LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

**NEVIENDABĪGU INTEGRĒTU DATU AVOTU EVOLŪCIJAS APSTRĀDE**

BAKALAURA DARBS

Autors: **Lauma Svilpe**

Studenta apliecības Nr.: ls16043

Darba vadītājs: asoiciētā profesore Dr. dat. Darja Solodovņikova

RĪGA 2020

ANOTĀCIJA

ABSTRACT

Processing of integrated heterogeneous data sources evolution

SATURA RĀDĪTĀJS

[IEVADS 6](#_Toc38612652)

[1. [..teorētiskā daļa..] 7](#_Toc38612653)

[2. evolūcijas apstrādes mehānisma izstrāde 8](#_Toc38612654)

[2.1. Evolūcijas apstrādes metadati un to glabāšana 8](#_Toc38612655)

[2.1.1. Izmaiņu adaptācijas scenāriji un operācijas 10](#_Toc38612656)

[2.1.2. Izmaiņu adaptācijas scenāriju zarošanās nosacījumi 11](#_Toc38612657)

[2.1.3. Izmaiņu adaptācijas procesa papildus informācija 13](#_Toc38612658)

[2.2. Izmaiņu adaptācijas scenāriji 13](#_Toc38612659)

[2.2.1. Datu maģistrāles līmeņa pievienošana 14](#_Toc38612660)

[2.2.2. Datu avota pievienošana 15](#_Toc38612661)

[2.2.3. Datu kopas pievienošana 15](#_Toc38612662)

[2.2.4. Metadatu īpašības pievienošana 17](#_Toc38612663)

[2.2.5. Datu vienības pievienošana 17](#_Toc38612664)

[2.2.6. Datu avota dzēšana 19](#_Toc38612665)

[2.2.7. Datu maģistrāles līmeņa dzēšana 20](#_Toc38612666)

[2.3. Evolūcijas apstrādes funkcionalitāte 21](#_Toc38612667)

[2.3.1. Pirmreizējā izmaiņas apstrāde 21](#_Toc38612668)

[2.3.2. Scenārija iegūšana 24](#_Toc38612669)

[2.3.3. Scenārija izpilde 25](#_Toc38612670)

[REZULTĀTI 27](#_Toc38612671)

[SECINĀJUMI 28](#_Toc38612672)

[IZMANTOTĀ LITERATŪRA UN AVOTI 29](#_Toc38612673)

[PIELIKUMI 30](#_Toc38612674)

APZĪMĒJUMU SARAKSTS

IEVADS

1. [..teorētiskā daļa..]
2. evolūcijas apstrādes mehānisma izstrāde

Lai apstrādātu datu avotu evolūcijas rezultātā radušās izmaiņas un veiksmīgi adaptētu tās sistēmā, nepieciešams izstrādāt mehānismu, kurš darbojas, balstoties uz dažādiem metadatiem:

* dati par radušos izmaiņu;
* dati par manuāli veicamajām darbībām;
* dati par automātiski izpildāmām darbībām;
* kopējie izmaiņu adaptācijas scenāriji;
* izstrādātāja pievienotie papildus dati;
* scenāriju zarošanās nosacījumi.

Tāpat kā jau esošās sistēmas metadatu glabāšanas mehānisms, arī evolūcijas apstrādes mehānisms izstrādāts, izmantojot *Oracle SQL* relāciju datu bāzi.

* 1. Evolūcijas apstrādes metadati un to glabāšana

Evolūcijas metadatu glabāšanai datu bāze papildināta ar vairākām tabulām, kuras ar esošajām sistēmas tabulām saistītas ar ārējām atslēgām. Lai realizētu evolūcijas metadatu glabāšanu, nepieciešamas trīs esošās sistēmas tabulas. Tabulā *Change* glabājas informācija par identificētajām izmaiņām, kā arī, apstrādājot šīs tabulas ierakstus, iespējams noteikt, kāda tipa izmaiņa ir radusies. Uz šīs tabulas ierakstu balstās viss kopējais izmaiņas adaptācijas scenārijs un scenārija izpildes procesa apraksts. Tabulā *Author* glabājas informācija par sistēmas lietotājiem. Šī tabula izmantota, lai identificētu, kurš lietotājs darbojies ar konkrētas izmaiņas adaptāciju sistēmā. Savukārt tabula *Type* sistēmā izmantota kā klasifikators, kurš nosaka dažādu sistēmā izmantoto vienumu tipus. Evolūcijas mehānisma realizācijai izveidoti vairāki jauni tipi un apakštipi, kas paredzēti dažādu statusu un datu veidu glabāšanai jaunizveidotajās tabulās.

A close up of text on a white background

Description automatically generated

*2.1. att.* Evolūcijas apstrādes metadatu glabāšanai paredzētās datu bāzes shēma

Attēlā 2.1. redzama nepieciešamā datu bāzes shēma, lai glabātu datu avotu evolūcijas apstrādes metadatus. Ar raustīto līniju apzīmētas esošās sistēmas tabulas. Izmaiņu adaptācijas scenāriju un veicamo operāciju glabāšanai paredzētas tabulas *ChangeAdaptionOperation*, *ChangeAdaptionScenario* un *ChangeAdaptationProcess*. Lai glabātu scenāriju zarošanās nosacījumus un pārvaldītu nosacījumu izpildi, paredzētas tabulas *ChangeAdaptationCondition*, *CA\_ConditionMapping* un *CA\_ManualConditionFulfillment*. Ja izmaiņu adaptācijas laikā nepieciešama kāda papildus informācija no izstrādātāja, tā tiek glabāta tabulā *ChangeAdaptationAdditionalData.* Sīkāk tabulas aprakstītas nodaļās 2.1.1., 2.1.2. un 2.1.3.

* + 1. Izmaiņu adaptācijas scenāriji un operācijas

Izmaiņu adaptācijas operācijas ir darbības jeb soļi, kas jāveic, lai adaptētu izmaiņas sistēmā. Tās ir iespējami īsas un universālas.

*2.1. tabula*

Datu bāzes tabula *ChangeAdaptationOperation*

|  |  |  |  |
| --- | --- | --- | --- |
| **Kolonnas nosaukums** | **Datu tips** | **Obligāts** | **Ierobežojums** |
| CAO\_ID | number(10,0) | Jā | Primārā atslēga |
| CAO\_OPERATIONTYPE\_ID | varchar2(10 byte) | Jā | Ārējā atslēga |
| CAO\_OPERATION | varchar2(4000 byte) | Jā | Nav |

Katra no darbībām tiek glabāta kā ieraksts tabulā *ChangeAdaptationOperation* (tabulas aprakstu skat. tabulā 2.1.). Darbībai ir tips, kas norāda uz to, vai darbība ir manuāli veicama (šādā gadījumā tabulā glabājas tekstuāls apraksts ar to, kas izstrādātājam jāveic, lai paveiktu darbību) vai automātiski izpildāma (šajā gadījumā tabulā glabājas izpildāmās procedūras nosaukums). Vairākas dažādu tipu izmaiņu adaptāciju veikšanai izmantotās darbības pārklājas, tāpēc tās tiek glabātas atsevišķā tabulā, lai novērstu informācijas dublēšanos datu bāzē.

Izmaiņu adaptācijas scenārijs ir secīgu darbību virkne, kas tiek veikta, lai veiksmīgi adaptētu izmaiņu sistēmā. Katram no izmaiņu veidiem ir atšķirīgs izmaiņu adaptācijas scenārijs.

*2.2. tabula*

Datu bāzes tabula *ChangeAdaptationScenario*

|  |  |  |  |
| --- | --- | --- | --- |
| **Kolonnas nosaukums** | **Datu tips** | **Obligāts** | **Ierobežojums** |
| CAS\_ID | number(10,0) | Jā | Primārā atslēga |
| CAS\_PARENTSCENARIO\_ID | number(10,0) | Jā | Ārējā atslēga |
| CAS\_CHANGETYPE\_ID | number(10,0) | Jā | Ārējā atslēga |
| CAS\_OPERATION\_ID | varchar2(10 byte) | Jā | Ārējā atslēga |

Katra scenārija soļi glabājas tabulā *ChangeAdaptationScenario* (tabulas aprakstu skat. tabulā 2.2.). Katrs no scenārija soļiem satur ārējo atslēgu uz adaptācijas darbību (tabulu *ChangeAdaptationOperation*), izmaiņas tipu, kā arī vecākieraksta identifikatoru – ārējo atslēgu uz šo pašu tabulu. Vecākieraksta identifikatora glabāšana nodrošina darbību secības uzturēšanu, kā arī atvieglo korekciju veikšanu (piemēram, pievienojot scenārija vidū kādu darbību, jārediģē tikai divi tabulas ieraksti, taču, ja secība tiktu glabāta kā kārtas skaitļi, jārediģē būtu gandrīz viss scenārijs).

Lai sekotu līdzi izmaiņu adaptācijas procesam un glabātu informāciju par katras darbības izpildi, nepieciešama tabula, kurā glabājas katrai reāli notikušai izmaiņai atbilstošais adaptācijas scenārijs.

*2.3. tabula*

Datu bāzes tabula *ChangeAdaptationProcess*

|  |  |  |  |
| --- | --- | --- | --- |
| **Kolonnas nosaukums** | **Datu tips** | **Obligāts** | **Ierobežojums** |
| CAP\_ID | number(10,0) | Jā | Primārā atslēga |
| CAP\_SCENARIO\_ID | number(10,0) | Jā | Ārējā atslēga |
| CAP\_DATETIME | timestamp(6) | Jā | Nav |
| CAP\_AUTHOR\_ID | number(10,0) | Jā | Ārējā atslēga |
| CAP\_STATUSTYPE\_ID | varchar2(10 byte) | Jā | Ārējā atslēga |
| CAP\_CHANGE\_ID | number(10,0) | Jā | Ārējā atslēga |

Tabula *ChangeAdaptationProcess* (tabulas aprakstu skat. tabulā 2.3.) glabā arējo atslēgu uz tabulu *ChangeAdaptationScenario*, kas nodrošina katras operācijas atšķiramību, kā arī ārējo atslēgu uz tabulu *Change,* lai identificētu, kura no izmaiņām tiek adaptēta. Lai sekotu līdzi izmaiņu adaptācijas procesam, tiek glabāts arī izpildāmās darbības statuss (darbība ir vai nav izpildīta), kā arī datums, laiks un lietotājs, kurš piedalījies šīs darbības izpildē.

* + 1. Izmaiņu adaptācijas scenāriju zarošanās nosacījumi

Neskatoties uz to, ka izmaiņas tips ir nosakāms jau pie tās rašanās, tas negarantē viennozīmīga izmaiņas adaptācijas scenārija esamību. Ir dažādi nosacījumi, kuru izpildes rezultātā scenārijs var zaroties. Tie var būt gan automātiski izpildāmi (šajā gadījumā sistēma pati nosaka, pa kuru no scenārija ceļiem nepieciešams virzīties), gan manuāli apstrādājami (šajā gadījumā izstrādātājam nepieciešams izvērtēt konkrēto situāciju un pieņemt lēmumu).

*2.4. tabula*

Datu bāzes tabula *ChangeAdaptationCondition*

|  |  |  |  |
| --- | --- | --- | --- |
| **Kolonnas nosaukums** | **Datu tips** | **Obligāts** | **Ierobežojums** |
| CAC\_ID | number(10,0) | Jā | Primārā atslēga |
| CAC\_CONDITON | varchar2(4000 byte) | Jā | Nav |
| CAC\_CONDITIONTYPE\_ID | number(10,0) | Jā | Ārējā atslēga |

Tāpat kā darbības, arī nosacījumi dažādu tipu izmaiņu adaptāciju veikšanā pārklājas, līdz ar to tie tiek glabāti atsevišķā tabulā *ChangeAdaptationCondition* (tabulas aprakstu skat. tabulā 2.4.). Katram ierakstam tiek glabāts nosacījuma tips (nosacījums ir manuāli vai automātiski izpildāms), kā arī pats nosacījums – ja tas ir manuāli izpildāms, tiek glabāts tekstuāls apraksts, kas jāizvērtē pašam izstrādātājam. Ja nosacījums ir automātiski izpildāms, tiek glabāts izpildāmās funkcijas nosaukums.

*2.5. tabula*

Datu bāzes tabula *CA\_ConditionMapping*

|  |  |  |  |
| --- | --- | --- | --- |
| **Kolonnas nosaukums** | **Datu tips** | **Obligāts** | **Ierobežojums** |
| CACM\_ID | number(10,0) | Jā | Primārā atslēga |
| CACM\_SCENARIO\_ID | number(10,0) | Jā | Ārējā atslēga |
| CACM\_CONDITION\_ID | number(10,0) | Jā | Ārējā atslēga |

Tā kā katrai no scenārija ietvaros izpildāmajām darbībām var būt vairāki nosacījumi, tabula *CA\_ConditionMapping* (tabulas aprakstu skat. tabulā 2.5.) tiek izmantota, lai nodrošinātu N:N saiti starp tabulām *ChangeAdaptationCondition* un *ChangeAdaptationScenario*.

Automātiski izpildāmos nosacījumus iespējams pārbaudīt pirms katras darbības veikšanas un tam nav nepieciešama izstrādātāja iejaukšanās, līdz ar to iespējams automātiski virzīties pa kādu no scenārijā paredzētajiem ceļiem. Tomēr, veicot manuālu nosacījuma apstrādi (izstrādātājs pieņem lēmumu par turpmāko scenārija gaitu), nepieciešams glabāt informāciju par to, kādu lēmumu ir pieņēmis izstrādātājs, lai šis lēmums tiktu ņemts vērā visā izmaiņas adaptācijas procesā.

*2.6. tabula*

Datu bāzes tabula *CA\_ManualConditionFulfillment*

|  |  |  |  |
| --- | --- | --- | --- |
| **Kolonnas nosaukums** | **Datu tips** | **Obligāts** | **Ierobežojums** |
| CAMCF\_ID | number(10,0) | Jā | Primārā atslēga |
| CAMCF\_CONDITION\_ID | number(10,0) | Jā | Ārējā atslēga |
| CAMCF\_CHANGE\_ID | number(10,0) | Jā | Ārējā atslēga |
| CAMCF\_FULFILLMENTSTATUS\_ID | varchar2(10 byte) | Jā | Ārējā atslēga |

Lēmumu glabāšanai izmantota tabula *CA\_ManualConditionFulfillment* (tabulas aprakstu skat. tabulā 2.6.), kurā tiek glabāta ārējā atslēga uz tabulu *Change*, kā arī uz nosacījumu un nosacījuma izpildes statusu (nosacījums izpildīts vai nav izpildīts).

* + 1. Izmaiņu adaptācijas procesa papildus informācija

Izmaiņu adaptācijas procesa laikā no izstrādātāja var tikt prasīti dažādi papildus dati, kas būs nepieciešami tālāko scenāriju darbību izpildei un nosacījumu pārbaudei.

*2.7. tabula*

Datu bāzes tabula *ChangeAdaptationAdditionalData*

|  |  |  |  |
| --- | --- | --- | --- |
| **Kolonnas nosaukums** | **Datu tips** | **Obligāts** | **Ierobežojums** |
| CAAD\_ID | number(10,0) | Jā | Primārā atslēga |
| CAAD\_DATA\_TYPE\_ID | varchar2(10 byte) | Jā | Ārējā atslēga |
| CAAD\_CHANGE\_ID | number(10,0) | Jā | Ārējā atslēga |
| CAAD\_DATA | clob | Jā | Nav |

Šiem papildus datiem tiek izmantota tabula *ChangeAdaptationAdditionalData* (tabulas aprakstu skat. tabulā 2.7.), kas glabā ārējo atslēgu uz tabulu *Change*, datu tipu (informācija par to, kādam nolūkam izmantojami papildus dati), kā arī pāši glabājamie dati. Glabājamo datu formāts atkarīgs no datu veida. Piemēram, glabājot tikai identifikatoru, tas būs vienkāršs skaitlis, taču iespējamas situācijas, kad dati būs kādā citā formātā (JSON, XML u.c.), tāpēc izmantots universāls datu tips CLOB, kam ir liela ietilpība.

* 1. Izmaiņu adaptācijas scenāriji

Katras izmaiņas adaptācijai sistēmā atbilst scenārijs jeb veicamo darbību kopums, kas nepieciešams izmaiņas veiksmīgai apstrādei. Darba ietvaros netiek apskatīti visu iespējamo izmaiņu apstrādes scenāriji. Katra izstrādātā scenārija mērķis ir panākt, lai tas būtu pēc iespējas automātiski izpildāms (lai būtu nepieciešama pēc iespējas mazāka izstrādātāja iejaukšanās izmaiņu adaptācijas procesā), tomēr konceptuālu lēmumu pieņemšana par datu izmantošanu sistēmā nav iespējama bez cilvēka iejaukšanās.

Turpmākajās apakšnodaļās detalizēti aprakstīti 7 izmaiņu adaptācijas scenāriji.

* + 1. Datu maģistrāles līmeņa pievienošana

Lai pievienotu jaunu datu maģistrāles līmeni, izstrādātājam nepieciešams sagatavot gan jaunā datu maģistrāles līmeņa struktūras aprakstu, gan identificēt nepieciešamos ELT procesus, kā arī nepieciešamības gadījumā sagatavot datu kopu piemērus.

A close up of a device

Description automatically generated

*2.2.att*. Datu maģistrāles līmeņa pievienošanas peldceliņu diagramma

Attēlā 2.2. iespējams apskatīt pilnu datu maģistrāles līmeņa pievienošanas scenāriju. Kā redzams, lielākā daļa darbību ir veicamas manuāli no izstrādātāja puses, taču sistēma automātiski spēj iegūt datu kopas struktūru un pievienot to izveidotajam datu maģistrāles līmenim.

* + 1. Datu avota pievienošana

Lai pievienotu jaunu datu avotu, izstrādātājam nepieciešams sagatavot datu kopu piemērus, lai būtu iespējams iegūt to struktūru, kā arī definēt datu maģistrāles līmeņus, kur tiks glabāti transformētie dati.

A screenshot of a cell phone

Description automatically generated

*2.3.att.* Datu avota pievienošanas peldceliņu diagramma

Kā redzams attēlā 2.3., vispirms izstrādātājs manuāli pievieno datu avota datu kopas piemērus un pēc tam uzreiz iespējams definēt citus datu maģistrāles līmeņus un ELT procesus. Datu kopas struktūru no pievienotajiem datu kopas piemēriem iespējams iegūt automātiski, kā rezultātā arī datu kopas pievienošana pirmajam datu maģistrāles līmenim ir automātiska.

* + 1. Datu kopas pievienošana

Datu kopas pievienošanas scenārijs lielā mērā ir atkarīgs no tā, vai izstrādātājs datu kopu vēlas pievienot datu maģistrāles līmenim vai esošam datu avotam.

A close up of a map

Description automatically generated

*2.4. att.* Datu kopas pievienošanas peldceliņu diagramma

Pievienojot jaunu datu kopu (skat. att. 2.4.), jau sākumā tiek veikta pārbaude, kādai datu struktūrai jauno datu kopu nepieciešams pievienot. Ja datu kopa jāpievieno esošam datu avotam, tad adaptācijas process ir gandrīz pilnībā automātisks – izstrādātājam nepieciešams tikai pievienot datu kopu piemērus struktūras noskaidrošanai.

* + 1. Metadatu īpašības pievienošana

Metadatu īpašības pievienošana ir pilnībā manuāls process, tāpēc vienīgā scenārija darbība tiek norādīta kā informācija izstrādātājam, ka viņam nepieciešams izlemt par tālāko pievienotās īpašības izmantošanu.

* + 1. Datu vienības pievienošana

Datu vienības pievienošanas scenārijs pilnībā atkarīgs no tā, vai izstrādātājs datu vienību vēlas pievienot esošai datu kopai, kas atrodas kādā datu maģistrāles līmenī (izņemot pirmo datu maģistrāles līmeni) vai datu avota datu kopai.

*A close up of a map

Description automatically generated*

*2.5. att.* Datu vienības pievienošanas peldceliņu diagramma

Pievienojot jaunu datu kopu (skat. att.2.5.), jau sākumā tiek veikta pārbaude, kādai datu struktūrai jauno datu kopu nepieciešams pievienot. Ja datu vienība jāpievieno esošai datu kopai kādā datu maģistrāles līmenī (izņemot pirmo datu maģistrāles līmeni), izstrādātājam nepieciešams pievienot papildus informāciju un izdarīt lēmumus par pievienojamo datu vienību un ar to saistīto struktūru. Citādi process ir gandrīz automātisks – nepieciešams tikai pievienot datu vienības piemērus. Ar raustīto līniju apzīmēts datu avota pievienošanas scenārijs (skat. nodaļu 2.2.2.).

* + 1. Datu avota dzēšana

Lai dzēstu datu avotu, izstrādātājam nepieciešams izvērtēt, vai dzēšamā avota datu kopas datu vienības iespējams aizvietot ar datiem no citiem avotiem.

A close up of text on a white background

Description automatically generated

*2.6. att*. Datu avota dzēšanas peldceliņu diagramma

Datu avota dzēšanas procesā (skat. att. 2.6.) ietilpst arī datu avota pievienošana (attēlā atzīmēta ar raustīto līniju, skat. nodaļu 2.2.2.), ja ir iespējama datu avota aizvietošana ar datiem no citiem avotiem un nepieciešams jauns datu avots. Norādot alternatīvās datu vienības no citiem datu avotiem, tiek izveidots ieraksts tabulā *ChangeAdaptationAdditionalData*.

* + 1. Datu maģistrāles līmeņa dzēšana

Maģistrāles līmeņa dzēšana ir pavisam vienkāršs process, ja datu maģistrāles līmenis nav saistīts ar nevienu datu kopu. Šādā gadījumā nekādi adaptācijas procesi nav nepieciešami. Tomēr šādi gadījumi ir salīdzinoši reti un funkcionējošam datu maģistrāles līmenim jau ir piesaistītas kādas datu kopas.

A close up of a map

Description automatically generated

*2.7. att.* Datu maģistrāles līmeņa dzēšanas peldceliņu diagramma

Datu maģistrāles līmeņa dzēšanas procesā (skat. att. 2.7.) iesaistīti arī automātiskie nosacījumi, kuru pārbaudei nav nepieciešama izstrādātāja iejaukšanās. Ar raustīto līniju apzīmēta datu avota dzēšana (skat. nodaļu 2.2.6.).

* 1. Evolūcijas apstrādes funkcionalitāte

Kad izmaiņu apstrādes scenāriji saglabāti tabulās *ChangeAdaptationOperation, ChangeAdaptationScenario, ChangeAdaptationCondition* un *CA\_ConditionMapping*, iespējams sākt izmaiņu apstrādi. Lai izmaiņu veiksmīgi adaptētu sistēmā, izstrādāta funkcionalitāte, kas, savācot tabulā *Change* saglabātos izmaiņas metadatus un izmantojot noteiktos apstrādes scenārijus, aizpilda pārējās evolūcijas apstrādei paredzētās tabulas un pēc iespējas mēģina pielietot automātiskos nosacījumus un darbības.

Tiek izdalīti divi evolūcijas apstrādes posmi:

* pimreizējā izmaiņas apstrāde – nosaka izmaiņas adaptācijas scenāriju un izveido sākotnējos ierakstus tabulās *ChangeAdaptationProcess* un *CA\_ManualConditionFulfillment*;
* scenārija izpilde – pēc iespējas tiek izpildīti automātiskie nosacījumi un operācijas, kā arī tabulā *ManualConditionFulfillment* saglabātas izstrādātāja manuālās izvēles un tabulā *ChangeAdaptationAdditionalData* saglabāti papildus dati, ja tādi nepieciešami.

Katrs no minētajiem evolūcijas apstrādes posmiem detalizēti aprakstīts nodaļās 2.3.1., 2.3.2. un 2.3.3.

* + 1. Pirmreizējā izmaiņas apstrāde

Pirmreizējā izmaiņas apstrādē tiek noteikts izmaiņas tips un atrasts izmaiņas tipam atbilstošais adaptācijas scenārijs. Katrai no scenārijā izpildāmajām darbībām tiek izveidots ieraksts tabulā *ChangeAdaptationProcess* ar statusu “Nav adaptēts”. Tiek saglabāts arī laiks, kad procesa solis izveidots, kā arī procesa soļa autors (šajā gadījumā tabulā *Author* jābūt definētam lietotājam “Sistēma”).

A close up of a map

Description automatically generated

*2.8.att.* Pirmreizējās izmaiņas apstrādes aktivitāšu diagramma

Veicot pirmreizējo izmaiņas apstrādi (skat. att. 2.9.), tiek apstrādātas tikai izmaiņas ar statusu “Jauns”. Nosakot izmaiņas tipu (izmaiņu tipus skatīt nodaļā 2.2.), pareizajā secībā tiek atlasīti visi izmaiņas tipam atbilstošie scenārija soļi, pēc kā izveidoti konkrētajai izmaiņai atbilstošie procesa soļi. Pēc tam tiek atlasīti konkrētajajam procesa solim atbilstošie scenārija soļu manuālie nosacījumi un saglabāti ar statusu “Nav izpildīts”. Līdz ar konkrētās izmaiņas procesu izveidošanu, izmaiņas statuss tiek nomainīts uz “Procesā”, līdz ar to šīs izmaiņas adaptāciju var uzskatīt par sākušos.

![A screenshot of a cell phone

Description automatically generated]()

*2.9. att.* Pirmreizējās izmaiņas apstrādes programmkoda fragments

Attēlā 2.9. redzams programmkoda fragments algoritmam, kas veic pirmreizējo izmaiņu apstrādi. Funkcija *Define\_change\_type*, balstoties uz ierakstu tabulā *Change,* nosaka izmaiņas tipu (to iespējams noteikt, pārbaudot, kuri no tabulas *Change* laukiem ir aizpildīti). Katram no procesa soļiem tiek izsaukta funkcija *Insert\_change\_adaptation\_process*, kura ievieto atbilstošo ierakstu tabulā *ChangeAdaptationProcess*. Funkcijā *Insert\_manual\_condition\_fulfillment* tiek atlasīti scenārija solim atbilstošie manuālās izpildes nosacījumi un saglabāti tabulā *CA\_ManualConditionFulfillment*. Funkcija *Update\_change\_in\_progress* uzstāda izmaiņas statusu “Procesā”*­*.

* + 1. Scenārija iegūšana

Lai iegūtu konkrētajai izmaiņai atbilstošu scenāriju, jāņem vērā gan izpildāmās darbības, gan nosacījumi, kas attiecas uz katru no tām.

![A screenshot of text

Description automatically generated]()

*2.10.att.* Izmaiņai atbilstoša adaptācijas scenārija datu atlases vaicājums

Attēlā 2.10. redzams datu atlases vaicājums, kas paredzēts secīga konkrētās izmaiņas scenārija atlasei. Ieejas parametrs ir izmaiņas identifikators (attēlā 2.10. – *in\_change\_id*) Vaicājumā izmantota *Oracle* iebūvētā funkcija *Listagg*, kas ļauj vairāku atsevišķu ierakstu datus sarindot vienā kolonnā, atdalot tos ar izvēlēto atdalītāju. Šādā veidā tiek nodrošināta vieglāk lasāma informācija par katrai operācijai atbilstošajiem nosacījumiem.

* + 1. Scenārija izpilde

Scenārija izpilde ir balstīta uz nosacījumu pārbaudēm un uz darbību izpildes. Ja scenārija izpildei nepieciešama izstrādātāja iejaukšanās, algoritms tiek pārtraukts un turpināts tikai tad, kad izstrādātājs ir izdarījis savu izvēli par manuālajiem nosacījumiem vai izpildījis noteikto darbību.

![A screenshot of a social media post

Description automatically generated]()

*2.11. att.* Scenārija izpildes programmkoda fragments

Attēlā 2.11. redzama datu bāzes procedūra *Run\_change\_adaptation\_scenario*, kas, ieejas parametrā saņemot izmaiņas identifikatoru, apstrādā izmaiņu. Vispirms ar funkciju *Get\_change\_adaptation\_scenario* tiek iegūts kursors ar pašu scenāriju. Šajā funkcijā tiek izmantots nodaļā 2.3.2. aprakstītais datu atlases vaicājums. Tālāk adaptācijas process tiek turpināts tikai tad, ja nepieciešams izpildīt automātisko darbību, kā arī atbilstošajiem nosacījumiem jābūt izpildītiem. Manuālie nosacījumi tiek pārbaudīti, izmantojot tabulu *CA\_ManualConditionFulfillment*. Automātisko nosacījumu pārbaudei tiek izmantots *Dynamic SQL*, kas ļauj no tabulas iegūto procedūras nosaukumu izpildīt kā procedūru, iekļaujot to “*execute immediate”* blokā. Pēc šī paša principa tiek izsauktas arī procedūras izmaiņu adaptācijas procesa soļu izpildei.

REZULTĀTI

SECINĀJUMI

IZMANTOTĀ LITERATŪRA UN AVOTI

PIELIKUMI

Bakalaura darbs „Neviendabīgu integrētu datu avotu evolūcijas apstrāde” izstrādāts LU Datorikas fakultātē.

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

Autors: Lauma Svilpe \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Rekomendēju/nerekomendēju darbu aizstāvēšanai *(nederīgo svītro vadītājs)*

Vadītājs: asociētā profesore Dr.dat. Darja Solodovņikova \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_.06.2020.

Recenzents: <amats> <grāds> <vārds uzvārds>

Darbs iesniegts Datorikas fakultātē \_\_.06.2020.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

\_\_\_\_.06.2020. prot. Nr. \_\_\_\_\_

Komisijas sekretārs(-e): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_