

Datorzinātnes un Informācijas tehnoloģijas fakultāte

**Mācību kurss**

##### Lielas datu bāzes

**2. praktiskais darbs**

**“Temporālas datu bāzes”**

Izstrādāja: Aleksandrs Koroļko

Grupas numurs: 2

Apliecības numurs: 171RDB270

2020./21. māc. gads

#### SATURS

[UZDEVUMS 4](#_Toc58450270)

[1. Temporālo datu datubāzes tabulas definēšana 5](#_Toc58450271)

[2. Temporālie vaicājumi 7](#_Toc58450272)

[2.1 Pirmais vaicājums 7](#_Toc58450273)

[2.2 Otrais vaicājums 7](#_Toc58450274)

[2.3 Trešais vaicājums 8](#_Toc58450275)

[2.4 Ceturtais vaicājums 9](#_Toc58450276)

[2.5 Piektais vaicājums 9](#_Toc58450277)

[2.6 Sestais vaicājums 10](#_Toc58450278)

[3. Tabulas izveidošana un datu ievade 11](#_Toc58450279)

[3.1 Objektu tipu un tabulas izveidošana 11](#_Toc58450280)

[3.2 Datu ievade INSERT 14](#_Toc58450281)

[3.3 Datu ievade SQL\*LOADER 16](#_Toc58450282)

[4. Objektu skati 19](#_Toc58450283)

[4.1 Java klase un metodes deklarēšana PL/SQL vidē 19](#_Toc58450284)

[4.2 Objektu skatu izveide 21](#_Toc58450285)

[5. Temporālo vaicājumu izpildīšana un pārbaudīšana 24](#_Toc58450286)

[5.1 Pirmais vaicājums 24](#_Toc58450287)

[5.2 Otrais vaicājums 25](#_Toc58450288)

[5.3 Trešais vaicājums 26](#_Toc58450289)

[5.4 Ceturtais vaicājums 27](#_Toc58450290)

[5.5 Piektais vaicājums 29](#_Toc58450291)

[5.6 Sestais vaicājums 32](#_Toc58450292)

[6. WINDOWING klona tabula un vaicājumi 34](#_Toc58450293)

[7. Secinājumi 36](#_Toc58450294)

[PIELIKUMI 38](#_Toc58450295)

[1. Pielikums 38](#_Toc58450296)

[2. Pielikums 39](#_Toc58450297)

[3. Pielikums 41](#_Toc58450298)

[4. Pielikums 42](#_Toc58450299)

[5. Pielikums 45](#_Toc58450300)

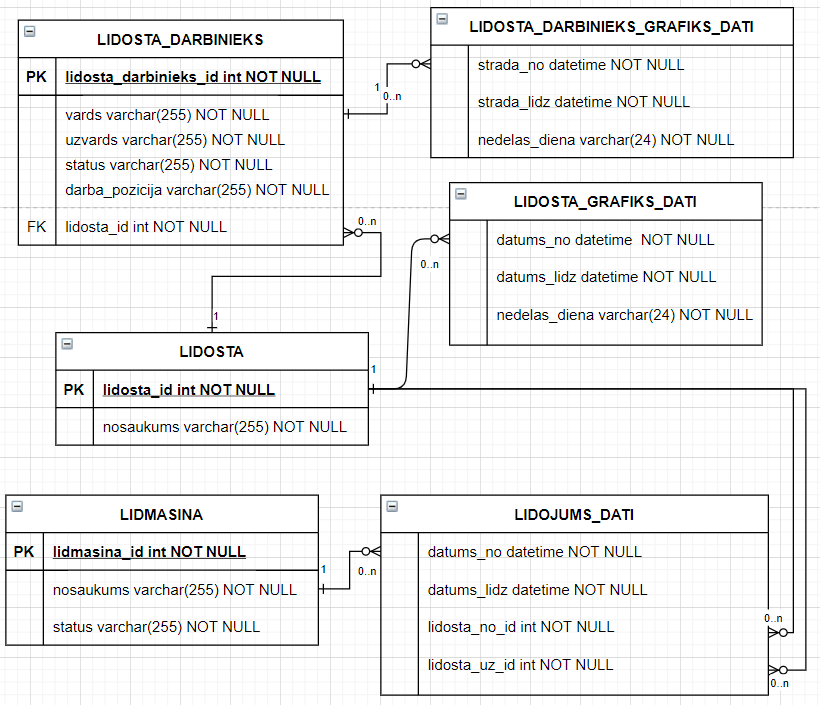
# UZDEVUMS

**2. praktiskā darba uzdevums:**  
1. Tiek definētas vismaz 3 **savstarpēji saistītas** laika sērijas.  
2. Tiek definēti 6 dažādas sarežģītības pakāpes datu izgūšanas **temporālie** vaicājumi, kuru realizēšanai ir nepieciešams izmantot temporālās algebras darbības . Atgādinājums: Temporālie vaicājumi operē ar laika **periodiem** un intervāliem. Temporālo vaicājumu definēšanā lietderīgi izmantot laika diagrammu grafiskos attēlojumus.  
3. Temporālajiem vaicājumiem tiek noskaidrots kādas temporālās algebras darbības jārealizē to izpildei (tiek izmantotas vismaz 3 tipiskākās temporālās algebras darbības).  
4. Temporālo datu glabāšanai tiek izmantotas **relāciju-objektu datu glabāšanas struktūras**: tabula ar kolekciju, tabulas ar atsaucēm (references).  
5. Tiek veikta datu ievade relāciju – objektu DB struktūrās (INSERT, SQL\*Loader).  
6. Tiek izveidotas relāciju – objektu DB virtuālās struktūras (objektu skati), no kurām tiks izgūti dati. Veidojot objektu tipus objektu skatiem, tiek definētas arī **metodes** (ORDER, MEMBER) temporālās algebras darbību izpildīšanai. **Metožu definēšanai tiek izmantotas PL/SQL un Java** programmēšanas valoda.  
7. Tiek izpildīti temporālie vaicājumi izmantojot objektu skatus un definētās metodes temporālās algebras darbību realizēšanai. Rezultāti tiek izanalizēti vai tie ir korekti.  
8. Temporālo vaicājumu realizācijai tiek izmantota arī klona tabulu (windowing) tehnoloģija.  
9. Tiek izdarīti secinājumi, kā vienkāršāk un efektīvāk SQL valodā pierakstīt temporālos vaicājumus.  
10. Kopējie secinājumi.

# Temporālo datu datubāzes tabulas definēšana

Šajā darbā, ka problemvīde bija izvēlēta vienkāršota lidostas grafika sistēma, kurā tiek definētas sekojošas relācijas un tabulas, kurus ir iespējams apskatīt attēlā (att. 1.1):

1. Lidosta tabula (LIDOSTA):
   * lidosta\_id – lidostas identifikators
   * nosaukums – lidostas nosaukums
   * grafiki – lidostas grafiki (nested table)
2. Lidostas grafika datu tabula (LIDOSTA\_GRAFIKS\_DATI):
   * datums\_no – grafika posma sākuma datums un laiks
   * datums\_līdz – grafika posma beigas datums un laiks
   * nedelas\_diena – grafika nedēļas dienas nosaukums
3. Lidmašīnas tabula (LIDMASINA):
   * lidmasina\_id – lidmašīnas identifikators
   * nosaukums – lidmašīnas nosaukums
   * status – lidmašīnas tiekošais status
   * lidojumi – lidmašīnas lidojumi (nested table)
4. Lidmašīnas lidojuma datu tabula (LIDOJUMS\_DATI):
   * datums\_no – lidosta posma sākuma datums un laiks
   * datums\_lidz – lidosta posma sākuma beigas datums un laiks
   * lidosta\_no\_id – lidostas identifikators no kurienes tiek veikts izlidojums
   * lidosta\_uz\_id – lidostas identifikators uz kurienes tiek veikts izlidojums
5. Lidostas darbinieks (LIDOSTA\_DARBINIEKS):
   * lidosta\_darbinieks\_id – lidostas darbinieka identifikators
   * vards – darbinieka vārds
   * uzvards – darbinieka uzvārds
   * status – darbinieka tiekošais status
   * darba\_pozicija – darbinieka darba pozīcija
   * lidosta\_id - (FK) darbinieka lidostas identifikators
   * grafiki – darbinieka grafiki (nested table)
6. Lidostas darbinieka grafika datu tabula (LIDOSTA\_DARBINIEKS\_GRAFIKS\_DATI):
   * strada\_no – darbinieka darba sākums datums un laiks
   * strada\_lidz - darbinieka darba beigas datums un laiks
   * nedelas\_diena – grafika nedēļas dienas nosaukums



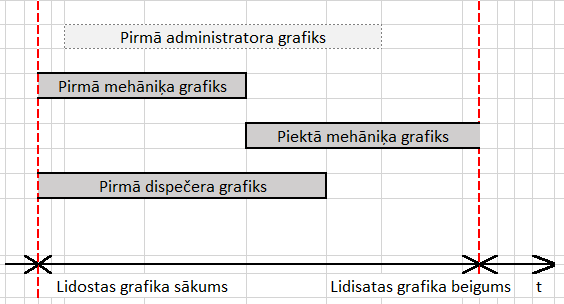
att. 1.1 **Datubāzes relāciju shēma**

# Temporālie vaicājumi

Šajā nodaļā tiek nodefinētas, aprakstītas un attēlotas 6 dažādas sarežģītības pakāpes datu izgūšanas temporālie vaicājumi, kuru realizēšanai noskaidrotas temporālās algebras darbības.

## 2.1 Pirmais vaicājums

Pirmā vaicājumā būtu nepieciešams nodefinēt lidostas darbiniekus, kurās darba grafiks vienlaicīgi uzsāks ar lidostas grafiku vai darba grafiks vienlaicīgi beidzas ar lidostas grafiku. Ar šo vaicājumu ir iespējams nodefinēt darbiniekus, kuri bija atbildīgi lidostas atvēršanai un nepieciešamas darbības uzsākšanai darba dienas sākumā, kā arī, kuri bija atbildīgi lidostas slēgšanai un nepieciešamas darbības beigšanai. Vaicājumu ir iespējams redzēt attēlā (att. 2.1), kurā ir redzams, kā pirmais mehāniķis un dispečeris bija atbildīgi lidostas atvēršanai, jo tiem sakrīt darba grafiks sākums ar lidostas grafika sākumu, un piektais mehāniķis bija atbildīgs lidostas slēgšanai, jo sakrīt darba grafika beigums ar lidostas grafika beigumu.

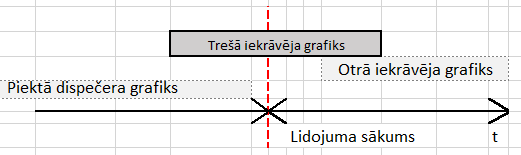


att. 2.1 Pirmais temporālais vaicājums

Lai realizēt vaicājumu, būtu nepieciešams izmantot Alenna temporālās algebras STARTS, kā arī FINISHES laika periodu attiecības, kur arī loģiskais operators būs VAI, un izmantojot jo vaicājuma ir nepieciešams atrast darbiniekus, kurās darba grafiks vienlaicīgi uzsāks ar lidostas grafiku vai darba grafiks vienlaicīgi beidzas ar lidostas grafiku.

## 2.2 Otrais vaicājums

Otrā vaicājumā būtu nepieciešams nodefinēt lidostas darbiniekus, kuriem darba grafiks daļēji nosegt lidmašīnas lidojumu, kas nozīme, ka darbinieki sniedza kādus pakalpojums pirms lidojuma sākuma, kas nozīme, ka darbinieki ir atbildīgi par lidojuma uzticamību un lidmašīnas sagatavošanu pirms lidojuma. Vaicājumu ir iespējams redzēt attēlā (att. 2.2), kurā ir redzams, ka atbildīgs par bagāžu iekraušanu, priekš kaut kāda lidojuma, ir trešais iekrāvējs, jo darbinieka darba grafiks daļēji nosegt lidmašīnas lidojuma grafiku.

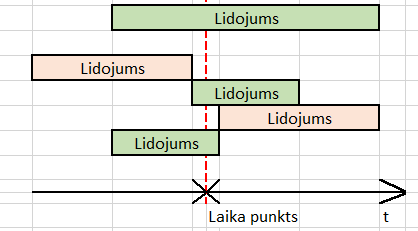


Att. 2.2 Otrais temporālais vaicājums

Lai realizēt vaicājumu, būtu nepieciešams izmantot Alenna temporālās algebras OVERLAPS laika periodu attiecību, kā arī, ir nepieciešams izvadīt tikai tādus datus, kur darbinieka lidosta sakrīt ar lidostu no kurienes tiek veikts izlidojums, jo darbinieks var sniegt savus pakalpojumus tikai savā lidostā.

## 2.3 Trešais vaicājums

Trešā vaicājumā, būtu nepieciešams atrast lidojumus kādā noteiktā laika punktā, lai apskatīt visus lidojumus sistēmā noteiktā dienā un laikā. Vaicājumu ir iespējams redzēt attēlā (att. 2.3), kurā ir redzams, ka lidojumi ar zaļo fonu ir nepieciešami lidojumi noteiktā laika punktā.

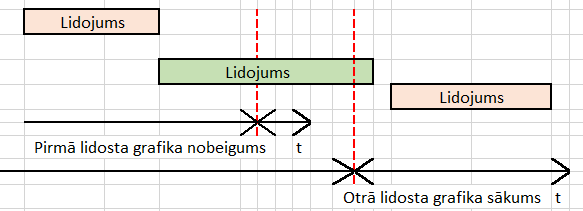


Att. 2.3 Trešais temporālais vaicājums

Lai realizēt vaicājumu, būtu nepieciešams izmantot temporālās algebras darbību ATLASE, ar kuru būtu iespējams definēt lidojumus noteiktā dienā un laikā.

## 2.4 Ceturtais vaicājums

Ceturtā vaicājumā, būtu nepieciešams nodefinēt tādas lidmašīnas, kuriem izlidojums no pirmā lidosta bija pirms lidostas grafika nobeigumā, bet atlidošana otrā lidostā tiek veikta tās jauna grafika sākumā. Ar sekojošo vaicājumu, būtu iespējams atrast lidojumus, kurās laikā ir laiks logs, kurā lidmašīna lido bez pirmā un otrā lidostas radistu atbalstu, kas ir bīstami un varētu noved pie lidmašīnas lidojuma kursa dezorientēšanu. Vaicājumu ir iespējams redzēt attēlā (att. 2.4), kurā ir redzams, ka lidojums ar zaļo fonu ir aprakstītas problēmas lidojums, jo ir laika brīdis, starp pirmā lidosta grafika nobeiguma un otrā lidosta grafika sākuma, kad lidmašīna lido bez lidostas atbalsta.

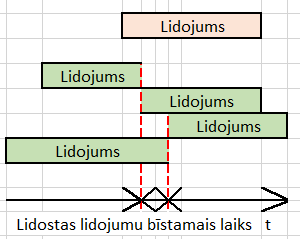


Att. 2.4 Ceturtais temporālais vaicājums

Lai realizēt tāda veida vaicājumu, būtu nepieciešams izmantot Alenna temporālās algebras OVERLAPED BY un OVERLAPS laika periodu attiecības, kā arī, ir nepieciešams izmantot PROCEDES starp pirmā lidosta grafika un otrā lidosta, lai ignorēt tādus gadījumus, kad grafiki savā starpā saskaries vai daļēju nosegt.

## 2.5 Piektais vaicājums

Piektā vaicājumā, būtu nepieciešams nodefinēt lidmašīnas, kā arī lidmašīnas reģistrēto lidostu, kuriem varētu būt izlidojuma vai atlidošanas laikā sastrēgums, piemērām, kad lidmašīnas izlidojuma laiks sakrīt ar kādu citu lidmašīnas ielidojuma laiku vienā lidostā, kas varētu pievest pie sastrēgumiem vai, sliktākā gadījumā, avārijām. Vaicājumu ir iespējams redzēt attēlā (att. 2.5), kurā ir redzams, ka lidojumi ar zaļo fonu - tas ir lidostas bīstamie lidojumi un attēlā ir paradīts tikai situācija, kad lidmašīnas izlidojuma laiks sakrīt ar kādu citu lidmašīnas ielidojuma laiku vienā lidostā, bet varētu būt arī tādas situācijas, kad lidmašīnas izlidojuma laiks sakrīt ar kādu citu lidmašīnas izlidojuma laiku vienā lidostā, vai arī lidmašīnas ielidojuma laiks sakrīt ar kādu citu lidmašīnas ielidojuma laiku vienā lidostā.

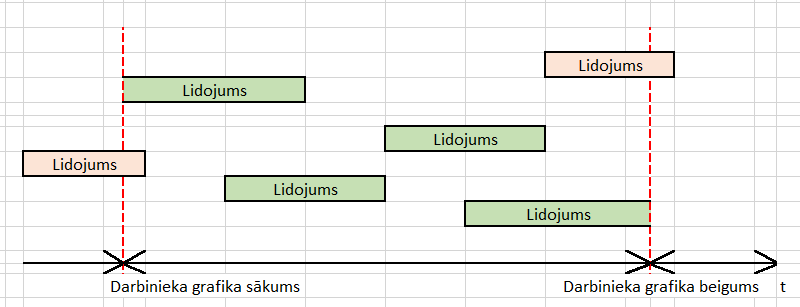


Att. 2.5 Piektais temporālais vaicājums

Lai realizēt tāda veida vaicājumu, būtu nepieciešams izmantot Alenna temporālās algebras MEETS vai MET BY vai STARTS vai STARTED BY vai FINISHES vai FINISHED BY laika periodu attiecības, jo ir nepieciešams pārbaudīt visas iespējamas lidmašīnas lidojumu saskares variācijas.

## 2.6 Sestais vaicājums

Sestajā vaicājumā, būtu nepieciešams atrast tādus dispečerus, kuriem darbā laikā bija pilnīgi paveikti lidmašīnas lidojumi, tas nozīme, ka dispečeris kontrolēja savā darbā laikā visu lidojumu no tās sākuma līdz tās beigām. Vaicājumu ir iespējams redzēt attēlā (att. 2.6), kurā ir redzams, ka lidojumi ar zaļo fonu - tas ir lidojumi kurus pilnīgi kontrolēja dispečeris.



Att. 2.6 Sestais temporālais vaicājums

Lai realizēt tāda veida vaicājumu, būtu nepieciešams izmantot Alenna temporālās algebras CONTAINS, kā arī STARTED BY vai FINISHED BY, lai atrast visus lidmašīnas lidojumus, kurus pilnīgi kontrolēja dispečeris savā darbā grafika laikā.

# Tabulas izveidošana un datu ievade

Šajā nodaļā tiek paradīts, kā bija veikta tabulas ar objektu tipu izveidošana, savukārt relāciju definēšana starp temporālo datu kolekcijām un tabulām, un datu ievade izmantojot INSERT un SQL\*LOADER.

## Objektu tipu un tabulas izveidošana

Lai izveidot visus nepieciešamus savienojumus starp tabulām un kolekcijās tipa datiem, kā arī lai labāk saprast kā to ir iespējams veikt izmantojot objektu tipus un vienkāršas tabulas bija izveidotas vairākas relāciju starp tabulām savienojumiem. Priekš temporāliem datiem bija izmantota “NESTED TABLE” un atsauces “REF” relācijas veidi.

Priekš lidosta un lidostas grafikiem bija izmantotas tikai objektu tipi, kurus ir iespējams redzēt sekojošas vaicājumos:

* Lidosta grafika datu objekta tipa izveidošana:

/\* LIDOSTAS GRAFIKA DATI TIPS \*/  
create or replace type LIDOSTA\_GRAFIKS\_DATI\_TYPE as object (  
 datums\_no TIMESTAMP,  
 datums\_lidz TIMESTAMP,  
 nedelas\_diena varchar2(24)  
);  
/

* Lidosta grafika datu objektu kolekcijas tipa izveidošana:

/\* TABULA LIDOSTAS GRAFIKA DATI TIPS \*/  
create or replace type TABLE\_LIDOSTA\_GRAFIKS\_DATI\_TYPE as table of LIDOSTA\_GRAFIKS\_DATI\_TYPE;

* Lidosta objekta tipa izveidošana:

/\* LIDOSTA TIPS \*/  
create or replace type LIDOSTA\_TYPE as object (  
 lidosta\_id number,  
 nosaukums varchar2(255),  
 grafiki TABLE\_LIDOSTA\_GRAFIKS\_DATI\_TYPE  
);  
/

* Savienošana starp temporāliem datiem (lidostas grafiki), izmantojot NESTED TABLE un tabulu (LIDOSTA) izveidošana, kā arī PRIMARY KEY definēšanā:

/\* TABULA LIDOSTA ar kollekciju grafiki \*/  
create table LIDOSTA of LIDOSTA\_TYPE nested table grafiki store as LIDOSTA\_GRAFIKS\_DATI;  
alter table LIDOSTA add primary key (lidosta\_id);

Lai izveidot relāciju starp lidostas darbiniekiem un viņas grafikiem (temporālie dati) bija veikta relācija izveidojot uzreiz priekš darbiniekiem tabulu, jo ir nepieciešams uzreiz noradīt darbinieka lidostu izmantojot FOREIGN KEY, un priekš darbinieku grafikiem tās objektu tipu ar kolekcijas tipu un pēc tām viss bija savienots izmantojot NESTED TABLE:

* Lidostas darbinieka grafika objekta datu tipa izveidošana

/\* LIDOSTAS DARBINIEKS GRAFIKA DATI TIPS \*/  
create or replace type LIDOSTA\_DARBINIEKS\_GRAFIKS\_DATI\_TYPE as object (  
 strada\_no TIMESTAMP,  
 strada\_lidz TIMESTAMP,  
 nedelas\_diena varchar2(24)  
);  
/

* Lidostas darbinieka grafika objektu kolekcijas tipa izveidošana

/\* TABULA LIDOSTAS DARBINIEKS GRAFIKA DATI TIPS \*/  
create or replace type TABLE\_LIDOSTA\_DARBINIEKS\_GRAFIKS\_DATI\_TYPE as table of LIDOSTA\_DARBINIEKS\_GRAFIKS\_DATI\_TYPE;

* Lidostas darbinieka tabula izveidošana un ar temporālas datiem relāciju definēšana

/\* TABULA LIDOSTA\_DARBINIEKS ar kollekciju grafiki \*/  
create table LIDOSTA\_DARBINIEKS (  
 lidosta\_darbinieks\_id number PRIMARY KEY,  
 vards varchar2(255),  
 uzvards varchar2(255),  
 status varchar2(255),  
 darba\_pozicija varchar2(255),  
 lidosta\_id number,  
 CONSTRAINT FK\_PersonOrder FOREIGN KEY (lidosta\_id)  
 REFERENCES LIDOSTA(lidosta\_id),  
 grafiki TABLE\_LIDOSTA\_DARBINIEKS\_GRAFIKS\_DATI\_TYPE  
) nested table grafiki store as LIDOSTA\_DARBINIEKS\_GRAFIKS\_DATI;

Un priekš lidojumiem un lidmašīnām bija nepieciešams veikt vairākas relāciju veidus, jo lidojumos ir nepieciešams definēt lidostas – uz kurienes tiek bija veikts lidojums, kā arī no kurienes, kas bija izveidots izmantojot atsauces “REF” uz lidostas objektiem, kā arī savienojums starp lidojumus (temporāli dati) ar lidmašīnu bija veikta izmantojot NESTED TABLE:

* Lidojuma objekta tipa izveidošana, kurā ir definētas arī atribūti, kuros tiek glabāts atsauce uz lidostām

/\* LIDOJUMS DATI TIPS \*/  
create or replace type LIDOJUMS\_DATI\_TYPE as object (  
 datums\_no TIMESTAMP,  
 datums\_lidz TIMESTAMP,  
 lidosta\_no REF LIDOSTA\_TYPE,  
 lidosta\_uz REF LIDOSTA\_TYPE  
);  
/

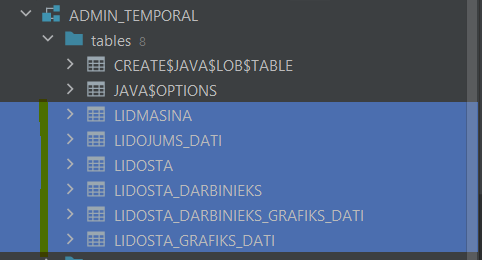
* Lidojuma objektu kolekcijas tipa izveidošanā

/\* TABULA LIDOJUMS DATI TIPS \*/  
create or replace type TABLE\_LIDOJUMS\_DATI\_TYPE as table of LIDOJUMS\_DATI\_TYPE;

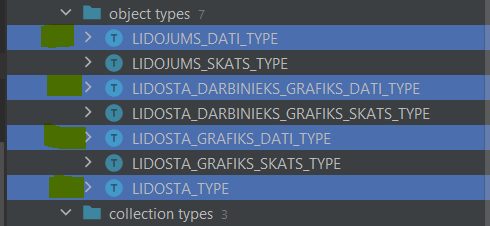
* Lidmašīnas tabula izveidošana un ar temporālas datiem relāciju definēšana

/\* TABULA LIDMASINA ar kollekciju lidojumi \*/  
create table LIDMASINA (  
 lidmasina\_id number PRIMARY KEY,  
 nosaukums varchar2(255),  
 status varchar2(255),  
 lidojumi TABLE\_LIDOJUMS\_DATI\_TYPE  
) nested table lidojumi store as LIDOJUMS\_DATI;

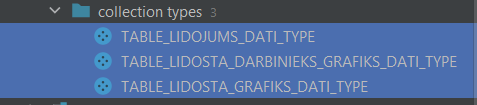
Rezultātā bija izveidotas 6 tabulas (att. 3.1), 4 objektu tipi (att. 3.2) un 3 kolekcijas (att. 3.3).



Att. 3.1 Izveidotas tabulas



Att. 3.2 Izveidoti objekti



Att. 3.3 Izveidotas objektu kolekcijas

## Datu ievade INSERT

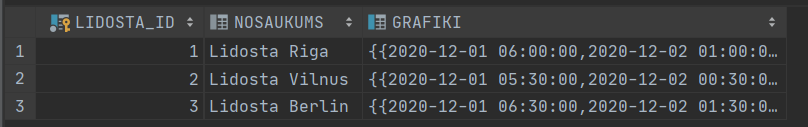
Lai ievadīt datus izmantojot INSERT bija nepieciešams izmantot vienkāršo SQL INSERT, kā arī INSERT PL/SQL jomā, priekš atsauces definēšanas. Arī no tā kā priekš laika datiem bija izmantos datu tips TIMESTAMP visus laika dati bija formatēti izmantojot TIMESTAMP formātu.

Priekš “LIDOSTA” un tās temporāliem datiem bija izmantots vienkāršs INSERT, kuru piemēru ir iespējams redzēt sekojošā vaicājumā (visu INSERT vaicājumu ir iespējams redzēt 1. pielikumā):

insert into LIDOSTA VALUES (1, 'Lidosta Riga', TABLE\_LIDOSTA\_GRAFIKS\_DATI\_TYPE (  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-01 6:00:00', TIMESTAMP '2020-12-02 1:00:00', 'otrdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-02 6:00:00', TIMESTAMP '2020-12-03 1:00:00', 'tresdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-03 6:00:00', TIMESTAMP '2020-12-04 1:00:00', 'ceturtdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-04 6:00:00', TIMESTAMP '2020-12-05 1:00:00', 'piektdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-05 6:30:00', TIMESTAMP '2020-12-06 0:30:00', 'sestdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-06 6:30:00', TIMESTAMP '2020-12-07 0:30:00', 'svetdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-07 6:00:00', TIMESTAMP '2020-12-08 1:00:00', 'pirmdiena')  
 ));

Vaicājumā ir redzams, kā tiek nodefinēta lidostas grafika kolekcijas dati izmantojot tās kolekcijas tipu “TABLE\_LIDOSTA\_GRAFIKS\_DATI\_TYPE” un tiek definēti jaunie kolekcijas objekti izmantojot nodefinēto jau objektu tipi “LIDOSTA\_GRAFIKS\_DATI\_TYPE”.

Rezultātā bija veiksmīgi ievadīti dati priekš lidostām un tās grafikiem (att. 3.4).



Att. 3.4 Ievadītas datnes priekš lidostām

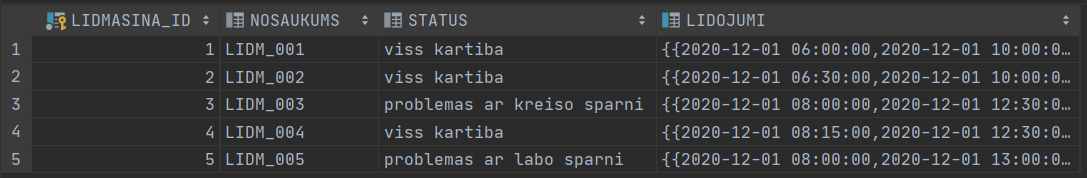
Bet lai realizēt datu ievade priekš lidmašīnām un tās lidojumiem bija nepieciešams izmantot PL/SQL, jo lidojumā objektā tiek jādefinē lidostas objektu atsauces saglabājot atsauces mainīgos. Kā piemērs tiek paradīts datu ievade priekš pirmā lidmašīna (visu INSERT vaicājumu ir iespējams redzēt 2. pielikumā):

DECLARE  
 lidosta1 REF LIDOSTA\_TYPE;  
 lidosta2 REF LIDOSTA\_TYPE;  
 lidosta3 REF LIDOSTA\_TYPE;  
BEGIN  
 SELECT REF(l)  
 INTO lidosta1  
 FROM LIDOSTA l  
 WHERE LIDOSTA\_ID = 1;  
  
 SELECT REF(l)  
 INTO lidosta2  
 FROM LIDOSTA l  
 WHERE LIDOSTA\_ID = 2;  
  
 SELECT REF(l)  
 INTO lidosta3  
 FROM LIDOSTA l  
 WHERE LIDOSTA\_ID = 3;  
  
 INSERT INTO LIDMASINA VALUES (1, 'LIDM\_001', 'viss kartiba', TABLE\_LIDOJUMS\_DATI\_TYPE (  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 06:00:00', TIMESTAMP '2020-12-01 10:00:00', lidosta1, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 14:00:00', TIMESTAMP '2020-12-01 18:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 00:00:00', TIMESTAMP '2020-12-02 07:00:00', lidosta2, lidosta1),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 12:00:00', TIMESTAMP '2020-12-02 16:00:00', lidosta1, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 01:00:00', TIMESTAMP '2020-12-03 06:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 11:00:00', TIMESTAMP '2020-12-03 17:00:00', lidosta2, lidosta1),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 22:00:00', TIMESTAMP '2020-12-04 01:00:00', lidosta1, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-04 06:30:00', TIMESTAMP '2020-12-04 12:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-04 15:00:00', TIMESTAMP '2020-12-04 20:00:00', lidosta2, lidosta1)  
 ));

END;

Ir redzams, ka visas lidostas tiek saglabāti “DECLARE” nodaļā mainīgos “lidosta1”, “lidosta2”, “lidosta3”, kuras vērtības tiek definētās speciālās “SELECT INTO” vaicājamas. Kā arī priekš lidojuma kolekcija tiek izmantota tās kolekcijas tips “TABLE\_LIDOJUMS\_DATI\_TYPE” un lidojuma objekti definēti izmantojot lidojuma objekta tipu “LIDOJUMS\_DATI\_TYPE”.

Rezultātā bija veiksmīgi ievadīti dati priekš lidmašīnām un tās lidojumiem (att. 3.5).



Att. 3.5 Ievadītas datnes priekš lidmašīnām

## Datu ievade SQL\*LOADER

Priekš SQL\*LOADER datu ievades metode bija izvēlēts ievadīt datus priekš lidostas darbiniekiem un tās grafikiem. No tā kā datu ievade ir nepieciešams arī ievadīt kolekcijas datus nebija iespējams izmantot “\*.par” failu, kurā tiek deklarēta visa nepieciešama informācija par datu ievade, jo SQL\*LOADER pagaidām nav iespējams darboties ar to un izmantot NESTED TABLE kolekcijas, kā bija minēts Oracle dokumentācija. No tā bija nodefinēta vienkāršā veida “\*.ctl” fails, kurā tiek minēts, kāds datu fails ir nepieciešams izmantot priekš ievadātiem. Kā piemērs tiek paradīts “darbinieks.ctl” fails priekš “Rīga Lidosta”:

LOAD DATA  
INFILE 'darbinieks\_riga.dat'  
insert into table LIDOSTA\_DARBINIEKS  
fields terminated by ","  
TIMESTAMP FORMAT "yyyy-mm-dd HH24:MI:SS"  
(  
 lidosta\_darbinieks\_id INTEGER EXTERNAL,  
 vards,  
 uzvards,  
 status,  
 darba\_pozicija,  
 lidosta\_id INTEGER EXTERNAL,  
 grafiki\_file FILLER CHAR,  
 grafiki NESTED TABLE SDF(grafiki\_file) COUNT (CONSTANT 5)  
 (  
 LIDOSTA\_DARBINIEKS\_GRAFIKS COLUMN OBJECT  
 (  
 strada\_no POSITION(1:20) TIMESTAMP,  
 strada\_lidz POSITION(21:40) TIMESTAMP,  
 nedelas\_diena POSITION(41:51) CHAR(11)  
 )  
 )  
)

Šajā piemērā ir redzams, ka izmantots “darbinieks\_riga.dat” fails, kurā ir ievadītas datnes par lidostas darbiniekiem. Arī tiek definēts TIMESTAMP formāts, lai visas laika ievad datnes būtu vienādā formātā. Vēl ir izmantots palīg lauks “grafiki\_file”, kurā tiek jāievada “\*.dat” failā nepieciešamu faila nosaukumu, kurā tiek glabātas darbinieka grafika temporālas datnes. Un priekš NESTED TABLE bija izmantots COUNT (CONSTANT 5), lai kontrolēt datu ievadē darbinieka grafika izmēru uz piecām dienām, jo kolekcijām ir nepieciešams nodefinēt ievad datnes izmērs, kas arī ir iespējams veikt izmantojot palīg lauku. Pēc tām ir definēta objektu tips “LIDOSTA\_DARBINIEKS\_GRAFIKS” kas ir COLUMN OBJECT un lai veiksmīgi ievadīt nepieciešamas datnes no objekta faila bija pieņemts izmantot POSITION funkciju ar kuru ir iespējams kontrolēt lauka izmēru, kas ir svarīgi priekš laika tipa datiem.

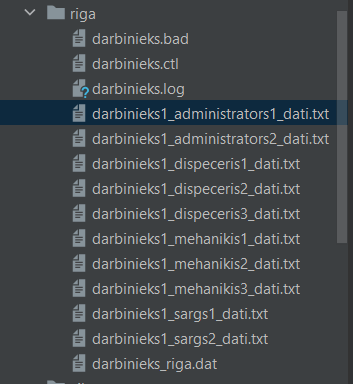
Kā piemērs “\*.dat” faila, ir paradīts “darbinieks\_riga.dat” faila dati:

1, Aleksejs, Krums, administre, administrators, 1,darbinieks1\_administrators1\_dati.txt  
2, Olegs, Olaps, administre, administrators, 1,darbinieks1\_administrators2\_dati.txt  
3, Mihails, Namps, strada, despetceris, 1,darbinieks1\_dispeceris1\_dati.txt  
4, Janis, Vanags, strada, despetceris, 1,darbinieks1\_dispeceris2\_dati.txt  
5, Kristaps, Jansons, strada, despetceris, 1,darbinieks1\_dispeceris3\_dati.txt  
6, Elena, Fjodorova, strada, mehanikis, 1,darbinieks1\_mehanikis1\_dati.txt  
7, Nikolajs, Rumbulis, strada, mehanikis, 1,darbinieks1\_mehanikis2\_dati.txt  
8, Toms, Kruze, strada, mehanikis, 1,darbinieks1\_mehanikis3\_dati.txt  
9, Rudolfs, Rasa, sarga, apsargs, 1,darbinieks1\_sargs1\_dati.txt  
10, Dmitrijs, Ivanovs, sarga, apsargs, 1,darbinieks1\_sargs2\_dati.txt

Ir redzams, ka pēdējā kolonnā ir ievadīta darbinieka grafika faila nosaukums, kuru kā piemērs, ir paradīts grafika datu fails priekš pirmā administratore :

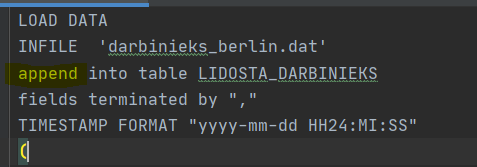
2020-12-01 08:00:00 2020-12-01 17:00:00 otrdiena  
2020-12-02 15:00:00 2020-12-02 23:00:00 tresdiena  
2020-12-03 15:00:00 2020-12-03 23:00:00 ceturtdiena  
2020-12-04 08:00:00 2020-12-04 17:00:00 piektdiena  
2020-12-05 14:30:00 2020-12-05 22:30:00 sestdiena

Kopumā, kopēja failu struktūra bija sekojoša (att. 3.6).



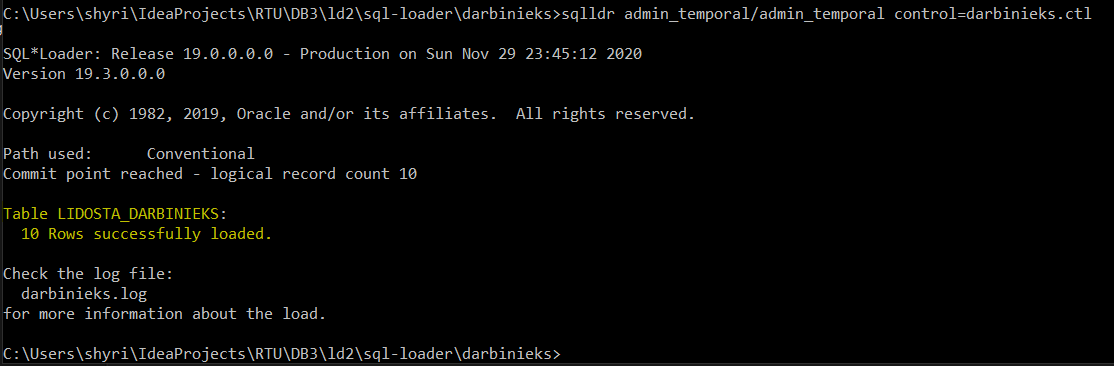
Att. 3.6 SQL\*LOADER faila ievad datnes struktūra

Lai paplašināt ar jauniem datiem LIDOSTA\_DARBINIEKS tabulu, bija nepieciešams izmainīt “\*.ctl” faila komandu no INSERT uz APPEND (att. 3.7), jo ar to ir iespējams pievienot jaunus ierakstus, izmantojot citus “\*.ctl” failus.



Att. 3.7 SQL\*LOADER APPEND

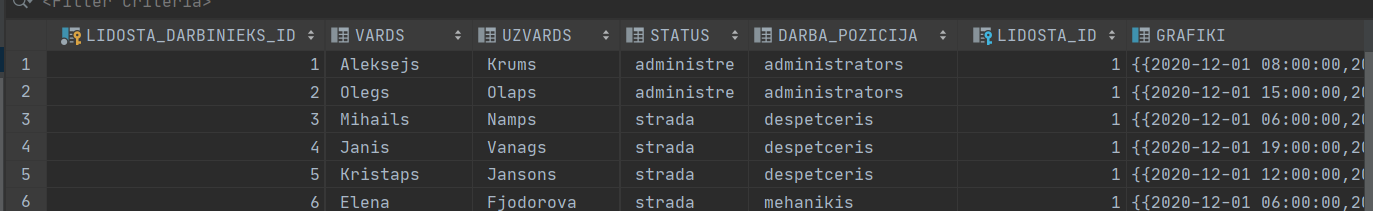
Lai palaist SQL\*LOADER terminālā bija izpildīta komanda (att. 3.8), priekš kurai bija nepieciešams definēt lietotāja vārdu un parole, kā arī “control” failu.



Att. 3.8 SQL\*LOADER komandas izpildījums

Ir redzams, ka visi dati tiek veiksmīgi ievadīti LIDOSTA\_DARBINIEKS tabulā.

Tādas pašas darbības bija veiktas priekš citiem lidostas darbiniekiem un rezultātā bija veiksmīgi ievadīti dati priekš lidostas darbiniekiem un tās darba grafikiem (att. 3.9).



Att. 3.9 Ievadītas datnes priekš lidostas darbiniekiem

# Objektu skati

Lai izveidot objektu skatus priekš temporāliem vaicājumiem bija nepieciešams izveidot jaunus objektu tipus, kuros tiek apvienota informācija no tabulām un temporāliem datiem, kā arī tiek izveidoti objektu ķermeņi (body), kuros tiek veikta Java funkcijas izsaukšana savukārt tos no sākuma bija jādefinē.

## Java klase un metodes deklarēšana PL/SQL vidē

Priekš šīm darbam bija izveidota tikai viena Java klase, kuru bija nepieciešams pārnest Oracle datubāzē. Lai to izdarīt, no sākuma ir jāizveido Java klase, kuru ir iespējams redzēt 3. pielikumā. Pēc tām izmantojot komandu “loadjava” Java klase “KorolkoClass.java” tika pārnesta Oracle kontekstā.

Komanda:

loadjava -v -u admin\_temporal/admin\_temporal@localhost:1522/shyrik -resolve -t "C:\Users\shyri\IdeaProjects\RTU\DB3\ld2\src\main\java\KorolkoClass.java"

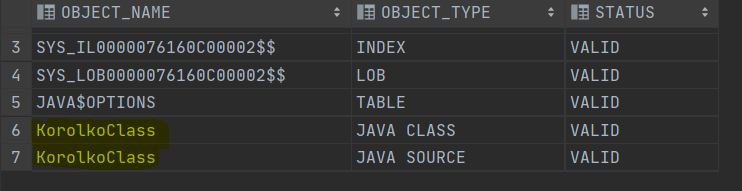
Lai veikt klases dzēšanu no Oracle vidē ir jāizmanto sekojošo komandu:

dropjava -v -u admin\_temporal/admin\_temporal@localhost:1522/shyrik KorolkoClass.java

Lai pārlecināties, ka Java klase veiksmīgi pārnesējas Oracle vidē bija izpildīts sekojošais vaicājums:

select OBJECT\_NAME, OBJECT\_TYPE, STATUS  
from USER\_OBJECTS  
where CREATED >= TO\_DATE('02-12-2020', 'DD-MM-YYYY');

Rezultātā Java klase bija ar statusu “VALID”, kas nozīme, ka klases pārnēsāšanas laikā nebija nekādas problēmas (att. 4.1):



Att. 4.1 Ievadītas datnes priekš lidostas darbiniekiem

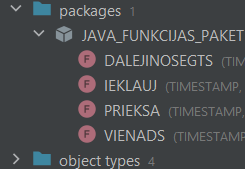
Pēc tām, lai būtu iespējams lietot Java klases funkcijas, bija pieņemts izveidot funkcijas pakete, kurā tiek nodefinēta visas Java funkcijas izmantošana.

Paketes izveidošanas vaicājums:

create or replace package java\_funkcijas\_pakete as  
 function *prieksa*(t1 in TIMESTAMP, t2 in TIMESTAMP)  
 return BOOLEAN as language java name  
 'KorolkoClass.prieksa(oracle.sql.TIMESTAMP, oracle.sql.TIMESTAMP) return boolean';  
  
 function *vienads*(t1 in TIMESTAMP, t2 in TIMESTAMP)  
 return BOOLEAN as language java name  
 'KorolkoClass.vienads(oracle.sql.TIMESTAMP, oracle.sql.TIMESTAMP) return boolean';  
  
 function *dalejiNosegts*(t1\_no in TIMESTAMP,  
 t1\_lidz in TIMESTAMP,  
 t2\_no in TIMESTAMP,  
 t2\_lidz in TIMESTAMP)  
 return BOOLEAN as language java name  
 'KorolkoClass.dalejiNosegts(oracle.sql.TIMESTAMP, oracle.sql.TIMESTAMP, oracle.sql.TIMESTAMP, oracle.sql.TIMESTAMP) return boolean';  
  
 function *ieklauj*(t1\_no in TIMESTAMP,  
 t1\_lidz in TIMESTAMP,  
 t2\_no in TIMESTAMP,  
 t2\_lidz in TIMESTAMP)  
 return BOOLEAN as language java name  
 'KorolkoClass.ieklauj(oracle.sql.TIMESTAMP, oracle.sql.TIMESTAMP, oracle.sql.TIMESTAMP, oracle.sql.TIMESTAMP) return boolean';  
end;  
/

Ir redzams, ka funkcijās tiek deklarētās tikai nepieciešamas metodes, lai būtu iespējams izveidot temporālus vaicājumus, kā arī, funkcijas parametri ir ar datu tipu TIMESTAMP, kas ir svarīgi, jo Java definētās funkcijās parametri ir ar TIMESTAMP datu tipu. Un atgriešanas vērtībās tiek izmantots BOOLEAN datu tips, jo funkcijās tiek veikts pārbaudījums uz to, vai izpildās nepieciešamas temporālās algebras darbības vai nē.

Rezultātā bija pievienots datubāzes kontekstā jauna funkcijas pakete, kuru ir iespējams lietot PL/SQL vidē (att. 4.2).



Att. 4.2 Funkcijas pakete

## Objektu skatu izveide

Pirms objektu skatiem izveidošanas bija nepieciešams izveidot tās objektu tipa struktūru, kā arī, nodefinēt tajos visas nepieciešamas metodes, lai tīrāk un vieglāk veikt temporālas algebras darbības. Objektu skatu tipos bija iekļauti visi lauki no tabulām un tās temporāliem objektiem, lai būtu iespējams attēlot visu informāciju par datubāzes datiem. Kā piemērs tiek paradīts objektu skata izveidošana priekš lidostas grafikiem un visus parējos ir iespējams apskatīt 4. pielikumā.

Objektu skata tipa izveidošana ar metodēm:

create or replace type LIDOSTA\_GRAFIKS\_SKATS\_TYPE as object (  
 lidosta\_id number,  
 nosaukums varchar2(255),  
 datums\_no TIMESTAMP,  
 datums\_lidz TIMESTAMP,  
 nedelas\_diena varchar2(24),  
 order member function *SORT\_BY\_DATUMS\_NO* (grafiks LIDOSTA\_GRAFIKS\_SKATS\_TYPE) return number,  
 member function *DALEJI\_NOSEGT* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number,  
 member function *IR\_PRIEKSA* (obj\_no in TIMESTAMP) return number  
);  
/

Objekta skata ķermeņa definēšana:

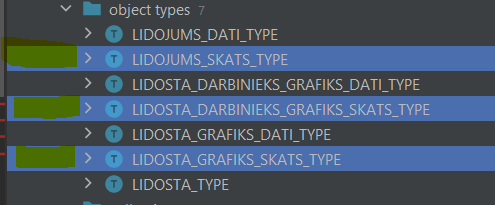
create or replace type body LIDOSTA\_GRAFIKS\_SKATS\_TYPE as  
 order member function *SORT\_BY\_DATUMS\_NO* (grafiks LIDOSTA\_GRAFIKS\_SKATS\_TYPE) return number is  
 begin  
 if (JAVA\_FUNKCIJAS\_PAKETE.*PRIEKSA*(SELF.DATUMS\_NO, grafiks.DATUMS\_NO)) then  
 return -1;  
 end if;  
 return 1;  
 end;  
 member function *DALEJI\_NOSEGT* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number is  
 begin  
 if (JAVA\_FUNKCIJAS\_PAKETE.*DALEJINOSEGTS*(SELF.DATUMS\_NO, SELF.DATUMS\_LIDZ, obj\_no, obj\_lidz)) then  
 return 1;  
 end if;  
 return 0;  
 end;  
 member function *IR\_PRIEKSA* (obj\_no in TIMESTAMP) return number is  
 begin  
 if (JAVA\_FUNKCIJAS\_PAKETE.*PRIEKSA*(SELF.DATUMS\_LIDZ, obj\_no)) then  
 return 1;  
 end if;  
 return 0;  
 end;  
end;  
/

Pēc dota piemēra ir redzams, ka ir izmantotas Java funkcijas, izsauktas no funkcijas pakete, PL/SQL vidē, kā arī tiek nodefinētas visas funkcijas tipi (ORDER un MEMEBER). Kā funkcijas atgriešanas vērtības tips, tiek ņemts NUMBER datu tips, jo bija problēmas ar BOOLEAN datu tipa izmantošanu PL/SQL funkcijās.

Pēc visas objektu skatu definēšanu un izveidošanu ir iespējams pārbaidīt, vai eksistē kaut kādas problēmas ar tiem. Lai to izdarīt ir nepieciešams vaicājumu, kurā būtu paradīts kaut kas, jā eksistē problēmas ar datu tipiem vai ķermeņa realizēšanu:

SELECT owner, object\_type, object\_name  
FROM all\_objects  
WHERE status = 'INVALID';

Rezultātā bija veiksmīgi izveidoti visi objektu skata tipi (att. 4.3).



Att. 4.3 Objektu skati tipi

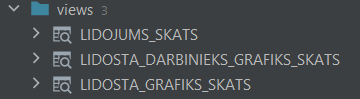
Kad tiek izveidoti visi objektu skata tipi ir iespējams definēt un izveidot objektu skatus. Objektu skatos tiek pielietotas visas objektu skata tipi, lai būtu iespējams izmantot tās nodefinētas funkcijas. Kā piemērs ir paradīts objektu skata izveidošana priekš lidostas grafikiem, bet pārējos ir iespējams apskatīt 5. pielikumā.

Objektu skata definēšana:

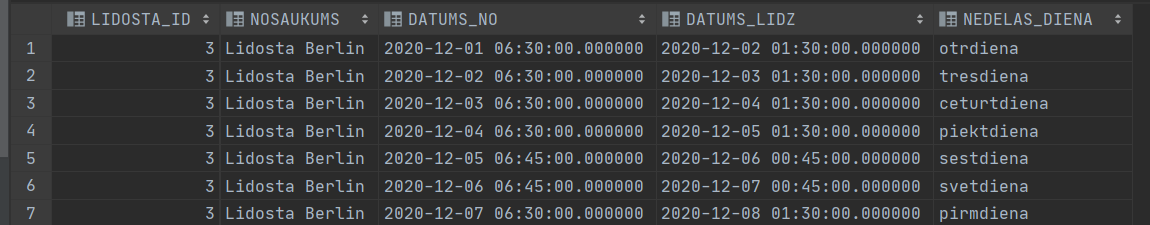
create or replace view LIDOSTA\_GRAFIKS\_SKATS of LIDOSTA\_GRAFIKS\_SKATS\_TYPE  
 with object identifier(datums\_no, datums\_lidz) as  
select L.LIDOSTA\_ID, NOSAUKUMS,  
 LG.DATUMS\_NO, LG.DATUMS\_LIDZ, LG.NEDELAS\_DIENA  
from LIDOSTA L, TABLE(L.GRAFIKI) LG;

Ir redzams, ka tiek ņemti visi lauki un tās vērtības no LIDOSTA tabulā, kā arī tiek pievienoti lauki no temporālas datnes kolekcijas GRAFIKI, kura bija pārdefinēta uz tabulu ar TABLE funkciju. Arī skats bija izveidots izmantojot nodefinēto objektu skata tipu “LIDOSTA\_GRAFIKS\_SKATS\_TYPE” un tiek uzradīts par ieraksta identifikatoriem laika vērtības, kas ir nepieciešams, lai viss būtu sakārtots pēc laika vērtībām vienā objekta kontekstā.

Rezultātā visi objektu skati bija izveidoti (att. 4.4), kā arī tajos tiek saglabāti visi nodefinētie dati (att. 4.5).



Att. 4.4 Objektu skati



Att. 4.4 Objektu skata LIDOSTA\_GRAFIKS\_SKATS datnes

# Temporālo vaicājumu izpildīšana un pārbaudīšana

Šajā nodaļa tiek veiktas vaicājumu izpildes analīze un tās parbaudīšana.

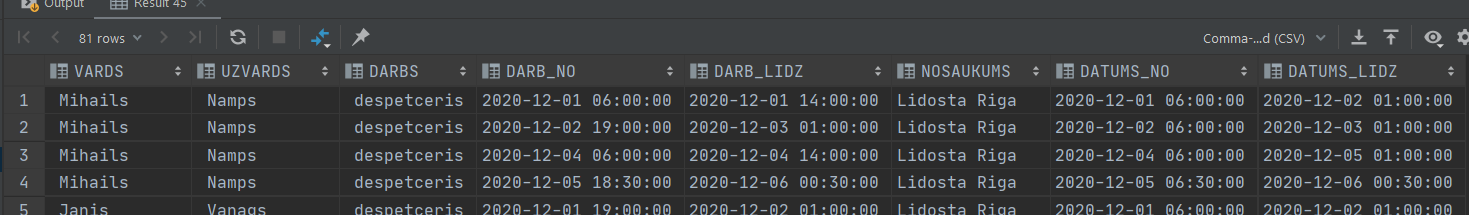
## Pirmais vaicājums

Lai izveidot pirmo temporālo vaicājumu bija nepieciešams izmantot lidostas un tās darbinieku skatus savienojot tos savā starpā ar INNER JOIN, lai izvadīt tādus darbiniekus, kuriem darba laiks sakrīt ar viņas lidostas grafika sākšanu vai nobeigšanu. Kā arī bija izmantotas objektu funkcijas SAKTS\_AR un BEIDZAS\_AR lai ērtāk veikt pārbaudījumus vaicājumā.

Vaicājuma kods:

select LDGS.VARDS as vards,  
 LDGS.UZVARDS as uzvards, LDGS.DARBA\_POZICIJA as darbs,  
 TO\_CHAR (LDGS.STRADA\_NO, 'YYYY-MM-DD HH24:MI:SS') as darb\_no, TO\_CHAR (LDGS.STRADA\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') as darb\_lidz,  
 LGS.NOSAUKUMS, TO\_CHAR (LGS.DATUMS\_NO, 'YYYY-MM-DD HH24:MI:SS') as datums\_no, TO\_CHAR (LGS.DATUMS\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') as datums\_lidz  
from LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS LDGS  
 inner join LIDOSTA\_GRAFIKS\_SKATS LGS on LDGS.LIDOSTA\_ID = LGS.LIDOSTA\_ID and LDGS.NEDELAS\_DIENA = LGS.NEDELAS\_DIENA  
where (VALUE (LDGS).SAKTS\_AR(LGS.DATUMS\_NO, LGS.DATUMS\_LIDZ)) = 1 or (VALUE (LDGS).BEIDZAS\_AR(LGS.DATUMS\_NO, LGS.DATUMS\_LIDZ)) = 1  
;

Ir redzams, ka ir izmantots loģiskais operators OR, lai atrast sakarības starp lidostas grafika un tās darbinieku grafikiem, kā arī INNER JOIN tiek veikts pārbaudījums uz lidostas sakarību un grafika nedēļas dienas, lai izvadīt tikai nepieciešamus darbiniekus, kuriem darba diena sākas ar lidostas atvēršanu vai slēgšanu laikiem. Vaicājumā izpildīšanas rezultātā bija saņemtas lidostas darbinieku grafiki un lidostas grafiki (att. 5.1).



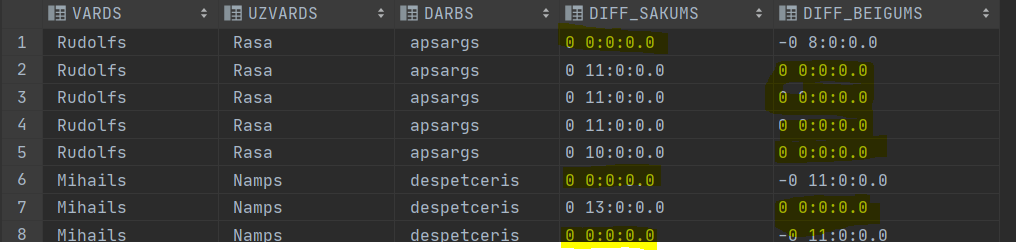
Att. 5.1 Pirmā vaicājuma rezultāts

Lai pārbaudīt vaicājumu uz pareizību ir iespējams izveidot citu vaicājumu, kurā tiek veikta laika starpība aprēķināšanā starp grafikas sākumiem un beigumiem un jā vienā no tās kolonas ir 0, tad šīs vērtības sakrīt:

Vaicājuma kods:

select LDGS.VARDS as vards,  
 LDGS.UZVARDS as uzvards, LDGS.DARBA\_POZICIJA as darbs,  
 (LDGS.STRADA\_NO - LGS.DATUMS\_NO) diff\_sakums,  
 (LDGS.STRADA\_LIDZ - LGS.DATUMS\_LIDZ) diff\_beigums  
from LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS LDGS  
 inner join LIDOSTA\_GRAFIKS\_SKATS LGS on LDGS.LIDOSTA\_ID = LGS.LIDOSTA\_ID and LDGS.NEDELAS\_DIENA = LGS.NEDELAS\_DIENA  
where (VALUE (LDGS).SAKTS\_AR(LGS.DATUMS\_NO, LGS.DATUMS\_LIDZ)) = 1 or (VALUE (LDGS).BEIDZAS\_AR(LGS.DATUMS\_NO, LGS.DATUMS\_LIDZ)) = 1  
;

Vaicājuma rezultāts (att. 5.2):



Att. 5.2 Pirmā vaicājuma pārbaudīšana

Pēc pārbaudīšanas ir redzams, ka vaicājums strādā korekti un ir iemesls turpināt ar nākamo vaicājumu.

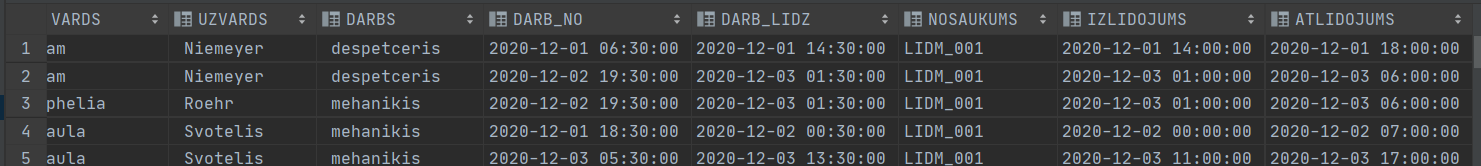
## Otrais vaicājums

Lai izveidot otro vaicājumu bija nepieciešams izmantot lidmašīnas lidojumu un lidostas darbinieku skatus savienojot tos savā starpā ar INNER JOIN, lai izvadīt tādus rakstus, kur darbinieka lidosta sakrīt ar lidmašīnas izlidojuma lidostu, lai noteikt kādi darbinieki savā darba laikā apkalpoja kādus lidmašīnas pirms tās izlidojuma no lidostas. Arī lai veikt pareizas temporālas algebras darbības ir nepieciešams izmantot funkciju DALEJI\_NOSEGT.

Vaicājuma kods:

select LDGS.VARDS as vards,  
 LDGS.UZVARDS as uzvards, LDGS.DARBA\_POZICIJA as darbs,  
 TO\_CHAR (LDGS.STRADA\_NO, 'YYYY-MM-DD HH24:MI:SS') as darb\_no, TO\_CHAR (LDGS.STRADA\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') as darb\_lidz,  
 LS.NOSAUKUMS,  
 TO\_CHAR (LS.DATUMS\_NO, 'YYYY-MM-DD HH24:MI:SS') as izlidojums, TO\_CHAR (LS.DATUMS\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') as atlidojums  
from LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS LDGS  
 inner join LIDOJUMS\_SKATS LS on  
 LDGS.LIDOSTA\_ID = DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID  
where (VALUE (LDGS).DALEJI\_NOSEGT(LS.DATUMS\_NO, LS.DATUMS\_LIDZ)) = 1  
;

Vaicājuma rezultāts (5.3):



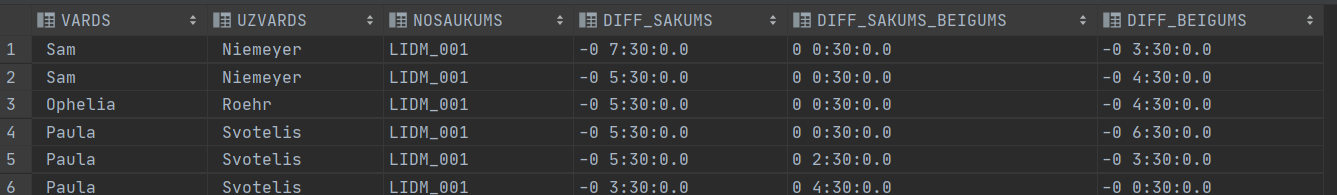
Att. 5.3 Otrā vaicājuma rezultāts

Lai pārbaudīt vaicājumu uz pareizību bija izveidots cits vaicājums, kurā veiktas laika starpība aprēķināšanā starp darbinieka grafika sākumu ar lidojuma sākumu (vērtība jābūt mazāk par 0), darbinieka grafika beigšanu un ar lidojuma sākumu (vērtība jābūt lielāk par 0) un darbinieka grafika beigšanu un ar lidojuma beigšanu (vērtība jābūt mazāk par 0).

Vaicājuma kods:

select LDGS.VARDS as vards,  
 LDGS.UZVARDS as uzvards,  
 LS.NOSAUKUMS,  
 (LDGS.STRADA\_NO - LS.DATUMS\_NO) diff\_sakums,  
 (LDGS.STRADA\_LIDZ - LS.DATUMS\_NO) diff\_sakums\_beigums,  
 (LDGS.STRADA\_LIDZ - LS.DATUMS\_LIDZ) diff\_beigums  
from LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS LDGS  
 inner join LIDOJUMS\_SKATS LS on  
 LDGS.LIDOSTA\_ID = DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID  
where (VALUE (LDGS).DALEJI\_NOSEGT(LS.DATUMS\_NO, LS.DATUMS\_LIDZ)) = 1  
;

Vaicājuma rezultāts (att. 5.4):



Att. 5.4 Otrā vaicājuma pārbaudīšana

Pēc pārbaudīšanas ir redzams, ka vaicājums strādā korekti un ir iemesls turpināt ar nākamo vaicājumu.

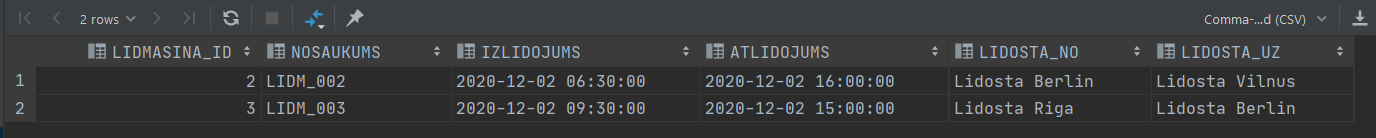
## Trešais vaicājums

Tā kā trešā vaicājumā bija nepieciešams nodefinēt lidojumus 02.12.2020 datuma pusdienās laikā ar temporālas algebras darbību atlase, tad nav nepieciešams izmantot nekādas nodefinētas objekta skata funkcijas.

Vaicājuma kods:

select LS.LIDMASINA\_ID, LS.NOSAUKUMS,  
 TO\_CHAR (LS.DATUMS\_NO, 'YYYY-MM-DD HH24:MI:SS') izlidojums,  
 TO\_CHAR (LS.DATUMS\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') atlidojums,  
 DEREF(LS.LIDOSTA\_NO).NOSAUKUMS lidosta\_no, DEREF(LS.LIDOSTA\_UZ).NOSAUKUMS lidosta\_uz  
from LIDOJUMS\_SKATS LS  
where LS.DATUMS\_NO < (TIMESTAMP '2020-12-02 12:00:00') and  
 LS.DATUMS\_LIDZ > (TIMESTAMP '2020-12-02 12:00:00')  
;

Vecajumā rezultāts (att. 5.5):



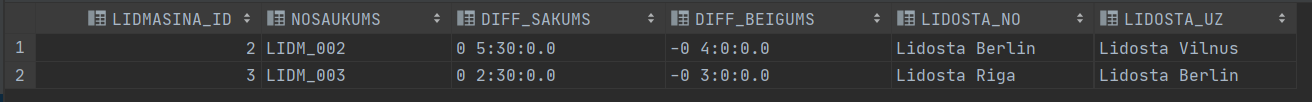
Att. 5.5 Trešā vaicājuma rezultāts

Lai pārbaudīt vaicājumu uz pareizību bija izveidots cits vaicājums, kurā veiktas laika starpība aprēķināšanā starp 02.12.2020 datuma pusdienās laika un lidostas sākumu (vērtība jābūt lielāk par 0) un nobeigumu (vērtība jābūt mazāk par 0).

Vaicājuma kods:

select LS.LIDMASINA\_ID, LS.NOSAUKUMS,  
 ((TIMESTAMP '2020-12-02 12:00:00') - LS.DATUMS\_NO) diff\_sakums,  
 ((TIMESTAMP '2020-12-02 12:00:00') - LS.DATUMS\_LIDZ) diff\_beigums,  
 DEREF(LS.LIDOSTA\_NO).NOSAUKUMS lidosta\_no, DEREF(LS.LIDOSTA\_UZ).NOSAUKUMS lidosta\_uz  
from LIDOJUMS\_SKATS LS  
where LS.DATUMS\_NO < (TIMESTAMP '2020-12-02 12:00:00') and  
 LS.DATUMS\_LIDZ > (TIMESTAMP '2020-12-02 12:00:00')  
;

Vecajumā rezultāts (att. 5.6):



Att. 5.6 Trešā vaicājuma pārbaudīšana

Pēc pārbaudīšanas ir redzams, ka vaicājums strādā korekti un ir iemesls turpināt ar nākamo vaicājumu.

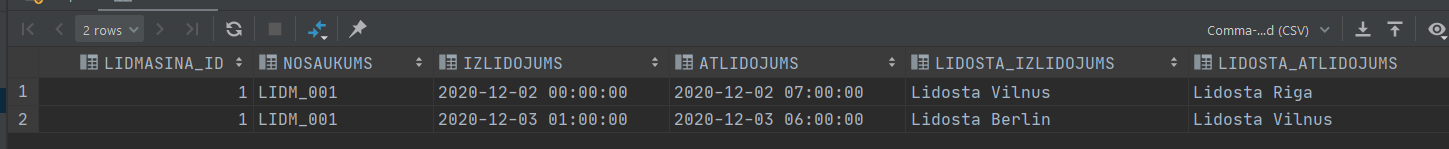
## Ceturtais vaicājums

Lai realizēt ceturto vaicājumu, kurā ir nepieciešams atrast lidmašīnas, kuriem lidojums tiek veikts naktī, kad lidosta no kurienes tiek veikts izlidojums un kurā tiek veikta atlidošana ir cietas lidojuma laikā, ir nepieciešams veikt temporālas algebras “daļēji nosegt” starp lidostas, no kurienes tiek veikts izlidojums, ar lidojumiem un starp lidojumiem un lidostām uz kurienes tiek veikts atlidojams, kā arī savā starpā pārbaudīt vai ir priekša pirmās lidostas grafika nobeigšana salīdzinot ar otrā lidosta grafika sākumu.

Vaicājuma kods:

select LS.LIDMASINA\_ID, LS.NOSAUKUMS,  
 TO\_CHAR (LS.DATUMS\_NO, 'YYYY-MM-DD HH24:MI:SS') izlidojums,  
 TO\_CHAR (LS.DATUMS\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') atlidojums,  
 DEREF(LS.LIDOSTA\_NO).NOSAUKUMS lidosta\_izlidojums,  
 DEREF(LS.LIDOSTA\_UZ).NOSAUKUMS lidosta\_atlidojums  
from LIDOJUMS\_SKATS LS  
 inner join LIDOSTA\_GRAFIKS\_SKATS LGS\_NO on LGS\_NO.LIDOSTA\_ID = DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID  
 inner join LIDOSTA\_GRAFIKS\_SKATS LGS\_UZ on LGS\_UZ.LIDOSTA\_ID = DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID  
where VALUE(LGS\_NO).DALEJI\_NOSEGT(LS.DATUMS\_NO, LS.DATUMS\_LIDZ) = 1 and  
 VALUE(LS).DALEJI\_NOSEGT(VALUE(LGS\_UZ).DATUMS\_NO, VALUE(LGS\_UZ).DATUMS\_LIDZ) = 1 and  
 VALUE(LGS\_NO).IR\_PRIEKSA(VALUE(LGS\_UZ).DATUMS\_NO) = 1  
;

Vaicājuma rezultāts (att. 5.7):



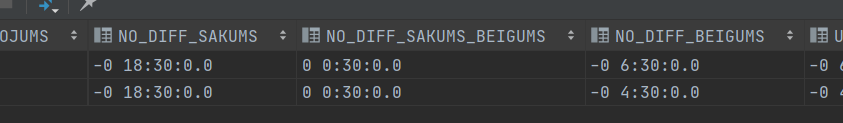
Att. 5.7 Ceturtā vaicājuma rezultāts

Lai pārbaudīt vaicājumu uz pareizību, bija izveidots cits vaicājums, kurā tiek veikta laika starpība aprēķināšanā starp lidostās grafikiem (vērtība jābūt lielāk par 0), jo tiek izmantota temporālas algebras darbība PROCEDES, lidostas (izlidojums) grafika sākumu ar lidojuma sākumu (vērtība jābūt mazāk par 0), lidostas (izlidojums) grafika beigšanu ar lidojuma sākumu (vērtība jābūt lielāk par 0), lidostas (izlidojums) grafika beigšanu un ar lidojuma beigšanu (vērtība jābūt mazāk par 0), lidojuma sākumu ar lidostas (atlidošana) grafika sākumu (vērtība jābūt mazāk par 0), lidojuma sākumu ar lidostas (atlidošana) grafika beigšanu (vērtība jābūt lielāk par 0) un lidojuma beigšanu ar lidostas (atlidošana) grafika beigšanu (vērtība jābūt mazāk par 0)

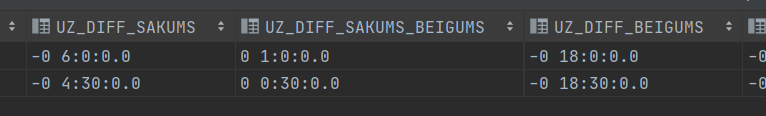
Vaicājuma kods:

select LS.NOSAUKUMS,  
 DEREF(LS.LIDOSTA\_NO).NOSAUKUMS lidosta\_izlidojums,  
 DEREF(LS.LIDOSTA\_UZ).NOSAUKUMS lidosta\_atlidojums,  
 (VALUE(LGS\_NO).DATUMS\_NO - LS.DATUMS\_NO) no\_diff\_sakums,  
 (VALUE(LGS\_NO).DATUMS\_LIDZ - LS.DATUMS\_NO) no\_diff\_sakums\_beigums,  
 (VALUE(LGS\_NO).DATUMS\_LIDZ - LS.DATUMS\_LIDZ) no\_diff\_beigums,  
 (LS.DATUMS\_NO - VALUE(LGS\_UZ).DATUMS\_NO) uz\_diff\_sakums,  
 (LS.DATUMS\_LIDZ - VALUE(LGS\_UZ).DATUMS\_NO) uz\_diff\_sakums\_beigums,  
 (LS.DATUMS\_LIDZ - VALUE(LGS\_UZ).DATUMS\_LIDZ) uz\_diff\_beigums,  
 (VALUE(LGS\_NO).DATUMS\_LIDZ - VALUE(LGS\_UZ).DATUMS\_NO) lidostas\_diff  
from LIDOJUMS\_SKATS LS  
 inner join LIDOSTA\_GRAFIKS\_SKATS LGS\_NO on LGS\_NO.LIDOSTA\_ID = DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID  
 inner join LIDOSTA\_GRAFIKS\_SKATS LGS\_UZ on LGS\_UZ.LIDOSTA\_ID = DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID  
where VALUE(LGS\_NO).DALEJI\_NOSEGT(LS.DATUMS\_NO, LS.DATUMS\_LIDZ) = 1 and  
 VALUE(LS).DALEJI\_NOSEGT(VALUE(LGS\_UZ).DATUMS\_NO, VALUE(LGS\_UZ).DATUMS\_LIDZ) = 1 and  
 VALUE(LGS\_NO).IR\_PRIEKSA(VALUE(LGS\_UZ).DATUMS\_NO) = 1  
;

Vaicājuma rezultāts (att. 5.8, att. 5.9, att. 5.10):



Att. 5.8 Lidostas (izlidojums) grafika ar lidmašīnas lidojuma pārbaude uz OVERLAPS



Att. 5.9 Lidmašīnas lidojuma ar lidostas (atlidošana) grafika pārbaude uz OVERLAPS



Att. 5.10 Lidostas (izlidojums) grafika ar lidostas (atlidošana) grafika pārbaude uz PROCEDES

Pēc pārbaudīšanas ir redzams, ka vaicājums strādā korekti un ir iemesls turpināt ar nākamo vaicājumu.

## Piektais vaicājums

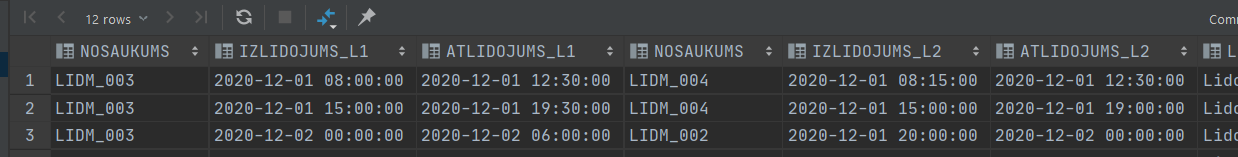
Lai realizēt piekto vaicājumu, kurā bija nepieciešams nodefinēt lidmašīnas, kā arī lidmašīnas reģistrēto lidostu, kuriem varētu būt izlidojuma vai atlidošanas laikā sastrēgums, bija vajadzīgs atrast tādus ierakstus, kuriem izlidojuma laiks sakrīt un izlidojums tiek veikts no vienāda lidostā vai atlidošanas laiks sakrīt un atlidošana tiek veikta uz vienādu lidostu vai izlidojuma laiks satiekas ar atlidošanas laiku vienā un tāda pašā lidostā. Arī neiekļaut tādus gadījumus, kad lidojumi ir vienādas.

Vaicājuma kods:

select LS.NOSAUKUMS,  
 TO\_CHAR (LS.DATUMS\_NO, 'YYYY-MM-DD HH24:MI:SS') izlidojums\_l1,  
 TO\_CHAR (LS.DATUMS\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') atlidojums\_l1,  
 LS2.NOSAUKUMS,  
 TO\_CHAR (LS2.DATUMS\_NO, 'YYYY-MM-DD HH24:MI:SS') izlidojums\_l2,  
 TO\_CHAR (LS2.DATUMS\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') atlidojums\_l2,  
 DEREF(LS.LIDOSTA\_NO).NOSAUKUMS lidosta\_izlidojums\_l1,  
 DEREF(LS.LIDOSTA\_UZ).NOSAUKUMS lidosta\_atlidojums\_l1,  
 DEREF(LS2.LIDOSTA\_NO).NOSAUKUMS lidosta\_atlidojums\_l2,  
 DEREF(LS2.LIDOSTA\_UZ).NOSAUKUMS lidosta\_atlidojums\_l2  
from LIDOJUMS\_SKATS LS  
 inner join LIDOJUMS\_SKATS LS2 on (DEREF(LS2.LIDOSTA\_NO).LIDOSTA\_ID =  
 DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID or  
 DEREF(LS2.LIDOSTA\_UZ).LIDOSTA\_ID =  
 DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID or  
 DEREF(LS2.LIDOSTA\_NO).LIDOSTA\_ID =  
 DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID or  
 DEREF(LS2.LIDOSTA\_UZ).LIDOSTA\_ID =  
 DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID)  
where VALUE(LS) != VALUE(LS2) and  
 (((VALUE(LS).SAKTS\_AR(VALUE(LS2).DATUMS\_NO, VALUE(LS2).DATUMS\_LIDZ) = 1 or  
 VALUE(LS2).SAKTS\_AR(VALUE(LS).DATUMS\_NO, VALUE(LS).DATUMS\_LIDZ) = 1) and  
 DEREF(LS2.LIDOSTA\_NO).LIDOSTA\_ID = DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID) or  
 (VALUE(LS).SATIEK(VALUE(LS2).DATUMS\_NO) = 1 and  
 DEREF(LS2.LIDOSTA\_NO).LIDOSTA\_ID = DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID) or  
 (VALUE(LS2).SATIEK(VALUE(LS).DATUMS\_NO) = 1 and  
 DEREF(LS2.LIDOSTA\_UZ).LIDOSTA\_ID = DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID) or  
 ((VALUE(LS).BEIDZAS\_AR(VALUE(LS2).DATUMS\_NO, VALUE(LS2).DATUMS\_LIDZ) = 1 or  
 VALUE(LS2).BEIDZAS\_AR(VALUE(LS).DATUMS\_NO, VALUE(LS).DATUMS\_LIDZ) = 1) and  
 DEREF(LS2.LIDOSTA\_UZ).LIDOSTA\_ID = DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID))  
;

(LS2.LIDOSTA\_UZ).LIDOSTA\_ID = DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID))  
;

Vaicājuma rezultāts (att. 5.11):



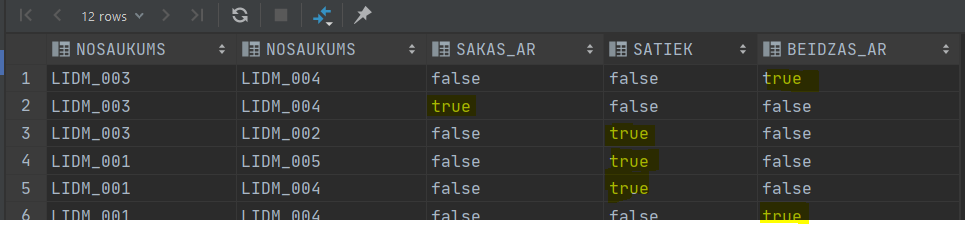
Att. 5.11 Piektā vaicājuma rezultāts

Lai pārbaudīt vaicājumu uz pareizību, bija izveidots cits vaicājums, kurā bija veikta laika starpība aprēķināšanā starp lidojumiem uz sekojošiem temporālas algebras darbībām : sakas ar, beidzas ar vai satiek. Arī bija samainīts pārbaudīšanas princips, jo ir daudz vērtības, kurus ir jāsalīdzina, lai pārbaudīt vaicājumu uz pareizību, no tā, izmantojot CASE WHEN bija veikta īsāka pārbaudes izvade, kurā tiek izvadīts “true”, vai izpildās temporālas algebras nosacījums un “false”, kā nē.

Vaicājuma kods:

select LS.NOSAUKUMS,  
 LS2.NOSAUKUMS,  
 case when (  
 ((LS.DATUMS\_NO - LS2.DATUMS\_NO) = interval '0' second and (LS.DATUMS\_LIDZ - LS2.DATUMS\_LIDZ) < interval '0' second) or  
 ((LS2.DATUMS\_NO - LS.DATUMS\_NO) = interval '0' second and (LS2.DATUMS\_LIDZ - LS.DATUMS\_LIDZ) < interval '0' second)  
 ) then 'true' else 'false' end sakas\_ar,  
 case when ((LS.DATUMS\_LIDZ - LS2.DATUMS\_NO) = interval '0' second or (LS2.DATUMS\_LIDZ - LS.DATUMS\_NO) = interval '0' second) then 'true' else 'false' end satiek,  
 case when (  
 ((LS.DATUMS\_LIDZ - LS2.DATUMS\_LIDZ) = interval '0' second and (LS.DATUMS\_NO - LS2.DATUMS\_NO) < interval '0' second) or  
 ((LS2.DATUMS\_LIDZ - LS.DATUMS\_LIDZ) = interval '0' second and (LS2.DATUMS\_NO - LS.DATUMS\_NO) < interval '0' second)  
 ) then 'true' else 'false' end beidzas\_ar  
from LIDOJUMS\_SKATS LS  
 inner join LIDOJUMS\_SKATS LS2 on (DEREF(LS2.LIDOSTA\_NO).LIDOSTA\_ID =  
 DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID or  
 DEREF(LS2.LIDOSTA\_UZ).LIDOSTA\_ID =  
 DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID or  
 DEREF(LS2.LIDOSTA\_NO).LIDOSTA\_ID =  
 DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID or  
 DEREF(LS2.LIDOSTA\_UZ).LIDOSTA\_ID =  
 DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID)  
where VALUE(LS) != VALUE(LS2) and  
 (((VALUE(LS).SAKTS\_AR(VALUE(LS2).DATUMS\_NO, VALUE(LS2).DATUMS\_LIDZ) = 1 or  
 VALUE(LS2).SAKTS\_AR(VALUE(LS).DATUMS\_NO, VALUE(LS).DATUMS\_LIDZ) = 1) and  
 DEREF(LS2.LIDOSTA\_NO).LIDOSTA\_ID = DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID) or  
 (VALUE(LS).SATIEK(VALUE(LS2).DATUMS\_NO) = 1 and  
 DEREF(LS2.LIDOSTA\_NO).LIDOSTA\_ID = DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID) or  
 (VALUE(LS2).SATIEK(VALUE(LS).DATUMS\_NO) = 1 and  
 DEREF(LS2.LIDOSTA\_UZ).LIDOSTA\_ID = DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID) or  
 ((VALUE(LS).BEIDZAS\_AR(VALUE(LS2).DATUMS\_NO, VALUE(LS2).DATUMS\_LIDZ) = 1 or  
 VALUE(LS2).BEIDZAS\_AR(VALUE(LS).DATUMS\_NO, VALUE(LS).DATUMS\_LIDZ) = 1) and  
 DEREF(LS2.LIDOSTA\_UZ).LIDOSTA\_ID = DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID))  
;

Vaicājuma rezultāts (att. 5.12):



Att. 5.12 Piektā vaicājuma pārbaude

Pēc pārbaudīšanas ir redzams, ka vaicājums strādā korekti un ir iemesls turpināt ar nākamo vaicājumu.

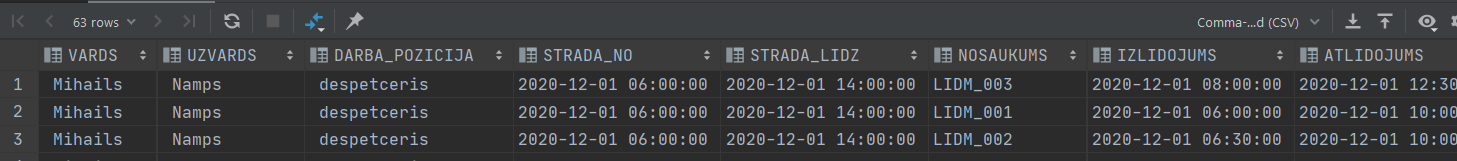
## Sestais vaicājums

Lai realizēt sesto vaicājumu, kurā ir nepieciešams atrast tādus dispečerus, kuriem darbā laikā bija pilnīgi paveikti lidmašīnas lidojumi, tas nozīme, ka dispečeris kontrolēja savā darbā laikā visu lidojumu no tās sākuma līdz tās beigām, bija vajadzīgs atrast tādus ierakstus, kur lidojuma laiks sākas ar dispečera darba grafika laiku vai lidojums ir iekļauts dispečera darba grafika laikā vai lidojuma laiks beidzas ar dispečera darba grafika laiku. Arī ir paņemt tādus rakstus, kur dispečera lidosta sakrīt ar vienu no lidmašīnas lidojumu lidostām.

Vaicājuma kods:

select LDGS.VARDS, LDGS.UZVARDS,  
 LDGS.DARBA\_POZICIJA,  
 TO\_CHAR (LDGS.STRADA\_NO, 'YYYY-MM-DD HH24:MI:SS') strada\_no,  
 TO\_CHAR (LDGS.STRADA\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') strada\_lidz,  
 LS.NOSAUKUMS,  
 TO\_CHAR (LS.DATUMS\_NO, 'YYYY-MM-DD HH24:MI:SS') izlidojums,  
 TO\_CHAR (LS.DATUMS\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') atlidojums,  
 DEREF(LS.LIDOSTA\_NO).NOSAUKUMS, DEREF(LS.LIDOSTA\_UZ).NOSAUKUMS  
from LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS LDGS  
 inner join LIDOJUMS\_SKATS LS on (DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID = LDGS.LIDOSTA\_ID or  
 DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID = LDGS.LIDOSTA\_ID)  
where (VALUE(LS).SAKTS\_AR(LDGS.STRADA\_NO, LDGS.STRADA\_LIDZ) = 1 or  
 VALUE(LS).BEIDZAS\_AR(LDGS.STRADA\_NO, LDGS.STRADA\_LIDZ) = 1 or  
 VALUE(LDGS).IEKLAUJ(LS.DATUMS\_NO, LS.DATUMS\_LIDZ) = 1) and  
 LDGS.DARBA\_POZICIJA like '%despetceris%'  
;

Vaicājums rezultāts (att. 5.13):



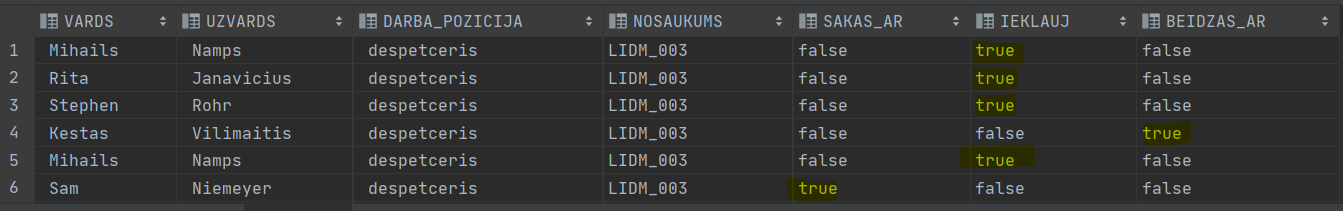
Att. 5.13 Sestā vaicājuma rezultāts

Lai pārbaudīt vaicājumu uz pareizību, bija izveidots cits vaicājums, kurā bija veikta laika starpība aprēķināšanā starp lidojumiem un darbinieka grafikiem, kuros bija jāpārbauda uz sekojošiem temporālas algebras darbībām: sākts ar, beidzas ar vai iekļauj. Arī tā pat kā un 5. vaicājumā pārbaudīšana bija izmantots CASE WHEN, jo ar to ir saprotamāk un vieglāk analizēt ierakstus un vaicājuma pareizību.

Vaicājuma kods:

select LDGS.VARDS, LDGS.UZVARDS,  
 LDGS.DARBA\_POZICIJA,  
 LS.NOSAUKUMS,  
 case when (  
 ((LS.DATUMS\_NO - LDGS.STRADA\_NO) = interval '0' second and (LS.DATUMS\_LIDZ - LDGS.STRADA\_LIDZ) < interval '0' second)  
 ) then 'true' else 'false' end sakas\_ar,  
 case when ((LDGS.STRADA\_NO - LS.DATUMS\_NO) < interval '0' second and (LDGS.STRADA\_LIDZ - LS.DATUMS\_LIDZ) > interval '0' second) then 'true' else 'false' end ieklauj,  
 case when (  
 ((LS.DATUMS\_LIDZ - LDGS.STRADA\_LIDZ) = interval '0' second and (LDGS.STRADA\_NO - LS.DATUMS\_NO) < interval '0' second)  
 ) then 'true' else 'false' end beidzas\_ar  
from LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS LDGS  
 inner join LIDOJUMS\_SKATS LS on (DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID = LDGS.LIDOSTA\_ID or  
 DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID = LDGS.LIDOSTA\_ID)  
where (VALUE(LS).SAKTS\_AR(LDGS.STRADA\_NO, LDGS.STRADA\_LIDZ) = 1 or  
 VALUE(LS).BEIDZAS\_AR(LDGS.STRADA\_NO, LDGS.STRADA\_LIDZ) = 1 or  
 VALUE(LDGS).IEKLAUJ(LS.DATUMS\_NO, LS.DATUMS\_LIDZ) = 1) and  
 LDGS.DARBA\_POZICIJA like '%despetceris%'  
;

Vaicājuma rezultāts (att. 5.14):



Att. 5.14 Sestā vaicājuma pārbaude

Pēc pārbaudīšanas ir redzams, ka vaicājums strādā korekti.

# WINDOWING klona tabula un vaicājumi

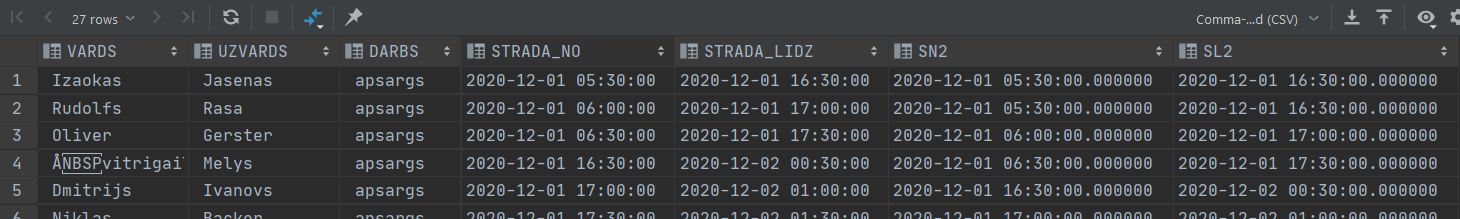
Lai pamēģināt izveidot klona tabulas windowing kontekstā, bija izveidoti vaicājumi, kuros tiek veikta OVER (ORDER BY) darbības.

Prieks pirmā klona tabulas vaicājuma bija paņemts pirmais vaicājums un izmainīts, lai ierakstu atlase tiek veikta tikai priekš apsargiem, kuriem ir nepieciešams atvērt vai ciet lidostas, pamatojoties uz darba grafika.

Vaicājuma kods:

select LDGS.VARDS as vards,  
 LDGS.UZVARDS as uzvards, LDGS.DARBA\_POZICIJA as darbs,  
 TO\_CHAR (LDGS.STRADA\_NO, 'YYYY-MM-DD HH24:MI:SS') strada\_no,  
 TO\_CHAR (LDGS.STRADA\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') strada\_lidz,  
 MIN(LDGS.STRADA\_NO) OVER (ORDER BY LDGS.STRADA\_NO ROWS 1 PRECEDING) as SN2,  
 MIN(LDGS.STRADA\_LIDZ) OVER (ORDER BY LDGS.STRADA\_LIDZ ROWS 1 PRECEDING) as SL2  
from LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS LDGS  
 inner join LIDOSTA\_GRAFIKS\_SKATS LGS on LDGS.LIDOSTA\_ID = LGS.LIDOSTA\_ID and LDGS.NEDELAS\_DIENA = LGS.NEDELAS\_DIENA  
where ((VALUE (LDGS).SAKTS\_AR(LGS.DATUMS\_NO, LGS.DATUMS\_LIDZ)) = 1 or (VALUE (LDGS).BEIDZAS\_AR(LGS.DATUMS\_NO, LGS.DATUMS\_LIDZ)) = 1)  
 and LDGS.DARBA\_POZICIJA like '%apsargs%'  
;

Vaicājuma rezultāts (att. 6.1):



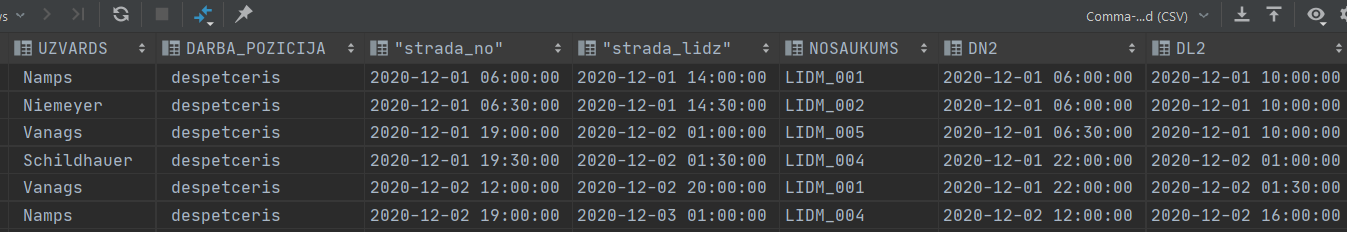
Att. 5.13 Pirmā WINDOWING vaicājuma rezultāts

Priekš otrā vaicājuma, bija izmantots sestā temporāla vaicājuma kods, bet tiek veikta pārbaude uz tādiem lidojumiem, kuri sakas ar vai beidzas ar dispečera darba grafiku.

Vaicājuma kods:

select LDGS.VARDS, LDGS.UZVARDS,  
 LDGS.DARBA\_POZICIJA,  
 TO\_CHAR (LDGS.STRADA\_NO, 'YYYY-MM-DD HH24:MI:SS') "strada\_no",  
 TO\_CHAR (LDGS.STRADA\_LIDZ, 'YYYY-MM-DD HH24:MI:SS') "strada\_lidz",  
 LS.NOSAUKUMS,  
 MIN(TO\_CHAR (LS.DATUMS\_NO, 'YYYY-MM-DD HH24:MI:SS')) OVER ( ORDER BY LS.DATUMS\_NO ROWS 1 PRECEDING) as DN2,  
 MIN(TO\_CHAR (LS.DATUMS\_LIDZ, 'YYYY-MM-DD HH24:MI:SS')) OVER ( ORDER BY LS.DATUMS\_LIDZ ROWS 1 PRECEDING) as DL2  
from LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS LDGS  
 inner join LIDOJUMS\_SKATS LS on (DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID = LDGS.LIDOSTA\_ID or  
 DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID = LDGS.LIDOSTA\_ID)  
where ((VALUE(LS).SAKTS\_AR(LDGS.STRADA\_NO, LDGS.STRADA\_LIDZ) = 1 and DEREF(LS.LIDOSTA\_NO).LIDOSTA\_ID = LDGS.LIDOSTA\_ID) or  
 (VALUE(LS).BEIDZAS\_AR(LDGS.STRADA\_NO, LDGS.STRADA\_LIDZ) = 1 and DEREF(LS.LIDOSTA\_UZ).LIDOSTA\_ID = LDGS.LIDOSTA\_ID)) and  
 LDGS.DARBA\_POZICIJA like '%despetceris%';  
;

Vaicājuma rezultāts (att. 6.2):



Att. 5.13 Otrā WINDOWING vaicājuma rezultāts

# Secinājumi

Laboratorijas darbā bija apskatīts un iemācīts veids, kā realizēt temporālas datubāzes Oracle vidē, izmantojot NESTED TABLE un atsauces (REF) relācijas realizēšanai starp tabulām un tās temporālas datnēm, kur priekš temporālas datnēm bija nepieciešams definēt objektu tipus un tās kolekcijas tipus, kā arī, LIDOSTA tabula arī bija veikta no objekta tipa, jo bija nepieciešams izmantot objektu atsauces lidojumos. Tiek izveidoti, izanalizēti temporālie vaicājumi, kuros bija veiktas tādas temporālas algebras darbības, ka OVERLAPS, STARTS, FINISHES, MEETS, PROCEDES, CONTAINS un ATLASE. Visas temporālas algebras darbības tiek realizētas objektu skatos datu tipos metodēs, kuros kontekstā bija izmantotas Java klases nodefinētas metodes, ar kuriem bija iespējams veikt nepieciešamus pārbaudījumus priekš temporālas algebras darbībām.

Datu ievades posmā bija izmantots INSERT vienkārša un PL/SQL vidē, kas palīdzēja saprast, kā jāveic loģiskas funkcijas blokus PL/SQL vidē un strādāt ar objekta tipiem PL/SQL vidē. Arī priekš datu ievadē bija izmantots SQL\*LOADER programmatūra ar kuru bija veiksmīgi ievadītas visas nepieciešamas datnes, bet izmantojot to, bija vairākas problēmas, kā piemērs, SQL\*LOADER nebija iespējams veikt lauku formatēšanu temporāliem datnes failiem, no kas, bija jāveic stingras pozīcijas priekš katriem datiem, kā arī bija patērēts laiks, lai saprast ka pareizāk ierakstīt un ievadīt TIMESTAMP datnes SQL\*LOADER vidē.

Pēc datnes ievadīšanas bija izveidotas objektu skati ar tās objektu tipiem, kas ir ļoti laba struktūra, jo nav nepieciešams veikt izmaiņas galvenās objektu tipos un tabulās, kā arī katrā reizē pārdefinēt metodes un veikt visas datu ievade no sākuma, jā kaut kāda metode bija kļūda, kā arī izmantojot objektu skatus ir iespējams veikt ātri un ērti jebkāda veida datnes “glabātuve” definējot tās laukus ar objekta tipa atribūtiem un izmantojot objekta metodes ātri un ērti veikt nepieciešamas loģikas darbības.

Pēc objektu skatu izveidošanas un tās, pēc kļūdas struktūrā, pārbaudīšanas, bija realizēti visi uzmīnēti darbā vaicājumi, kā arī bija veiktas papildu vaicājumi, ar kuriem bija veiktas tās darbošanās pareizība, kas rezultātā paradīja, ka vaicājumi bija izveidoti korekti un visa temporālas algebras loģika bija novērota. Vaicājumos, kuros bija veikta pārbaudīšana, bija izmantots vienkāršs SQL laika datnes salīdzinājums, kas savā starpā, salīdzinājāt ar jau izveidotiem metodēm PL/SQL objektu tipos, bija grūtāk saprast un kontrolēt tās loģiku.

Jā salīdzināt divus piegājienus, kā SQL valodā vienkāršāk un efektīvāk SQL valodā pierakstīt temporālos vaicājumus, tad manuprāt tas ir atkarīgs no situācijas, jo, piemērām, izmantojot vienkāršu loģiku temporālas algebras kontekstā, būtu efektīvāk un vienkāršāk salīdzināt visus laika datus SQL vaicājumā, neveidojot liekus objektus un metodes, kuri varētu būt nebūtu vajadzīgi kopējā sistēmā. Bet, jā lietot temporālas algebras darbības, salīdzinot savā starpā vairākās temporālas ierakstus no dažādas tabulās, tad būtu nepieciešams izmantot objektu skatus, kuriem ir vienkāršāk izveidot un pārdefinēt kopējo objekta struktūru, nekā kolekcijas objektu tipiem vai arī tabulas objektu tipiem, un tajās izveidot aprakstījušas temporālas darbības metodes, kuros būtu veiktas temporālas algebras darbības. Kā arī, jā ir vairākas vienādas temporālas algebras darbības metodes dažādas vietās, tad labāk nodefinēt galveno loģiku PL/SQL funkcijas kontekstā vai arī izmantojot funkcijas paketes, jo izmainīt vienā vietā būtu vieglāk nekā meklēt un mainīt visā kodā, bet, kā bija pamanīts, izmantojot Java klase ar Oracle pakete ir iespējams ātri un viegli salīdzināt laika vērtības, izmantojot jau nodefinētas metodes, priekš temporāla datnes salīdzināšanai. Kā arī jā izmantot Java klase, tad PL/SQL vidē nebūtu nepieciešams veikt lielas salīdzināšanas darbības, kas ir būtiski priekš temporālas algebras darbībām, no tā tiek atstāta tīra kopēja struktūra, kur katrs datubāzes slānis ir atbildīgs tikai par savu daļu.

Darbā beigās bija izveidotas vienkārši, pēc savā būtības, WINDOWING klona tabulas un tās vaicājumi, kuri bija balstītās uz jau izveidotiem temporāliem vaicājumiem.

Kopumā izpildījāt laboratorijas darbu bija saņemtas jaunas zināšanas par temporālo algebru darbībām un tās realizēšanas iespējam datubāzes kontekstā, kā arī, šāda veida zināšanas ir ļoti svarīgas, jo darboties ar temporālajiem datiem – tas ir viens no svarīgiem elementiem datubāzes kontekstā, jo veikt nepareizas datu izvade temporālajos datubāzes ir ļoti vienkārši, kas rezultātā varētu pievest pie lielām problēmām datu analizēšanas posmā.

# PIELIKUMI

## Pielikums

/\* INSERT LIDOSTA dati \*/  
begin  
 insert into LIDOSTA VALUES (1, 'Lidosta Riga', TABLE\_LIDOSTA\_GRAFIKS\_DATI\_TYPE (  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-01 6:00:00', TIMESTAMP '2020-12-02 1:00:00', 'otrdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-02 6:00:00', TIMESTAMP '2020-12-03 1:00:00', 'tresdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-03 6:00:00', TIMESTAMP '2020-12-04 1:00:00', 'ceturtdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-04 6:00:00', TIMESTAMP '2020-12-05 1:00:00', 'piektdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-05 6:30:00', TIMESTAMP '2020-12-06 0:30:00', 'sestdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-06 6:30:00', TIMESTAMP '2020-12-07 0:30:00', 'svetdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-07 6:00:00', TIMESTAMP '2020-12-08 1:00:00', 'pirmdiena')  
 ));  
  
 insert into LIDOSTA VALUES (2, 'Lidosta Vilnus', TABLE\_LIDOSTA\_GRAFIKS\_DATI\_TYPE (  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-01 5:30:00', TIMESTAMP '2020-12-02 0:30:00', 'otrdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-02 5:30:00', TIMESTAMP '2020-12-03 0:30:00', 'tresdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-03 5:30:00', TIMESTAMP '2020-12-04 0:30:00', 'ceturtdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-04 5:30:00', TIMESTAMP '2020-12-05 0:30:00', 'piektdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-05 6:30:00', TIMESTAMP '2020-12-06 0:00:00', 'sestdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-06 5:30:00', TIMESTAMP '2020-12-07 0:00:00', 'svetdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-07 5:30:00', TIMESTAMP '2020-12-08 0:30:00', 'pirmdiena')  
 ));  
  
 insert into LIDOSTA VALUES (3, 'Lidosta Berlin', TABLE\_LIDOSTA\_GRAFIKS\_DATI\_TYPE (  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-01 6:30:00', TIMESTAMP '2020-12-02 1:30:00', 'otrdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-02 6:30:00', TIMESTAMP '2020-12-03 1:30:00', 'tresdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-03 6:30:00', TIMESTAMP '2020-12-04 1:30:00', 'ceturtdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-04 6:30:00', TIMESTAMP '2020-12-05 1:30:00', 'piektdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-05 6:45:00', TIMESTAMP '2020-12-06 0:45:00', 'sestdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-06 6:45:00', TIMESTAMP '2020-12-07 0:45:00', 'svetdiena'),  
 LIDOSTA\_GRAFIKS\_DATI\_TYPE (TIMESTAMP '2020-12-07 6:30:00', TIMESTAMP '2020-12-08 1:30:00', 'pirmdiena')  
 ));  
end;

## Pielikums

/\* INSERT LIDMASINA dati \*/  
DECLARE  
 lidosta1 REF LIDOSTA\_TYPE;  
 lidosta2 REF LIDOSTA\_TYPE;  
 lidosta3 REF LIDOSTA\_TYPE;  
BEGIN  
 SELECT REF(l)  
 INTO lidosta1  
 FROM LIDOSTA l  
 WHERE LIDOSTA\_ID = 1;  
  
 SELECT REF(l)  
 INTO lidosta2  
 FROM LIDOSTA l  
 WHERE LIDOSTA\_ID = 2;  
  
 SELECT REF(l)  
 INTO lidosta3  
 FROM LIDOSTA l  
 WHERE LIDOSTA\_ID = 3;  
  
 INSERT INTO LIDMASINA VALUES (1, 'LIDM\_001', 'viss kartiba', TABLE\_LIDOJUMS\_DATI\_TYPE (  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 06:00:00', TIMESTAMP '2020-12-01 10:00:00', lidosta1, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 14:00:00', TIMESTAMP '2020-12-01 18:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 00:00:00', TIMESTAMP '2020-12-02 07:00:00', lidosta2, lidosta1),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 12:00:00', TIMESTAMP '2020-12-02 16:00:00', lidosta1, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 01:00:00', TIMESTAMP '2020-12-03 06:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 11:00:00', TIMESTAMP '2020-12-03 17:00:00', lidosta2, lidosta1),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 22:00:00', TIMESTAMP '2020-12-04 01:00:00', lidosta1, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-04 06:30:00', TIMESTAMP '2020-12-04 12:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-04 15:00:00', TIMESTAMP '2020-12-04 20:00:00', lidosta2, lidosta1)  
 ));  
 INSERT INTO LIDMASINA VALUES (2, 'LIDM\_002', 'viss kartiba', TABLE\_LIDOJUMS\_DATI\_TYPE (  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 06:30:00', TIMESTAMP '2020-12-01 10:00:00', lidosta3, lidosta1),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 14:00:00', TIMESTAMP '2020-12-01 18:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 20:00:00', TIMESTAMP '2020-12-02 0:00:00', lidosta2, lidosta3),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 06:30:00', TIMESTAMP '2020-12-02 16:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 19:00:00', TIMESTAMP '2020-12-02 23:30:00', lidosta2, lidosta1),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 06:00:00', TIMESTAMP '2020-12-03 10:30:00', lidosta1, lidosta3),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 18:00:00', TIMESTAMP '2020-12-03 23:00:00', lidosta1, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-04 07:30:00', TIMESTAMP '2020-12-04 11:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-04 16:00:00', TIMESTAMP '2020-12-04 21:30:00', lidosta2, lidosta1)  
 ));  
 INSERT INTO LIDMASINA VALUES (3, 'LIDM\_003', 'problemas ar kreiso sparni', TABLE\_LIDOJUMS\_DATI\_TYPE (  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 08:00:00', TIMESTAMP '2020-12-01 12:30:00', lidosta1, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 15:00:00', TIMESTAMP '2020-12-01 19:30:00', lidosta2, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 00:00:00', TIMESTAMP '2020-12-02 06:00:00', lidosta3, lidosta1),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 09:30:00', TIMESTAMP '2020-12-02 15:00:00', lidosta1, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 19:30:00', TIMESTAMP '2020-12-03 00:00:00', lidosta3, lidosta1),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-04 07:45:00', TIMESTAMP '2020-12-04 12:15:00', lidosta1, lidosta2),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-04 14:00:00', TIMESTAMP '2020-12-04 18:00:00', lidosta2, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-04 21:30:00', TIMESTAMP '2020-12-05 01:00:00', lidosta2, lidosta1),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-05 08:15:00', TIMESTAMP '2020-12-05 11:45:00', lidosta1, lidosta3)  
 ));  
 INSERT INTO LIDMASINA VALUES (4, 'LIDM\_004', 'viss kartiba', TABLE\_LIDOJUMS\_DATI\_TYPE (  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 08:15:00', TIMESTAMP '2020-12-01 12:30:00', lidosta1, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 15:00:00', TIMESTAMP '2020-12-01 19:00:00', lidosta2, lidosta1),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 22:00:00', TIMESTAMP '2020-12-02 01:30:00', lidosta1, lidosta3),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 07:00:00', TIMESTAMP '2020-12-02 11:30:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 13:30:00', TIMESTAMP '2020-12-02 16:45:00', lidosta2, lidosta1),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 19:00:00', TIMESTAMP '2020-12-02 23:30:00', lidosta1, lidosta3),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 06:30:00', TIMESTAMP '2020-12-03 11:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 14:30:00', TIMESTAMP '2020-12-03 17:00:00', lidosta2, lidosta1),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 21:00:00', TIMESTAMP '2020-12-04 00:00:00', lidosta1, lidosta2)  
 ));  
 INSERT INTO LIDMASINA VALUES (5, 'LIDM\_005', 'problemas ar labo sparni', TABLE\_LIDOJUMS\_DATI\_TYPE (  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 08:00:00', TIMESTAMP '2020-12-01 13:00:00', lidosta2, lidosta3),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 16:00:00', TIMESTAMP '2020-12-01 19:00:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-01 22:00:00', TIMESTAMP '2020-12-02 01:00:00', lidosta2, lidosta1),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 07:00:00', TIMESTAMP '2020-12-02 10:00:00', lidosta1, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 12:00:00', TIMESTAMP '2020-12-02 15:00:00', lidosta2, lidosta1),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 17:00:00', TIMESTAMP '2020-12-02 21:00:00', lidosta1, lidosta3),  
  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-02 22:00:00', TIMESTAMP '2020-12-03 00:30:00', lidosta3, lidosta2),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 05:30:00', TIMESTAMP '2020-12-03 09:00:00', lidosta2, lidosta1),  
 LIDOJUMS\_DATI\_TYPE (TIMESTAMP '2020-12-03 13:30:00', TIMESTAMP '2020-12-03 16:45:00', lidosta1, lidosta2)  
 ));  
END;

## Pielikums

import java.sql.\*;  
import oracle.sql.TIMESTAMP;  
  
public class KorolkoClass {  
  
 public static boolean prieksa (  
 TIMESTAMP t1,  
 TIMESTAMP t2  
 ) throws SQLException {  
 return t1.timestampValue().before(t2.timestampValue());  
 }  
  
 public static boolean vienads (  
 TIMESTAMP t1,  
 TIMESTAMP t2  
 ) throws SQLException {  
 return t1.timestampValue().equals(t2.timestampValue());  
 }  
  
 public static boolean dalejiNosegts (  
 TIMESTAMP t1\_no,  
 TIMESTAMP t1\_lidz,  
 TIMESTAMP t2\_no,  
 TIMESTAMP t2\_lidz  
 ) throws SQLException {  
 return t1\_no.timestampValue().before(t2\_no.timestampValue()) &&  
 t1\_lidz.timestampValue().after(t2\_no.timestampValue()) &&  
 t1\_lidz.timestampValue().before(t2\_lidz.timestampValue());  
 }  
  
 public static boolean ieklauj (  
 TIMESTAMP t1\_no,  
 TIMESTAMP t1\_lidz,  
 TIMESTAMP t2\_no,  
 TIMESTAMP t2\_lidz  
 ) throws SQLException {  
 return t1\_no.timestampValue().before(t2\_no.timestampValue()) &&  
 t1\_lidz.timestampValue().after(t2\_lidz.timestampValue());  
 }  
  
}

## Pielikums

/\* LIDOSTAS GRAFIKA SKATA TIPS \*/  
create or replace type LIDOSTA\_GRAFIKS\_SKATS\_TYPE as object (  
 lidosta\_id number,  
 nosaukums varchar2(255),  
 datums\_no TIMESTAMP,  
 datums\_lidz TIMESTAMP,  
 nedelas\_diena varchar2(24),  
 order member function *SORT\_BY\_DATUMS\_NO* (grafiks LIDOSTA\_GRAFIKS\_SKATS\_TYPE) return number,  
 member function *DALEJI\_NOSEGT* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number,  
 member function *IR\_PRIEKSA* (obj\_no in TIMESTAMP) return number  
);  
/  
create or replace type body LIDOSTA\_GRAFIKS\_SKATS\_TYPE as  
 order member function *SORT\_BY\_DATUMS\_NO* (grafiks LIDOSTA\_GRAFIKS\_SKATS\_TYPE) return number is  
 begin  
 if (JAVA\_FUNKCIJAS\_PAKETE.*PRIEKSA*(SELF.DATUMS\_NO, grafiks.DATUMS\_NO)) then  
 return -1;  
 end if;  
 return 1;  
 end;  
 member function *DALEJI\_NOSEGT* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number is  
 begin  
 if (JAVA\_FUNKCIJAS\_PAKETE.*DALEJINOSEGTS*(SELF.DATUMS\_NO, SELF.DATUMS\_LIDZ, obj\_no, obj\_lidz)) then  
 return 1;  
 end if;  
 return 0;  
 end;  
 member function *IR\_PRIEKSA* (obj\_no in TIMESTAMP) return number is  
 begin  
 if (JAVA\_FUNKCIJAS\_PAKETE.*PRIEKSA*(SELF.DATUMS\_LIDZ, obj\_no)) then  
 return 1;  
 end if;  
 return 0;  
 end;  
end;  
/  
  
/\* LIDOSTAS DARBINIEKS GRAFIKA SKATA TIPS \*/  
create or replace type LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS\_TYPE as object (  
 lidosta\_darbinieks\_id number,  
 vards varchar2(255),  
 uzvards varchar2(255),  
 status varchar2(255),  
 darba\_pozicija varchar2(255),  
 lidosta\_id number,  
 strada\_no TIMESTAMP,  
 strada\_lidz TIMESTAMP,  
 nedelas\_diena varchar2(24),  
 order member function *SORT\_BY\_STRADA\_NO* (grafiks LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS\_TYPE) return number,  
 member function *SAKTS\_AR* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number,  
 member function *BEIDZAS\_AR* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number,  
 member function *DALEJI\_NOSEGT* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number,  
 member function *IEKLAUJ* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number  
);  
/  
  
create or replace type body LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS\_TYPE as  
 order member function *SORT\_BY\_STRADA\_NO* (grafiks LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS\_TYPE) return number is  
 begin  
 if (JAVA\_FUNKCIJAS\_PAKETE.*PRIEKSA*(SELF.STRADA\_NO, grafiks.STRADA\_NO)) then  
 return -1;  
 end if;  
 return 1;  
 end;  
 member function *SAKTS\_AR* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number is  
 begin  
 if ((JAVA\_FUNKCIJAS\_PAKETE.*VIENADS*(SELF.STRADA\_NO, obj\_no)) and  
 (JAVA\_FUNKCIJAS\_PAKETE.*PRIEKSA*(SELF.STRADA\_LIDZ, obj\_lidz))  
 ) then  
 return 1;  
 end if;  
 return 0;  
 end;  
 member function *BEIDZAS\_AR* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number is  
 begin  
 if ((JAVA\_FUNKCIJAS\_PAKETE.*VIENADS*(SELF.STRADA\_LIDZ, obj\_lidz)) and  
 (JAVA\_FUNKCIJAS\_PAKETE.*PRIEKSA*(obj\_no, SELF.STRADA\_NO))  
 ) then  
 return 1;  
 end if;  
 return 0;  
 end;  
 member function *DALEJI\_NOSEGT* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number is  
 begin  
 if ((JAVA\_FUNKCIJAS\_PAKETE.*DALEJINOSEGTS* (  
 SELF.STRADA\_NO, SELF.STRADA\_LIDZ, obj\_no, obj\_lidz  
 ))) then  
 return 1;  
 end if;  
 return 0;  
 end;  
 member function *IEKLAUJ* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number is  
 begin  
 if ((JAVA\_FUNKCIJAS\_PAKETE.*IEKLAUJ* (  
 SELF.STRADA\_NO, SELF.STRADA\_LIDZ, obj\_no, obj\_lidz  
 ))) then  
 return 1;  
 end if;  
 return 0;  
 end;  
end;  
/  
  
create or replace type LIDOJUMS\_SKATS\_TYPE as object (  
 lidmasina\_id number,  
 nosaukums varchar2(255),  
 status varchar2(255),  
 datums\_no TIMESTAMP,  
 datums\_lidz TIMESTAMP,  
 lidosta\_no REF LIDOSTA\_TYPE,  
 lidosta\_uz REF LIDOSTA\_TYPE,  
 order member function *SORT\_BY\_DATUMS\_NO* (lidojums LIDOJUMS\_SKATS\_TYPE) return number,  
 member function *SAKTS\_AR* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number,  
 member function *BEIDZAS\_AR* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number,  
 member function *DALEJI\_NOSEGT* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number,  
 member function *SATIEK* (obj\_no in TIMESTAMP) return number  
);  
/  
  
create or replace type body LIDOJUMS\_SKATS\_TYPE as  
 order member function *SORT\_BY\_DATUMS\_NO* (lidojums LIDOJUMS\_SKATS\_TYPE) return number is  
 begin  
 if (JAVA\_FUNKCIJAS\_PAKETE.*PRIEKSA*(SELF.DATUMS\_NO, lidojums.DATUMS\_NO)) then  
 return -1;  
 end if;  
 return 1;  
 end;  
 member function *SAKTS\_AR* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number is  
 begin  
 if ((JAVA\_FUNKCIJAS\_PAKETE.*VIENADS*(SELF.DATUMS\_NO, obj\_no)) and  
 (JAVA\_FUNKCIJAS\_PAKETE.*PRIEKSA*(SELF.DATUMS\_LIDZ, obj\_lidz))) then  
 return 1;  
 end if;  
 return 0;  
 end;  
 member function *BEIDZAS\_AR* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number is  
 begin  
 if ((JAVA\_FUNKCIJAS\_PAKETE.*VIENADS*(SELF.DATUMS\_LIDZ, obj\_lidz)) and  
 (JAVA\_FUNKCIJAS\_PAKETE.*PRIEKSA*(obj\_no, SELF.DATUMS\_NO))) then  
 return 1;  
 end if;  
 return 0;  
 end;  
 member function *DALEJI\_NOSEGT* (obj\_no in TIMESTAMP, obj\_lidz in TIMESTAMP) return number is  
 begin  
 if ((JAVA\_FUNKCIJAS\_PAKETE.*DALEJINOSEGTS* (SELF.DATUMS\_NO, SELF.DATUMS\_LIDZ, obj\_no, obj\_lidz))) then  
 return 1;  
 end if;  
 return 0;  
 end;  
 member function *SATIEK* (obj\_no in TIMESTAMP) return number is  
 begin  
 if (JAVA\_FUNKCIJAS\_PAKETE.*VIENADS*(SELF.DATUMS\_LIDZ, obj\_no)) then  
 return 1;  
 end if;  
 return 0;  
 end;  
end;  
/

## Pielikums

create or replace view LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS of LIDOSTA\_DARBINIEKS\_GRAFIKS\_SKATS\_TYPE  
 with object identifier(strada\_no, strada\_lidz) as  
select LD.LIDOSTA\_DARBINIEKS\_ID, LD.VARDS, LD.UZVARDS,  
 LD.STATUS, LD.DARBA\_POZICIJA, LD.LIDOSTA\_ID,  
 LDG.STRADA\_NO, LDG.STRADA\_LIDZ, LDG.NEDELAS\_DIENA  
from LIDOSTA\_DARBINIEKS LD, TABLE(LD.GRAFIKI) LDG;  
  
create or replace view LIDOSTA\_GRAFIKS\_SKATS of LIDOSTA\_GRAFIKS\_SKATS\_TYPE  
 with object identifier(datums\_no, datums\_lidz) as  
select L.LIDOSTA\_ID, NOSAUKUMS,  
 LG.DATUMS\_NO, LG.DATUMS\_LIDZ, LG.NEDELAS\_DIENA  
from LIDOSTA L, TABLE(L.GRAFIKI) LG;  
  
create or replace view LIDOJUMS\_SKATS of LIDOJUMS\_SKATS\_TYPE  
 with object identifier(datums\_no, datums\_lidz) as  
select L.LIDMASINA\_ID, NOSAUKUMS, STATUS,  
 LI.DATUMS\_NO, LI.DATUMS\_LIDZ, LI.LIDOSTA\_NO, LI.LIDOSTA\_UZ  
from LIDMASINA L, TABLE(L.LIDOJUMI) LI;