

RĪGAS TEHNISKĀ UNIVERSITĀTE

Datorzinātnes un informācijas tehnoloģijas fakultāte

5.praktiskais darbs

mācību priekšmetā

“Datu bāzes vadības sistēmas”

**Servera programmēšana**

Izstrādāja: Jolanta Bērziņa

Stud.apl.nr 181RDB178

Pārbaudīja: lektors A.Auziņš

2019./2020. m.g.

**Saturs**

[1. Ievads 2](#_Toc42437010)

[1.1. Darba mērķis 2](#_Toc42437011)

[1.2. Darba uzdevumi 2](#_Toc42437012)

[2. Datu bāzes tabulas izveide 3](#_Toc42437013)

[3. Datubāzes trigeris 7](#_Toc42437014)

[Secinājumi 9](#_Toc42437015)

# Ievads

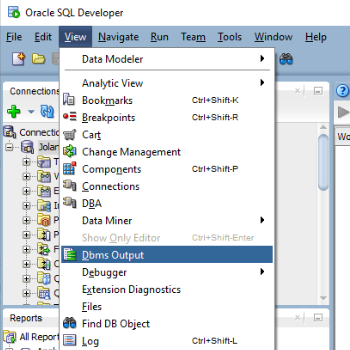
## Darba mērķis

Piektā prakstiskā darba studiju programmā “Datu bāzes vadības sistēmas” galvenais mērķis ir iepazīt servera programmēšanas ideju, izveidot nelielu PL/SQL programmu un trigeri.

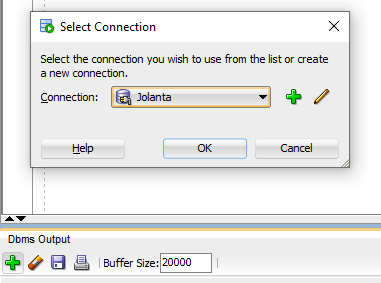
## Darba uzdevumi

1. Izveidot datu bāzes tabulu;
   1. Ievadīt datus;
   2. Uzrakstīt nelielu PL/SQL programmu, kurā izmantojot kursoru, datu bāzes tabulas dati tiek pārrakstīti kādā PL/SQL tabulā un tiek izvadīti uz ekrāna;
2. Izveidot datu bāzes trigeri un pārbaudīt tā darbību;
   1. Vizualizēt pārbaudes rezultātus, lai redzams, ka trigeris nostrādā un izpilda norādītās funkcijas;
3. Izdarīt secinājumu par apskatīto tehnoloģiju izmantošanas lietderību.

# Datu bāzes tabulas izveide

Pirms darba ar PL/SQL uzsākšanu ir nepieciešams DBMS output sakts (*View*). Strādājot ar SQLDeveloper, tas ir izdarāms izvēlnē *View* un iekš tā izvēlas DBMS output (skat. 2.1.att).

2.1.attēls. DBMS output atvēršana

DBMS Output View ir jāsavieno ar datubāzi, izmantojot to pašu savienojumu, kas tiek pielietots, lai SQLDeveloper savienotu ar Oracle datubāzi.

2.2.attēls. DBMS Output savienošana ar datubāzi

Pēc savienojuma izveides var uzsākt darbu un izveidot tabulu. Tabulas izveidei tika izmantots sekojošā komanda:

BEGIN

EXECUTE IMMEDIATE 'CREATE TABLE SPELETAJS\_PL (

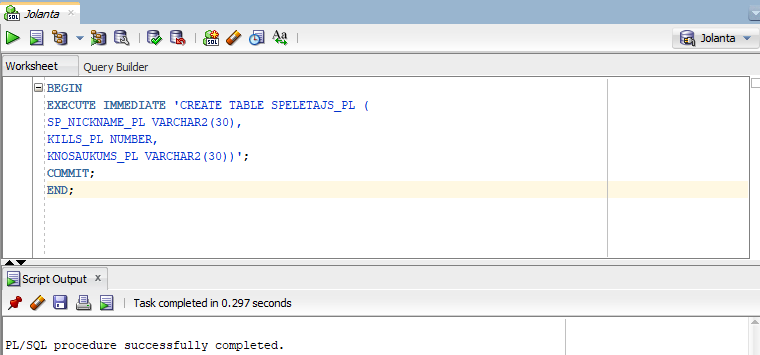
SP\_NICKNAME\_PL VARCHAR2(30),

KILLS\_PL NUMBER,

KNOSAUKUMS\_PL VARCHAR2(30))';

COMMIT;

END;

2.3.attēls. Veiksmīga tabulas SPELETAJS\_PL izveide

Ar augstākredzamās komandas palīdzību tiek izveidota tabula SPELETAJS\_PL, kas sevī ietver 3 kolonas – SP\_NICKNAME\_PL, KILS\_PL, KNOSAUKUMS\_PL.

Pēc tabulas izveides var uzsākt tās aizpildi ar ierasktiem. Šīs darbības veikšanai tiek izmantots kursors, kas iegūs nepieciešamos datus no tabulas SPELETAJS.

SET SERVEROUTPUT ON

DECLARE

SP\_NICKNAME\_PL VARCHAR2(30);

KILLS\_PL NUMBER;

KNOSAUKUMS\_PL VARCHAR (30);

CURSOR CURS1 IS SELECT SPELETAJS.SP\_NICKNAME, SPELETAJS.KILLS, SPELETAJS.K\_NOSAUKUMS

FROM SPELETAJS;

BEGIN

OPEN CURS1;

LOOP

FETCH CURS1 INTO SP\_NICKNAME\_PL, KILLS\_PL, KNOSAUKUMS\_PL;

INSERT INTO SPELETAJS\_PL

VALUES (SP\_NICKNAME\_PL, KILLS\_PL, KNOSAUKUMS\_PL);

DBMS\_OUTPUT.PUT\_LINE (SP\_NICKNAME\_PL || ' '||KILLS\_PL ||' '|| KNOSAUKUMS\_PL);

EXIT WHEN SP\_NICKNAME\_PL= 'NothingButAverage';

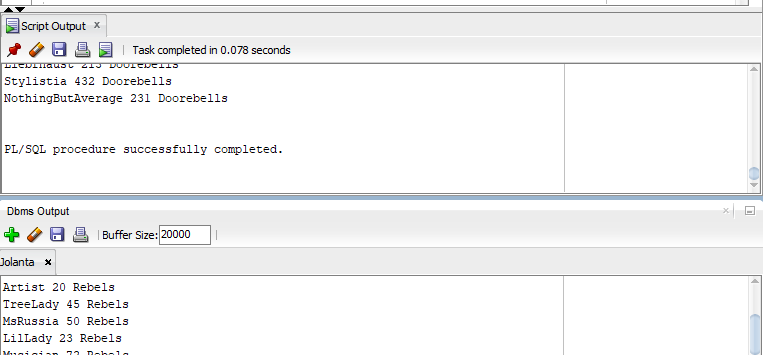
END LOOP;

CLOSE CURS1;

END;

/

Lai gan šis vaicājums nav deklarēts kā funkcija, tad to atkārtoti izmantot pagaidām nevar. Neraugoties uz to šis vaicājums tāpāt strādā.

2.4.attēls. Skripta rezultāts ar ierkastu izveidi

Vaicājums ne tikai aizpilda tabulu, bet arī izvada rezultātu teksta veidā *DBMS Output* logā. Skripta kodā tiek izmantots cikla, lai ar kursoru varētu iziet caur tabulu. Cikls tiek pārtraukts, kad tiek sasniegts pēdējāis spēlētāja ieraksts. Vaicājums strādās pareizi, līdz netiek izmainīta pēdējā ieraksta *SP\_NICKNAME* vērtība.

Lai varētu strādāt ar pašu PL/SQL tabulu, nepieciešams izveidot tipu – kas ir attiecīgā tabula. Šinī gadījumā tips tiek izveidots piešķirot iepriekš izveidotās tabulas datu tipus. Tas nozīmē, ka no struktūras ziņas tabulas ir identiskas.

Skripta kods pieminētajai darbībai:

SET SERVEROUTPUT ON

DECLARE

TYPE SQLPLTABLE IS TABLE OF SPELETAJS\_PL%ROWTYPE INDEX BY BINARY\_INTEGER;

SQLTABLE1 SQLPLTABLE;

SP\_NICKNAME\_PLS VARCHAR2(30);

KILLS\_PLS NUMBER;

KNOSAUKUMS\_PLS VARCHAR2(30);

I NUMBER (30):=1;

CURSOR CURS1 IS SELECT SPELETAJS.SP\_NICKNAME, SPELETAJS.KILLS, SPELETAJS.K\_NOSAUKUMS

FROM SPELETAJS;

BEGIN

OPEN CURS1;

LOOP

FETCH CURS1 INTO SP\_NICKNAME\_PLS, KILLS\_PLS, KNOSAUKUMS\_PLS;

INSERT INTO SPELETAJS\_PL

VALUES (SP\_NICKNAME\_PLS, KILLS\_PLS, KNOSAUKUMS\_PLS);

SQLTABLE1(I).SP\_NICKNAME\_PL:=SP\_NICKNAME\_PLS;

SQLTABLE1(I).KILLS\_PL:=KILLS\_PLS;

SQLTABLE1(I).KNOSAUKUMS\_PL:=KNOSAUKUMS\_PLS;

DBMS\_OUTPUT.PUT\_LINE (SQLTABLE1(I).SP\_NICKNAME\_PL|| ' : '|| SQLTABLE1(I).KILLS\_PL || ' : '|| SQLTABLE1(I).KNOSAUKUMS\_PL);

I:=I+1;

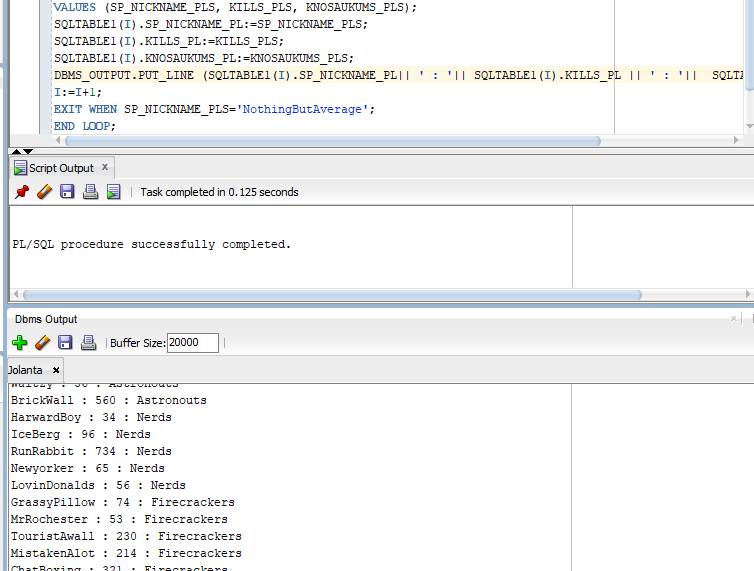
EXIT WHEN SP\_NICKNAME\_PLS='NothingButAverage';

END LOOP;

CLOSE CURS1;

END;

/

2.5.attēls. Skripta koda izvades rezultāti

Programma aizpilda tabulu ar kursoru, gluži tāpat kā iepriekšējā gadījumā, tikai šajā reizē *DBMS Output View* tiek izvadīts izmantojot izveidoto PLSQL tabulu.

# Datubāzes trigeris

Vieni no visbiežāk izmantotajiem datu bāzes trigera notikumiem ir komandu INSERT, UPDATE, DELETE izpilde kādai datubāzes tabulai. Ar datubāzes trigeru palīdzību tiek nodrošināts ka veicamās izmaiņas datubāzē tik tiešām tiek paveiktas.

Šajā gadījumā trigers tiek izveidots tabulai SPELETAJS, kad tiek editēta kāda no ieraskta vērtībām.

CREATE OR REPLACE TRIGGER EDITING\_WITHTRIGGER

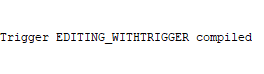
BEFORE UPDATE ON SPELETAJS

FOR EACH ROW

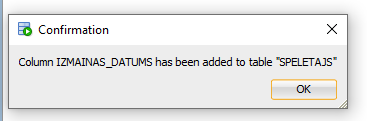
BEGIN

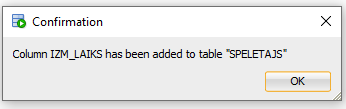
:NEW.IZMAINAS\_DATUMS:=SYSDATE;

:NEW.IZM\_LAIKS:=SYSTIMESTAMP;

END;

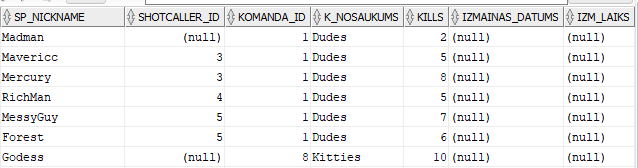
3.1.attēls. Veiksmīga trigger izveidošana

Lai varētu ar šādu trigeri strādāt, ir nepieciešams arī pievienot attiecīgās kolonas tabulai SPELKETAJS – kolonas IZMAINAS\_DATUMS un IZM\_LAIKS.

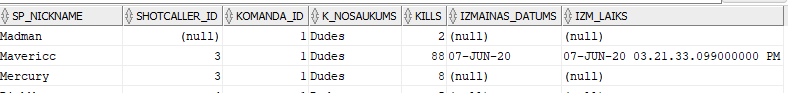
3.2.attēls.Kolonas IZMAINAS\_DATUMS veiksmīga pievienošana tabulai SPELETAJS

3.3.attēls. Kolonas IZM\_LAIKS veiksmīga pievienošana tabulai SPELETAJS

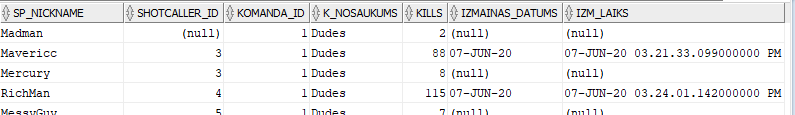
Pirms jebkādu pārmaiņu veikšanas vērtības kolonās IZMAINAS\_DATUMS un IZM\_LAIKS uzrādās (null), jeb nekādas izmaiņas nav veiktas (skat. 3.4.att).

3.4.attēls. Kolonu vērtības pirms izmaiņu veikšanas tabulā SPELETAJS

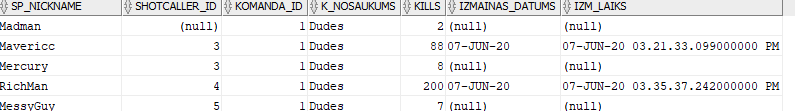
Nomainot speletajam *Mavericc*  KILLS vērtību no 5 uz 88, pēc tabulas atjaunošanas parādās datums, laiks kad pēdējās izmaiņas ir veiktas (skat.3.5. un 3.6.att.).



3.5.attēls. Izmaiņas rindas ierakstā pēc KILLS vērtību nomaiņas

3.6.attēls. Izmaiņas rindas ierakstā pēc KILLS vērtību nomaiņas

Pēc atkārtotas tā paša ieraksta izmaiņas kolonās IZMAINAS\_DATUMS un IZM\_LAIKS atkal nomainās (skat. 3.7.att.).

3.7.attēls. Izmaiņas datuma un laikā pēc atkārtota tā paša ieraksta UPDATE

# Secinājumi

Uzdevuma nosacījumos bija prasīts uzrakstīt nelielu Pl/SQL programmu ar kursoru un izveidot triggeri, notestēt to un vizuāli attainot tā darbības rezultātus. Nospraustie punkti tika veiksmīgi izpildīti. Pēc darba izpildes un mācību materiāla izpētes tika secināts, ka ar šādu rīku palīdzību ir iespējams izveidot sarežģītas konstrukcijas darbiem ar datiem, kas sniedz svas priekšrcības un lietderīgumu. Piemēram, ar līdzīga trigera realizāciju, var saglabāt ne tikai datus par laiku un datuu, ka dir veiktas izmaiņas, bet arī to kurš ir veicis izmaiņas datubāzē, attiecīgā tabulā vai kolonās. Šādā veidā var izsekot datu bāzes lietotāju, kas ir veicis izmaiņas un nepieciešamīabs gadījumā ir iespējams noskaidrot pamatojumu veiktajām pārmaiņām. Veiktās pārmaiņas var netikai saglabāt tajā pašā tabulā, kurā ir noptikušas pārmaioņas, bet pat saglabāt atsevišķā tabulā ar detalizētākiem ierakstiem. Tad ir piejama vesela editēšanas vēsture.

Darbā realizētais trigeris un PL/SQL programma ir ļoti vienkārši. Atskatoties uz iepriekšējiem veiktajiem darbiem saistībā ar datubāzēm, šis arī ir bijis diezgan īss. Bet paši pamati par datu bāzes serveri, PL/SQL funkcionalitāti ir izpētīti.