LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

REĀLLAIKA ANALĪZES FUNKCIONALITĀTES IZSTRĀDE DATU NOLIKTAVAI

MAĢISTRA DARBS

Autors: **Oskars Kalniņš**
Stud. Apl. Nr. ok11059
Darba vadītājs: profesore Dr. dat. Laila Niedrīte

RĪGA 2019

ANOTĀCIJA

Šī maģistra darba mērķis ir izstrādāt reāla laika datu noliktavas arhitektūru un realizēt tai atbilstošu risinājumu kā koncepta pierādījumu.

Darba ietvaros ir īsumā aprakstītas tipiskas datu noliktavas un pamatota reāla laika datu noliktavu nepieciešamība. Darba turpinājumā tika apkopota pieejamā akadēmiskā informācija par metodēm, kuras var izmantot, reāla laika datu noliktavu izstrādē. Darbā ir sniegts augsta līmeņa eksistējošās statiskās datu noliktavas apraksts, kā arī detalizēts reāla laika analīzei pielāgoto datu noliktavas izpildes pamatrādītāju ETL process. Turpinājumā ir detalizēti aprakstīta izstrādātā reāla laika risinājuma arhitektūra un tā izstrādes process, izmantojot rīku *Oracle GoldenGate*. Darba noslēgumā ir veikta izstrādātā risinājuma analīze.

Atslēgas vārdi: **datu noliktavas, reāla laika datu noliktavas, reāla laika BI, *Oracle GoldenGate***

ABSTRACT

Title: “Implementation of real-time analysis functionality for data warehouse”

The purpose of this master’s thesis is to develop an architecture for a real-time data warehouse and implement corresponding solution as a proof of concept.

This work includes a short summary of typical data warehouses and justifies the need for real-time data warehouses. Within the framework of this paper, available academic information on methods of development for real-time data warehouses was compiled. Additionally, the work includes a high-level description of the existing static data warehouse as well as a detailed description of the ETL process for key performance indicators that were adjusted for real-time analysis. The paper also includes a detailed description of the architecture of the developed solution and the development process using a tool Oracle GoldenGate. In conclusion an analysis of developed solution is given.

Key words: **data warehouse, real-time data warehouse, real-time BI, Oracle GoldenGate**

AUTOREFERĀTS

Maģistra darba ietvaros autors ir izstrādājis reāla laika datu noliktavas risinājumu un izveidojis detalizētu tā aprakstu. Darbā izstrādātajā risinājumā datu replicēšanai reālā laikā ir izmantota izmainīto datu vākšanas metode, kura ir realizēta replicējot izmaiņas datu noliktavā izmantojot rīku *Oracle GoldenGate*. Replicēto datu transformēšanai un integrēšanai ar statiskās datu noliktavas datiem ir izmantota klasisko un materializēto datu bāzes skatu kombinācija, kas nodrošina efektīvu datu apstrādi. Datu attēlošanai lietotājam ir izmantots *Oracle APEX* platformā izstrādāts rīks, kuru darba autors ir pielāgojis, lai tas attēlotu ar polišu transakciju skaitu saistīto izpildes pamatrādītāju vērtības reālā laikā, kā arī, ja nepieciešams, atjaunotu reāla laika datu bāzes materializētos skatus, kad lietotājs pieprasa apskatīt reāla laika datus.

Darba ietvaros ir radīts detalizēts izstrādātā risinājuma arhitektūras, ELT procesa un izstrādes procesa apraksts. Ir sniegts augsta līmeņa eksistējošās datu noliktavas vispārējs apraksts, kā arī detalizēts pielāgojamā datu noliktavas fakta struktūras un ETL procesa apraksts. Darbā ir apkopotas teorētiskas idejas un metodes, kuras var izmantot, lai realizētu reāla laika datu noliktavu. Rezultātā ir izstrādāts reāla laika datu noliktavas risinājuma koncepta apliecinājums un iegūts detalizēts un kvalitatīvs tā apraksts.

Par informācijas avotiem darbam izmantoti tikai zinātniski raksti, grāmatas un izstrādātāju dokumentācija, kas ļauj domāt, ka darbā paustās idejas ir korektas. Maģistra darbā izmantotie literatūras avoti tika meklēti populārākajos akadēmisko rakstu meklēšanas pakalpojumos, kas ļauj domāt, ka darba izstrādei ir izdevies iepazīties ar visaptverošu pieejamās literatūras pārskatu.

Pareizrakstības kļūdu pārbaudei izmantota Microsoft Word iebūvētā funkcionalitāte, papildus darbu pārlasot darba autoram. Darbā ir izmantota tikai oficiāli pieņemta datorzinātnes terminoloģija latviešu valodā. Teksts un virsraksti noformēti atbilstoši metodiskajiem norādījumiem, iekļaujot visas prasītās darba nodaļas.

Katram no izmantotajiem literatūras avotiem un aizgūtajiem attēliem ir atsauces.

Satura rādītājs

APZĪMĒJUMU SARAKSTS

Apzīmējums	**Skaidrojums**
Atkārtošanas žurnāla datnes	Atkārtošanas (no angļu *redo*) žurnāla datnes ir žurnāldatnes, kuras tiek izmantotas, lai anulētu izdarīto atsaukšanas operāciju.
BI	Biznesa inteliģence (no angļu *business intelligence*) ir tehnoloģiju vadīts datu analīzes un informācijas attēlošanas process, ar mērķi palīdzēt lietotājam pieņemt apzinātu lēmumu.
Datuve	Datu glabātuve, kurā savākti operatīvie dati un dati, kas nepieciešami noteiktai lietotāju grupai. Šos datus var iegūt no uzņēmuma datu bāzes, datu noliktavas vai kāda cita specifiska avota, lai noteiktas lietotāju grupas šos datus saņemtu ērti lietojamā formā un varētu veikt ar tiem nepieciešamās darbības.
DDL	Valoda, kas parasti ir datu bāzes pārvaldības sistēmas sastāvdaļa un ko izmanto, lai aprakstītu visus datu bāzes atribūtus un īpašības (piem., ierakstu izvietojumu, atslēgu laukus, datņu atrašanās vietas u.c.).
DML	Valoda, kas parasti ir datu bāzu pārvaldības sistēmas daļa un kas atvieglo datu manipulēšanas operāciju izpildi.
ETL	Process datu izvilkšanai no dažādiem datu avotiem, transformēšanai atbilstošos datu tipos un transformēto datu ielādēšanai izmantošanai lietotnē (no angļu *Extract – Transform - Load*).
ELT	ETL procesa variācija, kur datu transformācija tiek veikta pēc datu ielādes mērķa sisēmā (no angļu *Extract – Load - Transform*).
IT	Informācijas tehnoloģija
Izmainīto datu vākšana	Izmainīto datu vākšana (no angļu *changed data capture*) ir datu izvilkšanas metode, kurā tiek iegūti tikai izmainītie dati.
Izpildes pamatrādītājs	Iepriekš noteikts mērs, kurš tiek izmantots, lai mērītu stratēģiskā mērķa, plāna, iniciatīvas vai darījumu procesa izpildījumu.
Izplātā datne	Izplātā datne (no angļu *flat file*) ir datne, kas nav fizikāli saistīta ar citām datnēm un nesatur norādes uz tām. Izplātajai datnei nepiemīt hierarhiska struktūra.
Izstādes vide	Izstādes vide (no angļu *staging area*) ir pagaidu glabāšanas vide, kurā dati tiek ielādēti apstrādei ETL procesā.

Karstais nodalījums	Datu noliktavas nodalījums, kurš konkrētajā laika momentā tiek izmantots reāllaika datu noliktavas funkcionalitātes nodrošināšanai.
Kešdarbe	Nesen piekļūtās informācijas īslaicīga uzglabāšana īpašā atmiņas apakšsistēmā ātrākai piekļuvei.
MapReduce	Programmēšanas modelis, kas saistāms ar lielu datu kopu ģenerēšanu un apstrādi ar paralēliem, sadalītiem algoritmiem klasterī.
OLAP	Tiešsaistes analītiskā apstrāde (no angļu *online analytical processing*) - daudzdimensionālas datu bāzes analīzes metode, kas nodrošina specifisku datu bāzu indeksāciju, tādējādi paātrinot piekļuvi datiem gadījumos, kad jāpārskata lieli datu masīvi, kā arī ļaujot analizēt datus daudzos dažādos aspektos.
REST	Programmatūras arhitektūras stils, kas nosaka vadlīnijas un labo praksi mērogojamu tīmekļa programmu radīšanai.
Saliktā notikumu apstrāde	Saliktā notikumu apstrāde (no angļu *complex event processing*) ir dažādu avotu notikumu straumju nepārtraukta un inkrementāla apstrāde, kas balstīta uz deklaratīviem vaicājumiem un šablonu aprakstiem ar gaidīšanas laiku tuvu nullei.
Sūces un apmaiņas metode	Sūces un apmaiņas metode (no angļu *trickle and flip*) ir datu ielādes metode, kurā dati datu noliktavā nepārtraukti tiek ielādēti atsevišķā reāla laika nodalījumā, kurš periodiski aizstāj statisko faktu tabulu.
Tiešas sūces plūsmas metode	Tiešas sūces plūsmas metode (no angļu *direct trickle feed*) ir datu ielādes metode, kurā dati datu noliktavā nepārtraukti tiek ielādēti tieši faktu tabulā.

IEVADS

Mūsdienās pieaug nepieciešamība pēc risinājumiem, kas ļauj organizācijām ātri iegūt analītisku informāciju no pieaugoša datu daudzuma, kas tiek radīts uzņēmuma ikdienas darījumprocesos. Tā kā informācijas vērtība ar laiku samazinās, ir nepieciešams maksimāli samazināt laiku, kas nepieciešams, lai informāciju no operacionālās sistēmas apstrādātu un nogādātu gala lietotājam lēmuma pieņemšanai. Lai operacionālās sistēmas datus pārvērstu informācijā uz kuru var balstīt lēmumu pieņemšanu laikā, ir nepieciešams izstrādāt reāla laika datu noliktavas.

Šī maģistra darba mērķis ir izstrādāt funkcionalitāti eksistējošas klasiskās datu noliktavas vienas daudzdimensiju modeļa shēmas datu atjaunošanai reālā laikā, lai analizētu pieejamās arhitektūras, tehnoloģiju iespējas un izvērtētu ieguvumus.

Maģistra darba sākumā ir aprakstītas tipiskas datu noliktavas, to arhitektūra un BI sistēmas, kas uz tām ir balstītas. Darbā ir sniegts īss ieskats BI sistēmu izcelsmē un vēsturē, kā arī sniegts pārskats par šādu sistēmu arhitektūru, funkcionalitāti un problēmām, kuras ir jārisina tipisku BI risinājumu izstrādē. Darba turpinājumā ir aprakstītas reāla laika datu noliktavas un BI rīki, un metodes, kuras var izmantot, lai realizētu reāla laika datu noliktavu. Darbā ir tuvāk apskatīta reāla laika datu noliktavu arhitektūra un izmaiņas klasiskajos ETL procesos, kas ir jāveic, lai datus sistēmā būtu iespējam ielādēt reālā laikā.

Turpmākajās nodaļās ir sniegts eksistējošās datu noliktavas augsta līmeņa apraksts, kurai ir izstrādāta viena daudzdimensiju modeļa shēmas datu atjaunošana reālā laikā. Augstā līmenī ir aprakstīta šīs datu noliktavas arhitektūra, sniegts ieskats tajā realizētajā ETL procesā un aprakstītas tehnoloģijas, kas ir izmantotas datu noliktavas realizācijā. Darbā ir detalizēti aprakstīta atvasināta polises transakcijas fakta struktūra un ETL process tā iegūšanai.

Darba noslēgumā ir detalizēti aprakstīts izstrādātais reāla laika datu noliktavas risinājums atvasināta polises transakcijas fakta un uz tā balstīto izpildes pamatrādītāju atjaunošanai reālā laikā. Darbā ir detalizēti aprakstīta izstrādātā risinājuma arhitektūra, kā arī reāla laika datu noliktavas risinājuma izstrāde, kas ietver iesaistīto avota un mērķa datu bāzu sagatavošanu, nepieciešamās programmatūras uzstādīšanu un reāla laika datu integrācija ar statiskajiem datu noliktavas datiem. Ir veikta īsa neformāla izstrādātā risinājuma analīze.

Darbā izstrādātais reāla laika datu noliktavas risinājums ir pilnībā autora izstrādāts. Maģistra darbam nav pievienots izstrādātā risinājuma pirmkods, jo tas ir konfidenciāls un ir kompānijas īpašums. Pie aprakstītā risinājuma darbs tiks turpināts, lai to varētu piedāvāt klientiem kā daļu no datu noliktavas “XIA”.

1. DATU NOLIKTAVAS UN BIZNESA INTELIĢENCE

Šajā nodaļā ir sniegts vispārīgs biznesa informācijas sistēmu apraksts un neliels ieskats to rašanās vēsturē. Turpinājumā tiek aprakstīts biznesa informācijas sistēmu sadalījums pēc laika kurā informācijā tiek nogādāta no operacionālās sistēmas datu noliktavā, kur informācija kļūst pieejama biznesa informācijas sistēmai. Nodaļas nobeigumā tuvāk tiks apskatītas reāllaika biznesa informācijas sistēmas un to priekšrocības.

1.1. Biznesa inteliģences sistēmu vēsture

Biznesa inteliģences (BI) sistēmas ir rīku un metožu kopums, kas tiek izmantots, lai iegūtu un organizētu informāciju, kas ir nepieciešama organizācijas stratēģijas plānotājiem un lēmumu pieņēmējiem, lai palīdzētu pieņemt lēmumus, kas uzlabo uzņēmuma ienesīgumu. Tās sevī ietver uzņēmējdarbībai nozīmīgu informāciju un tās analīzi, kas ir nepieciešama, lai pieņemtu labākus lēmumus. Tas, kas ir uzskatāms par uzņēmumam labāku lēmumu, ir atkarīgs no katra uzņēmuma specifikas un nosaka to, kādai informācijai būtu jāparadās BI sistēmā. Papildus tam BI sistēmas var izmantot, lai izprastu kompānijas iekšējās iespējas, pieejamos resursus un to izmantojumu.

Lai arī BI kā termins IT pasaulē ir parādījies salīdzinoši nesen, tā pirmsākumi ir meklējami aptuveni 50 gadu senā pagātnē. Pieejas šāda tipa informācijas iegūšanai gadu gaitā ir attīstījušās un ar katru iterāciju to iespējas ir palielinājušās, pielāgojoties uzņēmumu pieaugošajai sarežģītībai un tehnoloģisko iespēju attīstībai. BI sistēmas aizstāja lēmumu atbalsta sistēmas un izpildvadītāju informācijas sistēmas. [1]

Lēmumu atbalsta sistēmas ir sistēmas, kuras palīdzēja risināt problēmas, kur vismaz daļai no problēmas nav iespējams izstrādāt realizējamu algoritmu tās atrisināšanai. Šādos gadījumos problēmas daļai, kurai ir iespējams izstrādāt algoritmu tās atrisināšanai tiek izstrādāta informācijas sistēma, bet, lai atrasinātu problēmas daļu, kurai algoritma nav, tiek piesaistīts cilvēks, kurš ar izstrādātās sistēmas sniegtās informācijas palīdzību šo problēmu var atrisināt. Šī iemesla dēļ šādas sistēmas sauca par lēmumu atbalsta sistēmām. Lēmumu atbalsta sistēmas parasti tika realizētas konkrētas problēmas atrisināšanai izmantojot atbilstošus datu modeļus, un tās tika realizētas izmantojot relāciju datu bāzes un dažādas izklājlapu pakotnes. [2]

Izpildvadītāju informācijas sistēmas attīstījās no lēmumu atbalsta sistēmām. Atšķirībā no lēmumu atbalsta sistēmām, kas bija paredzētas, lai palīdzētu atsevišķām grupām pieņemt lēmumus, izpildvadītāju informācijas sistēmas palīdzēja pieņemt lēmumus uzņēmuma līmenī. Šādas sistēmas vēljoprojām bija paredzētas lietošanai nelielam cilvēku skaitam, t.i. kompānijas augstākajai vadībai. Izpildvadītāju informācijas sistēmas parasti tika realizētas lieldatoros, kas nozīmēja to, ka šādu sistēmu izstrāde bija dārga un to pielietojums nebija elastīgs. [2, 3]

Šo vēsturisko sistēmu galvenā problēma ir tā, ka tās mēģina iegūt analītisku informāciju tieši no operacionālajām sistēmām. Informācija šajās sistēmas netiek glabāta piemērotās datu struktūrās un agregācijas pakāpēs, kā rezultātā informācija tiek iegūta neefektīvi un operacionālā sistēma tiek papildu noslogota. 1980-to gadu beigās uzņēmumos sāka veidot datu noliktavas, kurās tika glabāta stratēģiski svarīga informācija, kas bija atvasināta no operacionālajās sistēmās glabātiem datiem. Datu noliktavas parasti tika veidotas tādu sfēru uzņēmumos, kuros tika radīts daudz transakciju informācijas, piemēram, finanšu sektora, apdrošināšanas vai telekomunikāciju uzņēmumos, ar mērķi šo informāciju analizēt mārketinga nolūkos. Datu noliktavas pirmo reizi ļāva analītiķiem apstrādāt lielus datu apjomus, un iegūt no šī datu apjoma lietojamu informāciju. Šādā veidā datu noliktavas kļuva par galveno faktoru, kas ļauj izstrādāt efektīvus BI risinājumus. [3, 4]

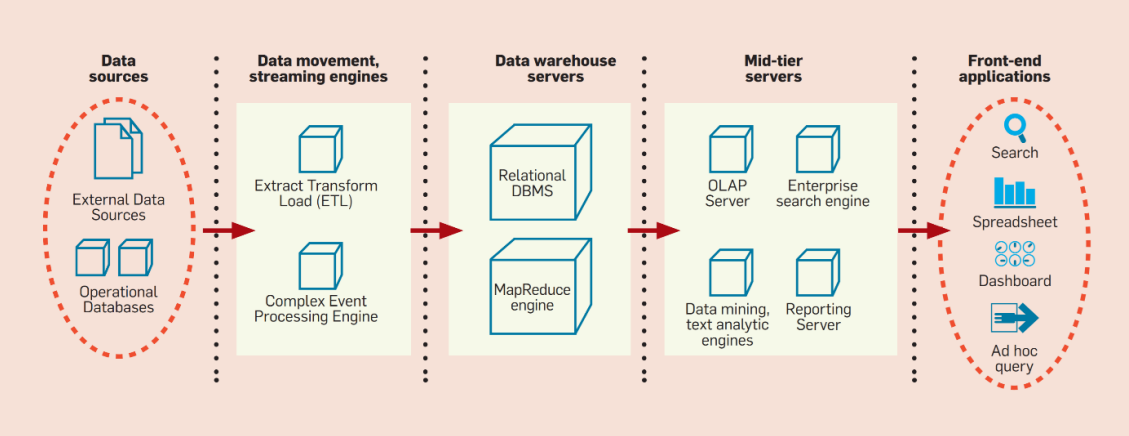
1.2. Tipiskās datu noliktavas un biznesa inteliģences sistēmas

BI sistēmas ir strauji attīstījušās pēdējo 20 gadu laikā. Šo attīstību ir sekmējis tas, ka izmaksas, lai iegūtu lielus datu apjumus no dažādiem datu avotiem un tos glabātu informācijas sistēmās ir samazinājušās. Tas ļauj uzņēmumiem strādāt pie risinājumiem, kas ļauj šos datus apstrādāt un uzglabāt. Mūsdienās lielākā daļa no veiksmīgajiem uzņēmumiem lieto kāda veida BI risinājumu, lai uzlabotu savu darbību. [5]

BI sistēmu lietošana uzņēmumam rada vairākus ieguvumus. BI sistēmu lietošana var palīdzēt uzņēmumam samazināt datu apjomu, ko uzņēmums glabā, izslēdzot datuvēs glabātos dublikāt datus un datus, kam nav pielietojuma. Datu noliktavas parasti glabā datus, kas ir priekšmeta orientēti, integrēti, laikā dalīti un nemainīgi. Tas nozīmē to, ka BI var atbildēt uz gala lietotāju jautājumiem par to, kas ir noticis pagātnē, jo tās ļauj efektīvi analizēt vēsturisko datu nozīmi. Veiksmīgi realizētas BI sistēmas sniedz tās lietotājam vairāk un labāku informāciju, kas ļauj lietotājam pieņemt labākus lēmumus. Šādu sistēmu izmantošana var

palīdzēt uzņēmumam uzlabot savus darījumprocesus un veicināt stratēģisku uzņēmuma mērķu sasniegšanu. [6, 7]

Tipiski biznesa inteliģences risinājumi kā izejas datus izmanto vienu vai vairākus iekšējos vai ārējos datu avotus. Dati šajos datu avotos var būt dažādas kvalitātes, datu objektu reprezentācija var atšķirties. Šī iemesla dēļ ir nepieciešams šos datus apstrādāt, lai tos atbrīvotu no šīm atšķirībām. Šim nolūkam parasti tiek realizēti ETL (no angļu *Extract – Transform - Load*) procesi, kas ir procesu kopums, lai datus izvilktu no datu avota, transformētu, lai atbrīvotos no pretrunām un reprezentācijas atšķirībām datos un integrētu atbilstošos ierakstus no dažādiem avotiem, un transformācijas rezultātu ielādētu datu noliktavā. Lai iegūtu datus, kas ir tuvāki reālam laikam var tikt izmantoti tādi pielāgoti risinājumi kā saliktās notikumu apstrādes (no angļu *complex event processing*) programmas. Pēc tam, kad dati ir transformēti tie tiek ielādēti datu noliktavās. Datu noliktava tipiski tiek realizēta izmantojot relāciju datu bāzes pārvaldības sistēmu vai arī gadījumos, kad datu apjoms ir liels, tā var tikt realizēta *MapReduce* bāzētā programmā. *MapReduce* ir programmēšanas modelis, kas saistāms ar lielu datu kopu apstrādi ar paralēliem, sadalītiem algoritmiem klasterī. Lai apstrādātu datu noliktavā ielādētos datus atgriešanai gala lietotājam, var tikt izmantoti dažādi starpserveri, kuru izvēle ir atkarīga no konkrētās BI sistēmas lietojuma. Pēdējais solis BI sistēmā ir datu attēlošana un atgriešana gala lietotājam. Arī šeit izmantotie risinājumi ir atkarīgi no konkrētās BI sistēmas lietojuma. Piemērs tipiskai BI sistēmas arhitektūrai ir redzams attēlā 1.1. [5, 8, 9, 10]



1.1. att. **Tipiska biznesa inteliģences sistēmas arhitektūra [5]**

Lai iegūtu datus no datu avotiem, ir jāatrisina vairākas problēmas. Ir svarīgi no datu avotiem atlasīt tieši klientam nepieciešamos datus, jo pretējā gadījumā, ielādējot datu noliktavā lielu nevajadzīgu datu apjomu, pasliktināsies datu noliktavas veiktspēja. Datu avoti, no kuriem dati tiek iegūti, var atšķirties pēc to strukturētības pakāpes un datu avota tipa. Pēc strukturētības pakāpes datus var iedalīt strukturētos datos, piemēram relāciju datu bāzes, daļēji strukturētos datos, kā ziņu raksti vai sistēmas žurnālu datnes, un nestrukturētos datos, kā attēli

vai video. Datu avoti var būt uzņēmuma iekšējs datu avots, piemēram, operacionāla informācijas sistēma, vai ārējs datu avots, piemēram, valsts automašīnu reģistrs. Pie tam, avoti, no kuriem dati ir jāiegūst, var būt realizēti dažādās tehnoloģijās, tādējādi radot papildu sarežģījumus datu atlasei. Atlasot datus, kuri tiks ielādēti datu noliktavā, ir jāņem vērā šie faktori un jāizvēlas attiecīgi risinājumi un pieejas, lai šos datus varētu efektīvi atlasīt. Piemēram, lai efektīvi iegūtu datus no iekšējas operacionālas sistēmas, tās datu bāzē var būt nepieciešams izveidot atbilstošas indeksu struktūras, materializētos skatus vai sadalīt datus nodalījumos, lai uzlabotu datu atlases efektivitāti. Nodalījums ir datu loģiska vai funkcionāla daļa. Pie tam, BI sistēmai ir jābūt izstrādātai tādā veidā, lai tai vienkārši varētu pievienot jaunus datu avotus neatkarīgi no to tipa, un lai tā būtu mērogojama, t.i. spētu apstrādāt pieaugošu datu apjomu. [1, 5, 11, 4]

Pēc tam, kad dati ir atlasīti no datu avotiem, notiek datu transformēšana, lai iegūtu datu noliktavā izmantojamu vienota formāta informāciju. Viens no galvenajiem uzdevumiem datu transformācijas solī risināmajiem uzdevumiem, ir datu kvalitātes uzlabošana. Galvenie datu kvalitātes kritēriji ir datu pilnīgums jeb tas, cik daudz no visiem nepieciešamajiem datiem ir pieejams datu noliktavā, nepretrunīgums, korektums, atbilstība formātu prasībām, precizitāte un integritāte. Tā kā dati var tikt iegūti no dažādām sistēmām, ir jāatpazīst un jāapvieno ieraksti, kas apraksta vienu un to pašu entītiju. Pie tam dati var būt iegūti no avotiem ar dažādiem datu kvalitātes standartiem, kas rada nepieciešamību aizpildīt iztrūkstošas vai standartizēt vērtības, kas ir iegūtas no datu avota. Datu transformācijas laikā veicamos uzdevumus var iedalīt sekojoši:

- Papildus datu atlase – atkarībā no datu avota struktūras visas nepieciešamās informācijas atlase datu izvilkšanas posmā var būt neefektīva. Šādos gadījumos iztrūkstošo informāciju var būt nepieciešams atlasīt datu transformācijas posmā;
- Ierakstu atdalīšana vai savienošana – iegūtos ierakstus var būt nepieciešams sadalīt, lai tie atbilstu datu noliktavas datu modelim, vai apvienot ierakstus no dažādiem datu avotiem, kas apraksta vienu un to pašu faktu;
- Pārvēršana – datu pārvēršana tiek veikta, lai standartizētu no dažādiem datu avotiem iegūtās vērtības, lai atbrīvotos no pretrunām vienas un tās pašas vērtības reprezentācijā, un padarītu vērtības saprotamas lietotājiem;
- Agregācija – visbiežāk datu noliktavas lietotājiem neinteresē informācija tādā pašā detalizācijas pakāpē, kā tā ir pieejama operacionālajā sistēmā, tāpēc datus var grupēt atbilstoši lietotāja prasībām;
- Papildināšana – ieraksta lauku vienkāršošana, lai tos padarītu lietojamākus datu noliktavā. [4, 5, 8]

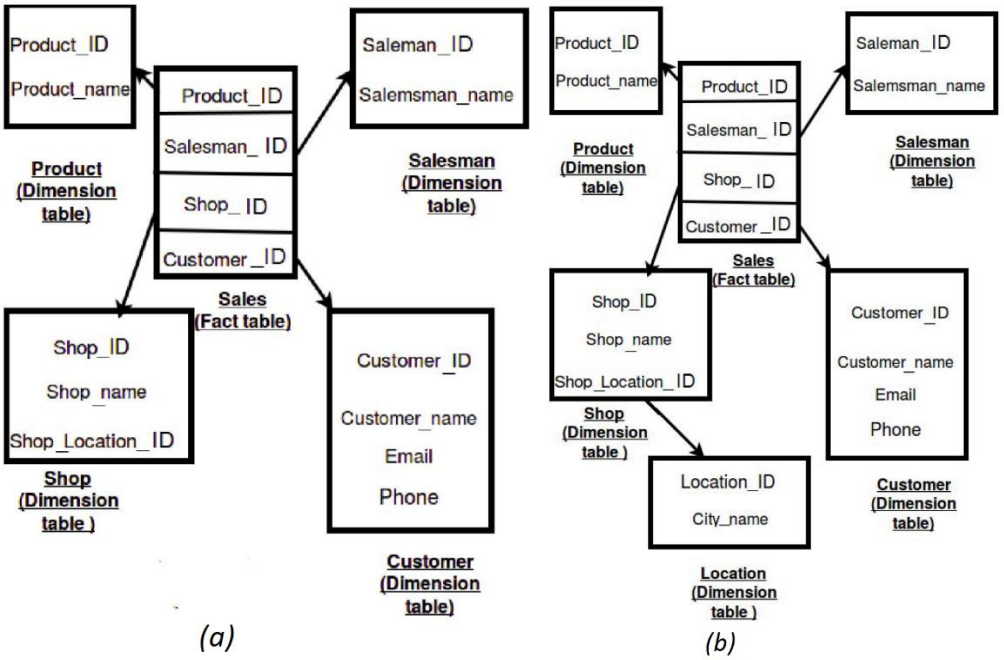
Tā kā datu kvalitātes nodrošināšana ir svarīgs faktors veiksmīga BI rīka realizācijā, tad šiem transformācijas procesiem ir jāpievērš īpaša uzmanība, jo tieši šajā solī tiek nodrošināta datu integrācija un datu kvalitāte. [11]

Tad, kad dati ir transformēti formā, kādā tos var ielādēt datu noliktavā, sākās datu ielādes solis. Šajā solī ievietošanai datu noliktavā sagatavotie dati tiek tajā saglabāti. Lielā datu apjoma dēļ, ko jāielādē datu noliktavā, un datu noliktavā izveidoto indeksu dēļ šis solis nav triviāls un prasa pietiekoši nopietnus plānošanas darbus. Lai paātrinātu datu ielādi datu noliktavā var būt nepieciešams dzēst implementētos indeksus pirms datu ielāde ir sākta un izveidot no jauna pēc datu ielādes pabeigšanas. Parasti datu noliktavās ir realizētas izstādes, integrācijas un pieejas slāņos, kur integrācijas slānī tiek glabāti neizmainīti dati, kas ir iegūti no avota sistēmas, bet integrācijas slāni tiek glabāti dati pēc to transformācijas, un pieejas slānis ir pati datu noliktava, kur dati glabājas sadalīti faktos un dimensijās. Pēc tam, kad ar datiem tiek veiktas datu noliktavas slānim atbilstošās darbības un transformācijas, tie tiek ielādēti nākamajā slānī. Līdz ar to datu ielāde var aizņemt ilgu laiku un datu noliktava ielādes laikā parasti nav spējīga strādāt, tāpēc ir jāmeklē laika periodi, kuros būtu pieņemams datu noliktavu padarīt nepieejamu. Kamēr notiek datu ielāde datu noliktavā, var izmantot datu bāzes trigerus, lai palīdzētu nodrošināt datu kvalitāti. Šajā procesā ir jāiestrādā algoritmi, kā apstrādāt gadījumus, kad dati netiek ielādēti datu noliktavā transformācijas procesā pieļautas kļūdas vai kāda cita iemesla dēļ. Tāpat ir nepieciešams izstrādāt procesus, ar kuru palīdzību būtu iespējams pārbaudīt datu kvalitāti pēc to ielādes datu noliktavā. [4, 8, 10]

Ir pieejami dažādi rīki un risinājumi, kas ļauj ērti realizēt un automatizēt ETL procesus. Šādi rīki piedāvā ne tikai ērtāku veidu, kā realizēt ETL procesus, bet arī papildus funkcionalitāti, kas palīdz uzturēt datu noliktavu, kā arī izsekot kļūdainu datu izcelsmi un transformācijas procesus. Ar šādu rīku palīdzību ir iespējams noteikt, vai dati ir bijuši kļūdaini jau avota sistēmā un kļūda nav korekti apstrādāta ETL procesā, vai arī kļūda ir radusies pašā ETL procesā, un ir nepieciešams šo procesu izlabot. Šajos rīkos var būt iekļauta arī funkcionalitāte, kas palīdz veikt izmaiņu ietekmes analīzi gadījumos, kad var būt nepieciešams veikt izmaiņas jau izveidotās tabulās, indeksos vai nodalījumos. Visām šim izmaiņām ir nepieciešams veikt izmaiņu ietekmes analīzi, lai apzinātu visus datu bāzes objektus, kurus ieplānotās izmaiņas var ietekmēt. Populārākie šāda veida rīki ir *Informatica, IBM Infosphere Datastage, Ab Initio, SQL Server Integration Services (SSIS)* un *Oracle Data Integrator.* [9]

Ir vairākas datu noliktavā ielādēto datu analīzes metodes. To, kādas metodes tiek izmantotas datu noliktavā ielādēto datu analīzei, parasti nosaka lietotāju prasības un jautājumi, uz kuriem datu noliktavai ir jāsniedz atbildes. Tipiskas metodes datu noliktavas datu analīzei

ir tiešsaistes analītiskā apstrāde jeb OLAP, dažādu atskaišu veidošana, datizrace un uzņēmumiem pielāgotas meklēšanas programmas, kas var veikt meklēšanu pilnā tekstā izmantojot atslēgas vārdus. Lai veiksmīgi ieviestu šīs analīzes metodes, datu noliktavas arhitektūrai ir jābūt tam piemērotai. Piemēram, lai veiksmīgi realizētu tiešsaistes analītisko apstrādi, datu noliktavai visticamāk ir jābūt realizētai izmantojot zvaigznes shēmu, kura paredz fakta un dimensiju arhitektūru un kura ar atbilstoši veidotiem skatiem datu noliktavā ļauj veikt fakta analīzi pa dimensijām. Populārs zvaigznes shēmas atvasinājums ir sniegpārslas shēma, kura pieļauj to, ka dimensiju tabulas atribūti var būt tālāk sadalīti apakšatribūtos. Šo shēmu piemēri ir redzami attēlā 1.2. Šīs analīzes metodes parasti ir iestrādātas starpserverī starp datu noliktavu un lietotāja saskarni, lai efektīvi iegūtu datus no datu noliktavas un sagatavotu tos attēlošanai lietotājam. [5, 9, 7]



1.2. att. **Datu noliktavu datu shēmu piemēri:**

(a) zvaigznes shēma, (b) sniegpārslas shēma [9]

Starpserveri, kuros ir pieejamas šīs analīzes metodes, parasti ir iestrādāti kādā BI lietotnē, kas ļauj ērti piekļūt šīs analīzes rezultātiem. Šādos rīkos parasti ir pieejamas tajos iepriekš iestrādāti uz noteiktiem parametriem balstīti datu attēlojumi, kā grafiki, diagrammas un informācijas paneļi, kā arī iespējas izpildīt ekspromtvaicājumus. Veiksmīgai datu attēlošanai lietotājam viegli uztveramā veidā ir svarīgi, lai BI rīkā būtu iestrādāti vairāki dažādi katra attēlošanas tipa veidi. Šādā veidā palielinās iespēja, ka risinājumā būs pieejams tāds attēlošanas veids, kas nepieciešams konkrētajā gadījumā. Var izmantot arī sarežģītākus risinājumus, kas balstoties uz informāciju datu noliktavā ar datizraces palīdzību spēj izstrādāt dažādus nākotnes prognozes modeļus. Visiem šiem pieejamajiem attēlošanas tipiem ir jāspēj attēlot lielas un sarežģītas rezultātu kopas, un tiem ir jābūt interaktīviem, lai lietotājs no

lietotāja saskarnes var ērti piekļūt jauniem rezultātiem balstoties uz to, kas jau ir redzams lietotāja saskarnē. [12, 4, 13]

Mūsdienās BI rīki attīstās no klasiskiem BI rīkiem, kas ir balstīti uz klasiskām datu pārvaldības metodēm un datu noliktavām uz tīmeklī bāzētiem rīkiem, kas ļauj analizēt no sīkdatnēm un serveru žurnālu datnēm ievadītus lietotāju datus, kā arī analizēt informāciju, ko lietotāji ir radījuši sociālo mediju platformās. Šādi rīki var atbildēt uz dabiskas valodas lietotāju vaicājumiem, veikt teksta analīzi, kā arī piedāvā dažādas attīstītas datizraces metodes. Tālāka šo rīku attīstība ļautu lietotājiem piekļūt BI sistēmām no savām viedierīcēm, un BI rīki spētu lietotājam piedāvāt viņu interesējošu informāciju balstoties uz ierīci, no kuras lietotājs ir piekļuvis sistēmai, lietotāja atrašanās vietas un cita konteksta. Šāda virziena attīstība ir svarīga, jo viedierīču skaits strauji pieaug, un nepieciešamība ērti piekļūt BI rīkiem no šīm ierīcēm palielināsies. Šī iemesla dēļ ir svarīgi strādāt arī pie veidiem, kā efektīvi vizualizēt datus šajās ierīcēs un nodrošināt efektīvu ierīces lietotāju saskarni. Citi nozīmīgi BI rīku attīstības virzieni ir lielo datu analīze, teksta analīze, tīmekļa analīze un tīkla analīze. [13]

Aprakstītie ETL procesi nav triviāli un to izpildes laiks var būt mērāms stundās, tāpēc tos parasti izpilda reizi noteiktā periodā, piemēram, mēnesī, dienā vai stundā. Līdz ar to dati, kas ir pieejami tipiskās datu noliktavās ir vēsturiski un piemēroti stratēģiskai lēmumu pieņemšanai. Mūsdienās industrijai palielinās tendences datu noliktavas izmantot, lai palīdzētu pieņemt taktiskus lēmums ikdienas uzņēmuma darbībā. Šī iemesla dēļ ir nepieciešami risinājumi, kas datu noliktavās un BI rīkos piegādā informāciju gandrīz bez aiztures, jeb reālā laikā. [4]

2. REĀLLAIKA DATU NOLIKTAVAS UN BI SISTĒMAS

Datu apstrāde reālā laikā ir arvien biežāk sastopama prasība darbā ar datu noliktavām, jo pieaug lietotāju prasības pēc svaigiem datiem un atsevišķos gadījumos datu piegāde ar pāris sekunžu vai minūšu novēlošanos var mazināt šo datu vērtību. Reāla laika BI sistēmas palielina informācijas vērtību saīsinot laika periodu, kas ir nepieciešams, lai analītiskā informācija nonāktu pie lēmuma pieņēmēja. Galvenās ar reāla laika datu noliktavām atrisināmās problēmas ir datu gaidīšanas laika samazināšana un izmantojamas informācijas iegūšana no liela apjoma datiem. Tas nozīmē, ka reāla laika datu noliktavas mērķis ir samazināt laiku no jaunas informācijas rašanās brīža līdz no tās izrietošai nepieciešamai darbībai. Galvenie ieguvumi no reāla laika BI sistēmas ir:

- Iespēja redzēt, kas notiek uzņēmumā, pirms tiek saņemtas finansiālās atskaites;
- Iespēja mācīties un veikt novērojumus no palielinātās informācijas plūsmas;
- Palielināta iespējas veiksmīgi paredzēt nākotnes notikumus;
- Iespēja uzņēmumam būt preventīvam lēmumu pieņemšanā;
- Atvieglota adaptīvu un automatizētu lēmumu pieņemšana.

Lai veiksmīgi realizētu reāla laika datu noliktavu un biznesa inteliģences sistēmu, ir svarīgi izprast piegādājamās informācijas darījumprasības un izmainīt ETL arhitektūru tā, lai tā spētu nogādāt datus pie lietotājiem ātrāk. Pie tam izmaiņas var būt nepieciešamas ne tikai datu noliktavā un izmantotajos BI rīkos, bet arī uzņēmuma darījumu procesos un organizācijā, lai veiksmīgāk izmantotu un atbalstītu reāllaika BI rīku. Tā kā reāla laika datu noliktavas galvenais mērķis ir datu gaidīšanas laika samazināšana pie lieliem datu apjomiem, tad šādās sistēmās var būt jārēķinās ar datu kvalitātes kompromisiem un specializētu tehnisko nodrošinājumu. [12, 14, 15, 16, 17]

Šobrīd nav izstrādāta precīza reāla laika datu noliktavas definīcija. Par reāla laika sistēmu ir uzskatāma sistēma, kurai ir jāapstrādā ārēji stimuli noteiktā laika periodā. Ja sistēmai stimula apstrādāšanai ir vismaz viens noteikts termiņš, tad sistēma ir uzskatāma par patiesu reāla laika sistēmu, pretējā gadījumā sistēma ir uzskatāma par nosacītu reāla laika sistēmu. Par reāla laika datu noliktavu var uzskatīt sistēmu, kas atspoguļo precīzu, saskaņotu un apvienotu skatu uz uzņēmuma stāvokli pirms noteikta laika perioda. Atšķirībā no klasiskām datu noliktavām, kas izmanto tikai vēsturiskos uzņēmuma datus, reāla laika datu noliktava satur arī šī brīža aktuālo uzņēmuma informāciju. Galvenās īpašības, kurām ir jāpiemīt reāla laika datu noliktavai ir augsta pieejamība, zems gaidīšanas laiks un horizontāla mērogojamība. Tas nozīmē to, ka datu noliktavai ir jāspēj apstrādāt jaunus datus jebkurā

laikā, jāspēj tos apstrādāt ātri, kā arī risinājuma veiktspējai ir jāuzlabojas, ja tam tiek piešķirti papildus resursi. Klasiskie ETL procesi nespēj nodrošināt visas šīs īpašības, tāpēc, lai realizētu reāla laika datu noliktavu ir jāmeklē jaunas pieejas un tehnoloģijas, ar kuru palīdzību šī prasības būtu iespējams nodrošināt. Jauni risinājumi ir nepieciešami katrā no ETL procesa posmiem. Ir iespējami vairāki risinājumi, kas nodrošinātu datu svaigumu, reāla laika datu noliktavas veiktspēju un pieejamību, kā arī tās iespēju atbalstīt pareģojošu taktisku lēmumu pieņemšanu. [18, 19, 20]

Ir vairāki veidi kā ir iespējams realizēt reāla laika datu noliktavas. Vienkāršākais veids, kā panākt to, ka dati datu noliktavā ir svaigāki, ir biežāka optimizētu pakešuzdevumu izpilde. Šādā veidā var panākt to, ka dati datu noliktavā tiek ielādēti reizi stundā vai biežāk, ja tas ir nepieciešams. Diemžēl, šāda risinājuma gadījumā datu noliktava biežāk nav pieejama lietotājiem, un nav iespējams nodrošināt svaigāko datu no avota sistēmām pieejamību datu noliktavā. Šo iemeslu dēļ šāds risinājums nav uzskatāms par reāla laika datu noliktavu, bet gan par gandrīz reāla laika datu noliktavu. Alternatīvs risinājums, lai nodrošinātu reāla laika datu noliktavas darbību, ir izmainīto datu vākšana (no angļu - *changed data capture*). Šī metode paredz to, ka datu noliktavā tiek nogādāti tikai tie dati, kas ir no jauna izveidoti vai izmainīti avota sistēmā kopš iepriekšējās ielādes. Vēl viena pieeja reāla laika datu noliktavas realizācijai ir sūces un apmaiņas metode, kurā dati tiek ielādēti pagaidu izstādes vidē (no angļu *staging area*), kas ir aktīvās datu noliktavas kopija. Šī pieeja paredz periodisku aktīvās datu noliktavas aizvietošanu ar pagaidu izstādes tabulām, un aizvietošanas brīdī tiek izveidota pagaidu izstādes vides kopija, kas kļūst par jauno pagaidu izstādes vidi. Reāla laika datu noliktavu ir iespējams izveidot arī ārējo reāla laika datu kešatmiņu, kura glabā reāla laika datus, un tiek integrēta ar statisko datu noliktavu. [9, 17, 18]

2.1. Reāla laika datu noliktavu arhitektūra

Reāla laika datu noliktavām ir jāspēj jaunākos datus integrēt jau eksistējošajā vēsturisko datu kopā. Visbiežāk, lai realizētu šo jaunāko datu integrēšanu datu noliktavā, izmanto atsevišķu reāla laika datu bāzes serveri vai nodalījumu. Ar šādu risinājumu pieļaujams, ka dati reāla laika nodalījumā var būt neprecīzi, jo tos vēlāk var pārrakstīt ar klasiskajiem ETL procesiem, nodrošinot to, kas sistēma ir galu galā precīza. Realizējot datu noliktavas karsto nodalījumu ir jāpatur prātā tas, ka datu noliktavai ir jāspēj tajā efektīvi veidot jaunus ierakstus, kā arī veikt datu atlasi, sapludinot iegūtos rezultātus ar jau eksistējošajiem datiem

statiskajā datu noliktavā. Lai gan datos, kas tiek ielādēti reāla laika datu noliktavas nodalījumā, ir pieļaujamas neprecizitātes, tiem vienalga ir jābūt pietiekoši precīziem, lai radītu korektu iespaidu par notikumiem avota sistēmās. Reālie datu bāzes nodalījumi ir atšķirīgi gadījumos, kad statiskā datu noliktava ir transakcijas detalizācijas līmenī, kad statiskā datu noliktava ir periodisks avota sistēmas momentuzņēmums un kad reālajā datu bāzes nodalījumā ir jāglabā uzkrājošs avota sistēmas momentuzņēmums. Visos gadījumos reāla laika datu bāzes nodalījumam būtu jāatbilst sekojošām prasībām:

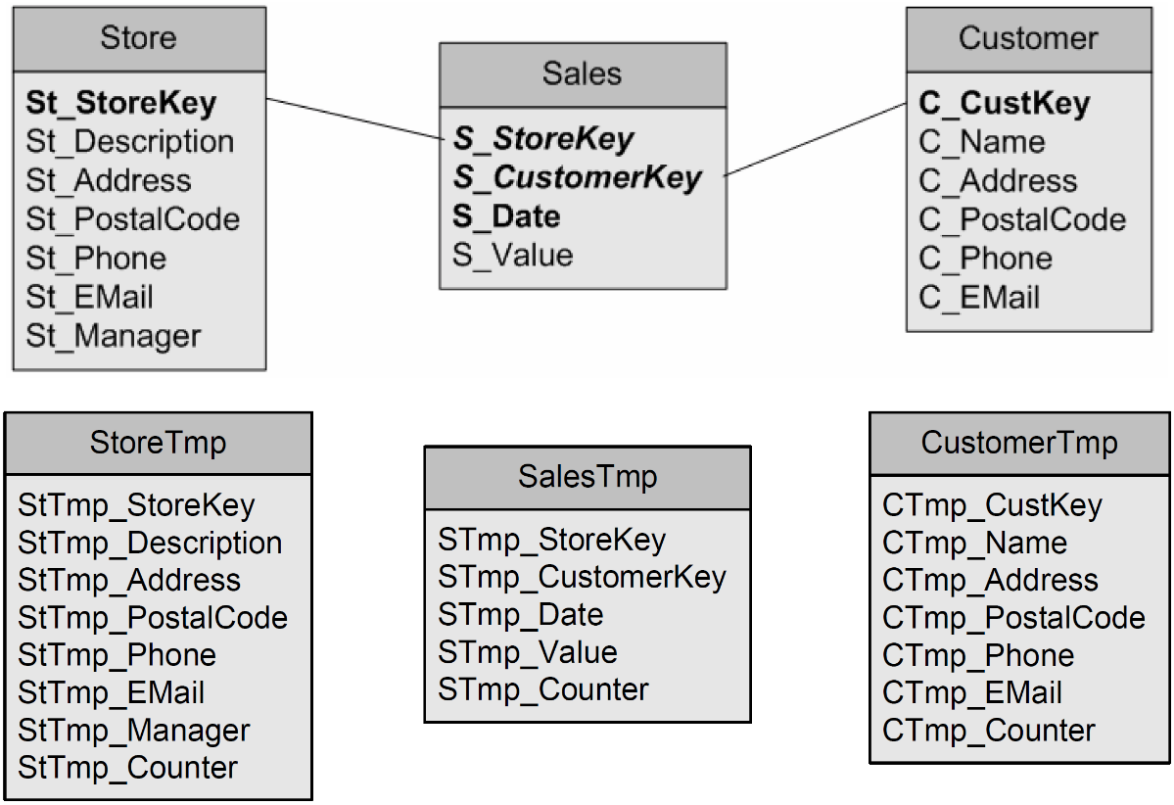
- Tam ir jāsatur visa informācija, kas ir radīta pēc tam, kad statiskā datu noliktava tika pēdējo reizi atjaunota;
- Tam ir nemanāmi jāsaplūst ar statiskās datu noliktavas faktu tabulas arhitektūru, tas varētu būt pat fizisks statiskās datu noliktavas faktu tabulas nodalījums;
- Tam ir jābūt minimāli indeksētam vai neindeksētam, lai atbalstītu nepārtrauktu jaunu datu ierakstīšanu;
- Jānodrošina īsi atbildes laiki lietotāju vaicājumiem, piemēram, glabājot nodalījumu atmiņā. [12, 20, 14, 16]

Ja statiskās datu noliktavas faktu tabula ir transakcijas detalizācijas līmenī, tad reāla laika nodalījuma faktu tabulai ir jāsatur tādas pašas dimensijas, kā statiskās datu noliktavas faktu tabulai. Reāla laika nodalījuma faktu tabulai ir jāsatur informācija par visām transakcijām, kuras ir radītas pēc statiskās datu noliktavas pēdējās atjaunošanas. Šī nodalījuma faktu tabulai ir jāsatur viens ieraksts par katru attiecīgo transakciju, kas ir notikusi avota sistēmā. Reāla laika nodalījums var būt neindeksēts, lai tajā varētu nepārtraukti rakstīt informāciju. Ideālā gadījumā, ja reāla laika nodalījuma faktu tabulai ir jāuztur informācija tikai par vienu dienu, tad tajā var nerealizēt laika dimensiju. Lai reāla laika nodalījums atbalstītu gan ātru rakstīšanu, gan īsu atbildes laiku lietotāju vaicājumiem, statisko datu noliktavu var būt nepieciešams atjaunot pietiekoši bieži, lai reāla laika nodalījums tiktu regulāri iztīrīts un tā datu apjoms nekļūtu tik liels, ka būtu nepieciešama indeksācija, kas, savukārt, var palēnināt datu rakstīšanu šajā nodalījumā. Lai palielinātu reālā laika nodalījuma veiktspēju gan datu rakstīšanai, gan atlasei, šo nodalījumu var glabāt atmiņā esošā datu bāzē, ja pieejamais atmiņas apjoms to atļauj. [12, 20, 16]

Ja statiskās datu noliktavas faktu tabula ir periodiskā detalizācijas līmenī, piemēram, fakti satur informācijas apkopojumu par mēnesi, tad reāla laika nodalījumu var uzskatīta par tekošā mēneša attēlojums. Šajā gadījumā reāla laika nodalījuma faktu tabula glabā informācijas apkopojumu par doto faktu. Tas nozīmē, ka reāla laika faktu tabulā fakti tiek pievienoti, kad datu noliktava saņem jaunus faktus, un izmainīti, kad datu noliktava saņem informāciju par izmaiņām jau reģistrētā faktā. Šādā veidā tiek nodrošināts, tas, ka reāla laika

datu nodalījuma faktu tabula satur visu informāciju par tekošo periodu, kas vēl nav ielādēta statiskajā datu noliktavā. Līdz ar to ierakstu skaits faktu tabulā visticamāk nebūs pārmērīgi liels arī perioda beigās, kas nozīmē to, ka faktu tabula var nebūt indeksēta, un efektīvai darbībai var tikt glabāta atmiņā. Šajā gadījumā, ja reāla laika nodalījums ir veiksmīgi realizēts, tad perioda beigās to var vienkārši pievienot statiskās datu noliktavas faktu tabulai, reāla laika nodalījumu iztukšot un procesu sākt no jauna. [12, 20, 16]

Reāla laika datu bāzes nodalījumus, kas glabā uzkrājošus avota sistēmas momentuzņēmumus, var izmantot gadījumos, kad reāla laika datu noliktavā ir jāglabā informācija par procesiem, kas notiek kādu laika periodu. Šādā gadījumā reāla laika nodalījums satur uzkrājošu informāciju par šo procesu un datu noliktavas fakts tiek atjaunots ar jaunāko saņemto informāciju par šo faktu. Piemēram, ja datu noliktavā tiek glabāta informācija par transakcijām, kura dienas laikā var mainīties, tad statiskā datu noliktava glabā informāciju par transakciju kāda tā bija dienas sākumā, bet reāla laika nodalījums glabā informāciju par aktuālo transakcijas statusu, kas ļauj dienas beigās statiskās datu noliktavas faktu aizstāt ar reāla laika nodalījuma faktu.



2.1. att. **Vienkāršs reāla laika nodalījuma tabulu shēmas piemērs [14]**

Tabulas datu noliktavas reāla laika nodalījumā var veidot kā pilnīgas attiecīgo statiskās datu noliktavas tabulu kopijas ar vienu papildus lauku, kurš glabā kārtību kādā ieraksti reāla laika nodalījumā ir izveidoti. Šāda veida reāla laika nodalījuma arhitektūras piemērs ir redzams attēlā 2.1. Ja reāla laika nodalījuma arhitektūra ir izveidota šādā veidā un to ir

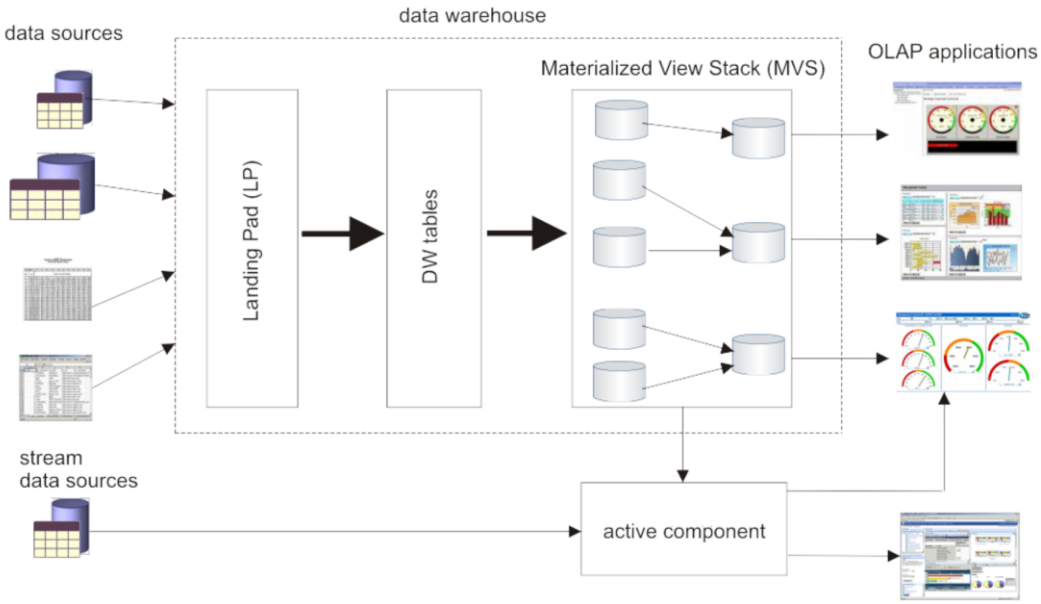
iespējams aizpildīt reālā laikā, tad šāda struktūra var atvieglot statiskās datu noliktavas atjaunošanu, jo tam par pamatu var ņemt šo reāla laika nodalījumu. Tā kā šādā arhitektūrā pieņem, ka dati reāla laika nodalījumā ir korekti, tad, lai atjaunotu statisko datu noliktavu pietiek veikt korektu datu agregāciju pēc statiskās datu noliktavas faktu tabulas atslēgām un saglabāt tos statiskajā datu noliktavā. Tā kā šādā veidā datu atjaunošana ir pietiekoši ātra, tad šādi ir iespējams samazināt laiku, kuru datu noliktava ir slēgta atjaunināšanai un nav pieejama lietotājiem. [14]

2.2. att. **Divu reāla laika nodalījumu uzturēšanas piemērs [21]**

Reāla laika datu noliktavā var būt nepieciešams uzturēt divus reāla laika nodalījumus. Tas ir nepieciešams, jo klasiskie ETL procesi izpildās pietiekoši ilgi, un šajā laikā var tikt radītas jaunas transakcijas avota sistēmā, kas arī ir jāieraksta reāla laika datu noliktavā. Šī iemesla dēļ rodas problēmas ar reāla laika nodalījuma iztīrīšanu. Ja reāla laika nodalījums tiek iztīrīts tad, kad klasiskais ETL process sāk darbu, tad uz šī procesa laiku reāla laika datu noliktavai nav pieejama informācija par periodu, kurš šobrīd tiek ielādēts statiskajā datu noliktavā, jo tas datu noliktavā ir pilnībā pieejams tikai pēc procesa beigām. Ja reāla laika nodalījums tiek iztīrīts tad, kad ETL process beidz darbu, tad transakcijas, kuras ir radītas kamēr statiskā datu noliktava tiek atjaunota, vairs nav pieejamas reāla laika nodalījumā un datu noliktavā kopumā. Šajā gadījumā šīs transakcijas datu noliktavā tiks ierakstītas tikai tad, kad statiskā datu noliktava tiks atjaunota nākamo reizi. Šo problēmu var risināt datu noliktavā realizējot divus reāla laika nodalījumus, kuros abos dati tiek rakstīti paralēli un lietotājam ir pieejams tikai viens no šiem nodalījumiem, kuru sauc par karsto nodalījumu. Uzsākot statiskās datu noliktavas ETL procesu, var iztīrīt neaktīvo reālā laika nodalījumu, un tad, kad ETL process darbu beidz, neaktīvais reāla laika nodalījums aizstāj karsto nodalījumu. Šāds risinājums nodrošina to, ka karstajā nodalījumā vienmēr ir pieejami tieši tie dati, kas ir radīti pēc tam, kad statiskā datu noliktava ir pēdējo reizi atjaunota. Šī risinājuma piemēra shēma ir redzama attēlā 2.2. [21]

Alternatīva datu noliktavas arhitektūra ir ELT (no angļu *Extract – Load – Transform).* ELT arhitektūru var izmantot, ja statiskā datu noliktava ir realizēta izmantojot materializētus datu bāzes skatus un šie skati piekļūst tikai noteiktām datu noliktavas datu apakškopām. Šādā arhitektūrā visi ienākošie dati tiek glabāti atsevišķi no statiskās datu noliktavas datiem slānī, ko sauc par ielādes laukumu. Šeit jaunie dati var tikt glabāti tabulās vai izplātā datnē.

Atšķirībā no ETL arhitektūras, dati ielādes laukumā tiek ielādēti tādā formātā, kādā tie ir pieejami avota sistēmā, un datu transformācija tiek veikta pēc šo datu ielādes, izmantojot datu bāzes materializētos skatus. Dati no ielādes laukuma tiek integrēti materializētajos skatos, kuri ir pieejami lietotājam. Materializētie skati tiek atjaunoti tikai tad, kad tiem piekļūst OLAP lietotne. Tas nozīmē to, ka materializētie skati var tikt atjaunoti jebkurā laikā balstoties uz lietotāja pieprasījumiem, un ka tiek atjaunoti tikai tie dati, kas ir nepieciešami lietotājiem, tādejādi efektīvāk izmantojot datu noliktavai pieejamos resursus. Šādā veidā var optimizēt arī datu noliktavas materializēto skatu atjaunošanu, jo vienā datu noliktavas atjaunošanas ciklā var tikt apstrādāti vairāki izmainīti ieraksti, kas nozīmē to, ka datu noliktava nav jāatjauno katru reizi, kad tā saņem jaunus datus, un tiek samazināts kopējais atjaunošanas ciklu skaits. Šādas ELT arhitektūras piemērs ir redzams attēlā 2.3. [22]

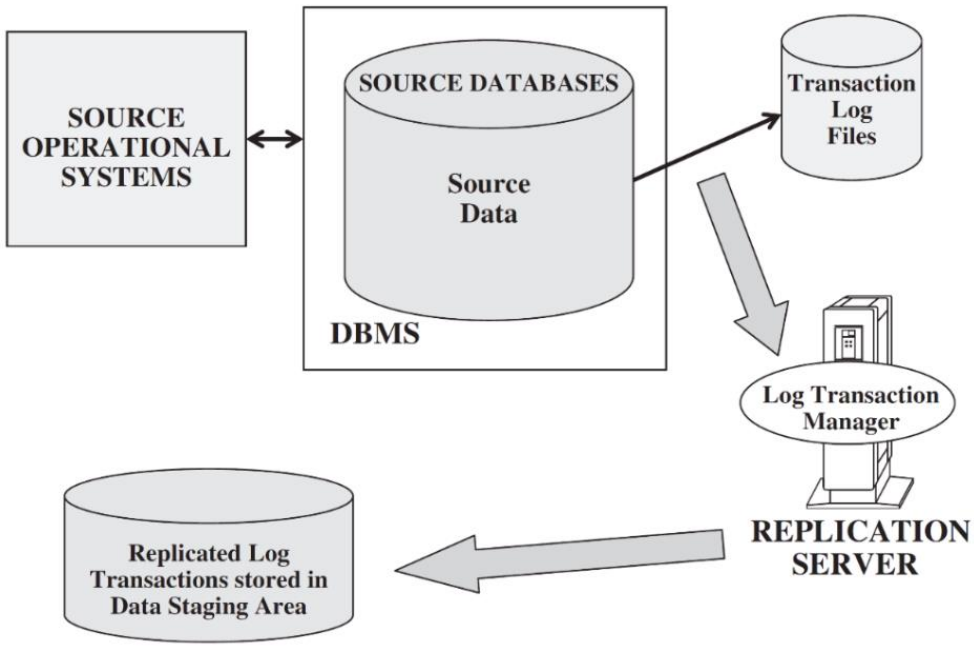


2.3. att. **ELT reāla laika datu noliktavas arhitektūra ar materializētiem skatiem [22]**

Ja datu noliktavai ir nepieciešams apstrādāt arī reāla laika datu straumi, šai arhitektūrai var pievienot aktīvo komponenti, kas apstrādātu šos datus un integrētu tos ar datu noliktavā esošajiem datiem. Aktīvā komponente var integrēt datu straumju datus ar datu noliktavā eksistējošiem datiem, jo tā var izpildīt vaicājumus datu noliktavā. Šādā veidā aktīvā komponente spēj noteikt, kad datu straume satur izlēcējus (no angļu *outlier*) un par šīm anomālijām brīdināt OLAP lietotni. Balstoties uz šiem brīdinājumiem var būt nepieciešams atjaunot datu noliktavu, lai tajā integrētu šos izlēcējus. [22]

2.2. Datu izvilkšana reālā laikā

Lai veiktu datu izmaiņu izvilkšanu no operacionālās sistēmas reālā laikā, ir pieejamas sekojošas iespējas: datu vākšana caur transakciju atkārtošanas (no angļu *redo*) žurnāla datnēm, datu vākšana caur datubāzes trigeriem, datu vākšana avota lietotnē, starplietojumprogrammu integrācija un uzņēmuma informācijas integrācija. Katrai no šīm iespējām ir savas pozitīvās un negatīvās īpašības. [4, 12]



2.4. att. **Shēma datu vākšanai no datu bāzes transakciju atkārtošanas žurnāla [4]**

Katra datu bāzes transakcija pievieno datu bāzes atkārtošanas žurnāla datnē ierakstu, par to, kas šajā transakcijā ir izdarīts, kas nozīmē to, ka datu bāzē veiktās izmaiņas var iegūt no šīm datnēm. Šo metodi sauc par datu izmaiņu vākšanu caur transakciju atkārtošanas žurnāla datnēm. Šī datu izvilkšanas metode lasa datu bāzes atkārtošanas žurnāla datnes un atlasa tās transakcijas, kuras ir neatgriezeniski iesūtītas (no angļu *commit*) datu bāzē. Šīs metodes shēma ir redzama attēlā 2.4. Šāds risinājums nerada papildu slodzi operacionālajai sistēmai, jo transakciju reģistrēšana ir daļa no datu bāzes pamata funkcionalitātes. Diemžēl, datu bāzē var tikt ierakstītas arī darījumu transakcijas, kas vēl nav pabeigtas un šī iemesla dēļ neatbilst visām darījumu prasībām. Tas nozīmē to, ka šie ieraksti var būt nepilnīgi vai nepareizi, jo papildu informācija datu bāzē var tikt ierakstīta vēlāk vai saturēt ārējās atslēgas, kuras datu noliktavā vēl nav ierakstītas. Šādu problēmu risināšanai var būt nepieciešami papildu risinājumi, piemēram, pakotnes darbs, kas pārraksta reāla laika datus reizi dienā ar datiem no transakciju datubāzes izmantojot klasisko ETL implementāciju. Tā kā, kad datu bāzes transakciju atkārtošanas žurnāla datnes kļūst pietiekoši lielas, tās tiek iztukšotas un izmantotas

no jauna, tad izmantojot šo pieeju ir svarīgi pārliecināties, ka visas atkārtošanas žurnāla datnē ierakstītās transakcijas iztukšošanas brīdī jau ir izvilktas no atkārtošanas žurnāla datnes. Šī datu izvilkšanas metode strādā tikai lietotnēm, kurām ir atkārtošanas žurnāla datnes. Tas nozīmē, ka, ja sistēmā tiek lietotas indeksētas vai atsevišķas datnes, kurām nav atkārtošanas žurnāla datņu, tad šīm datnēm ir jāizmanto cita datu izvilkšanas metode. [4, 12]

Arī datu vākšana caur datu bāzes trigeriem ir iespējama tikai lietotnēs, kuras ir balstītas uz datu bāzēm, kurās ir iespējams realizēt datu bāzes trigerus. Datu bāzes trigeris ir programma, kas tiek izpildīta datu bāzē, kad lietotājs veic izmaiņas datos, un tā var būt definēta specifiskai tabulai un specifiskam izmaiņu veidam. Lai realizētu datu vākšanu caur datu bāzes trigeriem, visiem notikumiem datu bāzē, kurus ir nepieciešams vākt apstrādei ETL procesā, ir jāizveido datu bāzes trigeri, kas šo izmaiņu ieraksta atsevišķā datnē, kas tiks izmantota, lai izvilktu datus apstrādei datu noliktavā. Ar šādu trigeru palīdzību ir iespējams tvert datu bāzes stāvokli gan pirms, gan pēc izmaiņām. Tā kā šis risinājums strādā tieši datu avotā, tas ir uzskatāms par drošu. Šī risinājuma negatīvās iezīmes ir tas, ka šādu datu bāzes trigeru izstrāde un uzturēšana ir papildu slogs izstrādātājiem un ka trigeru izpildes ir papildus darbības avota sistēmā, kas palēnina tās darbu. [4, 8]

Datu vākšana avota lietotnē nozīmē to, ka avota lietotne tiek izmainīta tā, ka tā visas izmaiņas, kas tajā tiek veiktas, raksta datnē, kas tiek izmantota izmaiņu vākšanai apstrādei ETL procesā. Atšķirībā no datu vākšanas caur datu bāzes trigeriem, šis izmaiņas tiek iestrādātas vispārējos avota sistēmas procesos, kas nozīmē to, ka šī izmaiņu vākšanas metode var tikt izmantota arī sistēmās, kas neizmanto datu bāzes. Diemžēl, arī šis risinājums ievieš papildus sarežģītību sistēmas izstrādes un uzturēšanas procesā, kas var nebūt triviāls sarežģītības palielinājums, ja avota sistēmu skaits ir liels. Pie tam, tāpat kā datu vākšana caur datu bāzes trigeriem, šāda papildus funkcionalitāte var samazināt avota sistēmu veiktspēju papildu slodzes dēļ, kas rodas saglabājot izmaiņas datnēs. [4, 8]

Lai veiktu datu izvilkšanu reālā laikā var izmantot arī starplietojumprogramu integrāciju. Starpprogrammu integrācijas nozīmē to, ka divas lietotnes ir savā starpā integrētas, tādejādi ļaujot tām apmainīties ar datiem, ziņām vai transakcijām. Tas nozīmē to, ka viena lietotne var brīvi piekļūt citas lietotnes datiem. Tā kā starplietojumprogrammu integrācija ir paredzēta, lai pārvietotu datus starp lietotnēm reālā laikā, tad to var izmantot, lai pārvietotu datus no operacionālās sistēmas uz ETL izstādes vidi. Šādā gadījumā operacionālā sistēma parasti nosūtīs izmaiņas uz izstādes vidi brīdī, kad tad tās tiek veiktas. Arī šī risinājuma negatīvās iezīmes ir līdzīgas, kā datu vākšanai caur datu bāzes trigeriem un datu vākšanai avota lietotnē, t.i. ir nepieciešams papildu darbs, lai realizētu un uzturētu operacionālās sistēmas integrāciju ar izstādes vidi, un šī metode var ietekmēt operacionālās

sistēmas veiktspēju papildu darbību dēļ, kas nepieciešamas, lai nosūtītu datus no avota sistēmas uz izstādes vidi. [4]

Vēl viena iespēja informācijas iegūšanai, lai to izmantotu BI sistēmās reālā laikā, ir uzņēmuma informācijas integrācija. Uzņēmuma informācijas integrācija ir metode, ar kuras palīdzību vairāki atšķirīgi uzņēmuma informācijas avoti tiek apvienoti vienā virtuālā skatā, pret kuru lietotājs var ērti izpildīt vaicājumus. Lietotājs šo skatu redz kā vienu informācijas avotu, bet skats lietotāja vaicājumu sadala apakšvaicājumos, kuri katrs tiek izpildīti uz atbilstošā informācijas avota. Šādi risinājumi vaicājumus izpilda avota sistēmā, kas nozīmē to, ka vaicājumi drīkst būt tikai ar ierobežotu sarežģītību, jo pretējā gadījumā tie var pārmērīgi noslogot avota sistēmu, un tādejādi palēninât tās darbu. Pie tam uzņēmuma informācijas integrācijas rīki iegūto informāciju nekur nesaglabā, kas nozīmē to, ka netiek izmantota reāla datu noliktava, bet rezultāti tiek nogādāti lietotājam tieši no avota sistēmas, bez jebkādiem starpniekiem. [4, 12]

2.3. Datu transformācija reālā laikā

Tā kā reāla laika datu noliktavās laiks, kas ir atvēlēts ETL procesiem ir ļoti mazs, tad var būt nepieciešams samazināt laiku, kas tiek tērēts veicot datu transformāciju. Lai to izdarītu, var būt nepieciešams vienkāršot transformācijas, kas tiek izpildītas ierakstiem, kas tiek nolasīti no avota sistēmas. Arī šim vienkāršotajam procesam vajadzētu mēģināt iespējami maksimāli nodrošināt datu attīrīšanu un pielāgošanu datu noliktavas datu formātam. To var izdarīt samērīgi samazinot datu kvalitātes pārbaudes, ierakstot faktus ar dimensijām un izslēdzot datu izstādes vidi. [12, 8]

Tā kā datu kvalitātes pārbaudes procesi var būt pietiekoši sarežģīti, lai veiktu datu transformāciju reālā laikā, var būt nepieciešams izslēgt sarežģītākās pārbaudes, kas tiek veiktas šajā procesā. Šī iemesla dēl, datu kvalitātes pārbaudes procesus ir ieteicams ierobežot tikai uz atsevišķu kolonu vērtību pārbaudi, izslēdzot datu struktūru pārbaudi un to atbilstības darījuma prasībām pārbaudi. Tas ir nepieciešams, jo datu struktūru pārbaude un darījuma prasību pārbaude var būt pietiekoši sarežģīti procesi, kas pārbaudi veic izmantojot vērtības no vairākām kolonām un tabulām, kā arī var veikt ārēju sistēmu izsaukumus. Savukārt, atsevišķu lauku pārbaudes parasti ir vienkāršas viena lauka validācija vai vērtības atrašana uzmeklēšanas tabulā, lai paplašinātu ierakstu. Diemžēl, sarežģītāko datu pārbaužu izslēgšana nozīmē to, ka kļūdas, ko šīs pārbaudes parasti atrastu nokļūst datu noliktavā, kas nozīmē to,

ka ir nepieciešami procesi, kas spēj attēlot šos kļūdainos datus, kā arī pakešuzdevumi, kas aizstāj reālā laikā iegūtos datus ar datiem, kas iegūti izmantojot tradicionālos ETL procesus. Lai gan zināmas neprecizitātes datos ir pieļaujamas, tiem ir jārada korekta avota sistēmu reprezentācija. [12, 20]

Reālās vidēs transakcijas var tikt radītas pirms datu bāzē tiek ierakstīts pilns dotās transakcijas konteksts. Tas nozīmē to, ka datu noliktavā fakts var ienākt pirms tam ir radītas visas nepieciešamās dimensijas. Tā kā reāla laika datu noliktava nevar gaidīt līdz brīdim, kad faktam tiks korekti ierakstītas visas tā dimensijas, šīs situācijas ir atsevišķi jāapstrādā. Šādos gadījumos faktus datu noliktavā var ierakstīt ar jau eksistējošu dimensiju kopijām vai arī izmantot noklusētas tukšas dimensijas. Kad informācijas avotā parādās pilna informācija par fakta dimensijām, to var uzreiz pārrakstīt karstajā reāla laika nodalījumā, vai arī gaidīt, kad pilnu informāciju datu noliktavā nogādās tradicionālais ETL process. [12, 20]

Ir iespējamas arī reāla laika datu noliktavu arhitektūras, kuras pilnībā izslēdz datu izstādes vides un datu ielādi pastāvīgā ETL uzglabāšanas vidē. Tas nozīmē, ka šādi risinājumi izpilda vaicājumus avota sistēmā un rezultātus atgriež sistēmas lietotājam. Šādas sistēmas nav ieteicams lietot kā daļu no datu noliktavas, jo visticamāk, ja šādas sistēmas tiek izmantotas, var nebūt iespējams nodrošināt datu noliktavas rezerves kopiju, atkopšanas, arhivēšanas un atbilstības prasības. Šādu sistēmu visbiežākais piemērs ir informācijas integrācijas sistēmas. [12]

2.4. Datu ielāde reālā laikā

Ir vairākas alternatīvas pieejas kā realizēt datu ielādi reāla laika datu noliktavā. Šīs pieejas ir tiešas sūces plūsma (no angļu *direct trickle feed*), sūces un apmaiņas metode (no angļu – *trickle and flip*) un ārēja reāla laika datu kešdarbe. [16, 12, 8, 7]

Tiešas sūces pieeja, ir pieeja, kad dati datu noliktavā tiek ielādēti nepārtraukti. Šo pieeju var realizēt divos dažādos veidos. Viens no veidiem ir datus nepārtraukti ielādējot statiskās datu noliktavas faktu tabulās. Tā kā statiskās datu noliktavas faktu tabulā var būt realizēti pietiekoši sarežģīti indeksi, lai paātrinātu datu atlasi no šīs faktu tabulas, tad šāds risinājums var nebūt efektīvs un datu ielādēšana var būt salīdzinoši lēna un prasīt daudz resursu. Otrs veids, kā realizēt tiešās sūces pieeju ir datus nepārtraukti ielādējot faktu tabulas atsevišķā reāla laika nodalījumā. Šādā veidā ir iespējams risināt neefektīvas datu ielādes problēmas indeksu dēļ, bet ir jārēķinās ar to, ka ir nepieciešams ieguldīt papildu darbu atbilstošu

vaicājumu izstrādei. Papildu tam šādā gadījumā ir jādomā par risinājumu, kā datus no reāla laika nodalījuma faktu tabulas ielādēt statiskās datu noliktavas faktu tabulā. Izvēloties šādu risinājumu ir jāņem vērā arī tas, ka nepārtraukti atjaunojot vienu un to pašu tabulu, vaicājumu izpildes laiks palielinās, kas ar laiku var kļūt par nopietnu problēmu. Tā kā datu ielāde datu noliktavā un vaicājumu izpilde datu noliktavā var būt ar dažādu noslogojumu, tad brīžos, kad datu noliktavas noslogojums ir lielāks, tās darbība var tikt apgrūtināta, jo datu bāzes pārvaldības sistēmas var bloķēt jaunus ienākošos datus, lai veiksmīgi atbildētu uz lietotāju vaicājumiem. Šo problēmu sauc par jauktā noslogojuma problēmu. [16, 8, 7]

Lai nodrošinātu datu noliktavas pastāvīgu pieejamību vaicājumiem un lēmumu pieņemšanas sistēmām un nodrošinātu tajā pieejamo datu integritāti, var lietot modeli, kas izmanto faktu tabulas shēmas duplikāciju ar visiem tās datu integritātes ierobežojumiem. Lai izmantotu šo modeli ir jāpieņem, ka operacionālā sistēma, no kuras tiek ņemti dati, pilnībā ievēro visus datu integritātes ierobežojumus. Datu duplikācija tiek veikta tikai dinamiskajiem reāla laika datiem, un tā tiek veikta tikai gadījumos, kad datu noliktava ir noslogota un nespēj integrēt jaunus ienākošos datus. Šajos gadījumos dati tiek ierakstīti pagaidu dublikāta shēmā, kurā papildus primārās faktu tabulas laukiem, ir lauki, kas parāda, vai dotais ieraksts ir integrēts primārajā faktu tabulā un laiku, kad šī izmaiņa ir veikta. Vaicājumi šīs dublikāta faktu tabulas savieno ar primārajām faktu tabulām, kas nozīmē to, ka lietotāji vēljoprojām var piekļūt reāla laika datiem. Tad, kad kļūst skaidrs, ka primārajā faktu tabulā var integrēt jaunus datus, dati no dublikāta faktu tabulas tiek ierakstīti primārajā faktu tabulā un tiek dzēsti no dublikāta faktu tabulas. Šis modelis paredz arī to, ka, gadījumos, kad dublikāta faktu tabula ir noslogota un nespēj apstrādāt jaunas izmaiņas, rekursīvi var veidot jaunas dublikāta faktu tabulas. Nepieciešamības gadījumā šis modelis paredz arī dimensiju tabulu apstrādi analogā veidā. [23]

Sūces un apmaiņas pieeja ir atvasināta no tiešās sūces pieejas ar mērķi risināt jauktā noslogojuma problēmu. Šī metode izmanto reāla laika nodalījumu, kurā ir realizēta izstādes faktu tabula ar tieši tādu pašu struktūru, kā atbilstošās faktu tabula statiskajā datu noliktavā. Arī šāda veida risinājumā dati tiek nepārtraukti ielādēti datu noliktavas reāla laika nodalījumā, bet atšķirībā no tiešās sūces pieejas, datu noliktava periodiski tiek slēgta un reāla laika nodalījuma faktu tabula tiek nokopēta, reāla laika nodalījums tiek pārsaukts par aktīvo datu noliktavas tabulu un process tiek atsākts no jauna, datus ielādējot jaunā izstādes tabulā. Šādā veidā var risināt jaukta noslogojuma problēmu, jo dati tiek rakstīti izstādes tabulās, bet vaicājumi tiek izpildīti uz aktīvās faktu tabulas, un izstādes tabulas periodiski tiek padarītas par aktīvajām faktu tabulām, tādējādi padarot jaunākos datus pieejamus lietotājam. Šo datu apmaiņu var realizēt vai nu, kā vienkāršu norāžu apmaiņu, kas ir ļoti ātra darbība, bet prasa

kopēt izstādes tabulu, lai radītu jaunu izstādes tabulu, vai arī izstādes tabulas datus pievienojot aktīvas datu noliktavas faktu tabulai. Šī risinājuma galvenā problēma ir tāda, ka gadījumos, kad faktu tabulas kļūst pietiekoši lielas, to kopiju izveidošana, lai radītu jaunas izstādes tabulas var prasīt daudz laika, kas nozīmē to, ka datu noliktava ilgstošus laika periodus var nebūt pieejam lietotājiem. Šajā gadījumā datu kopēšanas ilgums var svārstīties no vairākām minūtēm līdz vairākām stundām, kas nozīmē to, ka pie lieliem datu apjomiem šāds risinājums nespēj veikt datu apstrādi reālā laikā. [16, 8, 7]

Alternatīvs risinājums ir izmantot ārēju reāla laika kešdarbes metodi. Šādā risinājumā reāla laika datu iegūšanai tiek izmantots atsevišķs datu bāzes serveris, kurš izpilda ETL procesus reālā laikā. Lai šis atsevišķais reāla laika serveris spētu apstrādāt lielu apjomu datu reālā laikā, to var realizēt atmiņā esošā datu bāzē, kas spēj īsā laika periodā apstrādāt lielu daudzumu izmaiņu datos, kā arī ātri atbildēt uz lietotāju vaicājumiem. Ar šāda risinājuma palīdzību ir iespējams nodrošināt datu ielādi datu noliktavā reālā laikā. Šādā veidā var risināt arī datu noliktavas mērogojamības problēmu un jauktā noslogojuma problēmu, jo vaicājumi, kuri tiek izpildīti uz reāla laika datiem tiek novirzīti uz šo ārējo reāla laika kešatmiņu, kura ir spējīga tos apstrādāt. Šāds risinājums nerada papildu noslogojumu statiskajai datu noliktavai, jo reāla laika dati glabājas atsevišķā datu bāzē, un tas ļauj lietotājam ātri piekļūt datiem, kas ir tikko radīti avota sistēmā. Lielākā šāda risinājuma problēma ir tāda, ka gadījumos, kad ir nepieciešams atbildēt uz vaicājumiem, kuriem ir nepieciešami dati gan no statiskās datu noliktavas, gan no reāla laika datu bāzes servera vai nodalījuma, vaicājuma izpilde var prasīt daudz resursu. [16, 12, 8, 7]

Neatkarīgi no tā, kurš no aprakstītajiem risinājumiem tiek izmantots, reāla laika dati periodiski ir jāielādē statiskajā datu noliktavā, tāpēc ir svarīgi optimizēt šo procesu. Faktors, kas var atvieglot šo procesu ir tas, vai ir pieļaujami salīdzinoši īsi periodi, kad datu noliktava tiek padarīta nepieejama lietotājiem. Šis faktors ir svarīgs, jo paralēla lietotāju vaicājumu apstrāde un jaunu datu ielāde datu noliktavā, kardināli samazina datu ielādes veiktspēju, kas var nozīmēt to, ka datu noliktava nav spējīga apstrādāt jaunus datus, jo visi tās resursi ir noslogoti, lai apstrādātu lietotāju vaicājumus. Lai padarītu datu ielādi statiskajā datu noliktavā efektīvāku, ir svarīgi, vai ir iespējams statiskās datu noliktavas faktu tabulu sadalīt mazākos nodalījumos tā, lai ielādējot datus būtu jāizmaina pēc iespējas mazāks skaits nelielo nodalījumu. Šādā veidā ir iespējams samazināt laiku, kas ir nepieciešams datu ielādei statiskās datu noliktavas faktu tabulā un laiku, kas ir nepieciešams, lai pārrēķinātu datu noliktavā eksistējošos indeksus. Tā kā tieši indeksu pārrēķināšana, ja tie ir dzēsti pirms datu ielādes, vai pati datu ielāde, ja indeksi netiek dzēsti, visvairāk ietekmē datu ielādes laiku datu noliktavā, un nepieciešamais laiks ir tieši atkarīgs no ietekmēto datu apjoma, tad faktu tabulas sadalīšana

mazākos nodalījumos ir efektīvs veids, kā samazināt ielādes laiku. Faktu tabulu visbiežāk ir iespējams sadalīt nodalījumos pēc laika, bet tas, kā tieši faktu tabulu ir iespējams sadalīt ir atkarīgs no konkrētās datu noliktavas realizācijas un funkcionalitātes. [10]

Ja statiskās datu noliktavas faktu tabulu nav iespējams sadalīt mazākos nodalījumos, tā, lai datu ielāde izmainītu tikai nelielu daļu no tiem, tad iespējamais risinājums, lai samazinātu datu ielādei nepieciešamo laiku, ir dzēst faktu tabulas indeksus, pirms veikt datu ielādi. Tas vai ir ieteicams dzēst indeksus pirms datu ielādes, ir atkarīgs no ielādējamo datu apjoma un faktu tabulā jau eksistējošo datu apjoma. Ja ielādējamo datu apjoms ir liels, piemēram 10 miljonu ierakstu vai vairāk, tad ir ieteicams dzēst eksistējošos indeksus pirms uzsākt datu ielādi, un tos izveidot no jauna pēc tam, kad datu ielāde ir pabeigta. Pretējā gadījumā, ja ielādējamo datu apjoms ir neliels, ir ieteicams veikt ielādi indeksus no datu noliktavas nedzēšot. Tas ir tāpēc, ka pilnai indeksu pārrēķināšanai ir nepieciešams ilgs laiks, un ieguvumi no indeksu dzēšanas datu ielādes laikā ir lielāki tikai tad, ja tiek ielādēts liels skaits ierakstu. Datu ielādes laiks ir atkarīgs arī no metodes, kāda tiek izmantota datu ielādei. Gadījumos, kad statiskajā datu noliktavā vienlaicīgi ir jāielādē salīdzinoši liels ierakstu skaits, piemēram vairāk par tūkstoti, ir ieteicams izmantot datu bāzē iebūvētos lielapjoma ielādes rīkus. Pretējā gadījumā, kad ielādējamo ierakstu skaits ir neliels, efektīvāk var būt izmantot kādu izstādes videi pieejamu ielādes rīku. [10]

2.5. Datu analīze un attēlošana reāla laika BI rīkos

Reāla laika BI rīkiem ir jāspēj efektīvi un ātri veikt datu analīzi un iegūto rezultātu prezentāciju lietotājam. Reāla laika datu noliktavas rada papildu izaicinājumus datu analīzei un prezentēšanai, jo šādās datu noliktavās dati tiek atjaunoti nepārtraukti, kas var likt datu bāzes pārvaldības rīkiem padarīt datus nepieejamus analīzei un prezentācijai. BI rīki nevar gaidīt līdz datu noliktavas pārvaldības sistēma padara datus pieejamus, jo šāda aizture reāla laika sistēmā var nebūt pieņemama.

Klasiskie BI rīki var prasīt cilvēka līdzdalību, lai veiktu datu analīzi, kas rada papildu gaidīšanas laiku starp brīdi, kad dati ir ielādēti datu noliktavā un tie tiek padarīti pieejami gala lietotājam. Periodu starp brīdi, kad dati ir ielādēti datu noliktavā un tie kļūst pieejami gala lietotājam sauc par analīzes gaidīšanas laiku. Lai samazinātu šo gaidīšanas laiku, reāla laika BI rīkos ir nepieciešams automatizēt datu analīzi, izmantojot dažādus lietotāja definētus parametrus, pēc kuriem tiek sagatavotas attiecīgas atskaites, uz kurām balstoties lietotājs var

pieņemt lēmumu par turpmākajām darbībām. Ir iespējami vairāki tehniski risinājumi, ar kuru palīdzību var samazināt analīzes gaidīšanas laiku, kā piemēram iepriekšējo materializēto skatu izmantošana jaunu skatu pārrēķināšanā un dalītu sistēmu izmantošana paralēlu vaicājumu apstrādei. Izmantojot iepriekšējos materializētos skatus jaunu skatu pārrēķināšanā tiek samazināts datu apjoms, kas ir tiek apstrādāts jaunā skata iegūšanai, šādā veidā samazinot laiku, kas ir nepieciešams materializētā skata pārrēķināšanai. Dalītu sistēmu izmantošana paralēlai vaicājumu izpildei nozīmē to, ka katrs no paralēlajiem vaicājumiem strādās tikai ar apakškopu no visiem datu noliktavas datiem, tādā veidā paātrinot vaicājumu izpildi. Šajā gadījumā ir nepieciešams risinājums vaicājumu atgriezto rezultātu apvienošanai. [17]

Datizrace reāla laika BI rīkā attīstās no statiskas datizraces uz dinamisku straumju izraci. Straumju izrace ir process, kurš spēj apstrādāt nepārtrauktu datu straumi un veikt ātru datu analīzi un apstrādi. Lai realizētu straumju apstrādi reāla laika BI sistēmās, var būt nepieciešams attiecīgi pielāgot reāla laika datu noliktavu, piemēram, realizētu papildus datu glabātuvi, kurā dati tiek glabāti saspiestā veidā, lai ļautu straumes apstrādes algoritmiem veikt ātrāku datu apstrādi. Lai uzlabotu reāla laika datizraces algoritmu darbības laikus var izmantot pakāpeniskas mācīšanās algoritmus. [17]

3. DATU NOLIKTAVAS RISINĀJUMS XIA

Datu noliktava XIA ir datu noliktavas risinājums apdrošināšanas kompānijām, kas izmanto apdrošināšanas biznesa pārvaldības sistēmu TIA kā galveno rīku, lai pārvaldītu ar apdrošināšanas biznesu saistītos ikdienas darījumprocesus. Apdrošināšanas sistēmas TIA kodolā ir realizēti visa apdrošināšanas biznesa pārvaldībai nepieciešamie procesi atbilstoši apdrošināšanas biznesa standartiem, procesi ir pielāgojami katras apdrošināšanas kompānijas specifiskajām prasībām. Datu noliktavā XIA ir realizēts procesu kopums, kas ļauj veikt datu izvilkšanu no apdrošināšanas sistēmas TIA, datu transformāciju un ielādi datu noliktavā XIA, kur tiek veikta tālāka datu analīze izmantojot dažādas dimensijas un metrikas. Datu noliktavā XIA ir iestrādātas noklusētas atskaites, kuras apdrošināšanas kompānija var izmantot uzreiz pēc datu noliktavas uzstādīšanas, kā arī tā ļauj lietotājiem izveidot pašiem savas atskaites izmantojot izpildes pamatrādītājus no vairāk nekā 200 izpildes pamatrādītāju bibliotēkas.

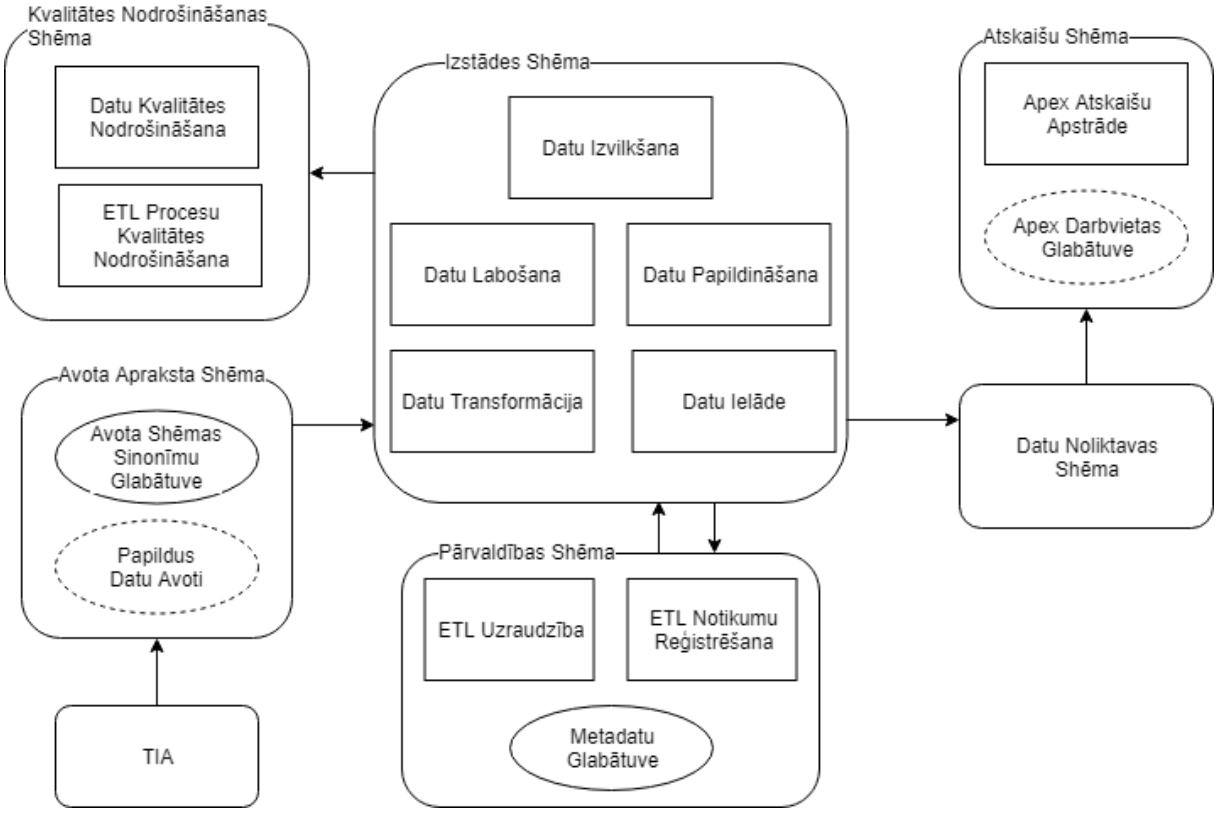
Šajā nodaļā ir detalizēti aprakstīta datu noliktavas XIA arhitektūra un tehnoloģijas, kas tiek izmantotas datu noliktavā XIA. Maģistra darba ietvaros šī datu noliktava ir papildināta ar reāla laika analīzes funkcionalitāti.

3.1. Datu noliktavas XIA arhitektūra

XIA datu noliktava ir Oracle datu bāze, kura satur 6 datu bāzes shēmas, kurai katrai ir savs specifisks mērķis. Šīs shēmas ir:

- Avota apraksta shēma – apraksta savienojumu starp datu noliktavu XIA un avota shēmas objektiem;
- Izstādes shēma – datu izstādes un transformācijas vide. ETL procedūru pirmkods tiek glabāts šajā shēmā;
- Datu noliktavas shēma – galvenā datu noliktavas shēma, kas satur dimensionālajā modelī transformētus un analīzei gatavus datus;
- Pārvaldības shēma – shēma datu noliktavas XIA konfigurācijai un ETL notikumu reģistrēšanai;
- Kvalitātes nodrošināšanas shēma – satur kvalitātes nodrošināšanas procedūras datu pareizības pārbaudei;
- Atskaišu shēma – satur atskaišu SQL vaicājumus, ja tie ir nepieciešami.

Visas šīs shēmas ir savā starpā saistītas, un, lai datu noliktava XIA varētu veiksmīgi funkcionēt, katrai no šīm shēmām ir jāspēj pildīt tai noteiktos uzdevumus. Datu noliktavas XIA arhitektūra ir redzama attēlā 3.1.



3.1. att. **Datu noliktavas XIA arhitektūra**

Avota apraksta shēma glabā sinonīmus, kas norāda uz avota sistēmas TIA datu bāzes objektiem. Šie sinonīmi tiek izveidoti datu noliktavas XIA uzstādīšanas laikā izmantojot datu bāzes skriptu, kas atlasa visus avota sistēmas datu bāzes objektus un izveido tiem sinonīmus avota apraksta shēmā. Šis skripts izveido sinonīmus visām avota sistēmas TIA procedūrām, pakotnēm, funkcijām, skatiem, tabulām un tipiem. Šie sinonīmi vēlāk tiek izmantoti datu noliktavas XIA datu izvilkšanas un transformācijas procesos. Šāda realizācija nozīmē to, ka datu noliktavu XIA ir iespējams novirzīt uz citu avota sistēmu vienkārši pārģenerējot sinonīmus, kas tiek glabāti avota apraksta shēmā. Avota sistēma TIA var atrasties gan tajā pašā datu bāzē, kurā ir realizēta datu noliktava XIA, gan arī atsevišķā datu bāzē. Gadījumos, kad datu noliktava XIA un avota sistēma TIA atrodas atsevišķās datu bāzēs, XIA datu bāzē ir jāizveido datu bāzes saite, kura tiek izmantota, lai piekļūtu avota sistēmai. Avota apraksta shēma tiek izmantota arī, lai izveidotu jebkādus papildu datu bāzes objektus, kas var būt nepieciešami ETL procesos, piemēram, materializētos skatus vai sekvences. Katram datu noliktavas XIA datu avotam būtu jāizveido atsevišķa avota apraksta shēma, ko var izmantot, lai piekļūtu datiem avota sistēmā.

Izstādes shēma ir shēma, kas nodrošina ETL procesu izpildi. Šī shēma satur procedūru kopu, kas veic datu izvilkšanu, labošanu, papildināšanu, transformāciju un ielādi datu noliktavā XIA. XIA ETL procesi ir realizēti izmantojot programmēšanas valodu PL/SQL, kurā ir realizēta arī avota sistēma TIA. Šādā veidā tiek atvieglota abu sistēma integrācija un tiek nodrošināts, tas, ka datu noliktava XIA var izmantot jau eksistējošo infrastruktūru. Datu transformācijas procesā datu noliktava XIA izmanto vairākas funkcijas, kas ir implementētas sistēmā TIA, šādā veidā nodrošinot datu atbilstību abās sistēmās. Lai datu noliktavai pievienotu jaunus datu avotus šī shēma ir jāpaplašina ar papildus funkcijām, kas nodrošina šo jauno datu avotu veiksmīgu apstrādi.

Datu noliktavas shēmā tiek glabāti visi datu noliktavā XIA ielādētie dati. Dati tiek glabāti dimensionālajā modelī, kas sastāv no faktu un ar tiem saistīto dimensiju tabulām. Datu noliktavā XIA noklusēti ir pieejami tādi fakti, kā polišu skaits, maksājumi un ierakstītā prēmija. Šajā shēmā glabājas arī tiltu tabulas, kas ir tabulas, ar kuru palīdzību tiek realizēta daudz pret daudz attiecība starp faktu un dimensiju, lai būtu iespējams glabāt faktus, kuriem ir vairākas konkrētās dimensijas vērtības, un nesabalansētas hierarhijas. Tā kā ar vienu polisi var apdrošināt vairākus objektus, tad polises faktam var būt nepieciešams izmantot tilta tabulu, lai saglabātu šos objektus, bet nesabalansēta hierarhija ir nepieciešama, lai saglabātu apdrošināto objektu hierarhiju, piemēram, komerciāla īpašuma apdrošināšanai. Rīki, kuri piekļūst datu noliktavā glabātajiem datiem izpilda vaicājumus šajā shēmā, lai iegūtu datus, no kuriem izveidot atskaites.

Pārvaldības shēmas mērķis ir ļaut lietotājam uzraudzīt ETL procesu izpildi un ziņot par jebkādām kļūdām, kas var rasties datu noliktavā. Šajā shēmā tiek glabāti dažādi parametri, kas ir nepieciešami ETL procesiem, kā arī datu noliktavas XIA konfigurācijas tabulas, kurās glabājas papildu klasifikatori atskaitēm un datu kartējumi. Konfigurāciju tabulas tiek izmantotas, lai glabātu informāciju par datu noliktavas XIA datu avotiem, datu noliktavas tabulām, dimensiju noklusētajām vērtībām, sekvencēm un ārējām atslēgām, lai tās būtu iespējams dzēst pirms datu ielādes un izvedot no jauna, kad datu ielāde ir pabeigta, un tās uztur izstrādātāju komanda. Datu kartējumi, kas tiek glabāti šajā shēmā, tiek izmantoti, pamērām, lai kartētu avota sistēmas konta ierakstu atribūtus uz izpildes pamatrādītājiem datu noliktavā XIA.

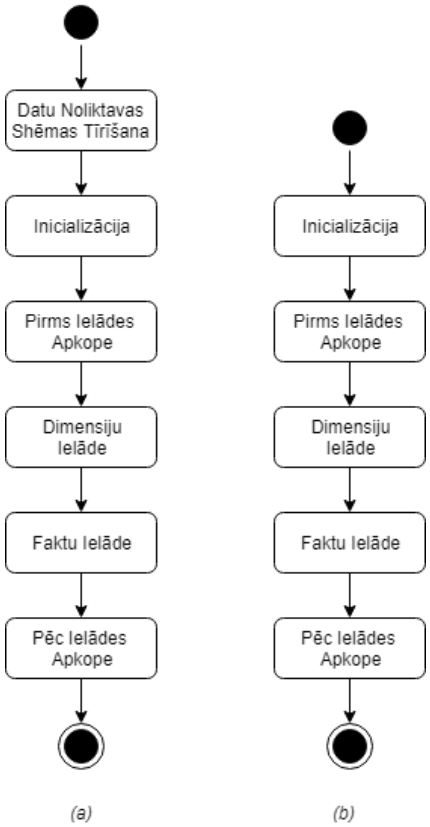
Kvalitātes nodrošināšanas shēma uzrauga ielādējamās datu kopas kvalitāti no diviem aspektiem. Tā pārbauda ielādējamo datu integritāti jeb to vai dati pārkāpj kādus datu struktūru ierobežojumus, piemēram, iztrūkstošas ārējās atslēgas, un vai datos ir kādas pretrunas, piemēram, vēsturiskajos datos pārklājas periodi. Šīs pārbaudes var uzskatīt par tehnisku datu validāciju un tās tiek veiktas ETL procesa ietvaros. Šī shēma pārbauda arī datu atbilstību

darījumprasībām. Atbilstība darījumprasībām tiek pārbaudīta salīdzinot datus dažādās iepriekš definētās darījumu atskaitēs. Šīs darījumprasības dažādiem klientiem var būt atšķirīgas un tās ir atkarīgas no klienta vajadzībām un datu atbilstības darījumprasībām pārbaude var tikt veikta jebkurā laikā pēc datu ielādes datu noliktavā XIA.

Atskaišu shēma ir nepieciešama tikai gadījumos, kad kā atskaišu veidošanas lietotne tiek izmantota Oracle APEX. Šajā gadījumā šī shēma ir nepieciešama, kā apstrādes vide datiem, kas tiek nosūtīti Oracle APEX. Datu apstrādes rezultātu kopa var tikt uzskatīta par izpildes pamatrādītāju bibliotēku, jo šie rezultāti tiek sagatavoti attēlošanai Oracle APEX un tiek nosūtīti pārlūkprogrammai izmantojot Oracle HTTP serveri vai iegultu PL/SQL vārteju.

3.2. Datu noliktavas XIA ETL process

Datu noliktavā XIA var izdalīt divu veidu ETL procesus: sākotnējo ielādi un izmaiņu ielādi. Abu šo procesu UML aktivitāšu diagrammas ir redzamas attēlā 3.2.



3.2. att. **ETL procesu aktivitāšu diagrammas:**

(a) sākotnējās ielādes aktivitāšu diagramma, (b) izmaiņu ielādes aktivitāšu diagramma

Šie procesi ir analogi viens otram. Galvenā atšķirība starp sākotnējās ielādes procesa soļiem un izmaiņu ielādes procesa soļiem ir tā, ka sākot sākotnējās ielādes procesu tiek iztīrīta

datu noliktavas datu bāzes shēma. Šajā solī tiek dzēsti visi dati no datu tabulām datu noliktavas shēmā, tiek izveidoti noklusētie dimensiju ieraksti, tiek atiestatītas datu bāzes sekvences uz to sākotnējām vērtībām un tiek atiestatīti visi ETL parametri uz to sākotnējām vērtībām. Lai kļūdu gadījumā būtu zināms, kurā no soļiem datu noliktavas tīrīšanas process ir apstājies, pēc katra soļa izpildīšanas datu bāzē tiek saglabāts karogs, kurš nozīmē to, ka šis solis ir pabeigts veiksmīgi. Pēc tam, kad datu noliktavas tīrīšanas process ir pabeigts datu bāzē tiek saglabāts karogs, kurš nozīmē to, ka viss tīrīšanas process ir veiksmīgi izpildīts.

Inicializācijas solī tiek iestatīti datu ielādei nepieciešamie globālie mainīgie, ja tie nav iestatīti iepriekš. Globālie mainīgie nosaka to kādā valodā tiks iegūti ierakstu apraksti, no kura un līdz kuram datumam atlasīt ierakstus avota sistēmā, kuru datu avotu izmantot u.tml. Šajā solī arī tiek izveidota jauna ielādes sesija, kura tiek izmantotu, lai reģistrētu ETL notikumus pārvaldības shēmā. Tas ir nepieciešams, lai būtu iespējams uzraudzīt ETL procesu norisi, kā arī, lai būtu iespējams ETL procesu atkļūdot, gadījumos, ja procesa laikā tiek atgriezta kļūda.

Pirms ielādes apkopes solī datu noliktava tiek sagatavota, lai tā spētu apstrādāt jaunos ienākošos datus. Šajā solī tiek atjaunoti datu noliktavā esošie materializētie skati, ievākta statistika no avota shēmas un tiek izslēgtas datu noliktavā esošās ārējās atslēgas. Datu noliktavā esošie materializētie skati tiek atjaunoti, lai tie saturētu svaigāko informāciju, kas tiek glabāta datu noliktavā un var būt nepieciešama datu ielādes laikā. Avota shēmas statistika tiek ievākta, lai rastu priekšstatu par to kāds datu apjoms tiks ielādēts datu noliktavā un atvieglotu datu noliktavas atkļūdošanu, ja ir nepieciešams to veikt. Datu noliktavas ārējās atslēgas tiek atslēgtas, lai paātrinātu datu ielādi datu noliktavas tabulās.

Pēc tam, kad datu noliktava ir sagatavota jaunu datu ielādei, sākas jauno datu ielādes solis. Datu ielāde tiek veikta divos soļos: vispirms tiek atjaunotas visas datu noliktavā esošās dimensiju tabulas un pēc tam tiek atjaunotas visas datu noliktavā esošās faktu tabulas. Ja kādas datu noliktavas tabulas atjaunošanas laikā rodas kļūda, tad viss ETL process tiek pārtraukts un atsaukts. Tas tiek darīts, jo tiek uzskatīts, ka datu noliktavas lietotājiem nebūtu jāpiekļūst daļēji ielādētiem datiem. Katras fakta vai dimensijas tabulas ETL process ir realizēts atsevišķā pakotnē, kur katra pakotne satur visas procedūras, lai varētu veiksmīgi izvilkt datus no avota sistēmas, tos transformēt un ielādēt attiecīgajā tabulā datu noliktavā. Gadījumā, ja datu noliktavā ir nepieciešams veikt tādu datu ielādi, kas nav daļa no avota sistēmas kodola risinājuma, ir nepieciešams pielāgot attiecīgās pakotnes, lai tās spētu šos datu apstrādāt, un, ja datu noliktavā ir izveidota klientam specifiska fakta vai dimensijas tabula, tad ir nepieciešams izveidot jaunu pakotni, kas veiktu nepieciešamās darbības datu ielādei šajā tabulā.

Noslēdzošais solis datu noliktavas XIA ETL procesā ir pēc ielādes datu noliktavas apkope. Šajā solī tiek atjaunoti datu noliktavas materializētie skati, ievākta statistika par datu noliktavas shēmu, ieslēgtas ārējās atslēgas, kā arī tiek veikta dimensiju vērtību korektuma pārbaude un kļūdu reģistrēšana. Materializētie skati, kuri tiek atjaunoti šajā solī, ir skati, kuri padara datu noliktavas datus pieejamus lietotājiem un tiek izmantoti datu noliktavas atskaišu sagatavošanai. Līdzīgi kā avota shēmas statistika, šajā solī ievāktā datu noliktavas shēmas statistika var būt noderīga, lai pārbaudītu vai datu ielāde ir bijusi veiksmīga un var palīdzēt ETL procesa atkļūdošanai, ja tā ir nepieciešama. Ārējās atslēgas pēc datu ielādes tiek ieslēgtas, jo tās ļauj efektīvāk izpildīt lietotāju vaicājumus datu noliktavā, šādā veidā paātrinot datu noliktavas darbību. Gadījumos, kad ETL procesos ir radusies kļūda vai pārbaudot datu noliktavas dimensiju vērtības ir atrasta kļūda, šī kļūda tiek ierakstīta ETL procesa izpildes žurnālā.

3.3. Datu noliktavā XIA izmantotās tehnoloģijas

Datu noliktava XIA ir realizēta Oracle datu bāzē un visi ETL procesi ir implementēti izmantojot PL/SQL. Tiek piedāvātas divas iespējas kā piekļūt datiem datu noliktavā atkarībā no lietotāja prasībām un vajadzībām: *Oracle Application Express* vai *SAP Business Objects*.

Datu noliktavas realizācijai izmantotās tehnoloģijas sakrīt ar tehnoloģijām, kas ir izmantotas, lai realizētu datu noliktavas primāro datu avotu apdrošināšanas biznesa vadības sistēmu TIA. Tas nozīmē to, ka izmantojot šīs tehnoloģijas datu noliktavu XIA ir salīdzinoši viegli integrēt ar avota sistēmu, un atsevišķos gadījumos ir iespējams izmantot avota sistēmas loģiku, lai veiktu datu transformāciju datu noliktavā. Papildu tam programmēšanas valoda PL/SQL ļauj tieši strādāt ar datu bāzes objektiem, šādā veidā atvieglojot efektīvu ETL procesu realizāciju.

Viena no tehnoloģijām, kas tiek piedāvāta klientiem piekļūšanai datu noliktavā ir *Oracle Application Express* jeb *Oracle APEX*. Oracle APEX ir uz Oracle datu bāzēm balstītu tīmekļa lietotņu izstrādes rīks. Ar šī rīka palīdzība ir iespējams izstrādāt ātras un drošas lietotnes, kuru izmantošanai ir nepieciešama tikai tīmekļa pārlūkprogramma. Pie tam Oracle APEX ir iekļauts Oracle datu bāzes licencē, kas nozīmē to, ka datu noliktavas pasūtītājam, tas neprasa papildu ieguldījumus. Šis rīks ir salīdzinoši vienkāršs un tā izmantošanai nav nepieciešami lieli papildus resursi. Tas atbalsta vienkāršus atskaišu veidošanas procesus, kā iepriekš definētu atskaišu izpildi un informācijas paneļus un vienkāršu rezultātu atlasi,

agregāciju un apstrādi. Papildus tam, *Oracle APEX* var cieši integrēt ar avota sistēmu TIA, kas nozīmē, ka lietotāji, piemēram, var izmantot tos pašus lietotāju kontus, ar kuriem tie piekļūst avota sistēmai. Diemžēl, izmantojot šo rīku ir nepieciešama izstrādātāju līdzdalība, lai izveidotu jaunas atskaites. Tas piedāvā tikai statisku datu vizualizāciju un lietotājiem ir pieejami tikai ierobežoti līdzekļi eksistējošo atskaišu pielāgošanai. Papildu tam, šajā rīkā trūkst dažādu biznesa inteliģences funkciju, kā piemēram, rezultātu vērtēšana.

Otra iespēja, kas tiek piedāvāta lietotājiem, lai piekļūtu datu noliktavai ir rīks *SAP Business Objects*. Šajā rīkā ir iekļautas daudzas datu analīzes funkcijas, kas palīdz iegūt lielāku pievienoto vērtību no datu noliktavā glabātajiem datiem. Šis rīks ļauj gala lietotājiem pašiem izveidot jaunas atskaites, bez izstrādātāju līdzdalības. Lietotāji var radīt un izpildīti ekspromtvaicājumus un ekspromtatskaites izmantojot grafisko lietotāja saskarni. Izmantojot šo rīku lietotāji var izmainīt atskaišu izvietojumu un pielāgot datu vizualizāciju. *SAP Business Objects* ir pieejama arī tāda biznesa inteliģencei svarīga funkcionalitāte kā datu izmaiņu izsekošana un iespējamo scenāriju analīze. Diemžēl, šim rīkam ir nepieciešams iegādāties atsevišķu licenci un tam ir lielākas resursu prasības, kas nozīmēt to, ka šī rīka lietošana prasa daudz lielākus finansiālos ieguldījumus nekā *Oracle APEX* lietošana.

4. STATISKĀS DATU NOLIKTAVAS FAKTA APRAKSTS

Šajā nodaļā ir aprakstīta atvasināta polises transakcijas fakta daudzdimensiju modeļa shēma, ETL process šī fakta iegūšanai un uz tā balstītai polišu skaita izpildes pamatrādītājs. Šī darba ietvaros ir izstrādāts risinājums apdrošināšanas polišu skaita izpildes pamatrādītāja atjaunošanai reālā laikā.

4.1. Polišu transakciju skaita izpildes pamatrādītāja struktūra

Šajā darbā ir izstrādāts risinājums polišu transakciju skaita izpildes pamatrādītāja atjaunošanai reālā laikā. Šis izpildes pamatrādītājs tiek izmantots, lai noteiktu to, kā ir mainījies polišu skaits noteiktā laika periodā. Šis izpildes pamatrādītājs tika izvēlēts atjaunošanai reālā laikā tāpēc, ka tas nav atkarīgs no virsgrāmatas grāmatojumiem, kuri tiek veikti tikai reizi dienā, kas nozīmē to, ka to atjaunošanai reālā lakā nav nepieciešamība, un šī izpildes pamatrādītāja iegūšanas ETL process nav triviāls. Polišu skaita izpildes pamatrādītājs ir balstīts uz atvasinātas polises transakcijas fakta. Šis izpildes pamatrādītājs ir pieejams datu noliktavas XIA polišu realizācijas (no angļu *sales*) daudzuma info panelī, kurš ir redzams attēlā 4.1, kā arī ir iespējama šīs informācijas analizēšana pēc darījuma līnijām, pārdošanas kanāliem, u.tml.

Reporting Period 2019-05-01 To 2019-05-31 Previous Period 2019-04-01 To 2019-04-30 Go

Quantity

Indicator	Current	Change %
GWP, €	0	0
GWP New Business, €	0	0
GWP Cancellations, €	0	0
GWP Renewals, €	0	0
GWP Expired, €	0	0
Written Instalments, €	0	0
No of Policies	2	-97.5
No of Policies New Business	4	-95.1
No of Policies Renewals	1	0
No of policies Cancellations	-3	-200
No of policies Expired	-2	0
No of policies In Force	81	0

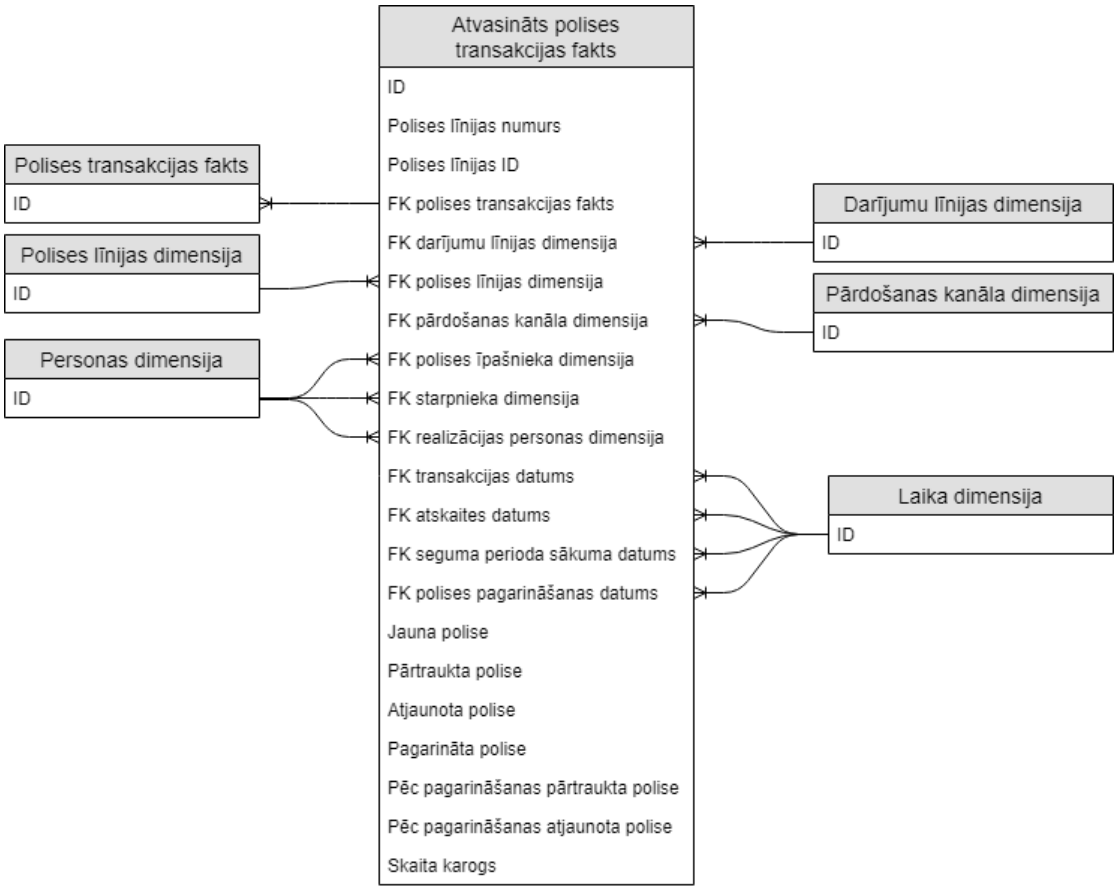
Quality

Indicator	Current	Change %
Renewal Ratio, %	50	0
Renewal Ratio in GWP, %	0	0
Termination Ratio, %	4	0
Termination Ratio in GWP, %	0	0
No of Customers - Start	1	0
No of Customers - End	1	0
No of New Customers	0	-100
No of Retained Customers	1	0
No of Lost Customers	0	0
New Customers Ratio, %	0	0
Retained Customers Ratio, %	100	0
Lost Customers Ratio, %	0	0

4.1. att. **Datu noliktavas XIA polišu realizācijas (no angļu *sales*) info panelis**

Atvasināta polises transakcijas fakta tabula glabā informāciju par to, kāda tipa polises transakcija ir katra datu noliktavā ierakstītā polises transakcija. Šajā faktu tabulā katram iespējamajam polises transakcijas tipam ir izveidots atsevišķs karoga lauks, un, ielādējot datus datu noliktavā, katrai atvasinātajai polises transakcijai tiek aizpildīts atbilstošais polises

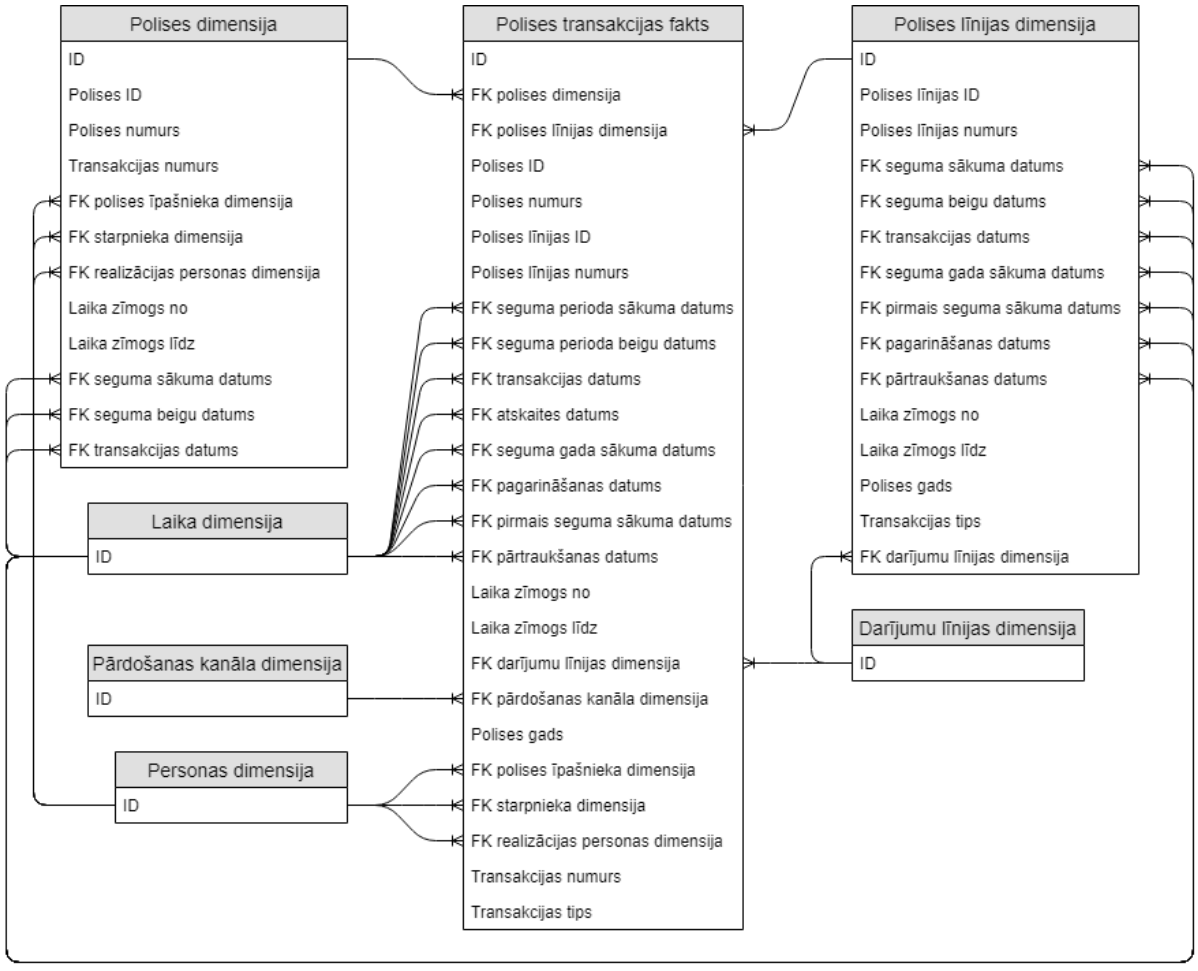
transakcijas tipa karoga lauks ar vērtību 1 vai -1 atkarībā no tā, vai polises transakcijas tips palielina polišu skaitu, vai to samazina. Polises transakciju tipi, kuri palielina polišu skaitu dotajā periodā ir jaunas apdrošināšanas polises izveidošana, pārtrauktas apdrošināšanas polises atjaunošana, pagarināta polise un pēc pagarināšanas pārtrauktas polises atjaunošana. Realizācijas info panelī šie transakcijas tipi tiek apvienoti jaunu polišu skaita un pagarinātu polišu skaita izpildes pamatrādītājos. Polises transakciju tipi, kas samazina polišu skaitu periodā ir polises pārtraukšana un polises pārtraukšana pēc polises pagarināšanas. Realizācijas info panelī šie transakcijas tipi tiek apvienoti vienā izpildes pamatrādītājā – pārtraukto polišu skaits. Papildu tam realizācijas info panelī ir redzama informācija par to kā ir mainījies apdrošināšanas polišu skaits atskaites periodā, cik apdrošināšanas polises atskaites periodā ir beigušās bez pagarināšanas un cik aktīvas polises ir dotā atskaites perioda beigās. Lai varētu veikt padziļinātu polišu transakciju skaita analīzi pēc darījumu līnijām, pārdošanas kanāliem, polises turētāja, starpnieka vai pārdevēja, atvasinātā polises transakcijas fakta tabulā ir ārējās atslēgas uz šīm dimensijām. Atvasinātas polises transakcijas fakta zvaigznes shēma ir redzama attēlā 4.2.



4.2. att. **Atvasināta polises transakcijas fakta zvaigznes shēma.**

Atvasināts polises transakcijas fakts ir balstīts uz polises transakcijas fakta tabulas. Šis fakts nesatur nevienu apkopojamu vērtību, bet tas atspoguļo pilnu polises transakciju vēsturi, kas nozīmē to, ka tas ir uzskatāms par bez fakta faktu. Šis fakts satur informāciju gan par

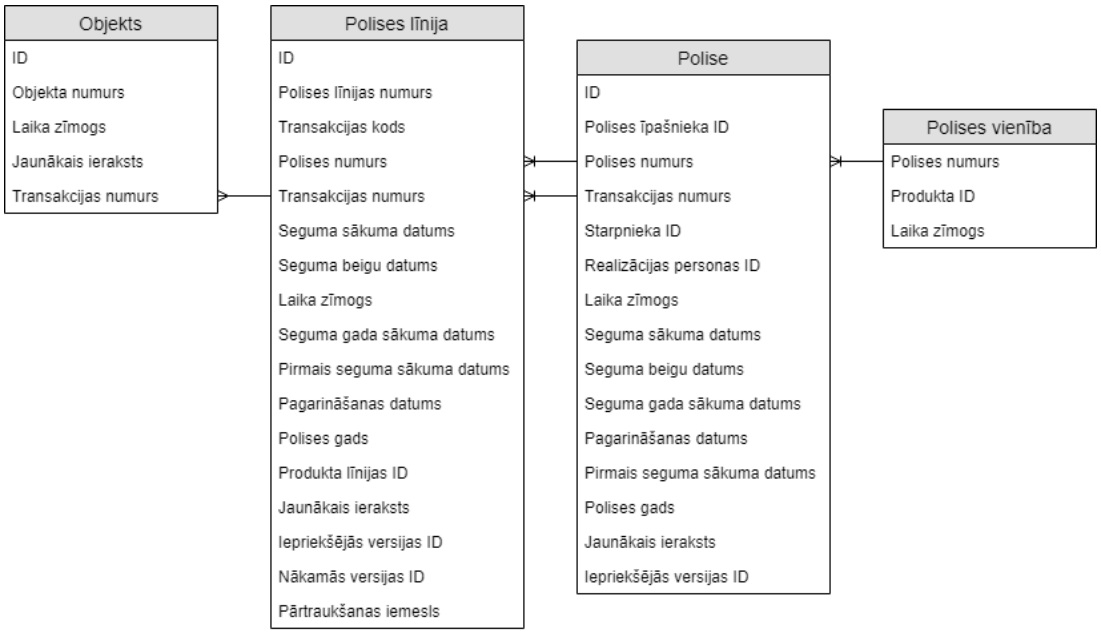
apdrošināšanas polisi, gan par polises līniju. Polises transakciju fakts satur visu polises vēsturi kāda tā ir bijusi jebkurā atskaites periodā. Lai to izdarītu katram ierakstam šajā tabulā ir laika zīmogi, kuri parāda tieši kādā laika periodā dotā polises transakcija ir bijusi aktuāla. Šie laika zīmogi var tikt pārrakstīti, kad datu noliktavā tiek ielādētas jaunas polises transakcijas, kuras pārraksta vēsturiskās polises transakcijas. Šis fakts satur aprakstošu informāciju par polises līnijas statusu pēc polises transakcijas izpildes. Polises transakcijas fakta zvaigznāja shēma ir redzama attēlā 4.3. Šī shēma nesatur polises transakciju aprakstošos laukus, jo tie nav svarīgi šī darba ietvaros izstrādātā risinājuma funkcionalitātes nodrošināšanai.



4.3. att. **Polises transakcijas fakta zvaigznāja shēma**

Polises transakcijas fakts tiek iegūts no polises un polises līnijas dimensijām. Polises dimensija satur aprakstošu informāciju par visām polises transakcijām, kas ir notikušas avota sistēmā, bet polises līnijas dimensija satur šo pašu informāciju par polises līnijām. Šīs dimensijas ir nepieciešams nodalīt, jo viena apdrošināšanas polise avota sistēmā var saturēt vairāk par vienu polises līniju. Šīs dimensijas tāpat, kā polises transakcijas fakts, satur pilnu vēsturisku informāciju par attiecīgi visām polises transakcijām un polises stāvokli dotā laika periodā un polises līnijām un to stāvokli dotā laika periodā, kas nozīmē, ka arī šīm dimensijām ir laika zīmoga lauki, kas nosaka laika periodu, kad dotais dimensijas ieraksts ir

bijis aktuāls. Šīs dimensiju tabulas tiek aizpildītas no datu noliktavas izstādes tabulām, un tajās tiek ierakstītas tikai tādas polises un polišu līniju versijas, kuras ir pabeigtas avota sistēmā. Lai to garantētu, tiek izmantota polises transakciju laika zīmogu izstādes tabula, kura glabā avota sistēmas transakcijas numuru, polises numuru un vēlāko laika zīmogu, kas ir atrodams uz polises, polises līnijas vai apdrošinātā objekta ieraksta. Šajā tabulā tiek izveidots ieraksts par polises transakciju tikai tad, kad visās minētajās entītijās, kurās ieraksti ar doto transakcijas numuru ir atzīmēti kā pabeigti. Šī tabula nodrošina arī to, ka transakcijas laika zīmogi uz polises un polises līnijas dimensiju ierakstiem ir vienādi.



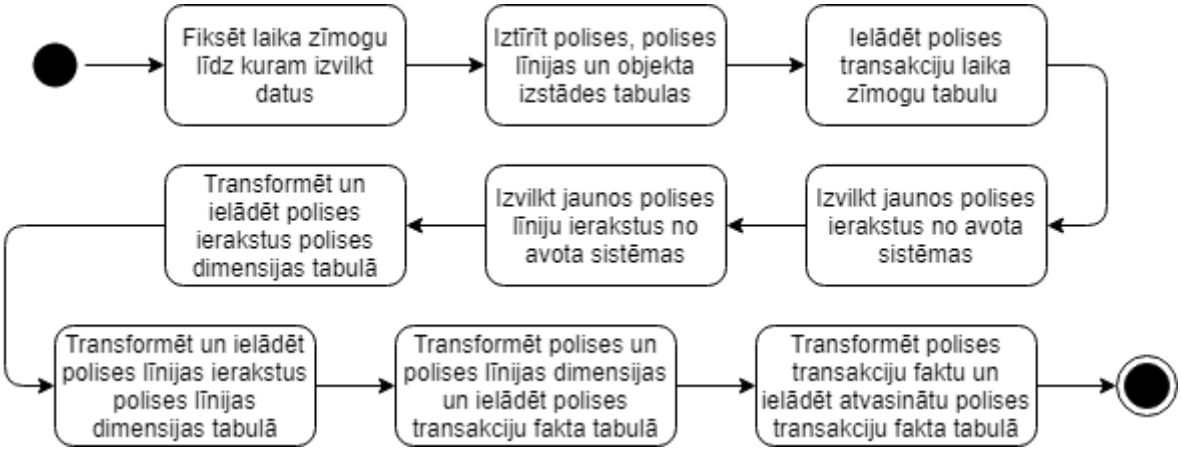
4.4. att. **Izmantojamo avota sistēmas tabulu relāciju modelis**

Lai iegūtu nepieciešamo informāciju polises un polises līniju dimensiju aizpildīšanai ir nepieciešams iegūt datus no četrām avota sistēmas tabulām: objekta, polises līnijas, polises un polises vienības tabulām. Šo tabulu relāciju modelis ir redzams attēlā 4.4. Lai noskaidrotu transakciju numurus, kuru ieraksti ir jāiegūst no avota sistēmas, tiek izmantots objekta, polises līnijas un polises tabulu apvienojums, kurā tiek pārbaudīts vai kāds ieraksts ar doto transakcijas numuru nav saglabāts vēlākai apstrādei. Visām izmaiņām, kas tiek veiktas polisei vai polises līnijai, avota sistēmā tiek izveidots jauns ieraksts. Ieraksts tiek atzīmēts kā saglabāts vēlākai apstrādei uzliekot atbilstošu karoga vērtību laukā "Jaunākais ieraksts", kas nozīmē to, ka, ja nevienā no tabulām ierakstam ar doto transakcijas numuru nav uzlikta šī atzīme, tad visi ieraksti ar doto transakcijas numuru ir jāielādē datu noliktavā. Tā kā polises līnijas dimensija glabā informāciju par produktu, kurā dotā polises līnija ir iekļauta, tad ir nepieciešams iegūt datus arī par polises vienību, kas ir vienīgā tabula, kas glabā informāciju par polises produktu. Polises vienības tabula glabā vienu ierakstu katrai avota sistēmā izveidotajai polisei. Nepieciešamo ierakstu polises vienības tabulā var iegūt pēc polises

numura, kurš tiek glabāts uz polises līnijas. Polises līnijas un polises tabulas glabā visu atlikušo informāciju, kas ir nepieciešama, lai izveidotu ierakstus polises līnijas un polises tabulās.

4.2. Polišu transakciju skaita izpildes pamatrādītāja ETL process

Polišu transakciju skaita izpildes pamatrādītāja ETL procesa ietvaros dati tiek iegūti no avota sistēmas, saglabāti izstādes vidē, transformēti, ielādēti polises un polises līnijas dimensiju tabulās un ielādēti polises transakcijas un atvasinātas polises transakcijas faktu tabulās. Attēlā 4.5. ir redzama aktivitāšu diagramma ETL procesa polišu transakciju skaita izpildes pamatrādītājā izmantotā fakta iegūšanai. Šajā diagrammā ir attēloti tikai soļi, kas ir tieši saistīti ar atvasināta polises transakcijas fakta iegūšanu un soļi citu datu noliktavas faktu iegūšanai šajā diagrammā nav iekļauti.

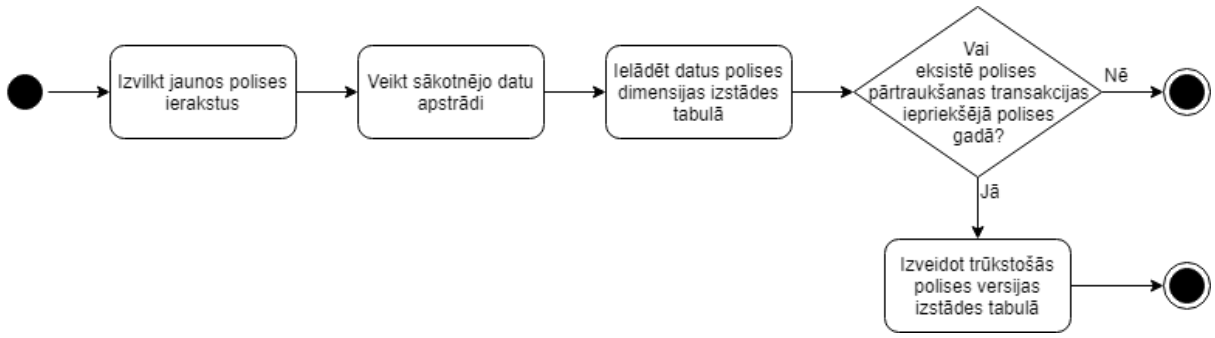


4.5. att. **Polišu transakciju skaita izpildes pamatrādītāja ETL procesa aktivitāšu diagramma**

Uzsākot ETL procesu datu noliktavā XIA tiek fiksēts laika zīmogs, līdz kuram atlasīt transakcijas avota sistēmā. Laika zīmoga vērtība tiek uzstādīta, kā mazākā aktīvās datu bāzes transakcijas sākuma datuma vērtībā ETL procesa uzsākšanas brīdī vai kā sistēmas datums gadījumos, kad datu noliktavā nav nevienas aktīvas datu bāzes transakcijas. Šis laika zīmogs tiek fiksēts ar mērķi novērst datu zudumu datu noliktavā gadījumos, kad tajā ir aktīvas datu bāzes transakcijas brīdī, kad tiek uzsākts ETL process, kā arī, lai datu noliktavā netiktu ielādētas transakcijas, kuras ir izveidotas ETL procesa darbības laikā, kas nozīmē to, ka daļa no šo transakciju veiktajām izmaiņām var nebūt ielādētas datu noliktavā. Papildus tam, katrai ielādētajai dimensijai un faktam datu noliktavas parametros tiek fiksēts lielākais avota ieraksta identifikators, kurš ir ielādēts datu noliktavā attiecīgajam faktam vai dimensija, lai novērstu avota ierakstu atkārtotu ielādēšanu datu noliktavā.

Visi ETL procesi, kas ir realizēti datu noliktavā XIA un izmanto izstādes tabulas, sākas ar izstādes tabulu iztīrīšanas soli. Šajā solī tiek dzēsti visi dati, kas atrodas izstādes tabulās kopš iepriekšējās reizes, kad ETL process ir izpildīts. Šie dati tiek saglabāti izstādes tabulās pēc ETL procesa beigām, lai gadījumos, kad datu noliktavā tiek atklātas kļūdas, būtu iespējams izsekot tam, tieši kādi dati ir ielādēti apstrādei datu noliktavā no avota sistēmas. Turpmākajos soļos šajās tabulās tiks ielādēti svaigi dati, kas ir iegūti no avota sistēmas tekošā ETL procesa izpildes laikā.

Pirmā informācija, kas tiek izvilkta no avota sistēmas ir informācija par transakcijām, kas ir radītas laika periodā starp iepriekšējo reizi, kad ETL process ir izpildīts un laika zīmogu, līdz kuram dati ir iegūstami šajā ETL procesā. Šī soļa ietvaros nepieciešamā informācija tiek iegūta no polises, polises līnijas un objekta tabulām avota sistēmā un saglabāta polises transakciju laika zīmogu tabulā datu noliktavā. No šīm tabulām tiek iegūti transakcijas numura, ieraksta laika zīmoga, polises numura un karoga, kura vērtība ir 0, ja dotais ieraksts ir saglabāts vēlākai apstrādei, vai 1 pretējā gadījumā, lauki. No tabulām iegūtie ieraksti tiek apvienoti un grupēti pēc transakcijas numura. Polises transakciju laika zīmogu tabulā tiek saglabātas tikai tās transakcijas, kuru karoga lauka minimālā vērtība ir lielāka par nulli jeb tās transakcijas, kuru visi ieraksti ir pabeigti. Jaunie ieraksti tiek sapludināti (no angļu *merge*) ar tabulā jau eksistējošajiem datiem, un gadījumos, kad tiek atjaunots tabulā jau eksistējošs ieraksts, tam tiek atjaunots laika zīmogs ar vērtību, kuru process ir atradis šajā sesijā. Polises transakciju laika zīmogu tabulā tiek ierakstīts transakcijas numurs, polises numurs, un lielākais laika zīmogs, kas tika atgriezts dotajai transakcijai no avota tabulām.



4.6. att. **Polises datu izvilkšanas aktivitāšu diagramma**

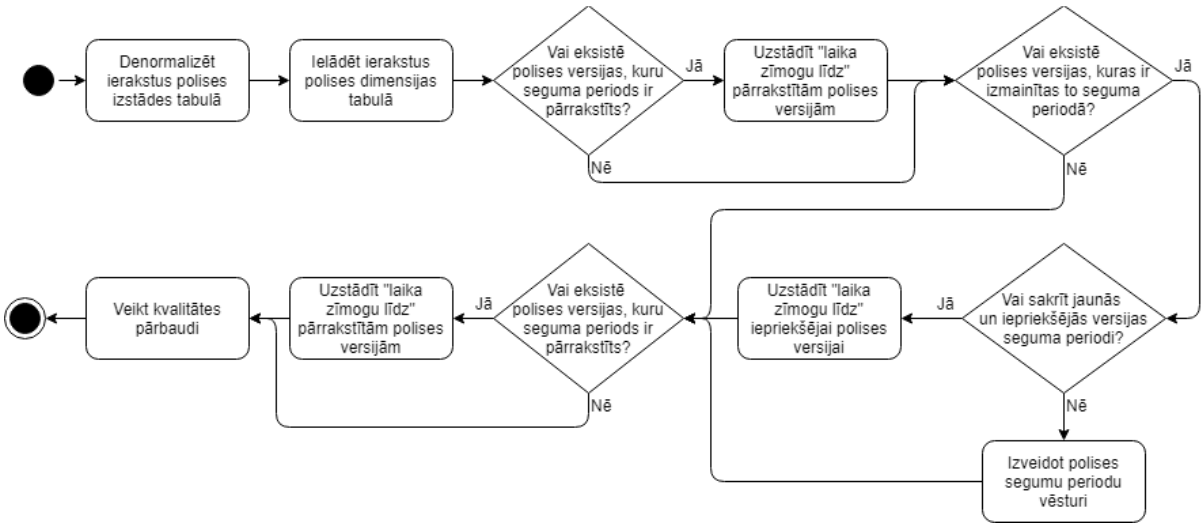
ETL procesa turpinājumā no avota sistēmas tiek izvilkti polises dati. Šī procesa aktivitāšu diagramma ir redzama attēlā 4.6. No avota sistēmas tiek izvilkti visi polises ieraksti, kuru transakcijas numurs ir atrodams polišu transakciju laika zīmoga tabulā un kuriem laika zīmogs avota sistēmā ir starp iepriekšējā ETL procesa sākuma laika zīmogu un tekošā ETL procesa fiksēto laika zīmogu. Ierakstiem, kuri tiek iegūti no avota sistēmas, tiek veikta sākotnējā apstrāde, kas ietver polises pārtraukšanas datuma un atbilstošo realizācijas personu uzstādīšanu izmantojot standarta SQL funkcijas, kā arī polises atjaunošanas datuma

uzstādīšanu no iepriekšējās polises versijas gadījumos, kad ir veikta polises pārtraukšana iepriekšējā polises gadā, izmantojot apakšvaicājumu, kurš tiek izpildīts avota sistēmā. Pēc tam, kad datiem ir veikta sākotnējā apstrāde tie tiek ielādēti polises dimensijas izstādes tabulā. Kad ielāde izstādes tabulā ir pabeigta, tiek veikta pārbaude vai ielādētajiem polises ierakstiem ir nepieciešams izveidot ierakstus par trūkstošajām polises versijām. Tas var būt nepieciešams gadījumos, kad apdrošināšanas polise ir pārtraukta vai izmainīta iepriekšējā polises gadā, un izveidotajai polises pārtraukšanas versijai ir uzstādīts polises gads, kurā tā ir pārtraukta, bet pagarināšanas datums ir tāds pats kā iepriekšējai polises versijai. Šādos gadījumos ir nepieciešams izveidot papildu ierakstu par polises pārtraukšanu arī par pēdējo polises gadu. Šī pārbaude tiek veikta izstādes tabulā ielādētajiem ierakstiem, iepriekšējās polises versijas meklējot avota sistēmā.

Process jauno polišu līniju ierakstu izvilkšanai no avota sistēmas ir analogs jauno polišu datu izvilkšanai no avota sistēmas. Arī šī procesa ietvaros no avota sistēmas polises līnijas tabulas tiek atlasīti tie ieraksti, kuru transakcijas numurs ir atrodams polises transakciju laika zīmoga tabulā un kuru laika zīmogs ir starp iepriekšējā ETL procesa sākuma laika zīmogu un tekošā ETL procesa fiksēto laika zīmogu. Papildus šiem nosacījumiem tiek pārbaudīts vai polise, kurai pieder dotā polises līnija, ir ierakstīta polises izstādes tabulā, jo avota sistēmā jaunai polises līnijas versijai ir jābūt atbilstošam ierakstam polises tabulā. Atrastajiem polises līnijas ierakstiem ar vienkāršām SQL funkcijām tiek apstrādāts pārtraukšanas datums, transakcijas kods, kurš norāda kāda veida izmaiņa ir veikta polises līnijai, un tiek uzstādīts polises karogs, kurš norāda vai polises līnija ir daļa no apdrošināšanas polises vai apdrošināšanas polises piedāvājuma. Analogi polises ierakstu apstrādes procesam ar apakšvaicājumu palīdzību gadījumos, kad polises līnija ir izmainīta iepriekšējā polises gadā, no iepriekšējās versijas tiek uzstādīts polises pagarināšanas datums, polises gada sākuma datums un polises gads. Arī polises līniju izstādes tabulā var būt nepieciešams izveidot ierakstus trūkstošajām polises līnijas versijām gadījumos, kad polises līnija ir pārtraukta iepriekšējos polises gados.

Polišu dimensijas transformēšanas un ielādes process sākas ar izstādes tabulā ielādētu ierakstu denormalizāciju. Šī procesa ietvaros ieraksti, polises izstādes tabulas ierakstu atsauču lauki un datumi tiek savienoti ar atbilstošajām dimensijām datu noliktavā, un atsauču identifikatori tiek aizvietoti ar dimensiju koda vērtībām un to aprakstiem, bet datuma lauki tiek aizvietoti ar laika dimensijas identifikatora vērtībām. Gadījumos, kad dimensiju ieraksti ar dotajiem identifikatoriem netiek atrasti datu noliktavā, uz polises dimensiju ierakstiem atbilstošajās kolonās tiek uzstādītas noklusētās vērtības, kas norāda uz dimensiju ierakstiem ar nedefinētu vērtību. Pēc tam, kad izstādes ielādētie ieraksti ir denormalizēti, tie tiek ielādēti

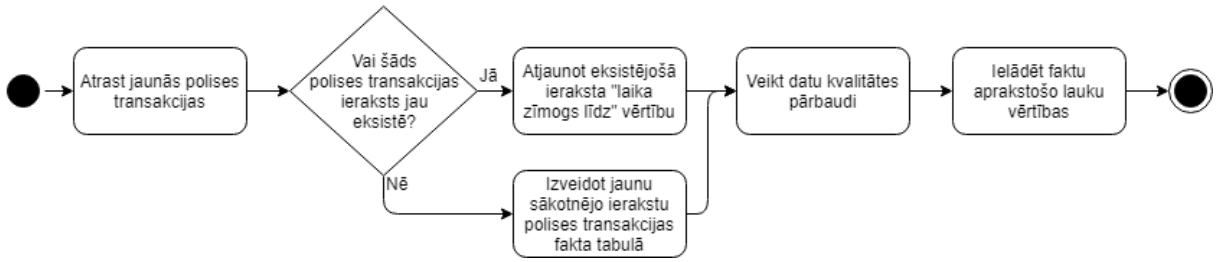
polises dimensijas tabulā. Kad polises ieraksti ir ielādēti polises dimensijas tabulā, tiek pārbaudīts vai dimensijas tabulā eksistē ieraksti, kuru seguma periods ir pilnībā pārrakstīts. Ja šādi ieraksti datu noliktavā eksistē, tad šiem ierakstiem tiek uzstādīta "laika zīmoga līdz" vērtība no jaunā ieraksta "laika zīmoga no" vērtības, šādā veidā norādot to, ka jaunais ieraksts pilnībā aizstāj iepriekšējo datu noliktavas ierakstu, bet saglabājot pilnu polises vēsturi. Pēc pārrakstīto versiju apstrādes tiek pārbaudīts vai polises dimensijā ir ielādēti ieraksti, kuru sākuma periods ir starp iepriekšējās versijas seguma sākuma un beigu datumu. Ja šāds ieraksts tiek atrasts, tad tam tiek uzstādīta "laika zīmoga līdz" vērtība, pretējā gadījumā tiek izveidots polises ieraksts par periodu no iepriekšējās polises versijas sākuma datuma līdz jaunās polises versijas sākuma datumam ar iepriekšējās polises versijas vērtībām. Šis ieraksts ir nepieciešams, lai datu noliktavā glabātos pilna polises vēsture. Pēc šī ieraksta izveidošanas tiek vēlreiz pārbaudīts vai eksistē polises versijas, kuras ir pārrakstītas un šīm versijām laukā "laika zīmogs līdz" tiek atzīmētas versijas aktualitātes perioda beigas. Kā pēdējā darbība pēc ielādes procesa ir polises dimensijas ierakstu kvalitātes pārbaude. Kvalitātes pārbaudes ietvaros tiek pārbaudīts, vai vienai polisei nav vairāki vienlaicīgi aktuāli ieraksti par vienu un to pašu seguma periodu. Polišu dimensijas transformēšanas un ielādes procesa aktivitāšu diagramma ir redzama attēlā 4.7.



4.7. att. **Polises dimensijas transformēšanas un ielādes procesa aktivitāšu diagramma**

Arī transformācijas un ielādes process polises līnijas dimensijai ir analogs polises dimensijas transformācijas un ielādes procesam. Polises līnijas dimensijas transformācijas process sākas ar izstādes tabulas ierakstu denormalizāciju, kas notiek tāpat kā polises dimensijas izstādes tabulas ierakstu denormalizācijas process. Denormalizētie ieraksti tiek ielādēti polises līnijas dimensijas tabulā, kur tie tiek apstrādāti tāpat, kā polises dimensijas tabulas ieraksti. Tas nozīmē to, ka pārrakstītajiem dimensijas ierakstiem tiek uzstādīta "laika zīmogs līdz" vērtība, lai noslēgtu periodu, kad tie ir aktuāli, tiek izveidoti vēsturiskie ieraksti,

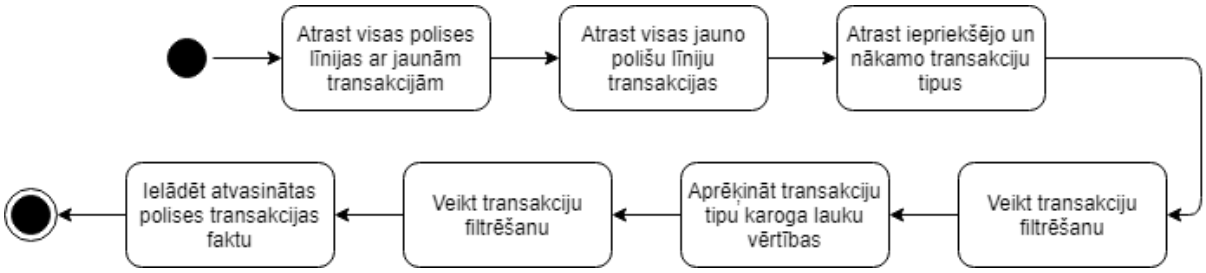
kur tas ir nepieciešams, un vēlreiz tiek apstrādāti pārrakstītie ieraksti. Atšķirībā no polises dimensijas transformācijas un ielādes procesa, polises līnijas transformācijas un ielādes procesā ir divi papildus soļi transakciju tipu koriģēšanai polises līnijas dimensijā. Gadījumos, kad polises līnija ir atjaunota pēc pārtraukšanas avota sistēmā ir iespējams koriģēt transakcijas tipu, kas tiek uzstādīts uz polises līnijas. Šī iemesla dēļ, lai datu noliktavā dati būtu korekti, uz polises līnijas dimensijas ierakstiem, kuru iepriekšējai versijai ir norādīts pārtraukšanas iemesls, bet dotajai versijā tas nav norādīts, tiek uzstādīts polises atjaunošanas transakcijas tips. Otrs polises līnijas transakcijas tipa koriģēšanas solis ir nepieciešams, lai uz polisēm, kuras ir pārtrauktas no to pirmā sākuma datuma un vēlāk atjaunotas, uzstādītu jaunas polises transakcijas tipu. Arī polises līnijas dimensijas tabulai tiek veikta kvalitātes pārbaude, kuras ietvaros tiek pārbaudīts, vai vienai polises līnijai nav vairāki vienlaicīgi aktuāli ieraksti par vienu un to pašu seguma periodu.



4.8. att. **Polises transakciju transformācijas un ielādes process**

Pēc polises un polises līnijas dimensiju ielādes var notikt polises transakcijas fakta ielāde. Polises transakcijas fakts tiek ielādēts balstoties uz šīm dimensijām un tā transformācijas un ielādes process ir redzams attēlā 4.8. Polises transakcijas tiek meklētas visām polisēm, kuras ir ielādētas datu noliktavā tekošajā ETL procesā. Par polises transakciju tiek uzskatīts polises un polises līnijas ierakstu savienojums, kur polises līnija pieder dotajai polisei un polises līnijas seguma periods ir polises seguma perioda ietvaros vai polises seguma periods ir polises līnijas seguma perioda ietvaros, un abi dimensiju ieraksti ir aktuāli vienā laikā. Šajā brīdī polises transakcijas ieraksts tiek identificēts ar polises dimensijas ieraksta identifikatoru un polises līnijas dimensijas ieraksta identifikatoru. Ja polises transakcijas fakta tabulā jau eksistē ieraksts ar šādu identifikatoru kombināciju, tad šim ierakstam tiek uzstādīta jauna “laika zīmogs līdz” vērtība, kas tiek ņemta kā mazākā šī lauka vērtība uz dotajiem polises un polises līnijas dimensiju ierakstiem. Pretējā gadījumā, polises transakcijas fakta tabulā tiek izveidots jauns ieraksts, kurš satur tikai informāciju par polises un polises līnijas identifikatoriem, kā arī informāciju par polises līnijas seguma periodu un laiku, kad transakcija notikusi. Laika informācija tiek ņemta no polises un polises līnijas dimensijām kā lielāka attiecīgā lauka vērtība perioda sākuma laukiem un mazākā attiecīgā vērtība perioda beigu laukiem.

Kad sākotnējie polises transakciju fakta ieraksti ir ielādēti tabulā, ETL process var veikt datu kvalitātes pārbaudi. Arī polises transakciju fakta datu kvalitātes pārbaude ietver pārbaudi vai vienai polises numura un polises līnijas numura kombinācijai neeksistē vairāk par vienu vienlaicīgi aktuālu ierakstu, kuram pārklājas polises transakcijas seguma periods. Ja šādi ieraksti neeksistē, tad polises transakcijām tiek ielādēta to aprakstošā informācija, pretējā gadījumā ETL process tiek atzīmēts kā kļūdains un izveidotie ieraksti netiek iesūtīti datu noliktavā. Faktu aprakstošā informācija tiek ielādēta atsevišķi no sākotnējo datu ielādes labākai procesa ātrdarbības uzlabošanai, jo sākotnējie ieraksti tiek meklēti atsevišķi polises līnijām, kuru seguma periods ir polises seguma perioda ietvaros un polisēm, kuru seguma periods ir polises līnijas seguma perioda ietvaros. Šie rezultāti tiek apvienoti un aprakstošās informācijas neiekļaušana šajā apvienojumā nozīmē to, ka tajā ir mazāk lauku, kas savukārt uzlabo vaicājuma izpildes laiku. Aprakstošo lauku iegūšanai polises transakcijas ieraksti, kuri ir izveidoti pirmajā procesa solī tiek savienoti polises un polises līnijas dimensijām, izmantojot to identifikatoru, un katra aprakstošā lauka vērtība pēc nepieciešamības tiek ņemta no polises vai polises līnijas dimensijas.



4.9. att. **Atvasināta polises transakcijas fakta transformācijas un ielādes aktivitāšu diagramma**

Pēdējais solis ETL procesā, lai iegūtu polišu skaita izpildes pamatrādītāju, ir datu atlase, transformācija un ielāde atvasinātas polises transakcijas fakta tabulā. Šī soļa aktivitāšu diagramma ir redzama attēlā 4.9. Dati ielādei atvasinātas polises transakcijas fakta tabulā tiek atlasīti no polises transakcijas fakta tabulas. Tā kā atvasinātas polises transakcijas fakts ir pamatā balstīts uz polises līnijas informāciju, tad pirmais solis šajā procesā ir atrast visas polises līnijas, kurām ir izveidota kaut viena jauna transakcija pašreizējā ETL procesā. Tas tiek darīts izmantojot pēdējās ielādētās transakcijas datu noliktavas parametru, meklējot visus polises līnijas numurus, kuriem ir transakcijas ar identifikatoru, kurš ir lielāks par šī parametra vērtību. Procesa turpinājumā šīm polises līnijām tiek atrastas visas transakcijas, kas ir ierakstītas polises transakcijas fakta tabulā. Procesa ietvaros tiek atrastas visas polises līnijai izmantotās transakcijas, jo nākamajā solī, lai atrastu iepriekšējo un nākamo transakciju tipus, tiek izmantotas *Oracle* datu bāzē iebūvētās analītiskās funkcijas, kuras strādā ar atlasīto datu kopu. Iepriekšējie un nākamie polises līnijas transakcijas tipi tiek meklēti katram polises gadam atsevišķi.

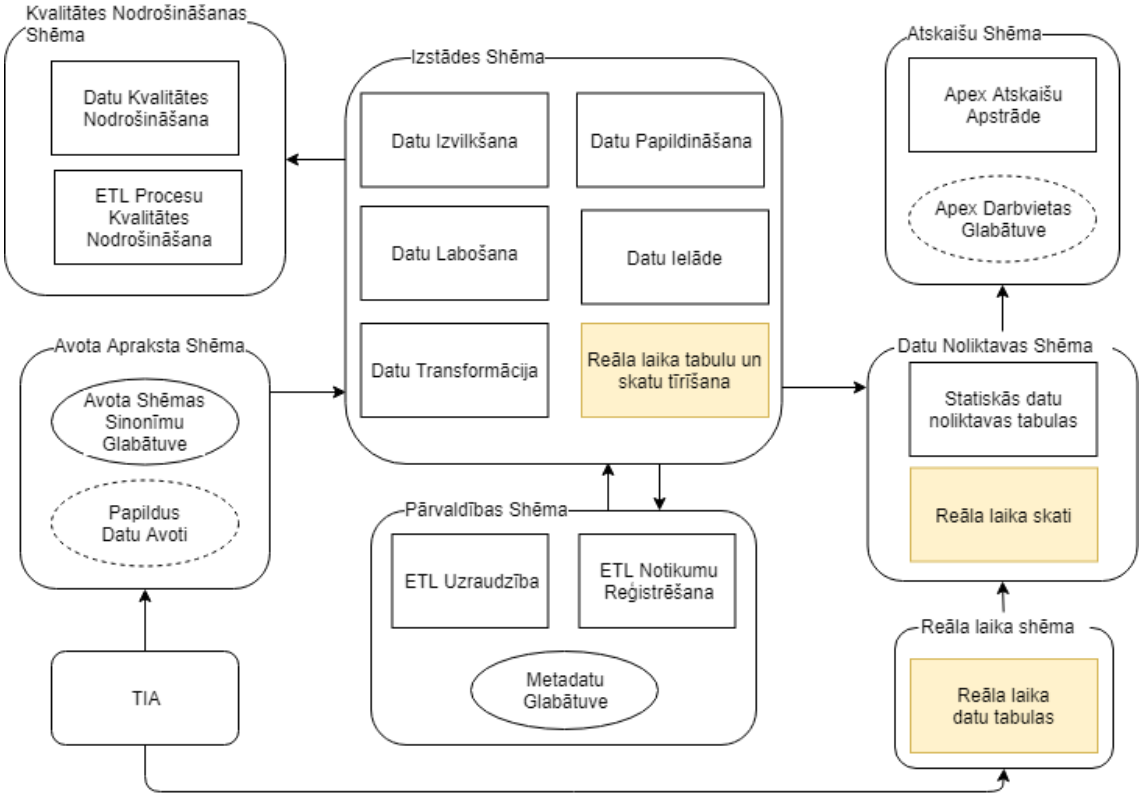
Pēc tam, kad visām atlasītajām polises transakcijām ir atrasti iepriekšējo un sekojošo polišu transakciju tipi, tiek veikta polises transakciju filtrēšana. Šajā solī tiek atsijātas transakcijas, kurām ir tāds pats transakcijas tips kā iepriekšējai transakcijai un transakcijas par periodiem, kad polise ir pārtraukta. Transakcijas, kurās nav mainījies transakcijas tips, var atsijāt, jo tās ir transakcijas, kuras ir izveidotas vēsturisko periodu nosegšanai vai arī vienkāršas polises izmaiņu transakcijas, kuras neizmaina polises statusu, līdz ar to tās neizmaina polišu skaitu sistēmā un nav nepieciešamas šajā fakta tabulā. Turpinājumā atlasītajiem polises transakcijas ierakstiem tiek uzstādītas atvasinātas polises transakcijas fakta karoga lauka vērtībās. Šīs vērtības tiek iegūtas ar vienkāršām SQL funkcijām salīdzinot dotās transakcijas stāvokli ar tās iepriekšējo vai sekojošo stāvokli, un uzstādot atbilstošā atvasinātas polises transakcijas fakta karoga lauka vērtību. Karoga lauks parāda, kāda tipa transakcija dotais fakta ieraksts ir un vai šī tipa transakcija palielina vai samazina kopējo polišu skaitu sistēmā. Kad ierakstu karoga lauku vērtības ir uzstādītas, tiek atlasītas tikai tās transakcijas, kuras vēl nav ielādētas datu noliktavā, izmantojot lielāko transakcijas identifikatoru, kas ir ielādēts datu noliktavā, un tās tiek ielādētas atvasināta polises transakcijas fakta tabulā.

5. REĀLLAIKA DATU NOLIKTAVAS IZSTRĀDE

Šajā nodaļā ir aprakstīta izstrādātā reāla laika datu noliktavas risinājuma arhitektūra un tehnoloģijas, kas šajā risinājumā ir izmantotas. Nodaļas turpinājumā ir aprakstīta datu replikācijas risinājuma izstrāde, kā arī risinājuma datu transformācijas un attēlošanai reālā laikā izstrāde. Nodaļas noslēgumā ir veikta izstrādātā risinājuma analīze.

5.1. Izstrādātā risinājuma arhitektūra un izmantotās tehnoloģijas

Lai realizētu polišu skaita izpildes pamatrādītāja atjaunošanu reālā laikā, datu noliktavā ir izveidota jauna reāla laika shēma, kurā tiek replicēti dati, kas ir izveidoti avota sistēmā kopš pēdējā ETL procesa izpildes, kā arī pielāgota jau eksistējošo shēmu funkcionalitāte, lai integrētu šo shēmu datu noliktavā. Datu noliktavas arhitektūras shēma pēc risinājuma realizācijas ir redzama attēlā 5.1, kurā jaunie datu noliktavas komponenti ir izcelti dzeltenā krāsā.



5.1. att. **Reāla laika datu noliktavas arhitektūra**

Datu noliktavā ir izveidota jauna datu bāzes shēma, kurā glabājas ieraksti no avota sistēmas, kas ir izveidoti pēc pēdējā ETL procesa izpildes datu noliktavā. Lai izveidotu

ierakstus reāla laika shēmā, ir izmantota datu vākšanas caur transakciju atkārtošanas žurnāla datnēm metode. Ar šīs metodes palīdzību izmaiņas avota sistēmā tiek izvilktas no avota datu bāzes transakciju atkārtošanas žurnāla datnēm un nosūtītas uz datu noliktavas serveri, kur tās tiek replicētas un saglabātas reāla laika shēmas tabulās. Šī procesa ietvaros no avota sistēmas tiek izvilktas izmaiņas tikai tajos datu bāzes laukos, kuri ir nepieciešami datu noliktavas ETL procesos.

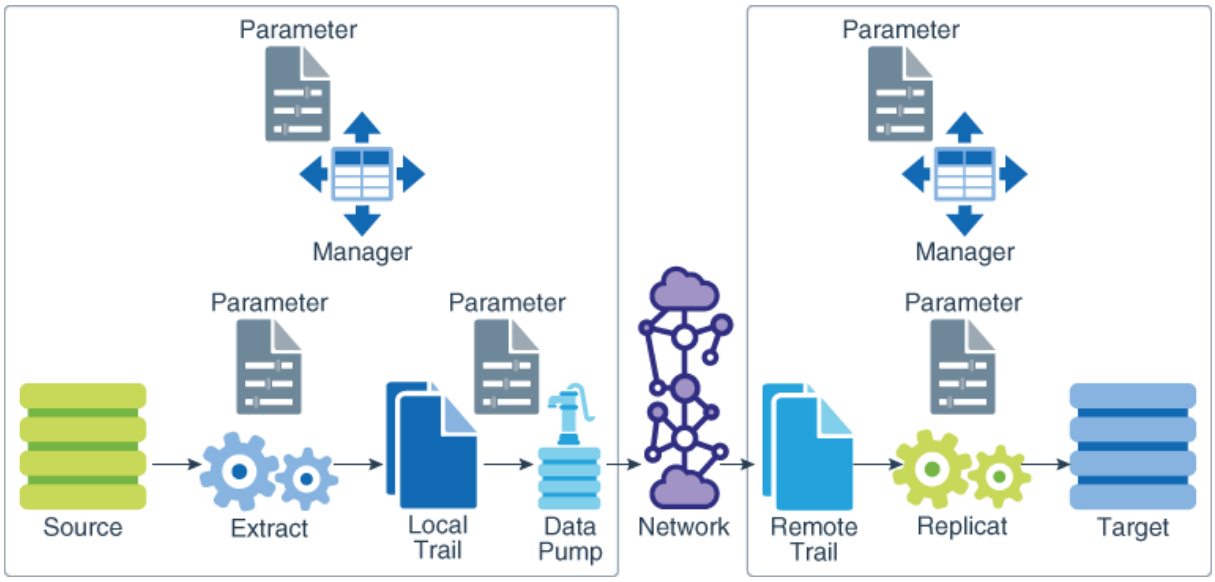
Lai integrētu reāla laika datus datu noliktavā, datu noliktavas shēmā ir izveidoti jauni reāla laika skati. Šie skati izpilda visu nepieciešamo loģiku, lai replicētu atbilstošo faktu un dimensiju transformācijas un ielādes procesus. Datu noliktavas shēma ir pielāgota, lai izmantotu attēlā 2.3. redzamajai arhitektūrai analogu procesu, kas nozīmē to, ka shēmā ir realizēti gan parasti datu noliktavas skati, gan materializēti datu noliktavas skati, lai efektīvi integrētu reāla laika datus ar eksistējošajiem statiskajiem datu noliktavas datiem. Datu noliktavā izveidotie reāla laika materializētie skati tiek atjaunoti tikai brīdī, kad lietotājs no datu noliktavas lietotnes piekļūst šiem skatiem. Šādā veidā ir samazināts laiks, kas ir nepieciešams, lai reāla laika datus integrētu ar eksistējošiem statiskajiem datu noliktavas datiem, gan arī laiks, kas ir nepieciešams, lai izpildītu vaicājumus pret reāla laika datiem.

Lai piekļūtu datu noliktavai ir izmantota *Oracle APEX* lietotne, kurā pēc noklusējuma ir pieejama funkcionalitāte izpildes pamatrādītāju aplūkošanai un analīzei. Lai attēlotu reāla laika datus un saglabātu lietotājam iespēju aplūkot tikai statiskos datu noliktavas datus, šajā lietotnē ir izveidota jauna ekrānforma, kurā lietotājam ir pieejams attēlā 4.1. redzamais realizācijas info panelis ar reāla laika datiem. Lai ļautu lietotājiem veikt tuvāku reāla laika polišu skaita izpildes pamatrādītāja analīzi, atskaišu shēmā tika izveidoti jauni izpildes pamatrādītāji, kuri iegūst datus no reāla laika skatiem. Papildu tam, atskaišu shēmas funkcijas, kuras tiek izmantotas no *Oracle APEX* lietotnes, lai iegūtu lietotnē attēlojamos datus, ir pielāgotas, lai gadījumos, kad reāla laika datu tabulās ir pieejami jaunāki dati nekā reāla laika materializētajos skatos, reāla laika materializētie skati tiktu atjaunoti pirms datu iegūšanas. Šādā veidā ir nodrošināts, lai lietotājam vienmēr tiktu atgriezti svaigākie dati, kas ir pieejami datu noliktavā.

Veiksmīgai reāla laika datu integrēšanai datu noliktavas statiskajos datos ir pielāgots arī datu noliktavas statiskais ETL process. Tajā ir veiktas izmaiņas, lai veiksmīgi izpildīta ETL procesa beigās no reāla laika tabulām tiktu dzēsti tie ieraksti, kas ir ielādēti statiskajā datu noliktavā. Šie ieraksti tiek dzēsti izmantojot ETL procesa sākumā fiksēto laika zīmogu, līdz kuram tiek atlasīti dati ielādēšanai datu noliktavā. No reāla laika datu tabulām tiek dzēsti visi ieraksti, kuru laika zīmogi ir pirms šī fiksētā laika zīmoga. Pēc ierakstu dzēšanas no reāla laika datu tabulām tiek atjaunoti arī reāla laika materializēties skati, lai nodrošinātu to, ka

šajos skatos netiek glabāti veci dati. Šādā veidā tiek nodrošināts, lai sistēmā viens un tas pats ieraksts netiktu vienlaicīgi apstrādāts kā statisks datu noliktavas ieraksts un reāla laika ieraksts.

Lai realizētu datu replikāciju reālā laikā ir izmantots *Oracle GoldenGate* rīks. Šis rīks ietver funkcionalitāti, kas ļauj veikt reāla laika datu integrāciju, transakcionālu datu izmaiņu notveršanu, datu replikāciju un transformāciju. Ar šī rīka palīdzību ir iespējams replicēt datus gan starp *Oracle* datu bāzēm, gan arī citām atbalstītām datu bāzēm vai datņu tipiem. *Oracle GoldenGate* rīks replicē tikai datu bāzē iesūtītas transakcijas, šādā veidā uzlabojot datu kvalitāti un rīka veiktspēju. Ir pieejamas divas *Oracle GoldenGate* arhitektūras – klasiskā arhitektūra, kura ietver visu nepieciešamo funkcionalitāti datu replicēšanai, un mikropakalpju (no angļu *microservices*) arhitektūra, kura papildu klasiskās arhitektūras funkcionalitātei piedāvā arī attālinātas uz REST servisiem balstītas lietotnes risinājuma realizācijai un uzturēšanai. Šajā darbā ir izmantota klasiskā arhitektūra. [23]



5.2. att. **Tipiska *Oracle GoldenGate* procesa realizācijas diagramma [25]**

Attēlā 5.2. ir redzama tipiska *Oracle GoldenGate* procesa realizācijas diagramma. Lai realizētu datu replikāciju ar *Oracle GoldenGate,* ir nepieciešams realizēt izvilkšanas procesu, kas iegūst datus no avota datu bāzes un saglabā tos lokālā pierakstu datnē vai pierakstu datnē uz mērķa servera. Ir iespējamas divas datu izvilkšanas metodes - klasiskā izvilkšanas metode, kas replicē datus no datu bāzes transakciju atkārtošanas žurnāla datnēm, un integrētā izvilkšanas metode, kas mijiedarbojoties ar datu bāzes žurnālizraces (no angļu *logmining*) serveri, iegūst datus no datu bāzes loģiskajiem izmaiņu ierakstiem. Integrētā izvilkšanas metode nodrošina efektīvāku datu izvilkšanu, filtrēšanu un atbalsta vairāk datu bāzes lauku datu tipus, bet tā pilnvērtīgai realizēšanai ir nepieciešama *Oracle* datu bāzes versija 11.2.0.4 vai jaunāka.

Izvilkšanas procesu ir ieteicams konfigurēt tā, lai tas pierakstu datni veido lokāli, un izveidot atsevišķu izvilkšanas procesu, ko sauc par datu sūkni, kas datus no lokālās pierakstu datnes pārraksta uz pieraksta datni mērķa serverī, šādā veidā nodrošinoties pret datu zudumu tīkla kļūmju dēļ. Datu sūknis tīkla kļūmju gadījumā pārtrauc datu pārnesi un saglabā vietu, pie kuras apstrādes tas ir apstājies pierakstu datnē. Darbu atsākot datu sūknis turpinās darbu no šīs vietas pierakstu datnē, tādā veidā nodrošinot, ka netiek pazaudēta informācija.

Datu replikācijas nodrošināšanai uz mērķa servera, uz šī servera ir jārealizē *Oracle GoldenGate* replikācijas process. Replikācijas process lasa datus no pierakstu datnes, replicē nolasītās DML komandas un izpilda šīs komandas datu bāzē mērķa serverī. Replikācijas procesu ir iespējams realizēt izmantojot vienu no divām metodēm – neintegrēto replikācijas metodi, kura replicē standarta SQL komandas, un integrēto replikācijas metodi, kura replicē loģiskos izmaiņu ierakstus un izmanto tos izmaiņu replikācijai. Integrētā izmaiņu replikācijas metode atbalsta izmaiņu asinhronu replikāciju, kas nozīmē to, ka izmaiņas nav obligāti jāreproducē tādā secībā, kā tās ir radītas avota sistēmā, bet neintegrētā izmaiņu replikācija ļauj izmantot paralēlus replikācijas procesus veiktspējas uzlabošanai. Integrētās datu replikācijas metodes izmantošanai ir nepieciešama *Oracle* datu bāzes versija 11.2.0.4 vai jaunāka. [24, 25]

Oracle GoldenGate procesu pārvaldībai, uz visiem serveriem, kuri ir iesaistīti replikācijas procesā, ir jābūt konfigurētam *Oracle GoldenGate* pārvaldības procesam, kas nozīmē arī to, ka uz visiem iesaistītajiem serveriem ir jābūt instalētam *Oracle GoldenGate* rīkam. Pārvaldības process veic citu *Oracle GoldenGate* procesu uzsākšanu, uztur procesu portus, iztīra vecās pierakstu datnes, kā arī izveido notikumu un kļūdu atskaites. Visu šo procesu konfigurēšana notiek izmantojot parametru datnes, kurās tiek norādīti visi procesu veiksmīgai darbībai nepieciešamie parametri. [23]

Risinājum izstrādē izmantotā avota sistēma ir izstrādāta izmantojot *Oracle Database 11g* datu bāzes versiju 11.2.0.3.0, bet datu noliktava ir izstrādāta izmantojot *Oracle Database 12c* datu bāzes versiju 12.2.0.1.0. Datu izvilkšanai no avota sistēmas un replicēšanai datu noliktavā ir izmantota *Oracle GoldenGate* versija 18.1.0.0.0, kas atbalsta darbu gan ar *Oracle Database 11g*, gan ar *Oracle Database 12c*. Risinājums ir izstrādāts izmantojot klasisko datu izvilkšanas metodi, jo avota sistēmas datu bāzes versija integrēto datu izvilkšanas metodi pilnībā neatbalsta. Datu replikācija ir izstrādāta izmantojot neintegrēto datu replikācijas metodi, jo avota sistēmas datu bāzes versija neatbalsta visus atbilstošai datu izvilkšanai nepieciešamos parametrus, piemēram, kompaktu atjaunošanas ierakstu formātu. Risinājums ir izstrādāts uz datora ar *Intel i7-782HQ* procesoru, 32GB operatīvo atmiņu un *Windows 10 Enterprise* operētājsistēmu. Avota sistēma un datu noliktava ir realizēta *VMware* virtuālajos serveros ar operētājsistēmu *Windows Server 2008 R2 Enterprise*, un avota sistēmas

virtuālajam serverim ir piešķirti 2 procesora kodoli un 4GB operatīvās atmiņas, bet datu noliktavas virtuālajam serverim ir piešķirti 4 procesora kodoli un 16GB atmiņas.

5.2. Datu replikācijas realizācija

Datu replikācijas no avota sistēmas uz datu noliktavu realizācijai ir nepieciešams veikt atbilstošos sagatavošanas darbus abās datu bāzēs, uz abiem serveriem ir jāinstalē *Oracle GodlenGate* programmatūra un jākonfigurē pārvaldības process, kā arī uz avota sistēmas ir jārealizē datu vākšanas process, bet uz mērķa sistēmas ir jārealizē datu replikācijas process.

5.2.1. Datu bāzes sagatavošanas process Oracle GoldenGate

Lai *Oracle GoldenGate* procesi varētu veiksmīgi strādāt ar datu bāzēm avota un mērķa serveros, datu bāzēs ir jābūt iespējotam minimālajam papildu žurnālam (no angļu *minimal supplemental logging*), izveidotam datu bāzes lietotājam ar atbilstošām privilēģijām un datu bāzēm ir jābūt *archivelog* režīmā. [26]

Oracle datubāzes minimālais papildu žurnāls ir nepieciešams, lai datu bāzes transakciju atkārtošanas žurnāla datnēm pievienotu ierakstu ķēdes informāciju ierakstu atjaunošanas operācijām. Informācija par to, vai minimāla papildu žurnāla funkcija ir iespējota, ir pieejama *v$database* datu bāzes skata *supplemental_log_data_min* laukā. Gadījumā, ja šī funkcija nav iespējota, lietotājam ar sistēmas izmainīšanas datubāzes privilēģiju ir jāizpilda attēlā 5.3. redzamā SQL komanda.

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
```

5.3. att. **Komanda minimālā papildu žurnāla funkcijas iespējošanai [25]**

Lai *Oracle GoldenGate* procesi varētu piekļūt datu bāzes objektiem, ir nepieciešams izveidot datu bāzes lietotājus ar šiem procesiem atbilstošām privilēģijām. Datu izvilkšanas procesa datu bāzes lietotājam ir nepieciešamas datu bāzes privilēģijas, kas ļauj tam izpildīt vaicājumus pret avota datu bāzes metadatiem un avota tabulām. Datu replikācijas datu bāzes lietotājam ir nepieciešamas datu bāzes privilēģijas, lai tas varētu veidot ierakstus kontrolpunkta tabulā un replicēt DML komandas mērķa datu bāzes tabulās. Papildu tam, gadījumos, kad tiek replicētas arī avota sistēmas DDL komandas, atsevišķs lietotājs ir nepieciešams arī pārvaldības procesam, lai uzturētu *Oracle GoldenGate* datu bāzes objektus, kas ir nepieciešami, lai vāktu avota sistēmā izpildītās DDL komandas. Visas datu bāzes

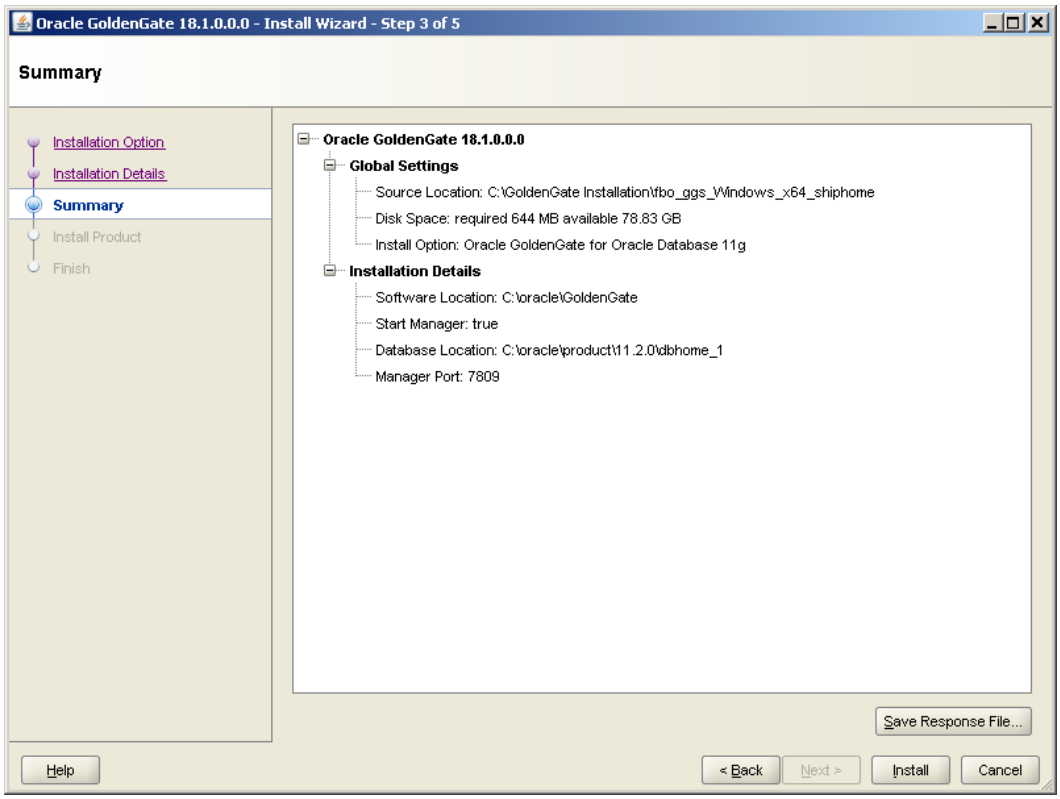
lietotājiem nepieciešamās privilēģijas ir redzamas 1. pielikumā. Lai piešķirtu lietotājiem nepieciešamās datu bāzes privilēģijas ir jāizmanto, gan *dbms_goldengate_auth* pakotnes *grant_admin_privilege* funkcija, gan ar atsevišķām privilēģiju piešķiršanas komandām. Gan avota sistēmas datu bāzei, gan datu noliktavas datu bāzei ir izveidoti divi datu bāzes lietotāji *Oracle GoldenGate* procesu vajadzībām – *ggadm* lietotājs, ar datu bāzes privilēģijām, kas nepieciešamas attiecīgi datu izvilkšanai un replikācijai, un *ggmgr* lietotājs ar datu bāzes privilēģijām, kas nepieciešamas pārvaldības procesam. [25]

Lai datu izvilkšanas process varētu reālā laikā izvilkt datus no avota sistēmas transakciju atkārtošanas žurnāla datnēm, avota sistēmai ir jābūt *archivelog* režīmā. Šis režīms nozīmē to, ka tā vietā, lai pilnās transakciju atkārtošanas žurnāla datnes vienkārši izdzēstu, tās tiek arhivētas un saglabātas datu bāzes parametros norādītajā datnē. Lai pārbaudītu to, vai datu bāze ir *archivelog* režīmā, pieslēdzoties datu bāzei ar *sysdba* lietotāju ir jāizpilda *archive log list;* komanda, un pie atgrieztajiem rezultātiem datu bāzes žurnāla režīmam ir jābūt norādītam kā arhīva režīmam.

Ja datu bāze nav arhīva režīmā, ir nepieciešams uzstādīt datu bāzes parametrus *db_recovery_file_dest*, kurš norāda mapi, kurā saglabāt arhivētās transakciju atkārtošanas žurnāla datnes, un *db_recovery_dest_size*, kurš norāda cik daudz vietas krātuvē ir atvēlēts šīm datnēm. Šī risinājuma ietvaros arhivētām transakciju atkārtošanas datnēm uz avota servera tika atvēlēts 2GB krātuves, bet uz mērķa servera tām tika atvēlēts 8GB krātuves, lai pietiktu vietas datu bāzes dublējuma datnēm. Mapi, kurā glabāt arhivētas transakciju atkārtošanas datnes ir vēlams norādīt ārpus datu bāzes instalācijas datnes, kas arī tika ievērots abos serveros. Pēc šo parametru uzstādīšanas datu bāzei ir nepieciešams pieslēgties ar *sysdba* lietotāju, izslēgt datu bāzi ar *shutdown immediate* komandu un atslēgties no tās. Tagad ir nepieciešams atkal pieslēgties datu bāzei ar *sysdba* un sāknēt datu bāzi ar *startup mount* komandu, bet nepadarīt to atvērtu. Kad datu bāze ir šādā stāvoklī ir nepieciešams izpildīt *alter database archivelog;* komandu, lai datu bāzi pārslēgtu *archivelog* režīmā. Pēc tam datu bāzi ir jāpadara atvērtu ar *alter database open;* komandu. Lai pārbaudītu, vai arhivētas datu bāzes transakciju atkārtošanas datnes tiek saglabātas norādītajā mapē, ir nepieciešams izpildīt *alter system swhitch logfile;* komandu, kas liek datu bāzei pārslēgties uz jaunu transakciju atkārtošanas žurnāla datni. Pēc šīs komandas izpildīšanas *db_recovery_file_dest* norādītajā mapē ir jābūt arhivētai transakciju žurnāla datnei. Šis process ir vienāds gan *Oracle Database 11g*, gan *Oracle Database 12c.* [25, 27, 28]

5.2.2. Oracle GoldenGate instalācija un pārvaldības procesa konfigurācija

Oracle GoldenGate 18c instalācijas process ir īss, un tajā ir ierobežots skaits iespēju, kuras var tikt pielāgotas lietotāja vajadzībām. Instalācijas procesa sākumā ir jānorāda kādai datu bāzei šis rīks tiek instalēts. *Oracle GoldenGate 18c* instalācija ļauj izvēlēties starp sekojošām *Oracle* datu bāzes versijām: *Oracle Database 18c*, *Oracle Database 12c* un *Oracle Database 11g*. Uz avota servera *Oracle GoldenGate* ir uzinstalēts *Oracle Database 11g* versijai, bet uz datu noliktavas servera tas ir uzinstalēts priekš *Oracle Database 12c* versijas. Turpmākajā instalācijas procesā ir jānorāda mape, kurā instalēt *Oracle GoldenGate*, datu bāzes instalācijas mape, kura tiek aizpildīta automātiski, ja ir norādīts sistēmas mainīgais *ORACLE_HOME*, ports, kuru izmantos *Oracle GoldenGate* pārvaldības process, un nosacījumu, vai sāknēt pārvaldības procesu pēc instalācijas. Instalācijas procesā uz abiem serveriem tika atstāta noklusētā pārvaldības procesa porta vērtība 7809. Attēlā 5.4. ir redzama *Oracle GoldenGate* instalācijas kopsavilkuma lapa no avota sistēmas. Ar analogiem parametriem *Oracle GoldenGate* ir uzstādīts gan uz avota sistēmas, gan uz datu noliktavas servera. [26]



5.4. att. ***Oracle GoldenGate* instalācijas kopsavilkuma lapa no avota sistēmas**

Pēc *Oracle GoldenGate* instalācijas ir nepieciešams izveidot šī rīka darba mapes. Lai to izdarītu ir nepieciešams izmantot *Oracle GoldenGate* komandu rindas saskarni *GGSCI*, kuru ir iespējams izpildīt startējot *ggsci.exe* izpildāmo datni mapē, kurā rīks ir uzinstalēts. Šī

saskarne ir izmantojama, lai izpildītu *Oracle GoldenGate* komandas procesu konfigurācijai, pārvaldībai un uzraudzībai. Lai izveidotu *Oracle GoldenGate* darba mapes, šajā rīkā ir jāizpilda *craete subdirs* komanda, kura izveidos visas *GoldenGate* procesiem nepieciešamās mapes. Šīs mapes tiek izmantotas, piemēram, lai uzturētu procesu parametru, pierakstu un atskaišu datnes. [26]

Ja ir nepieciešams, *Oracle GoldenGate* pārvaldības procesu var pievienot *Windows* servera pakalpojumiem. Lai izveidotu *Windows* servisu ar pielāgotu nosaukumu, šo nosaukumu ir jānorāda *Oracle GoldenGate* globālo parametru datnēs parametrā *mgrservicename*. Globālo parametru datne glabā parametrus, kuri tiek izmantoti visos *Oracle GoldenGate* procesos, bet pēc nepieciešamības šos parametrus var pārrakstīt katra atsevišķa procesa parametra datnēs. Lai izmainītu globālo parametru datni ir nepieciešams izmantot *Oracle GoldenGate* komandu rindas saskarni *GGSCI*. Globālo parametru datnes rediģēšanai šajā rīkā ir jāizpilda komanda *edit params ./globals*, kur, ja tas jau nav izdarīts, izveido globālo parametru un atver to rediģēšanai. Risinājumā izmantotās globālās parametru datnes ir redzamas attēlā 5.5. [26, 29]

```
MGRSERVNAME TIAGoldenGateManager
```

(a)

```
MGRSERVNAME XIAGoldenGateManager
CHECKPOINTTABLE ggadm.gg_checkpoint_table
```

(b)

5.5. att. **Risinājumā izmantotie globālās *GoldenGate* parametru datnes:**

(a) avota sistēmas globālā parametru datne, (b) datu noliktavas globālā parametru datne

Avota sistēmas globālajā parametru datnē ir nepieciešams norādīt tikai *Windows* pakalpojuma nosaukumu, jo visi procesiem nepieciešamie parametri tiks norādīti katra procesa parametru datnē atsevišķi. Datu noliktavas globālajā parametru datnē ir norādīts arī kontrolpunkta tabulas nosaukums, jo visi izveidotie replikācijas procesi izmantos vienu un to pašu kontrolpunkta tabulu. Lai pievienotu *GoldenGate* pārvaldības procesu *Windows* servera pakalpojumu sarakstam ar operētājsistēmas sistēmas komandrindu ir jāpāriet uz mapi, kurā ir instalēts *GoldenGate*, un jāizpilda *install addservice* komanda. Pēc šīs komandas izpildīšanas *GoldenGate* pārvaldības process ir pievienots *Windows* pakalpojumu sarakstam ar globālajā parametru datnē norādīto nosaukumu.

Datu bāzes lietotāju kontus, kurus izmanto *Oracle GoldenGate,* pierakstīšanās informācijas glabāšanai ir ieteicams izmantot šajā rīkā iebūvēto pierakstīšanās informācijas glabātuvi. Šīs glabātuves izmantošana nodrošina to, ka lietotāju pierakstīšanās informācija tiek glabāta šifrētā datnē un parametru datnēs neparādās datu bāzes lietotāju paroles, kā arī tā ļauj izmantot lietotāju segvārdus, kas atšķiras no datu bāzes lietotāja vārda. Lai izveidotu lietotāju pierakstīšanās informācijas glabātuvi *GGSCI* ir jāizpilda komanda *add*

credentialstore. Pēc šīs komandas izpildīšanas tiek izveidota tukša lietotāju pierakstīšanās informācijas glabātuve. Lai pievienotu lietotāju pierakstīšanās informāciju šai glabātuvei *GGSCI* ir jāizpilda komanda *alter credentialstore add user [username] password [password] alias [alias]*. Pēc šīs komandas izpildīšanas, lai procesu parametru datnēs norādītu lietotāja kontu, ar kuru pieslēgties datu bāzei, var izmantot parametru *useridalias [alias]*. Lai pārbaudītu vai ar izveidoto segvārdu var pieslēgties datu bāzei GGSCI var izmantot komandu *dblogin useridalias [alias]*, lai pieslēgtos datu bāzei. Šādā veidā gan uz avota sistēmas, gan uz datu noliktavas sistēmas ir izveidoti segvārdi datu bāzes lietotājiem *ggadm* un *ggmgr*. [25, 30]

Pirms uzsākt datu izvilkšanas un replikācijas procesu konfigurāciju, abos serveros ir nepieciešams konfigurēt *GoldenGate* pārvaldības procesu. Pārvaldības process pārvalda datu izvilkšanas un replikācijas procesus, piešķir tiem nepieciešamos portus, veic pierakstu datņu uzturēšanu un veido notikumu un kļūdu atskaites. Lai atvērtu pārvaldības procesa parametru datni rediģēšanai, *GGSCI* ir jāizpilda komanda *edit params mgr*. Izstrādātajā risinājumā izmantotās pārvaldības datnes ir redzamas attēlā 5.6.

```
PORT 7809
DYNAMICPORTLIST 7810-7820, 7830
AUTOSTART EXTRACT *
AUTORESTART EXTRACT *, RETRIES 4, WAITMINUTES 4
STARTUPVALIDATIONDELAY 5
USERIDALIAS ggmgr
PURGEOLDEXTRACTS dirdat\* , USECHECKPOINTS, MINKEEPHOURS 2
```

(a)

```
PORT 7809
DYNAMICPORTLIST 7810-7820
AUTOSTART REPLICAT *
AUTORESTART REPLICAT *, RETRIES 4, WAITMINUTES 4
STARTUPVALIDATIONDELAY 5
USERIDALIAS ggmgr
PURGEOLDEXTRACTS dirdat\*, USECHECKPOINTS, MINKEEPHOURS 2
```

(b)

5.6. att. **Risinājumā izmantotās *GoldenGate* pārvaldības procesu parametru datnes:**
(a) avota sistēmas pārvaldības procesa parametru datne, (b) datu noliktavas pārvaldības procesa parametru datne

Avota sistēmas pārvaldības procesa parametru datne un datu noliktavas pārvaldības procesa parametru datne ir analogas viena otrai. Vienīgā atšķirība ir tā, ka *autostart* un *autorestart* parametri strādā attiecīgi ar datu izvilkšanas procesiem un datu replikācijas procesiem. Pie parametra *port* pārvaldības procesa parametru datnē tiek norādīts, kāds ports ir atvēlēts pārvaldības procesam, lai mijiedarbotos ar ārējiem procesiem. Šis ports tika izvēlēts rīka instalēšanas procesā. Ar parametru *dynamicportlist* tiek norādīts saraksts ar portiem, ko pārvaldības process var piešķirt lokālajiem procesiem, lai tie varētu mijiedarboties ar *GoldenGate* procesiem uz citiem serveriem. *Autostart* parametrs norāda kādus *GoldenGate* procesus pārvaldības procesam izpildīt automātiski, pēc tam, kad tas ir izpildīts. Gadījumos, kad kāds no *GoldenGate* procesiem beidz savu darbību ar kļūdu, *autorestart* parametrā var norādīt, lai šis process tiktu automātiski sākts no jauna un cik reizes automātisku uzsākšanu atkārtot, un cik ilgi gaidīt starp atkārtotiem mēģinājumiem. Ar parametru *useridalias* tiek norādīts, kādu lietotāju no lietotāju pieslēgšanas informācijas glabātuves izmantot, lai pieslēgtos datu bāzei, kad tas ir nepieciešams. Lai sistēmā neglabātos vecas pierakstu datnes

pārvaldības procesam ir ieteicams norādīt *purgeoldextracts* parametru. Šo parametru var norādīt arī datu izvilkšanas vai replikācija procesa parametros, bet tā nenorādīšanas gadījumā vecās pierakstu datnes netiks dzēstas no servera, kas var novest pie tā, ka tās nevajadzīgi izmanto lielu krātuves apjomu. Parametrs *usecheckpoints* liek pārvaldības procesam pārliecināties, ka visi lokālie procesi ir beiguši darbu ar doto lokālo pierakstu datni pirms tās dzēšanas un *minkeephours* norāda, cik stundas pierakstu datnes ir jāpatur sistēmā. Pārvaldības procesus var uzsākt *GGSCI* izpildot komandu *start mgr* vai no *Windows* pakalpojumu saraksta, ja tas ir pievienots šim sarakstam. [29, 30]

5.2.3. Datu izvilkšanas reālā laikā realizācija

Datu izvilkšanas un nosūtīšanas realizācijai ir nepieciešams konfigurēt datu izvilkšanas procesu un datu sūknēšanas procesu. Šajā darbā ir aprakstīts polises tabulas izmaiņu izvilkšanas un nosūtīšanas process.

Pirmais solis datu izvilkšanas realizācijai ir iespējot papildu žurnālu tabulai, no kuras datu izvilkšanas procesam ir jāiegūst dati. Lai to izdarītu no *GGSCI* ir jāpieslēdzas datu bāzei ar lietotāju, kam ir nepieciešamās datu bāzes privilēģijas, lai izmainītu doto tabulu, un jāizpilda komanda *add trandata [table]*. Pēc šīs komandas izpildīšanas norādītajai tabulai tiks iespējots papildu žurnāls, un *Oracle GoldenGate* datu izvilkšanas process varēs notvert izmaiņas šajā tabulā.

```
EXTRACT p_ext
USERIDALIAS ggadmin
EXTTRAIL dirdat\pt
TABLE tia.polise, COLS (ID, Polises īpašnieka ID, Polises numurs, Transakcijas numurs,
Starpnieka ID, Realizācijas personas ID, Laika zīmogs, Seguma sākuma datums, Seguma beigu datums,
Seguma gada sākuma datums, Pagarināšanas datums, Pirmais seguma sākuma datums, Polises gads,
Jaunākais ieraksts, Iepriekšējās versijas ID), KEYCOLS (ID);
```

5.7. att. **Izstrādātā risinājuma datu izvilkšanas procesa parametru datne**

Attēlā 5.7. ir redzams risinājumā izmantotā datu izvilkšanas procesa parametru datne. Attēlā ir izmainīti tabulas un kolonnu nosaukumi. Lai izveidotu vai rediģētu jau eksistējošu procesa parametru datni, *GGSCI* ir jāizpilda komanda *edit params [process]*, kas šajā gadījumā nozīmē *edit params p_ext*. Datu izvilkšanas procesa parametru datnes pirmajā rindā ir jānorāda kāda tipa process tas ir un procesa nosaukums. Arī šajā datnē ir nepieciešams norādīt datu bāzes lietotāju, ar kuru šis process var pieslēgties pie datu bāzes, šajā gadījumā ir norādīts lietotājs no lietotāju pierakstīšanās glabātuves. Parametrs *exttrail* norāda lokālās pierakstu datnes ceļu, kurā procesam rakstīt notvertās izmaiņas. Parametrs *table* nosaka, no kuras tabulas datu izmaiņas tiks vāktas. Šī parametra *cols* papildparametrā var norādīt datu

bāzes laukus, kuru izmaiņas procesam ir jāreģistrē un papildparametrā *keycols* var norādīt laukus, kurus izmantojot var identificēt ierakstu. Šie ir pamata parametri, kas ir nepieciešami, lai datu izvilkšanas process varētu strādāt.

Pēc parametru datnes izveides, datu izvilkšanas procesu ir nepieciešams reģistrēt kā *Oracle GoldenGate* procesu. To var izdarīt izmantojot *GGSCI* komandu *add extract*. Šai komandai ir jānorāda avots, no kura datu izvilkšanas process iegūs datus, kā arī laiks, no kura sākt datu izvilkšanu. Šajā gadījumā dati tiks iegūti no datu bāzes transakciju atkārtošanas žurnāla datnes, un datu iegūšana tiks sākta brīdī, kad process tiks izpildīts. Tas nozīmē, ka komanda, kas ir jāizpilda ir *add extract p_ext, tranlog, begin now*. Pēc šīs komandas izpildīšanas, datu izvilkšanas process ir reģistrēts *Oracle GoldenGate*. Lai pabeigtu datu izvilkšanas procesa izveidi, ir nepieciešams reģistrēt lokālo pierakstu datni, kurā datu izvilkšanas process rakstīs datus. Lai to izdarītu ir jāizpilda komanda *add exttrail*, kurā norāda pierakstu datnes nosaukumu, datu izvilkšanas procesu, kuram pierakstu datne ir piesaistīta, un maksimālo datnes izmēru. Pilna komanda, kas ir izpildāma, lai piesaistītu pierakstu datni datu izvilkšanas procesam ir *add exttrail dirdat\pt, extract p_ext, megabytes 50*. Šajā brīdī datu izvilkšanas procesa izveide ir pabeigta, un, lai sāktu datu izvilkšanu no datu bāzes, ir jāizpilda komanda *start extract p_ext*. Pēc šīs komandas izpildes, norādītajā mapē ir jābūt izveidotai pierakstu datnei, kurā tiks rakstītas notvertās datu izmaiņas. Lai redzētu datu izvilkšanas procesa statusu *GGSCI* ir jāizpilda komandu *info extract* *. [25, 30]

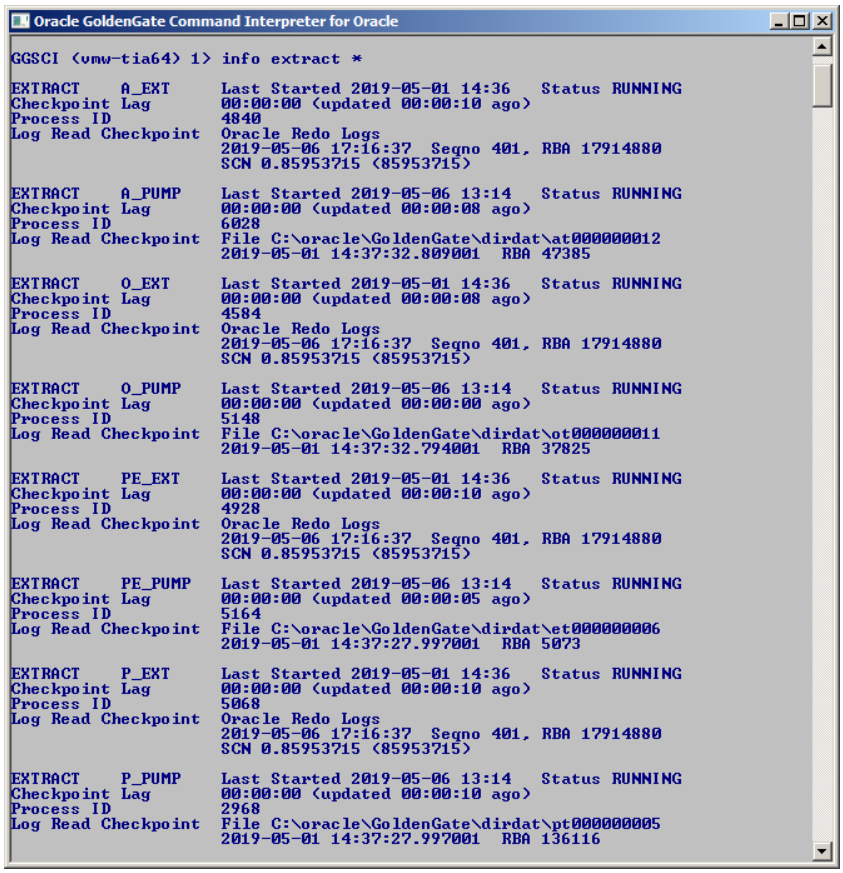
```
EXTRACT p_pump
RMTHOST xiahost, MGRPORT 7809
RMTTRAIL dirdat\rp
TABLE tia.polise, COLS (ID, Polises īpašnieka ID, Polises numurs, Transakcijas numurs,
Starpnieka ID, Realizācijas personas ID, Laika zīmogs, Seguma sākuma datums, Seguma beigu datums,
Seguma gada sākuma datums, Pagarināšanas datums, Pirmais seguma sākuma datums, Polises gads,
Jaunākais ieraksts, Iepriekšējās versijas ID), KEYCOLS (ID);
```

5.8. att. **Izstrādātā risinājuma datu sūkņa procesa parametru datne**

Datu nogādāšanai no avota sistēmas datu noliktavā ir nepieciešams izveidot datu sūkņa procesu. Šis process ir nepieciešams, lai nodrošinātos pret datu zudumiem, gadījumos, kad datus nav iespējams nogādāt mērķa sistēmā tīmekļa kļūmju dēļ. Izstrādātajā risinājumā izmantotā datu sūkņa parametru datne ar izmainītiem tabulas un kolonnu nosaukumiem ir redzama attēlā 5.8. Parametru datnes izveidošana notiek ar analogu komandu, komandai, kas ir izmantota, lai izveidotu datu izvilkšanas parametru datni. Šajā parametru datnē nav nepieciešams norādīt lietotāju piekļūšanai datu bāzei, jo šis process nestrādā ar datu bāzi. Atšķirībā no datu izvilkšanas procesa, datu sūkņa process datus raksta pierakstu datnē uz mērķa servera, jeb ārējā pierakstu datnē. Šī iemesla dēļ šajā parametru datnē ir jānorāda arējā servera IP adrese vai resursdatora nosaukums, arējā servera *GoldenGate* pārvaldības procesa ports un pierakstu datnes nosaukums ar parametriem *rmthost, mgrport* un *rmttrail*. Parametrs

table, tāpat kā datu izvilkšanas procesā, tiek izmantots, lai norādītu avota tabulu, kolonnas, kuras datnē ir iekļautas, un kolonnu, pēc kuras ierakstu var identificēt.

Datu sūkni, tāpat kā datu izvilkšanas procesu, ir nepieciešams reģistrēt *Oracle GoldenGate*. Arī šajā gadījumā ir jāizmanto *GGSCI* komanda *add extract*, bet atšķirībā no datu izvilkšanas procesa, kā datu avots ir jānorāda lokālā pierakstu datne un nav nepieciešams norādīt laiku, no kura sākt iegūt datus, jo šis process datus iegūs no datnes sākuma, vai vietas datnē, kur tas ir ticis pārtraukts. Komanda, kas ir jāizpilda, lai reģistrētu datu sūkņa procesu ir *add extract p_pump, exttrailsource dirdat\pt*. Atšķirībā no datu izvilkšanas procesa, datu sūkņa procesam ir nepieciešams reģistrēt ārējo pierakstu datni. Lai to izdarītu ir jāizpilda komanda *add rmttrail dirdat\rp, extract p_pump, megabytes 50*. Šo procesu var izpildīt ar komandu *start extract p_pump*. Pēc šīs komandas izpildes uz mērķa servera ir jābūt izveidotai pierakstu datnei, kurā tiks ierakstīti dati no avota sistēmas lokālās pierakstu datnes. [25, 30]



5.9. att. **Avota sistēmā realizētie datu izvilkšanas procesi**

Kopumā, lai realizētu visu četru attēlā 4.4. redzamo tabulu replikāciju datu noliktavā, ir nepieciešams izveidot 8 datu izvilkšanas procesus: 4 procesus, kas vāc izmaiņas no norādītajām tabulām, un 4 datu sūkņa procesus. Visi izveidotie procesi ir redzami attēlā 5.9.

5.2.4. Datu replikācijas reālā laikā realizācija

Notverto izmaiņu replicēšanai datu noliktavā, uz datu noliktavas servera ir nepieciešams izveidot *Oracle GoldenGate* datu replicēšanas procesu. Šis process strādā ar datu sūkņa izveidoto pierakstu datni, un brīdī, kad tajā tiek ierakstītas jaunas izmaiņas, tas izveido izmaiņām atbilstošas DML komandas, un izpilda tās mērķa datu bāzē.

Datu replikācijas procesam ir nepieciešama kontrolpunktu tabula. Šī tabula tiek izmantota, lai datu replikācijas process pēc tā apturēšanas varēt reģistrēt, kur tas ir beidzis apstrādāt pierakstu datni, un pēc procesa atsākšanas turpināt datnes apstrādi no šīs vietas, kā arī nodrošināt to, ka katra pierakstu datnē ierakstītā izmaiņa tiek replicēta tieši vienu reizi. Kontrolpunktu tabulas nosaukums ir definēts datu noliktavas servera *Oracle GoldenGate* globālajos parametros, kas ir redzami attēlā 5.5. Lai izveidotu šo tabulu no *GGSCI* ir jāpieslēdzas datu bāzei ar *dblogin* komandu un jāizpilda *add checkpointtable* komanda. Pēc šīs komandas izpildīšanas datu bāzē var pārliecināties par to, ka šī tabula ir izveidota.

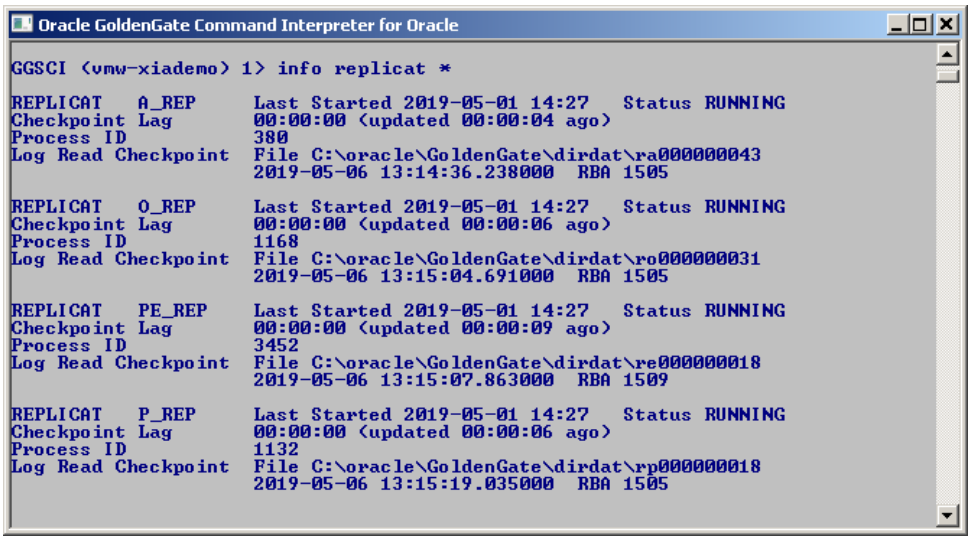
Lai realizētu datu replikāciju mērķa datu bāzē ir jāizveido tabulu, kura satur visus laukus, kuru izmaiņas vāc datu izvilkšanas process, ar tādiem pašiem datu tipiem, kādi tie ir avota sistēmā. Datu noliktavas *ggadm* shēmā, kura kalpo kā reāla laika datu shēma, ir izveidota šāda tabula rt_polise_izstade. Lai šī tabula spētu ātrāk apstrādāt datu izmaiņas un lietotāju vaicājumus, tā tika izveidota kā atmiņā esoša tabula, izmantojot *inmemory* atslēgas vārdu. Tas nozīmē to, ka tabulas ieraksti tiek glabāti operatīvajā atmiņā un vaicājumu vai datu izmaiņu izpildīšanai nav nepieciešams datus nolasīt no cietā diska. [25, 27, 29]

```
REPLICAT p_rep
USERIDALIAS ggadmin
ASSUMETARGETDEFS
REPERROR (1403, DISCARD)
MAP tia.polise, TARGET ggadm.rt_polise_izstade, KEYCOLS (ID);
```

5.10. att. **Izstrādātā risinājuma datu replikācijas parametru datne**

Datu replikācijas procesa konfigurācija notiek līdzīgi kā datu izvilkšanas procesu konfigurācija. Attēlā 5.10. ir redzama risinājumā izmantotā polises datu replikācijas procesa parametru datne ar izmainītiem tabulu un kolonu nosaukumiem. Tāpat kā datu izvilkšanas procesiem, arī, lai izveidotu un rediģētu datu replikācijas procesa parametru datni, *GGSCI* ir jāizpilda *edit params [process]* komanda. Datu replikācijas parametru datnes pirmajā rindā ir jānorāda procesa tips un nosaukums. Lai šis process varētu piekļūt datu bāzei un replicēt tajā no pierakstu datnes iegūtās datu izmaiņas, šajā parametru datnē ir nepieciešams norādīt datu bāzes lietotāju, kuram ir tiesības izveidot, izmainīt un dzēst ierakstus mērķa tabulā. Ar parametru *assumetargetdefs* datu replikācijas procesam tiek norādīts, ka saņemto lauku datu

tipi ir tādi paši, kā mērķa tabulas laukiem ar vienādiem nosaukumiem. Kad avota sistēma izveido jaunas polises transakcijas, tā ne tikai izveido jaunus ierakstus atbilstošajās tabulās ar jauno datu bāzes objekta informāciju, bet tā arī atjauno iepriekšējo šī objekta versijas ierakstu, ierakstot tajā nākamās versijas identifikatoru. Tā kā šajā risinājumā reāla laika tabulas neglabā visus avota sistēmas datu bāzes ierakstus, bet tikai tos ierakstus, kas ir izveidoti pēc pēdējā ETL procesa izpildes, tad gadījumos, kad iepriekšējā versija ir izveidota pirms šī procesa izpildes, replikācijas procesa mēģinājums atjaunot neeksistējošo ierakstu beigsies ar datu bāzes kļūdu ar kodu 1403. Tā kā datu noliktavas reāla laika tabulās ir nepieciešams iegūt tikai jaunus ierakstus, un datu noliktavā jau ielādēto ierakstu atjaunošana nav nepieciešama, tad ar parametra *reperror(1403, discard)* palīdzību replikācijas procesam tiek norādīts, ka šādus ierakstus procesā var ignorēt. Datu replikācijas procesa parametru datnes pēdējā rindā tiek norādīts, ka no avota sistēmas polises tabulas iegūtos ierakstus ir nepieciešams kartēt uz mērķa datu bāzes tabulu rt_polise_izstade, un, ka ierakstus var identificēt pēc lauka ID. Šī ir nepieciešamā parametru kopa, lai datu replikācijas process varētu veiksmīgi replicēt no pierakstu datnes nolasītās datu izmaiņas mērķa datu bāzē. Lai reģistrētu replikācijas procesu *GoldenGate, GGSCI* ir nepieciešams izpildīt komandu *add replicat p_rep, exttrail dirdat\rp*. Pēc šīs komandas izpildīšanas datu replikācijas process ir reģistrēts *Oracle GoldenGate* un to var izpildīta ar komandu *start replicat p_rep*.



5.11. att. **Datu noliktavā realizētie datu replikācijas procesi**

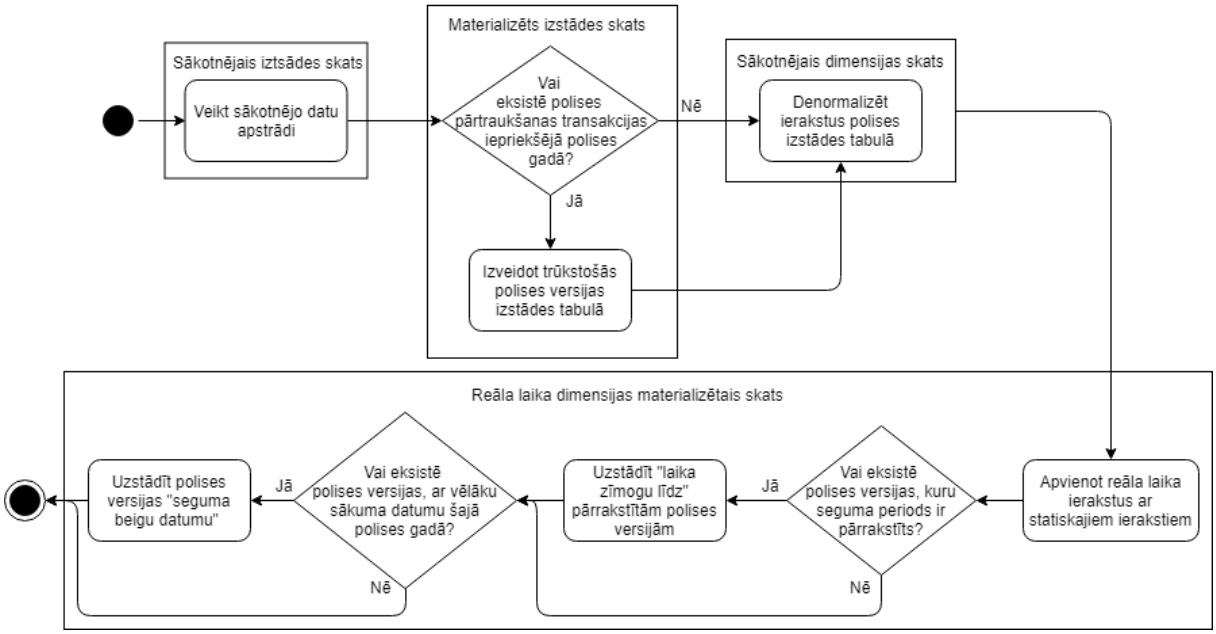
Kopumā, lai veiktu četru nepieciešamo tabulu datu replikāciju, ir nepieciešam realizēt četrus replikācijas procesus. Šo procesu realizācija ir analoga šajā apakšnodaļā aprakstītajam procesam, izņemot to, ka visi procesi izmanto vienu un to pašu kontrolpunktu tabulu, kas nozīmē, ka tā nav jāizveido atkārtoti. Visi risinājumā izveidotie datu replikācijas procesi ir redzami attēlā 5.11. Šādā veidā datu bāzes ieraksts nonāk datu noliktavas reāla laika tabulās aptuveni 7 sekundes pēc tam, kad tas ir izveidots vai izmainīts avota sistēmā.

5.3. Datu transformācija un attēlošana reālā laikā

Datu transformācijas realizēšanai reālā laikā, datu noliktavā ir iestrādāti vairāki klasiskie datu bāzes skati, kā arī materializēti datu bāzes skati. Šie skati iegūst datus viens no otra, pakāpeniski transformējot reāla laika datus un integrējot tos ar statiskajiem datu noliktavas datiem. Lai pilnībā integrētu reāla laika datu noliktavu ar statisko datu noliktavu ir nepieciešamas nelielas izmaiņas datu noliktavas lietotnē, kā arī datu noliktavas ETL procesā.

5.3.1. Polises un polises līnijas dimensiju transformācija reālā laikā

Pēc datu ielādes reāla laika datu noliktavas tabulās, lai šos datus integrētu datu noliktavā, tiem ir nepieciešams veikt transformācijas, kas datus pārvērš formā, kura iederas klasiskajā datu noliktavā gan pēc to formas, gan satura. Izstrādātajā risinājumā šī datu transformācija ir realizēta izmantojot vairāku datu bāzes skatu un materializētu datu bāzes skatu kombināciju. Attēlā 5.12. ir redzama polises dimensijas reāla laika datu transformācijas un integrācijas ar statiskās datu noliktavas datiem process.



5.12. att. **Reāla laika polises dimensijas transformācijas procesa aktivitāšu diagramma**

Pirmā darbība pēc datu ielādes datu noliktavas reāla laika tabulās ir datu sākotnējās apstrādes izpilde. Šī darbība ir realizēta izmantojot datu bāzes skatu, un tās izpildītās darbības ir analogas darbībām statiskajā datu noliktavas ETL procesā, kas tiek veiktas pirms datu ielādes polises dimensijas izstādes tabulā. Analogi statiskajam ETL procesam, gadījumos, ja eksistē polises pārtraukšanas transakcijas iepriekšējos polises gados, kuras atbilst statiskajā

ETL procesā izmantotajiem nosacījumiem, tad šīm polises transakcijām tiek izveidoti vēsturiskie transakciju izstādes ieraksti. Šis process tiek izpildīts materializētajā skatā, kurš apvieno sākotnējo izstādes skatu pašu ar sevi, lai izveidotu trūkstošos vēstures izstādes ierakstus. Šī reāla laika materializētā skata saturs atbilst polises dimensijas izstādes tabulas saturam statiskajā datu noliktavas ETL procesā.

Polises dimensijas reāla laika datu transformācijai un integrēšanai ar statiskās datu noliktavas polises dimensijas tabulu, tiek izmantoti vēl divi datu bāzes skati. Pirmā, klasiskā datu bāzes skata uzdevums ir denormalizēt reāla laika izstādes materializētajā skatā pieejamos datus. Šajā skatā ir pieejami visi reāla laika polises dimensijas ieraksti formātā, kas atbilst datu noliktavas polises dimensijas tabulai. Lai denormalizētos polises dimensijas reāla laika datus integrētu ar statiskajā datu noliktavā esošajiem dimensijas datiem ir realizēts materializēts skats. Integrējot datus visiem ierakstiem tiek pievienots jauns lauks, kurš norāda to vai ieraksts ir iegūs no reāla laika datu noliktavas tabulām vai statiskajām datu noliktavas tabulām. Pēc datu integrācijas šis skats pārbauda vai katram dotajam ierakstam eksistē polises versija, kas pilnībā pārraksta dotās versijas seguma periodu, un gadījumos, kad šāda versijas eksistē, polises dimensijai tiek uzstādīts ieraksta aktualitātes "laika zīmogs līdz", kas iegūts no šī ieraksta aktualitātes sākuma datuma. Papildu tam, šajā skatā tiek pārbaudīts vai eksistē tādas polises transakcijas, kuru seguma periods sākas dotās polises transakcijas seguma periodā. Šādos gadījumos šai polises transakcijai tiek izmainīts seguma beigu datums, lai tas sakristu ar dienu pirms nākamās versijas seguma sākuma datuma. Šādā veidā tiek panākts, tas, ka polises versijām ir pareizi aktualitātes un seguma periodi gadījumos, kad tie ir izmainīti, bet tas nozīmē arī to, ka reāla laika risinājums neglabā pilnu polises dimensijas vēsturi, kāda tā tiek glabāta statiskajā polises dimensijas tabulā.

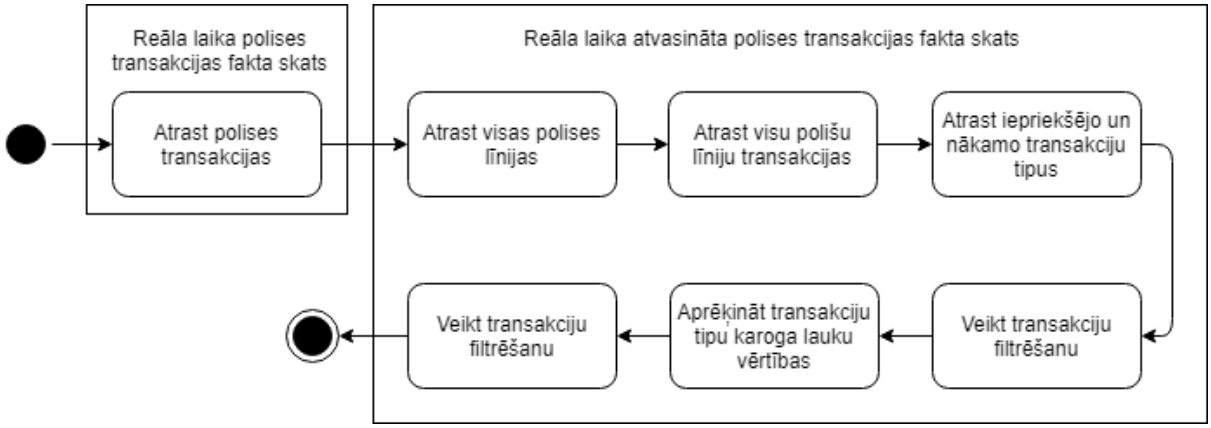
Tā kā statiskais polises līnijas dimensijas datu transformācijas process ir analogs statiskajam polises dimensijas datu transformācijas procesam, arī reāla laika polises līnijas dimensijas datu transformācijas process ir analogs reāla laika polises dimensijas datu transformācijas procesam. Arī reāla laika polises līnijas dimensijas datu transformācijas process ir realizēts izmantojot 2 klasiskos datu bāzes skatus un 2 materializētos datu bāzes skatus. Šo skatu funkcionalitāte ir analoga polises dimensijas datu transformācijā izmantotajiem skatiem, jo arī polises līnijas darījumprasības, kas nosaka to seguma un aktualitātes periodus, ir analogas polises seguma un aktualitātes periodu darījumprasībām. Arī datumu lauki, kas ir pieejami šo dimensiju izstādes tabulās un kas ir nepieciešami, lai realizētu datu transformāciju, kas atbilst darījumprasībām, ir analogi.

Datu noliktavas reāla laika dimensijas atšķirībā no statiskajām datu noliktavas dimensijām, neglabā pilnu dimensijas vēsturi jebkurā laika brīdī. Reāla laika dimensijas reāla

laika datiem ir pieejams tikai dimensijas šī brīža stāvoklis, jo, lai optimizētu ierakstu transformācijas procesu, netiek izveidoti vēsturiskie ieraksti, bet, kad tas ir nepieciešams tiek izmainīti jau eksistējošie dimensijas ieraksti datu noliktavā. Atšķirībā no statiskā polises līnijas transformācijas procesa, reāla laika transformācijas procesā netiek veikta polises transakcijas tipa koriģēšanas operācijas. Tas nozīmē to, ka atsevišķos gadījumos reāla laika polises līnijas dimensijas ierakstam var tikt uzstādīts nepareizs polises transakcijas tips, bet, tā kā šie gadījumi ir salīdzinoši reti, šāda neprecizitāte ir pieļaujama. Lai reāla laika transformācijas procesā iegūtu informāciju par iepriekšējām polises versijām, tās tiek meklētas gan reāla laika tabulās, gan statiskajās datu noliktavas tabulās.

5.3.2. Atvasināta polises transakcijas fakta transformācija reālā laikā

Atvasināta polises transakcijas fakta transformācija reālā laikā ir realizēta izmantojot divus klasiskus datu bāzes skatus. Viens skats tiek izmantots, lai iegūtu polises transakcijas faktu reālā laikā, bet otrs tiek izmantots, lai no šī skata iegūtu reāla laika atvasinātu polises transakcijas faktu. Attēlā 5.13. ir redzama aktivitāšu diagramma reāla laika atvasināta polises transakcijas fakta iegūšanai.



5.13. att. **Reāla laika atvasināta polises transakcijas fakta iegūšanas aktivitāšu diagramma**

Reāla laika polises transakcijas fakta skats iegūst polises transakcijas fakta ierakstu no reāla laika polises un polises līnijas transakcijām. Tas tiek realizēts izmantojot vaicājumu polises transakciju iegūšanai no statiskā polises transakcijas ielādes procesa, bet aizstājot statiskās dimensiju tabulas ar reāla laika dimensiju tabulām. To var darīt, jo reāla laika dimensiju skati satur gan visus statiskās datu noliktavas dimensiju ierakstus, gan reāla laika dimensiju ierakstus formā, kas sakrīt ar statiskās datu noliktavas dimensiju tabulām. Atšķirībā no statiskā polises transakcijas fakta ielādes procesa, šis vaicājums iegūst gan visus polises transakciju ierakstus, gan arī to aprakstošo informāciju. Šādā veidā tiek nodrošināts tas, ka

reāla laika polises transakcijas faktā ir pieejama tā pati informācija, kas ir pieejama statiskās datu noliktavas polises transakcijas fakta tabulā.

Arī atvasināta polises transakcijas fakta iegūšanai ir izmantots vaicājums, kas tiek izmantots šī fakta iegūšanai statiskā fakta ielādes procesā. Atšķirībā no statiskā atvasināta polises transakcijas ielādes fakta reāla laika vaicājums iegūst datus balstoties uz visām polises līnijām, kas ir pieejamas polises līnijas materializētajā reāla laika skatā. Tas tiek darīts, jo atšķirībā no statiskās datu noliktavas vaicājuma, kuram ir jāielādē tikai jaunās polises transakcijas, reāla laika skatam ir jāatgriež visas polises transakcijas, kas ir pieejamas sistēmā. Turpmākais atvasinātas polises transakcijas fakta iegūšanas process sakrīt ar šī fakta statiskās datu noliktavas ielādes procesu, par pamatu ņemot reāla laika polises transakcijas faktu. Arī reāla laika atvasinātajā polises transakcijas faktā pieejamās informācijas struktūra ir tāda pati, kā statiskās datu noliktavas atvasinātam polises transakcijas faktam.

5.3.3. Izmaiņas datu noliktavas lietotnē un ETL procesā

Veiksmīgai reāla laika datu attēlošanai datu noliktavas lietotnē tajā ir nepieciešams izveidot jaunu ekrānformu, un atbilstošo informāciju tajā balstīt uz jaunajiem reāla laika datiem. Papildus tam, lai reāla laika datus integrētu datu noliktavā, ir nepaciešams pielāgot statisko datu noliktavas ETL procesu, lai tas uzturētu reāla laika tabulas un skatus.

Quantity

Indicator	Current	Change %
GWP, €	0	0
GWP New Business, €	0	0
GWP Cancellations, €	0	0
GWP Renewals, €	0	0
GWP Expired, €	0	0
Written Instalments, €	0	0
RT No of Policies	7	-91.4
RT No of Policies New Business	10	-87.8
RT No of Policies Renewals	2	0
RT No of policies Cancellations	-5	-400
RT No of policies Expired	-3	0
RT No of policies In Force	84	3.7

5.14. att. **Reāla laika realizācijas info panelis**

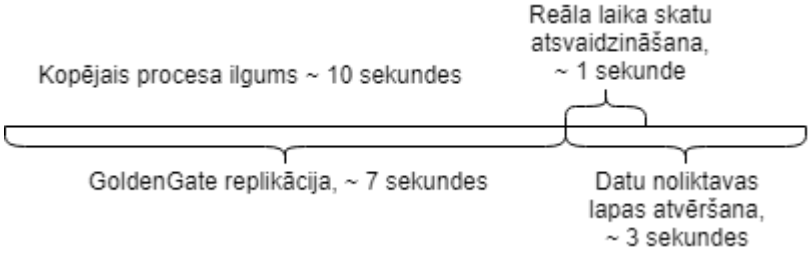
Maģistra darbā ir izmantota uz *Oracle APEX* balstīta datu noliktavas lietotne. Lai datu noliktavas lietotnē integrētu reāla laika datus, un saglabātu iespēju apskatīt informāciju, kas ir balstīta uz statiskajiem datu noliktavas datiem, datu noliktavas lietotnē ir izveidota jauna ekrānforma, kura attēlo realizācijas info paneli ar datiem, ka balstīti uz reāla laika skatiem. Lai iegūtu datus, ko attēlot datu noliktavas lietotnes realizācijas info panelī, tiek izmantota datu bāzes funkcija, kas iegūst nepieciešamo tipu transakciju skaitu norādītā laika periodā. Lai

iegūtu šo informāciju no reāla laika datiem, tika izveidota jauna datu bāzes funkcija, kas iegūst šo informāciju no reāla laika polises transakcijas fakta un atvasināta polises transakcijas fakta. Šādā veidā tiek nodrošināts tas, ka datu noliktavas lietotnē ir pieejami sekojoši izpildes pamatrādītāji reālā laikā: jaunu polišu skaits, pagarinātu polišu skaits, pārtrauktu polišu skaits, kopējo aktīvo polišu skaits, kopējā polišu skaita izmaiņas un polišu skaits, kuras ir beigušās bez pagarināšanas. Lai nodrošinātu to, ka info panelī vienmēr ir pieejami svaigākie dati, gadījumos, kad lielākais reāla laika izstādes tabulās pieejamas transakcijas numurs ir lielāks nekā lielākai reāla laika polises līnijas dimensijas materializētajā skatā pieejamais transakcijas numurs, reāla laika materializētie skati tiek atjaunoti. Materializēto skatu atjaunošanai ir nepieciešama aptuveni sekunde, tāpēc tā nepadara lietotni pārāk lēnu ērtai lietošanai.

Lai statiskais ETL process varētu veikt reāla laika tabulu uzturēšanu, tā noslēgumā ir realizēta funkcionalitāte, kas iztīra reāla laika tabulas. No reāla laika tabulām tiek dzēsti dati, kuru laika zīmogs ir agrāks par laika zīmogu, kurš tiek fiksēts statiskā ETL procesa sākumā, un līdz kuram tiek atlasīti dati no avota sistēmas. Šādā veidā tiek nodrošināts tas, ka reāla laika tabulās ir visi dati, kas nav izvilkti no avota sistēmas statiskā ETL procesa laikā. Pēc reāla laika tabulu iztīrīšanas tiek atjaunoti reāla laika skati, lai pēc ETL procesa beigām tie saturētu aktuālo informāciju. Reāla laika tabulu tīrīšana ir pēdējais solis statiskajā ETL procesā, un tas tiek izpildīts tikai gadījumos, ja ETL procesa laikā nav radušās kļūdas.

5.4. Izstrādātā risinājuma analīze

Izstrādātajā risinājumā reāla laika dati ir pieejami aptuveni 10 sekundes pēc to izveides avota sistēmā. Attēlā 5.15. ir redzams kādi procesi tiek izpildīti, lai padarītu reāla laika datus pieejamus datu noliktavas lietotnē, un to aptuvenais izpildes laiks.



5.15. att. **Reāla laika datu atjaunošana laiks**

Novērtējot procesu izpildes laiku ir jāņem vērā tas, ka šie procesi ir realizēti virtuālajos serveros ar ierobežotiem pieejamajiem resursiem, un to izpildes laiks var samazināties, ja šos procesus uzstādītu produkcijas vidēs. No procesiem, kas tiek izpildīti, lai padarītu reāla laika

datus pieejamus datu noliktavas lietotnē, visilgāk izpildās tieši datu replicēšanas process. Tā aptuvenais izpildes laiks ir 7 sekundes. Šis ir ierakstu replikācijas laiks gadījumos, kad avota tabulā polises transakcijas tiek veidotas manuāli pa vienai. Tas nozīmē, ka lai samazinātu laiku starp datu izveidi un brīdi, kad dati ir pieejami datu noliktavā, ir jāmeklē veidi kā optimizēt *Oracle GoldenGate* procesus. *Oracle GoldenGate* procesu izpildes laiku ir iespējams samazināt, piemēram, konfigurējot *Oracle GoldenGate* efektīvākai tīkla izmantošanai, pārvaldot procesiem pieejamo virtuālās atmiņas apjomu vai pielāgojot datu replikācijas procesus. [29]

Datu transformācijas un attēlošanas izpildes laiki ir apmierinoši, bet ir jāņem vērā, ka šie rezultāti ir iegūti veicot testus vidēs ar ierobežotu ierakstu skaitu. Darba turpinājumā būtu jāveic testi ar lielāku datu apjomu, lai noskaidrotu kā datu apjoma pieaugums ietekmē izstrādātā risinājuma veiktspēju. Tā kā mēneša laikā datu noliktavā kopējais paredzamais datu pieaugums ir tikai 100 MB, kas nozīmē, ka izmantotajās faktu tabulās datu pieaugums ir vēl mazāks. Var pieņemt, ka pie šāda neliela datu apjoma izstrādātais risinājums ir spējīgs veiksmīgi funkcionēt. Gadījumā, ja risinājuma veiktspēja tomēr ir pārāk maza, iespējams realizēt papildus materializētos skatus, lai uzlabotu vaicājumu efektivitāti. Sākotnēji risinājums tika izstrādāts bez materializētajiem skatiem, bet šādam risinājumam rezultātu atgriešanai bija nepieciešamas 9 minūtes, kas nozīmē to, ka materializēto skatu pievienošana var kardināli uzlabot risinājuma veiktspēju. Otra iespēja, kā uzlabot reāla laika datu transformēšanas un attēlošanas laiku, ir vienkāršot transformācijas procesus, jo izstrādātie procesi ir ļoti tuvi statiskās datu noliktavas procesiem, kas reāla laika risinājumā var nebūt nepieciešams.

REZULTĀTI

Darba izstrādes rezultātā ir izstrādāts reāla laika datu noliktavas risinājums, kas reālā laikā atjauno atvasinātu polises transakciju faktu un uz tā balstītos polises transakciju skaita izpildes pamatrādītājus. Izstrādātais risinājums ir reāla laika datu noliktavas koncepta apliecinājums, un pie tā tiks turpināts darbs, lai to padarītu par piegādājamu klientiem lietošanai klientu sistēmās. Reāla laika datu noliktavas risinājuma analīzes un koncepta apliecinājuma izstrādei tika pavadīti aptuveni 4 cilvēkmēneši.

Papildu tam ir radīts detalizēts izstrādātā risinājuma apraksts, kas ietver izstrādātā risinājuma arhitektūras, izstrādes procesa, datu noliktavas faktu struktūras un ETL procesa aprakstu. Šis apraksts sniedz ieskatu procesā, kas ir veikts, lai statisku datu noliktavas faktu pielāgotu atjaunošanai reālā laikā

Izstrādes gaitā ir radīts augsta līmeņa eksistējošās datu noliktavas XIA apraksts. Šādā veidā ir ar praktiskiem piemēriem demonstrēts, kā var realizēt klasiskas datu noliktavas, un kādas tehnoloģija var tikt izmantotas, lai realizētu uz *Oracle* tehnoloģijām balstītu datu noliktavu.

Darbā ir izstrādāts ETL procesu apraksts reāla laika datu noliktavās, kā arī ir apkopota informācija par reāla laika datu noliktavu arhitektūru. Reāla laika sistēma pilnībā neaizstāj klasiskos ETL procesus, bet strādā paralēli tiem, jo klasiskie ETL procesi var būt nepieciešami, lai padarītu reāla laika datus galu galā precīzus, un reāla laika ETL procesa realizācijā var būt nepieciešams to realizēt atsevišķā datu bāzes nodalījumā vai pat atsevišķā datu bāzē. Uz šī apraksta pamata ir izstrādāts reāla laika datu noliktavas risinājums.

SECINĀJUMI

Veicot darbu var secināt to, ka reāla laika datu noliktavas izstrādei ir nepieciešams izmantot pielāgotu statiskās datu noliktavas fakta transformācijas procesu. Tas ir nepieciešams, jo statisks ETL process var būt pārāk sarežģīts, lai to pieņemamā laikā izpildītu reāla laika datiem.

Ja realizējamais ETL process ir pietiekoši sarežģīts, tad, lai veiktu datu transformācijas 10 sekunžu laikā un datu noliktava būtu lietojama, var būt nepietiekami ar reāla laika datu glabāšanu atmiņā esošās tabulās. Šādos gadījumos var lietot materializētus skatus saglabājot tajos transformāciju starprezultātus, kurus var izmantot turpmākajos transformācijas procesa soļos. Secināts, ka šādā veidā ir iespējams gan samazināt laiku, kurš ir nepieciešams transformāciju veikšanai, gan laiku, kurš ir nepieciešams rezultātu atgriešanai datu noliktavā.

Pēc darba izstrādes secināts, ka reāla laika process nevar aizstāt eksistējoša datu noliktavas fakta statisko ETL procesu, jo reālā laikā nodrošināt visu funkcionalitāti, kas tiek nodrošināta statiskā ETL procesā, un veikt datu kvalitātes kontroli nav iespējams. Šī iemesla dēļ noliktavas faktu atjaunošanai reālā laikā ir jāizmanto atsevišķs, vienkāršots ETL process, un jāsaglabā statiskais ETL process, kurš periodiski pārrakstītu reāla laika datus, kas var būt kļūdaini, un šādā veidā saglabātu nelielu reālu laika datu apjomu, kas datu noliktavai ir jāapstrādā.

Izstrādātajā reāla laika ETL procesā lielāko daļu no procesa laika aizņem datu replikācija. No tā var secināt to, ka ir nepieciešams veikt datu replikācijas procesa optimizāciju vai datu replikācijai izmantot citu rīku, kurš spēj datu replikāciju izpildīt īsākā laika periodā, tādejādi samazinot laiku, pēc kura datus var apstrādāt datu noliktavā.

Darba turpinājumā ir nepieciešams veikt izstrādātā risinājuma slodzes testēšanu, lai pārliecinātos par to, ka šis risinājums ir lietojums pie slodzēm, kādas ir sagaidāmas uzstādot šo risinājumu reālās klientu produkcijas vidēs. Papildu tam, pirms uzstādīšanas klientu vidēs, risinājums ir jāpadara drošāks, piemēram, šifrējot pieraksta datņu saturu.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] S. Negash, "Business Intelligence," *Communications of the Association for Information Systems,* vol. 13, no. 15, pp. 177-195, 2004.

[2] J. Shim, M. Warkentin, J. F. Courtney, D. J. Power, R. Sharda and C. Carlsson, "Past, present, and future of decision support technology," *Decision Support System: Directions for the Nest Decade,* vol. 33, no. 2, pp. 111-126, 2002.

[3] N. W. Steve Williams, The Profit Impact of Business Intelligence, San Francisco: Elsevier, 2010, pp. 5-11.

[4] P. Ponniah, Data Warehousing Fundamentals for IT Professionals, 2nd ed., New Jersey: John Wiley & Sons, Inc., 2010.

[5] S. Chaudhuri, U. Dayal and V. Narasayya, "An overview of business intelligence technology," *Communications of the ACM,* vol. 54, no. 8, pp. 88-98, 2011.

[6] H. J. Watson and B. H. Wixom, "The Current State of Business Intelligence," *Computer,* vol. 40, no. 5, pp. 96-99, 2007.

[7] F. S. E. Ali, "A Survey of Real-Time Data Warehouse and ETL," *International Scientific Journal of Management Information Systems,* vol. 9, no. 3, pp. 3-9, 2014.

[8] K. Kakish and T. A. Kraft, "ETL Evolution for Real-Time Data Warehousing," in *Proceedings of the Conference on Information Systems Applied Research*, New Orleans Louisiana, USA, 2012.

[9] R. Mukherjee and P. Kar, "A Comparative Review of Data Warehousing ETL Tools with New Trends and Industry Insight," in *2017 IEEE 7th International Advance Computing Conference (IACC),* Hyderabad, 2017.

[10] N. Ferreira, P. Martins and P. Furtado, "Near real-time with traditional data warehouse architectures: factors and how-to," in *IDEAS '13 Proceedings of the 17th International Database Engineering & Applications Symposium*, Barcelona, 2013.

[11] W. Yeoh and A. Koronios, "Critical Success Factors for Business Intelligence Systems," *Journal of Computer Information Systems,* vol. 50, no. 3, pp. 23-32, 2010.

[12] R. Kimball and M. Ross, The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 3rd ed., Indiana: John Wiley & Sons, Inc., 2013.

[13] H. Chen, R. H. L. Chiang and V. C. Storey, "Business Intelligence and Analytics: From Big Data to Big Impact," *MIS Quarterly,* vol. 36, no. 4, pp. 1165-1188, 2012.

[14] R. J. Santos and J. Bernardino, “Real-time data warehouse loading methodology,” in *IDEAS '08 Proceedings of the 2008 international symposium on Database engineering & applications*, Coimbra, Portugal, 2008.

[15] K. Dobrev and M. Hart, “Benefits, Justification and Implementation Planning of Real-Time Business Intelligence Systems,” *The Electronic Journal Information Systems Evaluation*, vol. 18, no. 2, pp. 101-118, 2015.

[16] A. Vaisman and E. Zimányi, Data Warehouse Systems: Design and Implementation, Springer-Verlag Berlin Heidelberg, 2014.

[17] M. Nadj and C. Schieder, “Quo Vadis Real-Time Business Intelligence? A Descriptive Literature Review and Future Directions,” in *Proceedings of the 24th European Conference on Information Systems*, Istanbul, 2016.

[18] S. Bouaziz, A. Nabli and F. Gargouri, “From Traditional Data Warehouse To Real Time Data Warehouse,” *Intelligent Systems Design and Applications. ISDA 2016. Advances in Intelligent Systems and Computing*, vol. 557, pp. 467-477, 2017.

[19] A. Sabtu, N. F. M. Azmi, N. N. A. Sjarif, S. A. Ismail, O. M. Yusop, H. Sarkan and S. Chuprat, “The challenges of Extract, Transform and Loading (ETL) system implementation for near real-time environment,” in *2017 International Conference on Research and Innovation in Information Systems (ICRIIS)*, Langkawi, Malaysia, 2017.

[20] H. Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications, Springer Science & Business Media, 2011.

[21] N. Marz and J. Warren, Big Data: Principles and best practices of scalable realtime data systems, Shelter Island: Manning Publications Co., 2015.

[22] F. Waas, R. Wrembel, T. Freudenreich, M. Thiele, C. Koncilia and P. Furtado, “On-Demand ELT Architecture for Right-Time BI: Extending the Vision,” *International Journal of Data Warehousing and Mining*, vol. 9, no. 2, pp. 21-38, 2013.

[23] I. Lebdaoui, G. Orhanou and S. E. Hajji, “Data Integrity in Real-time Datawarehousing,” in *Proceedings of the World Congress on Engineering 2013 Vol III*, London, 2013.

[24] Oracle Corp., Oracle Fusion Middleware Understanding Oracle GoldenGate, 18c (18.1.0), Oracle, 2019.

[25] Oracle Corp., Oracle Fusion Middleware Using Oracle GoldenGate for Oracle Database, 18c (18.1.0), Oracle, 2019.

[26] Oracle Corp., Oracle Fusion Middleware Installing Oracle GoldenGate, 18c

(18.1.0), Oracle, 2019.

[27] R. Bhatiya, Oracle Database Database Administrator’s Guide, 12c Release 2 (12.2), Oracle, 2019.

[28] S. Fogel, Oracle Database Administrator's Guide, 11g Release 2 (11.2), Oracle, 2015.

[29] Oracle Corp., Oracle Fusion Middleware Administering Oracle GoldenGate, 18c (18.1.0), Oracle, 2019.

[30] Oracle Corp., Oracle Fusion Middleware Reference for Oracle GoldenGate, 18c (18.1.0), Oracle, 2019.

1. pielikums. *Oracle GoldenGate* datu bāzes lietotāju privilēģijas

Datu bāzes privilēģija	**Klasiskā izvilkšanas metode, datu bāzes versija 11.2.0.3 vai vecāka**	**Jebkāda replikācijas metode, datu bāzes versija 11.2.0.4 vai jaunāka**	**Pārvaldības process**	**Mērķis**
Izveidot sesiju un izmainīt sesiju	X	X		Pieslēgties datu bāzei.
Pieslēgties	X	X		Nepieciešama, ja procesam pieder mērķa objekti.
Resurss	X	X		Izveidot objektus.
Izmainīt sistēmu	X			Izpildīt administratīvas izmaiņas, piemēram, iespējot izmaiņu reģistrēšanu.
Privilēģijas, kas piešķirtas izmantojot *dbms_goldengate_auth.grant_admin_privilege* funkciju		X		Piešķir datu bāzes privilēģijas, kas nepieciešamas replikācijas procesam *Oracle* datu bāzē 11.2.0.4 vai jaunākā.
Ievietot, atjaunot vai dzēst datus no mērķa tabulām		X		Izpildīt replicētās DML komanda mērķa tabulā.
Izveidot tabulu		X	X	Izveidot kontrolpunkta tabulu mērķa datu

				bāzē.
Atlasīt jebkuru vārdnīcu	X	X		Nepieciešamas visas privilēģijas, lai nodrošinātu darbu ar vārdnīcu tabulām.
Izpildīt vaicājumus pret vēsturiskiem datu bāzes objektiem	X			
Izpildīt vaicājumus uz avota tabulām	X			
Izpildīt funkcijas *dbms_flashback* pakotnē	X			Izpildīt dbms*_flashback.get_system_change_number*

DOKUMENTĀRĀ LAPA

Maģistra darbs " Reāllaika analīzes funkcionalitātes izstrāde datu noliktavai" izstrādāts LU Datorikas fakultātē.

Darba teksta galīgā versija izgatavota 19.05.2019**.**

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

Autors: _____ (Oskars Kalniņš)

(Autora paraksts un datums)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **p i e m ē r o t u / n e p i e m ē r o t u** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītāja: _____ (Dr. dat. Laila Niedrīte)

(Vadītājas paraksts un datums)

Darbs iesniegts **maģistratūras sekretariātā** _____.

(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Studiju metodiķe: _____.

(Metodiķes paraksts)

Recenzents: _____

(Akad.amats, zin.grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____

(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____

(Sekretāra paraksts)