PUCP

1INF33 Base de Datos

# SQL-DML

2024

Profesores del curso

# ÍNDICE

# Saberes previos

- Data Definition Language (DDL)

- Técnicas de modelamiento de datos
  - Notación Barker
    - Tipos de datos
    - Modelamiento y convenciones
  - Notación IDEF1X

# Objetivos

# Objetivos

- Presentar la utilidad y uso de las principales sentencias de manipulación de datos (DML): Select, Insert, Update y Delete

# Sentencia SQL Select

# Sentencia SQL SELECT

- La sentencia **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 DML (en ese caso se dice “subconsulta” o vista temporal).

# Sentencia SQL SELECT

## Tipos de consultas

- Consultas simples
- Consultas con funciones agregadas
- Consultas con GROUP BY / HAVING
- Consultas con ORDER BY
- Consultas de unión
- Consultas anidadas
- Consultas correlacionadas
- Consultas con ALL/ANY
- Consultas con EXISTS
- Consultas con subconsultas en FROM/WHERE
- Consultas con operaciones de conjunto

# Sentencia SQL SELECT

| | | | |
|---|---|---|---|
| **SELECT** | lista_select | ← | proyección (π) |
| **FROM** | lista_tablas | ← | producto (X) |
| **WHERE** | condición | ← | selección (σ) |

Theta Join :

$\pi$ (lista_select) ( |X| (condición) (lista_tablas) )

# Sentencia SQL SELECT

- Sintaxis:

  **SELECT** items_seleccionables

  **FROM** Tabla1 [,Tabla2,...]

  **WHERE** condición_selección

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

# Sentencia SQL SELECT

- Selección de filas:

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

  SELECT **DISTINCT** job FROM emp;

- Condición de selección de filas específicas.

  **WHERE**: 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.

# Test de comparación ( =, <>, <, <=, >, >= )

| Operador | Propósito | Ejemplo |
|---|---|---|
| = | Compruebe si dos expresiones son o no iguales. Si las expresiones son iguales, la condición es verdadera y se devuelven los registros coincidentes | `SELECT * FROM emp WHERE sal=1500;` |
| != <> ^= | Compruebe si dos expresiones son o no iguales (diferentes) | `SELECT * FROM emp WHERE sal != 1500;` |
| > <br> < | Mayor que y menor que | `SELECT * FROM emp WHERE sal > 1500;` <br> `SELECT * FROM emp WHERE sal <1500;` |
| >= <br> <= | Mayor o igual que y menor o igual que | `SELECT * FROM emp WHERE sal >=1500;` <br> `SELECT * FROM emp WHERE sal <=1500;` |

# Test de pertenencia a conjunto (IN)

| Operador | Propósito | Ejemplo |
|---|---|---|
| IN | Igual a cualquier valor de una lista de valores. Equivalente a "=ANY" | `SELECT * FROM emp`<br>`WHERE job IN ('CLERK', 'ANALYST');`<br><br>`SELECT * FROM emp WHERE sal IN`<br>`(SELECT sal FROM emp WHERE deptno = 30);` |
| NOT IN | Compruebe si dos expresiones son o no iguales. Si las expresiones no son iguales, la condición es verdadera y devuelve los registros no coincidentes. | `SELECT * FROM emp`<br>`WHERE job NOT IN ('CLERK', 'ANALYST');`<br><br>`SELECT * FROM emp`<br>`WHERE sal NOT IN`<br>`SELECT sal FROM emp WHERE deptno = 30);` |

# Test de rango (BETWEEN) y existencia (EXISTS)

| Operador | Propósito | Ejemplo |
|---|---|---|
| [NOT] BETWEEN *x* AND *y* | Comprueba valores entre un rango determinado y devuelve los valores coincidentes. La condición BETWEEN es inclusiva. Los valores inicial y final están incluidos. | `SELECT * FROM emp WHERE sal BETWEEN 200 AND 300;` |
| EXISTS | El operador EXISTS devuelve verdadero si la subconsulta devuelve alguna fila, de lo contrario, devuelve falso. | `SELECT dname, deptno FROM dept WHERE EXISTS (SELECT * FROM emp WHERE dept.deptno = emp.deptno);` |

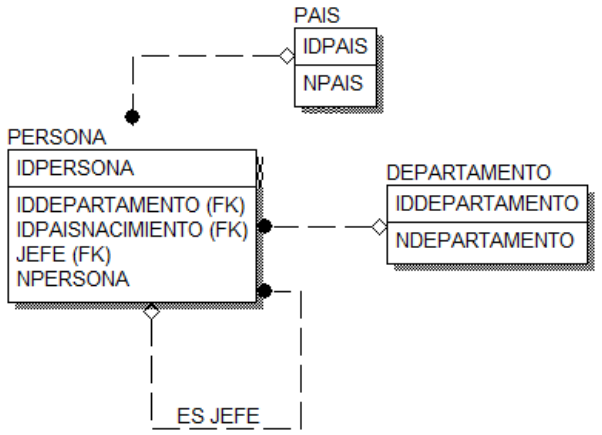
# Test de correspondencia con patrón (LIKE) y test de valor nulo (NULL)

| Operador | Propósito | Ejemplo |
|---|---|---|
| x [NOT] LIKE | Realiza y devuelve una coincidencia de patrón utilizando comodines en la cláusula WHERE de una sentencia SELECT | `SELECT * FROM tab1`<br>`WHERE col1 LIKE 'A_C%E%';` |
| IS [NOT] NULL | Es el operador para comprobar si un valor en una columna o una expresión es NULL o no. | `SELECT dname, deptno`<br>`FROM emp WHERE comm IS NULL;`<br><br>`SELECT dname, deptno`<br>`FROM emp WHERE comm IS NOT NULL;` |

# Sentencia SQL SELECT

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
```



# Sentencia SQL SELECT

## 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
…..

# Sentencia SQL SELECT

**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 |
| … |

# Sentencia SQL SELECT

**SELECT … UNION…,**
**SELECT … UNION ALL**

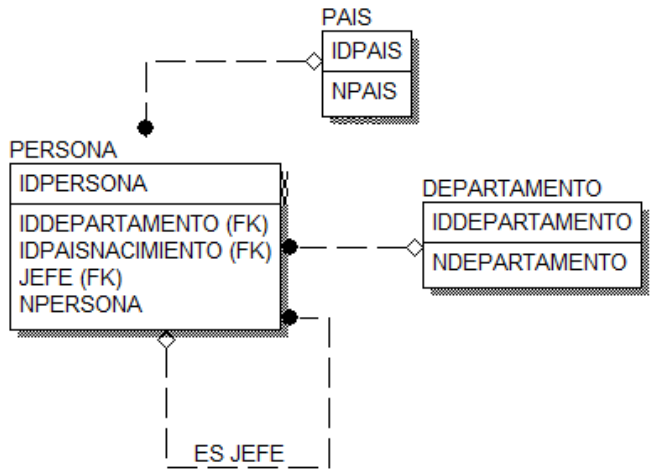
```
SELECT Sueldo FROM Persona
WHERE IdDepartamento = ‘001’
UNION
SELECT Sueldo FROM Persona
WHERE IdDepartamento = ‘002’;
```

| Sueldo |
|---|
| 1000 |
| 1300 |
| 1500 |
| … |

# Sentencia SQL SELECT

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
```




* Uso de alias de tablas afecta al nombre de la tabla en la instrucción SELECT. Este alias es temporal y dura únicamente mientras se ejecuta y no queda guardado en ningún lugar.

# 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.

Ejemplo: Mostrar 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

- 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.

Ejemplo: Listar 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.

Ejemplo: Listar 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 |

# Prioridad de los operadores

## 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 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 |

# Prioridad de los operadores

## Reglas de prioridad o qué ocurre en primer lugar

```
SELECT last_name||' '||salary*1.05 As "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 “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

# SQL - DML

## 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.

# SQL - DML

## INSERT

Permite insertar una nueva fila en una tabla.

- Sintaxis:
  - Inserción de todas las columnas:

    **INSERT INTO** *Tabla* **VALUES** (*valor1, valor2, ..., valorN*)

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

  - Inserción de algunas columnas:

    **INSERT INTO** *Tabla* (*col1, ....., colN*) **VALUES** (*val1, ..., valN*)

    Ejemplo: INSERT INTO emp (empno, ename, hiredate, sal, deptno)
    VALUES (1235, 'Jorge', '01-JAN-2024',2500,30);

# SQL - DML

## 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 |

# SQL - DML

**DELETE**
Elimina los datos de una tabla específica.

- Sintaxis:

  **DELETE from** *Tabla*
  **[WHERE** condición_selección**]**

- Ejemplo:

```
DELETE FROM Persona
WHERE idpersona = '0002';

DELETE FROM Persona
WHERE idpersona >= '0005' AND idpersona <= '0013' ;
```

# SQL - DML

## 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);
```

# Conclusiones

En esta sesión, debe haber aprendido lo siguiente:

- El comando de SQL que se utiliza para consultar los datos de la base de datos: SELECT.
- Los comandos de SQL que se utilizan para modificar los datos de la base de datos: INSERT, UPDATE y DELETE.

# Referencias


- AR. Elmasri y S.B. Navathe. (2007). Fundamentos de Sistema de Base de Datos, 5ta edición
- Oracle Help Center. (12 de septiembre de 2024). *SQL Language Reference*. https://docs.oracle.com/en/database/oracle/oracle-database/23/sqlrf/Conditions.html#GUID-C2E3ED44-16E7-4924-9125-E1693B1022A8

# ¡Gracias!

PUCP