

Nombre:

Fecha:

Profesor:

Materia:

Institución:

Curso:

Nota:

Que es Normalización?

La normalización es la transformación de las vistas de usuario complejas y del almacenamiento de datos a un juego de estructuras de datos más pequeñas y estables, consiste en:

- Reducir la repetición de datos
- Evitar anomalías de inserción, actualización y eliminación
- Mejorar la integridad de los datos

Ej:

id_estudiante	Nombre	Curso	Profesor
1	Ana Lopez	Matematicas	Carlos Gomez
1	Ana Lopez	Lengua	Laura Sanchez
2	Juan Ruiz	Matematicas	Carlos Gomez

Primera Forma Normal (1FN)

• Todos los atributos contienen valores atómicos (no multivalores ni listas)

• No hay grupos repetitivos

Error en ejemplo base:

La tabla no muestra listas, pero repite información

Ej:

id_estudiante	Nombre	Curso	Profesor
1	Ana Lopez	Matematicas	Carlos Gomez
1	Ana Lopez	Lengua	Laura Sanchez
2	Juan Ruiz	Matematicas	Carlos Gomez

Segunda Forma Normal (2FN)

Una tabla esta en segunda forma normal si:

- Esta en 1FN
- Todos los atributos no clave dependen completamente de la clave primaria

La clave podria ser compuesta: (id_estudiante, curso)
pero Nombre depende del id_estudiante y
Profesor depende solo de Curso

Ej: Tabla Estudiante:

id_estudiante	Nombre
1	Ana Lopez
2	Juan Ruiz

Tabla Curso:

Curso	Profesor
Matematicas	Carlos Gomez
Lengua	Laura Sanchez

Tabla Estudiante-Curso (Relacion):

id_estudiante	Curso
1	Matematicas
1	Lengua
2	Matematicas

Tercera Forma Normal (3FN)

Una tabla esta en tercera forma normal si:

- Esta en 2FN
- No existen dependencias transitivas (un atributo depende de otro que no es clave)

Possible error: Supon que en la tabla Curso tienes tambien la Facultad y esa facultad tiene un decano. Si ambos estan en Curso, entonces hay una dependencia: Curso \rightarrow Facultad \rightarrow Decano \rightarrow dependencia transitiva.

Ej: Tabla Curso:

id-curso	nombre	id-facultad
1	Matematicas	10
2	Lengua	11

Tabla Facultad:

id-facultad	nombre-facultad	decano
10	Ciencias	Dr. Suarez
11	Letras	Dra. Perez

1FN: Eliminar grupos repetitivos y listas, usar valores atómicos.

2FN: Eliminar dependencias parciales (en claves compuestas).

3FN: Eliminar dependencias transitivas.

5 Ejemplos de cada Función

SELECT

```
SELECT nombre FROM Estudiante;  
SELECT nombre, precio FROM Producto;  
SELECT * FROM Empleado;  
SELECT fecha_nacimiento FROM Cliente;  
SELECT descripcion FROM Curso;
```

JOIN

```
SELECT c.nombre, p.fecha FROM Cliente c  
JOIN Pedido p ON c.id_cliente = p.id_cliente;
```

```
SELECT e.nombre, d.nombre FROM Empleado e  
JOIN Departamento d ON e.id_departamento = d.id_departamento;
```

```
SELECT a.nombre, L.nombre FROM Autor a  
JOIN Libro L ON a.id_autor = L.id_autor;
```

```
SELECT u.nombre, r.nombre FROM Usuario u  
JOIN Rol r ON u.id_rol = r.id_rol;
```

```
SELECT s.nombre, c.nombre FROM Servicio s  
JOIN Categoria c ON s.id_categoria = c.id_categoria;
```

GROUP BY

```
SELECT ciudad, COUNT(*)  
FROM Cliente  
GROUP BY ciudad;
```

```
SELECT categoria, AVG(precio)
FROM Producto
GROUP BY categoria;
```

```
SELECT rol, COUNT(*)
FROM Usuario
GROUP BY rol;
```

```
SELECT estado, COUNT(*)
FROM Pedido
GROUP BY estado;
```

```
SELECT tipo, SUM(saldo)
FROM Cuenta
GROUP BY tipo;
```

HAVING

```
SELECT ciudad, COUNT(*)
FROM Cliente
GROUP BY ciudad HAVING COUNT(*) > 5;
```

```
SELECT categoria, AVG(precio)
FROM Producto
GROUP BY categoria HAVING AVG(precio) > 50000;
```

```
SELECT estado, COUNT(*)
FROM Pedido
GROUP BY estado HAVING COUNT(*) < 3;
```

```
SELECT rol, COUNT(*)
FROM Usuario
GROUP BY rol HAVING COUNT(*) >= 2;
```

```
SELECT tipo, SUM(saldo)
FROM Cuenta
ORDER BY tipo HAVING SUM(saldo) > 1000000;
```

ORDER BY

```
SELECT nombre FROM Producto
ORDER BY nombre ASC;
```

```
SELECT precio FROM Producto
ORDER BY precio DESC;
```

```
SELECT fecha_nacimiento FROM Estudiante
ORDER BY fecha_nacimiento;
```

```
SELECT nombre FROM Empleado
ORDER BY Salario DESC;
```

```
SELECT descripcion FROM Curso
ORDER BY descripcion ASC;
```

Nombre:

Fecha:

Año

Mes

Día

Profesor:

Materia:

Institución:

Curso:

Nota:

LIMIT

SELECT * FROM Producto LIMIT 10;

SELECT nombre FROM Cliente LIMIT 5;

SELECT nombre, precio FROM Producto LIMIT 3;

SELECT * FROM Pedido LIMIT 1;

SELECT descripcion FROM Curso LIMIT 2;

COUNT

SELECT COUNT(*) FROM Cliente;

SELECT COUNT(*) FROM Producto

SELECT COUNT (ciudad) FROM Cliente;

SELECT COUNT(*) FROM Pedido WHERE estado='Entregado';

SELECT COUNT(*) FROM Evaluacion WHERE nota >= 4.0;

SUM

SELECT SUM(precio) FROM Producto;

SELECT SUM(salario) FROM Empleado;

SELECT SUM(foto) FROM Factura;

SELECT SUM(saldo) FROM Cuenta;

SELECT SUM(precio * cantidad) FROM Detalle_Factura;

AVG

SELECT AVG(precio) FROM Producto;

SELECT AVG(salario) FROM Empleado;

SELECT AVG(notas) FROM Evaluacion;

SELECT AVG(costo) FROM Servicio;

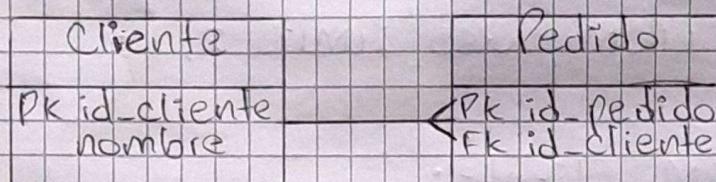
SELECT AVG(edad) FROM Estudiante;

DISTINCT

```
SELECT DISTINCT ciudad FROM Cliente;  
SELECT DISTINCT categoria FROM Producto;  
SELECT DISTINCT estado FROM Pedido;  
SELECT DISTINCT tipo FROM Cuenta;  
SELECT DISTINCT especialidad FROM Profesori;
```

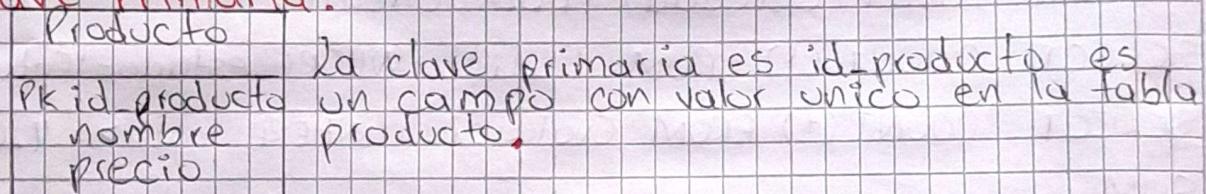
Ejemplo de cada exposición:

Clave Foranea + DELETE



```
DELETE FROM Cliente WHERE id-cliente = 1
```

Clave Primaria:



TRUNCATE:

```
TRUNCATE TABLE Producto;
```

El truncate sirve para borrar todos los datos de una tabla en este caso Producto sin eliminar su estructura.

UPDATE: Sirve para actualizar datos dentro de una tabla.

```
UPDATE Cliente SET nombre = 'Carlos Perez'  
WHERE id-cliente = 1;
```

Normalización:

Tabla No Normalizada:

id	nombre	Curso1	Curso2
1	Juan Ruiz	Matematicas	Fisica

No esta normalizada por lo que tiene campos repetitivos.

Tabla Primera Forma Normal (1FN):

id	nombre	Curso
1	Juan Ruiz	Matematicas
1	Juan Ruiz	Fisica

Ahora, la tabla esta en 1FN porque ya no hay campos repetitivos y todos los atributos contienen valores atómicos.