KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Algirdas Kepežinskas

VERSLO VALDYMO SISTEMOS NAVISION ATTAIN IR OLAP PRIEMONIŲ INTEGRAVIMAS

Magistro darbas

Vadovas
doc. L. Nemuraitė

KAUNAS, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

VERSLO VALDYMO SISTEMOS NAVISION ATTAIN IR OLAP PRIEMONIŲ INTEGRAVIMAS

Magistro darbas

Vadovas
doc. L. Nemuraitė

Recenzentas
dr. doc. S. Maciulevičius

Atliko
IFM 0/4 gr. stud.
A. Kepežinskas

KAUNAS, 2006

TURINYS

Lentelių sąrašas

Paveikslėlių sąrašas

ĮVADAS

Sėkmingam verslui nepakanka vien patikimų tiekėjų, ištikimų klientų ir gero kolektyvo. Apie savo verslą reikia žinoti viską, sugebėti rasti jo silpnąsias grandis, tobulintinas vietas, nenašius arba atvirkščiai – perspektyvius sprendimus. Nenaudojant specialių priemonių, tai pasiekti ganėtinai sunku. Laikas ir investicijos į verslo procesų analizę gali būti labai dideli, o naudos duoti mažai. Todėl prieš pradedant kurti naujus sprendimus, reikia deramai susipažinti su esančiomis technologijomis, pasirinkti labiausiai tinkančias ir duodančias daugiausiai naudos.

Šiuo metu Lietuvoje aktyviai diegiama Microsoft Business Solutions bendrovės verslo valdymo sistema Navision Attain. Tai vidutinėms bei stambioms (Lietuvos mastu) įmonėms skirta sistema, įgalinanti efektyviai valdyti įmonės verslo procesus, resursus, išteklius bei apskaitą. Sistema yra reguliariai atnaujinama, pritaikant ją prie naujausių tendencijų verslo informacinių sistemų srityje. Paketo galimybės plečiamos, ir šiuo metu leidžia ne tik valdyti finansus bei komercinę veiklą, bet ir gamybą, ryšius su klientais. Yra galimybė naudotis e-komercijos privalumais bei darbu su tais pačiais duomenimis „per atstumą“. Navision Attain yra pilnai išversta į lietuvių kalbą, o tai padėjo jai sėkmingai įsitvirtinti Lietuvos rinkoje.

Nepaisant visų minėtos sistemos privalumų, jų neužtenka norint užtikrinti sėkmingą verslą. Sistemoje numatyta patogi ataskaitų ruošimo aplinka, tačiau ji nesuteikia galimybių sistemos duomenis paprastai ir greitai pamatyti įvairiais pjūviais. Tokiu atveju kiekvienam naujam pjūviui turi būti kuriama nauja Navision Attain ataskaita. Į naujų ataskaitų kainą įeina sugaištas laikas, pinigai už ataskaitų sukūrimą, licencijos naujoms ataskaitoms kaina. Akivaizdu, jog šiuo atveju nėra naudinga iškilus naujiems poreikiams kurti naujas ataskaitas.

Be lankstumo problemos, didesnės įmonės, turinčios keliasdešimt vienu metu dirbančių vartotojų, dažnai susiduria su techninės įrangos apribojimais. Verslo valdymo sistemos Navision Attain teikiamos ataskaitų generavimo priemonės, nors ir ganėtinai patogios, tačiau reikalauja labai daug duomenų bazės serverio resursų. Kai kurias ataskaitas yra tiesiog būtina leisti dažnai. Tokios ataskaitos yra:

- atsargų apyvarta;
- klientų bei pirkėjų likučiai;
- pirkimo ir pardavimų suvestines.

Verslo valdymo sistemos Navision Attain darbo specifika lemia, kad su atsargų apyvarta susijusios ataskaitos ypač apkrauna darbinę sistemą, tačiau šio tipo ataskaitos yra gyvybiškai svarbios sėkmingai verslo analizei. Jeigu tokias ataskaitas leidžia keletas vartotojų vienu metu, sukurtas apkrovimas yra toks didelis, kad dauguma likusių vartotojų tiesiog nebegali dirbti su sistema, arba jų darbo kokybė labai nukenčia.

Daugelis didesnių Navision Attain vartotojų, taupydami sistemos resursus, stengiasi išnaudoti naktinį sistemos darbo laiką. Kadangi nakties metu dirbančių vartotojų kiekis yra ženkliai mažesnis, lieka daug nepanaudotų resursų. Standartinėmis ataskaitomis šiuos resursus panaudoti neefektyvu – išeinant iš darbo galima paleisti generuoti ataskaitą, tačiau dažnai prireikia tik šiek tiek pakeisti parametrus, o tam jau reikalinga ataskaitą generuoti iš naujo.

Žinant šiuos standartinių ataskaitų generavimo priemonių trūkumus, akivaizdu jog yra būtinybė ieškoti galingesnių duomenų analizės priemonių. Svarbiausi reikalavimai naujosioms priemonėms tiesiogiai atitinka esamų priemonių trūkumus:

- duomenų analizė turi būti lanksti – turi būti galimybė skirtingus duomenų pjūvius matyti be papildomų programavimo darbų. Reikalinga galimybė lengvai keisti ataskaitų išvaizdą, išdėstymą;
- duomenų analizė turi kuo mažiau apkrauti darbinę sistemą – duomenys analizei turėtų būti atkeliami iš darbinės sistemos, tam, kad juos analizuojant nebereikėtų kreiptis į verslo valdymo sistemą;
- duomenų analizės priemonės turėtų palaikyti galimybę daugiausiai resursų reikalaujančias operacijas atlikti nakties metu, tuo siekiant išvengti verslo valdymo sistemos našumo sumažėjimo.

Visas šias savybes atitinka OLAP priemonės [2]. OLAP – tai nuo 1970 metų vystoma duomenų analizės metodika, leidžianti greitai ir patogiai išgauti vartotojui reikiamus duomenis iš duomenų sandėlio. Šiuo metu rinkoje yra nemažai galingų OLAP priemonių, todėl svarbu pasirinkti tinkamą einamai situacijai – labiausiai atitinkančią vartotojų poreikius, geriausiai integruojamą į įmonės infrastruktūrą, ir pan.

Atliekant verslo valdymo sistemos Navision Attain, ir OLAP priemonių integraciją, svarbu pasiekti optimalų šių sistemų tarpusavio bendradarbiavimą. Optimalų tuo, kad duomenų apsikeitimas tarp jų turėtų vykti kuo sklandžiau, kuo mažiau apkrautų verslo valdymo sistemą. Tam reikia sukurti metodiką, kuri padėtų išvengti dažniausiai pasitaikančių klaidų, leistu sukurti sparčiai veikiančią platformą. Kuriant metodiką, svarbu atlikti literatūros analizę – vartotojo poreikiams pritaikyti ETL [11] [12] (duomenų išgavimo, transformavimo ir užkrovimo) procesus, pasirinkti tinkamas duomenų kubų saugyklas bei schemas [7] [8] [10], surasti bei panaikinti duomenų pralaidumą mažinančius, bei resursus naudojančius taškus [11] [13].

Todėl galima teigti, jog šio tyrimo objektas yra konkrečios įmonės, kurios verslo duomenų analizei atlikti nepakanka standartinių verslo valdymo sistemos Navision Attain ataskaitų generavimo priemonių galimybių, verslo analizės sistemos, pagrįstos OLAP technologijomis, sukūrimas. Norint tai pasiekti, reikia atlikti keletą uždavinių:

1. Atlikti įmonės infrastruktūros analizę, apžvelgti OLAP produktų rinką, ir parinkti tinkamą klientui produktą;
2. Atlikti įmonės verslo procesų analizę;
3. Identifikuoti skirtingus vartotojus bei jų poreikius;
4. Remiantis naujausiais literatūros šaltiniais, pasirinkti tinkamus ETL procesus, duomenų sandėlio ir duomenų kubų schemas, ir pan.;
5. Sukurti saugyklos schemą (schemas) atitinkančią skirtingų vartotojų poreikius;
6. Sukurti integravimo priemones duomenų paėmimui iš Navision Attain sistemos;
7. Sukurti duomenų transformavimo priemones, skirtas reikalingam duomenų apdorojimui;
8. Sukurti kiekvieno vartotojų tipo poreikius atitinkančius kubus;
9. Išbandyti ir ištestuoti sistemos prototipą;
10. Sudaryti saugyklų kūrimo bei plėtimo metodiką naujų veiklos procesų analizei.

Analizės kokybė yra kritiškai svarbi, nes tik turint tikslius reikalavimus ir būsimos sistemos parametrus galima sukurti pilnai vartotojų poreikius atitinkančią sistemą. Analizės etapo metu buvo apžvelgti ETL procesai, OLAP priemonės, išanalizuoti ir numatyti tinkami sprendimo keliai.

Nemažiau svarbu buvo ir ištirti bei numatyti būsimosios sistemos integraciją į esamą – tam buvo stengiamasi kiek įmanoma panaudoti jau esama techninę bazę, užtikrinti vartotojų teises bei dumenų sauumą, parinkti reikalingą techninę įrangą, nustatyti jos ryšius su esama įranga.

Be funkcinių reikalavimų buvo atsižvelgta į nefunkcinius reikalavimus saugumui, našumui, integracijai. Apibrėžta, jog sistemą būtina projektuoti taip, kad kubų duomenų atnaujinimas vyktų kuo sparčiau, ir kuo mažiau apkrautų duomenų bazės serverį. Kuriama sistema turi atsižvelgti į esamas *Windows Active Directory* vartotojų teises, turi būti galimybė apriboti kubų duomenų pasiekiamumą. Duomenų kubai turi būti prieinami per jiems skirtą terminalinį serverį.

Siekiant, kad sukurta sistema kuo tiksliau atitiktų vartotojų poreikius, buvo numatyti kokybės įvertinimo kriterijai:

- Plečiamumas - kokios reikalingos laiko ir resursų sąnaudos norint išplėsti kubą naujomis dimensijomis ar pjūviais?
- Našumas - kiek laiko trunka atnaujinimas? Kiek vidutiniškai trunka pjūvių generavimas?
- Patikimumas - ar sistemos duomenys korektiški? Ar dažnai pasitaiko klaidų, kurias reikia taisyti? Ar numatyti visi galimi naudojimo atvejai?
- Stabilumas - ar sistema veikia stabiliai? Ar sėkmingai vyksta automatinis atnaujinimas? Kiek buvo atnaujinimo sutrikimų?

Kad projektas vyktų sėkmingai, svarbu įvardinti galimus rizikos faktorius, bei būdus juos valdyti – tokiu atveju, susidūrus su numatyta problema, bus galima sutaupyti laiko jos sprendimui, bei sumažinti viso projekto žlugimo riziką.

Sukūrus sistemos prototipą, buvo tiriami optimizavimo būdai. Tam tikslui buvo pasinaudota turima literatūra bei žiniomis apie sistemą įgytomis vykdant projektą. Buvo atliktas keleto lygių optimizavimas:

- kubų schemų optimizavimas;
- techninės įrangos optimizavimas;
- duomenų sandėlio optimizavimas.

Įgytos žinios suformuluotos kaip bendro pobūdžio rekomendacijos, kurias galima naudoti diegiant kitas OLAP sistemas, ar kuriant priemones kitų verslo procesų analizei.

Darbo metu buvo sukurti keturi duomenų kubai – pirkimų, pardavimų, atsargų ir pirkėjų skolų. Kiekvienam kubui sukurtos reikalingi rodiniai, transformacijos, nustatytas jų atnaujinimas. Sukurti kubai įkelti į OLAP serverį bei atliktas jų testavimas. Vartotojams sukurtos pavyzdinės ataskaitos bei trumpas prisijungimo prie kubų žinynas.

Eksperimentinio tyrimo metu sukurti kubai optimizuoti pagal numatytą planą, tuo iki 80% sumažinant duomenų bazės serverio apkrovimą kubų naujinimo metu.

Darbo pabaigoje atliktas naujosios sistemos kokybės tyrimas, apklausiant sistemos vartotojus bei administratorius apie svarbiausius duomenų sandėlio kokybės rodiklius. Gauti rezultatai suklasifikuoti bei pateikti rezultatų apibendrinime.

Trumpai apžvelgta Microsoft verslo informacijos sistemų ateities vizija, pristatyta naujoji BI platforma susidedanti iš trijų pagrindinių komponentų:

- integravimo servisų
- analizės servisų
- ataskaitų servisų

Darbo struktūra:

1. Analizės etapae aprašomi įmonėje vykstantys verlso procesai, identifikuojami vartotojų tipai bei jų poreikiai, nustatomi nefunkciniai reikalavimai. Atliekama literatūros analizė ir pasirenkamas problemos sprendimo būdas.
2. Reikalavimų išgavimo etape pateikiami tikslūs vartotojų reikalavimai kiekvienam duomenų kubui.
3. Projektavimo etape išgauti reikalavimai paverčiami detalia specifikacija, nustatomos naudojamos verslo valdymo sistemos duomenų bazės lentelės, jų sąryšiai su dimensijomis, matais bei skaičiuojamaisiais laukais. Šio etapo metu taip pat aprašomi atnaujinimai, bei sukuriamas duomenų srautų modelis.
4. Eksperimentinio tyrimo skyriuje aprašomas sukurtų duomenų kubų optimizavimas.
5. Diegimo skyriuje aprašytas naujosios sistemos patalpinimas į kliento tinklą.

6. Sistemos kokybės tyrimo skyriuje pateikiami sistemos kokybės vertinimo kriterijai bei vartotojų apklausos rezultatai.
7. Darbo pabaigoje apibendrinami gauti rezultatai, apžvelgiama verslo informacijos rinkos ateitis, bei pateikiamos darbo išvados.

Darbo tema padarytas pranešimas „Verslo valdymo sistemos Navision Attain ir OLAP duomenų analizės integracija“, pristatytas 2005 metais sausio 27 dieną vykusioje konferencijoje „Informacinės Technologijos ‘2005“, išspausdintas straipsnis konferencijos leidinyje.

1. VERSLO VALDYMO SISTEMOS NAVISION ATTAIN IR OLAP PRIEMONIŲ INTEGRAVIMO ANALIZĖ

1.1. ANALIZĖS TIKSLAS

Navision Attain sistema Lietuvoje jau yra įdiegta daugeliui bendrovių, tame tarpe ir tokioms didelėms įmonėms kaip AB „Pieno žvaigždės“, AB „Rokiškio sūris“ ir t.t. Stambaus masto įmonėms, turinčioms keletą padalinių, aptarnaujančioms daugelį klientų per dieną, vykdančioms didelių apimčių pirkimo/pardavimo operacijos, yra kritiškai svarbu turėti kuo išsamesnę informaciją apie savo veiklos rodiklius. Tokią informaciją suteikti standartinėmis Navision Attain ataskaitų ruošimo priemonėmis nėra patogu:

- Techninė įranga: su sistema dirba daug vartotojų, o leidžiama ataskaita smarkiai apkrauna visą sistemą, kas daro neigiamą įtaką sistemos našumui ir stabilumui.
- Kiekvienam informacijos pjūviui turi būti ruošiama nauja ataskaita. Programavimo darbai trunka, bei kainuoja, tad dažnai vengiama užsakinėti smulkius pakeitimus.
- Naktimis techninė įranga nedirba – būtų naudinga tuo metu ją naudoti.

Žinant šiuos trūkumus, būtų naudinga rasti geresnį problemos sprendimo būdą. Išklausius vartotojų pageidavimus, galima matyti, jog OLAP duomenų analizė [2] ganėtinai tiksliai atitiktų reikalavimus.

Šios analizės tikslas yra išsiaiškinti Navision Attain ir OLAP duomenų analizės integravimo galimybes. Reikia nustatyti, ar abi sistemos yra techniškai suderinamos, ar OLAP duomenų analizė tiksliai atitinka vartotojų poreikius, ar sukurta sistema finansiniu požiūriu būtų efektyvesnė už esamas Navision Attain ataskaitų ruošimo priemones.

1.2. OLAP PRODUKTŲ APŽVALGA

Siekiant patenkinti užsakovų poreikius, tinkamai išspręsti pateiktą problemą, be galo svarbu yra tinkamai įvertinti galimus jos sprendimo variantus. Vienas iš svarbiausių pasirinkimų yra tinkamo OLAP produkto parinkimas. Šiuo metu yra siūloma nemažai skirtingas galimybes turinčių sistemų. Tai ir Oracle OLAP, IBM DB2, Hyperion, Business Objects, MicroStrategy, Cognos, Essbase, SAP BW, Microsoft Analysis Services.

Trumpas keleto didesnių OLAP produktų svarbiausių savybių palyginimas pateikiamos pirmoje lentelėje.

1.1 lentelė. OLAP produktų savybių palyginimas

Microsoft	IBM	Oracle	Hyperion
Pateikimas			
Visi Microsoft pateikiami duomenų sandėlių produktai yra pateikiami	Pateikiama kartu su IBM DB2	Parduodama atskirai	Tik OLAP produktas

kartu su Microsoft SQL Server 2000, ir papildomai nieko nekainuoja.			
Naudojama duomenų bazė			
Microsoft SQL Server 2000	IBM DB2	Oracle	Essbase, ar kita
Sandėlio kūrimas ir valdymas			
Duomenims užkrauti, transformuoti ir išsaugoti naudojami DTS (Data Transformation Services). DTS palaiko OLE DB duomenų šaltinius (ODBC, failai, Microsoft Office). Partneriai teikia papildomų šaltinių palaikymus. DTS COM architektūra leidžia lengvai plėsti DTS funkcionalumą. Standartiškai nepalaikomi papildantys atnaujinimai (angl. *incremental updates*). Siūlomos atskiros priemonės reliacinių ir OLAP duomenų saugyklų valdymui.	Duomenims užkrauti ir atnaujinti naudojamas Data Warehouse Center. Palaikomi duomenų šaltiniai yra IBM, Informix, Microsoft, Oracle, Sybase reliacinės duomenų bazės, tekstiniai failai, ODBC bei OLE DB sąsajos. DB2 Warehouse Manager praplečia duomenų sandėlių kūrimo bei valdymo galimybes, palaiko IMS ir VSAM duomenų šaltinius, papildomas transformacijas bei UDF ir SP (User Defined Functions ir Stored Procedures) transformacijas. DB2 Warehouse Manager taip pat palaiko paskirstytą, agentais apgrįstą duomenų išgavimą bei perkėlimą. Taip pat palaikomas vienas nuo kito priklausomų duomenų sandėlių kūrimas. ETL procesas gali būti išplėstas naudojant partnerių siūlomą ETI*EXTRACT papildą. Duomenų valymas gali būti išplėstas naudojant dar vieną papildą – Vality Integrity.	Reliaciniams duomenų sandėliams kurti ir valdyti naudojamas Oracle Warehouse Builder komponentas. Palaikomi duomenų šaltiniai yra reliacinės DB, tekstiniai failai, SAP duomenys. Rezultatai gali būti saugojami tik Oracle DB. Duomenims galima atlikti pirminį užkrovimą, atnaujinimą, papildančius atnaujinimus. Transformacijos aprašomos diagramomis, iš jų sugeneruojamas PL/SQL kodas. Pure*Extract tai atskirai parduodamas produktas, palaikantis DB2, IMS, QSAM, VSAM duomenų šaltinius. Pure*Integrate, tai taip pat atskirai parduodamas duomenų valymo priedas.	Duomenų sandėliams kurti ir valdyti yra skirtas Hyperion Integration Server. Palaikomi duomenų šaltiniai yra reliacinės duomenų bazės. Rezultatų duomenų bazė yra Essbase. Palaikomas pirminis užkrovimas, atnaujinimas, papildantis atnaujinimas. Laikui bėgant produktas bus integruotas į Essbase.
OLAP priemonės			
Microsoft Analysis Services teikia OLAP saugyklų kūrimo ir valdymo priemonės. Palaikomi MOLAP, ROLAP, HOLAP saugyklų modeliai. Siūlomas gausus analitinių funkcijų rinkinys duomenims apdoroti. Duomenų saugyklos gali būti padalintos į atskiras particijas, palaikomas sudėtingas apsaugos mechanizmas.	OLAP duomenų sandėliams kurti naudojamas pervardintas Hyperion produktas – DB2 Integration Server.	OLAP duomenų sandėliams kurti naudojamas Oracle OLAP Services komponentas, kuris yra parduodamas atskirai.	Pagrindinis produkto privalumas yra gausios OLAP galimybės. Palaikoma per 350 analitinių funkcijų. Ateityje numatoma palaikyti Java for OLAP API. Palaikomi MOLAP, ROLAP ir HOLAP duomenų saugyklų modeliai. OLAP saugyklos gali būti paskirstytos bei padalintos į atskiras particijas.
Duomenų išgavimas (angl. *data mining*)			
Palaikomas duomenų išgavimas. Siūlomi du algoritmai: sprendimų medžiai bei klasterizavimas.	Skirtas atskirai parduodamas tačiau pilnai integruojamas produktas Intelligent Miner. Jis gali naudoti DB ir kaip	Duomenų išgavimui skirtas vėlgi atskirai parduodamas Oracle Data Mining produktas. Pagristas Darwin technologija,	Nėra palaikomas.

	duomenų šaltinį, ir kaip rezultatų saugyklą. Produktas palaiko daugelį duomenų išgavimo algoritmų.	nupirkta iš Thinking Machines Corporation. Produktas palaiko daugelį duomenų išgavimo algoritmų.	
Sąsajos			
Palaikomos šios sąsajos: OLE DB OLE DB for OLAP OLE DB for Data Mining COM	Dauguma SQL dialektų ODBC OLE DB IMS VSAM HEAPI	PL/SQL Express SNAPI Express SPL	Hyperion Essbase API Java for OLAP API
Metaduomenys			
Už metaduomenis atsakingi Metadata Services (anksčiau – Microsoft Repository). Tai Microsoft SQL Server 2000 komponentas atsakingas už informacijos apie duomenų sandėlius bei jų sąryšius saugojimą bei pateikimą. Jis įgyvendintas naudojant rinkinį OIM (Open Information Model) ir COM interfeisus. OIM yra objektų modeliai apibūdinantys duomenų sandėlių duomenis, procesus bei sąsajas, o taip pat jų sąryšius. Čia taip pat saugoma informacija reikalingi pasiekti duomenų sandėlio informaciją.	Nuo 7 versijos, IBM palaiko centralizuotą metaduomenų saugyklą, kuri valdoma Data Center Manager komponento pagalba. Metamodeliai kuriami naudojantis arba IBM patentuotu formatu, arba CWM (Common Warehouse Model). Abu jie pasiekiami naudojantis SQL sąsaja.	Už metaduomenų valdymą reliacinėse duomenų bazėse, Express OLAP duomenų sandėliuose, Oracle Discoverer ir Oracle Reports BI įrankiuose yra atsakingas Oracle Warehouse Builder komponentas. Metaduomenys yra saugojami Oracle*I lentelėse ir pasiekiami naudojant PL/SQL. Metaduomenys yra saugojami CWM formatu.	Hyperion Integration Server metaduomenis (duomenų sąryšius, atvaizdus, dimensijas, hierarchijas, skaičiavimus, verslo taisykles) saugomos naudojamos duomenų bazės lentelėse. Metaduomenys pasiekiami naudojant SQL užklausas. Kaip metaduomenų saugykla gali būti naudojama ir Informatica PowerCenter.

Daugelis iš minėtų sistemų daugiau negu pilnai patenkintų šio projekto keliamus reikalavimus, todėl renkamasi buvo atsižvelgiant ne į siūlomas galimybes, tačiau į suderinamumą, diegimo ypatumus, ateities perspektyvas.

Remiantis šiais kriterijais, kaip OLAP analizės platforma buvo pasirinkta Microsoft Analysis Services [1] [3] [5]. Toks sprendimas buvo priimtas dėl keleto priežasčių:

- Navision Attain sistema turi galimybę būti įdiegta naudojant Microsoft SQL Server duomenų bazę, o ateityje tai bus vienintelė palaikoma duomenų bazė.
- Kaina. Kadangi Microsoft Analysis Services yra pateikiami kartu su Microsoft SQL Server duomenų baze, tai papildomos išlaidos minimalios.
- Microsoft netolimoje ateityje planuoja smarkiai plėsti savo verslo analizės rinką, siūlyti naujus produktus ir naujas analizės galimybes.

1.3. PROJEKTO APLINKA BEI SPRENDŽIAMA PROBLEMA

Analizės užsakovas yra stambi pieno produktų gamybos įmonė. Jų verslo apimtis yra pakankamai didelė, jog būtų poreikis ieškoti optimalesnių duomenų analizės būdų. Analizė apima šios įmonės veiklą, tačiau lengvai gali būti pritaikyta ir kitoms stambios įmonėms.

Tyrimo sritis – OLAP priemonių taikymas įmonėse, verslo valdymui naudojančiose Microsoft Business Solutions Navision Attain sistemą.

Tyrimo objektas – konkrečios įmonės, naudojančios Navision Attain apskaitos priemones, veiklos analizės sistema, kurią reikia sukurti naudojant Microsoft SQL Server duomenų analizės įrankius.

Tyrimo problema – pardavimų veikla įmonėje generuoja ypač intensyvius duomenų srautus, kurių analizei reikalingos galingos priemonės. Įprastinių ataskaitų nepakanka, todėl reikia ieškoti geresnių sprendimų.

Vienas iš jų – OLAP duomenų analizė. Reikia ištirti OLAP įrankių suderinamumą su įmonėje įdiegta Navision Attain verslo valdymo sistemą, jos tinkamumą vartotojų poreikiams tenkinti, jos našumą bei įtaką įmonės veiklai, palyginti SQL Server duomenų analizės įrankių galimybes su Navision ataskaitų kūrimo priemonėmis.

Norint įgyvendinti OLAP duomenų analizę, reikia įvykdyti keletą etapų:

- Saugyklos schemų ir poschemių projektavimas ir kūrimas
- Integravimo su Navision projektavimas ir kūrimas
- Duomenų transformacijų kūrimas bei pritaikymas Navision Attain duomenims
- Saugyklos pildymo procesų kūrimas
- Analizės kubų projektavimas, kūrimas bei testavimas

Reikia išsiaiškinti, ar šios veiklos atsiperka, tai yra ar OLAP duomenų analizės priemonės yra efektyvios laiko, kaštų bei galimybių atžvilgiu.

1.4. APLINKOS IR VERSLO PROCESŲ ANALIZĖ

Įmonės pardavimų procese dalyvauja daugybė objektų, tačiau galima išskirti keletą pagrindinių:

1.2 lentelė. Pagrindiniai pardavimų proceso objektai

Objektas	Aprašymas
Prekė	Parduodama produkcija
Tara	Apyvartinė tara – tara atiduodama pirkėjui, ir grįžtanti iš jo atgal
Pirkėjas	Produkcijos pirkėjas
Sistema, sistemos lygiai	Pirkėjų klasifikacija – skirstoma į keturis sistemų lygius
Mokėtojas	Klientas apmokantis už parduotą produkciją

Vadybininkas	Prekybos agentas atsakingas už tam tikrą pirkėją/mokėtoją

Kiekvieną iš šių objektų apibūdina keletas svarbiausių dimensijų:

1.3 lentelė. Prekės objekto dimensijos

Objektas	Aprašymas
Stambi prekės grupė	Produkcijos klasifikatorius
Prekės statistinė grupė	Produkcijos klasifikatorius
Prekės registravimo grupė	Produkcijos registravimo nustatymas iš Navision Attain
Riebumas	Produkcijos riebumas %
Gamintojas	Produkcijos gamintojo kodas
Brand	Produkcijos prekinis ženklas
Fasuotė	Produkcijos fasuotė
Pakuotė	Produkcijos pakuotė

1.4 lentelė. Pirkėjo/mokėtojo objekto dimensijos

Objektas	Aprašymas
Geografinė vieta	Pirkėjo/mokėtojo geografinė vieta
Funkcinis padalinys	Funkcinis padalinys kuriam priklauso pirkėjas/mokėtojas
Vadybininkas	Atsakingas vadybininkas
Rajonas	Rajonas
Mokėtojas	Už pirkėjo pirkimus mokantis mokėtojas
Sistema	Pirkėjo/mokėtojo klasifikavimas į sistemas
Sistemos klasė	
Sistemos lygis	
Sistemos 4 lygis	
Šalis	Pirkėjo/mokėtojo šalis
Pirkėjo registravimo grupė	Registravimo nustatymas iš Navision Attain
Regionas	Regionas

Pardavimų procesas susideda iš šių etapų:

1. Užsakymo pildymas - Navision Attain sistemoje tam tikram pirkėjui pildomas pardavimo užsakymas. Suvedamos perkamos prekės, įkeliama apyvartinė tara, patvirtinamos kainos. Užsakymas pateikiamas.
2. Užsakymo siuntimas – užsakytos prekės išsiunčiamos nurodytam pirkėjui (gavėjui).
3. Registruojama sąskaita faktūra pirkėjo mokėtojui.
4. Pirkėjui grąžinus apyvartinę tarą, registruojamas taros grąžinimas.

Atlikus visus šiuos žingsnius, užsakymas tampa pilnai įvykdytas. Analitikams reikalingi duomenys apie šį procesą pateikiami **1.5.2 skyriuje, Vartotojų tikslai ir problemos.**

1.5. VARTOTOJŲ ANALIZĖ

1.5.1. VARTOTOJŲ AIBĖ, TIPAI IR SAVYBĖS

Nagrinėjamoje įmonėje pardavimų procesas susideda iš daugelio skirtingų procesų. Už juos yra atsakingi skirtingi vartotojų tipai. Pagrindiniai vartotojų tipai yra šie:

- Finansininkai
- Pardavimų analitikai
- Sandėlio darbuotojai
- Personalo darbuotojai

Kadangi kiekvienas iš šių darbuotojų dirba su ta pačia Navision Attain sistema, tai jie visi turi priėjimą prie tų pačių duomenų, tačiau juos nori matyti skirtingais pjūviais. Kartais šių vartotojų poreikiai tiek skirtingi, kad jiems patenkinti neužtenka vien tos pačios duomenų struktūros skirtingo pateikimo, tačiau reikia ir skirtingų duomenų kubų.

1.5.2. VARTOTOJŲ TIKSLAI IR PROBLEMOS

Trumpai apžvelgsime skirtingus vartotojų tipus, jų pageidaujamus matyti duomenis bei šių duomenų pateikimo galimybes.

- **Finansininkai**. Šiems vartotojams aktualiausia yra informacija apie ilgalaikius įmonės veiklos rodiklių pokyčius, darbuotojų darbo efektyvumą, skirtingų geografinių vietovių ir funkcinių centrų darbo statistiką ir panašiai. Šie vartotojai turės turėti priėjimą **prie visų** duomenų, nes jiems svarbu analizuoti visas įmonės veiklos šakas.

  Antra nemažiau svarbi finansininkų darbo šaka yra pirkėjų skolų analizė – informacija apie pirkėjams atliktus pardavimus, gautus grąžinimus, mokėjimus, atiduotą ir grąžintą tarą, pradelstas skolas ir panašiai.

- **Pardavimų analitikai** – jiems svarbiausia yra matyti informacija apie įmonės atliktus faktinius pardavimus, jų pasiskirstymą tarp geografinių vietovių ir funkcinių padalinių, skirtingų prekių pardavimų kitimą, pardavimus pagal pirkėjų klasifikatorius, prekių grupes, suteiktas nuolaidas ir panašiai. Tai yra stambi duomenų grupė, todėl jai yra numatomas atskiras duomenų kubas.

- **Sandėlio darbuotojai** - šiems darbuotojams aktualiausia informacija yra žaliavų apyvartos žiniaraštis. Šiai ataskaitai generuoti Navision Attain priemonėmis yra sunaudojama daugiausia resursų, todėl ją yra aktualiausia perkelti į galingesnes analizės priemones. Svarbu, kad sandėlio darbuotojai galėtų matyti tik jų aptarnaujamo sandėlio informaciją. Norima matyti informacija yra gautas, perduotas, parduotas, pirktas, pajamuotas, nurašytas prekių kiekis bei savikainos, prekių neto ir bruto svoriai. Informaciją reikia klasifikuoti pagal prekes, jų grupes ir

pogrupius, pakuotes, fasuotes bei kitus prekių klasifikatorius. Taip pat svarbu matyti prekių pasiskirstymą tarp skirtingų sandėlių, perdavimus iš vienų sandėlių į kitus, konkrečias dėžes, ir t.t.

- **Personalo darbuotojai.** Šiems darbuotojams reikalinga matyti produkcijos pardavimus pagal konkrečius prekybos agentus. Šią informaciją galima gauti iš pardavimų analitikams numatyto pardavimų kubo.

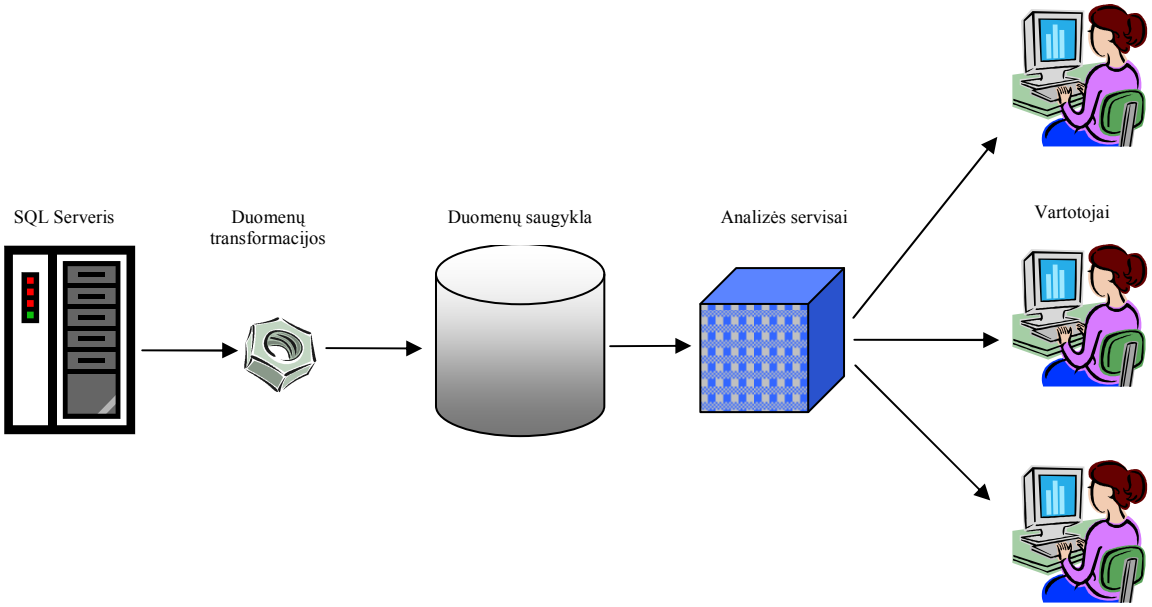
1.6. ARCHITEKTŪROS IR DIEGIMO ANALIZĖ

Verslo valdymo sistema Navision Attain gali būti įdiegta dviem variantais:

1) Naudojant SQL Server 2000 duomenų bazę
2) Naudojant Navision Attain gimtąją (angl. *native*) duomenų bazę

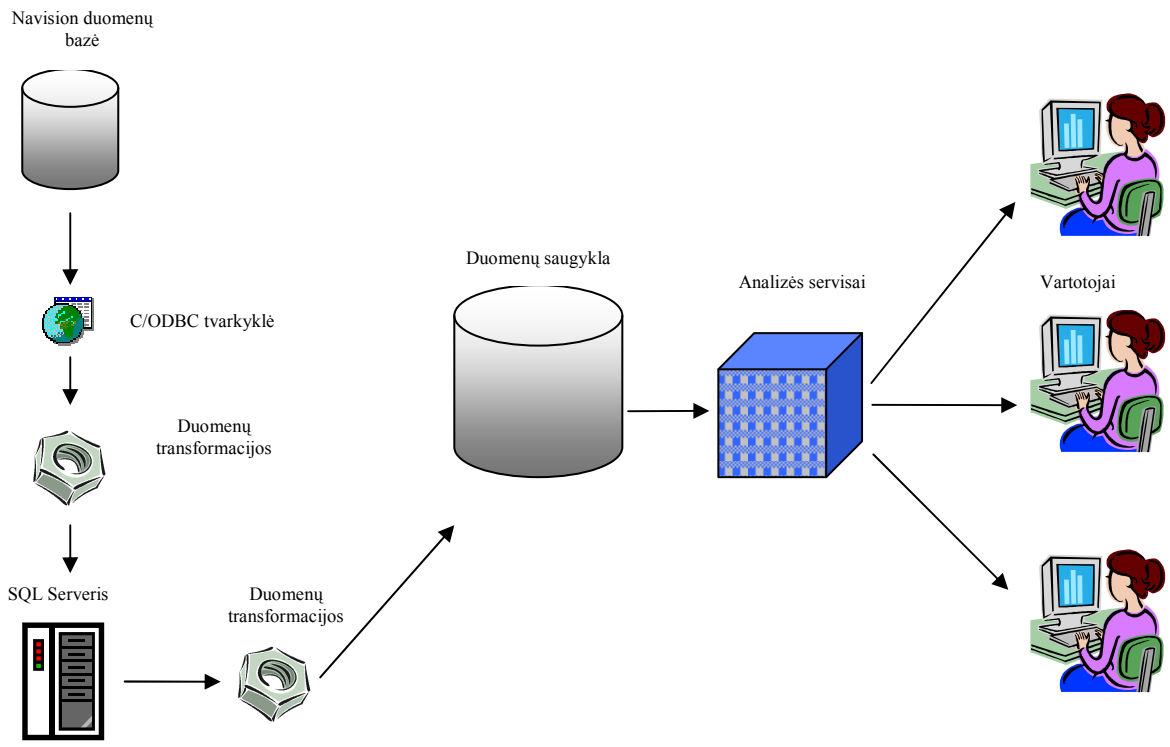
Priklausomai nuo sistemos diegimo, skiriasi ir integracijos su OLAP duomenų analize architektūra.

Pirmuoju atveju, sistemos architektūra atrodo štai taip:



1.1 pav. Integravimo architektūra naudojant SQL Server duomenų bazę

Antruoju atveju architektūra tampa sudėtingesnė:

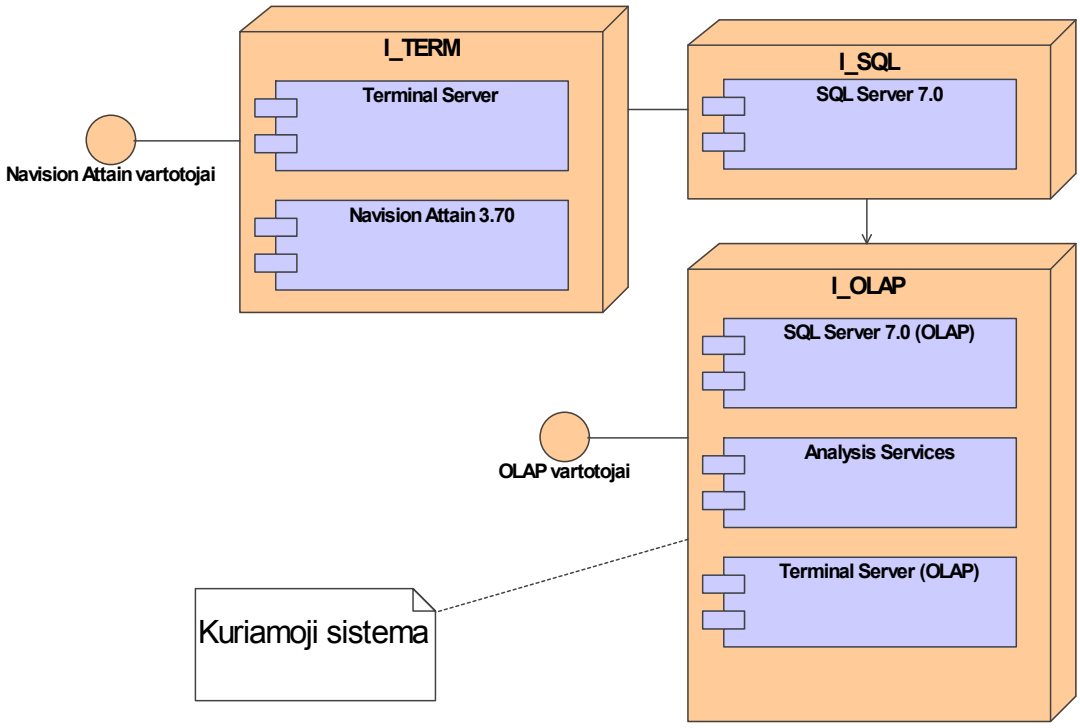


1.2 pav. Integravimo architektūra naudojant Navision Attain gimtąją duomenų bazę

Naujas tarpinis žingsnis (duomenų perkėlimas į SQL serverio lenteles) atsiranda dėl to, jog Navision Attain duomenų struktūra yra labai sudėtinga, laukų pavadinimuose naudojamos lietuviškos raidės bei specialieji simboliai (pvz. tarpai). Dėl to atsiranda problemos naudojant šias lenteles tiesiogiai OLAP kubų redaktoriuje. Nemažiau svarbu yra ir tai, jog įvairūs klasifikatoriai į duomenų lenteles nebūtinai yra įrašomi ir reikia numatyti situacijas, kai tam tikri įrašai iš duomenų lentelių bus praleisti, todėl, kad dimensijų lentelėse nėra atitinkamų reikšmių.

Matome, jog gauta sistemos architektūra beveik tiksliai atitinka ETL (angl. *extract, transform and load*) metodologiją. Pirminiai (Navision Attain sistemos) duomenys yra nuskaitomi iš esamos SQL duomenų bazės. Tada, naudojant įvairius vaizdus (angl. *view*), sąryšius bei DTS (angl. *data transformation services*) transformuojami į reikiamą struktūrą ir galiausiai išsaugojami naujojoje OLAP duomenų saugykloje.

Pritaikius naująją sistemą prie esančios įmonėje informacinės sistemos, galima sukurti tokį preliminarų naujosios infrastruktūros vaizdą:



1.3 pav. Preliminari sistemos diegimo schema

Čia I_TERM yra darbinis terminalinis serveris – prie jo jungiasi Navision Attain vartotojai, jame paleistas Navision Attain klientas. Iš šio kompiuterio yra jungiamasi į I_SQL serverį, kuris ir yra pagrindinė duomenų bazė. Mūsų diegiamas naujasis komponentas yra I_OLAP kompiuteris, kuris tarnaus ir kaip terminalinis serveris OLAP vartotojams, ir kaip OLAP serveris. Tai yra kompiuteris kuris tarnaus kaip naujosios sistemos serveris. Jis bendraus su pagrindiniu SQL duomenų bazės serveriu I_SQL – iš jo ims realius duomenis, apdoros, ir savyje saugos transformuotus duomenis.

1.7. DARBO TIKSLAI IR UŽDAVINIAI

Diegiama sistema turėtų įgyvendinti keletą jai numatytų tikslų:

1. Sumažinti pagrindinės sistemos apkrovimą (diskų masyvo naudojimą) tuo padidinant pagrindinės sistemos efektyvumą
2. Suteikti patogesnes priemones informacijos išgavimui iš Navision Attain sistemos, lengvinti vartotojų darbą
3. Efektyviau išnaudoti naktinį sistemos darbo laiką
4. Suteikti papildomą informaciją apie įmonės veiklą
5. Tarnauti kaip mėnesinių bei metinių ataskaitų pateikimo priemonė

Pilnai įdiegus sistemą, ir jai atliekant visus čia iškeltus tikslus, ji turėtų žymiai palengvinti įmonėje vykstantį pardavimų procesą. Pasiteisinus šiai sistemai, ji būtų diegiama ir į kitus įmonės veiklos procesus.

Kad šie tikslai būtų pasiekti, reikės atlikti tokias užduotis:

11. Sukurti saugyklos schemą (schemas) atitinkančią skirtingų vartotojų poreikius
12. Sukurti integravimo priemones duomenų paėmimui iš Navision Attain sistemos
13. Sukurti duomenų transformavimo priemones, skirtas reikalingam duomenų apdorojimui
14. Sukurti kiekvieno vartotojų tipo poreikius atitinkančius kubus
15. Išbandyti ir ištestuoti sistemos prototipą
16. Sudaryti saugyklų kūrimo bei plėtimo metodiką naujų veiklos procesų analizei.

1.8. REIKALAVIMAI DUOMENIMS

Kuriant duomenų analizės sistemą numatyta, jog už duomenų tikslumą, pilnumą ir korektiškumą atsakinga Navision Attain sistema. Tai ir būtina verslo procesams, nes šios duomenų savybės būtinos sėkmingam darbui su visa sistema. Atsižvelgiant į tai, atskirų reikalavimų duomenims nėra keliama.

Duomenys į analizės duomenų saugyklą bus perkeliami kiekvieną naktį, todėl svarbu pabrėžti, jog OLAP priemonėmis matomi duomenys **nebus šviežiausi**. Tai daryti verčia sistemos apkrovimas, kuris tiktai naktį yra šiek tiek mažesnis.

Svarbu patikrinti, jog OLAP priemonėmis atvaizduoti duomenys sutaptų su matomais Navision Attain sistemoje:

- išliktų lietuviškos raidės
- klasifikatoriai būtų pavadinti pavadinimais, o ne vien kodais
- duomenys tam tikrai datai sutaptų su Navision Attain ataskaitomis išgautais duomenimis

1.9. NEFUNKCINIAI REIKALAVIMAI IR APRIBOJIMAI

1.9.1. REIKALAVIMAI NAŠUMUI

Svarbus nefunkcinis reikalavimas sistemai yra našumas, tiek atnaujinant OLAP kubus, tiek ir generuojant jais įvairius pjūvius. Svarbu projektuoti sistemą taip, kad atnaujinimas vyktų kuo sparčiau, ir kuo mažiau apkrautų esančią sistemą. Tai pasiekti galima dviem būdais:

- Apribojant analizuojamų duomenų kiekį, griežtai apibrėžiant norimus matyti pjūvius ir prastinant duomenų saugyklą
- Replikuojant esančios sistemos duomenis į atskirą SQL serverį, kuris bus skirtas tiktai OLAP duomenų analizei

Kol numatytas įgyvendinti pirmasis sprendimas, tačiau prireikus sistema bus pertvarkyta į antrąjį variantą.

Nemažiau svarbus yra ir pjūvių generavimo greitis. Šį greitį padidinti galima keliais būdais:

- sukurti pakankamai pradinių apskaičiavimų (angl. *aggregations*)
- naudoti optimizacijas pagal panaudojimą (angl. *usage based optimizations*)

- apriboti analizuojamų duomenų kiekį, griežtai apibrėžiant norimus matyti pjūvius ir prastinant duomenų saugyklą

Pradiniam sistemos diegimui bus naudojami visi šie sprendimai. Vėliau, esant reikalui, įvairius apribojimus bus galima mažinti, naudojant galingesnę techninę įrangą.

1.9.2. REIKALAVIMAI INTEGRAVIMUI

Be našumo reikalavimų, sistemai keliami ir integravimo reikalavimai. Svarbu, jog nauja analizės sistema kuo labiau integruotųsi į esamą informacinę įmonės sistemą. Sistema turėtų dirbti be įsikišimo, todėl svarbu užtikrinti, kad:

- visada būtų nenutrūkstantis ryšys tarp OLAP serverio ir pagrindinio SQL serverio
- kiekvieną naktį be trikdžių įvyktų OLAP kubų atnaujinimas
- OLAP serveris integruotųsi su esančia Windows Active Directory vartotojų autentifikavimosi sistema
- OLAP serveris visada būtų pasiekiamas per terminalą, bei tiesiogiai per analizės klientus

Pabrėžtina, jog šiai įmonei ypatingai yra svarbus nenutrūkstantis darbas su sistema, todėl yra tiesiog būtina užtikrinti kuo geresnę visų sistemos komponentų integraciją.

1.9.3. REIKALAVIMAI SAUGUMUI

Akcentuojamas ir duomenų saugumas. Kadangi įmonė veikia visos Lietuvos mastu, turi daugelį atsakingųjų centrų, funkcinių padalinių, filialų ir kitų administracinių vienetų, yra svarbu užtikrinti, jog kiekvienas vienetas turi priėjimą tik prie „savo“ duomenų. Tai galima užtikrinti naudojantis MS SQL Server Analysis Services esančiomis autorizavimo priemonėmis. Skirtingoms Windows Actine Directory vartotojų grupėms bus suteiktas priėjimas prie skirtingų dimensijų verčių, tuo užtikrinant, kad jie matys tik dalį viso pjūvio.

1.10. RIZIKOS FAKTORIŲ ANALIZĖ

Turint omeny, jog sistemai be galo svarbus prieinamumas, reikia iš anksto numatyti galimus rizikos faktorius, ir būdus jas valdyti.

1.5 lentelė. Rizikos faktoriai

Rizikos faktorius	Svarbumas	Rizikos faktoriaus eliminavimo būdas
OLAP serverio nepasiekiamumas	Kritinis	Užtikrinti, jog OLAP serveris yra greitai pasiekiamas fiziškai, kad esant reikalui, jį būtų galima kuo greičiau sutvarkyti
Duomenų nekorektiškumas	Aukštas	Atlikti išsamų pradinį testavimą, tikrinant rezultatus su Navision Attain ataskaitų rezultatais; turėti

		pavyzdinius nekorektiškus duomenis greitam problemos identifikavimui; turėti terminalinį priėjimą prie OLAP serverio problemos tvarkymui
Nepavykęs automatinis atnaujinimas	Aukštas	Įdiegti pranešimų sistemą, perspėjančią apie nepavykusį atnaujinimą
Norimo pjūvio atvaizdavimo galimybės nebuvimas	Žemas	Iš anksto suderinti norimus matyti pjūvius, numatyti jų įgyvendinimą; sutarti bei parengti alternatyvius sprendimus
Sistemos lūžis	Žemas	Kurti kasdienines atsargines kopijas
Nenumatytos situacijos	Aukštas	Pasirašyti aptarnavimo sutartį, numatyti nenumatytų situacijų valdymo tvarką, gretai reaguoti į iškilusias kritines problemas

1.11. REZULTATO KOKYBĖS KRITERIJAI

Įvertinus visus numatytus funkcinius ir nefunkcinius reikalavimus, būsimą sistema bus galima įvertinti remiantis šiais kokybės kriterijais:

1.6 lentelė. Kokybės įvertinimo kriterijai

Kriterijus	Paaiškinimas
Plečiamumas	Kokios reikalingos laiko ir resursų sąnaudos norint išplėsti kubą naujomis dimensijomis ar pjūviais?
Stabilumas	Ar sistema veikia stabiliai? Ar sėkmingai vyksta automatinis atnaujinimas? Kiek buvo atnaujinimo sutrikimų?
Našumas	Kiek laiko trunka atnaujinimas? Kiek vidutiniškai trunka pjūvių generavimas?
Patikimumas	Ar sistemos duomenys korektiški? Ar dažnai pasitaiko klaidų, kurias reikia taisyti? Ar numatyti visi galimi naudojimo atvejai?
Efektyvumas	Ar pasiteisino sistemos diegimas? Ar sunaudojama mažiau resursų nei atitinkamus duomenis išgaunant Navision Attain ataskaitų priemonėmis? Ar pagreitėjo pagrindinės sistemos darbas?
Paprastumas ir patogumas	Ar vartotojai įsisavino sistemą? Ar sistema aiški, ir ar ja patogu naudotis nepatyrusiam vartotojui? Ar yra priemonių vartotojų darbui supaprastinti?
Išsamumas	Ar patenkinti visi vartotojų reikalavimai?

Įvertinus šiuos kriterijus, bus galima pasakyti ar įdiegta sistema pasiteisino, ar projektas buvo sėkmingas. Bus galima skaičiuoti per kiek laiko projektas atsipirko, kokią naudą jis davė, ir prireikus jį plėsti bei tobulinti, diegiant jį į kitas įmonės veiklos sritis (pirkimus, žaliavos supirkimą, gamybą, ilgalaikį turtą ir t.t.)

1.12. ANALIZĖS KUBŲ PROJEKTAVIMO METODŲ ANALIZĖ

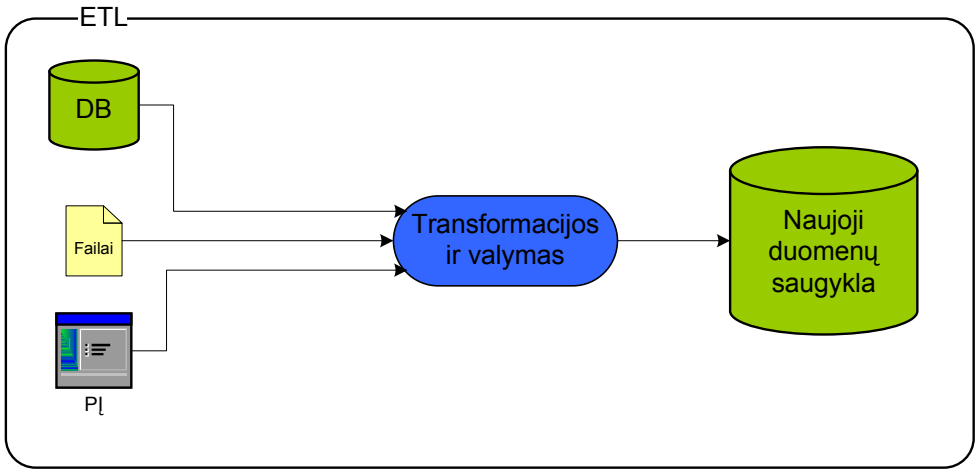
Norint sukurti optimaliai veikiančią sistemą reikia pasirinkti labiausiai reikalavimus atitinkančią projektavimo metodiką. Tuo tikslu buvo atlikta esamos literatūros analizė, siekiant išsiaiškinti esamus duomenų išgavimo, transformavimo, saugojimo, duomenų saugyklos projektavimo metodus bei jų tinkamumą kuriamai sistemai.

1.12.1. DUOMENŲ IŠGAVIMO (ETL) PROCESAI

Vienas iš svarbiausių veiksnių, užtikrinančių sėkmingą darbą su duomenimis bei optimalų sistemos resursų panaudojimą yra efektyvus duomenų išgavimas, bei jų transformavimas OLAP sistemos panaudojimui. Literatūroje tai vadinama E(C)TL – *Extract, (Cleanse), Transform and Load* [12]. Tai procesas, kurio metu reikalingi duomenys išgaunami iš šaltinio informacinės sistemos, jie išvalomi, apdorojami taip, kad optimaliai tiktų panaudojimui, ir išsaugomi naujojoje duomenų saugykloje.

Paprastai šis procesas susideda iš tokių žingsnių:

1. Identifikuojami reikalingi duomenys. Tai gali būti vienoje ar daugiau duomenų bazėje esančios lentelės, struktūrizuoti failai, ar kita.
2. Šie duomenys nuskaitomi, tuomet pašalinama nereikalinga informacija, ir jie pakeičiami pagal poreikius (atliekama agregacija, įvykdomi tarpiniai skaičiavimai ir pan.)
3. Naujai gauti duomenys išsaugomi naujojoje duomenų bazėje.



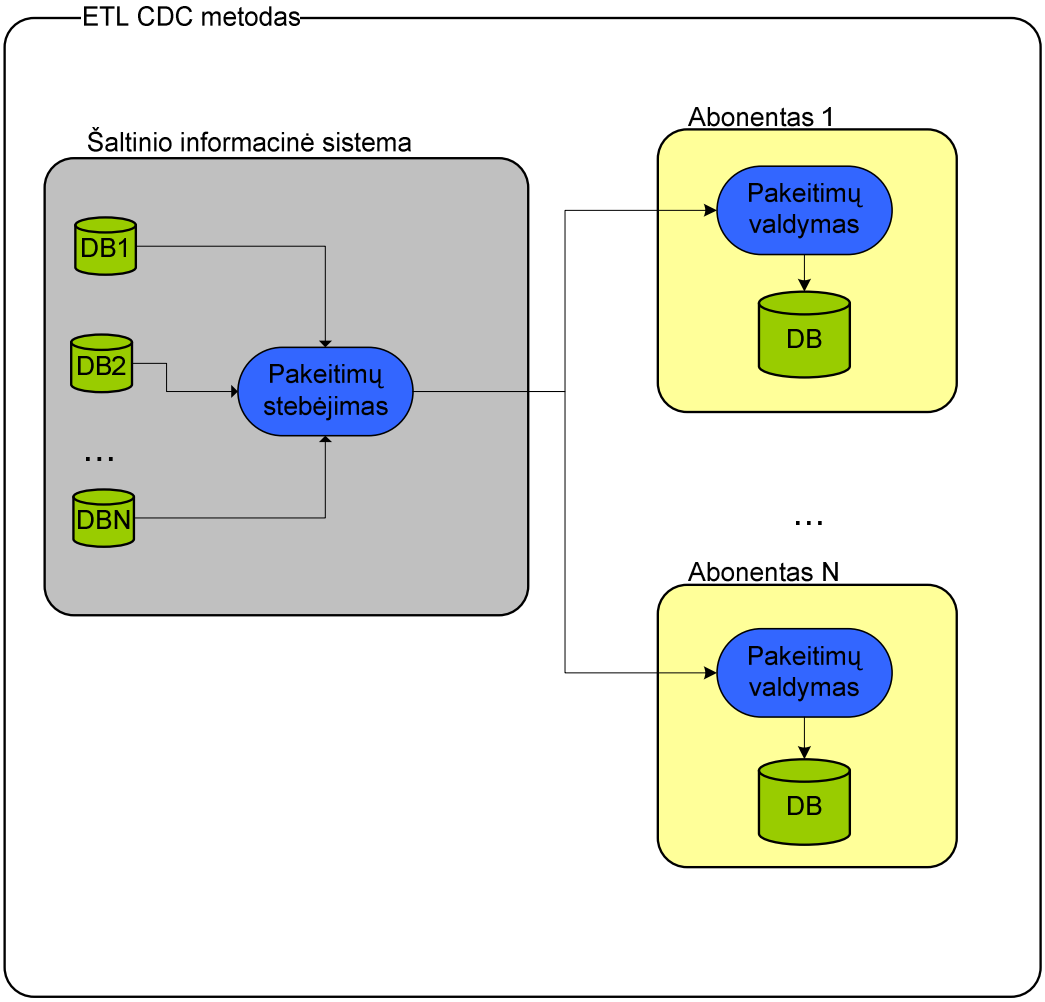
1.4 pav. ETL proceso diagrama

Pats paprasčiausias atlikimo būdas yra visų esančių duomenų nuskaitymas, apdorojimas, ir patalpinimas į naująją duomenų bazę. Tačiau tai yra daug laiko ir resursų reikalaujanti operacija, todėl ieškoma tobulesnių sprendimų. Tai gali būti tik naujų duomenų apdorojimas, visų pasikeitimų stebėjimas, ir pan. Vienas iš tokių metodų yra CDC metodas.

1.12.2. KINTANČIŲ DUOMENŲ IŠGAVIMO (CDC) METODAS

Remiantis I. Ankorion [11] CDC (angl. *Changed Data Capture*) metodas, tai ETL technologija, padedanti išspręsti vieną didžiausių ETL problemų: apdorojimo laikas didėja eksponentiškai didėjant duomenų kiekiams. Tai ypač svarbu kai yra reikalingi kuo naujesni duomenys, o jų kiekis yra labai didelis. Tokiu atveju, norint palaikyti greitą ir efektyvų duomenų atnaujinimą, reikia vis galingesnės techninės įrangos, arba tenka naudotis vis senesniais duomenimis. CDC technologija šią problemą išsprendžia identifikuodama, priimdama ir išsaugoma visus duomenų bazės pakeitimus realiu laiku.

Paprastai CDC veikia leidėjo / abonento principu. Leidėjo duomenų bazės valdymo sistema stebi duomenų bazės pasikeitimus, ir praneša apie juos abonentui. Tokiu būdu, abonentas visada turi šviežiausius duomenis.



1.5 pav. ETL CDC metodo schema

Pakeitimų stebėjimą galima realizuoti keletu skirtingų būdų:

- Stebint DBVS pakeitimų žurnalą
- Panaudoti šaltinio įrašų laiko žymes (angl. *timestamps*)

- Palyginti prieš ir po pakeitimų sukurtus DB vaizdus
- Panaudoti DBVS trigerius

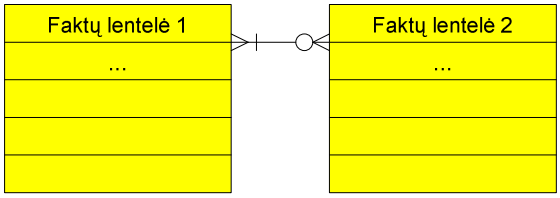
Kurį iš šių būdų panaudoti dažniausiai apsprendžia esama DBVS bei jos teikiamos galimybės. Šie būdai skiriasi ir naudojamais resursais, efektyvu, realizacijos sudėtingumu.

1.12.3. SAUGYKLŲ SCHEMOS

Nemažiau svarbu nei efektyvus reikalingų duomenų išgavimas yra ir tinkamas duomenų saugyklos parinkimas. Priklausomai nuo reikalavimų, galima pasirinkti iš daugelio pasiteisinusių duomenų saugyklų modelių.

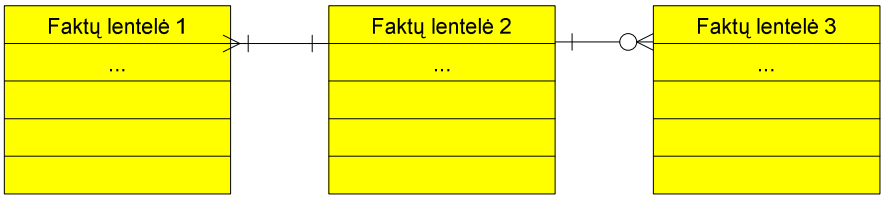
Remiantis D. Moody, M. Kortink [10], egzistuoja keletas pagrindinių duomenų saugyklos modelių. Kiekvienas iš jų turi savo privalumų ir trūkumų [8].

- **Plokščioji**. Tai paprasčiausia schema kurią galimą panaudoti neatsisakant jokios informacijos. Šiuo atveju visa informacija apjungiama į kuo mažesnį esybių skaičių, tuo minimizuojant reikalingų ryšių kiekį ir pagreitinant užklausų įvykdymą. Tačiau tokios schemos trūkumas yra tas, kad nors esybių skaičius yra minimizuojamas, tačiau jų sudėtingumas didėja, nes atsiranda perteklinė informacija.



1.6 pav. Plokščioji schema

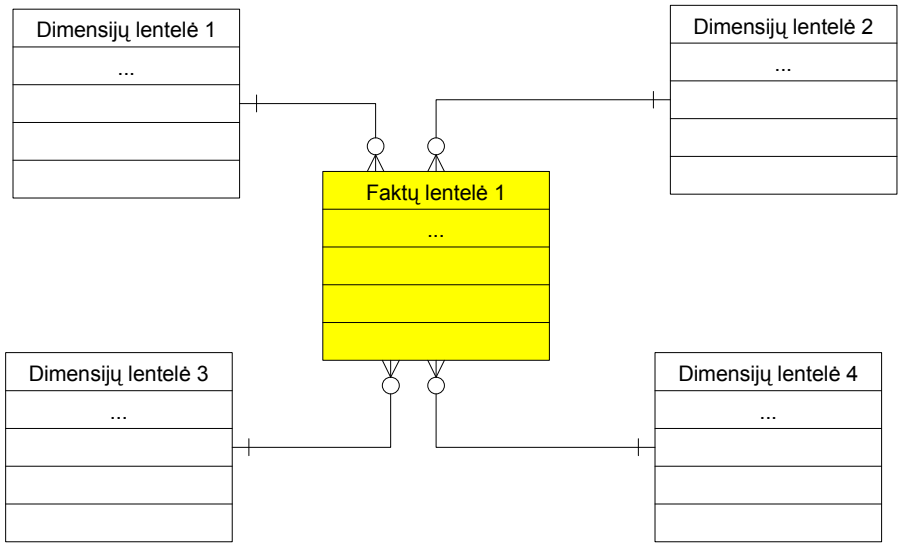
- **Terasos**. Tai schema panaši į plokščiąją, tačiau joje esybių skaičius minimizuojamas tik iki operacijos lygio – t.y. į vieną esybę įtraukiami visi vieną operaciją (transakciją) apibūdinantys atributai.



1.7 pav. Terasos tipo schema

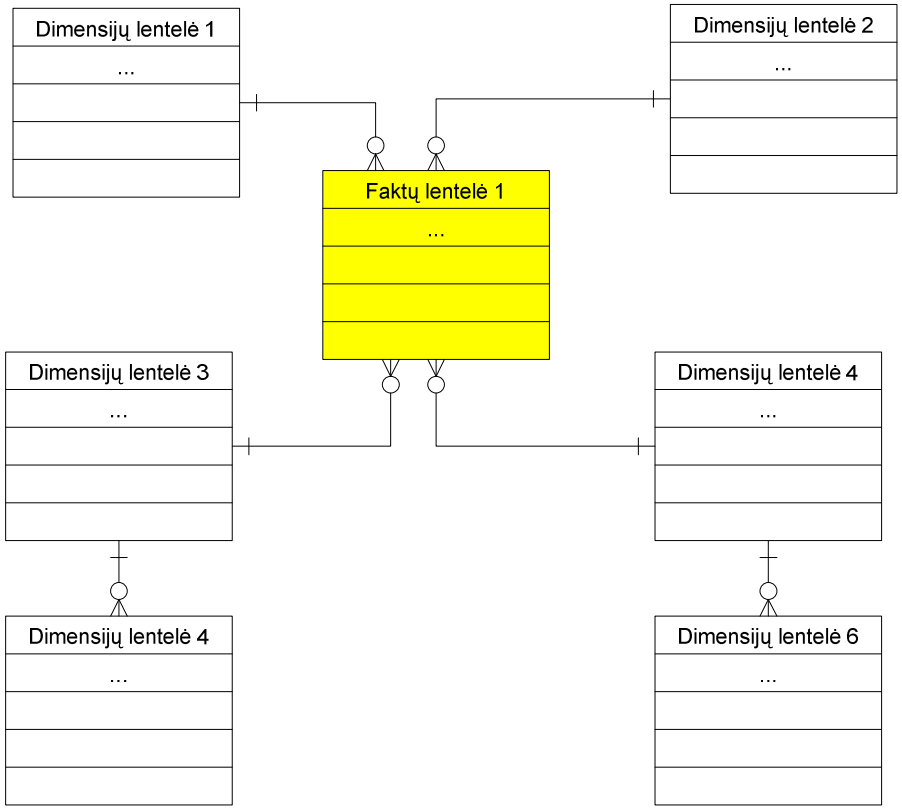
- **Žvaigždės**. Tai schema kuri dažniausiai atitinka duomenų bazės struktūrą – čia faktų lentelė padaroma iš operacijų aibės, o dimensijų lentelės sudaromos iš visų operacijoje dalyvaujančių esybių. Jeigu egzistuoja hierarchija tarp dimensijų, tai vaiko esybė paveldi visus atributus iš tėvinės esybės, taip paliekant tik vieną hierarchijos lygmenį. Ši schema

gali būti išplėsta iki **žvaigždžių spiečiaus** schemos, jeigu egzistuoja faktų lentelės (o tiksliau operacijų esybių sudarančių ją) hierarchija.



1.8 pav. Žvaigždės tipo schema

- **Snaigės**. Snaigės schema yra beveik identiška žvaigždės schemai, tik čia dimensijų esybės nėra sutraukiamos iki vieno hierarchijos lygmens, o išsaugojamos pilnai.



1.9 pav. Snaigės tipo schema

- **Žvaigždžių klasterio**. Ir žvaigždės ir snaigės schema turi trūkumų – snaigės atveju vietoj paprastos duomenų saugyklos gaunama sudėtinga, reikalaujanti daugybės ryšių, kai tuo tarpu žvaigždės atveju prarandama dimensijų hierarchijos informacija. Tokiu atveju

siūloma naudoti **žvaigždžių klasterio** schemą, kur dimensijos sutraukiamos tol kol atsiranda išsišakojimas (t.y. ta pati sutraukta dimensija tampa naudojama keletą kartų). Jam atsiradus, žemesnio hierarchinio lygmens dimensijos įtraukiamos į subdimensijų esybę.

Kiekviena iš schemų turi savų privalumų bei trūkumų, todėl svarbu pasirinkti labiausiai tinkančią atsižvelgiant į tinkamumą, realizacijos kainą bei sudėtingumą, palaikomumą bei teikiamus privalumus.

1.12.4. OPTIMIZACIJA

Kuriant duomenų saugyklą bei jos pildymo procesus svarbu numatyti jų spartų veikimą. Norint tai užtikrinti reikia remtis atliktais tyrimais bei skaičiavimais, panaudoti turimas žinias apie šių procesų optimizaciją.

Remiantis S. Goil ir A. Choudhary [7], galima išskirti tokius svarbiausius kubų optimizavimo žingsnius:

1. **Skirsnių (angl. *partitions*) naudojimas.** Suskaidant kubą į nepriklausomus skirsnius atsiranda galimybių panaudoti lygiagrečius skaičiavimus, kas pagreitiną veikimą daugiaprocesorinėse sistemose.
2. **Daliniai kubai.** Tai optimizacija, kuri gali būti panaudota tik paties OLAP serverio kubų apdorojimo algoritme (esminis optimizacijos principas yra tai, kad ne visus duomenis reikia atnaujinti norint apskaičiuoti galutinį rezultatą).
3. Svarbu minimizuoti bendravimą tarp procesų (tai padeda efektyviau išnaudoti daugiaprocesorines sistemas).

Microsoft [13] taip pat pateikia keletą OLAP kubų optimizavimo patarimų, pritaikytų specialiai Microsoft SQL Server Analysis Services programinei įrangai:

1. Efektyvus agregacijų projektavimas sutrumpina užklausų įvykdymo laiką. Agregacijos tai iš anksto suskaičiuotos tarpinės sumos kiekvienam hierarchijos lygmeniui. Agregacijoms kurti yra skirtos skirtingos priemonės – vienos iš jų numato tolydų užklausų pasiskirstymą pagal dimensijas, kitos leidžia agregacijas kurti pagal kubų naudojimo istoriją, ir pan. Reikia nepamiršti, kad didėjant dimensijų skaičiui, agregacijų kiekis didėja eksponentiškai, o tuo pačiu didėja ir saugyklos dydis, todėl svarbu rasti geriausią balansą tarp agregacijų ir atnaujinimo greičio bei duomenų saugyklos dydžio.
2. Saugyklos tipas. Analysis Services leidžia pasirinkti vieną iš trijų saugyklos tipų – *MOLAP* (angl. *multidimensional OLAP)*, *HOLAP* (angl. *hybrid OLAP)* arba *ROLAP* (angl. *relational OLAP)*. Šis pasirinkimas taip pat apsprendžia užklausos įvykdymo greitį. Apibendrinti kiekvieno iš tipų privalumus ir trūkumus galima taip:
    - MOLAP saugykloje saugoma tiek šaltinio duomenys, tiek ir suskaičiuotos agregacijos. Šis saugyklos tipas naudoja suspaudimą (angl. *compression*), todėl

sumažinama reikalinga saugojimui vieta. Kadangi naudojama daugiamatėm struktūrom saugoti optimizuota saugykla, įvairios užklausos įvykdomos daug greičiau. Duomenys dažniausiai saugojami tame pačiame serveryje, todėl jų pasiekimas ypač spartus. Šis saugyklos tipas naudoja daugiausiai disko vietos.

- HOLAP saugykla palieka šaltinio duomenis šaltinio saugykloje, o savojoje saugykloje saugo tik agregacijas. Taip naudojama mažiau disko vietos, tačiau prarandamas užklausų atlikimo greitis. Kad jas pagreitinti, dažnai kuriami indeksai šaltinio duomenų saugykloje ar papildomos agregacijos, tačiau taip prarandama sutaupyta vieta.
- ROLAP saugykla palieka duomenis šaltinio saugykloje, ir ten pat saugo agregacijas. Šis saugyklos tipas daugiausia naudojamas su ypač didelėmis dimensijomis, nes MOLAP dimensijos užkraunamos pilnai į atmintį. Taip pat naudojama realaus laiko OLAP sprendimuose.

3. Norint pagreitinti kubų atnaujinimą, svarbu nesukurti per daug agregacijų. Kadangi agregacijos taip pat turi būti atnaujintos, tai didelis jų kiekis gali ženkliai sulėtinti kubų atnaujinimą.
4. Kubų skaidymas particijomis leidžia jas atnaujinti paraleliai, atnaujinti tik tas particijas kurias reikia, kurti skirtingus agregacijų nustatymus particijoms ir pan.
5. Pašalinti perteklinius ryšius tarp faktų lentelės ir dimensijų lentelių – nurodant OLAP serveriui jog dimensijų narius galima rasti tiesiai iš dimensijų lentelių, nenaudojant sąryšio su faktų lentele, galima sutaupyti daug kubų atnaujinimo laiko.
6. Naudoti tik naujus duomenis – atnaujinant kubą (ar particijas) galima nurodyti kokius duomenis naudoti. Naudojant tik naujus duomenis, sutaupoma ypač daug atnaujinimo laiko.
7. Naudoti indeksus šaltinio duomenų bazėje – tai pagreitina užklausų vykdomų atnaujinant kubą išgavimą.
8. Svarbu minimizuoti naudojamas dimensijas, matus, lygmenis ir pan., nes kiekvienas iš jų prisideda prie agregacijų dydžio bei sudėtingumo, tuo lėtindamas sistemą.
9. Naudoti tinkamus duomenų tipus – paprastai naudojant skaitmeninius duomenų tipus raktiniams laukams užklausos apjungiančios kelias lenteles įvykdomos daug greičiau. Taip pat svarbu naudoti kuo trumpesnius duomenų tipus (pvz. 4 baitų skaičius vietoj 8 baitų).
10. Nemažiau svarbu yra ir techninės įrangos optimizavimas, tačiau tai nepatenka į šio projekto ribas.

1.13. ANALIZĖS IŠVADOS

Baigiant analizę, reiktų padaryti šias išvadas:

1. Tiriant įmonės, naudojančios Navision Attain verslo valdymo sistemą, veiklą, nustatyta problema – dabartinių duomenų analizės priemonių (Navision sistemos ataskaitų) nepakanka, jos nėra pakankamai lanksčios, naudoja per daug darbinės sistemos resursų, neišnaudojamas naktinis sistemos darbo metas. Todėl nutarta šiam tikslui ištirti OLAP priemones ir sudaryti jų taikymo metodiką įmonėms, verslo valdymui naudojančioms Navision Attain sistemą.

2. Išanalizuoti įmonėje vykstantys procesai, identifikuoti skirtingi vartotojų tipai – finansininkai, pardavimų analitikai, sandėlio darbuotojai, personalo darbuotojai. Vartotojų tipams numatyti duomenų kubai. Viso numatyta sukurti keturis duomenų kubus – pirkimų, pardavimų, atsargų bei pirkėjų skolų.

3. Numatyta galima būsimos sistemos architektūra bei diegimas – būsimai sistemai paskirtas atskiras serveris, numatyta jog bus naudojami esantys *Active Directory* tinklo vartotojai bei jais pagrįstas duomenų saugumas.

4. Išgauti funkciniai (kiekvienam duomenų kubui reikalingos dimensijos, matai, skaičiuojamieji laukai) bei nefunkciniai (saugumo, našumo, integracijos) reikalavimai, reikalavimai duomenims (už duomenų vientisumą atsako verslo valdymo sistema). Sudarytas svarbiausių rizikos faktorių sąrašas, numatytas jų valdymas.

5. Atlikus literatūros analizę, pasirinkti OLAP sistemos kūrimo sprendimai: atsisakyta CDC metodikos panaudojimo (reikalauja per daug resursų, netinka esamai sistemai), pasirinkta MOLAP tipo duomenų saugykla (siekiant greičio bei efektyvumo) ir snaigės tipo kubų schema; nuspręsta atlikti literatūros siūlomas optimizacijas.

6. Pasiūlytas klientui tinkamas problemos sprendimo būdas (bus naudojamos Microsoft SQL Server teikiamos OLAP priemonės, duomenų kubų serveris atskiriamas nuo verslo valdymo sistemos serverio) tinkamas ne tik šiai konkrečiai įmonei, tačiau ir bet kurioms kitoms besinaudojančioms Navision Attain sistema. Projektuojama sistema universali, lengvai plečiama ir pritaikoma kitiems verslo procesams.

7. Pasirengta sistemos diegimui – išanalizuota esama įmonės infrastruktūra, atliktas pirminis integravimo bandymas (išgaunant duomenis iš verslo valdymo sistemos ir juos patalpinant duomenų sandėlyje). Su klientu aptarta tolimesnė veiksmų seka bei terminai.

2. REIKALAVIMAI BŪSIMAI SISTEMAI

Analizės etapo metu išgauti vartotojų reikalavimai projektuojamai sistemai turi būti klasifikuoti bei patikslinti. Šioje dalyje surinkti reikalavimai priskiriami atskiriems kubams, susiejant juos su esama Navision Attain sistema ir patikslinant jų realizavimą.

2.1. VARTOTOJŲ POREIKIŲ IR KUBŲ SĄRYŠIAI

Analizės dalyje buvo identifikuoti būsimos sistemos vartotojų tipai ir jų tikslai. Šiuos vartotojų tikslus apibendrinsime būsimais kubais.

- **Finansininkai.** Naudojasi visais duomenų kubais:
  - **Pirkimų**
  - **Paradavimų**
  - **Skolų**
  - **Atsargų**
- **Pardavimų analitikai.** Naudojasi informacija teikiama **pardavimų** duomenų kubo.
- **Sandėlio darbuotojai**. Jiems skirtas **atsargų** kubas.
- **Personalo darbuotojai**. Naudojasi **pardavimų** kubu.

2.2. VARTOTOJŲ POREIKIAI DUOMENŲ ANALIZEI

Įvardinus kiekvienam vartotojui skirtus kubus, apibrėšime ir detalizuosime jų pateikiamą informaciją bei dimensijas.

2.2.1. PIRKIMŲ KUBAS

Šis kubas skirtas analizuoti informaciją apie žaliavos pirkimus pagal tiekėjus, jų registravimų nustatymus, vietas, grupes ir panašiai. Pardavimų procese ši informacija yra svarbi norint ištirti pelningumus, statistikas ir kitus aktualius duomenis.

Būsimo **pirkimų** kubo struktūra:

2.1 lentelė. Pirkimų kubo struktūra

Dimensijos	
Data	Operacijos data
Metai	Operacijos atlikimo metai
Ketvirtis	Operacijos atlikimo ketvirtis
Mėnuo	Operacijos atlikimo mėnuo
Diena	Operacijos atlikimo diena
Dokumento Nr.	Operacijos dokumento numeris
Prekės grupė	Prekės klasifikatorius

Prekės pogrupis	Prekės klasifikatorius
Prekė	Prekės klasifikatorius
Partijos numeris	Prekei suteiktas partijos numeris
Tiekėjo reg. gr.	Tiekėjo registravimo grupė (Navision sistemos nustatymas)
Tiekėjas	Tiekėjas, iš kurio pirkta prekė
Vieta	Sandėlys, į kurį padėta prekė
Padalinys	Administracinis vienetas
Atsargų reg. gr.	Atsargų registravimo grupė (Navision sistemos nustatymas)
Mat. Vnt.	Bazinis prekės matavimo vienetas
Dėžė	Dėžė į kurią padėta prekė
Geografinė vieta	Administracinis vienetas
Matai	
Pirkimo kiekis	Įsigytas prekės kiekis
Pirkimo suma	Įsigytų prekių savikaina
Skaičiuojamieji laukai	
Pirkimo kaina	Vieneto kaina
Likutis pradžiai kiekis	Iki nurodyto periodo įsigytų prekių kiekis
Likutis pabaigai kiekis	Nurodyto periodo pabaigai įsigytas prekių kiekis viso
Likutis pradžiai suma	Iki nurodyto periodo įsigytų prekių savikaina
Likutis pabaigai suma	Nurodyto periodo pabaigai įsigytų prekių savikaina viso

2.2.2. PARDAVIMŲ KUBAS

Šis kubas skirtas analizuoti informaciją apie produkcijos pardavimus pagal pirkėjus, mokėtojus, jų sistemas, prekių klasifikaciją, administracinius vienetus, ir panašiai.

Būsimo **pardavimų** kubo struktūra:

2.2 lentelė. Pardavimų kubo struktūra

Dimensijos	
Data	Operacijos data
Metai	Operacijos atlikimo metai
Ketvirtis	Operacijos atlikimo ketvirtis
Mėnuo	Operacijos atlikimo mėnuo
Diena	Operacijos atlikimo diena
Išorinis Dokumento Nr.	Pirkėjui pateiktas dokumento Nr.
Dokumento Nr.	Operacijos dokumento numeris
Savaitė	Operacijos atlikimo savaitė
Geografinė vieta	Administracinis vienetas
Funkcinis padalinys	Administracinis vienetas

Rajonas	Pirkėjo rajonas
Vieta	Sandėlys iš kurio parduota
Stambi prekės grupė	Prekės klasifikatorius
Prekės statistinė grupė	Prekės klasifikatorius
Prekės reg. gr.	Prekės registravimo grupė (Navision Attain sistemos nustatymas)
Riebumas	Prekės riebumo grupė
Gamintojas	Prekės gamintojas
Brand	Prekybinis ženklas
Pirkėjas	Pirkėjas, kuriam parduota prekė
Mokėtojas	Mokėtojas, kuriam pateikta SF
Sistema	Klientų klasifikatorius
Sistemos klasė	Klientų klasifikatorius
Sistemos lygis	Klientų klasifikatorius
Sistemos 4 lygis	Klientų klasifikatorius
Pirkėjo šalis	Pirkėjo šalis
Pirkėjo reg. gr.	Pirkėjo registravimo grupė (Navision Attain sistemos nustatymas)
Brokas	Ar prekė buvo brokuota
Grąžinimo priežastis	Dėl kokios priežasties prekė buvo gražinta
Fasuotė	Prekės fasuotė
Pakuotė	Prekės pakuotė
Prekė	Prekės kodas ir pavadinimas
Vadybininkas	Vadybininkas atsakingas už pardavimą
Regionas	Pirkėjo regionas
Gavėjo tipas	Gavėjo tipas (klasifikatorius)
Matai	
Kiekis	Parduotas prekės kiekis
Suma su nuolaida	Parduotų prekių kaina
Nuolaida	Suteikta nuolaida
Svoris bruto	Parduotų prekių svoris bruto
Svoris neto	Parduotų prekių svoris neto
Bazinė suma	Bazinė parduotų prekių kaina
PVM suma	PVM suma
Suma su PVM	Parduotų prekių kaina su PVM
Skaičiuojamieji laukai	
Suma	Parduotų prekių kaina be nuolaidos
Pokytis kiekis	Parduoto kiekio pokytis lyginant su praėjusiu periodu
Pokytis kiekis %	Parduoto kiekio pokytis lyginant su praėjusiu periodu, procentaliai
Pokytis svoris neto	Parduoto prekių svorio pokytis lyginant su praėjusiu periodu
Pokytis svoris neto %	Parduoto prekių svorio pokytis lyginant su praėjusiu periodu, procentaliai

2.2.3. ATSARGŲ KUBAS

Šis kubas skirtas analizuoti informaciją apie atsargų apyvartą tarp sandėlių, administracinių vienetų.

Būsimo **atsargų** kubo struktūra:

2.3 lentelė. Atsargų kubo struktūra

Dimensijos	
Data	Operacijos data
Metai	Operacijos atlikimo metai
Ketvirtis	Operacijos atlikimo ketvirtis
Mėnuo	Operacijos atlikimo mėnuo
Diena	Operacijos atlikimo diena
Dokumento Nr.	Operacijos dokumento numeris
Prekė	Prekės kodas ir pavadinimas
Atsargų reg. gr.	Prekės atsargų registravimo grupė (Navision Attain sistemos nustatymas)
Prekės grupė	Prekės klasifikatorius
Prekės pogrupis	Prekės klasifikatorius
Mat. Vnt.	Prekės bazinis matavimo vienetas
BPRG	Bendra prekės registravimo grupė (Navision Attain sistemos nustatymas)
Pakuotė	Prekės pakuotė
Fasuotė	Prekės fasuotė
Statistinė grupė	Prekės klasifikatorius
Grupė gamybai	Prekės klasifikatorius
Riebumas	Prekės riebumo grupė
Brand	Prekybinis ženklas
Vieta	Sandėlys
Dėžė	Dėžė
Geografinė vieta	Administracinis vienetas
Funkcinis padalinys	Administracinis vienetas
Perduota iš vietos	Perdavimo atveju rodo iš kurios vietos perduota
Perduota į vietą	Perdavimo atveju rodo į kurią vietą perduota
BVRG	Bendra verslo registravimo grupė (Navision Attain sistemos nustatymas)
Įrašo tipas	Pirkimas/Pardavimas/Perdavimas/Teig. koregavimas/Neig. Koregavimas
Gamintojas	Prekės gamintojas
Matai	
Pirkimo kiekis	Pirkimo atveju, rodomas pirktų prekių kiekis
Pirkimo suma	Pirkimo atveju, rodomas pirktų prekių savikaina
Pardavimo kiekis	Pardavimo atveju, rodomas parduotų prekių kiekis
Pardavimo suma	Pardavimo atveju, parduotų prekių savikaina
Teig. kor. kiekis	Teigiamo koregavimo atveju, prekių kiekis

Teig. kor. suma	Teigiamo koregavimo atveju, prekių savikaina
Gautas kiekis	Įeinančio perdavimo atveju, prekių kiekis
Gauta suma	Įeinančio perdavimo atveju, prekių savikaina
Perduotas kiekis	Išeinančio perdavimo atveju, prekių kiekis
Perduota suma	Išeinančio perdavimo atveju, prekių savikaina
Neig. kor. kiekis	Neigiamo koregavimo atveju, prekių kiekis
Neig. kor. suma	Neigiamo koregavimo atveju, prekių savikaina
Kiekis	Kiekis
Suma	Savikaina
Svoris neto	Prekių svoris neto
Svoris bruto	Prekių svoris bruto
Skaičiuojamieji laukai	
Likutis pradziai kiekis	Prekių likutis periodo pradžiai
Likutis pradziai suma	Prekių savikainos likutis periodo pradžiai
Likutis pabaigai kiekis	Prekių likutis periodo pabaigai
Likutis pabaigai suma	Prekių savikainos likutis periodo pabaigai
Viso gautas kiekis	Iš viso teigiamas prekių pokytis
Viso gauta suma	Iš viso teigiamas prekių savikainos pokytis
Viso sunaudotas kiekis	Iš viso neigiamas prekių pokytis
Viso sunaudotas kiekis	Iš viso neigiamas prekių savikainos pokytis
Pirkimo kaina	Prekių įsigijimo vieneto kaina

2.2.4. SKOLŲ KUBAS

Kubas, skirtas analizuoti pirkėjų skolas už pardavimus, bei tarą.

Būsimo **skolų** kubo struktūra:

2.4 lentelė. Skolų kubo struktūra

Dimensijos	
Data	Operacijos data
Metai	Operacijos atlikimo metai
Ketvirtis	Operacijos atlikimo ketvirtis
Mėnuo	Operacijos atlikimo mėnuo
Diena	Operacijos atlikimo diena
Dokumento Nr.	Operacijos dokumento numeris
Išorinis dokumento Nr.	Klientui pateiktas dokumento numeris
Mokėtojas	Klientas, kuriam išrašyta SF
Mokėtojo vadybininkas	Vadybininkas, atsakingas už mokėtoją
Gavėjas	Klientas, gaunantis prekes

Gavėjo agentas	Vadybininkas, atsakingas už gavėją
Gavėjo tipas	Klientų klasifikatorius
Rajonas	Gavėjo regionas
Geografinė vieta	Administracinis vienetas
Sistema	Klientų klasifikatorius
Pirkėjo reg. gr.	Pirkėjo registravimo grupė (Navision Attain sistemos nustatymas)
Funkcinis padalinys	Administracinis vienetas
Beviltiškas mokėtojas	Taip/Ne
Mokėjimo būdo kodas	SF apmokėjimo būdo kodas
Apmokama tara	Taip/Ne
Tara	Taip/Ne
Taros pavadinimas	Taros kodas ir pavadinimas
Pradelstų dienų skaičius	Dienų skaičius, kiek klientas vėluoja apmokėti SF
Matai	
Mokėjimai	Visų kliento atliktų mokėjimų suma
Pardavimai	Klientui išrašytų SF suma
Grąžinimai	Iš kliento gautų grąžinimo pažymų suma
Kita	Kitų operacijų suma
Parduota taros	Klientui atiduotos taros savikaina
Parduota taros kiekis	Klientui atiduotos taros kiekis
Grąžinta taros	Iš kliento gautos taros savikaina
Grąžinta taros kiekis	Iš kliento gautos taros kiekis
Likusi suma	Likusi neapmokėta SF suma
Skaičiuojamieji laukai	
Likutis pradziai kiekis	Taros kiekio likutis periodo pradžiai
Likutis pabaigai kiekis	Taros kiekio likutis periodo pabaigai
Grynasis pokytis kiekis	Taros kiekio pokytis per periodą
Bendras grynasis pokytis	Viso kliento skolos likučio pokytis per periodą
Bendras likutis pradžiai	Viso kliento skolos likutis periodo pradžiai
Bendras likutis pabaigai	Viso kliento skolos likutis periodo pabaigai
Produkcijos grynasis pokytis	Kliento skolos už produkciją pokytis per periodą
Produkcijos likutis pradžiai	Kliento skolos už produkciją likutis periodo pradžiai
Produkcijos likutis pabaigai	Kliento skolos už produkciją likutis periodo pabaigai
Taros grynasis pokytis	Kliento skolos už tarą pokytis per periodą
Taros likutis pradžiai	Kliento skolos už tarą likutis periodo pradžiai
Taros likutis pabaigai	Kliento skolos už tarą likutis periodo pabaigai

2.3. DETALI REIKALAVIMŲ SPECIFIKACIJA

Pagal vartotojų poreikius specifikavus būsimus kubus, reikia nustatyti iš kur kiekviena dimensija ir matas įgaus reikšmes. Pagal tai įgyvendinimo metu bus kuriami rodiniai bei duomenų perkėlimo servisai, iš darbinės duomenų bazės paimantys bei perkeliantys reikalingus duomenis į OLAP duomenų saugyklą.

2.3.1. PIRKIMŲ KUBAS

Šiam kubui kaip faktų lentelė bus naudojama Navision Attain sistemos *prekės knygos įrašų* bei *vertės įrašų* lentelės.

Prekės knygos įrašų lentelėje saugomos operacijos atliekamos su atsargomis (prekėmis).

Vertės įrašų lentelėje saugoma informacija apie *prekės knygos įrašų* savikainas.

Būsimo **pirkimų** kubo sąryšiai:

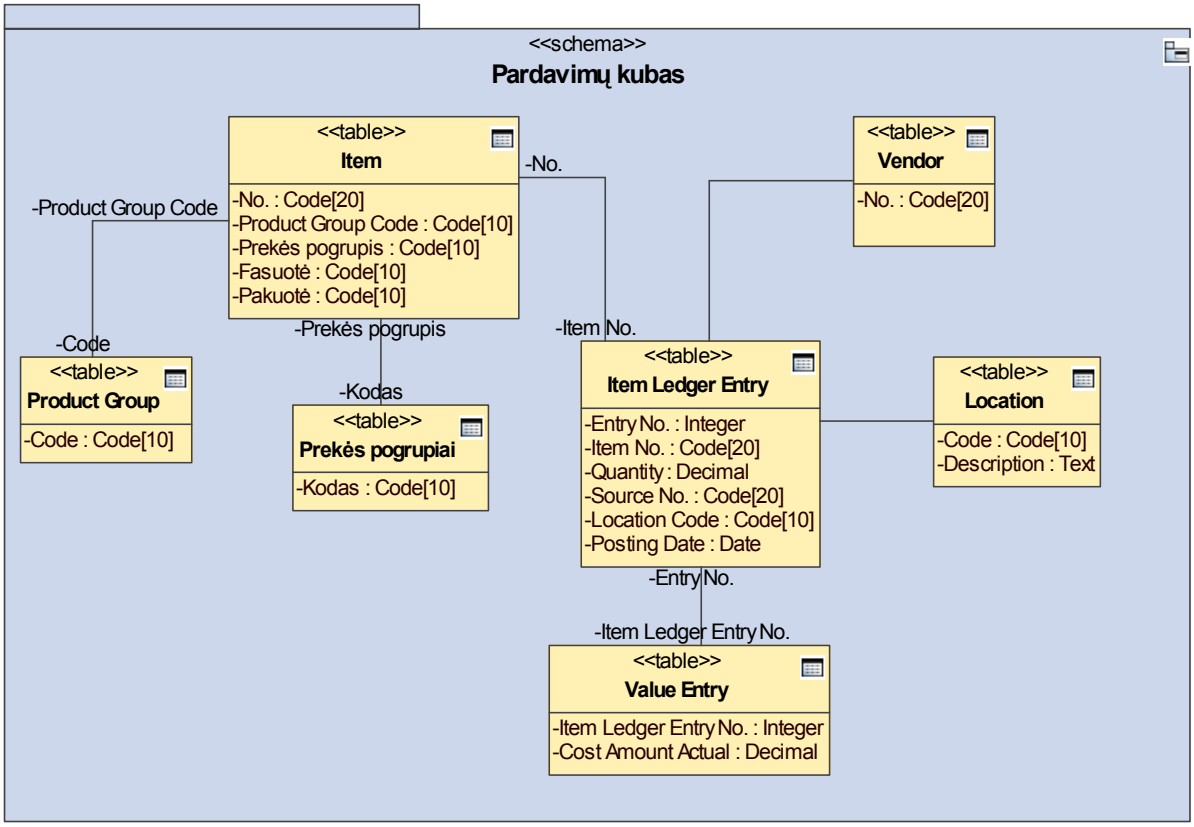
2.5 lentelė. Pirkimų kubo sąryšiai su Navision Attain sistemos lentelėmis

Dimensijos	**Lentelė**	**Laukas**
Data	CRONUS$Item Ledger Entry	Posting Date
Metai	CRONUS$Item Ledger Entry	Posting Date
Ketvirtis	CRONUS$Item Ledger Entry	Posting Date
Mėnuo	CRONUS$Item Ledger Entry	Posting Date
Diena	CRONUS$Item Ledger Entry	Posting Date
Dokumento Nr.	CRONUS$Item Ledger Entry	Document No.
Prekės grupė	CRONUS$Item CRONUS$Product Group	Product Group Code Code
Prekės pogrupis	CRONUS$Item CRONUS$Prekių pogrupiai	Prekės pogrupis Kodas
Prekė	CRONUS$Item Ledger Entry CRONUS$Item	Item No. No.
Partijos numeris	CRONUS$Item Ledger Entry	Lot No.
Tiekėjo reg. gr.	CRONUS$Vendor	Vendor Posting Group
Tiekėjas	CRONUS$Item Ledger Entry CRONUS$Vendor	Source No. No.
Vieta	CRONUS$Item Ledger Entry CRONUS$Location	Location Code Code
Padalinys	CRONUS$Item Ledger Entry	Global Dimension 2 Code
Atsargų reg. gr.	CRONUS$Item	Inventory Posting Group
Mat. Vnt.	CRONUS$Item Ledger Entry	Base Unit of Measure
Dėžė	CRONUS$Item Ledger Entry	Bin Code
Geografinė vieta	CRONUS$Item Ledger Entry	Global Dimension 1 Code
Matai	**Lentelė**	**Laukas**

Pirkimo kiekis	CRONUS$Item Ledger Entry	Quantity
Pirkimo suma	CRONUS$Value Entry	Cost Amount (Actual)
Skaičiuojamieji laukai	**Formulė**	
Pirkimo kaina	Pirkimo suma / Pirkimo kiekis	
Likutis pradžiai kiekis	Kiekio suma iki šio periodo	
Likutis pabaigai kiekis	Kiekio suma iki šio periodo imtinai	
Likutis pradžiai suma	Savikainos suma iki šio periodo	
Likutis pabaigai suma	Kiekio suma iki šio periodo imtinai	

Kai kurie laukai nurodomi kaip esantys iš dviejų skirtingų lentelių – tai rodo jog laukas siejasi su kita lentele. Tokiu atveju dažniausiai būtina nurodyti ir lauko kodą, ir susietoje lentelėje esantį pavadinimą (pvz. Prekės Nr. ir Prekės pavadinimas).

Šiame kube naudojamas lenteles ir jų sąryšius galima apibendrinti tokia konceptualia duomenų bazės schema (schemoje rodomi tik sąryšiams reikalingi laukai):



2.1 pav. Pardavimų kube naudojamų lentelių sąryšiai

Atsižvelgiant į naudojamas lenteles, bus kuriami tokie rodiniai:

- *Prekė* – skirtas pateikti informaciją apie prekę.
- *Tiekėjas* – skirtas pateikti informaciją apie tiekėjus.
- *Vieta* – skirtas pateikti informaciją apie sandėlį.

- *Vertės įrašai* – pateikia vertės įrašus, sugrupuotus pagal prekės knygos įrašo numerį.
- *Pirkimai – pagrindinis* – formuoja faktų lentelę.

2.3.2. PARDAVIMŲ KUBAS

Šiam kubui kaip faktų lentelė bus naudojama Navision Attain sistemos *prekės knygos įrašų* bei *vertės įrašų* lentelės.

Prekės knygos įrašų lentelėje saugomos operacijos atliekamos su atsargomis (prekėmis).

Vertės įrašų lentelėje saugoma informacija apie *prekės knygos įrašų* savikainas.

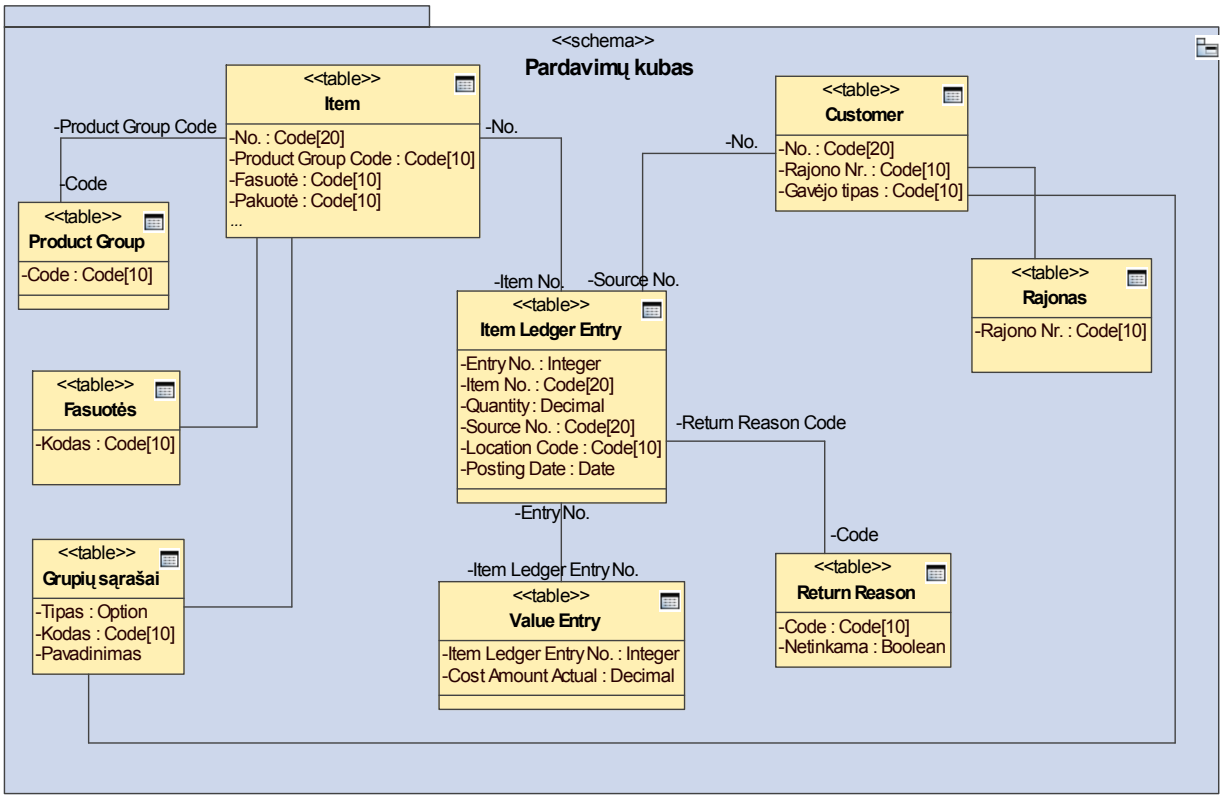
Būsimo **pardavimų** kubo sąryšiai:

2.6 lentelė. Pardavimų kubo sąryšiai su Navision Attain sistemos lentelėmis

Dimensijos	Lentelė	Laukas
Data	CRONUS$Item Ledger Entry	Posting Date
Metai	CRONUS$Item Ledger Entry	Posting Date
Ketvirtis	CRONUS$Item Ledger Entry	Posting Date
Mėnuo	CRONUS$Item Ledger Entry	Posting Date
Diena	CRONUS$Item Ledger Entry	Posting Date
Išorinis Dokumento Nr.	CRONUS$Item Ledger Entry	External Document No.
Dokumento Nr.	CRONUS$Item Ledger Entry	Document No.
Savaitė	CRONUS$Item Ledger Entry	Posting Date
Geografinė vieta	CRONUS$Item Ledger Entry	Global Dimension 1 Code
Funkcinis padalinys	CRONUS$Item Ledger Entry	Global Dimension 2 Code
Rajonas	CRONUS$Customer	Rajono Nr.
	CRONUS$Rajonas	Rajono Nr.
Vieta	CRONUS$Item Ledger Entry	Location Code
Stambi prekės grupė	CRONUS$Item	Product Group Code
	CRONUS$Product Group	Code
Prekės statistinė grupė	CRONUS$Item	Statistinė grupė
Prekės reg. gr.	CRONUS$Item	Inventory Posting Group
Riebumas	CRONUS$Item	Riebumo grupė
Gamintojas	CRONUS$Item	Manufacturer Code
Brand	CRONUS$Item	Brand
Pirkėjas	CRONUS$Item Ledger Entry	Source No.
	CRONUS$Customer	No.
Mokėtojas	CRONUS$Customer	Bill-to Customer No.
	CRONUS$Customer	No.
Sistema	CRONUS$Customer	Sistemos kodas
Sistemos klasė	CRONUS$Customer	Sistemos klasės kodas

Sistemos lygis	CRONUS$Customer	Sistemos lygio kodas
Sistemos 4 lygis	CRONUS$Customer	Sistemos 4-as lygis
Pirkėjo šalis	CRONUS$Customer	Country Code
Pirkėjo reg. gr.	CRONUS$Customer	Customer Posting Group
Brokas	CRONUS$Return Reason	Netinkama
Grąžinimo priežastis	CRONUS$Item Ledger Entry	Return Reason Code
	CRONUS$Return Reason	Code
Fasuotė	CRONUS$Item	Fasuotė
	CRONUS$Fasuotės	Kodas
Pakuotė	CRONUS$Item	Pakuotė
	CRONUS$Grupių sąrašai	Kodas, kur Tipas = Pakuotė
Prekė	CRONUS$Item Ledger Entry	Item No.
Vadybininkas	CRONUS$Item Ledger Entry	Vadybininko kodas
Regionas	CRONUS$Customer	Global Dimension 2 Code
Gavėjo tipas	CRONUS$Customer	Gavėjo tipas
	CRONUS$Grupių sąrašai	Kodas, kur Tipas = Gavėjo tipas
Matai	**Lentelė**	**Laukas**
Kiekis	CRONUS$Item Ledger Entry	Kiekis * -1
Suma su nuolaida	CRONUS$Value Entry	Cost Amount (Actual)
Nuolaida	CRONUS$Value Entry	Discount Amount
Svoris bruto	CRONUS$Item Ledger Entry	Quantity * -1 * Gross Weight
	CRONUS$Item	
Svoris neto	CRONUS$Item Ledger Entry	Quantity * -1 * Net Weight
	CRONUS$Item	
Bazinė suma	CRONUS$Item Ledger Entry	Bazinė kaina
Skaičiuojamieji laukai	**Formulė**	
Suma	Suma su nuolaida – Nuolaida	
PVM Suma	Suma * 0,18	
Suma su PVM	Suma * 1,18	
Pokytis kiekis	Kiekis – [ankstesnis kiekis]	
Pokytis kiekis %	Kiekis – [ankstesnis kiekis] / [ankstesnis kiekis]	
Pokytis svoris neto	Svoris neto – [ankstesnis svoris neto]	
Pokytis svoris neto %	Svoris neto – [ankstesnis svoris neto] / [ankstesnis svoris neto]	

Šiame kube naudojamas lenteles ir jų sąryšius galima apibendrinti tokia konceptualia duomenų bazės schema (schemoje rodomi tik sąryšiams reikalingi laukai):



2.2 pav. Pirkimų kube naudojamų lentelių sąryšiai

Atsižvelgiant į naudojamas lenteles, bus kuriami tokie rodiniai:

- *Prekė* – panaudojamas tas pats kaip ir pirkimų kube.
- *Pirkėjas* – skirtas pateikti informaciją apie pirkėjus.
- *Grąžinimo priežastis* – informacija apie grąžinimo priežastį.
- *Vertės įrašai* – naudojamas tas pats kaip ir pirkimų kube.
- *Pardavimai – pagrindinis* – formuoja faktų lentelę.

2.3.3. ATSARGŲ KUBAS

Šiam kubui kaip faktų lentelė bus naudojama Navision Attain sistemos *prekės knygos įrašų* bei *vertės įrašų* lentelės.

Prekės knygos įrašų lentelėje saugomos operacijos atliekamos su atsargomis (prekėmis).

Vertės įrašų lentelėje saugoma informacija apie *prekės knygos įrašų* savikainas.

Būsimo **atsargų** kubo sąryšiai:

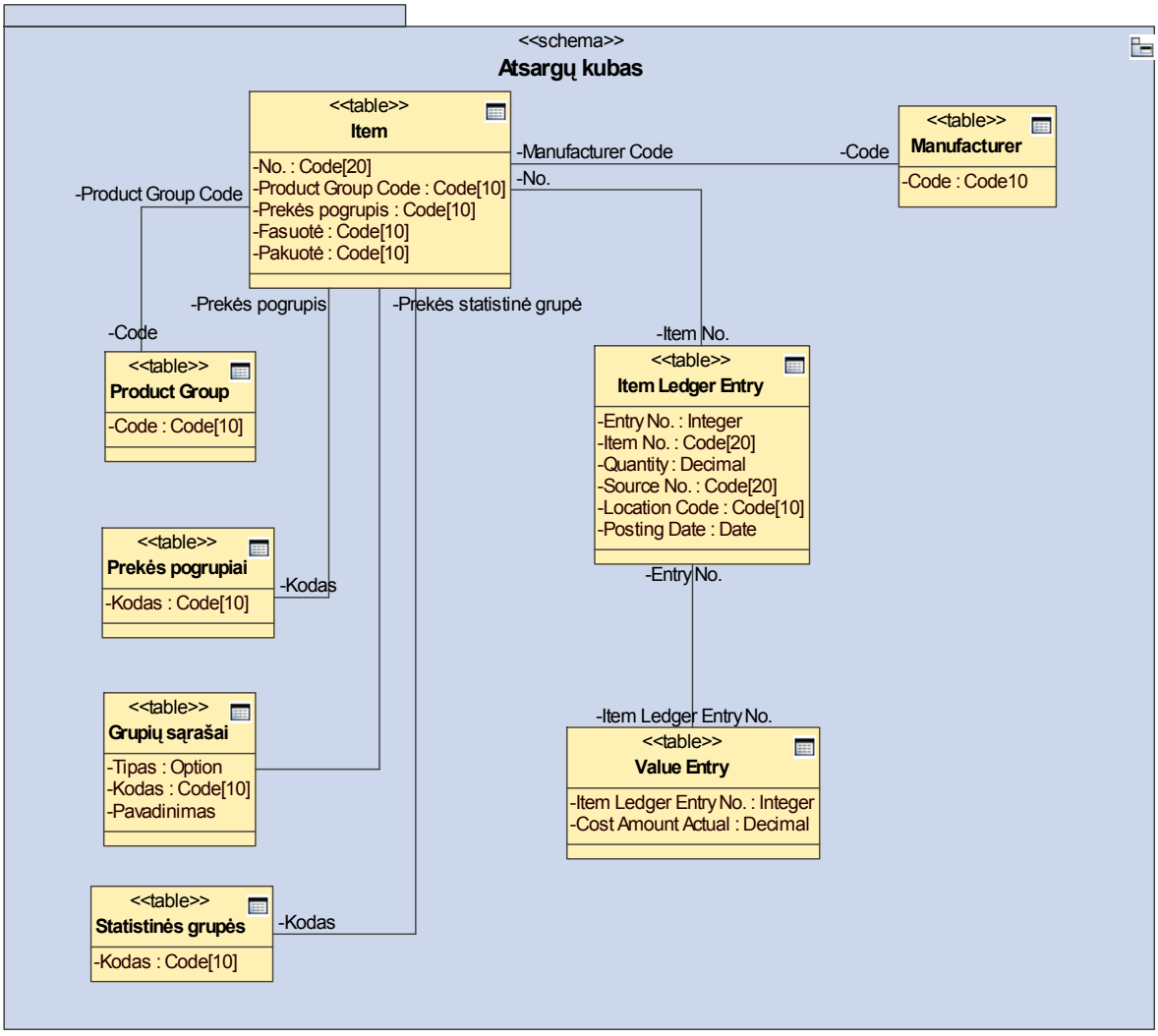
2.7 lentelė. Atsargų kubo sąryšiai su Navision Attain sistemos lentelėmis

Dimensijos	Lentelė	Laukas
Data	CRONUS$Item Ledger Entry	Posting Date
Metai	CRONUS$Item Ledger Entry	Posting Date
Ketvirtis	CRONUS$Item Ledger Entry	Posting Date

Mėnuo	CRONUS$Item Ledger Entry	Posting Date
Diena	CRONUS$Item Ledger Entry	Posting Date
Dokumento Nr.	CRONUS$Item Ledger Entry	Document No.
Prekė	CRONUS$Item Ledger Entry	Item No.
	CRONUS$Item	No.
Atsargų reg. gr.	CRONUS$Item	Inventory Posting Group
Prekės grupė	CRONUS$Item	Product Group Code
	CRONUS$Product Group	Code
Prekės pogrupis	CRONUS$Item	Prekės pogrupis
	CRONUS$Prekių pogrupiai	Kodas
Mat. Vnt.	CRONUS$Item	Base Unit of Measure
BPRG	CRONUS$Item	Gen. Prod. Posting Group
Pakuotė	CRONUS$Item	Pakuotė
	CRONUS$Grupių sąrašai	Kodas, kur Tipas = Pakuotė
Fasuotė	CRONUS$Item	Fasuotė
	CRONUS$Fasuotės	Kodas
Statistinė grupė	CRONUS$Item	Statistinė grupė
	CRONUS$Statistinės grupės	Kodas
Grupė gamybai	CRONUS$Item	Grupė gamybai
	CRONUS$Grupių sąrašai	Kodas, kur Tipas = Grupė gamybai
Riebumas	CRONUS$Item	Riebumo grupė
	CRONUS$Grupių sąrašai	Kodas, kur Tipas = Riebumas
Brand	CRONUS$Item	Brand
	CRONUS$Grupių sąrašai	Kodas, kur Tipas = Brand
Vieta	CRONUS$Item Ledger Entry	Location Code
Dėžė	CRONUS$Item Ledger Entry	Bin Code
Geografinė vieta	CRONUS$Item Ledger Entry	Global Dimension 1 Code
Funkcinis padalinys	CRONUS$Item Ledger Entry	Global Dimension 2 Code
Perduota iš vietos	CRONUS$Item Ledger Entry	Perduota iš vietos
Perduota į vietą	CRONUS$Item Ledger Entry	Perduota į vietą
BVRG	CRONUS$Item Ledger Entry	Bendr. verslo reg. grupė
Įrašo tipas	CRONUS$Item Ledger Entry	Entry Type
Gamintojas	CRONUS$Item	Manufacturer Code
	CRONUS$Manufacturer	Code
Matai	**Lentelė**	**Laukas**
Pirkimo kiekis	CRONUS$Item Ledger Entry	Quantity, jei Entry Type = Pirkimas
Pirkimo suma	CRONUS$Value Entry	Cost Amount (Actual), jei Entry Type = Pirkimas
Pardavimo kiekis	CRONUS$Item Ledger Entry	Quantity, jei Entry Type = Pardavimas
Pardavimo suma	CRONUS$Value Entry	Cost Amount (Actual), jei Entry Type

		= Pardavimas
Teig. kor. kiekis	CRONUS$Item Ledger Entry	Quantity, jei Entry Type = Pardavimas
Teig. kor. suma	CRONUS$Value Entry	Cost Amount (Actual), jei Entry Type = Pardavimas
Gautas kiekis	CRONUS$Item Ledger Entry	Quantity, jei Entry Type = Perdavimas, ir Quantity > 0, ir sandėlys ne tranzitinis
Gauta suma	CRONUS$Value Entry	Cost Amount (Actual), jei Entry Type = Perdavimas, ir Quantity > 0, ir sandėlys ne tranzitinis
Perduotas kiekis	CRONUS$Item Ledger Entry	Quantity, jei Entry Type = Perdavimas, ir Quantity < 0, ir sandėlys ne tranzitinis
Perduota suma	CRONUS$Value Entry	Cost Amount (Actual), jei Entry Type = Perdavimas, ir Quantity < 0, ir sandėlys ne tranzitinis
Neig. kor. kiekis	CRONUS$Item Ledger Entry	Quantity, jei Entry Type = Neig. koregavimas
Neig. kor. suma	CRONUS$Value Entry	Cost Amount (Actual), jei Entry Type = Neig. koregavimas
Kiekis	CRONUS$Item Ledger Entry	Quantity
Suma	CRONUS$Value Entry	Cost Amount (Actual)
Svoris neto	CRONUS$Item Ledger Entry CRONUS$Item	Quantity * Net Weight
Svoris bruto	CRONUS$Item Ledger Entry CRONUS$Item	Quantity * Gross Weight
Skaičiuojamieji laukai	**Formulė**	
Likutis pradziai kiekis	Kiekio suma iki šio periodo	
Likutis pradziai suma	Savikainos suma iki šio periodo	
Likutis pabaigai kiekis	Kiekio suma iki šio periodo imtinai	
Likutis pabaigai suma	Savikainos suma iki šio periodo imtinai	
Viso gautas kiekis	Pirkimo kiekis + Gautas kiekis + Teig. kor. kiekis	
Viso gauta suma	Pirkimo suma + Gauta suma + Teig. kor. suma	
Viso sunaudotas kiekis	Pardavimo kiekis + Perduotas kiekis + Neig. kor. kiekis	
Viso sunaudotas kiekis	Pardavimo suma + Perduota suma + Neig. kor. suma	
Pirkimo kaina	Pirkimo kaina / Pirkimo kiekis	

Šiame kube naudojamas lenteles ir jų sąryšius galima apibendrinti tokia konceptualia duomenų bazės schema (schemoje rodomi tik sąryšiams reikalingi laukai):



2.3 pav. Atsargų kube naudojamų lentelių sąryšiai

Atsižvelgiant į naudojamas lenteles, bus kuriami tokie rodiniai:

- *Prekė* – panaudojamas tas pats kaip ir pirkimų kube.
- *Vertės įrašai* – naudojamas tas pats kaip ir pirkimų kube.
- *Atsargos – pagrindinis* – formuoja faktų lentelę.

2.3.4. SKOLŲ KUBAS

Šiam kubui kaip faktų lentelės bus naudojamos Navision Attain sistemos *pirkėjų knygos įrašų*, detalių pirkėjų knygos įrašų, *prekės knygos įrašų* bei *vertės įrašų* lentelės.

Pirkėjų knygos įrašų lentelėje saugomos pirkėjų atliktos operacijos – pardavimai, grąžinimai, mokėjimai, ir pan.

Detalių pirkėjų knygos įrašų lentelėje saugoma informacija apie pirkėjų operacijų gretinimus (SF ir mokėjimų, ir pan.)

Prekės knygos įrašų lentelėje saugomos operacijos atliekamos su atsargomis (prekėmis).

Vertės įrašų lentelėje saugoma informacija apie *prekės knygos įrašų* savikainas.

Būsimo **skolų** kubo sąryšiai:

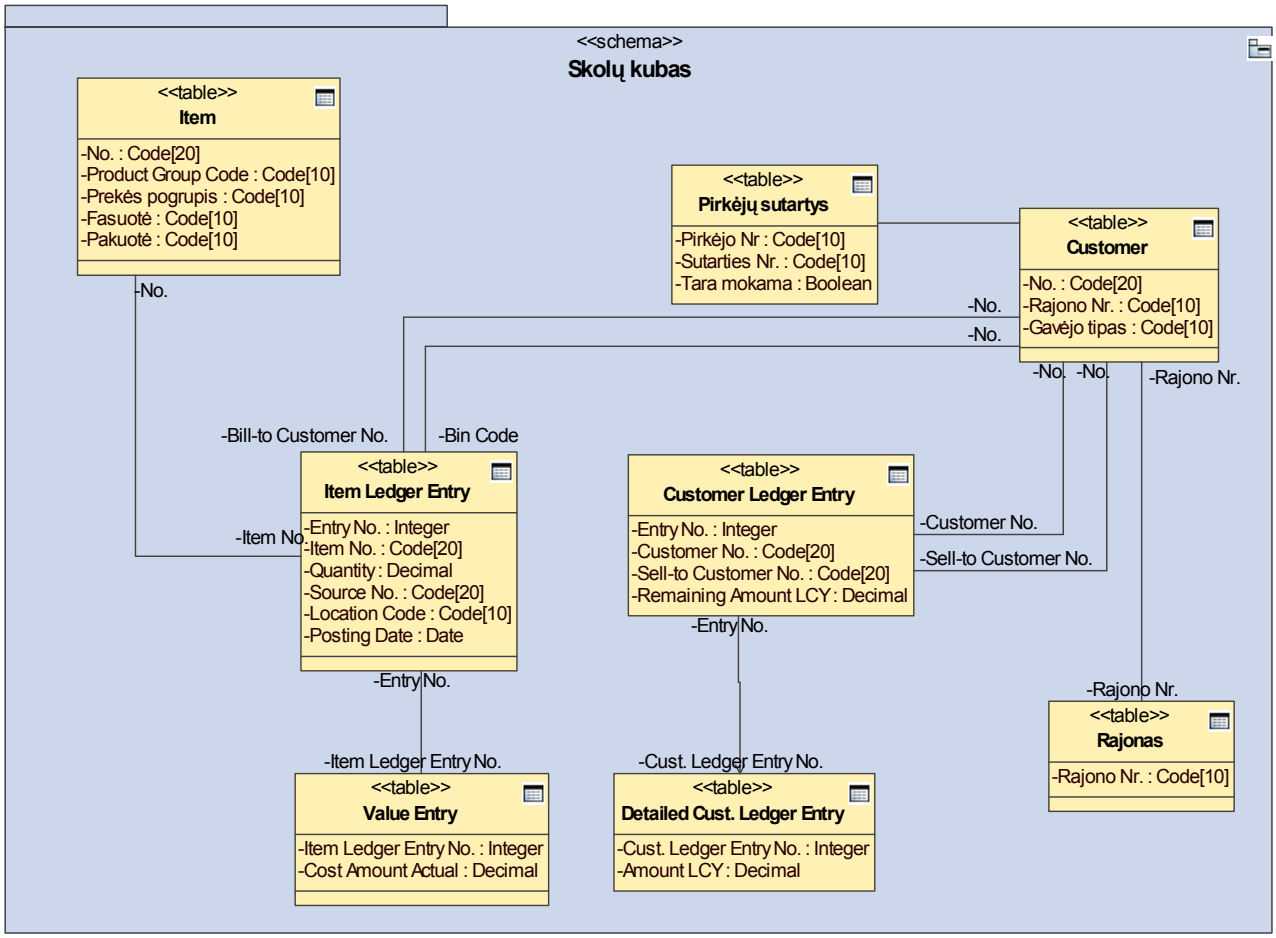
2.8 lentelė. Skolų kubo sąryšiai su Navision Attain sistemos lentelėmis

Dimensijos	Lentelė	Laukas
Data	CRONUS$Cust. Ledger Entry	Posting Date
	CRONUS$Item Ledger Entry	Posting Date
Metai	CRONUS$Cust. Ledger Entry	Posting Date
	CRONUS$Item Ledger Entry	Posting Date
Ketvirtis	CRONUS$Cust. Ledger Entry	Posting Date
	CRONUS$Item Ledger Entry	Posting Date
Mėnuo	CRONUS$Cust. Ledger Entry	Posting Date
	CRONUS$Item Ledger Entry	Posting Date
Diena	CRONUS$Cust. Ledger Entry	Posting Date
	CRONUS$Item Ledger Entry	Posting Date
Dokumento Nr.	CRONUS$Cust. Ledger Entry	Document No.
	CRONUS$Item Ledger Entry	Document No.
Išorinis dokumento Nr.	CRONUS$Cust. Ledger Entry	External Document No.
	CRONUS$Item Ledger Entry	External Document No.
Mokėtojas	CRONUS$Cust. Ledger Entry	Customer No.
	CRONUS$Item Ledger Entry	Bill-to Customer No.
Mokėtojo vadybininkas	CRONUS$Customer	Salesperson Code
Gavėjas	CRONUS$Cust. Ledger Entry	Sell-to Customer No.
	CRONUS$Item Ledger Entry	Bin Code
Gavėjo agentas	CRONUS$Customer	Salesperson Code
Gavėjo tipas	CRONUS$Customer	Gavėjo tipas
	CRONUS$Grupių sąrašai	Kodas, kur Tipas = Gavėjo tipas
Rajonas	CRONUS$Customer	Rajono Nr.
	CRONUS$Rajonas	Rajono Nr.
Geografinė vieta	CRONUS$Cust. Ledger Entry	Global Dimension 1 Code
	CRONUS$Item Ledger Entry	Global Dimension 1 Code
Sistema	CRONUS$Customer	Sistemos kodas
	CRONUS$Grupių sąrašai	Kodas, kur Tipas = Sistema
Pirkėjo reg. gr.	CRONUS$Customer	Customer Posting Group
Funkcinis padalinys	CRONUS$Cust. Ledger Entry	Global Dimension 2 Code
	CRONUS$Item Ledger Entry	Global Dimension 2 Code
Beviltiškas mokėtojas	CRONUS$Customer	Beviltiskas moketojas
Mokėjimo būdo kodas	CRONUS$Cust. Ledger Entry	Mokėjimo būdo kodas
	CRONUS$Item Ledger Entry	-
Apmokama tara	CRONUS$Pirkėjų sutartys	Tara mokama
Tara	-	‚Taip', jeigu duomenys iš *prekių knygos įrašų*

		‚Ne', jeigu duomenys iš *pirkėjų knygos įrašų*
Taros pavadinimas	CRONUS$Item	Description
Pradelstų dienų skaičius	CRONUS$Cust. Ledger Entry	Šiandien – Due Date
Matai	**Lentelė**	**Laukas**
Mokėjimai	CRONUS$Det. Cust. Ledger Entry CRONUS$Item Ledger Entry	Amount (LCY), jei Document Type = Payment 0
Pardavimai	CRONUS$Det. Cust. Ledger Entry CRONUS$Item Ledger Entry	Amount (LCY), jei Document Type = Invoice 0
Grąžinimai	CRONUS$Det. Cust. Ledger Entry CRONUS$Item Ledger Entry	Amount (LCY), jei Document Type = Credit Memo 0
Kita	CRONUS$Det. Cust. Ledger Entry CRONUS$Item Ledger Entry	Amount (LCY), jei Document Type <> Payment, Invoice, Credit Memo 0
Parduota taros	CRONUS$Cust. Ledger Entry CRONUS$Value Entry	0 Cost Amount (Actual), jei Entry Type = Positive Adjmt.
Parduota taros kiekis	CRONUS$Cust. Ledger Entry CRONUS$Item Ledger Entry	0 Quantity, jei Entry Type = Positive. Adjmt.
Grąžinta taros	CRONUS$Cust. Ledger Entry CRONUS$Value Entry	0 Cost Amount (Actual), jei Entry Type = Negative Adjmt.
Grąžinta taros kiekis	CRONUS$Cust. Ledger Entry CRONUS$Item Ledger Entry	0 Quantity, jei Entry Type = Negative Adjmt.
Likusi suma	CRONUS$Cust. Ledger Entry CRONUS$Item Ledger Entry	Remaining Amount (LCY) 0
Skaičiuojamieji laukai		
Likutis pradziai kiekis	Grynojo pokyčio suma iki šio periodo	
Likutis pabaigai kiekis	Grynojo pokyčio suma iki šio periodo imtinai	
Grynasis pokytis kiekis	Parduotas taros kiekis + Grąžintas taros kiekis	
Bendras grynasis pokytis	Pardavimai + Mokėjimai + Grąžinimai + Kita + Parduota taros + Grąžinta taros	
Bendras likutis pradžiai	Bendrojo grynojo pokyčio suma iki šio periodo	
Bendras likutis pabaigai	Bendrojo grynojo pokyčio suma iki šio periodo imtinai	
Produkcijos grynasis pokytis	Pardavimai + Mokėjimai + Grąžinimai + Kita	
Produkcijos likutis pradžiai	Produkcijos grynojo pokyčio suma iki šio periodo	

Produkcijos likutis pabaigai	Produkcijos grynojo pokyčio suma iki šio periodo imtinai
Taros grynasis pokytis	Parduota taros + Grąžinta taros
Taros likutis pradžiai	Taros grynojo pokyčio suma iki šio periodo
Taros likutis pabaigai	Taros grynojo pokyčio suma iki šio periodo imtinai

Šiame kube naudojamas lenteles ir jų sąryšius galima apibendrinti tokia konceptualia duomenų bazės schema (schemoje rodomi tik sąryšiams reikalingi laukai):



2.4 pav. Skolų kube naudojamų lentelių sąryšiai

Atsižvelgiant į naudojamas lenteles, bus kuriami tokie rodiniai:

- *Prekė* – panaudojamas tas pats kaip ir pirkimų kube.
- *Pirkėjas* – skirtas pateikti informaciją apie pirkėjus.
- *Pirkėjų sutartys* – surenka informaciją apie aktyvias pirkėjų sutartis.
- *Vertės įrašai* – naudojamas tas pats kaip ir pirkimų kube.
- *Detalūs pirkėjų įrašai* - grąžina detalius pirkėjų knygos įrašus, sugrupuotus pagal pirkėjų knygos įrašo numerį.
- *Skolos – pagrindinis* – formuoja faktų lentelę.

3. VERSLO ANALIZĖS SISTEMOS PROJEKTAS

Nemažiau svarbu nei detalus ir tikslus vartotojo reikalavimų išgavimas yra ir išbaigto projekto parengimas. Pilnas sistemos projektas sumažina įgyvendinimo klaidų tikimybę, padeda užtikrinti būsimos sistemos atitikimą vartotojų poreikiams. Projektas padeda parengti detalią veiksmų seką, įgalinančią atlikti sistemos realizavimą optimaliu būdu. Rengiant šį projektą, naudojama „iš viršaus žemyn“ metodika – t.y. iš pradžių specifikuojamas norimas gauti galutinis duomenų kubas, toliau detalizuojant iš ko jis bus sudarytas.

3.1. DUOMENŲ TRANSFORMACIJŲ SERVISAI

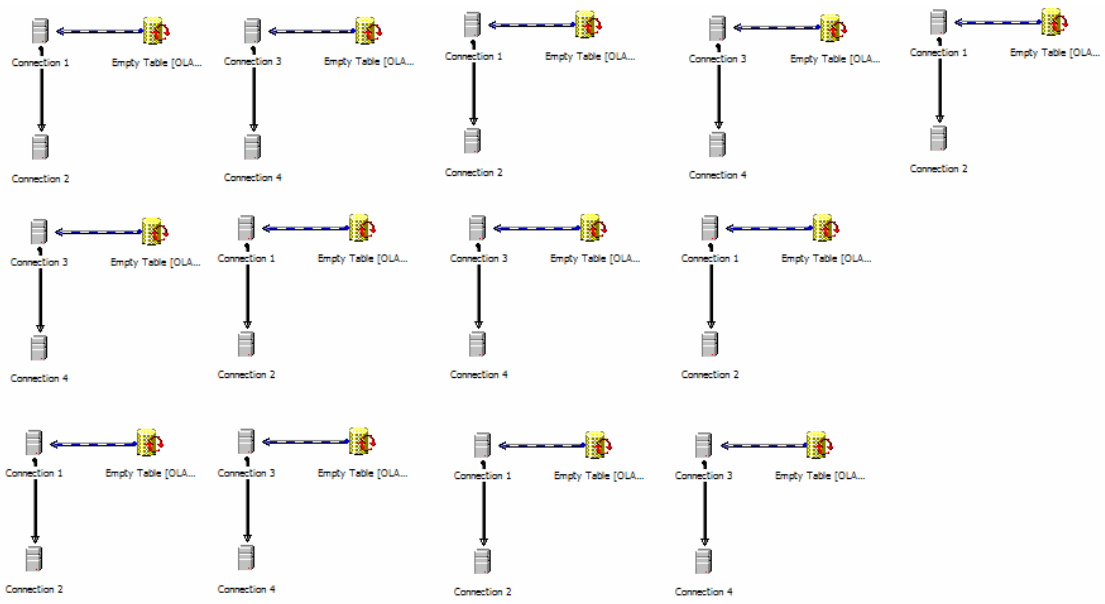
Kaip įvardinta analizės etapo metu, duomenys OLAP duomenų analizei bus saugomi atskiroje nuo darbinės duomenų bazėje. Šiems duomenims perkelti bei suteikti reikiamą formą bus sukurti duomenų transformacijų servisai [4] [6]. Jų pagalbą, naudojant rodinius, duomenys bus paimami iš darbinės sistemos, transformuojami, ir patalpinami OLAP duomenų saugykloje.

3.1.1. BENDRAI NAUDOJAMOS LENTELĖS

Dalis reikalingų duomenų yra bendri keliems kubams. Tai tokios lentelės kaip:

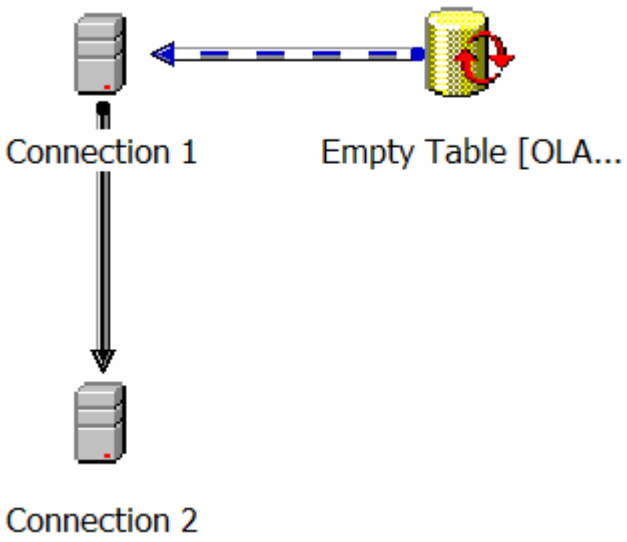
- CRONUS$Item
- CRONUS$Product Group
- CRONUS$Prekių pogrupiai
- CRONUS$Vendor
- CRONUS$Location
- CRONUS$Customer
- CRONUS$Rajonas
- CRONUS$Return Reason
- CRONUS$Fasuotės
- CRONUS$Grupių sąrašai
- CRONUS$Statistinės grupės
- CRONUS$Manufacturer
- CRONUS$Pirkėjų sutartys

Šioms lentelėms perkelti į OLAP duomenų saugyklą bus sukurtas atskiras DTS paketas. Su šiais duomenimis jokios transformacijos nebus atliekamos, jie bus perkeliami tiksliai tokie kokie yra.



3.1 pav. Bendrai naudojamų lentelių perkėlimo DTS paketas

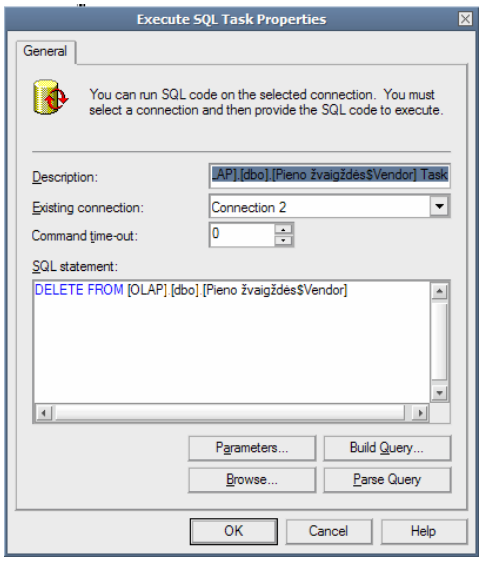
Šiame DTS pakete matyti už skirtingų lentelių perkėlimą atsakingos operacijos. Detaliau panagrinėsime vieną pavyzdį.



3.2 pav. Tiekėjų lentelės perkėlimo DTS

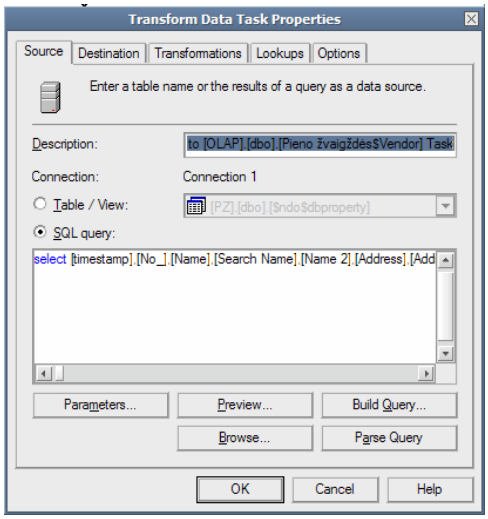
Vienos lentelės perkėlimui atliekami trys žingsniai:

1. Išvaloma esanti lentelė (Empty Table)



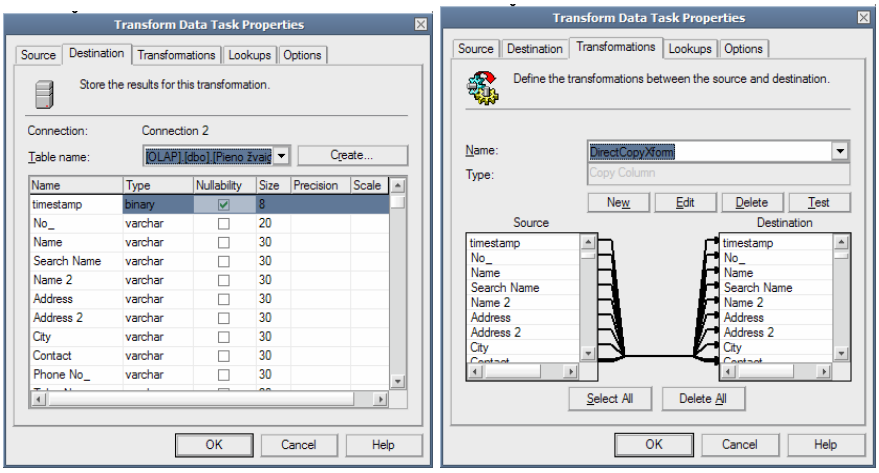
3.3 pav. Lentelės valymas

2. Prisijungiama prie šaltinio SQL serverio, ir nuskaitomi esantys duomenys



3.4 pav. Prisijungimas prie šaltinio serverio

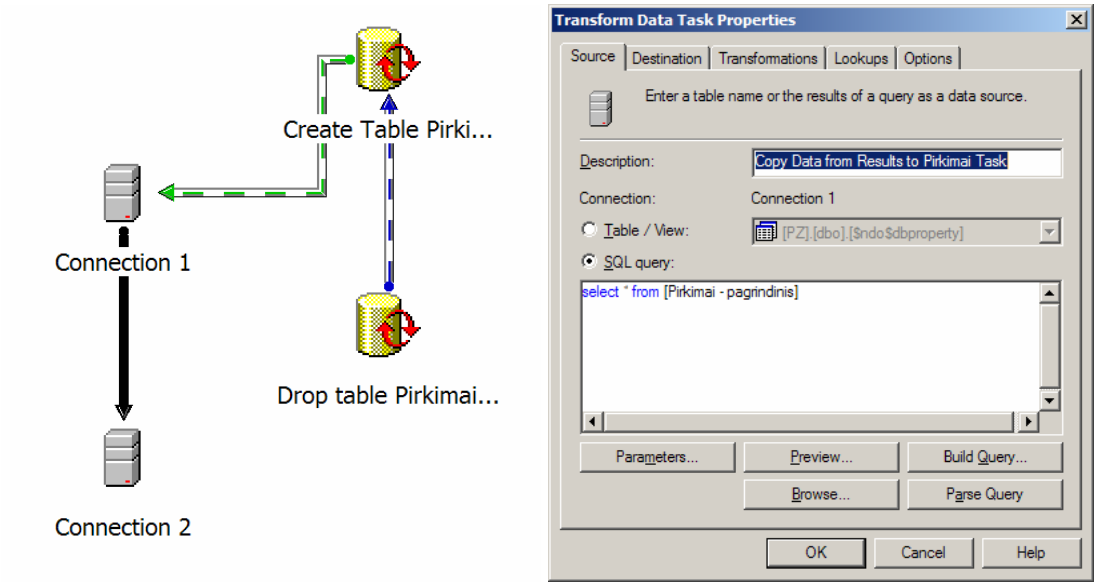
3. Šie duomenys įrašomi į naująjį serverį



3.5 pav. Rezultatų saugojimas

3.1.2. PIRKIMŲ KUBO TRANSFORMACIJOS

Šiam kubui sukurtas DTS paketas, naudodamas *Pirkimai – pagrindinis* rodinį (2 priedas, 4 skyr.), perkels transformuotus duomenis į OLAP duomenų saugyklos *pirkimai* lentelę – ši lentelė ir bus naudojama kaip pirkimų kubo faktų lentelė.



3.6 pav. Pirkimų kubo transformacijos

Šis paketas vykdo keturis žingsnius:

1. Ištrinama esanti faktų lentelė
2. Sukuriama nauja faktų lentelė
3. Suformuojami duomenys iš šaltinio serverio faktų lentelės rodinio
4. Gauti rezultatai patalpinami naujajame serveryje

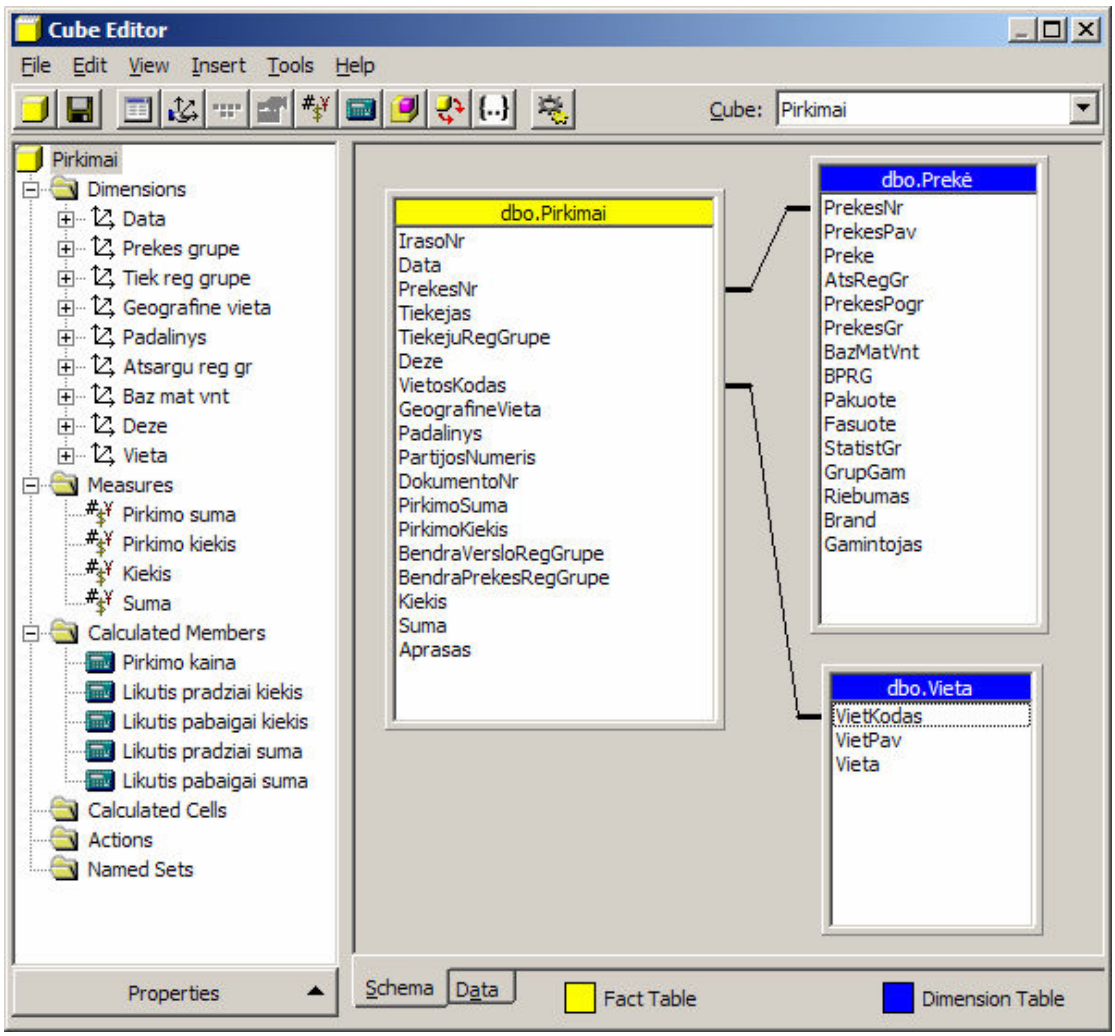
3.1.3. LIKUSIŲ KUBŲ TRANSFORMACIJOS

Likusiųjų kubų faktų lentelių perkėlimai atliekami analogiškai pirkimų kubui, tik naudojant kita rodinį (*Pardavimai – pagrindinis, Skolos – pagrindinis, Atsargos – pagrindinis*, žr. 2 priedą).

3.2. KUBŲ SCHEMOS

3.2.1. PIRKIMŲ KUBO SCHEMA

Naudojant perkeltas bendrai naudojamas duomenų lenteles, ir suformuotą faktų lentelę, sukuriama Pirkimų kubo schema. Schema apjungia visas lenteles pagal loginius sąryšius, įgalina suformuoti visas dimensijas ir matus, sukurti skaičiuojamuosius laukus.



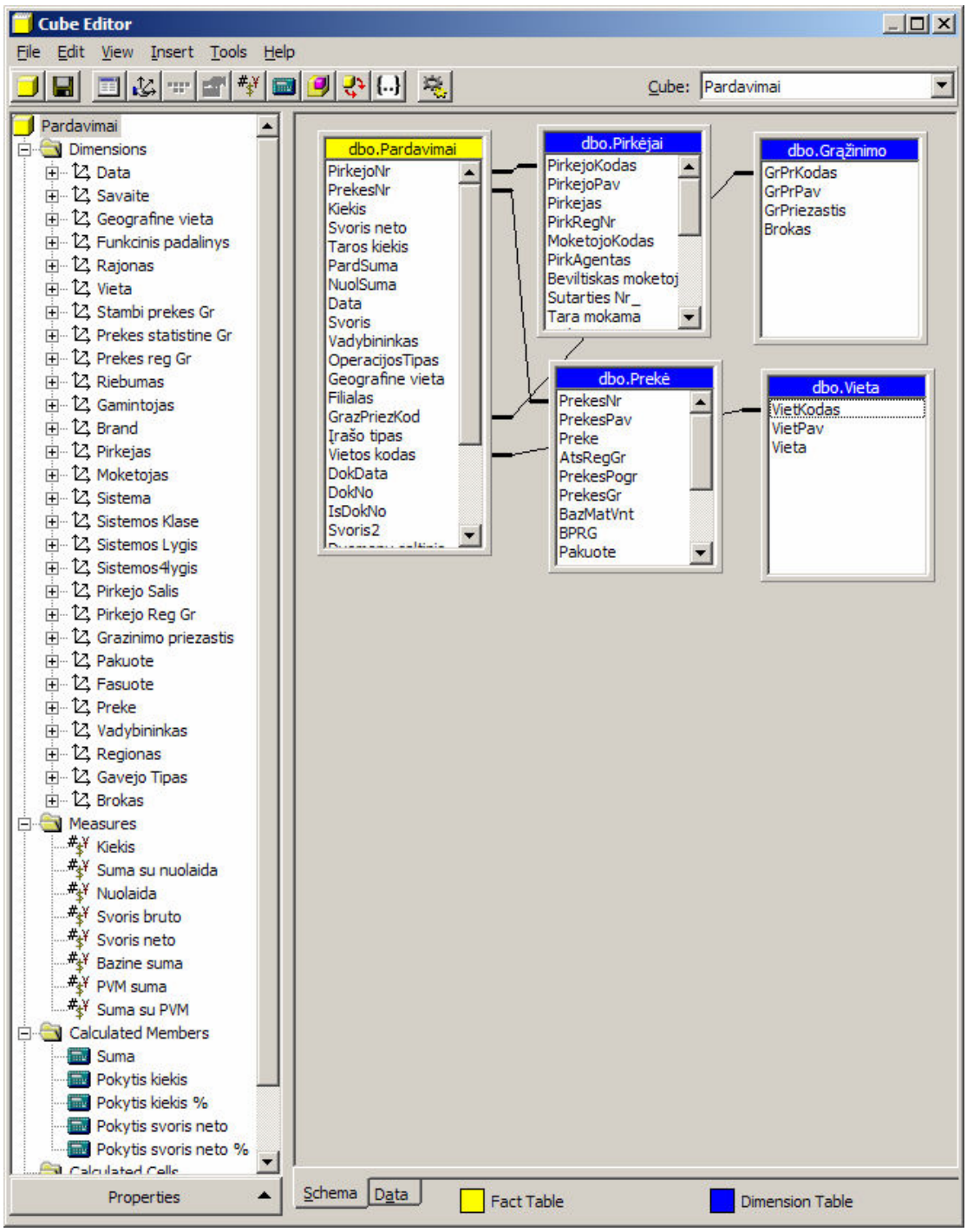
3.7 pav. Pirkimų kubo schema

Kaip matyti iš pavyzdžio, kube naudojama naujoji suformuota faktų lentelė (*pirkimai*), ir dimensijų rodiniai (*prekė, vieta*). Panaši struktūra kartojasi visuose kuriamuose kubuose.

Kaip numatyta, iš faktų bei dimensijų lentelių sukuriamos dimensijos, iš faktų lentelės sukuriami matai, o naudojant jų kombinacijas sukuriami skaičiuojamieji laukai (kairėje).

3.2.2. PARDAVIMŲ KUBO SCHEMA

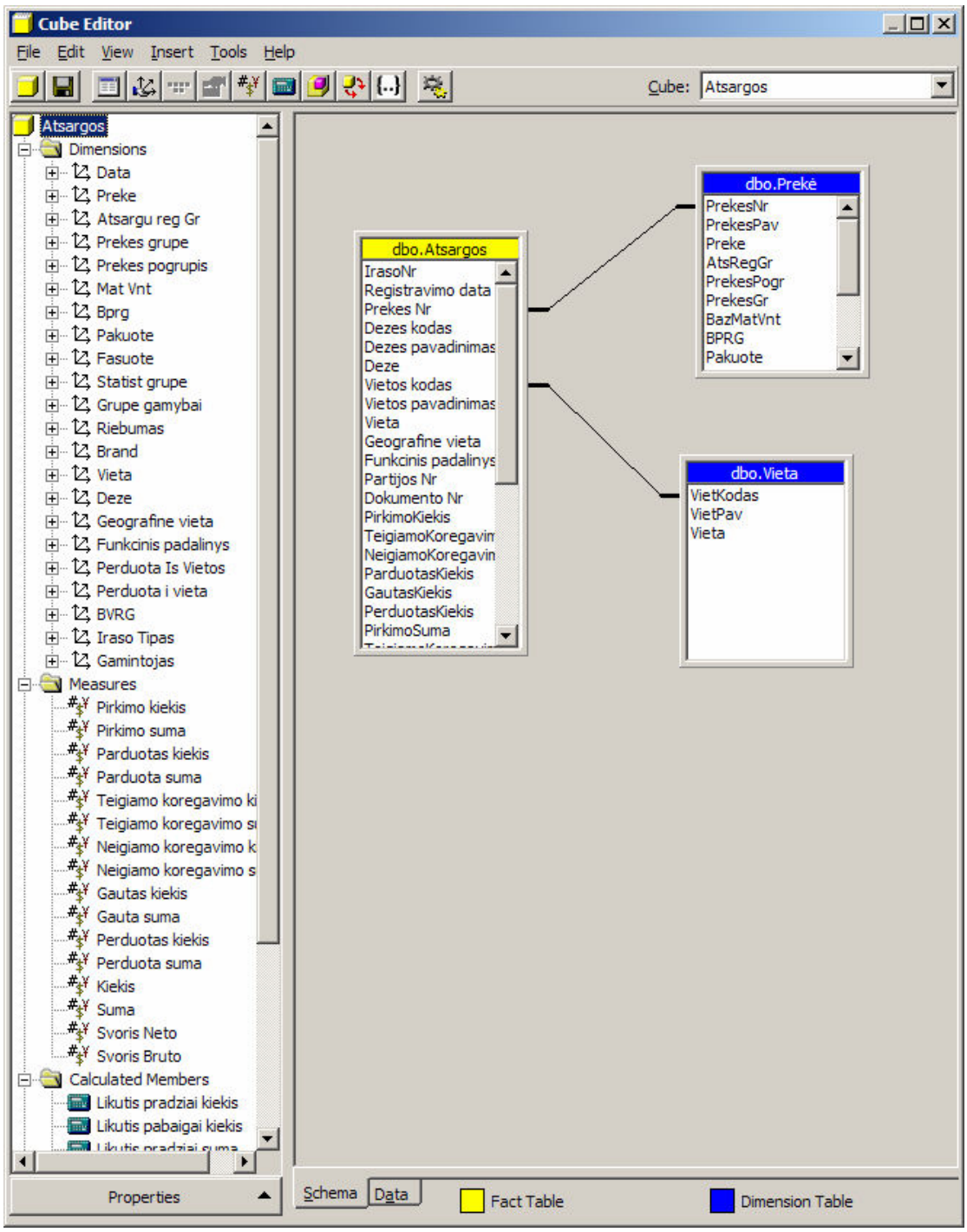
Analogiškai pirkimų kubui, sukuriama ir pardavimų kubo schema.



3.8 pav. Pardavimų kubo schema

3.2.3. ATSARGŲ KUBO SCHEMA

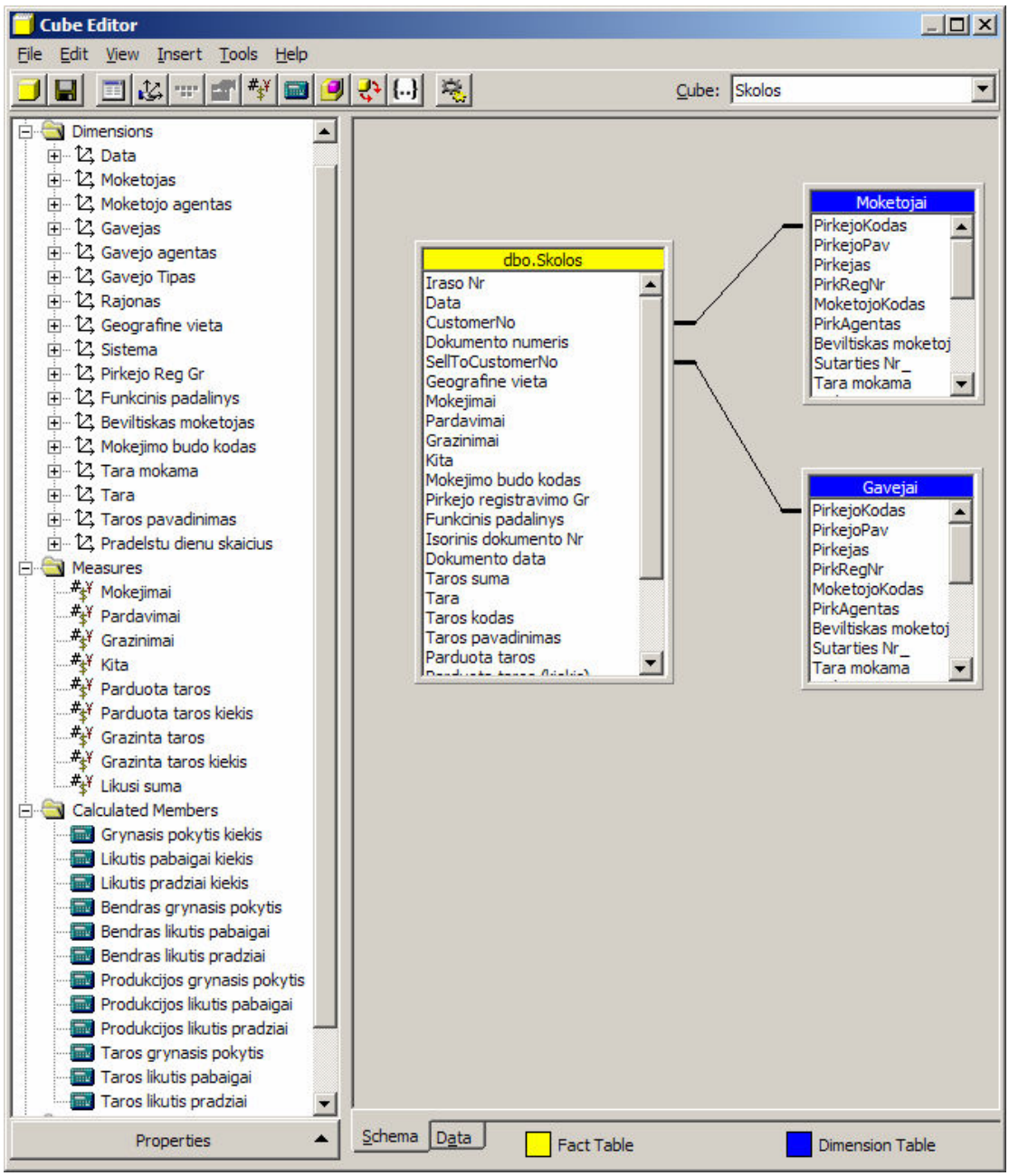
Analogiškai pirkimų kubui, sukuriama ir atsargų kubo schema.



3.9 pav. Atsargų kubo schema

3.2.4. SKOLŲ KUBO SCHEMA

Analogiškai pirkimų kubui, sukuriama ir skolų kubo schema.

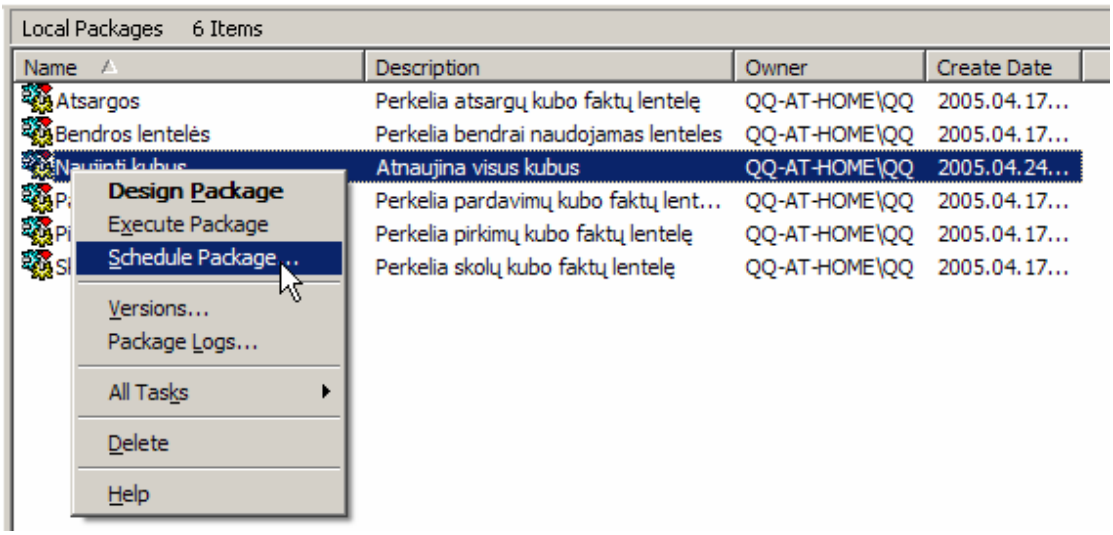


3.10 pav. Skolų kubo schema

3.3. ATNAUJINIMAI

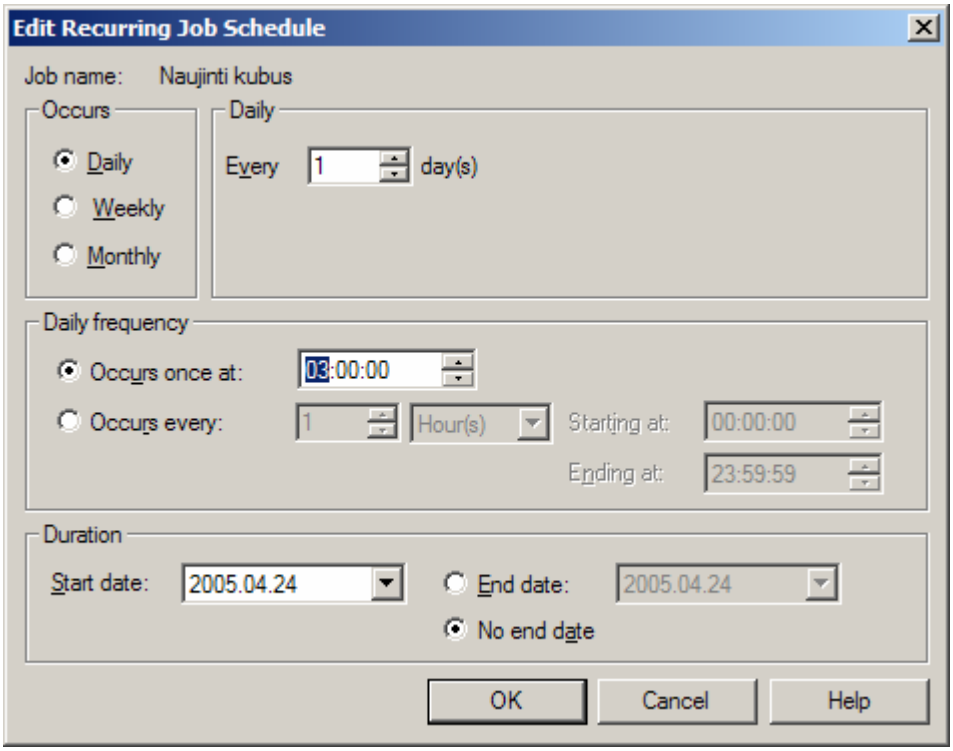
Kadangi visi sukurti kubai turi būti atnaujinami automatiškai nakties metu, jų atnaujinimui bus sukurtas DTS paketas, kuris, SQL Server 2000 priemonių pagalba, bus suplanuotas automatiškam paleidimui naktį.

Sukūrus DTS paketą, jam sukuriame atnaujinimo planą.



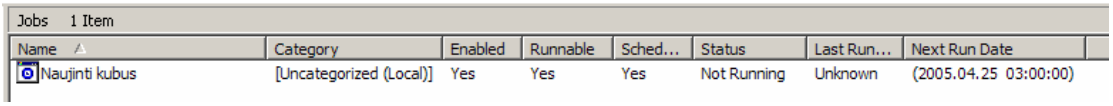
3.11 pav. Kubų atnaujinimo planavimas – pirmas žingsnis

Nurodome kokiu periodiškumu bus atnaujinami kubai.



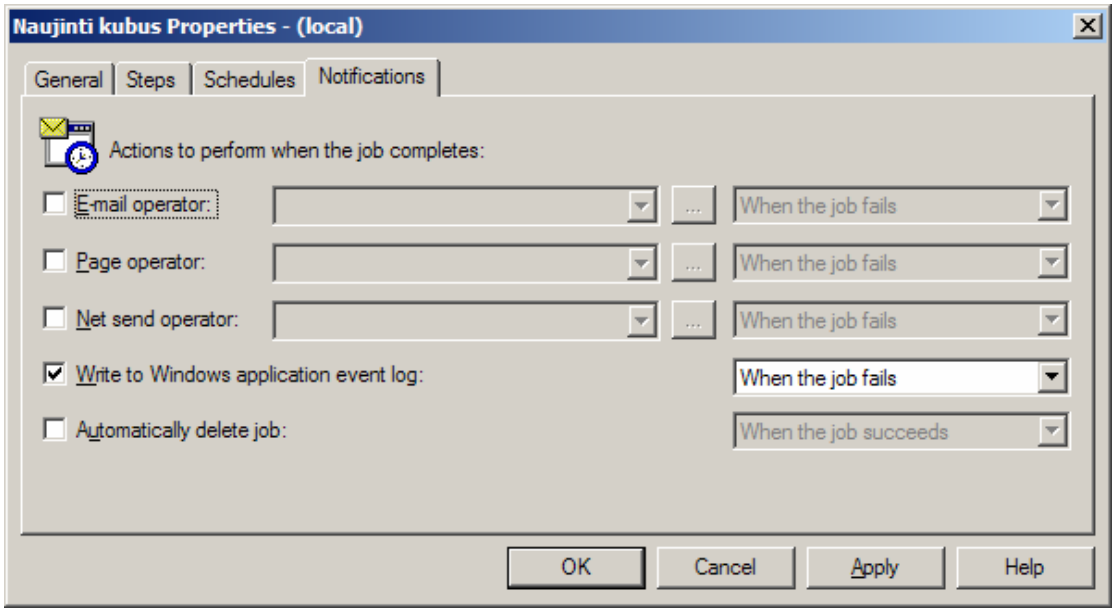
3.12 pav. Kubų atnaujinimo planavimas – antras žingsnis

Tai atlikus, gauname įrašą *SQL Server Agent / Jobs* lange.



3.13 pav. Kubų atnaujinimo planavimas – sukurtas planavimo įrašas

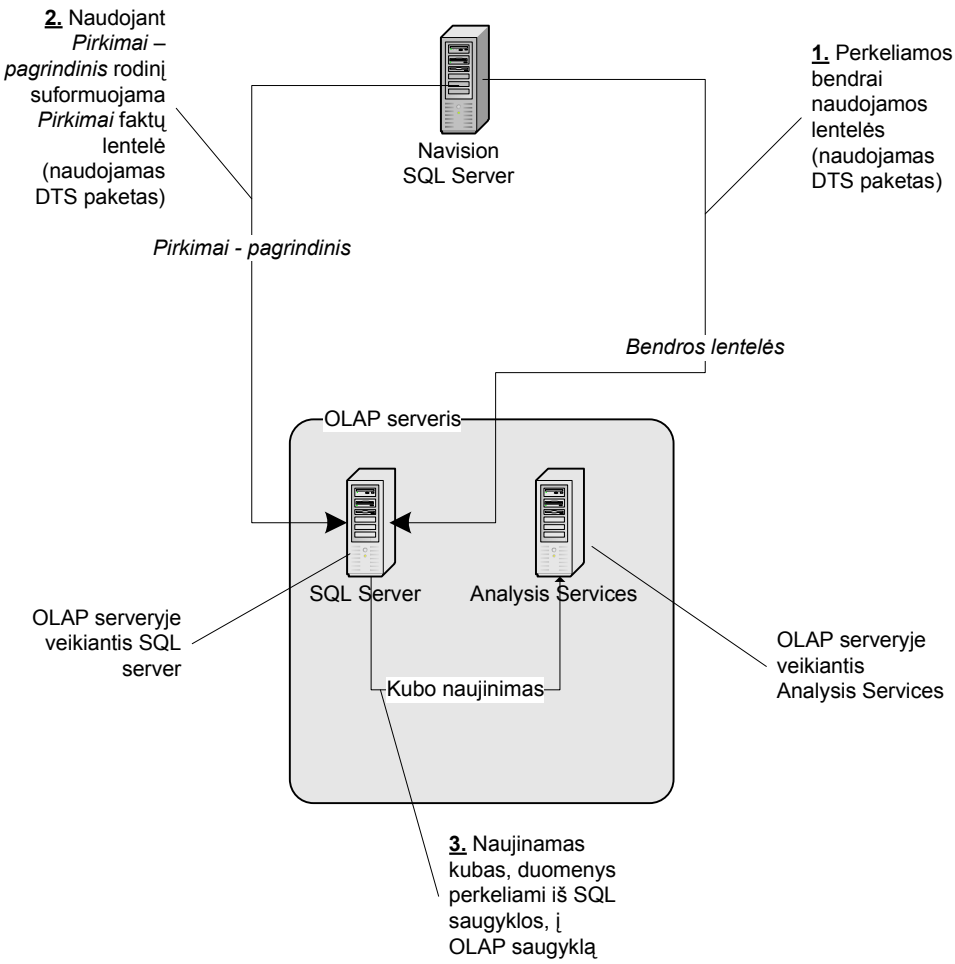
Toliau galima nustatyti įrašo platesnius parametrus, kaip perspėjimus apie nepasisekusį atnaujinimą.



3.14 pav. Kubų atnaujinimo planavimas – perspėjimų nustatymas

3.4. BENDRAS KUBŲ FORMAVIMO PROCESAS

Kiekvienas kubas formuojamas tam tikra procesų seka. Kiekvieno žingsnio metu vyksta duomenų išgavimas / transformacija. Pateikiamas Pirkimų kubo formavimo etapų pavyzdys.



3.15 pav. Kubų formavimo procesas

Formavimo žingsniai:

1. Perkeliamos bendrai naudojamos lentelės – DTS paketas *Bendros lentelės*
    a. Išvalomos senos lentelės
    b. Nuskaitomi nauji duomenys iš Navision SQL serverio
    c. Įrašomi nauji duomenys į OLAP SQL serverį
2. Formuojama faktų lentelė *Pirkimai*
    a. Ištrinama esanti faktų lentelė
    b. Perkuriama faktų lentelė
    c. Formuojami rodinio *pirkimai – pagrindinis* rezultatai
    d. Rezultatai įrašomi į OLAP SQL serveryje esančią faktų lentelę
3. Vykdomas kubo atnaujinimas
    a. Atnaujinamos dimensijos
    b. Duomenys perkeliami iš SQL saugyklos į OLAP saugyklą
    c. Suskaičiuojamos agregacijos

Paskutinio etapo (*vykdomas kubo atnaujinimas*) visi žingsniai atliekami automatiškai paleidus *Analysis Services* kubo atnaujinimo procesą.

4. OLAP METAMODELIS IR KŪRIMO PROCESAS

4.1. OLAP METAMODELIS

1999 metų balandžio 20 dieną, tuomet dar gyvavusi *Meta Data Coalition* paskelbė viešai priimanti *Open Information Model* metaduomenų modelį kaip naują standartą [15]. Šis universalus modelis buvo sukurtas Microsoft ir keletos dešimčių kitų įmonių bendromis pastangomis.

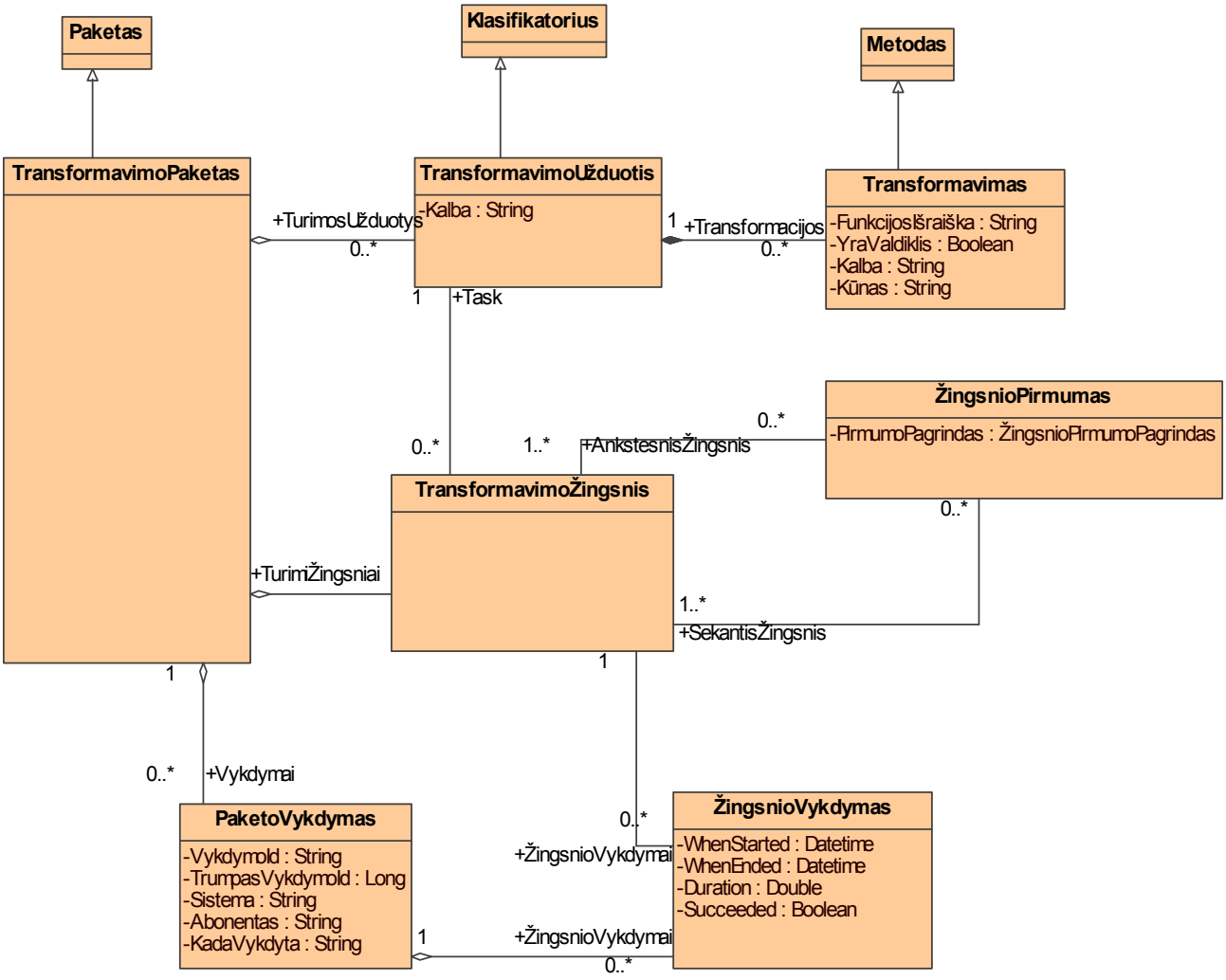
Microsoft SQL Server 2000 naudoja OIM metaduomenims apie visus duomenų tipus saugoti – lenteles, laukus, sąryšius, transformacijas. Analysis Services metaduomenys taip pat atitinka OIM.

OIM yra ypač detalus modelis, apimantis daugelį pagrindinių duomenų bazių valdymo sistemų bei duomenų sandėlių sistemų struktūrų. 2000 metų rugsėjio 25 dieną, MDC oficialiai prisijungė prie *Object Management Group*, o geriausios OIM savybės buvo panaudotos kuriant *Common Warehouse Model* (CWM) modelį.

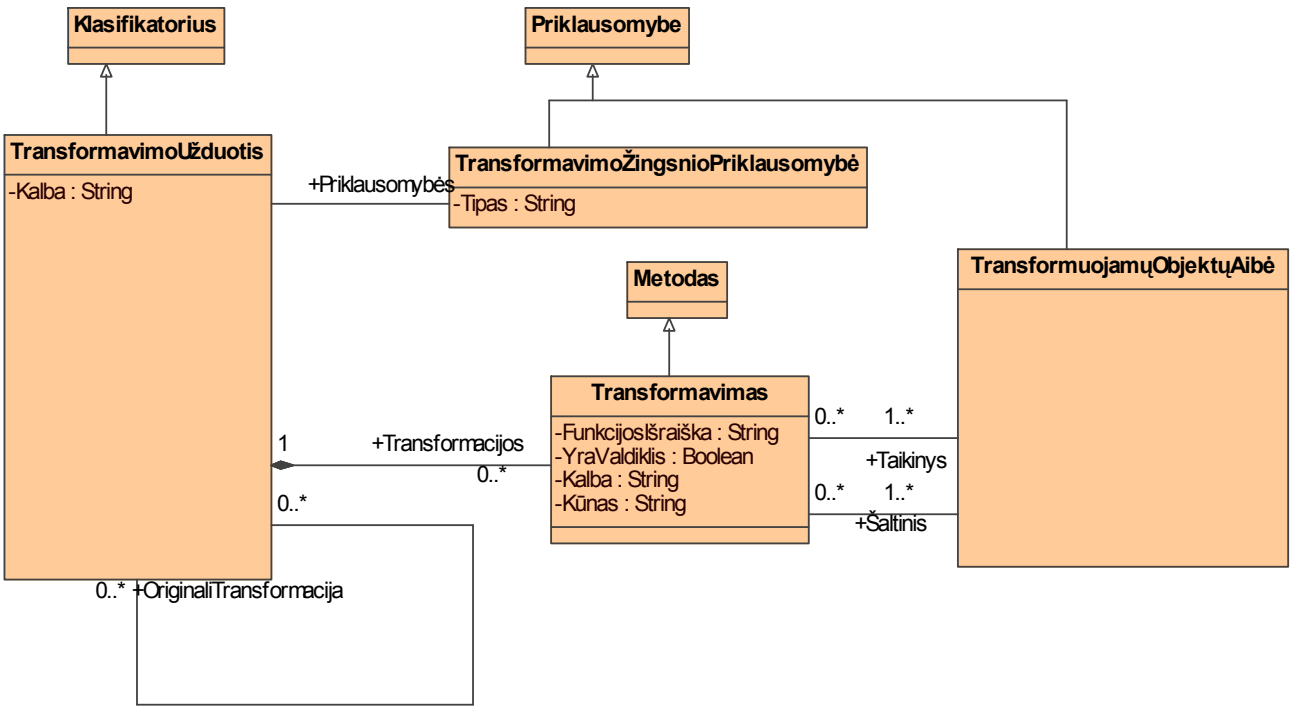
Svarbiausios OIM modelio schemos apima:

1. Duomenų bazių sistemos
    - Lentelės, laukai, rodiniai
    - Apribojimai
    - Trigeriai ir procedūros
    - Indeksai
    - Integralumas
    - Raktai
    - Katalogai ir prisijungimai
    - Duomenų tipų sąryšiai
2. Transformacijos
    - Transformacijų talpyklos
    - Transformacijų etapai
    - Apribojimai ir konvertavimas
3. Duomenų sandėlių sistemos
    - OLAP serveriai ir duomenų bazės
    - Saugyklos, kubai ir skirsniai
    - Hierarchijos ir lygiai

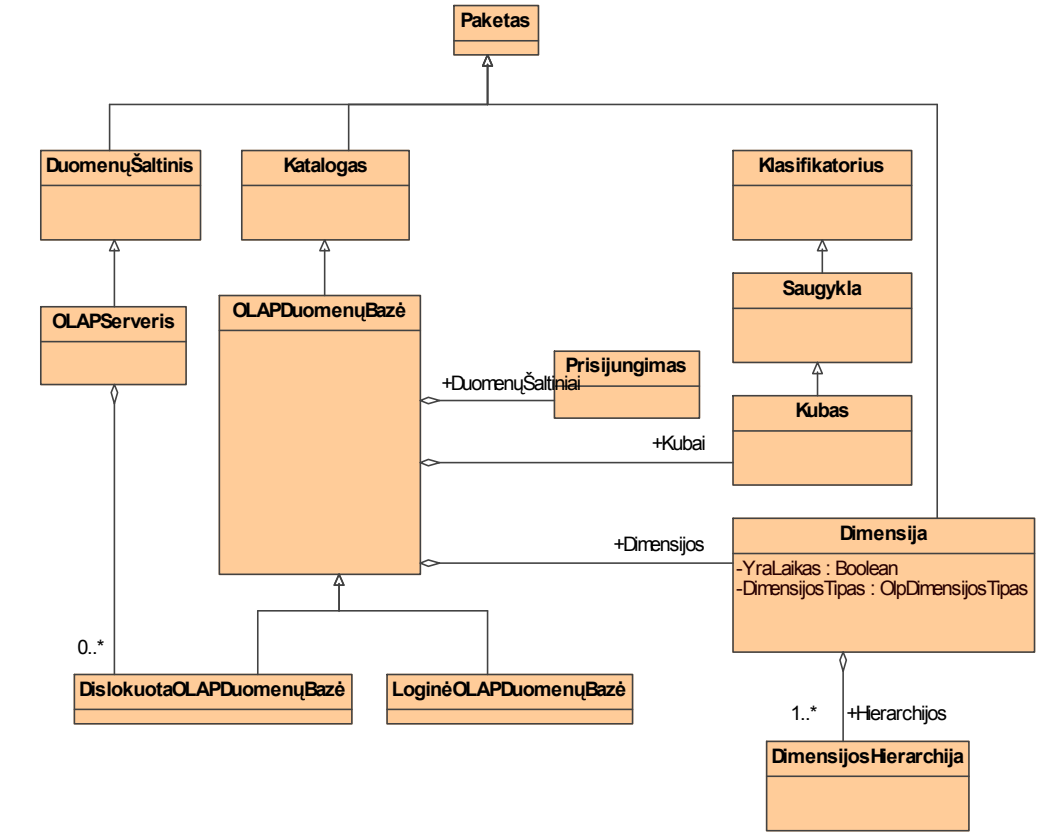
Šiam darbui aktualiausios keturios diagramos: transformacijų talpyklų, transformacijų etapų, OLAP serverių, duomenų bazių, bei saugyklų, kubų ir skirsnių. Supaprastinti jų variantai pateikiami žemiau.



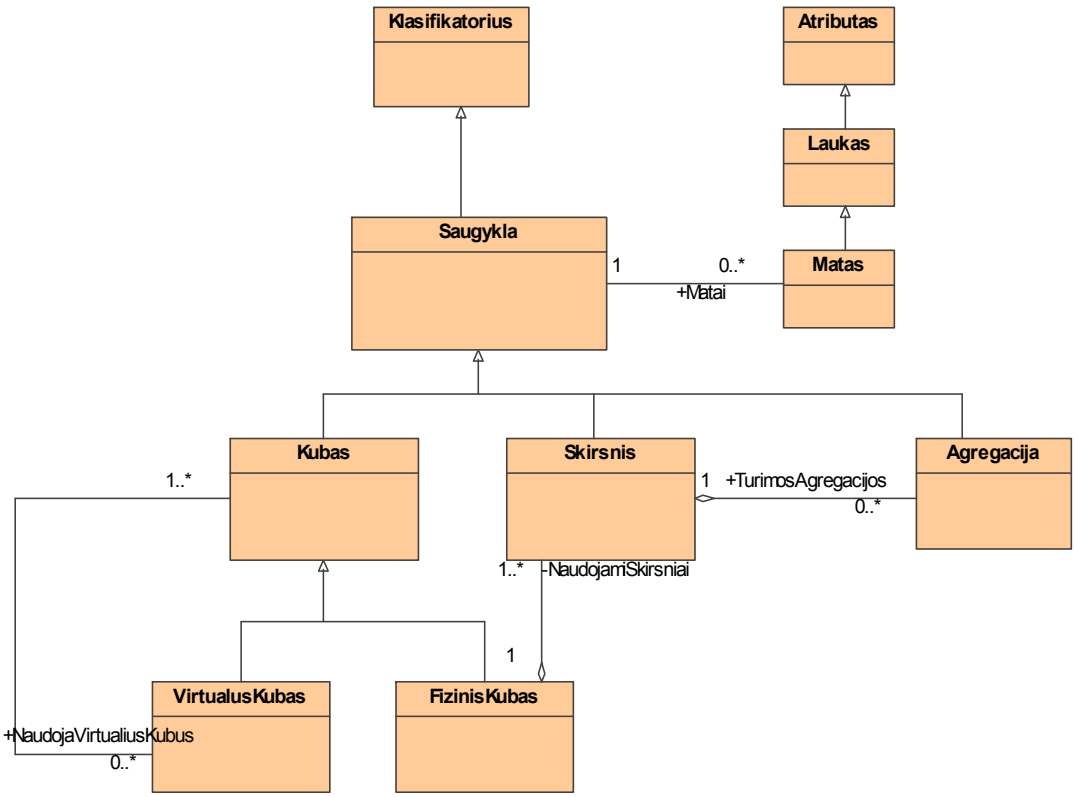
4.1 pav. Transformacijų talpyklos



4.2 pav. Transformacijų etapai



4.3 pav. OLAP serveriai ir duomenų bazės



4.4 pav. Saugyklos, kubai ir skirsniai

Likusios diagramos skirtos apibrėžti pradinių duomenų saugyklas, duomenų apribojimus ir pan.

4.2. ANALIZĖS MODELIŲ METADUOMNŲ EKSPORTAS NAUDOJANT XML

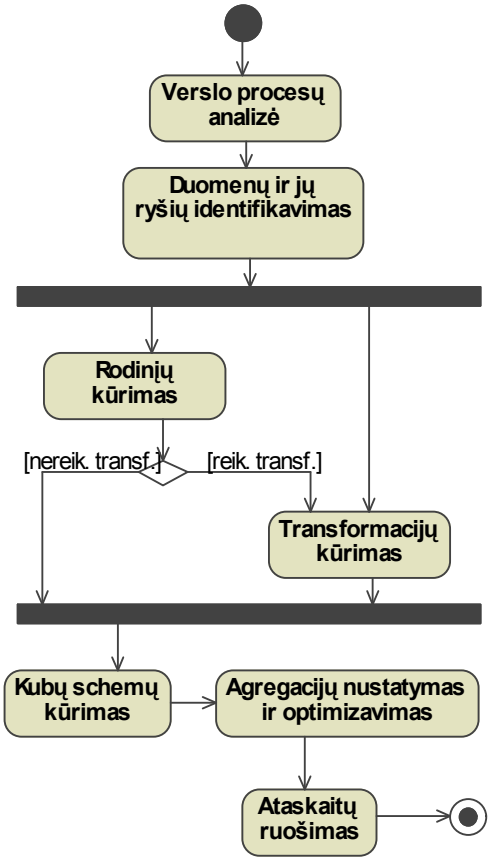
Turėti išsamų metaduomenų modelį nepakanka – reikia galimybės šiuos metaduomenis paimti ir apdoroti. Microsoft Analysis Services atveju, yra pateikiama priemonė, kurios pagalba sukurtus OLAP duomenų analizės kubus galima eksportuoti/importuoti iš/į XML dokumentus. Tokiu būdu XML struktūra pateikiamos visos kubo savybės, kurias galima redaguoti išorinių priemonių pagalba, ir atnaujinti Analysis Services serveryje. Visgi šiuo atveju Microsoft nesilaikė savo pačios sukurtų metaduomenų modelių. XML eksportas yra realizuotas naudojant *Decision Support Objects* (DSO) sąsają, kurios pagalba galima pasiekti visus Analysis Services saugomus metaduomenis.

Eksportuoto kubo XML pavyzdys pateikiamas 3 priede.

4.3. ANALIZĖS KUBŲ FORMAVIMO PROCESAS

Šio darbo tikslas yra ne tik sukurti konkrečią sistemą, integruojančią užsakovo verslo valdymo sistemą Navision Attain bei Microsoft teikiamas OLAP priemones, tačiau ir sukurti universalų šių platformų integravimo proceso modelį. Tai padėtų ateityje greitai diegti panašius sprendimus, išvengti dažniausiai pasikartojančių klaidų.

Supaprastintas pirminis proceso modelis, sukurtas remiantis literatūroje teikiamais modeliais [9] pateikiamas žemiau.



4.5 pav. Verslo valdymo sistemos ir OLAP priemonių integravimo proceso modelis

Kiekvienas iš etapų gali būti detalizuotas – dalis šio detalizavimo pateikiama kituose šio darbo skyriuose, kita dalis bus patikslinta vėlesnių diegimų metu.

4.4. UŽKLAUSŲ SPECIFIKAVIMAS KALBA MDX

Užklausoms į OLAP kubus formuoti naudojama kalba MDX (angl. *Multidimensional Expressions*). Ši kalba pirmą kartą pasirodė 1998, kai tapo naudojama keliose komercinėse programose. Kalba įgalina rašyti užklausas, kurių pagalba suformuojamos daugiadimensinės struktūros. Kalba yra panaši į SQL kalbą, tačiau turi specifines struktūras, kurių pagalba nurodomas eilučių, stulpelių turinys, apskaičiuojami dimensijų nariai, skaičiuojamieji laukai, ir t.t.

MDX kalbos pagalba iš šaltinio kubo, pritaikant nurodytas taisykles formuojamas rezultatų kubas. Bet kuri dimensija, matas ar suskaičiuotas laukas gali būti patalpintas ant bet kurios *ašies*. Ašių rezultatų kube gali būti bet kiek.

„{}“ skliausteliai žymi aibes. Aibės gali būti sudarytos iš dimensijų narių, suskaičiuotų narių, kelių aibių tarpusavio operacijų, funkcijų rezultatai, ir panašiai.

Vienoje ašyje galima pavaizduoti daugiau nei vieną dimensiją. Tokiu atveju naudojama dimensijų sankirta, ir suformuojamos angliškai *tuples*. Tai dimensijų narių kombinacijos.

Žemiau pateikiamas MDX užklausos pavyzdys, suformuojantis lentelę, kur eilutėse yra data, stulpeliuose tiekėjo registravimo grupė. Kaip duomenys rodomas matas *kiekis*, ir filtruojama pagal prekės grupę.

```
SELECT
{Hierarchize({{[Tiek reg grupe].[All Tiekeju Reg Grupe],AddCalculatedMembers({[Tiek reg grupe].[All Tiekeju Reg Grupe].children})}})}
ON
0,
{Hierarchize({{[Data].[All Data],AddCalculatedMembers({[Data].[All Data].children,[Data].[All Data].[2003].children})}})}
ON
1
FROM
Pirkimai]
WHERE
([Measures].[Kiekis],[Prekes grupe].[All Prekes Gr].[Maisto priedai])
      CELL PROPERTIES VALUE, FORMATTED_VALUE, FONT_NAME, FONT_SIZE, FONT_FLAGS,FORE_COLOR, BACK_COLOR, FORMAT_STRING
```

4.6 pav. Pavyzdinė MDX užklausa

5. VERSLO ANALIZĖS SISTEMOS PROTOTIPO EKSPERIMENTINIS TYRIMAS

Šiame skyriuje sukurti duomenų kubai yra optimizuojami naudojantis literatūroje pateiktomis bei kūrimo metu įgytomis žiniomis. Optimizavimu siekiama kiek įmanoma sumažinti kubų naujinimo trukmę (kuri yra tiesiogiai proporcinga duomenų bazės serverio apkrovimo padidėjimui) bei užklausų į kubus generavimo laikus. Numatoma atlikti trijų lygių optimizavimą:

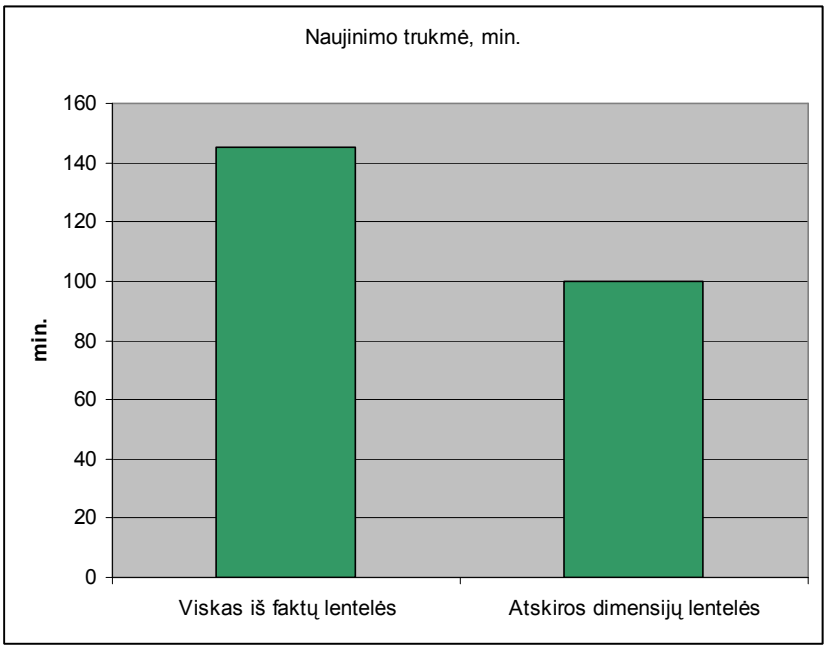
- kubų schemų optimizavimas;
- techninės įrangos optimizavimas;
- duomenų sandėlio optimizavimas.

Eksperimentinio tyrimo metu gauti rezultatai ir žinios bus apibendrintos ir pateiktos kaip bendro pobūdžio rekomendacijos [16].

5.1. OPTIMIZAVIMAS IR REKOMENDACIJOS

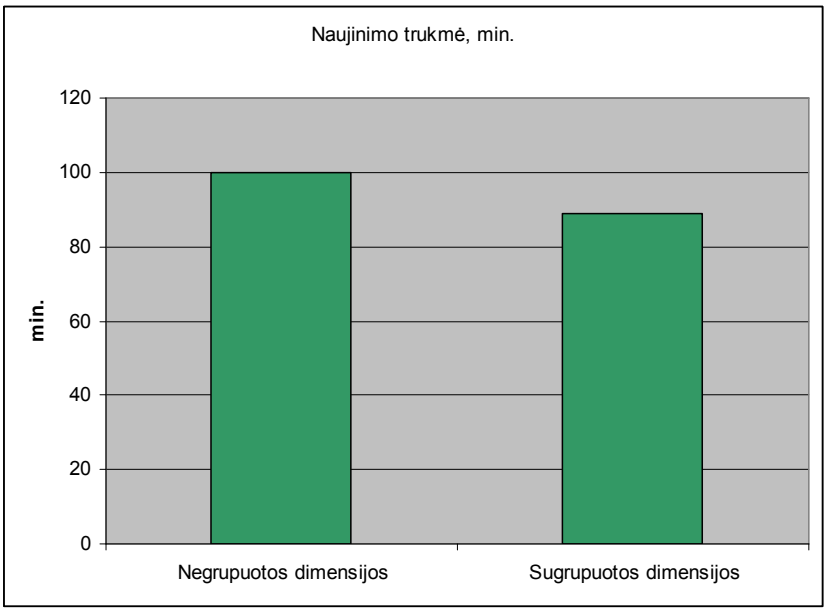
Bekuriant numatytus duomenų analizės kubus, buvo susidurta su keletu skirtingų problemų. Viena iš svarbiausių buvo ta, kad ir naudojant naująsias duomenų analizės priemonės, esama darbinė sistema buvo gana smarkiai apkraunama. Tai ypač pasireikšdavo kubų atnaujinimo metu. Šią problemą reikėjo spręsti greitai, nes vienas iš svarbiausių naujai kuriamos sistemos reikalavimų buvo sumažinti esamos darbinės sistemos apkrovas. Naudojantis esama literatūra ir eksperimentais, buvo imtasi tokių žingsnių:

1. Naudoti atskirus rodinius dimensijų lentelėms. Iš pradžių visos dimensijos buvo imamos iš faktų lentelės, kas labai prailgindavo kubų atnaujinimo laiką. Perkėlus dimensijas į atskirus rodinius, tapo įmanomas dimensijų pakartotinis panaudojimas, smarkiai sutrumpėjo jų atnaujinimo laikas.



5.1 pav. Atnaujinimo laiko sutrumpėjimas

2. Panaudotas dimensijų grupavimas. Pasak literatūros, sugrupuotos dimensijos atnaujinamos greičiau negu pavienės. Tai panaudota grupuojant datos dimensiją, bei prekių grupės / pogrupio / prekės dimensijas. Pastebėtas nežymus atnaujinimo pagreitėjimas.

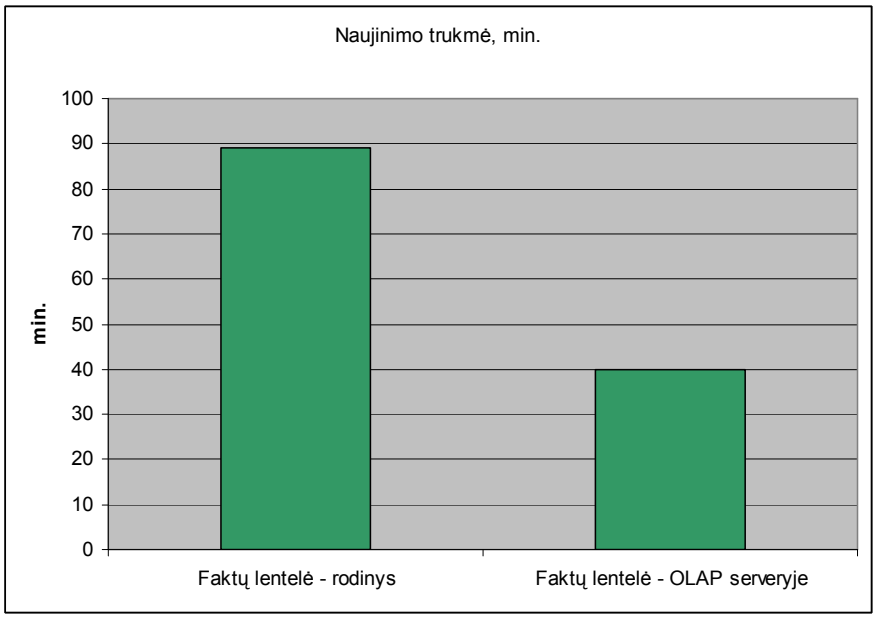


5.2 pav. Atnaujinimo laiko sutrumpėjimas

3. Particijų naudojimas [13]. Kadangi tikimybė praeitiems analizės periodams keistis mažėja tuo labiau, kuo senesni jie yra, todėl juos atnaujinti kiekvieną kartą nėra prasmės. Dėl to, visi esami kubai buvo suskirstyti į atskiras particijas pagal ketvirčius. Kiekvieną naktį atnaujinama tik paskutinio ketvirčio particija. Senesnės particijos atnaujinamos tik kartą į

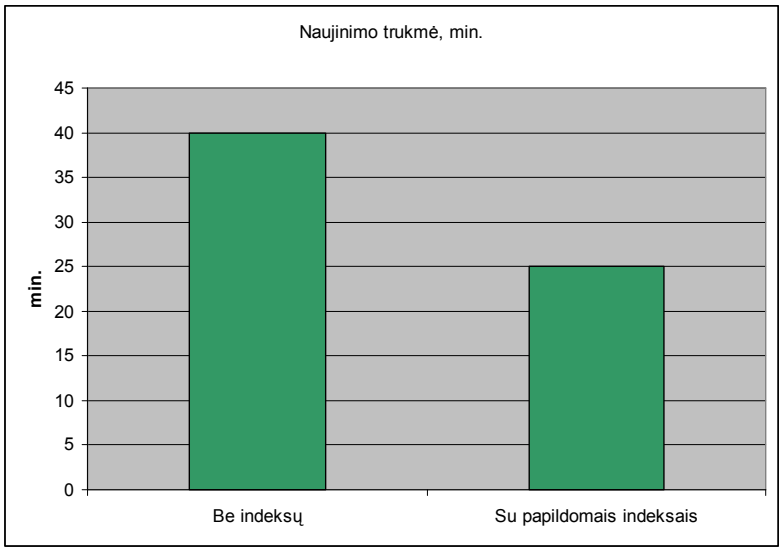
savaitę. Taip sutaupoma daug resursų, nes naujinimo laikas priklauso tiesiogiai nuo įrašų skaičiaus.

4. Faktų lentelės perkėlimas į OLAP serverį [13]. Naudojant rodinį kaip faktų lentelę, SQL serveris turėdavo kiekvieną kartą naujai formuoti jo rezultatus. Tai trukdydavo veikti SQL serverio vidinėms optimizacijoms, ir dėl to smarkiai sulėtėdavo kubų atnaujinimas. Tuo tikslu, faktų lentelės rodinio rezultatus išsaugome OLAP serveryje. Tai žymiai pagreitino kubų atnaujinimą:



5.3 pav. Atnaujinimo laiko sutrumpėjimas

5. Faktų lentelei tapus realia lentele, o ne rodiniu, atsirado galimybė dar vienai optimizacijai – papildomų indeksų bei statistikos generavimui. Įjungus dimensijas atitinkančius indeksus, bei statistikos generavimą naujoje faktų lentelėje, atnaujinimo greitis sumažėjo dar labiau. Tačiau tai turėjo kainą – smarkiai padidėjo naujosios lentelės užimama vieta, bei sulėtėjo jos kūrimas iš rodinio.



5.4 pav. Atnaujinimo laiko sutrumpėjimas

Kuriant naujas panašaus pobūdžio sistemas, naudinga būtų atsižvelgti į šiuos optimizavimus dar prieš pradedant projektavimą. Taip galima sutaupyti daug laiko ir resursų – tiek projektuojant, tiek ir naudojant sukurtą sistemą. Be jau išvardintų optimizavimų, dar galima paminėti keletą patarimų, kurių pagalba atnaujinimo laiką būtų galima sutrumpinti dar labiau:

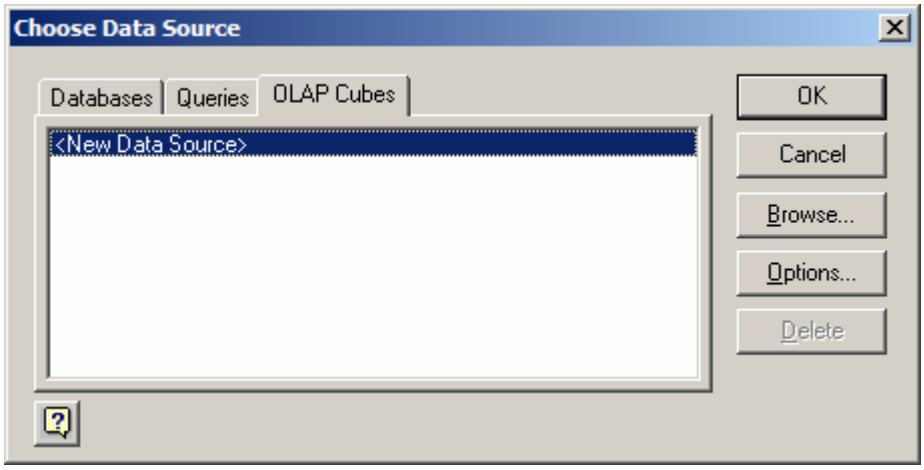
1. Atskira lentelė datos dimensijai. Literatūra teigia, jog datos dimensiją naudoti iš faktų lentelės yra nerekomenduotina. Viena, tai dėl to, jog ji turi daug lygių, o tai lėtina atnaujinimą. Antrą, ją naudinga būtų turėti vieną visiems kubams. Tam tikslui rekomenduotina sukurti atskirą lentelė, su visomis reikalingomis datomis, ir šią lentelę naudoti kaip datos dimensiją.
2. Naudoti bendro naudojimo (angl. *shared*) dimensijas. Tokias dimensijas atnaujinti reikia tik vieną kartą, tad jei jos naudojamos skirtinguose kubuose sutaupoma nemažai resursų.
3. Nenaudoti per daug agregacijų – reikalingų agregacijų kiekis didėja žymiai greičiau nei gaunamas užklausų pagreitėjimas. Todėl svarbu rasti balansą tarp užklausų greičio, ir agregacijų kiekio, nes kuo daugiau agregacijų, tuo lėčiau atnaujinamas kubas.

5.2. NAUDOJIMO INFORMACIJA IR ATASKAITŲ PAVYZDŽIAI

Sukurtiems kubams naudoti bus naudojama Microsoft Excel programinė įranga. Šia programa patogu kurti analizės lenteles bei grafikus, ji jau yra įdiegta užsakovo kompiuteriuose.

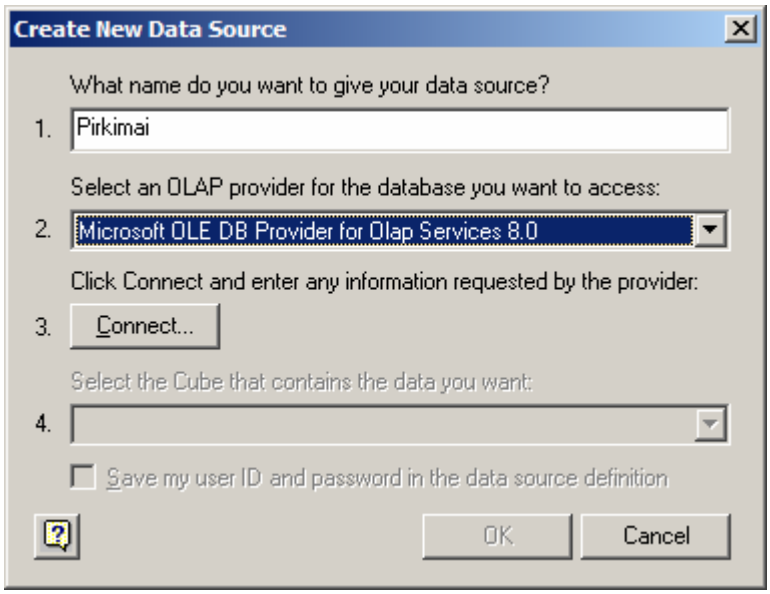
Pateikiama trumpa instrukcija kaip naudoti sukurtus duomenų analizės kubus:

1. Pasileidus *Microsoft Excel*, renkamės meniu punktą *Data -> Import External Data -> New Database Query*. Atsiradusiame lange, pasirenkame *OLAP Cubes* skirtuką:



5.5 pav. Duomenų šaltinio kūrimas

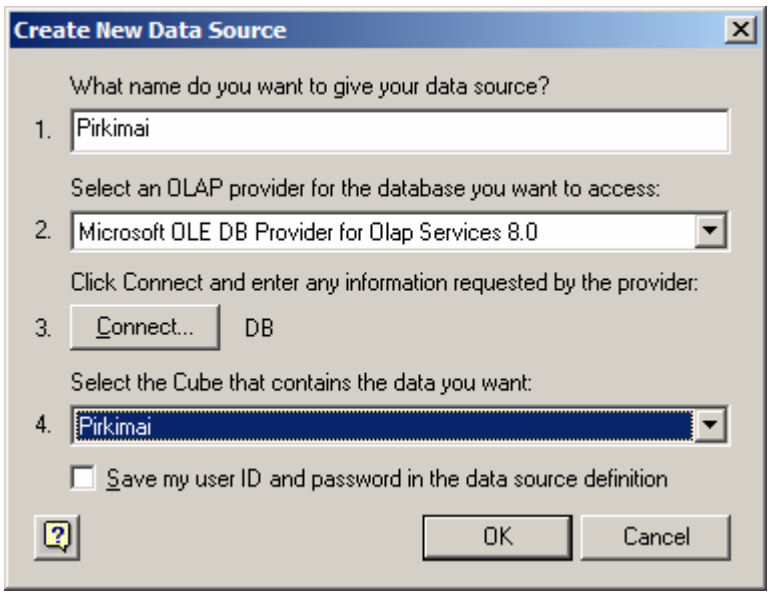
2. Pažymime <*New Data Source*> ir spaudžiame OK.



5.6 pav. Suteikiamas pavadinimas

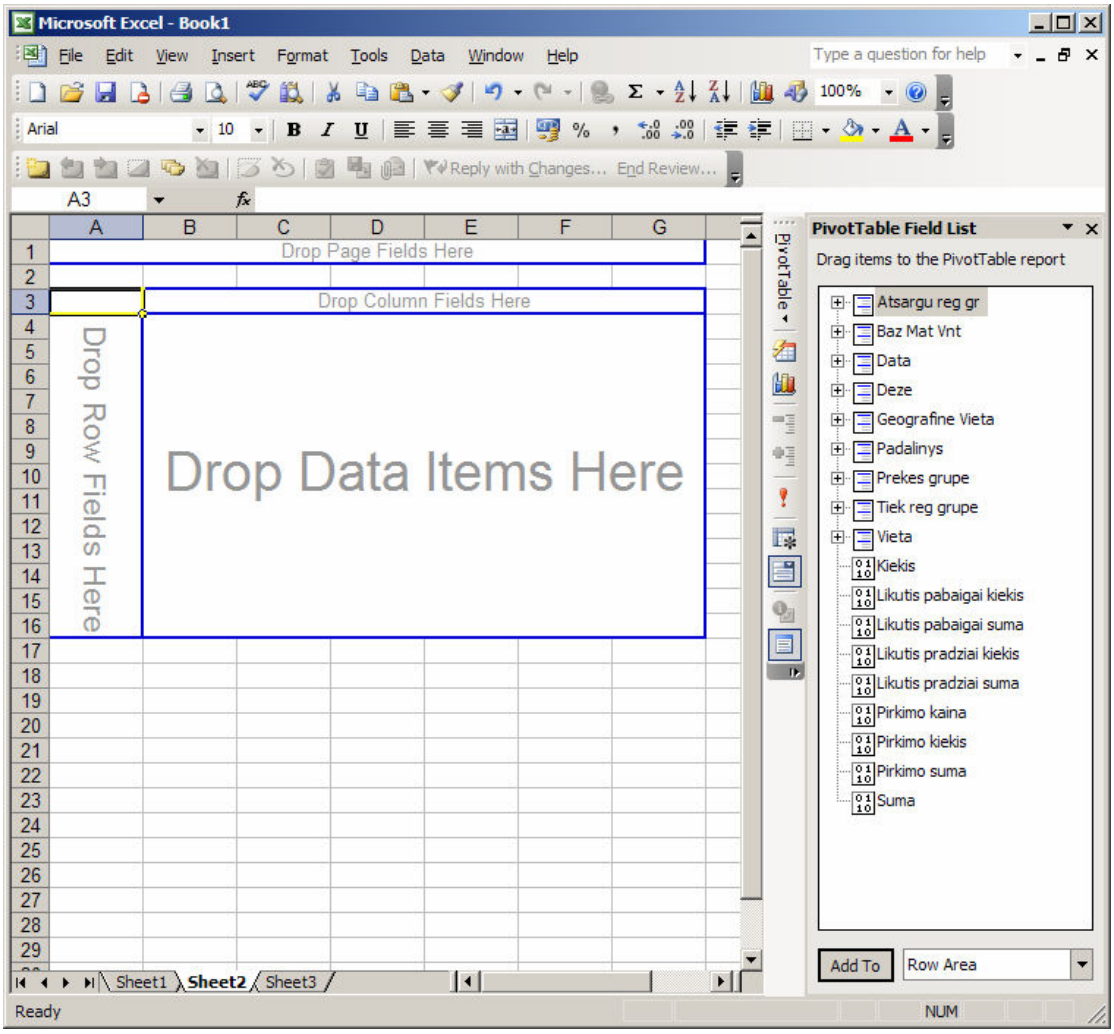
Atsiradusiame lange užpildome kubo pavadinimą, ir pasirenkame *Microsoft OLE DB Provider for Olap Services 8.0* tiekėją. Tai atlikus spaudžiame *Connect*.

3. Sekančiuose languose nurodome OLAP serverio adresą, bei pasirenkame duomenų bazę.
4. Paskutiniu žingsniu pasirenkame patį kubą:



5.7 pav. Pasirenkamas duomenų kubas

5. Viską teisingai atlikus grįžtame į *Microsoft Excel* programą, ir turime panašų vaizdą:



5.8 pav. Atidaryto kubo vaizdas

6. Toliau jau galime kurti reikalingas ataskaitas ar grafikus.

Keletas jau sukurtų ataskaitų pavyzdžių:

Prekes grupe	Maisto priedai				
Vieta	All Vieta				
Kiekis		Tiekeju Reg Grupe			
Metai	Ketvirtis	Menuo	LIETUVA	UZSIENIS	Grand Total
2003					
	Quarter 3				
		September	214864.96	117050.00	331914.96
	Quarter 3 Total		214864.96	117050.00	331914.96
	Quarter 4				
		October	198462.16	55519.00	253981.16
	Quarter 4 Total		198462.16	55519.00	253981.16
2003 Total			413327.12	172569.00	585896.12
Grand Total			413327.12	172569.00	585896.12

5.9 pav. Ataskaitos pavyzdys – maisto priedų pirkimas pagal datą iš Lietuvos ir užsienio tiekėjų

Geografine Vi	KP				
Vieta	All Vieta				
Kiekis		Ats Reg Gr			
Metai	Ketvirtis	Menuo	ATSARGINĖS	KURAS	MEDŽIAGOS Grand Total
2003					
	Quarter 3				
		July		134549.60	134549.60
		August		128985.57	128985.57
		September	4575.99	16.00	8594.96 13186.95
	Quarter 3 Total		4575.99	263551.17	8594.96 276722.12
	Quarter 4				
		October	4724.00	14.60	9365.42 14104.02
	Quarter 4 Total		4724.00	14.60	9365.42 14104.02
2003 Total			9299.99	263565.77	17960.38 290826.14
2004			305.00	238.00	127.00 670.00
Grand Total			9604.99	263803.77	18087.38 291496.14

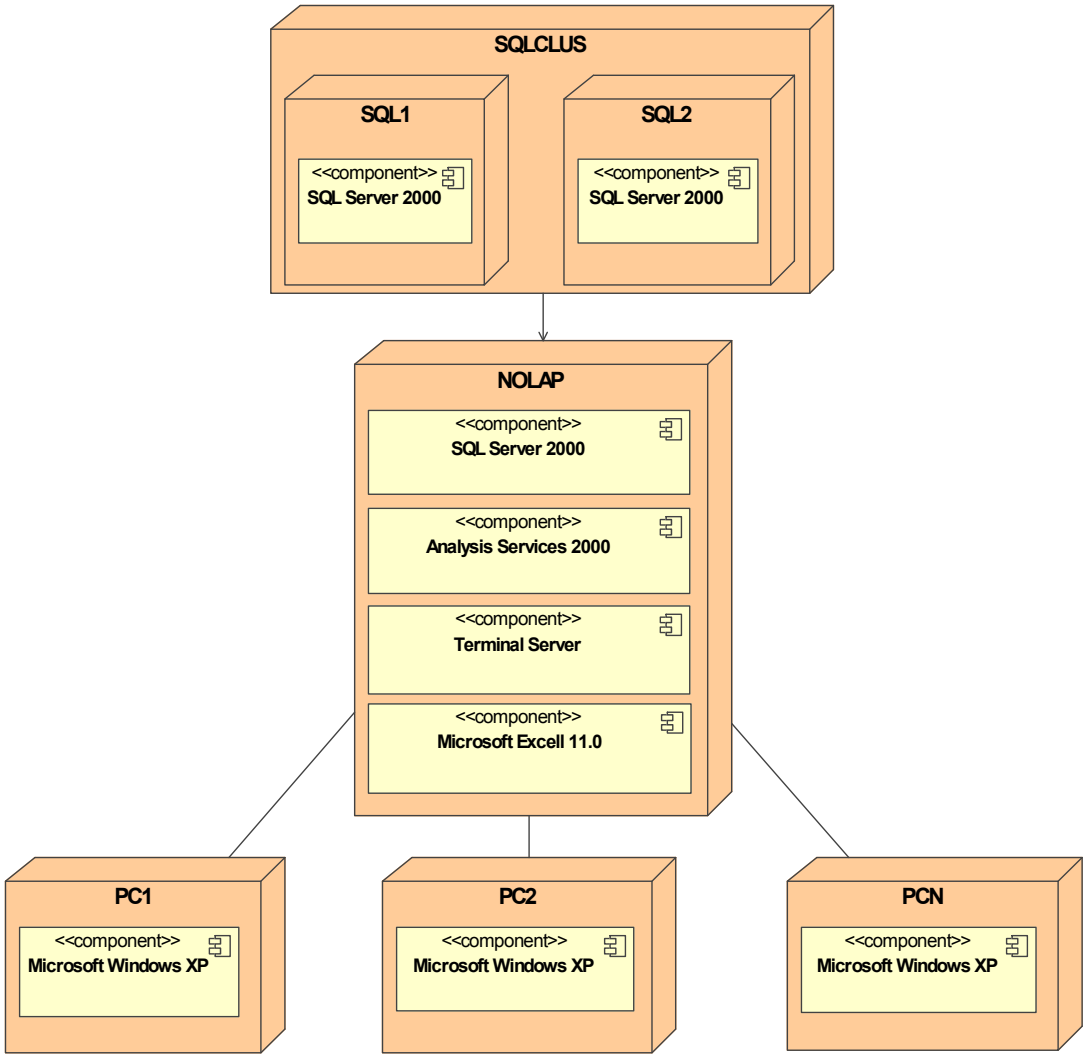
5.10 pav. Ataskaitos pavyzdys – prekių pirkimas Kauno sandėlyje pagal datą ir prekių grupes

Vieta	KAU Filialas "Kauno pienas"				
Metai	Ketvirtis	Likutis pradziai kiekis	Viso gautas kiekis	Viso sunaudotas kiekis	Likutis pabaigai kiekis
2003					
	Quarter 1		9.373.879,30	-444.958,00	8.928.921,30
	Quarter 2	8.928.921,30	12.794.442,00	-3.505.319,00	18.218.044,30
	Quarter 3	18.218.044,30	39.019.838,30	-48.650.552,60	8.587.330,00
	Quarter 4	8.587.330,00	4.773.891,00	-1.626.477,00	11.734.744,00
2003 Total			65.962.050,60	-54.227.306,60	11.734.744,00
2004					
	Quarter 1	11.734.744,00	2.489,10	-185,70	11.737.047,40
	Quarter 2	11.737.047,40	104.414,00	-44.435,52	11.797.025,88
	Quarter 3	11.797.025,88	2.253,75	-1.711,59	11.797.568,04
	Quarter 4	11.797.568,04	495,00	-353,80	11.797.709,24
2004 Total		11.734.744,00	109.651,85	-46.686,61	11.797.709,24
Grand Total			66.071.702,45	-54.273.993,21	11.797.709,24

5.11 pav. Ataskaitos pavyzdys – Kauno pieno prekių apyvarta pagal datą

6. DIEGIMAS

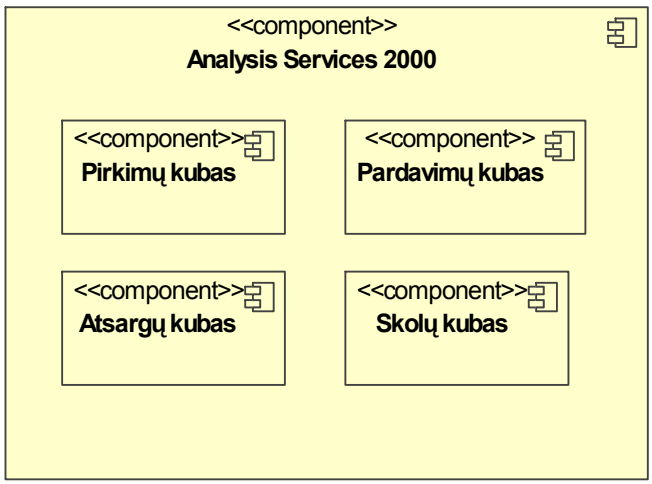
Apibrėšime, kokie pagrindiniai techniniai-programiniai komponentai sudarys būsimą sistemą, ir kokiais sąryšiais jie susiję.



6.1 pav. Projektuojamos sistemos komponentai.

Į sistemą įeina šie komponentai:

- *Microsoft SQL Server 2000* – naudojamas OLAP duomenų saugyklai. Į šią saugyklą perkeliami duomenys iš darbinės duomenų bazės, esančios SQLCLUS klasteryje.
- ***Analysis Services 2000* – naudoja duomenis esančius OLAP duomenų saugykloje, juos apdoroja ir pateikia vartotojui. Susideda iš šių sub-komponentų:**



6.2 pav. Analysis Services komponento sub-komponentai

- *Terminal Services* – įgalina vartotojus prie NOLAP serverio pasijungti per atstumą, ir naudotis jame esančiais resursais.
- *Microsoft Excel 11.0* – tarnauja kaip OLAP serverio vartotojo interfeisas. Leidžia vartotojams kurti sukinius (angl. *pivot tables*), diagramas, ir jų pagalba analizuoti OLAP serverio pateikiamus duomenis.
- *PC1..PCN* – vartotojų kompiuteriai. OLAP serverį galima pasiekti ir per tinklą – tai atliekama tiesiai iš vartotojų kompiuterių.

7. SISTEMOS KOKYBĖS TYRIMAS

Remiantis S. Mohanty [14], duomenų kokybė tai yra jų gebėjimas patenkinti verslo tikslus. Duomenų saugyklose, duomenų saugyklos kokybė ir efektyvumas dažnai tiesiogiai susiję su teikiamų duomenų kokybe. Norint šią kokybę užtikrinti, dažnai yra svarbu teisingai ir tiksliai išmatuoti įvairius rodiklius apibūdinančius saugyklos darbą bei duomenis.

Yra be galo daug kriterijų nusakančių duomenų saugyklos kokybę. Juos galima skirstyti į grupes. Pvz., nefunkcinių reikalavimų grupės:

- Našumas
- Saugumas
- Prieinamumas
- Patogumas

Daugelis šių kriterijų yra subjektyvūs, t.y. juos išmatuoti ir įvertinti yra neįmanoma ar bent jau labai sunku. Tokiais kriterijais remtis vertinant sistema yra ganėtinai rizikinga, todėl kad nežinoma kaip duomenų saugykla kinta laike (ar duomenys „gerėja“, ar „blogėja“? ar saugykla naudojama optimaliai?).

Norint objektyviai įvertinti sistemos kokybę, reikia remtis išmatuojamais kriterijais. Juos galima suskirstyti į du tipus:

1. Vidinės metrikos. Tai dažnai paprastai išmatuojami rodikliai: procesoriaus apkrovimas, diskų masyvo panaudojimas, laikinosios atminties rodikliai, sistemos prieinamumo statistika ir pan. Šie rodikliai dažniausiai padeda atsakyti ar dauguma nefunkcinių reikalavimų patenkinami, ar ne.
2. Išorinės metrikos. Tai panaudojimo iš vartotojo pusės statistika. Ji padeda atsakyti į klausimus kaip:
    - kas naudoja sistemą?
    - kada sistema naudojama?
    - kaip sistema naudojama?
    - kaip dažnai sistema naudojama?
    - ir pan.

Šiame darbe sukurtos sistemos kokybei įvertinti reikia:

1. išmatuoti naujosios sistemos darbo rodiklius
2. atsakyti ar buvo patenkinti visi nefunkciniai reikalavimai?

Rodiklių sistemos kokybės įvertinimui yra be galo daug, tačiau atsižvelgiant funkcinius ir nefunkcinius vartotojo reikalavimus buvo pasirinkti šie (Pateikiama kartu su įvertinimu. Įvertinimą atliko 6 skirtingų tipų sistemos vartotojai bei 2 sistemos administratoriai):

1. ETL etapas:

   a. Faktų lentelės generavimo trukmė trukmė. Vidutiniškai vieno kubo faktų lentelė generuojama 20 minučių. Visų kubų faktų lentelės sugeneruojamos per 80 minučių – toks rezultatas užsakovą pilnai tenkina.

   b. **Dimensijų lentelių perkėlimo trukmė** – visos kubams reikalingos dimensijų lentelės perkeliamos per 3 minutes.

   c. **ETL etapo įtaka pagrindinės sistemos procesorių darbui** – ETL etapo metu pagrindinės sistemos procesorių apkrovimas padidėja 26%. Rodiklis vertinamas labai gerai.

   d. **ETL etapo įtaka pagrindinės sistemos diskų masyvui** – ETL etapo metu pagrindinės sistemos diskų masyvų apkrova padidėja 48%. Tai priimtinas rodiklis.

   e. **ETL etapo įtaka pagrindinėje sistemoje vykstantiems procesams** – nakties metu suplanuoti darbai ETL etapo vykdymo metu vykdomi vidutiniškai 20% ilgiau. Rodiklis vertinamas labai gerai.

2. Kubų atnaujinimo etapas:

   a. Etapo vykdymo trukmė – visų kubų atnaujinimas trunka vidutiniškai 2 valandas 30 minučių.

3. Kubų naudojimo įvertinimas:
   a. Našumas

      i. Tipinės ataskaitos generavimo greitis – viena ataskaita vidutiniškai generuojama 20 sekundžių. Rodiklis vertinamas gerai.

      ii. Procesoriaus apkrovimas dirbant 10 vartotojų – OLAP serverio procesoriaus apkrovimas vidutiniškai 80%. Rodiklis vertinamas teigiamai.

      iii. Diskų apkrovimas dirbant 10 vartotojų – diskų masyvas apkraunamas vidutiniškai 12%. Rodiklis vertinamas labai gerai.

   b. Tinkamumas

      i. Klaidų duomenyse skaičius mėnesio bėgyje. Mėnesio bėgyje pastebėtos 6 klaidos duomenyse. 4 iš jų buvo dėl klaidingų duomenų darbinėje sistemoje. Rodiklis vertinamas gerai.

      ii. Situacijų, kuomet negalima atlikti norimos analizės, skaičius mėnesio bėgyje. Atvejų nepastebėta – sistema pilnai atitinka vartotojų poreikius.

   c. Prieinamumas

i. **Sistemos darbo sutrikimų skaičius mėnesio bėgyje**. Pastebėtas vienas sutrikimas, kuris įvyko dėl techninės įrangos.

ii. **Vidutinė sutrikimo trukmė –** 8 valandos.

d. Saugumas. Rodiklis įvertintas labai gerai – visos vartotojų teisės atitinka numatytas.

e. Patogumas. Rodiklis įvertintas patenkinamai – pasirinktas OLAP klientas (Microsoft Excel) pasirodė nesantis labai patogus įrankis, todėl ateityje numatyta ieškoti galingesnių.

4. Sistemos palaikymo įvertinimas:

a. Atnaujinimų nustatymo patogumas. Atnaujinimų nustatymo lankstumą ir patogumą sistemos administratoriai įvertino gerai.

b. Integracija su esama sistema – naujoji sistema pilnai integravosi su esančia, vartotojų teisių nustatymai su esančiais tinklo vartotojais.

c. Vidutinė klaidų ištaisymo trukmė – 2 valandos. Įvertinta gerai.

8. PROJEKTO REZULTATŲ APIBENDRINIMAS BEI ŽVILGSNIS Į ATEITĮ

Sistemos kūrimo projekto vykdyme galima išskirti du svarbius etapus: sukurtos sistemos optimizavimą, ir baigiamąjį sistemos kokybės tyrimą. Glaustai pateikiame šių etapų rezultatus.

Sistemos optimizavimo etapas:

8.1 lentelė. Optimizavimo etapo rezultatai

Žingsnis	Pasiekti rezultatai
Dimensijų iškėlimas į atskirus rodinius	Sutaupytos 45 minutės atnaujinimo laiko
Dimensijų grupavimas	Sutaupyta 11 minučių atnaujinimo laiko
Particijų naudojimas	Sutaupomi resursai išvengiant nesikeičiančių duomenų atnaujinimo
Faktų lentelės perkėlimas į OLAP serverį	Sutaupytos 49 minutės atnaujinimo laiko
Papildomi indeksai faktų lentelėje	Sutaupyta 15 minučių atnaujinimo laiko
Viso	Sutaupyta 120 minučių (82%) viso naujinimo laiko. Sutaupytas laikas tiesiogiai proporcingas sumažėjusiam darbinės sistemos apkrovimui

Sistemos kokybės įvertinimas:

8.2 lentelė. Kokybes įvertinimo apibendrinimas

Kriterijus	Įvertinimas
ETL etapas	
Faktų lentelės generavimo trukmė	Labai geras
Dimensijų lentelių perkėlimo trukmė	Puikus
ETL etapo įtaka pagrindinės sistemos procesorių darbui	Labai geras
ETL etapo įtaka pagrindinės sistemos diskų masyvui	Priimtinas
ETL etapo įtaka pagrindinėje sistemoje vykstantiems procesams	Labai geras
Kubų atnaujinimo etapas	
Etapo vykdymo trukmė	Labai geras
Kubų naudojimo įvertinimas	
Našumas	
Tipinės ataskaitos generavimo greitis	Geras
Procesoriaus apkrovimas	Teigiamas
Diskų apkrovimas	Labai geras
Tinkamumas	
Klaidų skaičius	Geras
Kritinių situacijų skaičius	Puikus
Prieinamumas	

Darbo sutrikimų skaičius	Puikus
Vidutinė sutrikimo trukmė	8 valandos
Saugumas	Labai geras
Patogumas	Patenkinamas
Sistemos palaikymo įvertinimas	
Atnaujinimo nustatymai	Geras
Integracija su esama sistema	Puikus
Vidutinė klaidų ištaisymo trukmė	Geras

Apibendrinus visus įvertinimus, galima teigti jog naujoji sistema pilnai pateisino vartotojų lūkesčius. Esamos darbinės sistemos apkrovimas buvo ženkliai sumažintas visiškai atsisakius Navision Attain sistemos duomenų analizės priemonių. Rastų klaidų skaičius yra mažas, o pačios klaidos smulkios bei greitai ištaisytos.

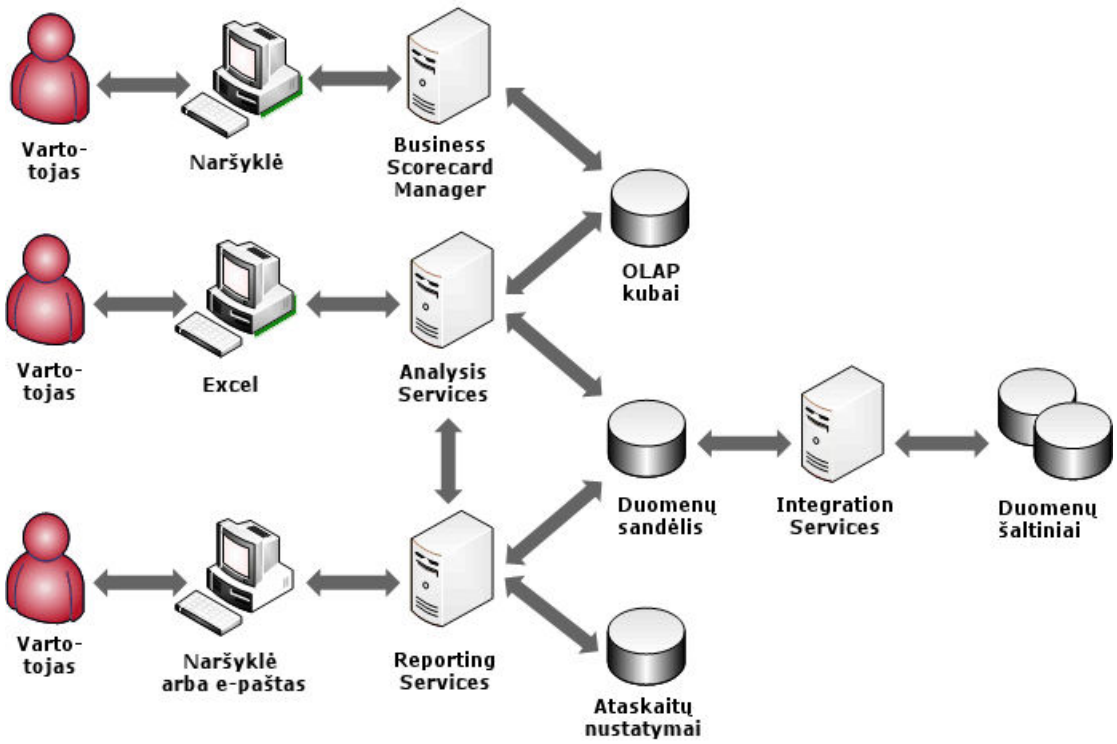
Žvelgiant į sistemos tobulinimą bei panaudojimą ateityje, reikėtų atkreipti dėmesį į plačius Microsoft verslo analizės sistemų planus.

2005 metų pabaigoje, Microsoft išleidžia Microsoft SQL Server 2005 platformą. Šiame produkte visiškai perkurta verslo informacijos sritis – DTS (angl. *Data Transformation Services*) perdaryta į *Integration Services*, išplėstas *Reporting Services* komponentas. Stambiausi pakeitimai:

1. *Analysis Services* buvo pagerintas našumas, didelių duomenų sandėlių palaikymas. Įgyvendintas išankstinis duomenų užkrovimas (angl. *Pro-Active Caching*) – duomenų kubas naujinas save savaime, fone, vartotojui nepastebint. Priklausomai nuo poreikio, *Analysis Services* automatiškai skaičiuos reikalingas agregacijas, užkraus duomenis. Tai padės išvengti periodinio duomenų sandėlių administratorių darbo – nebereikės kas kartą pasikeitus vartotojų poreikiams skaičiuoti naujų agregacijų, naujinti duomenų, dimensijų.
2. Jau egzistuojančią duomenų kubų užklausų kalbą MDX papildys XML/A (angl. *XML for Analysis*). Tai XML pagrįstas MDX plėtinys, palaikomas ne tik Microsoft, bet ir SAP bei Hyperion. XML/A bus SOAP tinklo servisas, o tai reiškia kad kubų duomenis bus galima pasiekti iš bet kur, naudojant HTTP protokolą.
3. *Reporting Services* teiks galimybę vartotojams patiems kurti jiems reikalingas ataskaitas, ir visa tai daryti iš jau paruoštų duomenų analizei duomenų sandėlių. Tai leis dar labiau atskirti darbinę verslo valdymo sistemą nuo duomenų analizės priemonių – taip taupant resursus bei laiką. Patogios ataskaitų kūrimo priemonės leis sumažinti duomenų analizės priemonių kūrimo sąnaudas.
4. *Integration Services* bus žymiai galingesnė priemonė duomenų perkėlimui, transformacijai atlikti. Pats procesas bus vizualizuotas, vartotojas tarytum nurodys duomenų išgavimo kelią, o *Integration Services* atliks visą likusį darbą.

5. *Business Scorecard Manager* tai dar vienas duomenų analizės produktas, skirtas aprašyti įmonės veiklą apibūdinančius rodiklius, bei juos patogiai stebėti.

Vizualiai naująją Microsoft verslo informacijos (angl. *Business Intelligence*) produktų platformą galima pamatyti žemiau pateiktame paveiksle.



8.1 pav. Microsoft verslo informacijos platforma

Išnagrinėjus Microsoft ateities planus verslo informacijos srityje, akivaizdu, jog jau dabar reikia pradėti investuoti į verslo analizės produktus. Ypač į šias naujoves derėtų atsižvelgti įmonėms jau turinčioms Microsoft produktų bazę.

IŠVADOS

1. Šiame darbe sprendžiama verslo duomenų analizės įgyvendinimo problema – konkrečios įmonės duomenų analizės procesai, atliekami jų naudojamos verslo valdymo sistemos Navision Atain priemonėmis, generuoja ypač didelius duomenų srautus, tos priemonės yra nelanksčios, per daug apkrauna verslo valdymo sistemą.

2. Siekiant pilno integravimo su esama technine baze bei minimalių finansinių sąnaudų, iš galimų sprendimo variantų pasirinktas labiausiai vartotojo poreikius tenkinantis sprendimas - Microsoft SQL Server siūlomos duomenų analizės (OLAP) priemonės.

3. Atlikta įmonės verslo procesų analizė, kurios metu identifikuoti duomenų analizės vartotojai ir jiems reikalingi keturi duomenų analizės kubai (pirkimų, pardavimų, atsargų bei pirkėjų skolų).

4. Kuriamiesiems duomenų kubams pasirinkta MOLAP tipo duomenų saugykla – tokio tipo saugykla užtikrina minimalų verslo valdymo sistemos apkrovimą. Kadangi tiek duomenys, tiek agregacijos yra pilnai atskirtos nuo verslo valdymo sistemos, tad į ją kreipiamasi tiktai kubų naujinimo metu. Kubai atnaujinami naktį, todėl vartotojai beveik nepajunta darbo našumo sumažėjimo. Pasirinktos snaigės tipo kubų schemos – kiekvieno kubo faktinius duomenis apibūdinančios dimensijos perkeliamos į atskiras lenteles, kas labiausiai atitinka esamą duomenų struktūrą.

5. Darbe buvo sukurta konkrečios įmonės verslo analizės sistema, sudaryta iš keturių kubų. Eksperimentinio tyrimo metu ji buvo optimizuota norint sutrumpinti duomenų kubų atnaujinimo ir užklausų generavimo trukmes. Optimizavimas atliktas keletu etapų: optimizuotos kubų schemos (dimensijos iškeltos į atskirus rodinius, jos sugrupuotos), optimizuotas duomenų sandėlis (kubai išskaidyti į atskirus skirsnius) bei techninė įranga (faktų lentelės pilnai perkeltos į OLAP serverį, joms sukurti papildomi indeksai). Optimizavimo metu buvo išplėstos iš literatūros gautos žinios, praktikoje pritaikyti teoriniai samprotavimai, sukurtos rekomendacijos naujų sistemų diegimui.

6. Sukurtas sprendimas yra universalus – jis gali būti pritaikomas įvairiems verslo valdymo sistemos Navision Attain diegimo variantams (tiek naudojant gimtąją bazę, tiek Microsoft SQL Server), suderinamas su ateinančiomis Navision versijomis. Kuriant sistemą stengtasi kiek įmanoma atsiriboti nuo konkretaus užsakovo specifinio funkcionalumo, taip palengvinant ateities diegimus.

7. Sukurtas verslo valdymo sistemos Navision Attain bei Microsoft SQL Server duomenų analizės priemonių integravimo metodas, kurio svarbiausias akcentas yra dažniausiai pasitaikančių klaidų išvengimas bei optimalus sistemos darbas, taupant pagrindinės sistemos resursus. Dalis šio metodo principų buvo pristatyti konferencijoje.

8. Kuriant sprendimą, stengtasi orientuotis į verslo valdymo sistemos Navision Attain specifiką – duomenų struktūras, jų išgavimo ypatybes (faktų lentelių, sumų saugojimas, *flowfield* ir *flowfilter*

naudojimas), tačiau daugumą eksperimentinio tyrimo metu atlikto optimizavimo metodų galima taikyti ir kitoms verslo valdymo sistemoms.

9. Gauta patirtis gali būti ir bus panaudota ateityje. Išnagrinėjus Microsoft verslo analizės planus, pastebėta ryški tendencija plėsti šią sferą, todėl panašaus pobūdžio metodika yra aktuali. Planuojama įgytas žinias plėsti tolimesnių diegimų metu, pritaikyti jas naujoms technologijoms.

LITERATŪROS SĄRAŠAS

[1] BAIN T., BENKOVICH M. ir kiti. *Professional SQL Server 2000 Data Warehousing with Analysis Services*“, 2nd. ed. Wrox Press, 2001.

[2] THOMSON E.. *OLAP Solutions. Building Multidimensional Information Systems*, 2nd. ed. John Wiley & Sons, 2002.

[3] MICROSOFT. *Designing and Implementing OLAP Solutions with Microsoft SQL Server 2000.* 2000.

[4] PAUL S., GUATAM N., BALINT R.. *Preparing and Mining Data with Microsoft SQL Server 2000 and Analysis Services.* Microsoft, 2002.

[5] JACOBSON R.. *Microsoft SQL Server 2000 Analysis Services Step by Step.* Microsoft, 2000.

[6] MICROSOFT. *Delivery Guide Populating a Data Warehouse with Microsoft® SQL Server™ 2000 Data Transformation Services.* 2001.

[7] GOIL S., CHOUDHARY A.. *High Performance Multidimensional Analysis and Data Mining*.

[8] TONKŪNAITĖ J., NEMURAITĖ L.. *Duomenų saugyklos projektavimo metodika*.

[9] TONKŪNAITĖ J., NEMURAITĖ L.. *Duomenų saugyklos projektavimo proceso modelis*.

[10] MOODY D., KORTINK M., *From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design // Proceedings of the International Workshop on Design and Management of Data Warehouses.* Stockholm, Sweden: 2000. Prieiga per internetą: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-28/paper5.pdf

[11] VASSILEV H.. *ETL: Efficiency by Design.* Caribou Lake Inc. Prieiga per internetą: http://datawarehouse.ittoolbox.com/browse.asp?c=DWPeerPublishing&r=http%3A%2F%2Fhosteddocs%2Eittoolbox%2Ecom%2FHV051305%2Epdf

[12] Pervasive Software Inc. *ETL – The Secret Weapon in Data Warehousing and Business Intelligence*. 2003. Preiga per internetą http://wp.bitpipe.com/resource/org_1013484794_935/ETLTheSecretWeapon_bitpipe.pdf

[13] Microsoft Corp. *Microsoft SQL Server 2000 Analysis Services Performance Guide*. 2003.

[14] MOHANTY S.. *Beyond Monitoring - Measuring the Effectiveness of a Data Warehouse.* 2004.

[15] META DATA COALITION. *Open Information Model v1.1.* 1999.

[16] KEPEŽINSKAS A., NEMURAITĖ L.. *Verslo valdymo sistemos Navision Attain ir OLAP duomenų analizės integracija, Informacinės Technologijos ‘2005.* 2005.

TERMINAI

DTS – data transformation services, duomenų transformavimo įrankiai

Navision Attain – Microsoft Business Solutions verslo valdymo sistema

ETL – extract, transform and load – duomenų išgavimas, transformavimas ir išsaugojimas

aggregations – pirminiai apskaičiavimai, pagreitinantys OLAP užklausas

usage based optimisation – užklausų optimizavimo priemonė, naudojanti užklausų statistiką

Windows Active Directory – Windows tinklų vartotojų autentifikavimo paketas

OLAP – On-Line Analytical Processing – priemonė greitai ir patogiai išgauti reikiamus duomenis įvairiais pjūviais

shared – bendrai naudojama – dimensijų atveju, ši dimensija gali būti naudojama keliuose skirtinguose kubuose

Analysis Services – Microsoft SQL Server 2000 platformos OLAP komponentas

DBVS – duomenų bazės valdymo sistema

OIM – Open Information Model, metaduomenų modelis

CWM – Common Warehouse Model, metaduomenų modelis

DSO – Decision Support Objects, programinė sąsaja metaduomenims iš *Analysis Services* paimti

SANTRAUKA ANGLŲ KALBA

ENTERPRISE RESOURCE PLANNING SOFTWARE NAVISION ATTAIN DATA ANALYSIS USING OLAP TOOLS

This work investigates the problem of company not able to handle it's data analysis using Navision Attain tools alone. A more powerful system is needed, and Microsoft SQL Server OLAP tools are selected as such. The work carried out covers data extraction, transformation and loading (ETL) for analyzing Navision Attain data using OLAP data analysis tools. Different integration architectures, as well as needed transformations and most often encountered problems are covered, together with possible solutions to them.. After completing business process analysis, and identifying user roles and required data, four different data cube were created on data from Navision Attain. Namely cubes for sales analysis, purchase analysis, item inventory tracking and customer debt analysis. In addition, an example schedules for cube updating are created, and wider usage guidelines are provided. The complete system is then optimised to provide efficient performance and low main system load. The optimisation results are compiled into generic suggestions for further analysis system development.

1. PRIEDAS. TECHNINĖ UŽDUOTIS

Tvirtinu … … … …
Vedėjas … … … …
Suderinta … … … …
Vadovas … … … …
Data … … … …

TECHNINĖ UŽDUOTIS

1. TEMA:
   Verslo valdymo sistemos Navision Attain ir OLAP duomenų analizės integravimas
2. ANALITINIS IR TIRIAMASIS DARBAS:
   2.1. Užsakovo įmonėje vykdomų procesų analizė.
   2.2. Vartotojų poreikių duomenų analizei analizė.
   2.3. Integracijos poreikių analizė.
   2.4. Nefunkcinių reikalavimų analizė.
3. SUPROJEKTUOTI, REALIZUOTI IR PARUOŠTI VARTOJIMUI OLAP DUOMENŲ ANALIZĖS KUBUS:
   3.1. Pirkimų kubą
   3.2. Pardavimų kubą
   3.3. Atsargų kubą
   3.4. Skolų kubą
4. PARUOŠTI SISTEMOS PANAUDOJIMO DOKUMENTUS:
   4.1. Trumpą vartotojo vadovą
   4.2. Trumpa atnaujinimų nustatymo vadovą
   4.3. Ataskaitų pavyzdžius
5. REIKALAVIMAI PROJEKTAVIMUI, PROGRAMINEI IR TECHNINEI ĮRANGAI:
   5.1. Esama duomenų bazė realizuota Microsoft SQL Server 2000 programine įranga
   5.2. Kubus realizuoti Microsoft SQL Server 2000 Analysis Services priemonėmis
   5.3. Nauja sistema turi lengvai integruotis į esamą įmonės infrastruktūrą: veikti esamuose vartotojų kompiuteriuose, naudoti esamą tinklo apsaugą (Windows Active Directory)
6. REIKALAVIMAI DARBO PRISTATYMUI:
   6.1. Pateikti spausdintą ataskaitos variantą
   6.2. Paruošti skaidres
   6.3. Pateikti sistemos kopiją skaitmeniniame formate

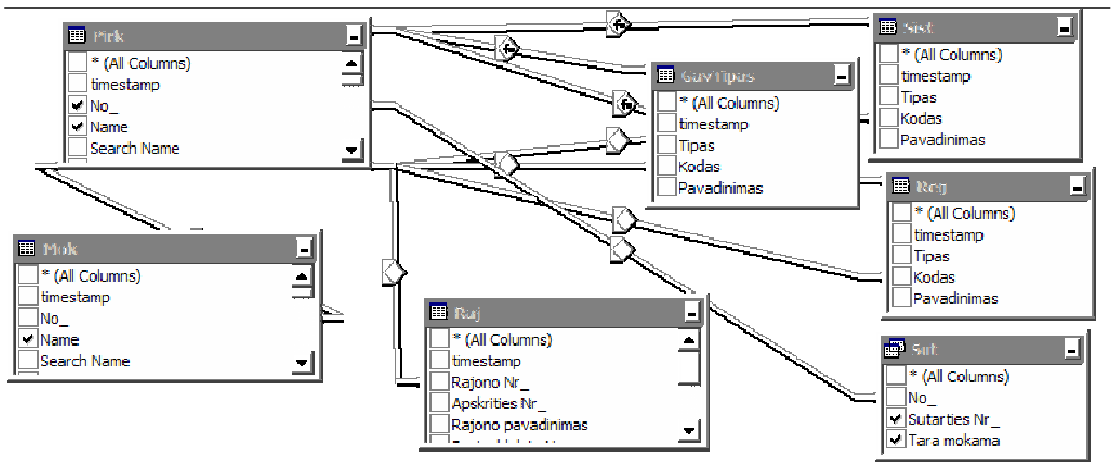
IFM-0/4 grupės studentas
Algirdas Kepežinskas

2. PRIEDAS. RODINIAI

Pateikiamas svarbesnių rodinių užklausų tekstas, ir, kur įmanoma, schemos.

1. PIRKĖJAS

```
1.    SELECT
2.        Pirk.No_ AS PirkejoKodas,
3.        Pirk.Name AS PirkejoPav,
4.        Pirk.No_ + ' ' + Pirk.Name AS Pirkejas,
5.        Pirk.[Registration No_] AS PirkRegNr,
6.        Pirk.[Bill-TO Customer No_] AS MoketojoKodas,
7.        Pirk.[Salesperson Code] AS PirkAgentas,
8.        Pirk.[Beviltiškas mokėtojas] AS [Beviltiskas moketojas],
9.        Sut.[Sutarties Nr_],
10.       Sut.[Tara mokama],
11.       Mok.Name AS MoketojoPav,
12.       Mok.No_ + ' ' + Mok.Name + ' ' + Mok.[Name 2] AS Moketojas,
13.       Pirk.[Rajono Nr_] + ' ' + Raj.[Rajono pavadinimas] AS Rajonas,
14.       Reg.Kodas + ' ' + Reg.Pavadinimas AS Regionas,
15.       GavTipas.Kodas + ' ' + GavTipas.Pavadinimas AS GavejoTipas,
16.       Sist.Kodas + ' ' + Sist.Pavadinimas AS Sistema,
17.       Pirk.[Sistemos klasĖs kodas] AS SistemosKlase,
18.       Pirk.[Sistemos lygio kodas] AS SistemosLygis,
19.       Pirk.[Sistemos 4-AS lygis] AS Sistemos4Lygis,
20.       Pirk.[Country Code] AS PirkejoSalis,
21.       Pirk.[Customer Posting GROUP] AS PirkejoRegGr
22.   FROM          dbo.[CRONUS$Customer] Pirk
23.   LEFT OUTER JOIN
24.       dbo.[CRONUS$Grupių sąrašai] Sist
25.       ON Pirk.[Sistemos kodas] = Sist.Kodas
26.       AND Sist.Tipas = 13
27.   LEFT OUTER JOIN
28.       dbo.[CRONUS$Grupių sąrašai] GavTipas
29.       ON Pirk.[Gavėjo tipas] = GavTipas.Kodas
30.       AND GavTipas.Tipas = 20
31.   LEFT OUTER JOIN
32.       dbo.[CRONUS$Grupių sąrašai] Reg
33.       ON Pirk.[Regiono kodas] = Reg.Kodas
34.       AND Reg.Tipas = 21
35.   LEFT OUTER JOIN
36.       dbo.[CRONUS$Rajonas] Raj
37.       ON Pirk.[Rajono Nr_] = Raj.[Rajono Nr_]
38.   LEFT OUTER JOIN
39.       dbo.[Pirkėjų sutartys] Sut
40.       ON Pirk.No_ = Sut.No_
41.   LEFT OUTER JOIN
42.       dbo.[CRONUS$Customer] Mok
43.       ON Pirk.[Bill-TO Customer No_] = Mok.No_
```



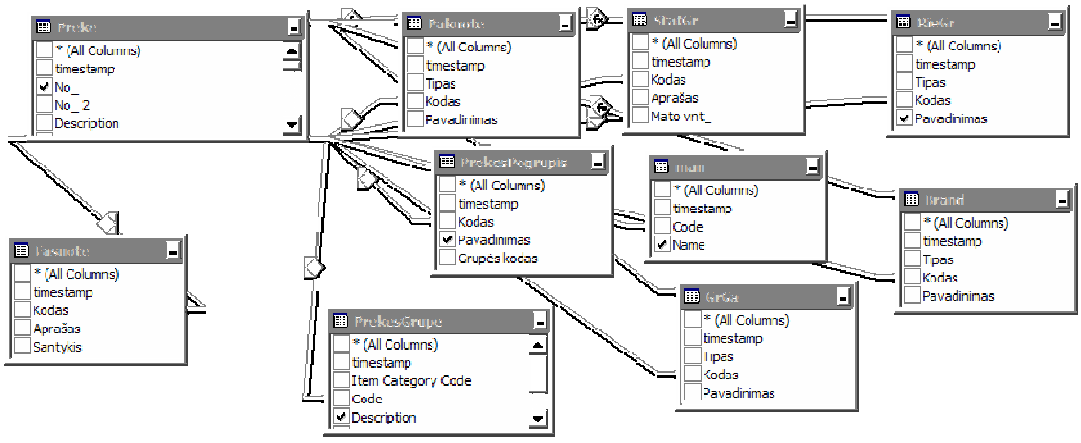
2. PIRKĖJŲ SUTARTYS

```
CREATE
    VIEW dbo.[Pirkėjų sutartys] AS
SELECT
    TOP 1 c.[No_],
    s.[Sutarties Nr_],
    CASE
        WHEN s.[Tara mokama] = 1
        THEN 'Taip'
        ELSE 'Ne'
    END AS [Tara mokama]
FROM [CRONUS$Customer] c
LEFT OUTER JOIN
    [CRONUS$Pirkėjų sutartys] s
    ON c.[No_] = s.[Pirkėjo Nr_]
    AND
    (
        (
            s.[Sutarties galiojimo pradþia] = { d '1753-01-01' }
        )
        OR
        (
            s.[Sutarties galiojimo pradþia] <= getdate()
        )
    )
    AND
    (
        (
            s.[Sutarties galiojimo pabaiga] = { d '1753-01-01' }
        )
        OR
        (
            s.[Sutarties galiojimo pabaiga] >= getdate()
        )
    )
```

3. PREKĖ

```
SELECT DISTINCT
    Preke.No_ AS PrekesNr,
    Preke.Description + ' ' + Preke.[Description 2] AS PrekesPav,
    Preke.No_ + ' ' + Preke.Description + ' ' + Preke.[Description 2] AS Preke,
    Preke.[Inventory Posting GROUP] AS AtsRegGr,
    PrekesPogrupis.Pavadinimas AS PrekesPogr,
    PrekesGrupe.Description AS PrekesGr,
    Preke.[Base Unit of Measure] AS BazMatVnt,
    Preke.[Gen_ Prod_ Posting GROUP] AS BPRG,
    Pakuote.Kodas + ' ' + Pakuote.Pavadinimas AS Pakuote,
    Fasuote.Kodas + ' ' + Fasuote.Aprašas AS Fasuote,
    StatGr.Kodas + ' ' + StatGr.Aprašas AS StatistGr,
    GrGa.Kodas + ' ' + GrGa.Pavadinimas AS GrupGam,
    RieGr.Pavadinimas AS Riebumas,
    Brand.Kodas + ' ' + Brand.Pavadinimas AS Brand,
    man.Name AS Gamintojas
FROM dbo.[CRONUS$Item] Preke
LEFT OUTER JOIN
    dbo.[CRONUS$Prekių pogrupiai] PrekesPogrupis
    ON PrekesPogrupis.Kodas = Preke.[Prekės pogrupis]
LEFT OUTER JOIN
    dbo.[CRONUS$Product GROUP] PrekesGrupe
    ON Preke.[Product
GROUP Code] = PrekesGrupe.Code
LEFT OUTER JOIN
    dbo.[CRONUS$Grupių sąrašai] Pakuote
    ON Pakuote.Tipas = 8
    AND Pakuote.Kodas = Preke.PakuotĖ
LEFT OUTER JOIN
    dbo.[CRONUS$Fasuotės] Fasuote
    ON Preke.FasuotĖ = Fasuote.Kodas
LEFT OUTER JOIN
    dbo.[CRONUS$Statistinės grupės] StatGr
    ON Preke.[Statistinė grupė] = StatGr.Kodas
LEFT OUTER JOIN
    dbo.[CRONUS$Grupių sąrašai] GrGa
    ON GrGa.Tipas = 3
    AND GrGa.Kodas = Preke.[GrupĖ gamybai]
```

```
39.   LEFT OUTER JOIN
40.       dbo.[CRONUS$Grupių sąrašai] RieGr
41.       ON RieGr.Tipas = 5
42.       AND RieGr.Kodas = Preke.[Riebumo grupĖ]
43.   LEFT OUTER JOIN
44.       dbo.[CRONUS$Grupių sąrašai] Brand
45.       ON Brand.Tipas = 10
46.       AND Brand.Kodas = Preke.Brand
47.   LEFT OUTER JOIN
48.       dbo.[CRONUS$Manufacturer] man
49.       ON man.Code = Preke.[Manufacturer Code]
```



4. PIRKIMAI - PAGRINDINIS

```
1.    SELECT DISTINCT
2.        PrekesIrasai.[Entry No_] AS IrasoNr,
3.        PrekesIrasai.[Posting Date] AS DATA,
4.        PrekesIrasai.[Item No_] AS PrekesNr,
5.        Tiekejas.[No_] + ' ' + Tiekejas.Name AS Tiekejas,
6.        Tiekejas.[Vendor Posting GROUP] AS TiekejuRegGrupe,
7.        Dezes.Code + ' ' + Dezes.Description AS Deze,
8.        PrekesIrasai.[Location Code] AS VietosKodas,
9.        PrekesIrasai.[Global Dimension 1 Code] AS GeografineVieta,
10.       PrekesIrasai.[Global Dimension 2 Code] AS Padalinys,
11.       PrekesIrasai.[Lot No_] AS PartijosNumeris,
12.       PrekesIrasai.[Document No_] AS DokumentoNr,
13.       CASE
14.           WHEN PrekesIrasai.[Entry Type] = 0
15.           THEN VertesIrasai.SavikSuma
16.       END AS PirkimoSuma,
17.       CASE
18.           WHEN PrekesIrasai.[Entry Type] = 0
19.           THEN Quantity
20.       END AS PirkimoKiekis,
21.       PrekesIrasai.[Bendr_ verslo reg_ grupė] AS BendraVersloRegGrupe,
22.       PrekesIrasai.[Bendr_ prekės reg_ grupė] AS BendraPrekesRegGrupe,
23.       PrekesIrasai.Quantity AS Kiekis,
24.       VertesIrasai.SavikSuma AS Suma,
25.       PrekesIrasai.Description AS Aprasas
26.   FROM dbo.[CRONUS$Item Ledger Entry] PrekesIrasai
27.   INNER JOIN
28.       dbo.[Vertės įrašai] VertesIrasai
29.       ON PrekesIrasai.[Entry No_] = VertesIrasai.[Item Ledger Entry No_]
30.   LEFT OUTER JOIN
31.       dbo.[CRONUS$Vendor] Tiekejas
32.       ON Tiekejas.[No_] = PrekesIrasai.[Source No_]
33.   LEFT OUTER JOIN
34.       dbo.[CRONUS$Bin] Dezes
35.       ON PrekesIrasai.[Bin Code] = Dezes.Code
36.       AND PrekesIrasai.[Location Code] = Dezes.[Location Code]
37.   WHERE
38.       (
39.           PrekesIrasai.[Completely Invoiced] = '1'
40.       )
```

```
41.       AND
42.       (
43.           PrekesIrasai.[Entry Type] = 0
44.       )
```

5. PARDAVIMAI - PAGRINDINIS

```
1.    SELECT
2.        pki.[Source No_] AS PirkejoNr,
3.        pki.[Item No_] AS PrekesNr,
4.        pki.Quantity * - 1 AS Kiekis,
5.        pki.[Svoris neto]*ABS(pki.Quantity)/pki.Quantity * - 1 AS [Svoris neto],
6.        pki.[Taros kiekis],
7.        ValueEntry.PardSuma,
8.        ValueEntry.NuolSuma,
9.        pki.[Posting Date] AS DATA,
10.       pki.[Bendrasis svoris]*ABS(pki.Quantity) / pki.Quantity * - 1 AS Svoris,
11.       pki.[Vadybininko kodas] AS Vadybininkas,
12.       CASE
13.           WHEN ValueEntry.PardSuma > 0
14.           THEN 'Pardavimas'
15.           WHEN ValueEntry.PardSuma < 0
16.           THEN 'Grazinimas'
17.       END AS OperacijosTipas,
18.       pki.[Global Dimension 1 Code] AS [Geografine vieta],
19.       pki.[Global Dimension 2 Code] AS [Filialas],
20.       [RETURN Reason Code] AS [GrazPriezKod],
21.       CASE
22.           WHEN pki.[Entry Type] = 1
23.           THEN 'Pardavimas'
24.           WHEN pki.[Entry Type] = 4
25.           THEN 'Perdavimas'
26.       END AS [?ra?o tipas],
27.       pki.[Location Code] AS [Vietos kodas],
28.       pki.[Document Date] AS [DokData],
29.       pki.[Document No_] AS [DokNo],
30.       pki.[External Document No_] AS [IsDokNo],
31.       pki.Quantity * - 1 * Item.[Gross Weight] AS Svoris2,
32.       'PZ' AS [Duomenu saltinis],
33.       pki.Quantity * - 1 * Item.[Net Weight] AS Svoris3,
34.       [Entry No_] AS IrasoNr,
35.       CASE
36.           WHEN pki.Quantity > 0
37.           THEN pki.[Bazin? kaina] * - 1
38.           WHEN pki.Quantity < 0
39.           THEN pki.[Bazin? kaina]
40.       END AS BazineSuma,
41.       ValueEntry.PardSuma * 0.18 AS PVMSuma,
42.       ValueEntry.PardSuma * 1.18 AS SumaSuPVM
43.   FROM dbo.[CRONUS$Item Ledger Entry] pki
44.   INNER JOIN
45.       dbo.[Vertės įrašai] ValueEntry
46.       ON pki.[Entry No_] = ValueEntry.[Item Ledger Entry No_]
47.   INNER JOIN
48.       dbo.[CRONUS$Item] Item
49.       ON pki.[Item No_] = Item.[No_]
50.   WHERE
51.       (
52.           (
53.               pki.[Entry Type] = 1
54.           )
55.           OR
56.           (
57.               (
58.                   pki.[Entry Type] = 4
59.               )
60.               AND
61.               (
62.                   pki.[Source Type] = 1
63.               )
64.           )
65.        )
```

6. ATSARGOS - PAGRINDINIS

```
1.    SELECT
```

```
2.        PrekesIrasai.[Entry No_] AS IrasoNr,
3.        PrekesIrasai.[Posting Date] AS [Registravimo DATA],
4.        [Item No_] AS [Prekes Nr],
5.        Deze.Code AS [Dezes kodas],
6.        Deze.Description AS [Dezes pavadinimas],
7.        Deze.Code + ' ' + Deze.Description AS Deze,
8.        Vieta.Code AS [Vietos kodas],
9.        Vieta.Name + ' ' + Vieta.[Name 2] AS [Vietos pavadinimas],
10.       Vieta.Code + ' ' + Vieta.Name + ' ' + Vieta.[Name 2] AS Vieta,
11.       PrekesIrasai.[Global Dimension 1 Code] AS [Geografine vieta],
12.       PrekesIrasai.[Global Dimension 2 Code] AS [Funkcinis padalinys],
13.       PrekesIrasai.[Lot No_] AS [Partijos Nr],
14.       PrekesIrasai.[Document No_] AS [Dokumento Nr],
15.       CASE
16.           WHEN PrekesIrasai.[Entry Type] = 0
17.           THEN Quantity
18.       END AS PirkimoKiekis,
19.       CASE
20.           WHEN PrekesIrasai.[Entry Type] = 2
21.           THEN Quantity
22.       END AS TeigiamoKoregavimoKiekis,
23.       CASE
24.           WHEN PrekesIrasai.[Entry Type] = 3
25.           THEN Quantity
26.       END AS NeigiamoKoregavimoKiekis,
27.       CASE
28.           WHEN PrekesIrasai.[Entry Type] = 1
29.           THEN Quantity
30.       END AS ParduotasKiekis,
31.       CASE
32.           WHEN (PrekesIrasai.[Entry Type] = 4
33.           AND PrekesIrasai.Quantity > 0
34.           AND Vieta.[USE AS In-Transit] = 0)
35.           THEN Quantity
36.       END
37.       GautasKiekis,
38.       CASE
39.           WHEN (PrekesIrasai.[Entry Type] = 4
40.           AND PrekesIrasai.Quantity < 0
41.           AND Vieta.[USE AS In-Transit] = 0)
42.           THEN Quantity
43.       END
44.       PerduotasKiekis,
45.       CASE
46.           WHEN PrekesIrasai.[Entry Type] = 0
47.           THEN VertesIrasai.SavikSuma
48.       END AS PirkimoSuma,
49.       CASE
50.           WHEN PrekesIrasai.[Entry Type] = 2
51.           THEN VertesIrasai.SavikSuma
52.       END AS TeigiamoKoregavimoSuma,
53.       CASE
54.           WHEN PrekesIrasai.[Entry Type] = 3
55.           THEN VertesIrasai.SavikSuma
56.       END AS NeigiamoKoregavimoSuma,
57.       CASE
58.           WHEN PrekesIrasai.[Entry Type] = 1
59.           THEN VertesIrasai.SavikSuma
60.       END AS ParduotaSuma,
61.       CASE
62.           WHEN (PrekesIrasai.[Entry Type] = 4
63.           AND PrekesIrasai.Quantity > 0
64.           AND Vieta.[USE AS In-Transit] = 0)
65.           THEN VertesIrasai.SavikSuma
66.       END
67.       GautaSuma,
68.       CASE
69.           WHEN (PrekesIrasai.[Entry Type] = 4
70.           AND PrekesIrasai.Quantity < 0
71.           AND Vieta.[USE AS In-Transit] = 0)
72.           THEN VertesIrasai.SavikSuma
73.       END
74.       PerduotaSuma,
75.       PerdIVieta.Code + ' ' + PerdIVieta.Name AS PerduotaIVieta,
76.       PerdIsVieta.Code + ' ' + PerdIsVieta.Name AS PerduotaIsVietos,
77.       PrekesIrasai.[Bendr_ verslo reg_ grupĖ] AS [Bendra verslo reg grupe],
78.       PrekesIrasai.Quantity AS Kiekis,
79.       PrekesIrasai.[Kiekis 2] AS Kiekis2,
80.       VertesIrasai.SavikSuma AS Suma,
81.       PrekesIrasai.Quantity * Preke.[Net Weight] AS SvorisNeto,
```

```
82.       PrekesIrasai.Quantity * Preke.[Gross Weight] AS SvorisBruto,
83.       CASE
84.           WHEN PrekesIrasai.[Entry Type] = 0
85.           THEN 'Pirkimas'
86.           WHEN PrekesIrasai.[Entry Type] = 1
87.           THEN 'Pardavimas'
88.           WHEN PrekesIrasai.[Entry Type] = 2
89.           THEN 'Teigiamas koregavimas'
90.           WHEN PrekesIrasai.[Entry Type] = 3
91.           THEN 'Neigiamas koregavimas'
92.           WHEN PrekesIrasai.[Entry Type] = 4
93.           THEN 'Perdavimas'
94.       END AS IrasoTipas
95.   FROM dbo.[CRONUS$Item Ledger Entry] PrekesIrasai
96.   INNER JOIN
97.       dbo.[CRONUS$Item] Preke
98.       ON PrekesIrasai.[Item No_] = Preke.[No_]
99.   LEFT OUTER JOIN
100.      dbo.[CRONUS$Location] PerdIsVieta
101.      ON PrekesIrasai.[Perduota iš vietos] = PerdIsVieta.Code
102.  LEFT OUTER JOIN
103.      dbo.[CRONUS$Location] PerdIVieta
104.      ON PrekesIrasai.[Perduota į vietą] = PerdIVieta.Code
105.  LEFT OUTER JOIN
106.      dbo.[CRONUS$Location] Vieta
107.      ON PrekesIrasai.[Location Code] = Vieta.Code
108.  LEFT OUTER JOIN
109.      dbo.[CRONUS$Bin] Deze
110.      ON PrekesIrasai.[Bin Code] = Deze.Code
111.      AND PrekesIrasai.[Location Code] = Deze.[Location Code]
112.  LEFT OUTER JOIN
113.      [Vertės įrašai] VertesIrasai
114.      ON VertesIrasai.[Item Ledger Entry No_] = PrekesIrasai.[Entry No_]
```

7. SKOLOS - PAGRINDINIS

```
1.    SELECT
2.        pki.[Entry No_] AS [Iraso Nr],
3.        pki.[Posting Date] AS DATA,
4.        pki.[Customer No_] AS [CustomerNo],
5.        pki.[External Document No_] AS [Dokumento numeris],
6.        pki.[Sell-TO Customer No_] AS [SellToCustomerNo],
7.        pki.[Global Dimension 1 Code] AS [Geografine vieta],
8.        CASE
9.            WHEN pki.[Document Type] = 1
10.           THEN lpki.Suma
11.           ELSE 0
12.       END AS [Mokejimai],
13.       CASE
14.           WHEN pki.[Document Type] = 2
15.           THEN lpki.Suma
16.           ELSE 0
17.       END AS [Pardavimai],
18.       CASE
19.           WHEN pki.[Document Type] = 3
20.           THEN lpki.Suma
21.           ELSE 0
22.       END AS [Grazinimai],
23.       CASE
24.           WHEN (pki.[Document Type] <> 1)
25.           AND
26.           (
27.               pki.[Document Type] <> 2
28.           )
29.           AND
30.           (
31.               pki.[Document Type] <> 3
32.           )
33.           THEN lpki.Suma
34.           ELSE 0
35.       END AS [Kita],
36.       pki.[Mokėjimo būdo kodas] AS [Mokejimo budo kodas],
37.       pki.[Customer Posting GROUP] AS [Pirkejo registravimo Gr],
38.       pki.[Global Dimension 2 Code] AS [Funkcinis padalinys],
39.       pki.[External Document No_] AS [Isorinis dokumento Nr],
40.       pki.[Document Date] AS [Dokumento DATA],
41.       0 AS [Taros suma],
42.       'Ne' AS Tara,
```

```
        '' AS [Taros kodas],
        '' AS [Taros pavadinimas],
        0 AS [Parduota taros],
        0 AS [Parduota taros (kiekis)],
        0 AS [Grazinta taros],
        0 AS [Grazinta taros (kiekis)],
        lik.Suma AS [Likusi suma],
        CASE
            WHEN datediff(day, pki.[Due Date], getdate()) > 0
            THEN datediff(day, pki.[Due Date], getdate())
            ELSE 0
        END AS [Pradelstu dienu skaicius]
    FROM dbo.[CRONUS$Cust_ Ledger Entry] pki
    INNER JOIN
        dbo.[Detalūs pirkĖjų knygos įrašai] lpki
        ON pki.[Entry No_] = lpki.[Cust_ Ledger Entry No_]
    INNER JOIN
        dbo.[Pirkėjų knygos įrašų likusi suma] lik
        ON pki.[Entry No_] = lik.[Cust_ Ledger Entry No_]
    UNION ALL
    SELECT
        pki.[Entry No_] AS [Iraso Nr],
        pki.[Posting Date] AS DATA,
        g.[Bill-TO Customer No_] AS [CustomerNo],
        pki.[External Document No_] AS [Dokumento numeris],
        pki.[Bin Code] AS [SellToCustomerNo],
        pki.[Global Dimension 1 Code] AS [Geografine vieta],
        0 AS [Mokejimai],
        0 AS [Pardavimai],
        0 AS [Grazinimai],
        0 AS [Kita],
        '' AS [Mokejimo budo kodas],
        '' AS [Pirkejo registravimo Gr],
        pki.[Global Dimension 2 Code] AS [Funkcinis padalinys],
        pki.[External Document No_] AS [Isorinis dokumento Nr],
        pki.[Document Date] AS [Dokumento DATA],
        pki.Quantity * - 1 * i.[Taros kaina] AS [Taros suma],
        'Taip' AS Tara,
        pki.[Item No_] AS [Taros kodas],
        pki.[Item No_] + ' ' + i.Description + ' ' + i.[Description 2] AS [Taros pavadinimas],
        CASE
            WHEN pki.[Entry Type] = 2
            THEN pki.Quantity * 1 * i.[Taros kaina]
            ELSE 0
        END AS [Parduota taros],
        CASE
            WHEN pki.[Entry Type] = 2
            THEN pki.Quantity * 1
            ELSE 0
        END AS [Parduota taros (kiekis)],
        CASE
            WHEN pki.[Entry Type] = 3
            THEN pki.Quantity * 1 * i.[Taros kaina]
            ELSE 0
        END AS [Grazinta taros],
        CASE
            WHEN pki.[Entry Type] = 3
            THEN pki.Quantity * 1
            ELSE 0
        END AS [Grazinta taros (kiekis)],
        0 AS [Likusi suma],
        0 AS [Pradelstu dienu skaicius]
    FROM dbo.[CRONUS$Item Ledger Entry] pki
    INNER JOIN
        dbo.[Vertės įrašai] lpki
        ON pki.[Entry No_] = lpki.[Item Ledger Entry No_]
    INNER JOIN
        [CRONUS$Item] i
        ON pki.[Item No_] = i.[No_]
    INNER JOIN
        [CRONUS$Customer] g
        ON pki.[Bin Code] = g.[No_]
    WHERE
        (
            pki.[Entry Type] IN (2, 3)
        )
        AND
        (
            pki.[Atsargų reg_ grupė] = 'APYV TARA'
        )
```

```
123.      AND
124.      (
125.          pki.[Location Code] = 'PIRKĖJAI'
126.      )
```

3. PRIEDAS. KUBO XML APRAŠO PAVYZDYS

1. PIRKIMAI


```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <DSOXML80:Server Method="Connect" Name="localhost" ClassType="1" ConnectTimeout="0" Description="" Edition="0" IsValid="1"
LockTimeout="0" ProcessingLogFileName="" ServiceState="4" State="1" Timeout="0" Version="8.0"
xmlns:DSOXML80="http://www.microsoft.com/bipractices/DSOXML80">
    - <DSOXML80:Database Method="None" Name="DB" SubClassType="0">
        <DSOXML80:DataSource Method="Add" Name="localhost - OLAP" SubClassType="0" ClassType="6" CloseQuoteChar=""""
        ConnectionString="Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial
        Catalog=OLAP;Data Source=localhost" Description="" IsReadOnly="0" IsValid="1" OpenQuoteChar=""""
        SupportedTxnDDL="8" />

        + <DSOXML80:DatabaseDimension Method="Add" Name="Pirkimai^Atsargu reg gr" SubClassType="0"
        AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="7"
        DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension=""
        Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo"."PrekÄ—"" IsChanging="0"
        IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" JoinClause=""
        LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="1"
        SourceTable=""dbo"."PrekÄ—"" SourceTableAlias=""dbo"."PrekÄ—"" SourceTableFilter="" State="4" StorageMode="1">
        + <DSOXML80:DatabaseDimension Method="Add" Name="Pirkimai^Data" SubClassType="0"
        AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="7"
        DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension=""
        Description="" DimensionType="1" EnableRealTimeUpdates="0" FromClause=""dbo"."Pirkimai"" IsChanging="0"
        IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" JoinClause=""
        LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="1"
        SourceTable=""dbo"."Pirkimai"" SourceTableAlias=""dbo"."Pirkimai"" SourceTableFilter="" State="4" StorageMode="1">
        + <DSOXML80:DatabaseDimension Method="Add" Name="Pirkimai^Deze" SubClassType="0"
        AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="7"
        DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension=""
        Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo"."Pirkimai"" IsChanging="0"
        IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" JoinClause=""
        LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="1"
        SourceTable=""dbo"."Pirkimai"" SourceTableAlias=""dbo"."Pirkimai"" SourceTableFilter="" State="4" StorageMode="1">
        + <DSOXML80:DatabaseDimension Method="Add" Name="Pirkimai^Padalinys" SubClassType="0"
        AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="7"
        DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension=""
        Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo"."Pirkimai"" IsChanging="0"
        IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" JoinClause=""
        LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="1"
        SourceTable=""dbo"."Pirkimai"" SourceTableAlias=""dbo"."Pirkimai"" SourceTableFilter="" State="4" StorageMode="1">
        + <DSOXML80:DatabaseDimension Method="Add" Name="Pirkimai^Prekes grupe" SubClassType="0"
        AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="7"
        DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension=""
        Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo"."PrekÃ  ", "dbo"."Pirkimai""
        IsChanging="0" IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0"
        JoinClause="("dbo"."PrekÃ  "."PrekesNr"="dbo"."Pirkimai"."PrekesNr")" LastProcessed="2005.04.24 13:42:02"
        LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="1" SourceTable=""dbo"."Pirkimai""
        SourceTableAlias=""dbo"."Pirkimai"" SourceTableFilter="" State="4" StorageMode="1">
        + <DSOXML80:DatabaseDimension Method="Add" Name="Pirkimai^Tiek reg grupe" SubClassType="0"
        AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="7"
        DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension=""
        Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo"."Pirkimai"" IsChanging="0"
        IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" JoinClause=""
        LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="1"
        SourceTable=""dbo"."Pirkimai"" SourceTableAlias=""dbo"."Pirkimai"" SourceTableFilter="" State="4" StorageMode="1">
        + <DSOXML80:DatabaseDimension Method="Add" Name="Pirkimai^Vieta" SubClassType="0"
        AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="7"
        DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension=""
        Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo"."Vieta"" IsChanging="0"
        IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" JoinClause=""
        LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="1"
        SourceTable=""dbo"."Vieta"" SourceTableAlias=""dbo"."Vieta"" SourceTableFilter="" State="4" StorageMode="1">
        - <DSOXML80:Cube Method="Add" Name="Pirkimai" SubClassType="0" AggregationPrefix="" ClassType="9"
        Description="" EstimatedSize="6244524" LastUpdated="1899.12.30 00:00:00" OlapMode="0" State="4" AllowDrillThrough="0"
        DataSource="localhost - OLAP" DefaultMeasure="" DrillThroughColumns="" DrillThroughFilter="" DrillThroughFrom=""
        DrillThroughJoins="" EstimatedRows="0" FromClause=""dbo"."PrekÃ  ", "dbo"."Pirkimai", "dbo"."Vieta""
        IsReadWrite="0" IsTemporary="0" IsValid="1" JoinClause="("dbo"."PrekÃ  "."PrekesNr"="dbo"."Pirkimai"."PrekesNr")
        AND ("dbo"."Pirkimai"."VietosKodas"="dbo"."Vieta"."VietKodas")" ProcessingKeyErrorLimit="0"
```

ProcessingKeyErrorLogFileName="" ProcessOptimizationMode="0" SourceTable=""dbo"."Pirkimai"" SourceTableAlias=""dbo"."Pirkimai"" SourceTableFilter="" IsVisible="1">
<DSOXML80:CustomProperty Method="Update" Name="SchemaLayout" DataType="0" Value="({localhost - OLAP}{dbo.Pirkimai}{dbo.Pirkimai}{1200}{600}{2760}{4400}{1})({localhost - OLAP}{dbo.PrekÄ }{dbo.PrekÄ }{900}{5610}{1935}{3885}{1})({localhost - OLAP}{dbo.Vieta}{dbo.Vieta}{5700}{5325}{1800}{2200}{1})" />
<DSOXML80:CustomProperty Method="Update" Name="SchemaJoins" DataType="0" Value="("dbo."PrekÄ "."PrekesNr"="dbo."Pirkimai"."PrekesNr") AND ("dbo."Pirkimai"."VietosKodas"="dbo."Vieta"."VietKodas")" />
+ <DSOXML80:CubeDimension Method="Add" Name="Pirkimai^Data" SubClassType="0" AggregationUsage="0" AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="11" DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension="" Description="" DimensionType="1" EnableRealTimeUpdates="0" FromClause=""dbo."Pirkimai"" IsChanging="0" IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" IsVisible="1" JoinClause="" LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="1" SourceTable=""dbo."Pirkimai"" SourceTableAlias=""dbo."Pirkimai"" SourceTableFilter="" StorageMode="1">
+ <DSOXML80:CubeDimension Method="Add" Name="Pirkimai^Prekes grupe" SubClassType="0" AggregationUsage="0" AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="11" DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension="" Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo."PrekÄ ", "dbo."Pirkimai"" IsChanging="0" IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" IsVisible="1" JoinClause="("dbo."PrekÄ "."PrekesNr"="dbo."Pirkimai"."PrekesNr")" LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="2" SourceTable=""dbo."Pirkimai"" SourceTableAlias=""dbo."Pirkimai"" SourceTableFilter="" StorageMode="1">
+ <DSOXML80:CubeDimension Method="Add" Name="Pirkimai^Tiek reg grupe" SubClassType="0" AggregationUsage="0" AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="11" DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension="" Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo."Pirkimai"" IsChanging="0" IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" IsVisible="1" JoinClause="" LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="3" SourceTable=""dbo."Pirkimai"" SourceTableAlias=""dbo."Pirkimai"" SourceTableFilter="" StorageMode="1">
+ <DSOXML80:CubeDimension Method="Add" Name="Pirkimai^Geografine vieta" SubClassType="0" AggregationUsage="0" AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="11" DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension="" Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo."Pirkimai"" IsChanging="0" IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" IsVisible="1" JoinClause="" LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="4" SourceTable=""dbo."Pirkimai"" SourceTableAlias=""dbo."Pirkimai"" SourceTableFilter="" StorageMode="1">
+ <DSOXML80:CubeDimension Method="Add" Name="Pirkimai^Padalinys" SubClassType="0" AggregationUsage="0" AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="11" DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension="" Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo."Pirkimai"" IsChanging="0" IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" IsVisible="1" JoinClause="" LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="5" SourceTable=""dbo."Pirkimai"" SourceTableAlias=""dbo."Pirkimai"" SourceTableFilter="" StorageMode="1">
+ <DSOXML80:CubeDimension Method="Add" Name="Pirkimai^Atsargu reg gr" SubClassType="0" AggregationUsage="0" AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="11" DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension="" Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo."PrekÄ "" IsChanging="0" IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" IsVisible="1" JoinClause="" LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="6" SourceTable=""dbo."PrekÄ "" SourceTableAlias=""dbo."PrekÄ "" SourceTableFilter="" StorageMode="1">
+ <DSOXML80:CubeDimension Method="Add" Name="Pirkimai^Baz mat vnt" SubClassType="0" AggregationUsage="0" AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="11" DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension="" Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo."PrekÄ "" IsChanging="0" IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" IsVisible="1" JoinClause="" LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="7" SourceTable=""dbo."PrekÄ "" SourceTableAlias=""dbo."PrekÄ "" SourceTableFilter="" StorageMode="1">
+ <DSOXML80:CubeDimension Method="Add" Name="Pirkimai^Deze" SubClassType="0" AggregationUsage="0" AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="11" DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension="" Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo."Pirkimai"" IsChanging="0" IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" IsVisible="1" JoinClause="" LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="8" SourceTable=""dbo."Pirkimai"" SourceTableAlias=""dbo."Pirkimai"" SourceTableFilter="" StorageMode="1">
+ <DSOXML80:CubeDimension Method="Add" Name="Pirkimai^Vieta" SubClassType="0" AggregationUsage="0" AllowSiblingsWithSameName="0" AreMemberKeysUnique="0" AreMemberNamesUnique="0" ClassType="11" DataMemberCaptionTemplate="(* data)" DataSource="localhost - OLAP" DefaultMember="" DependsOnDimension="" Description="" DimensionType="0" EnableRealTimeUpdates="0" FromClause=""dbo."Vieta"" IsChanging="0" IsReadWrite="0" IsShared="0" IsTemporary="0" IsValid="1" IsVirtual="0" IsVirtual70="0" IsVisible="1" JoinClause="" LastProcessed="2005.04.24 13:42:02" LastUpdated="1899.12.30 00:00:00" MembersWithData="0" OrdinalPosition="9" SourceTable=""dbo."Vieta"" SourceTableAlias=""dbo."Vieta"" SourceTableFilter="" StorageMode="1">
<DSOXML80:CubeMeasure Method="Add" Name="Pirkimo suma" SubClassType="0" AggregateFunction="0" ClassType="10" Description="" FormatString="Standard" IsInternal="0" IsValid="1" IsVisible="1" OrdinalPosition="1" SourceColumn=""dbo."Pirkimai."PirkimoSuma"" SourceColumnType="5" />

```
    <DSOXML80:CubeMeasure Method="Add" Name="Pirkimo kiekis" SubClassType="0" AggregateFunction="0"
ClassType="10" Description="" FormatString="Standard" IsInternal="0" IsValid="1" IsVisible="1" OrdinalPosition="2"
SourceColumn=""dbo"."Pirkimai"."PirkimoKiekis"" SourceColumnType="5" />
    <DSOXML80:CubeMeasure Method="Add" Name="Kiekis" SubClassType="0" AggregateFunction="0" ClassType="10"
Description="" FormatString="Standard" IsInternal="0" IsValid="1" IsVisible="1" OrdinalPosition="3"
SourceColumn=""dbo"."Pirkimai"."Kiekis"" SourceColumnType="5" />
    <DSOXML80:CubeMeasure Method="Add" Name="Suma" SubClassType="0" AggregateFunction="0" ClassType="10"
Description="" FormatString="Standard" IsInternal="0" IsValid="1" IsVisible="1" OrdinalPosition="4"
SourceColumn=""dbo"."Pirkimai"."Suma"" SourceColumnType="5" />
    + <DSOXML80:Command Method="Add" Name="Pirkimo kaina" SubClassType="0" ClassType="13" CommandType="1"
Description="" IsValid="1" OrdinalPosition="1">

    CREATE MEMBER CURRENTCUBE.Measures.[Pirkimo kaina] AS '[Measures].[Pirkimo
    Suma]/[Measures].[Pirkimo Kiekis]'
    + <DSOXML80:Command Method="Add" Name="Likutis pradziai kiekis" SubClassType="0" ClassType="13"
    CommandType="1" Description="" IsValid="1" OrdinalPosition="2">

    CREATE MEMBER CURRENTCUBE.Measures.[Likutis pradziai kiekis] AS
    'Sum(PeriodsToDate([Data].[(All)],[Data].PrevMember),[Measures].[Kiekis])'
    + <DSOXML80:Command Method="Add" Name="Likutis pabaigai kiekis" SubClassType="0" ClassType="13"
    CommandType="1" Description="" IsValid="1" OrdinalPosition="3">

    CREATE MEMBER CURRENTCUBE.Measures.[Likutis pabaigai kiekis] AS
    'Sum(PeriodsToDate([Data].[(All)]),[Measures].[Kiekis])'
    + <DSOXML80:Command Method="Add" Name="Likutis pradziai suma" SubClassType="0" ClassType="13"
    CommandType="1" Description="" IsValid="1" OrdinalPosition="4">

    CREATE MEMBER CURRENTCUBE.Measures.[Likutis pradziai suma] AS
    'Sum(PeriodsToDate([Data].[(All)],[Data].PrevMember),[Measures].[Suma])'
    + <DSOXML80:Command Method="Add" Name="Likutis pabaigai suma" SubClassType="0" ClassType="13"
    CommandType="1" Description="" IsValid="1" OrdinalPosition="5">

    CREATE MEMBER CURRENTCUBE.Measures.[Likutis pabaigai suma] AS
    'Sum(PeriodsToDate([Data].[(All)]),[Measures].[Suma])'
    + <DSOXML80:Partition Method="Add" Name="Pirkimai" SubClassType="0" AggregationPrefix="Pirkimai_Pirkimai_"
    ClassType="19" Description="" EstimatedSize="0" LastUpdated="1899.12.30 00:00:00" OlapMode="0" State="4"
    AllowDrillThrough="0" DataSource="localhost - OLAP" DefaultMeasure="" DrillThroughColumns="" DrillThroughFilter=""
    DrillThroughFrom="" DrillThroughJoins="" EnableRealTimeUpdates="0" EstimatedRows="0" FromClause=""dbo"."Pirkimai",
    "dbo"."PrekÄ  ", "dbo"."Vieta"" IsDefault="1" IsReadWrite="0" IsTemporary="0" IsValid="1"
    JoinClause="("dbo"."PrekÄ  "."PrekesNr"="dbo"."Pirkimai"."PrekesNr") AND
    ("dbo"."Pirkimai"."VietosKodas"="dbo"."Vieta"."VietKodas")" ProcessingKeyErrorLimit="0"
    ProcessingKeyErrorLogFileName="" ProcessOptimizationMode="0" SourceTable=""dbo"."Pirkimai""
    SourceTableAlias=""dbo"."Pirkimai"" SourceTableFilter="" RemoteServer="" AccumulatedSize="258325171"
    AggregationsCount="272" PercentageBenefit="25">
    </DSOXML80:Cube>
  </DSOXML80:Database>
</DSOXML80:Server>
```

4. PRIEDAS. KONFERENCIJOJE PRISTATYTAS IR IŠSPAUSDINTAS STRAIPSNIS

Verslo valdymo sistemos Navision Attain ir OLAP duomenų analizės integracija

Algirdas Kepežinskas, Line Nemuraitė
Kauno technologijos universitetas
Informacijos sistemų katedra, Studentų 50-308

Straipsnyje analizuojama verslo valdymo sistemos Navision Attain duomenų išgavimas, transformacija ir galiausiai panaudojimas OLAP duomenų analizės priemonėse. Nagrinėjamos šių dviejų sistemų integracijos savybės, reikalingos transformacijos bei dažniausiai kylančios problemos. Sudarytas pavyzdinis duomenų kubas prekių apyvartai stebėti. Sukurtas automatinis kubo atnaujinimas, apibendrintos platesnės panaudojimo galimybės.

1. ĮVADAS

Lietuvoje vis daugiau ir daugiau vidutinių ir stambių įmonių diegiasi verslo valdymo sistemas. Ar tai būtų Navision Attain, SAP, Axapta, ar bet kuri kita VVS, jos suteikia įmonėms įrankį efektyviai valdyti resursus, išteklius, finansus, gamybą, bei kitus įmonėje vykstančius verslo procesus. Visos šios sistemos savyje turi numatytas ataskaitų generavimo priemonės, tačiau vis dažniau ir dažniau šių sąlyginai paprastų priemonių nebeužtenka norint patenkinti vis augančius vartotojų poreikius.

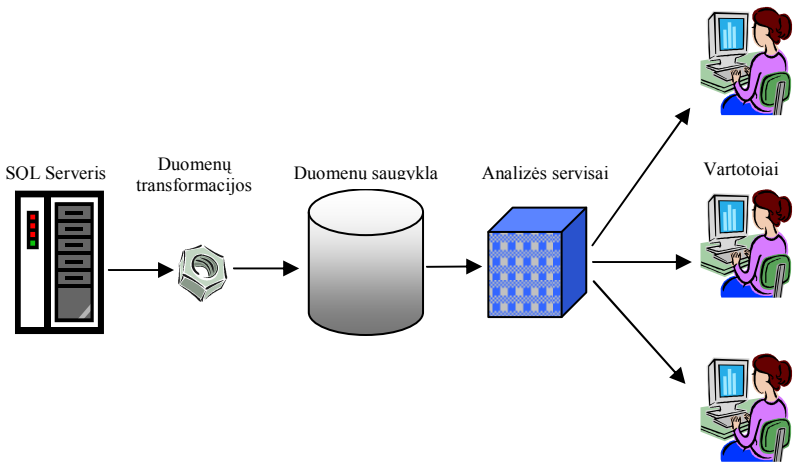
Ar dėl patogumo, ar dėl resursų stygiaus, ar dėl plėtimo galimybių, vis daugiau įmonių renkasi duomenų analizę atlikti naudojantis OLAP priemonėmis. Naudojantis šiomis priemonėmis, dalis realios duomenų bazės duomenų, reikalingų analizei, perkeliami į duomenų saugyklą (*data warehouse*), kur jie transformuojami, pritaikant juos specifiniams poreikiams, saugomi specifinėse duomenų saugyklose (*data marts*) ir toliau apdorojami OLAP priemonėmis. Iš jų kuriami duomenų analizės kubai, kurie leidžia duomenis matyti įvairiais pjūviais, analizuoti juos pagal įvairias dimensijas, grupuoti, skirstyti, laisvai pasirenkant jų pateikimo formą.

Microsoft SQL Server Analysis Services suteikia galimybę atlikti visas šias operacijas, tuo labai palengvindamas didelių duomenų kiekių analizę. Duomenų išgavimui ir transformavimui naudojama SQL Server Data Transformation Services, kurie įgalina duomenis pakeisti bei pritaikyti taip kaip reikalauja konkretus atvejis. Pakeisti duomenys saugojami tame pačiame SQL Serveryje, toje pačioje ar naujose duomenų bazėse. Iš ten juos paima Analysis Services, ir pagal suprojektuotą kubų schemą juos apdoroja, bei saugo galutinėje jų saugykloje (priklausomai nuo saugyklos tipo – pasirinkus saugyklos tipą MOLAP, duomenys perkeliami į Analysis Services duomenų bazę, pasirinkus ROLAP ar HOLAP tipą, duomenys lieka originalioje bazėje).

2. NAVISION ATTAIN ĮDIEGIMO VARIANTAI

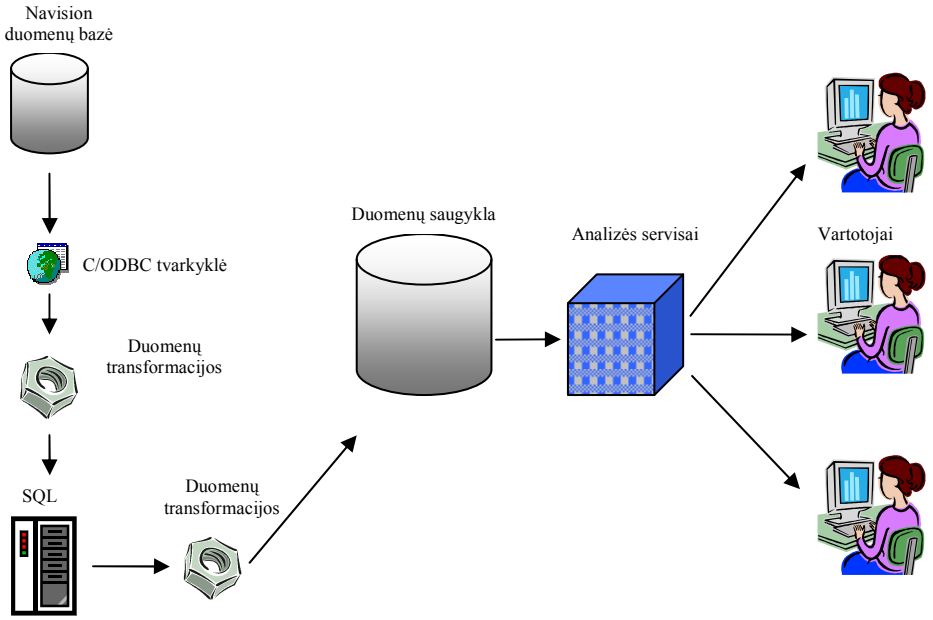
Navision Attain ir OLAP duomenų analizės priemonių integracijos architektūra, priklauso nuo Navision Attain įdiegimo tipo. Šiuo metu yra siūlomi du Navision Attain diegimo variantai:

3) Naudojant SQL Server 2000 duomenų bazę. Šis variantas skirtas didesnės apimties įmonėms, reikalaujančioms didesnių galimybių bei našumo. Jis supaprastina Navision Attain ir OLAP duomenų analizės integraciją, nes visi Navision Attain duomenys jau yra saugomi SQL Serveryje, t.y. galima praleisti vieną išgavimo žingsnį. Tokiu atveju, integracijos architektūra atrodo taip:



1. pav. integracijos architektūra naudojant SQL Server duomenų bazę

4) Naudojant Navision Attain gimtąją (*native*) duomenų bazę. Šiuo atveju visi sistemos duomenys yra saugomi Navision Attain serverio gimtojoje duomenų bazėje. Norint juos sėkmingai panaudoti OLAP analizėje, juos reikia perkelti į SQL Serverį, o tam reikalingas papildomas išgavimo žingsnis naudojant C/ODBC duomenų bazės tvarkyklę.



2. pav. integracijos architektūra naudojant Navision Attain gimtąją duomenų bazę

Nepriklausomai nuo to, koks diegimo variantas yra pasirinktas įmonėje, šią integraciją pilnai galima atlikti naudojantis SQL Server 7.0 esančiomis priemonėmis.

3. NAVISION ATTAIN DUOMENŲ IŠGAVIMAS, TRANSFORMAVIMAS, IR UŽKROVIMAS

Šiame straipsnyje analizuosime atvejį, kai Navision Attain sistema įdiegta naudojant SQL Server duomenų bazę.

Norint atlikti sėkmingą Navision Attain ir OLAP duomenų analizės integraciją, reikia atlikti šiuos etapus:

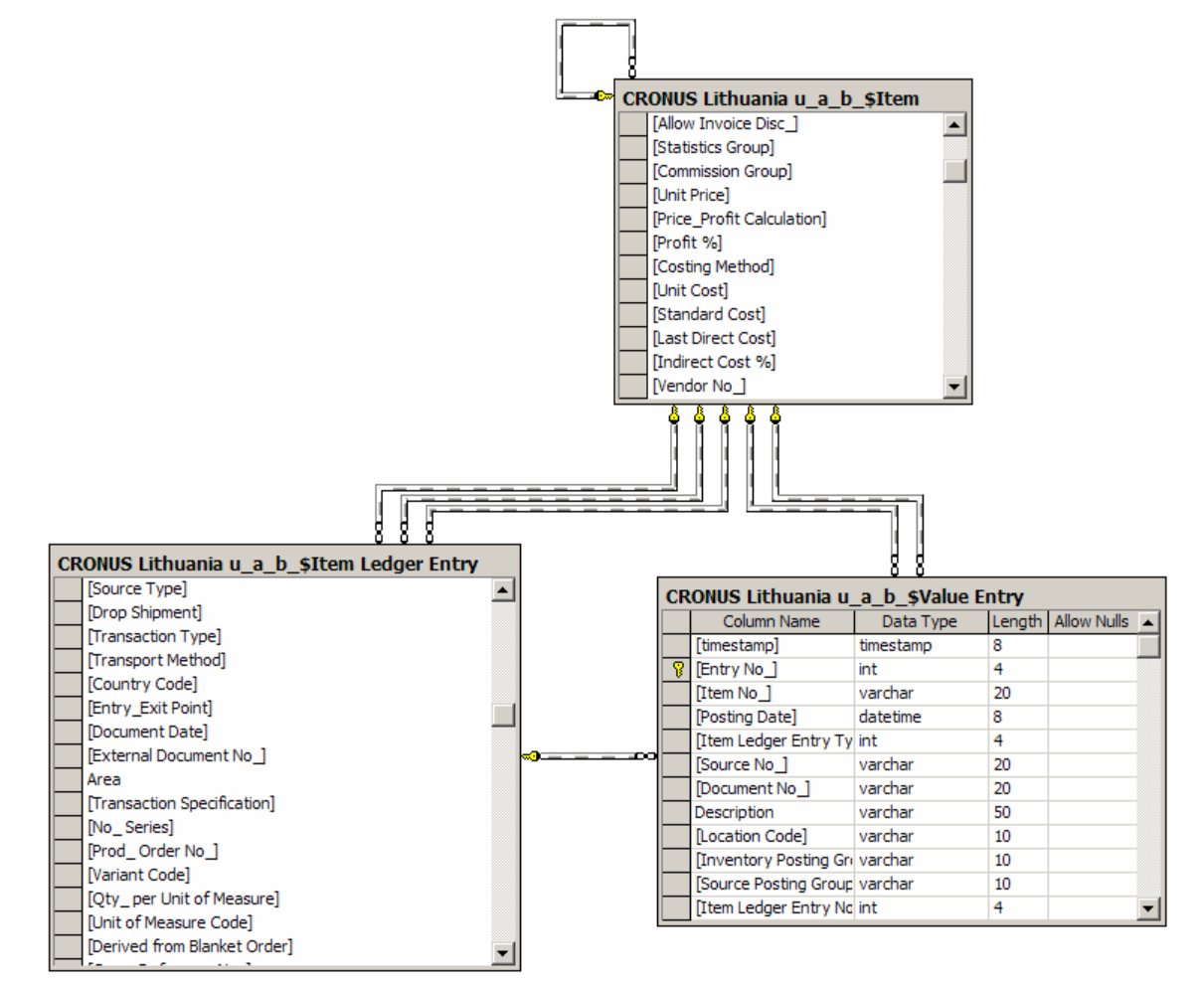
1) Identifikuoti reikalingas duomenų bazės lenteles. Navision Attain sistemoje yra apie 1000 skirtingų lentelių, todėl ypač svarbu yra tiksliai bei teisingai identifikuoti reikalingas analizei. Priešingu atveju, rizikuojama susidurti su našumo problemomis, atsiranda didesnė klaidų tikimybė, bei sudėtingėja projektavimo ir kūrimo procesas.
2) Sukurti rodinius (*views*) ir DTS (*data transformation services*) atliekančius reikiamas duomenų transformacijas. Navision Attain naudojami specifiniai lentelių bei laukų pavadinimai, kurie dažniausiai sukelia problemų dirbant su Analysis Services paketu. Be to, daugelis operacijų nereikalauja būtinų klasifikatorių (ir kitų dimensijų) reikšmių, todėl šiuo atveju reikia užtikrinti, kad net ir nesant tuštiems įrašams dimensijų lentelėse, faktų lentelės įrašai vistiek bus įtraukti į kubą (tai aktualu, nes Analysis Services neleidžia kurti OUTER JOIN tipo ryšių). Šios problemos lengvai išsprendžiamos panaudojus minėtus rodinius bei DTS.
3) Suprojektuoti bei sukurti kubo schemą. Šiame žingsnyje yra svarbu numatyti ir tinkamai sukurti sąryšius tarp skirtingų lentelių, užtikrinti jų korektiškumą ir teisingumą.
4) Sukurti automatinį kubo atnaujinimo mechanizmą, nustatyti atnaujinimo intervalus ir patikrinti jo veikimą.

4. PAVYZDINIS ANALIZĖS KUBAS

Vaizdumui, bus sukurtas pavyzdinis Navision Attain duomenų analizės kubas, skirtas prekių apyvartai stebėti.

1) Šiam kubui mus reikės šių lentelių: Prekė, Prekės knygos įrašas, Vertės įrašas. Sukūrus pavyzdinę diagramą, iškart matyti kokie sudėtingi yra Navision Attain lentelių sąryšiai. Mes

naudosimės tik keliais iš jų: Prekės knygos įrašas.Prekės Nr. -> Prekė.Nr. ir Vertės įrašas.Prekės knygos įrašo Nr. -> Prekės knygos įrašas.Įrašo Nr.



3. pav. pavyzdinio kubo lentelių schema

2) Iš schemos matyti, jog yra daugybė mums nereikalingų laukų. Taip pat matyti, jog laukų pavadinimai turi savyje tarpų, kas sukeltų problemų dirbant su kubais. Vadinasi, teks parašyti keletą rodinių kad ištaisyti šias problemas

Atsargos – vertes irasai:

```
CREATE VIEW dbo.[Atsargos - vertes irasai]
AS
SELECT         [Item Ledger Entry No_] AS PrekesKnIrasoNr, SUM([Cost Amount (Actual)]) AS Suma
FROM         dbo.[CRONUS Lithuania u_a_b_$Value Entry]
GROUP BY [Item Ledger Entry No_]
```

Atsargos – preke:

```
CREATE VIEW dbo.[Atsargos - preke]
AS
SELECT     No_ AS PrekesNr, Description AS PrekesPavadinimas
FROM         dbo.[CRONUS Lithuania u_a_b_$Item]
```

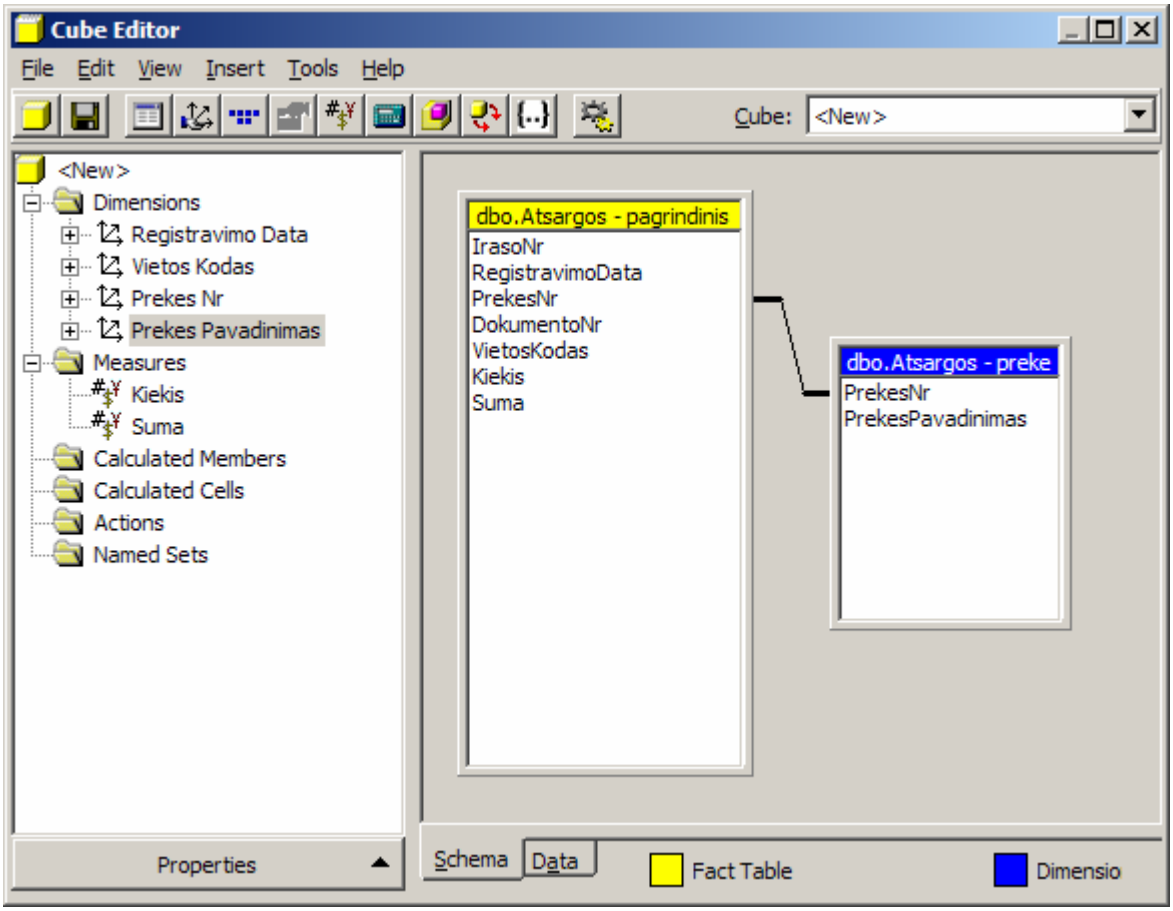
Atsargos – pagrindinis:

```
CREATE VIEW dbo.[Atsargos - pagrindinis]
AS
SELECT      pki.[Entry No_] AS IrasoNr, pki.[Posting Date] AS RegistravimoData, pki.[Item No_]AS PrekesNr, pki.[Document No_] AS DokumentoNr,
```

```
                    pki.[Location Code] AS VietosKodas, pki.Quantity AS Kiekis,
vi.Suma
FROM        dbo.[CRONUS Lithuania u_a_b_$Item Ledger Entry] pki INNER JOIN
                    dbo.[Atsargos - vertes irasai] vi ON pki.[Entry No_] =
vi.PrekesKnIrasoNr
```

Atsargos – pagrindinis rodinys tarnaus kaip faktų lentelė, *Atsargos – prekė* – kaip dimensijų lentelė.

3) Sudėjus lenteles į kubų redaktorių, sukūrus sąryšius bei dimensijas, gautas naujasis kubas:



4. pav. gautasis atsargų kubas

Vietos Kodas	MĖLYNAS						
						Data	
Year	Quarter	Month	Day	Dokumento Nr	Prekes Pavadinimas	Kiekis	Suma
2001	Quarter 1	January	3	107018	INNSBRUCK saugykla/st.durys	25,00	25.075,00
				107018 Total		25,00	25.075,00
			3 Total			25,00	25.075,00
			6	107019	Stiklinės durys	52,00	10.670,40
					Varžtas	500,00	18.525,00
				107019 Total		552,00	29.195,40
			6 Total			552,00	29.195,40
			9		Dažai, geltoni	800,00	6.400,03
			9 Total			800,00	6.400,03
			15		Kompiuteris - Highline paketas	-3,00	0,00
					Serveris - Enterprise paketas	-1,00	0,00
			15 Total			-4,00	0,00
			17		INNSBRUCK saugykla/st.durys	-10,00	-10.030,00
					Stiklinės durys	-5,00	-1.080,00
			17 Total			-15,00	-11.110,00
			22		OSLO saugykla/lentyna	7,00	3.836,00
			22 Total			7,00	3.836,00
			25		75W garsiakalbis, vyšninis	11,00	0,00
					Pagrindas	-15,00	-1.800,00
					Užpakalinė dalis	-31,00	-2.728,00
					Viršutinė dalis	-3,00	-258,00
			25 Total			-38,00	-4.786,00
		January Total				1.327,00	48.610,43
	Quarter 1 Total					1.327,00	48.610,43
2001 Total						1.327,00	48.610,43
Grand Total						1.327,00	48.610,43

5. pav. naujojo kubo duomenų pavyzdys

4) Kuriant automatinį kubo atnaujinimą, pasinaudosime SQL Server DTS įrankiais – sukursime pakuotę kubo atnaujinimui, ir nustatysime kad jį būtų paleidžiama kiekvieną naktį.

Jobs 1 Item

Name	Category	Enabled	Scheduled	Next Run Date
Naujinti atsargu kuba	[Uncategorized (Local)]	Yes	Yes	(2004.12.20 02:00:00)

6. pav. sukurtas automatinis kubo atnaujinimas

Norint sukurti kubą, išgaunantį duomenis iš Navision Attain sistemos įdiegtas gimtojoje bazėje, reikėtų papildomų žingsnių išgaunant duomenis naudojantis C/ODBC tvarkyklėmis ir saugant juos SQL bazėje. Sekantys žingsniai išlieka tokie patys.

5. DAŽNIAUSIAI KYLANČIOS PROBLEMOS

Būtina paminėti, jog naudojant Navision Attain duomenis OLAP duomenų analizės priemonėse, susiduriama su keletu rimtų problemų. Kai kurių iš jų sprendimas yra paprastas, kitų sudėtingesnis, tačiau norint sukurti optimaliai veikiančią sistemą, būtina atsižvelgti į jas visas.

1. Navision Attain naudoja specialius laukų tipus, vadinamus *FlowFields* ir *FlowFilters*. Šie laukų tipai skirti greitam duomenų filtravimui ir išrinkimui iš susietų lentelių. Kadangi tai yra virtualūs laukai, jie SQL duomenų bazėje neturi atitikmens, ir norint juos panaudoti tenka išrinkimo užklausas formuoti atskirai. Tam galima naudoti du būdus:
   - naudoti Navision Attain kuriamas ir palaikomas *FlowFields* skaičiavimas skirtas lenteles. Šiose lentelėse saugomos dalinės sumos ir kiti skaičiavimai, naudojami laukų atvaizdavimui. Tai yra didžiausią spartą užtikrinantis būdas, tačiau sunkiai pritaikomas praktiškai – bet koks pakeitimas Navision Attain sistemoje padarytų šiuos skaičiavimus nebetinkamais.
   - naudoti atskirus rodinius šių laukų skaičiavimui. Tokiu būdu formuojamos užklausos kurių pagalba galima imituoti virtualius laukus. Rodiniai toliau naudojami pagrindinėje užklausoje.
2. Lėtas kubų atnaujinimo procesas. Šią problemą dažniausiai sukelia keletas veiksnių:

- Didelis naudojamų dimensijų kiekis. Dažniai dimensijų kiekio sumažinti negalima, todėl reikia ieškoti kitų sprendimų, kaip pavyzdžiui kuo daugiau dimensijų iškelti į dimensijų lenteles, kad kuo mažiau jų liktų faktų lentelėje. Taip pat svarbu datos dimensijoms naudoti atskiras lenteles, nes jų išrinkimas ir atnaujinimas trunka labai ilgai.
- Pagrindinės duomenų bazės apkrovimas. Šiuo atveju reikia apsvarstyti galimybę duomenis replikuoti į tik OLAP analizei skirtą duomenų saugyklą, ir kubų atnaujinimus vykdyti iš ten.

3. Lėtas pjūvių generavimas. Dažniausiai šią problemą nulemia tai, jog generuojami labai smulkūs pjūviai, to visai nenorint. Todėl reikėtų dimensijas grupuoti į logines grupes, pvz.: Prekių kategorija, Prekių grupė, Prekė. Tai atlikus, pjūvius galima generuoti stambesnius, tuo sumažinant reikalingų skaičiavimų kiekį, ir tuo pačių jų atlikimo trukmę.
4. Per didelis dimensijų narių kiekis. Norint analizuoti duomenis pagal, pvz., dokumento numerį, susiduriama su 64 tūkstančių dimensijų narių apribojimu. Tokiu atveju šias dimensijas reikia grupuoti pagal, pvz., pirmus simbolius, pagal datą, ar panašiai.

6. IŠVADOS

Apibendrinant Navision Attain verslo valdymos ir OLAP duomenų analizės integraciją, reiktų pasakyti, kad:

1. Poreikis naudoti galingesnes duomenų analizės priemonės verslo valdymo sistemų duomenims analizuoti vis didėja.
2. Galimybė panaudoti OLAP duomenų analizės priemones Navision Attain sistemoms duomenims analizuoti yra visada – nesvarbu ar Navision Attain serveris įdiegtas SQL Server bazėje, ar gimtojoje bazėje.
3. Analizuojant Navision Attain duomenis kyla keletas standartinių kliučių, tačiau visoms joms išspręsti pakanka turimų priemonių.
4. Analizės galimybės yra neribotos – jas riboja tik turimi techninės įrangos resursai. Galima išgauti bet kokį norimą duomenų vaizdą.
5. OLAP duomenų analizę galima panaudoti bet kurioje įmonėje, nepriklausomai nuo joje vykstančių verslo procesų tipo bei sudėtingumo.

Reikia manyti, jog laikui bėgant tokių diegimų ir integracijų su Navision Attain bei kitomis VVS tik daugės, todėl jau dabar yra svarbu įsisavinti pagrindus, kad prireikus būtų galima greitai ir efektyviai pritaikyti juos konkrečiai integracijai. Tai yra sąlyginai nauja ir perspektyvi rinka Lietuvoje, todėl būtina kuo plačiau ją išnaudoti, siekiant tiek naudos sau, tiek informacinių sistemų plėtrai Lietuvoje.

7. LITERATŪRA

[1] **T.BAIN, M. BENKOVICH ir kiti**, *„Professional SQL Server 2000 Data Warehousing with Analysis Services“*, 2nd. ed., Wrox Press, 2001.

[2] **S. GOIL, A. CHOUDHARY**, *“High Performance Multidimensional Analysis and Data Mining“*.

[3] **R. JACOBSON**, *„Microsoft SQL Server 2000 Analysis Services Step by Step“*, Microsoft, 2000.

[4] **D. MOODY, M.KORTINK**, *„From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design // Proceedings of the International Workshop on Design and Management of Data Warehouses“*, Stockholm, Sweden, 2000. Prieiga per internetą: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-28/paper5.pdf

[5] **Microsoft Corp.**, *„Designing and Implementing OLAP Solutions with Microsoft SQL Server 2000“*, 2000.

[6] **Microsoft Corp.**, *„Delivery Guide Populating a Data Warehouse with Microsoft® SQL Server™ 2000 Data Transformation Services“*, 2001.

[7] **S. PAUL, N. GUATAM, R. BALINT**, *„Preparing and Mining Data with Microsoft SQL Server 2000 and Analysis Services“*, Microsoft, 2002.

[8] **E. THOMSON**, *„OLAP Solutions. Building Multidimensional Information Systems“*, 2nd. ed., John Wiley & Sons, 2002.

MICROSOFT BUSINESS SOLUTIONS NAVISION ATTAIN AND OLAP DATA ANALYSIS INTEGRATION

Algirdas Kepežinskas, Lina Nemuraitė

This work covers data extraction, transformation and loading (ETL) for analyzing Navision Attain data using OLAP data analysis tools. Different integration architectures, aswell as needed transformations and most often problems are also covered, together with possible solutions to them. A sample data cube is created from Navision Attain, for item inventory tracking. In addition, an example schedule for cube updating is created, and wider usage guidelines are provided.