VIDZEMES AUGSTSKOLA

INŽENIERZINĀTŅU FAKULTĀTE

VILCIENU MARŠRUTU DATUBĀŽU SISTĒMAS PIELIETOJUMS APLIKĀCIJĀ

KURSA DARBS

Autors: Artūrs Gailītis

Stud. Apl. Nr.: IT23005

Valmiera, 2025

# SATURS

[SATURS 2](#_Toc198467134)

[IEVADS 3](#_Toc198467135)

[1. TEORĒTISKAIS APRAKSTS 4](#_Toc198467136)

[1.1 ORACLE APEX APRAKSTS 4](#_Toc198467137)

[1.2 INFORMĀCIJA PAR SQL VALODU 4](#_Toc198467138)

[1.3 ER DIAGRAMMAS 5](#_Toc198467139)

[1.4 GTFS FORMĀTS 6](#_Toc198467140)

[2. PAR DATIEM 7](#_Toc198467141)

[2.1 TABULU DATU UZBŪVE 7](#_Toc198467142)

[2.2 TABULU ATSLĒGAS 9](#_Toc198467143)

[2.3 SEKVENCES 10](#_Toc198467144)

[2.4 TRIGERI 10](#_Toc198467145)

[2.5 PROCEDŪRAS 11](#_Toc198467146)

[2.6 REST DATU AVOTS 14](#_Toc198467147)

[2.7 ER DIAGRAMMA 15](#_Toc198467148)

[3. LIETOTNE ORACLE VIDĒ 16](#_Toc198467149)

[SECINĀJUMI 23](#_Toc198467150)

[SAĪSINĀJUMI UN ATSLĒGVĀRDI 24](#_Toc198467151)

[LITERATŪRAS SARAKSTS 25](#_Toc198467152)

[PIELIKUMS 26](#_Toc198467153)

[IZVEIDOŠANAS SKRIPTS 26](#_Toc198467154)

[IZDZĒŠANAS SKRIPTS 30](#_Toc198467155)

# IEVADS

Mūsdienās viena no risinājumiem, kā cilvēki pārvietojās vidēji lielus attālumus ir pasažieru vilciens. Lai plānotu ērtu un efektīvu nokļūšanu galamērķi, ir nepieciešams pareiza un viegli saprotama dzelzceļa satiksmes informācija. Piemēram, no kurienes uz kurieni vai arī kurās dienās vilciens kursēs.

Kursa darba aplikācija ir izstrādāta Oracle Apex datubāžu vidē ar mazliet HTML palīdzību. Aplikācijas izmantotie dati tika iegūti no Latvijas atvērtajiem datiem mājaslapā: "Iekšzemes dzelzceļa vilcienu kustības saraksts GTFS formātā”.

**Problēma:** GTFS formātā izveidotā vilcienu kustību saraksts nav pasažieriem viegli saprotams. Topošajiem pasažieriem, kas mēģina ieplānot savu maršruta gaitu, ir nepieciešams, lai vilcienu kustības informācija būtu viegli saprotama, tā lai visa vecuma spētu orientēties tajā.

**Mērķis:** Izmantot Oracle Apex datubāžu objektus, piemēram, indeksus, sekvenci, procedūras, trigerus, lai izveidotu vilciena kustības saraksta aplikāciju. Kurā var apskatīt vilciena kustību, kā arī izveidot jaunu vai pārmainīt esošu kustību priekš administratoriem.

**Darba uzdevumi:**

* Apskatīt vajadzīgo teoriju;
* Ielādēt datu kopas Oracle vidē;
* No dotajā datu tabulām izveidot datubāžu objektus;
* Izstrādāt aplikāciju.

**Izmantotās metodes:**

* Literatūras analīze;
* SQL programmēšanas valoda;
* HTML mājas lapu veidošanas valoda.

# 1. TEORĒTISKAIS APRAKSTS

## 1.1 ORACLE APEX APRAKSTS

Oracle APEX, jeb Oracle Application Express ir Oracle kompānijas izveidotā datubāžu izstrādes rīks. Šajā modernajā izstrādes platformā, tās lietotāji viegli un ātri izveidot, importēt datu tabulas. kuras var likt iekšā šajā pašā programmas izveidotajās aplikācijās.

Kā viena no priekšrocībām atšķirībā no citām SQL programmām ir tāda ka lietotājam iepriekš nevajag zināt programmēšanas valodas. Tādējādi Oracle APEX piedāvā izstrādes rīka lietotājiem draudzīgu grafisko vidi. Papildus tam, lai lietotu šo programmu nevajag lādēt savā datorā specifisku lietotni, jo APEX ir pieejams ikvienā interneta pārlūkprogrammā.

Integrācija ar citām Oracle datubāzēm ir arī priekšrocība, ko piedāvā Oracle APEX. Tas nozīmē, ka piekļūt vajadzīgajiem datiem ir diezgan ātri un elementāri. Ja iegādājās maksas versiju, tad arī var ielikt un tos apstrādāt liela izmēra datu kopas. Taču, ja izmanto brīvo versiju, tad lietotājs var, tik un tā, ar SQL vaicājumiem apstrādāt un vizuāli parādīt šos datus mazākā izmērā.

## 1.2 INFORMĀCIJA PAR SQL VALODU

SQL jeb Structured Query Language ir plaši izmantota programmēšanas valoda, priekš datubāžu izmantošanas vajadzībām. Ar šo valodu var izveidot, apstrādāt, pārvaldīt un izdzēst datubāzes un tajā esošās tabulas un datus. Šo datu modificēšanas un pārvaldīšanas valodu atbalsta dažādas datubāžu pārvaldes sistēmas. Piemēram, MySQL, PostgreSQL, kā arī Oracle. Tā rezultātā katrai pārvaldes sistēmā mazliet atšķiras SQL valodas sintakse.

Šī valoda ir speciāli izstrādāta, lai strādātu ar dažādām datubāzēm, neskatoties uz to izmēru. Šajās izveidotās datubāzēs, dati tiek glabāti tabulās, kurās tie ir izkārtoti pa rindām un kolonnām. Ar šo datubāžu programmēšanas valodu, var izveidot jaunu datu ierakstu, rediģēt individuāli vai arī kopēji esošos datus, kā arī tos izdzēst, ja lietotājam vairs nevajag tos. Vel viena funkcija ir apskatīt jebkādā situācijā datus. Kā galvenās komandas SQL ir SELECT, UPDATE, INSERT un DELETE.

SELECT komandu izmanto, lai iegūtu specifisku ierakstu no datubāžu tabulām. Šī komanda neietekmē tabulu ierakstus un to struktūru. Piemēram, šis vaicājums “SELECT \* FROM ceļi” nodrošina ka lietotājam tiks parādīti visi datu ieraksti no tabulas ”ceļi”. Papildus var pielikt komandu WHERE, kas šajā gadījumā parādīs visus datus, kuri filtrēšanas gadījumā sakritīs ar WHERE vaicājumu. Piemēram, šis vaicājums “SELECT \* FROM mašīnas WHERE krāsa = ‘sarkana’” parādīs visus tabulas “mašīnas” ierakstus kuros krāsa būs sarkana.

Ar UPDATE komandu var rediģēt un izmainīt datu ierakstus. Izmantojot nosacījumus, lietotājs varēs izmainīt konkrētu datu ierakstu. Piemēram, šis vaicājums “UPDATE ceļi SET bedres = true WHERE id = 2” nodrošinās ka tabulā “ceļi” bedres kolonnā pārveidos vērtību uz true vai 1, ja tam id būs 2.

INSERT komandu izmanto lai izveidotu jaunus objektus datubāzē, vai arī izveidotu jaunu datubāzi. Piemēram šis vaicājums “INSERT INTO ceļi (id, bedres) VALUES (10, false)” nodrošinās ka tabulā “ceļi” tiks pievienota jauns ieraksts par id un bedrēm kur pirmais ir 10 un otrais ir false.

DELETE komandu izmanto lai izdzēstu nevajadzīgos ierakstus no datubāzes tabulas. Piemēram, šis vaicājums “DELETE FROM ceļi WHERE id = 1” nodrošinās ka tabulā “ceļi” tiks izdzēsts ieraksts kurā id ir 1.

## 1.3 ER DIAGRAMMAS

ER diagramma, jeb Entity-Relationship diagramma ir viens no veidiem, kā grafiski var apskatīt un secināt tabulu attiecības vienu pret otru. Šīs diagrammas palīdzību var novērtēt un saprast esošās tabulas, pievienot datubāzē pareizas tabulas, kā arī, ja rodas problēmas, tās novērst.

ER diagrammas esošie elementi:

* **Entitāte**: tabula vai tabulas grupa, kura ir vajadzīga datubāžu vidē. Tā ir attēlota, kā taisnstūris. Katras nosaukums ir unikāls datubāzē;
* **Atribūti**: parāda un raksturo tabulu parametrus jeb kolonnas. Tām ir unikāls nosaukums, kas nevar atkārtoties esošajā tabulā, kā arī datu tips un citi parametri;
* **Atslēgas**: nosaka ar kuru kolonnu tabula tiek savienota ar citu tabulas kolonnu. Tās var būt Primārās, Svešās vai Unikāla atslēga;
* **Attiecības:** savieno divas vai vairākas tabulas. Ir trīs attiecību veidi: viens pret vienu, viens pret daudziem un daudzi pret daudziem. Šīs attiecības tiek apzīmētas ar rombu un līniju.

## 1.4 GTFS FORMĀTS

GTFS jeb General Transit Feed Specification formāts ir sabiedrisko transportu datu apmaiņas standartizēts formāts. Šo formātu izveidoja Google, lai vienotu pieeju apmaiņai transporta informācijai.

Kā struktūra GTFS formāts parasti ir CSV failu kopās, kas ir saspiesti, tā lai būtu patērētu mazāk vietas datu transportēšanai ZIP failos. Pārsvarā šie faila satur informāciju par maršrutiem, pieturām, kustības grafikiem un transporta līdzekļiem utt.

Šos datus izmanto sabiedriskā transporta aģentūras, lai autoostas vai stacijas varētu publicēt šos datus. Tā lai sabiedrisko transporta lietotāji varētu būt pieeja reāllaika informācija par autobusa vai vilciena kustībām. Kura rezultātā lietotāji varētu izplānot savu laiku un ceļu.

# 

# 2. PAR DATIEM

Darba procesa laikā bija nepieciešams izvērtēt kuras tabulas izvēlēties no ZIP failā ieliktajām tabulām. Izvēlētās galvenās 3 tabulas no kurām taisīs šo kursa darbu tika ņemta no Latvijas atvērto datu portāla, bet pārējās 4 ir taisītas ar procedūras palīdzību. Tātad kopumā priekš šī darba ir izveidotas 7 tabulas.

## 2.1 TABULU DATU UZBŪVE

Tabulas datu uzbūve pārsvarā nav mainīta. Tās ir atstātas tāda pašas, kā tās ir ieliktas Latvijas atvērto datu portālā.

**Tabula 2.1.1** Datu struktūra “Braucieni” tabulā.

| Kolunnu nosaukums | Datu tips |
| --- | --- |
| ID | Number |
| Route\_ID | Number |
| Service\_ID | Varcha2r(50 Byte) |
| Trip\_ID | Number |
| Trip\_Headsign | Varchar2(50 Byte) |

**Tabula 2.1.2** Datu struktūra “Kalendārs” tabulā.

| Kolunnu nosaukums | Datu tips |
| --- | --- |
| ID | Number |
| Service\_ID | Varchar2(50 Byte) |
| Monday | Number |
| Tuesday | Number |
| Wednesday | Number |
| Thursday | Number |
| Friday | Number |
| Saturday | Number |
| Sunday | Number |
| Start\_Day | Number |
| End\_Day | Number |

**Tabula 2.1.3** Datu struktūra “Maršruti” tabulā.

| Kolunnu nosaukums | Datu tips |
| --- | --- |
| ID | Number |
| Route\_ID | Number |
| Route\_Short\_Name | Varchar2(4000 Byte) |
| Route\_Long\_Name | Varchar2(100 Byte) |
| Route\_Desc | Varchar2(4000 Byte) |
| Route\_Type | Number |
| Route\_Url | Varchar2(4000 Byte) |
| Route\_Color | Varchar2(50 Byte) |
| Route\_Text\_Color | Number |

**Tabula 2.1.4:** Datu struktūra “Dienu\_skaits” ko izveido procedūra “dienu\_skaitu\_saskaitisana”

| Kolunnu nosaukums | Datu tips |
| --- | --- |
| ID | Number |
| Diena | Varchar2(20 Byte) |
| Pieejami | Number |
| Nepieejami | Number |

## 2.2 TABULU ATSLĒGAS

Veidojot šo darbu, galvenajām tabulām tika izstrādāti indeksi jeb atslēgas. Importējot datus no Latvijas atvērto datu portāla uz Oracle APEX, automātiski tika izveidoti indeksi, kuri pārbauda vai “id” ir unikāls un to uztaisa, kā Primāro jeb Galveno atslēgu. Tad papildus tika uztaisīti priekš “Service\_ID” un “Route\_ID” kā Unikālu atslēgu, tā lai katrs id būtu unikāls un to varētu savienot ar citām tabulām.

**Tabula 2.2.1** Izveidoto tabulu indeksi.

| Indeksu nosaukumi | Indeksu tips | Tabula |
| --- | --- | --- |
| ServiceID\_Con | Unique | Braucieni |
| SYS\_C00176316784 | Primary | Braucieni |
| Routes\_ID\_F\_Con | Foreign | Braucieni |
| ServiceID\_F\_Con | Foreign | Kalendārs |
| SYS\_C00176316756 | Primary | Kalendārs |
| SYS\_C00176316764 | Primary | Maršruti |
| RoutesID\_Con | Unique | Maršruti |

## 

## 2.3 SEKVENCES

Šajā darbā tika radītas divas sekvences - “Braucieni\_SEQ” un “Maršruti\_SEQ” kuru mērķis ir procedūru laikā lietotājam nevajag rakstīt vai nu “Trip\_ID” vai arī “Route\_ID”. Jo tas tiks uzrakstīts lietotāja vietā.

Tika izveidots “Braucieni\_SEQ” ar šādu kommandu:

CREATE SEQUENCE "Braucieni\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 75663 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL ;

Tika izveidots “Maršruti\_SEQ” ar šādu kommandu:

CREATE SEQUENCE "Maršruti\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 10 START WITH 1800 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL ;

## 2.4 TRIGERI

Priekš šī darba tika izveidoti divi trigeri, kuru mērķis ir atvieglot aplikācijas lietotāju. Tas nozīmē, ka lietotājam nav obligāti jāraksta konkrētais id, jo ja lietotājs rakstot jaunu ierakstu neieraksta, piemēram, “Route\_ID” un izpilda SQL vaicājumu. Tad automātiski, ar sekvences palīdzību, tiks uzģenerēts id numurs.

Tika izveidots “Braucieni\_TRG” priekš “Trip\_ID”:

create or replace trigger "Braucieni\_TRG"

before insert on BRAUCIENI

for each row

begin

if :NEW."TRIP\_ID" is null then

select "Braucieni\_SEQ".nextval into :NEW."TRIP\_ID" from sys.dual;

end if;

end;

/

Tika izveidots “Maršruti\_TRG” priekš “Route\_ID”:

create or replace trigger "Maršruti\_TRG"

before insert on MARŠRUTI

for each row

begin

if :NEW."ROUTE\_ID" is null then

select "Maršruti\_SEQ".nextval into :NEW."ROUTE\_ID" from sys.dual;

end if;

end;

/

## 2.5 PROCEDŪRAS

Šajā darbā tika izveidoti divas procedūras. Pirmā no tām ir “Sadalit\_Marsrutus”. Šīs procedūras mērķis ir sadalīt “Maršruti” tabulu skatoties pēc ierakstu “Route\_ID”. Piemēram tiek izveidots viena no jaunajām tabulām “Maršruti\_ar\_1000\_ID”. Šajā tabulā tiks ielikti visi braucieni, kuriem “Route\_ID” ir no 1000 līdz pat 1999.

Tika izveidota procedūra “Sadalit\_Marsrutus” ar šādu kommandu:

create or replace PROCEDURE Sadalit\_Marsrutus IS

i NUMBER := 1000;

step NUMBER := 1000;

max\_route NUMBER;

sql\_text VARCHAR2(4000);

drop\_sql VARCHAR2(4000);

BEGIN

SELECT MAX(ROUTE\_ID) INTO max\_route FROM MARŠRUTI;

WHILE i <= max\_route LOOP

drop\_sql := 'BEGIN ' ||

' EXECUTE IMMEDIATE ''DROP TABLE "MARŠRUTI\_AR\_' || i || '\_ID"''; ' ||

'EXCEPTION ' ||

' WHEN OTHERS THEN ' ||

' IF SQLCODE != -942 THEN RAISE; END IF; ' ||

'END;';

EXECUTE IMMEDIATE drop\_sql;

sql\_text := 'CREATE TABLE "MARŠRUTI\_AR\_' || i || '\_ID" AS ' ||

'SELECT \* FROM MARŠRUTI WHERE ROUTE\_ID BETWEEN ' || i || ' AND ' || (i + step - 1);

EXECUTE IMMEDIATE sql\_text;

i := i + step;

END LOOP;

END Sadalit\_Marsrutus;

/

Tad lai šī procedūra izpildās, tiek izmantota šī kommanda:

BEGIN

Sadalit\_Marsrutus;

END;

/

Otrās procedūras - “dienu\_skaitu\_saskaitisana” mērķis ir no dotā “kalendārs‘ tabulas saskaitīt cik katrā dienā ir pieejams maršruts un cik ir šīs dienas ir nepieejamas. Kad šo procedūru izpilda, tad tiek uztaisīta jauna tabula - “dienu\_skaits”, kuru vajadzēs izmantot priekš RESTful Services modeļa izveidei.

Tika izveidota procedūra “dienu\_skaitu\_saskaitisana” ar šādu kommandu:

create or replace PROCEDURE dienu\_skaitu\_saskaitisana AS

TYPE dienas\_tips IS TABLE OF VARCHAR2(20);

dienas dienas\_tips := dienas\_tips('MONDAY', 'TUESDAY', 'WEDNESDAY', 'THURSDAY', 'FRIDAY', 'SATURDAY', 'SUNDAY');

sql\_text VARCHAR2(4000);

drop\_sql VARCHAR2(4000);

BEGIN

drop\_sql := 'BEGIN ' ||

' EXECUTE IMMEDIATE ''DROP TABLE DIENU\_SKATS''; ' ||

'EXCEPTION ' ||

' WHEN OTHERS THEN ' ||

' IF SQLCODE != -942 THEN RAISE; END IF; ' ||

'END;';

EXECUTE IMMEDIATE drop\_sql;

sql\_text := 'CREATE TABLE DIENU\_SKATS (

ID NUMBER GENERATED ALWAYS AS IDENTITY,

DIENA VARCHAR2(20),

PIEEJAMI NUMBER,

NEPIEEJAMI NUMBER

)';

EXECUTE IMMEDIATE sql\_text;

FOR i IN 1..dienas.COUNT LOOP

sql\_text := 'INSERT INTO DIENU\_SKATS (DIENA, PIEEJAMI, NEPIEEJAMI) ' ||

'SELECT ''' || dienas(i) || ''', ' ||

'SUM(CASE WHEN ' || dienas(i) || ' = 1 THEN 1 ELSE 0 END), ' ||

'SUM(CASE WHEN ' || dienas(i) || ' = 0 THEN 1 ELSE 0 END) ' ||

'FROM KALENDĀRS';

EXECUTE IMMEDIATE sql\_text;

END LOOP;

END dienu\_skaitu\_saskaitisana;

/

Tad lai šī procedūra izpildās, tiek izmantota šī kommanda:

BEGIN

dienu\_skaitu\_saskaitisana;

END;

/

## 2.6 REST DATU AVOTS

Izmantojot procedūras izveido tabulu “DIENU\_SKAITS” tika izveidota RESTful datu servisa modelis.

Šajā modelī tika ielikts vaicājums:

select

ID,

DIENA,

PIEEJAMI,

NEPIEEJAMI

from DIENU\_SKATS

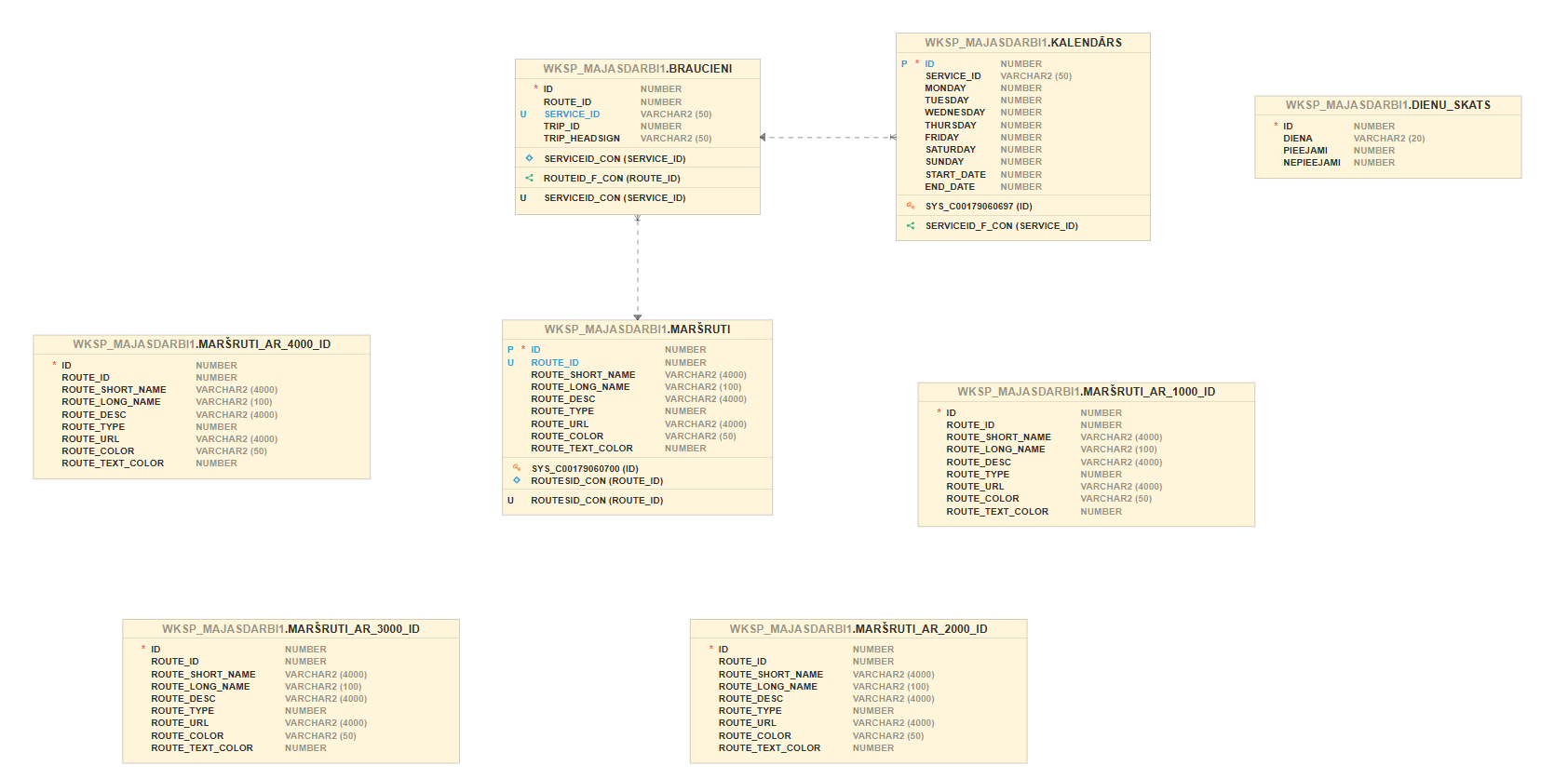
No datu servisa modeļa tika izveidots URL links - “https://apex.oracle.com/pls/apex/majasdarbi/diena/pieejamiba”, kurš ir JSON dokuments. Šajā JSON dokumentā atrodas visi tabulas dati. Tad, ar šo URL linku, izveidota lietotnes REST datu avots - “dienu pieejamiba”. Šis avots ir domāts, lai izveidotu lietotnē divas pie tipa diagrammas par pieejamām un nepieejamām dienām.

## 

## 2.7 ER DIAGRAMMA

Lai vizuāli parādītu datubāžu tabulu, tika izveidota ER diagramma. Šajā diagrammā ir parādīti gan tabulas, gan arī to attiecības. Skatoties šīs attiecības var secināt, ka visas attiecības ir viens pret daudziem (1:N). Taču piecas tabulas, kuras tika veidotas ar procedūras palīdzību, nav atzīmētas nekādas attiecības.

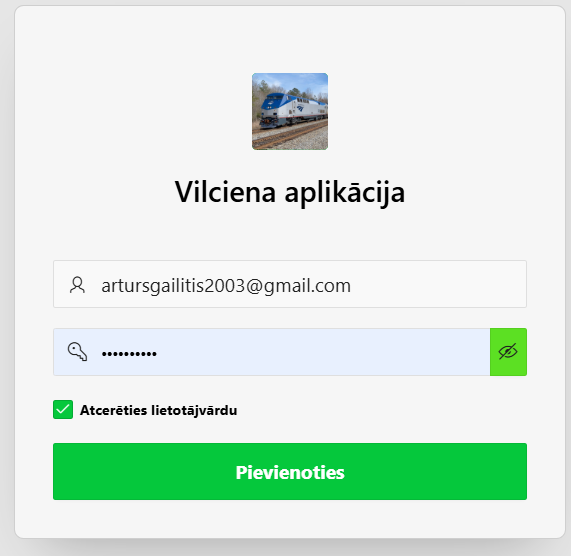
**Attēls 2.7.1:** ER diagramma.



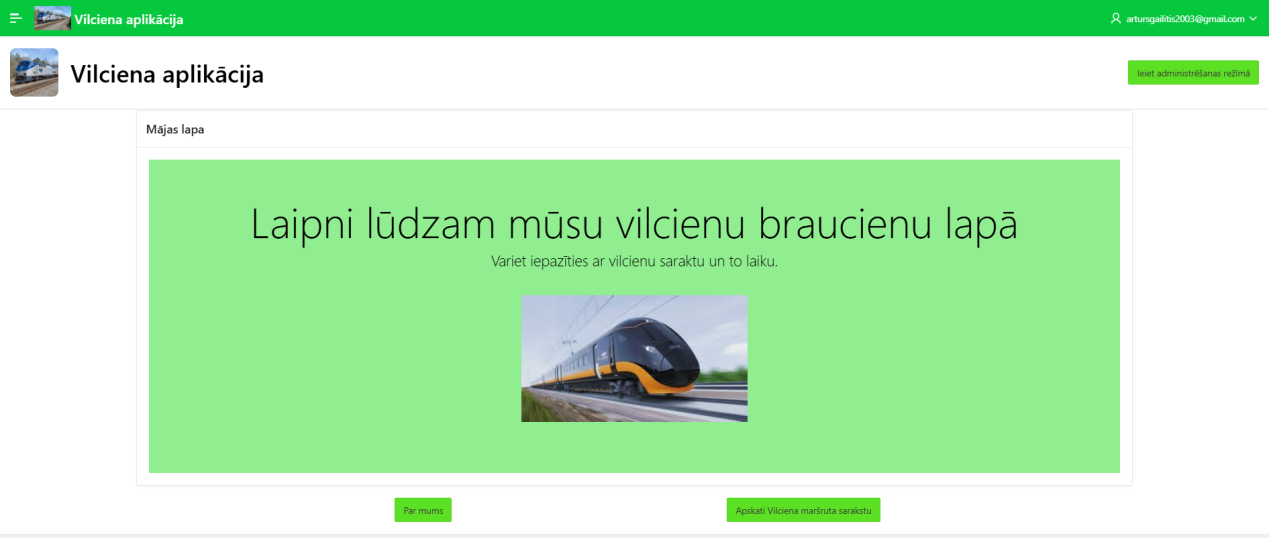
# 3. LIETOTNE ORACLE VIDĒ

Izstrādājot šo lietotni, tika izveidotas astoņas lapas. Trīs no viņām satur tabulas ar datiem. Tikai divas no tām var ietekmēt pašus datus. Viena lapa ir galvenā lapa ko ielādē vispirms. Tad viena lapa ir informācija par lietotni. Vel viena lapa ir par pieteikšanos pašā lietotnē. Un viena ir pieteikšanās administrācijas sadaļai, kurā var ietekmēt datu kopas. Pēdējā lapa ir izmantojot REST datu avotu, tiek parādīti diagrammas, par pieejamām un nepieejamām dienām.

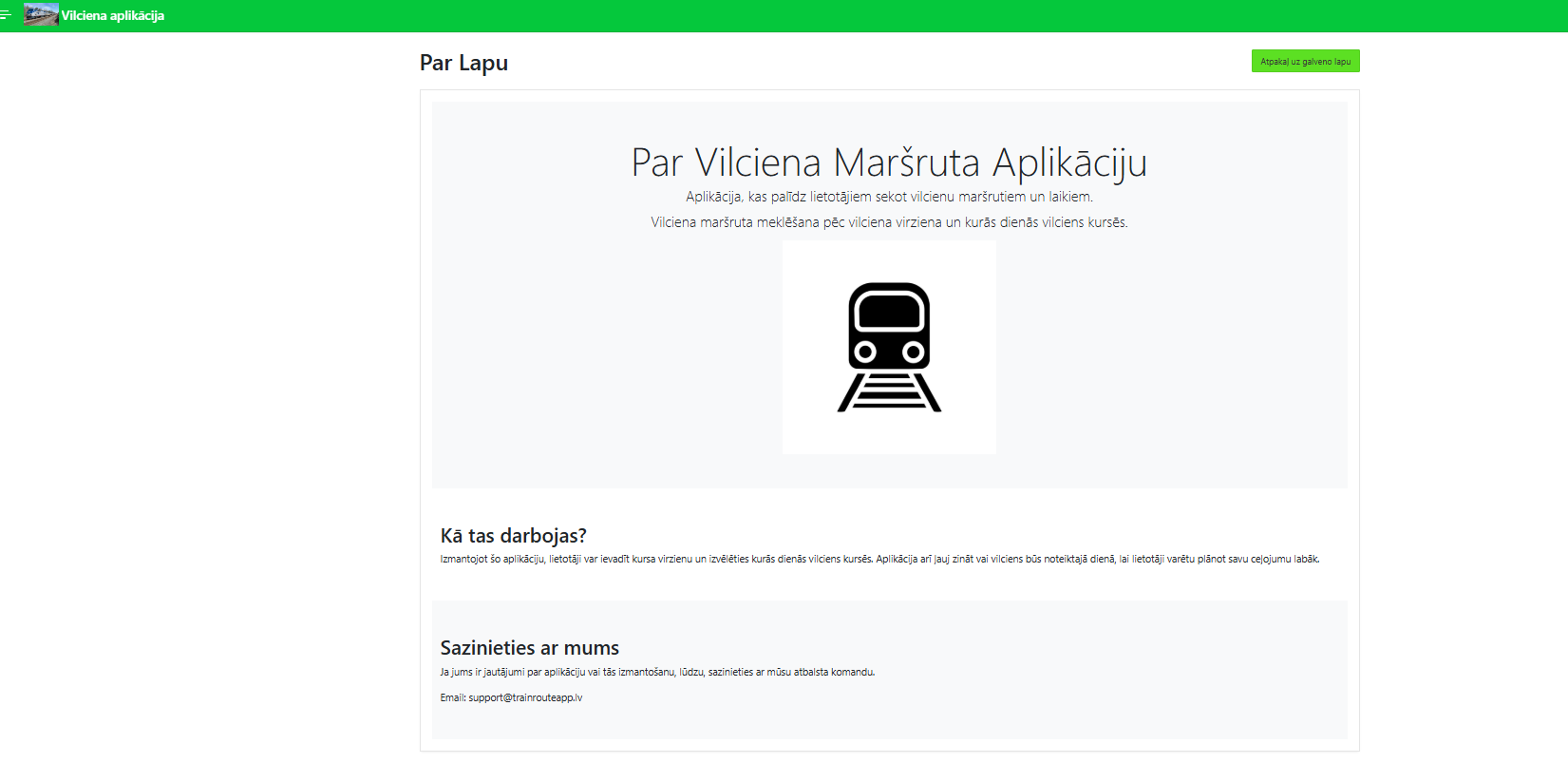
**Attēls 3.1:** Mājaslapas pieslēgšanās



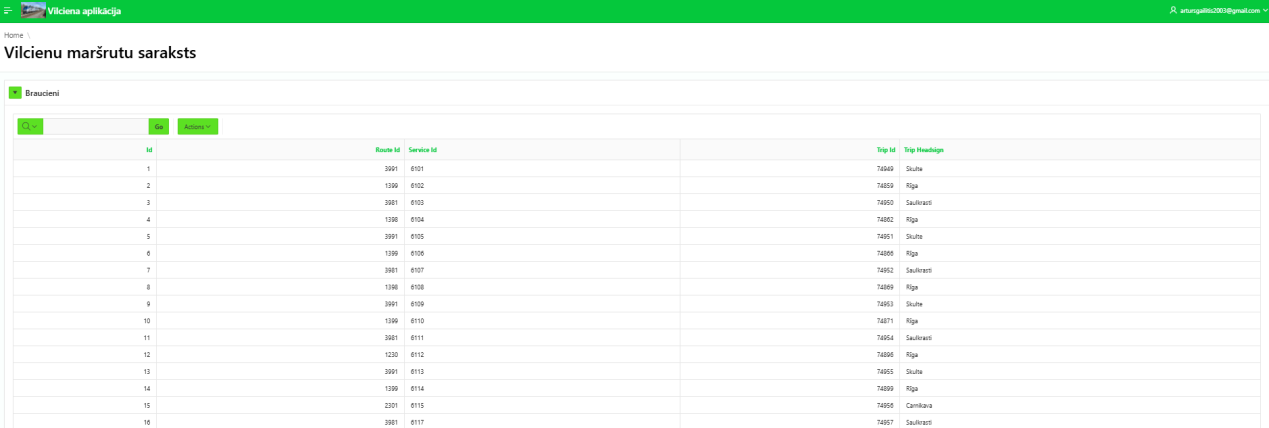
**Attēls 3.2:** Mājaslapas galvenā lapa.



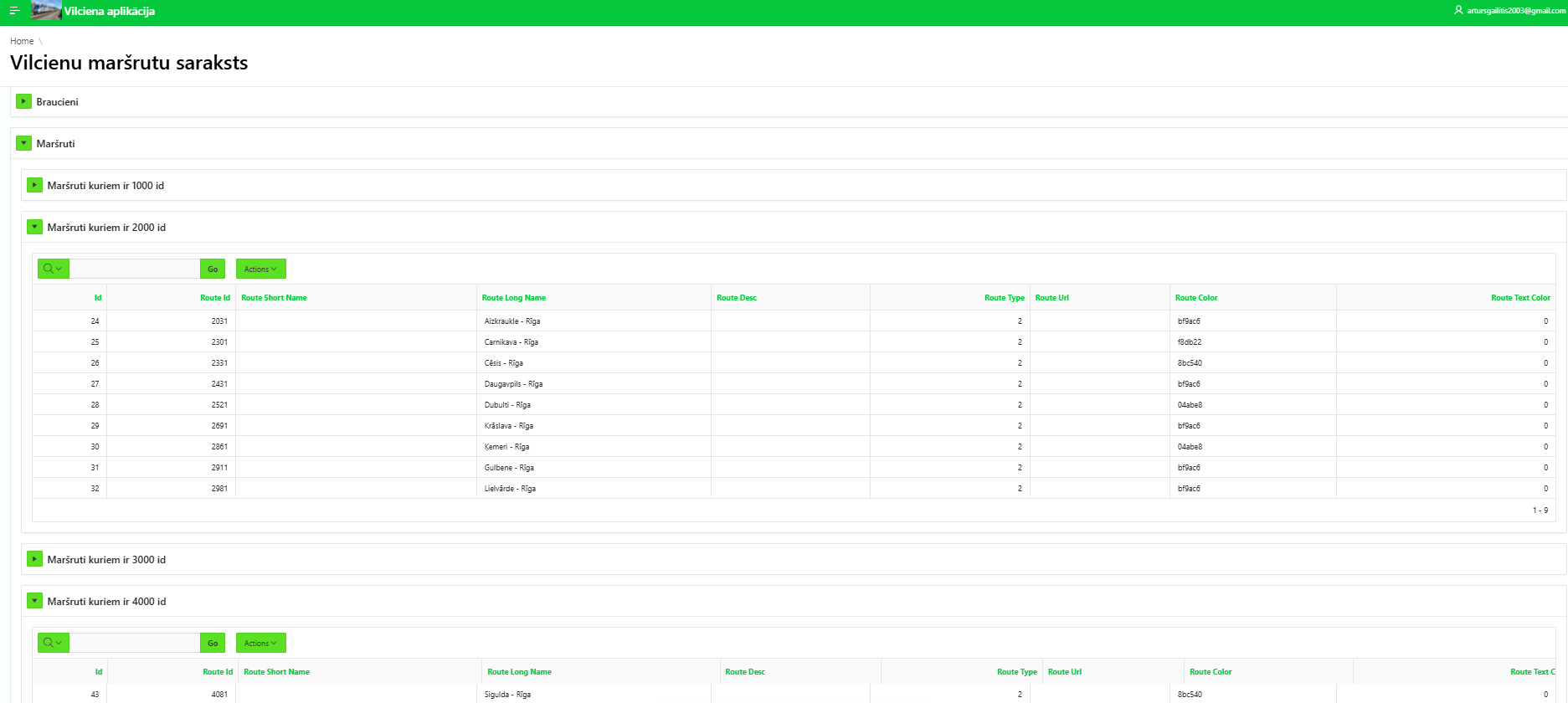
**Attēls 3.3:** Informācija par lietotni,



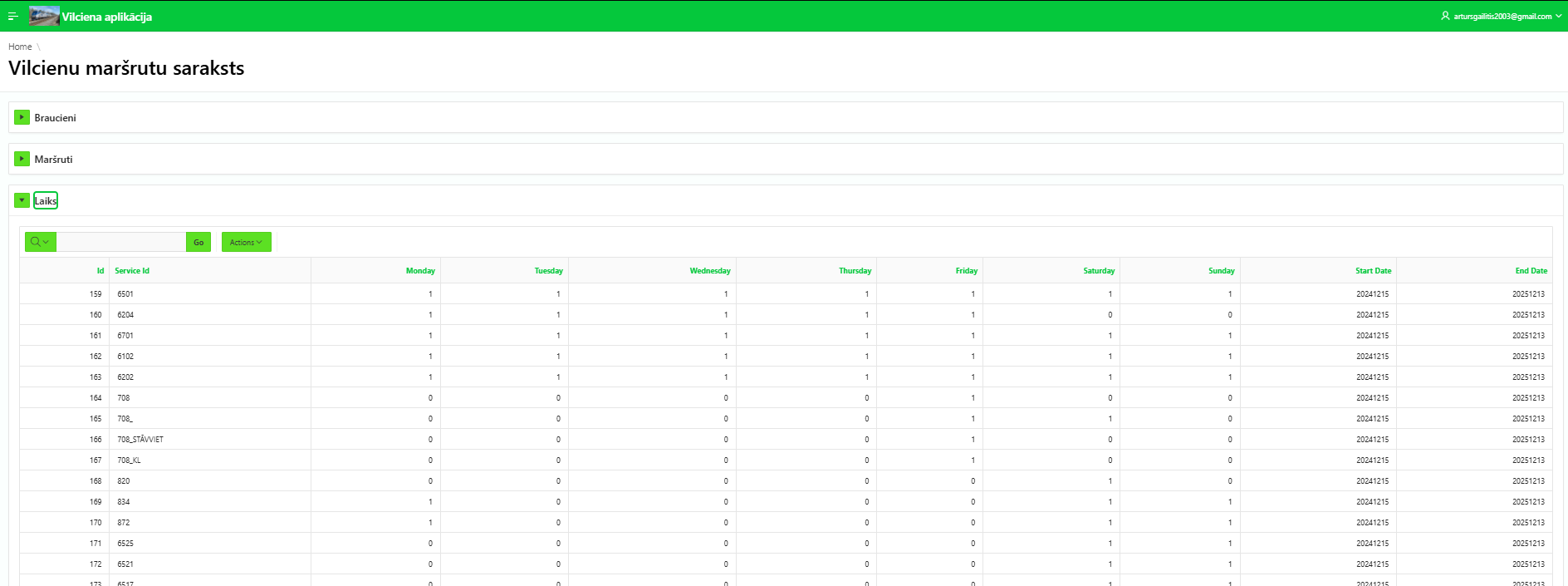
**Attēls 3.4:** Lietotāji var apskatīties ievadītos braucienus.



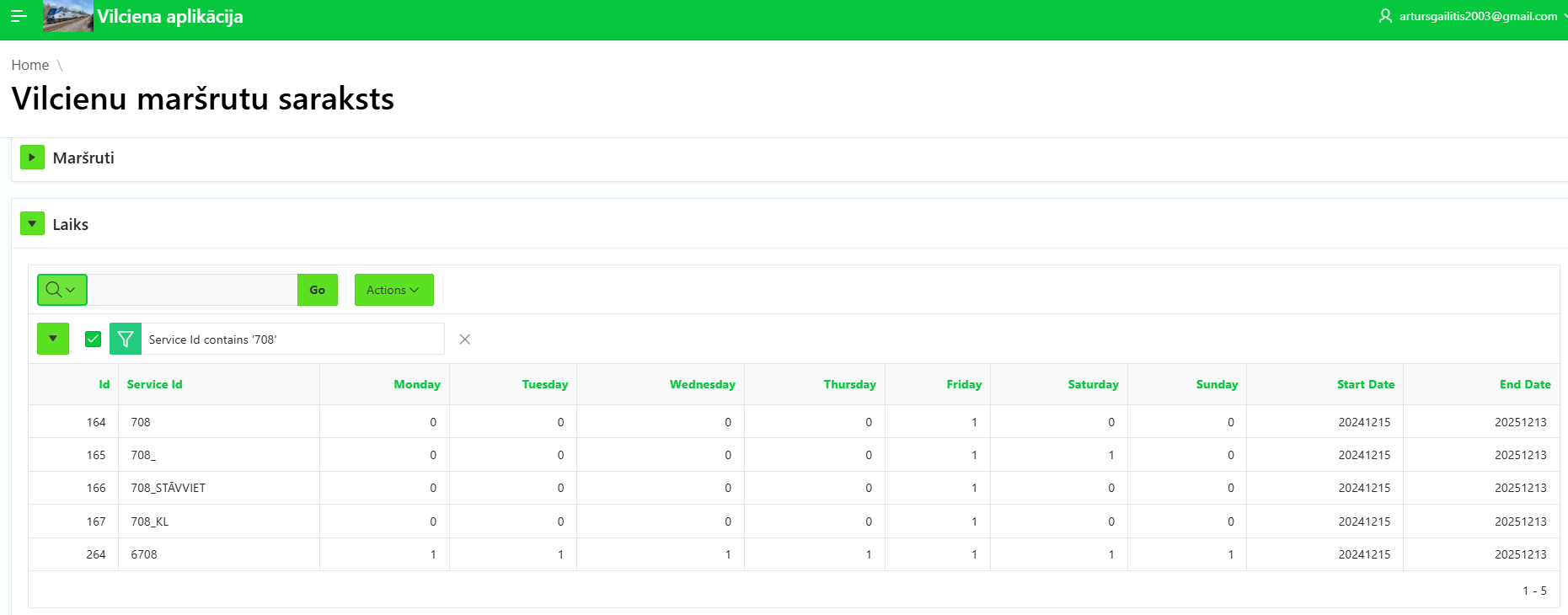
**Attēls 3.5:** Lietotāji var apskatīties maršrutus balstoties uz to id.



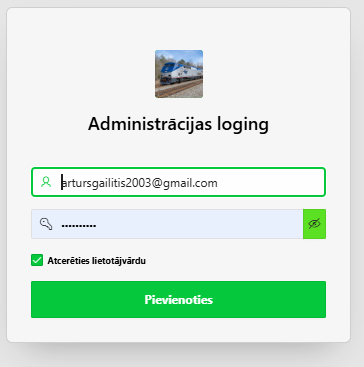
**Attēls 3.6:** Lietotāji var apskatīties kurās dienās vilciena kustība būs.



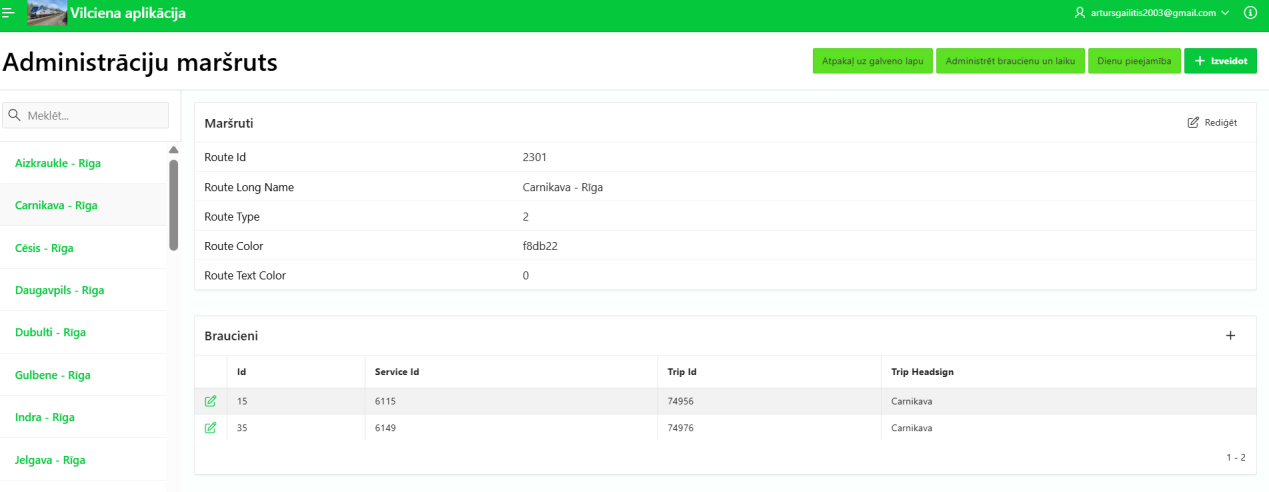
**Attēls 3.7:** Lietotājs var filtrēt datus balstoties uz vajadzību.



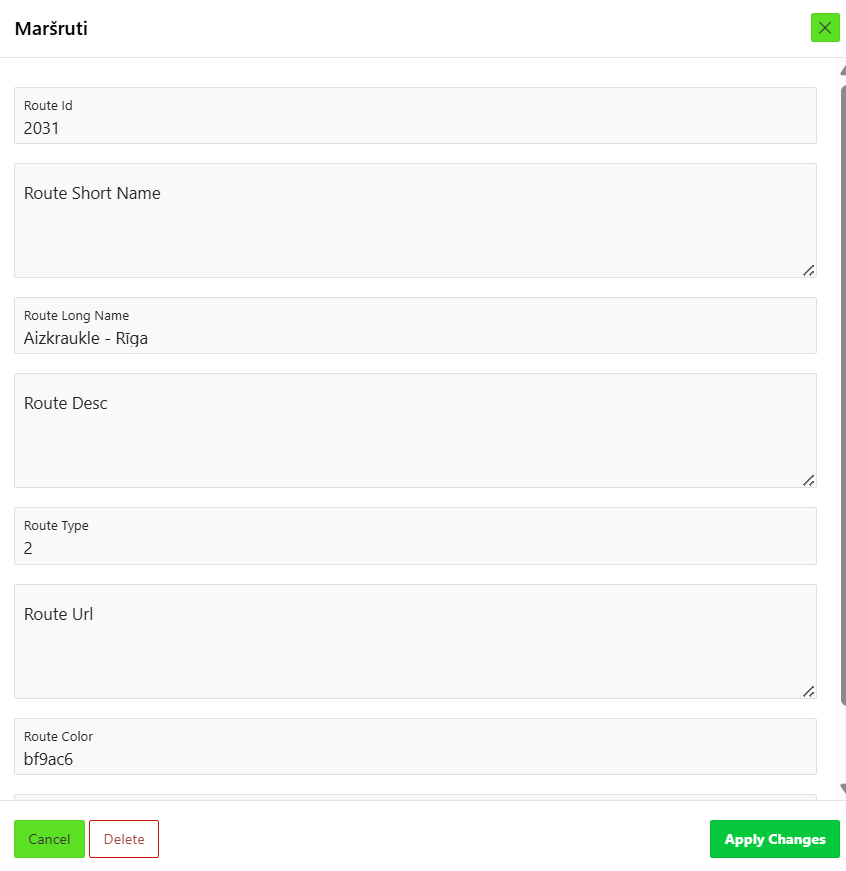
**Attēls 3.8:** Administrēšanas daļas pieslēgšanās.



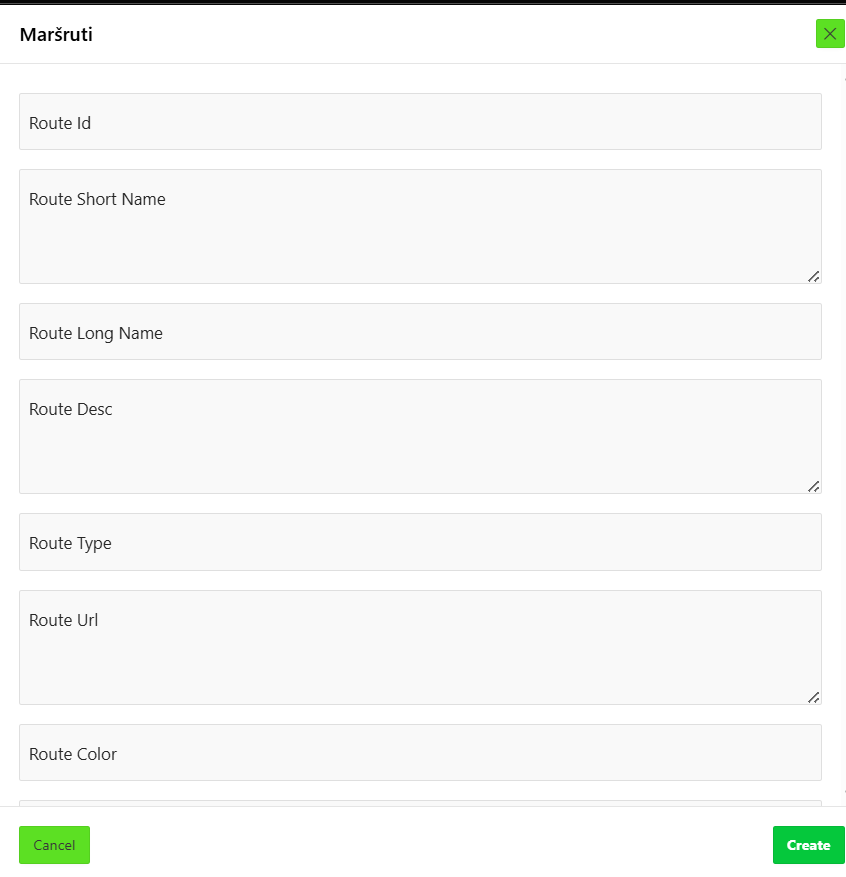
**Attēls 3.9:** Vieta lietotnē kurā var rediģēt vai pievienot jaunu maršrutu, braucieni.



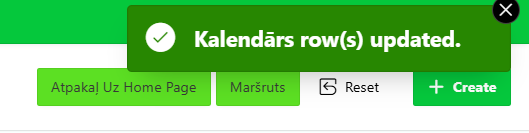
**Attēls 3.10:** Rediģēt esošo ierakstu.



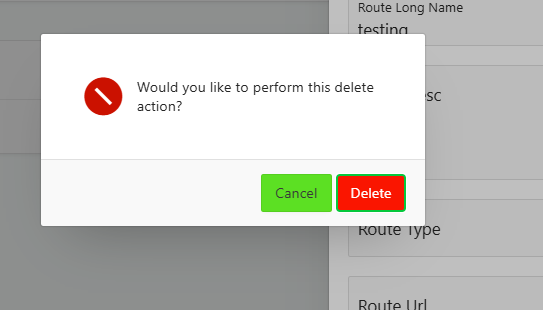
**Attēls 3.11:** Pievieno jaunu ierakstu.



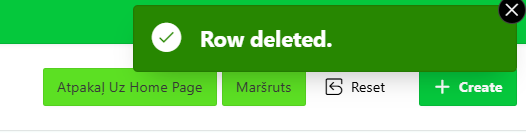
**Attēls 3.12:** Ieraksts ir pievienots tabulā.



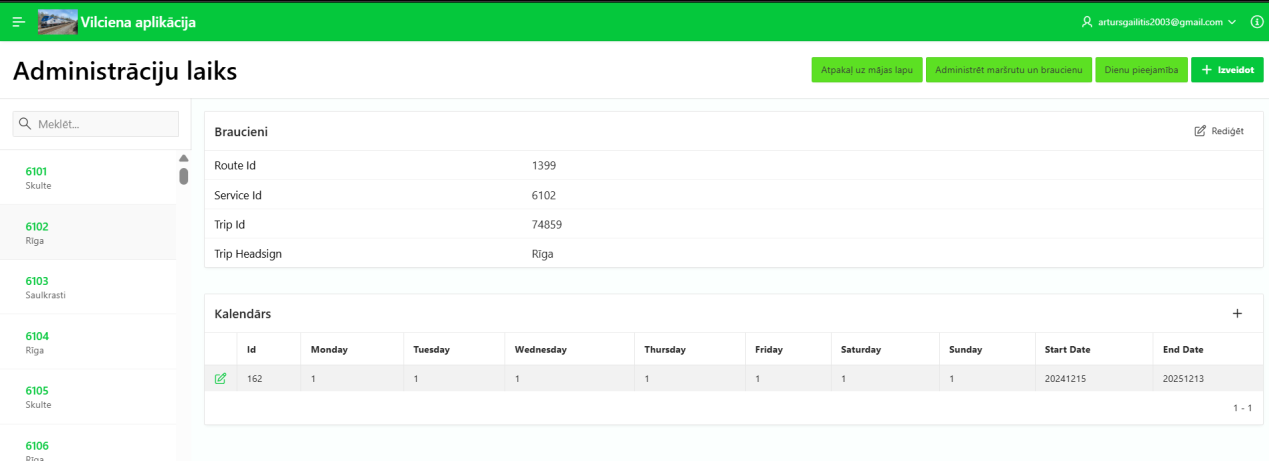
**Attēls 3.13:** Papildus jautā vai tiešām izdzēst ierakstu



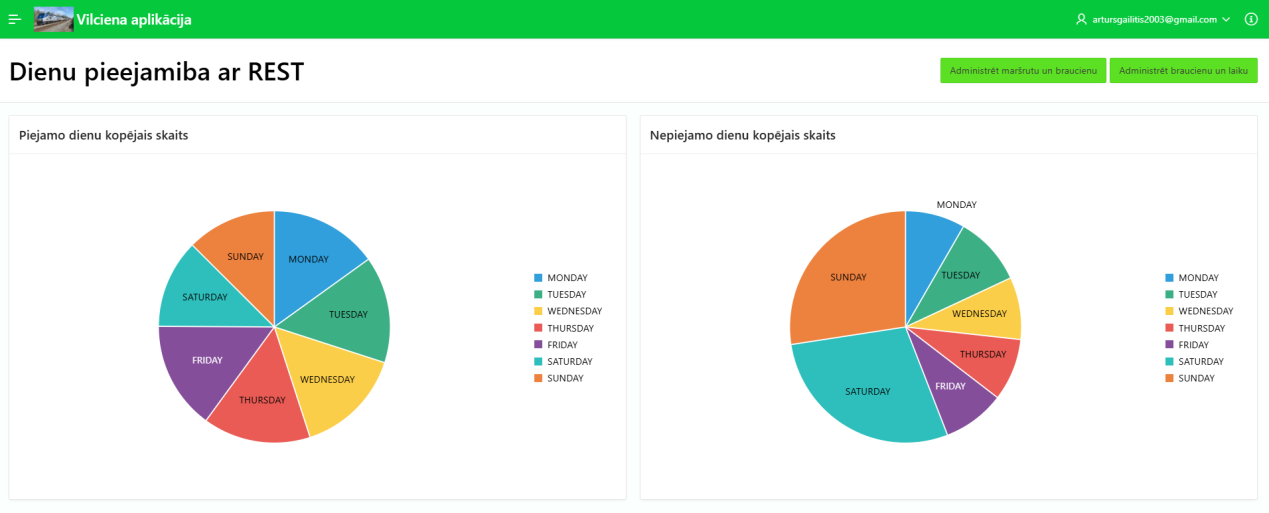
**Attēls 3.14:** Tabulas ierakstu izdzēš



**Attēls 3.15:** Vieta lietotnē kurā var rediģēt vai pievienot jaunu kalendāru, braucienus.



**Attēls 3.16:** Dienu pieejamība ar REST datu avotu



# SECINĀJUMI

Galvenais mērķis, kāpēc šis darbs tika veikts, ir sasniegts, jo pielietojot indeksus, sekvences un procedūru, tika izveidots vienkārša lietotne par vilciena kustību. Šajā lietotnē var filtrēt esošos datus, gan arī, priekš lietotnes administratoriem, ievietot jaunus datus vai rediģēt esošos.

Taču viens no uzlabojumiem varētu būt pievienot pogu administrēšanas sadaļā. Lai nospiežot pogu izpildās no jauna procedūra “Sadalit\_Marsrutus”. Kas atjaunina lietotājiem maršrutu tabulas pēc to id. Savādāk administratoriem ir jāiet uz SQL commands, lai to atjaunotu.

Izmantojot Oracle Apex var secināt, ka, lai izveidotu diez ko modernu aplikāciju ir jābūt zināšanas, kurā vietā var izveidot elementiem funkcijas. Tas nozīmē topošajam vai esošajam Oracle Apex lietotājam, ir jāzina, kura sadaļa elementā, viņam ir nepieciešama. Tas padara lietotnes taisīšanu sarežģītāku, priekš topošajiem izstrādātajiem. Taču ja zina ko darīt, tad Oracle vidē var taisīt labas lietotnes.

Nākamais secinājums ir tas, ka lai apskatītu tabulu indeksus “Constraint” sadaļās, ir jāgaida kāds laiks. Ja grib skatīties tikai vienu tabulu, tad tas ir paciešami. Bet ja grib vairākus, tad tas gaidīšanas laiks traucē darboties ar datubāze.

# SAĪSINĀJUMI UN ATSLĒGVĀRDI

HTML – (HyperText Markup Language) ir iezīmēšanas valoda, ko izmanto tīmekļa lapu veidošanai un strukturēšanai.

MySQL - populāra atvērtā pirmkoda relāciju datubāžu pārvaldības sistēma, ko izmanto datu glabāšanai, pārvaldībai un manipulācijām.

PostgreSQL - relāciju datubāžu pārvaldības sistēma, kura ir piemērota sarežģītiem datu modeļiem un datu analīzēm.

CSV - (Comma-Separated Values) ir faila formāts, kur datus tabulās, viens ar otru atdala ar komatu, vai citu zīmi.

ZIP - ir formāts, kuru izmanto failu saspiešanai, lai tas samazinātu kopējo izmēru.

# LITERATŪRAS SARAKSTS

* Amazon. (bez datuma). *What is SQL (Structured Query Language)?* Ielādēts 2025. gada 8. Maijs no amazon.com: https://aws.amazon.com/what-is/sql/
* GeeksforGeeks. (2025. gada 8. Aprīlis). *Introduction of ER Model*. Ielādēts 2025. gada 8. Maijs no geeksforgeeks.org: https://www.geeksforgeeks.org/introduction-of-er-model/
* General Transit Feed Specification. (bez datuma). *GTFS: Making Public Transit Data Universally Accessible*. Ielādēts 2025. gada 8. Maijs no gtfs.org: https://gtfs.org/getting-started/what-is-GTFS/
* Latvijas Atvērto datu portāls. (2021. gada 11. Janvāris). *Vilcienu kustības saraksts iekšzemes dzelzceļa maršrutos (GTFS formātā)*. Ielādēts 2025. gada 21. Februāris no data.gov.lv: https://data.gov.lv/dati/lv/dataset/iekszemes-dzelzcela-vilcienu-kustibas-saraksts-gtfs-formata
* Syntax. (2021. gada 27. Jūlijs). *5 Reasons to Choose Oracle Application Express (APEX)*. Ielādēts 2025. gada 8. Maijs no syntax.com: https://www.syntax.com/5-reasons-to-choose-oracle-application-express-apex/

# PIELIKUMS

## IZVEIDOŠANAS SKRIPTS

(bez datu ievietošanas, jo tas aizņemtu pārāk daudz vietas šajā dokomentācijā)

CREATE TABLE BRAUCIENI (

ID NUMBER GENERATED BY DEFAULT AS IDENTITY,

ROUTE\_ID NUMBER,

SERVICE\_ID VARCHAR2(50 BYTE),

TRIP\_ID NUMBER,

TRIP\_HEADSIGN VARCHAR2(50 BYTE),

CONSTRAINT ServiceID\_Con UNIQUE (SERVICE\_ID)

);

CREATE TABLE KALENDĀRS (

ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

SERVICE\_ID VARCHAR2(50 BYTE),

MONDAY NUMBER,

TUESDAY NUMBER,

WEDNESDAY NUMBER,

THURSDAY NUMBER,

FRIDAY NUMBER,

SATURDAY NUMBER,

SUNDAY NUMBER,

START\_DATE NUMBER,

END\_DATE NUMBER,

CONSTRAINT SERVICEID\_F\_CON FOREIGN KEY (SERVICE\_ID) REFERENCES BRAUCIENI (SERVICE\_ID)

);

CREATE TABLE MARŠRUTI (

ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

ROUTE\_ID NUMBER,

ROUTE\_SHORT\_NAME VARCHAR2(4000 BYTE),

ROUTE\_LONG\_NAME VARCHAR2(100 BYTE),

ROUTE\_DESC VARCHAR2(4000 BYTE),

ROUTE\_TYPE NUMBER,

ROUTE\_URL VARCHAR2(4000 BYTE),

ROUTE\_COLOR VARCHAR2(50 BYTE),

ROUTE\_TEXT\_COLOR NUMBER,

CONSTRAINT RoutesID\_Con UNIQUE (ROUTE\_ID)

);

ALTER TABLE BRAUCIENI

ADD CONSTRAINT ROUTEID\_F\_CON

FOREIGN KEY (ROUTE\_ID) REFERENCES MARŠRUTI (ROUTE\_ID);

CREATE SEQUENCE "Braucieni\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 75663 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL ;

CREATE SEQUENCE "Maršruti\_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 10 START WITH 1800 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL ;

create or replace trigger "Braucieni\_TRG"

before insert on BRAUCIENI

for each row

begin

if :NEW."TRIP\_ID" is null then

select "Braucieni\_SEQ".nextval into :NEW."TRIP\_ID" from sys.dual;

end if;

end;

/

create or replace trigger "Maršruti\_TRG"

before insert on MARŠRUTI

for each row

begin

if :NEW."ROUTE\_ID" is null then

select "Maršruti\_SEQ".nextval into :NEW."ROUTE\_ID" from sys.dual;

end if;

end;

/

create or replace PROCEDURE Sadalit\_Marsrutus IS

i NUMBER := 1000;

step NUMBER := 1000;

max\_route NUMBER;

sql\_text VARCHAR2(4000);

drop\_sql VARCHAR2(4000);

BEGIN

SELECT MAX(ROUTE\_ID) INTO max\_route FROM MARŠRUTI;

WHILE i <= max\_route LOOP

drop\_sql := 'BEGIN ' ||

' EXECUTE IMMEDIATE ''DROP TABLE "MARŠRUTI\_AR\_' || i || '\_ID"''; ' ||

'EXCEPTION ' ||

' WHEN OTHERS THEN ' ||

' IF SQLCODE != -942 THEN RAISE; END IF; ' ||

'END;';

EXECUTE IMMEDIATE drop\_sql;

sql\_text := 'CREATE TABLE "MARŠRUTI\_AR\_' || i || '\_ID" AS ' ||

'SELECT \* FROM MARŠRUTI WHERE ROUTE\_ID BETWEEN ' || i || ' AND ' || (i + step - 1);

EXECUTE IMMEDIATE sql\_text;

i := i + step;

END LOOP;

END Sadalit\_Marsrutus;

/

BEGIN

Sadalit\_Marsrutus;

END;

/

create or replace PROCEDURE dienu\_skaitu\_saskaitisana AS

TYPE dienas\_tips IS TABLE OF VARCHAR2(20);

dienas dienas\_tips := dienas\_tips('MONDAY', 'TUESDAY', 'WEDNESDAY', 'THURSDAY', 'FRIDAY', 'SATURDAY', 'SUNDAY');

sql\_text VARCHAR2(4000);

drop\_sql VARCHAR2(4000);

BEGIN

drop\_sql := 'BEGIN ' ||

' EXECUTE IMMEDIATE ''DROP TABLE DIENU\_SKATS''; ' ||

'EXCEPTION ' ||

' WHEN OTHERS THEN ' ||

' IF SQLCODE != -942 THEN RAISE; END IF; ' ||

'END;';

EXECUTE IMMEDIATE drop\_sql;

sql\_text := 'CREATE TABLE DIENU\_SKATS (

ID NUMBER GENERATED ALWAYS AS IDENTITY,

DIENA VARCHAR2(20),

PIEEJAMI NUMBER,

NEPIEEJAMI NUMBER

)';

EXECUTE IMMEDIATE sql\_text;

FOR i IN 1..dienas.COUNT LOOP

sql\_text := 'INSERT INTO DIENU\_SKATS (DIENA, PIEEJAMI, NEPIEEJAMI) ' ||

'SELECT ''' || dienas(i) || ''', ' ||

'SUM(CASE WHEN ' || dienas(i) || ' = 1 THEN 1 ELSE 0 END), ' ||

'SUM(CASE WHEN ' || dienas(i) || ' = 0 THEN 1 ELSE 0 END) ' ||

'FROM KALENDĀRS';

EXECUTE IMMEDIATE sql\_text;

END LOOP;

END dienu\_skaitu\_saskaitisana;

/

BEGIN

dienu\_skaitu\_saskaitisana;

END;

/

## IZDZĒŠANAS SKRIPTS

BEGIN

EXECUTE IMMEDIATE 'DROP PROCEDURE dienu\_skaitu\_saskaitisana';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP PROCEDURE Sadalit\_Marsrutus';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TRIGGER "Maršruti\_TRG"';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TRIGGER "Braucieni\_TRG"';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP SEQUENCE "Maršruti\_SEQ"';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP SEQUENCE "Braucieni\_SEQ"';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE DIENU\_SKATS';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

DECLARE

i NUMBER := 1000;

max\_route NUMBER;

BEGIN

SELECT NVL(MAX(ROUTE\_ID), 0) INTO max\_route FROM MARŠRUTI;

WHILE i <= max\_route LOOP

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE "MARŠRUTI\_AR\_' || i || '\_ID"';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

i := i + 1000;

END LOOP;

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE KALENDĀRS';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE BRAUCIENI';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE MARŠRUTI';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/