Riigi Infosüsteemi Amet

RELATSIOONILISTE ANDMEMUDELITE KOOSTAMISE JUHEND

Ver. 1.0

Tallinn 2015

Sisukord

Dokumendi eesmärk

Käesaoleva juhendi eesmärk on Eesti Vabariigi infosüsteemide relatsiooniliste andmemudelite ja nende alusel loodud andmebaaside ühtlase kvaliteedi tagamine infosüsteemide erinevate komponentide kavandamise, arendamise ja rakendamise protsessis. Relatsioonilise andmemudeli kvaliteedi all mõistetakse käesolevas dokumendis andmemudeli omaduste kogumit, mis kindlustab loodud andmemudelite ning nende alusel loodud andmebaaside struktuuride aegpüsivuse.

Andmemudeli ja selle alusel loodud andmebaasi struktuuri aegpüsivuse all mõistetakse käesolevas dokumendis andmemudeli arengu kriteeriumit, kus juba projekteeritud ja rakendatud andmemudelid säilitavad oma seoste struktuuri pikema aja jooksul muutumatuna (ideaalis „igavesti“) ning muudatused mudelis on kirjeldatavad peamiselt kui uute komponentide (olemite ja olemi atribuutide) lisamine mudelisse, nende seostamine omavahel ning varasemalt mudelis eksisteerivate komponentidega.

Dokumendi skoop

Dokument kirjeldab reeglid, mida tuleb järgida riigi infosüsteemi relatsiooniliste andmemudelite ja mudelite alusel relatsiooniliste andmebaaside loomisel. Metoodika on loodud fookusega kasutamiseks füüsiliste andmemudelite loomisel. Samas ei ole mingeid takistusi selle kasutamisel loogiliste andmemudelite loomisel.

Käesolev dokument ei sisalda andmemudeli loomise aluseks olevat uuringut ja selle läbi viimise metoodikat.

Dokument on mõeldud realtsiooniliste andmemudelite loojatele.

Dokumendi ülesehitus

Käesolev dokument on jagatud kahest suuremaks alajaotiseks, millest mõlemal on oma iseseisev tähendus ja on vaadeldav ka täiesti iseseisva dokumendina.

Esimeses jaotises kirjeldatakse relatsiooniliste andmemudelite loomise põhimõtteid, reegleid, mustreid ja piiranguid.

Teises jaotises on esitatud andmete modelleerimise mõistete süsteem, mis loob üldise aluse dokumendi mõistmiseks, kuid on käsitletav ka iseseisva andmete modelleerimise sõnastikuna.

Lisaks sellele esitatakse dokumendis põhimõtted konkreetse ülesande lahendamiseks erinevat tüüpi andmestute vahel valiku tegemiseks. Jaotise eesmärk on näidata relatsiooniliste andmemudelite ja andmebaaside valiku põhjuslikke kriteeriume kõrvutades ja eristades neid teis tüüpi andmestuste valiku kriteeriumitest.

1. Andmeskogumi füüsilise vormingu valimine

Erineva kasutusloogikaga andmete puhul on otstarbekas kasutada andmete talletamise erinevaid füüsilisi vorminguid:

1. Relatsioonilised andmebaasid – andmekogumid, mille sisemine struktuur (seosed) on keerukas ja kus tuleb rangelt tagada ACID kriteeriumite täitmine. Selle, kas ACID kriteeriumid peavad olema täidetud või mitte, määrab rakenduste iseloom, mis kasutavad seda andmekogumit. Enamiku „tööstusrakenduste“ korral on ACID kriteeriumite täitmine vältimatu.
2. JSON ja XML andmestruktuurid – andmekogumid, mille maht on väike ja mida kasutatakse peamiselt andmete (kogu andmekogum ühe korraga) lugemiseks (Näit: süsteemide häälestusparameetrid, klassifikaatorite kehtivad versioonid, andmestruktuuride kirjeldused, rakenduste kirjeldused, avaandmed jms.). Paljudel juhtudel on need andmekogumid mingi rakenduse töö eksponeeritav tulemus (näit. avaandmed). Kasutatakse ka andmevahetuseks erinevate infosüsteemide vahel.

   Oluliseks tingimuseks JSON ja XML andmestruktuuride valikul on andmete hierarhilisena esitamise võimalikkus.
3. CSV-andmestruktuur (fail) – peamiselt andmevahetuseks kasutatav andmestruktuur, mida kasutatakse kui XML või JSON struktuuride loomine pole otstarbekas või ajalooliselt on CSV-andmestruktuur juba olemas ja selle ringi tegemiseks puudub otsene vajadus. Sobib hästi ka suuremahuliseks andmete ülekandeks, kuna on oluliselt väiksema ajakuluga töödeldav kui XML või JSON ja andmete kogumaht on nendest väiksem, kuna kirjelduslik osa piirneb peamiselt esimese kirjega (faili päisega) või puudub üldse. Sobib ka lihtsa struktuuriga avaandmete esitamiseks. Puuduseks on programmikoodi sisse kirjutatud faili struktuuri tundmine.

   Oluliseks tingimuseks CSV andmestruktuuride valikul on lihtne, lameda kirjena andmete esitamise võimalikkus.
4. NoSQL andmebaasid – suure andmemahuga andmekogumid, millel on horisontaalselt suur skaleeruvus (st sama tüüpi andmeid kirjeldatakse väga erineva omaduste loendi abil) ja mille muutmisega ei tegeleta erinevates kohtades (Näit. veebi kaudu vastu võetavad avaldused, laimale ringile levitatav informatsioon, logid jms.) Siia hulka kuuluvad ka andmepilves hoitavad andmestud. Oluliseks kasutamise piiranguks on loodava andmebaasi BASE kriteeriumite täitmise piisavus (st puudub vajadus rakendada vastavust ACID piirangutele).

2. Relatsioonilise andmebaasi andmemudeli koostamine

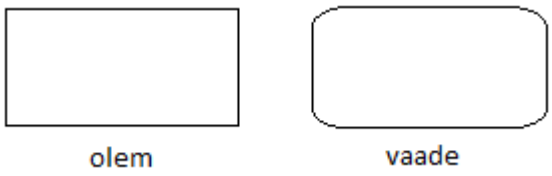
Käesolevas jaotises kirjeldatakse reeglid relatsiooniliste andmemudelite loomiseks. Relatsioonilise andmemudeli koostamise metoodika aluseks on ERD mudeli loomise reeglistik koos “varese jala” notatsiooniga (*crow’s foot notation*), millele lisanduvad käesolevas dokumendis kirjeldatud laiendused ja piirangud.

„Varese jala“ notatsiooni on käesolevas dokumendis kasutatud põhjusel, et see on erinevatest kasutusel olevatest notatsioonidest kõige ülevaatlikum ja metoodiliselt täiuslikum. Mudelite loomisel võib kasutada mistahes teist notatsiooni, mis on konkreetses organisatsioonis traditsiooniliselt kasutusel. Eelistatud on Enterprise Architect (EA) tarkvara poolt toetatud notatsioonid („crow foot“, IE ja UML).

2.1.1. ERD (Entity Relationship Diagram / olemi-suhte graaf) ja “varese jala” (“crow’s foot“) notatisoon

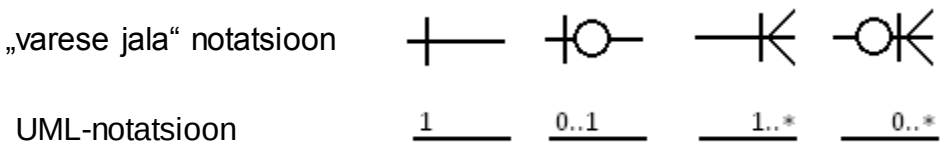
“Varese jala” notatsioon on üks levinumaid ERD notatsioone, mis on kasutusel ühe või ainsa notatsioonina enamikes ERD-l põhinevates andmemudelite projekteerimise rakendustes. “Varese jala” notatsioonist on olemas mitmeid erinevaid, väikeste erinevusega teisendusi. Lubatud on kasutada mistahes sellist notatsiooni teisendust tingimusel, et oleksid täidetud järgmised nõuded:

1. graafiliselt on eristatavad olemid (*entity/table*), mis tähistatakse ristkülikuna ja vaated (*view*), mis tähistatakse ümar-nurkadega ristülikutena:



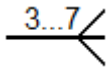
Joonis 1. Olemi ja vaate tähistus

2. Igal olemil on primaarvõti (primary key, PK), mis on surrogaatvõti so kunstlikult genereeritud väärtusega võti, mis ei oma seost reaalse eluga. Primaarvõtme väärtus on kas järjenumber, stringina esitatud järjenumber, millele on lisatud täiendavaid elemente või juhuslikult genereeritud string-väärtus (vt. täpsemalt jaotis 2.5)
3. olemite vaheliste suhete kirjeldamisel eristuvad graafiliselt järgmised seose tugevused: üks; üks või mitte ühtegi; üks või mitu; mitte ühtegi, üks või mitu:



Joonis 2. Seoste tugevused

4. erijuhtudel (see peab olema mudeli kirjelduses (selgitusena) eraldi välja toodud piisava põhjendatusega) on lubatud kasutada ka seose tüüpi “mitu” (st välistades nii “mitte ühtegi” kui ka “üks”). Sellisel juhul tuleb seose “mitu” otsa juures kirjeldada minimaalne ja maksimaalne võimalik seotud objektide arv (Näiteks: 2...n, 2...2, 3...7, 6…n, vms.):

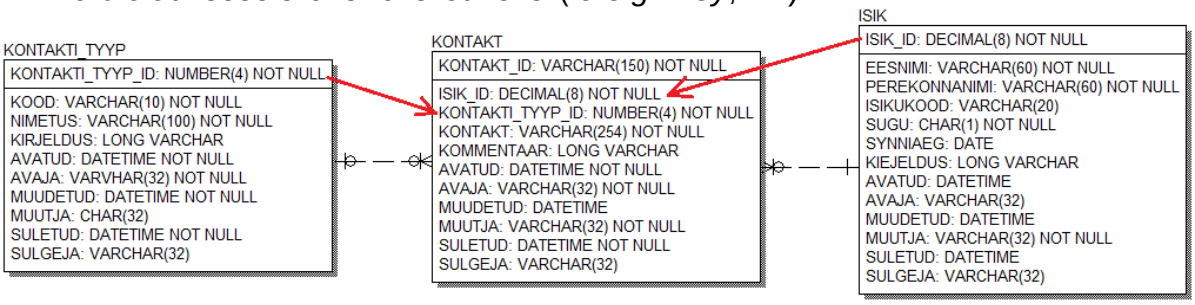


Joonis 3. Seose tugevus „mitu“

Selliseid seoseid tuleb kasutada väga kaalutletult, sest nende kasutamine seab mudelile jäiku piiranguid, mida tegelikus elus tavaliselt ei ole. Nende kasutamisel on tavaliselt rangelt tehniline põhjendus või on nad tingitud hetkel kehtivast seadusandlusest või muudest piirangutest, mis võivad tulevikus muutuda. Seepärast peaks püüdma neid vältida.

Olemite vaheliste suhete kirjeldamisel kehtivad järgmised piirangud:

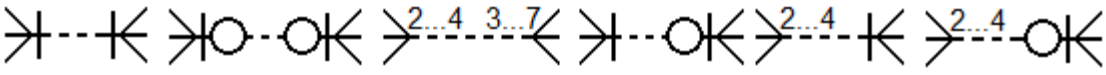
1. olemite vahele on lubatud kirjeldada ainult mitte-identifitseerivad (non-identify) suhted st. keelatud on identifitseerivad (identify) suhted. Mitte-identifitseeriv suhe tähendab seda, et seost valdava olemi primaarvõti (*primary key*. PK) ei pärandu kunagi seostatud olemi primaarvõtme koosseisu vaid alati iseseisvaks välisvõtmeks (*foreign key*, FK):



Joonis 4. Mitte-identifitseerivad suhted olemite vahel

Kui kasutatav ERD Case süsteem seda võimaldab tuleb mitteidentifitseerivad seosed tähistada katkendliku joonega nagu seda on näha joonisel 4.

2. füüsilistes andmemudelites ei olemite vahel lubatud seosed, kus seose mõlemas otsas on mistahes vormingus seose tugevus “mitu” (üks või mitu; mitte ühtegi, üks või mitu; mitu):

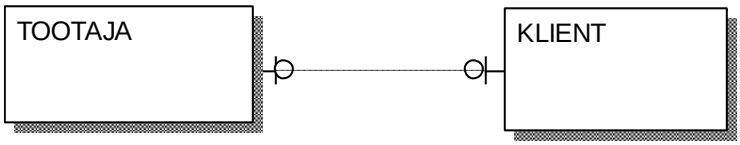


Joonis 5. Keelatud mitu-mitmesed seosed

Kontseptuaalsete andmemudelite kohta see piirang ei kehti.

3. üks-ühesed seosed olemite vahel on lubatud ainult tehnilistel põhjustel ja need kirjeldavad samasust st. seovad ära kaks olemit näitamaks (seostamaks) objekte, mis on kirjeldatud mõlemas olemis. Sellised seosed

tekivad tavaliselt olemi (mõiste) rollipõhisel tükkideks jagamisel. Kirjeldataval seosel on alati mõlemas otsas seose tugevuseks „üks või ei ühtegi“:



Joonis 6. Samasuse kirjeldamine

Seos tehakse kas seose ühe poole olemi primaarvõtme ülekandmisega seose teise olemi välisvõtmeks või mõlema seotud olemi primaarvõtmete vahetamisel välisvõtmetena. Mõlemal juhul kehtestatakse välisvõtmetele analoogiliselt primaarvõtmega unikaalsuse piirang. See tähendab, et sama välisvõtme sama väärtus tohib samas olemis eksisteerida ainult üks kord.

4. Ei ole lubatud sellised üks-ühesed seosed olemite vahel, mille kummaski otsas ei ole seose tugevust “ei ühtegi”. Sisuliselt tähendab selline seos, et tegemist on sama objekti kirjeldusega, mis on mingil põhjusel jagatud kaheks osaks. Sellise seose tekkimisel tuleb need kaks olemit, mille vahele selline seos tekkib, ühendada kokku üheks terviklikuks olemiks, mille atribuudid sisaldavad mõlema liidetud olemi sisulisi atribuute. Atribuudid (ka metaandmeid sisaldava atribuudid) liidetakse uude olemisse kokku unikaalsuse printsiibil – igat sama tähendusega (sama semantikaga) andmeelementi võib loodavas (liit-)olemis olla ainult üks.

2.1.2. ERD skeemide “tükeldamine” ja “värvimine”

Kui olemite hulk ERD skeemis kasvab selliseks, et skeemi kui tervikut on raske jälgid, tuleb ERD jaotada osadeks – funktsionaalseteks vaadeteks. ERD skeem tuleb jaotada funktsionaalseteks vaadeteks selliselt, et ühte vaatesse kuuluvad loogiliselt mingi konkreetse, tervikliku funktsionaalsusega kaetavad olemid (andmemudeli osa).

Funktsionaalse vaate suurus ei tohi üldjuhul ületada viitkümmet olemit. Samas ei ole kvantitatiivne andmemudeli tükeldamise piirang olulisem semantilisest tükeldamise reeglist – iga alamskeem peab olema semantiliselt terviklik.

Igal funktsionaalsel vaatel tuleb määrata ainult sellele vaatele ainuomane olemite taustavärv, millega värvitakse antud vaatesse kuuluvate olemite taustad. Kui olem kuulub mitmesse funktsionaalsesse vaatesse, siis tuleb otsustada, milline on see vaade, mida võib käsitleda selle olemi n.n. “põhilise asukohana”. See funktsionaalne vaade määrab antud olemi taustavärvi ja see värv on selle olemi taustavärviks läbi kõikide vaadete. Olemite taustavärvide kaudu kirjeldatakse erinevate funktsionaalsete vaadete vahelised seosed.

Olemi “põhilise asukoha” vaate määramise aluseks on olemiga seotud funktsionaalsus. Olemi “põhiliseks asukohaks” on sobiv valida see funktsionaalne

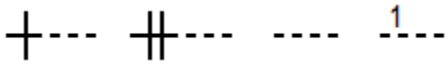
vaade, mis on seotud sellise funktsionaalsusega, mis tekitab olemisse esmaseid väärtusi (kirjeid). Kui neid väärtusi (kirjeid) tekitavad mitmete erinevate vaadetega seotud funktsionaalsused siis on sobiv valida olemi "põhiliseks asukohaks" vaade, mis on seotud sellise funktsionaalsusega, mis on ajalises järjestuses teistest samasse olemisse andmeid tekitavatest funktsionaalsustest ees pool.

Funktsionaalse jaotuse aluse (reeglistiku) loomine ja funktsionaalse jaotuse tegemine on andmemudeli looja pädevuses. Kui andmemudel on jaotatud funktsionaalseteks vaadeteks siis peab neid vaateid olema nii palju, et kogu andmemudel (kõik olemid ja seosed) on vaadetega kaetud.

2.1.3. Suhte tugevuste graafilise notatsiooni selgitused

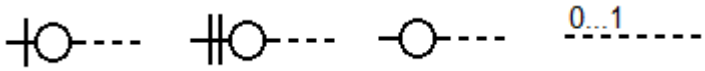
Kuna "varese jala" notatsioonist on käibel erinevaid versioone, siis esitame siinkohal võimalike kasutavate graafiliste sümbolite tähendused. Viimasena antakse iga tähistuse grupi juures võrdluseks UML notatsiooni vastava suhte tugevuse tähistus:

1. suhte tugevus "üks":



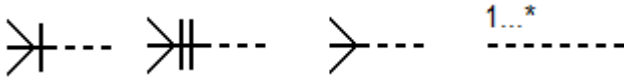
Joonis 7. Suhte tugevuse „üks" tähistused

2. suhte tugevus "üks või ei ühtegi":



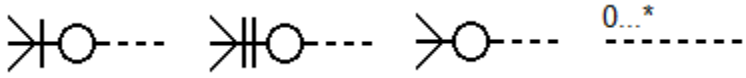
Joonis 8. Suhte tugevuse „üks või ei ühtegi" tähistused

3. suhe tugevus "üks või mitu":



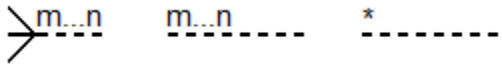
Joonis 9. Suhte tugevuse „üks või mitu" tähistused

4. suhte tugevuse "üks, mitu või ei ühtegi" ("üks, mitu või mitte ühtegi"; UML tähistus: 0..*)



Joonis 10. Suhte tugevuse „üks, mitu või ei ühtegi" tähistused

5. suhte tüüp "mitu" (st rohkem kui üks; UML tähistus puudub)



Joonis 11. Suhte tugevuse „mitu" tähistused

kus m>=2 ja n>=m. Selline seos on siiski harva esinev ja selle tekkimist peaks üritama vältida mõeldes üle, kas selline seos on ikka sisuline.

6. suhe tüüpi “mitu või ei ühtegi” (“mitu või mitte ühtegi”) ei kasutata.

Katkendlik seose joon seose kirjeldamisel tähistab mitte-identifitseerivat seost ja pidev joon identifitseerivat seost (viimase kasutamine ei ole lubatud).

2.1.4. Olemite piiritlemine

Kõik maailmas olemas olev on klassifitseeritav „subjektideks“, „objektideks“, „sündmusteks“ ja „seosteks“.

Olemid piiritletakse selliselt, et andmemudeli üks olem hõlmab ainult ühe reaalse maailma objekti, nähtuse, subjekti, seose või sündmuse andmeid. Kuna nähtus on “kehatu objekt” siis edaspidi käsitletakse objekti ja nähtust kui ühetaolist ja koondatakse need ühtse mõiste “objekt” alla. Olemitena väljendatakse skeemis ka muutuva iseloomuga väärtuste loendeid ehk klassifikaatoreid ja teatmikke, mille abil kirjeldatakse subjekt-, objekt-, sündmus- ja seos-tüüpi olemite omaduste väärtuste loendeid..

REEGEL: Olemil peab olema semantiline nimi, mis võimaldab mõista olemisse koondatud andmete tähendust. Olemi nimi väljendatakse ainsuse nimetavas käändes.

Igale olemile määratakse tüüp (“objekt”, “subjekt” , “sündmus”, “seos” või “teatmik”), mida kasutatakse olemi atribuutide mustri valimisel ja seoste määramisel teiste olemitega (vt. all pool).

Kõik klassifikaatorid koondatakse üldjuhul ühte tehnilisse olemisse, kus nad on üksteisest eristatavad klassifikaatori nime järgi. On lubatud ka klassifikaatorite hoidmine eraldi seisvates olemites.

2.1.5. Identifitseerimine – primaarvõtmed (primary key/master key)

REEGEL: Igal olemil peab olema primaarvõti. Primaarvõtmel on relatsioonilises mudelis kolmene tähendus, millest kaks on seotud andmemudeli enesega ja kolmas selle mudeli kasutamisega:

1. võimaldavad lihtsalt eristada kirjeid ja teostada ühte kirjet hõlmavaid operatsioone päring (SELECT) kustutamine (DELETE) ja uuendamine (UPDATE) võtme alusel,
2. nende kaudu tehakse seoseid olemite vahel (primaarvõti – välisvõti),
3. objekt-orienteeritud programmeerimistehnikates seostatakse andmeid ja objekte (klassi-ilminguid).

Olemites kasutatakse primaarvõtmetena surrogaatvõtmeid. Eelistatud on järjenumbritena kirjelduvad surrogaatvõtmed. Keelatud ei ole ka juhulikkuse alusel (hash) genereeritavad surrogaatvõtmed.

REEGEL: Primaarvõtme nimi (atribuudi nimi) on olemi nimi, millele on lisatus sufiks "_ID". (Näiteks: olem – ISIK, primaarvõti ISIK_ID).

Andmemudel peab olema projekteeritud nii, et iga kaks kirjet, mis luuakse sama olemi kirjelduse alusel oleksid väärtuste komplektina unikaalsed ja unikaalsus ei seisneks ainult primaarvõtme erinevuses vaid neid kirjeid oleks võimalik eristada üksteisest ka muude, sisuliste andmete alusel – kirje väärtuste komplekt sama olemi piires ei tohi kaotada oma unikaalsust pärast primaarvõtme väärtuse eemaldamist. Vastasel juhul on tegemist kas sama reaalse maailma subjekti, objekti, sündmuse või seose korduva kirjeldamisega andmebaasis või on olemi omaduste kirjeldus puudulik st. kirjeldatud on liiga väike valik omadusi selleks, et kirjeldatavaid asju üksteisest korrektselt eristada.

Rakendamine:

Järjenumber-tüüpi surrogaatvõtmete genereerimiseks kasutatakse meetodeid, kus uue primaarvõtme väärtuse genereerimine tehakse andmeid lisava andmebaasi transaktsiooni osana andmebaasi mootori poolt.

Sellistes andmebaasi mootorites, kus on olemas andmetüüp või andmetüübi lisaomadus "auto increment" või midagi sarnast (MySql: „auto increment", MS SQLerver: „Identity(start, samm)", Oracle: „auto_increment"), kasutatakse primaar-võtmete andmevälja kirjeldamisel seda.

Sellistes andmebaasimootorites, kus on olemas instrumendid "sequence" ja trigerid, (aga puudub „auto increment" tüüpi kirjeldus) kasutatakse primaarvõtmete väärtuste genereerimiseks neid.

Juhuslikkuse alusel võtme väärtuste genereerimine eeldab instrumentide "hash" ja trigerid olemasolu konkreetses andmebaasimootoris. Selle meetodi kasutamine primaarvõtme väärtuse genereerimisel ei ole lubatud kohtades, kus on vajalik kirjete

lisandumise järjestuse üle arvestuse pidamine – loendurina kasvav primaarvõti kõlbab ka kirjete lisandumisjärjekorra kirjeldamiseks.

Kui konkreetses andmebaasimootoris on kasutatavad kõik kirjeldatud meetodid siis otsustab tarkvara arendaja, lähtuvalt lahendatava ülesande spetsiifikast, millist meetodit kasutatakse. Eelistatud on esimesed kaks meetodit st. primaatvõtme väärtuse arvutamine “auto increment” või “sequence” meetodil.

Rakendused peavad olema loodud nii, et primaarvõtme väärtus ei muutu kunagi. See on ka põhjus, miks loomulikke andmeid ei tohi kasutada primaarvõtmetena – loomulikud andmed võivad elust enesest tingituna muutuda.

Andmebaasimootori väline primaarvõtmete väärtuste genereerimine ei ole lubatud.

2.1.6. Alternatiivvõtmed

Kui mistahes olemi kirje struktuurist eemaldada pimaarvõti peavad kõik selle olemi kirjed jääma üksteise suhtes (sama olemi piires) endiselt unikaalseks. See tähendab seda, et kirjetes kirjeldatud andmekomplektid peavad jääma eristatavateks.

REEGEL: Kirjete unikaalsuse tagamiseks tuleb igale olemile kirjeldada vähemalt üks alternatiivvõti st. unikaalne võti, mis hõlmab ühte või enamat andmeelementi ja mille väärtus üle kõigi kirjete on unikaalne.

Olemile võib olla kirjeldatud üks või enam alternatiivvõtit.

REEGEL: Kui kirjele on kirjeldatud üks alternatiivvõti, peab see hõlmama endas minimaalse komplekti selle olemi selliste atribuutide väärtusi, mille kombinatsioon selle olemi piires on alati unikaalne üle kõigi kirjete.

REEGEL: Primaarvõtme väljad ei tohi kunagi kuuluda alternatiivvõtme koosseisu.

2.1.7. Andmete seostamine – välisvõtmed (foreign key)

REEGEL: Välisvõtme andmetüüp on sama selle primaarvõtme andmetüübiga, millega välisvõti on seostatud. Välisvõti ei saa olla tüüpi “auto increment” (või sellega sarnanev). Seetõttu, kui primaarvõtme andmetüübiks on “auto increment” (või sellega sarnanev) määratakse välisvõtme andmetüübiks kas “INTEGER”, “LONG INTEGER”, NUMBER või NUMERIC vastavalt sellele, milline nendest on baasandmetüübiks “auto increment” (või sellega sarnanevale) andmetüübile.

REEGEL: Välisvõtme nimeks on üldjuhul sellega seotud vastava primaarvõtme nimi. Kui olem on teise olemiga seotud mitme suhtega siis on lubatud kasutada välisvõtme nime kahte erinevat kuju:

1. Tehniline kuju – igale välisvõtme nimele lisatakse pärast sufiksit “_ID” järjenumber (Näiteks: ISIK_ID1, ISIK_ID2, jne.)
2. Loogiline kuju – igale välisvõtmele lisatakse primaarvõtme nimele semantiline nimi, mis selgitab selle tähendust antud kohas ja antud kontekstis (Näiteks: ISIK_OSTJA_ID, ISIK_MYYJA_ID)

Olemite vahelise ühe suhte kirjeldamisel on eelistatud välisvõtme nime füüsiline kuju (ilma järjenumbrita). Mitme suhte kirjeldamisel kahe olemi vahel on eelistatud välisvõtme nime loogiline kuju.

REEGEL: Välisvõtme kirjeldamisel tuleb kirjeldada ka üks piirangutest: RESTRICT või CASCADE. Mudeli koostamine selliselt, et tekkiks vajadus piirangute NO ACTIONS või SET NULL järele, on keelatud.

Rakendamine

Kõik välisvõtmed peavad olema andmebaasis füüsiliselt kirjeldatud. st. ei tohi tekkida situatsiooni, kus andmemudelis kirjeldatud seos moodustub andmebaasis vaid SELECT-lausete JOIN-tingimuste kaudu.
Kui konkreetse andmebaasisüsteemi SQL-keeles puudub võimalus kirjeldada seose loomisel tabelite vahel piiranguid RESTRICT ja CASCADE peavad vastavad piirangud olema loodud kasutades trigereid ja andmebaasiprotseduure.

2.1.8. Mustrid andmemudelite koostamisel

Andmemudeli loomisel on kõige olulisem eraldada kohe õigesti olemid ja kirjeldada nende vahelised suhted sellisel tasemel, et neid kirjeldusi ei peaks hiljem muutma st. ei peaks olemite vahel ümber jaotama atribuute (andmeelemente) ja ei peaks muutma seoseid juba eksisteerivate olemite vahel. Tegelikult on need muudatused ühe asja kaks poolt ja üks tingib teist.

Käesolevas jaotises kirjeldatakse andmemudelite koostamise struktuuriliste mustrite kogumit, mille järgimine võimaldab luua aegpüsivaid andmemudeleid. Andmemudeli aegpüsivus on andmemudeli arengu kriteerium, mis tagab, et juba projekteeritud ja rakendatud andmemudelid säilitavad oma olemas olevate seoste struktuuri pikema aja jooksul ning muudatused mudelis on kirjeldatavad kui uute komponentide (olemid, atribuudid ja suhted) lisamine mudelisse, nende seostamine omavahel ning varasemas mudelis eksisteerivate komponentidega. Ideaalsel juhul ei kao ega muuda oma semantikat ükski mudelis olev olem ega muutu ka juba loodud olemite vahelised suhted.

Vaadeldavad mustrid kirjeldavad erinevat tüüpi olemeid (olemite klassifitseerimist), nende vaheliste suhete loomise reegleid ja olemite sisemist, atribuutidest koosnevat struktuuri.

Vaadeldavad mustrid ei taga olemite atribuutide “lõpliku” loendi (olemi struktuuri) loomist vaid ainult annab juhiseid olemis sisalduvate andmete ja metaandmete kirjeldamiseks.

Käesolevas jaotises kirjeldatud metoodika on valdavalt soovituslik, kui konkreetse mustri (reegli) juures ei ole öeldud teisiti. Lubatud on kasutada mistahes teist metoodikat, mis tagab samasuguse tulemuse st. tagatud on loodava andme mudeli aegpüsivus.

2.1.9. Olemite klassifitseerimine

Kogu maailmas eksisteeriv on võimalik jaotada subjektideks, objektideks, nähtusteks, sündmusteks ja seosteks. Sama moodi on võimalik jaotada andmemudelis olevad olemid subjekt-, objekt-, sündmus- ja seos-tüüpi olemiteks. Nagu juba varem öeldud on nähtus “kehatu objekt” ja seega käsitletakse objekti ja nähtust kui ühetaolisi ning koondatakse need ühtse mõiste “objekt” alla - neid mõlemaid käsitletakse andmemudelis objekt-tüüpi olemitena.

Subjekt-tüüpi olemiteks klassifitseeruvad kõik olemid, mis kirjeldavad reaalse elu subjekte ja kes/mis on loodava mudeli raames käsitletavad kui tegutsejad. Alati on subjekt-tüüpi olemina klassifitseeritavad kõik olemid, mis ühes või teises rollis kirjeldavad inimest (**„elusad“ subjektid:** ISIK, MYYJA, OSTJA, KLIENT, TOOTAJA jms.). Samuti on subjekt tüüpi olemiteks alati ka asutusi, ettevõtteid, organisatsioone, ühendusi jms. kirjeldavad olemid (**„elutud“ subjektid:** ISIK, FIRMA, ETTEVOTE, ASUTUS, FIE, YHENDUS jms.). Loomade andmeid sisaldavad olemid on subjekt-tüüpi sellisel juhul, kui nad sisaldavad taas-tuvastatavate loomade andmeid st. kui olemis registreeritud andmete järgi on võimalik konkreetne loom üles leida või vastupidi – konkreetse looma andmed on võimalik olemist üles leida. Kui tegemist on identifitseerimatute loomade (loom kui kontsept, märgistamata metsloom jms.) andmeid sisaldava olemiga siis on tegemist objekt-tüüpi olemiga.

Tegelikult kehtib identifitseerimise/identifitseerimatuse reegel subjektide eristamisel ka inimeste ja organisatsioonide puhul – kui need on taastuvastatavad (identifitseeritavad) on tegemist subjektidega, kui aga mitte siis objektidega.

Objekt-tüüpi olemiteks klassifitseeruvad kõigi nende reaalse maailma komponentide (nii nähtavad ja käega katsutavad, kui ka nähtamatud ja käega katsutamatud) andmeid sisaldavad olemid, mis ei ole võimalik hõlmata mõistega “subjekt-tüüpi olem”.

Sündmus- ja seos-tüüpi olemid on suhteliselt sarnased ja teinekord on neid eristad üsna raske. Mõlemad on nad ära tuntavad selle järgi, et nad kumbki ei oma iseseisvat tähendust vaid nad saavad tähenduse läbi subjekt- ja/või objekt-tüüpi olemite, mida nad seovad. Iga seos seob omavahel vähemalt kahte objekt- ja/või subjekt-tüüpi olemit. Sündmus võib olla seotud ka ühe subjekt- või objekt-tüüpi

olemiga. Samas ei ole ei seose ega sündmuse puhul välistatud seos enama kui kahe objekt- või subjekt-tüüpi olemiga.

Selleks, et eristada sündmus- ja seos-tüüpi olemeid tuleb mõista sündmuste ja seoste olemust. Üsna täpselt väljendab sündmuse ja seose erinevust lause: iga sündmus on ka seos aga ükski seos ei ole kunagi sündmus. Sündmust ja seost eristab see, et sündmusel on toimumise koht, alguse aeg ja lõpu aeg aga seosel on ainult seose alguse aeg ja seose lõpu aeg. Lisaks sellele on neil muidugi konkreetset tüüpi sündmusi või seoseid iseloomustavad omadused. Isegi kui andmemudelis mingil põhjusel ei kajastata sündmuse toimumise kohta ei muuda see sündmuse olemust – sündmus ei muutu seoseks. Näiteks on „pulmapidu“ sündmus samas kui „abielu“ on seos.

Olemid, mis kirjeldavad sündmusi on **sündmus-tüüpi olemid** ja olemid mis kirjeldavad seoseid on **seos-tüüpi olemid**.

Sündmus- ja seos-tüüpi olemite projekteerimisel tuleb arvesse võtta tõsiasja, et nende olemite alusel loodavad andmebaasitabelid on üldjuhul kiiresti kasvava andmemahuga. See võib tingida vajadust planeerida täpsemalt nende paiknemist füüsilistel andmekandjatel (näiteks nende paigutamist andmebaasi eraldi partitsioonidesse).

Lisaks “maailma kirjeldavatele olemi tüüpidele” vajame ka ühte tehnilist olemi tüüpi so. teatmik. **Teatmik-tüüpi olemite** abil kirjeldatakse eelkirjeldatud väärtuste loendeid, mida kasutatakse teiste olemitüüpidega kirjeldatud olemite omaduste väärtuste määramiseks. Teatmik-tüüpi olemiteks on näiteks liigi, tüübid, grupid jms.

Üsna tihti kerkib küsimus, kas tegemist on objekt-tüüpi olemiga või teatmik-tüüpi olemiga. Sellisel juhul tuleb vastata lihtsale küsimusele, mis otsustab kummaga on tegemist. Küsimus on: “kas selle olemi alusel tehtud andmebaasitabeli saab andmetega laadida enne infosüsteemi esmast käivitamist?” (Arvesse ei lähe andmed, mis laetakse eelmisest infosüsteemist infosüsteemi välja vahetamisel). Kui vastus on “jah” on tegemist teatmik-tüüpi olemiga, kui “ei” (st. andmed tekkivad töö käigus) siis objekt-tüüpi olemiga.

Teatmik tüüpi olemeid eristab teiste olemi tüüpide olemitest see, et nendes olevate andmete muutumise intensiivsus on väga madal – andmete muutusi nendes tehakse harva. Tavaliselt ei muutu juba lisatud andmed enam üldse – neid vaid kustutatakse (kuupäevaga). Teatmike muutumine tähendabki enamasti juba olemasolevate andmete sulgemist ja uute lisamist.

Puhtalt tehnilistest kaalutlustest lähtuvalt on mõistlik teatmikest eristada teatmike alamliik **klassifikaator**. Kui (tava)teatmikku koosseisu on õigus muuta kõigil selleks õigust omavatel süsteemi kasutajatel, siis klassifikaator on teatmik, mille koosseisu on õigus muuta ainult tarkvara loojatel. Seda selle pärast, et klassifikaatorite väärtuste loendid on tugevasti seotud programmi koodiga ja ühe või teise klassifikaatori väärtuse sidumine mingi subjekti, objekti, sündmuse, seose või ka teatmiku reaga tähendab tunduvalt enamat kui ainult väärtuse määramist. Suure

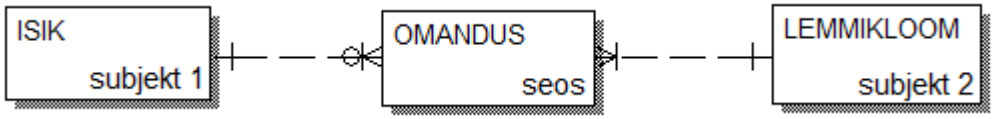
tõenäosusega mõjutab see programmi koodi täitmist (valikute tegemist, erinevate alamprogrammide käivitamist jms.).

Mudeli seisukohalt käitub klassifikaator sama moodi nagu mistahes teatmik ja seepärast kirjeldatavates mustrites neid teineteisest ei eristata vaid mõlemaid vaadeldakse kui teatmikke.

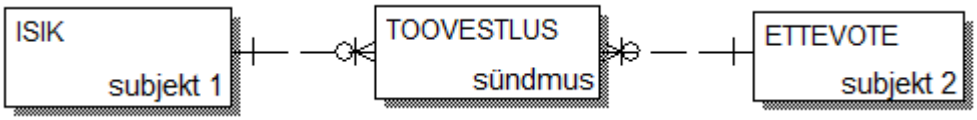
2.1.10. Alam-objektid, -sündmused, –seosed ja –teatmikud

Käesoleva jaotise piires on teadlikult loobutud kasutamast mõistet “olem” ja räägitakse subjekt-tüüpi olemi asemel “subjektist” või „subjektidest”, objekt-tüüpi olemi asemel “objektist” või “objektidest” jne. Seda on tehtud teksti loetavuse parandamiseks. Igas teksti lõigus, kus kasutatakse sõna “subjekt” võib mõelda ka “subjekt-tüüpi olem”, “objekt” asemel “objekt-tüüpi olem” jne. Mõiste “alam-sündmus” korrektne lahti kirjutus oleks “sündmus-tüüpi olemi sündmus-tüüpi alam-olem”. Sellisest keelelisest keerukusest loobumiseks on tehtud ka kirjeldatud mugandus.

REEGEL: „Elusal“ subjektil ei saa olla alam-subjekti. „Elus“ subjekt on teise subjektiga seotud alati kas läbi sündmuse, alam-sündmuse, seose või alam-seose. See tähendab seda, et kaks (omavahel eristatavat) subjekt-tüüpi olemit ei saa kunagi olla omavahel vahetult seotud.

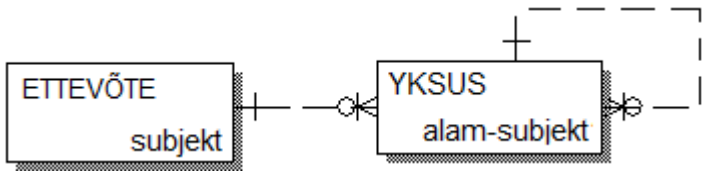


Joonis 12.“Elusate“ subjektide seostamine seose kaudu



Joonis 13. „Elusa“ ja „elutu“ subjektide seostamine sündmuse kaudu

REEGEL: „Elutul“ subjektil saab olla kuitahes palju alluvaid „elutuid“ subjekte. Alluvad „elutud“ subjektid on ülem-subjekti iseseisvalt identifitseerimatud komponendid. Üks („elutu“) alam-subjekt saab olla ainult ühe („elutu“) subjekti komponendiks.



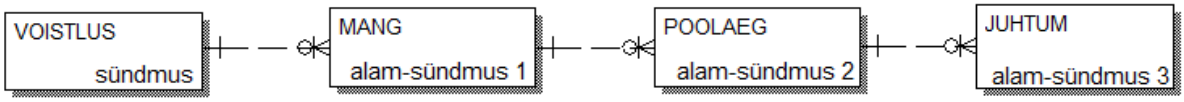
Joonis 12a. „Elutu“ subjekti seostamine „elutu“ (iseseisvalt identifitseerimatu) subjektiga

REEGEL: Objektiga saab olla seotud alam-objekt (Näiteks: MAJA ja KORTER) ja sellega viimasega omakorda alam-objekt (Näiteks: RUUM). Alam-objekt on objekti lahutamatu komponent ja kirjeldab ühe (alam-)osa sellest objektist, mille osa ta ise on. Objekti suhe tema komponentsesse alam-objekti moodustatakse vahetu üks-mitmese suhtega objekti poolt alam-objekti poole. Üks alam-objekt saab olla ainult ühe objekti komponendiks (alam-objektiks). Objekte ja erinevate objektide alam-objekte (mis ei ole omavahel seotud komponentsuse reegli abil st. on sõltumatud) seotakse omavahel seostega (seos-tüüpi olemitega) ja sündmustega (sündmustüüpi olemitega).

Joonis 14. Alam-objektid

Jooniselt on näha, et korter ei oma tähendust ilma majata, kus see asub ja ruum ei oma tähendust ilma korterita, mille koosseisu see ruum kuulub. Komponentsuse üheks vältimatuks omaduseks ongi alam-objekti identifitseerimatus ilma objektita, mille komponent see on. Korteri number ilma maja teadmata on üsna vähe ütlev.

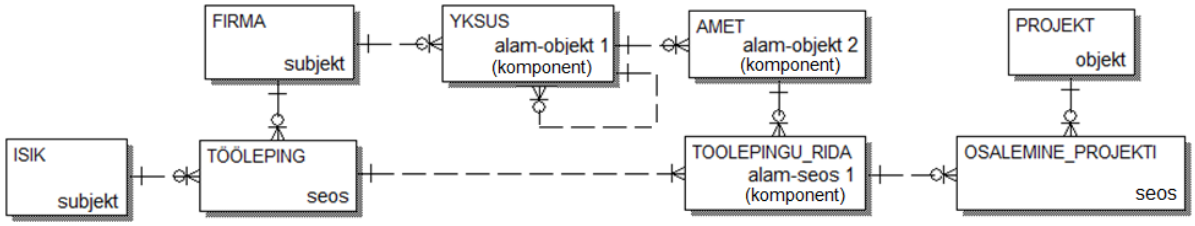
REEGEL: Sündmusel saab olla alam-sündmus (Näiteks: VÕISTLUS ja MÄNG) ja sellel omakorda alamsündmus (Näiteks: POOLAEG). Alam-sündmus on sündmuse lahutamatu komponent ja kirjeldab ühe (alam-)osa sellest sündmusest, mille osa ta ise on. Sündmuse suhe tema alam-sündmusega moodustatakse vahetu üks-mitmese suhtega sündmuse poolt alam-sündmuse poole. Üks alam-sündmus saab olla ainult ühe sündmuse alam-sündmuseks. Sündmusi ja erinevate sündmuste alam-sündmusi seotakse omavahel seostega (seos-tüüpi olemitega).



Joonis 15. Alam-sündmused

Alam-sündmuste omavahelises seoses ei ole midagi erinevat võrreldes alam-objektide vahelise seosega.

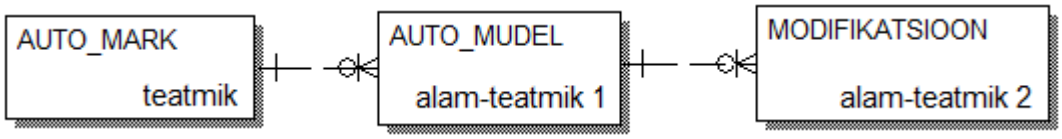
REEGEL: Seosel saab olla alam-seos (Näiteks: TÖÖLEPING ja TÖÖLEPINGU_RIDA) ja sellel omakorda alam-seos (Näiteks: OSALEMINE_PROJEKTIS). Alam-seos on seose lahutamatu komponent ja kirjeldab ühe (alam-)osa sellest seosest, mille osa ta ise on. Seose suhe tema alam-seosega moodustatakse vahetu üks-mitmese suhtega seose poolt alam-seose poole. Üks alam-seos võib olla ka mitme seose alam-seos. Sellisel juhul on ta neid seoseid omavahel siduvaks seoseks.



Joonis 16. Alam-seosed

Seoseid ja alam-seoseid on raske näidata ilma nende olemiteta mida nad seovad. Seda selle pärast, et oma semantika saab seos alati nendest olemitest, mida ta seob.

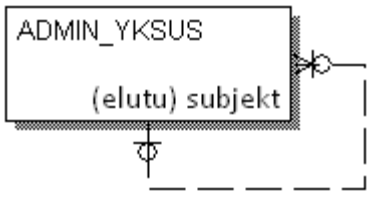
REEGEL: Teatmikul saab olla alam-teatmik (Näiteks: AUTO_MARK ja AUTO_MUDEL) ja sellel jällegi alam-teatmik (Näiteks: MODIFIKATSIOON). Alam-teatmik on teatmiku lahutamatu komponent ja kirjeldab teatmiku väärtuse jagunemist täpsemateks väärtusteks. Teatmiku suhe tema alam-teatmikuga moodustatakse vahetu üks-mitmese suhtega teatmiku poolt alam-teatmiku poole. Üks alam-teatmik saab olla ainult ühe teatmiku detailiseeringuks (alam-teatmikuks):



Joonis 17. Teatmikud ja alam-teatmikud

Hierarhia ahel nii objektide, sündmust, seoste kui ka teatmike puhul võib loogiliselt olla kui tahes pikk, kuid praktiliselt tuleb püüda vältida hierarhia sügavust üle kolme taseme – inimese tunnetus piirdub kolmemõõtmelise ruumiga ja sügavamatel tasemetel läheb enamiku jaoks asi keeruliseks. Lisaks sellele muutub keerukaks ka sellise puu haldamine ja see tekitab probleeme tarkvara arendajatele kasutatava kasutajaliidese loomisel. Muidugi ei ole seda võimalik alati vältida. Seega ei ole alluvuse piiramine kolme tasemega absoluutne nõue vaid ainult soovitus

REEGEL: Kui kõik olemid hierarhia erinevatel tasemetel on sarnase struktuuriga on soovitav need üldistada ühte ühtsesse olemi struktuuri ja realiseerida need ühe, iseendaga rekursiivses suhtes oleva olemina:



Joonis 18. Rekursiivne komponentsus

REEGEL: Rekursiivse seose puhul on alati seose mõlemas otsas „nullid“ (olenemata sellest, kas tegemist on objekti, subjekti, sündmuse, seose või teatmiku rekursiivselt kirjeldatud komponentsusega), sest alati on objekte, mis ei ole ühegi

teise objekti alam-objekt ja alati on objekte, millel alam-objektid puuduvad. Vastasel korral tekkiks kirjeldatavas hierarhias suletud ahel.

PIIRANG: Sellise lahenduse korral ei tohi kunagi kasutada seose piirangut CASCADE vaid alati tuleb kasutada piirangut RESTRICT. CASCADE kasutamine võib tekitada kirjete ahelkustutamise.

2.1.11. Erinevat tüüpi olemite seostamine

Käesolevas jaotises kirjeldatakse olemite omavahel seostamise reegleid st. olemite selliseid seoseid, mis ei ole jaotises 2.1.10 kirjeldatud komponentsusseosed. Erinevalt komponentsusseostest, kus kirjeldati sisalduvust, kirjeldatakse siin sõltumatute olemite omavahelise seostamise reegleid. Kui komponentsusseoses sai olem olla ainult ühe olemi komponendiks, siis siin kirjeldatavate seoste kaudu saab olem olla seotud mistahes arvu teise olemitega või ka iseendaga. Ka samade olemite vahel on lubatud mitmed erineva semantikaga seosed.

2.1.12. Subjekt-tüüpi olemite seostamine

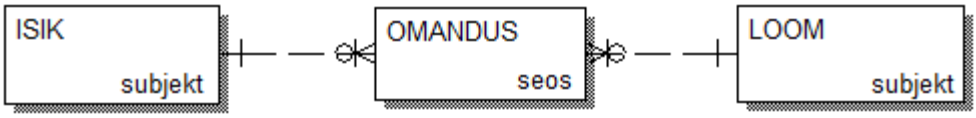
Käesolevas jaotises vaadatakse olemite vahelisi seoseid subjekt-tüüpi olemite seisukohalt.

2.1.12.1. „Elus“ subjekt-tüüpi olemi seostamine subjekt-tüüpi olemiga

REEGEL: „Elus“ subjekt-tüüpi olemi seostamine teise subjekt-tüüpi olemiga (nii „elus“ kui „elutu“ subjektiga) saab toimuda ainult kas läbi seos- või sündmus-tüüpi (ka alamseos- või alamsündmus-tüüpi) olemite kaudu – subjektid seostuvad läbi nende vahel olevate seoste või läbi osalemise samas sündmuses. Teiste sõnadega öeldes seovad subjekte sündmused ja seosed, milles need subjektid osalevad. Kahe „elus“ subjekt-tüüpi olemi omavaheline otse-seostamine on keelatud.

TÄPSUSTUS: Sama reegel kehtib ka sõltumatute „eluta“ subjekt-tüüpi olemite korral. Sõltumatud „eluta“ subjekt-tüüpi olemid on sellised subjekt-tüüpi olemid, millest üks ei ole teise komponent.

Näiteks seostab isikut ja tema lemmiklooma seos „omandus“:

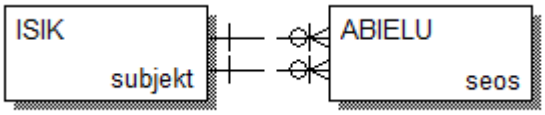


Joonis 19. Kahe „elus“ subjekt-tüüpi olemi seostamine

Iga seos-tüüpi olem kannab (vähemalt) subjektide vahelisele seose ajalist dimensiooni (alates, kuni), mille kirjeldamine on kohustuslik. Sama subjekt (loom) võib olla aja jooksul erinevate subjektide (isikute) omanduses. Üldjuhul tuleb programselt (näiteks trigeritega) tagada see, et erinevate seoste perioodid ei kattuks. Samas ei ole see siin kohustuslik, kuna me ei pruugi kirjeldada mitte omandust vaid omandust kui seost sama looma ja erinevate inimeste vahel - loom võib olla samal ajal seotud mitme subjektiga (perekonna liikmetega). Seega tuleb alati seoste korral määratleda, kas tegemist on kattuvate või mitte kattuvate seostega (üksteist välistavate seostega või mitte).

REEGEL: Olemite vahelisi seoseid moodustavate relatsioonide “üheses otsas” ei saa olla kunagi “nulli” st. seose liige (kumbki subjekt) ei saa kunagi puududa sest vastasel korral oleks seos määramata.

Seos-tüüpi olemi kaudu võib olla subjekt-tüüpi olem seotud ka iseendaga. Seos “abielu” on seos, kus meie kultuuris on lubatud seos ainult kahe inimese vahel:



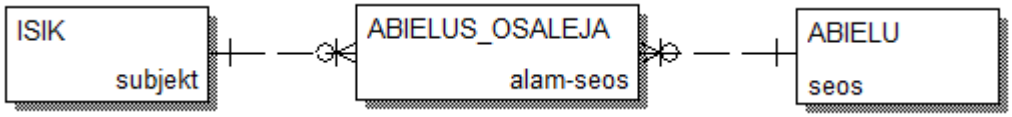
Joonis 20. Subjekt-tüüpi olemi seostamine iseendaga

Siin on samuti, nagu eelmiseski näites, oluline koht seose ajalisel dimensioonil (alates, kuni) mis peavad olema igal juhul määratud. Samas, praeguste seaduste järgi, tuleb rangelt kontrollida, et erinevatel sama isiku seostel (abieludel) ei oleks kattuvaid perioode. Ka siin tuleb hoolikalt mõelda kas on tegemist üksteis välistavate seostega või mitte (kas seosed võivad ajaliselt kattuda või mitte).

Selliseid seoseid, kus seos seob kahte (või mõnda suuremat fikseeritud arvu: 3, 4 jne.) subjekte, on elus palju aga tavaliselt on sellised fikseeritud seose osaliste arvuga subjektide seosed enamasti seadustega määratud seosed. “Looduses” on selliseid seoseid, kus suhte fikseeritud osaliste arv on läbiv üle kõikide selle suhte erinevate variantide, kaduv-vähe. Sellise seose joonistamine mudelisse seab

mudelile piirangu, mis kunagi hiljem võib hakata infosüsteemi arengut piirama. Näiteks praeguse seaduse ja mudeli seisukohalt (joonis 20) on võimalik registreerida ainult monogaamseid (paariabielusid). Kui nüüd aga võetakse vastu polügaamiat (grupiabielu) lubav seadus, siis muutub kirjeldatud mudel selle ülesande lahendamiseks ebasobivaks.

Selleks, et vältida seoses osalevate subjektide piiramist fikseeritud arvu osalejatega, tuleb subjekt seosega siduda läbi alam-seose. Joonisel 20 toodud näide teisendub sellisel juhul järgmiseks:

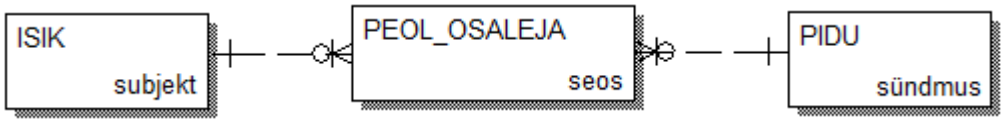


Joonis 21. Määramata arvu samas olemis olevate subjektide seostamine

Nüüd on läbi sama seose alam-seoste võimalik seostada mistahes hulk subjekte. Seejuures tekkib võimalus kirjeldada ajaliselt (ABIELU: alates, kuni) nii seose kehtivust tervikuna kui ka iga seoses osaleja individuaalset seoses osalemise aega (ABIELUS_OSALEJA: alates, kuni).

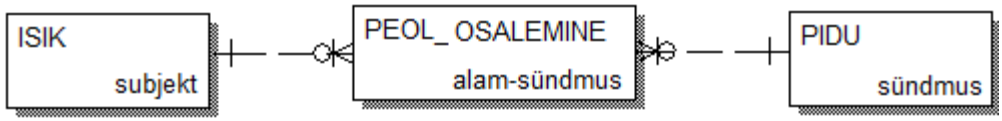
REEGEL: Subjektide seostamine läbi sündmuste toimub läbi seose sama sündmusega.

Kuna maailmas ei ole olemas sündmusi, kus kõigis sama tüüpi sündmustes osaleb alati ainult kaks (või mingi muu fikseeritud arv) subjekti, siis subjektide sidumine toimub alati läbi seose ja sündmuse. St subjektide seostamiseks läbi sündmuse tuleb seostatavad subjektid seostada sündmus-tüüpi olemiga seos-tüüpi olemi kaudu:



Joonis 22 Subjekt-tüüpi olemite seostamine läbi sündmuse ja seose

Tegelikkuses võib siduva olemi semantika ümbersõnastamise tulemusena öelda, et subjekte saab läbi sündmuste siduda kasutades sündmust ja alam-sündmust:



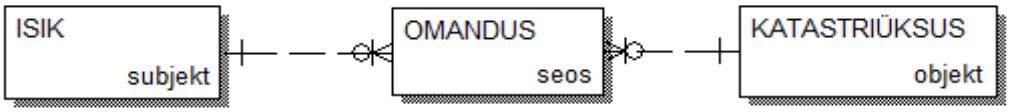
Joonis 23. Subjekt-tüüpi olemite seostamine läbi sündmuse ja alam-sündmuse

Sellel, kumba variant me kasutame, ei ole mingit tähtsust, sest selles kohas alam-seos ja alam-sündmus toimivad sama moodi – erinevus on semantika nüansis ja selle, kumba variant kasutatakse, peab otsustama mudeli looja.

2.1.12.2. Subjekt-tüüpi olemi seostamine objekt-tüüpi olemiga

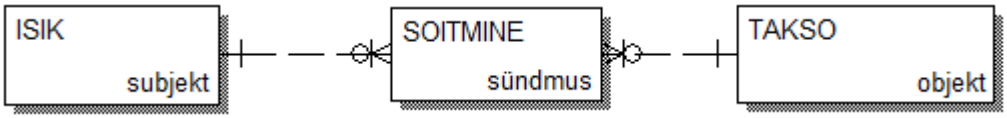
Subjekt-tüüpi olemi seostamine objekt-tüüpi olemiga on ligikaudu sama nagu on seda subjekt-tüüpi olemi seostamine subjekt-tüüpi olemiga (vt. 2.1.12.1). Vahe on selles, et objektid ja subjektid ei saa asuda samas olemis ja seega jääb ära variant, kus seotavad subjektid ja objektid on samas olemis (vt näited ja joonised 20, 21, 22 ja 23) – subjektid ja objektid paiknevad alati erinevates olemites.

REEGEL: Subjekt-tüüpi olemeid ja objekt-tüüpi olemeid seostatakse alati läbi seos- ja sündmus-tüüpi (alamseos- ja alamsündmus-tüüpi) olemite. Heaks näiteks on siin lemmiklooma omandusele sarnane näide kinnisvara omamisest:



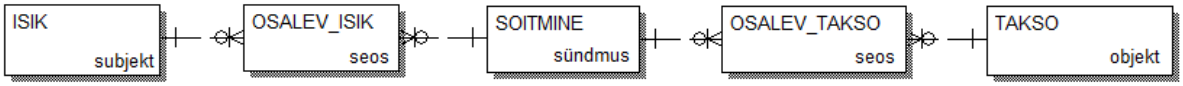
Joonis 24. Subjekt- ja objekt-tüüpi olemite seostamine

Analoogiline on ka subjekt-tüüpi olemi ja objekt-tüüpi olemi sidumine läbi sündmuse:



Joonis 25. Subjekt- ja objekt-tüüpi olemite seostamine sündmus-tüüpi olemi abil.

Kui semantiliselt on samas kohas võimalik subjekti ja objekti siduda kas sündmuse või seose kaudu tuleb hoolega mõelda kumba varianti kasutada, kuna subjekti ja objekti sidumisel sündmuse kaudu on olemas veel ka laiendatud verisoon, kus sama sündmusega saab olla seotud rohkem subjekte ja/või objekte:



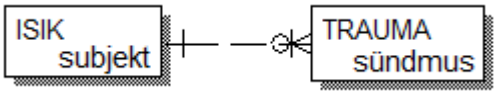
Joonis 26. Mitme subjekt- ja objekt- tüüpi olemi seostamine sündmustüüpi olemi abil

Põhimõtteliselt on analoogiline konstruktsioon võimalik ka subjekti ja objekti sidumisel läbi seose, kuid elus on selliseid semantilisi juhtumeid vähe ja sellise situatsiooni tekkimisel tuleb hoolega mõelda, kas midagi pole kahe silma vahele jäänud st. kas mingi tähelepanematuse tõttu on tekkinud tegelikkuses mitte eksisteeriv konstruktsioon. Keelatud see ei ole, kui konstruktsiooni kasutamine on argumenteeritult põhjendatud.

2.1.12.3. Subjekt-tüüpi olemi seosed sündmus- ja seos-tüüpi olemitega

Subjekt-tüüpi olemitel ei ole seos-tüüpi olemitega mingeid muid seoseid kui need, mis tulenevad subjekt-tüüpi olemite sidumisel subjekt-, objekt- ja sündmus-tüüpi olemitega (vt. 2.1.12.1 ja 2.1.12.2)

ERIJUHTUM: Subjekt-tüüpi olemite seostamisel sündmus-tüüpi olemitega on olemas üks erijuhtum – mudelis võib tekkida vajadus registreerida ühe subjektiga seotud sündmusi:



Joonis 27. Ühe subjekti seostamine sündmusega

Siin käitub sündmus-tüüpi olem nagu komponendina kasutatav objekt-tüüpi olem kirjeldades subjekti mingeid „juhtumisi“, mis on subjekti „elukäigu“ kirjeldamise komponendid. See tähendab seda, et mistahes sündmus võib olla ka subjekti (andmete) kirjeldamise komponent.

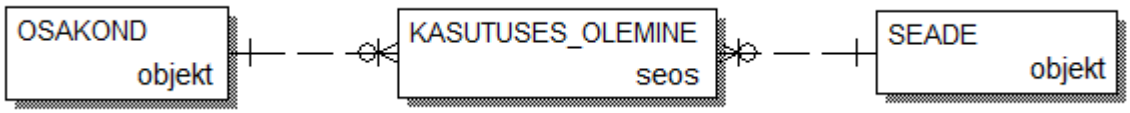
2.1.13. Objekt-tüüpi olemite seostamine

Käesolevas jaotises vaadatakse olemite vahelisi seoseid objekt-tüüpi olemite seisukohalt.

2.1.13.1. Objekt-tüüpi olemi seostamine objekt-tüüpi olemiga

REEGEL: Kahe sõltumatu objekt-tüüpi olemi seostamine teise objekt-tüüpi olemiga saab toimuda ainult kas läbi seos- või sündmus-tüüpi (ka alamseos- või alamsündmus-tüüpi) olemite kaudu – objektid seostuvad läbi nende vahel olevate seoste või läbi osalemise samas sündmuses. Teiste sõnadega öeldes seovad objekte sündmused ja seosed, milles need objektid osalevad. Kahe objekt-tüüpi olemi omavaheline otse-seostamine muudab ühe objekti teise alam-objektiks.

Näiteks seostab osakond ja inventari (seadet) selle seadme kasutamine osakonna poolt:

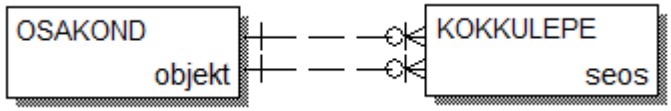


Joonis 28. Kahe sõltumatu objekt-tüüpi olemi seostamine

Seos-tüüpi olem kannab (vähemalt) objektide vahelisele seose ajalist dimensiooni (alates, kuni), mille kirjeldamine on kohustuslik. Sama seade võib olla aja jooksul erinevate objektide (osakondade) kasutuses. Üldjuhul tuleb programselt (näiteks trigeritega) tagada see, et erinevate seoste perioodid ei kattuks. Samas ei ole see alati kohustuslik, kuna ka selles näites võib esineda juhtumeid, kus mingi seade on kahe osakonna ühiskasutuses. Seega tuleb alati seoste korral määratleda, kas tegemist on kattuvate või mitte kattuvate seostega (üksteist välistavate seostega või mitte).

REEGEL: Objektide seoseid moodustavate relatsioonide “üheses otsas” ei saa olla kunagi “nulli” st. seose liige (kumbki objekt) ei saa kunagi puududa, sest vastasel korral oleks seos määramata.

Seos-tüüpi olemi kaudu võib olla objekt-tüüpi olem seotud ka iseendaga.

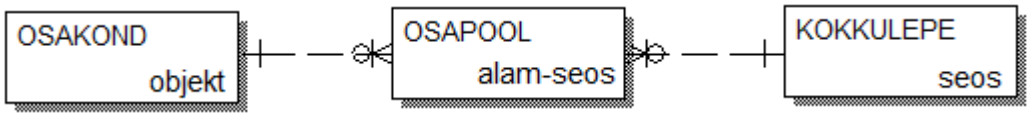


Joonis 29. Objekt-tüüpi olemi seostamine „iseendaga“

Siin on samuti, nagu eelmiseski näites, oluline koht ajalisel dimensioonil (alates, kuni) mis peavad olema igal juhul määratud. Antud näites pole ajalise kattuvuse kontrollil mingit tähendust, kuna erinevad kokkulepped võivad kehtida samaaegselt.

Selliseid seoseid, kus seos seob kahte (või mõnda suuremat fikseeritud arvu: 3, 4 jne.) objekte, on elus palju. Paraku eksisteerib palju ka olukordi, kus sama seoses eksisteerib palju osapooli (näiteks mitme osapoolega lepingud ja kokkulepped).

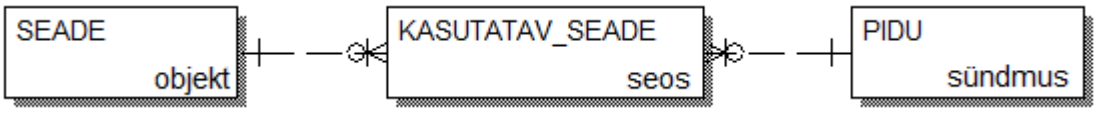
Selleks, et siduda samasse seosesse mitu objekti tuleb need siduda alam-seoste kaudu:



Joonis 30. Määramata arvu samas olemis olevate objektide seostamine

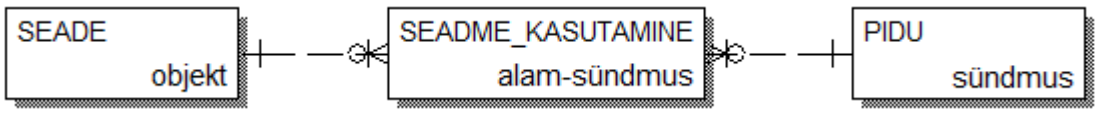
Nüüd on läbi sama seose alam-seoste võimalik seostada mistahes hulk objekte. Seejuures tekkib võimalus kirjeldada ajaliselt (KOKKULEPE: alates, kuni) nii seose kehtivust tervikuna kui ka iga seoses osaleva objekti seoses osalemise aega (OSAPOOL: alates, kuni).

REEGEL: Objektide seostamine läbi sündmuste toimub läbi seose sama sündmusega. Kuna maailmas ei ole olemas sündmusi, kus kõigis sama tüüpi sündmustes osaleb alati ainult kaks objekti, siis objektide sidumine toimub alati läbi seose ja sündmuse. St objektide seostamiseks läbi sündmuse tuleb seostatavad objektid seostada seos-tüüpi olemi kaudu sündmus-tüüpi olemiga:



Joonis 31 Objekt-tüüpi olemite seostamine läbi sündmuse ja seose

Tegelikkuses võib siduva olemi semantika ümbersõnastamise tulemusena öelda, et objekte saab läbi sündmuste siduda kasutades sündmust ja alam-sündmust:



Joonis 32. Objekt-tüüpi olemite seostamine läbi sündmuse ja alam-sündmuse

Sellel, kumba variant me kasutame, ei ole mingit tähtsust, sest selles kohas alam-seos ja alam-sündmus toimivad sama moodi – erinevus on semantika nüansis ja selle, kumba variant kasutatakse, peab otsustama mudeli looja.

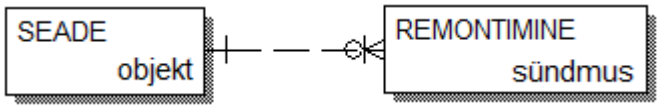
2.1.13.2. Objekt-tüüpi olemi seostamine subjekt-tüüpi olemiga

Objekt-tüüpi olemi seostamist subjekt-tüüpi olemiga on vaadeldud jaotises 2.1.12.1.

2.1.13.3. Objekt-tüüpi olemi seosed sündmus- ja seos-tüüpi olemitega

Objekt-tüüpi olemitel ei ole seos-tüüpi olemitega mingeid muid seoseid kui need, mis tulenevad subjekt-tüüpi olemite sidumisel subjekt-, objekt- ja sündmus-tüüpi olemitega (vt. 2.1.12.1 ja 2.1.12.2)

ERIJUHTUM: Objekt-tüüpi olemite seostamisel sündmus-tüüpi olemitega on olemas üks erijuhtum – mudelis võib tekkida vajadus registreerida ühe objektiga seotud sündmusi:



Joonis 33. Ühe objekti seostamine sündmusega

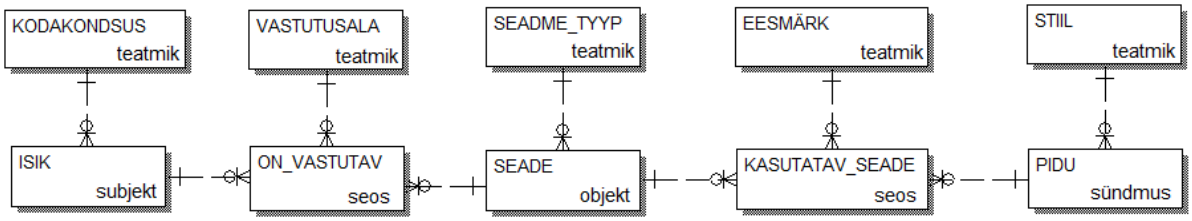
Siin käitub sündmus-tüüpi olem nagu komponendina kasutatav objekt-tüüpi olem kirjeldades objekti mingeid „juhtumisi“, mis on objekti „elutsükli“ kirjeldamise komponendid. See tähendab seda, et mistahes sündmus võib olla ka objekti (andmete) kirjeldamise komponent.

2.1.14. Teatmik-tüüpi olemite seostamine

Teatmik-tüüpi olemid seostuvad teiste olemitüüpide olemitega sõltumata nende tüübist alati ühte moodi. Seetõttu ei ole seoseid teatmik-tüüpi olemitega kirjeldatud iga olemi-tüübi juures eraldi vaid see kirjeldus on koondatud siia jaotisse kokku.

Nagu juba jaotises 2.1.9. mainitud, kirjeldatakse teatmik-tüüpi olemite abil eelkirjeldatud väärtuste loendeid, mida kasutatakse teiste olemitüüpidega kirjeldatud olemite omaduste väärtuste määramiseks.

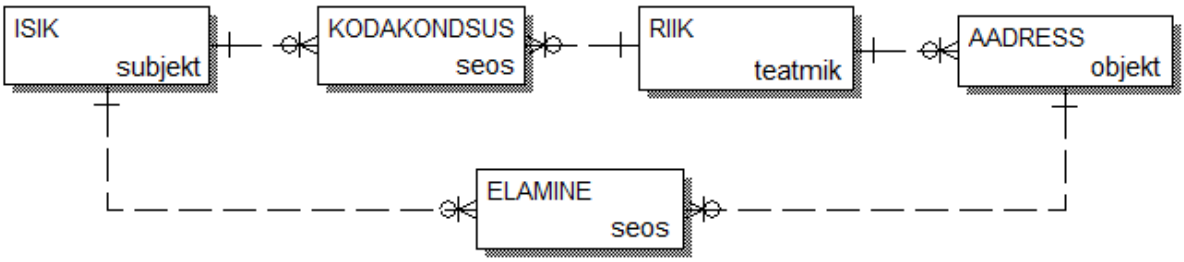
REEGEL: Seos teatmik-tüüpi olemi ja mistahes teist-tüüpi olemi vahel on alati suunaga teatmik-tüüpi olemi poolt selle teist-tüüpi olemi poole – suhte „ühene ots“ on teatmik-tüüpi olemi pool ja suhte „mitmene ots“ „selle teis-tüüpi olemi“ poole:



Joonis 34. Teatmik-tüüpi olemite seostamine

REEGEL: Teatmiktüüpi olemite seostamisel tuleb vältida olukorda, kus seose teatmik-tüüpi olemi poolsesse otsa tekkib „null“ st. võimalus, et seos puudub. Selle välistamiseks defineeritakse kõikidesse teatmikesse tühiväärtus „<määramata>“ (või semantiliselt sarnane väärtus) , millega saab seostada kõik need objektid, mille vastava teatmiku väärtust pole võimalik määrata (või mida ei teata). Et tagada andmete üldine käsitlus peab kõikides teatmikes sellise tühiväärtuse primaarvõtme (surrogaatvõtme) väärtus olema sama (Näit. „0“ või „-1“).

Sama teatmikku võib kasutada erinevate olemite omaduste väärtuste määramiseks:



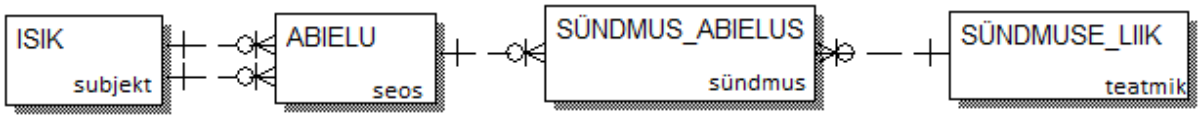
Joonis 35. Sama teatmiku kasutamine erinevate olemite omaduste määramiseks

Toodud näites määrab teatmik „RIIK“ isiku kodakondsused ja samas ka aadresside asukoha riigid.

2.1.15. Sündmus- ja seos-tüüpi olemite seostamine

Sündmus- ja seos-tüüpi olemite seostamise enamik mustreid on kirjeldatud jaotistes 2.1.12 ja 2.1.13.

ERIJUHTUM: Seos-tüüpi olemite seostamisel sündmus-tüüpi olemitega on olemas erijuhtum, mis on põhjustatud sellest, et mudelis võib tekkida vajadus registreerida seose kehtimise ajal toimunud sündmusi:



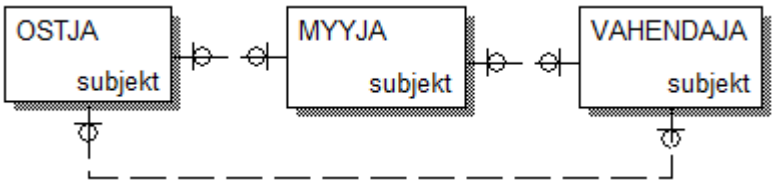
Joonis 36. Seose-tüüpi olemi seostamine sündmus-tüüpi olemiga

Siin käitub sündmus-tüüpi olem nagu komponendina kasutatav objekt-tüüpi olem kirjeldades seose kehtimise ajal toimunud „juhtumisi“, mis on seose „elutsükli“ kirjeldamise komponendid. See tähendab seda, et mistahes sündmus võib olla ka seose kirjeldamise komponent.

2.1.16. Olemite rolli-põhine jaotamine osadeks ja üks-ühene suhe

Vahel on otstarbekas jagada mingi olem, mis koondab struktuurilt ja loogikalt samasuguseid andmeid, kaheks või enamaks rollipõhiseks olemiks. Näiteks saab olemi ISIK jagada kolmeks rollipõhiseks olemiks OSTJA, MYYJA ja VAHENDAJA. Kuna sama subjekt võib teatud tingimustel olla üheaegselt nii ostja, müüja kui ka vahendaja (või olla ka ainult kahes rollis kolmest) siis tuleb leida metoodika erinevates olemites paiknevate sama subjekti andmete samasuste kirjeldamiseks. Üldisemalt öeldes, kui mingi olem on jagatud rollipõhiselt mitmeks erinevaks olemiks, siis eksisteerib võimalus, et andmed sama objekti, subjekti, sündmuse või seose kohta (sõltuvalt olemi tüübist) tekkivad rollipõhiselt mitmesse erinevasse olemisse.

REEGEL: Erinevates olemites olevate andmete samasuse kirjeldamiseks kasutatakse rist-viidatud (ID-de vahetamisega so. vastastikku välisvõtmete moodustamisega) üks-üheseid suhteid (null või üks – null või üks). Sellised seosed tuleb tekitada kõigi ühe olemi rollipõhise jaotamise tulemusena tekkinud olemite paaride vahel:



Joonis 37. Üks-ühesed seosed rollipõhise jaotuse korral

Selliselt moodustatud välisvõtmetele rakendub väärtustatud välisvõtme väärtuse unikaalsuse piirang – sama välisvõtme väärtus saab olla määratud ainult olemi ühes kirjes. See unikaalsuse piirang ei kehti tühjale väärtusele (määramata seoseid võib olla kuitahes palju).

Selle mustri kasutamine on kohustuslik.

2.1.17. Seostamisreegilte kokkuvõte olemitüüpide kaupa

Käesolevas jaotises on kõik eelnevalt kirjeldatud olemite seostamisreeglid võetud kokku olemitüüpide kaupa so. ühe olemitüübi seisukohalt vaadatuna

Käesolevas jaotises kasutatakse olemite vaheliste suhete kirjeldamisel mõisteid "suubuv suhe" ja "väljuv suhe". "Suubuva suhte" all mõistetakse siin olemite vahelise suhte "mitu"-poolset otsa ja "väljuva suhte" all olemite vahelise suhte "üks"-poolset otsa.

Erinevat tüüpi olemite vaheliste suhete moodustamisel kehtivad järgmised piirangud:

1. Subjekt-tüüpi olemisse saavad suubuda suhted vaid teatmik-tüüpi olemitest. Iga selline suhe kirjeldab subjekt-tüüpi olemi jaoks ühe omaduse (ühe) väärtuse.
2. Subjekt-tüüpi olemitest saavad suhted väljuda:
    a. objekt-tüüpi olemitesse – sellisel juhul kirjeldab seotud objekt-tüüpi olem antud subjekt-tüüpi olemi komponente/omadusi
    b. sündmus-tüüpi olemitesse – sellisel juhul kirjeldab suhe subjekti osalemise mingis sündmuses
    c. seos-tüüpi olemitesse – sellisel juhul määrab suhe subjekti osalemise mingis seoses.
3. Objekt-tüüpi olemisse saavad suhted suubuda:
    a. teatmik-tüüpi olemitest – iga selline suhe kirjeldab objekt-tüüpi olemi jaoks ühe omaduse väärtuse.
    b. subjekt-tüüpi olemitest – sellisel juhul kirjeldab antud objekt-tüüpi olem seotud subjekt-tüüpi olemi komponente/omadusi
    c. objekt-tüüpi olemist, mille komponendiks on vaadeldav objekt-tüüpi olem

4. Objekt-tüüpi olemitest saavad suhted väljuda:
    a. sündmus-tüüpi olemitesse – sellisel juhul kirjeldab suhe objekti osalemise mingis sündmuses
    b. seos-tüüpi olemitesse – sellisel juhul kirjeldab suhe subjekti osalemise mingis seoses.
    c. objekt-tüüpi olemitesse, mis kirjeldavad vaadeldava objekt-tüüpi olemi komponente
5. Sündmus-tüüpi olemitesse saavad suhted suubuda:
    a. teatmik-tüüpi olemitest – iga selline suhe kirjeldab sündmus-tüüpi olemi jaoks ühe omaduse väärtuse.
    b. subjekt-tüüpi olemitest – iga selline suhe määrab sündmuses osaleva subjekti
    c. objekt-tüüpi olemitest – igas selline suhe määrab sündmuses osaleva objekti
    d. sündmus-tüüpi olemist mille alamsündmused on kirjeldatud vaadeldavas sündmus-tüüpi olemis
6. Sündmus-tüüpi olemitest saavad suhted väljuda:
    a. Seos-tüüpi olemisse – iga selline suhe määrab sündmusega osalemise seoses.
    b. sündmus-tüüpi olemisse, mis kirjeldavad vaadeldava sündmus-tüüpi olemi alamsündmusi
7. Seos-tüüpi olemitesse saavad suhted suubuda:
    a. teatmik-tüüpi olemitest – iga selline suhe kirjeldab seos-tüüpi olemi jaoks ühe omaduse väärtuse.
    b. subjekt-tüüpi olemitest – iga selline suhe määrab seoses osaleva subjekti
    c. objekt-tüüpi olemitest – igas selline suhe määrab sündmuses osaleva objekti
    d. sündmus-tüüpi olemitest – igas selline suhe määrab seoses osaleva sündmus
    e. seos-tüüpi olemitest – iga selline suhe kirjeldab seose jaoks seose, mille alam-seos vaadeldav seos on

8. Sündmus-tüüpi olemitest saavad suhted väljuda:
    a. Sündmus-tüüpi olemisse – iga selline suhe määrab alam-sündmused, mis toimuvad antud sündmuse toimumise ajal
    b. seos-tüüpi olemisse – iga selline suhe kirjeldab vaadeldava sündmuse seose mingi subjekti, objekti, sündmuse või seosega
9. Seos-tüüpi olemitest saavad suhted väljuda:
    a. Sündmus-tüüpi olemisse – iga selline suhe määrab sündmused, mis toimuvad antud seose kehtimise ajal
    b. seos-tüüpi olemisse – iga selline suhe kirjeldab vaadeldava seose alam-seose
10. Seos-tüüpi ja ka sündmus-tüüpi olemitel, vaatamata nende välisele sarnasusele, on üks erinevus. Kui sündmus-tüüpi olem võib olla ainult ühe sündmus-tüüpi olemi alluvuses (olla selle ühe sündmuse alam-sündmuseks) siis seos-tüüpi olem võib enda kaudu ühendada kuitahes palju teiste olem-tüüpide olemeid ja ka seos-tüüpi olemeid.
11. Teatmik-tüüpi olemitesse saavad seosed suubuda ainult teistest teatmik-tüüpi olemitest. Seejuures tohib ühte teatmik-tüüpi olemisse suubuda seos ainult ühest teatmik-tüüpi olemist – sellest teatmik-tüüpi olemist, mille täpsustusi kirjeldab antud teatmik-tüüpi olem.
12. Teatmik-tüüpi olemitest saavad suhted väljuda kõikidesse teiste olemi tüüpide olemitesse. Iga seos määrab seotud olemi ühe atribuudi (omaduse) ühe väärtuse, mida saab valida teatmik-tüüpi olemis salvestatud loendist.
13. Kui samas olemis hoitakse andmeid mitmete erinevat tüüpi sündmuste ja seoste kohta (mille atribuutide struktuur on sama) siis tuleb sellise sündmus- või seos-tüüpi olemitega siduda vähemalt üks teatmik-tüüpi olem, mis kirjeldab kõik sündmuse või seose tüübid ja võimaldab määrata igale sündmusele või seosele, mis tüüpi sündmuse või seosega on tegemist.
14. Sama kehtib subjekt- ja objekt-tüüpi olemite korral.

Kui mudeli loomisel selgub, et mõni kirjeldatud reeglitest ei ole täidetav siis on mudelis selline viga, mis hiljem hakkab mõjutama mudeli aegpüsivust.

Ükski nendest reeglitest ei ole kohustuslik. Samas, kui mõni nendest reeglitest jäetakse arvesse võtmata, peab seda otsust eraldi põhjendama ja näitama ära miks seda tehti ja kuidas see mõjutab mudeli aegpüsivust.

2.1.18. Olemite sisemine struktuur

Olemite sisemise (atribuutide) struktuuri baas-mustri järgimine on kohustuslik. Kõikide olemite struktuuri baas-muster on sama:

<ID> (surrogaatvõti)
<omadus 1>
...
<omadus n>
<viide olemile 1> (välisvõti)
...
<viide olemile m> (välisvõti)
<algus-aeg> (alates)
<lõpp-aeg> (kuni)
KOMMENTAAR
AVATUD
AVAJA
MUUDETUD
MUUTJA
SULETUD
SULGEJA
SULGEMISE_KOMMENTAAR

Nagu juba käesolevas dokumendis ees pool mainitud, on kõikidel olemitel primaarvõtmeks (<ID>) surrogaatvõti. See on tingimatu nõue. Primaarvõtme nimi moodustatakse olemi nimest, millele lisatakse "_ID".

Igal olemil on kindlasti teda ennast iseloomustavad spetsiifilised atribuudid (omadused), mis on võimalik jaotada tinglikult kaheks - elementaaromadused (<omadus 1> … <omadus n>) ja viited (välisvõtmed) teistele olemitele (<viide olemile 1> … <viide olemile m>). Iga konkreetse olemi puhul saavad need olemi semantikast tingitud konkreetse tähenduse ja nimetuse.

Igas olemis peavad olema järgmised (meta)andmed:

KOMMENTAAR – vabatekstiline kommentaar. Seda võib kasutada mitmel eesmärgil aga enamuses peaks seda kasutama selleks, et anda süsteemi kasutajale (isikule, kellel on õigus muuta kirje sisu) võimalus märkida andmete juurde vabatekstilisi kommentaare, mis aitaks kasutajal meeles pidada erisusi, mis konkreetsete andmetega on seotud.

Järgnevad blokid kirje loomise-muutmise-sulgemise metaandmetega. Iga paar kirjeldab ühe sündmuse aja ja sündmuse tegija.

Atribuudid AVATUD ja AVAJA kirjeldavad kirje andmebaasi salvestamise aja (vähemalt sekundi täpsusega) ja viidet kirje loojale. Need atribuudid tuleb väärtustada kirje loomisel andmebaasi mootori poolt (trigeriga) ja nende väärtus ei muudeta enam kunagi. Kui andmebaasimootor seda võimaldab tuleb nende väärtuste muutmist takistada trigerit abil. Need väärtused peavad olema alati määratud.

Atribuudid MUUDETUD ja MUUTJA kirjeldavad kirje väärtuste muutmise viimast aega (vähemalt sekundi täpsusega) ja viidet muutuse teostajale. Need atribuudid väärtustatakse kirje loomisel andmebaasi mootori poolt samade väärtustega kui atribuudid AVATUD ja AVAJA, sest kirjel loomine on tema esimene muutus. Need väärtused muutuvad igakord, kui kirje mistahes sisulise atribuudi (<omadus 1>…<omadus n>, <viide olemile 1>…<viide olemile n>, KOMMENTAAR) väärtust muudetakse. Nende väljade väärtusi peab muutma andmebaasi mootor (näiteks trigeri abil). Metaandmete muutmine (näiteks kirje sulgemine väärtustades väljad SULETUD ja SULGEJA) atribuutide MUUDETUD ja MUUTJA väärtusi ei muudeta. Kui andmebaasimootor seda võimaldab tuleb nende väärtuste muutumist juhtida trigeri abil. Need väärtused peavad olema alati määratud.

Kui olem on projekteeritud nii, et tema andmed kunagi ei muutu vaid andmete muutumisel kehtiv kirje suletakse ja luuakse uus kirje uute andmetega, siis atribuudid MUUDETUD ja MUUTJA võivad puududa olemi struktuuris. Nende olemasolul tuleb nende muutmist takistada andmebaasi mootori vahenditega (trigeritega).

Atribuudid SULETUD ja SULGEJA võimaldavad kirje loogiliselt kustutada (sulgeda selle kirje kasutus). Need kaks välja on tingitud faktist, et andmebaasis olevate andmetega töötamise käigus ei tohi andmeid andmebaasist füüsiliselt kustutada.

Seda kas või juba selle pärast, et konkreetsete andmetega võib olla seotud olulisi andmeid, mis kaotavad oma tähenduse kui kustutada ära seotud andmed (näiteks teatmiku read). Kirje loomisel väärtustatakse atribuut SULETUD mingi väga suure (hilise) kuupäevaga milleni antud mudeli alusel loodud andmebaasi kasutus kunagi ei jõua (EOD - End of Days). Kirje sulgemisel kirjutatakse sinna kirje sulgemise kuupäev ja kellaaeg vähemalt sekundi täpsusega. Atribuut SULGEJA jääb kirje loomisel tühjaks ja väärtustatakse kirje loogilise kustutamise (sulgemise) hetkel andmetega, mis viitavad kustutajale, kes kirje sulges. Kuna andmete sulgeja võib olla ka mõni infosüsteemi automaatprotsess, siis peavad ka sellised protsessid oleme registreeritud kasutajana (või siis terve infosüsteem kui üks kasutaja).

Viide kirje loojale (AVAJA), muutjale (MUUTJA) ja sulgejale (SULGEJA) võib olla kirjeldatud kas andmebaasi kasutaja nimena või kui konkreetses andmemudelis on olemas olem, kus on registreeritud kasutajad, siis selles tabelis olev unikaalne kasutajanimi või ID. See viimane pädeb eriti just veebirakenduste korral, kus kõik kasutajad võivad olla andmebaasi logitud ühe kasutajana (rakenduse nimel). Sellisel juhul tuleb kasutada just viimast varianti.

Atribuut SULGEMISE_KOMMENTAAR on vajalik ebaregulaarsete “käsitsi” sulgemiste puhul (see ei välista muidugi selle kasutamist ka muudel juhtudel). Kui toimub kirje sulgemine, mis tehakse mingitel muudel põhjustel kui seda tehakse tavalise rutiini käigus siis kirjutatakse siia lühike kommentaar sulgemise põhjuste kohta. See võimaldab hiljem (aja möödudes, kui asi on ununenud) vältida küsimusi selle kohta, miks selline ebareeglipärane andmete sulgemine tehti.

Vajadusel võib samasuguse tähendusega atribuudi luua ka kirje avamise metaandmete juurde (AVAMISE_KOMMENTAAR), kus saab kirjeldada ebareeglipäraselt loodud kirjete loomise põhjuseid.

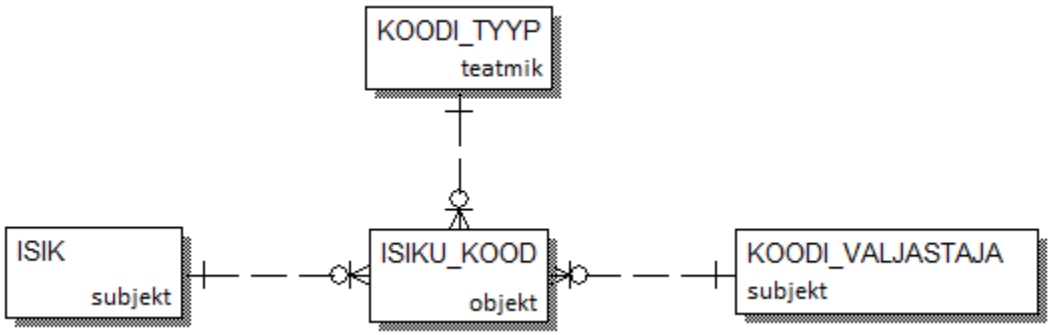
Rakendamine

Metaandmeid ei näidata tavakasutajale mitte kunagi.

2.1.18.1. Subjekt-tüüpi olemi struktuuri täpsustused

Subjekti omaduste hulka kuuluvad alati kohustuslike omadustena nimed (füüsilisel isikul: eesnimi, lisanimed, perekonnanimi, näidatavad nimed; juriidilisel isikul: nimi, lühinimi; identifitseeritaval loomal: nimi, mis võib olla ka määramata)

Lihtsamatel juhtudel on isikul üks kuni kaks koodi – füüsilistel isikutel isikukood, juriidilistel isikutel registrikood ja maksuameti kood ja loomadel registri kood. Keerulisematel juhtudel, kui on vajadus pidada muutuvat, allika põhist koodide loendit ja/või ajas muutuvate koodide loendit, viiakse koodid välja eraldi olemisse, kus see seotakse ühelt poolt subjektiga ja teiselt poolt koodi tüübi (teatmik) ja koodi allikaga (subjekt), mis võib asuda ka samas olemis kus subjekt, kelle/mille koode kirjeldatakse.



Joonis 38. Subjekti mitu, erineva allika poolt antud, koodi

Subjekt-tüüpi olemi välisvõtmed saavad viidata vaid teatmik-tüüpi olemitele.

Subjekt-tüüpi olemi atribuute <algus-aeg> ja <lõpp-aeg> interpreteeritakse üldjuhul kui “algus-kuupäeva” ja “lõpp-kuupäeva”. Elusate subjektide (inimesed, loomad) puhul on need “sündimise kuupäev” ja “surma kuupäev”. Eluta subjektide puhul on need “tekkimise/asutamise kuupäev” ja “lõppemise/likvideerimise kuupäev”. Enamikul juhtudel on nad kuupäevad ilma kellaajata. Kuid sõltuvalt infosüsteemi iseärasustest on võimalik ka kellaaja vajadus. Näiteks sünnitusmaja puhul on ilmselt lisaks lapse sündimise kuupäevale oluline ka sündimise kellaaeg.

Rakendamine

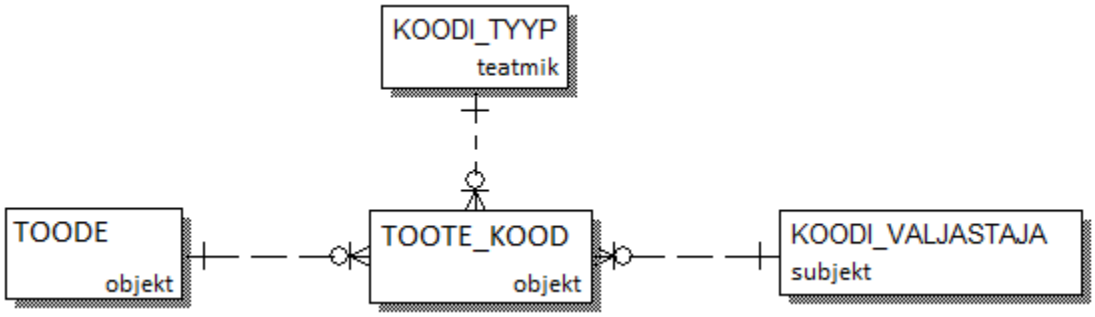
Kui konkreetses andmebaasi tootes puudub andmetüüp DATE, mis sisaldaks ainult kuupäeva (st. alati on koos kuupäevaga olemas ka kellaaeg) lisatakse algus-kuupäevale kellaaeg “00:00:00” ja lõpp-kuupäevale kellaaeg “23:59:59” või kui andmeformaat sisaldab ka millisekundeid siis kellaaeg “23:59:59,9999”. Võimalusel tuleb see tagada trigeriga.

2.1.18.2. Objekt-tüüpi olemi struktuuri täpsustused

Objekt-tüüpi olemi omaduste hulka kuulub alati kohustusliku omadustena üks nimi. Rohkemate (erinevate tähendustega) nimede kasutamise otsustab projekteerija.

Objektil võib olla kood aga see ei ole kohustuslik – koodi lisamine objektile ei ole eesmärk oma ette vaid kodeeritakse ainult need objektid, mille käsitlemiseks on kood vajalik st. on olemas seadusest või ajaloolisest kasutusest tulenev kood.

Keerulisematel juhtudel, kui on vajadus pidada muutuvat, allika põhist koodide loendit ja/või ajas muutuvate koodide loendit, viiakse koodid välja eraldi olemisse, kus see seotakse ühelt poolt objektiga ja teiselt poolt koodi tüübi (teatmik) ning koodi allikaga (subjekt).



Joonis 39. Objekti mitu, erineva allika poolt antud, koodi

Objekt-tüüpi olemi välisvõtmed saavad viidata teatmik-tüüpi olemitele ja kuni ühele Objekt-tüüpi olemile. Viimasele küll ainult sellistel juhtudel kui käsitletav objekt-tüüpi olem kirjeldab viidatava objekt-tüüpi olemi sisemise struktuuri komponente.

Objekt-tüüpi olemi atribuute <algus-aeg> ja <lõpp-aeg> interpreteeritakse üldjuhul kui “tekkimise/loomise/soetamise kuupäeva” ja “kadumise/hävitamise/likvideerimise kuupäeva”. See tähendab enamikul juhtudel on nad kuupäevad ilma kellaajata. Siiski võib tihti esineda juhtumeid, kus nii ühte kui teist on vaja kirjeldada kellaaja täpsusega. <algus-aega> ja <lõpp-aega> kirjeldatakse alati sama täpsusega st kas kuupäevana või kuupäev-kellaajana.

Rakendamine

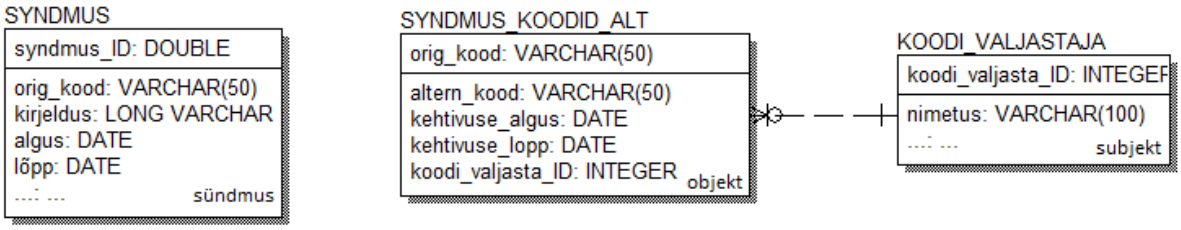
Kui <algus-aeg> ja <lõpp-aeg> on kirjeldatud kuupäevana ja konkreetses andmebaasi tootes puudub andmetüüp DATE, mis sisaldaks ainult kuupäeva (st. alati on koos kuupäevaga olemas ka kellaaeg) lisatakse algus-kuupäevale kellaaeg “00:00:00” ja lõpp-kuupäevale kellaaeg “23:59:59” või kui andmeformaat sisaldab ka millisekundeid siis kellaaeg “23:59:59,9999”. Võimalusel tuleb see tagada trigeriga.

2.1.18.3. Sündmus-tüüpi olemi struktuuri täpsustused

Sündmus-tüüpi olemite korral ei ole sündmust kirjeldavate omaduste kogum (<omadus 1>...<omadus 2>) kuidagi reglementeeritud.

Sündmusel võib olla kood (või number) aga see ei ole kohustuslik – koodi lisamine sündmusele ei ole eesmärk oma ette vaid kodeeritakse ainult need sündmused, mille käsitlemiseks on kood vajalik st. on olemas seadusest või ajaloolisest kasutusest tulenev kood.

Keerulisematel juhtudel, kui on vajadus pidada muutuvat, allika põhist koodide loendit siis projekteeritakse mudelisse vastavustabelid, mis seostatakse põhikoodiga so. koodiga, mis on sündmuse olemis.



Joonis 40. Sündmuse mitu, erineva allika poolt antud, koodi

Sündmus-tüüpi olemi välisvõtmed saavad viidata ühele sündmus-tüüpi olemile (sellele sündmus-tüüpi olemile, mille alamsündmused on kirjeldatud käesoleva olemiga) ja mistahes arvule teatmik-, subjekt-, objekt- ja seos-tüüpi olemitele.

Sündmus-tüüpi olemite puhul tuleb ühe viitena määrata suhe olemiga, mis kirjeldab toimumise koha. Toimumise koht ei pea olema geograafiline vaid võib olla ka “virtuaalne” (st. osakond, infosüsteem, kanal vms.). Mõningatel juhtudel võib toimumiskohaks olla ka “maailm” st. täpselt määratlemata. Sellisel juhul võib selle suhte määratluse ära jätta. Näiteks on PUHKUSEL_VIIBIMINE sündmus, millel on ajaline (kuupäevaline) algus ja lõpp aga puudub konkreetne üheselt määratud toimumise koht – see toimub “kusagil maailmas”.

Sündmus-tüüpi olemi atribuute <algus-aeg> ja <lõpp-aeg> interpreteeritakse üldjuhul kui “toimumise alguse aega” ja “toimumise lõppemise aega”. Seda, kas “aeg” sisaldab ka kellaaega või on määratud kuupäeva täpsusega, määrab kirjeldatava sündmuse iseloom. <algus-aega> ja <lõpp-aega> kirjeldatakse alati sama täpsusega st. kas kuupäevana või kuupäev-kellaajana.

Rakendamine

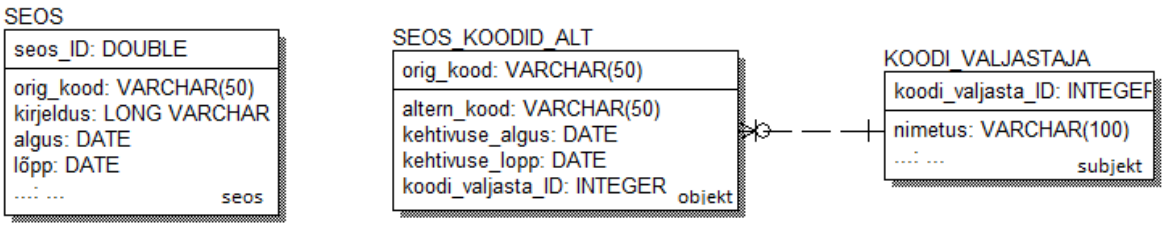
Kui <algus-aeg> ja <lõpp-aeg> on kirjeldatud kuupäevana ja konkreetses andmebaasi tootes puudub andmetüüp DATE, mis sisaldaks ainult kuupäeva (st. alati on koos kuupäevaga olemas ka kellaaeg) lisatakse algus-kuupäevale kellaaeg “00:00:00” ja lõpp-kuupäevale kellaaeg “23:59:59” või kui andmeformaat sisaldab ka millisekundeid siis kellaaeg “23:59:59,9999”. Võimalusel tuleb see tagada trigeriga.

2.1.18.4. Seos-tüüpi olemi struktuuri täpsustused

Seos-tüüpi olemite korral ei ole seost kirjeldavate omaduste kogum (<omadus 1>...<omadus 2>) kuidagi reglementeeritud.

Seosel võib olla kood (või number) aga see ei ole kohustuslik – koodi lisamine sündmusele ei ole eesmärk oma ette vaid kodeeritakse ainult need seosed, mille käsitlemiseks on kood vajalik st. on olemas seadusest või ajaloolisest kasutusest tulenev kood või number. Seda on siiski vaja aruharva.

Keerulisematel juhtudel, kui on vajadus pidada muutuvat, allika põhist koodide loendit siis projekteeritakse mudelisse koodide vastavustabelid, mis seostatakse põhikoodiga so. koodiga, mis on seose olemis.

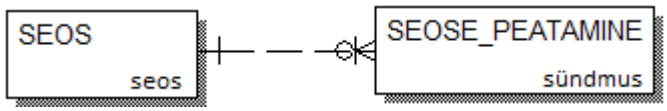


Joonis 41. Seose mitu, erineva allika poolt antud, koodi

Seos-tüüpi olemi välisvõtmed saavad viidata mistahes arvule teatmik-, subjekt-, objekt-, sündmus- ja ka seos-tüüpi olemitele. St., et seos-tüüpi olemiga saab siduda kõiki teist tüüpi olemeid ja ka teisi seos-tüüpi olemeid.

Seos-tüüpi olemi atribuute <algus-aeg> ja <lõpp-aeg> interpreteeritakse üldjuhul kui “seose algamise aega” ja “seose lõppemise aega”. Seda, kas “aeg” sisaldab ka kellaaega või on määratud kuupäeva täpsusega, määrab kirjeldatava seose iseloom. <algus-aega> ja <lõpp-aega> kirjeldatakse alati sama täpsusega st kas kuupäevana või kuupäev-kellaajana.

Kui on vaja kirjeldada seose peatumist mingiks ajaks, siis tehakse seda eraldi sündmus-tüüpi olemis, kus näidatakse ära seose peatamise algus ja lõpp.



Joonis 42. Seose peatamine

Rakendamine

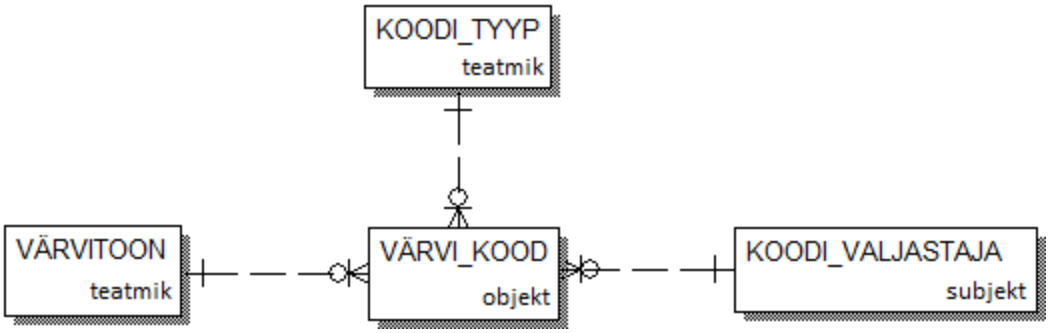
Kui <algus-aeg> ja <lõpp-aeg> on kirjeldatud kuupäevana ja konkreetses andmebaasi tootes puudub andmetüüp DATE, mis sisaldaks ainult kuupäeva (st. alati on koos kuupäevaga olemas ka kellaaeg) lisatakse algus-kuupäevale kellaaeg “00:00:00” ja lõpp-kuupäevale kellaaeg “23:59:59” või kui andmeformaat sisaldab ka millisekundeid siis kellaaeg “23:59:59,9999”. Võimalusel tuleb see tagada trigeriga.

2.1.18.5. Teatmik-tüüpi olemi struktuuri täpsustused

Teatmik-tüüpi olemi omaduste hulka kuulub alati kohustusliku omadustena üks nimi. Üldjuhul ongi nimesid ainult üks.

Teatmik-tüüpi olemis võib olla kood aga see ei ole kohustuslik – koodi lisamine teatmikule ei ole eesmärk oma ette vaid kodeeritakse ainult need teatmikud, mille käsitlemiseks on kood vajalik st. on olemas seadusest või ajaloolisest kasutusest tulenev kood (näiteks teatmiktüüpi olem, mis on loodud riikliku klassifikaatori rakendamiseks).

Keerulisematel juhtudel, kui on vajadus pidada muutuvat, allika põhist koodide loendit ja/või ajas muutuvate koodide loendit, viiakse koodid välja eraldi olemisse, kus see seotakse ühelt poolt teatmikuga ja teiselt poolt koodi tüübi (teatmik) ning koodi allikaga (subjekt).



Joonis 43. Teatmiku mitu, erineva allika poolt antud, koodi

Teatmik-tüüpi olemi välisvõtmed saavad viidata ainult teatmik-tüüpi olemitele. Sellisel juhul on vaadeldav teatmik kas viidatava teatmiku alam-teatmik (täpsustus) või kirjeldab viidatav teatmik vaadeldava teatmiku mõne omaduse väärtuse. Viimasel juhul on see teatmik tavaliselt klassifikaator.

Kuna teatmik-tüüpi olemid on tehnilise olemusega siis enamikul juhtudel teatmik-tüüpi olemitel atribuute <algus-aeg> ja <lõpp-aeg> ei ole – teatmiku väärtus hakkab kehtima kohe pärast andmebaasi kirjutamist (alates ajast AVATUD) ja lõpetab kehtivuse tema loogilisel kustutamisel (kehtib kuni ajani SULETUD). Seejuures tuleb tähele panna, et teatmiku ridu saab sulgeda ka ette haaravalt tulevikus kirjutades atribuuti SULETUD tuleviku väärtuse.

Kui siiski soovitakse lahutada kirje tehniline andmebaasi kirjutamise aeg ja loogilise kustutamise aeg ning kehtima hakkamise aeg ja kehtivuse lõppemise aeg tuleb selleks kasutada atribuute <alguse-aeg> ja <lopp-aeg>.

Olemi atribuute <algus-aeg> ja <lõpp-aeg> interpreteeritakse üldjuhul kui “kehtima-hakkamise aeg” ja “kehtivuse aegumise/lõppemise aeg”. Enamikul juhtudel on nad kuupäevad ilma kellaajata. Siiski võib esineda juhtumeid, kus nii ühte kui teist on vaja kirjeldada kellaaja täpsusega. <algus-aega> ja <lõpp-aega> kirjeldatakse alati sama täpsusega st kas kuupäevana või kuupäev-kellaajana.

Rakendamine

Kui <algus-aeg> ja <lõpp-aeg> on kirjeldatud kuupäevana ja konkreetses andmebaasi tootes puudub andmetüüp DATE, mis sisaldaks ainult kuupäeva (st. alati on koos kuupäevaga olemas ka kellaaeg) lisatakse algus-kuupäevale kellaaeg “00:00:00” ja lõpp-kuupäevale kellaaeg “23:59:59” või kui andmeformaat sisaldab

ka millisekundeid siis kellaaeg “23:59:59,9999”. Võimalusel tuleb see tagada trigeriga.

2.1.18.6. Aja käsitlust selgitavad märkused

Käesolev jaotis puudutab peamiselt piiranguid, mis esitatakse andmemudelis kajastatud aegade (kuupäevade ning kuupäevade ja kellaaegade) kasutamisel andmemudeli rakendamisel. Samas tuleb kõiki neid piiranguid arvesse võtta juba andmemudeli projekteerimisel:

1. Kirje loomise aeg (AVATUD) on kirje muutmise ajaga (MUUDETUD) kas võrdne (vahetult pärast kirje loomist) või sellest varasem (alates pärast esimest muutmist).
2. Kirje loomise aeg (AVATUD) on alati varasem kui kirje sulgemise aeg (SULETUD).
3. Kehtivuse algusaja (<algus-aeg>) tühiväärtus ei ole lubatud. Tühi väärtus asendatakse BOD (begin of days) konstandiga (N: 01.01.1800 00:00:00,0). See muudab päringutingimused lühemaks ja võimaldab vältida vigu.
4. Kehtivuse lõppaja (<lõpp-aeg>) tühiväärtus ei ole lubatud. Tühi väärtus asendatakse EOD (end of days) konstandiga (N: 31.12.3000 23:59:59,9999). See muudab päringutingimused lühemaks ja võimaldab vältida vigu.
5. Kirje kehtivuse ajavahemik <algus-aeg> kuni <lõpp-aeg> ei ole mitte kuidagi seotud vastava kirje tekkimise ja sulgemise (loogilise kustutamise) ajavahemikuga AVATUD kuni SULETUD.
6. Alam-olemi kirje <algus-aeg> ja <lõpp-aeg> peavad alati jääma ülemus-olemis oleva seotud kirjes kirjeldatud <algus-aeg> kuni <lõpp-aeg> ajavahemikku.
7. Kui andmebaasist päritakse andmeid mingil konkreetsel ajal (näiteks ajahetkel <NOW>) on kehtivad kõik kirjed, mis vastavad tingimusele:

   <algus-aeg> <= <NOW> AND <lopp-aeg> >= <NOW> AND SULETUD > <NOW>

8. Kui soovitakse näha andmeid, mis olid määratud ajal andmebaasis olemas siis päritakse välja kirjed, mis vastavad tingimusele:

   AVATUD <= <NOW> AND SULETUD >= <NOW>

9. Kui soovitakse näha andmeid, mis määratud ajal olid andmebaasis olemas ja samal ajal ka kehtivad siis päritakse välja kirjed, mis vastavad tingimusele:

   AVATUD <= <NOW> AND SULETUD >= <NOW> AND
   <algus-aeg> <= <NOW> AND <lopp-aeg> >= <NOW>

10. Kui andmebaasist päritakse andmeid mingil määratud ajaperioodil (näiteks ajaperioodil <ALGUS> kuni <LOPP>) on kehtivad kõik kirjed, mis vastavad tingimusele:

    <algus-aeg> <= <LOPP> AND <lopp-aeg> >= <ALGUS> AND
    SULETUD >= <ALGUS>

11. Kui soovitakse näha andmeid, mis olid määratud perioodil andmebaasis olemas siis päritakse välja kirjed, mis vastavad tingimusele:

    AVATUD <= <LOPP> AND SULETUD >= <ALGUS>

12. Kui soovitakse näha andmeid, mis määratud ajavahemikul olid andmebaasis olemas ja samal ajal ka kehtivad siis päritakse välja kirjed, mis vastavad tingimusele:

    AVATUD <= <LOPP> AND SULETUD >= <ALGUS> AND
    <algus-aeg> <= <LOPP> AND <lopp-aeg> >= <ALGUS>

2.1.19. Relatsioonilise andmemudeli dokumenteerimine

Andmemudel peab olema esitatud graafilisel (ERD), formaliseeritud ja verbaalsel kujul. Graafiliselt esitatakse andmemudel andmeloogilisel kujul st. iga olemi kohta peavad olema näha kõik tema atribuudid koos andmetüüpide kirjeldusega ja tühiväärtuse võimalikkuse (NULL / NOT NULL) kirjeldusega. Eraldi peavad olema tähistatud primaar- ja välisvõtmed.

Andmemudeli graafiline esitus peab olema teostatud mõne ERD case-süsteemiga sellislet, et kogu andmemudeli formaliseeritud ja verbaalne kirjeldus on adresseeritav (ligipääsetav) läbi ERD mudeli, elektroonilisel viisil.

Paberile trükituna või leheküljestatud elektroonilise dokumnedina esitatuna peab andmemudel koosnema vähemalt järgmistest komponentidest:

1. Üldosa:
    a. Preambula, mis sisaldab: andmemudeli tähendus – millise eesmärgi täitmiseks on andmemudel loodud, kasutajad, kaetud funktsionaalsuse lühikirjeldus, andmemudeli kirjeldamisel kasutatavad standardid ja mustrid,
    b. ERD-skeem, mis on esitatud funktsionaalsete vaadetena (vt. 2.1.2.) andmeloogilisel tasemel.
    c. olemite loetelu koos olemite semantika kirjeldusega, mis on esitatud kahe veerulise tabelina: olemi nimi; olemi semantika kokkuvõtlik kirjeldus, mis toetab ERD-skeemide mõistmist,
    d. seosed teiste andmemudelitega: kust andmemudelist saadakse andmeid ja kuhu neid antakse; milliseid andmeid, millise sageduse ja millise sünkronisatsioonimustri alusel vahetatakse,
2. Iga olemi kohta:
    a. olemi semantika põhjalik verbaalne kirjeldus (olemi tähendus, andmete tekke põhjus ja koht, andmete muutmise põhjused ja kohad, andmete kustutamise põhjused ja kohad, erijuhtumid, vastutajad)
    b. olemi atribuutide kirjeldused tabelina:
        i. nimi,
        ii. andmetüüp koos andmete pikkustega,
        iii. tühiväärtuse kirjeldus (NULL / NOT NULL),

iv. vaikeväärtus või reegel vaikeväärtuse arvutamiseks/pärimiseks kui selline on olemas,
v. minimaalne väärtus kui selline on olemas,
vi. maksimaalne väärtus kui selline on olemas,
vii. esitusformaat kui selline on määratud,
viii. valideerimise reeglid või valideerimisprotseduuri kirjeldus (ükski atribuut ei saa olla ima valideerimise reegliteta),
ix. võimalike väärtuse loend või viide klassifikaatorile kui sellised on määratud,
x. atribuudi verbaalne semantika kirjeldus: mida tähendab, kus ja millal tekkib tekkib (kui on erinev olemi üldisest,

c. olemi teiste olemitega suhete (relation) kirjeldused koos suhtete semantilise kirjeldustega (see võib olla liidetud ka välismvõtmena kirjeldatud atribuutide senmantika kirjeldustega),
d. olemi indeksite kirjeldused; iga indeksi kohta indeksi tüüp, unikaalsustunnus (unique or not), nimi ja sisalduvad tabeli veerud õiges järjestuses,
e. olemi trigerite ja seotud protseduuride kirjeldused või viited nendele kirjeldustele,

3. Andmemudeli kui tervikuga seotud trigerite ja protseduuride kirjeldused
4. Andmebaasi loomise skript kui on teada konkreetne andmebaasisüsteem (toode), milles antud mudeli järgi andmebaas luuakse.

Mõisted

Andmed – kvanitatiivsete ja kvalitatiivsete andmeelementide (väärtuste) kogum. Kõige üldisem mõiste andmete kohta. Ei määratle andmete struktuuri, talletamise viise, talletamise kohti, kasutamisviise, esitlusvorminguid, omanikke ega kasutajaid.

Andmekooslus e. kirje - Keerukaim lubatud konstruktsioon sama andmekoosluse sees on korduvgrupp, mis võib omakorda sisaldada korduvgruppi jne. Andmekoosluste koostamisel esitatakse konkreetsed piirangud andmekoosluse struktuuri kohta sh. piirangud korduvgruppide esinemise ja sisalduvuse taseme sügavuse kohta. Samuti korduvgruppide struktuuri kohta.

Andmekogum – piiritletud ja semantiliselt terviklik kollektsioon sama struktuuriga andmekooslusi, mis on käsitletav ühtse tervikuna. Sama andmekogumi sees olevad andmekooslused ei pea olema omavahel loogiliselt seotud aga nad võivad seda olla. Enamikel juhtudel on ainsaks seoseks samasse andmekogumisse kuuluvate andmekoosluste vahel kuuluvus samase andmekogumisse so. semantiline samasus.
Kasutataksega üldmõistena mistahes suuruse ja struktuuriga andmehulga kohta.

Andmekogumi struktuur – mingi konkreetse andemkogumi sisemine andmete organisatsioon.

Andmekogumi struktuuri mudel – mingis formaalses notatsioonis kirjeldatud mudel, mis kirjeldab konkreetse andmekogumi struktuuri ja võimaldab selles olevaid andmeelementide väärtusi formateerida ja/või tõlgendada. Minimaalselt kirjeldab andmekogumi struktuuri mudel andmeelementide loendi ja alam-loendid (korduvgrupid) ning andmeelementide formaadid.

Andmestu – kogum omavahel loogiliselt ja füüsiliselt seotud andmekooslusi. Lihtsamal juhul koosneb andmestu ühest andmekooslusest. Siia alla kuuluvad SQL-andmebaasides, NoSql-andmebaasides, XML-failide, JSON-faiides ja mistahes muudes struktuursetes failides talletatud andmed.

Andmestu struktuur – mingi konkreetse andemestu sisemine andmete organisatsioon.

Andmemudel (andmestu struktuuri mudel) - mingis formaalses notatsioonis kirjeldatud mudel, mis kirjeldab andmestu struktuuri ja võimaldab tõlgendada selles sisalduvaid andmeid.

Andmemudeli aegpüsivus – andmemudeli võime säilitada jooksvalt oma komponentide vaheliste seoste struktuuri stabiilsena. Ideaalsel juhul andmemudeli selline organisatsioon ja seisund, kus juba loodud mudelite komponentide vahelisi seoseid ei muudeta (ei ole vaja muuta) kunagi. Aegpüsiva mudeli muutus seisneb uute komponentide lisandumises mudelisse ja nende uute komponentide sidumises oma vahel ja juba olemasolevate komponentidega.

Andmestu kirjeldus – omavahel seotud formaliseeritud ja formaliseerimata kirjelduste kogum, mis kirjeldab mingi konkkreetse andmestu sisemise struktuuri kõik aspektid. Sama andmestruktuuri kirjeldust implementeerivaid andmestuid võib olla mitu.

Andmestruktuur – üldmõiste kõikide erinevate andmestruktuuride kohta sh. andmete struktuur, andmekoosluse struktuur, andmestu struktuur jms.

I**nformatsioon** – andmed konkreetses kasutuskontekstis e. kontekstsed andmed e. eesmärgistatud andmed.

Andmebaas – andmestu, kus samas keskkonnas koos andmetega hoitakse nende samade andmete koosluse ja struktuuri (seoste) masin-loetavas vormingus kirjeldusi.

Andmebaasijuhtimisüsteem e. andmebaasisüsteem – (lühend ABJS, ingl. k. DBMS – DataBase Management System) mingi konkreetse tootja või vabavara alliansi poolt loodud ühtse nimega tähistatav vahendite komplekt andmebaaside loomiseks, hävitamiseks, haldamiseks (andmestruktuuride loomime, muutmine, protseduuride loomine, kasutajate loomine ning nendele õiguste andmine, varundamine, taastamine jms.) ja kasutamiseks (andmete lisamine, uuendamine, kustutamine, pärimine, protseduuride käiivtamine jms.)

Andmebaasiserver - (lühend ABS, ingl. k. DBS) füüsilise või virtuaalse serveri peale installeeritud andmebaasi(juhtimis)süsteem. Mõningatel juhtudel vaadatakse seda koos hallatavate andmebaasidega. Väiksemate infosüsteemide puhul ongi see tavaliselt nii, et andmebaasid asuvad samal füüsilisel või virtuaalsel serveril, kus asub andmebaasiserver. Suuremate infosüsteemide puhul moodustab

andmebaaside talletamise füüsiline ja virtuaalne keskkond andmebaasi(juhtimis)süsteemidest eraldi seisvaid vahendite kogumeid.

Andmebaasimootor – Andmebaasijuhtimissüsteemi komponent. Installeeritud andmebaaiserveri keskne täiturprogramm, mis võtab vastu kasutajatelt saabunud korraldusi, täidab need ja tagastab tulemuse korralduse esitajale.

Stuktuurne andmete kogum – andmete kogum, kus sama kogumi raames mitme andmekomplekti kooseis on kirjeldatav/kontrollitav sama reegliga. Reegel ei pea olema kirjutatud andmete juurde. See tähendab seda, et ka andmestu, mis ei ole andmebaas, võib olla struktuurne.

Riiklik andmekogu – Seaduses sätestatud piirides andmeid haldav infosüsteem. Koosneb andmestust, tarkvarast selle andmestu haldamiseks, reeglitest ja protsessidest,

Andmemudelite normaliseerimine – relatsiooniline andmebaas-tüüpi andmestu andmemudeli viimine vastavusse mingi konkreetse normaalkujuna kirjeldatud reeglistikule.

Normaalkuju – relatsiooniline andmebaas-tüüpi andmestu andmemudeli orgeniseerimisreeglite kogum, mis võimaldab hallata anomaaliaid.

Andmemudelite koostamise mustrid – käesolevas dokumendis kirjeldatud reeglite kogumid, mida kasutatakse andmete modelleerimisel ja millega tagatakse andmemudelite aegpüsivus.

XML – (Extensible Markup Language) standardiseeritud struktuuriga märgistuskeel andmekoosluste esitamiseks, mis on SGML alamosa (http://www.w3.org/TR/xml/).

JSON – (JavaScript Object Notation) lihtne andmeesitusformat, mis baseerub programmeerimiskeele JavaScript alamosal. (http://www.json.org/ ja http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf)

Transaktsioon (andmebaasi mõttes) – (ehk tehing) teistest transaktsioonidest eraldatud terviklik tegevuste jada andmebaasisüsteemides, kus on realiseeritud transaktsiooni käsitlus. Transaktsioon hõlmab tööprotsessi minimaalse tegevuste kogumi, mis kirjeldab tööprotsessi mõistes tervikliku ja jagamatu toimingu. Tuntud ka COMMIT/ROLLBACK loogika nime all. Transaktsioon kirjeldub ACID piiranguga.

Transaktsiooniandmebaas – andmebaasijuhtimissüsteem, milles on realiseeritud transaktsioonkäsitlus või andmebaasiserver mis on sellise andmebaasijuhtimissüsteemi implementatsioon (andmebaasiserver) mingil infrastruktuuril (füüsilisel või virtuaalsel serveril).

SQL - andmepäringu keel, mida kasutavad anmebaasijuhtimissüsteemid. Structured Query Language. (http://www.iso.org/iso/home/store/catalogue_tc/catalogue_tc_browse.htm?commid=45342)

SQL-andmebaas – andmebaasijuhtimissüsteem või selle implementatsioon (andmebaasiserver) mingil infrastruktuuril (füüsilisel või virtuaalsel serveril), mis kasutab päringukeelena SQL-keelt. On olemas andmebaasijuhtimissüsteeme, mis peale SQL-keele implemeneteerivad veel mõnda teist päringukeelt koos SQL-keelega. Ka sellised andmebaasid kuuluvad SQL-andmbaaside mõiste alla.

NoSql-andmebaa – (not only SQL) andmebaasisüsteem, mis võimaldab andmete salvestamist mitte-tabeli kujul. Eesmärgiks on andmestruktuuride "horisontaalse skaleeritavuse" suurendamine ja samade andmete mitmetesse eksemplaridesse paljundamise lihtsustamine. Praegusel hetkel on seda tüüpi andmebaasid kasutusel peamiselt andmete kogumise ja esitamise valdkonnas, mis ei eelda suuremat andmetöötlust (andmete teisendamist).

Andmete terviklikkus – mõiste viitab loogilisele kontseptile, mida erinevat tüüpi andmestustes võib käsitleda erinevalt. Relatsioonilised andmemudelite ja andmebaaside puhul seob endas kahte mõistet seose terviklikkust ja olemi terviklikkust.

Seose terviklikkus – relatsioonilise andmemudeli seose terviklikkuse all mõistetakse reeglite komplekti, mis kirjeldab olemite vahelisi seoseid kirjeldavate väärtuste ilmnemise võimalusi. Iga reegel kirjeldab seose kahe olemi vahel või hierarhilise seose ühe olemi sees. Reegel koosneneba kahest osas. Reegli esimene osa kirjeldab olemi mingisuguse atribuudi või atribuutide grupi kui unikaalse primaarvõtme (primary key) st. määrab sellel atribuudile või atribuutide grupile unikaalsuse piirangu, mis piirab antud olemi raames sellele atribuudile või atribuutide grupile kui kogumile korduvate väärtuste tekkimist ja määratleb selle atribuudi või atribuutide grupi kui primaarvõtme. Reegli teine pool kirjeldab mingi

olemi mingi atribuudi kui viida mingi olemi (kas mingi muu olemi või sama olemi) primaarvõtmele. Seda kirjeldust nimetatakse välisvõtmeks ja sellega piiratakse välisvõtme atribuudile või atribuutide gruppi kuuluvate atribuutide väärtuste ilmnemist selliselt, et ükski välisvõti ei saa atribuutide väärtusena omada sellist väärtuste komplekti, mida pole olemas seotud olemi primaarvõtme väärtuste komplektide hulgas. Alternatiivina saab välisvõtme väärtus olla tühi (NULL), kui selline situatsioon on mudelis lubatud st. viide võib ka puududa.

Andmebaasi seose terviklikkuse all mõistetakse olukorda, kus omavahel seotud andmed (andmebaasitabelis olevad kirjed/tabeli read) viitavad üksteisele ja ei tekki situatsiooni, kus mingid andmed viitavad sellistele andmetele (“on seotud” selliste andmetega) mida pole olemas. Relatsioonilistes andmebaasides kirjeldatakse selle mõiste kaudu olukorda, kus tabeli kõigi ridade kõigi välisvõtmete (foreign key) väärtused viitavad mingi tabeli mingi kirje primaarvõtme (primary key) väärtusele või on nende väärtus määramata (NULL). Seejuures iga välisvõtme puhul on kirjeldatud, millise konkreetse tabeli välisvõtme väärtusele nad viitavad

NoSql-andmebaasidega seoses üldiselt seda mõistat ei kasutata. Väidatakse, et seda pole võimalik saavutada. Samas NoSql andmebaaside kasutamise seisukohalt on vaja seose terviklikkuse mõiste defineerida.

Transaktsioon (ärilises mõttes) – ärilises mõttes on transaktsioon minimaalne terviklik tegevuste jada. “Minimaalne terviklik” tähistab siin tegevuste kogumit, mille jagamine osadeks ei ole võimalik seetõttu, et jaotamise tulemusena saadud osategevustel puudub iseseisev äriline tähendus.

Identifitseeriv suhe – suhe mudeli kahe olemi vahel, kus suhtes ülemusena osaleva olemi (suhte ühesel poolele oleva olemi) primaarvõti päritakse suhtes alluvana osaleva olemi primaarvõtme koosseisu. Üldjuhul tähendab see seda, et alluv olem ei ole oma tähendust ilma ülemus-olemita. Meetodit ei kasutata (kuna muutub selisel kujul mõttetuks) kui olemite identifitseerimiseks ja sidumiseks kasutatakse surrogaatvõtmeid.

Mitte-identifitseeriv suhe – suhe mudeli kahe olemi vahel, kus suhtes ülemusena osaleva olemi (suhte ühesel poolele oleva olemi) primaarvõti päritakse suhtes alluvana osaleva olemi koosseisu väljaspoole primaarvõtit.

Olemi terviklikkus - igal relatsioonilise andmemudeli olemil on primaarvõti, kõik primaarvõtmesse kuuluvad andmeväljad on alati väärtustatud ja nende väärtuste kombinatsioon on alati selle olemi piires unikaalne.

Primaarvõti (primary key / masterKey) – relatsioonilistes andmemudelites ja andmebaasides olemi atribuutide või tabeli veergude lühim grupp, mille väärtus on lühim ja millel ilmnev väärtus on üle kõigi andmete unikaalne. Kui primaarvõti koosneb mitmest väärtusest ei tohi ühegi andmekomlekti (kirje) puhul olla ükski nendest väärtustest tühi (NULL). Andmebaasides ei tohi primaarvõtme ühegi komponendi väärtust muuta.

NoSql andmemudelites ja andmebaasides – võib selle mõite samastada kirje võtme väärtusega kuigi ametlikus terminoloogias seda nii ei kasutata. Kehtib samuti unikaalsuse reegel ja reegel, mis keelab kirje võtme väärtuse muutmist.

ACID - (relatsioonilisele / transakstsioonilisele) andmebaasile seatud piirangute komplekt, mille täitmine tagab, et andmebaasitransaktsioonid on usaldusväärsed (so. terviklikud ja üheselt mõistetavad):

Atomicity – transaktsiooni atomaarsus (terviklikkus), mis tähendab seda, et iga transaktsioon on terviklik või pole teda üldse – kui tranksaktsiooni alamosa teostamisel tekkib viga siis on vigane terve transakstioon ja andmed jäävad muutmata.

Consistency – järjepidevus (stabiilsus) on piirang, mis tähendab seda, et transakstioon peab viima andmebaasi ühest lubatud olekust teise (järgmisse) lubatud olekusse. St .et iga andmebaasi kirjutatud andmekomplekt peab vastama kõikidele andmebaasis kirjeldatud reeglitele (piirangud (constraints), tiggeritega juhitavale andmekontollile jms.)

Isolation – eraldatus (isoleeritus) on reegel, mis piirab, et korraga täidetakse ainult ühte transaktsiooni. See tähendab omakorda seda, et transaktsioonide täitmisel on järjestus ja järgmist transaktsiooni saab täitma asuda alles pärast seda, kui eelmise täitmine on lõppenud. Seda võib implementeerida ka nii, et lõpetamata transaktsioonide andmed ei ole nähtavad kellelegi teisele peale transaktsiooni omaniku.

Durability – andmete püsivus on reegel, mis ütleb, et pärast transaktsiooni lõpetamist (commit) säilivad andmed samas olekus, millised nad olid transaktsiooni

lõppedes olenemata tulevikus toimuvates mistahes kriisidest (elektri katkestus, programmi vead, riistvara/tarkvara riknemine jms.)

ACID piirangute tagamiseks kasutatakse andmebaasides andmete lukustamist - transaktsiooni ajaks so. kuni muutmise lõpuni ja muudatuste jõustamiseni (commit).

BASE – (**B**asically **A**vailable, **S**oft state, **E**ventual consistency)

Surrogaatvõti – võti mis ei oma reaalses maailmas mingit tähendust ja mille ainuke eesmärk on olla unikaalne. Surrogaatvõtme "tähendusetu tähendus" loob olukorra, kus sellel väärtust ei ole mõtet muuta, kuna uus väärtus on "sama mõttetu" kui eelmine väärtus.

Relatsioonilistes andmebaasides kasutatakse primaarvõtmete väärtustamisel, et luua "muutuste kindlaid" seoseid andmete vahel. Kui primaarvõtmena kasutada mõnda loomulikku, elust pärinevat andmeelementi või andmeelementide gruppi, siis jääb alati võimalus, et mõnda nendest väärtustest muudetakse. Sellisel juhul läheb muutmisele ka vastava primaarvõtme väärtus. See omakorda põhjustab välisvõtmete väärtuste muutmise vajaduse. Surrogaatvõtmete kasutamisel selline anomaalia puudub.

Nendes NoSql andmebaasides, kus andmete jaoks kasutatakse genereeritud tehnilisi võtmeid, kasutatakse surrogaatvõtit sisuliselt võtmete genereerimise ühe osana võtme struktuurist või siis ka tervikvõtme genereerimisel kuigi otseselt seda terminit (surrogaatvõti) ei kasutata.

Surrogaatvõtmete genereerimiseks kasutatakse peamiselt kahte meetodit:

1. surrogaatvõti on järjenumber
2. surrogaatvõti on ette antud pikkusega, juhulikult genereeritud sümbolite jada (hash)

Mõlemal juhul on võimalik surrogaatvõtme väärtust arvutada kahes põhimõtteliselt erinevas kohas - surrogaatvõtme väärtust saab arvutada andmebaasis või mingis teises tarkvara kihis (mõnes vahekihis või rakenduses eneses).

Esimesel juhul, võtmeväärtuse arvutamisel andmebaasis, kasutatakse enamasti andmebaasi poolt pakutavat instrumenti, mille nimi on sequence ja mis igakordsel küsimisel annab järgmise järjenumbri. Samas on erinevatel aegadel üsna palju kasutatud meetodit, miile nimeks on MAX+1. Selle meetodi puhul küsitakse

andmebaasitabelist välja konkreetse tabeli maksimaalse võtme väärtus ja lisatakse sellele 1. Samas on võimalik kasutada andmebaasis ka võtme genereerimist hash-meetodil. Kuna võtmed genereeritakse andmete lisamise transaktsiooni sees, siis mingit lisakontrolli võtme unikaalsuse kohta pole sellisel juhul vaja teha.

Teisel juhul, võtmeväärtuse arvutmaisel andmebaasist väljaspool, saab surrogaatvõtmeid genereerida kas MAX+1 meetodil või hash-meetodil. Sellisel käsitlusel tuleb iga andmete lisamise operatsiooni juures kontrollida võtme unikaalsust baasis ja unikaalsusreegli rikkumise tekkimisel genereerida uus võti. Unikaalsusreegli rikkumine võib tekkida põhjustatuna kahe erineva kasutaja üheaegsest soovist samal ajal samasse tabelisse uusi andmid lisada.

Big data – selle mõistega katekase kõik andmekogumid, mis on mahult liiga suured ja liiga varieeruva (horisontaalse) komplekssusega, et neid töödelda “klassikliste infotöötluse vahenditega”. Viimaste all mõistatakse tavalislt relatsioonilisi andmebaasisüsteeme. Vaadeldakse kui ühte ja ainsat omavahel seotud andmete kogumit (data set) või kui paljusid andmekogumeid, millede sosed ja andmemaht on sama võrreldes ühtse andmekogumiga. Andmed mida pole “üldlevinud” tarkvaravahendite abil võimalik “mõistliku ajaga” koguda, kureerida (curate, curation), hallata ja töödelda. Andmed, mille maht, liikumiskiirus (sisse-välja) ja struktuurilinr varieeruvus on suur. Nende käsitlust piiratakse tihti tegevustega: kogu, talleta, reorganiseeri, analüüsi, jaga.

BOD (Begin of Days) – universaalselt kasutatav, väga kauges minevikus asuv kuupäev koos kellaajaga (timestamp). Konkreetse andmebaasi puhul ideaalsel juhul varaseim kuupäev, mida andmebaas on DATETIME (TIMESTAMP) tüüpi väljas suuteline talletama. Paljudel juhtudel piisab ka väärtusest 01.01.1800 00:00:00, kuid tegelikkuses peab vaatama, et see oleks piisavalt varajane aeg, kuhu reaalseid andmeid ei sattu. Kasutatakse mineviku asuva “lõpmatuse” (määramata kehtivuse alguse aja) kirjeldamiseks.

EOD (End of Days) – universaalselt kasutatav, väga kauges tulevikus asuv kuupäev koos kellaajaga (timestamp). Konkreetse andmebaasi puhul ideaalsel juhul hiliseim kuupäev, mida andmebaas on DATETIME (TIMESTAMP) tüüpi väljas suuteline talletama. Piisab ka väärtusest 31.12.3000 23:59:59. Kasutatakse tulevikus asuva “lõpmatuse” (määramata kehtivuse lõpu aja) kirjeldamiseks.