9-10-2022

# Database Programming with SQL 2 / 2

JUAN CARLOS HERRERA HERNANDEZ
UNIVERSIDAD POLITECNICA DE AGUASCALIENTES

# Contenido

Section 11 – Ensuring Quality Queries Part I ........ 2
11-1 Ensuring Quality Query Results ........ 2
Section 12 – DML ........ 2
12-1 INSERT Statements ........ 2
12-2 Updating Column Values and Deleting Rows ........ 3
12-3 DEFAULT Values, MERGE, and Multi-Table Inserts ........ 4
Section 13 – DDL ........ 5
13-1 Creating Tables ........ 5
13-2 Using Data Types ........ 6
13-3 Modifying a Table ........ 7
Section 14 – Constraints ........ 8
14-1 Intro to Constraints; NOT NULL and UNIQUE Constraints ........ 8
14-2 PRIMARY KEY, FOREIGN KEY, and CHECK Constraints ........ 9
14-3 Managing Constraints ........ 9
Section 15 – Views ........ 10
15-1 Creating Views ........ 10
15-2 DML Operations and Views ........ 10
15-3 Managing Views ........ 10

## Section 11 – Ensuring Quality Queries Part I

### 11-1 Ensuring Quality Query Results

Solve a series of problems:

- Solve a series of problems Create a query to produce specified data
- Modify a query to produce specified data

| | |
|---|---|
| Select * from user_tables | PURGE RECYCLEBIN; |
| select * from tab; | |

## Section 12 – DML

### 12-1 INSERT Statements

| | |
|---|---|
| **USER** | Someone doing "real work" with the computer, using it as a means rather than an end |
| **Transaction** | Consists of a collection of DML statements that form a logical unit of work. |
| **Explicit** | Fully and clearly expressed; leaving nothing implied |
| **INSERT INTO** | Adds a new row to a table |

The table copies will not inherit the associated primary-to-foreign-key integrity rules (relationship constraints) of the original tables.

| Copy structure and data | Copy only structure |
|---|---|
| `CREATE TABLE copy_departments`<br>`as`<br>`SELECT * FROM departments;` | `CREATE TABLE copy_departments`<br>`   as (SELECT * FROM departments`<br>`       where 1=2);` |

| | |
|---|---|
| `Describe employees` | `SALARY NUMBER(6,2)   Precision 6,  Scale 2  [-9999.99, 9999.99]` |
| `user` | `select user from dual;` |
| `sysdate` | `select sysdate from dual;          -- default   DD-Mon-YYYY` |
| | `select to_char(sysdate, 'Month fmdd, yyyy')  from dual;` |

| select columns | all columns |
|---|---|
| `INSERT INTO copy_departments`<br>`   (department_id, department_name, location_id)`<br>`   VALUES (200,'Human Resources', 1500);` | `INSERT INTO copy_departments`<br>` VALUES (210,'Estate Management', 102, 1700);` |

```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, email, hire_date, salary)
  VALUES
  (302,'Grigorz','Polanski', 'GPolanski', TO_DATE('2017-07-20', 'yyyy-mm-dd'), 4200);
```

| Insert multiple records at the same time |
|---|
| `INSERT INTO sales_reps(id, name, salary, commission_pct)`<br>`    SELECT employee_id, last_name, salary, commission_pct`<br>`    FROM employees`<br>`    WHERE job_id LIKE '%REP%';` |

## 12-2 Updating Column Values and Deleting Rows

| | |
|---|---|
| **UPDATE** | Modifies existing rows in a table |
| **Correlated subquery UPDATE** | retrieves information from one table & uses the information to update another table |
| **Integrity Constraint** | Ensures that the data adheres to a predefined set of rules |
| **Correlated subquery DELETE** | deletes information on a linked table based on what was deleted on the other table |
| **Delete** | Removes existing rows from a table |

**Not Correlated**

```
UPDATE copy_employees
SET hire_date = sysdate
WHERE employee_id = 206;
```

**Correlated**

```
UPDATE copy_employees
SET hire_date = sysdate,
    salary = (SELECT salary FROM copy_employees
              WHERE employee_id= 205),
    job_id = (SELECT job_id FROM copy_employees
              WHERE employee_id= 205)
WHERE employee_id = 206;
```

**Correlated**

```
ALTER TABLE copy_employees
ADD (department_name varchar2(30));

select e.department_id, d.department_id, d.department_name
from employees e, departments d
where e.department_id = d.department_id;

UPDATE copy_employees e
SET e.department_name= (SELECT d.department_name
                        FROM departments d
                        WHERE e.department_id= d.department_id);
```

Not Correlated

```
DELETE FROM departments
WHERE department_id = 50;

DELETE FROM copy_employees
WHERE department_id = 50;
```

Correlated

```
DELETE FROM copy_employees
WHERE department_id =
     (SELECT department_id FROM departments
      WHERE department_name= 'Shipping');
```

Be carefully

```
SELECT * FROM copy_employees e
WHERE e.manager_id IN
     (SELECT d.manager_id
      FROM employees d
      GROUP BY d.manager_id
      HAVING count(d.department_id) < 2);
```

```
DELETE FROM copy_employees e
WHERE e.manager_id IN
     (SELECT d.manager_id
      FROM employees d
      GROUP BY d.manager_id
      HAVING count(d.department_id) < 2);
```

row-level locks, until you issue a COMMIT or ROLLBACK

```
SELECT e.employee_id, e.salary, d.department_name
FROM employees e JOIN departments d USING (department_id)
WHERE location_id = 1500 AND job_id= 'ST_CLERK'
FOR UPDATE
ORDER BY e.employee_id;

GRANT update, select ON employees TO schemas

User: SCHEMAS
update ESQUEMAS.employees e set salary = salary
where e.employee_id = 141;
```

## 12-3 DEFAULT Values, MERGE, and Multi-Table Inserts

A **data warehouse** is a collection of data designed to support business-management decision making. Data warehouses contain a wide variety of data, such as sales data, customer data, payroll, accounting, and personnel data, which presents a coherent picture of business conditions at a single point in time.

```
CREATE TABLE my_employees (
hire_date DATE DEFAULT SYSDATE,
first_name VARCHAR2(15),
last_name VARCHAR2(15));
```

```
-- Explicit
INSERT INTO my_employees
(hire_date, first_name, last_name)
VALUES (DEFAULT, 'Angelina','Wright');
```

```
-- Implicit
INSERT INTO my_employees
(first_name, last_name)
VALUES('Angelina','Wright');
```

```
UPDATE my_employees
SET hire_date = DEFAULT
WHERE last_name = 'Wright';
```

```
UPDATE my_employees
SET hire_date = '21-SEP-89'
WHERE last_name = 'Wright';
```

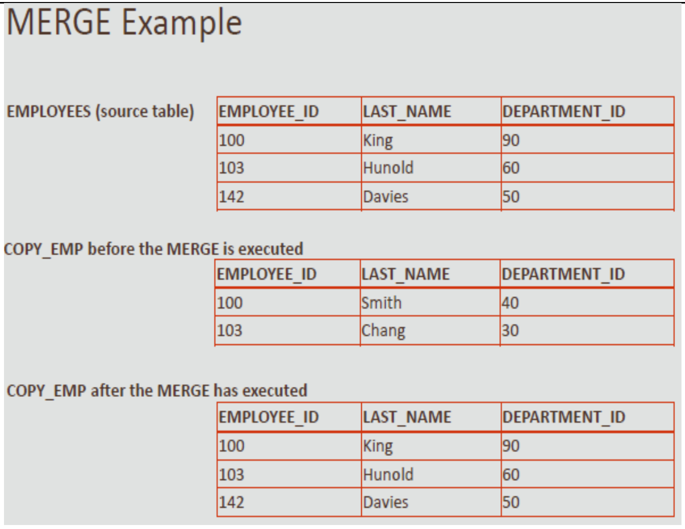
```
UPDATE copy_employees
SET hire_date = to_date('1989-09-21', 'yyyy-mm-dd')
WHERE employee_id = 100;
```

```
MERGE will INSERT and UPDATE
simultaneously.

MERGE INTO destination-table
     USING source-table
ON matching-condition
WHEN MATCHED THEN UPDATE
SET ……
WHEN NOT MATCHED THEN INSERT
VALUES (……);
```

```
MERGE INTO copy_emp c USING employees e
ON (c.employee_id = e.employee_id)
WHEN MATCHED THEN UPDATE
    SET
        c.last_name      = e.last_name,
        c.department_id  = e.department_id
WHEN NOT MATCHED THEN INSERT
    VALUES (e.employee_id, e.last_name, e.department_id);
```

ALL , FIRST



MERGE Example

EMPLOYEES (source table)

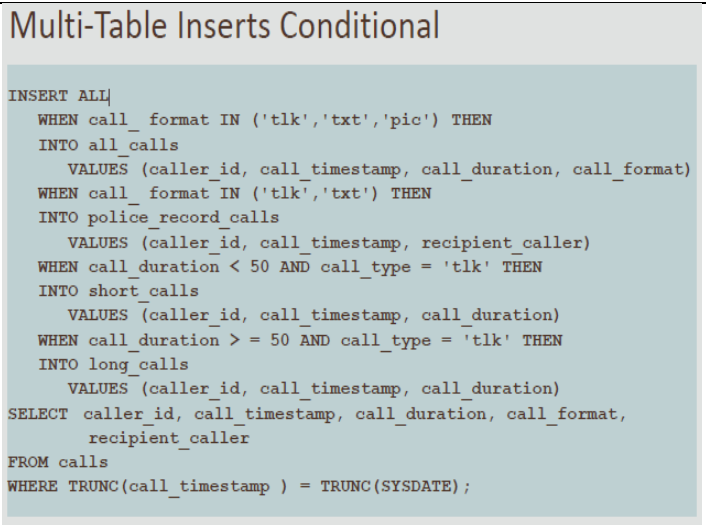
| EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID |
|---|---|---|
| 100 | King | 90 |
| 103 | Hunold | 60 |
| 142 | Davies | 50 |

COPY_EMP before the MERGE is executed

| EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID |
|---|---|---|
| 100 | Smith | 40 |
| 103 | Chang | 30 |

COPY_EMP after the MERGE has executed

| EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID |
|---|---|---|
| 100 | King | 90 |
| 103 | Hunold | 60 |
| 142 | Davies | 50 |



Multi-Table Inserts Conditional

```
INSERT ALL
   WHEN call_ format IN ('tlk','txt','pic') THEN
   INTO all_calls
      VALUES (caller_id, call_timestamp, call_duration, call_format)
   WHEN call_ format IN ('tlk','txt') THEN
   INTO police_record_calls
      VALUES (caller_id, call_timestamp, recipient_caller)
   WHEN call_duration < 50 AND call_type = 'tlk' THEN
   INTO short_calls
      VALUES (caller_id, call_timestamp, call_duration)
   WHEN call_duration > = 50 AND call_type = 'tlk' THEN
   INTO long_calls
      VALUES (caller_id, call_timestamp, call_duration)
SELECT  caller_id, call_timestamp, call_duration, call_format,
        recipient_caller
FROM calls
WHERE TRUNC(call_timestamp ) = TRUNC(SYSDATE);
```

## Section 13 – DDL

### 13-1 Creating Tables

| | |
|---|---|
| **Data dictionary** | Created and maintained by the Oracle Server and contains information about the database |
| **Schema** | A collection of objects that are the logical structures that directly refer to the data in the database |
| **DEFAULT** | Specifies a preset value if a value is omitted in the INSERT statement |
| **Table** | Stores data; basic unit of storage composed of rows and columns |
| **CREATE TABLE** | Command used to make a new table |

Table names are not case sensitive.
Table names should be plural, for example STUDENTS, not student

The main database object types are:

| | | | | | |
|---|---|---|---|---|---|
| Table | Index | Constraint | View | Sequence | Synonym |

```
CREATE TABLE my_cd_collection (
cd_number NUMBER(3),
title VARCHAR2(20) not null,
artist VARCHAR2(20) check(regexp_like(artist,'[a-zA-Z .]')),
purchase DATE DEFAULT SYSDATE);
```

```
-- External Tables Example
CREATE TABLE emp_load (
    employee_number CHAR(5),
    employee_dob CHAR(20),
    employee_last_name CHAR(20),
    employee_first_name CHAR(15),
    employee_middle_name CHAR(15),
    employee_hire_date DATE )
ORGANIZATION EXTERNAL (
    TYPE ORACLE_LOADER
    DEFAULT DIRECTORY def_dir1
    ACCESS PARAMETERS
     (RECORDS DELIMITED BY NEWLINE
       FIELDS (employee_number CHAR(2),
               employee_dob CHAR(20),
               employee_last_name CHAR(18),
               employee_first_name CHAR(11),
               employee_middle_name CHAR(11),
               employee_hire_date CHAR(10) date_format DATE mask "mm/dd/yyyy"))
    LOCATION ('info.dat') );
```

| User tables: | Data Dictionary tables (Only Select): |
|---|---|
| Employees<br>Departments | SELECT * FROM DICTIONARY;<br>SELECT * FROM USER_TABLES;<br>SELECT * FROM USER_INDEXES;<br>SELECT * FROM user_objects WHERE object_type= 'SEQUENCE';<br>SELECT * FROM USER_SEGMENTS;<br>SELECT * FROM ALL_TABLES; |

## 13-2 Using Data Types

| | |
|---|---|
| **BLOB** | Binary large object data up to 4 gigabytes |
| **CLOB** | Character data up to 4 gigabytes |
| **INTERVAL YEAR TO MONTH** | Allows time to be stored as an interval of years and months |
| **INTERVAL DAY   TO SECOND** | Allows time to be stored as an interval of<br>days to hours, minutes, and seconds |
| **TIMESTAMP** | Allows the time to be stored as a date with fractional seconds |
| **TIMESTAMP WITH TIMEZONE** | stores a time zone value as a displacement from<br>Universal Coordinated Time or UCT |
| **TIMESTAMP WITH LOCAL TIMEZONE** | when a column is selected in a SQL statement<br>the time is automatically converted to the user’s  timezone |

| | | |
|---|---|---|
| • CHAR | (fixed size, | maximum 2000 characters) |
| • VARCHAR2 | (variable size, | maximum 4000 characters) |
| • NUMBER | (variable size, | maximum precision 38 digits) |
| • DATE | | range yyyy-mm-dd hh24:mi:ss |
| • TIMESTAMP | | range yyyy-mm-dd hh12:mi:ss and fractions of a second |
| • INTERVAL DAY [(day_precision)] TO SECOND | | The default precisión value is 2 |

```
select current_timestamp, SYSTIMESTAMP from dual
```

| | |
|---|---|
| current_timestamp | 03-OCT-22 05.22.33.598000000 PM AMERICA/MEXICO_CITY |
| SYSTIMESTAMP        UCT | 03-OCT-22 05.22.33.598000000 PM -05:00 |

| MySQL      Date   yyyy-mm-dd | ORACLE       Date yyyy-mm-dd hh:mi:ss |
|---|---|
| create table tmp_Formatos(<br>Fecha date,<br>FechaTiempo datetime,<br>TiempoMarca timestamp); | create table tmp_Formatos(<br>Fecha date,<br>TiempoMarca timestamp); |
| select now(), sysdate(), current_timestamp(); | select sysdate, current_date,<br>       current_timestamp, SYSTIMESTAMP<br>from dual; |
| insert into tmp_Formatos<br>   values(sysdate(), sysdate(), sysdate()); | insert into tmp_Formatos<br>   values(sysdate, sysdate); |
| Select * from tmp_formatos; | select * from tmp_formatos; |
| select  second(fechaTiempo),<br>        extract(second from TiempoMarca)<br>from tmp_formatos; | select to_char(fecha, 'ss'),<br>       extract(second from TiempoMarca)<br>from tmp_formatos; |

| | |
|---|---|
| create table tmp_Horarios (<br>Fecha date,<br>TS TIMESTAMP,<br>TS_TZ TIMESTAMP WITH TIME ZONE,<br>TS_LTZ TIMESTAMP WITH LOCAL TIME ZONE); | create table tmp_Intervalos (<br>loan1 INTERVAL YEAR TO MONTH,<br>loan2 INTERVAL YEAR TO MONTH); |
| insert into tmp_horarios values<br>(sysdate, sysdate, SYSTIMESTAMP, sysdate); | INSERT INTO tmp_Intervalos (loan1, loan2)<br>VALUES (INTERVAL '121' MONTH(3),<br>        INTERVAL '3-6' YEAR TO MONTH); |
| | select sysdate+loan1 from tmp_intervalos; |

## 13-3 Modifying a Table

You can add or modify a column in a table, but you cannot specify where the column appears

```
ALTER TABLE tablename
ADD (column_name data_type [DEFAULT expression],
     column_name data_type [DEFAULT expression], ...);
```

```
ALTER TABLE mod_emp
MODIFY (salary NUMBER(8,2) DEFAULT 50);
```

```
ALTER TABLE tablename
DROP COLUMN columnname;
```

```
-- Dropping a column from a large table can take a long time
ALTER TABLE tablename SET UNUSED (column_name);
```

```
-- when you want to reclaim the extra disk space
ALTER TABLE copy_employees
   DROP UNUSED COLUMNS;
```

```
ALTER SESSION SET RECYCLEBIN = ON;
DROP TABLE table_name;
```

| | |
|---|---|
| `-- Recovery a Table`<br>`FLASHBACK TABLE table_name TO BEFORE DROP;` | `-- Show deleted tables`<br>`select * from USER_RECYCLEBIN;` |
| `-- Drop a table definitely`<br>`DROP TABLE Table_Name PURGE;` | `-- Rename a table`<br>`RENAME old_name to new_name;` |

| it does not release storage space | Free up storage space |
|---|---|
| Delete from Table_Name; | Truncate Table Table_Name; |

```
COMMENT ON TABLE Employees is 'Tabla de empleados';
comment on column Employees.last_name is 'Apellido Paterno';
```

```
select * from user_tab_comments;
SELECT * FROM USER_COL_COMMENTS;
```

| | Review the changes made (UNDO tablespace)<br>SCN (System Change Number) |
|---|---|
| UPDATE EMPLOYEES<br>SET LAST_NAME = 'King Kong'<br>where employee_id = 100; | select * from Employees<br>VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE<br>WHERE employee_id= 100; |

## Section 14 – Constraints

### 14-1 Intro to Constraints; NOT NULL and UNIQUE Constraints

| **Constraint** | Database rule. |
|---|---|
| **PRIMARY KEY** | Constraint ensures that the column contains no null values and uniquely identifies each row of the table |
| **UNIQUE KEY** | An integrity constraint that requires every value in a column or set of columns be unique |
| **UNIQUE constraint** | Every value in a column or set of columns (a composite key) must be unique |
| **FOREIGN KEY** | Designates a column (child table) that establishes a relationship between a primary key in the same table and a different table (parent table) |
| **REFERENCES** | Identifies that table and column in the parent table |
| **NOT NULL constraint** | For every row entered into the table, there must be a value for that column |
| **CHECK constraint** | Specifies a condition for a column that must be true for each row of data |
| **Table level constraint** | References one or more columns and is defined separately from the definitions of the columns in the table |
| **Column-level constraint** | Database rule that references a single column |

### 5 Types of constraints: All the constraints have a name

| | | | | |
|---|---|---|---|---|
| NOT NULL | PRIMARY KEY | FOREIGN KEY | UNIQUE | CHECK |

There are two different places in the CREATE TABLE statement that you can specify the constraint details:

- At the **column level** next to the name and data type
- At the **table level** after all the column names are listed

| Constraints at the Column Level | Constraints at the Table Level |
|---|---|
| `CREATE TABLE clients (`<br>`IDnumber NUMBER(4) primary KEY,`<br>`LastName VARCHAR2(20) constraint nn_LN not null,`<br>`Email VARCHAR2(20) UNIQUE,`<br>`HireDate date default sysdate,`<br>`Salary number(6,2) check(salary > 0)`<br>`);`<br><br>`system gives the constraint a name,`<br>`such as` **`SYS_C00585417`**<br><br>`The NOT NULL constraint can be specified only at the column level, not the table level` | `CREATE TABLE clients (`<br>`IDNumber NUMBER(4),`<br>`LastName VARCHAR2(20),`<br>`Email VARCHAR2(20),`<br>`HireDate date default sysdate,`<br>`Salary number(6,2),`<br>`CONSTRAINT` **`Clients_IDNumber_pk`** `primary key (IDNumber),`<br>`CONSTRAINT` **`uk`**`_Email UNIQUE(Email),--unique(email,phone)`<br>`CONSTRAINT check_Salary check(Salary>0)`<br>`);`<br><br>`ALTER TABLE clients MODIFY (LastName NOT NULL);` |

## 14-2 PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

| | |
|---|---|
| **PRIMARY KEY constraint** | A column or set of columns that uniquely identifies each row in a table |
| **FOREIGN KEY constraint** | Establishes a relationship between the foreign key column and a primary key or unique key in the same table or a different table |
| **NOT NULL** | Constraint ensures that the column contains no null values |
| **CHECK constraint** | Explicitly defines a condition that must be met |
| **ON DELETE SET NULL** | Allows a child row to remain in a table with null values when a parent record has been deleted |
| **ON DELETE CASCADE** | Allows a foreign key row that is reference to a primary key row to be deleted |

Column-level syntax example:

```
CREATE TABLE employees(
employee_id NUMBER(6,0) CONSTRAINT emp_pk PRIMARY KEY,
first_name VARCHAR2(20),
last_name VARCHAR2(25),
department_id NUMBER(4,0) CONSTRAINT emp_department_id_fk
    REFERENCES departments(department_id) ON DELETE SET NULL,
email VARCHAR2(25));
```

Table-level syntax example:

```
CREATE TABLE employees(
employee_id NUMBER(6,0) CONSTRAINT emp_pk PRIMARY KEY,
first_name VARCHAR2(20),
last_name VARCHAR2(25),
department_id NUMBER(4,0),
email VARCHAR2(25),
CONSTRAINT emp_dept_id_fk FOREIGN KEY (department_id)
                REFERENCES departments(department_id) ON DELETE CASCADE);
```

Composite primary key

```
CREATE TABLE job_history (
employee_id   NUMBER(6,0),
start_date    DATE,
end_date      DATE,
job_id        VARCHAR2(10),
department_id NUMBER(4,0),
CONSTRAINT jh_pk PRIMARY KEY(employee_id, start_date),
CONSTRAINT jh_endDate_ck CHECK (end_date> start_date));
```

```
CHECK constraint cannot contain
functions:
SYSDATE, UID, USER, or USERENV
Example: SYSDATE >'05-May-1999'

CHECK constraint cannot use:
CURRVAL, NEXTVAL, LEVEL, or ROWNUM
```

## 14-3 Managing Constraints

| | |
|---|---|
| **DISABLE CONSTRAINT** | To deactivate an integrity constraint |
| **CASCADE clause** | Disables dependent integrity constraints |
| **ALTER TABLE** | To add, modify, or drop columns from a table |
| **ENABLE CONSTRAINT** | To activate an integrity constraint currently disabled |
| **DROP CONSTRAINT** | Removes a constraint from a table |
| **DROP COLUMN** | Allows user to delete a column from a table |

|  |  |
|---|---|
| **CASCADE CONSTRAINT clause** | Defines the actions the database server takes when a user attempts to delete or update a key to which existing foreign keys point |

## Section 15 – Views

### 15-1 Creating Views

### 15-2 DML Operations and Views

### 15-3 Managing Views