades seda reeglit andmemudeliga on näha, et tõepoolest on siin võimalik registreerida kahe 2 ja rohkema inimese seos, mida me võime nimetada ka abieluks. On võimalik registreerida isegi ühe inimese seos iseendaga, aga see tuleb tarkvaraliselt välistada.

Kas ongi kõik? Ei, mitte veel päris kõik. Antud skeem sobiks ju ime hästi ka vabaabielude registreerimiseks. Praegusel juhul ei suuda me neid aga eristada registreeritud abieludest. Selleks et see puudus likvideerida peame tabelisse ABIELU tegema veeru SEOSE TYYP, mida võib siis väärtustada R -registreeritud abielude korral ja "V" - vabaabielude korral.

lõpmatu-lõpmatu areng ja muutumine

Nüüd on tõe poolest kõik. (ehkki kui keegi tahaks antus mudeli registreerida "üle aisa löömisi" tuleks veerule SEOSE TYYP lisada veel ühe väärtuse võimalus "A" - armu suhe :-)

Kas nüüd on kõik? Ega vist... ikka leiab mõne asja, mis jäi õigel hetkel meelde tulemata, või mis on vahepeal elus eneses juurde tekkinud. Selle mudeli kohta võib aga kindlalt (kindlalt? kindlalt!) öelda üht. Need tabelid ja need seosed, mis on üles joonistatus jäävad igavesest ajast igavesti selliseks nagu nad on. Juurde võib tull veerge tabelitesse või väärtuste loenditesse (nii nagu lisasime SEOSE TYYP-le armusuhte tunnuse), juurde võib tulla ka uusi tabeleid, mis seostatakse olemasolevatega, kuid olemasolevate tabelite struktuur ja seosed nende vahel säilivad edasi.

Esitan veel kokkuvõtteks lõpliku andmemudeli koos kõigi veergudega:

"lõplik" andmemudel