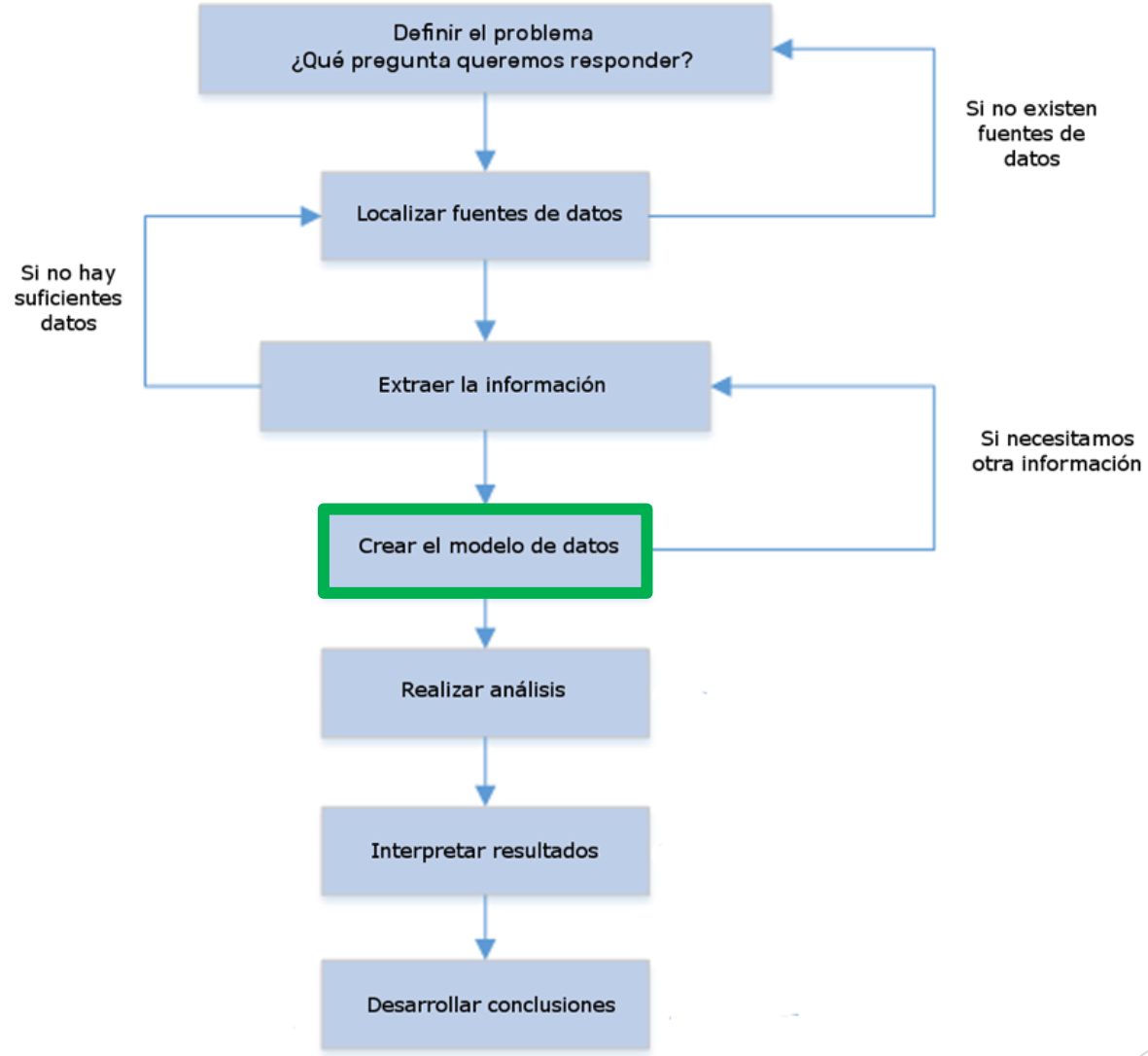
El modelo de datos

Dr. Francisco E. Cabrera

El proceso de un análisis de datos



El modelo de datos

Se trata del lugar donde vamos a almacenar la información para que pueda ser analizada.

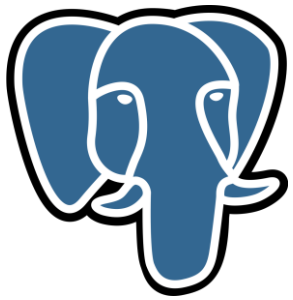
Tenemos varias opciones

- Modelo Relacional (SQL)
- Modelo Clave-Valor
- Modelo de Documentos
- Modelo de Columnas Anchas
- Modelo de Grafos
- Formatos de almacenamiento optimizado

Modelos de bases de datos

Modelo Relacional

- Se basa en tablas con filas y columnas.
- Usa claves primarias y foráneas para definir relaciones entre tablas.
- Las consultas se realizan mediante SQL.
  - SQL es un estándar.
- Garantiza ACID, evitando errores en las transacciones.




ACID

Transacciones ACID

- Atomicidad:
  - Una transacción debe ejecutarse completamente o no ejecutarse en absoluto.
  - Si falla en algún punto, la base de datos debe revertir todos los cambios previos (rollback) para mantener la integridad.

ACID

Transacciones ACID

- Consistencia
  - Una transacción debe llevar la base de datos de un estado coherente a otro estado coherente.
  - No puede violar restricciones (como claves primarias, foráneas, unicidad, etc.).

```
INSERT INTO usuarios (id, nombre) VALUES (1, 'Ana');
INSERT INTO usuarios (id, nombre) VALUES (1, 'Lucas');
```

ACID

Transacciones ACID

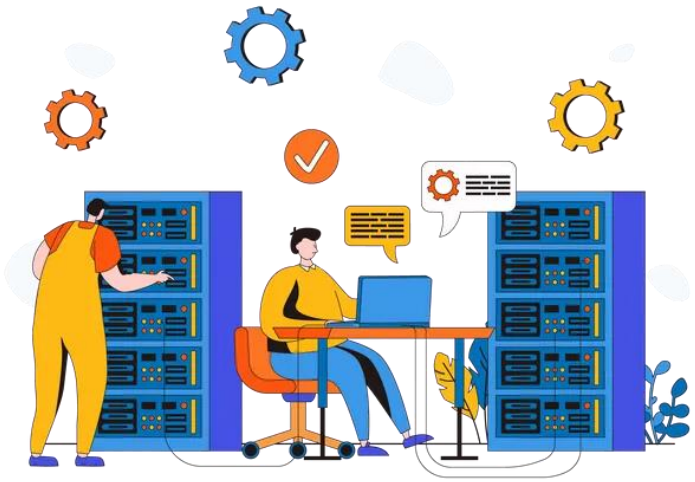
- Aisalmiento:
  - Si múltiples transacciones ocurren al mismo tiempo, sus operaciones no deben afectar el resultado final.
  - SQL permite niveles de aislamiento para controlar esto.



ACID

Transacciones ACID

- Durabilidad:
  - Una vez que una transacción es confirmada los cambios deben permanecer en la base de datos.
  - Incluso si hay un fallo en el sistema.



Relaciones entre tablas

- Cada tabla representa una entidad
  - Por ejemplo: Clientes, pedidos, productos, etc.
- Relaciones entre tablas:
  - Los datos se vinculan mediante claves primarias y claves foráneas.
    - Clave primaria: Identifica un elemento en una tabla de manera única.
    - Clave foránea: Referencia a la clave primaria de otra tabla.
- Usa operaciones relacionales:
  - JOIN, SELECT, INSERT, UPDATE, DELELE, etc.

Modelo Relacional

Ejemplo:

- Recuperar los últimos pedidos de “Juan López”

```
SELECT p.pedido_id, p.fecha_pedido,
FROM Pedidos p
JOIN Clientes c ON p.cliente_id = c.cliente_id
WHERE c.nombre = 'Juan' AND c.apellido = 'López'
ORDER BY p.fecha_compra DESC;
```

pedido_id	fecha_pedido
546	21/01/2025
520	19/12/2024
177	08/04/2024

Modelo Clave-valor

- Cada dato se almacena como un par clave-valor.
  - Es similar a un diccionario de Python o un objeto a JSON.
- No tiene esquema fijo.
- Es muy rápido y simple
  - Únicamente se necesita la clave para acceder al valor.

Amazon DynamoDB

Modelo Clave-valor

Ventajas

- Lecturas y escrituras rápidas
- Buena escalabilidad horizontal
- Simples y eficientes
  - Cache, info de sesión…

Inconvenientes

- No permiten consultas complejas
- Dificultad en la relación entre datos

Amazon DynamoDB

Base de datos Documental

- Utiliza un esquema flexible como JSON, BSON o XML.
- Modelo basado en documentos.
- Los documentos pueden contener estructuras anidadas y no requieren un esquema rígido.
- Alta escalabilidad.
- Optimizado para ciertos casos de uso.
  - Especialmente para grandes volúmenes de datos anidados.
  - No recomendables para casos de uso donde haya muchas relaciones entre datos.

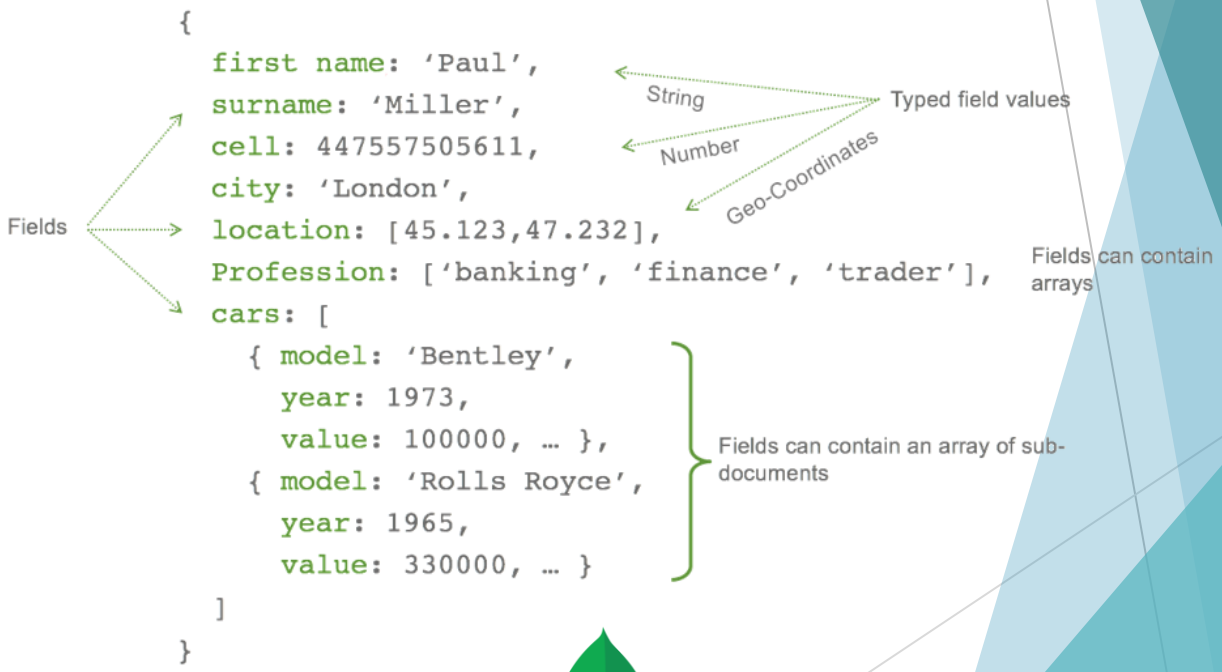
Base de datos Documental

Ventajas

- Esquema flexible
- Alta escalabilidad
- Eficiencia en lectura/escritura
- Útil para aplicaciones web y móviles.

Inconvenientes

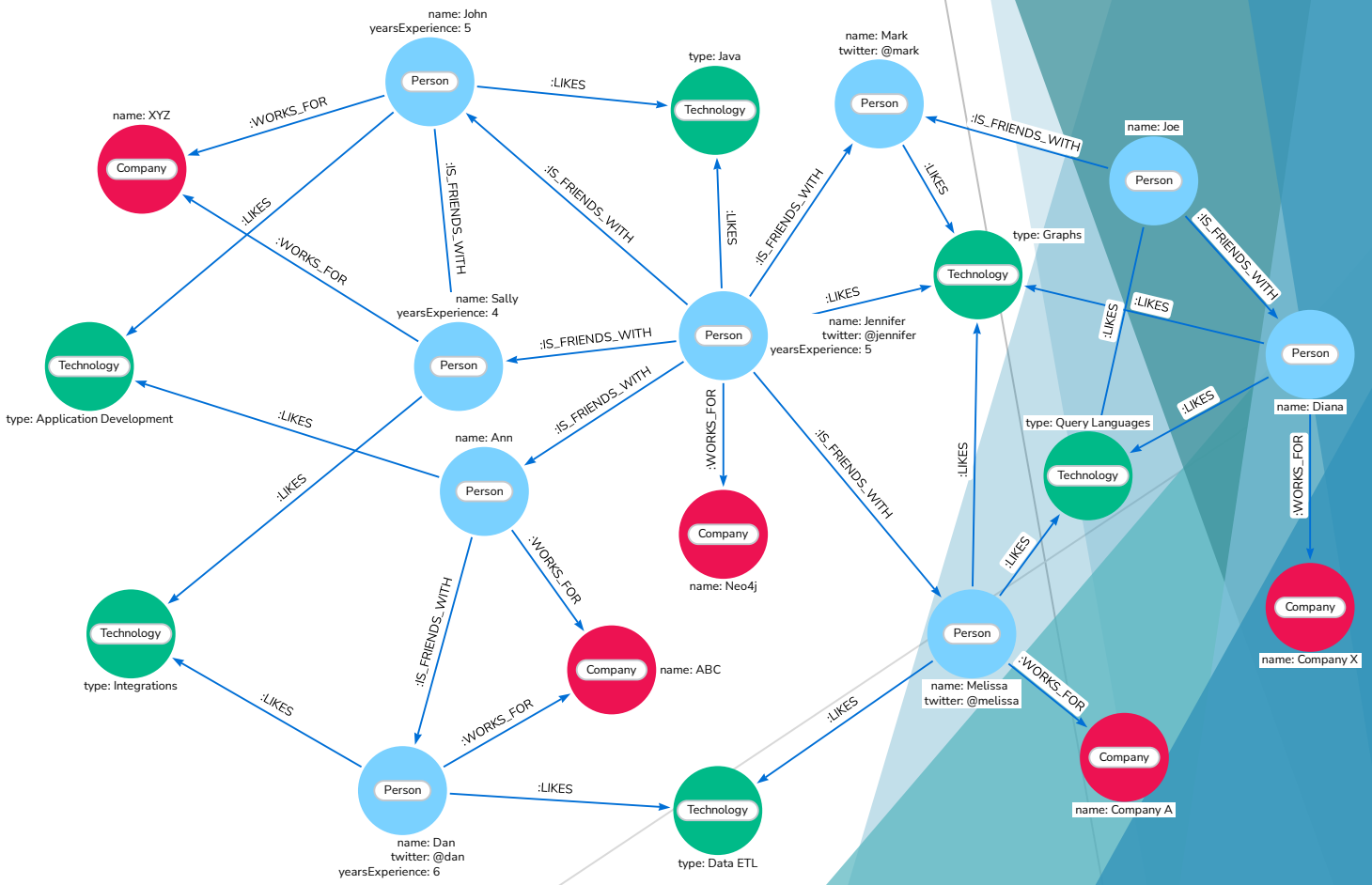
- No es ideal para relaciones complejas
- Ineficiente para consultas que requieren una fuerte consistencia



mongoDB®

Bases de datos de Grafos

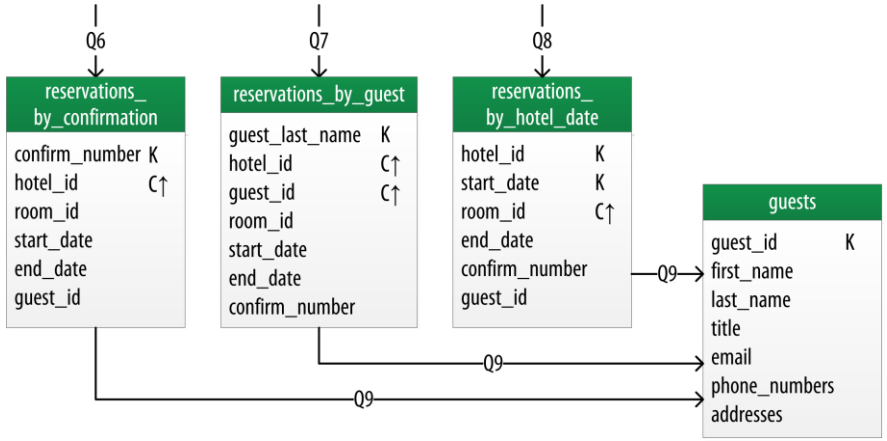
- Almacenan los datos en nodos y aristas.
  - Los nodos representan entidades.
  - Las aristas representan relaciones.
- Diseñadas para manejar datos interconectados eficientemente.



neo4j

ArangoDB

Bases de datos de Columna Ancha



- Organiza los datos en columnas en lugar de filas
- Cada fila puede tener un número variable de columnas
  - Agrupadas en familias de columnas

Ventajas

- Buen almacenamiento para grandes volúmenes
- Buen rendimiento y escalabilidad horizontal
- Diseñadas para escritura intensiva

Inconvenientes

- Inadecuado para transacciones ACID complejas
- Mayor latencia en consultas individuales

cassandra

APACHE HBASE

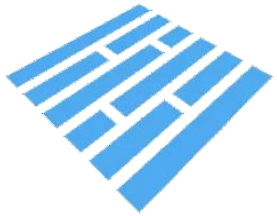
Formatos de archivo para análisis

Formatos de almacenamiento optimizado

Formatos de archivos para análisis de grandes volúmenes.

- Usados en Data Lakes y Big Data.
- Son eficientes en el almacenamiento y la consulta
- Tipos principales
  - Parquet
  - Avro
  - ORC, JSON, CSV, etc…

Parquet

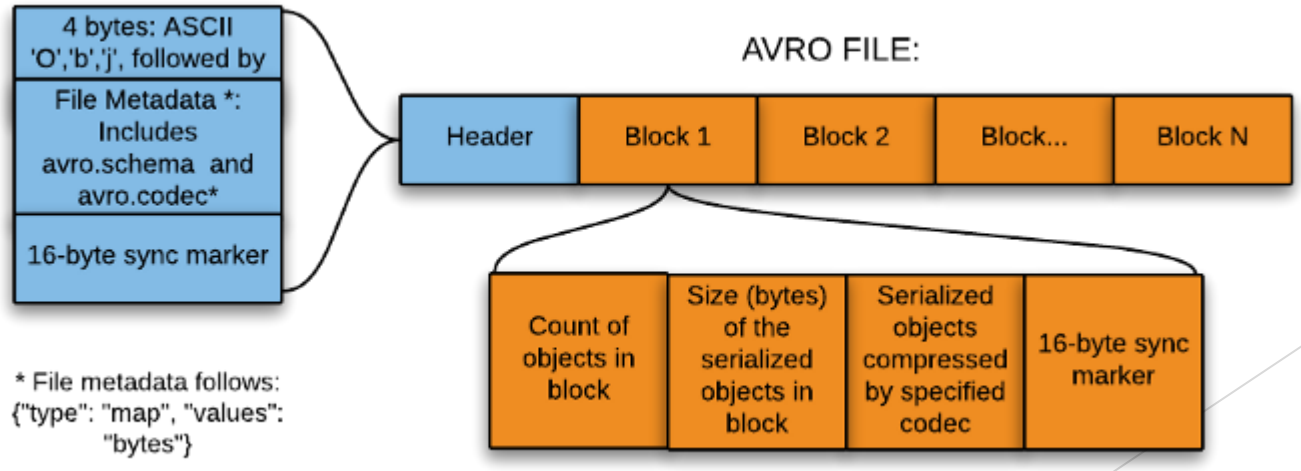


- Es un formato columnar que almacena los datos en grupos de filas
  - Permite acceder a datos de una columna concreta de un gran dataset rápidamente
- Compresión eficiente
- Optimizado para análisis en Spark

	Column 1	Column 2	Column 3	Column 4	Column 5
	Product	Customer	Country	Date	Sales Amount
Row Group 1	Ball	John Doe	USA	2023-01-01	100
	T-Shirt	John Doe	USA	2023-01-02	200
Row Group 2	Socks	Maria Adams	UK	2023-01-01	300
	Socks	Antonio Grant	USA	2023-01-03	100
Row Group 3	T-Shirt	Maria Adams	UK	2023-01-02	500
	Socks	John Doe	USA	2023-01-05	200

AVRO

- Almacenamiento orientado a filas
- Bueno para transmisión de datos (Kafka, ETL, etc.)



¿Cómo se usan estos formatos?

- Las bases de datos almacenan datos estructurados y semiestructurados
- Los DataLakes almacenan los datos en su forma original
- El almacenamiento distribuido nos permite manejar grandes volúmenes de datos de manera escalable
- Los formatos tipo Parquet y AVRO almacenan los archivos en DataLakes de manera eficiente

Caso práctico

Ejemplo: Tienda Online

- Usa MySQL para las transacciones de clientes.
- Va almacenando logs de usuarios en MongoDB
- Exporta los datos en formato Parquet a un Data Lake montado en HDFS
- Analiza estos datos usando Spark en busca de patrones de comportamiento.
  - Ejemplo: ¿Qué productos miran los clientes, pero luego no compran?
  - Otro ejemplo: ¿Cómo puedo ajustar el stock que necesito de un producto?