TALLINNA ÜLIKOOL
Informaatika Instituut

Rivo Lemmik

OLAP KUUBIL RAJANEV INFOSÜSTEEMI ARENDUSMEETOD

Magistritöö

Juhendaja: Priit Parmakson

Autor : ……..”…….” ……. 2008.a.

Juhendaja : ….……..”…….” ……. 2008.a.

Instituudi direktor : ……..”…….” ……. 2008.a.

Tallinn 2008

SISUKORD

SISSEJUHATUS

Töö keskendub laiemalt ettevõtte infosüsteemide, kitsamalt ERP (Enterprise Resource Planning) majandusinfosüsteemi arendamisele ja juurutamisele metoodiliselt ning modelleeritult, kus primaarseks lähtepunktiks on võetud tegevusepõhine vaade lähtuvalt ärianalüüsi vajadustest. Põhilisteks märksõnadeks selles vaates on informatsioon, äriprotsess, ärijuhtimine ja majandusanalüüs, mis on tarkvara seisukohalt põhilised tunnused, mis eristavad ERP majandustarkvara tavalisest FIS (Financial Information System) raamatupidamisprogrammist.

Uurimishüpotees

Lähtudes tegevusepõhisest OLAP (On-Line Analytical Processing) vaatest ettevõtte infosüsteemi juurutus- ja arendustööde teostamisel, saavutab ettevõte majandustarkvarasse tehtud investeeringu maksimaalse väärtuse ja äriprotsessi maksimaalselt efektiivse toimivuse. Selliselt arendatud ja juurutatuid ettevõtte infosüsteemi saab tervikuna nimetada majandustarkvara lahenduseks, mis igapäevaselt toetab ettevõtte strateegilist juhtimist ja võimaldab teha kiiresti õigeid otsuseid äriprotsessi juhtimise erinevatel tasanditel.

Ühtlasi arendan edasi oma bakalaureusetöös tõestatud hüpoteesi: *„Eeldusel, et ettevõttes on kirjeldatud kõik protsessid ja juurutatud kvaliteedijuhtimissüsteem, on mõistlik majandustarkvaras realiseeritud protsessid läbi tarkvara kohaldamise viia vastavusse ettevõtte protsessidega.”*, leides sobiva arendusmetoodika, et süsteemi kasutegur tervikuna suureneks. Väidan, et ettevõtte infovarade planeerimisel ning edasi majandusinfosüsteemi ülesehitamisel on kriitilise tähtsusega arendus- ja juurutusmetoodika valik, et ära kasutada süsteemi poolt tervikuna pakutav maksimaalne väärtus.

Võtan eelduseks hüpoteesi, et ettevõtte infosüsteemi arendamisel on võimalik saavutada madalam TCO (Total Cost of Ownership), kui äriprotsessi mudelist luua OLAP (On-Line Analytical Processing) mudel paralleelselt ja seotult OLTP (On-Line Transaction Processing) mudeliga. Selliselt loodud süsteemi juures on võimalik rakendada virtuaalset andmeladu ja

automaatselt genereerida staatilise andmelao ETL (Extract, Transform, Load) äriloogika koos OLAP dimensioonilise andmebaasiga.

Uurimismetoodika

Baasina kasutan oma bakalaureusetöö, teemal: *„Ettevõtte infosüsteemi loomine majandustarkvara Microsoft Navision Attain baasil"*, raames arendatud ja juurutatud ettevõtte majandusinfosüsteemi.

Magistritöös võtan aluseks OLAP kuubil põhinevad andmeanalüüsi teostamise metoodikad, koos majandustarkvara juurutusmetoodikaga *Microsoft Dynamics Sure Step* , kõrvutades neid seni teostatud arendus- ja juurutustöödega, mille tulemusena leian olemasoleva süsteemi kitsaskohad, mis on takistuseks hästi toimiva andmelao ja ärianalüüsi lahenduse loomisel.

Töö eesmärk

Leian antud ettevõtte majandusinfosüsteemi kontekstis sobivaima andmelao ja ärianalüüsi lahenduse, kombineerituna arendus- ja juurutusmetoodikaga, mis toetab süsteemi tervikuna elujõulist arengut ajas ja tagab süsteemi jätkusuutlikkuse. Selle põhjal teostan antud ettevõttes virtuaalse andmelao realisatsiooni.

Töö tulemusena valmib kohaldatud infosüsteemi arendusmeetod, mis keskendub ettevõtte infosüsteemide arendamisele ja juurutamisele, arvestades jooksvalt andmelao ja ärianalüüsi vajadustega, tagades seeläbi süsteemi tervikuna efektiivse toimivuse. Töö tulemus on formuleeritud universaalselt rakendatava ja uudse tehnoloogilise meetodina, virtuaalse andmelao rakenduse realiseerimiseks.

Taust

Käsitletavas ettevõttes tekkis vajadus infosüsteemi vahetamise järele aastal 2001. Olemasoleva süsteemina oli kasutusel raamatupidamisprogramm RV-Soft ja aastatel 1999 kuni 2001 arendatud infotarkvara Favourite. Ettevõtte põhitegevuseks on jae- ja hulgimüük, tootmine ning teenindus, aastal 2001 töötas ettevõttes ~70 töötajat ning tegutseti neljas linnas. Ettevõtte arenedes muutusid tootmis- ja teenindusprotsessid järjest mahukamaks ning 1999 aastal alustati ise enda tarvis infotarkvara arendamist, mis kataks põhilised tootmis- ja teenindustoimingud ning oleks elementaarses osas integreeritud raamatupidamistarkvaraga.

Ettevõtte kasvades aga tekkis järjest süvenev vajadus infosüsteemi horisontaaltasandi ehk raamatupidamistarkvara väljavahetamise järele.

Samal ajal, aastatel 2001 kuni 2002 toimus ettevõttes kvaliteedijuhtimissüsteemi ISO 9001:2000 juurutamine ja sertifitseerimine, sellest tulenevalt otsustati infosüsteemi vahetamisega tegelema hakata peale kvaliteedijuhtimissüsteemi juurutamist kuna selle tegevuse käigus muutusid oluliselt mitmed ettevõtte tööprotsessid. Aasta 2002 lõpus moodustati ettevõttes IT osakond, põhiülesandega viia läbi infosüsteemi vahetamise protsess.

Majandustarkvara kohaldamine

Majandustarkvarale esitatavate nõudmiste loetelu baseerus ettevõtte igapäevases põhitegevuses toimuvatele protsessidele ehk lähtus ainult OLTP (On-Line Transaction Processing) süsteemi andmesisestamise tasandist, vaatluse alt jäid täielikult välja nõudmised analüüsi osas ehk OLAP (On-Line Analytical Processing) vaade.

Välja valitud majandustarkvara Microsoft Navision Attain (tänapäeval nimega: Microsoft Dynamics NAV) on väga hästi kasutaja vajadustele kohaldatav kuna kogu tarkvara funktsionaalsus, sealhulgas tabelite struktuur, äriloogika, ekraanivormid jne. on kõik täies ulatuses algkoodi tasandil muudetavad, kasutades spetsiaalset arenduskeskkonda C/SIDE (Client/Server Integrated Development Environment). Sellest tulenevalt otsustati mitte otsida kõrvalteid vaid arendada või muuta majandustarkvara toimivusloogikat nii, et see kataks kõik ettevõtte protsessidest tulenevad vajadused

Vertikaallahenduse arendamine

Tootmis- ja teenindustarkvara on ettevõttes arendatud ise aastast 1999, mille esimene versioon oli integreeritud raamatupidamistarkvaraga RV-Soft. Tegemist on läbiproovitud ning toimiva lahendusega, mida on võimalik väga operatiivselt kohaldada tootmis- ja teenindusprotsessi muutustele, seega otsustati samal põhimõttel jätkata ning arendada infotarkvarast Favourite uus versioon, mis oleks majandustarkvaraga Microsoft Navision Attain täielikult reaalajas integreeritud vertikaallahendus Favourite II.

1. MAJANDUSTARKVARA JUURUTUSMETOODIKA

Majandustarkvara juurutades ja spetsiaaltarkvara arendades on esimesel ja kõige olulisemal kohal valitud metoodika, millest oleneb väga suurem määral saavutatava tulemuse kvaliteet ning TCO (Total Cost of Ownership). Tarkvaraarenduse valdkonnas on üha rohkem hakatud panustama metoodilisele ning modelleeritud arendusprotsessi kujutamisele, juhtivate terminitena võib siinkohal nimetada UML (Unified Modeling Language) ja RUP (Rational Unified Process), mis oma olemuselt on tihedalt seotud tehisintellekti valdkonnaga teadus-arenduses.

Ideaalis peaks modelleerimine ja metamodelleerimine välja jõudma maailma (eksistentsi) mudelini, mis oleks täiuslik „Tehisintellekt". Reaalses elus aga on suudetud arusaadavalt (meta)-modelleerida eksistentsist oluliselt primitiivsem tasand, mida on raske nimetada tehisintellektiks, küll aga on sellest väga palju kasu ettevõtete protsesse peegeldava spetsiaaltarkvara arendamisel. Oluline on rõhk sõnal „arusaadavalt", kuna indiviidi tasandilt vaadatuna on suudetul luua oluliselt mahukamaid ja sügavamaid mudeleid kuid probleemiks on sedasorti metatasandite abstraktsioonide kujutelma edastamine maailmale, kuna inimeste suhtlusvahenditel puudub suuresti selline metatasand, mis võimaldaks nimetatud mõtteid teiste inimeste jaoks arusaadavalt formuleerida.

Majandustarkvara juurutusmetoodikad on spetsiaaltarkvara arendusmetoodikatega võrreldes väiksema mahuga ja veidi erineva eesmärgiga. Majandustarkvara puhul on tegemist tavaliselt väga mahuka ja võimalusterohke valmistootega, mida aga enamasti ei ole võimalik kasutusele võtta „karbitootena". Juurutusmetoodika peaks hõlmama kõiki tegevusi, mida tehakse alates karbi ostmisest kuni süsteemi elutsükli lõpuni, näiteks:

- Tarkvara üldine seadistamine: Seadusandlusest ja raamatupidamise sisekorra eeskirjadest tulenevad finantsarvestuse põhimõtted
- Püsiandmete importimine: Kliendid, kaubad jne.
- Tarkvara kohaldamine: Ekraanivormide, dokumendivormide ja aruannete kujundamine vastavalt vajadusele, arvestades kasutamismugavust

- Tarkvara andmestruktuuri kohaldamine: Tabelitesse väljade lisamine, et saaks sisestada tehinguga seotud vajalikku lisainfot või tehinguid spetsiifiliselt klassifitseerida
- Tarkvara äriloogika kohaldamine: Tabelitesse lisatud väljadele andmetöötluse reeglite kehtestamine
- Tarkvara äriloogika arendamine: Olemasolevate, tarkvaras standardsete andmetöötluse reeglite muutmine

Tavaliselt jääb enamustes majandustarkvara juurutustes, kui välja arvata vertikaallahendused, selliste juurutustööde maht alla 1% kogu tarkvara mahust küll aga on absoluutarvudes väljendades piisavalt mahukas, et tegelikult karbitootel baseeruvat süsteemi saab nimetada ettevõtte-spetsiifiliseks majandustarkvara lahenduseks. Selline lähenemine on hädavajalik kuna vastasel juhul ettevõte ei saa või ei taha kasutada lahendust, mis ei haaku kokku ettevõtte spetsiifiliste nüanssidega. Reeglina ettevõtted, kus on protsessid juba välja töötatud ja selgelt formuleeritud ning põhjendatud, ei soovi hakata oma äriprotsesse muutma, tarkvara poolt pakutava funktsionaalsuse järgi.

Probleemiks on asjaolu, et sellise juurutustegevusega on ära kaotatud infosüsteemi dünaamika, mida pakuvad olemaslikult karbitooted ehk ettevõte ei saa lihtsalt vahetada toote versiooni või otsustada konkureeriva toote kasuks. Loodud on staatiline süsteem, millega ollakse seotud järgmised 6 kuni 8 aastat, mis on tavapärane majandustarkvara lahenduste amortiseerumise aeg. Arvestades tehnoloogiate arengu kiirust ja sellest tulenevaid üha uusi võimalusi on see kindlasti liialt pikk aeg, mil ettevõte võib kaotada oma kasumit, kui konkurendid samal ajal võtavad kasutusele uusi tehnoloogiaid.

Selline lähenemine tarkvara arendusele pärineb 1980-ndatest aastatest, kus maailmas domineerisid paindumatud ja suured tootmisettevõtted ja äris primaarsel kohal oli mõiste „Toode". Tänapäevaks on olukord muutunud selliselt, et primaarne mõiste äris on „Teenus", milles sisaldub ühe osana toode ja lisaks hulgaliselt pakutavat lisaväärtust. Teenuse pakkumine aga eeldab dünaamikat äriprotsessides ja seega ka infosüsteemis ning suurema konkurentsieelisega on ettevõtted, kes suudavad kõige kiiremini ja efektiivsemalt pakkuda sellist teenust, mille järgi on parasjagu nõudlus.

Selle probleemi lahendamisel tulevad appi majandustarkvara juurutusmetoodikad, mis ideaalis peaks toimima nii, et kõik eelpool kirjeldatud juurutusprotsessi tegevused teostatakse metoodikas kirjeldatud metatasandi struktuuriga juurutusmudeli loomisel. Selliselt loodud juurutusmudel on sõltumatu majandustarkvara versioonist ja ehk isegi tootest ja konkreetse tarkvara kasutusele võtmisel juurutusmudel lihtsalt rakendatakse majandustarkvaras. Sellist lahendust saaks nimetada ettevõtte-spetsiifiliseks standardlahenduseks kuna juurutustööd on teostatud tarkvarast üks abstraktsioon kõrgemal, mis on selgelt eristuv ning korduvkasutatav.

Reaalselt olemas olevad vahendid on primitiivsemad ning samuti puudub ühtne juurutusmetoodika standard, mida aktsepteeriksid konkureerivad majandustarkvara tootjad. Probleemiks on asjaolu, et standardiseeritud metoodika kasutamine kaotaks majandustarkvara tootja jõupositsiooni oma klientide kinni hoidmisel. Viimasel ajal on väga tõsiselt oma juurutusmetoodika arendamise ette võtnud ka Microsoft, kus 2007 aasta keskel anti välja metoodika „Microsoft Dynamics Sure Step", mis on teistest analoogsetest sammu võrra ees selles osas, et katab standardse metoodikana kogu Microsoft Dynamics ERP tooteperekonna, kuhu kuulub 4 erinevat majandustarkvara toodet.

1.1. Microsoft Dynamics Sure Step

Metoodika kirjeldamisel on kasutatud juhendmaterjali [2].

Microsoft Dynamics Sure Step on laiahaardeline juurutusmetoodika, mis pakub korralduslikku juhendust, projektijuhtimise strateegiat, tööriistu ja malle, et juurutada Microsoft Dynamics majandustarkvara tooteid. Metoodika pakub detailset juhendust kogu juurutamise elutsüklile, sisaldades Microsoft'i juurutuspartnerite poolt läbiproovitud parimaid praktikaid. Microsoft Dynamics Sure Step metoodika diagramm on toodud lisas 2.

Metoodika on olemuselt HTML tööriist, mida saab kasutada veebilehitsejaga. Metoodika on sobilik kõikide Microsoft ERP perekonna toodete juurutamiseks:

- Microsoft Dynamics™ CRM
- Microsoft Dynamics™ AX
- Microsoft Dynamics™ GP
- Microsoft Dynamics™ NAV
- Microsoft Dynamics™ SL

Metoodika käsitleb kõiki juurutusprojekti etappe alates planeerimisest ja analüüsist, läbides kavandamise, arendamise ja käivitamise tegevused. Samuti annab metoodika juhendust olemasoleva süsteemi optimeerimiseks ning uuendamiseks.

Metoodika on skaleeritav, võimaldades metoodikat kasutada nii väikeste, keskmise suurusega kui ka suurte korporatsioonide projektide puhul. Konfigureeritavus võimaldab seda kasutada kõigi Microsoft ERP tooteperekonna toodete jaoks. Samuti saab metoodikat kasutada nii kiire juurutuse kui täismahuliste juurutusprojektide korral.

Metoodika on süstemaatiline, järjepidev ja korduvkasutatav erinevate projektide puhul. Mudel on jagatud faasideks ja iga faas tegevusteks ning ülesanneteks. Enamikel juhtudel on tegevuse väljundiks selgepiiriline dokumentatsioon, mis on kasutatav järgmises faasis või on ise järgmiseks faasiks. Ühe faasi väljund on alati sisendiks järgnevale faasile. Tegevused ja ülesanded on paigutatud hierarhilisse struktuuri, kus faaside esimene alamtasand on tegevused ja tegevuste alamtasand on ülesanded. Mõned ülesanded jagunevad alamülesanneteks, mis on hierarhia madalaim tasand.

1.1.1. Juurutuse faasid

Metoodika koosneb kuuest põhifaasist ja kahest lisafaasist, mis hõlmavad optimeerimist ja (versiooni) -uuendamist. Põhifaasid on järgnevad:

- Planeerimine
- Analüüs
- Kavandamine
- Arendamine
- Prototüüpimine
- Rakendamine

Planeerimise faas – on üleminekufaas müügietapist juurutamisele. Põhiliselt sisaldab see faas protsessida ja infrastruktuuri üldist planeerimist ja analüüsi, vahest käsitletakse seda faasi ka kui eelanalüüsi. Selle faasi eelmärk on hankida üldist informatsiooni, et defineerida üldine projekti skoop ja koostada projekti pakkumine või eelarve.

Planeerimise faasil on järgmised väljundid:

- Äriprotsessi kirjeldus ja kattuvus tarkvara funktsionaalsusega
- Infrastruktuuri määratlus
- Tööde kokkulepe
- Pakkumine või eelarve

Analüüsi faas – osaliselt kattub planeerimise faasiga, käsitledes äriprotsessi detailselt. Dokumenteeritakse äriprotsess ja kirjeldatakse vajalikud muudatused, et tarkvara kataks äriprotsessi. Siin faasis kaasatakse aktiivselt äri-poole võtmeisikud, viiakse läbi tarkvara tutvustus ja võtmekasutajate koolitused. Koolituse käigus selgitatakse välja olemasolevatest süsteemidest migreerimist vajavad andmed, nende maht ja struktuur. Võttes aluseks saadud infot koostatakse projekti plaan.

Analüüsi faasi põhilised väljundid on:

- Projekti- ja kasutajakoolituste plaan
- Detailne äriprotsesside kirjeldus
- Andmete migratsiooni plaan
- Nõudmised tarkvara funktsionaalsusele

Kavandamise faas – võtab aluseks eelneva ja põhirõhk on süsteemi muudatuste spetsifikatsiooni loomisel, hõlmates tarkvara kohaldusi, integratsioone ja andmete migratsiooni. Kavandamisel lähtutakse nõudmistest, mis on esitatud tarkvara funktsionaalsusele. Andmete migratsiooni spetsifikatsioonis luuakse seosed olemasolevate andmeallikate väljade ja juurutatava tarkvara andmeväljade vahel, kui on vajalik, kirjeldatakse detailselt andmete konverteerimise loogika. Lisaks luuakse detailne ja tehniline arenduse spetsifikatsioon, mis kirjeldab tarkvara kasutajaliidese, äriloogika ja andmestruktuuri vajalikke muudatusi.

Kavandamise faasil on järgmised väljundid:

- Üldine spetsifikatsioon äriprotsessi baasil
- Muudatuste tehniline spetsifikatsioon
- Andmete migratsiooni tehniline ja loogiline spetsifikatsioon
- Testimisplaan ja testilood

Arendamise faas – lähtub eelnevast ja peamiseks ülesandeks on siinkohal tarkvara kohalduste, integratsioonide ja andmete migratsiooni programmeerimine. Peale programmeerimist arendused testitakse funktsionaalsete nõudmiste osas ja lisaks testilugude põhjal. Programmeerimise eelduseks võetakse keskkonna nõudmised rakendustarkvara, riistvara ja infrastruktuuri osas. Selle faasi lõpus kaasatakse taas aktiivselt võtmekasutajad ja hinnatakse arenduste vastavust ning vajadusel läbitakse arenduse faasi korduv iteratsioon.

Arenduse faasil on järgmised väljundid:

- Andmete migratsiooni funktsionaalsus kodeerituna ja testituna
- Tarkvara muudatused ja täiendused kodeerituna ja testituna
- Testitulemuste raport
- Tehniline ja lõppkasutaja dokumentatsioon

Prototüüpimise faas – võtab kokku eelneva ja peamiseks ülesandeks on süsteemi kasutuskõlbuliku prototüübi loomine. Majandustarkvara standardfunktsionaalsus seadistatakse lõplikult kas käsitsi või kasutades Sure Step RIM metoodikat. Kasutades eelnevalt loodud andmete migratsiooni funktsionaalsust, imporditakse kõik vajalikud andmed. Luuakse kaks süsteemi koopiat, üks testimiseks, teine valmis rakendamiseks. Süsteem testitakse lõplikult, kaasates võtmekasutajaid, mittevastavuste ilmnemisel läbitakse arenduse ja prototüüpimise faasi korduv iteratsioon. Seejärel viiakse läbi süsteemi kasutajate koolitused.

Prototüüpimise faasil on järgmised väljundid:

- Süsteemi rakendamise plaan
- Testimise plaan
- Süsteemi konfiguratsioon

Rakendamise faas – teostatakse kui eelnevalt loodud süsteem on valmis kasutamiseks. Peamiseks ülesandeks on süsteemi käiku laskmise esmasel perioodil kasutajate toetamine. Seejärel toimub projekti lõpetamine, mille käigus analüüsitakse projekti kulgu võrreldes projekti plaaniga, hinnatakse kliendi rahuolu. Lepitakse kokku täiendava toe või kasutajate koolituse vajadused.

Rakendamise faasil on järgmised väljundid:

- Tehniline dokumentatsioon
- Lõppkasutaja dokumentatsioon
- Funktsioneeriv, kasutuses olev süsteem
- *Go-Live Celebrating!*

1.1.2. RIM (Rapid Implementation Methodology)

Metoodika kirjeldamisel on kasutatud juhendmaterjali [3].

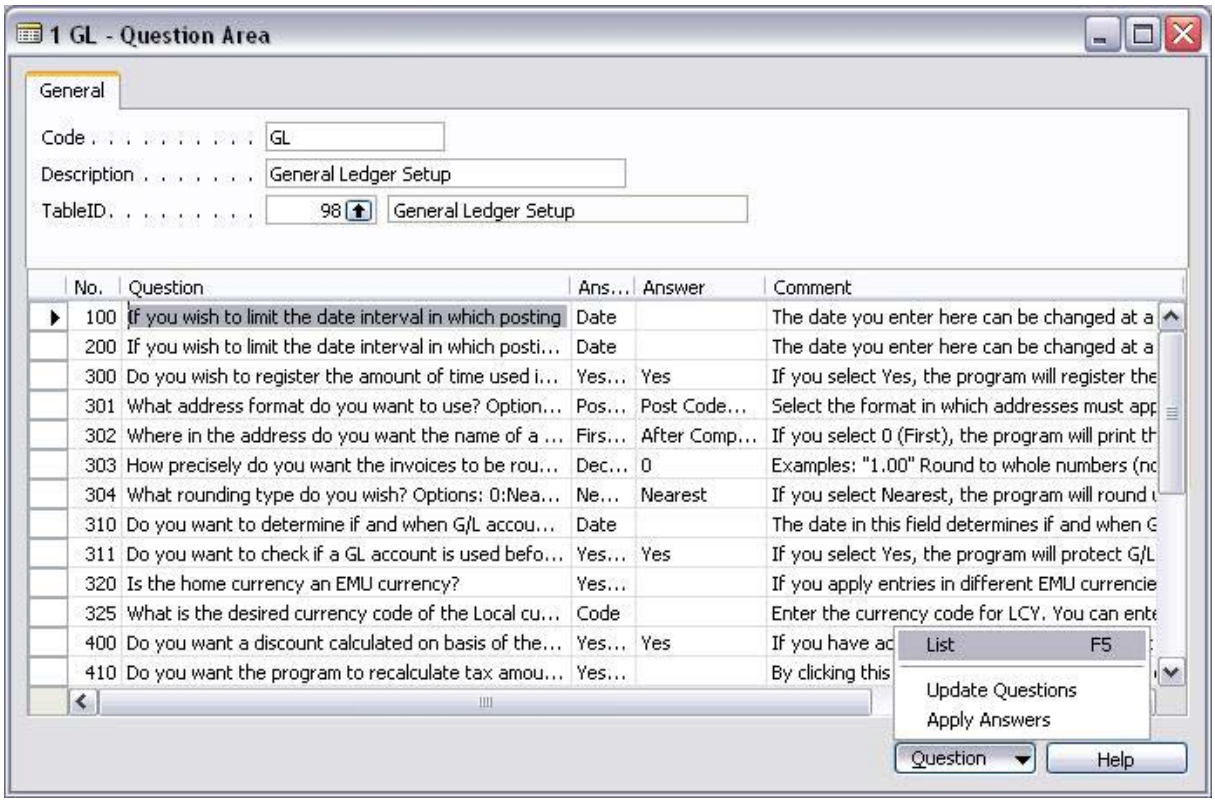
Navisioni kiire juurutamise metoodika tööriistakomplekt (Navision Rapid Implementation Methodology Toolkit) on tööriist, mis on välja töötatud Navisioni uue versiooni installeerimiseks ja juurutamiseks.

Tööriista eesmärgiks on juurutamiseks kuluva aja lühendamine järgmiste võtetega:

- projektianalüüsi malli järgi tegutsemine,
- tegevusvaldkonna kohaste andmekomplektide importimine,
- süsteemi seadistuse määratlemine Excelis ja selle importimine Navisioni,
- kliendiandmete importimine XML-i abil (põhiandmete üleviimine),
- importimise lihtsustamine ja andmete loomine (põhiandmete mallid),
- väiksem vajadus võtmeisikute kaasamiseks ja koolitamiseks esialgse juurutamise faasis.

Projektianalüüsi tarvis sisaldab tööriistakomplekt MS Project rakenduses loodud vaikimisi projekti analüüsi, mis on kasutatav projektianalüüsi mallina.

Küsimustiku põhises seadistuses tuleb anda selgelt formuleeritud küsimustele valikvastused. Küsimustik täidetakse Excelis ja imporditakse XML kujul või täidetakse kasutades joonisel 1 kujutatud küsimustiku kontroll-loendit. Peale küsimustiku importimist või täitmist rakendatakse see kontroll-loendi kaudu, mispeale teostatakse automaatselt süsteemi seadistamine Navisioni seadistuste tabelites.

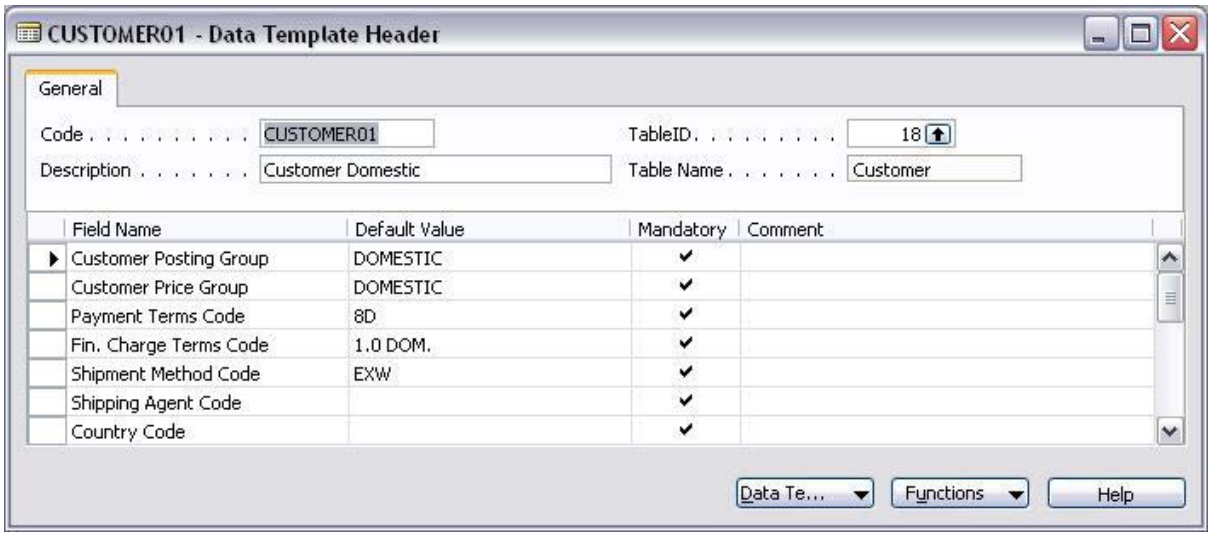


Joonis 1. Küsimustiku kontroll-loend

Allikas: Navision RIM Toolkit [3]

Tegevusvaldkonna kohases seadistuste komplektis sisaldub vastava tegevusvaldkonna jaoks loodud põhiandmete konteiner. Näiteks sisaldades klientide jt. püsiandmete klassifikatsiooni, konteerimise maatrikseid, dokumentide numbriseeriaid jne. Lisas 1 on toodud seadistusandmete tabelite loetelu, mida tegevusvaldkonna kohane seadistuste komplekt hõlmab.

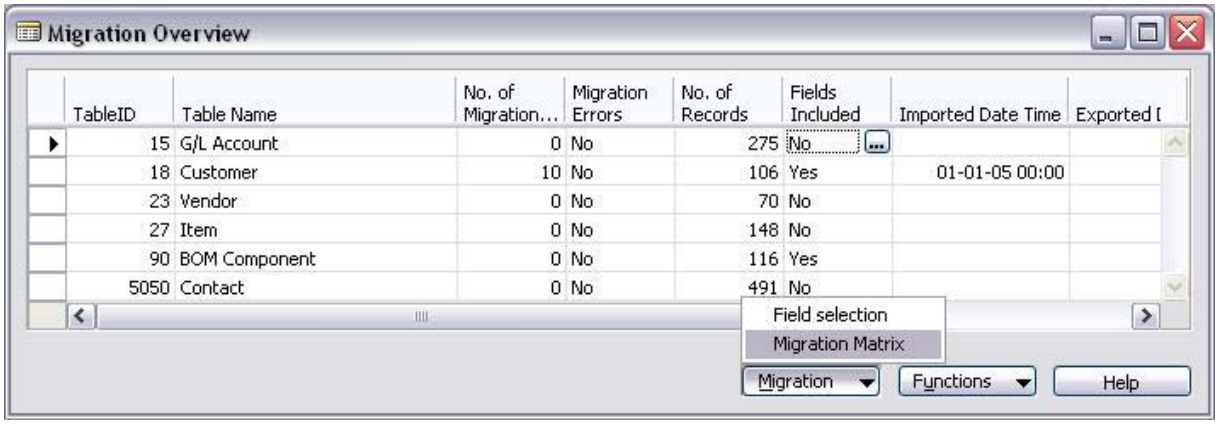
Põhiandmete mallid on loodud PR kontode, klientide, hankijate, kaupade ja kontaktide tabelite jaoks, võimaldades uue kirje loomisel automaatselt vastava malli vaikeväärtustega täita kohustuslikud väljad. Põhiandmete malli kirjeldamise aken on kujutatud joonisel 2, kus saab luua kohustuslike välja määratlusega ja vaikeväärtustega malle piiramatul hulgal kõigi Navisioni tabelite jaoks.



Joonis 2. Põhiandmete mall – kodumaine klient

Allikas: Navision RIM Toolkit [3]

Põhiandmete migratsiooni tarvis on loodud XML andmeliides, mis lihtsustab põhiandmete importimist suvalisest seni kasutusel olnud andmeallikast. Migreeritavate andmete kooslust ja struktuuri saab kerge vaevaga muuta vastavalt konkreetse juurutuse vajadustele, kasutades joonisel 3 kujutatud tööriista akent. Vaikimisi on tegevusvaldkonna põhiselt seadistatud XML andmeliidesed PR kontode, klientide, hankijate, kaupade ja kontaktide tabelite kohta.



Joonis 3. Põhiandmete migratsiooni ülevaade

Allikas: Navision RIM Toolkit [3]

2. VERTIKAALLAHENDUSE ARENDUSMETOODIKA

Vertikaallahendus on ettevõtte mingit tegevust peegeldav infosüsteemi osa, mida võib käsitleda kui terviklikku, eraldiseisvat süsteemi, mis on infosüsteemi horisontaaltasandiga integreeritud kindla sisendi ja väljundi kaudu. Käsitletavas ettevõttes on esimeseks põhitegevuseks müügiprotsess, mida võik käsitleda infosüsteemi horisontaaltasandina ning seda tasandit katab kasutatav majandustarkvara, kus sisendiks on kliendi soov ja väljundiks kliendi rahulolu. Teiseks põhitegevuseks on tootmine, mida saab käsitleda infosüsteemi vertikaaltasandina, kus sisendiks on tootmisvajadus ja väljundiks valmistoode. Selline tasandite jaotus tuleneb rõhuasetusest ettevõtte põhitegevustes ning samahästi võib teises olukorras, näiteks suurte tootmisorganisatsioonide puhul seda vaadet 90 kraadi pöörata, kus ettevõtte põhitegevuseks oleks tootmine ning teiseks põhitegevuseks müük. Kolmas võimalus on mitme põhitegevusega organisatsioon restruktureerida kontserniks, kus igal tütarettevõttel on vaid üks põhitegevus ja seega eksisteerib vaid infosüsteemi horisontaaltasand.

Väikesed ja keskmised tootmisettevõtted on hädas kuna nende ladu on kontrolli alt väljas. Sellist tarkvara mida nad vajaksid ei ole, kuna tavapärases tootmistarkvaras on liiga palju funktsioone mida nad ei vaja. Paljudel juhtudel kasutavad ettevõtted siiamaani ajutiselt loodud süsteeme mis põhinevad Accessi andmebaasidel või Exceli töölehtedel. Seega paljudel juhtudel ei vaja tootmine viimast tehnoloogiat vaid hoopis tootmisprotsessi kontrollimise võimalust ja baasfunktsionaalsust ärijuhtimiseks. Navision Attain'i tugev funktsionaalsus on komplekteerimine aga see on suunatud ühes asukohas tegutsevale, seeriatootmisega tegelevale üksusele [11:38-40].

Vertikaallahendused on vahendid tagamaks edu, tunnistab Navision'i tootejuht Janet Kahr. Nõudmised, mis baseeruvad toiduainetööstuse tootmisprotsessile on totaalselt erinevad elektroonikakomponentide tootja omadest [9:40].

Kenneth Shroeder, KLA-Tencor'i peadirektor ja tema firma töötajad on heidutamatud evangelistid laiendatud protsessi kontroolimise (APC) praktikas Sellegipoolest KLA andis alla ja lõpetas oma tarkvara müümise aastal 2003 kuna suured kliendid ei ostnud seda. Selgus,

et suured tootjad tahavad ise kirjutada oma tootmistarkvara. Probleem väljaspool toodetud lahendustega on selles, et nende lahenduste loojad ei tööta tehases. Tootmistarkvara kirjutamiseks peab teadma kõike mis toimub tehase tootmisprotsessis ja ainus viis seda teada saada, on igapäevaselt töötada selles tootmisettevõttes [10:53-54].

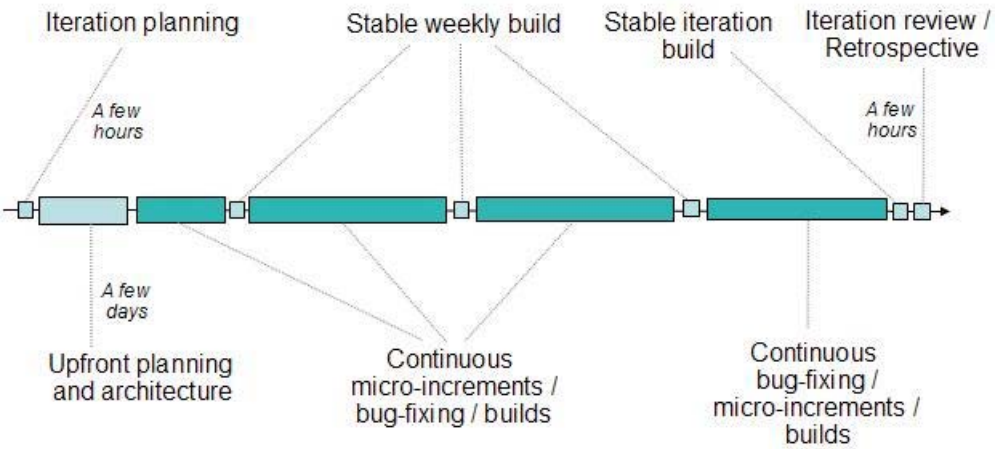
2.1. IS arendusprotsess OpenUP

Metoodika kirjeldamisel on kasutatud juhendmaterjali [4].

OpenUP baseerub neljal üksteist vastastiku toeaval baasprintsiibil

- Balanseerida konkureerivad prioriteedid, saavutamaks maksimaalset väärtust
- Teha koostööd huvide ühtlustamiseks ja arusaama jagamiseks
- Fokusseerida varakult arhitektuur, et maandada riske ja organiseerida arendust
- Tagada püsiv areng, hankides tagasisidet ja teostades parendusi

OpenUP iteratsioon – on fokuseeritud süsteemi poolt pakutava väärtuse järjepidevale kasvatamisele, väljastades iga paari nädala tagant täielikult testitud demonstreeritavaid või kasutatavaid süsteemi osi – toote iteratsioone. Otsuste tegemine peab toimuma seega operatiivselt kuna ei ole aega lõpututeks vaidlusteks. Iteratsiooniline arendus keskendub töötava arenduse loomisele ja vähendab riske seoses analüüsi paralüüsidega. Sage iteratsioonide demonstreerimine annab hulgaliselt vajalikku tagasisidet järgmiste iteratsioonide tarvis. Iteratsiooni elutsükkel on kujutatud joonisel 4.

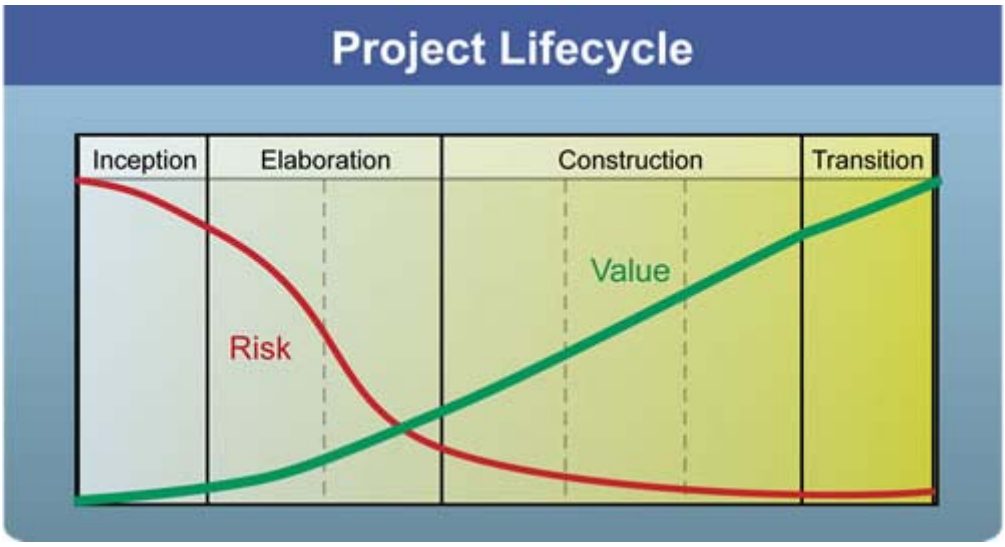


Joonis 4. Iteratsiooni elutsükkel

Allikas: OpenUP Plug-in [4]

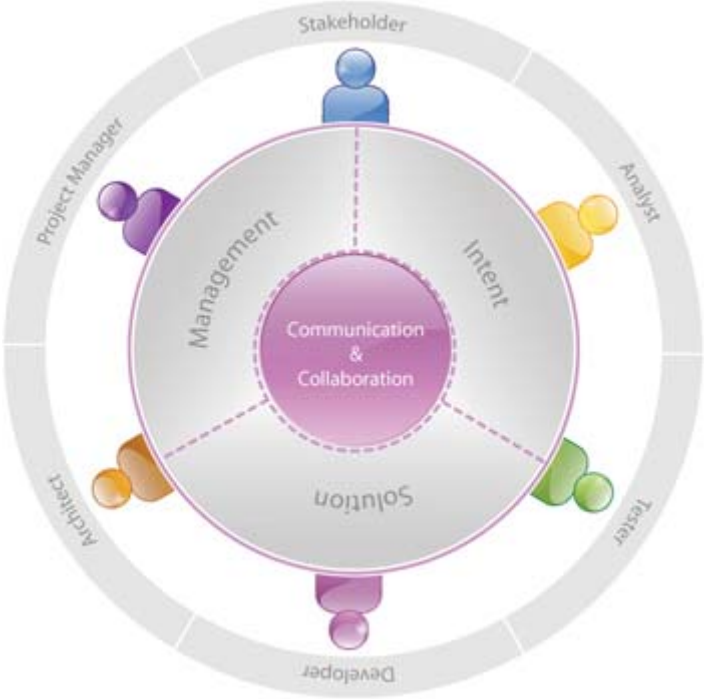
OpenUP projekti elutsükkel – pakub järelevalvet, läbipaistvust ja juhtmehhanisme, et kontrollida projekti finantseerimist, skoopi, riske, saavutatavat väärtust ja muid aspekte. Iga iteratsiooniga toode kasvab, mis annab võimaluse aru saada, kui suur on juba loodud väärtus ja kui hästi on projekt kontrolli all. Samuti annab see võimaluse arendajatele võimaluse projekti korrigeerida, saavutamaks optimaalsemat tulemust. Joonisel 5 on kujutatud projekti elutsükkel, kus iteratsioonid on jagatud nelja faasi.

- Teke – Projekti skoobi ja eesmärkide kinnitamine
- Ettevalmistamine – Loodava tarkvara tehnilise arhitektuuri kinnitamine ja riskide analüüs seni loodud väärtuse alusel
- Konstrueerimine – Seni loodud tarkvara valmidusastme hindamine ja primaarse fookuse seadmine optimeerimisele ning viimistlemisele
- Üleminek – Tarkvara valmiduse hindamine kasutusele võtmise seisukohalt



Joonis 5. Projekti elutsükkel
Allikas: OpenUP Plug-in [4]

OpenUP rollid – on jagatud projektiga seotud huvide, tegevuste ja oskuste järgi. Üks roll ei pruugi alati olla vastavuses ühe inimesega, üks inimene võib kanda mitut rolli erinevates projekti etappides ja vastupidi, mitu inimest sama rolli. Selliste rollidega meeskonnamäng aitab meeskonnal näha asja erinevate vaatenurkade alt ja erinevatest huvidest lähtuvalt. Joonisel 6 on kujutatud rollide jaotus ja nende koostoime.



Joonis 6. Rollide jaotus ja koostoime
Allikas: OpenUP Plug-in [4]

2.2. Tootmisprotsessi analüüs

Tootmisprotsessi analüüsimiseks on kasutatud OpenUP arendusprotsessi koos UML märgisüsteemiga [8]. Tabelis 1 on toodud OpenUP probleemipüstitus ja tabelis 2 OpenUP osapoolte kirjeldused.

Tabel 1. Probleemipüstitus

Probleem	Peale majandustarkvara Microsoft Navision Attain juurutamist puudub ülevaade tootmisressursi planeerimisest ja tootmisprotsessist. Puudub valmistoodete kohta tootmisajalugu. Vaja on luua süsteem tootmise ressursi- ja ajaplaneerimiseks ning tootmisprotsessi etappide info talletamiseks
Probleem mõjutab	Tootmisosakond, müügiosakond, garantiiosakond, kommertsosakond.
Probleemi mõju väljendub	Kliendi rahulolu, laokäibe kiirus.
Lahendus	Tootmistarkvara vertikaallahendusena majandustarkvara juurde.

Tabel 2. Osapoolte kirjeldus

Osapoole nimetus	**Osapoole kirjeldus**
Tootmisosakond	Toodab arvuteid, vajab infot mida ja kuna toota. Tahab paremini planeerida aega

Müügikonsultandid	Võtavad vastu tellimusi klientidelt, müüvad klientidele valmistoodangut. Tahavad saavutada kliendi rahulolu lubades reaalseid tootmise tähtaegu
Tootejuhid	Hangivad detaile ja komponente vastavalt vastuvõetud tellimustele, planeerivad ettetootmist. Tahavad saavutada optimaalse laokäibe kiiruse, varustades tootmist õigeaegselt õigete materjalidega
Juhtkond	Kujundab strateegia ja poliitika, tahab koostada võimalikult täpse kasumiplaani ja teada täpset valmistoodangu omahinda
Kliendid	Esitavad tellimusi ja ostavad valmistoodangut. Tahavad teada täpseid tarneaegu.
Konkurendid	Toodavad ja müüvad analoogseid tooteid. Tahavad saavutada kliendi rahulolu oma positsiooni tugevdamiseks.
Tarnijad	Tarnivad detaile ja komponente, tahavad aegsasti teada, mida ja kuna tarnida.
Omanikud	Rahastavad tegevust ja tahavad teenida kasu
IT osakond	Haldab süsteemi ja tahab, et oleks võimalikult vähe probleeme

2.2.1. Kasutajate IT-töökeskkonna kirjeldus

Tootmistarkvara Favourite II kasutajaliides on kavandatud klient – server rakendusena, andmebaasiserverina on kasutusel MS SQL Server 2000. Rakenduse tüübi valikul on lähtutud eeldusest, et tarkvara peab olema visuaalselt ja kasutamise loogikalt võimalikult sarnane kasutatavale majandustarkvarale Microsoft Navision Attain 3.6, selleks et ettevõtte töötajatel oleks võimalikult lihtne igapäevatöös majandustarkvara ja tootmistarkvara paralleelne kasutamine. Andmebaasina on kasutusel ühine andmebaasiserver majandustarkvaraga, mis tagab kahe tarkvara reaalajas koos toimivuse ja maksimaalse ühilduvuse. Tarkvara kasutajaliides on disainitud MDI aplikatsioonina, kus töö toimub põhiakna tööväljal alamakendes.. Tegevuste valimine toimub peamenüü aknas. Kasutajaliidese üldise loogikana on kasutatud register – kaart põhimõtet, kus mingi tegevuse valimisel avatakse esmalt registriloend, mis kuvab varasemaid tehinguid ja pakub väga paindlikke otsingu ja filtreerimise võimalusi. Registriloendis saab avada olemasoleva või luua uue kaardi.

2.2.2. Lahenduse üldkirjeldus

Infosüsteemi baasiks on majandustarkvara Microsoft Navision Attain versioon 3.6 ja andmebaasiserver Microsoft SQL Server 2000. Majandustarkvara vertikaallahendus Favourite II on loodud programmeerimisvahendiga Borland Delphi 7.

Püsiandmed

Püsiandmete haldamine toimub majandustarkvaras, vertikaallahenduses kasutatakse nende esitamiseks registriloendit. Kõik registriloendid on üles ehitatud sama struktuuriga ja toimivad sama vormi baasil. Põhiliste tegevustena saab kasutaja loendis seadistada kuvatavate väljade loetelu ja väljade järjestust, lisaks saab määrata tabeli filtreid ja sorteerimise järjestusi. Kõik loendi seadistused salvestatakse kasutajapõhiselt andmebaasis. Kiireks otsimiseks aktiivselt väljalt on loendis välja filtri sisestuslahter. Vaikimisi kuvab loend esimesed 100 kirjet.

- **Müügitellimused -** Registriloend kuvab reaalajas infot majandustarkvaras koostatud müügitellimuste kohta.
- **Kaubad -** Registriloend kuvab reaalajas infot majandustarkvaras sisestatud kaupade ja laoseisude kohta.
- **Kliendid -** Registriloend kuvab reaalajas infot majandustarkvaras sisestatud klientide kohta.

Tootmismoodul

Tarkvara tootmismooduli põhiülesandeks on tootmisprotsessi kõikide etappide fikseerimine ja tootmisvõimsuse planeerimine. Tootmismooduli sisendiks on majandustarkvaras ette valmistatud müügitellimused ja väljundiks on lattu arvele võetud valmistoodang.

- **Tootmistellimused -** Kui majandustarkvaras on müügitellimus ette valmistatud, tuleb selle alusel koostada vertikaallahenduse tootmismoodulis tootmistellimus. Tootmistellimused salvestatakse vertikaallahenduse andmebaasis.
- **Valmistuskaardid -** Valmistuskaardi eesmärgiks on säilitada informatsioon iga toote tootmisprotsessi kohta, monteerimise alguse ja lõpu ajad, testitingimused, paigaldatud detailide loend ja seerianumbrid jm..
- **Tootmise kalender -** Tootmise planeerimine ja tellitud toote valmimistähtaja prognoosimine toimub tootmiskalendri abil.
- **Tootmise loend -** Tootmise loend on abivahend tootmisosakonna vanemtehnikule tööde planeerimisel. Loend annab ülevaate ootel, tootmisse määratud ja töös toodetest.

2.2.3. Alternatiivlahendused

Arvestades keskmise suurusega eesti ettevõtte vajadusi ja võimalusi ei ole mõistlik ERP (Enterprise Resource Planning) majandustarkvara ja selle vertikaallahenduste juurutamine „karbitoodetena". Majandustarkvaras realiseerimata, ettevõtte spetsiifiliste protsesside katmiseks on mõistlik kasutada spetsiaalset vertikaallahendust. Võttes arvesse vertikaalsete protsesside keerukust ja unikaalsust on tihtipeale standardse lahenduse asemel mõistlikum ja kasulikum arendada spetsiaalne vertikaallahendus, mis täielikult peegeldab ettevõtte vertikaalseid protsesse ja tagab sellega tööprotsessi sujuva ja efektiivse kulgemise.

Stsenaarium 0: Jätkata samal viisil

+ Harjumuspärased töömeetodid, madalad IS hooldus- ja arenduskulud
– Eelpoolkirjeldatud probleemid

Stsenaarium 1: Hankida standardne tootmistarkvara

S Senine kogemus karbitoodete kasutuselevõtmisel
W Protsesside jäikus tulenevalt KJS-ist mistõttu neid muuta on raske
O Lai valik konkureerivate toodete vahel
T Kasutajatoe kadumine, suur sõltuvus konkreetse toote pakkujast

Stsenaarium 2: Arendada spetsiaaltarkvara majandustarkvara vertikaallahendusena

S Paindlik, ettevõtte vajadusi arvestav juurutusprotsess
W Vähene kogemus ja kompetents
O Paindlikud B2B ja B2C lahendused, võimalus tarkvara juurutada väheste kuludega ka teistes korporatsiooni ettevõtetes
T Suur sõltuvus tarkvara arenduspartneritest.

Otsus: Arendada spetsiaaltarkvara majandustarkvara vertikaallahendusena

2.2.4. Talitlusprotsesside kirjeldus

Talitusprotsesside kirjeldusi kasutatakse, et skemaatiliselt visualiseerida protsessid ehk tegevused, mida teostatakse, jõudmaks soovitud lõpptulemuseni. Lisas 3, joonisel 1 toodud põhiprotsessis on kujutatud kõik tegevused, kus sisendiks on kliendi soov ja väljundiks

kliendi rahulolu. Lisas 3, joonisel 2 toodud tootmisprotsessis on kujutatud detailselt tegevused, mida teostatakse toote komplekteerimisel. Allolevas tabelis 3 on toodud sõnastik, mis annab detailsemad seletused skeemidel ja kasutuslugudes esinevate terminite kohta.

Tabel 3. Sõnastik

Termin, lühend	**Seletus**
Valmistuskaart	Vorm, mille salvestatakse kogu info konkreetse valmistoote eksemplari komplektsuse ja tootmisprotsessi kohta
Tootmistellimus / Müügitellimus	Vorm, mis väljendab kliendi soovi
Staatus	Ühtne klassifikaator erinevate vormide jaoks: Salvestatud; Kinnitatud; Ootel; Töös; Valmis; Väljastatud; Tühistatud
Tehnik / Tootmistehnik	Isik, kes monteerib komponentidest toote ja testib selle vastavust nõuetele
Vanemtehnik	Isik, kes planeerib tootmist ja konteerib tooted peale monteerimist majandustarkvaras
Tootejuht	Isik, kes hangib detaile vastavalt tootmise vajadusele
Müüja / Müügikonsultant	Isik, kes vastutab kliendiga suhtlemise eest, sisestab tootmistellimused ja väljastab valmistooted
Favourite	Tootmistarkvara vana versioon, mis oli integreeritud vana raamatupidamistarkvaraga
Favourite II / Tarkvara / Tootmistarkvara / Infotarkvara	Planeeritav tootmistarkvara, mis saab olema integreeritud hetkel kasutatava majandustarkvaraga
Microsoft Navision Attain / Majandustarkvara	Hetkel kasutatav majandustarkvara, juurutatud moodulitega: Pearaamat; Ost; Müük; Ladu;
Toode / Valmistoode	Komponentidest vastavalt kliendi soovile komplekteeritud ja testitud lõpptoode.
Detail / Komponent	Algosad, millest komplekteeritakse lõpptoode
Test	Valmistoote nõuetele vastavuse kontroll
Klient	Isik või äriühing, kes on avaldanud soovi toote tellimiseks
KJS	Ettevõttes juurutatud ja sertifitseeritud kvaliteedijuhtimissüsteem vastavalt standardile ISO 9001:2000

2.2.5. Funktsionaalsed nõuded

Võttes aluseks talitusprotsesside kirjeldused, koostatakse kasutuslood, mis kirjeldavad detailselt kasutaja suhtlust süsteemiga ehk andmete sisestamise vajadust ja sisestatavate andmete struktuuri ning vajalikku tagasisidet erinevates talitusprotsessides. Näites 1 on toodud kasutuslugu tootmisprotsessi kohta.

Näide 1. Kasutuslugu KL-001 – Tootmisprotsessi info sisestamine

Eesmärk — Infotarkvaras on kvaliteetsed ja detailsed andmed toote komplektsuse ja tootmisprotsessi kohta

Tegutseja — Tootmistehnik, vanemtootja

Eeltingimused — Tegutseja on autoriseeritud ning tal on olemas vastavad kasutajaõigused

Päästik — Tegutseja valib rakenduse menüüst valiku „Valmistuskaardid"

Peastsenaarium

1. Tarkvara kuvab valmistuskaartide loetelu
    a. Toote seerianumber (valmistuskaardi number)
    b. Toote kood
    c. Toote nimetus
    d. Kliendi kood
    e. Kliendi nimi
    f. Tähtaeg
    g. Tellimuse number
    h. Tellimuse kuupäev
2. Tegutseja sisestab soovitud otsingukriteeriumi, valdavalt seerianumbri
3. Tarkvara kuvab kriteeriumile vastavad valmistuskaardid
4. Tegutseja valib soovitud valmistuskaardi
5. Tarkvara kuvab valmistuskaardi vormi (päis)
    a. Toote seerianumber (valmistuskaardi number)
    b. Toote kood
    c. Toote nimetus
    d. Kliendi kood
    e. Kliendi nimi
    f. Tähtaeg
    g. Tellimuse number
    h. Tellimuse kuupäev
    i. Valmistoote kood
    j. Staatus
    k. Tootmistehniku kood
    l. Monteerimise algusaeg
    m. Monteerimise lõppaeg
    n. Test läbitud
6. Tarkvara kuvab valmistuskaardi vormi (sisu)
    a. Paigaldatud detaili kood
    b. Paigaldatud detaili nimetus
    c. Seerianumber
    d. Tellitud detaili kood
7. Tegutseja alustab monteerimist valmistuskaardi funktsiooniga „Alusta tööd"
8. Tarkvara muudab toote staatust
9. Tegutseja sisestab tootmistehniku koodi (päis)
10. Tegutseja sisestab detailide seerianumbrid, kasutades võimalusel triipkoodilugejat ja vajadusel muudab detaili koodi kui tootesse paigaldati tellimusest erinev asendusdetail (sisu)
11. Tegutseja sisestab märke testi läbimise kohta (päis)

12. Tegutseja lõpetab monteerimise valmistuskaardi funktsiooniga „Kinnita ja Prindi”
13. Tarkvara muudab toote staatust ja genereerib tootele valmistoote koodi
14. Tarkvara prindib tootmislehe
15. Tegutseja prindib toote markeerimiskleebised valmistuskaardi funktsiooniga „Prindi kleebised”

Sellega on kasutuslugu lõppenud

Alternatiivstsenaariumid

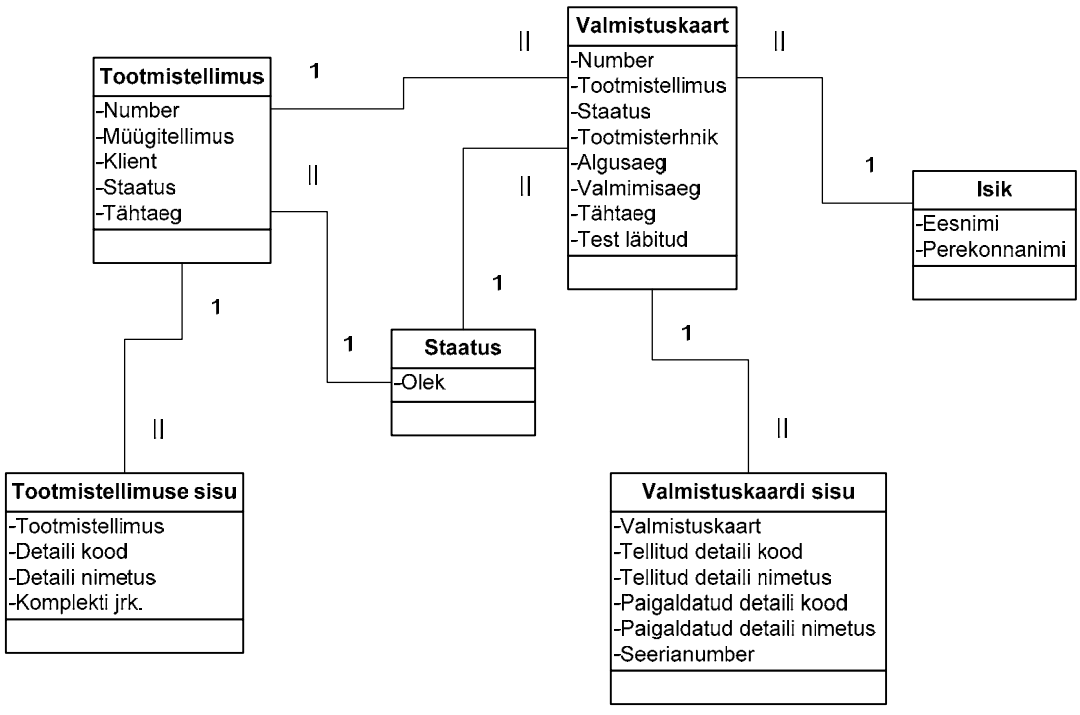
A. Tarkvaras ei leidu otsingukriteeriumidele vastavat valmistuskaarti
    a. Valmistuskaardi vormi ei saa avada
    b. Kasutaja saab täpsustada/parandada otsingukriteeriumid
B. Tarkvaras ei leidu sisestatud tootmistehniku koodi
    a. Tarkvara ei kuva tootmistehniku nime
    b. Kasutaja peab sisestama korrektse koodi

Järeltingimused
Kvaliteetsed andmed toote ja tootmisprotsessi kohta on salvestatud andmebaasi

Kasutusloospetsiifilised tehnilised nõuded
öökoha arvutiga peavad olema ühendatud ja korrektselt installeeritud triipkoodilugeja ja kleebiseprinter

2.2.6. Ainevaldkonna mudel

Kasutuslugude alusel luuakse klassiskeemid, mis kirjeldavad protsessis osalevaid objekte ning nendevahelisi seoseid. Joonisel 7 on kujutatud tootmisprotsessi kasutusloole vastav klassiskeem, millest edasi saab moodustada süsteemi tehnilise arhitektuuri osasse kuuluva ER mudeli, mis on aluseks andmebaasi struktuuri loomisel.



Joonis 7. Klassiskeem

2.2.7. Infosüsteemi loogiline ja füüsiline arhitektuur

Tehnilised nõuded

1. Süsteem peab vastama ISKE klassifikatsiooni järgi turvaklassile K1T1S1, seega on nõutav madal (L) turbeaste:
    a. K1 – töökindlus – 90% (lubatud summaarne seisak nädalas ~ ööpäev); lubatav nõutava reaktsiooniaja kasv tippkoormusel – tunnid (1÷10);
    b. T1 - info allikas, selle muutmise ja hävitamise fakt peavad olema tuvastatavad; info õigsuse, täielikkuse, ajakohasuse kontrollid erijuhtudel ja vastavalt vajadusele;
    c. S1 - info asutusesiseseks kasutamiseks: juurdepääs teabele on lubatav juurdepääsu taotleva isiku õigustatud huvi korral;
2. Süsteemi peab saama kasutada ainult sisevõrgus
3. Tarkvara arhitektuur peab olema võimalikult lihtsalt liidestatav väliste süsteemidega
4. Tarkvara peab olema realiseeritud platvormil, millel on Eestis piisavalt arenduskompetentsi
5. Maksimaalne taasteaeg 1 päev
6. Päringute kestvus kuni 5 sekundit 95% juhtudest
7. Terviklus – kinnitatud vorme ei saa muuta ega kustutada, saab märkida tühistatuks
8. Rakendus peab olema Inglise keelne, võimaldades valida kohalikku keelt, mille puhul kasutatakse vormide terminite tõlkeid andmebaasist.

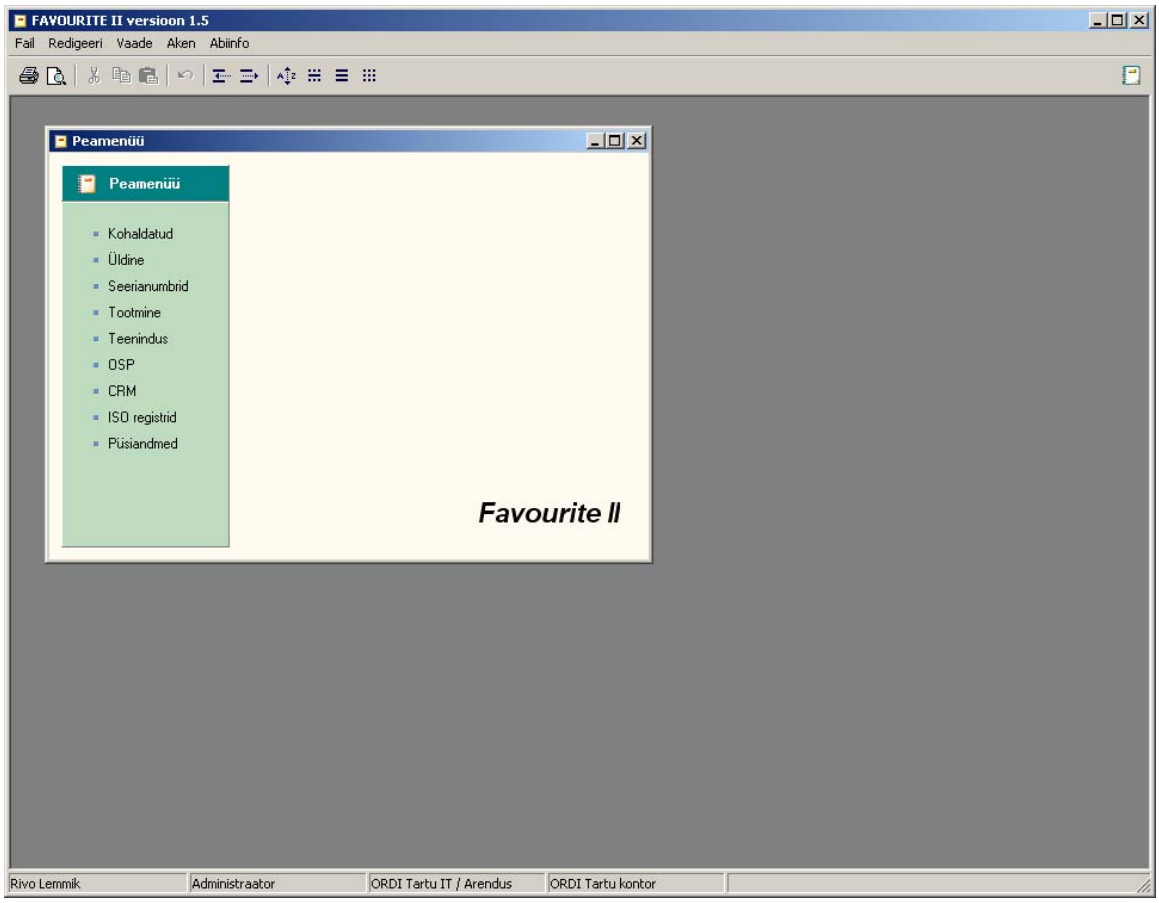
Süsteem on keskse andmesäilitusega, töökohajaamades andmeid ei paikne. Rakendus võib paikneda töökoha arvutites või Terminali lahenduse korral serveris.

2.3. Vertikaallahenduse kavandamine

Käesolevas alapeatükis on kasutatud materjali [1].

Vertikaallahenduse kavandamisel on tegevuse põhiste protsessi kirjelduste alusel loodud objekti põhine OLTP (On-Line Transaction Processing) andmebaasi struktuur, andmelao ja analüüsi jaoks vajalikku tegevuse põhist OLAP (On-Line Analytical Processing) vaadet struktuurselt kirjeldatud ei ole.

Tarkvara kasutajaliides on disainitud MDI aplikatsioonina, kus töö toimub põhiakna tööväljal alamakendes.. Tegevuste valimine toimub joonisel 8 kujutatud peamenüü aknas. Kasutajaliidese üldise loogikana on kasutatud register – kaart põhimõtet, kus mingi tegevuse valimisel avatakse esmalt registriloend, mis kuvab varasemaid tehinguid ja pakub väga paindlikke otsingu ja filtreerimise võimalusi. Registriloendis saab avada olemasoleva või luua uue kaardi.



Joonis 8. Vertikaallahenduse kasutajaliides

2.3.1. Püsiandmed

Püsiandmete haldamine toimub majandustarkvaras, vertikaallahenduses kasutatakse nende esitamiseks registriloendit. Kõik registriloendid on üles ehitatud sama struktuuriga ja toimivad sama vormi baasil. Põhiliste tegevustena saab kasutaja loendis seadistada kuvatavate väljade loetelu ja väljade järjestust, lisaks saab määrata tabeli filtreid ja sorteerimise järjestusi. Kõik loendi seadistused salvestatakse kasutajapõhiselt andmebaasis. Kiireks otsimiseks aktiivselt väljalt on loendis välja filtri sisestuslahter. Vaikimisi kuvab loend esimesed 100 kirjet.

Müügitellimused – registriloend kuvab reaalajas infot majandustarkvaras koostatud müügitellimuste kohta. Info päritakse otse majandustarkvara andebaasist vastava *View* vahendusel. Loend on kasutusel tootmistellimuse kaardil müügitellimuse numbri sisestamise lihtsustamiseks *Lookup* funktsiooni kaudu.

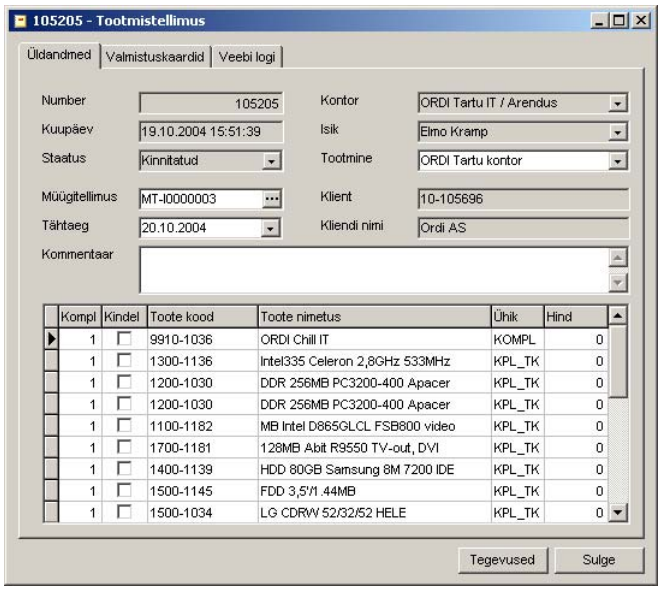
Kaubad – registriloend kuvab reaalajas infot majandustarkvaras sisestatud kaupade ja laoseisude kohta. Info päritakse otse majandustarkvara andebaasist vastava *View* vahendusel. Loend on kasutusel kõikides moodulites kauba numbri sisestamise lihtsustamiseks *Lookup* funktsiooni kaudu.

Kliendid – registriloend kuvab reaalajas infot majandustarkvaras sisestatud klientide kohta. Info päritakse otse majandustarkvara andebaasist vastava *View* vahendusel. Loend on kasutusel kõikides moodulites kliendi numbri sisestamise lihtsustamiseks *Lookup* funktsiooni kaudu.

2.3.2. Tootmismoodul

Tarkvara tootmismooduli põhiülesandeks on tootmisprotsessi kõikide etappide fikseerimine ja tootmisvõimsuse planeerimine. Tootmismooduli sisendiks on majandustarkvaras ette valmistatud müügitellimused ja väljundiks on lattu arvele võetud valmistoodang.

Tootmistellimused – majandustarkvaras ette valmistatud müügitellimuse alusel koostatakse vertikaallahenduse tootmismoodulis tootmistellimus. Tootmistellimused salvestatakse vertikaallahenduse andmebaasis, tootmistellimuste tabeli väljaloend on toodud lisas 4, tabelis 1. Tootmistellimuse aken on kujutatud joonisel 9, kus tuleb vastavasse lahtrisse valida või sisestada majandustarkvara müügitellimuse number, selle peale tõlgendatakse müügitellimuse sisu kindla loogika alusel tootmistellimuse sisuks. Enne tootmistellimuse kinnitamist saab üle kontrollida kas automaatne loogika on müügitellimuse sisu õigesti arvutikomplektideks jaganud ning vajadusel saab teha parandusi ja sisestada vabas vormis kommentaare. Kliendi soovil võib teha märke väljale „Kindel” mis tähendab, et tellitud artiklit ei tohi tootmises analoogse vastu muuta ilma kliendi nõusolekuta.

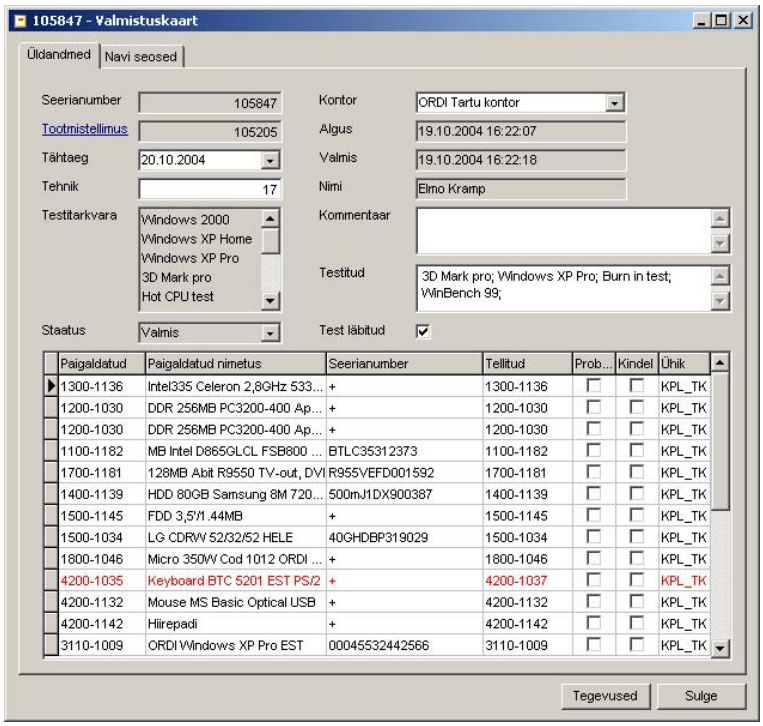


Joonis 9. Tootmistellimuse aken

Tootmistellimuse kinnitamisel kontrollitakse, kas majandustarkvaras on müügitellimus arhiveeritud, et vajadusel saaks taastada tootmisse andmise eelse seisu. Seejärel kustutab vertikaallahendus müügitellimuselt kõik arvutikomplekti päise ja sisu read ning edasi toimub toodetava arvuti sisu haldamine vertikaallahenduses. Peale tootmistellimuse kinnitamist luuakse automaatselt vastav arv valmistuskaarte, iga arvuti kohta üks. Tootmistellimuse kinnitamisega lõpeb müüja töö arvutikomplekti tootmisse panemisel. Lisas 4, tabelis 2 on kirjeldatud tootmistellimuse sisu tabeli väljad.

Valmistuskaardid

Valmistuskaardi eesmärgiks on säilitada informatsioon iga arvuti tootmisprotsessi kohta, monteerimise alguse ja lõpu ajad, testitingimused, paigaldatud detailide loend ja seerianumbrid jm. Valmistuskaartide tabeli väljaloend on toodud lisas 4, tabelis 3. Lähtuvalt graafikust monteerib tootmisosakond arvutid. Monteerimise alguses avatakse joonisel 10 kujutatud valmistuskaart. Monteerimise alustamisel fikseeritakse automaatselt töö alustamise kuupäev ja kellaaeg. Järgmisena sisestatakse detailide seerianumbrid ja vajadusel muudetakse paigaldatud artikli välja kui arvutisse paigaldati mingil põhjusel tellimusest erinev detail, vahetatud artikli read kajastuvad valmistuskaardil punast värvi.



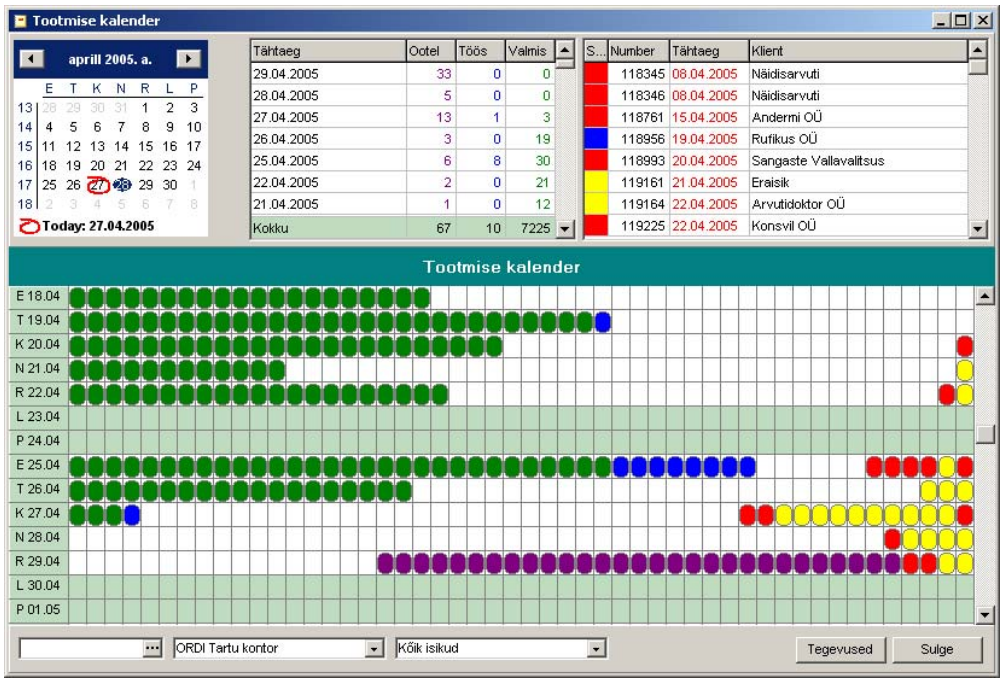
Joonis 10. Valmistuskaardi aken

Kui arvuti on monteeritud ja testi läbinud siis valitakse testitarkvara loendist testimisel kasutatud programmid ja märgitakse linnuke Test läbitud. Valmistuskaardi detailide loendis on veel kaks linnukest lisainformatsiooni tarvis. Kui arvutit ei saa mingil põhjusel monteerida siis tehakse märge takistuse põhjustanud artikliga rea „Probleem" väljale. Väljal „Kindel" on märge kui klient soovis, et mõnd artiklit ei tohi tootmine analoogse vastu vahetada ilma kliendi nõusolekuta. Peale arvuti valmimist valmistuskaart kinnitatakse, selle peale märgitakse automaatselt valmimise kuupäev ja kellaaeg ning prinditakse arvuti markeerimise kleebised. Majandustarkvaras tekib komplektižurnaali rida, mille konteerimisel kantakse arvuti detailid kulusse ja arvele võetakse valmis arvuti. Lisas 4, tabelis 4 on kirjeldatud valmistuskaardi sisu tabeli väljad.

.

Tootmise kalender

Tootmise planeerimine ja tellitud arvuti valmimistähtaja prognoosimine toimub joonisel 11 kujutatud tootmiskalendri abil. Tootmiskalender kuvab üleval keskel summaarse info, üleval paremal loendina kõik ootel ja töös arvutid ning graafiliselt suures graafikus kõik arvutid staatust iseloomustavates värvides. Valides kalendri jalusest müügiisiku filtri, kuvatakse täisvärvis pallikestena ainult valitud müügiisikuga seotud arvutid, teised pallikesed kuvatakse piirjoonena, samuti kuvab loend üleval paremal ainult valitud müügiisikuga seotud arvutid.



Joonis 11. Tootmise kalender

Tootmise loend

Tootmise loend on abivahend tootmisosakonna vanemtehnikule tööde planeerimisel. Loend annab ülevaate ootel, tootmisse määratud ja töös arvutitest, kuvades jooksvalt akna alumises pooles aktiivse arvuti konfiguratsiooni. Arvuti sisu osas saab ilma valmituskaarti avamata, otse loendi kaudu muuta paigaldatavat artiklit ja märkida probleemi välju. Loendi põhifunktsiooniks on arvutite tootmisse määramine, selleks kasutades funktsiooni „Märgista", märgistatakse soovitud arvutid ja funktsiooniga „Määra töösse" muudetakse kõikide märgitud arvutite staatus.

2.4. Kasutajatugi ja arendus

Peale infosüsteemi kasutuselevõttu algab uus etapp, kus pidevalt on vaja pakkuda kiiret kasutajatuge ja süsteemi edasi arendada. Kasutajatoe, veaparanduste ja arenduste maht on otseselt sõltuv kogu eelnenud protsessist, olenedes põhiliselt loodud lahenduse kvaliteedist.

Käesolevas töös kirjeldatud infosüsteemi hakkasid päevapealt kasutama kõik ettevõtte 90 töötajat, kellest ~50 sõltuvad otseselt oma igapäevatöö tegemisel infosüsteemist. Kasutajatoe pakkumine sellise süsteemi käivitamisel tähendas umbes 2 kuu pikkust perioodi kus iga päev laekus erinevaid kanaleid pidi kümneid küsimusi, mis kõik pidid saama operatiivselt

lahendatud. Kuna sellise toe pakkumise mahu juures ei ole võimalik teha praktiliselt ühtegi arendustööd siis on ülimalt oluline, et enne süsteemi käivitamist ollakse täiesti veendunud, et süsteem on korduvalt testitud ja läbi katsetatud ning on võimeline tõrgeteta töötama. Kui sellel perioodil oleks ilmnenud mõni tegemata jätmine või ebakvaliteetne teostus eelnenud tegevustes, võinuks see tähendada praktiliselt kogu ettevõtte tegevuse lakkamist ja sellest tulenevat otsest rahalist kahju.

Peale aktiivse kasutajatoe pakkumise perioodi, kui pole enam massilist vajadust toe järele, algab süsteemi arendamise periood. Arendusperioodi alguses on oht uuest süsteemist vaimustuses olevate kasutajate poolt tulev arendusettepanekute uputus, kus kõigil tekib kohutavalt palju ideid mis kõik võiks veel olla süsteemis realiseeritud. Oluline on mitte sattuda kaosesse, kus tegeldakse mõttetute arendustöödega ja olulisemad ettepanekud on jäänud tahaplaanile. Sellist olukorda aitab vältida konkreetne prioriteetide määramine ja tööde planeerimine. Ilma kindla plaanita, kui pidevalt väikesi arendustöid vahele võtta tekib endalegi arusaamatult väga suur ajakadu, kus näiteks kuu aja jooksul on tehtud kohutavalt palju tööd aga kui hakata analüüsima, mis tööd tehtud said siis selgub, et 90% ulatuses oli tegemist just väikeste eelisjärjekorras graafikusse vahele võetud töödega.

Lisaks arendustöödele tuleb pidevalt jälgida ja analüüsida süsteemi toimivust andmemahtude kasvades nii, et oleks võimalik teostada juba ennetavalt vajalikke regulaarseid hooldustöid, vältimaks süsteemi ülekoormatustest põhjustatud eksponentsiaalselt langevat töökiirust, mis võib kriitilise piiri ületamisel lõppeda sisuliselt süsteemi toimivuse lakkamisega.

3. INFOSÜSTEEMI ANDMEMUDELID JA ANDMEANALÜÜS

Infosüsteemi arendusprotsess algab alati ideest, kus on mõistlik mingi protsessi juhtimise või automatiseerimise abivahendina kasutusele võtta infosüsteem. Igal infosüsteemil on alati kaks poolt, sisend ja väljund ehk võimalus süsteemi andmeid sisestada ning edasi võimalus süsteemist andmeid kätte saada. Mõlemad pooled on siinkohal sama olulised kuna süsteem, kuhu ei saa andmeid sisestada on täiesti kasutu ja samuti süsteem, millest pole võimalik sisestatud andmeid kätte saada. Infosüsteemi kasulikkuse määrab suurel määral süsteemi võime andmeid töödelda, piltlikult kujutades on kõige primitiivsem infosüsteem selline, kuskohast on võimalik sisestatud andmeid väljundina kätte saada samal kujul ja samas järjestuses nagu neid sisestati, näiteks avades tühja faili ning sisestades sinna suvalisi andmeid suvalises järjekorras ja suvalise struktuuriga. Ka selline infosüsteem näib kasutu ehk esimene elementaarne vajadus on andmed struktureerida

Valides infosüsteemi loomiseks eelistatava metoodika ning sobiva mudeli on võimalik informatsioonile hakata lähenema süstemaatiliselt, läbides metoodika etappe ning kirjeldades süsteemi mudelit. Tavaliselt läbitakse süsteemi loomisel järgmised etapid:

- Idee ja eelanalüüs süsteemi vajalikkuse osas
- Analüüs, mis hõlmab üldiselt protsesse ja tegevusi
- Protsesside kirjeldamine tegevuste jadana
- Tegevustest objektide leidmine ja objekt-orienteeritud mudeli loomine
- Objektide vaheliste seoste määratlemine
- Kasutuslooskeemide kirjeldamine
- Süsteemi kasutajaliidese kavandamine
- Süsteemi relatsioonilise andmemudeli loomine
- Programmeeritakse rakendus
- Disainitakse OLTP (On-Line Transaction Processing) andmebaas
- Kasutajate koolitamine ja süsteemi kasutusele võtmine

3.1. Klassikaliselt arendatud süsteemide probleem

Klassikalise lähenemise tulemusena on loodud süsteem, mis katab täielikult kõigis protsessides andmete sisestamise vajaduse ning on väga hästi struktureeritud, normaliseeritud ning minimaalse andmeliiasusega, võimaldades kasutajal arusaadavalt ja mugavalt sisestada süsteemi andmeid tegevuse kohta, mida ta parasjagu sooritab. Kui selline süsteem on juurutatud ja kasutusele võetud, tekib peagi vajadus sisestatud andmeid süsteemist kätte saada ning kuna kõik andmed on objekti põhiselt klassifitseeritud ja objektid omavaheliste relatsioonidega seotud, on ka väga lihtne saada andmetest objektist lähtuvat väljundit. Lisaks objekte kirjeldatavatele andmetele on aga vaja ka süsteemist saada väljundit tegevuste kohta, mis on juba komplekssem soov ja siinjuures tekivad tavaliselt järgnevad probleemid:

1. OLTP (On-Line Transaction Processing) andmebaas on analüütiliste päringute sooritamiseks aeglane kuna erinevaid objekte tuleb siduda lugematute relatsioonide kaudu, et tulemus väljendaks tegevust.
2. Keerukad päringud koormavad süsteemi ja segavad andmesisestajate tööd kuna süsteem muutub aeglaseks.
3. OLTP andmebaasid võivad paikneda lokaalselt ettevõtte erinevates asukohtades ja ei ole tsentraalset andmebaasi
4. Aruannete loomine on alati süsteemi arhitekti töö kuna tavakasutaja ei suuda orienteeruda andmebaasi objektides ja relatsioonides
5. Tihtipeale ei ole võimalik vaadelda andmete seisu tagasiulatuvalt ehk taastada suvalisel ajahetkel mineviku seisu

Sellises olukorras tavaliselt lahendatakse probleem esmalt tehniliste vahendite abil ehk luuakse andmebaasist identne koopia, mida värskendatakse perioodiliselt, näiteks igal öösel, samuti konsolideeritakse koopiasse kokku perioodiliselt ettevõtte erinevates asukohtades paiknevad andmebaasid. Selline lihtne lahendus aitab aga täielikult vaid 2. Ja 3. probleemi kõrvaldamisel ehk aruandlus ei sega enam andmesisestajate tööd ja on olemas ettevõtte tsentraalne andmebaas. Lisaks leevendatakse veidi 1. probleemi järgmiste meetoditega:

- Arvestades, et paljud analüütilised aruanded on seotud samade objektidega, näiteks klientidega, siis on mõistlik luua andmebaasi koopiasse vaated, mis grupeerivad hulga alamobjekte kokku üheks mahukaks baasobjektiks

- Arvestades, et osade objektide vahelised relatsioonid võivad olla väga keerukad, on mõistlik luua lisaks uusi staatilisi objekte, mille andmed genereeritakse keerulise äriloogika alusel üks kord perioodis ning mida kasutatakse analüüsi tarvis sama perioodi jooksul korduvalt

Olemuselt on need mõlemad tegevused seotud andmestruktuuri denormaliseerimisega ja sisaldavad andmeliiasust, mida relatsiooniline andmebaas ei sisalda ehk tegemist on andmelao OLAP (On-Line Analytical Processing) struktuurile omaste tunnustega. Samas on endiselt osaliselt lahendamata 1. probleem kuna analüütilised päringud on endiselt väga keerukad ja täielikult on lahendamata 4. Ja 5. Probleem. Sellises olukorras jõutakse tavaliselt mõne aja pärast välja tõdemuseni, et süsteemi juurde on vaja ehitada klassikaline andmeladu.

3.1.1. Ebaefektiivne andmeladu

Operatiivsüsteemi transaktsiooniline vaade andmetele ja andmelao analüütiline vaade andmetele on piltlikult omavahel 90 kraadi pööratud vaated, kus transaktsiooniline vaade lähtub andmete sisestamise vajadustest objekti põhiselt ning analüütiline vaade andmete analüüsimise vajadusest tegevuste põhiselt. Analüütilise vaate vajadus on triviaalne kuna süsteemi loomisel on ju alati esimeseks lähtepunktiks protsess ehk tegevused, mida soovitakse infosüsteemis kajastada. Sellest tulenevalt on ka informatsiooni analüüsimisel primaarse tähtsusega tegevuste analüüs ning sekundaarse tähtsusega objektide statistika. Andmelao loomisel läbitakse tavaliselt järgnevad etapid:

- Võetakse aluseks eelnevalt loodud protsesside kirjeldused ning eristatakse olulised tegevused
- Võetakse aluseks eelnevalt loodud objekt-orienteeritud mudel ning pöörd-projekteeritakse objektide seosed tegevustega
- Luuakse mõistlik hulk OLAP (On-Line Analytical Processing) kuubid, lähtudes olulistest tegevustest, kus iga kuup esindab ühte tegevust ja väljendab selle analüütilist vaadet
- Luuakse OLAP kuupide struktuurile vastav andmemudel ja andmebaas
- Luuakse OLTP ja OLAP andmebaaside vahele ETL (Extract, Transform, Load) äriloogika, mis regulaarselt täidab andmeladu transaktsioonilisse andmebaasi sisestatud andmete baasil

- Juurutatakse aruandluse tööriist, mille abil suudavad süsteemi lõppkasutajad luua omale meelepäraseid aruandeid kuna OLAP andmestruktuur andmelaos on neile loogiliselt või intuitiivselt mõistetav

Selliselt loodud süsteem võib olla näiliselt täiuslik ja täita kõiki vajalikke funktsioone kuid üha enam kerkib selliste süsteemidega esile probleeme, mis on seotud asjaoluga, et see süsteem töötab hästi staatiliselt. Tänapäeva maailmas, kus üheks suureks konkurentsieeliseks on ettevõtte dünaamika, hakkab aga kirjeldatud süsteem takistama enama väärtuse loomist järgmistel põhjustel:

- Süsteemi kui terviku arendamisel toimub lahknevus peale protsesside kirjeldamist, kus edasi arendatakse kahte paralleelset, OLTP ja OLAP süsteemi mistõttu nende arendus ei toimu kunagi sünkroonis. Tulemusena ei kajasta OLAP süsteemi kogu OLTP süsteemi andmeid või halvemal juhul on täiesti halvatud
- Neid kahte eraldiseisvat süsteemi seob ETL, mille arendaja peab tundma lõpuni mõlema süsteemi äriloogikat, keda tavaliselt ei ole olemas kuna mõlemad süsteemid on loodud erinevate arendajate poolt. Tulemusena on OLAP süsteem ebakvaliteetne või ebatäielik
- ETL osa arendab OLAP süsteemi arendaja ning teeb pidevalt ebaotstarbekalt liigset tööd OLTP süsteemi transaktsioonilise mudeli ja protsesside tegevuste vaheliste seoste pöördprojekteerimisel

Sellest tulenevalt on taolise süsteemi arendus dünaamilises keskkonnas ebaefektiivne ja liigselt kulukas ning sealjuures saavutatud süsteem on kvaliteedilt ebaühtlane või ebakvaliteetne.

3.1.2. Efektiivne andmelao lahendus

Infosüsteemi arendusprotsessi tuleb vaadata kui tervikut ja seda strateegilises osas mitte jagada eraldiseisvateks süsteemideks. Nagu eelnevalt vaadeldud, kuna iga süsteem saab alguse tegevustest, on tegevuste analüüsimise vajadus triviaalne, seega on mõistlik arvestada OLAP (On-Line Analytical Processing) vaate realiseerimisega kohe süsteemi esmasel planeerimisel ning mitte jätta seda süsteemi arenduse teiseks etapiks.

Kõik eelnevalt vaadeldud metoodikad ning sinna juurde kuuluvad mudelid sisaldavad sarnast probleemi, kus modelleeritakse äriprotsess ning järgmise sammuna pööratakse vaadet nii, et primaarseks muutub tegevuse asemel objekt ning edasi modelleeritakse süsteemi objekt-orienteeritult. Selline käitumine on loogiline ja õige transaktsioonilise süsteemi seisukohalt kuid jätab kõrvale analüütilise süsteemi arhitektuuri. Samas ei saa ka primaarseks võtta analüütilist arhitektuuri sest sellesse süsteemi ei ole võimalik efektiivselt andmeid sisestada. Lahendus on transaktsioonilise süsteemi mudeli loomine läbi OLAP kuubi, kuna tegelikult seda etappi alati läbitakse, paraku enamasti informaalselt ja ilma kindla struktuurita.

Süsteemi loomise selles etapis, modelleerides mõistliku valiku OLAP kuupidest ja sidudes need rangete relatsioonide kaudu loodavate objekt-orienteeritud mudelitega, on võimalik peale OLTP andmebaasi struktuuri valmimist genereerida automaatselt OLAP andmebaasi struktuur ja ETL äriloogika. Selline lahendus osutub võimalikuks tänu asjaolule, et enam ei pea leiduma arendajat, kes tunneks mõlema süsteemi arhitektuuri kuna OLAP kuubid tunnevad OLTP andmebaasi arhitektuuri, loodud seoste kaudu.

Kõige olulisema tähtsusega siinkohal on eelpoolnimetatud OLAP kuupide mõistliku valiku moodustamine ehk süsteemi arenduse algfaasis tuleb väga täpselt defineerida loodava süsteemi andmeanalüüsi vajadused, mis ehk on veidi harjumatu mõtteviis kuna tavaliselt hakatakse analüütilise vaate ehk aruandluse peale mõtlema alles süsteemi loomise lõppjärgus. Oluline on see siinkohal seepärast, et üks OLAP kuup väljendab ühte tegevust ja võimaldab seda tegevust analüüsida.

Tegevused aga on sarnaselt objektidele hierarhilised, kus ühte tegevust saab jagada mitmeteks tegevusteks, mis selles sisalduvad. Jättes kõrvale tegevuste mõistliku valiku määratluse on põhimõtteliselt võimalik igast transaktsioonilisest andmebaasist moodustada automaatselt üks OLAP kuup, sidudes kõik andmebaasis sisalduvad andmed relatsioonide kaudu. Probleemiks on aga see, et selline OLAP kuup väljendab ja võimaldab analüüsida vaid ühte peamist tegevust, mis on eksistentsiaalne, hõlmates kogu antud süsteemi. Selge on, et sellisel analüüsil puudub reaalne rakendus.

Tulemusena saadud mudel on terviklik süsteemi mudel, kus ei eksisteeri OLTP ja OLAP eraldi harudena vaid on üksteisele järgnevad struktuurid. Samuti on see mudel lahutamatult seotud, realiseeritud süsteemiga ehk ei ole abstraktne eeldus süsteemi loomisele vaid on

loodud süsteemi lahutamatu osa. Samuti tuleb kõik edasised arendused teostada, alustades OLAP kuupide struktuuri muutmisest, seotuna transaktsioonilise andmestruktuuri muudatustega. Ideaalis välistab selline süsteemi arenduse metoodika ja mudeli täiustus kõik eelpoolkirjeldatud probleemid seoses süsteemi analüütilise osaga, samuti on kokkuvõttes efektiivsem. Tulemuseks on kvaliteetsem süsteem ja sealjuures madalam TCO (Total Cost of Ownership).

3.2. Andmelao loomise metoodika

Metoodika kirjeldamisel on kasutatud juhendmaterjale [5] ja [6] ning artikleid majandus-tarkvarade relatsiooniliste andmebaaside kohta [12] ja [13]

Andmeladu on üldsuse poolt heakskiidetud kui parim integreeritud ja järjepidev andmeallikas, teostamaks andmeanalüüsi ja seeläbi äriotsuseid. Põhiküsimuseks on, kuidas peaks olema disainitud andmelao andmebaas, et see parimal viisil toetaks andmelao kasutajate vajadusi. Sellele küsimusele vastuse leidmisel on põhiline ülesanne andmemudelil, mis on reeglina igasuguse andmetöötluse juures lahutamatuks osaks. Kasutatavate andmemudelite kaks põhilist tüüpi on ER mudel ja dimensiooniline mudel. Infosüsteemi operatiivtasandi tarvis on sobivaim andmemudel ER mudel, mida saab teatud mööndustega kasutada ka andmelao jaoks kuid siiski on andmeladudes põhirõhk dimensioonilise mudeli kasutamisel.

Modelleerimine on oluline kuna selle abil on võimalik visualiseerida ärimaailma ehk mudel on abstraktne peegeldus reaalsest maailmast. Traditsioonilised andmemudelid on ER mudelid, mis on loodud lähtuvalt äriprotsessi lõppkasutajate vajadustest ehk andmesisestajate tegevuse põhjal. Dimensiooniline mudel annab seevastu oluliselt laiema vaate abstraktsemate küsimuste osas, millele äriprotsessis osalejad vajavad vastuseid. Dimensioonilises mudelis suudavad kasutajad kergesti navigeerida ning soovitud infot leida.

3.2.1. ER mudel

ER mudelit esitatakse traditsiooniliselt ER diagrammina, kus kasutatakse kolme põhilist tüüpi elemente: olem, seos ja atribuut.

Olem – väljendab persooni, kohta, asja või sündmust, mis on oluline käsitletavas protsessis. Olem esindab objektide klassi, mis on reaalses maailmas esindatud ja mida saab klassifitseerida nende parameetrite või karakteristika alusel.

Seos – kirjeldab olemite vahelisi seoseid ja grammatiliselt väljendub verbina, näiteks: omab, kuulub, on. Seoseid liigitatakse selle järgi, mitut objekti ühest ja teisest olemist seotakse ehk võimalikud variandid on üks-ühele (1:1), üks-mitmele (1:M) ja mitu-mitmele (M:M) seosed. Normaliseeritud ER mudelis ei kasutata M:M seoseid, mis on lahendatud kahe olemi vahele seatud kolmanda ühendava olemi abil.

Atribuut – kirjeldab olemi karakteristikat ja parameetreid näiteks: toote number, nimetus, pilt ja kategooria. Atribuudid jagunevad järgnevalt: primaarne võti, kohustuslik, mittekohustuslik ja võõrvõti. Antud näites toote number on primaarne võti, nimetus on kohustuslik, pilt on mittekohustuslik ja kategooria on võõrvõti, mis viitab kategooriate olemile.

ER mudeli olulisim kontseptsioon on normaliseerimine, mille käigus määratakse olemitele atribuudid sellisel viisil, mis vähendab andmeliiasust ning väldib andmeanomaaliaid. Normaliseerimine pakub sobilikku arhitektuuri andmete sisestamise ja muutmise tarvis ning tugevat andmete integreeritust. Tavaliselt kasutatakse kolmandal normaalkujul ER mudeleid. Normaliseerimine on ka näiteks M:M seoste likvideerimise protsess.

3.2.2. Dimensiooniline mudel

Dimensiooniline modelleerimine on tehnika, millega visualiseeritakse andmemudelid mõõdikute komplektidena, mis kirjeldavad ärianalüüsi põhiaspekte. Selline mudel on kasulik andmete summeerimisel ja ümberpaigutamisel ning on fokuseeritud valdavalt numbrilistele andmetele nagu väärtused, kogused, kaalud, nõuded ja sündmused. Dimensioonilises mudelis kasutatakse mitmeid baasmõisteid, põhilised neist on: fakt, dimensioon ja mõõdik.

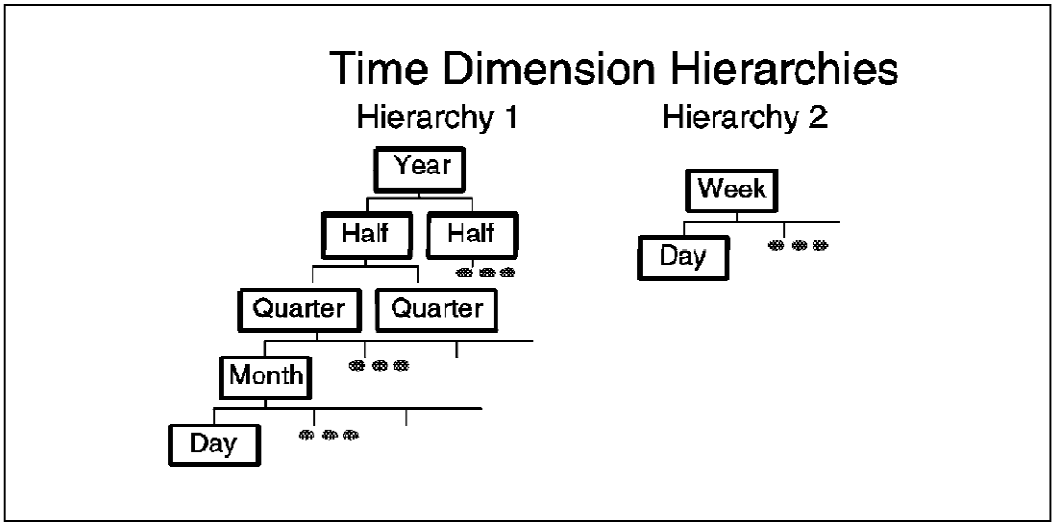
Fakt – on omavahel seotud andmeühikute kogum, sisaldades kirjeldavaid ja mõõdetavaid andmeid. Iga fakt tüüpiliselt esindab tehingut või sündmust, mida saab otseselt kasutada äriprotsessi analüüsimisel. Andmelaos on faktid kajastatud põhitabelitena, milles hoitakse kõiki numbrilisi andmeid.

Dimensioon – on kogumik samatüübilisi liikmeid või ühikuid. Diagrammis kujutatakse dimensiooni tavaliselt teljena ning iga andmeühik faktide tabelis on seotud ühe ja ainult ühe liikmega igast faktiga seotud dimensioonist. Seega dimensioonid kirjeldavad faktide kontekstilist tausta. Dimensioonid on parameetrid, mille abil saab sooritada OLAP (Online Analytical Processing) andmeanalüüsi. Näiteks müügitehingute analüüsimisel on vajalikud järgmised põhidimensioonid:

- Aeg
- Asukoht/regioon
- Klient
- Müügiisik

Dimensioon sisaldab hulka liikmed, kus iga liige väljendab fakti andmeühiku positsiooni. Näiteks kõik päevad, kuud, kvartalid ja aastad moodustavad aja dimensiooni ning kõik linnad, maakonnad ja riigid moodustavad geograafilise dimensiooni.

Dimensiooni liikmed on jagatud ühte või mitmesse hierarhiasse ning iga hierarhia koosneb ühest või mitmest tasandist. Hea näide dimensioonide hierarhiast on kujutatud joonisel 12, mis kujutab aja dimensiooni. Antud juhul kasutatakse kahte hierarhiat kuna nädalate põhist jaotust ei saa kirjeldada esimese hierarhia alamtasandina sest kuud ei koosne kunagi kindlast hulgast täisnädalatest.



Joonis 12. Aja dimensiooni hierarhiad ja tasandid

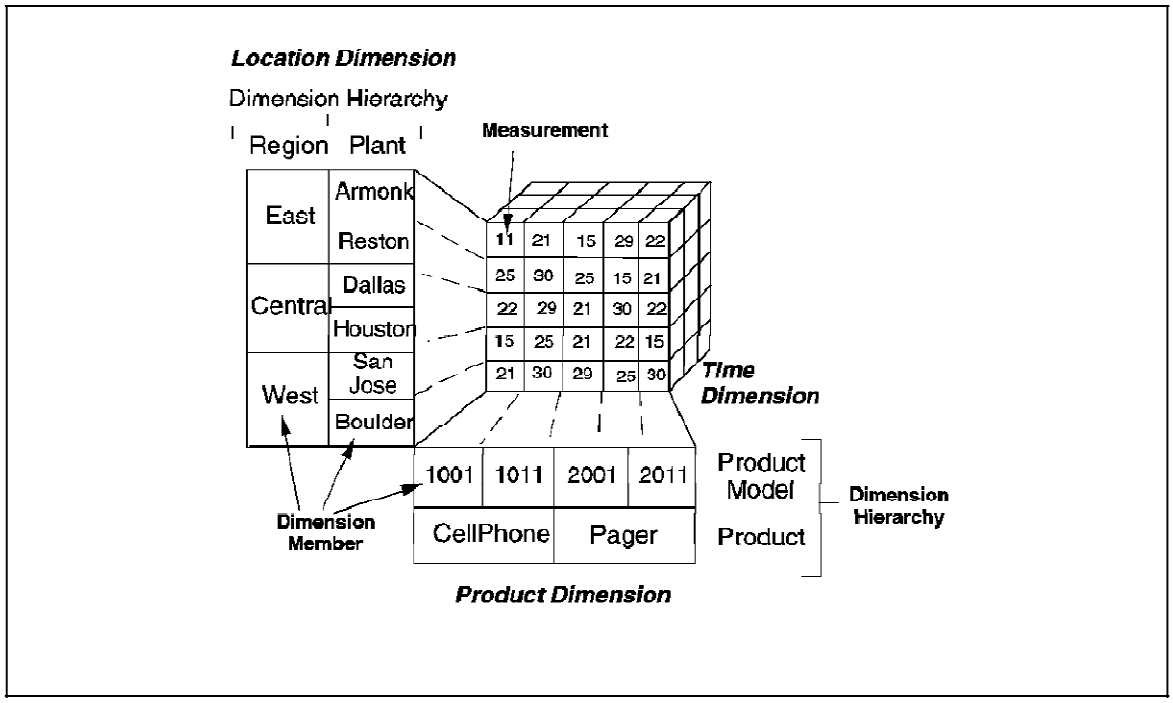
Allikas: Data Modeling Techniques for Data Warehousing [6:43]

Mõõdik – on fakti numbriline atribuut, kirjeldades fakti tehingu või sündmuse võimsust vastavalt dimensioonide kombinatsioonidele. Näiteks müügitehingu mõõdikuteks on müügisumma, müüdud kogus, kulu summa jne.

Dimensioonilise mudeli visualiseerimine

Populaarseim viis dimensioonilise mudeli visualiseerimiseks on kuubi kujutamine. Kuubi abil saab visualiseerida kolmedimensioonilist mudelit kuid tavaliselt sisaldab dimensiooniline mudel enamat kui kolme dimensiooni, mis viitab hüperkuubile. Kuna hüperkuupi on keeruline ette kujutada, kastutakse tavaliselt ikkagi kuubi mõistet.

Joonisel 13 on kujutatud tootmismahtu väljendav kuup, mis sisaldab mõõdikuid kolme dimensiooni kombinatsioonis: asukoht, toode ja aeg. Asukoha ja toote dimensioonid on omakorda kahetasandilise hierarhiaga.



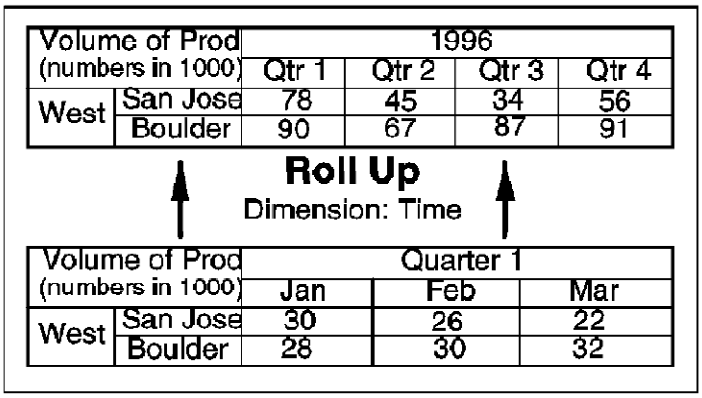
Joonis 13. Kuup – dimensioonilise mudeli metafoor

Allikas: Data Modeling Techniques for Data Warehousing [6:44]

3.2.3. OLAP põhioperatsioonid

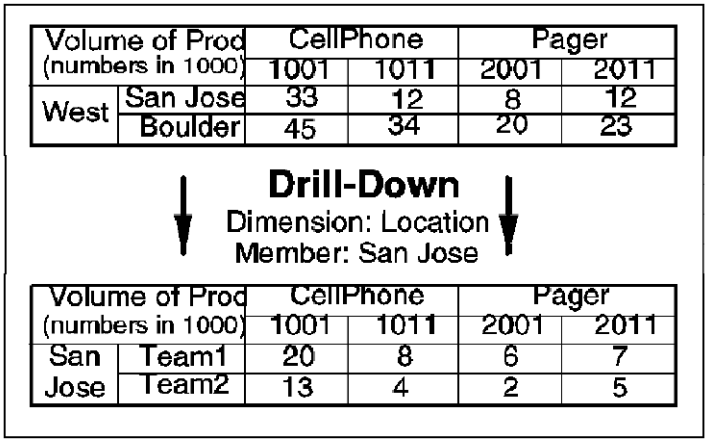
Dimensioonilise mudeli põhiülesanne on toetada OLAP ärianalüüsi, mille käigus teostatakse nelja tüüpi operatsioone. Dimensioonide hierarhiat kasutades puurimist ja kokku rullimist (drill down and roll up) ning dimensioonide kombinatsioone kasutades viilutamist ja pööramist (slice and dice).

Puurimine ja kokku rullimine – on tegevused, kus liigutakse dimensioonide hierarhia erinevate tasandite vahel. Puurimisel vaadatakse fakti sündmust detailsemalt ning kokku rullimisel summaarsemalt. Navigeeritavuse piirid määrab, mitmetasandiliselt on moodustatud dimensioonide hierarhia. Joonisel 14 on kujutatud fakti sündmuse esituse muutumine kokku rullimisel ning joonisel 15 sama puurimisel.



Joonis 14. Fakti sündmuse kokku rullimine

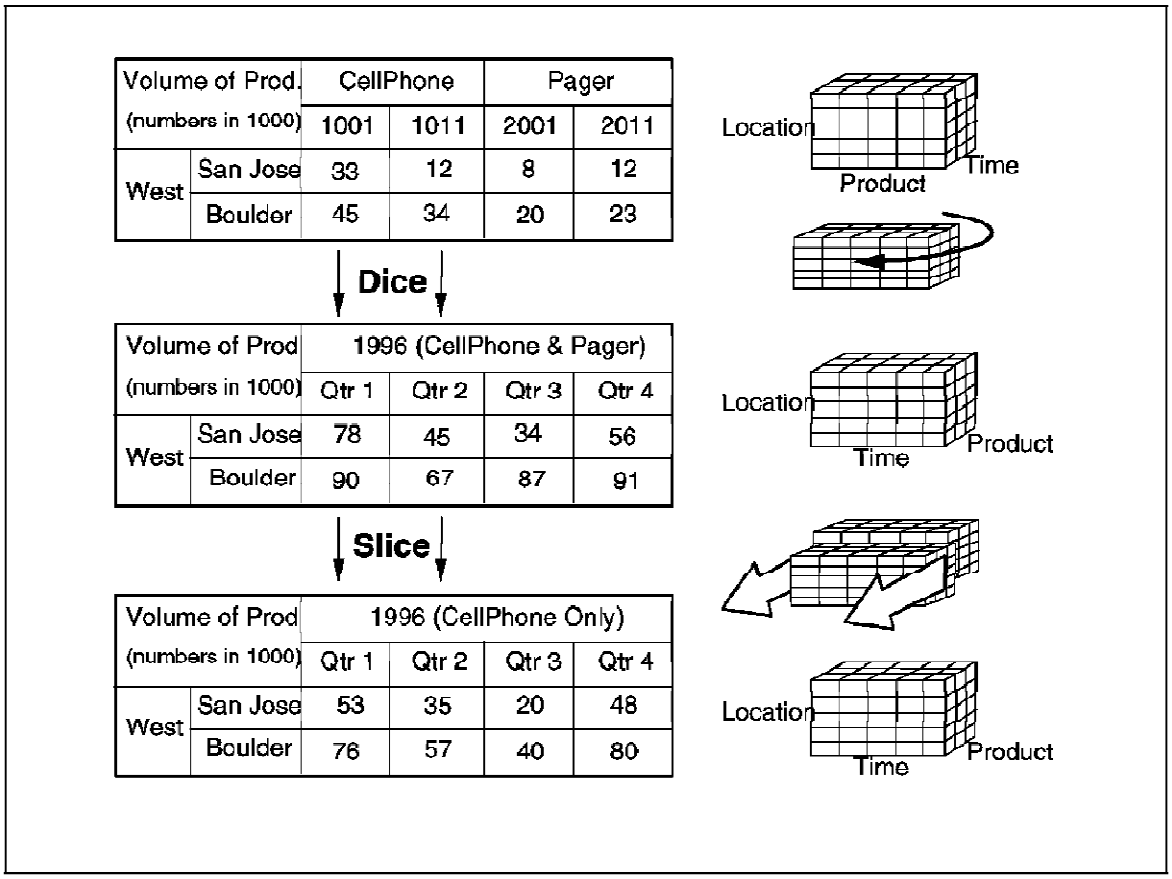
Allikas: Data Modeling Techniques for Data Warehousing [6:45]



Joonis 15. Fakti sündmuse puurimine

Allikas: Data Modeling Techniques for Data Warehousing [6:45]

Viilutamine ja pööramine – on andmete otsimine kuubist. Joonisel 16 kujutatud viilutamisel eraldatakse kuubist huvipakkuva dimensiooniväärtusega andmed ning pööramisel muudetakse perspektiivi nii, et lõpuks moodustub soovitud kahedimensiooniline analüüsivaade.



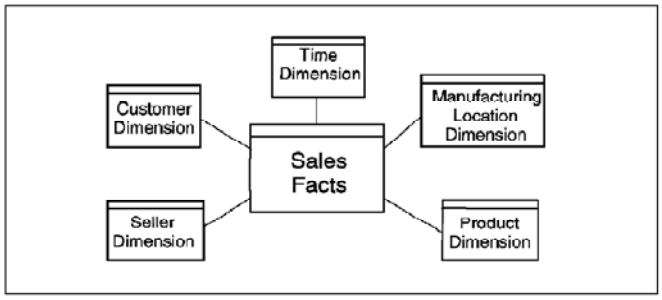
Joonis 16. Viilutamine ja pööramine
Allikas: Data Modeling Techniques for Data Warehousing [6:46]

Kirjeldatud põhioperatsioonid on hädavajalikud, teostamaks andmeanalüüsi ning seda tüüpi operatsioonide teostamiseks peavad andmed olema salvestatud spetsiifilisel viisil, mida nimetatakse dimensiooniliseks andmemudeliks.

3.2.4. Täht- ja lumehelbe mudel

Dimensioonilise mudeli puhul kasutatakse kahte põhilist andmemudeli tüüpi, mis on tähtmudel ja lumehelbe mudel. Vahest kasutatakse veel ka termineid tähtkuju- või mitmik-täht mudel, mis on täht- ja lumehelbe mudeli laiendused.

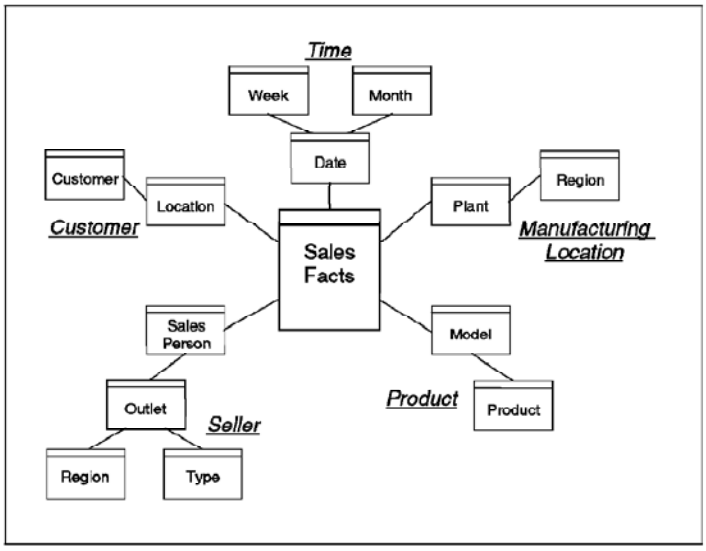
Tähtmudel – on põhiline termin, mida kasutatakse, rääkides dimensioonilisest mudelist. Selle mudeli nime on tulnud sellest, et loodud struktuur näeb välja nagu täht ning loogiline diagramm peegeldab füüsilist andmebaasi skeemi. Joonisel 17 kujutatud tähtmudelil on tüüpiliselt suur kesktabel, fakti tabel ning komplekt juurdekuuluvaid väiksemaid tabeleid, dimensioonide tabelid, mis on paigutatud ringis ümber fakti tabeli.



Joonis 17. Tähtmudel

Allikas: Data Modeling Techniques for Data Warehousing [6:47]

Lumehelbe mudel – on tähtmudeli edasiarendus, kus mitmetasandilise hierarhiaga dimensioonid on jagatud omavahel 1:M seoses olevateks dimensioonide tabeliteks. Joonisel 18 kujutatud lumehelbe mudel visualiseerib väga hästi dimensioonide hierarhiat ning on sobivaks aluseks füüsilise andmemudeli loomisel kuna peegeldab väga hästi andmebaasi füüsilist struktuuri. Samuti on lumehelbe mudel dimensioonide osas normaliseeritud ning välistab seega andmeliiasuse tekkimist ning sellest tulenevaid probleeme.



Joonis 18. Lumehelbe mudel

Allikas: Data Modeling Techniques for Data Warehousing [6:48]

3.2.5. C/SIDE hübriidne andmemudel

Majandustarkvara Microsoft Dynamic NAV arenduskeskkond C/SIDE (Client/Server Integrated Development Environment) kasutab loogilise andmebaasi arhitektiuurina teatavat ER ja dimensioonilise mudeli kombinatsiooni, kus primaarseks on ER mudel, mida saab laiendada dimensioonilise funktsionaalsusega ehk ER olemit saab käsitleda kui OLAP fakti ja defineerida, milliseid võõrvõtmeid kasutatakse fakti dimensioonidena. Selline hübriidne struktuur ei sobi kindlasti kõigi andmeanalüüsi vajaduste jaoks kuid katab reeglina ära igapäevase operatiivaruandluse vajadused.

Spetsiaalne andmetüüp – Flow Field

FlowField andmetüüp on väga võimas iseärasus *C/SIDE* andmebaasis. *FlowField* on peamine kontseptsioon mis tugevalt mõjutab *C/SIDE* aplikatsiooni loomise viisi. *FlowField* ja selle aluseks olev *SumIndexFields* kontseptsioon on loodud selleks et suurendada arvutuste võimsust summaarsete andmete arvutamisel, näiteks pearaamatu konto saldo arvutamisel. Tavapärases andmebaasis tuleb selliste tulemuste saamiseks teostada mitmeid erinevaid operatsioone. [5]

FlowField väärtus ei ole permanentne osa tabeli andmetest, seda võib käsitleda kui virtuaalset välja. *FlowField* väljade väärtused arvutatakse jooksvalt töö käigus ja salvestatakse spetsiaalsetes abitabelites. Vaikimisi on *FlowField* välja väärtus 0, see arvutatakse uuesti või loetakse abitabelist *C/AL* funktsiooni *CALCFIELDS* kasutamisel, automaatselt arvutatakse väärtus siis kui *FlowField* välja kasutatakse vormi või raporti juhtelemendi kaudu. *FlowField* välju on seitset tüüpi, tabel 4 iseloomustab erinevate tüüpide kasutatavust.

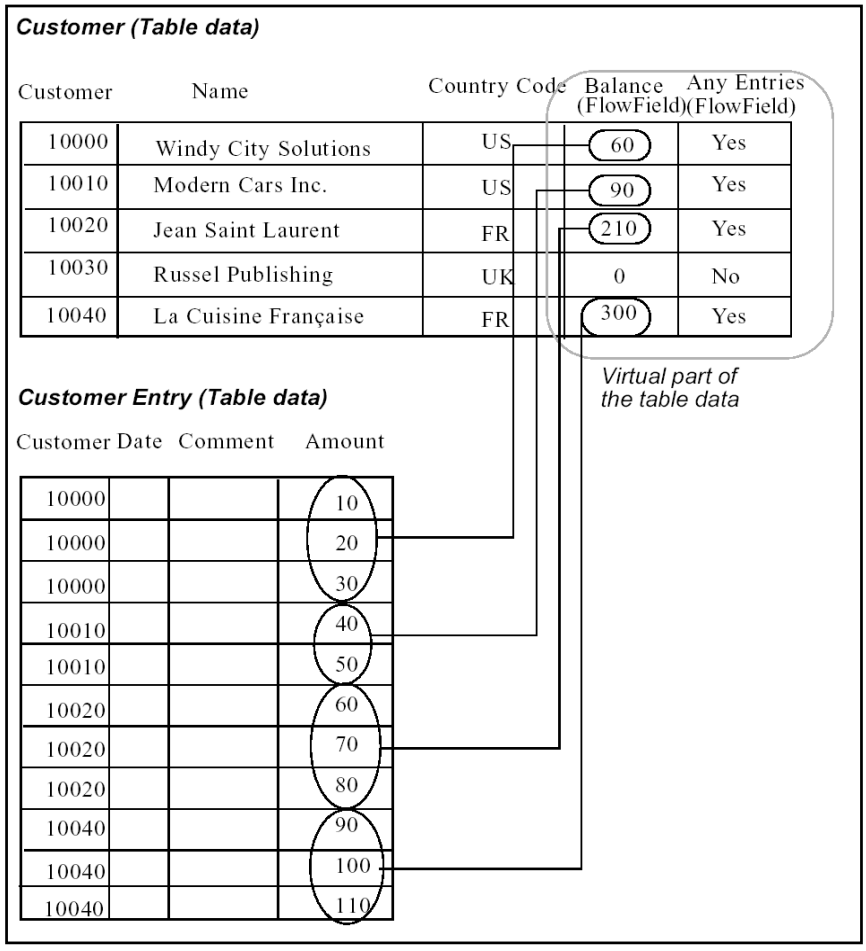
Tabel 4. FlowField välja tüübid

FlowField välja tüüp	Andmetüüp	Kirjeldus
Sum	decimal	määratud tingimustel kirjete summaarne väärtus
Average	decimal	määratud tingimustel kirjete keskmine väärtus
Exist	boolean	määratud tingimustel kirjete eksisteerimine
Count	integer	määratud tingimustel kirjete hulk
Min	suvaline	määratud tingimustel kirjete minimaalne väärtus
Max	suvaline	määratud tingimustel kirjete maksimaalne väärtus
Lookup	suvaline	määratud tingimustel kirje väärtus

***FlowField* kasutamise näide**

Customer tabelis on kaks *FlowField* välja. Väli *Any Entries* on *Exist* tüüpi ja *Balance* on *Sum* tüüpi. Alltoodud joonis 19 näitab et kliendi number 10000 saldo arvutatakse *Customer Entry* tabelist *Amount* välja pealt summeerides selliseid kirjeid kus *Customer* välja väärtus on 10000. Kliendi number 10030 saldo on 0 ja *Any Entries* väärtus on *No (false)* kuna sellise kliendi numbriga pole *Customer Entry* tabelis ühtegi kirjet. FlowField väärtuse arvutamise tingimused määratakse arvutuse valemiga, näiteks *Balance* välja arvutusvalem on järgmine:

Sum("Customer Entries".Amount WHERE(CustNo=FIELD(CustNo)))



Joonis 19. FlowField välja väärtuse arvutamine

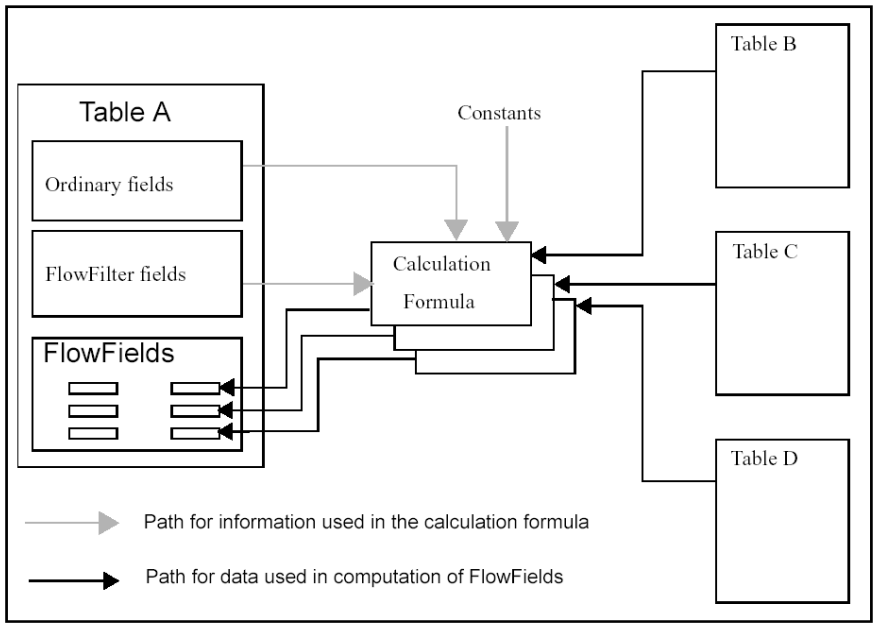
Allikas: Application Designer's Guide [5:43]

FlowField välja loomine toimub analoogselt tavaliste väljade loomisele, esmalt tuleb luua tavaline väli sisestades number ja nimi ning valides andmetüüp, seejärel tuleb avada parameetrite aken *View* menüüst *Properties kus FieldClass* parameetri väärtuseks tuleb

Normal asemel valida *FlowField* ja seejärel sisestada arvutuse valem *CalcFormula* parameetri väärtuseks. Arvutuse valemit saab sisestada assisteerija kaudu vajutades *CalcFormula* parameetri väärtuse lahtris *Assist-Edit* nuppu. Assisteerijas *Table Filter* parameetri väärtuse sisestamiseks saab omakorda kasutada assisteerijat. Lisaks staatilistele tingimustele saab *Table Filter* parameetris kasutada *FlowFilter* väljade väärtuseid.

Spetsiaalne andmetüüp – Flow Filter

FlowFilter andmetüüp on loodud spetsiaalselt *FlowField* andmetüübi kasutusvõimaluste laiendamiseks. *FlowFilter* tüüpi väli *C/SIDE* andmebaasi tabelis on samuti virtuaalne väli mille väärtuse saab kasutaja töö käigus ise sisestada. *FlowFilter* väärtus sisestatakse kasutatavale tabelile mitte igale väljale eraldi. Tüüpiliselt kasutatakse näiteks *FlowFilter* tüüpi väljana pearaamatu kontode tabelis välja *Kuupäevafilter*. *FlowField* väljad on näiteks *Saldo* ja *Muutus* mida arvutatakse pearaamatu kannete tabelist, kusjuures lisaks staatilistele *Table Filter* kirjeldustele on kirjeldatud ka filtreerimise tingimus vastavalt *FlowFilter* välja *Kuupäevafilter* väärtusele. Tulemusena saab kasutaja töö käigus sisestada kuupäevafiltri väärtuseks soovitud kuupäeva või kuupäevade vahemiku ning näeb kohe näha perioodi saldot ja muutust. [5]



Joonis 20. FlowFilter välja väärtuse kasutamine FlowField välja väärtuse arvutamisel
Allikas: Application Designer's Guide [5:48]

Ülaltoodud joonis 20 illustreerib seoseid erinevat tüüpi andmebaasi väljade ja arvutuse valemi vahel. Arvutuse valemis defineeritud filtrid võivad sisaldada väärtusi tavalistelt väljadelt ja parameetreid *FlowFilter* väljadelt. *FlowFilter* väljadele saab kasutaja töö käigus *C/SIDE* aplikatsioonis sisestada väärtusi mis määravad *FlowFilter* väärtuste kalkuleerimist.

3.2.6. ER mudel ja dimensiooniline mudel

Kirjeldatud kaks mudelit paistavad olemuselt ja struktuurilt väga erinevad kuid samas on neil ka sarnasused. Dimensioonilise mudeli puhul saab kasutada ER mudeliga sama notatsiooni: olem, seos, atribuut ja primaarne võti. Üldjoontes võib öelda, et fakt on olem, mille primaarne võti on võõrvõtmete kombinatsioon ja võõrvõtmed vastavad dimensioonidele. Selliselt loodud dimensioonilist mudelit saab nimetada ER mudelist tuletatud esituseks kuid reaalselt tekib siin erinevaid probleeme.

Dimensioonide seisukohast on probleemiks aja dimensiooni esitus, mis ER mudelis ei ole kunagi kajastatud võõrvõtmena kuid igas dimensioonilises mudelis on alati vajalik selle dimensiooni olemasolu eelnevalt kirjeldatud klassikalise hierarhia kujul.

Andmeanalüüsi seisukohast on probleemiks see, et ER mudeli olemid on tavaliselt liialt detailsed, näiteks majandustarkvara Microsoft Dynamics NAV olemi mudelis on müügiarved ja müügi kreeditarved eraldi olemid kuigi struktuurilt väga sarnased, samas dimensioonilise mudeli seisukohast huvitab meid müügitehingute fakt, kus ühe dimensioonina on kajastatud see, kas tegu on deebet- või kreeditarvega. Teine võimalus oleks käsitleda kogu ER andmemudelit kui ühte olemit ning moodustada sellest üks dimensiooniline mudel, lugematute dimensioonide hulgaga kuid sellise mudeli baasil andmeanalüüsi tegemise on liialt keerukas või pakub liialt eksistentsiaalset ja üldistatud tulemit.

Seega järeldub, et praktiliselt võimatu on leida kompromissi ER mudeli ja dimensioonilise mudeli detailsuse vahel ning igal juhul on vajalik kahe andmemudeli olemasolu: ER mudel andmete sisestamiseks ning dimensiooniline mudel andmete analüüsimiseks. Klassikaliste tarkvaraarenduse metoodikate probleem on aga selles, et neid kahte andmemudelit käsitletaksegi kui täiesti eraldiseisvaid ja paralleelselt või järjestikuselt arendatavaid. Sealjuures arendatakse esmalt ER mudel ja seejärel selle denormaliseerimise teel dimensiooniline mudel.

4. OLAP KUUBIL RAJANEV ARENDUSMEETOD

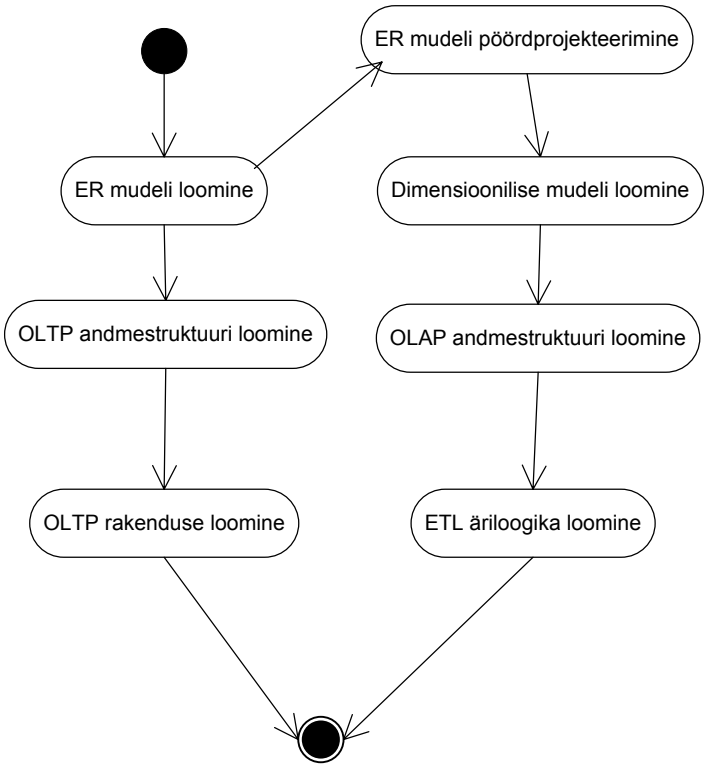
Eelnevates peatükkides on käsitletud tarkvara arendus- ja juurutusprotsessis erinevatel etappidel kasutatavaid metoodikaid, mis katavad kõik olulised tegevused, alates süsteemi loomise ideest kuni valmis realisatsiooni kasutamiseni. Eelneva põhjal võib välja tuua järgmised olulised järeldused:

- Majandustarkvara juurutusmetoodikad on väga sarnased spetsiaaltarkvara arendusmetoodikatele ning on olulises osas kattuvad
- Majandustarkvara juurutusmetoodikad ei paku juurutusmudelit, mis võimaldaks kogu juurutusprotsessi kõrgemal abstraktsiooni tasemel modelleerida
- Spetsiaaltarkvara arendusmetoodikaid kasutatakse koos UML mudelitega kuid süsteemi terviklik modelleerimine kõrgemal abstraktsiooni tasemel on liialt keerukas ehk võimatu
- Infosüsteemide arhitektuur jaguneb kaheks osaks, mis on omavahel pehmes seoses
    - Loogiline arhitektuur – planeerimine, analüüs, kavandamine
    - Tehniline arhitektuur – andmestruktuuri loomine, rakenduse arendamine
- Andmestruktuuri loomisel tõlgendatakse denormaliseeritud talitusprotsessi mudelist normaliseeritud andmestruktuur, mis vastab andmete sisestamise vajadustele
- Andmeanalüüsi tarvis hiljem pöördprojekteeritakse normaliseeritud andmestruktuurist denormaliseeritud mudelid, mille alusel koostatakse dimensioonilised andmemudelid
- Klassikaliselt realiseeritud andmelaod on staatilised, kohmakad ja andmekvaliteedi probleemidega kuna andmelao ja operatiivsüsteemi arendus ei ole sünkroonis

Käesoleva peatüki eesmärk on välja töötada süsteemi tehnilise arhitektuuri osana, olemasolevatest andmemudelitest üks abstraktsiooni tase madalamal asuv metamudel, mis sisaldab seotuna transaktsioonilise ja dimensioonilise mudeli metaandmeid, võimaldades luua dünaamilise andmeanalüüsi süsteemi, kus kasutaja poolt defineeritud OLAP vaade luuakse perioodiliselt operatiivsüsteemi andmete baasil.

Joonisel 21 on kujutatud tegevused, mida teostatakse klassikalise andmelaoga infosüsteemi arendamisel. Põhiliselt tasuks selle meetodi puhul tähelepanu pöörata järgmistele tegevustele, milles seisneb antud meetodi nõrkus:

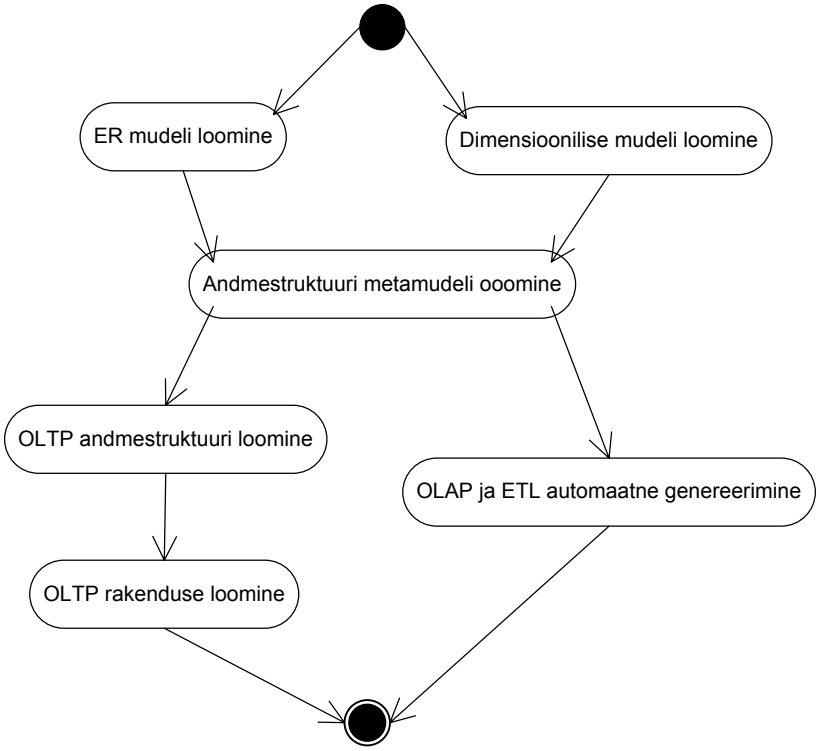
- Kuna arendust teostatakse kahes eraldiseisvas etapis, moodustatakse esmalt protsessi mudelist ER mudel ja teises etapis pöördprojekteeritakse ER mudelist uuesti protsessi mudeli analoog, mille põhjal luuakse dimensiooniline mudel
- Tulenevalt arendusprotsessi etapilisusest, esineb kiiresti ja dünaamiliselt arendatava transaktsioonilise süsteemi puhul alati OLAP ja ETL (Extract, Transform, Load) mahajäämus ning sellest tulenevad andmekvaliteedi probleemid OLAP andmebaasis



Joonis 21. Klassikalise andmelaoga infosüsteemi arendusmeetod

Alternatiivse meetodina on joonisel 22 kujutatud uudse lähenemisega arendusmeetod (OLAP kuubil rajanev infosüsteemi arendusmeetod), kus kogu arendusprotsess toimub ühes seotud etapis ja tulemusena on loodud virtuaalne andmeladu. Põhiliste erinevustena, võrreldes klassikalise meetodiga võib välja tuua järgmist:

- Protsessi mudelist moodustatakse korraga ER mudel ja dimensiooniline mudel, mis kirjeldatakse süsteemi ühtse metamudelina
- Ühtse metamudeli metaandmete baasil saab universaalse OLAP ja ETL rakenduse abil moodustada virtuaalse andmelao, mis on alati täieliku andmekvaliteediga, kuna reaalselt on infosüsteemi andmed salvestatud vaid ühes, transaktsioonilises andmebaasis.



Joonis 22. Dünaamilise andmelaoga infosüsteemi arendusmeetod

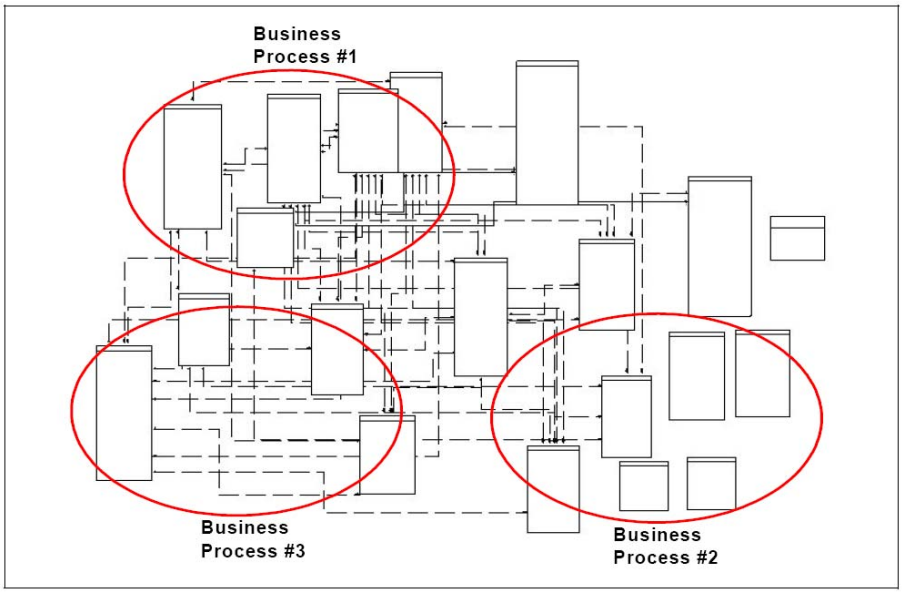
Virtuaalse andmelao piirangud, võrreldes klassikalise andmelaoga:

- Ei paku mineviku seisu taastamise funktsionaalsust
- Kõik OLTP andmebaasid peavad olema reaalajas ligipääsetavad
- Keerukate analüütiliste vaadete poolt põhjustatav suur koormus OLTP andmebaasile
- OLAP põhitegevuste nagu puurimine, kokku rullimine, viilutamine ja pööramine poolt põhjustatav suur koormus OLTP andmebaasile

4.1. ER mudelist dimensioonilise mudeli loomine

Metoodika kirjeldamisel on kasutatud juhendmaterjali [7]

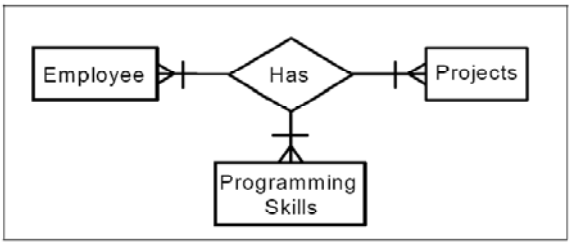
Kuna käsitletavas süsteemis on seni loodud vaid ER (Entity - Relationship) andmemudelid siis esmalt tuleb klassikaliselt lähenedes moodustada nendest sobivad dimensioonilised mudelid. Siinkohal on kõige olulisem jälgida, et moodustataks mõistlik hulk OLAP kuupe, mis katab andmeanalüüsi vajadused. Kuna OLTP süsteem tervikuna katab tavaliselt mitmeid äriprotsesse, tuleb esmalt identifitseerida protsessid ja nendega seotud tabelid ER mudelis. Joonisel 23 on piltlikult kujutatud ER mudeli jagunemine äriprotsessideks, samuti on tavapärane, et erinevad äriprotsessid kasutavad mõnd ühist andmeobjekti ER mudelis.



Joonis 23. ER mudeli jagunemine äriprotsessideks
Allikas: Dimensional Modeling: In a Business Intelligence Environment [7:209]

Peale äriprotsesside identifitseerimist on järgmise sammuna vaja leida algsed M:M seosed ER mudelist, mis normaliseerimise käigus on lahti kirjutatud 1:M seosteks. Eelnevas peatükis kirjeldatud M:M seose normaliseerimisel moodustati tavaliselt lisatabel, mis väljendas tegevust ehk kui näiteks denormaliseerimata müügisüsteemis olid objektid: klient ja kaup omavahel M:M seoses siis moodustati nende vahele tehingute objekt, mis on mõlema eelneva objektiga 1:M seoses ning väljendab tegevust, sisaldades tavaliselt numbrilisi andmeid. Sellised tegevusi väljendavad tabelid kujunevadki tavaliselt OLAP kuubi keskseks faktiks. Joonisel 24 on kujutatud näitena M:M seoses olevad objektid, kus töötajad omavad erinevaid

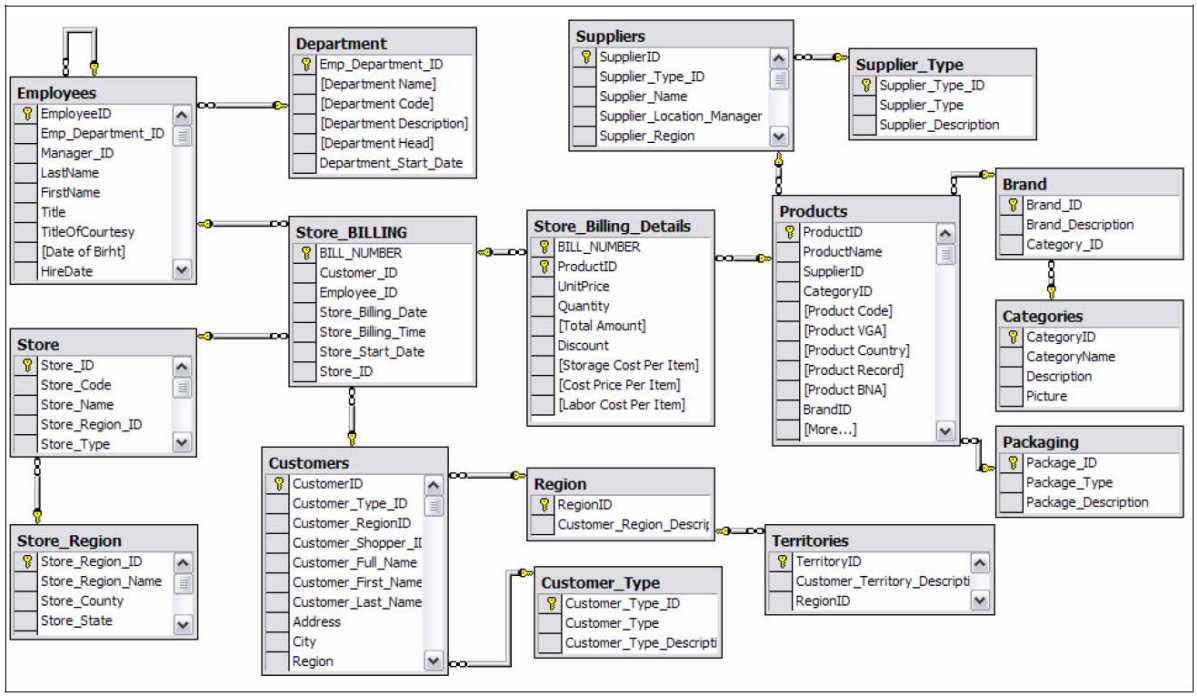
kompetentse ja osalevad mitmetes projektides ning iga projekti juurde kuulub hulk töötajaid ning vajalikud kompetentsid.



Joonis 24. Mitu-mitmele seosed

Allikas: Dimensional Modeling: In a Business Intelligence Environment [7:211]

Peale tegevuste tabelitest OLAP faktide moodustamist tuleb ER mudeli püsiandmete ja klassifikaatorite tabelitest moodustada OLAP dimensioonid. Viimasena tuleb ER mudelist leida kuupäeva sisaldaval väljad ning määratleda need OLAP aja dimensioonina. Eelmises peatükis oli juttu, et OLAP kuubis on alati üheks põhidimensiooniks aeg, mis on jagatud standardsesse hierarhiasse, samas ER mudelis ei eksisteeri aja klassifikaatorit vaid lihtsalt tehingute tabelites kuupäevade väljad. Näiteks moodustame joonisel 25 kujutatud ER mudelist dimensioonilise mudeli.



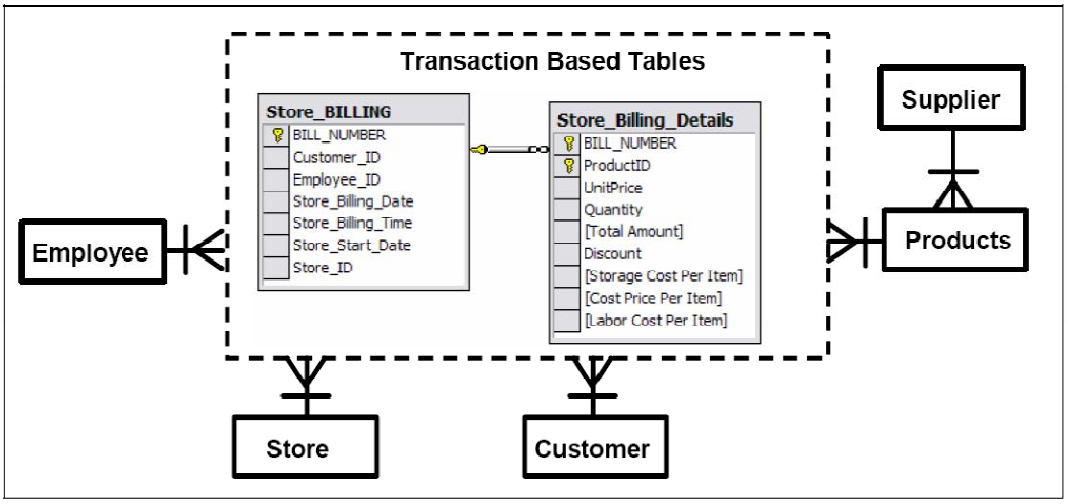
Joonis 25. Jaemüügi süsteemi ER mudel

Allikas: Dimensional Modeling: In a Business Intelligence Environment [7:212]

4.1.1. Äriprotsessi identifitseerimine

Esmalt identifitseerime tehingutega seotud tabelid ja püsiandmete ning klassifikaatorite tabelid. Joonisel 26 on kujutatud tehingutega seotud Store_BILLING ja Store_Billing_Details tabelid, mis väljendavad M:M seoseid töötajate, kaupluste, klientide ja toodete vahel.

- Iga töötaja müüb erinevaid tooteid ja iga toodet müüvad erinevad töötajad
- Iga kauplus müüb erinevaid tooteid ja iga toodet müüakse erinevates kauplustes
- Iga klient ostab erinevaid tooteid ja iga toodet ostavad erinevad kliendid
- Iga töötaja müüb erinevatele klientidele ja iga klient ostab erinevatelt müüjatelt

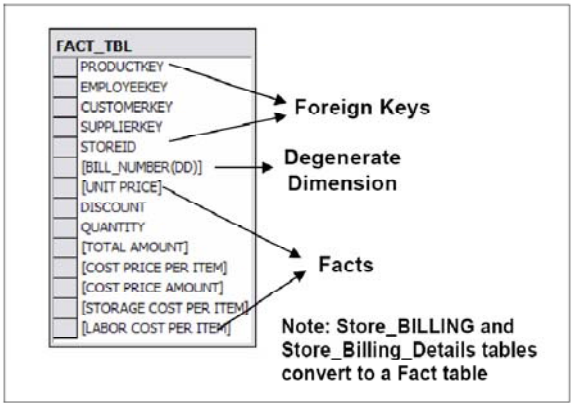


Joonis 26. Mitu-mitmele seosed ER mudelis

Allikas: Dimensional Modeling: In a Business Intelligence Environment [7:216]

4.1.2. Fakti tabeli moodustamine

Järgnevalt tuleb leitud tehingute tabelid denormaliseerimise teel konverteerida üheks lamedaks fakti tabeliks. Joonisel 27 on kujutaud loodud fakti tabel, mis sisuliselt sisaldab iga tehingu rea juures ka tehingu päise andmeid ehk on teatava andmeliiasusega. Samuti on näha, millised väljad loodud tabelis on seosed dimensioonidega ning millised on fakti ennast kirjeldavad väljad. Huvitav on siin jälgida, mis on saanud arve numbri väljast, mis algses tehingute päise tabelis on primaarne võti kuid siinkohal enam seda ei ole. Nimetame seda degenereerunud dimensiooniks, mis tähendab dimensiooni ilma atribuutideta ehk dimensioon mis on moodustunud tehingu numbrist algsete 1:M seoses olnud faktide tabelite denormaliseerimise tagajärjel.

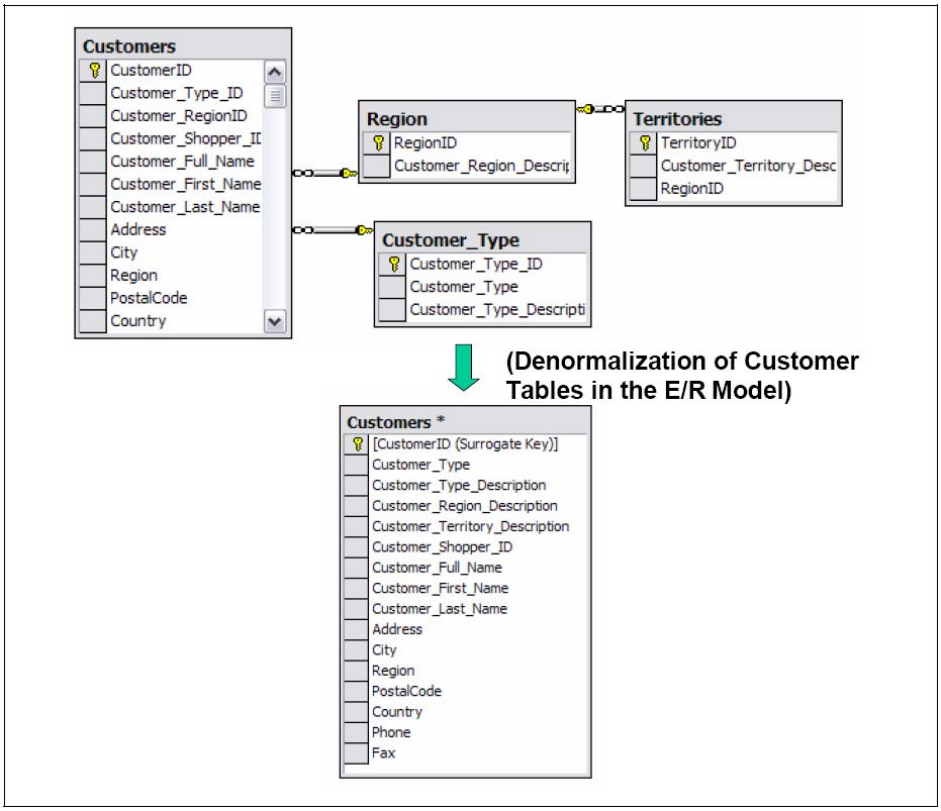


Joonis 27. Fakti tabel

Allikas: Dimensional Modeling: In a Business Intelligence Environment [7:216]

4.1.3. Dimensioonide tabelite moodustamine

Järgmise sammuna tuleb järelejäänud tabelid denormaliseerida lamedateks dimensioonide tabeliteks. Näitena on joonisel 28 on kujutatud tabelite: kliendid, regioonid, territooriumid ja kliendi tüübid; denormaliseerimine lamedaks klientide dimensiooni tabeliks.

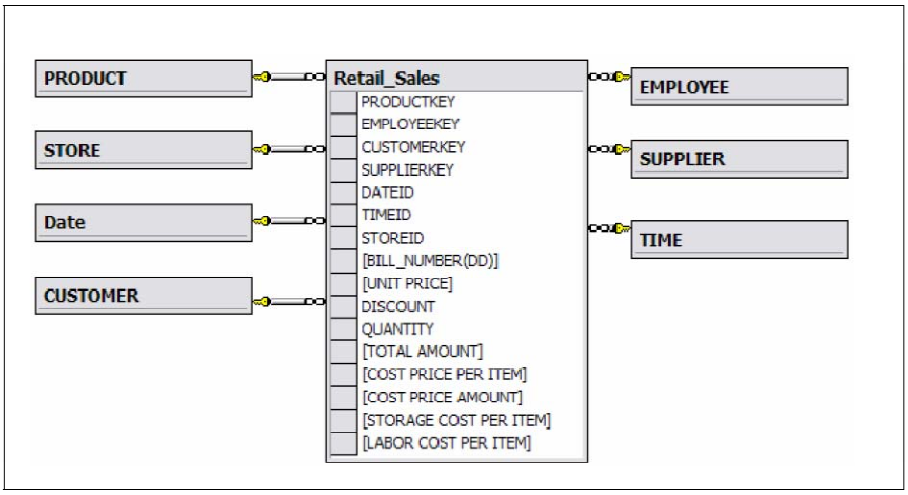


Joonis 28. Kliendi dimensiooni tabel

Allikas: Dimensional Modeling: In a Business Intelligence Environment [7:218]

4.1.4. Kuupäeva ja kellaaja dimensioonide moodustamine

Viimase sammuna tuleb identifitseerida kuupäeva ja kellaaja dimensioonid. Antud näite puhul on tabelis STORE_Billing väljad Store_Billing_Date ja Store_Billing_Time, mis on lihtsalt kuupäeva ja kellaaja tüüpi väljad ehk erinevalt teistest dimensioonidest ei ole võõrvõtmed. Nende väljade alusel tuleb moodustada kuupäeva ja kellaaja dimensioonid ning lõplik dimensiooniline mudel on kujutatud joonisel 29.



Joonis 29. Lõplik dimensiooniline mudel

Allikas: Dimensional Modeling: In a Business Intelligence Environment [7:222]

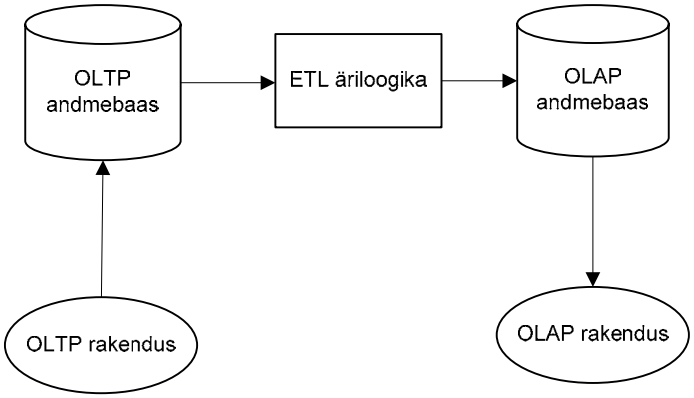
4.2. Virtuaalne andmeladu

Eelnevates peatükkides ja alapunktides on kirjeldatud infosüsteemide olemust ja nende arendamise metoodikaid, erinevaid andmemudeleid, nende eripärasid ja kasutatavust ning kõik see kokku moodustab paremini või halvemini toimiva infosüsteemi. Samuti selgus, et infosüsteemi loogilise arhitektuuri tasandil, näiteks ülesandepüstitusi ja protsesside kirjeldusi on suhteliselt keerukas formuleerida UML (Unified Modeling Language) märgisüsteemi poolt pakutaval metatasandil, küll aga on süsteemi tehnilise arhitektuuri osal olemas väga selgepiiriline metatasand, näiteks andmemudelite tabelite struktuuride näol. Samuti jõudsime arusaamani, et infosüsteem vajab reeglina kahte andmestruktuuri, ühte OLTP (On-Line Transaction Processing) ja teist OLAP (On-Line Analytical Processing) tarvis. Paraku kui need kaks andmestruktuuri on tehnilisel tasandil teineteisest sõltumatud, kaasneb sellega mitmeid probleeme:

- Süsteemid ei ole tavaliselt hästi dokumenteeritud ja arendaja peab juhinduma andmebaasi struktuurist, mõistmaks millised andmed, kus asuvad
- Ettevõttes võib olla kasutusel rohkem kui üks OLTP süsteem, millega kaasneb veel mitmeid teineteisest tehnilisel tasandil sõltumatuid andmestruktuure
- Tihtipeale ei ole OLTP süsteemi tabelite ja väljade nimed tähenduslikud, mistõttu ei ole lõppkasutaja suuteline nendes orienteeruma ja sobivaid päringuid kirjeldama

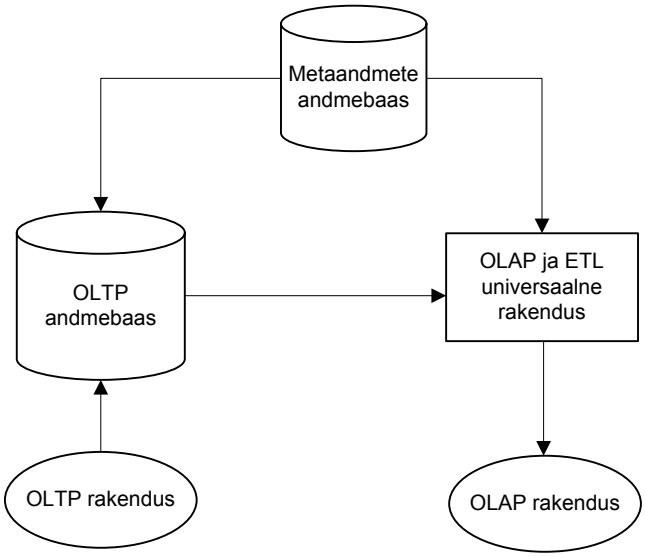
Käesoleva peatüki eesmärgiks on välja töötada virtuaalne andmelao lahendus, mis pakuks infosüsteemi lõppkasutajale funktsionaalsust, mis on väga sarnane klassikalisele andmelaole. Virtuaalse ja klassikalise andmelao põhiline erinevus tehnilisel tasandil seisneb selles, et kui klassikalise andmelao puhul eksisteerib andmebaas, mis sisaldab OLAP kuupide andmeid siis virtuaalse andmelao puhul eksisteerib andmelao struktuur metaandmetena ning päringutele vastuse leidmine toimub transaktsioonilisest andmebaasist. Antud ettevõtte kontekstis nimetame seda: „Öine aruandegeneraator".

Võrreldes virtuaalse andmelaoga, on klassikalise andmelao puhul tehnilise tasandi põhirõhk andmetabelite struktuuril, mis võimaldaks kasutajatel esitada soovitud päringuid. Kui klassikalist andmeladu hakatakse arendama andmelao tabelite loomisest ning ETL äriloogika arendamisest, mis on põhimõtteliselt võimalik sest metatasandi kirjeldamine ei ole kohustuslik analoogselt OLTP süsteemile siis jõutakse välja samade probleemideni, mida eelnevalt on kirjeldatud, kus andmelaos andmekvaliteet on madal ning esineb probleeme OLTP ja OLAP arenduse sünkroonsuses. Joonisel 30 on kujutatud klassikalise andmelaoga infosüsteemi põhikomponendid.



Joonis 30. Klassikalise andmelaoga infosüsteemi komponendid

Virtuaalse andmelao puhul on tehnilise tasandi põhirõhk andmelao struktuuri kirjeldamisel metatasandil, mis on rangelt seotud transaktsioonilise andmebaasi struktuuriga. Sisuliselt lisandub juurde üks abstraktsiooni tase madalamal asuv andmemudel, mis sisaldab kõigi ülejäänud andmemudelite metaandmeid ja ühendab need ühtseks süsteemiks, mida võiks nimetada: „Andmemudelite metaandmete repositoorium". Joonisel 31 on kujutatud virtuaalse andmelaoga infosüsteemi põhikomponendid.

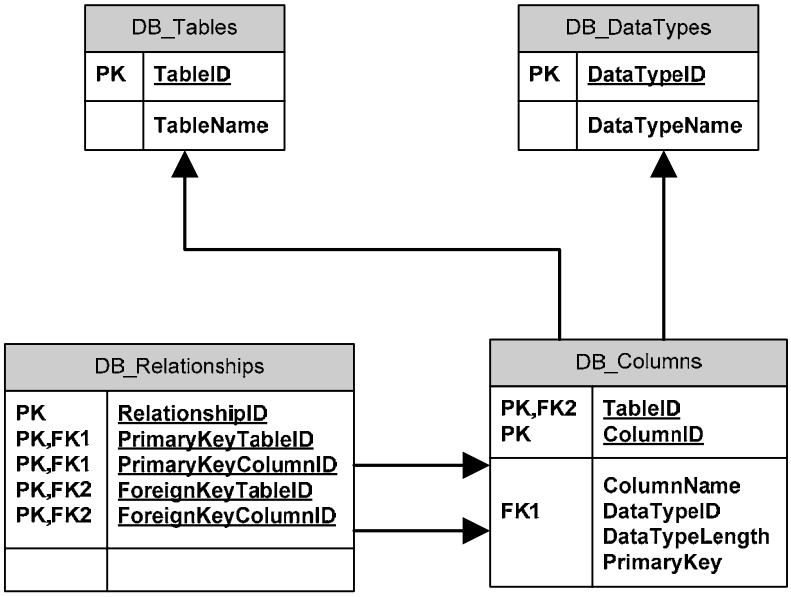


Joonis 31. OLAP kuubil rajaneva infosüsteemi komponendid

Selliselt loodud süsteem ei pea piirduma pelgalt virtuaalse andmelao võimalusega, mis on süsteemi poolt esmalt pakutav väärtus vaid kerge vaevaga saab edasi arendada ka juba klassikalise andmelao kuid seda juba täisautomaatselt genereerides andmelao andmetabelid koos ETL äriloogikaga.

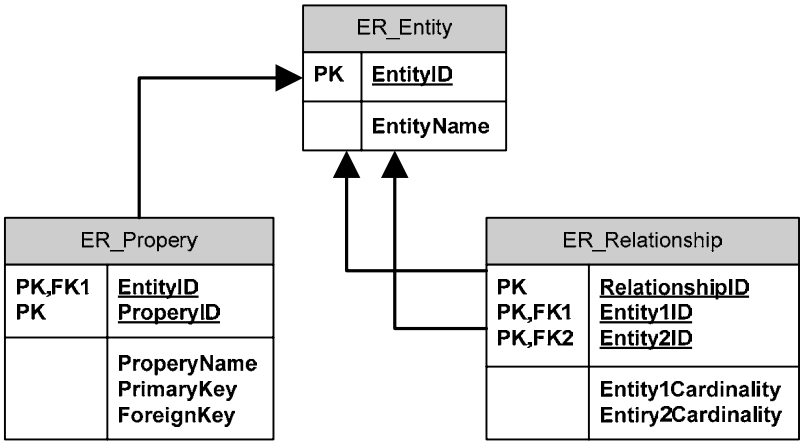
4.2.1. Virtuaalse andmelao metamudel

Erinevas kirjanduses käsitletakse nii transaktsioonilise kui dimensioonilise andmebaasi metamudeleid ning esitatakse neid kohati väga võimsate ja keerulistena, kujul millistena ei ole need praktikas otse realiseeritavad. Dünaamilise andmelao andmemudel peab ühendama nii transaktsioonilise kui dimensioonilise andmebaasi metamudelid. Alustades lihtsate põhiprintsiipide realiseerimisest, selgitan dünaamilise andmelao ideed ning arendan seda iteratiivselt, tuginedes käsitletava ettevõtte vajadustele. Joonisel 32 on kujutatud lihtsustatult relatsioonilise andmemudeli metamudel.



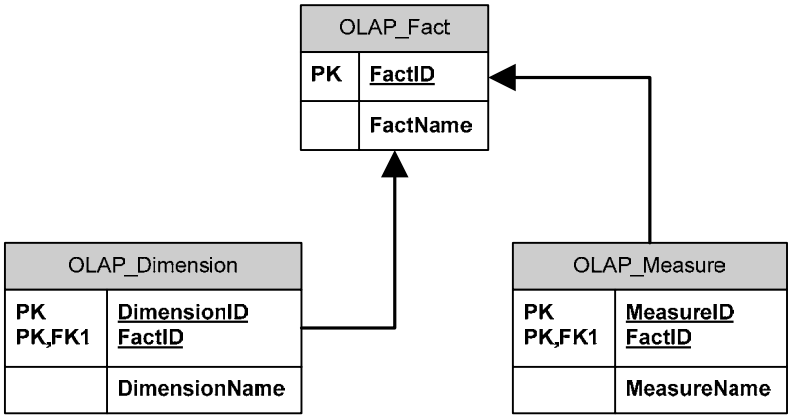
Joonis 32. Relatsioonilise andmemudeli metamudel

Transaktsioonilise ehk OLTP andmebaasi mudelit kujutatakse tavaliselt ER mudelina, kus eksisteerivad olem, seos ja atribuut. Joonisel 33 on kujutatud ER mudeli metamudel. Võrreldes ER metamudelit eelneva andmemudeli metamudeliga, näeme et ER mudel on veidi üldisem kuid suhteliselt selgelt tuleb välja, et ER olemile vastab andmemudelis tabel, ER atribuudile vastab andmemudelis väli ning ER seosele vastab andmemudeli seos.



Joonis 33. ER mudeli metamudel

Analüütilise ehk OLAP andmebaasi mudelit kujutatakse täht- või lumehelbe mudelina, kus eksisteerivad fakt, dimensioon ja mõõdik. Joonisel 34 on kujutatud OLAP tähtmudeli metamudel, mida eelnevatega võrreldes võib öelda, et fakt on olem, mille primaarne võti on võõrvõtmete kombinatsioon ja võõrvõtmed vastavad dimensioonidele.

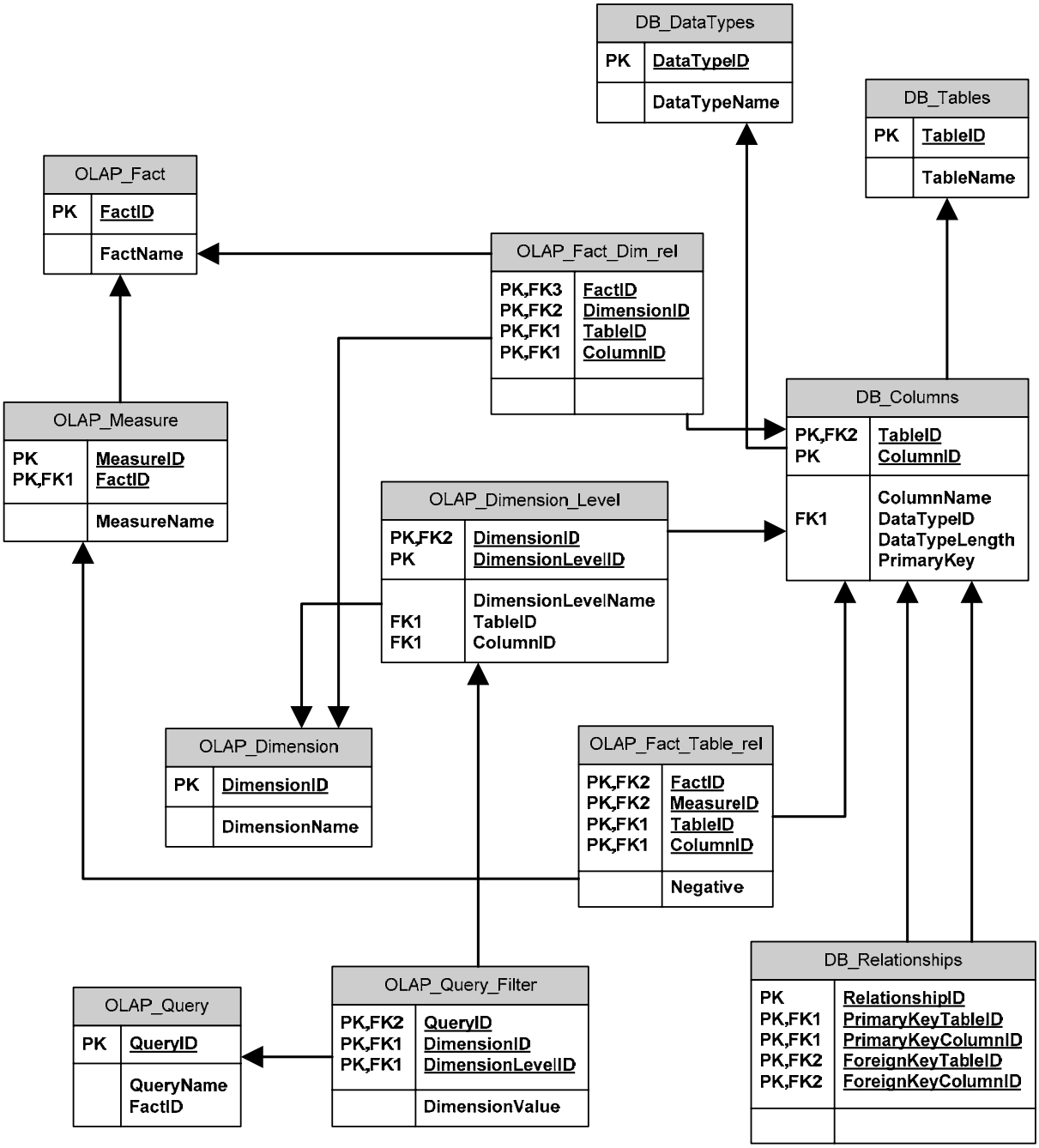


Joonis 34. OLAP tähtmudeli metamudel

Virtuaalse andmelao tarvis mudelit edasi arendades selgub eelnevast, et ER metamudeli võib vaatluse alt välja jätta kuna see kattub olemuslikult relatsioonilise andmebaasi metamudeliga seega dünaamilise andmelao mudelis on oluline luua seos OLAP metamudeli ja andmebaasi metamudeli vahel. Arvestades et andmebaasi metamudeli tabelid saab täita andmebaasi metaandmetega automaatselt, on mõistlik luua seos OLAP metamudeli poole peale. Kuna aga OLAP fakt võib olla andmebaasi tabeliga M:M seoses siis tuleb selle seose normaliseerimiseks luua uued objektid, mis seovad meid kahte metamudelit.

Järgneval joonisel 35 on kujutatud eelneva alusel loodud virtuaalse andmelao metamudel, mis sisaldab endas nii transaktsioonilise kui dimensioonilise andmemudeli metamudeleid ning neid ühendavaid metaseoseid. Kirjeldatud metamudel ja metaandmed on süsteemi tehnilise arhitektuuri lahutamatu osa ning kogu süsteemi edasine arendamine rajaneb sellele mudelile, kus paralleelselt transaktsioonilise andmemudeli arendamisele, lisatakse vajalikud metaandmed ka sellesse mudelisse ning tulemusena on kasutajate jaoks olemas virtuaalne andmeladu, mis on alati ajakohane ning olemuslikult ei sisalda probleemi andmekvaliteediga.

Sellise meetodi põhjal arendatud infosüsteemi teise etapina on võimalik luua universaalse äriloogika abil virtuaalse andmelao jaoks klassikaline OLAP andmebaas, mille struktuur moodustatakse automaatselt ja kus ETL toimub automaatselt. Selline süsteemi edasiarendus pakuks kasutajale täielikku klassikalise andmelao funktsionaalsust ja töökiirust, samas tagades alati reaalajas andmelao struktuuri vastavuse transaktsioonilisele süsteemile ning pakkudes täielikku andmekvaliteeti.



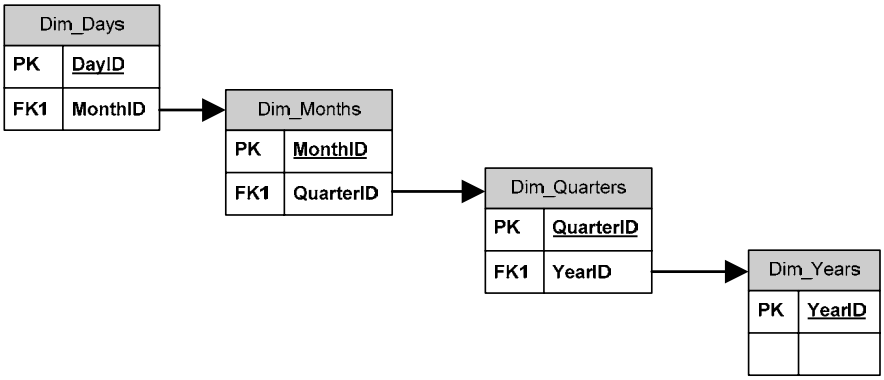
Joonis 35. Virtuaalse andmelao metamudel

Metamudeli tabelis DB_Tables on metaandmetena kirjeldatud transaktsioonilise andmemudeli tabelid, mis on OLAP kuupidesse kaasatud nii faktiliste kui dimensiooniliste andmetabelitena. Lisaks on igale tabelile määratud unikaalne võti. Tähelepanu võiks siin juhtida neljale tabelile, mis on lisatud transaktsioonilisse andmemudelisse: Dim_Days, Dim_Months, Dim_Quarters, Dim_Years; mis oma olemuselt kirjeldavad OLAP jaoks vajalikku aja dimensiooni ning transaktsioonilise süsteemi seisukohalt ei ole vajalikud.

Tabel 5. Metamudeli tabeli DB_Tables andmed

TableID	TableName
1	navision.dbo.[Ordi AS$Sales Invoice Header]
2	navision.dbo.[Ordi AS$Sales Invoice Line]
3	navision.dbo.[Ordi AS$Sales Cr_Memo Header]
4	navision.dbo.[Ordi AS$Sales Cr_Memo Line]
5	navision.dbo.[Ordi AS$Item]
6	navision.dbo.[Ordi AS$Item Category]
7	navision.dbo.[Ordi AS$Customer]
8	navision.dbo.[Ordi AS$Country]
9	navision.dbo.[Ordi AS$Salesperson_Purchaser]
10	navision.dbo.[Ordi AS$Responsibility Center]
11	Dim_Days
12	Dim_Months
13	Dim_Quarters
14	Dim_Years

Joonisel 36 on kujutatud aja dimensiooni abitabelite struktuur. Nendes tabelites sisalduvad andmed on genereeritud automaatselt ja katavad kogu ajalist perioodi, mida OLAP abil analüüsitakse. Päevade tabelis sisalduvad kirjetena kuupäevad alates 01.01.1754 kuni 31.12.2050, kus alguskuupäeva puhul on aluseks võetud Microsoft Navision Attain C/SIDE arenduskeskkonna „big bang", mis tähendab, et kui Navisionis kuupäeva väljale ei ole väärtust sisestatud siis SQL tabelisse salvestatakse väärtus 01.01.1753. Kuude tabelis on kirjed sama ajavahemiku kõikide kuude alguskuupäevadega, kvartalite tabelis kõikide kvartalite alguskuupäevadega ning aastate tabelis kõikide aastate alguskuupäevadega. Need tabelid on omakorda 1:M seoses, kus päevade tabelis on viide kuude tabelisse, mis kajastab seda, millises kuus konkreetne päev asub jne. Selliselt abitabelitega loodud aja dimensioon pakub OLAP tarvis vajalikku klassifikaatorit puurimise ja lahti rullimise funktsionaalsuse tarvis.



Joonis 36. Aja dimensiooni abitabelite struktuur

Metamudeli tabelis DB_Columns on metaandmetena kirjeldatud transaktsioonilise andmemudeli tabelite väljad, mis on OLAP kuupidesse kaasatud nii fakti dimensiooniväärtuse kui mõõdikutena, samuti dimensioonide klassifikaatorite moodustamiseks vajalikud andmeväljad. Tabeli esimene väli TableID viitab tabelile DB_Tables ja on koos ColumnID väljaga primaarseks võtmeks. Väljad DataTypeID ja DataTypeLength on vajalikud hiljem OLAP andmevaate moodustamisel Exceli tabelina. Välja PrimaryKey väärtus on 0 kui kirjeldatav andmeväli ei ole primaarne võti ega selle osa, väärtused 1..n näitavad, et andmeväli on primaarne võti või selle osa, väljendades andmeväljade järjestust võtmes.

Tabel 6. Metamudeli tabeli DB_Columns andmed

TableID	ColumnID	ColumnName	DataTypeID	DataTypeLength	PrimaryKey
1	1	No_	0	0	1
1	2	[Sell-to Customer No_]	0	0	0
1	3	[Posting Date]	0	0	0
1	4	[Salesperson Code]	0	0	0
1	5	[Responsibility Center]	0	0	0
2	1	[Document No_]	0	0	1
2	2	[Line No_]	0	0	2
2	3	No_	0	0	0
2	4	Quantity	0	0	0
2	5	Amount	0	0	0
3	1	No_	0	0	1
3	2	[Sell-to Customer No_]	0	0	0
3	3	[Posting Date]	0	0	0
3	4	[Salesperson Code]	0	0	0
3	5	[Responsibility Center]	0	0	0
4	1	[Document No_]	0	0	1
4	2	[Line No_]	0	0	2
4	3	No_	0	0	0
4	4	Quantity	0	0	0
4	5	Amount	0	0	0
5	1	No_	0	0	1
5	2	Description	0	0	0
5	3	[Item Category Code]	0	0	0
6	1	Code	0	0	1
6	2	Description	0	0	0
7	1	No_	0	0	1
7	2	Name	0	0	0
7	3	[Country Code]	0	0	0
8	1	Code	0	0	1
8	2	Name	0	0	0
9	1	Code	0	0	1

9	2	Name	0	0	0
10	1	Code	0	0	1
10	2	Name	0	0	0
11	1	DayID	0	0	1
11	2	MonthID	0	0	0
12	1	MonthID	0	0	1
12	2	QuarterID	0	0	0
13	1	QuarterID	0	0	1
13	2	YearID	0	0	0
14	1	YearID	0	0	1

Metamudeli tabelis DB_DataTypes on metaandmetena kirjeldatud transaktsioonilise andmemudeli tabelite väljade tüübid, mida hiljem kasutatakse OLAP andmevaate moodustamisel Exceli tabelina.

Tabel 7. Metamudeli tabeli DB_DataTypes andmed

DataTypeID	DataTypeName
0	Variant

Metamudeli tabelis DB_Relationships on metaandmetena kirjeldatud transaktsioonilise andmemudeli tabelite väljade vahelised 1:M seosed. Väli RelationshipID on seose identifikaator ning võimaldab defineerida ka seoseid, mis on 2..n välja põhised ehk sellisel juhul on kirjeldatud sama RelationshipID numbriga 2..n PK-FK seosesse kaasatud välja.

Tabel 8. Metamudeli tabeli DB_Relationships andmed

RelationshipID	PrimaryKeyTableID	PrimaryKeyColumnID	ForeignKeyTableID	ForeignKeyColumnID
1	1	1	2	1
2	3	1	4	1
3	5	1	2	3
4	5	1	4	3
5	6	1	5	3
6	7	1	1	2
7	7	1	3	2
8	8	1	7	3
9	9	1	1	4
10	9	1	3	4
11	10	1	1	5
12	10	1	3	5
13	14	1	13	2
14	13	1	12	2

15	12	1	11	2
16	11	1	1	3
17	11	1	3	3

Metamudeli tabelis OLAP_Fact on metaandmetena kirjeldatud dimensioonilise andmemudeli faktid, ehk OLAP kuubid, millistest kasutaja saab moodustada andmevaateid.

Tabel 9. Metamudeli tabeli OLAP_Fact andmed

FactID	FactName
1	Sales

Metamudeli tabelis OLAP_Measure on metaandmetena kirjeldatud dimensioonilise andmemudeli mõõdikud iga kuubi kohta, milliseid kasutaja saab lisada andmevaadetesse.

Tabel 10. Metamudeli tabeli OLAP_Measure andmed

MeasureID	FactID	MeasureName
1	1	Quantity
2	1	Amount

Metamudeli tabelis OLAP_Fact_Table_rel on metaandmetena kirjeldatud dimensioonilise andmemudeli iga fakti, iga mõõdiku seos transaktsioonilise andmemudeli andmeväljaga. Olukorras, kus ühe OLAP fakti andmevaade moodustub kokku 2..n transaktsioonilise andmemudeli tabelist, saab sama FactID ja MeasureID kombinatsiooniga viidata 2..n andmeväljale transaktsioonilises andmemudelis.

Tabel 11. Metamudeli tabeli OLAP_Fact_Table_rel andmed

FactID	MeasureID	TableID	ColumnID	Negative
1	1	2	4	0
1	1	4	4	1
1	2	2	5	0
1	2	4	5	1

Metamudeli tabelis OLAP_Dimension on metaandmetena kirjeldatud dimensioonilise andmemudeli dimensioonid, mis võivad olla ühised erinevatele faktidele ehk võimaldab kirjeldada ka OLAP mitmik-täht tüüpi andmemudelit.

Tabel 12. Metamudeli tabeli OLAP_Dimension andmed

DimensionID	DimensionName
1	Date
2	Customer
3	Item
4	Salesperson
5	Store

Metamudeli tabelis OLAP_Dimension_Level on metaandmetena kirjeldatud dimensioonilise andmemudeli dimensioonide hierarhia, võimaldades kirjeldada OLAP lumehelbe tüüpi andmemudeleid. Samuti sisaldab see tabel viiteid transaktsioonilise andmemudeli andmeväljadele, millistel asuvad dimensiooniväärtuse kirjeldused ehk selle alusel moodustatakse kasutaja jaoks kergesti kasutatav dimensioonide klassifikaator, kus kasutaja näeb lisaks dimensiooniväärtuse primaarsele võtmele ka kirjeldust.

Tabel 13. Metamudeli tabeli OLAP_Dimension_Level andmed

DimensionID	DimensionLevelID	DimensionLevelName	TableID	ColumnID
1	1	Days	11	1
1	2	Months	12	1
1	3	Quarters	13	1
1	4	Years	14	1
2	1	Customers	7	2
2	2	Countries	8	2
3	1	Items	5	2
3	2	Item categories	6	2
4	1	Salespersons	9	2
5	1	Stores	10	2

Metamudeli tabelis OLAP_Fact_Dim_rel on metaandmetena kirjeldatud dimensioonilise andmemudeli faktide ja dimensioonide vahelised seosed ehk see, milliseid dimensioone saab kasutaja valida, moodustades konkreetse OLAP kuubi põhjal andmevaadet. Samuti on selles tabelis viide transaktsioonilise andmemudeli andmetabelile, millega on OLAP fakt seotud ning andmeväljale, millisega on antud dimensioon seotud fakti andmetabelis. Kui OLAP faktile vastab transaktsioonilises andmemudelis mitu andmetabelid, on siin kirjeldatud kõik viited, kasutades korduvalt sama FactID ja DimensionID kombinatsiooni.

Tabel 14. Metamudeli tabeli OLAP_Fact_Dim_rel andmed

FactID	DimensionID	TableID	ColumnID
1	1	1	3
1	1	3	3
1	2	1	2
1	2	3	2
1	3	2	3
1	3	4	3
1	4	1	4
1	4	3	4
1	5	1	5
1	5	3	5

4.2.2. Virtuaalse andmelao rakendus

Virtuaalse andmelao puhul ei saa me klassikaliselt, kasutades andmebaasi päringute töövahendit, teostada päringuid andmelao tabelitest kuna meil ei ole reaalselt olemas andmelao struktuuriga tabeleid, mis sisaldaks denormaliseeritud kujul andmeid vaid on olemas andmelao struktuuri kirjeldus metaandmetena koos seostega transaktsioonilisse andmebaasi. Sellest tulenevalt on vaja ka päringud kirjeldada metaandmetena ning päringu vastusena moodustada dünaamilised andmevaated vastava universaalse töövahendi abil. Tabelis OLAP_Query on kirjeldatud OLAP andmevaate päringud, mis viitavad faktile, mida soovitakse analüüsida.

Tabel 15. Metamudeli tabeli OLAP_Query andmed

QueryID	QueryName	FactID
1	Sales - Item Category 4100, Period 31.12.2007	1

Tabelis OLAP_Query_Filter on kirjeldatud OLAP andmevaate põhifiltrid, millega on võimalik esmaselt piirata andmevaadet, et vältida liialt mahukaid päringuid transaktsioonilisse andmebaasi. Sisuliselt on tegemist OLAP kuubi viilutamise (Slice) funktsionaalsusega.

Tabel 16. Metamudeli tabeli OLAP_Query_Filter andmed

QueryID	DimensionID	DimensionLevelID	DimensionValue
1	1	1	20071231
1	3	2	4100

Eelnevalt kirjeldatud virtuaalse andmelao metaandmete põhjal genereerib universaalne rakendus automaatselt transaktsioonilise andmebaasi SQL päringu. Rakenduse kood on toodud lisas 5, näites 1. Lisaks transaktsioonilise andmebaasi andmetele on virtuaalse andmelao rakenduse tarvis loodud spetsiaalsed aja dimensiooni klassifikaatori tabelid, mille struktuuri on kirjeldatud eelnevalt. Aja dimensiooni tabelite automaatne andmetega täitmine toimub ühekordselt lisas 5, näites 2 toodud programmikoodiga. Järgnevas näites 2 on toodud metaandmete baasil, universaalse rakenduse poolt, genereeritud SQL päring.

Näide 2. Dünaamilise andmelao rakenduse poolt genereeritud SQL päring

```
SELECT navision.dbo.[Ordi AS$Sales Invoice Line].Quantity AS Quantity,
navision.dbo.[Ordi AS$Sales Invoice Line].Amount AS Amount,
Dim_Days.DayID AS Days,
Dim_Months.MonthID AS Months,
Dim_Quarters.QuarterID AS Quarters,
Dim_Years.YearID AS Years,
navision.dbo.[Ordi AS$Customer].No_ AS Customers,
navision.dbo.[Ordi AS$Country].Code AS Countries,
navision.dbo.[Ordi AS$Item].No_ AS Items,
navision.dbo.[Ordi AS$Item Category].Code AS Item_categories,
navision.dbo.[Ordi AS$Salesperson_Purchaser].Code AS Salespersons,
navision.dbo.[Ordi AS$Responsibility Center].Code AS Stores
FROM navision.dbo.[Ordi AS$Sales Invoice Line],
navision.dbo.[Ordi AS$Sales Invoice Header],
Dim_Days,
Dim_Months,
Dim_Quarters,
Dim_Years,
navision.dbo.[Ordi AS$Customer],
navision.dbo.[Ordi AS$Country],
navision.dbo.[Ordi AS$Item],
navision.dbo.[Ordi AS$Item Category],
navision.dbo.[Ordi AS$Salesperson_Purchaser],
navision.dbo.[Ordi AS$Responsibility Center]
WHERE navision.dbo.[Ordi AS$Sales Invoice Header].No_ = navision.dbo.[Ordi AS$Sales Invoice
Line].[Document No_]
AND Dim_Days.DayID = navision.dbo.[Ordi AS$Sales Invoice Header].[Posting Date]
AND Dim_Months.MonthID = Dim_Days.MonthID
AND Dim_Quarters.QuarterID = Dim_Months.QuarterID
AND Dim_Years.YearID = Dim_Quarters.YearID
AND navision.dbo.[Ordi AS$Customer].No_ = navision.dbo.[Ordi AS$Sales Invoice Header].[Sell-to
Customer No_]
AND navision.dbo.[Ordi AS$Country].Code = navision.dbo.[Ordi AS$Customer].[Country Code]
AND navision.dbo.[Ordi AS$Item].No_ = navision.dbo.[Ordi AS$Sales Invoice Line].No_
AND navision.dbo.[Ordi AS$Item Category].Code = navision.dbo.[Ordi AS$Item].[Item Category
Code]
AND navision.dbo.[Ordi AS$Salesperson_Purchaser].Code = navision.dbo.[Ordi AS$Sales Invoice
Header].[Salesperson Code]
AND navision.dbo.[Ordi AS$Responsibility Center].Code = navision.dbo.[Ordi AS$Sales Invoice
Header].[Responsibility Center]
AND Dim_Days.DayID = '20071231'
AND navision.dbo.[Ordi AS$Item Category].Code = '4100'
UNION ALL
SELECT -navision.dbo.[Ordi AS$Sales Cr_Memo Line].Quantity AS Quantity,
-navision.dbo.[Ordi AS$Sales Cr_Memo Line].Amount AS Amount,
Dim_Days.DayID AS Days,
Dim_Months.MonthID AS Months,
Dim_Quarters.QuarterID AS Quarters,
Dim_Years.YearID AS Years,
navision.dbo.[Ordi AS$Customer].No_ AS Customers,
navision.dbo.[Ordi AS$Country].Code AS Countries,
navision.dbo.[Ordi AS$Item].No_ AS Items,
navision.dbo.[Ordi AS$Item Category].Code AS Item_categories,
navision.dbo.[Ordi AS$Salesperson_Purchaser].Code AS Salespersons,
navision.dbo.[Ordi AS$Responsibility Center].Code AS Stores
FROM navision.dbo.[Ordi AS$Sales Cr_Memo Line],
```

```
navision.dbo.[Ordi AS$Sales Cr_Memo Header],
Dim_Days,
Dim_Months,
Dim_Quarters,
Dim_Years,
navision.dbo.[Ordi AS$Customer],
navision.dbo.[Ordi AS$Country],
navision.dbo.[Ordi AS$Item],
navision.dbo.[Ordi AS$Item Category],
navision.dbo.[Ordi AS$Salesperson_Purchaser],
navision.dbo.[Ordi AS$Responsibility Center]
WHERE navision.dbo.[Ordi AS$Sales Cr_Memo Header].No_ = navision.dbo.[Ordi AS$Sales Cr_Memo
Line].[Document No_]
AND Dim_Days.DayID = navision.dbo.[Ordi AS$Sales Cr_Memo Header].[Posting Date]
AND Dim_Months.MonthID = Dim_Days.MonthID
AND Dim_Quarters.QuarterID = Dim_Months.QuarterID
AND Dim_Years.YearID = Dim_Quarters.YearID
AND navision.dbo.[Ordi AS$Customer].No_ = navision.dbo.[Ordi AS$Sales Cr_Memo Header].[Sell-to
Customer No_]
AND navision.dbo.[Ordi AS$Country].Code = navision.dbo.[Ordi AS$Customer].[Country Code]
AND navision.dbo.[Ordi AS$Item].No_ = navision.dbo.[Ordi AS$Sales Cr_Memo Line].No_
AND navision.dbo.[Ordi AS$Item Category].Code = navision.dbo.[Ordi AS$Item].[Item Category
Code]
AND navision.dbo.[Ordi AS$Salesperson_Purchaser].Code = navision.dbo.[Ordi AS$Sales Cr_Memo
Header].[Salesperson Code]
AND navision.dbo.[Ordi AS$Responsibility Center].Code = navision.dbo.[Ordi AS$Sales Cr_Memo
Header].[Responsibility Center]
AND Dim_Days.DayID = '20071231' AND navision.dbo.[Ordi AS$Item Category].Code = '4100'
```

Tabelis 17 on kujutatud transaktsioonilise andmebaasi andmete põhjal moodustatud OLAP andmevaade, kus selle esitlemiseks ja OLAP kuubi kasutaja funktsionaalsuse (Drill-Down, Roll-Up, Slice, Dice) tarvis on kasutatud Excel'i Pivot tabelit.

Tabel 17. Dünaamilise andmelao rakendusega loodud andmevaade Excel Pivot tabelina

Sum of Amount	**Stores**					
Items	**NARVA**	**TLN-HULGI**	**TLN-TONDI**	**TRT-HULGI**	**TRT-KYYNI**	**Grand Total**
4100-1396		6888,98				**6888,98**
4100-1441		2448,31				**2448,31**
4100-1488		3585,59				**3585,59**
4100-1503			2254,24			**2254,24**
4100-1543		9118,65	4991,53	4559,32	2279,66	**20949,16**
4100-1556			2016,95		2016,95	**4033,9**
4100-1558					2288,13	**2288,13**
4100-1564				2661,02		**2661,02**
4100-1566	2233,06	4466,1				**6699,16**
4100-1586		2394,07	2542,37	4936,44		**9872,88**
4100-1591					3961,86	**3961,86**
4100-1592	3288,13					**3288,13**
4100-1599		2033,9				**2033,9**
4100-1607				4101,7	6186,44	**10288,14**
Grand Total	**5521,19**	**30935,6**	**11805,09**	**16258,48**	**16733,04**	**81253,4**

KOKKUVÕTE

Käesoleva magistritöö esimeses peatükis on kirjeldatud Microsoft ERP (Enterprise Resource Planning) tarkvaratoodete juurutusmetoodikat „Microsoft Dynamics Sure Step" koos majandustarkvara Microsoft Dynamics NAV juurutusmudeliga RIM (Rapid Implementation Methodology). Teises peatükis on kirjeldatud spetsiaaltarkvara arendusprotsessi OpenUP koos käsitletava ettevõtte vertikaallahenduse arendusprotsessi näidetega, samuti on kirjeldatud ja analüüsitud selliselt arendatud infosüsteemi probleeme ja kitsaskohti.

Juurutus- ja arendusmetoodikate võrdleva analüüsi tulemusena selgub, et majandustarkvara juurutusmetoodikad sarnanevad kõigis olulistes etappides klassikalise spetsiaaltarkvara arendusmetoodikaga ning ei paku universaalset metatasandit, mille abil oleks võimalik modelleerida kogu juurutusprotsessi. Olemasolev RIM metamudel pakub vaid väga primitiivseid võimalusi juurutusprotsessis püsiandmete ja süsteemi seadistuste kirjeldamisel.

Kolmandas peatükis on kirjeldatud infosüsteemide andmeanalüüsi teostamise kontseptsiooni ja andmelao loomise põhimõtteid. Analüüsitud on klassikalise andmelao kitsaskohti ja problemaatikat, mille põhjal on jõutud järeldusele, et andmeladu ei peaks olema infosüsteemi arendamise teises etapis loodud eraldiseisev lahendus vaid infosüsteemi arendamisel peaks andmete sisestamist pakkuv OLTP (On-Line Transaction Processing) ja andmete analüüsi pakkuv OLAP (On-Line Analytical Processing) vaade olema lahutamatult seotud ning pidevas kooskõlas arendatud.

Neljandas peatükis on eelnev kokku võetud ning selle põhjal välja töötatud kohaldatud, OLAP kuubil rajanev infosüsteemi arendusmeetod, mis lähtub põhimõttest, et OLTP ja OLAP kontseptsioone tuleb käsitleda koos ja moodustada nende põhjal ühine, seotud infosüsteemi andmemudel. Kirjeldatud meetodi realiseerimiseks on välja töötatud uudne, virtuaalse andmelao metamudel ja universaalne virtuaalse andmelao rakendus, mis kasutades nimetatud metamudelit koos transaktsioonilise andmebaasi andmetega, moodustab klassikalisele andmelaole sarnaselt OLAP andmevaateid, mida kasutaja saab OLAP kuubi tööriista, näiteks Excel'i Pivot tabeli, abil töödelda ja visualiseerida.

Käesoleva magistritöö teise eesmärgina on loodud käsitletava ettevõtte jaoks virtuaalse andmelao realisatsioon, nimetatuna „Öine aruandegeneraator", kus kasutaja saab kirjeldada soovitud OLAP vaate, kasutades intuitiivselt arusaadavat virtuaalset OLAP andmestruktuuri. Rakendus genereerib igaöiselt soovitud andmevaated ja tulemused salvestatakse Excel'i Pivot tabelina serverikettale.

Magistritöös realiseeritud idee jätkuna on plaanis välja töötada täiustatud metamudel koos edasi arendatud universaalse rakendusega, mis võimaldaks lisaks virtuaalse andmelao funktsionaalsusele ka automaatselt genereerida klassikalise andmelao OLAP andmestruktuuri koos ETL (Extract Transform Load) äriloogikaga. Selline metamudelil põhinev ja automaatselt loodud andmeladu omaks olulisi eeliseid võrreldes klassikalise andmelaoga, kuna OLAP ja OLTP oleks reaalajas seotud, millega oleks välistatud andmemudelite kooskõla probleemid. Automaatselt genereeritud ETL äriloogika välistaks, klassikalistes andmeladudes enamasti esinevad, OLTP ja OLAP vahelised andmekvaliteedi probleemid.

KASUTATUD KIRJANDUS

1. **Lemmik, R.** (2005); Ettevõtte infosüsteemi loomine majandustarkvara Microsoft Navision Attain baasil, Audentese Ülikool, Bakalaureusetöö
2. **Microsoft Corporation** (2007); Using the Microsoft Dynamics™ Sure Step methodology. Microsoft Official Training Materials for Microsoft Dynamics
3. **Microsoft Corporation** (2008); Navision Rapid Implementation Methodology Toolkit. Users Guide
4. **OpenUP.** OpenUP Plug-in Version 1.0. [http://epf.eclipse.org/wikis/openup/index.htm]. 13. märts 2008.
5. **Navision a/s.** Application Designer's Guide. [CD-ROM]. Navision Attain version 3.6 Installation CD.
6. **IBM International Technical Support Organization** (1998); Data Modeling Techniques for Data Warehousing
7. **IBM International Technical Support Organization** (2006); Dimensional Modeling: In a Business Intelligence Environment
8. **Martin Fowler** (2006); UMLi kontsentraat. Cybernetica AS, 140 lk.
9. **McCausland, R.** (2001); Manufacturing Plus Accounting: Made to Order. – Accounting Technology, Vol. 17, Issue 7, pp 38-42.
10. **Orenstein, D.** (2004); Hard Sell for Manufacturing Software. – Electronic Business, Vol. 30, Issue 11, pp 52-56.
11. **Scott, R.W.** (2005); Manufacturing a Solution. – Accounting Technology, Vol. 21, Issue 4, pp 38-41.
12. **Tribunella, T., Warner, P.D., Smith, L.M.** (2002); Designing Relational Database Systems. – CPA Journal, Vol. 72, Issue 7, pp 69-72.
13. **West, R.N.** (2000); How To Design a Database. – Management Accounting Quarterly, Vol. 2, Issue 1, pp 18-24.

LISAD

Lisa 1

Sure Step RIM seadistusandmete tabelid

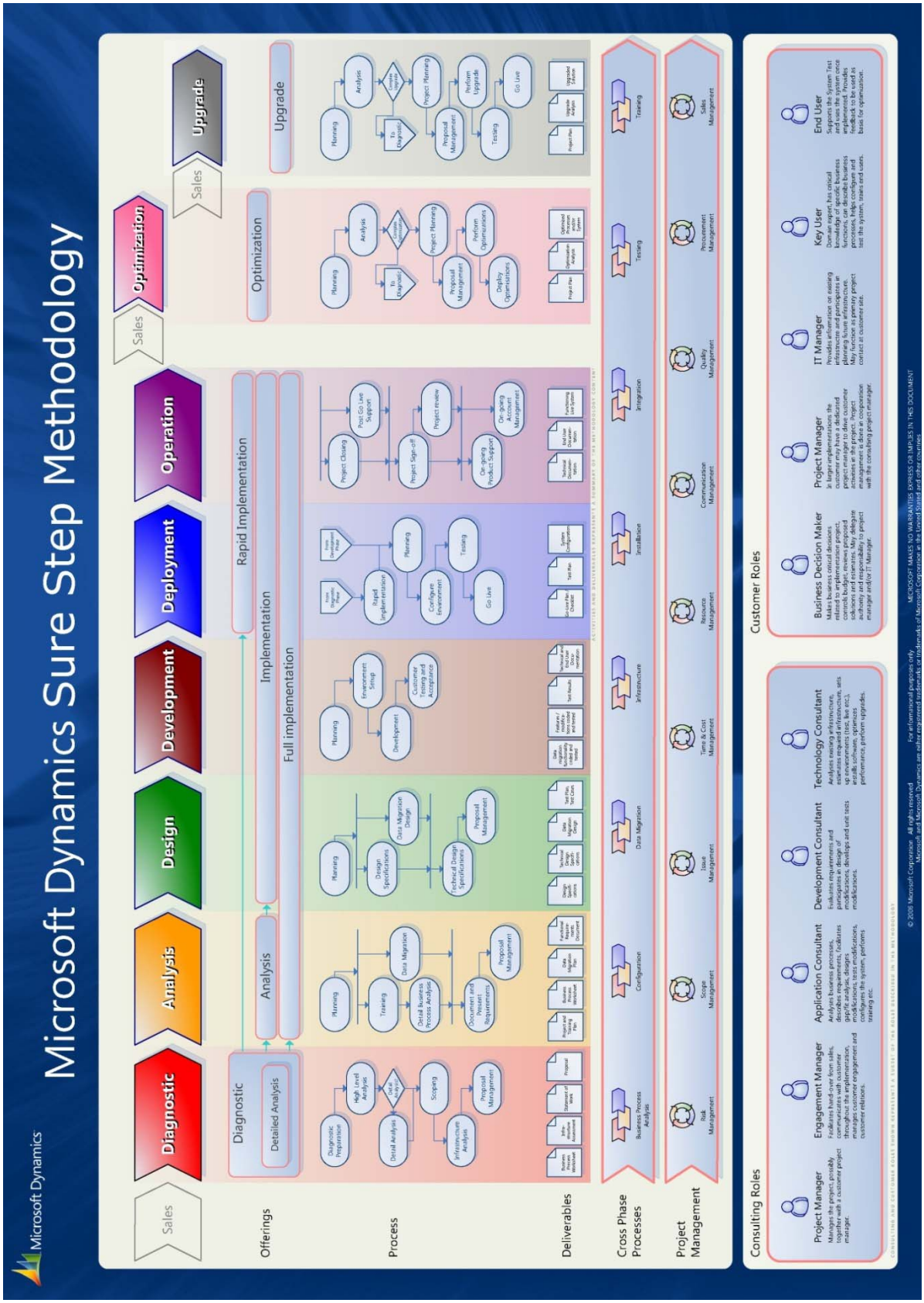
Tabeli nr	Tabeli nimi	Andmete päritolu
3	Maksetingimused	Ettevõtte sätete XML-failid
4	Valuuta	Ettevõtte sätete XML-failid
5	Viivisetingimused	Ettevõtte sätete XML-failid
6	Kliendihinnarühmad	Ettevõtte sätete XML-failid
8	Keel	Ettevõtte sätete XML-failid
9	Riik	Ettevõtte sätete XML-failid
10	Lähetusviis	Ettevõtte sätete XML-failid
42	Ümardusviis	Ettevõtte sätete XML-failid
50	Arvestusperiood	Ettevõtte sätete XML-failid
79	Ettevõtte andmed	Ettevõtte sätete XML-failid / küsimustik
80	Peažurnaali mall	Ettevõtte sätete XML-failid
82	Kaubažurnaalimall	Ettevõtte sätete XML-failid
92	Kliendi konteeringurühm	Ettevõtte sätete XML-failid
93	Hankija konteeringurühm	Ettevõtte sätete XML-failid
94	Varude konteeringurühm	Ettevõtte sätete XML-failid
98	Pearaamatu seadistamine	Ettevõtte sätete XML-failid / küsimustik
204	Mõõtühik	Ettevõtte sätete XML-failid
232	Peažurnaali tööleht	Ettevõtte sätete XML-failid
250	Üldine äri konteeringurühm	Ettevõtte sätete XML-failid
251	Üldine toote konteeringurühm	Ettevõtte sätete XML-failid
252	Üld. konteeringu seadistamine	Ettevõtte sätete XML-failid
289	Makseviis	Ettevõtte sätete XML-failid
291	Ekspediitor	Ettevõtte sätete XML-failid
308	Numbriseeria	Ettevõtte sätete XML-failid
309	Numbriseeria rida	Ettevõtte sätete XML-failid
311	Müügi ja müügivõlgade seadistamine	Ettevõtte sätete XML-failid / küsimustik
312	Ostude ja ostuvõlgade seadistamine	Ettevõtte sätete XML-failid / küsimustik
313	Varude seadistamine	Ettevõtte sätete XML-failid / küsimustik
314	Ressursiarvestuse seadistamine	Ettevõtte sätete XML-failid
315	Tööde seadistamine	Ettevõtte sätete XML-failid
323	KM äri konteeringurühm	Ettevõtte sätete XML-failid
324	KM toote konteeringurühm	Ettevõtte sätete XML-failid
325	KM konteerimise seadistamine	Ettevõtte sätete XML-failid
340	Kliendi hinnaalandi rühm	Ettevõtte sätete XML-failid

5053	Ärisuhted	Ettevõtte sätete XML-failid
5055	Postitusrühm	Ettevõtte sätete XML-failid
5070	Organisatsiooni tase	Ettevõtte sätete XML-failid
5079	Kliendisuhete halduse seadistamine	Ettevõtte sätete XML-failid
5087	Profiiliküsimustiku päis	Ettevõtte sätete XML-failid
5088	Profiiliküsimustiku rida	Ettevõtte sätete XML-failid
5218	Personaliarvestuse seadistamine	Ettevõtte sätete XML-failid
5500	Tootmisgraafiku seadistus	Ettevõtte sätete XML-failid
5603	PV seadistamine	Ettevõtte sätete XML-failid
5604	PV konteeringu liigi seadistamine	Ettevõtte sätete XML-failid
5611	Kulumiraamat	Ettevõtte sätete XML-failid
5619	PV žurnaali mall	Ettevõtte sätete XML-failid
5620	PV žurnaali tööleht	Ettevõtte sätete XML-failid
5700	Laohoiuühik	Ettevõtte sätete XML-failid
5714	Vastutuskeskus	Ettevõtte sätete XML-failid
5769	Lao seadistus	Ettevõtte sätete XML-failid
5790	Ekspediitoriteenused	Ettevõtte sätete XML-failid
5911	Hooldushalduse seadistamine	Ettevõtte sätete XML-failid / küsimustik
6081	Hooldus- hinnarühma seadistamine	Ettevõtte sätete XML-failid
6502	Kauba jälgimistähis	Ettevõtte sätete XML-failid
7303	Aluse tüüp	Ettevõtte sätete XML-failid
7304	Lao klass	Ettevõtte sätete XML-failid
7335	Aluse mall	Ettevõtte sätete XML-failid
7336	Aluse loomise töölehe mall	Ettevõtte sätete XML-failid
7337	Aluse loomise töölehe nimi	Ettevõtte sätete XML-failid
7600	Aluskalender	Ettevõtte sätete XML-failid
99000750	Töövahetus	Ettevõtte sätete XML-failid
99000751	Töögraafik	Ettevõtte sätete XML-failid
99000752	Töögraafiku tööpäevad	Ettevõtte sätete XML-failid
99000756	Töökeskuse rühm	Ettevõtte sätete XML-failid
99000765	Tootmise seadistamine	Ettevõtte sätete XML-failid
99000778	Standardülesanne	Ettevõtte sätete XML-failid
99000780	Võimsuse mõõtühik	Ettevõtte sätete XML-failid

Allikas: Navision Rapid Implementation Methodology Toolkit [3]

Lisa 2

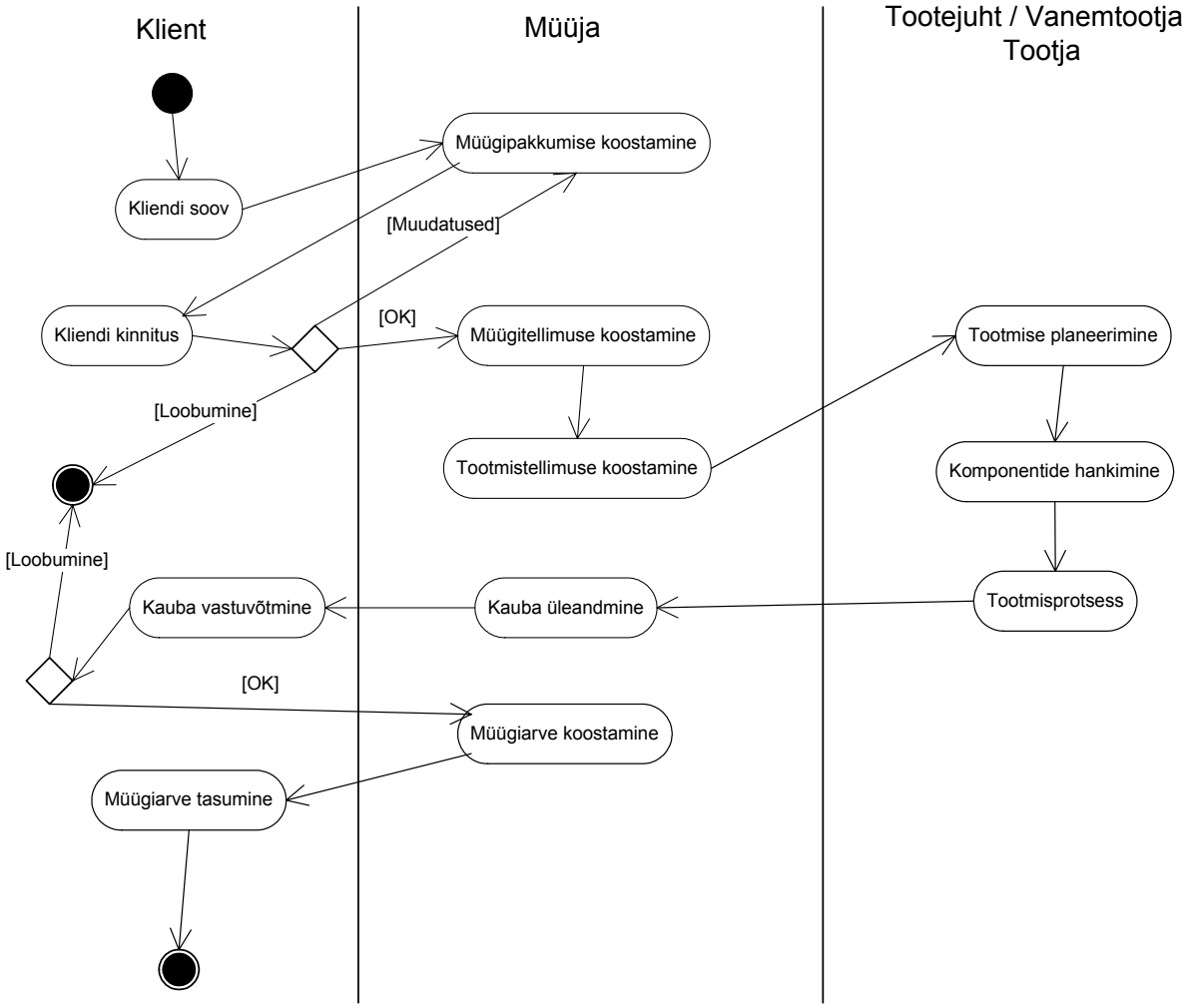
Microsoft Dynamics Sure Step diagramm



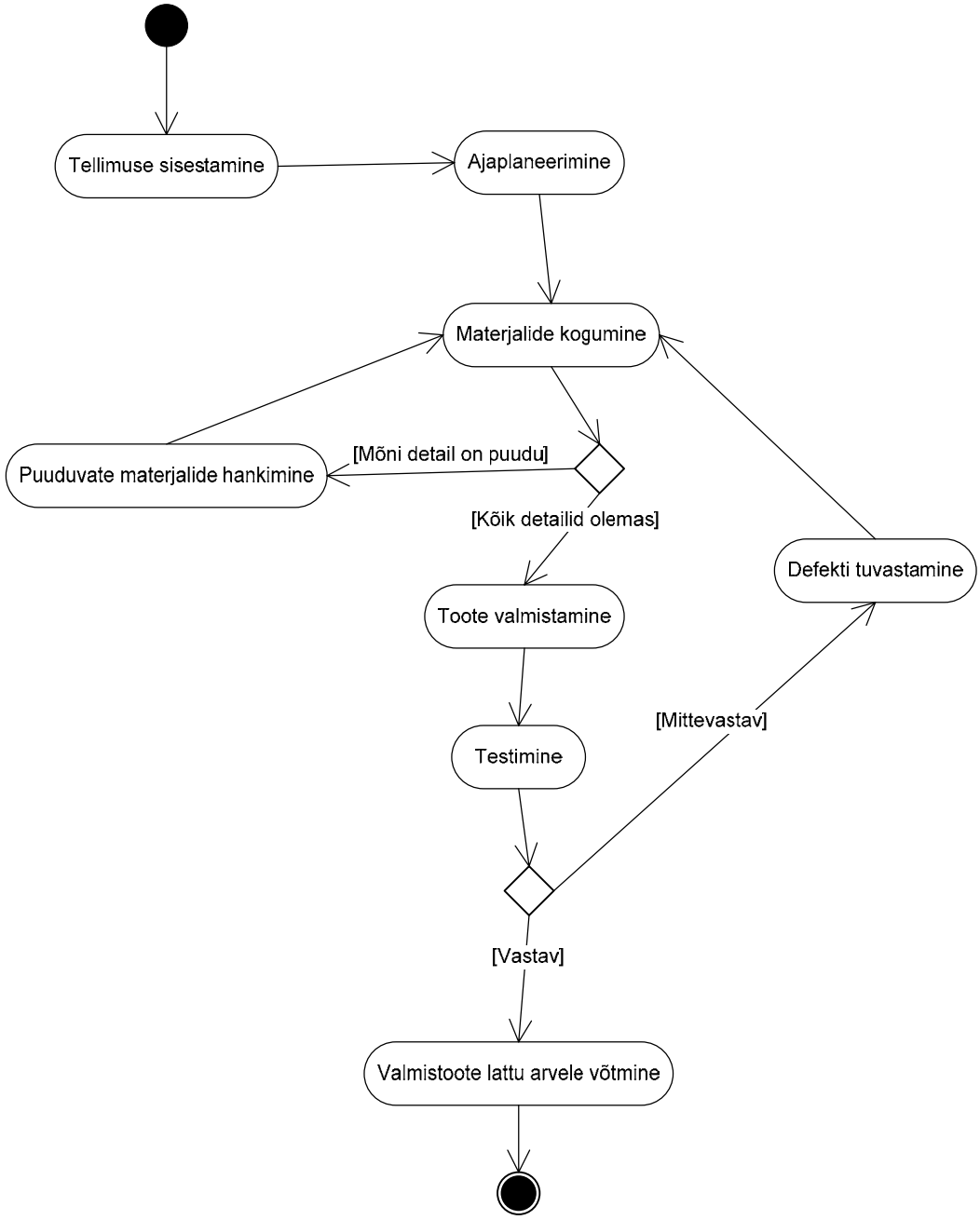
Allikas: Using the Microsoft Dynamics™ Sure Step methodology [2]

Lisa 3

Talitusprotsesside joonised



Joonis 1. Põhiprotsess



Joonis 2. Tootmisprotsess

Lisa 4

Vertikaallahenduse tabelite struktuur

Tabel 1. Tootmistellimuste tabeli väljaloend

Välja nimi	Andmetüüp	Pikkus	Võtmed, seosed	Kirjeldus
id	int	4	PK	Tootmistellimuse number
tellimus	varchar	20	FK müügitellimuste tabelist	Müügitellimuse number
kuup	datetime	16		Loomise kuupäev
kontor	varchar	20	FK kontorite tabelist	Loomise koht
klient	varchar	20	FK klientide tebelist	Kliendi kood
komment	varchar	1000		Kommentaar
staatus	int	4	FK staatuste tabelist	Staatus
userid	varchar	20	FK isikute tabelist	Looja isik
nimi	varchar	50		Kliendi nimi
tahtaeg	datetime	16		Prognoositav tähtaeg
tootmine	varchar	20	FK kontorite tabelist	Tootmisosakond

Tabel 2. Tootmistellimuse sisu tabeli väljaloend

Välja nimi	Andmetüüp	Pikkus	Võtmed, seosed	Kirjeldus
id	int	4	PK	Rea ID
viide	int	4	FK tootmistellimuste tabelist	Tootmistellimuse number
artikkel	varchar	20	FK kaupade tabelist	Kauba kood
nimetus	varchar	50		Kauba nimetus
komplekt	int	4		Komplekti jrk. number
kindel	tinyint	1		Kindel linnuke
yhik	varchar	20		Kauba mõõtühik
hind	money	21		Rea ühiku hind

Tabel 3. Valmistuskaartide tabeli väljaloend

Välja nimi	Andmetüüp	Pikkus	Võtmed, seosed	Kirjeldus
id	int	4	PK	Valmistuskaardi number
tellimus	int	4	FK tootmistellimuste tabelist	Tootmistellimuse number
staatus	int	4	FK staatuste tabelist	Staatus
tehnik	int	4	FK isikute tabelist	Monteerija kood
kontor	varchar	20	FK kontorite tabelist	Tootmisosakond
algus	datetime	16		Monteerimise algus
valmis	datetime	16		Monteerimise lõpp
tahtaeg	datetime	16		Valmimise tähtaeg
komment	varchar	1000		Kommentaar
testsoft	varchar	1000		Testitarkvara loend
testok	tinyint	1		Test läbitud
probleem	tinyint	1		Probleem mingi detailiga

tehniknimi	varchar	50		Monteerija nimi
artikkel	varchar	20	FK kaupade tabelist	Komplektimalli kood
nimetus	varchar	50		Komplektimalli numetus
artikkel_navi	varchar	20	FK kaupade tabelist	Valmis arvuti kood
hind	money	21		Ridade summaarne hind
mark	varchar	20		Märgistamise abiväli

Tabel 4. Valmistuskaardi sisu tabeli väljaloend

Välja nimi	**Andmetüüp**	**Pikkus**	**Võtmed, seosed**	**Kirjeldus**
id	int	4	PK	Rea ID
viide	int	4	FK valmistuskaartide tabelist	Valmistuskaardi number
artikkel_tell	varchar	20	FK kaupade tabelist	Tellitud kauba kood
nimetus_tell	varchar	50		Tellitud kauba nimetus
artikkel_paig	varchar	20	FK kaupade tabelist	Paigaldatud kauba kood
nimetus_paig	varchar	50		Paigaldatud kauba nimetus
probleem	tinyint	1		Probleemi linnuke
kindel	tinyint	1		Kindel linnuke
serial	varchar	50		Seerianumber
yhik	varchar	20		Mõõtühik

Lisa 5

Dünaamilise andmelao rakendus

Näide 1. Dünaamilise andmelao andmevaate genereerimine

```
procedure OLAPQuery;
var OLAPQueryID: integer;
    OLAPSQL: string;
    SQLPart: array of string;
    SQLTables, SQLRel: array of array of integer;
    i, j, k, l: integer;
    a: string;
    b: boolean;
begin
  OLAPQueryID := 1;
  DB_Tables.Open;
  DB_Columns.Open;
  DB_Relationships.Open;
  OLAP_Dimension.Open;
  OLAP_Dimension_Level.Open;
  OLAP_Measure.Open;
  OLAP_Fact_Dim_Rel.Open;
  OLAP_Fact_Table_Rel.Open;
  OLAP_Query.Open;
  OLAP_Query_Filter.Open;

  OLAP_Query.Locate('QueryID',OLAPQueryID,[]);

  OLAP_Measure.Filtered := false;
  OLAP_Measure.Filter := 'FactID = '+inttostr(OLAP_Query['FactID']);
  OLAP_Measure.Filtered := true;
  OLAP_Measure.First;
  repeat
    OLAP_Fact_Table_Rel.Filtered := false;
    OLAP_Fact_Table_Rel.Filter := 'FactID = '+inttostr(OLAP_Measure['FactID'])+' and MeasureID
= '+inttostr(OLAP_Measure['MeasureID']);
    OLAP_Fact_Table_Rel.Filtered := true;
    OLAP_Fact_Table_Rel.First;
    repeat
      if Length(SqlPart) < OLAP_Fact_Table_Rel.RecNo then
      begin
        SetLength(SqlPart,OLAP_Fact_Table_Rel.RecNo);
        SetLength(SqlTables,OLAP_Fact_Table_Rel.RecNo);
        SetLength(SqlRel,OLAP_Fact_Table_Rel.RecNo);
      end;

      DB_Tables.Locate('TableID',inttostr(OLAP_Fact_Table_Rel['TableID']),[]);

DB_Columns.Locate('TableID;ColumnID',VarArrayOf([inttostr(OLAP_Fact_Table_Rel['TableID']),intt
ostr(OLAP_Fact_Table_Rel['ColumnID'])]),[]);

      if OLAP_Measure.RecNo = 1 then
        SqlPart[OLAP_Fact_Table_Rel.RecNo - 1] := 'SELECT '
      else
        SqlPart[OLAP_Fact_Table_Rel.RecNo - 1] := SqlPart[OLAP_Fact_Table_Rel.RecNo - 1] + ',
';

      if OLAP_Fact_Table_Rel['Negative'] = 1 then
        a := '-'
      else
        a := '';

      SqlPart[OLAP_Fact_Table_Rel.RecNo - 1] := SqlPart[OLAP_Fact_Table_Rel.RecNo - 1] + a +
Db_Tables['TableName'] + '.' + DB_Columns['ColumnName'] + ' AS ' +
OLAP_Measure['MeasureName'];

      OLAP_Fact_Table_Rel.Next;
    until OLAP_Fact_Table_Rel.Eof;

    OLAP_Measure.Next;
  until OLAP_Measure.Eof;
```

```
  OLAP_Dimension.First;
  repeat
    OLAP_Fact_Dim_Rel.Filtered := false;
    OLAP_Fact_Dim_Rel.Filter := 'FactID = '+inttostr(OLAP_Query['FactID']) + ' and DimensionID
= ' +inttostr(OLAP_Dimension['DimensionID']);
    OLAP_Fact_Dim_Rel.Filtered := true;
    OLAP_Fact_Dim_Rel.First;
    if not (OLAP_Fact_Dim_Rel.Bof and OLAP_Fact_Dim_Rel.Eof) then
    begin
      repeat
        OLAP_Dimension_Level.Filtered := false;
        OLAP_Dimension_Level.Filter := 'DimensionID =
'+inttostr(OLAP_Dimension['DimensionID']);
        OLAP_Dimension_Level.Filtered := true;
        OLAP_Dimension_Level.First;
        repeat
          DB_Tables.Locate('TableID',inttostr(OLAP_Dimension_Level['TableID']),[]);

DB_Columns.Locate('TableID;PrimaryKey',VarArrayOf([inttostr(OLAP_Dimension_Level['TableID']),1
]),[]);

          SqlPart[OLAP_Fact_Dim_Rel.RecNo - 1] := SqlPart[OLAP_Fact_Dim_Rel.RecNo - 1] + ', '
+ Db_Tables['TableName'] + '.' + DB_Columns['ColumnName'] + ' AS ' +
OLAP_Dimension_Level['DimensionLevelName'];

          OLAP_Dimension_Level.Next;
        until OLAP_Dimension_Level.Eof;

        OLAP_Fact_Dim_Rel.Next;
      until OLAP_Fact_Dim_Rel.Eof;
    end;
    OLAP_Dimension.Next;
  until OLAP_Dimension.Eof;

  OLAP_Measure.Filtered := false;
  OLAP_Measure.Filter := 'FactID = '+inttostr(OLAP_Query['FactID']);
  OLAP_Measure.Filtered := true;
  OLAP_Measure.First;
  repeat
    OLAP_Fact_Table_Rel.Filtered := false;
    OLAP_Fact_Table_Rel.Filter := 'FactID = '+inttostr(OLAP_Measure['FactID'])+' and MeasureID
= '+inttostr(OLAP_Measure['MeasureID']);
    OLAP_Fact_Table_Rel.Filtered := true;
    OLAP_Fact_Table_Rel.First;
    repeat
      b := false;
      for i := 1 to Length(SqlTables[OLAP_Fact_Table_Rel.RecNo - 1]) do
      begin
        if SqlTables[OLAP_Fact_Table_Rel.RecNo - 1,i - 1] = OLAP_Fact_Table_Rel['TableID']
then
          b := true;
      end;

      if not b then
      begin
        i := Length(SqlTables[OLAP_Fact_Table_Rel.RecNo - 1]);
        setlength(SqlTables[OLAP_Fact_Table_Rel.RecNo - 1],i + 1);
        SqlTables[OLAP_Fact_Table_Rel.RecNo - 1,i] := OLAP_Fact_Table_Rel['TableID'];
      end;

      OLAP_Fact_Table_Rel.Next;
    until OLAP_Fact_Table_Rel.Eof;

    OLAP_Measure.Next;
  until OLAP_Measure.Eof;

  OLAP_Dimension.First;
  repeat
    OLAP_Fact_Dim_Rel.Filtered := false;
    OLAP_Fact_Dim_Rel.Filter := 'FactID = '+inttostr(OLAP_Query['FactID']) + ' and DimensionID
= ' +inttostr(OLAP_Dimension['DimensionID']);
    OLAP_Fact_Dim_Rel.Filtered := true;
    OLAP_Fact_Dim_Rel.First;
    if not (OLAP_Fact_Dim_Rel.Bof and OLAP_Fact_Dim_Rel.Eof) then
    begin
      repeat
```

```
        b := false;
        for i := 1 to Length(SqlTables[OLAP_Fact_Dim_Rel.RecNo - 1]) do
        begin
          if SqlTables[OLAP_Fact_Dim_Rel.RecNo - 1,i - 1] = OLAP_Fact_Dim_Rel['TableID'] then
            b := true;
        end;

        if not b then
        begin
          i := Length(SqlTables[OLAP_Fact_Dim_Rel.RecNo - 1]);
          setlength(SqlTables[OLAP_Fact_Dim_Rel.RecNo - 1],i + 1);
          SqlTables[OLAP_Fact_Dim_Rel.RecNo - 1,i] := OLAP_Fact_Dim_Rel['TableID'];
        end;

        OLAP_Dimension_Level.Filtered := false;
        OLAP_Dimension_Level.Filter := 'DimensionID =
'+inttostr(OLAP_Dimension['DimensionID']);
        OLAP_Dimension_Level.Filtered := true;
        OLAP_Dimension_Level.First;
        repeat
          b := false;
          for i := 1 to Length(SqlTables[OLAP_Fact_Dim_Rel.RecNo - 1]) do
          begin
            if SqlTables[OLAP_Fact_Dim_Rel.RecNo - 1,i - 1] = OLAP_Dimension_Level['TableID']
then
              b := true;
          end;

          if not b then
          begin
            i := Length(SqlTables[OLAP_Fact_Dim_Rel.RecNo - 1]);
            setlength(SqlTables[OLAP_Fact_Dim_Rel.RecNo - 1],i + 1);
            SqlTables[OLAP_Fact_Dim_Rel.RecNo - 1,i] := OLAP_Dimension_Level['TableID'];
          end;

          OLAP_Dimension_Level.Next;
        until OLAP_Dimension_Level.Eof;

        OLAP_Fact_Dim_Rel.Next;
      until OLAP_Fact_Dim_Rel.Eof;
    end;
    OLAP_Dimension.Next;
  until OLAP_Dimension.Eof;

  for i := 1 to length(SqlTables) do
    for j := 1 to length(SqlTables[i - 1]) do
    begin
      DB_Tables.Locate('TableID',SqlTables[i - 1,j - 1],[]);

      if j = 1 then
        SqlPart[i - 1] := SqlPart[i - 1] + ' FROM ' + DB_Tables['TableName']
      else
        SqlPart[i - 1] := SqlPart[i - 1] + ', ' + DB_Tables['TableName'];

      DB_Relationships.Filtered := false;
      DB_Relationships.Filter := 'PrimaryKeyTableID = '+ inttostr(SqlTables[i - 1,j - 1]);
      DB_Relationships.Filtered := true;
      DB_Relationships.First;
      if not (DB_Relationships.Bof and DB_Relationships.Eof) then
        repeat
          for k := 1 to length(SqlTables[i - 1]) do
          begin
            if DB_Relationships['ForeignKeyTableID'] = SqlTables[i - 1,k - 1] then
            begin
              b := false;
              for l := 1 to Length(SqlRel[i - 1]) do
              begin
                if SqlRel[i - 1,l - 1] = DB_Relationships['RelationshipID'] then
                  b := true;
              end;

              if not b then
              begin
                l := Length(SqlRel[i - 1]);
                setlength(SqlRel[i - 1],l + 1);
                SqlRel[i - 1,l] := DB_Relationships['RelationshipID'];
              end;
```

```
          end;
        end;
        DB_Relationships.Next;
      until DB_Relationships.Eof;
    end;


  for i := 1 to length(SqlRel) do
    for j := 1 to length(SqlRel[i - 1]) do
    begin
      DB_Relationships.Filtered := false;
      DB_Relationships.Filter := 'RelationshipID = ' + inttostr(SqlRel[i - 1,j - 1]);
      DB_Relationships.Filtered := true;
      DB_Relationships.First;

      repeat
        if (j = 1) and (DB_Relationships.RecNo = 1) then
          SqlPart[i - 1] := SqlPart[i - 1] + ' WHERE '
        else
          SqlPart[i - 1] := SqlPart[i - 1] + ' AND ';

        DB_Tables.Locate('TableID',inttostr(DB_Relationships['PrimaryKeyTableID']),[]);

DB_Columns.Locate('TableID;ColumnID',VarArrayOf([inttostr(DB_Relationships['PrimaryKeyTableID'
]),inttostr(DB_Relationships['PrimaryKeyColumnID'])]),[]);

        SqlPart[i - 1] := SqlPart[i - 1] + DB_Tables['TableName'] + '.' +
DB_Columns['ColumnName'] + ' = ';

        DB_Tables.Locate('TableID',inttostr(DB_Relationships['ForeignKeyTableID']),[]);

DB_Columns.Locate('TableID;ColumnID',VarArrayOf([inttostr(DB_Relationships['ForeignKeyTableID'
]),inttostr(DB_Relationships['ForeignKeyColumnID'])]),[]);

        SqlPart[i - 1] := SqlPart[i - 1] + DB_Tables['TableName'] + '.' +
DB_Columns['ColumnName'];

        DB_Relationships.Next;
      until DB_Relationships.Eof;
    end;

  OLAP_Query_Filter.Filtered := false;
  OLAP_Query_Filter.Filter := 'QueryID = ' + inttostr(OLAPQueryID);
  OLAP_Query_Filter.Filtered := true;
  OLAP_Query_Filter.First;
  repeat
    OLAP_Fact_Dim_Rel.Filtered := false;
    OLAP_Fact_Dim_Rel.Filter := 'FactID = '+inttostr(OLAP_Query['FactID']) + ' and DimensionID
= ' +inttostr(OLAP_Query_Filter['DimensionID']);
    OLAP_Fact_Dim_Rel.Filtered := true;
    OLAP_Fact_Dim_Rel.First;
    if not (OLAP_Fact_Dim_Rel.Bof and OLAP_Fact_Dim_Rel.Eof) then
    begin
      repeat
        OLAP_Dimension_Level.Filtered := false;

OLAP_Dimension_Level.Locate('DimensionID;DimensionLevelID',vararrayof([OLAP_Query_Filter['Dime
nsionID'],OLAP_Query_Filter['DimensionLevelID']]),[]);

        DB_Tables.Locate('TableID',inttostr(OLAP_Dimension_Level['TableID']),[]);

DB_Columns.Locate('TableID;PrimaryKey',VarArrayOf([inttostr(OLAP_Dimension_Level['TableID']),1
]),[]);

        SqlPart[OLAP_Fact_Dim_Rel.RecNo - 1] := SqlPart[OLAP_Fact_Dim_Rel.RecNo - 1] + ' AND '
+ DB_Tables['TableName'] + '.' + DB_Columns['ColumnName'] + ' = ''' +
OLAP_Query_Filter['DimensionValue'] + '''';

        OLAP_Fact_Dim_Rel.Next;
      until OLAP_Fact_Dim_Rel.Eof;
    end;
    OLAP_Query_Filter.Next;
  until OLAP_Query_Filter.Eof;

  for i := 1 to length(SqlPart) do
  begin
    if i = 1 then
```

```
      OLAPSQL := SqlPart[i - 1]
    else
      OLAPSQL := OLAPSQL + ' UNION ALL ' + SqlPart[i - 1];
  end;

  Memo1.Text := OLAPSQL;
end;
```

Näide 2. Aja dimensiooni abitabelite andmete genereerimine

```
procedure FillDateDim;
var DateDT: TDateTime;
    DayI, MonthI, QuarterI, YearI: integer;
begin
  Dim_Query.SQL.Text := 'delete from dim_days';
  Dim_Query.ExecSQL;
  Dim_Query.SQL.Text := 'delete from dim_months';
  Dim_Query.ExecSQL;
  Dim_Query.SQL.Text := 'delete from dim_quarters';
  Dim_Query.ExecSQL;
  Dim_Query.SQL.Text := 'delete from dim_years';
  Dim_Query.ExecSQL;

  Dim_Days.Open;
  Dim_Months.Open;
  Dim_Quarters.Open;
  Dim_Years.Open;
  DateDT := StrToDate('01.01.1754');
  YearI := 0;
  QuarterI := 0;
  MonthI := 0;
  DayI := 0;
  SetRoundMode(rmUp);
  repeat
    if YearOf(DateDT) <> YearI then
    begin
      YearI := YearOf(DateDT);
      Dim_Years.Insert;
      Dim_Years['YearID'] := StrToDate('01.01.'+inttostr(YearI));
      Dim_Years.Post;
    end;

    if Round(MonthOf(DateDT) / 3) <> QuarterI then
    begin
      QuarterI := Round(MonthOf(DateDT) / 3);
      Dim_Quarters.Insert;
      Dim_Quarters['QuarterID'] := StrToDate('01.'+inttostr(QuarterI * 3 -
2)+'.'+inttostr(YearI));
      Dim_Quarters['YearID'] := StrToDate('01.01.'+inttostr(YearI));
      Dim_Quarters.Post;
    end;

    if MonthOf(DateDT) <> MonthI then
    begin
      MonthI := MonthOf(DateDT);
      Dim_Months.Insert;
      Dim_Months['MonthID'] := StrToDate('01.'+inttostr(MonthI)+'.'+inttostr(YearI));
      Dim_Months['QuarterID'] := StrToDate('01.'+inttostr(QuarterI * 3 -
2)+'.'+inttostr(YearI));
      Dim_Months.Post;
    end;

    if DayOf(DateDT) <> DayI then
    begin
      DayI := DayOf(DateDT);
      Dim_Days.Insert;
      Dim_Days['DayID'] := StrToDate(inttostr(DayI)+'.'+inttostr(MonthI)+'.'+inttostr(YearI));
      Dim_Days['MonthID'] := StrToDate('01.'+inttostr(MonthI)+'.'+inttostr(YearI));
      Dim_Days.Post;
    end;

    DateDT := DateDT + 1;
  until DateToStr(DateDT) = '1.01.2051';
end;
```

SUMMARY

„An OLAP Cube Based Development Method of Information Systems”

This Master Thesis hypothesis is that information systems development is possible with lower TCO (Total Cost of Ownership), when developing OLAP (On-Line Analytical Processing) model together with OLTP (On-Line Transaction Processing) model and those two models are in relation to each other. That kind of combined model make possible to develop virtual data warehouse solution, with automatically generated ETL (Extract, Transform, Load) business logic.

The result of the Master Thesis is the customized development method of information systems, which is handling together OLAP and OLTP conceptions for creating the combined data model. Developed innovative meta-model is base for virtual data warehouse, that is providing functionality like classic OLAP dimensional database. In addition is developed universal application for virtual data warehousing, which is using that meta-model’s meta-data together with OLTP database to afford OLAP data-views for end-users.

The second result is virtual data warehouse realization for current company, named like “Nightly Report Generator”, where end-user can define OLAP view by using intuitively understandable OLAP data structure. The application generates then data views nightly and saves the results in Excel Pivot table format.

To continue evolvement of that idea, the plans contain improving meta-model and universal application for realizing automatically generated classic data warehouse together with automatically generated ETL business logic. That kind automatically generated data warehouse can exclude database structure accordance and data quality problems, as most manually developed data warehouses have.