TARTU ÜLIKOOL

MATEMAATIKA-INFORMAATIKATEADUSKOND

Arvutiteaduse instituut

Informaatika eriala

Juta Vaks

Mõõduka kiirusega muutuva dimensiooni käsitlemise meetodid andmeladudes

Bakalaureusetöö

Juhendaja: Konstantin Tretjakov, MSc

Autor: ……..... "….." juuni 2010

Juhendaja: ……..... "….." juuni 2010

Lubada kaitsmisele

Professor: ……..... "….." …….. 2010

TARTU 2010

Sisukord

Sissejuhatus ... 3
1. Taust ... 4
1.1. Andmelao mõiste ... 4
1.2. Andmelao andmemudelid ... 5
1.3. Aja mõõde andmelaos ja muutuvad dimensioonid ... 7
1.4. Muutuvate dimensioonide haldamise tehnikad ... 8
1.4.1 Tüüp 1 – väärtuse ülekirjutamine ... 8
1.4.2 Tüüp 2 – uue rea lisamine ja ajaloo säilitamine ... 9
1.4.3 Tüüp 3 – kahe versiooni säilitamine ehk paralleelne reaalsus ... 10
1.4.4 Tüüp 6 – ajalooline ja viimane seis kõrvuti, hübriidmeetod ... 10
1.4.5 Tüüp 2 meetodi täiendused ... 11
2. Probleemipüstitus ... 13
3. Ülevaade kirjandusest ... 15
3.1. Minidimensioon ... 15
3.2. Minidimensiooni puudused ... 18
3.3. Alternatiivid ... 18
4. Wela projektis teostatud lahendus ... 20
4.1. Dimensioonitabelite jagamine ... 20
4.1. Atribuutide muutuste salvestamine ... 22
5. Kirjeldatud meetodi analüüs ... 23
5.1. Dimensioonitabelite arvu suurenemine ... 23
5.2. Keerukuse kasv ... 24
5.3. Andmemahu kasv ... 24
6. Eeltööd meetodi rakendamiseks andmelaos ... 26
6.1. Atribuutide stabiilsuse hindamine semantika ja taustainfo abil ... 26
6.2. Atribuutide metainfo kaevandamine sisendandmetest ... 27
6.3. Muutumiskiiruse arvutamine ... 30
6.4. Stabiilsuse järgi grupeerimine ja tulemuste võrdlemine ... 31
Kokkuvõte ... 34
Abstract (in english) ... 35
Kirjandus ... 36

Sissejuhatus

Multidimensionaalne modelleerimine on praeguseks kujunenud *de facto* standardiks andmeladude projekteerimisel. Dimensioonitabelitel on tähtis roll igas andmelaos – siduda faktitabelis salvestatud sündmus vajalike olukorra kirjelduste ja muu informatsiooniga olemite kohta, mis sündmuses osalesid.

Andmelao oluline ülesanne on salvestada ajalugu, see tähendab ajalise mõõte lisamist igale salvestatud infokillule. Ka dimensiooniolemid on ajas muutuvad. Kui sündmusega seotud olemid on aja jooksul muutunud, peab iga sündmuse kohta olema võimalik tuvastada toimumishetkel kehtinud tingimused ehk dimensiooni atribuutide toonased väärtused.

Muutuvate dimensioonide käsitlemise jaoks on väljakujunenud kindlad meetodid, kuid enamus nendest meetoditest eeldab, et dimensioonid muutuvad suhteliselt aeglaselt ja etteaimatavalt. Antud töö eesmärgiks on analüüsida võimalusi muutuste efektiivseks haldamiseks, kui dimensioonid ei vasta nimetatud eeldustele.

Töö on motiveeritud reaalse andmelao projekteerimise käigus esilekerkinud probleemidest. Antud andmehoidlaks on tehnoloogia arenduskeskuse STACC ja AS Regio koostöös valmiva Wela rakenduse andmeladu. Traditsioonilised meetodid ei olnud antud projektis sobilikud ning oli vaja otsida alternatiivseid lahendusi.

Töö esimeses peatükis antakse ülevaade andmelao ja multidimensionaalse mudeli põhimõistetest ja tutvustatakse traditsioonilisi muutuva dimensiooni käsitlemise meetodeid. Järgmistes peatükkides selgitatakse põhjalikumalt vajadust alternatiivsete meetodite jaoks ning tutvustatakse kirjanduses esitatud lahendusi. Neljandas peatükis on kirjeldatud Wela rakenduses kasutatud meetod, mis on modifikatsioon olemasolevatest lahendustest. Viies peatükk analüüsib kasutatud meetodi eeliseid ning võimalikke puudusi. Viimases peatükis selgitatakse põhjalikumalt vajalikke eeltöid meetodi efektiivseks kasutamiseks ja analüüsitakse võimalusi nimetatud tööde automatiseerimiseks.

1. Taust

Käesolevas peatükis selgitame andmelao mõistet ja andmelaondust üldisemalt, anname ülevaate andmete struktuurist andmelaos ja muutuvate dimensioonide probleemist.

1.1. Andmelao mõiste

On üldteada, et andmelaod on oma eesmärkidelt ja projekteerimispõhimõtetelt erinevad operatsioonilistest andmebaasidest. Erinevalt operatsioonilistest andmebaasidest, mis on suunatud transaktsiooniliste tegevuste toetamisele, on andmeladude eesmärk pakkuda tuge nõustus- ehk otsusetegemisabisüsteemidele. Kuigi operatsioonilistes süsteemides võib vajalik info olemas olla ei ole see kasutajale kergesti kasutatav analüütiliseks töötlemiseks.

Andmelao üldtunnustatud definitsioon pärineb Bill Inmon'ilt ja kõlab järgmiselt: andmeladu on struktureeritud kogum integreeritud, kindlale teemale suunatud, püsiva loomuga ja ajast sõltuvaid andmeid, mille ülesandeks on toetada otsuste tegemist [7].

Teemale orienteeritus tähendab, et andmeladu on ärivaldkonna mingitest konkreetsetest vajadustest sõltuv ja andmed on koondatud ümber kesksete teemade. Integreeritus viitab sellele, et sageli on andmelattu koondatud andmeid mitmetest eraldiseisvatest süsteemidest ning salvestatud andmed on omavahel ühilduvad. Andmete allikateks võivad olla nii organisatsiooni operatsioonilised süsteemid (erinevad andmebaasid, rakenduste logifailid) või ka välised süsteemid näiteks avalikud demograafilised andmebaasid, valuutakursside tabelid, ilmavaatlused. Püsiva loomuse nõue tagab andmete kestvuse ja säilitamise. Ajast sõltuvus viitab, et igal andmeüksusel on ajaline mõõde ja samal andmeüksusel võib olla erinev väärtus erinevatel ajahetkedel. Seega kaks viimast karakteristikut tähendavad, et andmeladu peab võimaldama muutusi andmete väärtustes ilma nende eelnevalt salvestatud väärtusi ülekirjutamata.

Lisaks erinevatele eesmärkidele (ning nende tõttu) erinevad andmelaod ja operatsioonilised andmebaasid veel järgnevates punktides:

i. Andmelao ja operatiivse andmebaasi lõppkasutajad on erinevad. Wiley'd tsiteerides - operatsioonilise süsteemi kasutajad liigutavad organisatsiooni rattaid, andmelao kasutajad jälgivad kuidas rattad liiguvad [10].

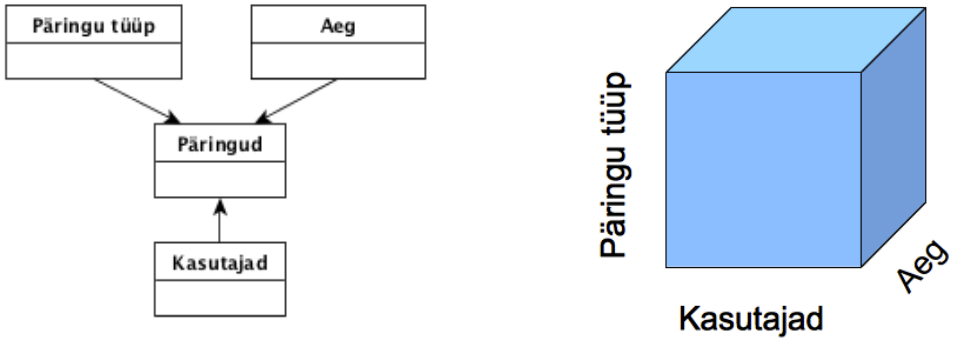
ii. Erinevad kasutajad sooritavad andmebaasidest erinevaid päringuid. Andmeladudest tehakse harilikult analüütilisi päringuid, operatsioonilistes andmeladudes sooritatakse transaktsioone. Analüütilised päringud sisaldavad tavaliselt erinevaid grupeerimis- ja agregeerimisoperatsioone üle suurte andmehulkade, samas kui transaktsioone iseloomustavad lühikesed lugemis- ja kirjutamisoperatsioonid.
iii. Andmete sisestusprotsessid on erinevad. Andmelao allikad pärinevad harilikult operatsioonilistest süsteemidest, nende sisestusprotsess on sageli pakktöötlus, mille käigus andmed integreeritakse, puhastatakse ja struktureeritakse andmelao jaoks.
iv. Keskkonna karakteristikud on erinevad. Nõuded kiirusele, kättesaadavusele, andmete värskusele, salvestusruumile. Andmeladusid iseloomustavad sageli suured andmemahud.

1.2. Andmelao andmemudelid

Andmelao projekteerimises eristatakse harilikult järgmisi samme: nõuete kogumine ja analüüs, kontseptuaalse andmemudeli loomine, andmelao loogilise ja füüsilise mudeli loomine [7, 3, 8, 2, 15].

Kontseptuaalse modelleerimise eesmärgiks on teostus-sõltumatu ja kõrge abstraktsioonitasemega kirjeldav multidimensionaalne mudel, mille tegemist alustatakse kasutaja vajaduste ja sisendsüsteemide analüüsist.

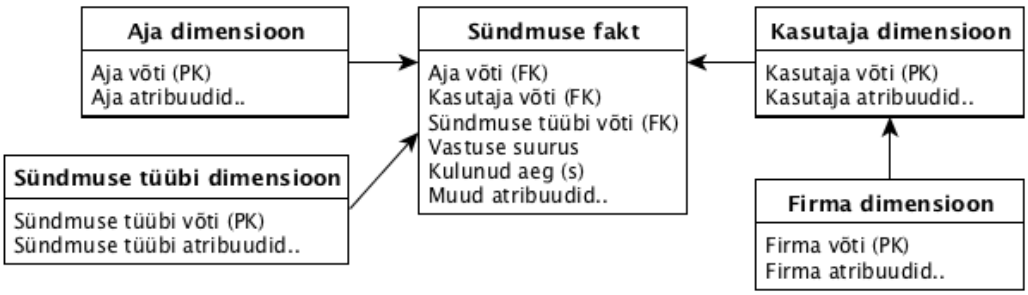
Traditsiooniliselt on nõustussüsteemide kontseptuaalseks mudeliks multidimensionaalne mudel, kus andmed on organiseeritud kesksete teemade ehk faktide ümber ja erinevaid vaatenurki nimetatakse dimensioonideks. Seda võib visualiseerida andmekuubina, kus kuubi telgedeks on dimensioonid ja erinevate dimensioonide väärtuste kokkupuutepunktides on faktid. Faktideks on mingid sündmused või mõõtmistulemused. Dimensioonitelgedel võib andmeid kategoriseerida hierarhilistesse tasemetesse, võimaldades andmeid vaadata suurema või väiksema granulaarsusastmega. Säärase mudeli kasutamisel nõustussüsteemides ja andmeladudes on kaks peamist hüve. Ühest küljest on see sarnane äri- ja andmeanalüütikute mõtteviisile, mistõttu ta on kasutajatele kergesti mõistetav. Teisalt on mudel analüütiliste päringute jaoks jõudlusefektiivne, kuna selle lihtne struktuur võimaldab andmelao disaineril ette aimata kasutajate käitumist. Joonis 1 illustreerib võimalikke kontseptuaalseid mudeleid.



Joonis 1. Erinevad viisid esitada sama kontseptuaalset mudelit.

Andmelao andmemudel ehk loogiline mudel koostatakse kontseptuaalse mudeli pealt. Lisaks kontseptuaalsele mudelile on oluliseks sisendiks ettevõtte või organisatsiooni operatsiooniliste andmebaaside skeemid (kui need on andmete allikateks). Mudeli koostamisel võetakse arvesse andmelao konkreetset teostusplatvormi (näiteks relatsiooniline või objektorienteeritud andmebaas) ja mittefunktsionaalseid nõudeid (salvestusruumi optimeerimine, jõudlus ehk päringute vastuse kiirus jms).

Mitmemõõtmeline modelleerimine ja analüütiline kasutuskoormus nõuavad spetsiaalseid projekteerimistehnikaid ka loogilise mudeli jaoks. Relatsiooniliste andmebaasisüsteemide jaoks on väljakujunenud spetsiifilised mudelid ehk skeemid, mis on tänapäevaks üldtunnustatud tehnikad ning toetatud paljude tootjate poolt. Tuntumad neist on täht-, lumehelbe ja tähtkuju skeemid. Joonis 2 kujutab lumehelbeskeemi, kus keskseteks faktideks on veebirakenduses sooritatud päringud.



Joonis 2. Lumehelbeskeem veebirakenduse päringutest.

Sagedasemad küsimused loogilise andmemudeli koostamisel on aja mõõte lisamine andmeelementidele, viiteterviklikkuse tagamine, dimensioonide versioneerimine,

normaliseerimine ja denormaliseerimine, andmete eelagregeerimine ja fragmenteerimine, vajaliku granulaarsuse määramine ning andmete stabiilsuse analüüs. Igale küsimusele lahendust otsides tuleb silmas pidada, kuidas saavutada soovitud tasakaal salvestusmahu, jõudluse ja mudeli keerukuse vahel. Antud töö keskendutakse peamiselt loogilise modelleerimise etapis esile kerkivatele probleemidele ja nende lahendusmeetoditele.

Loogilises mudelis on juba selgesti tuvastatavad andmelao tabelid ja vajalikud võtmed tabelites. Vahel koostatakse viimase sammuna ka veelgi täpsem füüsiline mudel, mis keskendub konkreetse andmebaasihaldussüsteemi küsimustele. Füüsilises mudelis asenduvad olemite nimed ja atribuudid konkreetsete tabelite ja veerunimedega, lisatakse andmetüübid, piirangud, indekseerimisinfo.

Projekteerimist ei pruugi teha jäigalt eelpool kirjeldatud järjekorras, tihti koostataksegi mudeleid paralleelselt. Seda enam, et sageli jäetakse kontseptuaalne mudel ka koostamata, piirdudes vaid nõuete analüüsiga ning alustatakse otse loogilise skeemi koostamisest.

1.3. Aja mõõde andmelaos ja muutuvad dimensioonid

Ajaline mõõde on olemas peaaegu kõigis andmeladudes. Seda seepärast, et peaaegu iga andmeladu on sisuliselt aegrida (kindlatel ajahetkedel saadud mõõtmistulemuste või sündmuste rida) ning sõltuvus ajast on üks andmehoidla põhilistest omadustest.

Ajalise mõõte lisamine võimaldab hoida ja kasutada ajaloolist informatsiooni. Ligipääs ajaloolistele andmetele ja vahendid nende analüüsiks on üks otsusetegijate peamisi ootusi andmelaole. Ajaloolised andmed on vajalikud, et otsida neist käitumismustreid, tuvastada ja jälgida trende äritegevuses, teha võrdlusi erinevate ajaperioodide tulemuste vahel ning teha nendest ennustusi tuleviku tarbeks. Analüüsi käigus on ärianalüütikutel vaja vaadelda andmeid erinevatest vaatenurkadest, grupeerida ja summeerida erinevate atribuutide kaupa või vaadelda ettevõtte varude või muude karakteristikute seisu suvalisel mineviku ajahetkel.

Ajaline mõõde ja muutused ei kehti ainult faktide kohta, ka dimensioonitelgedel olev info ei ole ajast puutumatu. Dimensiooniolemite tunnused võivad omada erinevatel ajahetkedel erinevad väärtusi. Andmelao oluline nõue on, et info oleks püsiv ja faktid säilitaksid

muutumatu seose dimensioonidega. Kasutajatel on vaja analüüsida fakte nende toimumishetkel kehtinud tunnuste järgi. Sellest järeldub, et me peame ka dimensioonide muutumisel salvestama nende erinevatel ajahetkedel kehtinud seisud. Näiteks, kui firmade dimensioonis firma X regioon muutub, on vaja säilitada mõlemad versioonid. Vajadus erinevate versioonide järele tekib näiteks juhul, kui ärianalüütikud soovivad analüüsida eelmise aasta fakte regioonide kaupa. Kui me regiooni lihtsalt üle kirjutaksime, satuksid kõik eelmise aasta faktid uude regiooni.

Dimensioonide atribuutide muutumise kohta on kujunenud välja oma termin – aeglaselt muutuvad dimensioonid (ingl.k. *slowly changing dimensions,* lühend *SCD*). Väga kiiresti muutuvaid dimensioone nimetatakse vastavalt kiiresti muutuvad dimensioonid (ingl.k. *rapidly changing dimensions*). Tasub märkida siiski, et viimane termin ei ole väga laialt kasutusel ning sageli kasutatakse terminit *SCD* ka mõõdukalt või isegi kiiresti muutuvate dimensioonide puhul, sest *SCD* alla koondatakse baastehnikad, mida võib rakendada teatud mööndustega ka kiiresti või mõõdukalt muutuvate dimensioonide puhul.

1.4. Muutuvate dimensioonide haldamise tehnikad

Klassikaliselt on olemas kolm meetodit ehk käitumismalli aeglaselt muutuvate dimensioonide muutuste haldamiseks [10] Neist on omakorda on tekkinud hulk modifikatsioone ja hübriidmeetodeid eri liiki situatsioonide lahendamiseks. Meetodeid nimetatakse lihtsalt Tüüp 1, Tüüp 2 ja Tüüp 3. Tinglikult võib lisada ka null tüübi, mis väljendab lihtsalt juhtu, kus muutusi ei arvestata ja dimensioone ei muudeta kunagi.

Kõikide dimensioonide iga atribuudi muutuseid pole vaja säilitada. Mõningad muutused tulenevad lihtsalt andmete parandamisest ning mõnede atribuutide puhul pole erinevate väärtuste teadmisest mingit praktilist kasu. Seepärast tuleb enamasti sobiv lähenemine otsustada atribuudi kaupa, mitte terve dimensiooni kaupa.

1.4.1 Tüüp 1 – väärtuse ülekirjutamine

Ülekirjutamine tähendab, et eelnevalt salvestatud dimensiooni atribuudi väärtus kirjutatakse üle uue informatsiooniga. Muutub ainult atribuudi väärtus, muutus ei mõjuta faktitabelit ega võtmeid tabelite vahel. Seda lähenemist kasutatakse sageli korrektsioonide tegemiseks. Kui tervet dimensiooni esimese tüübina käsitleda, on iga loogilise dimensiooni olemi kohta dimensioonitabelis ainult üks rida. Iga fakt on seotud dimensiooni hetkel

kehtiva e. atribuudi viimase väärtusega. Meetodi plussiks on kindlasti lihtsus ja kerge teostatavus, ometi on selge, et mingit ajalugu ei säilitata. Sellest hoolimata on tal oma kindel koht andmelaos, näiteks juhul kui atribuudi muutuse säilitamine ei anna mingit praktilist kasu.

1.4.2 Tüüp 2 – uue rea lisamine ja ajaloo säilitamine

Kõik atribuudi erinevatel ajahetkedel kehtinud väärtused säilitatakse andmebaasis. Iga kord kui mingi atribuudi väärtus muutub, lisatakse tabelisse uus rida, kus muutunud atribuudil on uus väärtus, kõik teised aga jäävad samaks (kui nad juhuslikult samaaegselt ei muutunud). Oluline on mainida, et antud lähenemise jaoks on vajalik kindlasti eraldi surrogaatvõtmete kasutamine primaarvõtmetena dimensioonitabelis. Kui dimensioonitabeli primaarvõtmeks oleks olemi naturaalne või operatsioonilises süsteemis kasutatav võti, siis rikuks uus rida primaarvõtme unikaalsust, sest naturaalvõti on mõlemil real sama. Seepärast kasutatakse primaarvõtmena surrogaatvõtmeid ja uus rida saab uue unikaalse väärtuse, millele edasised faktid hakkavad viitama. Naturaalvõti on dimensioonitabelis harilik atribuut ning jääb mõlemil real (ja ka kõigil edaspidistel ridadel) samaks. Naturaalvõti võimaldab ühe olemi read kokku siduda. Surrogaatvõtmena on soovitav kasutada unikaalseid numbrijadasid. Ei soovitata kasutada kombinatsioone naturaalvõti+versiooninumber ega mitmeveerulisi võtmeid (näiteks versiooni või ajatempliga).

Lisaks faktide sidumisele korrektsete ajalooliste väärtustega, on Tüüp 2 meetodil ka teine positiivne omadus - ta võimaldab jälgida, millised muutused on dimensioonis toimunud (loomulikult nende atribuutide ulatuses, millede muutusi antud viisil salvestatakse). Iga muudatusega kasvab suureneb dimensioonitabeli ridade arv ehk dimensioonitabel kasvab vertikaalselt.

Tabel 1. Näide atribuudi Osakond muutuste salvestamine Tüüp 2 meetodil

Kasutaja võti	Kasutaja ID	Osakond	Muud atribuudid
12345	444	Müük	...
23456	444	Klienditugi	...

1.4.3 Tüüp 3 – kahe versiooni säilitamine ehk paralleelne reaalsus

Atribuudi eelmise ja praeguse väärtuse hoidmiseks tekitatakse dimensioonitabelisse

lisaveerg. Meetod võimaldab kasutada atribuudi erinevaid väärtusi andmete grupeerimisel paralleelselt käitudes justkui muutust poleks toimunud, siit ka nimetus paralleelne reaalsus. Kaheks korraga eksisteerivaks versiooniks on harilikult kas "praegune" ja "eelmine" või "praegune" ja "esimene". Strateegia puuduseks on ilmselgelt, et sellisel meetodil saab hoida ainult ühte muutust, enamate muutuste jaoks tuleks lisada uusi veerge, mille tulemusel muutub andmebaasi skeem väga keerukaks. Dimensioonitabel kasvab horisontaalselt (iga atribuudi kohta mida me soovime selliselt säilitada, tuleb teha uus veerg). Täielik ajalooline informatsioon ei säili, samuti pole teada millal vana väärtus uue vastu vahetati. Isegi kui muutuste toimumishetk eraldi välja salvestada, on selle informatsiooniga töötamine ebamugav.

Tabel 2. Näide atribuudi Osakond muutuste salvestamine Tüüp 3 meetodil.

Kasutaja võti	Kasutaja ID	Originaal osakond	Praegune osakond	Muud atribuudid
12345	444	Müük	Klienditugi	...

1.4.4 Tüüp 6 – ajalooline ja viimane seis kõrvuti, hübriidmeetod

Esineb olukordi kus ärianalüütikud soovivad fakte vaadelda osadel juhtudel dimensiooni mingi tunnuse ajaloolisest perspektiivist teinekord aga hetkel kehtiva seisu järgi. Teiste sõnadega, atribuut peab vahel käituma kui Tüüp 2 (ajalooline väärtus) vahel aga kui Tüüp 1 (ülekirjutatud viimase seisuga).

On selge, et ükski eelpool mainitud meetoditest seda kergesti ei võimalda. Tabeli iseendaga liitmise (ingl k. *self join*) või alampäringute abil on see küll teostatav, kuid vajalikud päringud on tavakasutaja jaoks keerukad või aeganõudvad. Seepärast on välja pakutud strateegia, mis on omamoodi hübriid kõigist kolmest, ja kannab seetõttu nime Tüüp 6 (1+2+3). Samas nõuab see lähenemine suuremat salvestusmahtu ning keerulisemat laadimisprotsessi, mistõttu ei pruugi alati otstarbekas olla.

Lahenduseks on lisada atribuut dimensioonitabelisse kaks korda – näiteks Firma Regioon ja Firma Praegune Regioon. Esimeses hoitakse ajaloolist väärtust ja ta käitub kui Tüüp 2 atribuut, teise väärtus kirjutatakse muudatustega üle – ehk Tüüp 1 atribuut. Seega, kui muutus toimub, lisatakse tabelisse uus kirje analoogselt Tüüp 2 meetodile. Uues kirjes on atribuudi ajaloolise ja hetkel kehtivat infot näitavas veerus sama väärtus. Kõiki eelnevaid

dimensiooni versioone aga tuleb värskendada uue kehtiva väärtusega. Seda tuleks siis teha kõikide atribuutide jaoks, mida ärianalüütikud sääraselt kasutada soovivad.

Kui dimensioon on juba algselt väga lai ning enamus atribuutidest käituvad Tüüp 2 meetodi järgi, siis tundub dimensiooni tabeli laiuse kahekordistamine ebaotstarbekas. Sellisel juhul on juba mõtekam tekitada eraldi dimensioonid – Firma ja Pragune Firma. Praeguse Firma tabel oleks seotud ajalooliste andmetega välisvõtmega Firma dimensioonitabelis.

1.4.5 Tüüp 2 meetodi täiendused

Teisest tüübist on olemas mitmeid modifitseeritud versioone - lihtsamate ja keerulisemate täiustustega erijuhtumite ja/või ebaharilike ärinõuete lahendamiseks. Ka kolmandast tüübist on olemas modifikatsioone, kuid need on väga spetsiifilised ning jäävad antud töö skoobist välja. Kirjeldame siin paari enamlevinud modifikatsiooni Tüüp 2 tehnikast. Mõlemad muutused on lihtsad täiendused ja ei muuda põhikäitumist.

Üks täiendus, on lisada dimensioonile viimase väärtuse lipp-atribuut, mis on tõene ainult kõige viimasel versioonil dimensioonist. Atribuut võimaldab kasutajatel kiiresti piirata oma päringuid ainult praegu kehtivate profiilide kohta, see on sageli vajalik, kui kasutajad analüüsivad dimensioone ka iseseisvana, mitte ainult faktide kaudu.

Teine sage modifikatsioon on ridade ajatembeldamine. Ajatempel näitab hetke, kui atribuudi väärtused hakkasid kehtima või lõppesid. Kehtivusaja atribuute on kindlasti vaja andmete laadimisprotessis, kui andmed saabuvad erinevatel ajahetkedel või, kui me lisame ajaloolisi andmeid tagantjärgi, sest me peame teadma, milline dimensiooni kirje on sobiv faktile tema toimumishetkel.

Ajatemplid on ka mugav lisaväärtus dimensiooni enda analüüsiks. Me saame vaadelda nende abil dimensiooni seisu suvalisel ajahetkel või analüüsida atribuutide stabiilsust. Dimensioonitabelis on ajatemplid seega vajalikud laadimiseks ja dimensiooni uurimiseks, kuid nad ei ole vajalikud faktitabeliga sidumiseks. Kuna faktitabeli välisvõti viitab niigi korrektsele toimumishetkel kehtinud dimensioonile, on kehtivusaja kasutamine üleliigne.
Variatsioon ajatemplitest, mida Wela projekti andmelaos kasutatati, on kehtivusaja märkimine laadimistsükli vahemike abil. Andmelao laadimisprotsessi toimumiste kohta

peetakse eraldi logitabelit (kus on ka ajatempel). Igas dimensioonitabelis on kehtivusaeg defineeritud kui viide laadimisprotsessile, mil rida hakkas kehtima ning millises töötluses ta veel viimati kehtis. Sellise salvestuse eeliseks on fakt, et dimensiooni välisvõtmed (mis viitavad laadimistabelile) on täisarv tüüpi muutujad, mille salvestusmaht on tunduvalt väiksem kui oleks kahe kuupäev tüüpi muutuja puhul. Väikeste andmemahtude puhul triviaalne aspekt, kuid potentsiaalselt mitmete miljonite ridade puhul muutub see juba oluliseks.

2. Probleemipüstitus

Tüüp 2 ja tema modifikatsioonid on andmeladudes enim kasutatavad tehnikad. Seda seepärast, et selline lähenemine rahuldab kaks kõige tüüpilisemat ärinõuet – fakti saab vaadelda toimumishetkel kehtinud väärtuste järgi ning dimensiooniolemi muutusi võib vaadelda läbi ajaloo.

Vaatleme lähemalt kuidas toimub dimensioonitabeli kasv Tüüp 2 tehnika kasutamisel. Tabel 3 esitab võimalikke ridu kliendi dimensioonis peale aadressi muutumist kaks korda.

Tabel 3. Väljavõte ühe kliendiga seotud andmetest dimensioonitabelis.

Primaarvõti	Kliendi identifikaator	Aadress (Linn)	Kehtiv alates	Kehtiv kuni	Muud atribuudid
12345	567	Tartu	5	39	x, y, z..
29506	567	Kuressaare	40	87	x, y, z..
50794	567	Tallinn	88	..	x, y, z..

On ilmne, et Tüüp 2 muutuste lisamine tekitab andmebaasis andmeliiasust. Iga atribuudi muutuse kohta tekitatakse baasi terve uus rida, kus ülejäänud atribuudid on kõik samad. Andmeliiasust ei peeta dimensionaalses mudelis iseenesest probleemiks (võrreldes näiteks relatsioonilise mudeliga), kuid kui dimensioonitabel on suur või muutused toimuvad sageli, muutub kõikide andmete kordamine juba probleemiks. Kiiresti muutuva dimensiooni puhul võib mitmekordsete Tüüp 2 muutuste salvestamisel tabel kiiresti kasvada väga suureks. Mida enam on sageli muutuvaid atribuute või mida kiiremini atribuudid muutuvad, seda tormilisem kasv on.

Suur ja kiiresti kasvav on suhtelised mõisted. Suurte müügi-, teenindus- ja kindlustus-ettevõtete puhul on kliendi dimensioon sageli 100 või enama atribuudiga ning klientide arv ulatub sadadesse tuhandetesse ja isegi miljonitesse. Selge on ka see, et mida rohkem on atribuute, seda tõenäolisem on, et mõni neist muutub. Näiteks tabel 50 000 kliendiga, kus toimub keskmiselt 10 muutust kliendi kohta aastas, kasvab 10 aastaga umbes 5 miljoni realiseks (eeldades, et klientide arv ei kasva). See võib olla aktsepteeritav kasv. Teisalt, kui klientide arv on juba alguses 10 miljonit, siis kõigest paar muutust aastas kasvatab tabeli ridade arvu 10 aastaga sadadesse miljonitesse.

Sama probleem kehtib väga laiade dimensioonide korral. Kuna dimensioonid on oma olemuselt fakte kirjeldavad, siis domineerivad nendes tekstilised atribuudid. Ülejäänud atribuutideks on andmelao ja operatiivse süsteemi primaarvõtmed, mõned kuupäeva veerud ning numbrilised atribuudid (harilikult on need agregeeritud faktid).

Väga laiade dimensioonide korral pole võimatu, et ühele reale vastab ligi 1kB andmeid. Näiteks võime kujutleda olukorda, kus 100-veerulises normaliseerimata dimensioonis on ligi 75 tekstilist veergu, keskmise pikkusega 10 tähemärki. Ülejäänud atribuudid on 4-baidised täisarvud või kuupäeva veerud. Ühebaidiliste märgistike puhul kulub ühe tähemärgi salvestamiseks üks bait ning lisaks iga sõne kohta 1 või enam baiti üldkulu andmebaasi enda tarbeks. Seega kokku ühel real 75x11 + 25x4 = 925 baiti. Lisaks indekseerimisele kuluv andmemaht.

Tõsi, tähemärkide oletused sõltuvad tugevalt salvestatava info keelest ja keskmisest sõna pikkusest, kasutatavast märgistikust ja andmetüübist. Enamikes andmebaasisüsteemides saab keskmise rea pikkuse baitides mingil lõigul sisendandmetest lihtsalt arvutada. Lisaks tuleb arvestada, et paljud andmebaasimootorid tihendavad pikad sõned automaatselt, seega füüsiline salvestusmaht kettal võib olla väiksem. Kuid selge on, et ühe atribuudi muutmisel dubleerime me samaaegselt suure hulga mahukaid atribuute.

Kuigi andmelaod on projekteeritud haldama suuri andmemahte, ei saa neid laiendada lõpmatult. Põhiosa andmeladude mahust lasub küll faktitabelitel, kuid ka dimensioonitabelite kasvul tasub silma peal hoida. Kui hulk atribuute muutub sageli või pidevalt, ei tasu oodata probleemi lahendamisega. Võimalik, et projekteerimisprotsessi alguses pole andmebaasi analüütikul piisavalt taustateadmisi, et mõista andmete stabiilsust. Selle probleemi leevendamiseks on välja pakutud lahendusi töö teises osas.

Kokkuvõttes võib öelda, et Tüüp 2 toimib reeglina hästi, kui dimensioonitabel ei ole liiga lai (palju atribuute) või mahukas (palju kirjeid) ning kui muutused ei juhtu sageli. Antud töös käsitletakse võimalusi Tüüp 2 või tema hübriidmeetodite kasutamiseks muutuste haldamiseks, kui dimensioonitabel ei vasta nendele ootustele.

3. Ülevaade kirjandusest

Suurte ja kiiresti muutuvate dimensioonide teemat käsitlenud autorite lähenemised on üldjoontelt sarnased. Esimeseks sammuks on eraldada kiiresti muutuvad atribuudid originaaldimensioonist. See on ka loomulik lahendus. Lähenemised erinevad edasistes sammudes - kuidas seostada laialijagatud atribuudid faktidega ning algdimensiooniga.

Enne, kui atribuutide eraldamise juurde läheme, vaatleme kahte triviaalset viisi probleemi lahendamiseks. Esimene viis on väärtusgruppide kasutamine. Täpsete väärtuste asendamine väärtusvahemikega on lihtne võimalus vähendada muutuste arvu ühtlaselt muutuvate atribuutide korral. Näiteks atribuut kuusissetulek kliendi dimensioonis. Konkreetse arvu asemel salvestada kuusissetulek vahemikena: 5 tuhat ja alla, 5-10 tuhat, 10-15 tuhat jne. Selge, et väärtusgruppide kasutamisel salvestame vähem detaile, samuti on gruppide eeldefineerimine pööramatu protsess ja uusi gruppe on keeruline koostada. Teisalt ei pruugi lõppkasutajad konkreetsest numbrist huvitatud olla, vaid kasutavadki oma mudelites vahemikke. Seega on see ärianalüütikute otsus, milliste atribuutide puhul väärtusgruppide kasutamine on õigustatud ja millistesse gruppidesse andmed jagada.

Teine variant on muutlikke atribuute salvestada faktitabelis koos teiste transaktsiooni andmetega. Sellel on kaks puudust, esiteks pole faktitabelis olevat väärtust mugav kasutada dimensiooni olemite analüüsimisel. Teiseks, kui atribuudi muutumiskiirus osutub siiski piisavalt aeglaseks, ei pruugi korduva väärtuse salvestamine miljonitel faktiridadel olla enam otstarbekas.

3.1. Minidimensioon

Kimball, kes muutuvaid dimensioone kõige põhjalikumalt on käsitlenud, pakub suurte ja kiiresti muutuvate dimensioonide probleemi lahenduseks nn. minidimensioonide kasutamist [10-13].

Minidimensioon on defineeritud kui alamhulk algdimensiooni atribuutidest, mis on eraldatud väiksesse, iseseisvasse, kunstlikku dimensiooni, eesmärgiga hoida suure muutuva dimensiooni kasvu kontrolli all. Minidimensiooni sobib kasutada ka juhtudel, kus teatud hulk atribuute esineb päringutes sageli omavahelistes kombinatsioonides. Sageli

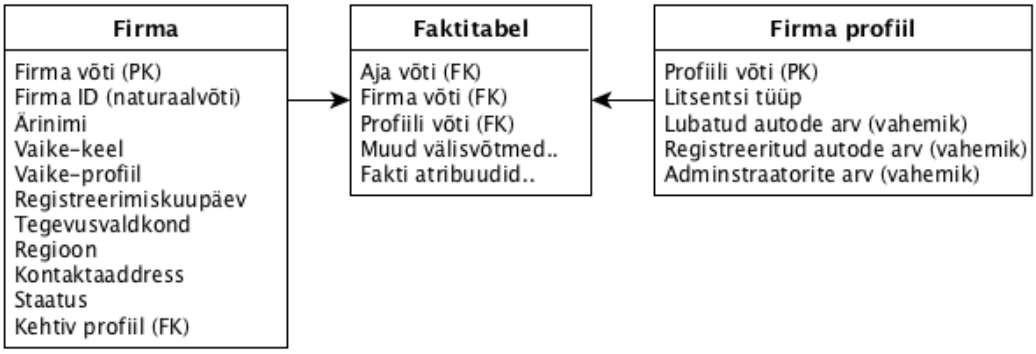
esineb atribuutide vahel ka teatav loogiline seos ning neid võib käsitleda kui iseloomustavat profiili – nt meessoost kliendid vanuses 20 kuni 25 aastat, keskmise sissetulekuga ja keskharidusega. Päringud, mis analüüsivad fakte ainult profiili kuuluvate atribuutide järgi, saavad nüüd suure osa klienditabeli andmetest tabelite liitmisest hoopis välja jätta.

Minidimensiooni tehnika rakendamine on võrdlemisi lihtne. Sageli päritavad või sageli muutuvad atribuudid eraldatakse algdimensioonist uude tabelisse. Uues tabelis hoitakse ühte rida atribuutide omavaheliste kombinatsioonide kohta, mitte enam ühte rida iga dimensiooni olemi kohta (näiteks erinevad võimalikud kliendi-profiilid). Seega on minidimensiooni suurus kordades väiksem algdimensioonist. Eelistatult koondatakse mindimensiooni väikese kardinaalsusega atribuudid. Algdimensiooni, millest on eraldatud minidimensioon, nimetame selguse mõttes põhidimensiooniks.

Kui kiiresti muutuv atribuut on arvulise väärtusega (näiteks kliendi sissetulek või vanus), siis tuleks täpsed väärtused asendada väärtuste vahemikega nagu eelpool selgitatud. Täpne väärtus koos kõikvõimalike teiste väärtuste kombinatsioonidega tekitaks minidimensiooni samapalju ridu kui algdimensioonis. Väärtusvahemike kasutamine on peamine kompromiss, mida minidimensiooni kasutamisel tuleb teha. Erandjuhul on võimalus täpset väärtust lisaks ka faktitabelis hoida, väärtusvahemik mindimensioonis tuleks siiski alles jätta, päringukiiruse ja kasutajasõbralikkuse huvides.

Faktitabelisse lisatakse uus välisvõti minidimensiooni primaarvõtmele, sel moel seotakse faktiga toimumishetkel kehtinud atribuutide kombinatsioon. Kui minidimensiooni eraldatud atribuutides toimub mingi muutus, hakkavad faktitabeli edasised read viitama lihtsalt uuele kombinatsioonile. Põhidimensiooni tabelis pole midagi vaja muuta. Faktitabeli kaudu on ka põhidimensioon seotud toimumishetkel kehtinud minidimensiooni väärtustega.

Joonisel 3 on kujutatud andmelao skeem, kus algsest Firma dimensioonist on osa iseloomustavaid tunnuseid eraldatud minidimensiooni nimega Firma profiil. Kuna profiili atribuudid on diskreetsed arvulised tunnused, on need asendatud vahemikega.



Joonis 3. Firma profiil minidimensioonina

Ridade arv minidimensioonis sõltub ilmselgelt atribuutide arvust ning iga atribuudi erinevate väärtuste arvust. Näiteks kolme atribuudi puhul, mis igaüks võivad saada 10 erinevat väärtust, on tabelis maksimaalselt 10x10x10 = 1000 rida. Kõikvõimalikke kombinatsioone ei pea tabeli loomisel valmis arvutama, neid võib lisada ka laadimisprotsessi käigus.

Minidimensioon käitub justkui harilik dimensioonitabel, selle erinevusega, et kõikide atribuutide kõikvõimalike muutuste jaoks on tabelisse vastavad read juba eelgenereeritud (või vähemalt võimalik genereerida) ja dimensioonitabeli suurus seega ette teada. Kui üks minidimensioon osutub liiga suureks, võime alati teha uue jagamise. Kui üksikud minidimensioonid on liiga väikesed, võib nad kokku liita.

Tasub mainida, et ka uus andmeskeem vastab endiselt tähtskeemi reeglitele. Kimball on tuline tähtskeemi pooldaja ning rõhutab sageli lumehelbe skeemi puudusi.

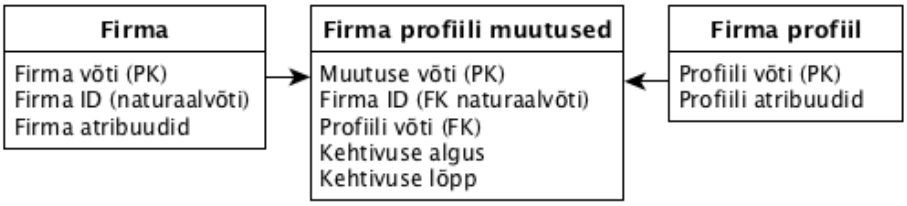
Oma olemuselt on minidimensioon sarnane nn kola-dimensioonile (ingl. k. *junk dimension*). Kola-dimensioon on abstraktne dimensioon, kuhu koondatakse kokku algandmetest väikese kardinaalsusega indikaator-atribuudid nagu näiteks erinevad tõeväärtustüüpi lipp-atribuudid [10]. Me võiksime nad jätta faktitabelisse jätta või lisada mingitele dimensioonidele, kuid seal tekib oht, et nad sageli muutudes muudavad kogu dimensiooni ebastabiilseks. Sarnaselt minidimensioonile tuleb ka kola-dimensiooni koostamisel silmas pidada maksimaalset ridade arvu. Kuna dimensioon on abstraktne ja väärtused omavahel nõrgalt korrelleeruvad on erinevate kombinatsioonide esinemine väga tõenäoline.

3.2. Minidimensiooni puudused

Põhidimensioon on seotud minidimensiooniga faktitabeli kaudu. See tähendab, et kui ärianalüütikud soovivad vaadelda algdimensiooni tervikuna, peavad nad liitma päringusse ka faktitabeli. See on minidimensiooni suurim puudus, kaob võimalus vaadata dimensiooniobjekte ilma faktitabelit kaasamata. Suure faktitabeli kaasamine päringutesse ainult välisvõtmete sidumiseks on äärmiselt ebaefektiivne. Meenutame, et dimensiooni ajaloo analüüsimine on üks Tüüp 2 tehnika kasutamise põhjustest.

Hetkel kehtivate väärtuste jaoks on lahendus lihtne – lisada põhidimensiooni välisvõti kehtivale minidimensiooni reale. Iga kord, kui profiili muutus tuvastatakse, tuleks värskendada dimensiooni välisvõtit viitega uuele minidimensiooni reale (ehk välisvõti käitub Tüüp 1 atribuudina). Sellisel juhul nimetatakse minidimensiooni põhidimensiooni tugijalaks (ingl k *outrigger table)*. Kindlasti ei tohiks kasutada Tüüp 2 tehnikat sellel välisvõtmel, muidu oleksime tagasi algpunktis.

Kuid keerulisem probleem on dimensiooni ajaloo analüüsimisega, info on küll kättesaadav, kuid selleks peaksime ikka kasutama faktitabelit. Lisaks, minidimensiooni uus atribuutide kombinatsioon seotakse põhidimensiooniga alles sel hetkel, kui mingi fakt või sündmus toimub. Minidimensiooni viidud atribuutide muutus aga võib toimuda juba varem. See ei pruugi olla aktsepteeritav. Parim lahendus on tekitada üks uus tabel nn. faktideta faktitabel, mis salvestab atribuutide muutusi koos ajatemplitega (vt Joonis 4). Tabelit nimetatakse faktideta faktitabeliks seepärast, et seal pole tavapäraseid mõõteandmeid.



Joonis 4. Faktideta faktitabeli kasutamine profiili muutuste salvestamiseks

3.3. Alternatiivid

Teine tunnustatud autor, kes on andmeladude arhitektuurist ja projekteerimisest kirjutanud, on Bill Inmon. Inmon ei pea multidimensionaalset lähenemist andmelao projekteerimiseks sobilikuks ning pakub välja alternatiivsed loogilised ja füüsilised mudelid. Kuid ka tema

projekteerimisprotsessis on eraldi sammuna olemas andmete stabiilsuse analüüs (ingl. k. stability analysis)[7]. Stabiilsusanalüüs on viimane samm loogilise andmemudeli koostamisel ning tema eesmärk on grupeerida andmed muutumise kiiruse järgi tabelitesse nii, et muutumissagedus tabelites oleks enam-vähem võrdne. Sama põhimõtet on kasutatud ka järgmises peatükis tutvustattava Wela meetod puhul, kuigi multidimensionaalses mudelis.

4. Wela projektis teostatud lahendus

Antud projektis oli vaja lahendada olukord, kus dimensioonid olid võrdlemisi laiad ning sisaldasid väga erineva muutumiskiirusega atribuute. Ühes dimensioonis leidus samaaegselt palju klassikalisi aeglaselt muutuvaid atribuute ning atribuute, mis olid väga ebastabiilsed, kuid mitte piisavalt ebastabiilsed, et neid faktitabelisse paigutada. Ometi oli vaja peaaegu kõikide atribuutide muutusi käsitleda Tüüp 2 tehnika järgi. Oli selge, et ebastabiilsete atribuutide tõttu kasvab dimensioonitabeli ridade arv kiiresti ning dubleeritakse palju aeglaselt muutuvaid atribuute.

4.1. Dimensioonitabelite jagamine

Welas kasutatud meetodi idee on lihtne ja baseerub samuti dimensioonitabeli lõhkumisel väiksemateks tabeliteks. Selgitame seda näite varal. Vaatleme lihtsuse mõttes ainult kahest tabelist koosnevat tähtskeemi – faktitabel ja dimensioonitabel. Oletame siinkohal, et andmelao projekteerijal on piisavalt taustainfot dimensiooni atribuutide tähenduse ja käitumise kohta, et nende muutumiskiirust hinnata. Järgmistes peatükkides käsitleme võimalusi muutumiskiiruste arvutamiseks.

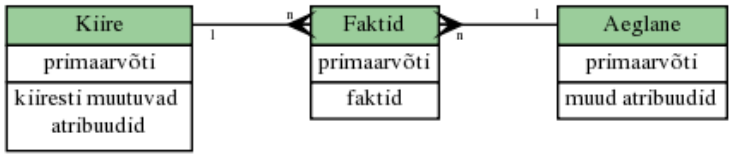
Esimese sammuna jagame dimensioonitabeli atribuudid omavahel stabiilsuse järgi gruppidesse ja moodustame igast grupist eraldi tabelid. Esialgu vaatame lihtsuse mõttes kaheks jagamist, aga võime ka moodustada rohkem tabeleid.

Seega tekib 3 tabelit: faktitabel, stabiilne ehk aeglaselt muutuv dimensioon ja ebastabiilsem ehk kiiresti muutuv dimensioon. Siinkohal ei kasuta me põhi- ja minidimensiooni mõisteid, sest üks uus dimensioon ei ole teisest tähtsam, samuti ei pruugi nad olla väga ebavõrdse suurusega. Eristamiseks viitame neile lihtsalt kui kiire ja aeglane.

Igale tabelile lisame primaarvõtme ja kehtivust näitavad ajatemplite veerud. Primaarvõtmena kasutame kindlasti surrogaatvõtit. Surrogaatvõti on siin ka hädavajalik, kuna igas tabelis hakkame jälgima kõikide atribuutide muudatusi Tüüp 2 meetodil.

Senimaani on sammud sarnased minidimensiooni meetodiga. Erinevus seisneb selles, kuidas tekkinud kolm objekti omavahel siduda - võimalikke kombinatsioone on ainult kolm:

i. Faktid on mõlemi tabeliga mitu-ühele seoses ehk eelnevalt kirjeldatud minidimensiooni meetod, Joonis 5.

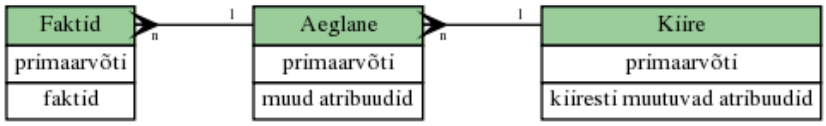


Joonis 5. Faktid seotud kiire ja aeglase dimensiooniga otse.

ii. Faktid on mitu-ühele seoses kiire dimensiooniga, mis on omakorda mitu-ühele seoses aeglase dimensiooniga, Joonis 6.

Joonis 6. Faktid seotud kiire dimensiooniga.

iii. Faktid on mitu-ühele seoses aeglase dimensiooniga, mis on omakorda mitu-ühele seoses kiire dimensiooniga, Joonis 7.



Joonis 7. Faktid seotud aeglase dimensiooniga.

Wela projektis teostati variant number kaks. Variant kolm osutub ebaotstarbekaks ja ei paku tegelikult mitte mingit leevendust algseisule. See saab ilmseks, kui vaadelda muutuste ja välisvõtmete haldamist dimensionitabelites.

Mitu-ühele seoste teisendamisel andmebaasi tabeliteks tuleb seega faktitabelile lisada välisvõti kiirele dimensioonile ja kiiresse tabelisse omakorda välisvõti aeglasele dimensioonile.

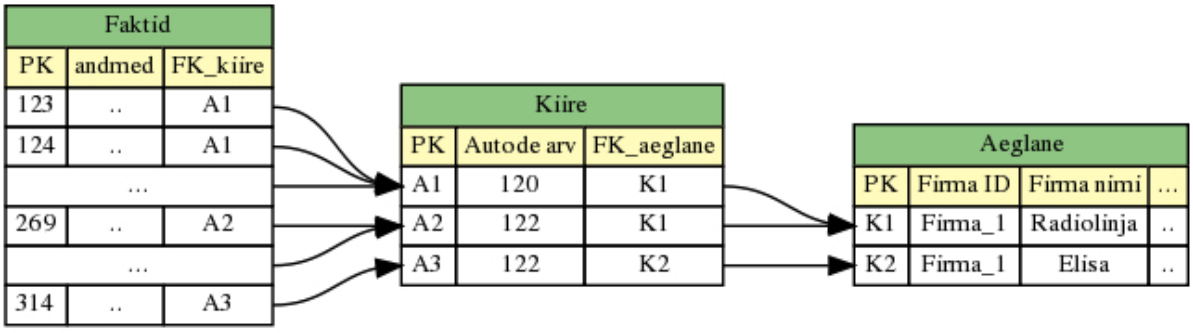
Kui algdimensioon jagatakse kolmeks või enamaks tabeliks, tuleb tabelid samamoodi stabiilsuse järgi kasvavalt järjestada seespoolt väljapoole. Tulemus on lumehelbe skeem, kus keskel on kõige ebastabiilsem faktitabel ja äärtes kõige püsivamate atribuutidega tabelid. Kuna algne dimensioon on kontseptuaalselt siiski tervik, hoolimata sellest, et on

eraldi tabelitesse jaotatud, peab tabelitesse jaotatud atribuutide komplekt moodustama fakti ajahetkel kehtiva terviku. Seega ühel ajahetkel viitab faktitabel kindlale reale kiires dimensioonis ja sealse välisvõtme kaudu ka ühele kindlale reale aeglases dimensioonis.

4.1. Atribuutide muutuste salvestamine

Kui dimensiooniolemi kiiresse tabelisse viidud atribuutides toimub muutus, märgitakse tabelis nende eelmine kirje kehtetuks ja lisatakse uus kirje, millele edaspidised faktid viitama hakkavad. Uues kirjes on siis uuendatud atribuutide väärtused, kuid aeglasele tabelile viitava välisvõtme väärtus on endine – sest aeglaselt muutuvas dimensiooniosas muutust ei toimunud ja ka uus kirje viitab endistele aeglastele atribuutidele. Selliselt võivad ebastabiilsed atribuudid muutuda järjest mitmeid korda, viidates ikka kõik ühele ülemtabeli kirjele. (Vt Joonisel 5 toodud kiire tabeli esimesi ridu, mis viitavad samale ülemtabeli kirjele K1)

Samas, kui aeglaselt muutuvates atribuutides toimub muutus, on lähenemine veidi keerulisem. Kõigepealt tuleb sisestada aeglaselt muutuvasse tabelisse uue primaarvõtme ja muutunud väärtustega kirje. Selleks, et edaspidiseid fakte selle uue kirjega siduda, tuleb lisada uued kirjed kõikidesse vahepeal asetsevatesse tabelitesse, uuendades vastavaid välisvõtmeid. Selliselt on säilitatud ajalooline korrektsus fakti kehtivushetke väärtustega.



Joonis 5. Firma_1 atribuutide muutuste salvestamine erinevates tabelites.

Joonis 5 illustreerib kuidas muutuste tulemusena erinevate tabelite read teineteisele viitavad. Kujutletud on olukord, kus ühel olemil on atribuudid muutunud nii kiires kui aeglases dimensiooniosas.

5. Kirjeldatud meetodi analüüs

Kirjeldatud meetodi peamine eelis on ilmne, muutused kiirete atribuutide väärtustes ei too enam kaasa kogu dimensiooni info üleliigset dubleerimist. Erinevalt minidimensiooni meetodist transformeerub algne tähtskeem lumehelbe skeemiks. Selle abil on ületatud minidimensiooni nõrk koht – vajadus kasutada faktitabelit dimensiooni iseseisvaks analüüsiks. Samas nii dimensioonide lõhkumine kui lumehelbe skeemi tekitamine võivad endaga kaasa tuua negatiivseid kõrvalnähte.

5.1. Dimensioonitabelite arvu suurenemine

On selge, et dimensioonide lõhkumisega tekitame me dimensioonitabeleid juurde. Enamus andmelao konstrueerimise põhisuunistest soovitab vältida ülearu suurt dimensioonide arvu. Harilikus andmelao modelleerimisprotsessis võib dimensioonide arvu kasv tuleneda kas lumehelbeskeemi kasutamisest või faktitabeli ülenormaliseerimisest.

Kimball, kes on tuntud tähtskeemi pooldaja, väidab, et normaliseeritud lumehelbe mudeli dimensioonitabelid mõjuvad negatiivselt erinevate atribuutide ristkasutusele ning takistavad *bitmap*-indeksite kasutamist. Kettaruumi võit, mis tabelite normaliseerimisega saavutatakse on harilikult väiksem kui 1 protsent kogu andmehoidla mahust [10]. (Märkus: antud kettaruumi hinnangus ei arvesta ta kiiresti muutuvate dimensioonide täisajaloo säilitamisel tekkivat hüppelist kasvu).

Faktitabeli ülenormaliseerimine ohustab tähtskeemi kasutajaid. Andmelao konstrueerijal võib tekkida kiusatus siduda kõik hierarhilised dimensioonid otse välisvõtmetega faktitabelisse, et säilitada tähtskeemi kuju. Kimball nimetab tulemust "sajajalgse" skeemiks (faktitabelil näib olevat tohutu hulk jalgu). Kuid säärane mudel toob endaga tegelikkuses kaasa hoopis kettaruumi kasvu, sest iga välisvõtme salvestamine miljonitel faktitabeli ridadel on juba arvestatav maht.

Kuigi dimensioonide arvu kasvul võivad olla objektiivsed põhjused, on selge, et dimensioonide arvu suurenemisega kaasneb alati risk, päringu vastuse kiirus väheneb. Mida rohkem on päringus erinevaid tabeleid omavahel liidetud, seda keerulisem on andmebaasimootoril koostada optimaalset päringuplaani. Samuti suureneb ketta-

operatsioonide arv andmete lugemiseks. Samas tuleb mainida, et tänapäevaste relatsiooniliste andmebaasimootorite jaoks pole mitme tabeli liitmine enam nii suur probleem, kui see oli 10 aastat tagasi.

5.2. Keerukuse kasv

Teine negatiivne külg dimensioonide lõhkumisel on keerukuse kasv. Infosüsteemi loogilised komponendid (valdkonna olemid) ei ole enam füüsilises mudelis kergelt tuvastatavad. Üks olem võib olla jagatud kaheks või enamaks füüsiliseks tabeliks. Ühes tabelis võivad olla koos erinevate olemite andmed jne. Tabelitele tekib väga palju kunstlikke lisaveerge (surrogaatvõtmed, välisvõtmed, kehtivusaega näitavad atribuudid). Kõik see tähendab, et päringu koostaja peab hoolikalt jälgima, et vajalikud alamtabelid saaksid omavahel õigete väljade abil liidetud ning atribuudi filtrid oleksid rakendatud vastavalt soovitud aja-vaatele (viimase kehtiva väärtuse, fakti tekkehetkel kehtinud väärtuse või mõne muu meelevaldse ajahetke järgi).

Kui andmelao lõppkasutajad kasutavad andmete pärimiseks alati vahekihti, mis nende eest füüsilise mudeli keerukuse peidab, pole see probleem. Kuid kui kasutajad koostavad ise päringuid, nõuab keerulisem mudel neilt lisateadmisi ning oskusi keerulisemate päringute koostamiseks. Keerukusega suureneb alati inimliku eksimise oht.

Keerukuse kasv ei ole probleemiks mitte ainult andmebaasi lõppkasutajatele. Ka andmelao laadimisprotsess muutub sellevõrra kindlasti komplitseeritumaks ja vähem läbipaistvaks. Erinevate dimensioonide ja alamdimensioonide naturaal- ja surrogaatvõtmete ning omavaheliste välisvõtmete seoste käitlemine vajab lisahoolt. Samuti sõltub dimensioonide arvust muutuste tuvastamine. Kuna sõltuvuste ahel algab välimistest tabelitest, tuleb nendest alustada ka laadimisprotsessi. Kõigepealt laadida andmed välimistesse tabelitesse, lisades sinna vajadusel uued read (kas muutunud atribuutide või uute objektide jaoks), enne kui struktuuris allpool olevaid tabeleid saab andmetega täita.

5.3. Andmemahu kasv

Kuigi töös kirjeldatud tehnikate eesmärgiks on dimensioonitabelite mahtu just kontrolli all hoida, võib vale käitumise korral tegelikult hoopis vastupidise efekti saavutada.

Meeldetuletuseks: minidimensiooni tehnika kasutamine lisab välisvõtmeid faktitabelisse, seega nende liigne kasutamine võib ootamatult hoopis salvestusmahu kasvu tekitada (eelpoolkirjeldatud „sajajalgse“-näide).

Lumehelbe skeemi puhul on oht dimensiooni ahelate muutuste kaskaad. Nagu eelpool juba kirjeldasime, tuleb dimensioonitabelite ahela kõige välimise tabeli muutuse korral teha muudatus ka kõigis ahela vahepealsetes tabelites. Seejuures pole oluline, kas ahel on moodustatud loogilise hierarhia tabelitest (näiteks poemüüja → osakonnajuhataja → poejuhataja → piirkonnajuhataja) või kirjeldatud meetodil tekkinud ahel.

Toodud meetod on küll oma olemuselt selle ohu eest paremini kaitstud, sest kogu tabelite ahel koostatakse muutumiskiiruse järgi ning välimistes tabelites eeldatakse väga harva muutusi, kuid viga võib tekkida muutumiskiiruste valest hinnangust. Samal põhjusel oleks eelpoolkirjeldatud variant 3 ebatarbekas, sest seal paikneksid välimistes tabelites kõige ebastabiilsemad atribuudid.

6. Eeltööd meetodi rakendamiseks andmelaos

Kirjeldatud meetodil dimensioonide jagamise eesmärk on moodustada atribuutide grupid nii, et atribuutide täisajaloo salvestamisel oleks mahu kasv minimaalne. Gruppide moodustamise peamine kriteerium on atribuutide muutumiskiirus. Teatud tingimustel võib olla eelistatum mudel, kus salvestusmaht on veidi suurem, kuid on saavutatud parem päringukiirus.

Optimaalse mudeli leidmiseks tuleb seega teada atribuutide tähendust, muutumiskiirust, salvestusmahtu ja kui võimalik, siis hinnata ka kasutajate päringus esinemise sagedust.

Antud peatükis kirjeldatakse muutumiskiiruse hindamist subjektiivselt taustainfo abil. Juhul kui taustainfo on puudulik, analüüsitakse võimalusi kaevandada taustainfot ja muutumiskiirusi sisendandmetest. Seejärel kirjeldatakse, kuidas arvutada tabelite kasvuprognoose, teades üksikute atribuutide muutumiskiirusi.

6.1. Atribuutide stabiilsuse hindamine semantika ja taustainfo abil

Harilikult võime eeldada, et dimensiooni füüsiliselt fikseeritud atribuudid nagu näiteks firma dimensiooni puhul aadress, riik ja nimi muutuvad suhteliselt harva.

Lisaks dimensiooni füüsilistele atribuutidele esinevad andmeladudes sageli nn agregeeritud atribuudid. Wela rakenduse pilootprojekti iseloomustas olukord, et paljusid operatsioonilises süsteemis hoitud andmeid ei peetud vajalikuks täies mahus andmelattu salvestada. Ometi oli ärianalüütikutelel soov analüüsida fakte nende puuduvate andmete agregeeritud väärtuste järgi. Näiteks firma autode üksikuid kirjeid pole mõtet andmelaos salvestada, kuid autopargi suurus on huvipakkuv firma tunnusena. Säärased atribuudid on dimensioonis sageli pidevad väärtused ning üpris ebastabiilsed.

Teistliiki agregeeritud faktid on andmelao siseselt faktitabelite pealt arvutatud atribuudid. Kasutajad on sageli huvitatud võimalusest näha käitumist iseloomustavaid mõõdikuid dimensioonide tunnustena, et kasutada neid oma päringutes piirangutena. On ju mugav, kui mingile tingimusele vastavaid firmasid ei pea kõigepealt eraldi päringuga faktide pealt

tuvastama, et siis edasi analüüsida ainult neile tingimustele vastavate firmade andmeid. Seda liiki agregeeritud faktide muudatusi ei pea samas Tüüp 2 tehnika järgi salvestama, harilikult käituvad nad kui Tüüp 1 atribuudid, misttõttu antud töös neid rohkem ei käsitleta.

6.2. Atribuutide metainfo kaevandamine sisendandmetest

Kuid kuidas hinnata stabiilsust, kui projekteerija ei oma piisavalt teadmisi ärivaldkonnast või dokumentatsiooni atribuutide tähenduse kohta, et otsustada atribuutide tähenduse ja eriti muutumiskiiruse kohta. Atribuutide nimetused võivad olla väheinformatiivsed lühendid ning nende tegelikku tähendust pole nimetustest ja andmetüüpidest alati võimalik järeldada. Seega tuleb vajalik info kaevandada sisendallikate andmetest.

Atribuutide stabiilsuse arvutamiseks on vajalik teatud kogus ajaliselt järjestatud transaktsioone. Iga transaktsiooni iseloomustab mingi sündmuse tulemus ning toimumishetkel kehtinud tingimused. Mida pikema ajaperioodi andmehulga peal muutusi jälgida, seda täpsema tulemuse saavutame.

Kirjanduses on andmekaeve tehnikate kasutamist andmelao projekteerimise abivahendina piiratud teadmiste keskkonnas analüüsinud mitmed autorid [16][17][8]. Sisendandmete analüüsis keskendutakse peamiselt tugevate ja ligilähedaste funktsionaalsete sõltuvuste leidmisele operatsioonilistes andmebaasides. See on samaväärne andmebaaside pöördprojekteerimisega.

Artiklis [16] kirjeldatakse funktsionaalsete sõltuvuste tuvastamist assotsiatsioonireeglite abil andmelao kontekstis. Sidudes leitud sõltuvused olemasolevate taustateadmiste ja ärimudelite kirjeldusega, saab teha järeldusi atribuutide semantika ja struktuuri kohta. Assotsiatsioonireeglite ja funktsionaalsete sõltuvuste omavahelistest seosest ja erinevustest annab ülevaate ka [1].

Assotsiatsioonireeglid on olemuselt mustrid atribuutide väärtuste vahel ja neid väljendatakse implikatsioonina $X \rightarrow Y$, kus X ja Y on erinevate atribuutide väärtustest koosnevad hulgad. Eeldatakse, et atribuutide väärtuste hulgad on teineteist välistavad, kui nad seda ei ole võib väärtuse alati esitada koos atribuudi nimega. Reeglile määratakse ka usaldustegur c, mis mõõdab reegli tugevust, ja tugi a, mis näitab olulisust statistilises mõttes.

Näiteks rahvusvahelises autoparandustööde tabelis, kus on olemas veerud registreerimis-riik ja roolitüüp, võib tuvastada reegli {riik.GB} → {roolitüüp.parem}, c=90%, a=20%. See tähendab, et 90% Suur-Britannias registreeritud autodest on parempoolse rooliga ja 20% kõigist andmebaasi sisestatud autodest on registreeritud Suur-Britannias ja parempoolse roolitüübiga.

Funktsionaalsed sõltuvused on aga määratud atribuutide endi vahel, mitte nende väärtuste vahel. Olgu antud relatsioon R, atribuutide hulgad X ja Y selles relatsioonis, siis X määrab funktsionaalselt Y, kui igale atribuudi X väärtusele vastab alati sama Y väärtus. Ehk Y on funktsionaalselt sõltuv X-st.

Kahe mõiste omavaheliseks sidumiseks defineeritakse assotsiatsioonireegli mall ehk reegli kavatsus (ingl k. *association rule intention*) [16]. Mall, sarnaselt funktsionaalsetele sõltuvustele, kirjeldab seoseid atribuutide vahel mitte väärtuste vahel. Ülaltoodud autoparandustööde assotsiatsioonireegli malliks oleks Riik → Roolitüüp. Igale mallile võib vastata hulk eksemplare ehk assotsiatsioonireegleid. Antud mallist võib seega lisaks olla veel eksemplar {riik.EST} → {roolitüüp.vasak}. Malli eksemplarid on atribuutide väärtuste omavahelised kombinatsioonid. Funktsionaalse sõltuvuse saab nüüd defineerida assotsiatsiooni reegli mallina, mille kõikvõimalikud eksemplarid omavad usaldustegurit 100%, kui nende tugi on suurem kui 0%.

Lisaks assotsiatsioonireeglitele on olemas ka palju teisi andmekaevanduse algoritme funktsionaalsete sõltuvuste leidmiseks. Näiteks TANE [5] ja tema modifikatsioonid CTANE, FastCFD, CFDMiner [4], Coords [6]. Tasub tähele panna, et andmehulkadest tasub kaevandada ligilähedasi funktsionaalseid sõltuvusi, mitte nn tugevaid funktsionaalseid sõltuvusi, sest me oletame, et piisavalt pika perioodi sisendandmetes on muutuvaid dimensioone.

Ligilähedase funktsionaalse sõltuvuse algoritmid defineerivad sõltuvuse X→Y ligiläheduse ridade arvu kaudu, mis tuleks relatsioonist eemaldada, et sõltuvus kehtiks. Sõltuvuse viga ε, $0 \leq \varepsilon \leq 1$, iseloomustab kui suure osa moodustavad erandid või vead relatsioonis, mis mõjutavad sõltuvust. Funktsionaalse sõltuvuse algoritmid ei vaatle andmeid järjestatult vaid relatsiooni tervikuna.

Kuid ka leitud ligilähedased funktsionaalsed sõltuvused ei ole piisavad ega sobilikud atribuutide muutumiskiiruste hindamiseks. Suure andmehulga peal, kus dimensiooni atribuudid mitu korda muutuvad, näitavad leitud funktsionaalsed sõltuvused ainult mõningaid väga püsivaid seoseid ja atribuute ning väikse andmehulga peal võib tulemus olla eksitav.

Selgitame funktsionaalsete sõltuvuste ja muutumiskiiruse mõistete seost ning erinevust näite varal. Olgu meil relatsioon R={A,B,C,D,E}, mis kajastab andmelao sisendandmeid ajaliselt kasvavas järjekorras. Tabel 4 demonstreerib näidisväljavõtet säärasest sisendandmete tabelist.

Antud tabelis tuvastaks algoritm sõltuvuse A→B, ε = 0,1. See tähendab, et üheksal real kümnest sõltuvus kehtib. Eemaldada tuleks rida number 7, et funktsionaalne sõltuvus oleks täielik. (Muudel ridadel kehtib: A=2 → B=1, kuid seitsmendal real A=2 → B=2).

Tabel 4. Näidisväljavõte järjestatud tabeli ridadest.

Reanumber	A	B	C	D	E
1	1	1	a	xy	2
2	1	1	a	xx	4
3	2	1	b	xy	2
4	3	2	b	xx	2
5	2	1	b	xx	3
6	3	2	b	xx	4
7	2	2	b	xy	1
8	1	1	a	xy	3
9	2	1	b	xy	3
10	3	2	b	xy	2

Ligilähedane funktsionaalne sõltuvus tuvastab seega relatsioonil statistiliselt kõige enam esinevad seosed. Andmelao seisukohalt huvitab meid aga atribuudi muutumiskiirus ehk tema väärtuste muutuste arv mingis ajaperioodis. Muutumiskiirus ei ole võrreldav erinevate väärtuste arvuga ajaperioodis või keskmiselt kõige sagedamini esineva väärtusega. Antud sõltuvuses A→B toimus tegelikult kaks korda muutus: esmalt seitsmendal real ja teisel korral üheksandal real. Atribuut võib omandada sama väärtust mitu korda erinevatel ajahetkedel, kuid andmelao kontekstis huvitab meid ainult muutuse toimumine. See, kas muutuse käigus saadud uus väärtus on varem esinenud või mitte, ei oma antud juhul tähtsust.

Seega funktsionaalsete sõltuvuste leidmine võib pakkuda mingil määral huvi andmelao projekteerijale, kuid pole piisav ajaliselt muutuva dimensiooni optimeerimiseks atribuutide stabiilsuse järgi. Kirjandusest ei õnnestunud ka leida viiteid otseselt stabiilsuse hindamise algoritmide kohta.

Funktsionaalseid sõltuvusi võib muidugi kasutada võimalike võtmete tuvastamiseks sisendandmetes. Samuti ei saa alahinnata leitud tugevaid funktsionaalseid sõltuvusi – need identifitseerivad muutumatud või väga aeglaselt muutuvad atribuudid.

6.3. Muutumiskiiruse arvutamine

Oletame, et meie näidisandmetes veerg A on naturaalvõti ja määrab teised atribuudid üheselt antud kirje lisamise ajahetkel. Päris andmetes on sääraseks võtmeks harilikult olemi identifikaator operatsioonilises süsteemis.

Vaatleme Tabelis 4 sõltuvust A→B. On lihtne märgata, et kuni reani seitse määrab A→B. Seejärel sõltuvuses toimub muutus, mis kestab 2 rida, seejärel toimub uus muutus. Sõltuvus A→C on muutumatu tervel sisendandmete hulgal.

Süstemaatiline lähenemine on siis tegelikult lihtne. Alustame esimesest võtmest sõltuvast atribuudist. Liigume läbi ajaliselt järjestatud andmete, võrreldes rea kaupa, kas sama võtme korral ühtib sõltuva atribuudi väärtus varem teadaolnud väärtusega. Kui leiame, et samale võtmele vastab mingil real uus väärtus, on toimunud atribuudi muutus. Seni edukalt läbivaadatud kirjete arv on võrdne antud funktsionaalse sõltuvuse kehtivusperioodiga.

Liigume edasi ridade kaupa, salvestades pidevalt leitud üksikud kehtivusperioodid. Kui sisendandmed on kõik läbivaadatud, võime leida pikima ja keskmise kehtivusperioodi sõltuva atribuudi jaoks. Sääraselt loeme kokku kõik üksikud seosed ja saame teada nende sõltuvuste keskmised kehtivusperioodid.

Kui eeldada lisaks, et me võtmeatribuuti ei tea, tuleb võrrelda kõikvõimalikke atribuutide kombinatsioone omavahel. Võrdluste arv sõltub siis väljade arvust k ja sisendi ridade arvust n. Kõikide kombinatsioonide läbivaatamise ajaline keerukus on $O(n \cdot 2^k)$. Esimeses lahenduses pole seega kindlasti tegu väga optimaalse algoritmiga, kuid samas on kogu

lähenemine on kergesti automatiseeritav.

Läbivaadatavate atribuudipaaride hulka saab vähendada, kasutades funktsionaalsete sõltuvuste algoritme võimalike võtmeatribuutide ning nende suhtes väga aeglaselt muutuvate atribuutide määramiseks.

6.4. Stabiilsuse järgi grupeerimine ja tulemuste võrdlemine

Kui atribuutide stabiilsus on teada, saame sarnase stabiilsusega atribuudid eraldada tabelitesse ja tabelid omavahel välisvõtmetega siduda nagu eelnevalt kirjeldatud. Tulemuste võrdlemiseks tuleks koostada kasvuprognoosid algsele jagamata tabelile ning uuele skeemile, et hinnata eeldatavat andmemahu säästu.

Võime võrdlustes ignoreerida uute dimensiooniolemite lisandumisi, sest need mõjutavad mõlemat skeemi ühesuguselt. Samuti võime võrrelda ainult ridade arvu ja ridade suurusi, mitte konkreetseid andmeblokkide salvestusmahte kettal. Andmeblokkide võrdlus annaks samad tulemused, lisaks sõltub see konkreetsest andmebaasimootorist.

Eeldame lihtsuse mõttes, et dimensiooni atribuudid muutuvad teineteisest eraldi, ehk atribuudid on teineteisest sõltumatud. See tähendab, et iga atribuudi iga muutuse kohta tekib uus rida. Järelikult on dimensioonitabeli kasvuprognoos perioodi kohta lihtsalt summa iga atribuudi perioodis ennustavatest muutustest korrutatuna dimensiooni olemite arvuga. Ei tohi unustada, et kui tabelis on välisvõti, siis on ka see muutuv atribuut, kusjuures tema muutumiskiirus on võrdne välimisse tabelisse lisanduvate ridade arvuga.

Olgu A_1 kuni A_n teineteisest sõltumatud atribuudid tabelis T, muutused(A_i) atribuudi A_i perioodis ennustavate muutuste arv, N dimensiooni olemite arv ning T.FK välisvõtmega FK viidatud teine tabel. Dimensiooni tabeli T ridade arvu kasv avaldub siis valemiga:

$$\text{kasv}(T) = N * \text{muutused}(A_1) + .. \ N * \text{muutused}(A_n) + \text{kasv}(T.FK) \qquad (1)$$

Näide1. Vaatleme seda näite varal. Olgu algses dimensioonis atribuudid A, B, C. Atribuut A on muutumatu, B ja C on muutuvad atribuudid, muutumiskiirustega vastavalt 5 ja 10 korda aastas (A suhtes). Dimensiooni olemeid on 100.

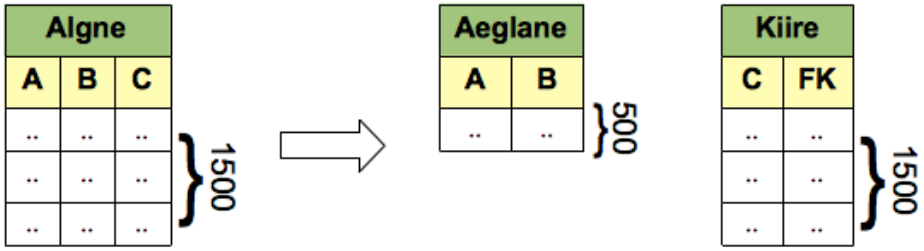
Jagame atribuudid kahte tabelisse: Aeglane {A, B} ja Kiire {C} ning arvutame lisanduvate ridade arvud aasta lõpuks algskeemi ja jagatud skeemi jaoks.

kasv({A, B, C}) = 100x0 + 100x5 + 10x100 = 1500.

kasv({A, B}) = 100x0 + 100x5 = 500

kasv({C, FK}) = 100x10 + kasv({A,B}) = 1500

Kahe uue tabeli peale kokku on ridade arv andmebaasis tegelikult kasvanud. Kuid võimalik andmemahu sääst tuleb tabelite laiusest ehk veergude arvust, andmetüüpidest ja ühe väärtuse keskmisest salvestusmahust. Säästu suurus sõltub ainult aeglases tabelis olevatest atribuutidest. Kiires tabelis olevate atribuutide salvestusmaht ei ole oluline, sest mõlemis skeemis on neid atribuute sisaldavaid ridu samapalju. Vt. Joonis 6.



Joonis 6. Tabelite kasvud algskeemis ja jagatud skeemis (Näide 1).

Antud näites oleme saavutanud olukorra, kus A ja B salvestusmaht on vähenenud kaks kolmandikku. Täpse säästu arvutamiseks tuleb sellest maha lahutada kiiresse tabelisse lisandunud välisvõtmele kuluv andmemaht (1500-l real) ning teha primaarvõtmete võrdlused uues ja algskeemis. Kuid, meenutame, et võtmed on harilikult täisarvtüüpi atribuudid, mille andmetüüp sõltub eeldatavast tabeli ridade arvust, igal juhul pole tegu väga suurt salvestusruumi vajavate atribuutidega.

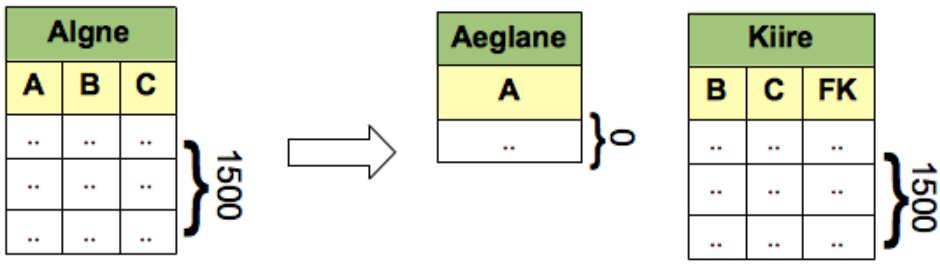
Näide 2. Uurime, kuidas muutuks kasvuprognoos, kui ka atribuut B eraldada kiiresse tabelisse. Teeme analoogsed arvutused:

kasv({A}) = 0

kasv({B, C, FK}) = 100x5 + 100x10 = 1500

Koguridade kasv on algskeemis ja jagatud skeemis on sama. Atribuudi A tõttu lisanduv salvestusmaht on vähenenud nullini, B ja C kasvud on muutumatud, lisandub kiires tabelis välisvõti 1500-l real. Algskeemiga võrreldes sõltub oletatav võti siis ainult sellest, kas

võidetud 1500xA on väiksem või suurem kui 1500x välisvõti. Vt Joonis 7.



Joonis 7. Tabelite kasvud algskeemis ja jagatud skeemis (Näide 2).

Minidimensiooni tehnikaga saavutatud skeemidega võrdluseks on vaja juba rohkem taustainfot. Täpseteks arvutusteks on vaja teada minidimensioonidesse viidud atribuutide kardinaalsust ning faktitabelisse perioodis lisanduvate ridade arvu. Meenutame, et minidimensiooni ridade arv sõltub sealsete atribuutide omavahelistest kombinatsioonidest ja faktitabelisse lisatavaid välisvõtmeid salvestatakse juba märkimisväärselt rohkematel ridadel.

Kokkuvõte

Käesoleva töö eesmärgiks oli uurida võimalusi muutuvate dimensioonide andmete hoidmiseks andmelaos ja leida sobiv lahendus mõõduka kiirusega muutuva dimensiooni jaoks. Isegi kui klassikalised lähenemised muutuvate dimensioonide probleemile on hästi dokumenteeritud ja praktikas ammu edukalt rakendatud, esineb olukordi, kus üldtuntud tehnikad ei ole sobivad ja vajavad täiendusi. Üheks sääraseks olukorraks on mõõduka kiirusega muutuvad dimensioonid. Vastavalt seatud eesmärgile kajastati olulisemaid lähenemisi ning selgitati nende põhilisi omadusi ja kasutusvõimalusi.

Jõuti järeldusele, et olemasolevatest lahendustest ei rahulda ükski piisavalt hästi kõiki konkreetse projekti ärinõudeid. Seepärast tuli kasutada modifikatsiooni olemasolevatest tehnikatest. Töös on toodud kasutatud meetodi põhjalik kirjeldus ning analüüsitud tema eeliseid ja puudusi võrreldes traditsiooniliste meetoditega.

Töö käigus selgus, et probleemi lahendus on peaaegu alati seotud dimensiooni lõhkumisega, eraldades erineva stabiilsusega atribuudid teineteisest. Lõhkumisega kaasnevad aga teatud riskid. Seega tuleb jagamist teha kaalutletult ja ettevaatlikult.

Kaalutletud otsuste vastuvõtmiseks on vaja omada põhjalikke teadmisi ärivaldkonnast ning andmelao sisendandmetest. Teine võimalus on vajalik info sisendandmetest kaevandada. Töös pakuti lahendusi atribuutide tähenduse ja omavaheliste seoste kaevandamiseks sisendandmetest ning kirjeldati viisi atribuutide muutumiskiiruste arvutamiseks. Lisaks demonstreeriti näidete varal kuidas erinevad dimensiooni lõhkumised andmelao tabelite suurusi mõjutavad.

Muutumiskiiruse arvutamine olemasolevatest andmetest ei ole teatud piiratud tingimustes keeruline ülesanne, kuid suurte andmemahtude ja paljude atribuutide korral on see andmelao projekteerija jaoks siiski ajamahukas ettevõtmine. Samuti tuleb projekteerijal läbi mängida erinevaid dimensioonide jaotamise variante, et võrrelda salvestusmahu prognoose. Teema edasiarenduseks võiks olla reaalne töövahend, mis kirjeldatud sammud automatiseerib, muutes seeläbi projekteerimisprotsessi efektiivsemaks.

Managing rapidly changing dimensions in data warehouses

Bachelor thesis

Juta Vaks

Summary

Traditional techniques for managing changing dimensions in data warehouses are well documented, but those techniques become unsuitable if dimension change speeds up. The aim of this thesis is to study the existing solutions and find the best technique for managing large dimension with unpredictable changes.

The most known technique for managing rapidly changing dimensions is based on using minidimensions. Author investigates the minidimension technique and compares it to the idea of snowflaking the database schema via stability analysis. Comparison is done in a real world data warehouse context, where a snowflaked schema was used.

It is shown that a data warehouse is very sensitive to the rate of change of data within the warehouse. The optimal organization of data is where the data in one table changes with similar speed. In order to model the data warehouse in this way, it is necessary to have information about the data beforehand, in form of domain knowledge or documentation. Another option is to extract necessary information from the data by using data mining techniques. An overview of suitable data mining techniques is presented, together with a simple algorithm for stability analysis and estimating storage space growth.

.

Kirjandus

[1] A. Alashqur. *Expressing Database Functional Dependencies in Terms of Association Rules.* European Journal of Scientific Research, 32(2), lk. 260-267, 2009.

[2] J. Atoum, D. Bader, A. Awajan. *Mining Functional Dependency from Relational Databases Using Equivalent Classes and Minimal Cover.* Journal of Computer Science, 4(6), lk. 421-426, 2008.

[3] O. Boussaid, A. Tanasescu, F. Bentayeb, J. Darmont. *Integration and dimensional modeling approaches for complex data warehousing.* Journal of Global Optimization, 37(4), lk. 571-591, 2007.

[4] W. Fan, F. Geerts, L. V. Lakshmanan, M. Xiong. *Discovering Conditional Functional Dependencies*. Proceedings of the 2009 IEEE international Conference on Data Engineering, Washington, DC. lk. 1231-1234, 2009

[5] Y. Huhtala, J. Kärkkainen, P. Porkka, and H. Toivonen. *TANE: An efficient algorithm for discovering functional and approximate dependencies.* The Computer Journal, 42(2), lk. 100–111, 1999.

[6] I. F. Ilyas et al. *CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies.* Proceedings of the 2004 ACM SIGMOD international conference on Management of data, lk. 647-658, 2004.

[7] W. H. Inmon. *Building the Data Warehouse*, third ed., John Wiley & Sons Inc, 2002.

[8] M. R. Jensen, T. Holmgren, T. B. Pedersen . *Discovering Multidimensional Structure in Relational Data.* Data Warehousing and Knowledge Discovery (3181), 2004.

[9] R. Kimball, J. Caserta, *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data*, Wiley Publishing Inc, 2004.

[10] R. Kimball, M. Ross. *The Data Warehouse Toolkit: the Complete Guide to Dimensional Modeling*, second ed., John Wiley & Sons Inc, 2002.

[11] R. Kimball. *Realtime partitions*, Intelligent Enterprise Magazine 5 (3), 2002.

[12] R. Kimball. *Tricky time spans*, Intelligent Enterprise Magazine 5 (10), 2002.

[13] R. Kimball. *When A Slowly Changing Dimension Speeds Up*, Intelligent Enterprise Magazine 2 (11), 1999.

[14] S. Rizzi, A. Abello, J. Lechtenborger, J. Trujillo. *Research in data warehouse modeling and design: dead or alive?* Proceedings of the 9th ACM international workshop on Data warehousing and OLAP, lk. 3-10, 2006.

[15] O. Romero, D. Calvanese, A. Abello, M. Rodriguez-Muro. *Discovering Functional Dependencies for Multidimensional Design.* Proceedings of the ACM 12th international workshop on Data warehousing and OLAP, 2009.

[16] C. Sapia, G. Höfling, M. Müller, C. Hausdorf, H. Stoyan, U. Grimmer. *On Supporting the Data Warehouse Design by Data Mining Techniques.* GI-Workshop: Data Mining und Data Warehousing, September, Magdeburg, 1999.

[17] B. Vrdoljak, M. Banek, and S. Rizzi. *Designing web warehouses from XML schemas*, Proceedings of the 5th international Data Warehousing and Knowledge Discovery Conference, lk. 89–98, 2003.