Curso de Base de Datos
Sesión 4: SQL

2022-2

Ingeniería Informática

Semana 4
SQL DDL

Profesor del curso:
César Aguilera
Luis Ríos

Elaborado por:
César Aguilera
Luis Ríos

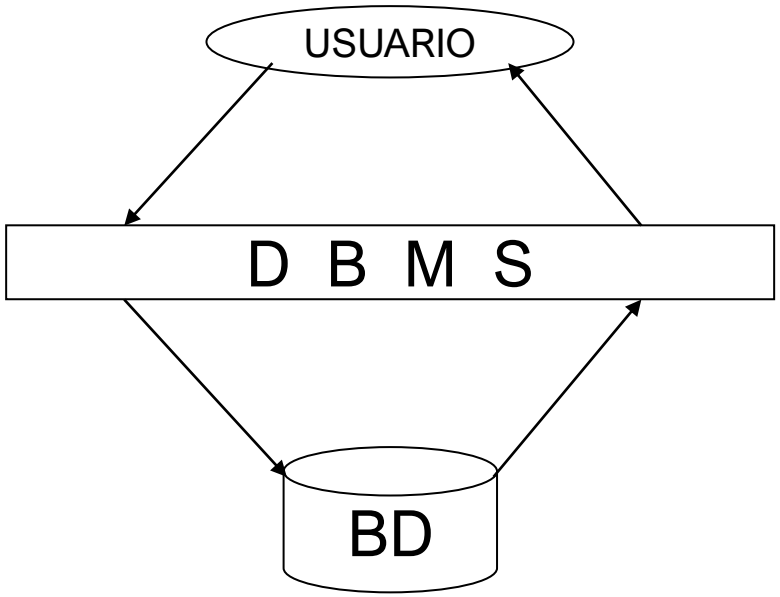
Revisado por:
Rony Cueva
César Aguilera

Saberes previos

- El Modelo Relacional:
  - Estructura y operaciones relacionales
  - Reglas de integridad
- Técnicas de modelamiento de datos
  - IDEF1X
    - Tipos de datos
    - Modelamiento y convenciones
  - Notación Barker

OBJETIVO

Proveer un ambiente que sea conveniente y eficiente para almacenar datos y extraer información.



Cualquiera

Fácil empleo
Orientado a Usuarios

Todos los datos

Agenda

- Structured Query Language (SQL)
  - SQL DDL: Lenguaje para definición de datos
  - SQL DQL: Lenguaje de consulta de datos
  - SQL DML: Lenguaje de manipulación de datos

Structured Query Language

El lenguaje de consulta estructurado (SQL) es el conjunto de declaraciones con las que todos los programas y usuarios acceden a los datos en una base de datos Oracle.

Mandato	Tipo
Create	DDL: Data Definition Language
Alter	
Drop	
Select	DQL: Data Query Language
Insert	DML: Data Manipulation Language
Delete	
Update	
Grant	DCL: Data Control Language
Revoke	

Definición de esquemas (DDL)

CREATE DATABASE	crea una nueva base de datos especificando características físicas de la misma.
CREATE SCHEMA	engloba múltiples comandos CREATE TABLE, CREATE VIEW y GRANT en una sola transacción.
CREATE TABLE	crea una tabla, definiendo sus columnas, restricciones y ubicaciones de almacenamiento.
ALTER TABLE	modifica (redefine) o añade columnas, restricciones o ubicaciones de almacenamiento.
DROP	elimina (remueve) objetos de la base de datos : tablas, vistas, procedimientos, disparadores, índices, paquetes, funciones, usuarios, etc.

Definición de esquemas (DDL)

```
CREATE TABLE table-Name
  {
      ( {column-definition | Table-level constraint}
      [ , {column-definition | Table-level constraint} ] * )
  |
      [ ( column-name [ , column-name ] * ) ]
      AS query-expression
      WITH NO DATA
   }
```

column-definition

```
Simple-column-Name [ DataType ]
    [ Column-level-constraint ]*
    [ [ WITH ] DEFAULT DefaultConstantExpressi
       | generated-column-spec
       | generation-clause
    ]
    [ Column-level-constraint ]*
```

Table-level constraint

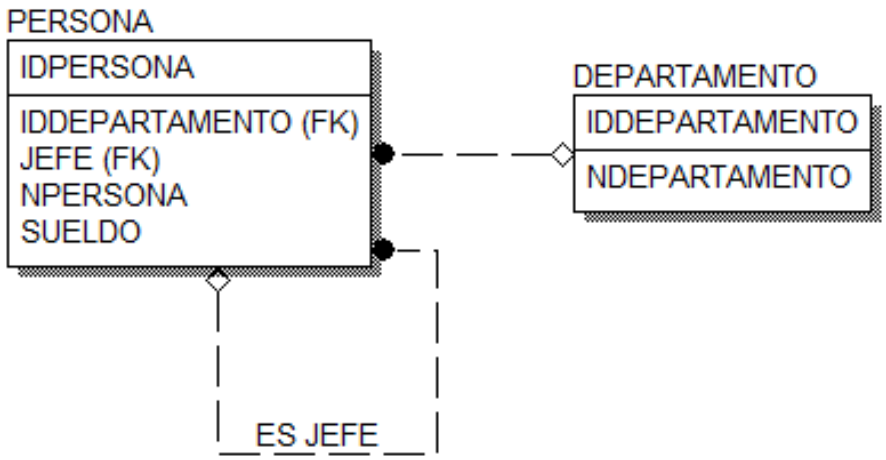
```
[CONSTRAINT constraint-Name]
{
    CHECK (searchCondition) |
    {
        PRIMARY KEY ( Simple-column-Name [ , Simple-column-Name ]* ) |
        UNIQUE ( Simple-column-Name [ , Simple-column-Name ]* ) |
        FOREIGN KEY ( Simple-column-Name [ , Simple-column-Name ]* )
            REFERENCES clause
    }
}
```

Definición de esquemas (DDL)

```
CREATE TABLE scott.emp
    (empno      NUMBER          CONSTRAINT pk_emp PRIMARY KEY,
     ename      VARCHAR2(10)    CONSTRAINT nn_ename NOT NULL
                                CONSTRAINT upper_ename
                                  CHECK (ename = UPPER(ename)),
     job        VARCHAR2(9),
     mgr        NUMBER          CONSTRAINT fk_mgr
                                  REFERENCES scott.emp(empno),
     hiredate   DATE            DEFAULT SYSDATE,
     sal        NUMBER(10,2)    CONSTRAINT ck_sal
                                  CHECK (sal > 500),
     comm       NUMBER(9,0)     DEFAULT NULL,
     deptno     NUMBER(2)       CONSTRAINT nn_deptno NOT NULL
                                CONSTRAINT fk_deptno
REFERENCES scott.dept(deptno) )
                              ;
```

1. Crear la tabla Persona considerando la relación recursiva "es jefe de"

```
CREATE TABLE   Persona (
        IdPersona           CHAR(8) PRIMARY KEY,
        IdDepartamento      CHAR(2) REFERENCES Departamento,
        NPersona            VARCHAR2(40)    NOT NULL,
        Jefe                CHAR(8) REFERENCES Persona,
        Puesto              VARCHAR2(20)    NOT NULL,
        Sueldo              NUMBER (6,2)    DEFAULT 1200    ) ;
```



ALTER TABLE

Las sentencias ALTER TABLE se utilizan para:

- Agregar una nueva columna
- Modificar una columna existente
- Definir un valor DEFAULT para una columna
- Borrar una columna

Puede agregar o modificar una columna de una tabla, pero no puede especificar donde aparece la columna.

2. Añadir la columna Fechaing tipo **date** que represente la fecha de ingreso a laborar y además la columna Sexo tipo **char** de tamaño uno con la integridad NOT NULL.

   alter table Persona add (Fechaing DATE, Sexo CHAR(1) not null)

3. Modificar la columna Sueldo a un tamaño de 7 dígitos, y con un salario de defecto de 1500 soles.

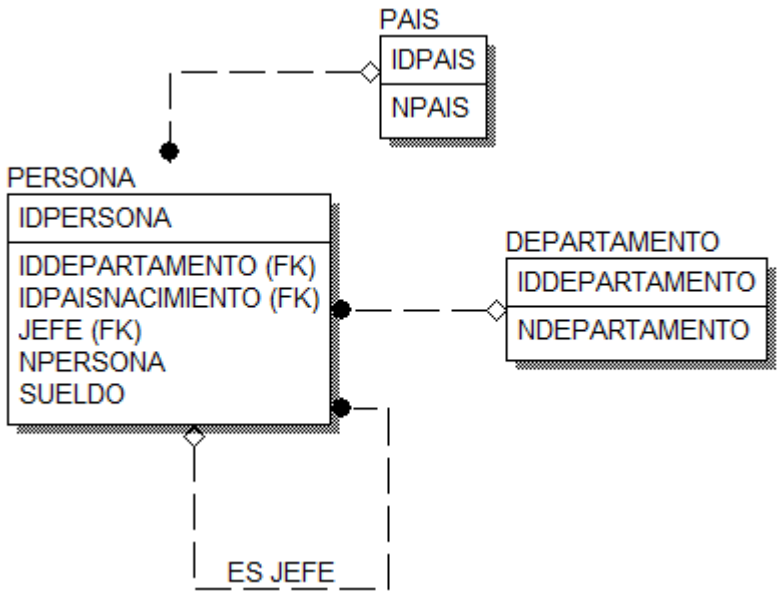
   alter table Persona modify Sueldo NUMBER(7,2) default (1500)

4. Modificar la columna Fechaing para que por defecto se registre la fecha del día.

   alter table Persona modify Fechaing default (sysdate)

5. Agregar la columna país de nacimiento y modificar sueldo con la restricción del mínimo legal.

```
ALTER TABLE Persona
   ADD (PaisNacimiento     CHAR(2) REFERENCES Pais )
   MODIFY ( Sueldo NUMBER (9,2) CHECK ( Sueldo > 800 and
            Sueldo <15000 )  );
```



ALTER TABLE

Al borrar una columna se aplican las siguientes reglas:

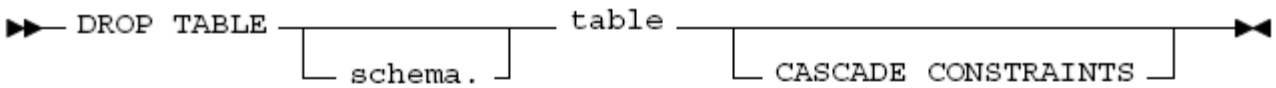
- Se puede borrar una columna que contenga datos
- Solo se puede borrar una columna cada vez
- No puede borrar todas las columnas de una tabla; debe permanecer al menos una columna
- Una vez borrada la columna, los valores de datos de esta no se pueden recuperar

Sintaxis:

ALTER TABLE *tablename*

DROP COLUMN *column name;*

Definición de esquemas (DDL)



schema is the schema containing the table. If you omit *schema*, Oracle7 assumes the table is in your own schema.

table is the name of the table to be dropped.

CASCADE CONSTRAINTS

drops all referential integrity constraints that refer to primary and unique keys in the dropped table. If you omit this option, and such referential integrity constraints exist, Oracle7 returns an error message and does not drop the table.

Example The following statement drops the TEST_DATA table:

```
DROP TABLE test_data
```

DROP TABLE

Permite eliminar la tabla nombrada.

Al borrar una tabla:

- Desaparecen todos los datos
- Cualquier vista y sinónimo referente a la tabla seguirá existiendo, pero ya no funcionarán (conviene eliminarlos)
- Las transacciones pendientes son aceptadas (COMMIT)

El borrado de una tabla es irreversible, y Oracle Server no cuestiona la decisión y borra la tabla inmediatamente.

ÍNDICES

Un índice de Oracle Server es un objeto de esquema que puede acelerar la recuperación de filas mediante un puntero. Los índices se pueden crear explícita o automáticamente.

- Si no hay un índice en la columna seleccionada, se produce una exploración de tabla completa
- Un índice proporciona acceso directo y rápido a las filas de una tabla
- El índice lo utiliza y mantiene automáticamente Oracle Server. Una vez creado un índice, no será necesaria ninguna intervención directa por parte del usuario
- Los índices son lógica y físicamente independientes de la tabla que indexan
- Esto significa que se pueden crear o borrar en cualquier momento sin que afecten a las tablas base o a otros índices

ÍNDICES

Se pueden definir dos tipos de índices:

- Índice único: Oracle Server crea automáticamente este índice al definir una restricción de clave PRIMARY KEY o UNIQUE en una columna de la tabla
- Índice no único: este es un índice que un usuario puede crear para acelerar el acceso a las filas

Para crear un índice en una o más columnas, utilice la sentencia CREATE INDEX:

```
CREATE INDEX index_name
ON table_name (column…, column);


CREATE INDEX wf_cont_reg_id_idx
ON wf_countries (region_id);
```

ÍNDICES

Se debe crear un índice solo si:

- La columna contiene una amplia variedad de valores
- Una columna contiene un gran número de valores nulos
- Una o más columnas se utilizan con frecuencia en conjunto en una cláusula WHERE o una condición de unión
- La tabla es grande y se espera que la mayoría de las consultas recuperen menos del 2 al 4% de las filas

ÍNDICES

Por lo general, no merece la pena crear un índice si:

- La tabla es pequeña
- No se suelen utilizar las columnas como condición en la consulta
- Se espera que la mayoría de las consultas recuperen más del 2 al 4% en la tabla
- La tabla se actualiza con frecuencia
- Se hace referencia a las columnas indexadas como parte de una expresión

ÍNDICES COMPUESTOS

- Un índice compuesto (también denominado índice "concatenado") es un índice creado en varias columnas de una tabla
- Las columnas del índice compuesto pueden aparecer en cualquier orden y no es necesario que sean adyacentes en la tabla

```
CREATE INDEX emps_name_idx
ON employees (first_name, last_name);
```

USO DE VISTAS

Una vista, como una tabla, es un objeto de base de datos. Sin embargo, tenemos que:

- Las vistas no son tablas "reales"
- Son representaciones lógicas de tablas existentes o de otra vista
- Las vistas no contienen datos propios
- Funcionan como una ventana por la que se pueden ver o cambiar los datos de las tablas
- Las tablas en las que se basa la vista se denominan tablas "base"

USO DE VISTAS

La vista es una consulta almacenada como una sentencia SELECT en el diccionario de datos.

```
CREATE VIEW view_employees
AS SELECT employee_id, first_name, last_name, email
FROM employees
WHERE employee_id BETWEEN 100 and 124;


SELECT *
FROM view_employees;
```

Consulta de datos (DQL)

SELECT

Consulta (recupera datos) de una o más tablas o vistas. El resultado es una tabla (filas y columnas) que a su vez puede ser usada en otra sentencia DQL (en ese caso se dice “subquery” o vista temporal).

SELECT

SELECT lista_select ← proyección (π)

FROM lista_tablas ← producto (X)

WHERE condición ← selección (σ)

Theta Join :

$$\pi_{\text{(lista\_select)}} \left(|X|_{\text{(condición)}} \text{(lista\_tablas)} \right)$$

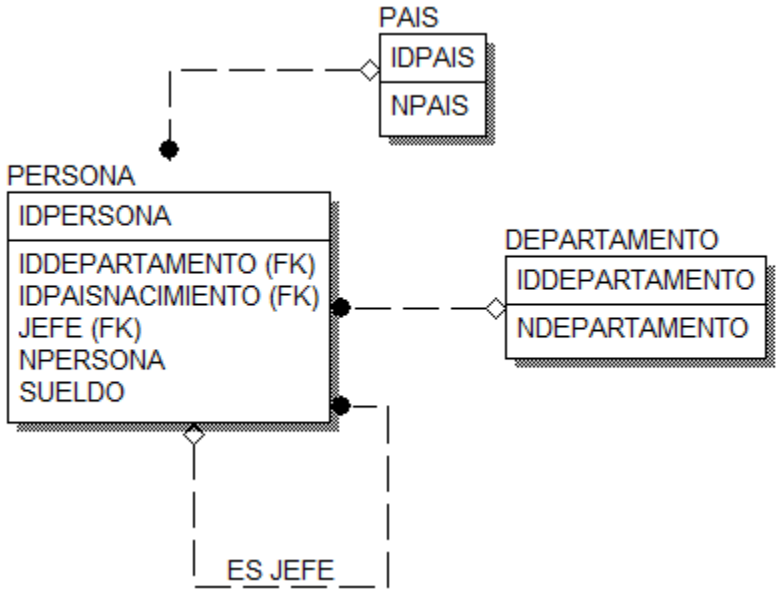
SELECT

Sintaxis:

SELECT items_seleccionables

FROM Tabla1 [,Tabla2,…]

WHERE condición_selección



Items seleccionables:

- Columnas
  - Todas: *
  - Algunas: se especifican
- Expresiones
- Constantes

SELECT

Selección de filas:

DISTINCT: permite eliminar filas duplicadas de la selección. Ejemplo:

SELECT **DISTINCT** job from emp;

WHERE: condición de selección de filas especificas. Está formado por expresiones lógicas.

- Columnas, expresiones o constantes.
- Operador de comparación: =, !=, <>, ^=, >, >=, <, <=, BETWEEN … AND…, IN (lista), IS NULL, LIKE %, _, NOT condición
- Columnas, expresiones o constantes unidas por operadores lógicos: AND, OR, NOT.

Condiciones de búsqueda

Test de comparación (=,<>,<,<=,>,>=)

Operator	Purpose	Example
=	Equality test.	SELECT * FROM emp WHERE sal = 1500
! = ^ = ¬ = < >	Inequality test. All forms of the inequality operator may not be available on all platforms.	SELECT * FROM emp WHERE sal != 1500
>	"Greater than" and "less than" tests.	SELECT * FROM emp WHERE sal > 1500
<		SELECT * FROM emp WHERE sal < 1500

Test de pertenencia a conjunto (IN)

Operator	Purpose	Example
>= <=	"Greater than or equal to" and "less than or equal to" tests.	`SELECT * FROM emp WHERE sal >= 1500` `SELECT * FROM emp WHERE sal >= 1500`

IN	"Equal to any member of" test. Equivalent to "= ANY".	`SELECT * FROM emp WHERE job IN ('CLERK','ANALYST')`
		
		`SELECT * FROM emp WHERE sal IN (SELECT sal FROM emp WHERE deptno = 30)`

Test de pertenencia (IN)

NOT IN	Equivalent to "!=ALL". Evaluates to FALSE if any member of the set is NULL.	SELECT * FROM emp WHERE sal NOT IN (SELECT sal FROM emp WHERE deptno = 30) SELECT * FROM emp WHERE job NOT IN ('CLERK', ANALYST')

Test de rango (BETWEEN)

Operator	Purpose	Example
`[NOT] BETWEEN x AND y`	[Not] greater than or equal to *x* and less than or equal to *y*.	`SELECT * FROM emp WHERE sal BETWEEN 2000 AND 3000`
`EXISTS`	TRUE if a subquery returns at least one row.	`SELECT dname, deptno FROM dept WHERE EXISTS (SELECT * FROM emp WHERE dept.deptno = emp.deptno)`

Test de correspondencia con patrón (LIKE)

`x [NOT] LIKE y [ESCAPE 'z']`	TRUE if *x* does [not] match the pattern *y*. Within *y*, the character "%" matches any string of zero or more characters except null. The character "_" matches any single character. Any character, excepting percent (%) and underbar (_) may follow ESCAPE; a wilcard character will be treated as a literal if preceded by the escape character.	`SELECT * FROM tab1 WHERE col1 LIKE 'A_C/%E%' ESCAPE '/'`

Test de valor nulo (NULL)

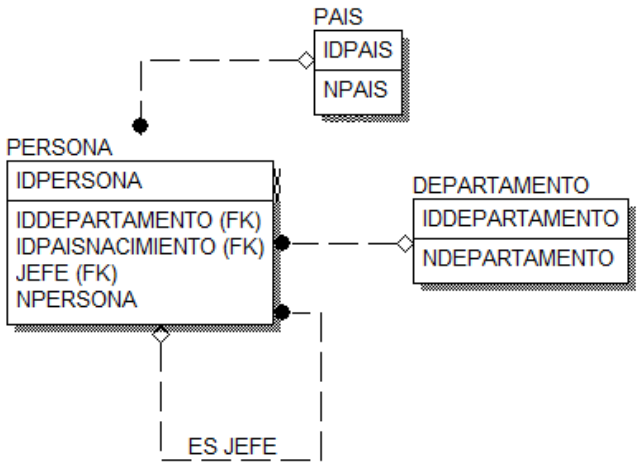
`IS [NOT] NULL`	Tests for nulls. This is the only operator that should be used to test for nulls.	`SELECT dname, deptno FROM emp WHERE comm IS NULL`

6. Obtener las personas con el nombre de su departamento y país de nacimiento.

```
SELECT  Persona.IdPersona, Departamento.NDepartamento, Npais

FROM    Persona, Departamento, Pais

WHERE   Persona.IdDepartamento = Departamento.IdDepartamento
        AND
        Persona.IdPaisNacimiento = Pais.IdPais
```



SELECT DISTINCT

SELECT Puesto FROM Persona;

PUESTO
GERENTE
GERENTE
SUBGERENTE
SUBGERENTE
SUBGERENTE
ASISTENTE
ASISTENTE
ASISTENTE
ASISTENTE
SECRETARIA
SECRETARIA
…..

SELECT DISTINCT Puesto FROM Persona;

PUESTO
GERENTE
SUBGERENTE
ASISTENTE
SECRETARIA
…..

SELECT … UNION…,
SELECT … UNION ALL

```
SELECT Sueldo FROM Persona  WHERE IdDepartamento = '001'
UNION ALL
SELECT Sueldo FROM Persona  WHERE IdDepartamento = '002';
```

Sueldo
1000
1300
1000
1300
1500
...

SELECT … UNION…,
SELECT … UNION ALL

```
SELECT Sueldo FROM Persona  WHERE IdDepartamento = '001'
UNION
SELECT Sueldo FROM Persona  WHERE IdDepartamento = '002';
```

Sueldo
1000
1300
1500
...

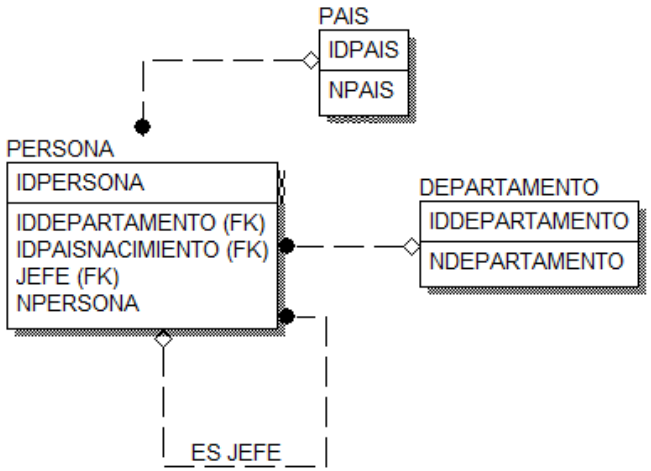
7. Obtener todos los subordinados de “FIESTAS" (podría haber más de un “FIESTAS" que sea jefe).

```
SELECT  J.NPersona "Jefe",  P.NPersona "Subordinado"

FROM    Persona P, Persona J

WHERE  ( P.Jefe = J.IdPersona  AND  J.NPersona LIKE “FIESTAS%"  )

ORDER BY   1 DESC, 2 ASC
```



Ordenación de los resultados de una consulta

Cláusula ORDER BY

Se puede pedir a SQL que ordene los resultados de una consulta incluyendo la cláusula ORDER BY en la sentencia SELECT.

La cláusula ORDER BY, consta de las palabras claves ORDER BY, seguidas de una lista de especificaciones de ordenación separadas por comas.

Muestra las ventas de cada oficina, ordenadas en orden por región y dentro de cada región por ciudad.

SELECT CIUDAD, REGION, VENTAS **FROM** OFICINAS
ORDER BY REGION, CIUDAD

CIUDAD	REGION	VENTAS
Chiclayo	Norte	$2,800,352.00
Trujillo	Norte	$756,000.00
Arequipa	Sur	$1,745,000.00

Ordenación de los resultados de una consulta

Cláusula ORDER BY

Utilizando la cláusula ORDER BY se puede solicitar la ordenación en secuencia ascendente o descendente, y se puede ordenar con respecto a cualquier elemento en la lista de selección de la consulta.
Por omisión, SQL ordena los datos en secuencia ascendente. Para solicitar ordenación en secuencia descendente, se incluye la palabra DESC en la especificación de ordenación.

Lista las oficinas, clasificadas en orden descendente de ventas, de modo que las oficinas con mayores ventas aparezcan en primer lugar.

```
SELECT CIUDAD, REGION, VENTAS FROM OFICINAS
ORDER BY  VENTAS DESC
```

CIUDAD	REGION	VENTAS
Chiclayo	Norte	$2,800,352.00
Arequipa	Sur	$1,745,000.00
Trujillo	Norte	$756,000.00

Ordenación de los resultados de una consulta

Cláusula ORDER BY

También se puede utilizar la palabra clave ASC para especificar orden ascendente, pero puesto que ésta es la secuencia de ordenación por omisión, la palabra clave se suele omitir.

Lista las oficinas, clasificadas en orden descendente de rendimiento de ventas, de modo que las oficinas con mejores rendimientos aparezcan primero.

```
SELECT CIUDAD, REGION, (VENTAS-OBJETIVO) FROM OFICINAS
ORDER BY 3 DESC
```

CIUDAD	REGION	(VENTAS-OBJETIVO)
Chiclayo	Norte	$112,352.00
Arequipa	Sur	$10,740.00
Trujillo	Norte	-$16,000.00

Reglas de prioridad o qué ocurre en primer lugar

- Tenga en cuenta que el operador AND se evalúa antes que el operador OR
- Esto significa que, para que una consulta, si no se cumple alguna de las condiciones de la sentencia AND, se utilizará el operador OR para seleccionar las filas
- Es importante recordar este concepto

ORDEN	OPERADORES
1	Aritméticos + - * /
2	Concatenación ||
3	Comparación <, <=, >, >=, <>
4	IS (NOT) NULL, LIKE, (NOT) IN
5	(NOT) BETWEEN
6	NOT
7	AND
8	OR

Reglas de prioridad o qué ocurre en primer lugar

```
SELECT last_name||' '||salary*1.05
As "Employee Aumento", department_id, first_name
FROM employees
WHERE department_id IN(50,80)
AND first_name LIKE 'C%'
OR last_name LIKE '%s%';

SELECT last_name||' '||salary*1.05
As "Employee Aumento", department_id, first_name
FROM employees
WHERE department_id IN(50,80)
OR first_name LIKE 'C%'
AND last_name LIKE '%s%';
```

Manipulación de datos (DML)

INSERT	Inserta nueva(s) fila(s) a una tabla o a una vista.
DELETE	Elimina las filas de una tabla o vista que cumplan la condición WHERE (el predicado). Si no se especifica la condición, todas las filas son eliminadas.
UPDATE	Actualiza (modifica) los valores de columnas en las filas que cumplan la condición WHERE. Si no se especifica la condición, todas las filas son actualizadas.

INSERT

Inserta una fila en una tabla.

Sintaxis:

Inserción de todas las columnas:
INSERT INTO *Tabla* **VALUES** (*valor1, valor2, …, valorN*)

Inserción de algunas columnas:
INSERT INTO Tabla (*col1, ….., colN*) **VALUES** (*val1, …, valN*)

Ejemplo:
INSERT INTO dept VALUES (50, ‘Finanzas’, ‘Lima’);

INSERT INTO emp (empno, ename, hiredate, sal, deptno)
VALUES (1235, ‘Jorge’, ’01-JAN-11’,2500,30);

8. Realicemos una inserción para observar los resultados de los datos ingresados.

INSERT INTO Persona (idpersona, id, npersona, puesto, sueldo, Paisnacimiento, sexo)
VALUES('10101010','01','Juan Perez','Asistente',1700,'51','M')

> SELECT * FROM Persona;

IDPERSON	ID	NPERSONA	JEFE	PUESTO	SUELDO	FECHAING	S	PA
10101010	01	Juan Perez		Asistente	1700	26/01/19	M	51

DELETE

Elimina los datos de una tabla específica.

Sintaxis:

DELETE from *Tabla*

[WHERE condicion_seleccion**]**

Ejemplo:

DELETE FROM Persona WHERE idpersona = ‘0002’;

UPDATE

Modifica los datos de una tabla específica.

Sintaxis:

UPDATE *Tabla*

SET *col1=val1, …, colN=valorN*

[WHERE condicion_seleccion**]**

Ejemplo:

```
UPDATE emp
SET job='salesman', sal=sal*1.5, deptno=30
WHERE deptno = (SELECT deptno FROM emp WHERE empno = 7788);
```

Resumen

En esta sesión, debe haber aprendido lo siguiente:

- SQL DDL son comandos que se utilizan para crear la estructura física de la base de datos
- SQL DQL es un comando que se utiliza para consultar los datos de la base de datos
- SQL DML son comandos que se utilizan para modificar los datos de la base de datos