Database Systems
Lab9

DCL (Data Control Language)

(GRANT and REVOKE)

Important command

- Use the following command whenever you want to work with users.

```
alter session set "_ORACLE_SCRIPT"=true;
```

Create/manipulate/delete users

- Create new user:
  - Create user user_name identified by user_password;
  - (GRANT CREATE SESSION TO user_name;)
- Change user password:
  - Alter user user_name identified by new_password;
- Lock user account:
  - Alter user user_name account lock;
- Unlock user account:
  - Alter user user_name account unlock;
- Delete user:
  - Drop user user_name;

Select data from other users:

Scott

Select * from user1.table1;

User1

Owner: table1

SQL> scott/tiger

Create user userA identified by abcd123;

Conn userA/abcd123;

Privileges

System privileges

Examples: create session, create table, create user

Command:

- Grant create table to scott;
- Grant create table, create session to scott;
- Grant all privileges to scott;
- Revoke create table from userA;

Object privileges

Examples: select, update, delete, reference

Command:

- Grant select on employee to userA;
- Grant select,update on employee to userA;
- Grant select on employee to userA;

Grant update on students to userA;

- Grant update(Major) on student to userA;
- Grant all on employee to userA;
- Grant select on student to public;
- Revoke delete on employee from userA;

Table space

- Grant unlimited tablespace to userA;

With grant option

- Create user user1 identified by u123;
- Grant create session to user1; with grant option;
- Grant create table to use1 with grant option;
- Grant unlimited table space to user1;
- Grant create user to user1;
- Conn user1/u123;
- Create table …
- Create user user2 identified by abc;
- Grant create session, create table to user2;

Roles

- Create role manager;
- Grant update on students to manager;
- Grant all on employee to manager;
- Grant create table, create session to manager;
- Grant unlimited tablespace to manager;
- Grant manager to userA;
- Drop role manager;

Synonyms

What is a synonym?

Oracle synonyms are aliases created for DB objects (like tables) in order to be easily used by users other than ADMIN.

Why do we need synonyms?

Example: Consider a table emp created by admin user ‘system’. Another user ‘abc’ can select the table as:

Select * from system.emp

(if no synonym is created)

Select * from emp

(if a synonym is created for table emp with the same name)

Why do we need synonyms?

- Admin user can be changed from a site to site. So, using the fixed name of the admin user in a select statement may need to be changed if the database is installed for a different client.

- If a user is granted a full privilege on a table, the user can use the table directly without the name of the admin user. But, most of the time limited privileges are given to user, and thus synonyms are created for this purpose.

CREATE SYNONYM

To create an Oracle synonym:

CREATE [OR REPLACE] [PUBLIC] SYNONYM [synonym_name] for [object_name];

- If [PUBLIC] is not use, you the created synonym can only be used by the owner (user).

How synonyms are used?

- Synonym name can be used as the original object. i.e. you can apply any operation allowed on the original object by using its synonym.

- Users other than the synonym owner can use it, if they are granted limited privileges, such as: SELECT, INSERT, …etc on tables.

- Users other than the synonym owner can use it and also they can use the original object name if they are granted ALL privileges on this object.

DROP SYNONYM

- DROP keyword is used to delete a synonym

DROP [PUBLIC] SYNONYM [synonym_name]

Introduction to PL/SQL

What is PL/SQL?

- Procedural programming language
  - Uses detailed instructions
  - Processes statements sequentially
- Combines SQL commands with procedural instructions
- Used to perform sequential processing using an Oracle database
- PL/SQL supports variables, conditions, loops and exceptions.
- PL/SQL blocks can include control flow and DML statements.

When is PL/SQL useful?

- When something is too complicated for SQL
- When conditional branching and looping are needed
- Very useful to develop code for a DB transaction

Basic Structure

```
DECLARE
   Variable declarations
BEGIN
   Program statements
EXCEPTION
   Error-handling statements
END;
```

Variable Declarations

Body

Exception Section

PL/SQL Program Lines

- May span multiple text editor lines
- Each line ends with a semicolon
- Text is not case sensitive

Comment Statements

- Block of comments are delimited with /* */

/* <comment that spans more than one line of code> */

- Single comment line starts with 2 hyphens

-- comment on a single line

Variables

- Variables can have:
  - any SQL data type, such as CHAR, DATE, or NUMBER
  - or any PL/SQL data type, such as BOOLEAN or BINARY_INTEGER.
  - Reference data types:
    - Reference a database item
    - Assume data type of item
      - %TYPE: assumes data type of field
      - %ROWTYPE: assumes data type of entire row

Variables

- Syntax for declaring a variable:

  variable_name *data_type_declaration*;

- Examples:
  - part_no NUMBER(4);
  - in_stock BOOLEAN;

Arithmetic Operations

**	Exponentiation	2 ** 3	8
*	Multiplication	2 * 3	6
/	Division	9/2	4.5
+	Addition	3 + 2	5
-	Subtraction	3 – 2	1
-	Negation	-5	Negative 5

Assigning values to variables

- **First:** Assignment Statement
  - Assignment operator: **:=**
  - Variable being assigned to a new value is on left side of assignment operator
  - New value is on right side of operator

Examples:

student_name := 'John Miller';

student_name := current_student;

tax := price * tax_rate;

Assigning values to variables

- **Second:** Selecting database value into a variable.

Examples:

```
SELECT FNAME, DNO
INTO V_FNAME, V_DNO
FROM EMPLOYEE
WHERE SSN = '1111111';
```

```
SELECT SALARY * 0.10
INTO   bonus
FROM   EMPLOYEE
WHERE  SSN = SSN_In;
```

Note: using INTO, enforce the developer to retrieve one-row only in a SELECT statement.

Displaying PL/SQL Output in SQL Plus

- Command to activate memory buffer in SQL*Plus to enable output from PL/SQL programs:

SQL> SET SERVEROUTPUT ON

- Command to output data from a PL/SQL program in SQL*Plus:

DBMS_OUTPUT.PUT_LINE('*output string*');
DBMS_OUTPUT.PUT_LINE('Employee Salary: '|| Salary);

- - || is a concatenation operator

Executing a PL/SQL Program in SQL Plus

- Copy program code from Notepad to SQL*Plus
- Type / to execute

Character Functions in PL/SQL

- Concatenating strings: joining 2 or more character strings into a single string

- Concatenation operator: ||

```
s_first_name := 'Sarah'
s_last_name := 'Miller'
s_full_name := s_first_name || ' ' ||
s_last_name
```

Character Functions in PL/SQL

- These functions were discussed before, but here are some examples using them in PL/SQL

RTRIM: removes blank trailing spaces

```
cust_address := RTRIM(cust_address);
```

LENGTH: returns string length (number of characters)

```
address_length := LENGTH(cust_address);
```

UPPER, LOWER: changes characters to all upper or lower case

```
s_name := UPPER(s_name);
s_name := LOWER(s_name);
```

Character Functions in PL/SQL

INSTR: searches a string and looks for a matching substring and returns its starting position

starting_position := INSTR(string_being_searched, search_string>);

blank_position := INSTR('Sarah Miller', ' ');

SUBSTR: extracts a specific number of characters from a string, starting at a given point

extracted_string := SUBSTR(*string_being_searched*, *starting_point*, *number_of_characters_to_extract*);

s_first_name := SUBSTR('Sarah Miller', 1,5);

Note: Most single-row functions can be used in PL/SQL in the same manner

NULL values in Assignment Statements

- Until a value is assigned to a variable, the variable's value is NULL
- Performing an arithmetic value on a NULL value always results in a NULL value
- Advice: Always initialize variable values