DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Escuela Técnica Superior de Ingeniería Informática

etsinf

# Databases and Information Systems

## Degree in Informatics Engineering

## Lab. Sessions Part II:

## SQL Data Definition Language and Transactions in Oracle

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

V. 21.2

# Contenido

# 1. Introduction

Only the ORACLE SQL Data Definition Language is presented in this document, emphasizing its differences with standard SQL. Also, some particularities of the ORACLE Data Manipulation Language are presented.

# 2. SQL Data Definition Language in ORACLE.

There is no database schema concept in the ORACLE system as in the standard SQL language. Associated to each user defined in the system, a database is created in which all the objects (tables, views, procedures,...) created by the user will be stored.

Of all the elements that can be included in a standard SQL schema, the following can be defined in the ORACLE system: basic tables (relations), views, and access privileges (domains cannot be defined). In addition, some details related to the internal representation of data (indexes, tablespace, cluster...); triggers to model active behavior; as well as programming elements (functions and procedures, procedure packages,...) can be defined.

## 2.1. Defining a table

```
Table_definition  ::=  CREATE TABLE  table_name
                              (Table_element1, Table_element2, ... )

Table_element  ::= Colum_definition
                          | Table_constraint

Colum_definition ::= column_name  Data_type
                          [DEFAULT (value | NULL)]
                          [Column_constraint1 Column_constraint2 ...]

Data_type ::=  CHAR (length)
               | VARCHAR2 (length)
               | NUMBER [(percision[, scale])]
               | DATE
```

```
Column_constraint::=  [CONSTRAINT constraint_name ]
                       {[NOT] NULL
                        | UNIQUE
                        | PRIMARY KEY
                        | REFERENCES table_name* [(colum_name*)]
                                  [ON DELETE {CASCADE  |  SET NULL}]
                        | CHECK (condition)}
                       [When_check]
```

```
Table_constraint ::=
[CONSTRAINT constraint_name]
        { UNIQUE (column_name1, column_name2, ...)
        | PRIMARY KEY (column_name1, column_name 2, ...)
        | FOREIGN KEY (column_name 1, column_name 2, ...)
            REFERENCES tabla_name [(column_name 1, column_name 2, ...)]
                                        [ON DELETE {CASCADE | SET NULL}]
        | CHECK (condition)}
        [When_check]

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

The available data types are:

Numeric:

a) NUMBER [(precision [, scale])] where precision is the total number of digits and scale is the number of decimal digits.
b) integer: NUMBER (precision)
c) real: NUMBER (precision, scale)

Alphanumeric:

d) fixed length: CHAR (length)
e) variable length: VARCHAR (length)

Dates:

f) DATE

The DEFAULT clause expression is constructed from constants of the predefined data types, operators, and system functions following the appropriate syntax. No other attributes of the table can be referenced in the expression. The type of the expression must match the data type of the column in which the clause is included.

In ORACLE, only the "weak referential integrity" type and the possibility of including the CASCADE or SET TO NULL referential integrity recovery directives are accepted.

The condition of the CHECK clause is a logical expression that is defined with the same syntax as the condition of the WHERE clause of the SELECT statement with the following limitations: only attributes of the table on which the constraint is defined can be referenced, and no subqueries or aggregate functions can be included. The table satisfies the integrity constraint defined in the CHECK clause if, for all tuples, the condition evaluates to true or undefined (due to the presence of null values).

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

Other clauses that do not appear in the standard SQL syntax can be included in the definition of a basic table in ORACLE to specify details of the physical representation of the table.

To delete a table, the DROP TABLE *table_name* statement is used.

### 2.2. Updating a table definition

```
Update_a_table ::= ALTER TABLE table_name
                   { ADD (Column_definition1, Column_definition2, ...)

                    | MODIFY (Column_definition1, Column_definition2, ...)
                     | DROP COLUMN (column_name1 , column_name2, ...)
                           [CASCADE]
                    | {DROP
                    | [VALIDATE | NOVALIDATE] ENABLE
                    | DISABLE } (constraint_name) }

Column_constraint ::= {PRIMARY  [CASCADE]
            | UNIQUE (column_name1, column_name2, ...) [CASCADE]
            | CONSTRAINT constraint_name }
```

The ADD option allows adding new columns or constraints to a table.

The MODIFY option allows you to modify the definition of columns.

The ENABLE (or DISABLE) option allows to activate (or deactivate) an integrity constraint. With the VALIDATE option (default option), the system ensures that when activating a constraint, the data stored in the database at the time of activation satisfies the constraint.

The DROP option allows to delete columns or integrity constraints. The CASCADE option deletes in cascade any integrity constraints defined in the schema that depends on the deleted element.

### 2.3. Defining a View

```
View_definition  ::= CREATE [OR REPLACE] VIEW view_name
                [(column_name1, column_name2, ...)] AS sentence_SELECT
                [WITH CHECK OPTION]
```

Because a view can be defined through any SELECT statement, the translation of an update operation on the view into updates on basic tables is not always possible since ambiguities may arise. For this reason, the update of views is subject to certain restrictions that avoid this possible ambiguity.

The restrictions in ORACLE are:

a) A view defined by a SELECT containing set operators (UNION, INTERSECT, ...), the DISTINCT operator, aggregate functions (SUM, AVG, ...) or the GROUP BY clause is not updateable.
b) If the view is defined on a single basic table, the system will translate the update on the view into an update operation on the basic table as long as no integrity constraint defined on that table is violated.
c) If the view is defined on a concatenation (join) of tables, the update is subject to the following additional restrictions:
    - The update can only modify one of the basic tables.
    - The update will modify the basic table that satisfies the key preservation property, i.e., that table such that its primary key could also be the key of the view if its attributes were selected by the SELECT that defines the view.

- The update will not be performed if it violates any of the constraints defined on the basic table to be updated.

### 2.4. Defining privileges

The owner of a database schema is the owner of all the objects defined in it. To allow another user to perform operations on the database objects, they must have the necessary authorization. These authorizations must be granted by the object owner using the GRANT statement.

```
Grant_privilege::= GRANT System_privilege1, System_privilege2,...
                    ON object
                        TO {PUBLIC | user1, user2, ...}
                        [WITH ADMIN OPTION]
```

a) *user* is the identifier of a user.
b) Using the PUBLIC clause, the privileges are assigned to all the users.
c) The WITH ADMIN OPTION clause allows to assign the received privileges to other users.

The privileges that can be granted are administrator privileges, i.e., create, delete or modify schema elements: tables, indexes, views, etc.

## 3. SQL Data Manipulation Language in ORACLE

The data manipulation SQL statements in ORACLE are the same as those of the standard SQL language studied, with the following exceptions related to combining tables with conjunctive operators.

- The SQL operator EXCEPT is called MINUS in ORACLE (with the same syntax).
- Except for UNION ALL, all operators remove duplicates.

## 4. Instructions for querying schema objects in ORACLE

- **Tables defined:** SELECT * FROM {USER_TABLES | ALL_TABLES} ;
- **Constraints that are defined:**
  SELECT * FROM {USER_CONSTRAINTS | ALL_CONSTRAINTS} ;
- **Table columns:** DESCRIBE table ;
- **View compilation errors of triggers:** SHOW ERRORS name ;

## Exercise 1:

We are interested in designing a database for managing a small library. After analyzing the system, the more frequent operations have been identified:

- Obtain the book information: book code, title, author (or authors), theme, and reader that has currently the book (if the book is currently borrowed).
- Obtain the information about the readers: reader ID, name, address, phone number, and the books that the reader currently has, and its loan date.
- Obtain a reader historical loan: book code, loan date, and return date.
- Add, remove, and update the reader's information.
- Manage the loans: Lend a book to a reader and record the return date.

Some integrity constraints have been identified:

- The book code uniquely identifies the book.
- The reader code uniquely identifies the reader.
- The themes to classify books are physics, electricity, mechanics, and optics.
- A book can only be borrowed once a day.
- The return date of a book must be after the loan date.

Using these requirements, we have designed the following relational schema:

```
READER(rcode:char(5), name:varchar2(60),address:varchar2(50),
   phone:varchar2(20))
   CP: {rcode}

BOOK(bcode:char(5), title:varchar(100),theme:varchar(15))
   CP: {bcode}
   RI1: theme es ('physics','electricity','mechanics','optics')

AUTHOR(bcode:char(5), author:varchar2(40) )
   CP:{bcode, author}
   CA:{bcode} -> book(bcode)

LOAN(rcode:char(5),  bcode:char(5),loan_date:date, return_date:date)
   CP:{bcode,loan_date}
   VNN:{rcode}
   CA:{rcode} -> reader(rcode)
   CA:{bcode} -> book(bcode)
RI2: return_date is null or return_date>loan_date
```

a) Define the database in ORACLE using the SQL definition language. All constraints must be deferrable.

b) Ad records to the database.

## 5. Transactions in Oracle

A transaction is a sequence of database manipulation instructions that constitutes a logical execution unit. A transaction must satisfy the properties of atomicity, consistency, isolation, and persistence. The maintenance of each of these properties is the responsibility of a DBMS module, atomicity and persistence are the responsibility of the recovery module, isolation is the responsibility of the concurrency control module, and consistency is the responsibility of the integrity checking module and the recovery module.

The SQL language offers the following statements for handling transactions:

- Start of a transaction: there is no statement to define the beginning of a transaction.
- End of a transaction (with commit): COMMIT [WORK] (partially committed transaction).
- End of a transaction (with rollback): ROLLBACK [WORK] (transaction rolled back).

It is important to remember that the completion (partial commit) of a transaction does not mean its final commit, as this may be conditional on the subsequent checking of integrity constraints. The DBMS can definitively commit or roll back a transaction that the user has finalized with commit. The DBMS can also interrupt, by rolling back, a transaction that has not yet been finalized by the user. To ensure good transaction properties, the DBMS must ensure that changes made by a committed transaction are permanently saved in the database, and that changes made by a rolled back or an aborted transaction that have already been saved are undone.

One of the essential properties of transactions is **consistency**, i.e., the transaction must lead the database to a consistent state in which all integrity constraints are met. Hence, it is essential to know the strategy followed by the system to perform this check.

The existence of the transaction concept and its property of logical unit of execution raises two possible strategies for integrity checking. A constraint can be checked in **immediate mode** (IMMEDIATE) after executing each SQL statement relevant to the constraint or in deferred mode (DEFERRED) after completing any transaction that includes a SQL statement relevant to the constraint. The mode of a constraint can be changed locally for a transaction if the constraint has been defined as DEFERRED. A constraint is DEFERRED if it can be checked at the end of the transaction. The checking mode of a constraint is defined in the database schema with the clause:

```
[[NOT] DEFERRABLE] [INITIALLY {IMMEDIATE | DEFERRED}]
```

The SQL statement that allows to change, locally in a transaction, the mode of a constraint defined as deferrable, is:

```
SET CONSTRAINT { constraint₁, ..., constraintₙ | ALL} {IMMEDIATE | DEFERRED}
```

In the interactive use of the ORACLE system, a transaction starts with the first SQL statement executed by the user since the last transaction ended or since the beginning of the session. A transactions ends when the user explicitly ends it with the COMMIT [WORK] statement (partially committed transaction), explicitly cancels it with the ROLLBACK [WORK] statement (rolled back transaction), when the system implicitly ends it due to the closing of the session (partially committed transaction), or implicitly cancels it due to the occurrence of an error (rolled back transaction). In other words, in ORACLE the operation that initiates a transaction is implicit and the operation that terminates a transaction can be explicit or implicit.

Regarding the constraint checking strategy in ORACLE, the standard SQL proposal is followed with the following behavior regarding the violation of a constraint: "if during the execution of a transaction a constraint is violated with immediate mode, the system undoes only the effect of the SQL operation that caused the constraint violation (statement rollback) and the transaction can continue. If

at the end of a transaction a constraint is violated with *deferred mode* the system cancels the transaction and undoes its global effect (transaction rollback)".

## Exercise 2:

Consider the database that you have created in the previous exercise. and perform the following exercises.

a) In ORACLE there is no cascading update option for restoring referential integrity. How could you modify the code of a reader who has loans, without violating the referential integrity of the historical loan table? (why do we need transactions?).

b) Design a transaction that performs inserts of tuples on the Reader table according to the following transaction schema:

```
COMMIT;
INSERT INTO reader VALUES ('s1', ..., ..., ...);
INSERT INTO reader VALUES ('s2', ..., ..., ...);
INSERT INTO reader VALUES ('s1', ..., ..., ...);
INSERT INTO reader VALUES ('s3', ..., ..., ...);
COMMIT;
```

(where s1, s2 and s3 are reader codes that you have not used up to the time of executing the transaction, and the ellipses represent any values for the attributes name, address and phone).

This transaction violates the primary key constraint of the Reader table.

Run the above transaction twice, once with the primary key constraint for the Reader table in immediate mode. Delete the inserted tuples and run the transaction again with the constraint in defered mode. What differences do you observe in both cases? (atomicity and consistency).

c) Start f two different sessions on the same database server (you can open two instances of SQL Developer) and in each session perform the following transactions (the ti's indicate the order in which the operations should be performed):

| **Session 1** | **Session 2** |
|---|---|
| $t_0$ "Consult the total number of readers" | |
| | $t_1$ "Consult the total number of readers" |
| $t_2$ "Insert a new reader" | |
| $t_3$ "Consult the total number of readers" | |
| | $t_4$ "Consult the total number of readers" |
| $t_5$ "Confirm the transaction" | |
| | $t_6$ "Consult the total number of readers" |
| | $t_7$ "Insert a new reader" |
| | $t_8$ "Consult the total number of readers" |
| | $t_9$ "Cancel the transaction" |
| $t_{10}$ "Consult the total number of readers" | |
| | $t_{11}$ "Consult the total number of readers" |

How are the results of the query operations t3, t4, t6 and t8 interpreted ? How can be the result of the final queries (t10 and t11) interpreted ? (isolation and persistence).

According to the previous results, Do you think that the ORACLE DBMS maintains the persistence property of a transaction?