Prof. Paula Ithurralde.
Prof. Miguel Silva.

BASES DE DATOS

Las bases de datos son sistemas de almacenamiento y organización de datos que permiten gestionar y acceder de manera eficiente a información estructurada. En términos más simples, son como "almacenes" digitales que almacenan datos de manera organizada y que pueden ser consultados, actualizados y gestionados de diversas formas. Las bases de datos son fundamentales en la informática y la gestión de información, y se utilizan en una amplia variedad de aplicaciones y sistemas.

En términos generales, una base de datos es un conjunto de datos estructurados que pertenecen a un mismo contexto y, en cuanto a su función, se utiliza para administrar de forma electrónica grandes cantidades de información.

Existen varios tipos de bases de datos, y se pueden clasificar en función de su estructura, su forma de acceso y su finalidad. Aquí tienes una descripción de algunos de los tipos más comunes:

1. **Bases de Datos Relacionales (SQL):** Son el tipo más común y ampliamente utilizado. Los datos se organizan en tablas con filas y columnas, y se utiliza un lenguaje de consulta estructurado (SQL) para acceder y gestionar los datos. Ejemplos de sistemas de gestión de bases de datos relacionales (SGBDR) incluyen MySQL, PostgreSQL, Microsoft SQL Server y Oracle.
2. **Bases de Datos NoSQL:** Están diseñadas para gestionar datos no estructurados o semiestructurados, y son altamente escalables. Incluyen bases de datos de documentos, bases de datos de grafos, bases de datos clave-valor y bases de datos de columnas. Se utiliza la estructura JSON para guardar los datos. Ejemplos de bases de datos NoSQL incluyen MongoDB, Cassandra y Neo4j.

CRUD

CRUD es un concepto fundamental en la gestión de bases de datos y aplicaciones, ya que engloba las operaciones más esenciales para interactuar con los datos. La mayoría de las aplicaciones y sistemas de bases de datos se basan en estas operaciones CRUD para gestionar la información de manera eficaz y segura. Cada operación se traduce a comandos SQL específicos en un sistema de gestión de bases de datos, y es fundamental para mantener la integridad y la consistencia de los datos.

El concepto de CRUD (Crear, Leer, Actualizar y Eliminar) se originó en el contexto de bases de datos relacionales, donde se utilizan estructuras tabulares y el lenguaje SQL para interactuar con los datos. No obstante, en las bases de datos NoSQL, que no siguen el modelo relacional, las operaciones pueden variar y no encajar directamente en el marco de CRUD. Aunque las bases de datos NoSQL tienen sus propias operaciones y estructuras, la esencia subyacente de CRUD sigue siendo aplicable a muchas de ellas.

CRUD es un acrónimo que se utiliza en el contexto de bases de datos y aplicaciones informáticas para describir las cuatro operaciones básicas que se pueden realizar en los datos almacenados en una base de datos. Cada letra de "CRUD" representa una de estas operaciones:

1. **CREATE (Crear):** La operación de "Crear" implica la inserción de nuevos datos en una base de datos. En un sistema de gestión de bases de datos (DBMS), esto se realiza generalmente mediante la sentencia SQL INSERT. Por ejemplo, puedes utilizar "Create" para agregar un nuevo registro a una tabla de empleados o clientes.
2. **READ (Leer):** La operación de "Leer" se utiliza para recuperar datos de la base de datos. Esto se logra mediante la sentencia SQL SELECT. La operación "Read" te permite recuperar registros específicos, realizar consultas y obtener datos según tus necesidades. Por ejemplo, puedes usar "Read" para mostrar la lista de empleados de una empresa.
3. **UPDATE (Actualizar):** La operación de "Actualizar" se refiere a la modificación de datos existentes en la base de datos. Esto se realiza con la sentencia SQL UPDATE. "Update" te permite modificar registros, cambiar valores en columnas específicas o realizar cambios en los datos almacenados. Por ejemplo, puedes usar "Update" para cambiar la dirección de un cliente o la descripción de un producto.
4. **DELETE (Eliminar):** La operación de "Eliminar" se usa para eliminar registros o datos de la base de datos. Esto se realiza con la sentencia SQL DELETE. "Delete" permite borrar registros o información obsoleta de la base de datos. Por ejemplo, puedes utilizar "Delete" para eliminar a un empleado que ya no trabaja en la empresa.

BASES DE DATOS RELACIONALES

Las bases de datos relacionales son ampliamente utilizadas debido a su estructura tabular y la capacidad de establecer relaciones entre los datos. En ellas, los datos se organizan en tablas, y cada tabla se relaciona con otras mediante claves primarias y foráneas.

FORMAS NORMALES (FN)

Las formas normales son reglas que se aplican en el diseño de bases de datos relacionales para garantizar la integridad y la eficiencia de la estructura de la base de datos. Estas reglas se utilizan para organizar la información de manera coherente y evitar problemas como la redundancia de datos y las anomalías en la actualización. Existen varias formas normales, siendo las más comunes:

Primera Forma Normal (1FN): En esta forma, se garantiza que cada columna de una tabla contiene valores atómicos (indivisibles) y que cada fila de la tabla es única. No debe haber grupos de valores repetidos o listas separadas por comas en una columna.

Segunda Forma Normal (2FN): Además de cumplir con 1FN, en 2FN se elimina la dependencia parcial. Esto significa que si una tabla tiene una clave primaria compuesta, cada columna no clave debe depender de toda la clave primaria, no solo de una parte de ella.

Tercera Forma Normal (3FN): Además de cumplir con 2FN, en 3FN se eliminan las dependencias transitivas. Esto implica que si una columna no clave depende de otra columna no clave, debe eliminarse y colocarse en otra tabla.

Existen otras formas normales, pero su existencia indica que deben crearse otras tablas para su correcta organización y manipulación:

Forma normal de Boyce-Codd (FNBC): La tabla se encuentra en FNBC si cada determinante, atributo que determina completamente a otro, es clave candidata. Deberá registrarse de forma anillada ante la presencia de un intervalo seguido de una formalización perpetua, es decir las variantes creadas, en una tabla no se llegarán a mostrar, si las ya planificadas, dejan de existir.

Cuarta Forma Normal (4FN): En 4FN se eliminan las dependencias multivaluadas, lo que significa que no puede haber conjuntos de datos relacionados que se puedan dividir en filas separadas. Si esto ocurre, se deben crear tablas adicionales.

Quinta Forma Normal (5FN): 5FN se refiere a las dependencias de unión. Esto implica que, si una tabla se puede descomponer en múltiples tablas sin perder información, entonces es necesario dividirla en varias tablas.

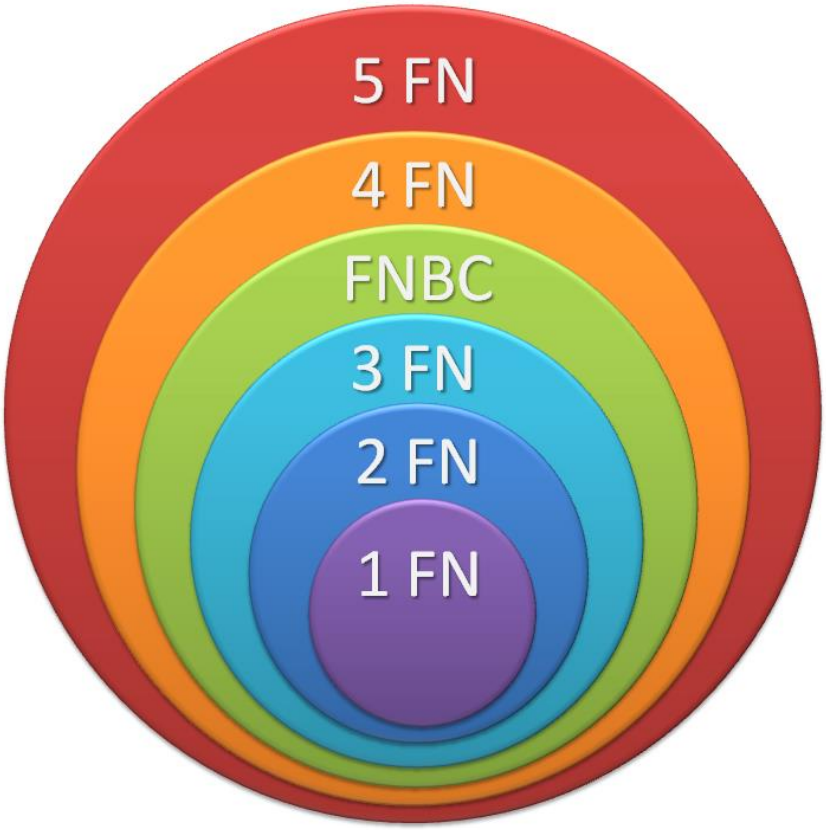

Diagrama de inclusión de todas las formas normales.

ESTRUCTURA BÁSICA DE UNA TABLA DE UNA BASE DE DATOS SQL

La estructura básica de una tabla en una base de datos SQL consta de varios componentes que definen cómo se almacenan y organizan los datos en esa tabla. A continuación, se presentan los elementos clave de la estructura de una tabla:

- **Nombre de la tabla:** Es el nombre único que se le da a la tabla. Debe seguir las reglas de nomenclatura, como no contener espacios ni caracteres especiales, y generalmente se utiliza en singular (por ejemplo, "Empleado" en lugar de "Empleados").
- **Columnas o campos:** Cada tabla consta de una serie de columnas que representan los atributos o propiedades de los datos que se almacenan. Cada columna tiene un nombre único y un tipo de datos que especifica qué tipo de información se almacenará en esa columna (por ejemplo, VARCHAR para texto, INT para enteros, DATE para fechas).
- **Clave primaria (Primary Key):** Es un campo o una combinación de campos que se utiliza para identificar de manera única cada fila en la tabla. La clave primaria garantiza que no haya duplicados y permite una rápida búsqueda y recuperación de datos. Generalmente, se denota con el atributo PRIMARY KEY y se puede comprender una o más columnas.
- **Clave foránea (Foreign Key):** Es un campo que establece una relación entre dos tablas. La clave foránea se usa para vincular registros en la tabla actual con registros en otra tabla. Ayuda a mantener la integridad referencial de la base de datos. Se define con el atributo FOREIGN KEY.
- **Restricciones (Constraints):** Las restricciones son reglas que se aplican a las columnas para garantizar la integridad de los datos. Algunas restricciones comunes son NOT NULL (para garantizar que un campo no esté vacío), UNIQUE (para garantizar que los valores sean únicos) y CHECK (para imponer condiciones específicas en los valores de una columna).
- **Otros atributos:** Además de los elementos anteriores, una tabla puede tener otros atributos, como restricciones adicionales, índices (para mejorar el rendimiento de las consultas), comentarios, etc.

MySQL Workbench

MySQL es un sistema de gestión de bases de datos relacional ampliamente utilizado. MySQL Workbench es una herramienta gráfica que facilita la administración y el desarrollo de bases de datos MySQL. A continuación, te explicaremos cómo instalar MySQL Workbench y crear una base de datos para un colegio como ejemplo.

Instalación de MySQL Workbench

Descarga MySQL Workbench desde el sitio web oficial:

https://dev.mysql.com/downloads/workbench/

Sigue las instrucciones de instalación proporcionadas para tu sistema operativo.

Deberá hacer la instalación “FULL”.

Durante la instalación, se te pedirá configurar una contraseña para el usuario "root" de MySQL. Asegúrate de recordar esta contraseña, ya que la necesitarás para conectarte a MySQL.

Ejemplo de Base de Datos para un Colegio

En este ejemplo, crearemos una base de datos para un colegio con 4 tablas: Alumnos, Cursos, Profesores y Materias. También definiremos las relaciones entre estas tablas.

Creación de la Base de Datos

Abre MySQL Workbench y conéctate a tu servidor MySQL utilizando la contraseña que configuraste durante la instalación.

Creación de la Base de Datos:

```
-- Crear la base de datos
CREATE DATABASE IF NOT EXISTS escuela;
-- Seleccionar la base de datos
USE escuela;
# BORRAR DB Y TABLAS:
#ÚSENLO CON MUCHA PRECAUCIÓN.
DROP DATABASE nombre_DB;
```

Creación de las Tablas

Tabla alumnos:

```
-- Crear la tabla Alumnos
CREATE TABLE Alumnos (
   DNI INT PRIMARY KEY,
   nombre VARCHAR(255),
   apellido VARCHAR(255),
   curso VARCHAR(10)
);
```

En esta tabla, hemos creado una columna DNI como clave primaria para identificar de manera única a cada alumno. La columna Curso servirá como clave foránea para establecer una relación con la tabla Cursos.

Tabla Cursos:

```
-- Crear la tabla Cursos
CREATE TABLE Cursos (
   idCurso INT AUTO_INCREMENT PRIMARY KEY,
   Curso INT
);
```

Hemos creado una columna idCurso como clave primaria para identificar los cursos de manera única y autoincremental.

Tabla Profesores:

```
-- Crear la tabla Profesores
CREATE TABLE Profesores (
   DNI INT PRIMARY KEY,
   Nombre VARCHAR(255),
   Apellido VARCHAR(255),
   Materia INT
);
```

Al igual que en la tabla alumnos, hemos creado una columna DNI como clave primaria para identificar de manera única a cada profesor. La columna Materia servirá como clave foránea para establecer una relación con la tabla Materias.

Tabla Materias:

```
-- Crear la tabla Materias
CREATE TABLE Materias (
   idMateria INT PRIMARY KEY,
   nombreMateria VARCHAR(255)
);
```

La columna IDMateria es la clave primaria para identificar las materias de manera única.

Modificación de las Columnas

Para modificar una columna existente en una tabla, puedes utilizar la sentencia **ALTER TABLE**. Por ejemplo, si deseas agregar una columna "Edad" a la tabla Alumnos, puedes hacerlo de la siguiente manera:

```
ALTER TABLE Alumnos
ADD Edad INT;
```

A continuación, haremos una modificación sobre la tabla Alumnos para que sea apropiada la creación de la llave foránea:

```
-- Modificar columna curso en tabla Alumnos:
ALTER TABLE Alumnos
MODIFY curso INT;
```

Creación de llaves foráneas:

```
-- Agregar FK a tabla Alumnos:
ALTER TABLE alumnos
ADD CONSTRAINT FK_Alumnos_curso
FOREIGN KEY (curso)
REFERENCES cursos(idCurso);
```

```
-- Agregar FK a tabla Profesores:
ALTER TABLE Profesores
```

```
ADD CONSTRAINT FK_Profesores_Materia
FOREIGN KEY (materia)
REFERENCES Materias(idMateria);
```

Ingreso de Datos en las Tablas

Puedes ingresar datos en las tablas utilizando la sentencia INSERT INTO. Aquí tienes un ejemplo para agregar un alumno a la tabla Alumnos:

```
INSERT INTO cursos (curso)
VALUES
    ('1º año'),
    ('2º año'),
    ('3º año'),
    ('4º año'),
    ('5º año');
```

```
INSERT INTO Alumnos (DNI, nombre, apellido, curso)
VALUES
    (1234578, 'Juana','Perez',1),
   (2345689, 'Luis','González',1),
   (3456790, 'Ana','Rodríguez',1),
   (4567801, 'Pedro','Martínez',2),
   (5678912, 'Sofía','López',2),
   (6789023, 'Diego','Fernández',2),
   (7890134, 'María','Torres',3),
   (8901245, 'Carlos','Sanchez',3),
   (9012356, 'Laura','Gómez',3),
   (9876532, 'Andrés','Pérez',4),
   (8765421, 'Luisa','Ramírez',4),
   (7654310, 'Javier','García',4),
   (5432198, 'Marta','Díaz',5),
```

```
(4321087, 'Joaquín','Mendoza',5),
(3210976, 'Valentina','Hernández',5);
```

```
INSERT INTO Materias (idMateria, nombreMateria)
VALUES
    (1, 'Matemáticas'),
(2, 'Historia'),
(3, 'Ciencias'),
(4, 'Literatura'),
(5, 'Inglés'),
(6, 'Geografía'),
(7, 'Química'),
(8, 'Biología'),
(9, 'Música'),
(10, 'Educación Física');
```

```
INSERT INTO Profesores (DNI, nombre, apellido, materia)
VALUES
    (12345678, 'Carlos', 'Chávez', 1),
(23456789, 'Laura', 'Gómez', 2),
(34567890, 'Javier', 'Hernández', 3),
(45678901, 'Sofía', 'Martínez', 4),
(56789012, 'Andrés', 'Pérez', 5),
(67890123, 'Valentina', 'Díaz', 6),
(78901234, 'Luis', 'Ramírez', 7),
(89012345, 'María', 'Sanchez', 8),
(90123456, 'Diego', 'García', 9),
(98765432, 'Joaquín', 'Mendoza', 10);
```

Búsquedas

Las búsquedas de información en MySQL se realizan con el comando **SELECT**, y existen diversas combinaciones que se pueden realizar, así como combinar búsquedas en distintas tablas.

Aquí algunos ejemplos:

```
SELECT * FROM Alumnos WHERE curso=5;


SELECT * FROM Alumnos WHERE curso=1 OR curso=5;


SELECT apellido, nombre FROM Profesores;


SELECT * FROM Profesores;


SELECT apellido, nombre FROM Profesores WHERE materia=1 OR materia=2;


SELECT Profesores.nombre, Profesores.apellido, Materias.nombreMateria AS materia
FROM Profesores
INNER JOIN Materias ON Profesores.materia = Materias.idMateria;
```

Actualizaciones

Para actualizar un registro en MySQL, se utiliza la sentencia **UPDATE**. A continuación, te mostraré un ejemplo simple de cómo usar UPDATE para actualizar un registro en una tabla.

```
-- Sentencia UPDATE para actualizar el salario de un empleado específico
UPDATE Profesores
SET nombre = 'Luis Angel'
WHERE DNI = 7;
```