I.E.S Rodrigo Caro

Departamento de Informática
www.informaticarodrigocaro.es
Informática Rodrigo Caro

Unit 7

SQL Introduction

ASIR

Introduction

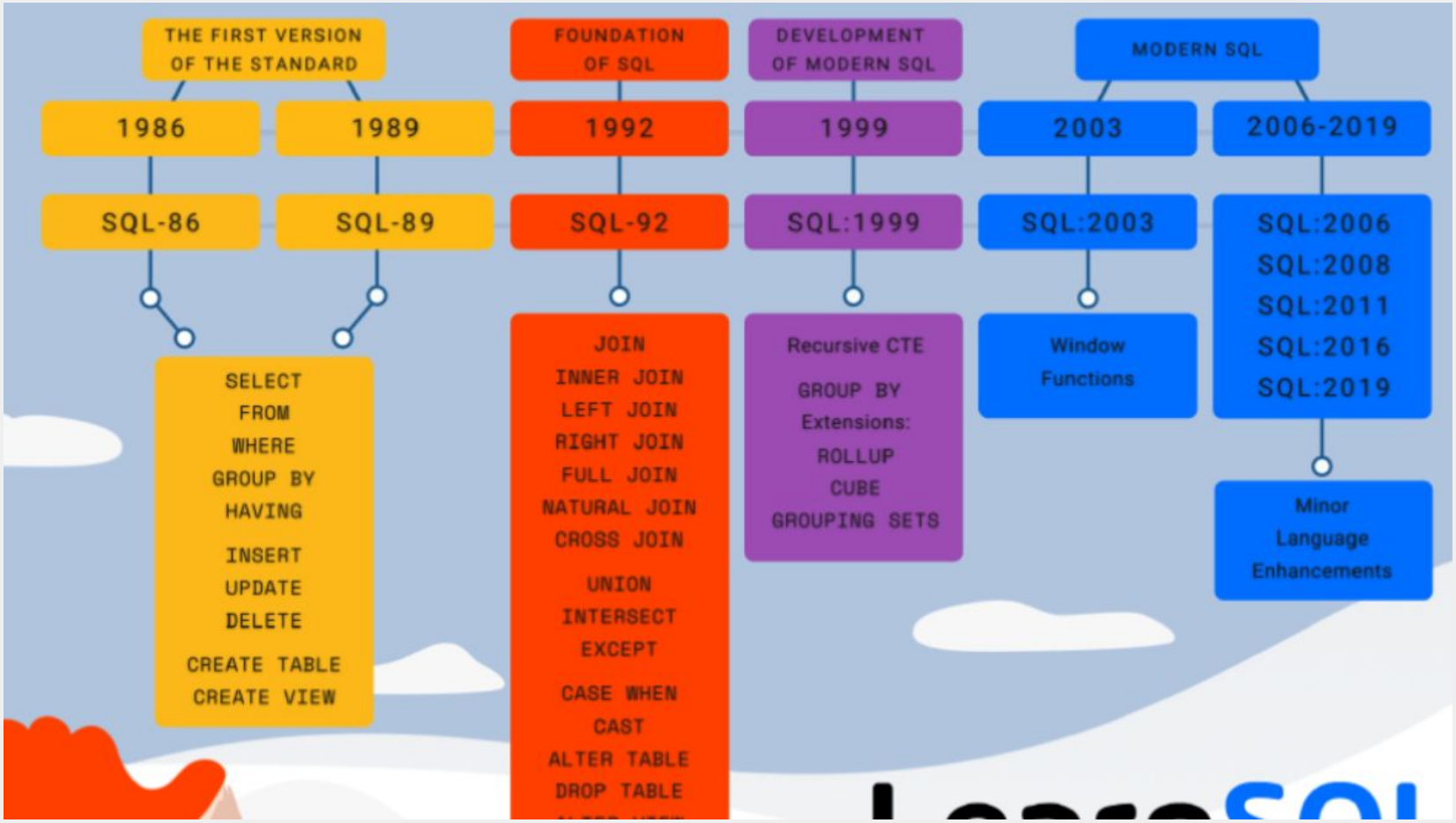
SQL (Structured Query Language) is a tool to organize, manage and retrieve a relational database;

Among the main features of

SQL we can highlight the following:

- It is a language for all types of DB users.
- It allows any data query to be made.
- It is possible to perform updates, data definitions and transaction control.
- It can be used interactively and in an embedded way.

Some History



SQL Name Convention

In a SQL-based database, objects such as a table or a view, are identified by unique names.

These names will be used in SQL statements to identify the database object on which the statement will act.

Rules for naming objects:

A name consists of a set of alphanumeric characters.

(letters and numbers) including underscore hyphen characters, '_' .

- It can begin with any of the above characters, but may not consist of numbers only.
- Names of databases, tables, and columns should not end with space characters.
- If an identifier is a reserved word or contains special characters, it should be enclosed in quotation marks whenever it is reference is made to it.

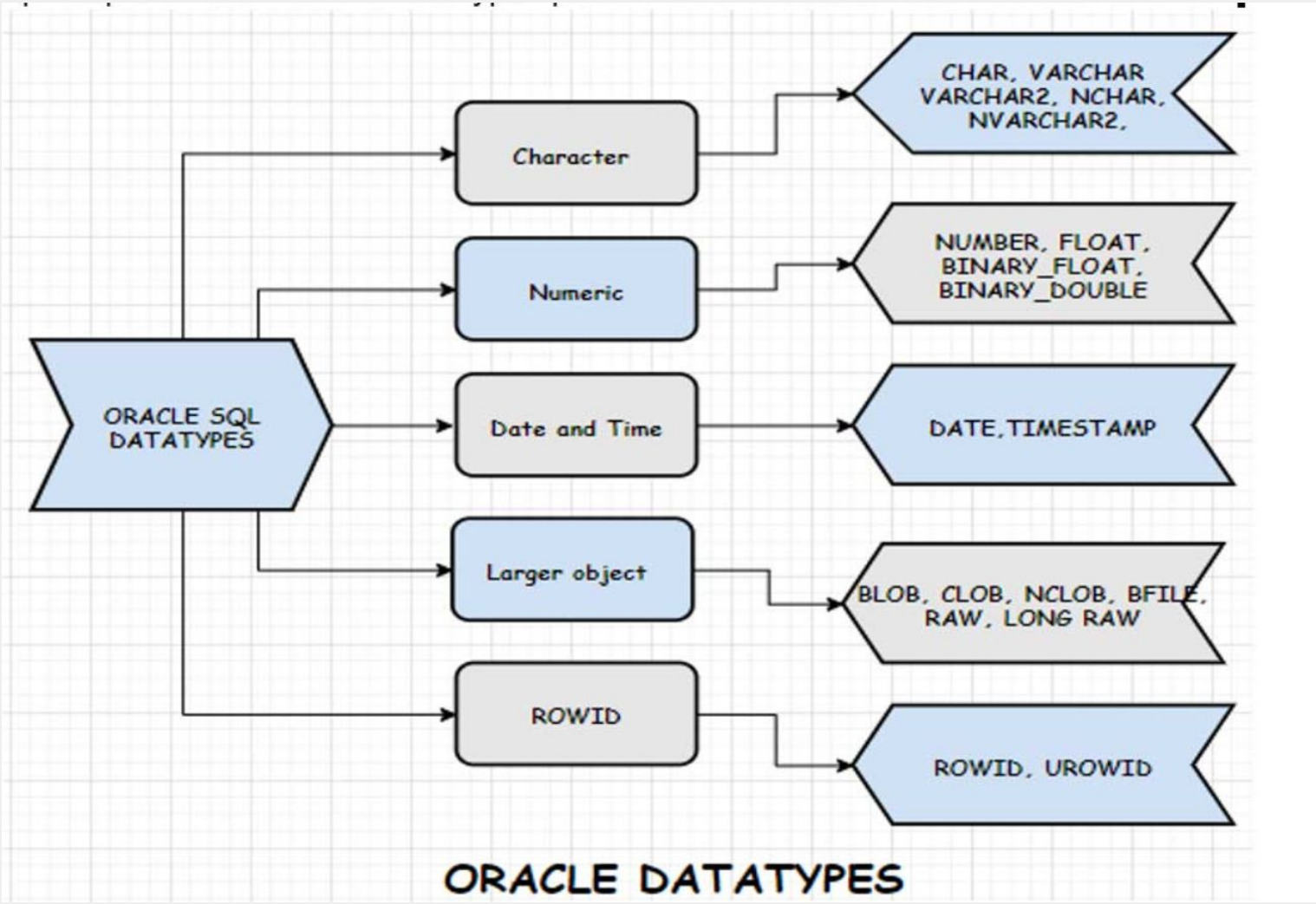
SQL Comparison Operators

Operator	Description
=	Equal to.
>	Greater than.
<	Less than.
>=	Greater than equal to.
<=	Less than equal to.
<>	Not equal to.
IS NULL	Is null
LIKE	Comparing with a pattern
BETWEEN	Between two values
IN	Belong to a a set
EXISTS	Exists

SQL Operator Priority

()
*, /
-, +, | |
=, !=, <, >, <=, >=
IS NULL, LIKE, BETWEEN, IN, EXITS
NOT
AND
OR

SQL Datatypes



SQL Character DataTypes

Data Type Syntax	Scale	Explanation
char(size)	Maximum size of 2000 bytes.	Where size is the number of characters to store. Fixed-length strings. Space padded.
nchar(size)	Maximum size o 2000 bytes.	Where size is the number of characters to store. Fixed-length NLS string Space padded.
nvarchar2(size)	Maximum size of 4000 bytes.	Where size is the number of characters to store. Variable-length NLS string.
varchar2(size)	Maximumsize of 4000 bytes. Maximum size of 32KB in PLSQL.	Where size is the number of characters to store. Variable-length string.
long	Maximum size of 2GB.	Variable-length strings. (backward compatible)
raw	Maximum size of 2000 bytes.	Variable-length binary strings
long raw	Maximum size of 2GB.	Variable-length binary strings. (backward compatible)

SQL Numeric Datatypes

Data Type Syntax	Scale	Explanation
number(p,s)	Precision can range from 1 to 38. Scale can range from -84 to 127.	Where p is the precision and s is the scale. For example, number(7,2) is a number that has 5 digits before the decimal and 2 digits after the decimal.
numeric(p,s)	Precision can range from 1 to 38.	Where p is the precision and s is the scale. For example, numeric(7,2) is a number that has 5 digits before the decimal and 2 digits after the decimal.
float		
dec(p,s)	Precision can range from 1 to 38.	Where p is the precision and s is the scale. For example, dec(3,1) is a number that has 2 digits before the decimal and 1 digit after the decimal.
decimal(p,s)	Precision can range from 1 to 38.	Where p is the precision and s is the scale. For example, decimal(3,1) is a number that has 2 digits before the decimal and 1 digit after the decimal.
integer		
int		
smallint		
real		
double precision		

SQL Date/Time Datatypes

Data Type Syntax	Scale	Explanation
date	A date between Jan 1, 4712 BC and Dec 31, 9999 AD.	3 bytes
timestamp (fractional seconds precision)	fractional seconds precision must be a number between 0 and 9. (default is 6)	4 bytes Includes year, month, day, hour, minute, and seconds. For example: timestamp(6)

SQL Larger Datatypes

Data Type Syntax	Scale	Explanation
bfile	Maximum file size of 264-1 bytes.	File locators that point to a **binary file** on the server file system (stored **outside the database**).
blob	Store up to (4 gigabytes -1) * (the value of the CHUNK parameter of LOB storage).	Stores unstructured **binary large objects:** images or sounds
clob	Store up to (4 gigabytes -1) * (the value of the CHUNK parameter of LOB storage) of character data.	Stores really **long characters strings** Example html for a web
nclob	Store up to (4 gigabytes -1) * (the value of the CHUNK parameter of LOB storage) of character text data.	Stores Unicode l**ong chararcter strings** Example html for a web

SQL SUMMARY

Relational Model	Oracle - SQL
character	VARCHAR2(length) CHAR(length)
integer	NUMBER(precision)
float	NUMBER(precision, scale)
date	DATE
time	DATE
boolean	CHAR(1) ('T' , 'F') or NUMBER(1) (0,1)
memo	CLOB
object	BLOB

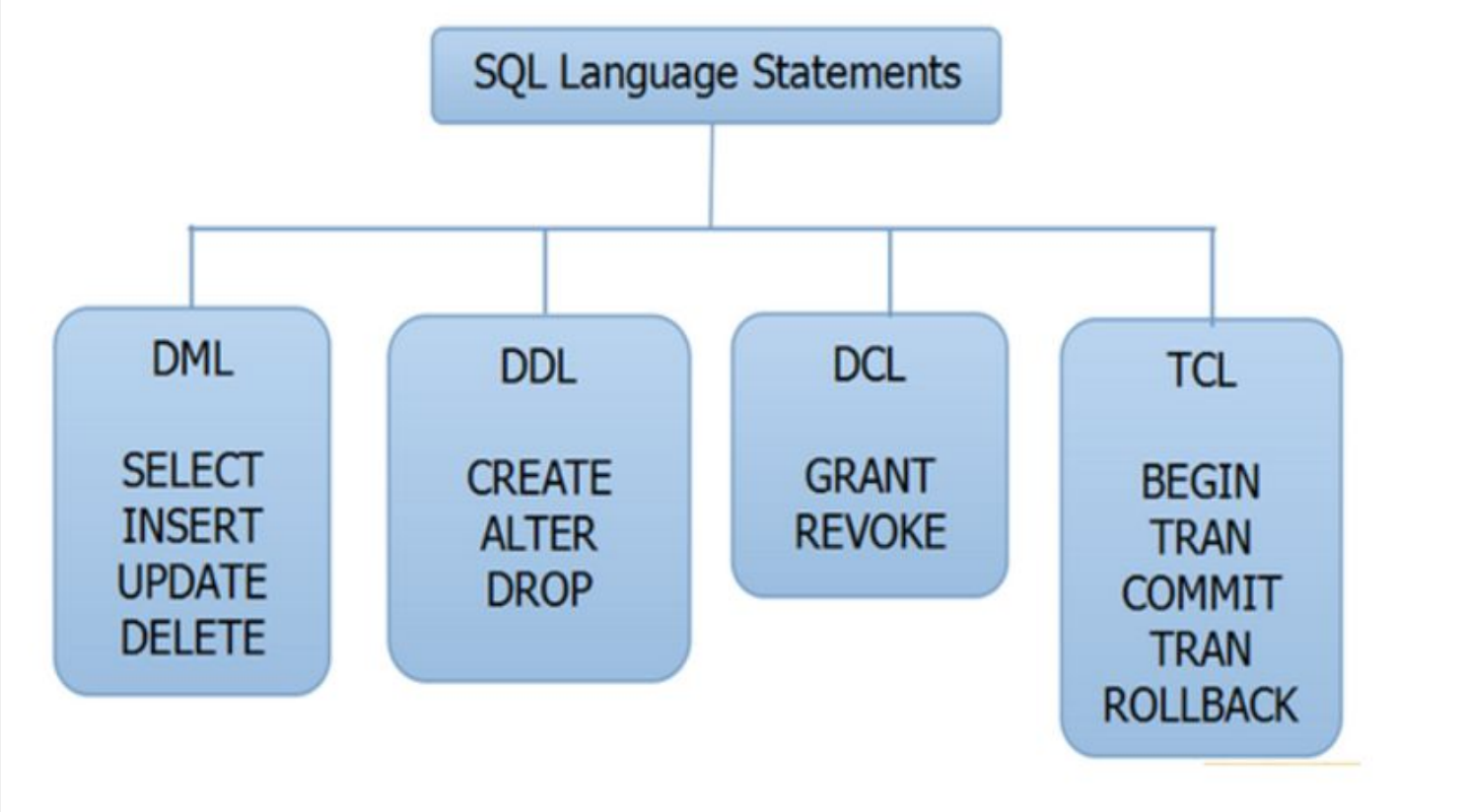
Comparing Sql Server and Oracle Datatypes

SQL Server	Oracle
Exact Numerics	
TINYINT	NUMBER(3)
SMALLINT	NUMBER(5)
INTEGER	NUMBER(10)
BIGINT	NUMBER(19)
DECIMAL(p,s)	NUMBER(p,s)
NUMERIC(p,s)	NUMBER(p,s)
SMALLMONEY	NUMBER(10,4)
MONEY	NUMBER(19,4)
Approximate Numerics	
REAL	BINARY_FLOAT
FLOAT	BINARY_DOUBLE

Comparing Sql Server and Oracle Datatypes

SQL Server	Oracle
Character strings	
CHAR(x)	CHAR(x)
VARCHAR(x)	ARCHAR2(x)
VARCHAR(MAX)	CLOB
TEXT	LONG
Binary strings	
BINARY(n)	RAW(n)
VARBINARY(n)	LONG RAW
VARBINARY(MAX)	LONG RAW or BLOB
IMAGE	LONG RAW
Binary strings	
XML	XMLTYPE
BIT	NUMBER(1)
TIMESTAMP	ORA_ROWSCN pseudo column
UNIQUEIDENTIFIER	RAW(16)
N/A	BFILE

SQL Sentences Types



SQL Sentences Types

SQL statements are categorized into four different types of statements, which are

- DDL (DATA DEFINITION LANGUAGE)
- DML (DATA MANIPULATION LANGUAGE)
- DCL (DATA CONTROL LANGUAGE)
- TCL (TRANSACTION CONTROL LANGUAGE)

SQL DDL: Create Table

This statement is used to create a new table in a database. In that table, if you want to add multiple columns, use the below syntax.

```
CREATE TABLE table_name
(
  column1 datatype null/not null,
  column2 datatype null/not null,
  ...
  CONSTRAINT constraint_name PRIMARY KEY (column1, column2, ... column_n)
);
```

Uppercase and lowercase letters are irrelevant when creating a table.

SQL DDL: Create Table

Before defining a table, it is very convenient to define :

- **Table' s name**. It must be a name that identifies its content.

  E.g. STUDENTS will contain data about students.

- **Columns' name**. It must be a descriptive name that identifies its content. E.g. ID, first name, last name…
- The **type of data and the size** that each column will have.
- The **restrictions** that each column will have: mandatory, default values, primary key, etc.

SQL DDL Create Table

- **Constraints Restrictions :**

  To define the constraints we use the CONSTRAINT CLAUSE.

  A CONSTRAINT can restrict a single column (column constraint) or a group of columns (table constraint).

  Thanks to the constraints we define:

  - primary keys
  - alternate keys
  - foreign keys
  - checks

There are also columns restrictions:

- null or not null
- default values

SQL DDL Create Table

- **Constraints Restrictions :**
  - **primary Keys**

    None of the fields that are part of the primary key can contain a null value.

```
CREATE TABLE supplier
(
  supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);
```

```
CREATE TABLE supplier
(
  supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id, supplier_name)
);
```

SQL DDL Create Table

- **Constraints Restrictions :**
  - **alternate keys**

```
[[CONSTRAINT constraint_name] UNIQUE (column), ]
```

```
CREATE TABLE supplier
( supplier_id numeric(10) NOT NULL,
  supplier_name varchar2(50) NOT NULL,
  contact_name varchar2(50),
  CONSTRAINT supplier_unique UNIQUE (supplier_id,
supplier_name)
 );
```

SQL DDL Create Table

- **Constraints Restrictions :**
  - **foreign keys**

```
CREATE TABLE table_name
(
  column1 datatype null/not null,
  column2 datatype null/not null,
  ...
  CONSTRAINT fk_column
    FOREIGN KEY (column1, column2, ... column_n)
    REFERENCES parent_table (column1, column2, ... column_n)
);
```

SQL DDL Create Table

- **foreign keys**

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);
CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  CONSTRAINT fk_supplier
    FOREIGN KEY (supplier_id)
    REFERENCES supplier(supplier_id)
);
```

SQL DDL Create Table

- **Constraints Restrictions :**
  - check. This constraint defines a condition that each row must satisfy.

```
CONSTRAINT constraint_name CHECK (column_name condition) [DISABLE]
```

```
CREATE TABLE suppliers
(  supplier_id numeric(4),
  supplier_name varchar2(50),
  CONSTRAINT check_supplierID
    CHECK (supplier_id BETWEEN 100 and 9999),
 CONSTRAINT check_supplierName
    CHECK (supplier_name=upper(supplier_name))
);
```

SQL DDL Create Table

- **Columns Restrictions :**
  - **default values** A column can be given a default value using the DEFAULT keyword. The default value can be literal value, an expression, or a SQL Function, such as SYSDATE.

```
CREATE TABLE demo_tbl
(
salary number(8,2) DEFAULT 9500,
hire_date DATE DEFAULT '01-JAN-2011' ,
birthdate DATE DEFAULT SYSDATE
);
```

SQL DDL Create Table

- **Columns Restrictions :**
  - **default values** A column can be given a default value using the DEFAULT keyword. The default value can be literal value, an expression, or a SQL Function, such as SYSDATE.

```
CREATE TABLE demo_tbl
(
salary number(8,2) DEFAULT 9500,
hire_date DATE DEFAULT '01-JAN-2011' ,
birthdate DATE DEFAULT SYSDATE
);
```

- **null or not null columns** NOT NULL indicates that the column must contain some information.

SQL DDL Drop table

Drop table statement deletes one or more DB tables of the database. Deleting a table also deletes the indexes and privileges associated with it are also deleted.

If there are associated views using this table, they will stop working but they won't be dropped.

```
DROP TABLE table_name

[ CASCADE CONSTRAINTS ]

[ PURGE ];
```

Important! You cannot delete a table that is referenced by foreign keys. We should delete the foreign key constraint (alter table) before dropping the table.

SQL DDL Drop table

- **CASCADE CONSTRAINTS** option will remove all referential integrity constraints which refer to primary and unique keys in the table. In case such referential integrity constraints exist and you don’t use this clause, Oracle returns an error.
- **PURGE** option will purge the table and its dependent objects so that they do not appear in the recycle bin. The risk of specifying the PURGE option is that you will not be able to recover the table. However, the benefit of using PURGE is that you can ensure that sensitive data will not be left sitting in the recycle bin.

SQL DDL Alter Table

The ALTER TABLE statement allows user to **modify an existing table** such as to **add, modify or drop** a column from an existing table.

```
ALTER TABLE table_name
  ADD/DROP/MODIFY/RENAME
column_name/constraint_name definition;
```

SQL DDL Alter Table

ADD

- to add a **column**

```
ALTER TABLE customers
    ADD customer_name varchar2(45) DEFAULT 'pending';
```

- to add **multiple columns**

```
ALTER TABLE customers
    ADD (
            customer name varchar2(45),
            city varchar2(40) DEFAULT 'Seattle'
        );
```

- to add a **constraint**

```
ALTER TABLE customers ADD CONSTRAINT validateCountry
   CHECK (
       BRANCH IN('PORTUGAL','SPAIN', 'FRANCE', 'ITALY')
          );
```

SQL DDL Alter Table

DROP:

- to drop a **column**

```
ALTER TABLE customers
    DROP customer_name;
```

- to drop **multiple columns**

```
ALTER TABLE customers
    DROP COLUMN
                date_of_birth,
                customer_name;
```

- to drop a **constraint**

```
ALTER TABLE table_name
    DROP CONSTRAINT constraint_name;
```

SQL DDL Alter Table

For our examples we are going to work with a table like this:

```
CREATE TABLE accounts (
    account_id NUMBER,
    first_name VARCHAR2(25),
    last_name VARCHAR2(25) NOT NULL,
    email VARCHAR2(100),
    phone VARCHAR2(12) ,
    status NUMBER( 1, 0 ) DEFAULT 1 NOT NULL,
    CONSTRAINT pkAccounts PRIMARY KEY(account_id)
);
```

SQL DDL Alter Table

- Modify to **allow or not allow null values**

```
ALTER TABLE accounts
    MODIFY email VARCHAR2( 100 ) NOT NULL;
```

This statement will not allow null values. In case there is any null value for that column, it will return a error.

- Change the **default value** of a column

```
ALTER TABLE accounts
    MODIFY status DEFAULT 0;
```

The status column was defined with the value 1 by default and now it is 0. Doing a insert of a row without a defining this column, now it is going to be set to 1.

SQL DDL Alter Table

- Widen or shorten the **size of a column**

```
ALTER TABLE accounts

    MODIFY phone VARCHAR2( 15 );
```

We are widening the column phone. It was a length of 12 and now it is 15.

To shorten the size of a column, you make sure that all data in the column fits the new size. If not, you will get an error after executing the statement.

SQL DDL Alter Table

MODIFY to modify a **column.**

- Modify the **column's visibility**. By default, table columns are visible. To make invisible a column could be because of legal or security reasons.

  You can define invisible column when you create the table or using ALTER TABLE MODIFY column statement.

```
ALTER TABLE customers

MODIFY customer_name INVISIBLE;
```

  Once a table's column is set to invisible, the following key statements will not work for the invisible column, for example doing `SELECT * FROM tableName`

SQL DDL Alter Table

- to modify multiple columns

```
ALTER TABLE customers
    MODIFY (
            customer name varchar2(100) NOT NULL,
             city varchar2(75) DEFAULT 'Seattle' NOT NULL
            );
```

SQL DDL Alter Table

MODIFY to modify a **column.**

SQL DDL Alter Table

RENAME:

- to rename **a table**

```
ALTER TABLE customers
        RENAME TO contacts;
```

- to rename a **column**

```
ALTER TABLE table_name
    RENAME COLUMN customer_name TO cname;
```

SQL DDL: Select Statement

The SELECT statement is used to select records from the table, with or without a condition.

Note that the SELECT statement is very complex that consists of many clauses such as **ORDER BY, GROUP BY, HAVING**, **JOIN.**

We can select:

- a single column
- several columns
- all the columns
- from one or more tables **(JOIN)**

SQL DDL: Select Statement

SELECT [column_name | * | expression]

FROM <table_name>

WHERE <expression>[comparison operator]<expression>;

Name	Description
column_name	Name of the column of a table.
*	Indicates all the columns of a table.
expression	Expression made up of a single constant, variable, scalar function, or column name and can also be the pieces of a SQL query that compare values against other values or perform arithmetic calculations.
table_name	Name of the table.
comparison operator	Equal to (=), not equal to(<>), greater than(>), less than(<), greater than or equal to (>=), less than or equal to (<=).

SQL DDL: Select Statement

For our example we are going to work with this table:

- Query data from a **single column**

```
SELECT   name

    FROM   customers;
```

SQL DDL: Select Statement

- Query data from **multiple columns**

```
SELECT    customer_id,    name,    credit_limit

    FROM    customers;
```

- Query data from all columns

```
SELECT    *

    FROM    customers;
```

SQL DDL: Select Statement

SELECT DISTINCT.

The DISTINCT clause is used in a SELECT statement to **filter duplicate rows** in the result set. It ensures that rows returned are **unique for the column or columns specified** in the SELECT clause.

	FIRST_NAME
1	Aaron
2	Adah
3	Adam
4	Adrienne
5	Agustina
6	Al
7	Aleshia
8	Alessandra
9	Alexandra

```
SELECT DISTINCT   first_name

  FROM   contacts

ORDER BY   first_name;
```

Result:

	FIRST_NAME
1	Aaron
2	Adah
3	Adam
4	Adrienne
5	Agustina
6	Al
7	Aleshia

SQL DDL: INSERT

Insert a **new row** into a table. Syntax:

```
INSERT INTO table_name
    VALUES (value_list);
```

Example:

```
INSERT INTO accounts(first_name,last_name,phone)
VALUES('Trinity',  'Knox',  '410-555-0197');
```

SQL DDL: INSERT ALL

Insert a **multiple rows** in a table or multiple tables. Syntax:

- Insert **multiple rows into a table**

```
INSERT ALL
    INTO table_name(col1,col2,col3) VALUES(val1,val2, val3)
    INTO table_name(col1,col2,col3) VALUES(val4,val5, val6)
    INTO table_name(col1,col2,col3) VALUES(val7,val8, val9)
Subquery;
```

SQL DDL: INSERT ALL

For our examples, we are going to work with this table:

```
CREATE TABLE fruits (
    fruit_name VARCHAR2(100) PRIMARY KEY,
    color VARCHAR2(100) NOT NULL
);
```

Example:

```
INSERT ALL
    INTO fruits(fruit_name, color)     VALUES ('Apple','Red')
    INTO fruits(fruit_name, color)     VALUES ('Banana','Yellow')
SELECT * FROM dual;
```

SQL DDL: DUAL TABLE

DUAL is a table automatically created by Oracle Database along with the data dictionary.

DUAL is in the schema of the user SYS but is accessible by the name DUAL to all users.

It has one column, DUMMY, defined to be VARCHAR2(1), and contains one row with a value X.

```
SELECT * FROM DUAL ;
```

Return x

```
CREATE TABLE Builder
(
      builderId number NOT NULL,
      builderName varhar2(15),
      country varchar2(20) DEFAULT 'SPAIN',
      CONSTRAINT clavePrimaria     PRIMARY KEY (builderId),
      CONSTRAINT mayuscNombre CHECK (builderName=upper(builderName)),
      CONSTRAINT mayuscPais      CHECK (country=upper(country)),
      CONSTRAINT compruebaID    CHECK  ( builderId BETWEEN 1 AND 255 )
);
```

References:

- https://www.oracletutorial.com/oracle-basics/
- https://www.c-sharpcorner.com/blogs/types-of-sql-statements-with-example

```
CREATE TABLE Product
(
    nameProduct varchar2(20),
    idProduct number NOT NULL,
    weight number,
    categoryP varchar2(10) NOT NULL,
    salePrice Number(6,2),
    costPrice Number(6,2),
    stockNumber Number,
    idBuilder Number,
    CONSTRAINT pkProductC  PRIMARY KEY (idProduct, categoryP),
    CONSTRAINT fkProductC  FOREIGN KEY (idBuilder) REFERENCES Builder(builderId),
    CONSTRAINT positiveWeightC  CHECK (weight >= 0),
    CONSTRAINT positiveSalePriceC CHECK (salePrice >= 0),
    CONSTRAINT positiveCostPriceC  CHECK (costPrice >= 0),
    CONSTRAINT validateCategoryC  CHECK (categoryP in  ('FIRST', 'SECOND', 'THIRD')),
    CONSTRAINT validateIDProductC   CHECK  ( idProduct BETWEEN 1 AND 255 )
);
```