

11 de marzo, 2024

## Partes de una aplicación

### Interfaz

Comunicación de la app con el exterior. No necesariamente interactúa con una persona, puede ser otro sistema.  
Envía y recibe información

### Núcleo.

Core de la aplicación.

Para lo que fue hecho el sistema

Todos los procesos que modifican datos o generan información.  
Cualquier modificación, traspaso (habitualmente backend)

## Base de datos

Archivos de texto donde se graban y leen datos

Lugar donde se almacena el dato

Motor de base de datos (Oracle)

Programa que hace óptimo el acceso a los datos en un tipo de formato de archivo.

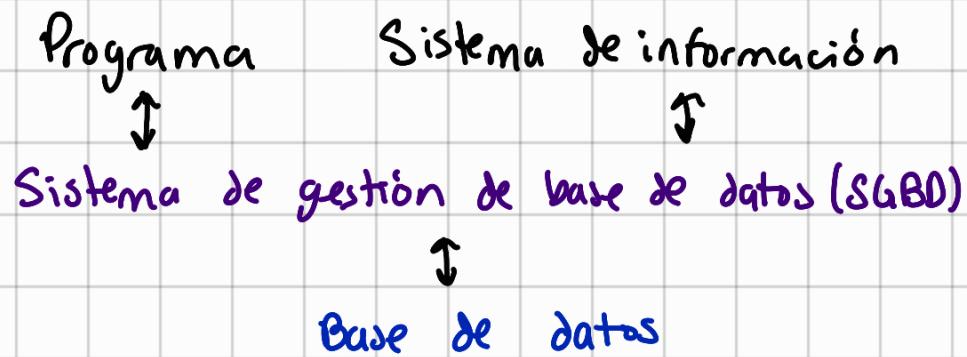
Aplicación especializada en almacenamiento y recuperación de datos de archivos.

Aplicación que administra datos en un tipo particular de archivo.

13 de marzo, 2024

## Sistema de gestión de base de datos

Consiste en que cambios en la estructura de la base de datos no repercuta en los programas o sistemas de información que la utilizan



## Base de datos.

- Lugar donde se almacenan datos de manera óptima.
- Conjunto. Nombre de la planilla de excel
- Dentro la base de datos tenemos tablas. Una tabla es una tabla, lo denominamos planilla.
- Cada tabla tiene un índice (uno o varios), que es una forma optimizada de recorrer la tabla a través de uno o varios campos. Estos pueden ser primarios (permite valores duplicados y ordena la tabla) o secundarios no permite valores duplicados y no ordena la tabla).
- Al dar nombre a un índice, la base de datos crea un árbol B-tree, el cual optimiza las búsquedas, haciéndolas más eficientes
- Mejoran el rendimiento. Las consultas se ejecutan más rápido
- Acceso rápido a la información. Se encuentran los datos específicos con mayor facilidad
- Eficiencia en la gestión de datos. Se optimiza el uso de recursos

## SQL

- Lenguaje estructurado de consulta.
- Traductor universal que convierte un lenguaje natural en un lenguaje de consulta de base de datos

**SQL** → **Lenguaje natural** → **Lenguaje de consulta**

Traductor Universal	Forma en la que el usuario expresa su necesidad de info	Lenguaje que la base de datos entiende
---------------------	---	--

## Beneficios

- Simple, unificado, independiente de cualquier lenguaje o motor de base de datos
- Interfaz única para el usuario. El usuario solo necesita entender el lenguaje natural para realizar consultas, no es necesario que conozca los detalles de la base de datos o el lenguaje de programación utilizado.

## Estandares

- ANSI SQL 92
- ANSI C
- Cada motor de base de datos tiene su propio dialecto SQL

15 de marzo, 2024

Motor de base de datos (postgres, mysql, sqlserver)  
Base de datos } distintos

Tupla. Conjunto de filas y columnas

## SQL

- Ni motor de base de datos, ni base de datos, es un lenguaje para acceder a la base de datos a través del motor de la base de datos.
- Me permite tener una capa intermedia y desentenderme de la base de datos con la que trabajo.
- Análisis lexicográfico. Ver si la estructura de consulta es correcta o no, si está bien escrita

## Transacción

- Conjunto de operaciones que quieren que se tomen de manera univoca, o se cumple todo, o no se cumple nada

- Cuando se hacen modificaciones a más de una tabla, se necesitan tomar una transacción para que no haya inconsistencias
- Cuando le digo al gestor que haga las modificaciones:
  - begin transaction (la hace pero no la ejecuta)
  - end transaction (si se cumplen, da el paso a ejecutar)

18 de marzo, 2024

### Motor de base de datos.

- Programa que accede a la base de datos.
- Gestor de base de datos

### Administrador de base de datos o gestor de base de datos

- Gestiona
  - Diseña
  - Planifica
  - Mantenimiento
  - Seguridad
  - Resguardo
- } La base de datos

### Base de datos. Archivo de almacenamiento.

## SQL

### ¿Qué es SQL?

- Lenguaje de consulta estructurado
- Permite acceder y manipular bases de datos
- Se convirtió en un estándar del ANSI en 1986 y de ISO en 1987

### ¿Qué puede hacer SQL? todo en una base de datos

- Ejecutar consultas
- Recuperar datos
- Insertar registros
- Actualizar registros
- Eliminar registros
- Crear nuevas bases de datos
- Crear nuevas tablas
- Crear procedimientos almacenados
- Crear vistas
- Establecer permisos sobre tablas, procedimientos y vistas

## RDBMS (Sistema de gestión de bases de datos relacionales)

- Es la base de SQL y de todos los sistemas de bases de datos modernos
- Los datos en RDBMS se almacenan en objetos de base de datos llamados tablas
  - Tabla. Colección de entradas de datos relacionados y consta de columnas y filas. Cada tabla se divide en entidades más pequeñas llamadas campos.
  - Campo. Columna de un tabla diseñada para mantener información específica sobre cada registro de la tabla.
  - Registro. Fila, es cada entrada individual que existe en una tabla. Entidad horizontal en una tabla
  - Columna. Entidad vertical en una tabla que contiene toda la información asociada con un campo específico en una tabla

## Backup

- Mantiene las bases de datos existentes

## Declaraciones

- Acciones → declaraciones SQL
- Constan de palabras clave que son fáciles de entender

## Ejemplo

```
SELECT * FROM customers;
```

↓  
extrae  
todos los nombres  
de campo

→ nombre de la tabla

Devuelve todos los registros de una tabla denominada clientes

- Las palabras clave no distinguen entre mayúsculas y minúsculas

## Comandos SQL más importantes

**SELECT**

extrae datos

**UPDATE**

actualiza datos

**DELETE**

elimina datos

**INSERT INTO**

inserta nuevos datos

**CREATE DATABASE**

crea una nueva base de datos

**ALTER DATABASE**

modifica una base de datos

**CREATE TABLE**

crea una nueva tabla

**ALTER TABLE**

modifica una tabla

**DROP TABLE**

elimina una tabla

**CREATE INDEX**

crea un índice (clave de búsqueda)

**DROP INDEX**

elimina un índice

## SELECT

- Se utiliza para seleccionar datos de una base de datos
- Si se desea devolver todas las columnas, sin especificar cada nombre de columna, puede utilizar **SELECT \***

→ nombre de la base de datos

**SELECT \* FROM nombre.Custumer**

→ todas las columnas

→ tabla de la base de datos

## SELECT DISTINCT

- Se utiliza para devolver solo valores distintos
- Elimina valores duplicados

**SELECT DISTINCT Country FROM Customers;** "corte vertical"

Selecciona todos los diferentes países de la tabla clientes

Podemos combinar esta declaración con una función llamada COUNT para devolver el número de países diferentes

```
SELECT COUNT(DISTINCT Country) FROM Customers;
```

## WHERE

- Filtra registros (filas)
- Extrae solo aquellos registros que cumplen con la condición específica

```
SELECT * FROM Customers
```

WHERE Country = 'Mexico'; "corte horizontal"

- Se pueden usar otros operadores además del = para filtrar la búsqueda	= igual	<> diferente de (!=)
	> mayor que	BETWEEN Entre un cierto rango
	< menor que	LIKE Busca un patrón
	>= mayor o igual	IN Especifica múltiples valores posibles para una columna
	<= menor o igual	

## ORDER BY

- Ordena el conjunto de resultados en orden ascendente o descendente

```
SELECT * FROM Products
```

```
ORDER BY Price;
```

- Los registros se ordenan de manera ascendente de forma predeterminada. Para ordenarlos en orden descendente, se utiliza DESC

```
ORDER BY Price DESC;
```

- Los strings se ordenan alfabéticamente

- Para ordenar por varias columnas

```
SELECT * FROM Customers  
ORDER BY DESCCountry, ASCCustomerName
```

Ordena por país, pero si algunas filas tienen el mismo, ordena por nombre

Se puede agregar ASC o DESC para ordenar de manera ascendente y descendente

## AND

- Usada con where
- Filtra registros en función de más de una condición

WHERE Country = 'Spain' AND CustomerName LIKE 'G%'

me va a devolver la tabla con los datos de España donde la columna CustomerName empieza con la letra G

## OR

- Usada con where
- filtra registros en función de más de una condición

WHERE Country = 'Spain' OR Country = 'Germany'

## NOT

- utilizado en combinación con otros operadores para dar el resultado opuesto

WHERE NOT Country = 'Spain';

Devuelve todos los clientes que no son de España

WHERE CustomerName NOT LIKE 'A%';

Selecciona el nombre de los clientes que no empiecen con la letra A

## INSERT INTO (no hay bloqueos)

- Insertar nuevos registros en una tabla
- Se puede especificar tanto el nombre de las columnas como los valores que se insertaran, o bien, si se agregaran valores para todas las columnas de la tabla, no es necesario especificar el nombre de estas

**INSERT INTO table-name (column1, column2...)**  
**VALUES (value1, value2, value3...)**

- También es posible insertar varias filas con esta declaración

**INSERT INTO table-name (column1, column2...);**  
**VALUES**  
**(value1, value2, value3...);**  
**(value1, value2, value3...);**  
**(value1, value2, value3...);**

## Valores nulos

### ¿Qué es un valor nulo?

- Un campo con valor NULL es un campo sin valor
- Es distinto de un campo con valor cero o un campo que contiene espacios
- Campo que se dejó en blanco durante la creación del registro
- No es posible probar con operadores de comparación, usaremos **IS NULL** y **IS NOT NULL**

### IS NULL y IS NOT NULL

- Prueba valores vacíos y no vacíos, respectivamente

**WHERE Address IS NULL**  
**IS NOT NULL**

## UPDATE

- Modifica los registros existentes en una tabla
- Se utiliza con where, que nos ayuda a especificar qué registros deben actualizarse, si se omite, se actualizarán todos los registros de la tabla

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

## DELETE Se generan bloques

- Elimina registros existentes en una tabla
- Como update, se utiliza con la cláusula where, para especificar qué registros deben eliminarse, si se omite where, se eliminarán todos los registros. Limpia la tabla, mantiene la estructura, pero no borra la tabla

```
DELETE FROM table_name WHERE condition
```

- Para eliminar una tabla completa usamos **DROP TABLE**

```
DROP TABLE customers;
```

20 de marzo, 2023

## Programación batch / shell

- Línea de comandos
- Archivo por lotes
- Archivo de texto con comandos del sistema operativo que se ejecutan en este caso windows (batch) y shell (linux) para automatizar tareas

## BCP

- Herramienta de linea de comandos para importar / exportar
- .exe que se conecta por la puerta de atrás del motor de base de datos e inserta o baja masivamente de registros, es un dump de la base de datos, este solo las trae y las pone en un .txt

## ETL (Extract, Transform, Load)

los procesos ETL permiten a las empresas convertir datos estructurados y sin estructura para tomar decisiones que afectan a sus negocios.

- Extracción. Origen de datos cualquier: página web, archivo, etc.
- Transformación. Procesar los datos, los dejo listos (corrección)
- Carga.

La idea es sacarle el esfuerzo al motor de base de datos procesándolo por afuera y con un programa batch hacer la carga y descarga.

Se puede aprovechar el orden de los datos para no tener un índice de más

Estructura de árbol.

Árbol binario.

Ordenamiento

## Modelos de Base de Datos

Uno debería de conocer al menos los siguientes tres modelos de base de datos

### Base de datos relacionales

- Guardan información tabular
- Datos altamente estructurados

## Base de datos NoSQL

- Bases de datos no distribuidas
- Tienen distintos nodos, se conectan entre sí y actualizan
- Guardan datos no estructurados
- Archivos, imágenes

## Bases de datos in memory

- Sistema de gestión de base de datos que almacenan y manipulan datos en RAM en lugar de discos físicos
- Más rápidos

	Relational database	NoSQL database	Time-series database	NewSQL database	In-memory database	Distributed database
Supported data types	Structured data	Unstructured and semi-structured data	Time-series data	Structured, unstructured and semi-structured data	Structured, unstructured and semi-structured data	Structured, unstructured and semi-structured data
Data storage models	Pre-defined, fixed relational model	Schema-less	Conceptual data model	Supports relational data model and schema-less	Varies depending on the database	Varies depending on the database
Main use case	Transaction processing and complex queries	High performance and availability	Storing and analyzing time-series data	Storing high data volumes and running complex queries	Fast query response times	Storing and analyzing large data volumes
ACID compliance	Yes	Varies depending on the database	No	No	Yes	Varies depending on the database
Scalability	Vertical scaling	Horizontal scaling	Vertical scaling	Horizontal scaling	Vertical scaling	Horizontal scaling

El método de acceso es un programa que me permite administrar acceso al medio o dispositivo de almacenamiento

22 de marzo, 2024

## SELECT TOP

- Especifica el número de registros a devolver
- útil en tablas grandes con miles de registros

SELECT TOP 3 \* FROM Customers;

selecciona los primeros tres registros de la tabla clientes

- MySQL admite LIMIT
- Oracle FETCH FIRST n ROWS ONLY ROWNUM

Cuantas filas que mostrar en la pantalla  $\times 2 = \text{numero}$

## Funciones de agregación

- Realiza un cálculo sobre un conjunto de valores y devuelve un valor único.

- Se utilizan a menudo con GROUP BY, SELECT

- Las más utilizadas

MIN() devolver el valor más pequeño dentro de la columna seleccionada

MAX() " " más grande "

COUNT() devolver el número de filas de un conjunto

SUM() devolver la suma total de una columna numérica

AVG() devolver el valor promedio de una columna promedio

- Ignoran los valores nulos (excepto COUNT())

## Ejemplo de sintaxis

función → nombre de columna  
SELECT MIN(Price)  
FROM Products;

## LIKE

- Se usa con la cláusula WHERE

- Busca un patrón específico en una columna

- Para campos de texto

- Busca los elementos que tengan cierta condición

- Comodines que se utilizan con LIKE

% representa zero, uno o varios caracteres

\_ representa un solo carácter

## Ejemplo

"a%" que empieza con la letra a  
↳ que luego tengo cualquier cosa

"% Moreno%" que tenga la palabra moreno  
↳ que tenga la palabra moreno  
↳ que empieza con cualquier cosa

"a\_\_%" tienen al menos 3 caracteres de longitud  
↳ que empiecen por a

"\_r%" tiene a r en la segunda posición

## Wildcard Characters

Se usan para sustituir uno o más caracteres en un string

% cero o más caracteres

\_ un solo carácter

[ ] cualquier carácter dentro de los corchetes

devuelve un resultado si alguno de los caracteres dentro coincide

^ cualquier carácter que no esté entre paréntesis

permite especificar una variedad de caracteres dentro de [ ] [a-F]

- cualquier carácter sin un rango específico

{ } cualquier carácter escapado

## IN

- Permite especificar múltiples valores en una cláusula where
- Abreviatura de múltiples OR condicionales

SELECT \* FROM Customers

WHERE Country IN ('Germany', 'France', 'UK')

Devuelve todos los clientes de Alemania, Francia o Reino Unido

- Si usamos NOT devolvemos todos los registros que NO son ninguno de los valores de la lista

## BETWEEN

- Selecciona valores dentro de un rango determinado
- Inclusivo (incluye los valores inicial - final)

`SELECT * FROM Products`

`WHERE Price BETWEEN 10 AND 20;`

Devuelve todos los productos con un precio entre 10 y 20

- Podemos agregar **NOT** para devolver los registros que no están en el rango

## AS (Alias)

- Da a una tabla, o a una columna en la tabla un nombre temporal que solo existe mientras dura esa consulta
- Trae una columna con otro nombre, sin cambiar el título de la base de datos original.
- Se utilizan a menudo para hacer que los nombres de las columnas sean más legibles

`SELECT column-name AS alias-name  
FROM table-name;`

`SELECT column-name(s)  
FROM table-name AS alias-name`

- Para incluir espacios, podemos encerrar el nombre entre corchetes [ ] o bien, utilizar comillas dobles
- Para concatenar usamos +, como lo hacemos en python

`SELECT CustomerName, Address + ';' + PostalCode AS Address`

## Metadatos

- Datos de los datos, estructura, los datos que vas a consultar.
- Conocimiento de cuál es la estructura de los datos con los que vamos a trabajar

25 de marzo, 2024

## Paginación

Técnica utilizada para dividir y presentar grandes conjuntos de datos en partes más pequeñas y manejables, lo que beneficia tanto a los usuarios como a los sistemas que gestionan esa información.

## Frameworks

Proporcionan herramientas y funcionalidades para implementar la paginación de manera efectiva

## Base de datos recurrente

- Donde estoy trabajando con varios usuarios a la vez
- Puede haber inconsistencias al subir datos manipulados

## LOCK (lockeo, bloqueo) de base de datos

- Es como la base de datos define que para hacer una actualización de un dato, tiene que impedir el acceso al mismo tiempo de otros usuarios
- El motor define como va a tomar de manera única un registro para que esa particula no sea accedida al mismo tiempo por otros usuarios mientras se modifica el dato en ese momento (evita problemas de inconsistencia)
- Los bloqueos en una base de datos son mecanismos utilizados para controlar el acceso concurrente a los datos y garantizar la consistencia y la integridad de la información en entornos multiusuarios.
- Cuando múltiples transacciones intentan acceder o modificar los mismos datos al mismo tiempo, es esencial tener un sistema de bloqueo.

para evitar problemas como lecturas sucias, escrituras perdidas y otros conflictos de concurrencia que pueden llevar a resultados inconsistentes.

- Por defecto la base de datos hace este lockeo automáticamente. Se genera un procedimiento por el cual según la cantidad de registros decide si es a nivel de tabla, registro, celda/columna de un registro
- El bloqueo de tablas en una base de datos ocurre cuando un proceso o transacción adquiere un bloqueo exclusivo en una tabla, impidiendo que otros procesos o transacciones realicen operaciones de lectura o escritura en la misma tabla hasta que el bloqueo sea liberado

### Tipos de bloqueo

- Lectura
- Escritura
  - Actualización
  - A nivel de fila

Tenemos que entender como trabaja el proceso de update/borrado en el motor de la base de datos que estoy trabajando. No es un conocimiento trivial, hay que informarse.

Lo que hemos visto hasta ahora son solo consultas, sin lockeos  
select → consulta → no hay bloqueos

- Por defecto las bases de datos van a intentar lockear lo menos posible
- Lockear impacta mucho en el motor.

La obligación del motor es recibir las consultas y ver si estas se encuentran en lo que está marcado como LOCK. Pero no puede verificar todas las consultas al mismo tiempo, tiene que esperar a que se haga un realce a la primera consulta para atender la segunda y así sucesivamente. Durante este proceso, el usuario lo que vio fue decaer el tiempo de performance en las respuestas. Esto es lo que sucede cuando una tabla, un registro de una BDD se lockea.

- El valor por defecto de cualquier motor es a nivel de REGISTRO. Todos los campos de un registro son lockeados.
- Si el motor ve que es una actualización masiva, puede lockear toda la tabla, es menos costoso

Todas las bases de datos tienen una interfaz gráfica. La interfaz gráfica de acceso a una base de datos es un programa que puede venir adentro del paquete pero es independiente a la misma.

## Views (vistas)

- Es una abstracción de una o varias tablas
- A nivel físico no existe, solo existe a nivel lógico
- Me permite trabajar con los usuarios, las personas que acceden a la base de datos y mostrarles un subconjunto de campos
- Se hacen para simplicidad y seguridad
- Puedo crear una consulta como una vista, le hago creer al usuario que esa vista es una tabla

## Triggers.

- Acción que se va a realizar en una base de datos en función de algo
- Se ejecutan antes o después de una consulta o transacción
- Se usa principalmente cuando se va a modificar, borrar o insertar un registro
- Especie de procedimiento almacenado que se ejecuta automáticamente cuando ocurre un evento específico en la base de datos.
- Se utilizan principalmente para mantener la integridad de los datos, realizar validaciones o ejecutar acciones adicionales en respuesta a ciertos cambios en la base de datos
- Por ejemplo, podrían utilizarse para actualizar automáticamente un campo en una tabla cuando se inserta una nueva fila, o para validar ciertos datos antes de permitir una operación

- Un ejemplo más cotidiano: Email de alerta
- El uso excesivo de triggers impacta en la performance de la base de datos

## Stored procedure

- Programa o procedimiento almacenado físicamente en una BD
- Conjunto de instrucciones SQL que se guardan en la base de datos y se pueden ejecutar de manera repetida cuando sea necesario
- Utilizados para realizar tareas repetitivas o complejas en una base de datos de una manera más eficiente y segura
- Usos comunes:
  - Procesamiento de datos
  - Validación y seguridad
  - Transacciones
  - Mejora el rendimiento
  - Reutilización de código

trigger → automático

Stored procedure → manual

1. SQL interpreta el análisis lexicográfico (valida si lo que escribí está ok)  
Analiza si las tablas y los campos que escribimos estaban bien
2. Analiza los permisos
3. Ejecuta la consulta  
(DB browser)

Si tengo todo en stored procedure, estos 3 pasos se hacen solo cuando guardo el stored procedure y no en cada consulta.

Ahora, cuando invoco al stored procedure solo se fija en el análisis de los permisos.

Así que abstractan el trabajo a los programadores: lo único que programamos son los stored procedure y desde el programa accedemos a ellos

## BCP (bulk copy program)

- Dependen del motor de base de datos.
- Copia masiva de datos
- Programa que puede estar separado de la base de datos o es parte misma de ésta que accede por una puerta de atrás a la base de datos y me permite bajar y subir de forma óptima grandes cantidades de registros.

## Índices

En una base de datos se utilizan para mejorar el rendimiento de las consultas al acelerar la búsqueda y recuperación de datos.

Para hacer una copia masiva de información, lo más conveniente sería:

1. Hacer un sort por fuera (ordenamiento), que coincida con el índice principal de la tabla
2. Borrar el índice en la tabla
3. Largar los datos en el orden correcto
4. Generar de nuevo el índice

## Importancia del borrado y la regeneración de índices

- Mejoran el rendimiento de las consultas y optimización de uso de recursos (**performance**)
- Aceleran la búsqueda

Recordatorio: El performance hace referencia a la eficiencia y velocidad con la que la base de datos puede procesar consultas y manipular datos.

## Problema del "queso bruller"

- Se presenta cuando se tienen tablas que se actualizan borrando y creando registros constantemente

- En el índice quedan huecos en las ramas (por ser un árbol binario), ya que muchas de estas ramas y bifurcaciones que tenemos al llegar a los nodos hojas van quedando desactualizadas, porque los nodos hojas o no existen o se quedaron nulos. Esto tira abajo la performance de los equipos.
- Los administradores de bases de datos borran y crean de nuevo el índice para evitar este problema.

### Manejos de acceso a una base de datos.

En términos generales existen dos principales:

#### 1. Librería propia de la base de datos (p: p install)

La ventaja es que no existe una capa de abstracción en el medio. Con esta manera gano performance pero pierdo movilidad

#### 2. Mediante una ODBC (option datos base conexión)

- Forma estándar de acceder a una base de datos (sobre todo cuando se trabaja con windows)

- Capa de abstracción

- Los inserts se hacen desde un archivo de configuración

#### PROS

Podemos configurar un acceso estándar (localmente en el servidor)

Todos pueden acceder

- Es una manera más simple de migrar entre bases de datos. Cuando me conecto directo a la capa no me conecto directo a la dirección de la base de datos, la cual puede cambiar, me conecto a su interfaz.

#### CONTRAS

Capa de abstracción extra

Tiempo de respuesta más lento

## Optimizar transacciones

Marcar la tabla como read only. Las consultas van a ser más performantes y rápidas. No bloquean los recursos ni afectan otras transacciones.

## Paginación

- División del contenido en páginas pequeñas
- Traslación con page up y page down (cursor)

## Tip (frameworks)

- Saber como se conecta el framework a una base de datos
- Buscar como pagina ese framework (habilidad de poder saltar entre frameworks)

## Plataforma de datos

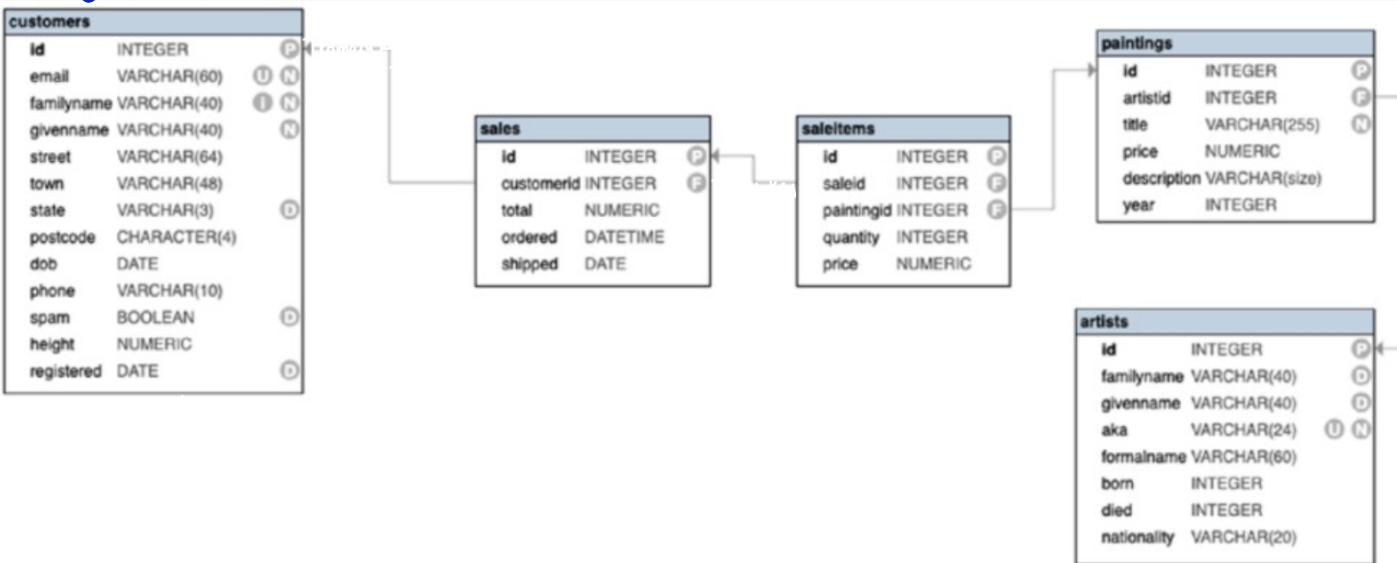
- Lugar donde vamos a consultar conjuntos de datos de manera segura
- Combinaciones y secuencias de datos, que pueden estar integradas a varios motores de base de datos
- Conjunto de herramientas, tecnologías y recursos que se utilizan para recopilar, almacenar, procesar, analizar y compartir datos de manera eficiente y segura.

## Buenas prácticas de programación al hacer una consulta.

1. Colocar autor (quien hizo la consulta)
2. Fecha
3. Breve descripción de lo que hace la consulta
4. Indicar los parámetros y tipos de entrada
5. Indicar la salida que obtiene el usuario

# Ejemplo de la estructura de una base de datos

## Diagrama Entidad / Relación (DER)



- La base de datos puede tener 0 o N tablas
- Un motor de base de datos administra 0 o N base de datos
- Los datos físicamente están guardados dentro de las tablas

### Primary Key

Son como físicamente se almacenan los datos en la base de datos

### Foreign Key

que dicen que es clave en otra tabla (de donde salen las flechas)

### Formas normales (normalización)

- Conjunto de reglas que me permite hacer un equilibrio entre performance y no repetir datos en una base de datos
- Tratan de reducir el espacio de almacenamiento de una forma óptima sin generar una sobrecarga en la base de datos

### DBMS (Administrador)

Software que administra los datos

### Tabla

Colección de filas y columnas

## Tupla.

- Una de las filas que se han extraído
- Combinación de registros de múltiples columnas

## Record

- Tupla
- Fila puntual de lo que estamos trabajando

OS de abril, 2024

## Lineamientos de base de datos

1. Lenguaje de consulta
2. Normalización
3. Acceso programático a la base de datos

Nota: normalizar la base de datos antes de ejecutar una linea de código.

## Datos dummy

- Conocidos también como datos ficticios o datos de ejemplo, son conjuntos de información creados específicamente para propósitos de demostración, entrenamiento o pruebas
- Suelen ser generados artificialmente y no representan necesariamente datos reales o históricos
- Cuando se trabaja con otros equipos, muchas veces no vamos a tener los otros módulos listos, para eso podemos parallelizar el desarrollo pidiendo previamente el diseño y usando datos dummy para poder avanzar

## Acceso programático

- Hace referencia a la capacidad de un programa para interactuar y manipular datos almacenados en una base de datos de manera

automatizada y mediante código de programación.

- Implica que el programa pueda realizar consultas, inserciones, actualizaciones y eliminaciones de datos, así como también ejecutar otros comandos y transacciones relacionadas con la base de datos.

## Normalización de base de datos

- Ahorrar espacio para ser mucho más funcional
- Ayuda a diseñar bases de datos de manera que los datos estén organizados de forma eficiente y sin redundancias innecesarias.
- El proceso de normalización se basa en una serie de reglas o formas normales (Boyce-Codd, 1NF, 2NF, 3NF, etc.) que se aplican a las tablas de la base de datos para eliminar la redundancia y asegurar la coherencia de los datos.

## Regresando a SQL

### JOIN

- Juntar 2 o N tablas en las cuales se une la información
- Estructura y clausula propia (Clausula: palabra clave, como se escribe la consulta)
- Relación entre las tablas
- Se usan lo más acotadamente posible porque es muy costoso
- Cláusula que se utiliza para combinar filas de dos o más tablas, en función de una columna relacionada entre ellas.

### INNER JOIN



Devuelve registros que tienen valores coincidentes en ambos tablas

### LEFT JOIN



Devuelve todos los registros de la tabla izquierda y los registros coincidentes de la tabla derecha

### RIGHT JOIN



Devuelve todos los registros de la tabla derecha y los registros coincidentes de la tabla izquierda

### FULL JOIN



Devuelve todos los registros cuando hay una coincidencia en la tabla izquierda o derecha.

## INNER JOIN

- Tipo de unión predeterminada para JOIN
- Devuelve solo las filas que coinciden en ambas tablas, es decir no puede devolver algo que se encuentra en una tabla pero no en otra.

```
SELECT column-name(s)  
FROM table1  
INNER JOIN table2  
ON table1.column-name = table2.column-name;
```

## LEFT JOIN

- Devuelve todos los registros de la tabla 1 y los registros coincidentes en la tabla 2
- Si no hay coincidencia el resultado es 0 del lado derecho (tabla 2).
- Devuelve todos los registros de la tabla izquierda (tabla 1) incluso si no hay coincidencias en la tabla derecha (tabla 2).

```
SELECT column-name(s)  
FROM table1  
LEFT JOIN table2  
ON table1.column-name = table2.column-name;
```

## RIGHT JOIN

- Devuelve todos los registros de la tabla 2 y los registros coincidentes en la tabla 1
- Si no hay coincidencia el resultado es 0 del lado izquierdo (tabla 1)
- Devuelve todos los registros de la tabla derecha (tabla 2) incluso si no hay coincidencias en la tabla izquierda (tabla 1)

```
SELECT column-name(s)
FROM table1
RIGHT JOIN table2
ON table1.column-name = table2.column-name;
```

## FULL JOIN

- Devuelve todos los registros cuando hay coincidencia en los registros de la tabla izquierda (tabla 1) o derecha (tabla 2)
- FULL OUTER JOIN
- Devuelve todos los registros coincidentes en ambas tablas, independientemente de que la otra tabla coincida o no

```
SELECT column-name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column-name = table2.column-name
WHERE condition;
```

## Self Join

- Es una unión normal, pero la tabla está unida consigo misma

```
SELECT column-name(s)
FROM table1 T1, table1 T2
WHERE condition
```

T1 y T2 son alias de tabla diferentes para la misma tabla

Nota. El problema del JOIN, es que si no sé que tipo de JOIN usar, puedo hacer consultas de forma incorrecta y que no me devuelvan los campos que necesito. Hay que tener cuidado y revisar si necesito hacer inner, left, right

## Performance JOIN

Hay dos consejos de Oro para hacer mas performante un JOIN:

1. Los campos por los que accedemos (donde se colora la igualdad), en la tabla donde se busca la información (no la que se recorre) sea indice (sea una clave). Así evitamos un full scan
2. Inferir sobre como optimizar cada consulta dependiendo del motor de base de datos. Pues cada motor de base de datos hacen optimizaciones, entonces cada uno tiene formas diferentes de resolver cada operación

## Clave foránea

- Clave que es clave en otra tabla
- Se relaciona generalmente con la clave primaria, pero no necesariamente tiene que ser

08 de abril, 2021

## Normalización de base de datos

- Proceso de organizar los datos en una base de datos
- Minimiza la redundancia y la dependencia de datos
- Disminuir problemas de actualización
- Proteger la integridad de los datos (cuando accedo a un dato en la base de datos se que es único y si estoy viendo ese valor ese es el valor apropiado)
- Se busca dividir las tablas en entidades más pequeñas y relacionadas entre si de manera lógica
- Se aplica a través de una serie de formas normales (1NF, 2NF, 3NF, 4NF, etc), cada una de las cuales tiene reglas específicas para garantizar la integridad y eficiencia de la base de datos

## Ejemplo.

Si tienes dos tablas con información: Clientes (nombre, dirección, teléfono) y Pedidos y Fechas, se puede normalizar al vincular los pedidos a los clientes a través de un identificador único. Así evitamos la repetición de

datos de cada cliente en cada registro de pedido.

Mas allá de la normalización: cada campo en la base de datos tiene un valor por defecto.

## Primer Forma Normal (1FN)

- Todos los atributos son individuales
- No existen valores repetidos
- Cada columna tiene un solo valor
- No hay grupos repetidos de columnas

## Segunda Forma Normal (2FN)

- Todo registro depende de la clave principal
- Existen columnas que pueden depender de otras tablas
- Está en 1NF y además y cada columna no clave está completamente dependiente de la clave prima

## Tercera Forma Normal (3FN)

- No hay columnas que dependan de otras tablas/columnas que no sean la clave principal
- Si estás en 2NF y no existen dependencias transitivas entre las columnas no clave

## Boyce - Codd

- Cumple con 3NF y no tiene dependencias funcionales no triviales (no tiene campos repetidos).
- Se aplica para eliminar ciertas anomalías de actualización que pueden surgir en bases de datos normalizadas hasta 3NF
- Se espera que esté entre la 3 y 4 (también es llamado 3.5NF)
- Es una extensión de 3NF y se aplica cuando existen dependencias funcionales no triviales entre las claves candidatas de una relación.

Dependencia funcional no trivial: una columna (o conjunto de columnas) determina otra columna, y esa determinación no es evidente o trivial

- En términos prácticos: para cada dependencia funcional no trivial  $X \rightarrow Y$ , donde:  
 $X$ , Superconjunto de claves candidatas  
entonces  $X$  debe ser una superclave, es decir, una clave candidata.

No siempre tenemos vamos a tener que llegar al último grado de normalización, por cuestiones de performance, lo que nos lleva al siguiente concepto:

## Desnormalización

- Ir bajando de nivel en los tipos de normalización
- Proceso contrario a la normalización
- Se combina información de varias tablas relacionadas para mejorar el rendimiento de consultas específicas.
- Se utiliza cuando se necesita optimizar el rendimiento de consultas complejas a costa de aumentar la redundancia de datos.

Boyce-Codd      →      3FN      →      2FN      →      1FN

Empiezo a normalizar inicialmente hasta la forma que considero sería la más apropiada, y llegado a ese punto voy para atrás, comienzo a desnormalizar si es que veo cuestiones de performance.

## Full Scan

- Proceso por el cual un motor de base de datos necesita recorrer toda la tabla obligatoriamente para saber si la respuesta a la consulta que he recibido es completa, es decir, si es todo lo que tiene que enviar como respuesta a la consulta

## Proceso Batch / lotes

- Se usa cuando tenemos que hacer modificaciones o actualizaciones masivas
- Va por fuera de la aplicación y toma el proceso habitual de la base de datos
- Es la modificación puntual.
- Tenemos dos formas diferentes de hacer esta actualización masiva:
  1. Se hace una pequeña aplicación que solamente hace la modificación que se le pide en la tabla. Tiene un proceso de log in. Suelen no tener interfaz gráfica. El administrador de base de datos toma ese programa y lo adjunta a un chop control chop control: proceso por el cual se encadenan tareas.
  2. Se usa cuando la base de datos es muy grande, si tabla es muy grande y la modificación es muy grande, y sobre todo si la modificación lleva consigo borrar registros, actualizar registros que son de campo de claves foraneas, secundarias o primarias. Entonces lo que se hace es un ETL
  - a. Se hace un dump de la base de datos, de los registros que se tienen que modificar
  - b. Se modifican todos esos registros en forma batch
  - c. Se hace un delete de esos registros en la base de datos <sup>no drop table</sup>
  - d. Se hace un load de los registros modificados

## Transacciones.

- Utilizadas cuando se hacen actualizaciones masivas
- Unidades lógicas de trabajo que consisten en una o varias operaciones de base de datos. Estas operaciones se ejecutan como una sola unidad (o todas las operaciones se cumplen con éxito o ninguna se lleva a cabo), manteniendo así la integridad de los datos
- Se basan en el principio ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad, por sus siglas en inglés)

**Atomicidad.** Unidad atómica de trabajo, implica que todas las operaciones de la transacción se ejecutan con éxito o ninguna se ejecuta en lo absoluto.

**Consistencia.** luego de que la transacción se completa con éxito, la base de datos debe mantenerse en un estado consistente, es decir, todas las restricciones de integridad deben cumplirse en todo momento.

**Aislamiento.** cada transacción debe ejecutarse como si fuera la única transacción en el sistema, es decir, las operaciones de una transacción no deben interferir con las operaciones de otras transacciones que se estén ejecutando simultáneamente.

**Durabilidad.** Despues de que la transacción se confirma en la base de datos, sus cambios deben ser permanentes y resistir a fallas del sistema.

**Commit.** Confirma y aplica los cambios realizados en una transacción, al ejecutarlo, los cambios realizados en la transacción se vuelven permanentes y se guardan en la base de datos. Estos cambios se hacen visibles para otros cambios y aplicaciones.

**Rollback.** Operación que deshace los cambios realizados en una transacción que no ha sido confirmada mediante un commit. Al ejecutarse, los cambios no se aplican y se restaura el estado anterior de la transacción, como si los cambios nunca hubieran ocurrido.

\* Un índice secundario no mueve datos, solo genera estructuras.

10 de abril, 2024

**DuckDB** → motor de base de datos

- Motor de base de datos
- Pensado para base de datos relacionales (fama tabular)

## OLAP (OnLine Analytical Processing)

Forma de tener una consola de comandos, mostrada en tiempo real de como se van viendo los datos de manera resumida

## Cursor

Mantiene conexiones e indices dentro del area de memoria del motor de base de datos para trabajar desde la parte programática de una manera sencilla (saber a donde estamos apuntando, cuando hicimos un select)

## SQL

### UNION

- Combinar el conjunto de resultados de dos o más SELECT
  - Cada SELECT dentro de UNION debe tener el mismo número de columnas
  - Las columnas deben tener tipo de dato similares
  - Las columnas de cada SELECT deben estar en el mismo orden

SELECT column-name(s) FROM table 1

UNION

SELECT column-name(s) FROM table 2;

- Selecciona solo valores distintos de forma predeterminada, usamos UNION ALL para permitir valores duplicados

## GROUP BY

- Agrupa filas que tienen los mismos valores en filas de resumen
- Se usa a menudo con funciones agregadas (COUNT, MAX, MIN, SUM, AVG).

SELECT column-name

FROM table-name

WHERE condition

GROUP BY column-name;

## HAVING

- Se agregó a SQL porque WHERE no se puede usar con funciones agrupadas

```
SELECT column_name  
FROM table-name  
WHERE condition  
GROUP BY column_name  
HAVING condition  
ORDER BY column_name;
```

## EXISTS

- Se utiliza para probar la existencia de cualquier registro en una subconsulta.
- Devuelve true si la subconsulta devuelve uno o más registros.

```
SELECT column_name  
FROM table-name  
WHERE EXISTS  
SELECT column_name FROM table-name WHERE condition;
```

## ANY and ALL

- Permiten realizar una comparación entre un valor de una sola columna y un rango de otros valores
- ANY:
  - Devuelve un valor booleano como resultado
  - Devuelve true si cualquiera de los valores de la subconsulta cumple la condición
  - Significa que la condición será verdadera si la operación es verdadera para cualquiera de los valores del rango

```
SELECT column_name  
FROM table-name  
WHERE column_name operator ANY  
SELECT column_name FROM table-name WHERE condition;
```

operador de comparación extendido

## - ALL

- Devuelve un valor booleano como resultado
- Devuelve true si todos los valores de la subconsulta cumplen la condición
- Se usa con SELECT, WHERE y HAVING
- Significa que la condición será verdadera solo si la operación es verdadera para todos los valores del rango

```
SELECT ALL column-name  
FROM table-name  
WHERE condition;
```

```
SELECT column-name  
FROM table-name  
WHERE column-name operator ALL  
SELECT column-name FROM table-name WHERE condition;
```

operator de comparación extendida

12 de abril, 2024

## SQL

### SELECT INTO

- Copia datos de una tabla en una tabla nueva

```
SELECT column1, column2, column3...  
INTO newtable [IN externaldb]  
FROM old-table  
WHERE condition;
```

ponemos \* para copiar todas las columnas en la nueva tabla

- Podemos usarlo también para crear una tabla nueva y vacía utilizando el esquema de otra

```
SELECT * INTO newtable  
FROM oldtable  
WHERE 1=0;
```

## INSERT INTO SELECT

- Copia datos de una tabla y los inserta en otra
- Requiere que los tipos de datos en las tablas de origen y de destino coincidan
- Los registros existentes en las tablas de destino no se ven afectados

```
INSERT INTO table2  
SELECT * FROM table1  
WHERE condition;
```

(Copiar solo algunas columnas)

```
INSERT INTO table2 (column1, column2, column3...)  
SELECT column1, column2, column3... FROM table1  
WHERE condition;
```

## CASE

- Pasa por condiciones y devuelve un valor cuando se cumple la primera condición
- Una vez que una condición es verdadera dejará de leer y devolverá el resultado
- Si ninguna condición es verdadera, devuelve el valor de la cláusa ELSE, si no hay ELSE y ninguna condición es verdadera, devuelve NULL

## CASE

```
WHEN condition1 THEN result1  
WHEN condition2 THEN result2  
WHEN conditionN THEN resultN  
ELSE result
```

```
END;
```

## Funciones NULL

### IFNULL()

- Permite devolver un valor alternativo si una expresión es NULL

IFNULL (column-name, valor a devolver)

(COALESCE(), ISNULL(), NVL())

- tienen la misma función y sintaxis que IFNULL()

## Stored Procedure

### ¿Qué es?

- Es un código SQL preparado que puede guardar, de modo que el código pueda reutilizarse una y otra vez.
- Si se tiene una consulta SQL que escribe una y otra vez, guardese como un stored procedure y luego simplemente llámelo para ejecutarlo
- También es posible pasar parámetros al stored procedure

CREATE PROCEDURE procedure-name

AS

sql-statement

GO;

Para ejecutarlo

EXEC procedure-name

Ejemplo cuando se pasa un parámetro:

CREATE PROCEDURE SelectAllCustomers @City nvarchar(30)

AS

SELECT \* FROM Customers WHERE city = @City

GO;

EXEC SelectAllCustomers @City = 'London';

- Configurar múltiples parámetros es muy fácil, simplemente enumere cada parámetro y el tipo de datos separados por una coma

## Buenas Prácticas (SP)

- Se optimizan las operaciones de base de datos.
- Se validan el tipo y el rango de datos para cada uno de ellos
- Integridad en mi base de datos
- Acceder a los datos y validar a los mismos y en el resultado de la consulta poder hacer una normalización dentro de mi base de datos.
- No está pensado para formatear datos, workflow o validar usuarios que pueden acceder o no a un proceso
- Meter todo en un SP es un error, pues están hechos para acceder de manera óptima a la base de datos

## Acceder desde una app a una base de datos

- El administrador le va a dar un **usuario genérico** que puede hacer lo justo y necesario
- Este usuario está relacionado con la aplicación
- El nombre que se le asigna al usuario suele tener el nombre de la aplicación

## Acceder de forma programática a SQL

- Revisar sqlalchemy y duckdb

\* Para comentar en SQL usamos -- ó /\* \*/ para bloques

Reposo

15 de abril, 2024

## Regla para el JOIN

- Si no estamos seguros de cuál usar, usemos LEFT JOIN

## Claves

### ¿Por qué las generamos?

Porque eso hace fuerza a la base de datos a generar en memoria un árbol que es persistente (no se pierde), con la estructura y el posicionamiento de los datos (arborB). También evitamos el full scan (que está prohibido)

### Clave primaria

- Es única e irrepetible (identificador)
- El orden físico del archivo es el mismo que vamos a encontrar en mi índice.

### Clave Secundaria.

- Nuevo índice para que la consulta acceda a través del índice, sea mucho más rápido y consuma menos recursos de CPU
- No es el mismo que se tiene físicamente
- Permitir tener campos repetidos

### Clave foránea

- Campos que son índices en otras tablas, por lo general claves primarias

### Normalización

- Tendemos a trabajar con la forma Boyce - Codd, o llamada de forma informal 3.5FN
- Se pierde tiempo en el acceso cuando se está tan normalizado, porque tenemos que actualizar muchas tablas ante cualquier normalización

## Problema del gueso bruller

Si yo tengo un indice creado, ese indice se va modificando, y en el proceso de modificación empieza a tener una estructura de arbol(nodo) que no llevan a ningun lado. Para resolver el problema solo hay que borrar el índice y crearlo nuevamente, para que esa recreación ayude a reducir el tamaño del árbol).

Este problema solo se presenta cuando se trabajan con muchas modificaciones

- \* **Regla de escritorio.** Si accedemos a un tiempo que no es índice, hay que pedir ensiguiente que se cree un índice.
- \* Cuando se hace una modificación en los datos, y esa modificación afecta a alguno de los indices que tenemos en los datos se hace en tiempo real en la estructura del arbol B

17 de abril, 2024

## Regla de escritorio

El SQL fue hecho para consultar datos (validación del formato, cambio, agrupación), esto es todo lo que yo debería poner en mi store procedure. Más cosas son propias de un programador de store procedure.

## Crear una base de datos

Forma de agrupación y darle coherencia al conjunto de tablas con el que estoy trabajando

Tablas que tienen algún tipo de relación

Es importante separar y crear diversas bases de datos en lugar de tener una sola para poder trabajar mejor, por cuestiones de seguridad, almacenamiento y performance.

Suelen agruparse por tareas similares o datos que comparten

## SQL

### CREATE DATABASE

- Declaración que se utiliza para crear una nueva base de datos SQL

`CREATE DATABASE databasename;`

- Hay que asegurarse de tener privilegios de administrador antes de crear cualquier base de datos
- Podemos consultar una base de datos una vez creada usando `SHOW DATABASES`

### DROP DATABASE

- Se utiliza para eliminar una base de datos SQL existente

`DROP DATABASE databasename;`

- Eliminar una base de datos resultará en la pérdida completa de la información completa almacenada en la base de datos

### BACKUP DATABASE

- Declaración que se utiliza para crear una copia de seguridad completa de una base de datos SQL existente.

`BACKUP DATABASE databasename  
TO DISK = 'filepath';`

`BACKUP DATABASE databasename  
TO DISK = 'filepath'  
WITH DIFFERENTIAL;`

Una copia de seguridad con diferencial solo respalda las partes de la base de datos que han cambiado desde la última copia de seguridad completa (solo respalda los cambios)

\* Base de datos relacionales → datos tabulares, lo que hemos trabajado con SQL

## Bases de datos NoSQL

- Todas aquellas bases de datos que no son tablas
- Por lo general son base de datos distribuidas
- Funcionan con un esquema dinámico
- Utiliza datos no estructurados, esto significa que puede crear una aplicación sin tener que crear primero un esquema de base de datos.
- Los requisitos de datos pueden evolucionar según sea necesario, sin causar problemas con la base de datos.

Cuando crea una aplicación con una base de datos relacional, debe saber como funcionará la aplicación con tablas específicas que se componen de filas y columnas. Antes de agregar ese primer dato a la base de datos, se debe definir un esquema. Si lo descuida, su aplicación no podrá funcionar con la base de datos.

- Puede trabajar con videoclips, actividad del dispositivo móvil, redes sociales, documentos de texto, imágenes.
- No está limitada a entradas solo de texto

# 7 mejores bases de datos NoSQL

	Type	Primary Model	Query Language	Transactions	Scalability
MongoDB	Document	Document Store	MongoDB Query Language	Yes (ACID for single documents)	Horizontal
Apache Cassandra	Wide-column	Wide Column Store	CQL (Cassandra Query Language)	Limited ACID	Horizontal
Redis	Key-value / Data structure store	In-memory Data Store	Redis commands	Transactions with optimistic locking	Master-slave replication
Couchbase	Document	Document / Key-Value	N1QL, SQL++	ACID (on document level)	Horizontal
Neo4j	Graph	Graph Database	Cypher	ACID Transactions	Horizontal
Amazon DynamoDB	Key-value / Document	Key-Value and Document Store	AWS proprietary	ACID with limitations	Managed, Horizontal
ArangoDB	Multi-model	Document, Graph, Key-value	AQL (ArangoDB Query Language)	ACID	Horizontal

# SQL vs NoSQL

Característica	Bases de datos relacionales: ventajas	Bases de datos NoSQL: ventajas
Esquema	El esquema fijo promueve la integridad y coherencia de los datos.	El esquema dinámico para datos no estructurados permite flexibilidad en la estructura y los tipos de datos.
Escalabilidad	Escalable verticalmente al aumentar la potencia informática.	Escalable horizontalmente, lo que permite agregar fácilmente más servidores.
Actas	El sólido soporte para las propiedades ACID garantiza un procesamiento de transacciones confiable.	Modelos de transacciones flexibles adaptados a casos de uso específicos, con algunas propiedades ACID compatibles.
Consultando	Potentes capacidades de consulta con un lenguaje estandarizado (SQL) para consultas complejas.	Capacidades de consulta flexibles adaptadas al tipo de base de datos (por ejemplo, clave-valor, documento, gráfico).
Casos de uso	Ideal para aplicaciones con estructuras y relaciones de datos bien definidas que requieren transacciones complejas.	Adecuado para manejar grandes volúmenes de datos no estructurados o semiestructurados y para aplicaciones escalables con modelos de datos en evolución.
Actuación	Optimizado para consultas y relaciones complejas, con rendimiento mantenido mediante indexación y optimización.	Alto rendimiento para operaciones de lectura/escritura, particularmente con grandes conjuntos de datos y funciones escalables para mantener el rendimiento bajo carga.

22 de abril, 2024

SQL

CREATE TABLE

- Declaración que se utiliza para crear una nueva tabla en una base de datos

```
CREATE TABLE Personas(  
    PersonuID int,  
    Last Varchar (255),  
    FirstName Varchar (255),  
    Address Varchar (255),  
    City Varchar (255)  
)
```

## DROP TABLE

- Se utiliza para eliminar una tabla existente en una base de datos
- Rompe una base de datos, TODO, no hay manera de recuperar, hay que empezar de nuevo

```
DROP TABLE table-name;
```

## TRUNCATE TABLE

- Se utiliza para eliminar los datos dentro de una tabla pero no la tabla en si:
- Llama todos los registros de una tabla manteniendo toda la estructura de la tabla

```
TRUNCATE TABLE table-name;
```

```
DELETE borrar registros con una condición, rangos
```

```
DELETE * FROM nombre-tabla borrar todos los registros
```

## ALTER TABLE

- Se utiliza para agregar, eliminar o modificar columnas en una tabla existente
- Tambien se utiliza para eliminar restricciones en una tabla existente
- Se usan con DROP y RENAME

Agregar columna a una tabla

```
ALTER TABLE table-name  
ADD column-name datatype;
```

Eliminar una columna en una tabla

```
ALTER TABLE table-name  
DROP COLUMN column-name;
```

Cambiar el nombre de una columna en una tabla

```
ALTER TABLE table-name  
RENAME COLUMN old-name TO new-name
```

### Constraints (restricciones)

- Reglas que debo ponerle a la base de datos
- Se pueden especificar cuando se crea la tabla o después de crear la tabla

```
CREATE TABLE table-name (  
    column1 datatype constraint,  
    column1 datatype constraint,  
    column1 datatype constraint,  
    ...  
);
```

- Las restricciones se utilizan para limitar el tipo de datos que pueden incluirse en una tabla
- Garantiza la precisión y confiabilidad de los datos de la tabla
- Si hay alguna infracción entre la restricción y la acción de datos, la acción se cancela
- Las restricciones pueden ser a nivel de columna o de tabla

- Las restricciones que comúnmente se utilizan en SQL

**NOT NULL** Garantiza que una columna no puede tener un valor nulo  
Obliga a la columna a NO aceptar valores nulos

Un campo debe contener siempre un valor, no puede insertar un nuevo registro ni actualizar un registro sin agregar un valor a este campo

**UNIQUE** Garantiza que todos los valores de una columna sean diferentes  
Se pueden tener muchas por tabla

**PRIMARY KEY** Identifica de forma única cada fila de una tabla  
Solo una por tabla (puede constar de una o varias columnas)  
Deben contener valores únicos y no pueden tener valores nulos

**FOREIGN KEY** Previene acciones que destruirían enlaces entre tablas  
Es un campo (o conjunto de campos) en una tabla que hace referencia a una primary key en otra tabla

**CHECK** Garantiza que los valores de una columna satisfagan una condición específica

Límita el rango de valores que se pueden colocar en una columna  
Solo permitiría ciertos valores para la columna(s)

**DEFAULT** Establece un valor predeterminado para una columna si no se especifica ningún valor  
El valor predeterminado se agregaría a todos los registros nuevos si no se especifica ningún valor

**CREATE INDEX** Se utiliza para crear y recuperar datos de la base de datos muy rápidamente.

Los usuarios no pueden ver los índices, solo se utilizan para acelerar las búsquedas / consultas.

24 de abril, 2024

## SQL

### AUTO INCREMENT

- Permite generar automáticamente un número único cuando se inserta un nuevo registro en una tabla
- Este es el campo de clave principal que nos gustaría que se creara automáticamente cada vez que se inserta un nuevo registro.
- De forma predeterminada el valor inicial de AUTO\_INCREMENT es 1 y se incrementará en 1 para cada nuevo registro
- IDENTITY (comienzo, incremento)

### DATES

- Hay que asegurarse que el formato de la fecha que se está intentando insertar coincida con el formato de la columna de fecha en la base de datos

DATE formato AAAA-MM-DD

DATETIME formato AAAA-MM-DD HH:MI:SS

TIMESTAMP formato AAAA-MM-DD HH:MI:SS

YEAR formato AAAA o AA

- No olvidar colocar las comillas, como un string, al hacer una consulta
- Se pueden comparar dos fechas si no hay ningún componente de tiempo involucrado

### VIEWS

- Tabla virtual basada en el conjunto de resultados de una declaración SQL

CREATE VIEW view-name AS

SELECT column1, column2, ...

FROM table-name

WHERE condition;

## Injection

- Técnica de inyección de código que podrían destruir la base de datos
- Técnica de piratería web más comunes
- Colocación de código malicioso en sentencias SQL, a través de la entrada de una página web
- Ocurre cuando le solicita información al usuario
- Para proteger un sitio web de la inyección SQL, se usan parámetros SQL, que son valores que se agregan a una consulta SQL en el momento de la ejecución, de forma controlada

## Hosting

- Si desea que su sitio web pueda almacenar datos de una base de datos, su servidor web debe tener acceso a un sistema de base de datos que utilice el lenguaje SQL
- Bases de datos hosting más comunes: MS SQL Server, Oracle, MySQL y MS Access.

## Tipos de datos

- Define qué valor puede contener una columna
- Guía para que SQL sepa qué tipo de dato se espera dentro de cada columna y también identifica como interactuará SQL con los datos almacenados

26 de abril, 2024

## Arquitectura de base de datos

MQ Server (Administrador de colas)

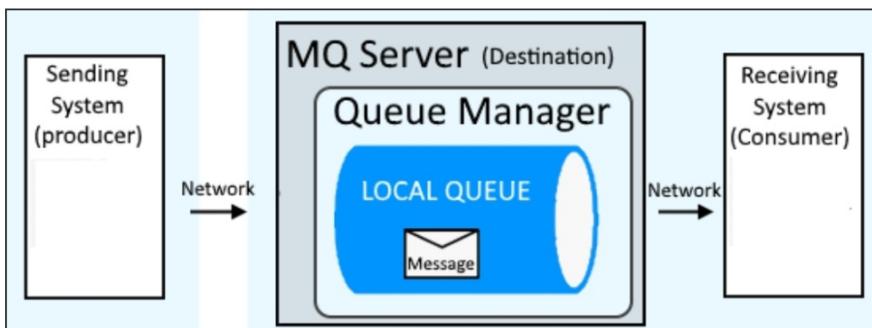
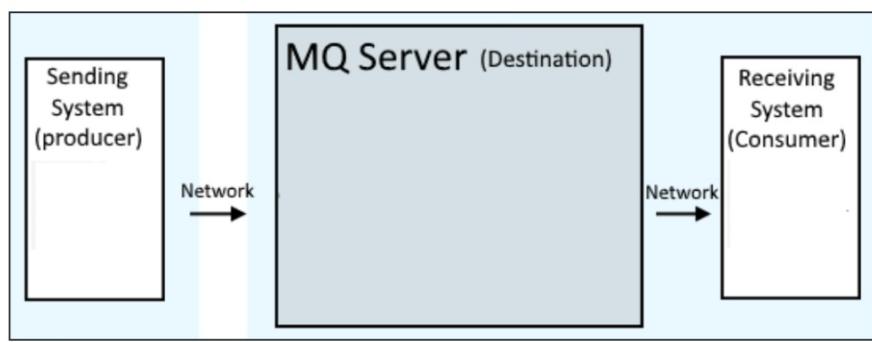
Cola de mensajería.

- Aplicación que se coloca enfrente del servidor
- No nos conectamos directamente a SQL, hacemos la conexión a un servidor de cola de mensajerías
- No podemos mezclar una conexión asíncrona (aplicación web)

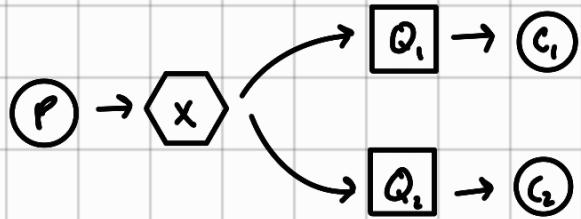
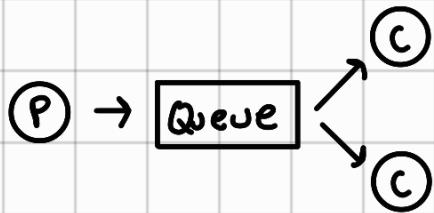
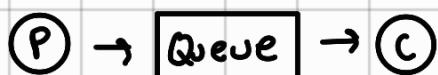
## Con una conexión síncrona (base de datos)



- Como programador, poniendo el MQ server en el medio, no le voy a estar pegando a la base de datos sino a un servidor de mensajería
- Este MQ server convierte una conexión síncrona en asíncrona
- MQ server fue creado específicamente para una sola tarea: recibir requerimientos y volcarlos en una cola de datos
- Se encuentra entre el sistema emisor y el sistema receptor (intermediario o proxy). Si el sistema receptor no está disponible, el sistema emisor no queda atrapado en una situación en la que no se puede pasar a su siguiente transmisión o se atasca al intentar enviar la transmisión una y otra vez. El sistema de envío simplemente envía los datos a MQ y luego continua (dispara y olvida). De esta forma MQ consigue que la transmisión de datos sea asíncrona.



### 3 modelos de trabajo



06 de mayo, 2024

### Temas para el parcial

1. Formas normales
2. Consultas SQL
3. Preguntas conceptuales, relacionada con lo hablado y explicado en clase
4. Situación hipotética a resolver (mejor forma de limpiar los datos, Armar una base de datos, tipo de base de datos, cuál sería la más apropiada) con una justificación

→ relational  
o NoSQL

### Base de datos NoSQL

- Surgieron por necesidades serias
- Formatos de base de datos que no funcionaban y no eran eficientes
- Evolución
- Surgieron en mundos distribuidos, nadie se podía pensar sin el uso de internet
- Base de datos distribuidas
- Han forzado nodos y arquitecturas en la red que son distribuidas
- Pensadas para resolver un problema particular

## Nombres importantes

- Base de datos orientada a documentos (.xml, .json)  
más populares y más usadas (mongodb)  
base de datos distribuidas  
¿cuáles son los métodos de replicación?  
Sincronización entre nodos y ajuste de esa sincronización (todas las bases de datos tienen un método de sincronización estándar)
- Bases de datos especializadas para una determinada tarea  
¿Para qué quiero utilizar mi base de datos?
- (MongoDB está hecho para consultas masivas)
- Cassandra para IoT, recibe actualizaciones de forma masiva

## Caso de estudio

- Método para presentar información
- Se presenta una situación (un dilema a resolver) con el nivel de profundidad y detalle que uno crea pertinente
- Fue creado para 2 tipos de situaciones
  1. Trabajar con un conjunto de personas donde se genere un debate y cada grupo argumente sus conclusiones ante la lectura del mismo
  2. Para algo que ya está resuelto, pero con la única diferencia que se corta la parte de la información para que se genere el debate y después se presenta la solución que se tomó y se pueden ver las diferencias o coincidencias
- En los casos de estudio no existen respuestas correctas o incorrectas

## Inteligencia artificial generativa

- No creativa
- Combinaciones de soluciones ya existentes

08 de mayo, 2024

## Data roaming

- Todos los procesos para mantener una determinada calidad de datos

## Nodo

- Instancia de una base de datos

Cantidad máx de usuarios vs cantidad máx de usuarios concurrentes

Tu pude haber tenido X cantidad de usuarios distintos que se conectaron a mi base de datos pero esos usuarios se conectaron en distintos momentos (cantidad máx) entonces, yo tuve como máximo un usuario conectado en cada unidad de tiempo X que le hayamos asignado (cantidad máx de usuarios concurrentes)

número mágico que yo quiero tener o entender, es lo que me va a permitir saber como tengo que armar mi aplicación, la arquitectura de mi aplicación va a estar dada no por las consultas, sino por la concurrencia (cuantas personas se conectan a mi base de datos al mismo tiempo)

## Block chain

La base de datos más conocida que replica todos sus datos de manera completa en cada uno de los nodos con los que cuenta es la block-chain (cadena de bloques)

Una block-chain es una base de datos distribuida y descentralizada que utiliza una tecnología de cadena de bloques para almacenar y gestionar datos. La cadena de bloques es una estructura de datos en la que los registros digitales (bloques) están vinculados de manera secuencial y segura mediante criptografía. Cada bloque contiene un conjunto de transacciones y un hash que apunta al bloque anterior, creando así una cadena inmutable de datos.

En una base de datos blockchain, cada nodo de la red tiene una copia idéntica de la cadena de bloques, lo que garantiza que todos los participantes tengan acceso a la misma información.

## Bloque (donde guardamos transacciones)

Eventualmente mis transacciones llenarán mi caja, cuando esa caja se llena se abre otra

Yo puedo tener una caja abierta en cada unidad de tiempo, pero no puedo tener dos cajas abiertas al mismo tiempo

Trabajamos con cajas que se cierran cuando se completan, eso te asegura que nunca pierdas una transacción porque las que no caen en una caja caen en la siguiente

Luego una caja cerrada pasa al siguiente estadio (con los mineros)

Los mineros lo que hacen es encontrar una representación criptográfica de esa caja. Pero ojo, los datos de la blockchain NO están encriptados, esa caja no se toca, los datos quedan igual que como en el preciso momento que esa caja se cerró.

nota: a través de un algoritmo de encriptación (hash 256) no se parte de la caja para llegar a la criptografía, se parte de un número, de una representación criptográfica que apunta a la caja, y es muy fácil validar si es la representación criptográfica ese número alfanumérico y hexadecimal que se ha generado.

Otra característica que tiene la blockchain es que es una red que intrínsecamente se va haciendo más segura cuanto más gente interactúe en ella, porque va a haber cada vez más personas que estén evaluando el hash, por tanto es menos probable que se puedan poner todos de acuerdo para concordar que un hash es la representación de una caja y realmente no lo sea.

En cuanto un minero ha encontrado el hash y, por ejemplo, 5 nodos más validaron eso, el minero recibe su recompensa en bitcoins y la caja se hace visible en la blockchain

Es el preciso instante donde Raul se ve ese 0.5 de bitcoin que Nell le envió

## Cadena

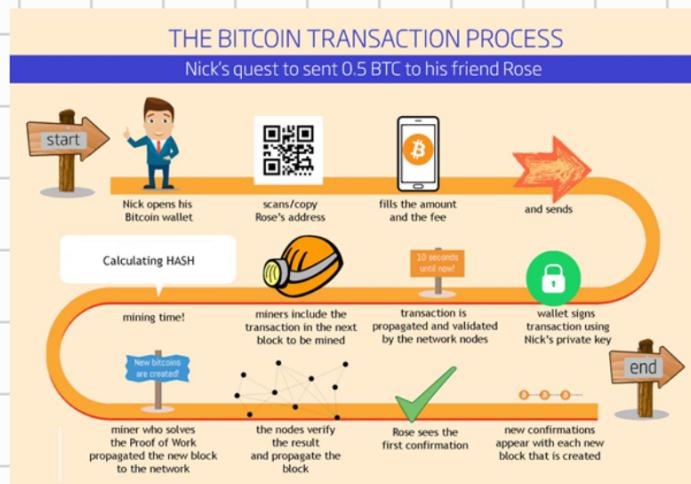
Cada caja tiene un lugar reservado para poner el hash de la caja anterior.

Cuando se cierra la caja yo no puedo meter más transacciones pero lo que voy a meter, porque tengo un lugar reservado específicamente en mi caja, es el hash encontrado de la caja anterior.

Ahora con las transacciones + el hash de la caja anterior hago todo este proceso de cálculo del hash de la caja actual & El cálculo del hash de la caja que acabo de cerrar tiene incluido el hash de la caja anterior \*

Esto es seguro porque si yo en cada caja tengo el hash de la caja inmediatamente anterior que ha sido cerrada, lo que me asegura es que en cada paso yo puedo validar que no se me ha modificado ningún dato. No se puede modificar la caja actual ni ninguna de toda la cadena de bloques, y eso se puede seguir validando (con la validación del hashing) hasta el **nodo génesis** (primer nodo creado de la blockchain)

En consecuencia, la blockchain al tener tantos nodos, no necesita un respaldo de datos, ya que su propia arquitectura de la distribución en nodos y la replicación completa de todas las transacciones en todos los nodos hace que no necesite tener ningún backup de la misma.





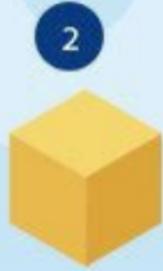
A quiere enviar dinero a B



El bloque se transmite a todas las partes de la red



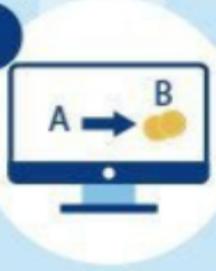
El bloque entonces puede añadirse a la cadena, lo que proporciona un registro indeleble y transparente sobre las transacciones



La transacción se representa en la red como un "bloque"



los que están en las redes aprueban que la transacción es válida



El dinero se mueve de A a B