Curso de Bases de Datos con SQL

Carolina Castañeda Castañeda

Sistema de Gestión de Bases de Datos

Maneja todas las operaciones relacionadas con la base de datos, como la inserción, actualización, eliminación y consulta de datos.

Gestión de transacciones

Soporte para ACID

Atomicidad, Consistencia, Aislamiento y Durabilidad, que garantizan la integridad y confiabilidad de las operaciones de la base de datos.

Control de concurrencia

Mecanismos para permitir que múltiples usuarios realicen operaciones al mismo tiempo sin conflictos, evitando problemas como condiciones de carrera o bloqueos de datos.

Lenguaje de consulta:

Interpretación y ejecución de consultas:

Los motores de bases de datos relacionales usan SQL para consultas y manipulación de datos, mientras que los motores NoSQL pueden usar otros lenguajes o APIs.

Optimización de consultas

Los motores implementan optimizadores que eligen la mejor estrategia para ejecutar una consulta de manera eficiente.

Manejo de datos

Almacenamiento

Los motores deben gestionar cómo se almacenan físicamente los datos en discos y cómo se recuperan de manera eficiente.

Seguridad y autenticación

- Gestión de usuarios y roles
- Cifrado de datos
- Auditoría

Escalabilidad y rendimiento:

- Escalabilidad horizontal y vertical
- Caché
- Replicación
- Particionamiento

Integridad y Consistencia de los datos

- Restricciones y disparadores

Soporte para tipos de datos avanzados:

- Compatibilidad con distintos tipos de datos y funciones avanzadas

Respaldo y recuperación

Backups

Herramientas para crear respaldos periódicos de los datos y restaurarlos en caso de fallas.

Compatibilidad y extensión

Interoperabilidad

Soporte para integrarse con otros sistemas o plataformas a través de APIs, ODBC/JDBC o controladores nativos.

Extensibilidad

Capacidad para agregar nuevas funcionalidades a través de módulos o plugins, como nuevos tipos de datos o funciones personalizadas.

Bases de datos

Bases de datos

Sistema que permite almacenar y organizar información.

Tipos de BD

- Relacionales (RDBMS)
- Bases de Datos NoSQL

RDBMS

Estructura: Los datos se almacenan en tablas organizadas en filas y columnas. Cada fila representa una entidad única, y cada columna representa un atributo de esa entidad.

NoSQL

Estructura: No utilizan tablas. Pueden usar estructuras como documentos, grafos, pares clave-valor o columnas anchas.

RDBMS

Esquema rígido: El esquema (estructura de las tablas) debe definirse de antemano y es rígido. Los datos deben ajustarse a este esquema.

NoSQL

Esquema flexible: No requieren un esquema fijo. Los datos pueden almacenarse en diferentes formatos, y se pueden agregar atributos nuevos sin modificar las estructuras existentes.

RDBMS

Relaciones: Las relaciones entre las tablas se definen mediante claves primarias y foráneas.

NoSQL

Relaciones: No necesariamente tienen relaciones explícitas entre las entidades. Las conexiones entre los datos suelen gestionarse a nivel de la aplicación.

RDBMS

Escalabilidad:
Tradicionalmente se escalan de manera vertical (mejorar hardware del servidor).

NoSQL

Escalabilidad: Se escalan fácilmente de forma horizontal, agregando más servidores o nodos a medida que crecen los datos.

RDBMS

Integridad: Garantizan la consistencia y la integridad de los datos mediante restricciones, como claves únicas, claves primarias y reglas de integridad referencial.

NoSQL

Flexibilidad: Son más adecuadas para almacenar datos no estructurados o semiestructurados, como JSON, XML, y pueden manejar grandes volúmenes de datos con estructuras variables.

RDBMS

Casos de uso: Aplicaciones con datos bien estructurados y relaciones complejas, como sistemas financieros, ERP, y gestión de inventarios.

NoSQL

Casos de uso: Aplicaciones con datos no estructurados o de rápido crecimiento, como aplicaciones web, redes sociales, big data, y sistemas de recomendación.

RDBMS

- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server
- SQLite

NoSQL

- MongoDB
- Cassandra
- Redis
- DynamoDB

SQL

Structured Query Language

Objetos en bases de datos

Esquema

Especifica cómo se organizan los datos dentro de la base de datos.

Tablas

Una tabla es una colección de datos organizados en filas y columnas.
Cada fila representa un registro único, y cada columna representa un atributo del registro.

Claves Primarias

Una columna (o conjunto de columnas) que identifica de manera única cada fila en una tabla. No puede contener valores nulos y debe ser única.

Claves Foráneas

Una columna en una tabla que crea una relación entre esa tabla y otra, apuntando a la clave primaria de la otra tabla.

Vistas

Consultas almacenadas que pueden ser tratadas como tablas virtuales. Se utilizan para simplificar consultas complejas.

Procedimientos almacenados

Bloques de código SQL que se guardan y se ejecutan en el servidor de la base de datos.

Operaciones Fundamentales

Create Read Update Delete

TIPOS DE COMANDOS

DQL DDL DML DCL TCL

DQL

Lenguaje de Consulta de Datos
Data Query Language

```
SELECT nombre, edad FROM estudiantes
WHERE edad > 18;
```

DDL

Lenguaje de Definición de Datos
Data Definition Language

```
CREATE TABLE estudiantes (id INT, nombre
VARCHAR(50), edad INT);

ALTER TABLE estudiantes ADD columna_grado
VARCHAR(10);

DROP TABLE estudiantes;
```

DML

Lenguaje de Manipulación de Datos
Data Manipulation Language

```
INSERT INTO estudiantes (id, nombre,
edad) VALUES (1, 'Juan', 20);

UPDATE estudiantes SET edad = 21 WHERE id
= 1;

DELETE FROM estudiantes WHERE id = 1;
```

DCL

Lenguaje de Control de Datos
Data Control Language

```
GRANT SELECT ON estudiantes TO usuario1;

REVOKE SELECT ON estudiantes FROM
usuario1;
```

TCL

Lenguaje de Control de Transacciones
Transaction Control Language

```
COMMIT;

ROLLBACK;

SAVEPOINT puntoA;
```

Operaciones más comunes

Funciones agregadas

SUM, AVG, COUNT, MIN, MAX.

```sql
SELECT departamento, SUM(salario) AS
total_salario
FROM empleados
WHERE salario > 40000
GROUP BY departamento;
```

```sql
SELECT departamento, COUNT(*) AS
numero_empleados
FROM empleados
WHERE fecha_contratacion > '2020-01-01'
GROUP BY departamento;
```

```
SELECT departamento, AVG(salario) AS
salario_promedio
FROM empleados
GROUP BY departamento
HAVING AVG(salario) > 50000;
```

```
SELECT MAX(salario) AS salario_maximo
FROM empleados
WHERE departamento = 'Ventas';

SELECT departamento, MIN(salario) AS
salario_minimo
FROM empleados
WHERE años_en_empresa > 5
GROUP BY departamento;
```

```
SELECT
     SUM(CASE
          WHEN salario < 50000 THEN salario
          ELSE 0
     END) AS total_junior,
     SUM(CASE
          WHEN salario >= 50000 THEN
salario
          ELSE 0
     END) AS total_senior
FROM empleados;
```

Uniones

INNER JOIN, LEFT JOIN,
RIGHT JOIN, FULL JOIN

Relacionamiento de datos
de múltiples tablas.

```
SELECT e.nombre, d.nombre_departamento
FROM empleados e
INNER JOIN departamentos d ON e.departamento_id =
d.departamento_id;

SELECT e.nombre, d.nombre_departamento
FROM empleados e
LEFT JOIN departamentos d ON e.departamento_id =
d.departamento_id;

SELECT e.nombre, d.nombre_departamento
FROM empleados e
RIGHT JOIN departamentos d ON e.departamento_id =
d.departamento_id;
```

```
SELECT
    c.cliente_id,
    c.nombre,
    SUM(v.cantidad * p.precio) AS total_ventas
FROM
    clientes c
INNER JOIN
    ventas v ON c.cliente_id = v.cliente_id
INNER JOIN
    productos p ON v.producto_id = p.producto_id
WHERE
    YEAR(v.fecha_venta) = 2023
GROUP BY
    c.cliente_id, c.nombre;
```

Condicionales

Permite filtrar la información.

```
SELECT * FROM empleados WHERE salario >
50000;

SELECT * FROM empleados WHERE salario >
50000 AND departamento = 'Ventas';

SELECT * FROM empleados WHERE salario >
50000 OR departamento = 'Ventas';
```

```
SELECT * FROM empleados WHERE NOT
departamento = 'Ventas';

SELECT * FROM empleados WHERE salario
BETWEEN 40000 AND 60000;


SELECT * FROM empleados WHERE
departamento IN ('Ventas', 'Marketing',
'IT');
```

```
SELECT * FROM empleados WHERE nombre LIKE
'J%';

SELECT * FROM empleados WHERE
fecha_termino IS NULL;
```

```
SELECT nombre,
       CASE
           WHEN salario > 50000 THEN
'Alto'
           WHEN salario BETWEEN 30000 AND
50000 THEN 'Medio'
           ELSE 'Bajo'
       END AS rango_salarial
FROM empleados;
```

Modelo entidad relación (ER)

Modelo Entidad Relación

Representación gráfica que se utiliza para modelar la estructura de una base de datos, mostrando cómo los datos están relacionados entre sí.

Componentes

Entidades

Una entidad representa un objeto concreto o conceptual que tiene relevancia para el sistema.

Entidades concretas

Una entidad puede ser un objeto con existencia física como: una persona, un salón, una casa.

Entidades abstractas

Objeto con existencia conceptual como: un puesto de trabajo, una asignatura de clases.

Atributos

Los atributos son las propiedades que describen una entidad.

Atributos Simples

Un atributo que no se puede dividir en más partes. Representa un solo valor. Ejemplo: estatura.

Atributos Compuestos

Un atributo que se puede dividir en componentes más pequeños y lógicos. Ejemplo: dirección.

Atributos Monovalorado (Claves)

Un atributo que tiene un único valor para cada entidad.
Ejemplo: Id.

Atributos Multivalorado

Un atributo que puede tener más de un valor asociado con una entidad.
Ejemplo: correos.

Atributos Derivados

Un atributo cuyo valor se puede calcular o derivar de otros atributos.
Ejemplo: edad.

Estudiante

Ejemplos de atributos incluyen nombre, celular, edad, email para la entidad Estudiante.

idEstudiante (Atributo Clave)

Relaciones

Cómo las entidades interactúan o se asocian entre sí.

Cardinalidad

Indican la forma en que las entidades interactúan entre sí. Pueden ser de varios tipos.

Uno a uno

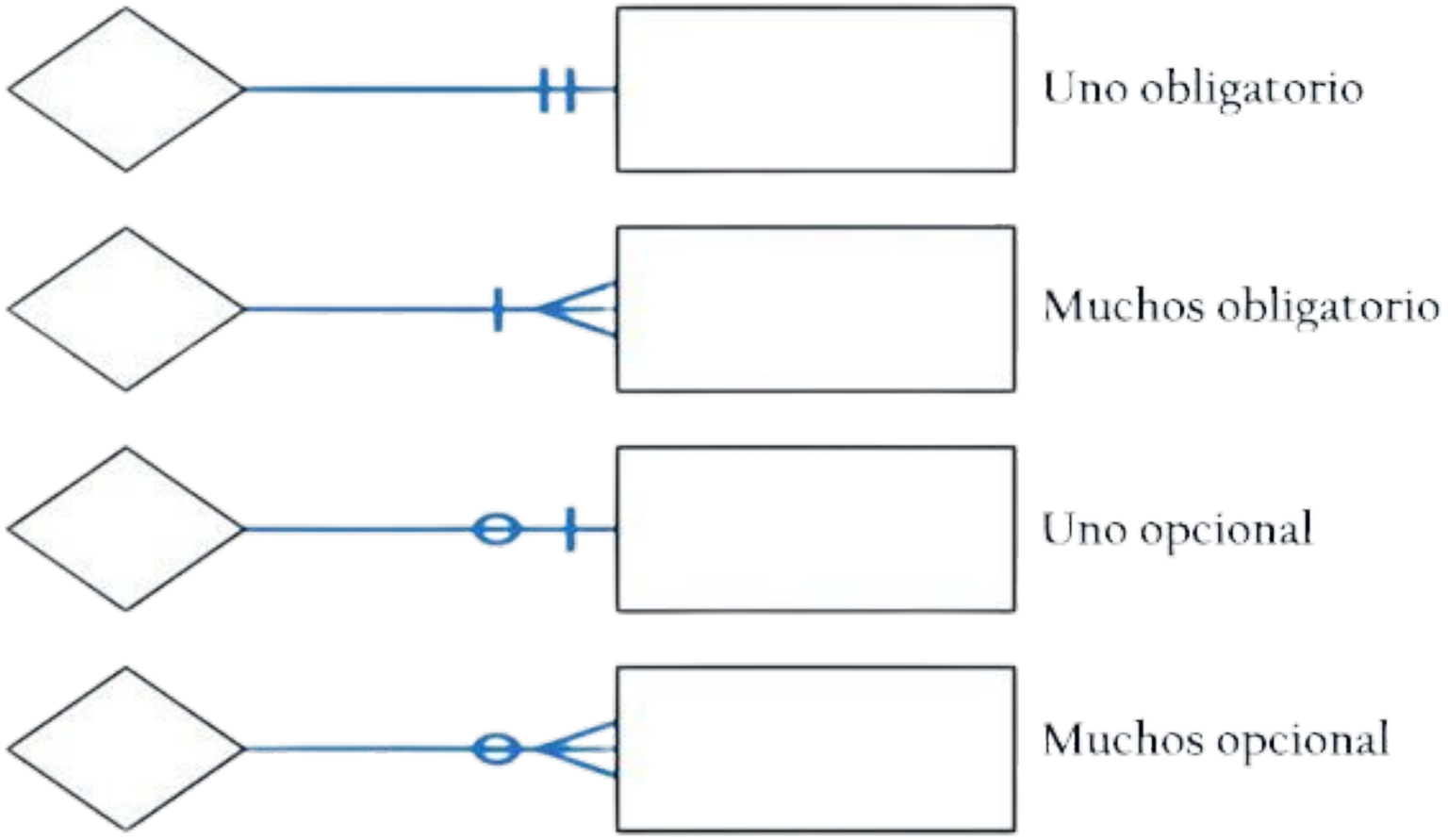
(1:1): Una entidad A está relacionada con una sola entidad B y viceversa.

Uno a muchos
Muchos a uno

(1:*),(1:N): Una entidad A puede estar relacionada con muchas entidades B, pero una entidad B está relacionada solo con una entidad A.

Muchos a muchos

(*:*),(N:M): Una entidad A puede estar relacionada con muchas entidades B y viceversa.


Uno obligatorio
Muchos obligatorio
Uno opcional
Muchos opcional

Normalización

La normalización en bases de datos es el proceso de estructurar una base de datos de manera que minimice la redundancia de datos y garantice la integridad. Se logra dividiendo una tabla grande en tablas más pequeñas y vinculándolas mediante relaciones.

1NF

- Elimina grupos repetitivos.
- Asegúrate de que cada columna contenga valores atómicos (no divisibles).
- Cada registro debe ser único.

Considera la siguiente tabla que almacena información sobre los alumnos y los cursos en los que están inscritos:

StudentID	StudentName	Courses	Professor
1	Carlos	Matemáticas, Física	Prof. López
2	Ana	Química, Biología	Prof. González

Aquí, las columnas Courses y Professor contienen múltiples valores, lo que viola la 1NF.

Alumnos:

StudentID	StudentName
1	Carlos
2	Ana

Matriculas:

StudentID	Course	Professor
1	Matemáticas	Prof. López
1	Física	Prof. López
2	Química	Prof. González
2	Biología	Prof. González

2NF

- Cumple con 1NF.
- Elimina dependencias parciales; es decir, cada atributo no clave debe depender completamente de la clave primaria.

Imagina que la institución también almacena la calificación obtenida por los alumnos en cada curso:

StudentID	Course	Professor	Grade
1	Matemáticas	Prof. López	A
1	Física	Prof. López	B
2	Química	Prof. González	A
2	Biología	Prof. González	B

Aquí, la columna Professor depende solo de Course, no de la clave compuesta (StudentID, Course), lo que viola la 2NF.

Matriculas:

StudentID	Course	Grade
1	Matemáticas	A
1	Física	B
2	Química	A
2	Biología	B

Cursos:

Course	Professor
Matemáticas	Prof. López
Física	Prof. López
Química	Prof. González
Biología	Prof. González

3NF

- Cumple con 2NF.
- Elimina dependencias transitivas; es decir, los atributos no clave no deben depender de otros atributos no clave.

Supongamos que tenemos una tabla que incluye la dirección del profesor:

Course	Professor	ProfessorAddress
Matemáticas	Prof. López	Calle Falsa 123
Física	Prof. López	Calle Falsa 123
Química	Prof. González	Avenida Siempreviva 742
Biología	Prof. González	Avenida Siempreviva 742

Course	Professor
Matemáticas	Prof. López
Física	Prof. López
Química	Prof. González
Biología	Prof. González

Professor	ProfessorAddress
Prof. López	Calle Falsa 123
Prof. González	Avenida Siempreviva 742

BCNF
Boyce-Codd

- Cumple con 3NF.
- Cada determinante debe ser una clave candidata.

Course	Professor	Classroom
Matemáticas	Prof. López	A101
Física	Prof. López	A101
Química	Prof. González	B202
Biología	Prof. González	B202

Course	Professor
Matemáticas	Prof. López
Física	Prof. López
Química	Prof. González
Biología	Prof. González

Classroom	Course
A101	Matemáticas
A101	Física
B202	Química
B202	Biología

4NF

- Cumple con BCNF.
- No debe haber dependencias multivaluadas.

Course	Professor	Classroom
Matemáticas	Prof. López	A101
Matemáticas	Prof. Pérez	A101
Química	Prof. González	B202
Química	Prof. Sánchez	B202

Course
Matemáticas
Química

Course	Professor
Matemáticas	Prof. López
Matemáticas	Prof. Pérez
Química	Prof. González
Química	Prof. Sánchez

Course	Classroom
Matemáticas	A101
Química	B202

5NF

- Cumple con 4NF.
- Divide la tabla en tablas más pequeñas cuando sea posible, de modo que la unión de estas tablas reproduzca la tabla original.

Ejemplo:
Supongamos que tenemos relaciones complejas entre cursos, profesores y salones, y que las asociaciones entre ellos pueden ser muchas a muchas.

Solución:
Necesitamos asegurarnos de que todas las posibles combinaciones de cursos, profesores y salones puedan reconstruirse sin pérdida de información a partir de sus relaciones más simples:

Course
Matemáticas
Química

Course	Professor
Matemáticas	Prof. López
Matemáticas	Prof. Pérez
Química	Prof. González
Química	Prof. Sánchez

Course	Classroom
Matemáticas	A101
Química	B202

Professor	Classroom
Prof. López	A101
Prof. Pérez	