

Databases

(Bases de Datos, BDA)

Escuela Técnica Superior de Informática Aplicada

Facultad de Informática

Lab exercise book n° 3: SQL (2nd part)

Data Definition

1. Introduction	2
2. Data definition in Oracle SQL.....	2
1.1. Definition of a basic relation.....	2
1.2. Modification of tables	4
1.3. View definition	4
1.4. Definition of privileges	5
1.5. Trigger definition	5
3. Data manipulation in Oracle SQL	7
4. Oracle Catalogue and other useful commands	7
5. Exercises:.....	8
6. Appendix: PL/SQL Language	9

1. Introduction

The data definition language of standard SQL can be found in many database books and online manuals. In this exercise book, we only present part of the SQL data definition language in Oracle, highlighting its differences with respect to the standard languages. In the same way, we present some particularities of the data manipulation language in Oracle.

2. Data definition in Oracle SQL

The Oracle DBMS does not provide the concept of database schemas as it appears in the standard SQL language. In Oracle, we have a database which is associated with each user. In each user's database we store all the objects created by the user (tables, views, procedures, etc.).

From all the elements that can be included in a standard SQL schema, Oracle allows the definition of the following objects: basic relations, views, and access privileges (that means that we cannot define domains). Additionally, we can define some details which are related to the physical data representation (index, *tablespace*, *cluster*...); triggers to model some active behaviour; as well as other programming elements (functions and procedures, procedure packages, ...).

1.1. Definition of a basic relation

```
Basic_relation_definition ::= CREATE TABLE relation_name
                             (table_element1, table_element2, ... )
```

```
table_element ::= attribute_definition
                | relation_constraint
```

```
attribute_definition ::= attribute_name data_type
                        [DEFAULT (expression)]
                        [attribute_constraint1, attribute_constraint2, ...]
```

```
data_type ::= CHAR (length)
            | VARCHAR2 (length)
            | NUMBER [(precision[,scale])]
            | DATE
```

```
attribute_constraint ::= [CONSTRAINT constraint_name]
                       {[NOT] NULL
                        | UNIQUE
                        | PRIMARY KEY
                        | REFERENCES relation_name * [(attribute_name *)]
                          [ON DELETE {CASCADE|SET NULL}]
                        | CHECK (condition)}
                       [when_to_check]
```

```

relation_constraint ::=
  [CONSTRAINT constraint_name]
    { UNIQUE (attribute_name1, attribute_name2, ...)
    | PRIMARY KEY (attribute_name1, attribute_name2, ...)
    | FOREIGN KEY (attribute_name1, attribute_name2, ...)
      REFERENCES relation_name [(attribute_name1, attribute_name2, ...)]
        [ON DELETE {CASCADE|SET NULL}]
    | CHECK (condition)}
    [when_to_check]

when_to_check ::=
  [NOT] DEFERRABLE [INITIALLY {IMMEDIATE | DEFERRED}]

```

The available data types are:

Numeric:

- NUMBER [(precision[, scale])] where "precision" is the total number of digits and "scale" is the number of decimal digits.
- integers: NUMBER (precision)
- real: NUMBER (precision, scale)

Alphanumerical:

- of fixed length: CHAR(length)
- of variable length: VARCHAR(length)

Dates:

- DATE

The expression of the DEFAULT clause is constructed from the constants of predefined data types, their operators and system functions, with the appropriate syntax in each case. The "expression" cannot refer to other attributes in the relation. The type of the expression must match the data type of the attribute which is included in the clause.

Oracle only includes the weak referential integrity and the possibility of including the referential integrity restoration directives: *delete in cascade* and *to nulls*.

The condition of the CHECK clause is a logic expression which is defined with the same syntax as the WHERE clause of the SELECT instruction with the following constraint: it can only refer to attributes of the relation where the constraint is defined and cannot include subqueries or aggregated functions. The relation complies with the integrity constraint which is defined with the CHECK clause if for every tuple the condition is evaluated to true or undefined (due to the presence of null values).

The directive *when_to_check* has the same meaning as in standard SQL.

In the definition of a basic relation in ORACLE, we can include other clauses which do not appear in the syntax of standard SQL in order to define some other details (which are specific to Oracle) for the physical representation of the relation.

In order to delete a relation we use the sentence DROP TABLE *relation_name*.

1.2. Modification of tables

```

table_modification ::= ALTER TABLE relation_name
                    {ADD (table_element1, table_element2, ...)
                     | MODIFY (attribute_definition1, attribute_definition2, ...)
                     | DROP COLUMN (attribute_name1 , attribute_name2, ...)
                       [CASCADE]
                     | {DROP
                       | [VALIDATE | NOVALIDATE] ENABLE
                       | DISABLE } (constraint) }

constraint ::= {PRIMARY [CASCADE]
               | UNIQUE (attribute_name1, attribute_name2, ...) [CASCADE]
               | CONSTRAINT constraint_name }

```

The option ADD allows new attributes or constraints to be added to a relation.

The option MODIFY allows the definition of some attributes to be modified.

The option ENABLE (respectively DISABLE) allows an integrity constraint to be enabled (respectively disabled). With the option VALIDATE (the default option), the system ensures that when a constraint is enabled the data which are stored in the database must comply with the constraint at the moment of enabling.

The option DROP allows the user to delete attributes or integrity constraints. The option CASCADE deletes in cascade every integrity constraint in the schema which depends on the element which is being deleted.

1.3. View definition

```

view_definition ::= CREATE [OR REPLACE] VIEW view_name
                  [(attribute_name1, attribute_name2, ...)] AS SELECT_instruction
                  [WITH CHECK OPTION]

```

Since a view can be defined through any SELECT, the translation of a modification on the view to the basic tables from which it is defined is not always possible since there can be ambiguities. Because of this, the modification of views is bound to some constraints which avoid these possible ambiguities.

These constraints are in ORACLE:

- a) A view which is defined by a SELECT that contains set operators (UNION, INTERSECT,...), the operator DISTINCT, aggregated functions (SUM, AVG, ...) or the clause GROUP BY is not updatable.
- b) If the view is defined over a single basic relation the system will translate the update over the view into an update operation over the basic relation if no integrity constraint is violated by the update.
- c) If the view is defined over a combination of relations, the update will be bound to the following additional constraints:
 - The update can only modify one of the basic relations.

- The update will modify the basic relation which has the property of *key preservation*, i.e., that relation such that its primary key could also be the key of the view if their attributes would be selected by the SELECT which defines the view.
- The update will not be performed if some of the constraints which are defined over the basic relation which is to be updated.

1.4. Definition of privileges

The owner of the database schema is the owner of all the objects that are defined in it. In order to allow other users to perform operations over the objects in the database, the user must have the appropriate authorisation; these authorisations must be given by the object's owner through the GRANT sentence.

```
grant_command_definition ::= GRANT privilege_system1, privilege_system2,...
                             ON object
                             TO {PUBLIC | user1, user2, ...}
                             [WITH GRANT OPTION]
```

- *user* is a user's identifier.
- With the clause PUBLIC the privilege is transmitted to all users.
- The clause WITH GRANT OPTION gives the permission to grant the obtained privilege to third people.

The privileges which can be granted are: create, delete or modify elements in the schemas: tables, indices, views, etc.

1.5. Trigger definition

In Oracle, the definition of a trigger is as follows.

```
trigger_definition ::=
{CREATE | REPLACE} TRIGGER trigger_name
  {BEFORE | AFTER | INSTEAD OF} event1 [OR event2 [OR ...]]
  ON {relation_name | view_name}
  [REFERENCING OLD AS reference_name [NEW AS reference_name]]
  [FOR EACH ROW [WHEN (condition)]]
  PL/SQL block

event ::= INSERT | DELETE | UPDATE [OF attribute_name1, attribute_name2, ...]
```

the *event* is any modification over the database; the *condition* is a logic expression which is written in SQL syntax and the action is a procedure written in the programming language PL/SQL. In this PL/SQL block we can include database manipulation instructions, error handling or data input/output.

By combining the options {BEFORE | AFTER} and FOR EACH ROW¹ we can define four kinds of triggers with different semantics:

	without FOR EACH ROW	FOR EACH ROW
AFTER	the trigger is executed once and after the execution of the update operation	the trigger is executed after and for each tuple affected by the update operation
BEFORE	the trigger is executed once and before the execution of the update operation	the trigger is executed before and for each tuple affected by the update operation

The execution of the event that activates the trigger is inserted between the executions of the triggers which are activated by the event, according to the semantics of the execution defined by each kind of trigger. The actions included in the executed triggers extend the transaction which includes the event that motivated the trigger. In what follows, we include some explanations about the three parts of a trigger as understood by Oracle:

Event:

- Only the events of the triggers of the kind FOR EACH ROW (tuple oriented) can be parameterised.
- The event parameterisation is implicit: 2*n parameters if the event is UPDATE and n parameters if the event is INSERT or DELETE (with n being the degree of the relation *relation_name* or the view *view_name*).
- The name of these parameters is the name of the relation attribute but preceded by the preserved word OLD if the event is DELETE or UPDATE and the preserved word NEW if the event is INSERT or UPDATE (these preserved words can be replaced by their corresponding *reference_name* which can be expressed in the REFERENCING clause).
- These parameters can be used in the condition.
- These parameters can also be used in the PL/SQL block. Here, the preserved words OLD and NEW must be preceded by a colon.

Condition:

- The condition is a logic expression which follows the syntax of the condition in a WHERE clause in a SELECT sentence, with the following constraints:
 - it cannot contain subqueries or aggregated functions
 - it can only refer to the event parameters
 (these constraints reduce the possibilities of the condition and hence many conditions are expressed in the PL/SQL block as IF/THEN/ELSE).
- If the condition refers to the event parameters, these are instantiated when the trigger is enabled.

¹ The absence of the clause FOR EACH ROW is equivalent to the clause FOR EACH STATEMENT in standard SQL.

Action:

- The PL/SQL block is a procedure which is written in a proprietary programming language by Oracle, whose syntax is included in the appendix.
- Database manipulation sentences: INSERT, DELETE, UPDATE, SELECT... INTO...
- Program sentences: Assignment, selection, iteration.
- Instructions for error handling and data input/output.
- These instructions can be structured by using the control structures in the PL/SQL language (selection, iteration, etc.). (see appendix)
- In the PL/SQL block we cannot include data definition sentences or instructions for transaction control (COMMIT, ROLLBACK, SAVEPOINT).
- The action of a trigger consists of a sequence of instructions defined by the structure of the PL/SQL block.
- The event parameter can be used in the PL/SQL block.

Comments:

- If the trigger is of the kind BEFORE *event* FOR EACH ROW, then the action of the trigger (PL/SQL block) can make an assignment over NEW.*attribute_name*.
- If the trigger can be activated by more than one kind of event, the PL/SQL block can use the logical predicates INSERTING, DELETING and UPDATING [*attribute_name*] to program different actions according to the kind of event which activated the trigger.
- In ORACLE the action of a trigger (PL/SQL block) cannot query or update the table which is modified by the trigger event.

Apart from the sentence for the creation of triggers, the ORACLE DDL provides the following sentences:

- DROP TRIGGER *trigger_name*: deletes the trigger.
- REPLACE TRIGGER *trigger_name*: modifies the definition of the trigger.
- ALTER TRIGGER *trigger_name* COMPILE: recompiles the trigger.
- ALTER TRIGGER *trigger_name* [ENABLE | DISABLE] ALTER TABLE *relation_name* [{ENABLE | DISABLE} ALL TRIGGERS]: enables or disables the triggers.

3. Data manipulation in Oracle SQL

The sentences of data manipulation in ORACLE SQL are the same as the standard SQL, with the following exceptions which are related to the set operators:

- The SQL operator **EXCEPT** is called **MINUS** in Oracle.
- Except from **UNION ALL**, all the set operators eliminate duplicate rows.

4. Oracle Catalogue and other useful commands

Like many other database systems, Oracle provides the user with a system dictionary or catalogue. This catalogue is a set of system tables and views which can be queried to see the objects which are defined in the database. In particular, we will need the following views for the exercises which are set in the following section.

- TABLES: We can check the tables which are defined by the user through


```
SELECT * FROM USER_TABLES;
```
- CONSTRAINTS: We can check the constraints which are defined by the user through

```
SELECT * FROM USER_CONSTRAINTS;
```

- TABLE ATTRIBUTES: We can check the attributes of a table through

```
DESCRIBE Table_Name;
```

- TRIGGERS: We can check the triggers which are defined by the user through

```
SELECT * FROM USER_TRIGGERS;
```

If we want to query all the objects instead of only the user's objects, we can use ALL instead of USER. I.e. to query all the tables in the database (owned or not by the user) we will use SELECT * FROM ALL_TABLES. Additionally, we can use DBA instead of USER to query the objects that can be modified by the database administrator, i.e. all the objects except the system objects.

Finally, when compiling triggers we can use the following command:

```
SHOW ERROR name;
```

5. Exercises:

We want to design a database to manage a small library for a university department. After analysing the system, we have identified the requirements which are more frequently performed; these are:

- Query the book information: book code, title, author(s), topic and, in the case of being borrowed, the user who has borrowed it.
- Query the user information: user code, name, address, telephone and books which are currently borrowed by the user, as well as the date it was borrowed.
- Query the historical loans made by a user: book code, date of loan and date of return.
- Insert, delete and modify the data for a user.
- Manage the loans: lend a book to a user and record the return date of a book by a user.

Some integrity constraints which have been detected are:

- The code of the book identifies the book uniquely.
- The code of the user identifies the user uniquely.
- The set of topics used to classify a book are: *physics, electricity, mechanics and optics*.
- The return date for a book must be later to the date of the loan.
- The total number of books which a user has at a certain same time is a *derived field* which will be automatically maintained by the system.

Perform the following tasks:

- Define the relational schema for the previous database (using the concept of the relational model).
- Define the database using ORACLE.
- Perform some insertions, updates and queries over the database.

6. Appendix: PL/SQL Language

Structure of a PL/SQL block:

DECLARE

Section for variable
declarations

BEGIN

Block sentences

END

Section for variable declarations:

- Local block variables:

```
variable_name datatype
datatype ::= {NUMBER | CHAR() | DATE }
```

(the declaration in this section must be separated by semicolon)

Sentences in the body of the PL/SQL block:

```
sequence_of_sentences ::= sentence; [sentence;] ...
```

- Selection sentence:

```
IF condition THEN sequence_of_sentences
    [ELSE sequence_of_sentences] END IF;
```

- Loop sentences:

```
WHILE condition LOOP
sequence_of_sentences ;
END LOOP;
```

```
FOR counter IN minimum ... maximum LOOP
sequence_of_sentences ;
END LOOP;
```

- Assignment :

```
variable_name := expression
```

- SQL sentences: **INSERT, DELETE, UPDATE, SELECT... INTO.....**

- Input-output sentences: `dbms_output.put_line ('message')`. In order to use this function, the package `dbms_output` must be enabled. This is made through the sentence **SQL SET SERVEROUTPUT ON**

Error handling:

If, during the execution of a trigger, an error appears (either predefined or defined by the user), then every modification which has been performed by the action in the trigger, as well as the event which activated it, are cancelled.

The sentence **RAISE_APPLICATION_ERROR** (*no_error*, 'message') causes the appearance of an internal error with number *no_error* and sends the user the message 'message'. *no_error* must be a negative number between -20000 and -20999).