Introduction:

Data Modelling → abstraction of data organization and interrelation.

↓ Translate

Database Modeling

1. Process to support users and business applications
2. Requirement gathering
3. Conceptual Data Modeling
4. Logical Data Modeling
5. Physical Data Modeling

Key Points on Data Modeling:

→ Data Organization: data organization

→ Use cases: taking into consideration data main usages, will help to create a "data model" more suited for organization requirements

→ Starting Early: planning is important

Iterative Process

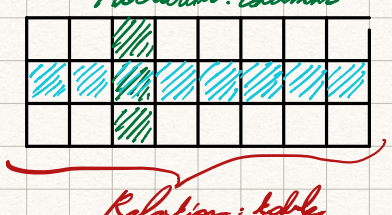
RDBMS:

* Relational Model: this model organizes data into one or more tables (or "relations") of columns and rows, with a unique key identifying each row; generally the table represents "one entity type"

Relational Database Management System

Attribute: column

Tuple: row



Relation: table

SQL: structured query language → language used for querying and maintaining the database.

→ MySQL → Postgre SQL → Sqlite

Structure: → Database / Schema: collection of tables

→ Tables / Relations: a group of rows sharing the same labeled element.

Advantages of Relational Databases:

→ Ease of use -- SQL
→ Ability to do joins
→ Ability to do aggregation and analytics
→ Smaller data volumes
→ Easier to change Business requirement
→ Flexibility for queries
→ Modeling the data not modeling queries
→ Secondary indexes (secondary key)
→ ACID Transaction - data integrity

ACID Transactions:

Atomicity → the whole transaction is processed or nothing is processed (all or any step is performed)

Consistency → only transactions that abide by constrains and rules are written into the DB.

Isolation → transactions are processed independently, order does not matter.

↓ isolation: simultaneous users 😃 concurrency effects 😟

↑ isolation: less concurrency effect 😃 more resources and transaction blocking 😟

Durability → completed transactions are saved to the database, even on the case of system failure.

Disadvantages of relational databases:

- → Have large amounts of data, not distributed
- → Handle data in different formats
- → Need high throughput - fast reads
- → Need a flexible schema (add columns not used by every row).
- → Need high availability → Relational DB are not distributed
- → Need horizontal scalability

NoSQL Databases:

→ Not Only

NoSQL databases has a simpler design, simpler horizontal scaling, and finer control of availability. Data structures used are different than those in "Relational Databases" and make some operations faster.

Types:

- Apache Cassandra (Partition row store) → data distributed partitions across nodes and servers and data is organized columns and rows format.
- Mongo DB (Document store) → in addition to "key lookup"; there is an api to retrieve documents based on its content.
- Dynamo DB (Key-Value store) → data represented as a collection of "key-value" pairs.

- Apache HBase (Wide Column store) → names and formats of the columns can vary from row to row.

- Neo4j (Graph database) → relationships between entities is the focus. Data is conformed by nodes and edges.

Data modelling for NoSQL DB:

Apache Cassandra → Keyspace: collection of tables

Table: a group of partitions

Rows: a single item

Partition:
- Fundamental unit of access
- Collection of rows
- How data is distributed

Primary key: is made up of a "partition key" and "clustering columns"

Columns:
- Clustering & Data
- Labeled Elements

Apache Cassandra:

... provides scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data.

→ Query language: CQL (Cassandra Query Language)

When to use a NoSQL database:

→ Larger amount of data: relational DB are not distributed → can only scale vertically // NoSQL databases are horizontally scalable.

→ Need to be able to store different

data formats: is able to handle different: structured, semi-structured and unstructured

→ Need horizontal scalability: is the ability to add more machines or nodes to a system to increase performance and data space.

→ Need high throughput: while ACID bring benefits → they also slow the process.

→ Need a flexible schema: allow to add columns that do not need to be used by every row.

→ Need high availability: traditional databases (relational) have a single point of failure

↳ NoSQL does not have this problem

When not to use a NoSQL database?

→ Small dataset: they are constructed for big datasets, not performant on small ones.

→ Need ACID Transactions: most NoSQL databases do not provide this (except "MongoDB")

→ Need ability to do joins across tables: No SQL does not allow it, as this will result in full table scans.

→ Want to do aggregations and analytics: /

→ Have changing business requirements: Ad-hoc queries are possible but difficult as the data model was done to fit particular queries.

→ If your queries are not available and you need flexibility: you need your queries in advance → construct appropriate data model ⇒ without this → Stick to relational databases.