Base de datos

¿Qué son las bases de datos

Las bases de datos son sistemas estructurados que permiten almacenar, gestionar y acceder a grandes volúmenes de datos de manera eficiente.

¿Por qué debemos estudiarlas?

Su estudio es fundamental en el ámbito informático y la tecnología, ya que las bases de datos son una parte clave de la mayoría de las aplicaciones y sistemas informáticos modernos.

Uso en ING de software

En ING Software los sistemas de gestión de bases de datos sirven para almacenar y gestionar datos de forma estructurada, lo que facilita el acceso a la información y su manipulación

Diferencia entre base de datos y sistema de archivos

La principal diferencia son

- Base de datos:
    - Se centran en datos estructurados y organizados para facilitar consultas, análisis e informes
    - Es un conjunto de datos estructurados que se almacenan y organizan de manera específica para facilitar su consulta y manipulación

- Sistema de archivos
    - Gestionan archivos de forma más genérica, priorizando el almacenamiento y la recuperación de datos
    - Organiza y gestiona la información en un medio de almacenamiento, como un disco duro.
    - Su principal función es permitir que el sistema operativo y las aplicaciones accedan a los archivos de manera eficiente, a través de una estructura jerárquica.

Sistema gestor de bases de datos

Es una aplicación software que sirve para interactuar con una base de datos de manera eficiente y segura. Proporcionando una interfaz para crear,modificar, almacenar y recuperar datos de la base de datos.

Además gestiona aspectos como la integridad de los datos, la seguridad, la concurrencia y la recuperación en caso de fallos.

También se encarga de gestionar eficientemente la base de datos, entre las que se encuentran su almacenamiento, la creacion y modificacion de esquemas y modelos de datos, la inserción, actualización, recuperación y eliminación de datos, el mantenimiento de la integridad de la información, control de procesos y transacciones o la realización de copias de seguridad y recuperación frente a fallos.

Niveles de abstracción en la gestión de los datos

- **Nivel físico:** En este nivel se definen detalles de la estructura de los archivos, los métodos de acceso a los datos y la organización
- **Nivel lógico**: Se describe la estructura lógica de la base de datos en términos de registros dependiendo del tipo de base de datos.
- **Nivel de vista:** Permiten ocultar ciertos datos o presentarlos de manera diferente según las necesidades de los usuarios

Definición y manipulación de los datos

- **DDL**: Se pueden definir la estructura de la base de datos, es decir: crear, modificar o eliminar tablas, índices, vistas, procedimientos almacenados, etc.
    - Create table, alter table, drop table, etc
- **DML**: Se realizan operaciones de manipulación de datos, insertar, actualizar, eliminar y recuperar datos en una base de datos
    - Select, insert, update, etc

Tipo de sistemas de gestión de bases de datos

- **Bases de datos relacionales**: Se organizan los datos en tablas con filas (tuplas) y columnas (atributos). Las relaciones entre tablas se definen mediante campos compartidos por dichas tablas
    - **Modelo de datos**: Los datos se organizan en tablas con columnas predefinidas y relaciones entre ellas definidas por claves foraneas. Los datos se almacenan en filas
    - **Lenguajes**: Utilizan SQL como DDL y DML
    - Escalabilidad: Suelen utilizar un escalado vertical añadir más recursos a un servidor para mejorar el rendimiento
    - **Flexibilidad**: Tienen reglas estrictas de integridad de datos y requieren esquemas predefinidos
    - **Grado de cumplimiento de las propiedades ACID**: Son estrictos con el cumplimiento de las propiedades ACID, lo que garantiza la coherencia de los datos y la integridad de las transacciones.
        - **Atomicidad**: Una transacción es atómica si se trata como una unidad de trabajo indivisible. Esto significa que, o bien todas las operaciones de la transacción se completan con éxito, o bien no se completa ninguna.
        - **Consistencia**: Una transacción mantiene la consistencia si transforma la base de datos de un estado válido a otro estado válido. Esto significa que la base de datos debe estar en un

estado coherente tanto antes como después de que se ejecute cada transacción.

        - **Aislamiento:** Una transacción se considera que cumple la propiedad de aislamiento si se ejecuta aislada de otras transacciones. Esto significa que los cambios realizados por una transacción no deben ser visibles para otras transacciones hasta que se éstos se completen.
        - **Durabilidad**: Una vez completada la transacción, sus cambios deben ser permanentes y sobrevivir a cualquier fallo posterior del sistema.

- **Bases de datos no relacionales:** Se caracterizan por su no adhesión al modelo relacional, las bases de datos NoSQL no se construyen principalmente sobre tablas. Están diseñadas para manejar grandes volúmenes de datos no estructurados o semiestructurados y las cuales pueden ser clave-valor, los documentos, los grafos o los objetos
    - **Modelo de datos**: Utilizan otras estructuras de datos como colecciones de documentos, grafos, objetos o simples pares clave-valor.
    - **Lenguajes**: Pueden tener o no un lenguaje de definición y/o manipulacion, pero suelen utilizar interfaces del programador o lenguajes específicos para acceder a los datos
    - **Escalabilidad**: Horizontal es decir pueden distribuirse en varios servidores
    - **Flexibilidad**: Son más flexibles y pueden manejar datos no estructurados o semiestructurados
    - **Grado de cumplimiento de las propiedades ACID**: Sacrifican en alguna medida las propiedades ACID o alguna de dichas propiedades.
        - **Disponibilidad básica:** Se garantiza que se genere una respuesta para cada petición a la base de datos
        - **Estado laxo**: El estado del sistema puede cambiar a lo largo del tiempo, incluso aunque no se haga ninguna petición

        - **Consistencia en último término**: La base de datos podría estar en algún momento en un estado inconsistente, pero se garantiza que tarde o temprano la base de datos pasará a un estado consistente.

Tipo de bases de datos noSQL:

- **Bases de datos de clave-valor:**
    - Almacenan datos en pares de clave-valor, donde la clave es un identificador único para recuperar el valor. Ejemplos: Redis, DynamoDB.
- **Bases de datos documentales:**
    - Almacenan datos en documentos, generalmente en formato JSON o XML, donde cada documento puede tener su propio esquema. Ejemplos: MongoDB, CouchDB.
- **Bases de datos orientadas a columnas:**
    - Almacenan datos en columnas, donde cada columna puede tener diferentes tipos de datos. Ejemplos: Cassandra, BigTable.
- **Bases de datos de grafos:**
    - Almacenan datos como nodos y aristas, representando relaciones entre ellos. Ejemplos: Neo4j.
- **Bases de datos en memoria:**
    - Almacenan datos en la memoria RAM, lo que permite un acceso muy rápido. Ejemplos: Redis.
- **Bases de datos multimodelo:**
    - Pueden manejar diferentes tipos de modelos de datos, como documentales, de clave-valor, de grafos, etc., en una misma base de datos.

Algebra relacional

El álgebra relacional es un conjunto de operaciones que describen paso a paso cómo computar una respuesta sobre las relaciones, tal y como éstas son definidas

en el modelo relacional. Denominada de tipo procedimental, a diferencia del cálculo relacional que es de tipo declarativo.

Describe el aspecto de la manipulación de datos. Estas operaciones se usan como una representación intermedia de una consulta a una base de datos y, debido a sus propiedades algebraicas, sirven para obtener una versión más optimizada y eficiente de dicha consulta.

Tuplas

Una tupla se define como una función finita que asocia unívocamente los nombres de los campos de una relación con los valores de una instanciación de la misma. En términos simples, es una fila de una tabla relacional.

Columnas

atributos: columnas

Union compatible

Una unión es compatible entre dos relaciones R, S, si ellas poseen el mismo grado y el dominio

Cardinalidad: se refiere al número de valores únicos en una columna de una tabla en relación con el número total de filas de la tabla

Grado: se refiere al número de atributos (columnas) que contiene

Aridad: número de atributos (columnas) que tiene una relación (tabla).

Tipo de operaciones

- **Operaciones básicas:** Cada operador del álgebra acepta una o dos relaciones y retorna una relación como resultado. σ y Π son operadores unarios, el resto de los operadores son binarios.:
    - **operaciones unarias**: trabajan en una relación
        - **Selección o restricción (σ):** permite seleccionar un subconjunto de tuplas de una relación todas aquellas que cumplan la(s) condición(es). **Select * from (nombre tabla o relación) where (condición)**
        - **Proyección (Π):** Permite extraer columnas (atributos) de una relación. **Select * from (nombre tabla o relación)**
    - **operaciones binarias:** trabajan en dos relaciones
        - **Producto cartesiano (x):** Muestra una nueva relación, cuyo esquema contiene cada una de las tuplas de las relaciones. **SELECT CUITS.Cuit_pais, CUITS.Pais, Personas.Persona FROM CUITS JOIN Personas**
        - **Unión (∪):** es una operación que combina dos relaciones (o tablas) para crear una nueva relación que contiene todos los registros de ambas relaciones originales. **SELECT * FROM (relación) UNION(o UNION ALL) SELECT * FROM (relacion);**
        - **Diferencia (-):** La diferencia en álgebra relacional (también conocida como diferencia de conjuntos) se utiliza para obtener las tuplas que están en una relación y no en la otra. **SELECT id, nombre FROM clientes_2024 WHERE (id, nombre) NOT IN (SELECT id, nombre FROM clientes_2025);**
- **Operaciones No básicas o derivadas:** son aquellas que no forman parte del conjunto fundamental de operaciones, pero que pueden expresarse usando combinaciones de las operaciones básicas.
    - **operaciones binarias:** trabajan en dos relaciones
        - **Intersección (∩):** devuelve las tuplas que están en ambas relaciones. **SELECT id, nombre FROM empleados_oficina1**

INTERSECT SELECT id, nombre FROM empleados_oficina2;

- **Unión natural (⋈) (Natural Join):** es una forma de unir dos tablas automáticamente usando las columnas con el mismo nombre en ambas tablas. **SELECT CUITS.Cuit_pais, CUITS.Pais, Personas.Persona FROM CUITS JOIN Personas ON CUITS.Tipo_sujeto = Personas.Sujeto;**
- **División (÷):** es una de las operaciones más potentes y conceptualmente complejas. **SELECT estudiante**
- **FROM estudiantes_cursos GROUP BY estudiante HAVING COUNT(DISTINCT curso) = (SELECT COUNT(*) FROM cursos);**
- **Agrupación (g)(UNION):** en álgebra relacional y su equivalente en MySQL, que es una de las partes más prácticas y comunes en bases de datos. **SELECT id_departamento, COUNT(*) AS total_empleados FROM empleados GROUP BY id_departamento;**

Comandos hasta ahora

(Los comandos deberian ir en MAYÚSCULA como Select y las tablas o atributos en minúscula pero meh)

Entrar en mysql

- sudo mysql
- mysql -u root -p
- mysql -h localhost -u root -p

Crear/dar privilegios/ cambiar contraseña/Verr privilegios/revocar privilegios

Crear usuario:

- CREATE USER "username"@"localhost" IDENTIFIED BY " passwordusser";

Dar privilegios:

- Totales
    - GRANT ALL PRIVILEGES ON *.* TO "ussername"@"localhost" WITH GRANT OPTION;
- A una base de datos especifica:
    - **GRANT ALL PRIVILEGES ON exampledb.* TO ‘newuser’@’localhost’;**
- Algunos privilegios en una base de datos especifica
    - **GRANT SELECT, INSERT, UPDATE ON exampledb.* TO ‘newuser’@’localhost’;**
- Privilegios en tablas:
    - **GRANT SELECT, INSERT ON exampledb.exampletable TO ‘newuser’@’localhost’;**

Cambiar contraseña:

- ALTER USER "ussername"@"localhost" IDENTIFIED WITH caching_sha2_password BY "newpassword";

Ver privilegios

- **SHOW GRANTS FOR ‘newuser’@’localhost’;**

Revocar privilegios

- **REVOKE INSERT ON exampledb.* FROM ‘newuser’@’localhost’;**

Comprobar estado

status

mostrar base de datos

show database

Usar base de datos

Use (nombre de base de dato)

Crear base de datos

Create database (nombre de la base de datos)

Borrar base de datos

Drop (nombre de la base de datos) (creo jaajjaja)

Mostrar tablas

SHOW TABLES

Crear tabla

Create table (nombre de la tabla) (atributos a definir)

ejemplo: **CREATE TABLE Empleados (IDE INT PRIMARY KEY, Nombre VARCHAR(50), Apellido VARCHAR(50), Email VARCHAR(100), Departamento INT, Salario DECIMAL(10, 2));**

Crear tabla con clave primaria compuesta: CREATE TABLE MiTabla (columna1 INT, columna2 VARCHAR(255), columna3 INT, PRIMARY KEY (columna1, columna2)

Describir tabla

DESCRIBE (nombre de la tabla)

Formas de borrar tablas

DROP (nombre tabla)

TRUNCATE (nombre tabla)

Modificar estructura de una tabla

ALTER TABLE

- Añadir clave ajena

    - ALTER TABLE Empleados **ADD FOREIGN KEY** (Departamento) **REFERENCES** Departamentos(IDDepartamento);
- Añadir clave primaria compuesta:
    - ALTER TABLE MiTabla **ADD PRIMARY KEY** (columna1, columna2);
- Modificar valor(creo) de una columna
    - ALTER TABLE CUITS MODIFY COLUMN Pais VARCHAR(100);

Proyectar tabla

SELECT * o (atributos a mostrar) FROM (nombre tabla)

Selección

SELECT * FROM (nombre tabla) WHERE (condicion)

- borrar registros duplicados de una visualizacion select **distinct** color, ciudad from COMPONENTES;
- SELECT **CONCAT**(apellido, departamento) FROM Empleados;
- para darle un **Alias** se usa: mysql> SELECT CONCAT(apellido, " ", departamento) **AS** Empleados FROM Empleados;

ejemplos de uso:

- si se sabe el nombre de la columna se pone mysql> SELECT nombre FROM Empleados;
- para mostrar mas de una columna se usa mysql> SELECT ide, nombre FROM Empleados;
- para ordenar (filtrar) usar mysql> SELECT * FROM Empleados **ORDER BY** salario;
- visualizar datos especificos coincidentes SELECT * FROM ARTICULOS WHERE ciudad **=** "caceres";
- buscar con una letra específica o palabra select ida, ciudad from ARTICULOS where ciudad **LIKE** "d%" (inicia) or ciudad **LIKE** "%e%"(contiene);
    - **%** Es un comodín que representa cualquier secuencia de caracteres, incluyendo cero caracteres, en una consulta SQL.
- concat + alias + like:
    - select concat(ida, "-", ciudad) as "ida-ciudad" from ARTICULOS where ciudad like "d % e%";
- Natural join:
    - select idc, ciudad from ENVIOS **natural join** ARTICULOS where ciudad="madrid";
- Ordenar el resultado del natural join:

    - select Anombre from ARTICULOS natural join ENVIOS where idp=1 order by Anombre;

Insertar tuplas en una relacion o tabla

INSERT INTO (nombre tabla) VALUES ()

Ejemplo:

- INSERT INTO Empleados VALUES (1, "Pepito", "Perez", "pepitoperez@gmail.com", 10, 300000);
- Insertar más de una tupla a la vez:
    - INSERT INTO ENVIOS VALUES (1, 1, 1, 200), (1, 1, 4, 700), (2, 3, 1 ,400), (2, 3, 2, 200), (2, 3, 3, 200), (2, 3, 4, 500), (2, 3, 5, 600), (2, 3, 6, 400), (2, 3, 7, 800), (2, 5, 2, 100), (3, 3, 1, 200), (3, 4, 2, 500), (4, 6, 3, 300), (4, 6, 7, 300), (5, 2, 2, 200), (5, 2, 4, 100), (5, 5, 4, 500), (5, 5, 7, 100),(5, 6, 2, 200), (5, 1, 4, 100), (5, 3, 4, 200), (5, 4, 4, 800), (5, 5, 5, 400), (5, 6, 4, 500);

Para eliminar un registro

DELETE FROM (nombre tabla)

Ejemplos:

- usar mysql> **DELETE** FROM Empleados WHERE IDE **=** 3;
- para eliminar más de un empleado usar mysql> DELETE FROM Empleados WHERE IDE **=** 1 **OR** IDE **=** 2 **OR** IDE **=** 4;
- o usar mysql> DELETE FROM Empleados WHERE IDE = 1 **<>** 8;
- o mysql> DELETE FROM Empleados WHERE IDE **<** 5;

Para modificar registros usar: los () no van

- para modificar todos
    - UPDATE (nombre tabla) SET (nombre atributo) = (nuevo dato)
- Para modificar un registro:
    - UPDATE Empleados SET email = "jsmith69@gmail.com" **WHERE** IDE = 8;

Cargar archivos csv a una tabla

LOAD DATA INFILE (ubicación del archivo)[NOMBRE ARCHIVO]" replace INTO TABIE CUITS FIELDS TERMINATED BY "," LINES TERMINATED BY "/n";

Respaldar datos

- desde la shell de comandos escribir el siguiente comando: **mysql -u root (o nombre del usuario) -p "base de datos" < "archivo.sql"**
- dentro de mysql: **source "archivo.sql"**