Resumen SQL

CONSULTAS

```
SELECT [ DISTINCT ] listaColumnas
FROM listaTablas
[ WHERE condición para filas]
[ GROUP BY listaColumnas por las que se quiere agrupar
[ HAVING condición para los grupos] ]
[ ORDER BY listaColumnas [ ASC | DESC ] ]
```

ARITMÉTICAS

```
COUNT( * ) número de filas
COUNT( [DISTINCT] expr ) número de valores distintos en expr
SUM( [DISTINCT] expr ) suma de todos los valores en expr
AVG( [DISTINCT] expr ) promedio de todos los valores en expr
MIN( expr ) el más pequeño de todos los valores en expr
MAX( expr ) el mayor de todos los valores en expr
```

TRIGGERS

- Funcionan como alarmas en la base de datos.
- Sirven para verificar datos o registrar lo que sucede en la base de datos.
- Se asocian con eventos de la base de datos, como INSERT, UPDATE o DELETE a tablas específicas.
- No se llaman, se activan automáticamente cuando ocurre el evento que están monitoreando (alarmas).

```
CREATE TRIGGER nombre_del_trigger
{BEFORE | AFTER} {INSERT | UPDATE | DELETE} ON nombre_de_tabla
FOR EACH ROW
BEGIN
    -- Código del trigger
END;
```

PROCEDIMIENTOS

- Son listas de tareas que se pueden almacenar, guardar y ejecutar en la base de datos.
- Son bloques de código que se almacenan en la base de datos y se llaman desde una aplicación o consulta SQL.
- Pueden aceptar parámetros de entrada y manipular datos, o realizar cálculos o transacciones.
- No devuelven datos.

```
CREATE PROCEDURE nombre_del_procedimiento
(parámetros)
BEGIN
    -- Código del procedimiento
END;
```

FUNCIONES

- Sirven para hacer cálculos o cambios de datos.

- Pueden aceptar parámetros de entrada y manipular datos, o realizar cálculos o transacciones.
- Devuelven datos.

```
CREATE FUNCTION nombre_de_función
(parámetros)
RETURNS tipo_de_dato
BEGIN
    -- Código de la función
END;
```

ÍNDICES

- Actúan como "filtro" ante ambigüedad en nombres de columnas y reducen la búsqueda solo a un subconjunto de una tabla.

```
CREATE INDEX idx_cliente ON cliente(provincia);
DROP INDEX nom_índice;
```

idx_cliente -> nombre del índice

EXPLAIN PLAN

- Da detalles sobre como se realizará una consulta específica, como el orden de acceso a tablas, los índices que se utilizarán y operaciones de filtro.
- La salida de explain plan se almacena en la tabla **PLAN_TABLE**, la cual contiene:
  - **Statement_id varchar2(30)** Valor opcional para identificar planes de ejecución.
  - **Operation varchar2(30)** Nombre de la operación interna realizada.
  - **Options varchar2(225)** Detalles sobre la operación.
  - **Object_name varchar2(30)** Nombre de la tabla o índice.
  - **Position numeric** Para la primera fila índice al coste estimado por el optimizador para realizar la sentencia, para el resto índice la posición relativa respecto al padre.

```
EXPLAIN PLAN FOR
SELECT column FROM table_name WHERE condition;
```

VISTAS

- Es una tabla lógica que no contiene sus propios datos, pero refleja los de la base de datos de manera dinámica.
- Si se modifica la base de datos también se actualizará la vista.
- "Simplificación" de una tabla para su visualización.
- Se pueden hacer consultas de visualización como las SELECT, JOIN, consultas con funciones... pero no de modificación como INSERT, UPDATE, DELETE, agregación sin GROUP BY...

```
CREATE VIEW nombre_de_vista AS
SELECT column FROM nombre_de_tabla WHERE condición;

CREATE VIEW VistaEjemplo AS
SELECT Nombre, Apellido, Edad FROM Empleados WHERE Departamento = 'Ventas';

-- Se puede acceder a los datos de las vistas como si se tratara de una tabla
```

```sql
SELECT * FROM VistaEjemplo;

DROP view VistaEjemplo;
```

```
```