LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

DATU NOLIKTAVAS ARHITEKTŪRA PARĀDU PIEDZIŅAS UZŅĒMUMĀ

BAKALAURA DARBS

Autors: **Viesturs Olekšs**
St. apl. Nr. vo13001
Darba vadītājs: Asoc. prof. Laila Niedrīte

RĪGA 2013

ANOTĀCIJA

Darba mērķis ir izpētīt datu noliktavu izstrādes pamatprincipus un metodes un pielietot iegūtās zināšanas praksē, izstrādājot datu noliktavu parādu piedziņas uzņēmumam.

Darba pirmajā daļā ir izklāstīts no kādām komponentēm sastāv datu noliktavas arhitektūra. Ir apskatīti arhitektūras tipi atkarībā no datu noliktavas izstrādes pieejas, to raksturīgākās iezīmes, kā arī to priekšrocības un trūkumi.

Darba otrajā daļā ir aprakstīta izstrādātās datu noliktavas arhitektūra. Ir apskatīts no kādām komponentēm sastāv šī datu noliktava, kāda pieeja un tehnoloģijas tika lietotas un kādi rīki tika izmantoti datu noliktavas izstrādē. Tiek atspoguļota risinājumu izvēle, kā arī to stiprās un vājās puses.

Atslēgvārdi: datu noliktava, datu noliktavas arhitektūra, datuve, ETL.

ABSTRACT

The goal of this paper is explore best practices and technologies of data warehousing and use that theoretical knowledge to design and develop data warehouse solution in debt collection company.

The first chapter describes components of data warehouse architecture. It describes data warehousing methodologies and the impact of chosen development approach on data warehouse architecture. It explains pros and cons of different design approaches.

The second chapter describes the components of developed data warehouse in debt collection company. It explains how the design was made and what approach has been taken.

Keywords: data warehouse, data warehouse architecture, ETL.

SATURS

APZĪMĒJUMI

Latviskais nosaukums	Angliskais nosaukums	Paskaidrojums
Datuve	Datamart	Datu kopa, ar optimizētu datu modeli priekš datu analīzes
Vecāku ieraksti	Parent record	Ieraksti citā tabulā, kas satur ārējās atslēgas, kas norada uz tekošās tabulas ierakstu unikālo identifikatoru.
Surogātatslēga	Surogatkey	Unikāls ieraksta identifikators datubāzes tabulā, kuram nav biznesa nozīmes
Naturālā atslēga	Natural key	Unikāls ieraksta identifikators, kas sastāv no atribūtiem, kas nodrošina atribūta unikalitāti pēc biznesa loģikas
Datu kartējums	Data mapping	Datu pārveidošana shēma starp diviem datu modeļiem
Metadati	Metadata	Dati par datiem datu noliktavā. Tie sniedz informāciju biznesa lietotājiem par datu nozīmi un klienta rīkiem par saitēm starp datiem

SAĪSINĀJUMI

DN	Datu noliktava
DBVS	Datu bāzu vadības sistēma
IS	Informācijas sistēma
OS	Operētājsistēma
BI	Biznesa inteliģence
ETL	Izlāde Transformācija Ielāde (no angliskā Extract Transform Load)
ER	Entītiju Relācijas (no angliskā Entity Relationship)
OWB	Oracle Warehouse Builder
PL/SQL	Programmēšanas valoda (no angliskā Procedural Language/Structured Query Language)

IEVADS

Lai pieņemtu ekonomiski pamatotus un panākumus nesošus lēmumus, vadītājiem nepieciešams rūpīgi analizēt pārdošanas rādītājus, vērot izmaiņas klientu vēlmēs un uzvedībā, izprast tendences. Informācijas apjoms ir pamatīgs pat vidēji lielos uzņēmumos. Lai neapjuktu plašajā datu jūklī, viens no uzticamākajiem, dzīvē pārbaudītiem risinājumiem ir centralizēt visu datu plūsmu un uzticēt tās apkopošanu specializētai informācijas apstrādes sistēmai jeb datu noliktavai.

Darba mērķis ir iepazīties ar literatūrā apkopotajām rekomendācijām par datu noliktavu arhitektūras tipiem un izstrādes pieejām, un balstoties uz tām, izstrādāt uzņēmuma datu noliktavas arhitektūru un pašu datu noliktavu.

Bakalaura darbā ir apskatīta vispārēja datu noliktavas arhitektūra un izvēlētais arhitektūras variants konkrētajai datu noliktavai. Ir izanalizēti konkrētās datu noliktavas arhitektūras plusi un mīnusi.

Uzņēmums piedāvā dažādus servisus parādu piedziņas, uzraudzības un rēķinu apstrādes jomā. Katram no šiem servisiem ir sava funkcionalitāte ko nodrošina dažādas informācijas sistēmas. Daudzi no uzņēmuma klientiem izmanto vairākus no šiem servisiem, līdz ar to ienākumi un izmaksas no klienta tiek uzskaitīti dažādās sistēmās. Tādēļ atskaites par kopējiem ieņēmumiem un izdevumiem no konkrētā klienta bija jāveido manuāli, kas ir ļoti laikietilpīgs process. Tika pieņemts lēmums izveidot datu noliktavu, kas apkopotu visus datus par klienta ieņēmumiem un izmaksām no dažādām IS, un automatizēt klienta peļņas un zaudējumu aprēķinus.

1. DN ARHITEKTŪRAS DEFINĪCIJA UN KOMPONENTES

Datu noliktavu un tās arhitektūru lieliski raksturo V. Inmana teiktais, ka datu noliktava pati par sevi ir arhitektūra nevis tehnoloģija. Tehnoloģija ir kā ķieģeļi, bet pilsētas arhitektūru neveido ķieģeļi, bet mājas, kas no tiem būvētas. Tā datu noliktavu veido dažādu elementu un procesu apkopojums, kur katru no tiem nodrošina noteiktas tehnoloģijas.[1]

Arī P. Ponnia salīdzina datu noliktavas arhitektūru ar skolas ēkas arhitektūru. Skolas ēkas arhitektūru neveido tās vizuālais izskats, bet gan klases, kabineti, bibliotēkas, koridori, logi un durvis, jumts un virkne citu komponentu. Arhitektūra ir struktūra, kas apvieno visas šīs komponentes , veidojot skolas ēkas arhitektūru. Pārnesot šo piemēru uz datu noliktavu izriet, ka datu noliktava un tās arhitektūra ir virkne komponentu un to mijiedarbība savā starpā.[2]

Daudz vienkāršāk un tiešāk datu noliktavēšanu un datu noliktavu definē C.Hammergrens un A. Simons. Viņi raksta, ka datu noliktavēšana ir koordinēta, arhitektēta un periodiska datu kopēšana no dažādiem datu avotiem, kā iekšējiem tā arī arējiem, uz vidi, kas ir optimizēta datu analīzei un atspoguļošanai.[3]

Datu noliktavas arhitektūras komponentes var iedalīt divās daļās - aparatūra un datu apstrāde. Katrā no tām var izdalīt sekojošas komponentes:

Aparatūra:

- Datori;
- Operētājsistēmas;
- Datoru tīkls;
- Datu bāzu vadības sistēmas;

Datu apstrāde:

- Datu avoti un datu izlāde;
- Datu ielāde un kvalitāte;
- Datu transformācija, agregācija un aprēķini;

- Datu atainošana;
- Pārvaldības un izstrādes rīki;

Turpmākajās apakšnodaļās tiks aprakstītas pieminētās DN arhitektūras komponentes, kā arī DN arhitektūras tipi atkarībā no izstrādes pieejas un DN datu bāzes iespējamie modeļi.

1.1. Aparatūra

Aparatūras izvēli pārsvarā nosaka sekojoši faktori:

- Datu noliktavas prasības, kas nosaka cik liels datu apjoms ir jāapstrādā noteiktā laikā;
- Esošā uzņēmuma aparatūras un datoru tīkla infrastruktūra, kas iekļauj noteiktas drošības prasības;

Šīs nodaļas mērķis ir aprakstīt nepieciešamās aparatūras vienību skaitu atkarībā no tā, kādas DN komponentes tiks izvietotas uz kopīgiem vai atsevišķiem serveriem vai datoriem.

1.2. Operētājsistēmas

Operētājsistēmas izvēli parasti nosaka jau esošās uzņēmuma infrastruktūras pamatlīnijas un DN izstrādes rīku un DBVS izvēle. Izvēli netieši var ietekmēt arī DN datu avotu izvēlētās OS un DBVS.

1.3. Datoru tīkls

Datoru tīkla infrastruktūras izveidē galvenokārt jāņem vērā, kādi datu apmaiņas veidi un protokoli tiks izmantoti datu apmaiņai starp datu avotiem un datu noliktavu kā arī datu izmantošanai datu atainošanas komponentē.

1.4. Datu bāzu vadības sistēmas

Izvēloties DN datu bāzu vadības sistēmu jāņem vērā sekojoši faktori:

- Esošā uzņēmuma infrastruktūra;
- Datu avotu DBVS;
- DN attīstības plāni un DBVS mērogojamība;
- Datu atainošanai paredzētie rīki un tehnoloģijas;

1.5. Datu avoti un datu izlāde

Šajā stadijā ir rūpīgi jāizplāno izmaiņu iegūšanas stratēģija un ierakstu vēstures uzkrāšanas mehānisms DN. Šeit var izdalīt sekojošus izlādes veidus:

- Datu izlāde notiek reizē ar ielādi Darba apgabalā un izmaiņu iegūšana tiek nodrošināta datu avotā;
- Datu izlāde notiek reizē ar Datu ielādi darba apgabalā un izmaiņu iegūšana tiek veikta Darba apgabalā;
- Datu izlāde notiek uz atsevišķu datu nesēju un izmaiņu iegūšana notiek datu avotā;
- Datu izlāde notiek uz atsevišķu datu nesēju un izmaiņu iegūšana notiek Darba apgabalā;

1.5.1. Izmaiņu iegūšana datu avotā

Pastāv ļoti dažādi veidi izmaiņu vākšanai datu avotu datu bāzēs. Vienkāršākais no veidiem ir izmantot kādu jau eksistējošu datuma lauku, kas tiek izmainīts izmaiņu gadījumā. Piemēram transakcijas datums, rēķina izrakstīšanas datums utt. Šeit gan ļoti rūpīgi jāizvērtē vai tiešām šis lauks maina savu vērtību pie visām ieraksta izmaiņām, kuras vēlamies redzēt.

Ja šāda lauka nav, tad iespējams pievienot speciālus audita laukus, kas tiks aizpildīti gadījumā, ja tiek mainīti kādi mums interesējoši atribūti šajā ierakstā. Vai arī iespējams pievienot audita tabulas, kuras saturēs izmainīto ierakstu primārās atslēgas. Šajā gadījumā gan ir jārēķinās ar papildus funkcionalitātes pievienošanu datu avotā, kas ne vienmēr ir iespējams. Ar šādu pieeju mēs vienmēr iegūsim mainīto ierakstu kopu un ieraksta pēdējo versiju uz izlādes brīdi, ja tas ticis manīts vairāk kārt kopš pēdējās izlādes. Pēc katras datu izlādes ir jāiztīra datu tabulas, kas nozīmē ka konkrētu izlādi nebūs iespējams atkārtot.

Nedaudz sarežģītāks mehānisms ir izmantojot audita tabulas, kas satur ne tikai izmainītā ieraksta atslēgas, bet arī atribūtus, kuru izmaiņas mums interesē vai pilnu ieraksta kopiju. Arī šajā gadījumā jārēķinās ar izmaiņu ieviešanu datu avotā, kā arī rūpīgi jāizvērtē nepieciešamība uzkrāt pilnu atribūtu vēsturi, jo šādas vēstures tabulas var sasniegt lielu datu apjomu. It sevišķi, ja šīs tabulas netiek iztīrītas pēc izlādes, bet tiek izlādēti tikai ieraksti, kas izveidoti pēc pēdējās izlādes.

Vēl sarežģītāks izmaiņu iegūšanas mehānisms ir datu avota datu bāzes žurnālu izmantošana. Šis veids gan ir tehnoloģiski sarežģītāks, bet var atvieglot noslodzi uz datu avota sistēmu.

1.5.2. Izmaiņu iegūšana darba apgabalā

Izmaiņu iegūšana darba apgabalā var tikt lietota papildus iegūtajām izmaiņām datu avotā vai kā vienīgais izmaiņu noteikšanas veids.

Šis izmaiņu veids tiek balstīts uz ierakstu atribūtu vai to pārbaudes summas salīdzināšanu ar datu noliktavā esošajiem ierakstiem.

Izmantojot šādu pieeju mēs varam ielādēt darba apgabalā ierakstus, kas ir izmainīti datu avotā, bet DN tabulā tiks pievienots ieraksts tikai tad, ja ticis mainīts atribūts, ko mēs izmantojam.

Šajā gadījumā gan jārēķinās, ka izmaiņu noteikšana var aizņemt daudz laika un radīt lielu noslodzi uz DN datubāzi.

1.5.3. Datu izlāde tieši darba apgabalā vai failos

Datu izlāde failos palīdz atrisināt datu apmaiņas problēmas gadījumā, ja datu avotos un DN izmantotas dažādas DBVS, kad tiešās sasaistes veidi bieži strādā neefektīvi vai nekorekti. Bet šāda pieeja sniedz arī papildus iespēju veidot izlādēto datu vēsturi, kas ļauj atkārtot datu ielādi darba apgabalā uz konkrēto laika brīdi. Šāds mehānisms ļauj saglabāt pilnu ierakstu vēsturi pat pie pilnīgas DN tabulas pārlādes.

1.6. Arhitektūra atkarībā no pieejas

Veidojot datu noliktavu pārsvarā tiek izmantotas divas atšķirīgas pieejas. Atkarībā no šīs pieejas arī mainās datu noliktavas arhitektūra. Viena no pieejām ir veidot kopēju uzņēmuma datu noliktavu, kas kalpo par datu avotu atsevišķām datuvēm. Otra pieeja ir veidot datuves, kuras apvienotas kopā veido datu noliktavu. Katrai no pieejām ir sekojošas pamata iezīmes:

1.1. tabula

Arhitektūras pieejas

Kopēja uzņēmuma DN	Datuves
• Korporatīva, visa uzņēmuma ietvaros; • Apvieno visus biznesa procesus; • ER bāzēts datu modelis; • Dati tiek saņemti no darba apgabala;	• Nodaļas ietvaros; • Noteiktam biznesa procesam; • Zvaigznes datu modelis; • Tehnoloģija optimizēta datu analīzei;

1.6.1. Lejupejoša jeb stratēģijas virzīta pieeja

Izmantojot lejupejošu pieeju vispirms tiek identificēta biznesa vajadzība un tikai pēc tam tiek izvērtēts, kādās sistēmās nepieciešamie dati ir pieejami un vai vispār šādi dati ir pieejami kādā no uzņēmuma IS. Šādai pieejai ir labi definēti datu noliktavas mērķi un tās datu modelis tiek būvēts atkarībā no risināmās biznesa problēmas, nevis pieejamajiem datiem. Tādējādi datu sasaiste un transformēšana ir daudz sarežģītāka kā augšupejošai jeb datu virzītai pieejai. Izmantojot lejupejošu stratēģiju DN funkcionalitāte ir daudz labāk definēta kā augšupejošā.[4]

Šādai pieejai var izdalīt sekojošus plusus un mīnusus: [2]

Plusi:

- Kopējs skatījums uz visiem uzņēmuma datiem;
- Atsevišķi projektēta un nav vairāku datuvju apvienojums;
- Kopēja centrāla datu uzglabāšana;
- Centralizēta kontrole;
- Iespējams sasniegt ātrus rezultātus, ja tiek izstrāde sadalīta iterācijās;

Mīnusi:

- Nepieciešams ilgs laiks, lai pilnībā uzbūvētu;
- Pastāv augsts neizdošanās risks;
- Nepieciešama augsta līmeņa un vairāku prasmes;
- Nepieciešami lieli izdevumi bez koncepcijas pierādījumiem;

1.6.2. Augšupejoša jeb datu virzīta pieeja

Augšupejošai pieejai kā pamats tiek ņemts tas, ka eksistē noteikti dati, kas var tikt izmantoti, lai uzlabotu biznesa lēmumu pieņemšanu. Izmantojot šo pieeju datu transformācijas ir daudz vienkāršākas un datu sasaiste un ielāde ir veicama daudz vienkāršāk kā lejupejošā pieejā. Bet pastāv liels risks, ka dati, kas tiks sagatavoti, nemaz nav nepieciešami, jo nav precīzi definētas vajadzības. [4]

Šai pieejai var izdalīt sekojošas pozitīvās un negatīvās puses: [2]

Plusi:

- Salīdzinoši ātra un viegla implementācija;
- Samērā ātra ieguldījumu atdeve;
- Zems neizdošanās risks;

- Iespēja sadalīt implementāciju un sākt ar kritiskākajām datuvēm;
- Projekta komandai ir iespēja mācīties un uzkrāt zināšanas;

Mīnusi:

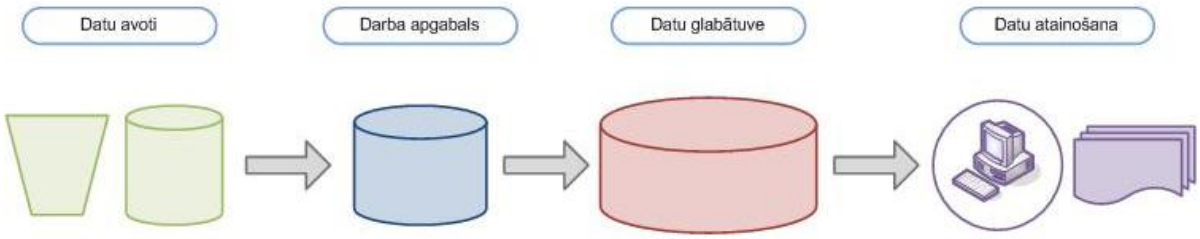
- Datuvei ir ierobežots skats uz datiem;
- Palielinās lieko datu apjoms;
- Dati var būt pretrunīgi un nesavienojami;

1.7. DN datu bāzes arhitektūra

Datu noliktavās var izdalīt divus galvenos datu modeļu veidus - dimensiju jeb datuvju modelis un relāciju modelis. Parasti datu modeļa izvēle ir atkarīga no izvēlētās datu noliktavas pieejas. Ja tiek izvēlēta lejupejoša pieeja, tad datu noliktavā tiek izmantots relāciju modelis. Bet ja tiek izmantota augšupejoša pieeja, tad parasti tiek izmantots dimensiju modelis. [1]

1.7.1 Relāciju modelis

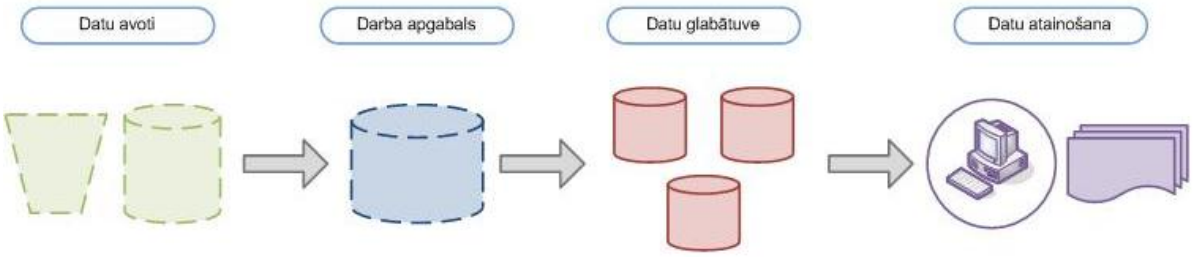
Šis modelis tiek parasti izvēlēts lietojot lejupejošu arhitektūras pieeju. Izmantojot šo modeli dati ir atomāri un tiek glabāti trešajā normālformā ar zemāko iespējamo granularitāti. Vaicājumiem un datu analīzei tiek izmantots šis datu modelis un netiek veidotas papildus datuves vai citas datu struktūras.[2]



1.1. att. **Relāciju modelis**

1.7.2. Dimensiju modelis

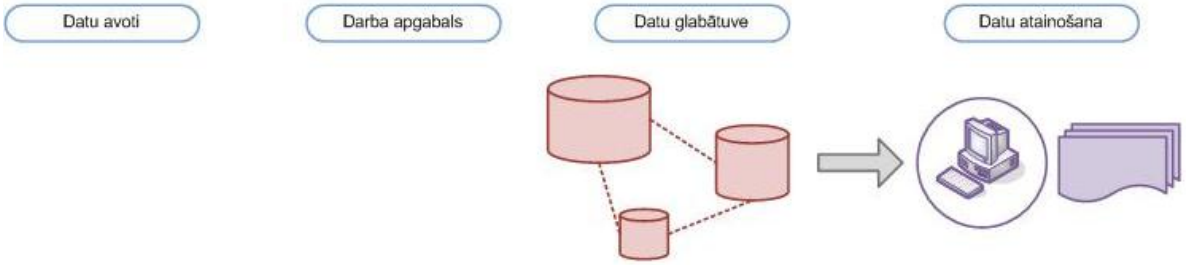
Šis datu modelis parasti tiek izmantots izmantojot augšupejošu arhitektūras pieeju, kad kompānijas struktūrvienības izstrādā atsevišķas datuves, lai analizētu viņām pieejamos datus. Katra no šīm datuvēm nodrošina specifiskas biznesa vajadzības un nekalpo kā vienots datu analīzes datu avots. Šīs datuves ir savā starpā neatkarīgas un ir grūti tās koplietot, lai analizētu datus, kas būtībā savā starpā ir saistīti.[2]



1.2. att. **Dimensiju modelis**

1.7.3 Apvienoto datu avotu modelis

Ir situācijas, kad kompānijai jau ir esošas datu kopas, kas tiek izmantotas datu analīzei. Ne vienmēr ir nepieciešams veidot jaunas datu izlādēs un ielādes komponentes. Var lietot jau esošās datu kopas vai dautuves un sasaistīt tās ar vienotām atslēgām un globāliem vaicājumiem. Ar šo pieeju netiek veidota ne kopēja datu noliktava ne datuves. Šajā gadījumā gan rūpīgi jāizvērtē tehniskās iespējas un apstrādājamais datu apjoms.[2]

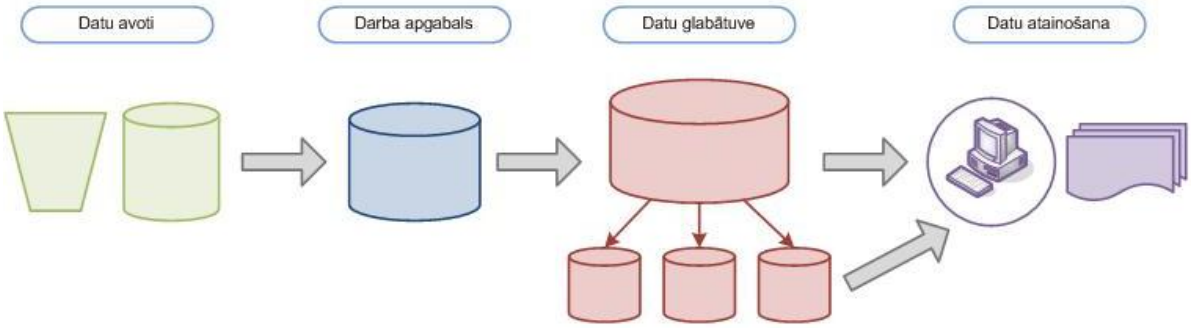


1.3. att. **Apvienoto datu avotu modelis**

1.7.4. Rumbas un spieķu modelis

Šis ir modelis, kura pamatā ir relāciju modelis, kur centralizēti trešajā normālformā glabājas atomāri dati. Tā galvenā atšķirība ir tā, ka tas satur arī savā starpā neatkarīgas datuves, kuru datu avots ir centralizētā datu noliktava. Šeit centralizētā DN ir kā rumbas, kas nodod tālāk datus pa spieķiem, kas ir datuves.

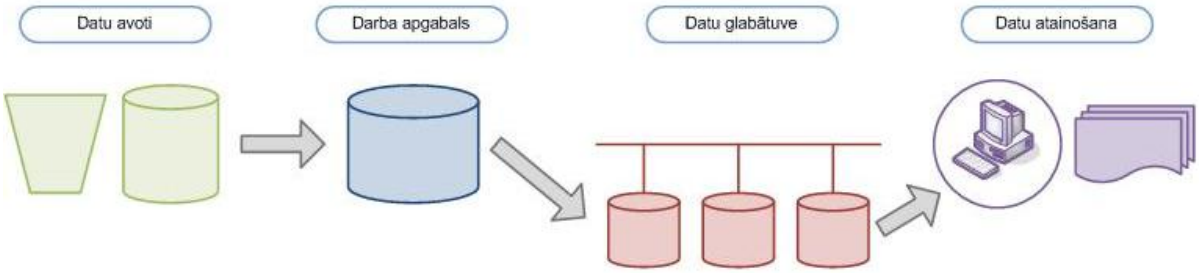
Datuves tiek veidotas dažādiem mērķiem, kā piemēram, departamentu datu analīzei, datu izracei vai specializētiem pieprasījumiem. Katra datuve satur denormalizētus, agregētus datus multidimensionālā struktūrā. Pielietojot šādu modeli dati var tikt analizēti tieši datu noliktavā vai kādā no specializētajām datuvēm.[2]



1.4. att. **Rumbas un spieķu modelis**

1.7.5. Apvienoto datuvju modelis

Šim modelim par pamatu tiek ņemts dimensiju modelis. Veidojot šo modeli tiek izveidota viena galvenā datuve ar dimensijām, kas tiks izmantotas citās datuvēs. Galvenā šāda modeļa ideja ir izmantot vienas un tās pašas dimensijas dažādās datuvēs, kas ļauj analizēt datus daudz augstākā līmenī, galvenajā datuvē. Šo modeli parasti pielieto izmantojot augšupejošu DN izstrādes pieeju.[2]



1.5. att. **Apvienoto datuvju modelis**

1.8. Datu ielāde un kvalitāte

Datu kvalitātes pārbaudes parasti tiek veiktas jau datu avotā vai pēc datu ielādes darba apgabalā. Šeit varētu izdalīt trīs galvenos datu pārbaudes virzienus:[5]

- Kolonnu pārbaude - tā ir vienkārša kādas kolonnas datu atbilstības pārbaude. Piemēram, pārbaude vai lauks vienmēr ir aizpildīts, vai vērtības ir noteiktā diapazonā vai formātā;
- Struktūras pārbaude - tā ir datu integritātes pārbaude, kur tiek noteikts vai atbilstošā ieraksta ārējās atslēgas ir sastopamas saistītajās tabulās un hierarhiju struktūrās;

- Biznesa noteikumu pārbaude - tā ir datu pārbaude pēc sarežģītiem nosacījumiem, kas var iekļaut vairākas pirmo divu tipu pārbaudes, kas nosaka ieraksta atbilstību noteiktai biznesa loģikai;

1.9. Datu transformācija, agregācija un aprēķini

Nelielas datu transformācijas parasti tiek veiktas visos datu apstrādes posmos, bet galvenās no tām tiek veiktas transformējot datus biznesa datu modelī, jeb izpildot prasībās noteiktās datu kalkulācijas un agregācijas. Šo transformāciju mērķis ir datu apvienošana un pielāgošana kopējiem standartiem.

Šā darba mērķis nav aprakstīt konkrētus transformāciju tipus. Galvenie jautājumi, kas jāapskata no arhitektūras puses ir, kur tiks veiktas transformācijas un kā tās tiks definētas. Abi šie jautājumi parasti tiek izskatīti kontekstā ar ETL rīku izvēli un to funkcionalitāti.

1.9.1. Transformēšanas pieeja

Pārsvarā tiek izmantotas divu veidu pieejas:

- ETL pieeja, kad transformācijas tiek veiktas uz ETL servera. Izmantojot šādu pieeju dati tiek izgūti no datu avotiem, transformēti izmantojot ETL servera resursus un ielādēti mērķa tabulās vai failos. Tādējādi tiek atvieglota noslodze uz datu avotu un mērķa datubāzi. Visbiežāk šī pieeja tiek izmantota kopā ar ETL rīkiem, kas nodrošina transformāciju veikšanu ETL servera atmiņā, nevis veicot datu operācijas datu bāzē.
- ELT pieeja, kad datu transformācijas tiek veiktas mērķa datu bāzē. Izmantojot šo pieeju dati tiek transformēti jau izmantojot mērķa datu bāzes servera resursus. Šajā gadījumā nav nepieciešams papildus ETL serveris uz kā veikt transformācijas. Visbiežāk šī pieeja tiek izmantota kopā ar ETL rīkiem, kas strādā ka koda ģeneratori un transformē datu transformāciju kartējumus datu bāzē izpildāmos skriptos;

Šīs ir divas visizplatītākās pieejas, bet transformācijas var tikt veiktas arī datu avotu datu bāzē pirms izlādes vai speciālā transformāciju datu bāzē. Ir pieejami arī ETL rīki, kas nodrošina datu kartējumu izpildi abos no iepriekš minētajiem variantiem. Tādējādi vajadzīgā pieeja un rīki jāizvēlas izejot no pieejamajiem resursiem datu avotu un mērķa datubāzu serveros.

1.9.2. Transformāciju definīcija

Atkarība no tā, kādas DBVS tiek izmantotas un kādi rīki ir pieejami, jāizlemj vai datu transformāciju definēšanai tiks izmantots kāds no ETL rīkiem.

Atsevišķos gadījumos, ja datu avoti izmanto to pašu DBVS un datu transformācijas var tikt izpildītas uz kāda no DB serveriem, datu transformācijas var tikt izstrādātas kā datu bāzē izpildāmi skripti, neizmantojot ETL izstrādes rīkus.

Vairumā gadījumu gan tiek izmantots kāds no ETL rīkiem, kas atvieglo kartējumu definēšanu un nodrošina citu papildus funkcionalitāti, kā piemēram, komandas darbu, versiju kontroli un ETL procesu definēšanu un izpildi.

1.10. Datu atainošana

Datu atainošanas komponentes arhitektūra atkarīga no daudziem faktoriem. Piemēram, kādas atskaites tiks veidotas izmantojot par avotu DN, tās būs vienkāršas statiskas atskaites, vai tiks izmantoti biznesa inteliģences vai datizraces rīki, vai atskaites tiks sagatavos IT komanda un nodos lietotājiem, vai lietotājiem būs iespēja veikt pieprasījumus tieši DN.

Lielākoties tiek izmantots kāds no biznesa inteliģences rīkiem, kas nodrošina dažādas informācijas analīzes un atainošanas iespējas:[6]

- Atskaišu izveide un darbināšana;
- Atskaišu nosacījumu mainīšana (filtri);
- Atskaišu publicēšana vai automātiska saņemšana;
- Atskaišu bibliotēka ērtai pārskatīšanai;
- Atskaišu izveidošana pie iepriekš noteiktu nosacījumu izpildes;

No arhitektūras viedokļa šeit var izdalīt divas atšķirīgas pieejas:

- Pieprasījumu bāzēti BI rīki. Tie ir BI rīki, kas datu atainošanai veido pieprasījumus DN datubāzei un izpilda to informācijas pieprasījuma brīdī. Šādu pieeju izmanto tādi BI rīki kā Oracle Business Inteligence un SAP Business Objects
- Atmiņā strādājoši BI rīki. Tie ir rīki, kas ielādē atmiņā nepieciešamos datu apgabalus un veic datu pieprasījumus tieši no atmiņas. Šādu pieeju izmanto tādi BI rīki kā Qlikview un Cognos BI.

1.11. DN pārvaldības un Izstrādes rīki

Veidojot datu noliktavu parasti tiek izmantoti rīki, kas atvieglo DN procesu pārvaldību un izstrādi.

1.11.1. Procesu kontroles rīki

Šie rīki tiek izmantoti, lai atvieglotu ielādes procesu definīciju, kārtošanu un izpildi.

Galvenās šo rīku funkcija ir:

- Izpildāmo procesu definēšana;
- Izpildes kārtības nodrošināšana;
- Automātiska kārtības noteikšana atkarībā no definētas mērķa tabulu atkarības;
- Paralēla vai virknes izpilde;
- Kārtība atkarībā no uzdevumu izpildes rezultāta;
- Automātiska procesu izpilde pēc noteikta grafika;
- Procesu izpildes žurnalēšana;

Procesu izpildes pārvaldībai var izmantot šos rīkus vai arī veidot konkrētai DN pielāgotus procesu izpildes risinājumus.

1.11.2. ETL rīki

ETL rīku galvenais uzdevums ir atvieglot datu kartējumu definēšanu starp datu avotiem un mērķa tabulām un to metadatu pārvaldību. Kā arī vajadzības gadījumā nodrošināt komandas izstrādes funkcionalitāti un versiju kontroli.

ETL rīkus var iedalīt divās kategorijās:

- ETL rīki, jeb rīki, kas veic datu izlādi no datu avota un datu transformācijas atmiņā un veic datu ielādi mērķa datu bāzē. Izmantojot šos rīkus galvenā noslodze ir uz ETL serveri, uz kura tiek izpildītas visas datu transformācijas. Pēc šāda darbības principa darbojas Informatica PowerCenter;
- ELT rīki, jeb rīki, kas datu transformācijas veic datu avota un/vai mērķa datubāzē. Šie rīki parasti darbojas kā skriptu ģeneratori un konvertē definētos datu kartējumus DBVS saprotamā skriptā, kas nodrošina kartējumā definētās datu transformācijas. Izmantojot šos rīkus parasti vislielākā noslodze ir uz mērķa datubāzes serveri, kur tiek izpildītas datu transformācijas. Pēc šāda darbības principa darbojas Oracle Warehouse Builder un Oracle Data Integrator;

1.11.3. Sasaistes rīki

Sasaistes rīki parasti ir arī daļa no ETL rīku funkcionalitātes, kas nodrošina darbu ar datu avotiem.

Galvenās funkcijas šādiem rīkiem ir:

- Sasaistes konfigurācijas glabāšana;
- Datu avota definīcijas transformācija atkarībā no datu avota tipa un DBVS;
- Datu tipu transformācijas;
- Dažādu kodējumu datu izgūšana;

1.11.4. Datu kvalitātes rīki

Datu kvalitātes rīki domāti, lai atvieglotu datu kvalitātes pārbaudes un datu labošanu. Datu kvalitātes rīkus var izmantot gan vienreizējai datu analīzei, lai identificētu nepieciešamās datu transformācijas pie datu izlādes vai ielādes, kā arī ikdienas lādējamo datu analīzei.

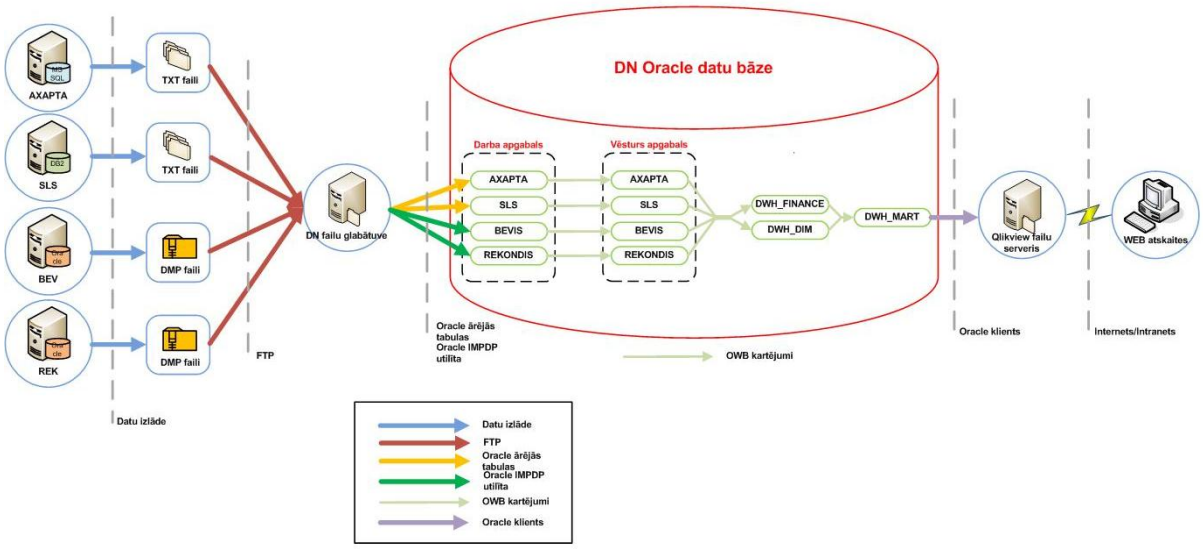
Galvenās funkcijas šiem rīkiem ir:

- Nekvalitatīvu datu identificēšana un atfiltrēšana;
- Automātiska identificēto problēmu labošana pēc iepriekš definētiem noteikumiem;
- Datu profilēšana;

2. UZŅĒMUMA DN ARHITEKTŪRA

Autors pievienojās konkrētās datu noliktavas izstrādei kā ETL izstrādātājs un izstrādāja datu ielādes procesus no vairākiem datu avotiem. Autors piedalījās arī DN arhitektūras definēšanā un DN datubāzes modelēšanā. Uz doto brīdi autors ir šīs DN arhitekts un nodarbojās ar sistēmas tālāku attīstību un izstrādi.

Attēlā nr.2.1. redzama uzņēmuma datu noliktavas augsta līmeņa arhitektūra. Īsumā raksturojot šo DN, tiek izmantoti dati no 4 dažādiem datu avotiem, kuri piegādā datu arhīvus uz DN failu glabātuvi. Tālāk dati tiek ielādēti datu noliktavā un sagatavoti datu atainošanai. Pēc tam dati tiek ielādēti biznesa inteliģences rika bināros failos, no kurienes ari tiek atainoti.



2.1. att. **DN arhitektūra**

2.1. Aparatūras komponentes

2.1.1. Datori, Serveri

Tā kā par datu izlādi atbild datu avotu sistēmu uzturētāji DN komandai nav precīzu ziņu par datu avotu serveru konfigurāciju un jaudu.

DN izmanto failu glabātuvi, kas atrodas uz servera ar sekojošu konfigurāciju:

2.1.tabula

Failu glabātuves servera konfigurācija

Vārds:	SRV2
Vide:	Izstrāde
Procesors:	8x2.33Ghz
Oracle licences:	4EE
Operatīvā atmiņa:	64GB
Cietais disks:	12TB

Šis pats serveris tiek izmantots arī kā izstrādes vide uz kura atrodas izstrādes DN datubāze.

DN produkcijas datubāzes serverim ir sekojoša konfigurācija:

2.2.tabula

DN servera konfigurācija

Vārds:	SRV1
Vide:	Produkcija
Procesors:	16x3Ghz
Oracle licences:	8EE
Operatīvā atmiņa:	140GB
Cietais disks:	3TB

Lai atainotu datus Qlikview izmanto sekojošus serverus:

2.3.tabula

Qlikview serveru konfigurācija

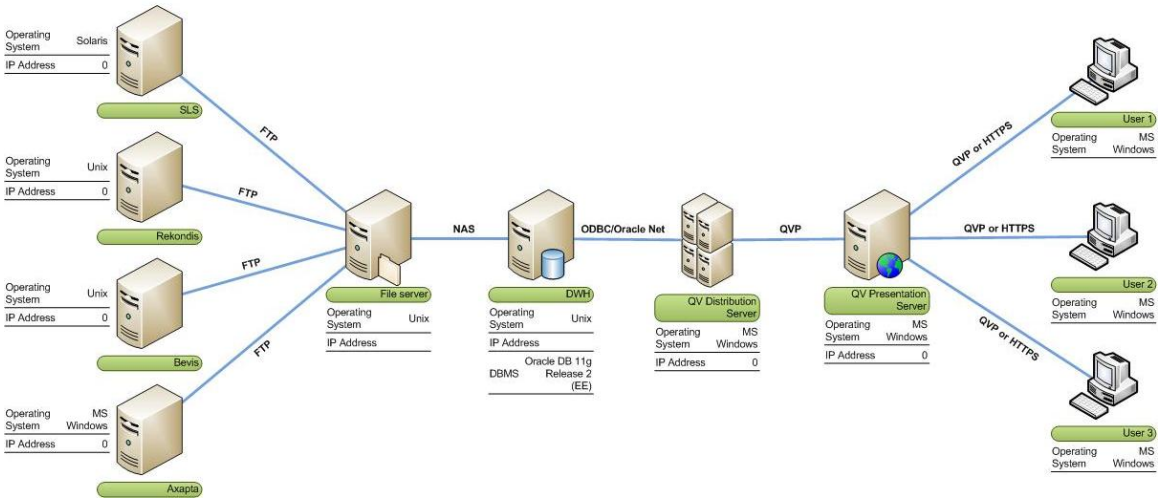
Vārds:	SRV3	SRV4
Vide:	-	-
Procesors:	2x2.33 GHz	2x3.07GHz
Oracle licences:	-	-
Operatīvā atmiņa:	11.4 GB	384 GB
Cietais disks:	10TB	10TB

SRV3 ir sadales serveris, kas nodrošina datu ielādi Qlikview failos un Qlikview aplikācijas procesus.

SRV4 ir prezentācijas serveris, kas nodrošina Qlikview failu izmantošanu WEB bāzētās atskaitēs.

2.1.2. Operētājsistēmas

Kā redzams attēlā nr.2.2. DN arhitektūrā ir pārstāvētas dažādas OS. DN datu avotu sistēmas izmanto gan Windows gan Unix bāzētas OS. Datu noliktavas failu glabātuve un datu bāzes serveri izmanto Unix operētājsistēmu. Qlikview serveris un Qlikview Publisher serveris izmanto Windows server 2003 un Windows server 2008 operētājsistēmas. Gala lietotāji izmanto Windows bāzētas OS.



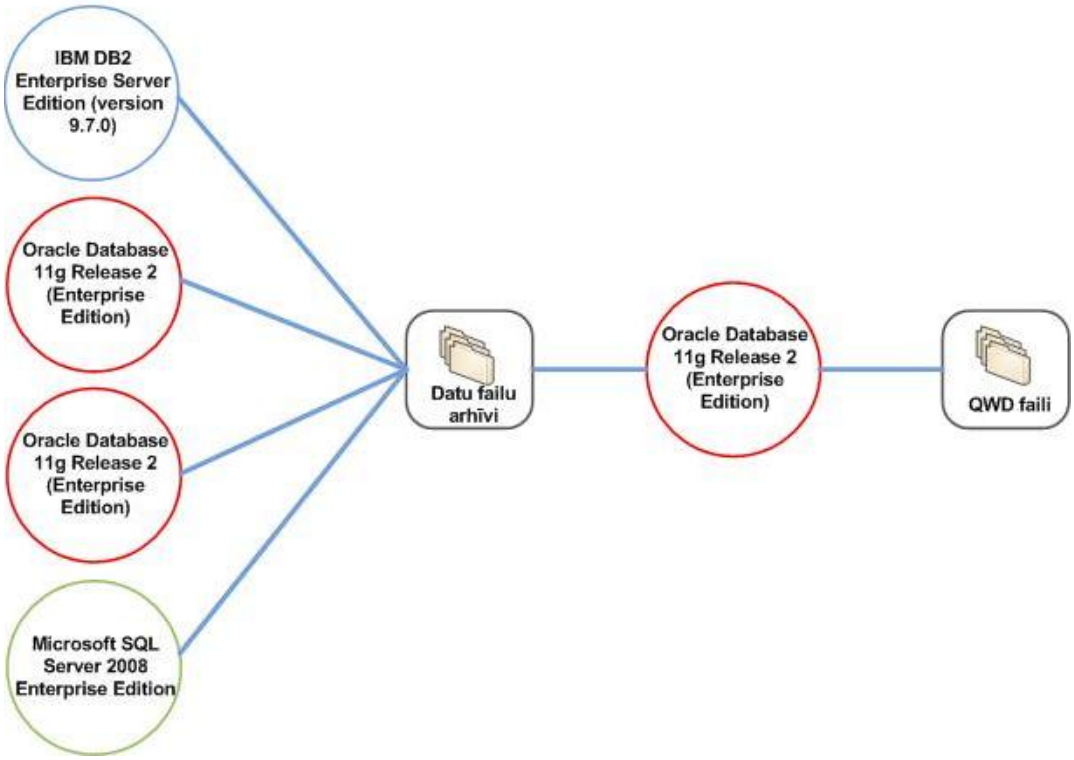
2.2. att. **Izmantotās operētājsistēmas**

2.1.3. Datoru tīkls

Kā redzams attēlā nr.2.2., faili no datu avotu datu bāzu serveriem tiek kopēti uz DN failu glabātuvi izmantojot FTP protokolu. Tālāk datu faili tiek kopēti no DN failu glabātuves uz datu bāzu serveri izmantojot piemontētas direktorijas. Qlikview serveris pieslēdzās pie DN izmantojot ODBC draiveri. Datu apmaiņa starp Qlikview serveriem notiek izmantojot savu QVP protokolu, bet lietotāji saziņai ar Qlikview serveri izmanto pārsvarā HTTPS protokolu.

2.1.4. Datu bāzu vadības sistēmas

Projektā ir pārstāvētas dažādas datu bāzu vadības sistēmas. Datu avoti izmanto sekojošas DBVS: Oracle DB 11g, MS SQL Server2008, DB2.



2.3. att. **Izmantotās DBVS**

Datu avotu datu bāzes izvēli pārsvarā nosaka IS izstrādātājs. Līdz ar datu avotiem, kurus izstrādā pats uzņēmums, tiek izmantota Oracle DB DBVS, bet pārējiem atkarībā no IS piegādātāja izvēles.

Datu noliktavas datubāze ir realizēta kā Oracle DB 11g datubāze. Šī DBVS tika izvēli galvenokārt ietekmēja uzņēmuma esošā infrastruktūra. Uzņēmums savu informācijas sistēmu izstrādei pārsvarā izmanto Oracle datu bāzes. Kā arī izvēli ietekmē fakts, ka datu avoti, no kuriem tiek lietots visvairāk datu , izmanto Oracle DB DBVS.

2.2. Datu izguve un datu avoti

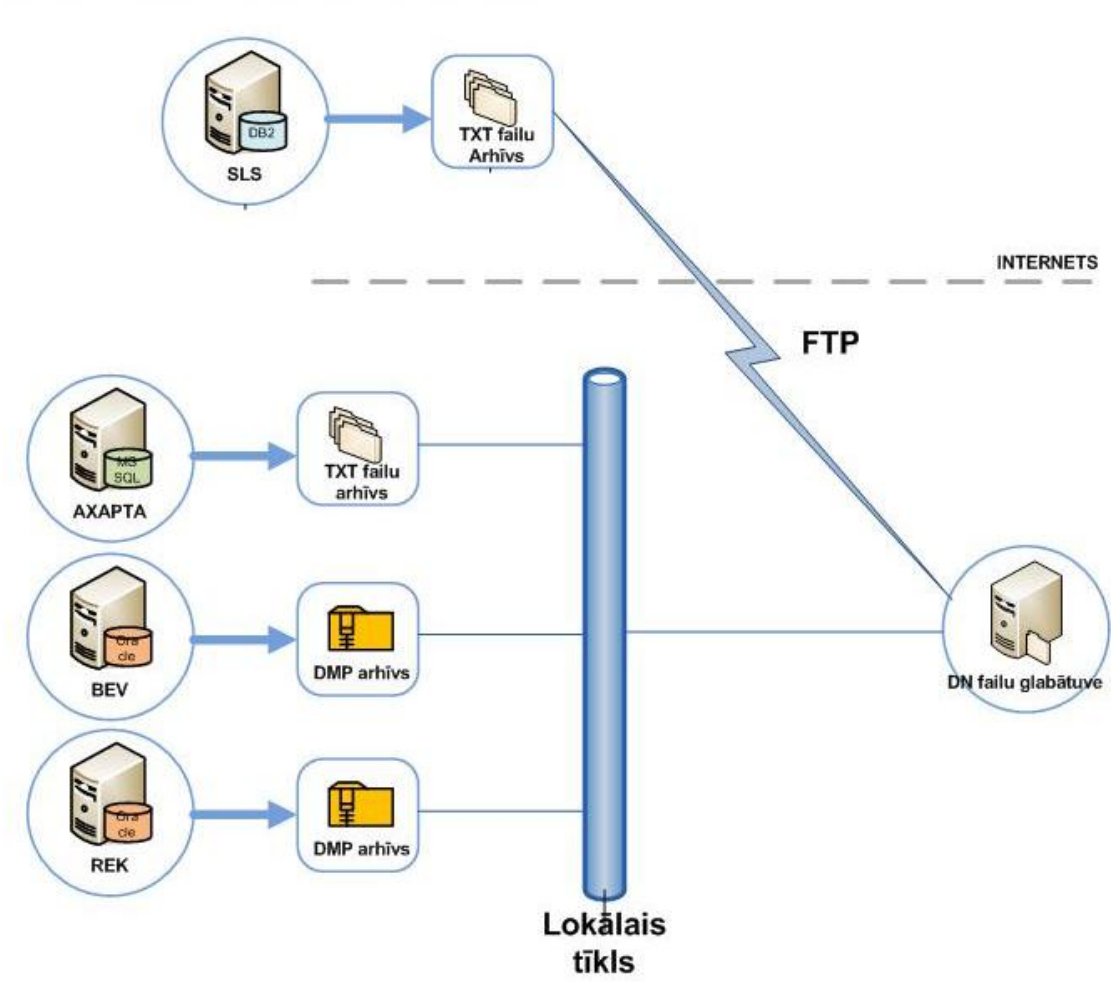
Datu noliktavā tiek izmantoti dati no četriem dažādiem datu avotiem.

2.4.tabula

Datu avoti

Datu avots	Kods	Apraksts
Rekondis	REK	Kredītu piedziņas sistēma
Bevis	BEV	Kredītu uzraudzības sistēma
SLS	SLS	Pārdošanas sistēma
Axapta	AXA	Klientu pārvaldības sistēma

Kā redzams attēlā nr.2.4., dati tiek izlādēti, saarhivēti un nogādāti DN failu glabātuvē.



2.4. att. **Datu izguve**

2.2.1. REKONDIS

REKONDIS ir sistēma, kas nodrošina parādu piedziņas procesu un satur informāciju par klientu, parādnieku, parādu un tā piedzīšanas aktivitātēm.

Sistēmas dati tiek glabāti Oracle datu bāzē. Datu izlāde tiek lietota Oracle datubāzes utilītprogramma expdp. Sagatavotais datu fails tiek arhivēts un pārvietots uz datu noliktavas failu glabātuvi.

Datu izlādi un failu piegādi nodrošina REK sistēmas uzturēšanas komanda. Datu izlādē iekļautās tabulas nosaka DN izstrādes komanda.[2.Pielikums]

2.2.2. BEVIS

BEVIS ir parādu uzraudzības sistēma, Ja parāds netiek piedzīts noteiktajā laika periodā, tad tas tiek pārvietots no parādu piedziņas sistēmas uz parādu uzraudzības sistēmu.

Sistēmas dati tiek glabāti Oracle datu bāzē. Datu izlāde tiek lietota Oracle datubāzes utilītprogramma expdp. Sagatavotais datu fails tiek arhivēts un pārvietots uz datu noliktavas failu glabātuvi.

Datu izlādi un failu piegādi nodrošina BEV sistēmas uzturēšanas komanda. Datu izlādē iekļautās tabulas nosaka DN izstrādes komanda. [2.Pielikums]

2.2.3. SLS

SLS nodrošina papildus servisus, kas saistīti ar parādu piedziņu, tajā skaitā klienta pārdošanas datu, rēķinu izrakstīšanas un atgādinājumu apstrādi.

Sistēmas dati tiek glabāti DB2 datu bāzē. Datu izlāde notiek uz teksta failiem. Katrs fails satur vienas tabulas datus ar fiksēta garuma laukiem. Teksta faili tiek arhivēti un pārvietoti uz DN failu glabātuvi.

Datu izlādi un piegādi uz DN failu glabātuvi nodrošina SLS uzturēšanas komanda. Datu izlādē iekļautās tabulas nosaka DN izstrādes komanda. [2.Pielikums]

2.2.4. AXAPTA

AXAPTA nodrošina klientu attiecību menedžmenta funkcijas un kalpo kā galvenais datu avots klientu informācijai.

Sistēma glabā datus MS SQL Server datu bāzē. Datu izlāde tiek veikta teksta failos. Katrs fails satur datus no vienas tabulas ar fiksēta garuma laukiem.

Datu izlādi un piegādi uz DN failu glabātuvi nodrošina AXAPTA uzturēšanas komanda. Datu izlādē iekļautās tabulas nosaka DN izstrādes komanda. [2.Pielikums]

2.2.5. Datu izlādes tipi

Datu izlāde nodrošina gan pilnu datu izlādi, gan datu izmaiņu izlādi. Katru dienu tiek sagatavots datu arhīvs, kas satur visus datus no izvēlētajām tabulām, kā arī datu arhīvs, kas satur izmainītos datus, kopš pēdējās datu izlādes.

Šāda situācija ir izveidojusies vēsturiski, jo sākotnēji tika ieviesta datu izlāde visiem tabulas datiem. Tā kā, augot datu apjomam, iepriekšējais DN risinājums nevārēja apstrādāt pilnu datu apjomu atvēlētajā laika sprīdī, tika ieviesta datu izlāde tikai izmainītajiem datiem. Tā kā datu izlādei atvēlētais laika sprīdis un resursi ir pietiekami, lai veiktu abu veidu datu izlādes, tad tika saglabāta arī pilnu tabulu izlāde.

Plusi:

- Izmantojot šādu pieeju pie DN ielādes ir iespēja noteikt kādu no datu failiem izmantot. Tādējādi jebkurā brīdī ir iespēja pārslēgties no pilnas ielādes uz inkrementālu un otrādi. Gadījumā, ja datu arhīvi par kādu no dienām ir bojāti, vai kādu iemeslu dēļ nav saņemti, tad nākamais inkremtālais datu arhīvs vairs nesaturēs datus par šo dienu, bet pilnais datu arhīvs saturēs visus datus, ko varēs izmantot pie nākamās ielādes. Tādējādi iespējams tiek samazināta iespēja pazaudēt datus un to vēsturiskumu;
- Izmantojot inkrementālo datu arhīvu iespējams paātrināt datu ielādi;

Mīnusi:

- Šāda pieeja pagarina datu izlādes laika sprīdi un rada papildus noslodzi uz datu avotu;
- Nepieciešama papildus disku vieta, lai glabātu abu veidu arhīvus;

2.3. DN failu glabātuve

Šī arhitektūras komponente ir iekļauta galvenokārt divu iemeslu dēļ:

- Lai atrisinātu datu pārnešanu starp dažādām DBVS;
- Lai atkārtoti pielietotu no datu avotiem jau izlādētus datus un neveidotu papildus noslodzi uz datu avotiem;

Tā kā datu izlāde un ielāde tiek nodrošināta caur datu failiem, tad tiek uzturēta DN failu glabātuve, kur tiek glabāti visi datu avotu datu faili par pēdējo gadu. Tas nodrošina pilnu datu vēsturiskuma saglabāšanu par pēdējo gadu DN pilnīgas pārlādes gadījumā. Salīdzināmā ar datu ielādi tieši no datu avota šādai pieejai ir sekojoši plusi un mīnusi:

Plusi:

- Nodrošina datu vēsturiskumu par pēdējo gadu, ja nepieciešama DN pārlāde;
- Pieejama vēsture atribūtiem, kas ir iekļauti datu failos, bet nav iekļauti DN ielādē;
- Nav nepieciešama sasaiste starp dažādām heterogēnām datu bāzēm;
- DN nav nepieciešama piekļuves tiesības datu avotu sistēmām;

Mīnusi:

- Datu izlāde ir pilnībā atkarīga no datu avotu uzturēšanas komandām;
- Papildus izmaksas par datu nesējiem;
- Papildus tīkla noslodze;

- Papildus laiks datu izlādei, failu piegādei un ielādei;

2.4. DN darba apgabals

Visa datu ielāde tiek piesaistīta konkrētam datumam, par kuru tiek ielādēti dati. Ielādējot datus kā parametrs tiek norādīts datums par kuru tiks lādēti dati. Tad tiek nokopēti un atarhivēti datu faili, kas satur datus uz šo datumu.

Regulārā ielādē darba dienās tiek izmantoti inkrementālie datu arhīvi, bet sestdienās tiek izmantoti pilnie datu arhīvi. Tā kā dati tiek lādēti par iepriekšējo dienu, tad datu ielāde nenotiek svētdienās un pirmdienās, jo tiek uzskatīts, ka datu avotos nenotiek izmaiņas.

Ielādējot datus šajā apgabalā tabulas tiek papildinātas ar sekojošiem sistēmas laukiem:

2.5.tabula

Sistēmas lauki

DW_SOURCE_ID	Lauks satur datu avota identifikatoru
DW_LOAD_ID	Ieraksta surogātatslēga
DW_LOAD_DATE	Datums par kuru tiek veikta ielāde
DW_CREATED	Ieraksta pievienošanas datums
DW_CHECKSUM	Ieraksta pārbaudes summa

Vislielāko uzmanību šeit jāpievērš atribūtam DW_CHECKSUM, kas tiek aprēķināts, lai izskaitļotu izmaiņas. Šādā veidā tiek iegūta ieraksta vēsture tikai pēc noteiktiem atribūtiem.

Pārbaudes summas aprēķina piemērs:

```
W_ADMIN.UTIL.GET_CHECKSUM(
  TO_CHAR(INGRP1.COLLECTOR_GROUP)  ||
     TO_CHAR(INGRP1.DEPT_NO)  ||
      INGRP1.EMAIL)
```

Šajā piemērā redzams, ka pārbaudes summa tiek izskaitļota no trīs atribūtiem. Ierakstiem ar vienādām šo atribūtu vērtībām šī summa būs vienāda, bet ja kāds no tiem atšķirsies, tad arī pārbaudes summa būs dažāda. Pārbaudes summas aprēķinā var tikt iekļauti visi atribūti, vai arī tikai tie, kuru vēsturiskās izmaiņas ir nepieciešams atspoguļot DN. Tādējādi, lai noteiktu vai ierakstā ir mainījušies atribūti, kuru vēsturiskums jāatspoguļo DN, nav nepieciešams salīdzināt visus nepieciešamos atribūtus, bet pietiek salīdzināt tikai ieraksta

atslēgu un tā pārbaudes summu. Ja atslēga un pārbaudes summa sakrīt, tad tiek uzskatīts, ka ieraksts nav mainījies, bet ja atslēga sakrīt un pārbaudes summa nesakrīt, tad tiek uzskatīts, ka ieraksts ir izmainīts un tiek pievienots jauns ieraksts vēstures apgabalā.

Ar pilnu funkcijas W_ADMIN.UTIL.GET_CHECKSUM tekstu var iepazīties pielikumā.[1.Pielikums]

Šajā apgabalā arī tiek pārbaudīta datu integritāte un tālāk datu uzglabāšanai tiek ielādēti tikai ieraksti, kuru vecāku ieraksti jau ielādēti vēstures apgabalā. Pārējie ieraksti tiek ignorēti un tiek piereģistrēta datu kļūda sistēmas tabulā.

Tā kā dati tiek lādēti arī no teksta failiem, tad ir iestrādātas lauku datu pārbaudes un datu formatēšana.

2.5. Datu vēstures apgabals

Šajā datu noliktavas apgabalā datiem ir tāda pati struktūra kā darba apgabalā. Šis apgabals satur tās pašas datu tabulas kā darba apgabals, tikai tajās tiek glabāta vēsturiska informācija.

Lai identificētu pēdējo aktuālo ierakstu tiek izmantota tabulas, kas satur unikālu surogātatslēgu un pēdējo ielādes datumu.

Piemēram:

2.6.tabula

Datu tabula

CLIENT_ID	CLIENT_NAME	EMAIL	DW_SOURCE_ID	DW_LOAD_ID	DW_LOAD_DATE	DW_CREATED	DW_CHECKSUM
1	Company1	c1@test.com	1	1909923569	1/10/2011	26/11/2012 11:01:37	3998360A515872E883C0606DF6EA4E04
2	Company2	c2@test.com	1	1909923570	1/10/2011	26/11/2012 11:01:38	D830DBA7392B60301273190D79C6DD82
1	Company1-1	c1@test.com	1	1909923571	2/10/2011	27/11/2012 11:01:39	B52FF3DD7C74A8647E8C1A6591A1FD3C

2.7.tabula

Tekošo ierakstu tabula

CLIENT_ID	DW_LOAD_DATE_MAX
1	2/10/2011
2	1/10/2011

Ielādējot datus vēstures apgabalā tiek salīdzinātas naturālās atslēgas un pārbaudes summas ierakstam, kas tiek lādēts un tekošajam ierakstam no vēstures apgabala. Ja vēstures apgabalā nav ieraksta ar šādu naturālo atslēgu un pārbaudes summu, tad ieraksts tiek pievienots vēstures apgabalam.

Piemēram:

2.8.tabula

Datu tabula pirms ielādes

ID	Name	EMAIL	DW_CHECKSUM
1	Company1	c1@test.com	979B902C58CF091F5235576185C83665
2	Company2	c2@test.com	B4AA8C76F7780DCE2D8518A8D7E1DB8E
3	Company3	c3@test.com	913923C0F64712F3C2FF22E41229BE04
4	Company4	c4@test.com	BA1267391758EC1EAC5AD46D9DF21287

2.9.tabula

Ienākošie ieraksti

ID	Name	EMAIL	DW_CHECKSUM
1	Company1	c1@test.com	979B902C58CF091F5235576185C83665
2	Company2-2	c2@test.com	520D61DE63DDE0F1B71E558E6117EA7A
3	Company3	c3-3@test.com	D27C51036CD016F88772B4A7AA665FAC

2.10. tabula

Datu tabula pēc ielādes

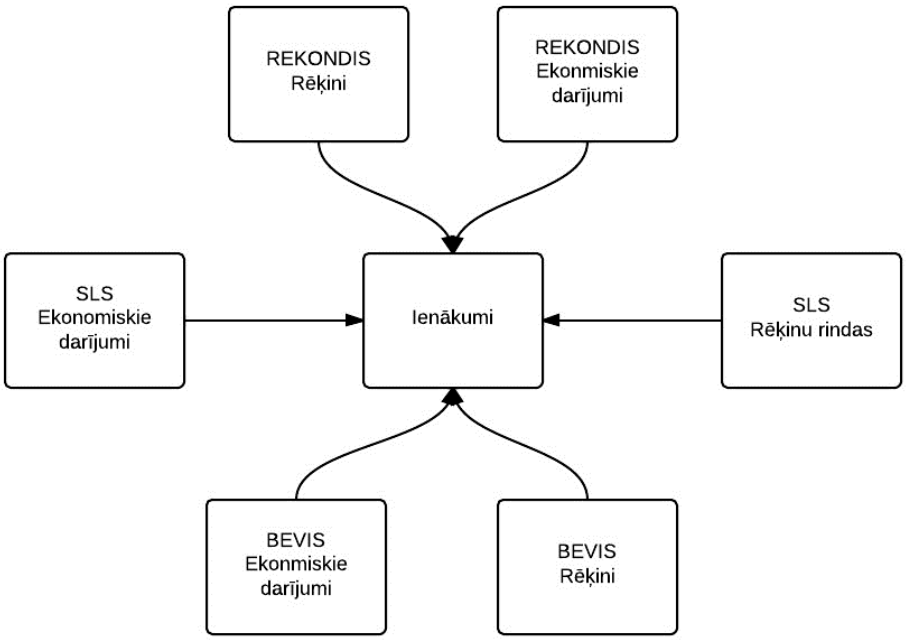
ID	Name	EMAIL	DW_CHECKSUM
1	Company1	c1@test.com	979B902C58CF091F5235576185C83665
2	Company2	c2@test.com	B4AA8C76F7780DCE2D8518A8D7E1DB8E
2	Company2-2	c2@test.com	520D61DE63DDE0F1B71E558E6117EA7A
3	Company3	c3-3@test.com	D27C51036CD016F88772B4A7AA665FAC
3	Company3	c3@test.com	913923C0F64712F3C2FF22E41229BE04
4	Company4	c4@test.com	BA1267391758EC1EAC5AD46D9DF21287

Piemērā redzams, ka klientam Company1 atribūtu vērtības un līdz ar to pārbaudes summa nav mainījusies, un ierakstam jauna versija netiek pievienota. Company2 nosaukums ticis nomainīts uz Company2-2, līdz ar to pārbaudes summa ir mainījusies, un tiek pievienots jauns ieraksts. Company3 e-pasts ir nomainīts, līdz ar to tiek pievienots jauns ieraksts. Company4 ieraksts nav sastopams ienākošajā datu failā, līdz ar to paliek nemainīgi viens ieraksts.

2.6. Datu agregācijas un dimensiju apgabals

Šajā datu apgabalā tiek veidotas faktu un dimensiju tabulas. Pēc būtības dati tiek sagatavoti zvaigznes modelim, tikai fakti un dimensijas netiek sasaistīti. Šajā solī datu modelis tiek mainīts no datu avota datu modeļa uz datu noliktavas modeli. Šajā posmā tiek pielietoti visi biznesa noteikumi un dati tiek agregēti līdz vajadzīgajam līmenim.

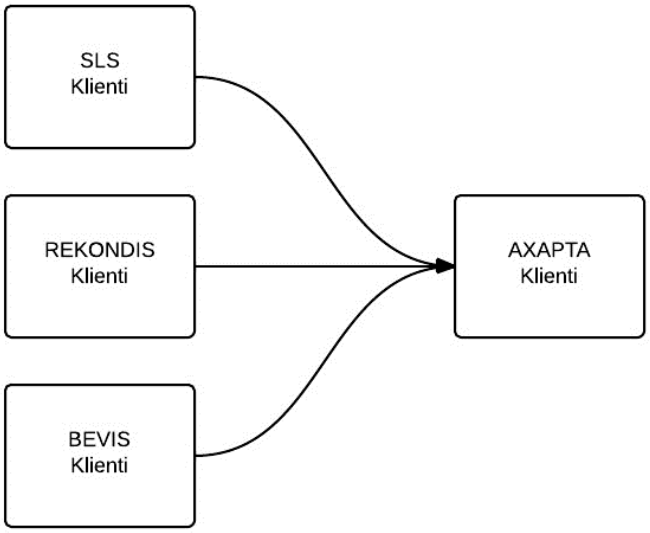
Kā piemēram, dažādas tabulas no dažādiem datu avotiem tiek transformētas uz vienu ienākumu datu struktūru.



2.5. att. **Datu transformācija**

Apvienotas dimensijas netiek veidotas. Visiem ienākumiem un izdevumiem ir tikai viena kopīga dimensija, kas veidojas dinamiski un tiek ielādēta no datu avotiem, tā ir klientu dimensija. Bet arī šajā gadījumā nenotiek ierakstu unificēšana.

Klienti no dažādām sistēmām tiek apvienoti kopējā dimensijā izmantojot vienu no datu avotiem kā galveno avotu šai dimensijai.



2.6. att. **Klientu sasaiste**

Klientu informācija no datu avotiem SLS, REKONDIS, BEVIS satur atslēgas laukus no AXAPTA klienta informācijas un tādējādi tiek savstarpēji sasaistīti. Tādējādi faktu tabulas ir piesaistītas divām klientu dimensijām. Viena no tām satur AXAPTA klientu informāciju otra datu avota klienta informāciju.

2.7. Datuves

Šajā DN apgabalā ir izveidotas savā starpā saistītas 4 zvaigznes shēmas. Ar datu modeļiem var iepazīties pielikumā [3.Pielikums].

- Ienākumu zvaigzne
- Izmaksu zvaigzne
- Izmaksu cenu zvaigzne
- Izmaksu komponentu zvaigzne

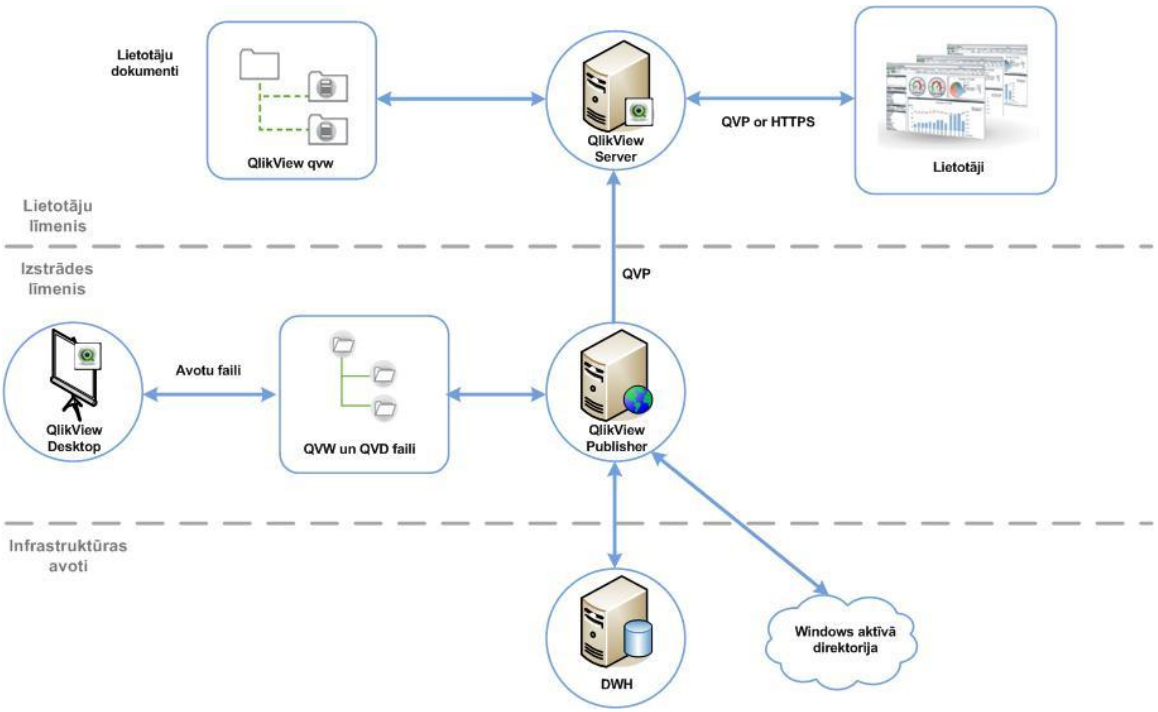
Galvenā atšķirība starp datiem agregāciju apgabalā un datuvē ir tā, ka ielādējot datus no agregāciju apgabala šajā apgabalā, datu ārējās atslēgas tiek aizstātas no naturālām atslēgām uz dimensiju surogātatslēgām.

Tā kā datuves ir savā starpā saistītas ar kopīgām dimensijām, tad var teikt ka DN tiek izmantots apvienoto datuvju modelis. Parasti šis modelis tiek lietots ar augšupejošu pieeju, tomēr konkrētās DN izstrādes pieeja gan vairāk līdzinās lejupejošai pieejai, jo visa pamatā bija biznesa vajadzība automatizēt peļņas un zaudējumu aprēķinu par katru klientu. Pēc tam tika apzināti datu avoti, kuros atrodama informācija par ienākumiem un izdevumiem katram klientam. Kaut arī tika apzināti vajadzīgie datu avoti un to dati, tie netiek transformēti pēc vienotiem standartiem un netiek apvienoti kopīgā relāciju modelī.

2.8. Datu atainošana

Kompānijā kā galvenais biznesa inteliģences rīks tiek ieviests Qlikview, līdz ar to arī šajā projektā datu atainošanai tiek izmantoti Qlikview risinājumi.

Qlikview ir biznesa inteliģences rīks, kas nodrošina datu apstrādi atmiņā. Atšķirībā no lielākās daļas BI rīkiem, Qlikview nav SQL pieprasījumu bāzēts rīks. Tas ielādē visus nepieciešamos datus operatīvajā atmiņā un izmanto pēc klienta pieprasījuma.



2.7. att. **Qlikview arhitektūra**

Kā redzams zīmējumā nr.2.7., Qlikview arhitektūra sastāv no sekojošām komponentēm:

1. Qlikview Desktop

Tas ir MS Windows videi domāts rīks, kas nodrošina datu modeļa un atskaišu izkārtojuma izstrādi Qlikview aplikācijai. Šis ir rīks, kur tiek definēta sasaiste ar datu avotiem, veicamās datu transformācijas kā arī lietotāja saskarnes izstrāde. Izveidotais rezultāts tiek saglabāts QVW vai tā sauktajos Qlikview failos. Šis fails var tikt izmantots, lai izveidotu QVD jeb Qlikview datu failu, kas ir binārs fails un satur tikai apstrādājamos datus.

2. Qlikview Server (QVS)

Šis ir servera puses produkts, kas nodrošina datu analīzi atmiņā kā arī nodrošina Qlikview klienta aplikāciju saziņu ar serveri. Tas iekļauj sevī arī administrācijas rīkus (Qlikview administrācijas konsole), kas nodrošina kontroli pār servera konfigurāciju, pieejas

tiesībām un citiem parametriem. Tas sevī iekļauj arī Web serveri, tajā skaita lietotāju portālu (Access Point).

3. Qlikview Publisher

Šis servera puses rīks, kam ir divas galvenās funkcijas:

- Lai ielādētu datus tieši QVW failos izmantojot definētās sasaistes.
- Lai izplatītu izveidotos dokumentus uz dažādiem serveriem vai statiskā (.pdf) formātā izmantojot e-pastu.

Konkrētajā datu noliktavā Qlikview tiek lietots kā atskaišu rīks, kas ataino datuvēs sagatavotos datus iepriekš definētās atskaitēs. Datu atainošanai dati tiek ielādēti no datuves *.QVD failos, ko Qlikview aplikācija lieto, lai atainotu datus atskaitēs.[7]

Plusi:

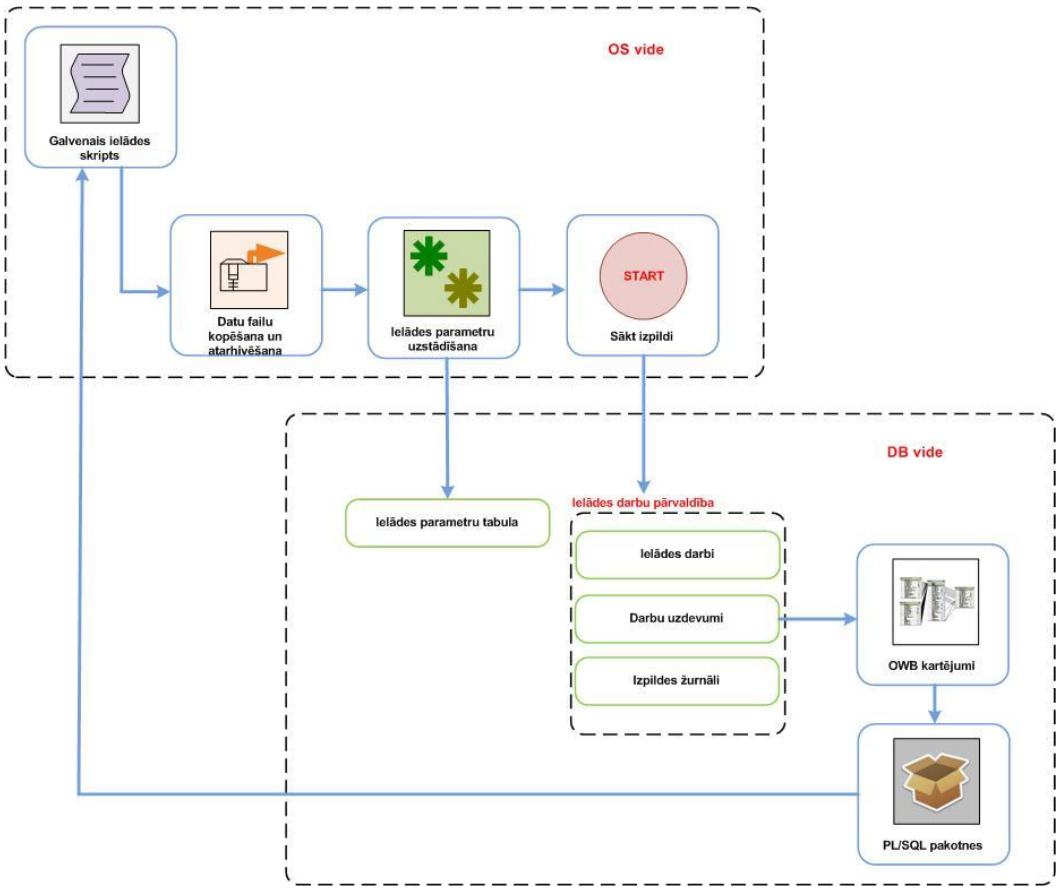
- Tā kā datu pieprasījumi tiek veikti no operatīvās atmiņas, tas paātrina datu iegūšanu, lai tos atainotu. Nav nepieciešams gaidīt kamēr dati tiek izgūti no DN datu bāzes.

Mīnusi:

- Nepieciešama jaudīga aparatūra, lai nodrošinātu liela datu apjoma apstrādi.
- Pašlaik lietotājiem netiek piedāvāta iespēja pašiem lietot Qlikview Desktop aplikāciju, lai veiktu dažādus datu pieprasījumus. Lietotāji var izmantot tikai statiskas, iepriekš definētas atskaites. Drīzumā ir plānots nodrošināt pieeju arī ar Qlikview Desktop aplikāciju.

2.9. DN pārvaldības un izstrādes rīki

Šajā datu noliktavā datu ielādes procesu pārvaldībai tiek izmantots pielāgots risinājums, nevis kāds konkrēts pārvaldības rīks.



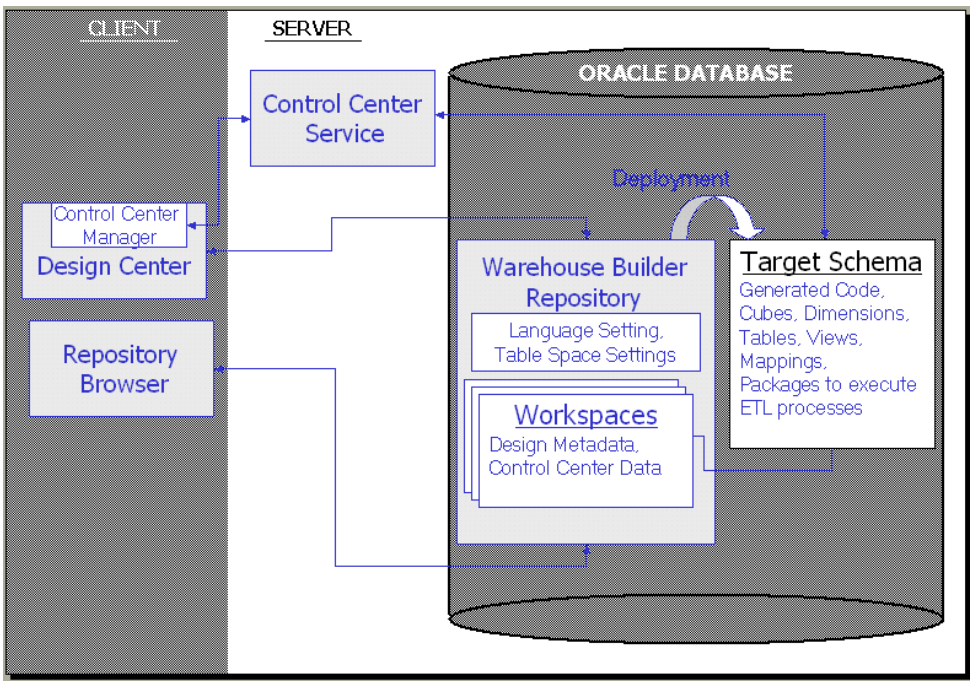
2.8. att. **DN pārvaldības risinājums**

Lai uzsāktu datu ielādi tiek izmantots Unix skripts ar parametriem, kas veic nepieciešamo datu failu piegādi no DN failu glabātuves uz DN datu bāzu serveri un to izpakošanu. Izmantojot šo skriptu uz DN datu bāzi tiek nodoti vajadzīgie ielādes parametri.

Datu bāzes vidē tiek glabāti ielādes darbi un to uzdevumi. Katrs no ielādes uzdevumiem atbilst vienam OWB kartējumam, kas savukārt atbilst vienai PL/SQL pakotnei. Tādējādi izejot no uzstādītajiem parametriem tiek noteikta izpildāmo dabu un uzdevumu kopa.

Datu ielādes automātiska izpilde tiek plānota izmantojot standarta Unix funkcionalitāti (crontab).

Kā datu kartējumu izstrādes rīks tika izvēlēts Oracle Warehouse Builder (OWB). Šis rīks darbojās kā ELT rīks. Līdz ar to visas transformācijas tiek veiktas DN datu bāzē izmantojot DN datu bāzu servera resursus.



2.9. att. **OWB arhitektūra[8]**

Izmantojot šo rīku, deklaratīvi tiek definēti datu kartējumi, uz kā balstoties tiek ģenerētas PL/SQL pakotnes. Kā redzams attēlā nr.2.9., datu kartējumu metadati glabājās DN datubāzē atsevišķā shēmā un PL/SQL pakotnes tiek glabātas mērķa tabulu shēmās. Izpildot datu datu pārlādes tiek izpildīts šis PL/SQL kods.[8]

Šajā gadījumā ETL rīka izvēli galvenokārt noteica sekojoši faktori:

- Datu noliktavas DBVS izvēle. Tā kā DN tiek izmantota Oracle datu bāze, tad OWB iespējams izmantot bez papildus licenču iegādes;
- Darbinieku kompetences. Kompānijā bija pieejami izstrādātāji ar OWB zināšanām;

Datu kvalitātes un sasaistes rīki netiek izmantoti. Tā kā visi dati tiek saņemti failos, tad nav nepieciešama sasaiste ar citām datu bāzēm. Savukārt datu kvalitātes pārbaudes ir iestrādātas OWB kartējumos. Tiek izmantotas datu tipu un formātu pārbaudes darba apgabalā, kā arī datu integritātes pārbaudes pie ielādes vēstures apgabalā.

Konkrētajam risinājumam var izdalīt sekojošas priekšrocības un trūkumus:

Plusi:

- OWB nav nepieciešamas papildus licences;
- Unix skriptu izmantošana procesu kontrolē nodrošina papildus funkcionalitāti darbībām ar failiem un DN failu glabātuves kontroli;
- OWB nodrošina deklaratīva kartējumu definēšanu un pārskatāmību;
- Nav nepieciešamas papildus licences datu kvalitātes rīkiem;

Mīnusi:

- OWB nav integrētas versiju kontroles;
- Nepieciešamas papildus zināšanas Unix skriptu izstrādē.
- Nav iespējas veikt datu profilēšanu

SECINĀJUMI

Darba mērķis bija izpētīt literatūrā apkopotās datu noliktavas arhitektūras pamatnostādnes un iespējamos arhitektūras pieejas variantus un pielietot iegūtās zināšanas parādu piedziņas uzņēmuma datu noliktavas izveidei. Darba gaitā arī izdevās sasniegt nospraustos mērķus.

Teorijas daļā tika apskatīts no kādām galvenajām komponentēm sastāv datu noliktavas arhitektūra. Kādi ir iespējamie datu noliktavas arhitektūras un līdz ar to arī izstrādes veidi. Tika apskatīts kādas ir katras pieejas priekšrocības un trūkumi. Praktiskajā daļā ir apskatīts kā ir realizēta katra no arhitektūras komponentēm un kāda no pieejām tiek izmantota DN izstrādē.

Apkopjot teorētisko informāciju var secināt, ka konkrētās DN implementācija nav tipiska kādai no populārākajām DN izstrādes pieejām. DN plānošana un prasību analīze tika veikta pēc lejupejošas pieejas, vispirms definējot konkrētu biznesa vajadzību, kas tiks atrisināta izmantojot DN. Izmantojot šādu pieeju DN datubāze parasti tiek veidota kā relāciju datu bāze, kurā dati tiek apkopoti pēc vienotiem standartiem, bet šajā gadījumā DN datu bāze ir veidota kā vairākas savā starpā savienotas datuves. Šāds datu modelis biežāk tiek pielietots izmantojot augšupejošu pieeju.

Pēc autora domām specifisks ir arī datu izmaiņu noteikšanas mehānisms, kas ir iestrādāts DN. Biežāk izmaiņu noteikšana notiek jau datu avotā. Interesants ir arī šīs funkcionalitātes tehniskais risinājums izmantojot pārbaudes summu, nevis vienkāršu atribūtu salīdzināšanu.

Izvērtējot tekošo situāciju un DN attīstības plānus, pēc autora domām, lietderīgāk būtu bijis sekot lejupejošai arhitektūras pieejai izstrādājot arī DN datu bāzes modeli un veidot vienotu relāciju modeli. Datu transformācijas, lai apkopotu datus vienotā modelī nemaz nebūtu tik sarežģītas, jo visās valstīs biznesa veids un metodes ir vienveidīgas. Daudzās no valstīm pat tiek izmantotas vienas un tās pašas sistēmas, bet ar nelieliem pielāgojumiem. Tādējādi apvienojot datus no vairāku valstu nodaļām, varētu iegūt korporatīvu skatījumu uz datiem un varētu analizēt datus par kompānijām, kas darbojas vairākās valstīs un ir uzņēmuma klienti vairāku valstu uzņēmuma nodaļām.

Bet, lai analizētu datus viena departamenta vai biznesa kategorijas ietvaros, veidot rumbas un spieķu datu modeli, kur datuves būtu iespējams veidot DN datu bāzē vai jau Qlikview datu struktūrās.

Viena no galvenajām problēmām DN izstrādē ir tā, ka datu izlādi un piegādi uz DN failu glabātuvi nodrošina datu avotu izstrādes un uzturēšanas komandas. Tādējādi jebkuras izmaiņas, piemēram papildus tabulu pievienošana, datu izlādē ir jāsaskaņo ar datu avotu izstrādes komandu un tālākā DN izstrāde ir atkarīga no to darbības ātruma un kvalitātes.

Līdzīgas neērtības sagādāja arī tas, ka Qlikview izstrāde kompānijā notiek centralizēti, ko nodrošina Qlikview izstrādātāji, kas nav piesaistīti konkrētam projektam, bet apkalpo visus projektus, kur nepieciešama Qlikview izstrāde. Tas ietekmē datu atainošanas komponentes gan izstrādes kvalitāti gan ātrumu.

Pēc autora domām ir ļoti lietderīgi ieviest tādu DN komponenti kā failu glabātuve un veikt datu izlādi uz to un ielādi no tās. Tas ļoti atvieglo datu apmaiņu gadījumos, ja DN un datu avoti izmanto dažādas DBVS un nav pieejami labi sasaistes rīki, kā arī gadījumā, ja datu avots neatrodas vienotā tīklā ar DN. Šāda pieeja ir arī salīdzinoši lēta, jo disku vietas izmaksas nav lielas salīdzinājumā ar ETL un sasaistes rīku licenču izmaksām.

Izmantojot šo pieeju datu izlādes un ielādes ātruma izmaiņas nebūs milzīgas, bet dos papildus iespēju glabāt datu vēsturi. Konkrētās DN failu glabātuvē tiek uzglabāti dati par pēdējiem 12 mēnešiem glabājot gan pilnos gan inkrementālos datu arhīvus. Pēc autora domām būtu lietderīgāk saglabāt datu arhīvus pēc to ielādes principa. Ja datu arhīvs tika izmantots datu ielādē, tad tas tiek saglabāts failu glabātuvē, tādējādi saglabājot tikai vienu no arhīviem par katru dienu. Tas ļautu paplašināt uzglabāšanas periodu ar tekošo disku vietas apjomu.

Šāda pieeja varētu nebūt labākais risinājums, ja DN dati ir jāuztur tuvu aktuālajiem datiem un tiek veiktas biežas datu izlādes un ielādes.

Jau ir nosprausti mērķi tālākajai DN attīstībai. Tekošā DN implementācija kalpo klientu peļņas analīzei uzņēmuma Zviedrijas nodaļai. Pašlaik jau notiek izstrāde, lai pievienotu peļņas aprēķinu par uzņēmuma Somijas nodaļas klientiem. Tiek veikta datu analīze un iespējas pievienot arī datus par uzņēmuma Igaunijas klientiem.

Uzņēmuma Zviedrijas nodaļa uztur papildus datu avotu REKONDIS un BEVIS datu bāzes kopijas, kas domātas atskaišu veidošanai izmantojot SAP Business Objects biznesa

inteliģences rīku. Ir pieņemts lēmums aizstāt šīs datu bāzu kopijas papildinot DN ar trūkstošo informāciju, kā arī aizstāt SAP Business Objects BI rīku ar Qlikview risinājumiem.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1]**William Inmon**, Building the Data Warehouse 4th Edition, Wiley, 2005

[2] **Ponniah Paulraj**, Data warehousing fundamentals for IT profession. 2nd ed, 2010

[3]**Thomas C. Hammergren , Alan R. Simon**, Data Warehouse for Dummies, Wiley, 2009

[4]**Sean Kelly,** Data warehousing in action, Wiley, 1997

[5]**Jack E. Olson,** Data Quality: The Accuracy Dimension, Morgan Kaufmann, 2003

[6]**Kimball, R., Caserta, J**., The Data Warehouse ETL Toolkit, Second edition. Wiley, 2008

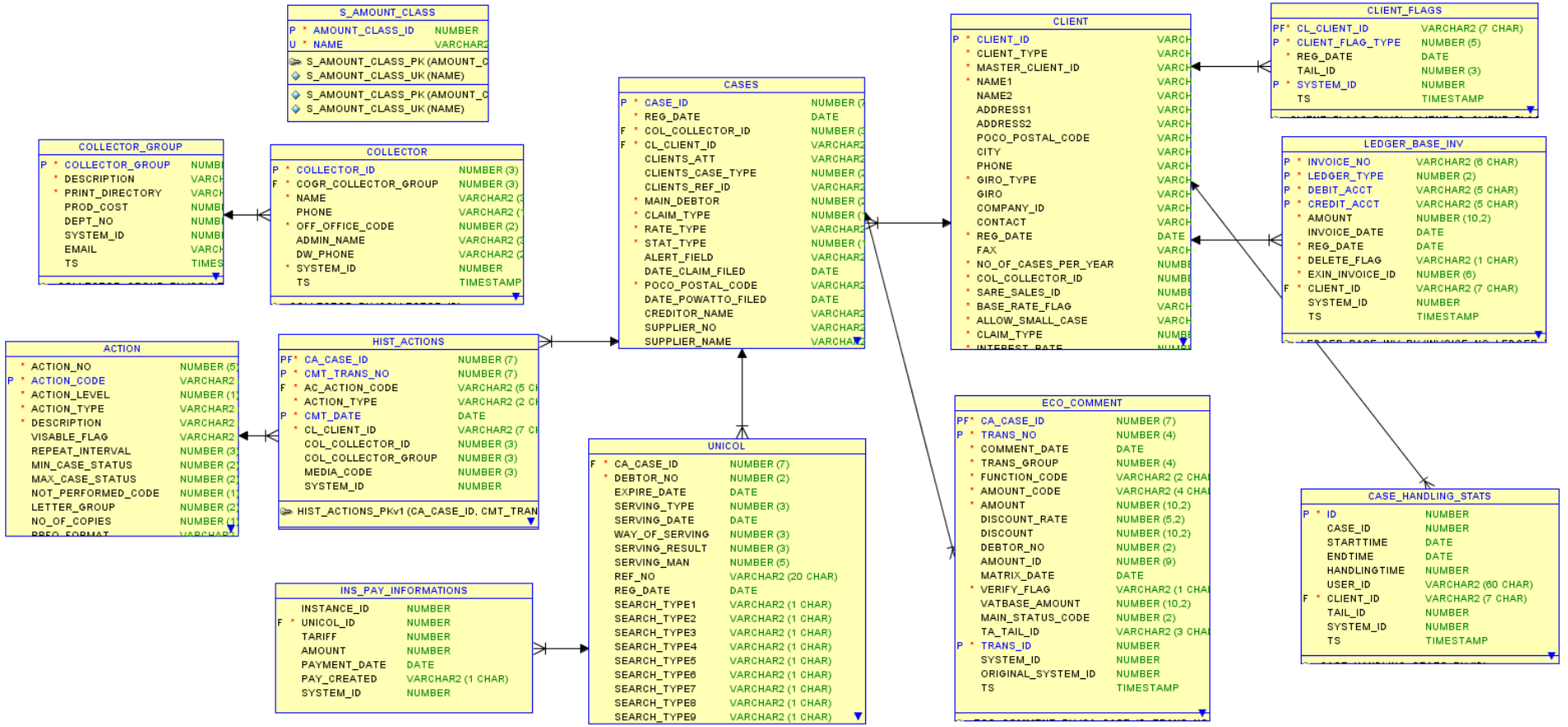
[7]**Qliktech,** Qlikview architectual overview, 2011 [tiešsaite] – [atsauce 15.05.2013] Pieejams: http://www.qlikview.com/us/explore/resources/whitepapers/qlikview-architectural-overview

[8]**Cathy Shea,** Data Oracle® Warehouse Builder Installation and Administration Guide, Oracle, 2012

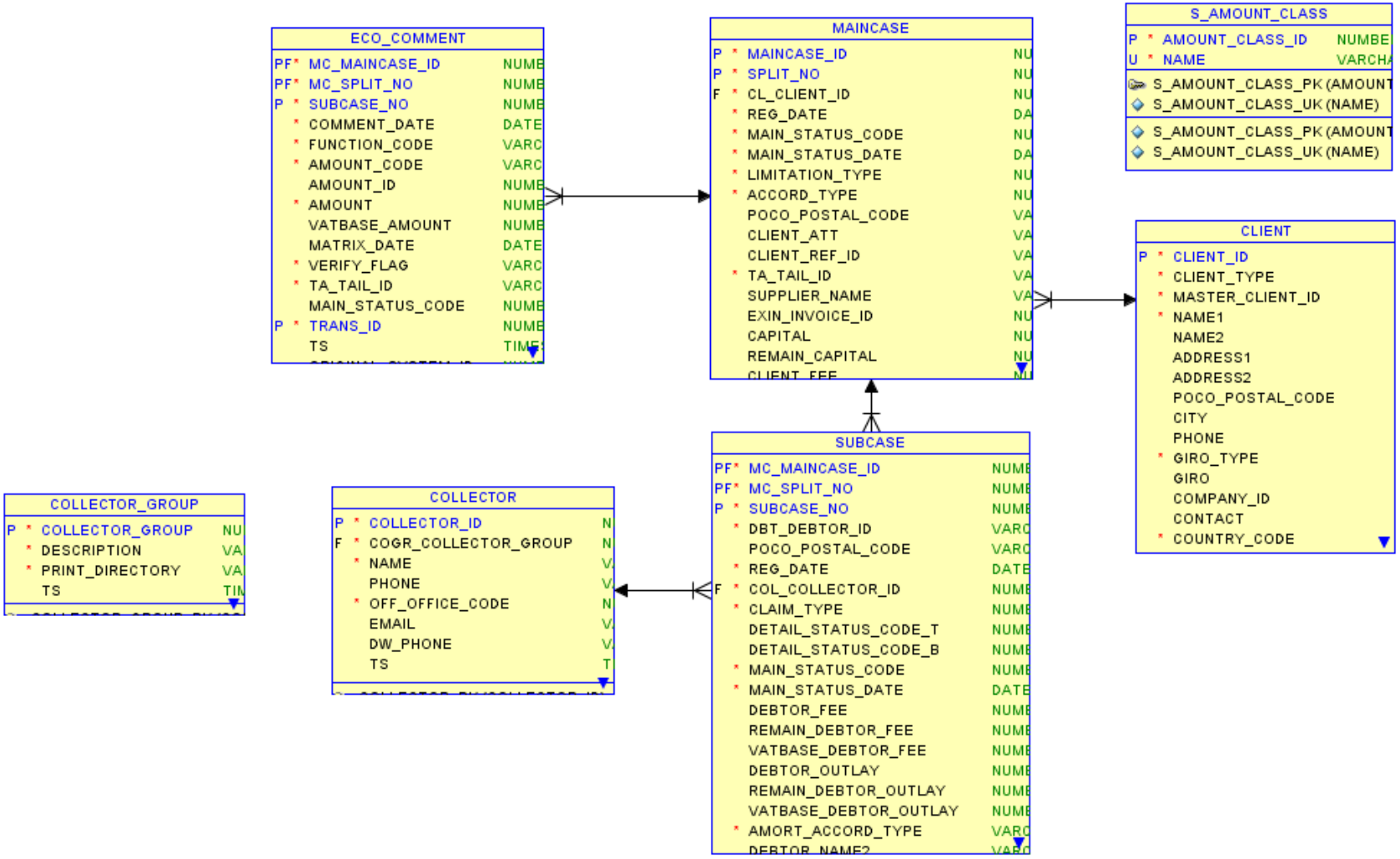
1. pielikums

Pārbaudes summas aprēķina funkcijas kods

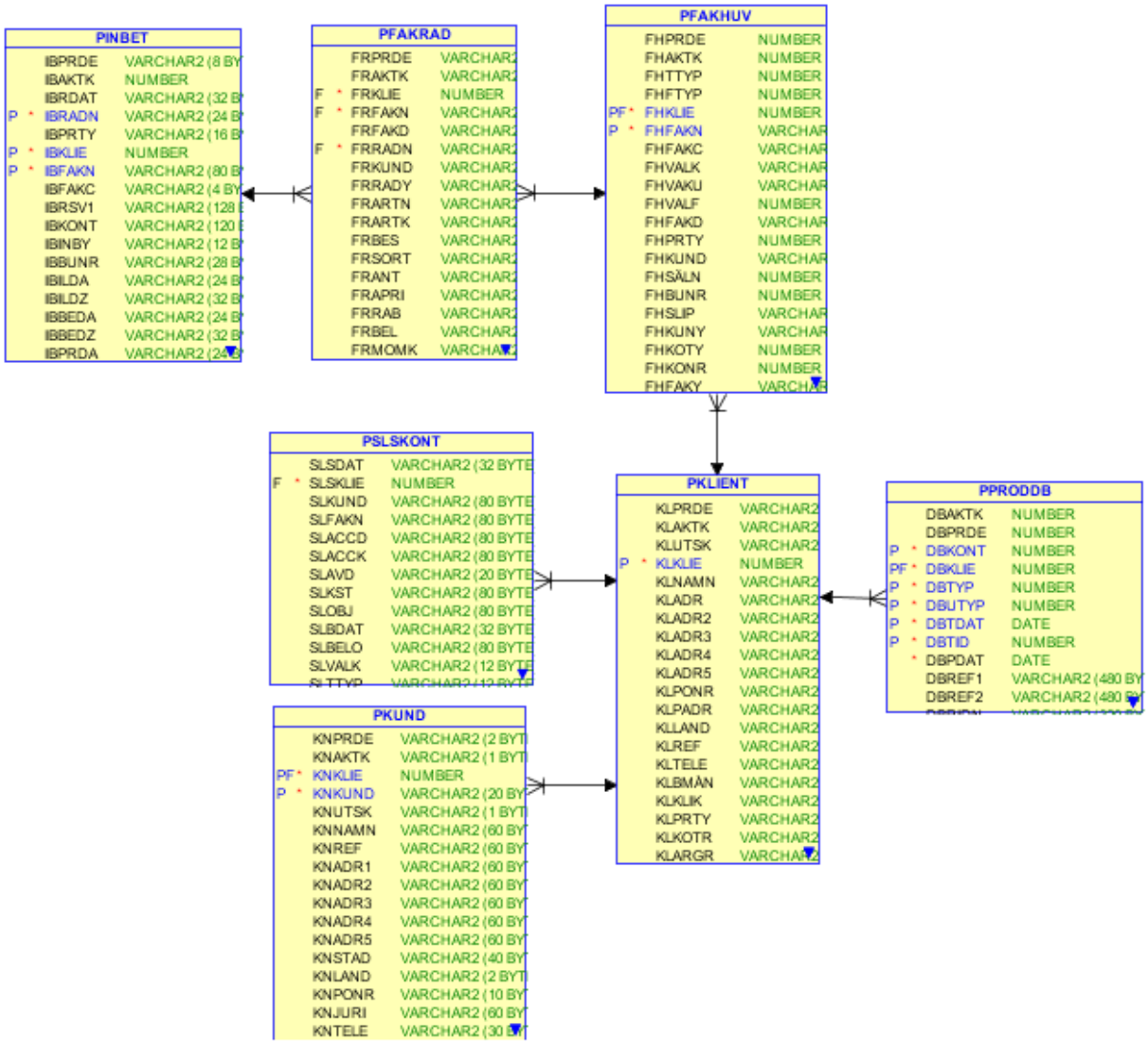
```
FUNCTION "GET_CHECKSUM" ("P_STRING" IN VARCHAR2)
    RETURN VARCHAR2
    DETERMINISTIC
    PARALLEL_ENABLE
  IS
    --initialize variables here
    v_md4hash   VARCHAR2 (32);
  -- main body
  BEGIN
    v_md4hash      :=
      RAWTOHEX (
        DBMS_CRYPTO.
         hash (typ   => DBMS_CRYPTO.hash_md4,
               src   => UTL_RAW.cast_to_raw (p_string)));
    RETURN v_md4hash;
  EXCEPTION
    WHEN OTHERS THEN
      RETURN 0;
  END get_checksum;
```



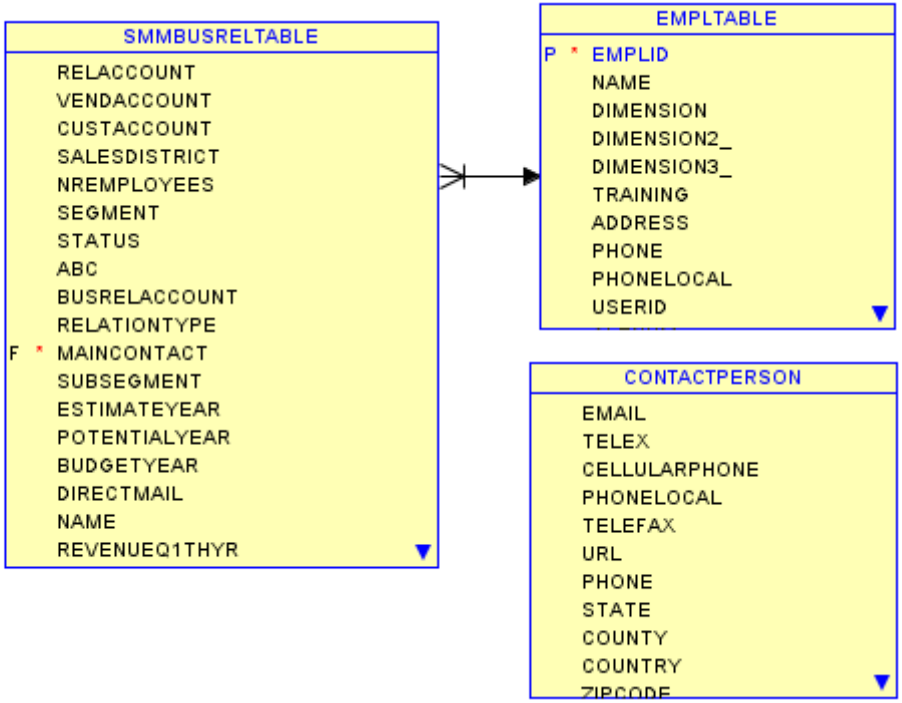
1. att. **Rekondis datu modelis**



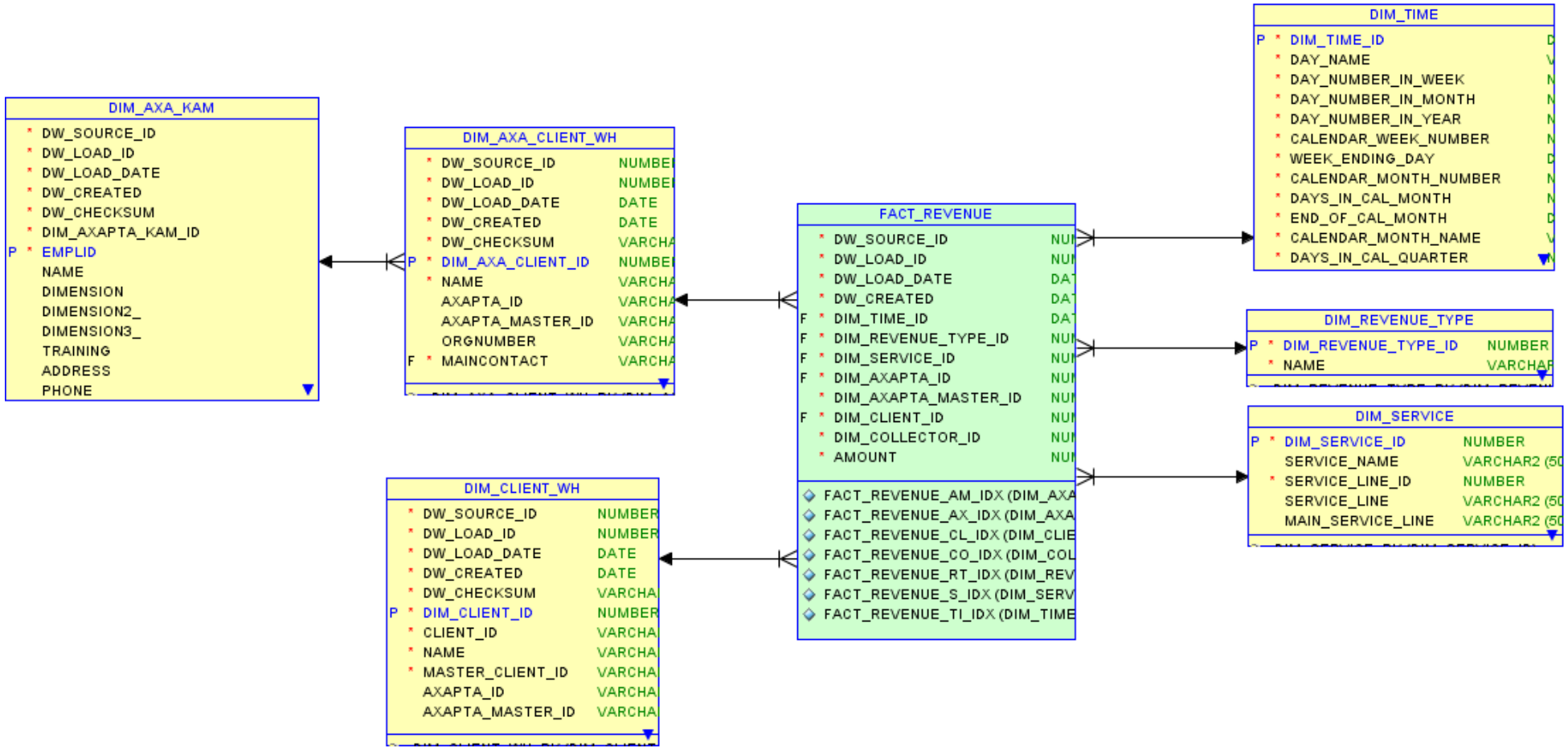
2. att. **Bevis datu modelis**



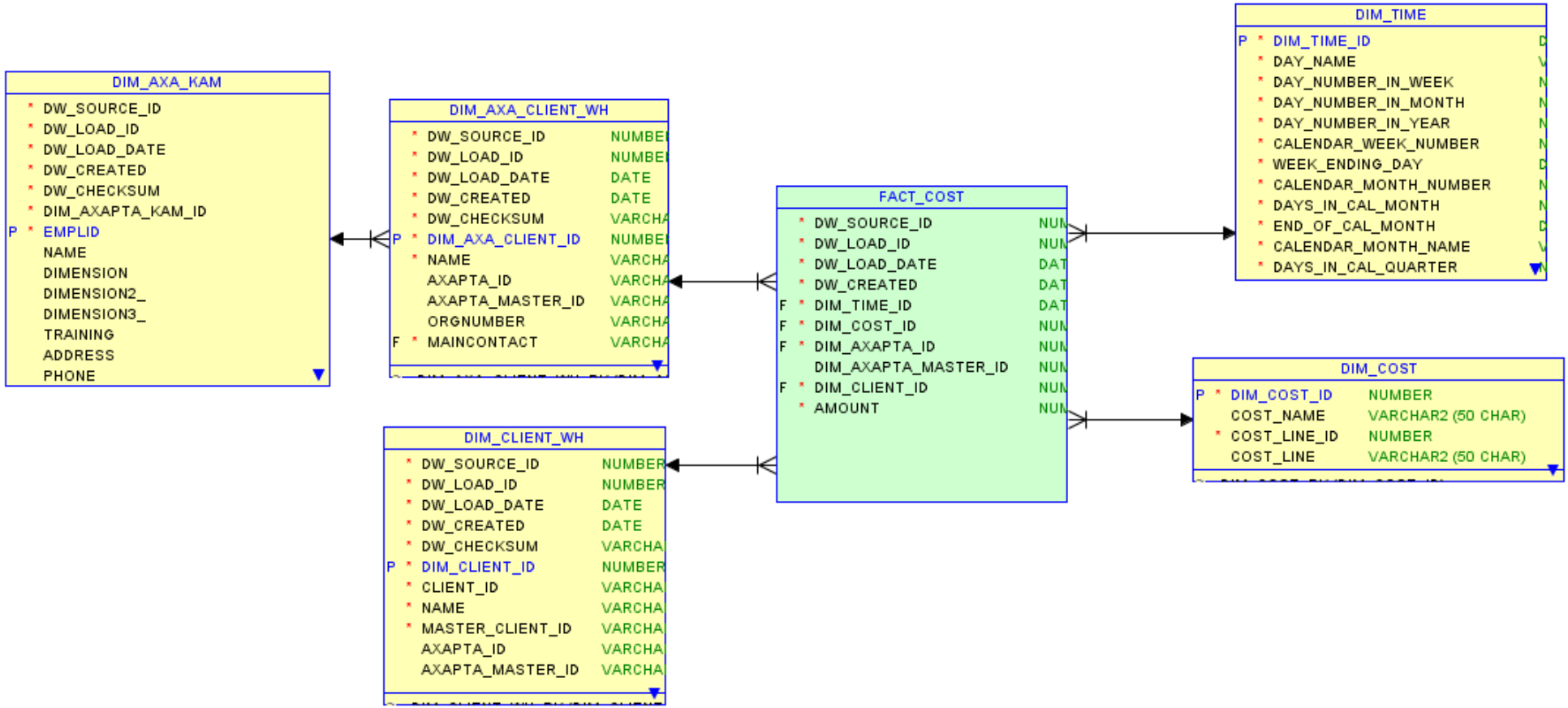
3. att. **SLS datu modelis**



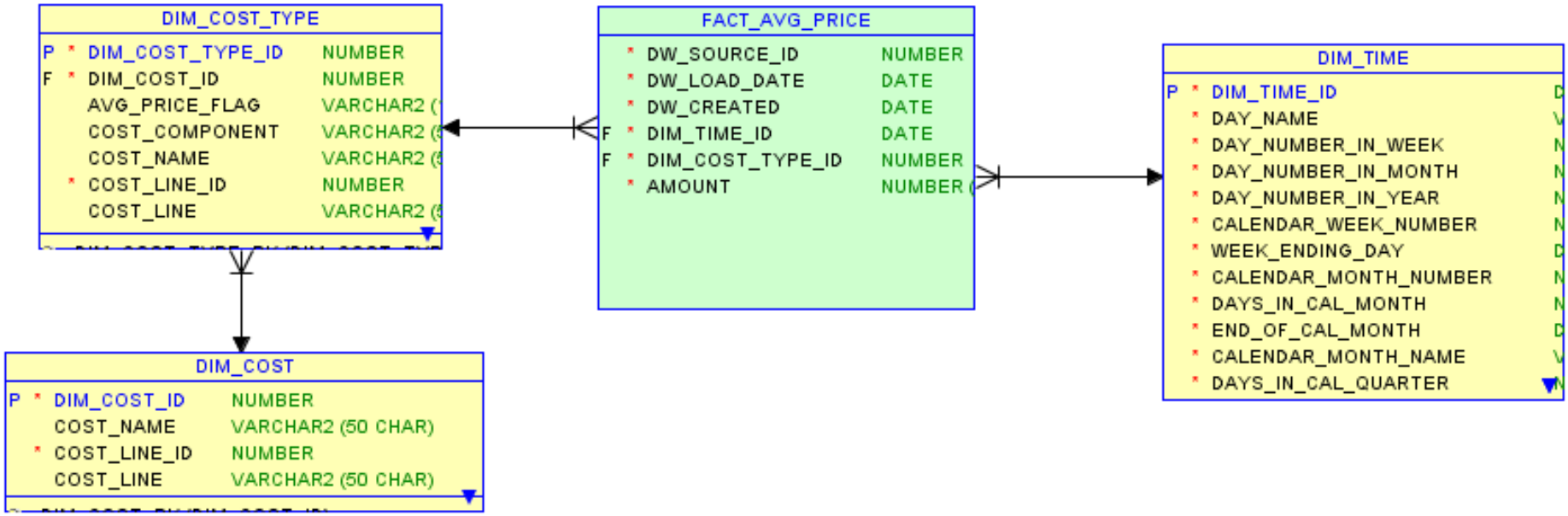
4. att. **Rekondis datu modelis**



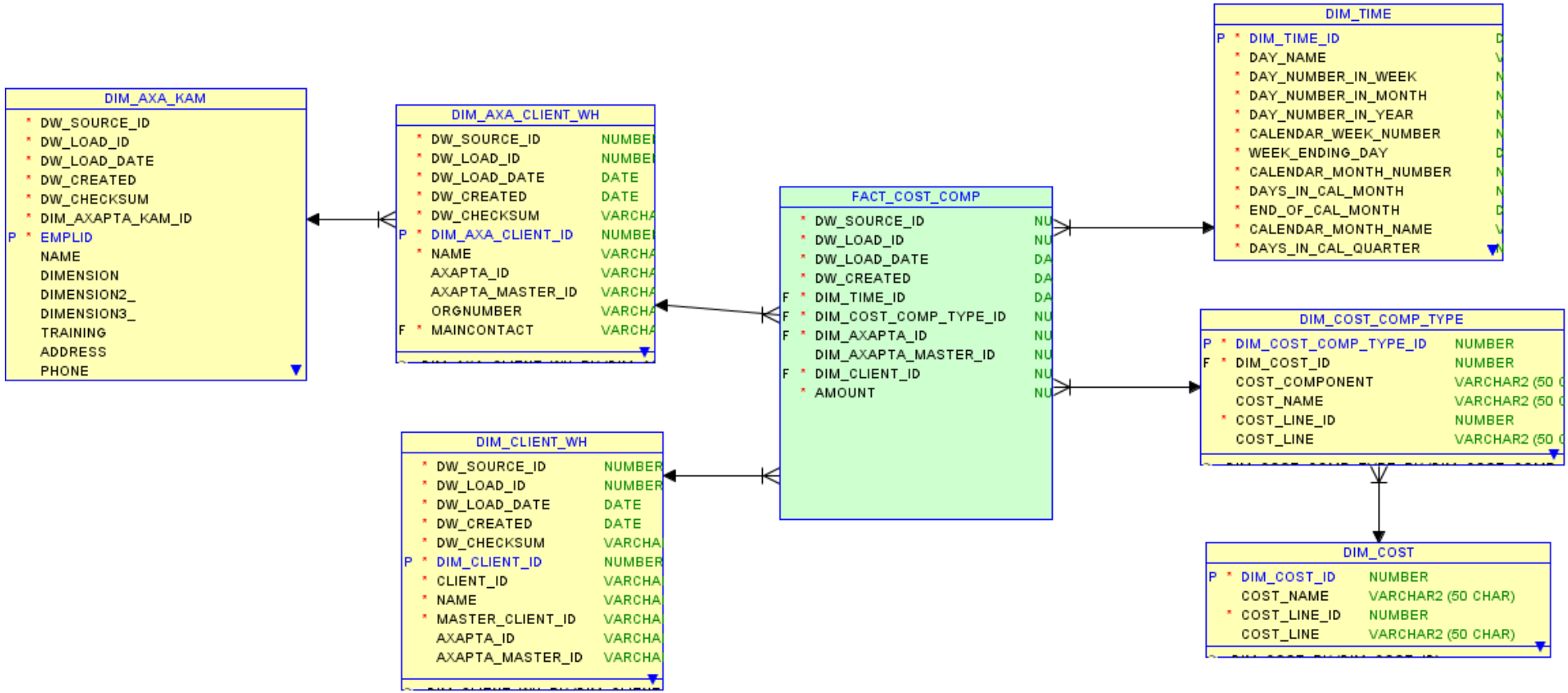
5. att. **Ienākumu datuves datu modelis**



6. att. **Izdevumu datuves datu modelis**



7. att. **Izdevumu cenu datuves datu modelis**



8. att. **Izdevumu komponentu datuves datu modelis**

DOKUMENTĀRA LAPA

Bakalaura darbs „Datu noliktavas arhitektūra parādu piedziņas uzņēmumā” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____ Viesturs Olekšs

(*personiskais paraksts)*

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Vadītāja: Asoc. prof. Laila Niedrīte _____.

(personiskais paraksts) (datums)

Recenzents: Mg. dat. Dita Gabaliņa.

Darbs iesniegts Datorikas fakultātē _____.

(datums)

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe _____.

(personiskais paraksts)

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē_____

(datums)

prot. Nr. _____.

Komisijas sekretāre: _____.

(amats, vārds, uzvārds, personiskais paraksts)