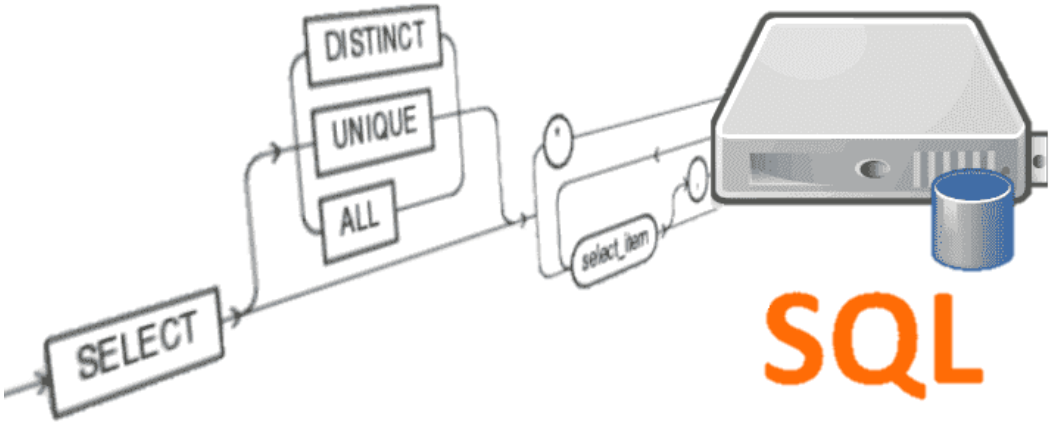
SQL

Structured Query Language

(Lenguaje de Consulta Estructurada)



Manipulación de datos - DISTINCT

DISTINCT elimina duplicados del resultados de una consulta, mostrando solo los valores únicos en una o más columnas seleccionadas.

SELECT DISTINCT columna1, columna2, ...
FROM nombre_tabla
WHERE condición;columna1, columna2, ...

Obtener las ciudades en las que viven tus clientes sin repetir valores

SELECT DISTINCT ciudad
FROM clientes;

Manipulación de datos - DISTINCT

Si tienes una tabla pedidos que incluye las columnas id_cliente y producto, y quieres obtener las combinaciones únicas de clientes y productos pedidos, puedes usar DISTINCT en ambas columnas.

SELECT DISTINCT id_cliente, producto
FROM pedidos;

DISTINCT elimina filas duplicadas donde las combinaciones de id_cliente y producto se repiten

Manipulación de datos - DISTINCT

DISTINCT también puede combinarse con funciones como **COUNT** para contar valores únicos.

Por ejemplo, si quieres saber cuántas ciudades distintas hay en la tabla clientes

```
SELECT COUNT(DISTINCT ciudad) AS ciudades_distintas
FROM clientes;
```

En este caso, **COUNT(DISTINCT ciudad)** cuenta solo las ciudades únicas en la tabla clientes.

Manipulación de datos - INSERT

INSERT agrega nuevos registros (filas) en una tabla de una base de datos.

Insertar una fila especificando todos los valores para cada columna. Los valores deben estar en el mismo orden de las columnas de la tabla

INSERT INTO nombre_tabla **VALUES** (valor1, valor2, valor3, ...)

Supongamos que tenemos una tabla llamada clientes con las columnas id_cliente, nombre, y edad

INSERT INTO clientes **VALUES** (1, 'Juan Pérez', 30);

Manipulación de datos - INSERT

Insertar una fila especificando columnas. Si alguna columna acepta nulos o tiene configurado un valor por defecto, se puede especificar solo las columnas en las que van a insertar datos.

INSERT INTO nombre_tabla (columna1, columna2, ...)
VALUES (valor1, valor2, ...)

Continuando con la tabla clientes, solo se agrega nombre y edad, idCliente se genera automáticamente

INSERT INTO clientes (nombre, edad) **VALUES** ('Ana Gómez', 25);

Manipulación de datos - INSERT

Insertar varias filas en una sola consulta.

```
INSERT INTO nombre_tabla (columna1, columna2, ...)
VALUES (valor1a, valor2a, ...),
       (valor1b, valor2b, ...),
        ...,
       (valor1n, valor2n, ...);
```

Agregar tres clientes en una sola consulta

```
INSERT INTO clientes (nombre, edad)
VALUES  ('Carlos López', 40),
        ('Maria Sanchez', 35),
        ('Jose Martinez', 28);
```

Manipulación de datos - INSERT

Insertar resultados de otra consulta (INSERT INTO ... SELECT).

INSERT INTO tabla_destino (columna1, columna2, ...)
SELECT columna1, columna2, ...
FROM tabla_origen **WHERE** condición

Insertar los clientes menores de 30 años de cliente a **cliente_joven**

INSERT INTO cliente_joven (id_cliente, nombre, edad)
SELECT id_cliente, nombre, edad **FROM** cliente **WHERE** edad < 30;

En caso de que las dos tablas tengan la misma estructura se puede usar

INSERT INTO cliente_joven
SELECT * **FROM** cliente **WHERE** edad < 30;

Manipulación de datos - INSERT

Insertar datos utilizando valores por defecto (DEFAULT). Si la tabla tiene columnas configuradas con valores por defecto se puede usar DEFAULT

INSERT INTO nombre_tabla (columna1, columna2, columna3)
VALUES (valor1, **DEFAULT**, valor3)

Agregar un cliente, con la columna ciudad configurada por defecto con un valor predeterminado

INSERT INTO cliente (id_cliente, nombre, ciudad)
VALUES (3, 'Luis Romero', **DEFAULT**);

Manipulación de datos - INSERT

Resumen

INSERT INTO ... **VALUES**: Añadir una sola fila.

INSERT INTO ... (col) **VALUES**: Añadir fila(s) especificando columnas.

INSERT INTO ... **VALUES** (), (), ...: Añadir múltiples filas.

INSERT INTO ... **SELECT**: Añadir datos basados en resultados de consulta.

INSERT INTO ... **VALUES** (..., **DEFAULT**, ...): Insertar valores con por default

Manipulación de datos - UPDATE

UPDATE modifica los valores de una o más columnas de una tabla.

Actualizar una fila específica usando WHERE. Con WHERE se define cuál fila debe actualizarse, pueden ser mas de una columna al mismo tiempo

UPDATE nombre_tabla **SET** columna1 = valor1, columna2 = valor2,
WHERE condición

Actualiza la cuidad y el teléfono del cliente 3 (id_cliente = 3)

UPDATE cliente **SET** ciudad = ‘Capital’, teléfono=22334455
WHERE id_cliente = 3;

Manipulación de datos - UPDATE

Actualizar varias filas con una condición. De manera similar a la anterior, las filas a modificar dependen de la condición en la cláusula WHERE

Aumentar el sueldo en 30 % a los empleados del sector ventas

UPDATE empleado **SET** sueldo = sueldo * 1.3
WHERE sector = 'ventas'

Actualizar todas las filas de una tabla (sin WHERE). Esta menara tan practica como peligrosa, se debe usar con mucho cuidado

Aumentar el sueldo en 30 % de todos los empleados

UPDATE empleado **SET** sueldo = sueldo * 1.3

Manipulación de datos - UPDATE

Actualizar datos usando JOIN: A veces es útil actualizar datos de una tabla basándote en información de otra tabla, lo cual puedes hacer con JOIN en el comando UPDATE

Actualizar el estado de clientes a ‘activo’ solo para aquellos que tienen una suscripción activa.

UPDATE clientes **SET** estado = ‘activo’
FROM clientes c
INNER JOIN suscripciones s **ON** c.id_cliente = s.id_cliente
WHERE s.estado = 'activa';

Manipulación de datos - UPDATE

Resumen

UPDATE ... **SET** ... **WHERE**: Actualizar una o más columnas en filas específicas.

UPDATE ... **SET** ...: Actualizar todas las filas de una tabla (sin WHERE).

UPDATE ... **SET** ... **WHERE** ... (**subconsulta**): Actualizar datos basados en resultados de una subconsulta.

UPDATE ... **JOIN**: Actualizar columnas en función de condiciones y valores de otras tablas.

Ejemplo - UPDATE con INNER JOIN

Actualizar el estado de los clientes a "con pedido pendiente" si tienen al menos un pedido con estado "pendiente".

Clientes(id_cliente, nombre, estado)
Pedidos(id_pedido, id_cliente, estado_pedido)

UPDATE clientes **SET** estado = 'con pedido pendiente'
FROM clientes c **INNER JOIN** pedidos p **ON** c.id_cliente = p.id_cliente
WHERE p.estado_pedido = 'pendiente'

La condición WHERE p.estado_pedido = 'pendiente' asegura que solo se actualizarán los clientes que tienen pedidos pendientes, cambiando su estado

Ejemplo - UPDATE con INNER JOIN

Aumentar el límite de crédito en 20% a los clientes que tienen un gasto total mayor a $5000

Clientes(id_cliente, nombre, limite)
Transacciones(id_transaccion, id_cliente, monto)

UPDATE clientes **SET** limite = limite * 1.2 **FROM** clientes c **INNER JOIN**

(**SELECT** id_cliente c, **SUM**(monto) **FROM** transacciones t
GROUP BY id_cliente **HAVING SUM**(monto) > 5000) **AS** gastosT

ON c.id_cliente = gastosT.id_cliente

La subconsulta calcula el gasto de cada cliente y filtra los mayor a $5000

Ejemplo - UPDATE con INNER JOIN

Cambiar el estado de los productos a “Prioritario" si el stock es menor que la cantidad

productos(id_producto, nombre, stock, estado)
pedido(id_producto, cantidad, fecha)

UPDATE productos **SET** estado = ‘Prioritario’
FROM productos p
INNER JOIN pedido pe **ON** p.id_producto = pe.id_producto
WHERE p.stock < pe.cantidad

Manipulación de datos - DELETE

DELETE elimina registros específicos de una tabla.

Eliminar filas específicas con una condición (WHERE). La cláusula WHERE especifica qué registros serán eliminados. Si omites (WHERE) se eliminarán todas las filas de la tabla

DELETE FROM nombre_tabla **WHERE** condición

Eliminar el cliente 3 (id_cliente = 3)

DELETE FROM clientes **WHERE** id_cliente = 3

Manipulación de datos - DELETE

Eliminar varias filas que cumplen con una condición específica. Se usa los operadores lógicos (**AND, OR**) o de comparación **<, >, IN**, etc.

Eliminar los empleados mayores de 70 años y que cobren mas de 990000

DELETE FROM empleado
WHERE edad > 70 **AND** sueldo > 990000

Eliminar todas las filas de una tabla (sin WHERE). Este comando elimina todas las filas y no se pueden recuperar sin un respaldo

DELETE FROM cliente

Eliminar todos los registros de la tabla clientes

Manipulación de datos - DELETE

Eliminar datos usando JOIN. Eliminar datos de una tabla basándose en criterios que involucran múltiples tablas.

Eliminar los clientes que no tienen una suscripción activa

DELETE cliente **FROM** clientes c
LEFT JOIN suscripciones s **ON** c.id_cliente = s.id_cliente
WHERE s.id_cliente **IS NULL**

Este comando elimina a todos los clientes que no tienen registros correspondientes en suscripciones (no tienen una suscripción activa).

Manipulación de datos - DELETE

Resumen

DELETE FROM ... **WHERE**: Eliminar filas específicas que cumplen una condición.

DELETE FROM ... **WHERE** condición_lógica: Eliminar varias filas que cumplen condiciones múltiples.

DELETE FROM ...: Eliminar todas las filas de una tabla (sin WHERE).

DELETE FROM ... **JOIN**: Eliminar filas usando condiciones basadas en otras tablas

Ejemplo - DELETE con INNER JOIN

Eliminar empleados que no están en ningún proyecto.

Empleado(id_empleado, nombre, departamento)
Asignado (id_proyecto, id_empleado)

DELETE e
FROM empleado e
LEFT JOIN Asignado a **ON** e.id_empleado = a.id_empleado
WHERE a.id_empleado **IS NULL**

Aquí usamos un LEFT JOIN para identificar a los empleados que no tienen coincidencias en la tabla Asignados (su valor es NULL), lo cual significa que no están asignados a ningún proyecto.

Ejemplo - DELETE con INNER JOIN

Eliminar pedidos con productos descontinuados. (disponible = FALSE)

Pedido(id_pedido, id_producto, cantidad)
Producto(id_producto, nombre, disponible)

```
DELETE p
FROM pedido p
INNER JOIN producto pr ON p.id_producto = pr.id_producto
WHERE pr.disponible = FALSE
```

En este caso, usamos un INNER JOIN para encontrar todos los registros de pedidos que tienen productos cuyo campo disponible está en FALSE.

Ejemplo - DELETE con INNER JOIN

Eliminar estudiantes sin calificaciones en ninguna materia

Estudiante (id_estudiante, nombre)
Calificacion (id_estudiante, id_materia, calificación)

```
DELETE e
FROM estudiante e
LEFT JOIN calificacion c ON e.id_estudiante = c.id_estudiante
WHERE c.id_estudiante IS NULL
```

LEFT JOIN identifica los estudiantes que no tienen calificaciones.

Manipulación de datos - Truncate

TRUNCATE elimina todos los registros de la tabla sin afectar su estructura (no afecta a las columnas)

DELETE, elimina las filas una por una, mientras que **TRUNCATE** elimina los datos en bloques, liberando el espacio de almacenamiento ocupado por la tabla y reinicia cualquier columna de identidad (autoincremental) a su valor inicial.

TRUNCATE TABLE nombre_tabla

Eliminar los datos de tabla cliente

TRUNCATE TABLE cliente

Muchas Gracias