

RĪGAS TEHNISKĀ UNIVERSITĀTE

Datorzinātnes un informācijas tehnoloģijas fakultāte

5. praktiskais darbs

mācību priekšmetā

“Datu bāzu pamati un Relāciju datu bāzes (DB1)”

**Datu bāzes servera programmēšanas valodas (PL/SQL) izmantošana**

Izstrādāja: Viktors Stepičevs

II.kurss, 7.grupa, 191RDB100

Pārbaudīja: lektors A.Auziņš

**2020./21/māc.gads**

**Darba saturs**

[**Darba uzdevumi** 3](#_Toc73509541)

[**Anotācija** 4](#_Toc73509542)

[**Tabulu struktūru definēšana** 5](#_Toc73509543)

[**Kursora definēšana** 6](#_Toc73509544)

[**Trigeru definēšana** 8](#_Toc73509545)

[“**Insert” tipa trigeris** 8](#_Toc73509546)

[“**Update” tipa trigeris** 9](#_Toc73509547)

[**WITH priekšraksta ar procedūru un funkciju definēšana** 10](#_Toc73509548)

[**Funkcijas piemēra definēšana** 10](#_Toc73509549)

[**WITH priekšraksts ar vairākām funkcijām** 11](#_Toc73509550)

[**WITH priekšraksts ar procedūru un funkciju** 12](#_Toc73509551)

[**Dinamiskais SQL un PL/SQL** 14](#_Toc73509552)

[**DBMS SQL 1. piemērs** 14](#_Toc73509553)

[**DBMS SQL 2. piemērs** 15](#_Toc73509554)

[**Secinājumi** 16](#_Toc73509555)

[**Literatūra** 17](#_Toc73509556)

[**Pielikums** 18](#_Toc73509557)

[**Darba izmantotas datu bāzes SQL kods** 18](#_Toc73509558)

# **Darba uzdevumi**

1. Izveidot datu bāzes tabulu. Ievadīt datus (5-7 ieraksti).
   * Uzrakstīt PL/SQL programmu, kurā izmantojot kursoru, datu bāzes tabulas dati tiek pārrakstīti kādā PL/SQL tabulā un tiek vizualizēti.
2. Izveidot divus dažādus datu bāzes trigerus un pārbaudīt to darbību. (Ja nepieciešams, var izveidot vēl kādu tabulu trigera darbības parbaudei).
   * Vizualizēt pārbaudes rezultātus, lai ir redzams, ka trigeris nostrādā un izpilda norādītās funkcijas.
3. SQL WITH priekšrakstā ievietot **funkciju**un **procedūru**. Pārbaudīt to darbību un vizualizēt rezultātus.
4. Izveidot PL/SQL programmu, kas demonstrē dināmiskā SQL un dināmiskā PL/SQL izmantošanu.
5. Izdarīt secinājumus par apskatīto tehnoloģiju izmantošanas lietderību.

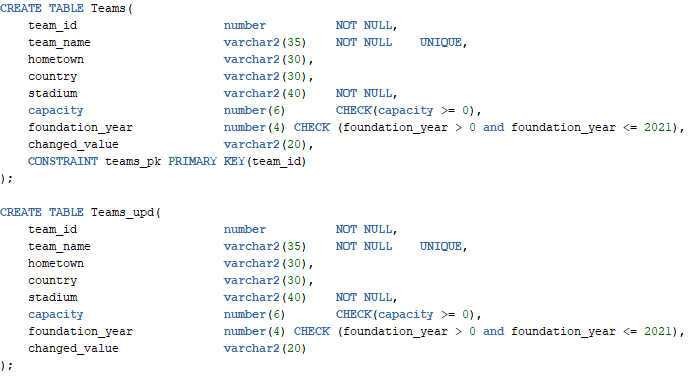
# **Anotācija**

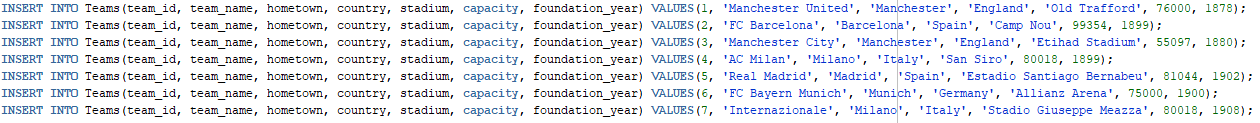
Balstoties uz darba uzdevumu nostādni, uzdevumu izpildēs laikā tika īstenotā aktīvas datubāzes sistēmas jēdzienu aplūkošana. Uzrakstot vairākās PL/SQL programmas un vizualizējot pārbaudes rezultātus ir iespējams saprast to darbības pamatprincipus un pārbaudīt izpildes korektumu. Darbs pilnīgi tika izstrādāts Oracle SQL Developer vidē.

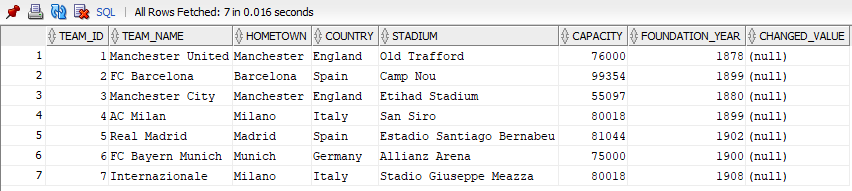
# **Tabulu struktūru definēšana**

Aplūkojot darbā definētos uzdevumus, pirmais ar ko ir jāsaka ir tabulu struktūru definēšana. Būtībā, ir teikts, ka darba ietvaros ir pietiekams ar vienu tabulu, bet dažādiem nolūkiem, kuri tiks apskatīti turpmāk es izveidoju arī otro tabulu, pirmās tabulas dublikātu. Darbā tika izmantota “Teams” tabula, kura varēja būt sastopama manos iepriekšējos darbos. Kopumā, runājot par izmaiņām atribūtu kopnē, tika noņemti visi FK, kuri bija saistīti ar citām tabulām, kuras nav izmantotas šī darba ietvaros. No jaunumiem tika ieviests atribūts “changed\_value” ar universālo datu tipu VARCHAR, lai varētu aizpildīt to kolonnu ne tikai ar cipariem, bet arī ar citiem simboliem. Šī kolonna tiks aizpildītā tad, kad tiks pielietoti datu bāzes trigeri datu atjaunošanai vai jauno iezīmju pievienošanai. Kā jau teicu agrāk, tabula “Teams\_upd” ir būtība tabulas “Teams” dublikāts, bet tā netiks sākotnēji aizpildīta ar ievades datiem.

Runājot par darba izmantotājiem ievades datiem, šī darba ietvaros samazināju iepriekšējo ierakstu skaitu līdz ieteiktajām darba uzdevumu nostādnē. Kopumā tika atstāti 7 ieraksti. Kā arī iepriekšējos darbos, darbā realizējamais kods varētu būt atrasts iesnieguma un atskaites pielikumos.

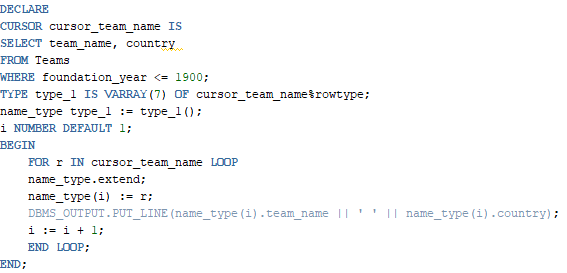




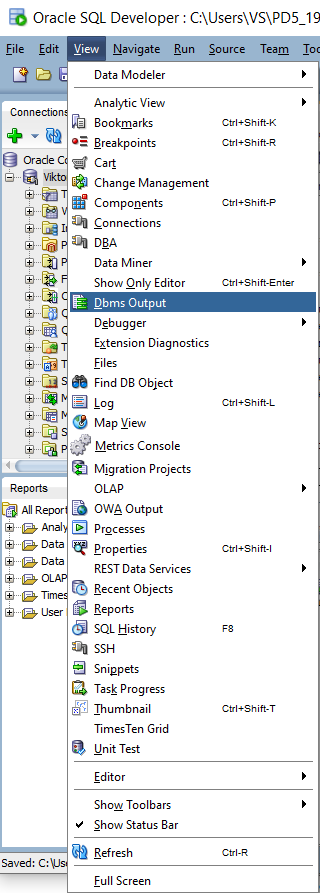


# **Kursora definēšana**

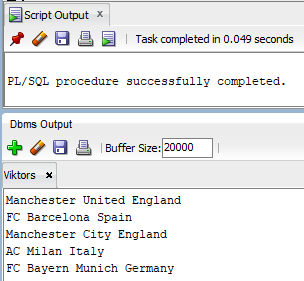
Ar šo kursora palīdzību iebūvētājā DBMS\_OUTPUT buferī tika nodotā informācija (komandas nosaukums un valsts) par komandām, kuras tika dibinātas 1900. gadā vai agrāk. Kā ir redzams, tika definēts kursora ar “for” ciklu “cursor\_team\_name”. Tā kā tabulā ir tikai 7 ieraksti, izvades masīva “VARRAY” maksimālas garuma vērtība ir līdzīga ar 7. Ar “%rowtype” atribūta palīdzību tika nolasīta katra tabulas rinda. Ar “DBMS\_OUTPUT” bufera paplašinājumu “PUT\_LINE” palīdzību izvades buferī tiek izprintētā katra noteiktā rinda. Katrā rindā izvadīju komandas nosaukumu un valsti.



Lai rezultātus būtu iespējams aplūkot, vispirms sadaļā “View” ir jāieslēdz “Dbms Output” buferi.



Palaižot kursoru, tika paziņots, ka procedūra tika veiksmīgi pabeigta un rezultāti tika izvadīti “Dbms Output” buferī.

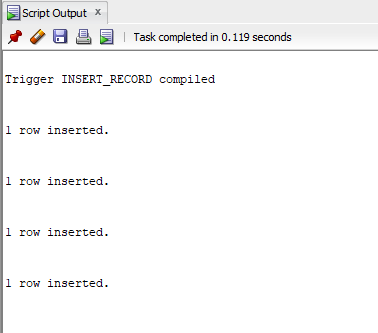


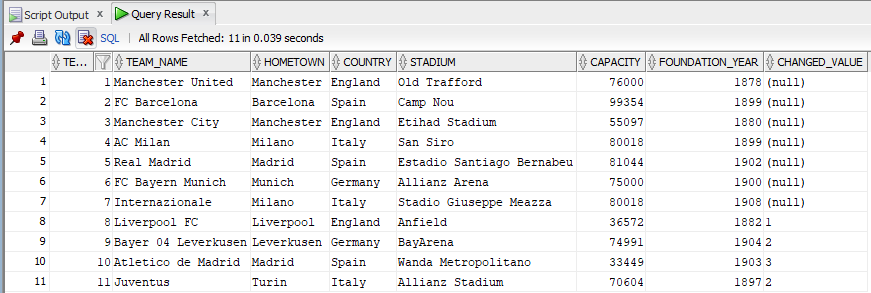
# **Trigeru definēšana**

## “**Insert” tipa trigeris**

Pēc šī trigera definēšanas, mums parādījas iespēja piešķirt jaunas stadiona kapacitātes (“capacity”) vērtības jaunajiem ierakstiem. “BEFORE INSERT” trigera definējumā nozīmē to, ka šīs trigeris tiks izpildīts pirms noteiktās operācijās, šajā gadījumā pirms ievietošanas. Gadījumā, jā jaunā ierakstā kolonnai “capacity” tiks piešķirta vērtības 1, tad šīm ierakstam kolonnai “capacity” tiks piešķirta gadījuma vērtība no 10000 līdz 50000. Pēc tādā paša principa piešķirot vērtību 2, gadījumā vērtības jau variēsies diapazonā no 50000 līdz 100000, bet jebkurā citā skaitļa (kurš ir lielāks par 0, jo tabulas definējuma tiek uzstādīta pārbaude) ievades gadījumā vienkārši tiek ģenerēts skaitlis no abiem iepriekšējiem diapazoniem. Arī, ir vērts pievienot, ka ar tām izmaiņām kolonnai “changed\_value” tiek pievienota atbilstoša iezīmē, lai varētu saprast pēc kura principa tiek izvēlētās gadījumā vērtības. Gadījumā ar “jebkuru” skaitli vienkārši tiek pievienota iezīme 3. Kā ir redzams, pēc trigera definēšanas tika ievietoti papildus ievades dati. Tā kā tabulas definējumā ari tiek uzstādīta pārbaude uz unikalitāti, pievienojot jaunas rindas ir jāpielieto jauno komandas nosaukumu. Pārbaudot rezultātus, kolonnā “changed\_value” ir redzams, redzams, kā tika noteikta jauna stadiona kapacitātes vērtība.

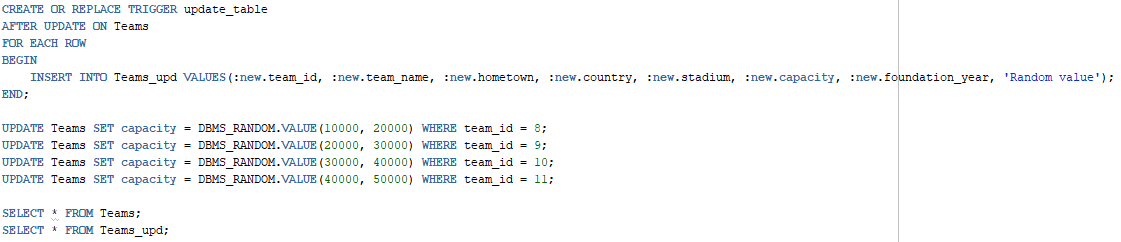


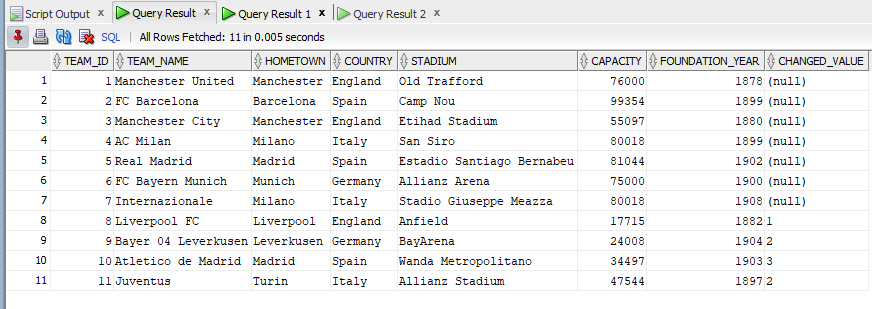


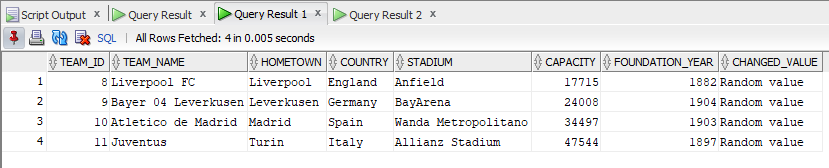


## “**Update” tipa trigeris**

Ar “update\_table” trigera palīdzību tiek atjaunotas stadiona kapacitātes vērtībās “jaunajām” iezīmēm. Šī informācija arī tiek ievietota dublikātu tabulā “Teams\_upd”



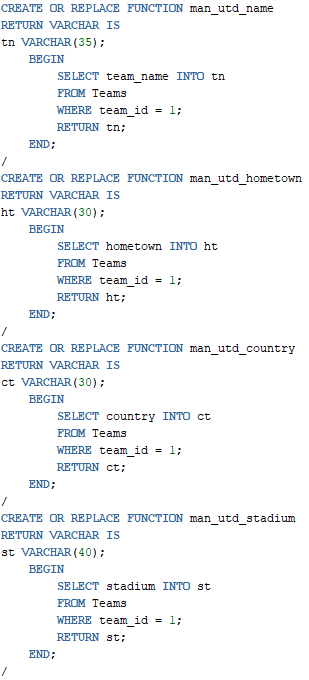


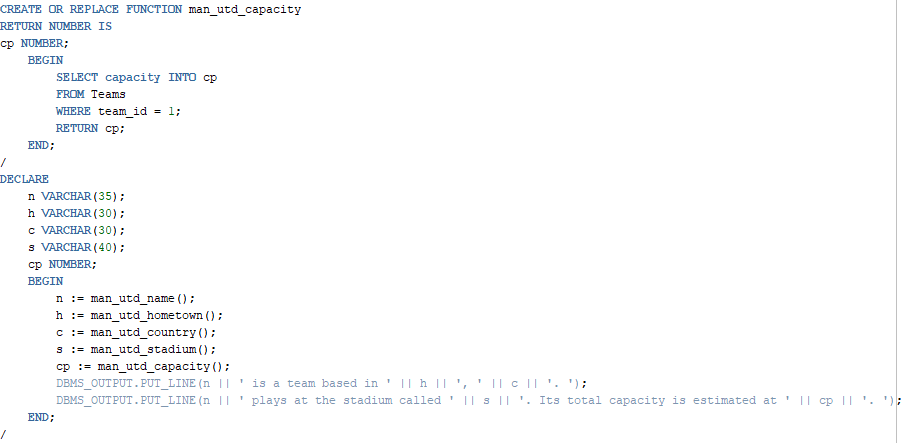


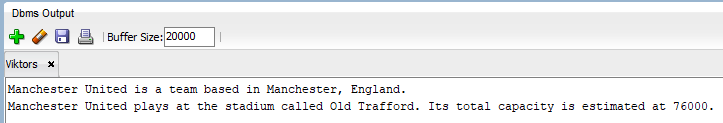
# **WITH priekšraksta ar procedūru un funkciju definēšana**

## **Funkcijas piemēra definēšana**

Šīs vaicājums nodrošināja informācijas izvadi par komandu ar “team\_id” = 1. Ar katru funkciju tika veikta piekļuve kādām no tabulas atribūtiem. Kā redzams, izvade tika nodrošināta ar “DBMS\_OUTPUT” buferi, noformējot to divu teikumu veidā.

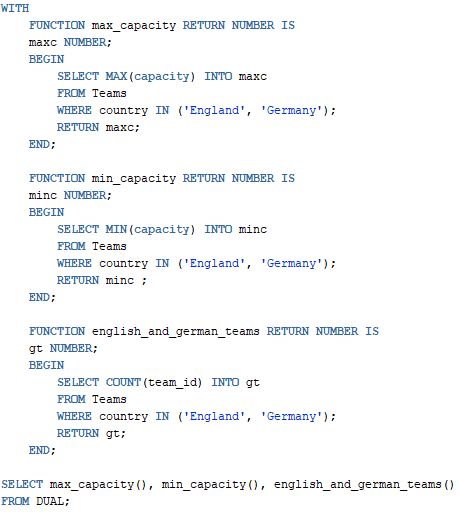


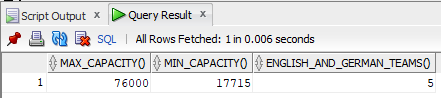




## **WITH priekšraksts ar vairākām funkcijām**

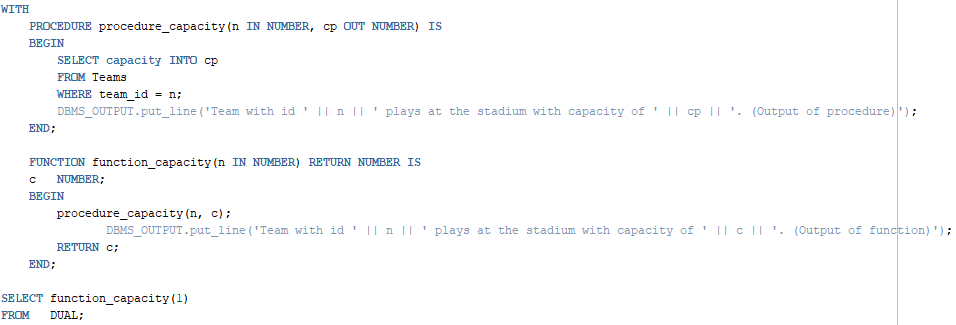
Šajā vaicājumā ietvaros tika pielietotās trīs funkcijas. Pirmā atgriež lielākās stadiona kapacitātes vērtību no angļu un vācu komandām, otrā tām pašām komandām atgriež vismazāko vērtību, bet trešā atgriež to komandu kopēja skaita aprēķinātu vērtību. Interesanti ir tas, ka rezultātā pie vismazākā stadiona tika izvadīta vērtība, kura tika ģenerēta trigeru pielietošanas laikā.

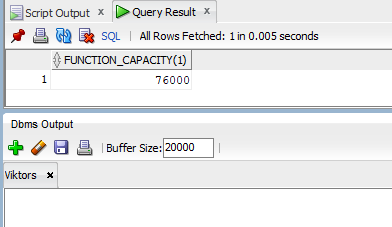




## **WITH priekšraksts ar procedūru un funkciju**

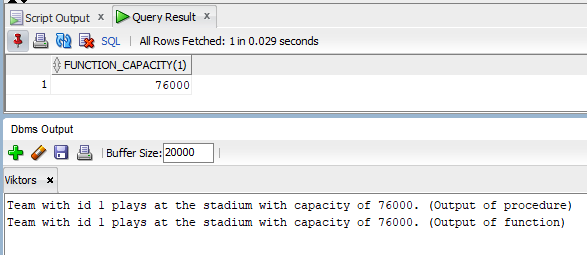
Ar šo WITH priekšrakstu, kurā tika ievietota procedūra un funkcija tika iegūtā komandas ar id 1 stadiona kapacitāte. Būtībā, ir saprotams, ka atšķirība starp procedūru un funkciju ir tā, ka funkcija obligāti atgriež kādu vērtību (kamēr procedūra neatgriež), bet procedūra satur kādu darbības nosacījumus. Ievietojot parametrus procedūrā, tā tiek izpildītā funkcijas ietvaros. Piemērā ir aplūkojams gadījums, kad funkcija par parametru saņēma skaitli n, kas procedūrā tika pielīdzināts pie “team\_id”. Procedūra padod funkcijai vērtību cp, kurā iepriekš bija ievietotas stadiona kapacitātes vērtības.





Būtībā, funkcijai ir jāizvada rezultātus ne tikai konsolē, bet arī ”DBMS OUTPUT” buferī, bet tomēr tas nenotiek. Nedaudz papētot šo trūkumu un iemeslus, es atradu veidu, kā iegūt izvadi arī buferī. To ir iespējams pagūt palaižot ne tikai WITH priekšrakstu, bet arī nodzēšot un pajaunu izveidojot tabulu un ievadot datus. Tādā gadījumā, rezultāts tiek piefiksēts arī “DBMS OUTPUT” buferī.

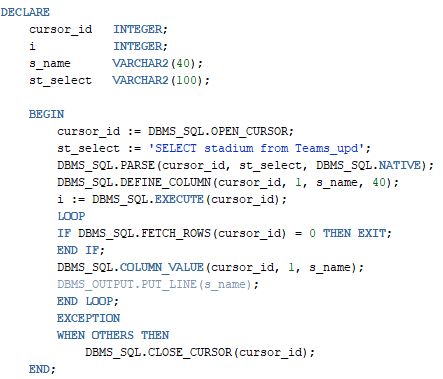


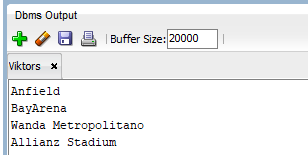


# **Dinamiskais SQL un PL/SQL**

## **DBMS SQL 1. piemērs**

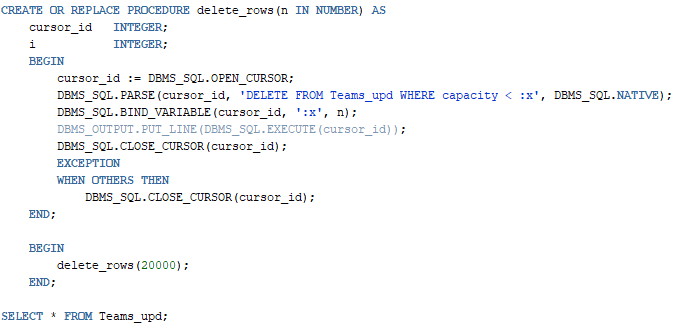
Būtībā, DBMS\_SQL ir programmu pakete, kura nodrošina interfeisu dinamiska SQL izmantošanai. Kopumā sākot, šajā piemērā tiek aplūkota iespēja iekš PL/SQL bloka. Ar vaicājuma palīdzību tika iegūti “jauno” komandu stadionu nosaukumi, kuri tika izvadīti “DBMS OUTPUT” buferī.

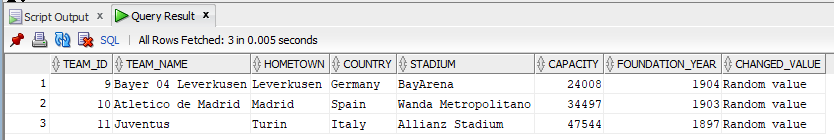




## **DBMS SQL 2. piemērs**

Ar sekojošo vaicājumu, procedūra “delete\_rows” izdzēš tabulas “Teams\_upd” rindas, kur stadionu kapacitāte ir mazāka par 20000. Iegūstot rezultātus ir redzams, ka patiešām tika izdzēsta viena rinda, kuras kapacitātes vērtībā bija redzama iepriekš, funkcijā, kura atgriež mazāko stadiona kapacitāti. Arī, “DBMS OUTPUT” buferī tika izvadīta vērtība 1, kas šajā kontekstā nozīmē izdzēsto rindu skaits.





# **Secinājumi**

Rūpīgi analizējot paveikto darbu varētu secināt, ka kopumā, manuprāt šīs darbs bija visgrūtākais no visiem šajā kursā. Šīs darbs, kā arī visi iepriekšējie aplūko SQL valodas pamatuzbūves vienības, vairākās sarežģītas konstrukcijas, struktūras un to praktisko pielietojumu. Izpildot šo darbu paradījās lieliska iespēja iepazīties aktīvas datubāzes jēdzienu un apzināt tādus terminus kā trigeri un kursori datubāzēs, kā arī saprast funkciju un procedūru darbību datu bāzes kontekstā. Ar pilnu pārliecību varu teikt, ka balstoties uz iemaņām, kuras tika iegūtas izpildot iepriekšējos darbus ļoti palīdzēja darba un to uzdevumu izpratnē un tās zināšanas viennozīmīgi ir iespējas klasificēt kā būtiskas un pielietojamas nākotnē. Darbu, kā iepriekšējos pildīju ORACLE SQL Developer rīkā.

Domājot par problēmām, kuras bija radušies darba izpildes gaitā es personīgi varētu izcelt dinamisko SQL un PL/SQL, jo būtībā sapratu, ka vajadzētu šai tēmai ieguldītu vairāk laika, nekā tas bija izdarīts, bet tā kā šīs darbs aizņēma novērojami vairāk laika salīdzinājumā ar iepriekšējo, kā arī novērojami vairāk, nekā bija plānots, pienākas nedaudz pasteigties.

Kopumā, visi darbi, kuri bija izpildīti šajā kursa ietvaros deva man lielu pamatu tālākai apmācībai, jo būtība pirms šī semestra vienīgā mana pieredze ar datu bāzēm bija MS Access vide vidusskolas līmenī. Izpildot visus darbus tika aplūkoti vairāki informatīvi un interesanti avoti, kuri viennozīmīgi deva iespēju apgūt un saprast radušos jautājumus, jo būtībā tādu avotu un resursu ir ļoti daudz, galvenais ir saprast, kā tos pareizi izprast, lai varētu veiksmīgi pielietot tās zināšanas un to informāciju praktisko uzdevumu izveidē.

# **Literatūra**

1. Prof. J. Eiduks. Lekciju konspekti priekšmetā “Datu bāzes vadības sistēmas”;
2. estudijas.rtu.lv
3. <https://datubaze.wordpress.com/>
4. <https://www.javatpoint.com/oracle-function#:~:text=%E2%86%92%20%E2%86%90%20prev-,Oracle%20Function,later%20in%20the%20same%20block>.
5. <https://www.oracletutorial.com/plsql-tutorial/plsql-varray/>
6. <https://docs.oracle.com/cd/B14117_01/appdev.101/b10807/13_elems042.htm>
7. <https://www.oracletutorial.com/plsql-tutorial/plsql-procedure/>
8. <https://docs.oracle.com/cd/B10501_01/appdev.920/a96590/adg09dyn.htm>
9. <https://oracle-base.com/articles/12c/with-clause-enhancements-12cr1#procedures>
10. <https://www.javatpoint.com/oracle-function#:~:text=%E2%86%92%20%E2%86%90%20prev-,Oracle%20Function,later%20in%20the%20same%20block>.

# **Pielikums**

## **Darba izmantotas datu bāzes SQL kods**

DROP TABLE Teams;

DROP TABLE Teams\_upd;

CREATE TABLE Teams(

team\_id number NOT NULL,

team\_name varchar2(35) NOT NULL UNIQUE,

hometown varchar2(30),

country varchar2(30),

stadium varchar2(40) NOT NULL,

capacity number(6) CHECK(capacity >= 0),

foundation\_year number(4) CHECK (foundation\_year > 0 and foundation\_year <= 2021),

changed\_value varchar2(20),

CONSTRAINT teams\_pk PRIMARY KEY(team\_id)

);

CREATE TABLE Teams\_upd(

team\_id number NOT NULL,

team\_name varchar2(35) NOT NULL UNIQUE,

hometown varchar2(30),

country varchar2(30),

stadium varchar2(40) NOT NULL,

capacity number(6) CHECK(capacity >= 0),

foundation\_year number(4) CHECK (foundation\_year > 0 and foundation\_year <= 2021),

changed\_value varchar2(20)

);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(1, 'Manchester United', 'Manchester', 'England', 'Old Trafford', 76000, 1878);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(2, 'FC Barcelona', 'Barcelona', 'Spain', 'Camp Nou', 99354, 1899);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(3, 'Manchester City', 'Manchester', 'England', 'Etihad Stadium', 55097, 1880);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(4, 'AC Milan', 'Milano', 'Italy', 'San Siro', 80018, 1899);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(5, 'Real Madrid', 'Madrid', 'Spain', 'Estadio Santiago Bernabeu', 81044, 1902);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(6, 'FC Bayern Munich', 'Munich', 'Germany', 'Allianz Arena', 75000, 1900);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(7, 'Internazionale', 'Milano', 'Italy', 'Stadio Giuseppe Meazza', 80018, 1908);

SELECT \* FROM Teams;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Kursors

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

CURSOR cursor\_team\_name IS

SELECT team\_name, country

FROM Teams

WHERE foundation\_year <= 1900;

TYPE type\_1 IS VARRAY(7) OF cursor\_team\_name%rowtype;

name\_type type\_1 := type\_1();

i NUMBER DEFAULT 1;

BEGIN

FOR r IN cursor\_team\_name LOOP

name\_type.extend;

name\_type(i) := r;

DBMS\_OUTPUT.PUT\_LINE(name\_type(i).team\_name || ' ' || name\_type(i).country);

i := i + 1;

END LOOP;

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Insert tipa trigeris

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE OR REPLACE TRIGGER insert\_record

BEFORE INSERT ON Teams FOR EACH ROW

WHEN(new.team\_id > 0)

BEGIN

IF :new.capacity = 1 THEN

:new.capacity := DBMS\_RANDOM.VALUE(10000, 50000);

:new.changed\_value := 1;

ELSIF :new.capacity = 2 THEN

:new.capacity := DBMS\_RANDOM.VALUE(50000, 100000);

:new.changed\_value := 2;

ELSE :new.capacity := DBMS\_RANDOM.VALUE(10000, 100000);

:new.changed\_value := 3;

END IF;

END;

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(8, 'Liverpool FC', 'Liverpool', 'England', 'Anfield', 1, 1882);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(9, 'Bayer 04 Leverkusen', 'Leverkusen', 'Germany', 'BayArena', 2, 1904);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(10, 'Atletico de Madrid', 'Madrid', 'Spain', 'Wanda Metropolitano', 3, 1903);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(11, 'Juventus', 'Turin', 'Italy', 'Allianz Stadium', DBMS\_RANDOM.VALUE(1, 2), 1897);

SELECT \* FROM Teams;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Update tipa trigeris

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE OR REPLACE TRIGGER update\_table

AFTER UPDATE ON Teams

FOR EACH ROW

BEGIN

INSERT INTO Teams\_upd VALUES(:new.team\_id, :new.team\_name, :new.hometown, :new.country, :new.stadium, :new.capacity, :new.foundation\_year, 'Random value');

END;

UPDATE Teams SET capacity = DBMS\_RANDOM.VALUE(10000, 20000) WHERE team\_id = 8;

UPDATE Teams SET capacity = DBMS\_RANDOM.VALUE(20000, 30000) WHERE team\_id = 9;

UPDATE Teams SET capacity = DBMS\_RANDOM.VALUE(30000, 40000) WHERE team\_id = 10;

UPDATE Teams SET capacity = DBMS\_RANDOM.VALUE(40000, 50000) WHERE team\_id = 11;

SELECT \* FROM Teams;

SELECT \* FROM Teams\_upd;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function example

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE OR REPLACE FUNCTION man\_utd\_name

RETURN VARCHAR IS

tn VARCHAR(35);

BEGIN

SELECT team\_name INTO tn

FROM Teams

WHERE team\_id = 1;

RETURN tn;

END;

/

CREATE OR REPLACE FUNCTION man\_utd\_hometown

RETURN VARCHAR IS

ht VARCHAR(30);

BEGIN

SELECT hometown INTO ht

FROM Teams

WHERE team\_id = 1;

RETURN ht;

END;

/

CREATE OR REPLACE FUNCTION man\_utd\_country

RETURN VARCHAR IS

ct VARCHAR(30);

BEGIN

SELECT country INTO ct

FROM Teams

WHERE team\_id = 1;

RETURN ct;

END;

/

CREATE OR REPLACE FUNCTION man\_utd\_stadium

RETURN VARCHAR IS

st VARCHAR(40);

BEGIN

SELECT stadium INTO st

FROM Teams

WHERE team\_id = 1;

RETURN st;

END;

/

CREATE OR REPLACE FUNCTION man\_utd\_capacity

RETURN NUMBER IS

cp NUMBER;

BEGIN

SELECT capacity INTO cp

FROM Teams

WHERE team\_id = 1;

RETURN cp;

END;

/

DECLARE

n VARCHAR(35);

h VARCHAR(30);

c VARCHAR(30);

s VARCHAR(40);

cp NUMBER;

BEGIN

n := man\_utd\_name();

h := man\_utd\_hometown();

c := man\_utd\_country();

s := man\_utd\_stadium();

cp := man\_utd\_capacity();

DBMS\_OUTPUT.PUT\_LINE(n || ' is a team based in ' || h || ', ' || c || '. ');

DBMS\_OUTPUT.PUT\_LINE(n || ' plays at the stadium called ' || s || '. Its total capacity is estimated at ' || cp || '. ');

END;

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

WITH for two functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

WITH

FUNCTION max\_capacity RETURN NUMBER IS

maxc NUMBER;

BEGIN

SELECT MAX(capacity) INTO maxc

FROM Teams

WHERE country IN ('England', 'Germany');

RETURN maxc;

END;

FUNCTION min\_capacity RETURN NUMBER IS

minc NUMBER;

BEGIN

SELECT MIN(capacity) INTO minc

FROM Teams

WHERE country IN ('England', 'Germany');

RETURN minc ;

END;

FUNCTION english\_and\_german\_teams RETURN NUMBER IS

gt NUMBER;

BEGIN

SELECT COUNT(team\_id) INTO gt

FROM Teams

WHERE country IN ('England', 'Germany');

RETURN gt;

END;

SELECT max\_capacity(), min\_capacity(), english\_and\_german\_teams()

FROM DUAL;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

WITH function and procedure

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DROP TABLE Teams;

CREATE TABLE Teams(

team\_id number NOT NULL,

team\_name varchar2(35) NOT NULL UNIQUE,

hometown varchar2(30),

country varchar2(30),

stadium varchar2(40) NOT NULL,

capacity number(6) CHECK(capacity >= 0),

foundation\_year number(4) CHECK (foundation\_year > 0 and foundation\_year <= 2021),

changed\_value varchar2(20),

CONSTRAINT teams\_pk PRIMARY KEY(team\_id)

);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(1, 'Manchester United', 'Manchester', 'England', 'Old Trafford', 76000, 1878);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(2, 'FC Barcelona', 'Barcelona', 'Spain', 'Camp Nou', 99354, 1899);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(3, 'Manchester City', 'Manchester', 'England', 'Etihad Stadium', 55097, 1880);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(4, 'AC Milan', 'Milano', 'Italy', 'San Siro', 80018, 1899);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(5, 'Real Madrid', 'Madrid', 'Spain', 'Estadio Santiago Bernabeu', 81044, 1902);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(6, 'FC Bayern Munich', 'Munich', 'Germany', 'Allianz Arena', 75000, 1900);

INSERT INTO Teams(team\_id, team\_name, hometown, country, stadium, capacity, foundation\_year) VALUES(7, 'Internazionale', 'Milano', 'Italy', 'Stadio Giuseppe Meazza', 80018, 1908);

WITH

PROCEDURE procedure\_capacity(n IN NUMBER, cp OUT NUMBER) IS

BEGIN

SELECT capacity INTO cp

FROM Teams

WHERE team\_id = n;

DBMS\_OUTPUT.put\_line('Team with id ' || n || ' plays at the stadium with capacity of ' || cp || '. (Output of procedure)');

END;

FUNCTION function\_capacity(n IN NUMBER) RETURN NUMBER IS

c NUMBER;

BEGIN

procedure\_capacity(n, c);

DBMS\_OUTPUT.put\_line('Team with id ' || n || ' plays at the stadium with capacity of ' || c || '. (Output of function)');

RETURN c;

END;

SELECT function\_capacity(1)

FROM DUAL;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dynamic PL/SQL - DBMS SQL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DECLARE

cursor\_id INTEGER;

i INTEGER;

s\_name VARCHAR2(40);

st\_select VARCHAR2(100);

BEGIN

cursor\_id := DBMS\_SQL.OPEN\_CURSOR;

st\_select := 'SELECT stadium from Teams\_upd';

DBMS\_SQL.PARSE(cursor\_id, st\_select, DBMS\_SQL.NATIVE);

DBMS\_SQL.DEFINE\_COLUMN(cursor\_id, 1, s\_name, 40);

i := DBMS\_SQL.EXECUTE(cursor\_id);

LOOP

IF DBMS\_SQL.FETCH\_ROWS(cursor\_id) = 0 THEN EXIT;

END IF;

DBMS\_SQL.COLUMN\_VALUE(cursor\_id, 1, s\_name);

DBMS\_OUTPUT.PUT\_LINE(s\_name);

END LOOP;

EXCEPTION

WHEN OTHERS THEN

DBMS\_SQL.CLOSE\_CURSOR(cursor\_id);

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dynamic PL/SQL - DBMS SQL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE OR REPLACE PROCEDURE delete\_rows(n IN NUMBER) AS

cursor\_id INTEGER;

i INTEGER;

BEGIN

cursor\_id := DBMS\_SQL.OPEN\_CURSOR;

DBMS\_SQL.PARSE(cursor\_id, 'DELETE FROM Teams\_upd WHERE capacity < :x', DBMS\_SQL.NATIVE);

DBMS\_SQL.BIND\_VARIABLE(cursor\_id, ':x', n);

DBMS\_OUTPUT.PUT\_LINE(DBMS\_SQL.EXECUTE(cursor\_id));

DBMS\_SQL.CLOSE\_CURSOR(cursor\_id);

EXCEPTION

WHEN OTHERS THEN

DBMS\_SQL.CLOSE\_CURSOR(cursor\_id);

END;

BEGIN

delete\_rows(20000);

END;

SELECT \* FROM Teams\_upd;