System Analysis and Design

Database Design

By: Vahid Rahimian

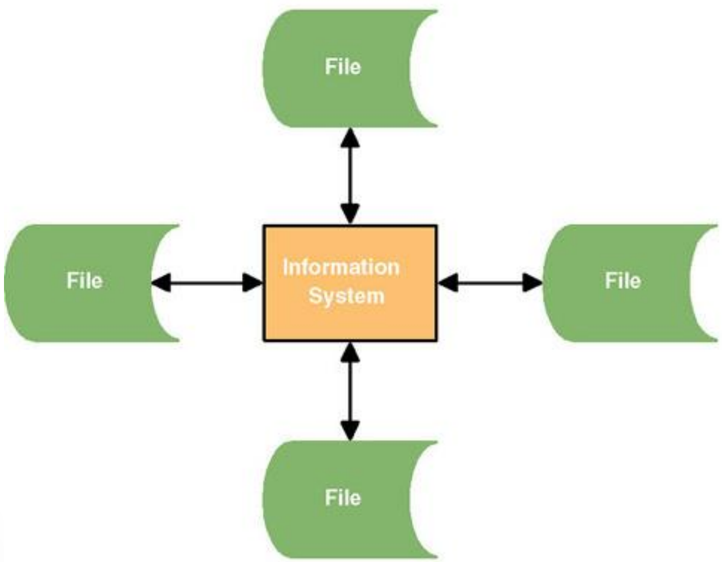
Spring 2022

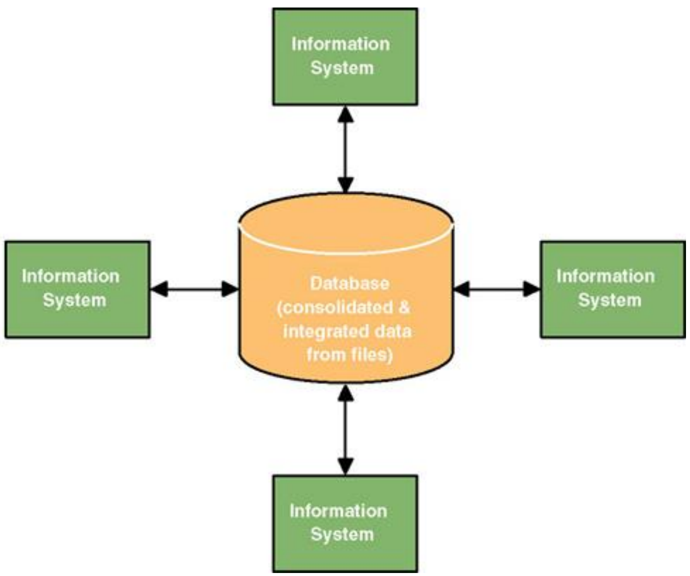
File vs. Database

- File: a collection of similar records.
  - Files are unrelated to each other except in the code of an application program.
  - Data storage is built around the applications that use the files.

- Database: a collection of interrelated files
  - Records in one file (or table) are physically related to records in another file (or table).
  - Applications are built around the integrated database

File vs. Database





Pros and Cons of Conventional Files

Pros

- Easy to design because of their single-application focus
- Excellent performance due to optimized organization for a single application

Cons

- Harder to adapt to sharing across applications
- Harder to adapt to new requirements
- Need to duplicate attributes in several files.

Pros and Cons of Databases

Pros

- Data independence from applications increases adaptability and flexibility
- Superior scalability
- Ability to share data across applications
- Less, and controlled redundancy (total non-redundancy is not achievable)

Cons

- More complex than file technology
- Somewhat slower performance
- Investment in DBMS and database experts
- Need to adhere to design principles to realize benefits
- Increased vulnerability due to consolidating in a centralized database

Relational Database

- A database that implements stored data in a series of two-dimensional tables that are “related” to one another via foreign keys.
- The physical data model is called a schema.
- The DDL and DML for a relational database is called SQL (Structured Query Language).
- Triggers – programs embedded within a database that are automatically invoked by updates.
- Stored procedures – programs embedded within a database that can be called from an application program.

A Good Data Model

- A good data model is simple
  - The data attributes that describe an entity should describe only that entity
- A good data model is nonredundant
  - Each data attribute exists in at most one entity (except for foreign keys)
- A good data model is flexible and adaptable to future needs

- *These goals are usually achieved through database normalization.*

Database Design

- The process of developing database structures from user requirements for data

- Objectives
  - 1. Derive relationships
  - 2. Evolve to meet user requirements
  - 3. Do it right the first time!

Major Steps in Database Design

- 1. Requirements Analysis
  - Talk to the potential users! Understand what data is to be stored, and what operations and requirements are desired.
- 2. Conceptual Database Design
  - Develop a high-level description of the data and constraints (we will use the ER data model)
- 3. Logical Database Design
  - Convert the conceptual model to a schema in the chosen data model of the DBMS. For a relational database, this means converting the conceptual to a relational schema (logical schema).

Major Steps in Database Design

- 4. Schema Refinement
  - Look for potential problems in the original choice of schema and try to redesign.
- 5. Physical Database Design
  - Direct the DBMS into choice of underlying data layout (e.g., indexes and clustering) in hopes of optimizing the performance.
- 6. Applications and Security Design
  - How will the underlying database interact with surrounding applications.

Database Design Goals

- Reduce data redundancy.
- Provide stable data structures that can be readily changed with changing user requirements.
- Allow users to make ad hoc requests for data.
- Maintain complex relationships between data elements.
- Support a large variety of decision needs

Logical Data Modeling

- 3 types of data objects: Entities, Attributes, Relationships
- **Entities**: Are persons, places, or things about which data is to be, or is, gathered
- **Attributes** : Are the properties of entities. Examples are Names, Tax Numbers, Age, Status
- **Relationships** : Describe how entities relate to each other. Examples are Customers BUY Products, Persons WORK_ON Jobs

Entities

- Are persons, objects or events about which information is, or will be, recorded in the Database

- The designation of a ‘thing’ about which data is to be collected, stored or processed.

- Many of these Entities can be identified with Business Activities (e.g. suppliers, purchase orders, customer)

3 Steps in Design

- 1. The tables that belong in the database.
  - What are the entities?
- 2. The columns that belong in each table.
  - What are the properties?
- 3. How tables and columns interact with each other.
  - What do they have in common

Entity Relationships

Type	Shown As	Example
One to One	1:1	Book -----> Title
One to Many	1:N	Publisher ---->Books
Many to Many	N:N	Books <--->Authors

SQL Databases

- SQL databases use structured query language (SQL) for defining and manipulating data.
- SQL is one of the most versatile and widely-used options available, making it a safe choice and especially great for complex queries.

SQL Databases

- It can be restrictive. SQL requires that you use predefined schemas to determine the structure of your data before you work with it.
- In addition, all of your data must follow the same structure. This can require significant up-front preparation, and it can mean that a change in the structure would be both difficult and disruptive to your whole system.

NoSQL databases

- NoSQL databases have dynamic schemas for unstructured data
- Data is stored in many ways: They can be column-oriented, document-oriented, graph-based or organized as a KeyValue store.
- You can create documents without having to first define their structure
- Each document can have its own unique structure
- The syntax can vary from database to database, and
- You can add fields as you go.

Horizontal vs. Vertical Partitioning

- In a vertically-partitioned table, entire columns are separated out and put into new, distinct tables.
- The data held within one vertical partition is independent from the data in all the others, and each holds both distinct rows and columns.

Horizontal vs. Vertical Partitioning

Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

Vertical Partitions

VP1

CUSTOMER ID	FIRST NAME	LAST NAME
1	TAEKO	OHNUKI
2	O.V.	WRIGHT
3	SELDA	BAĞCAN
4	JIM	PEPPER

VP2

CUSTOMER ID	FAVORITE COLOR
1	BLUE
2	GREEN
3	PURPLE
4	AUBERGINE

Horizontal Partitions

HP1

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN

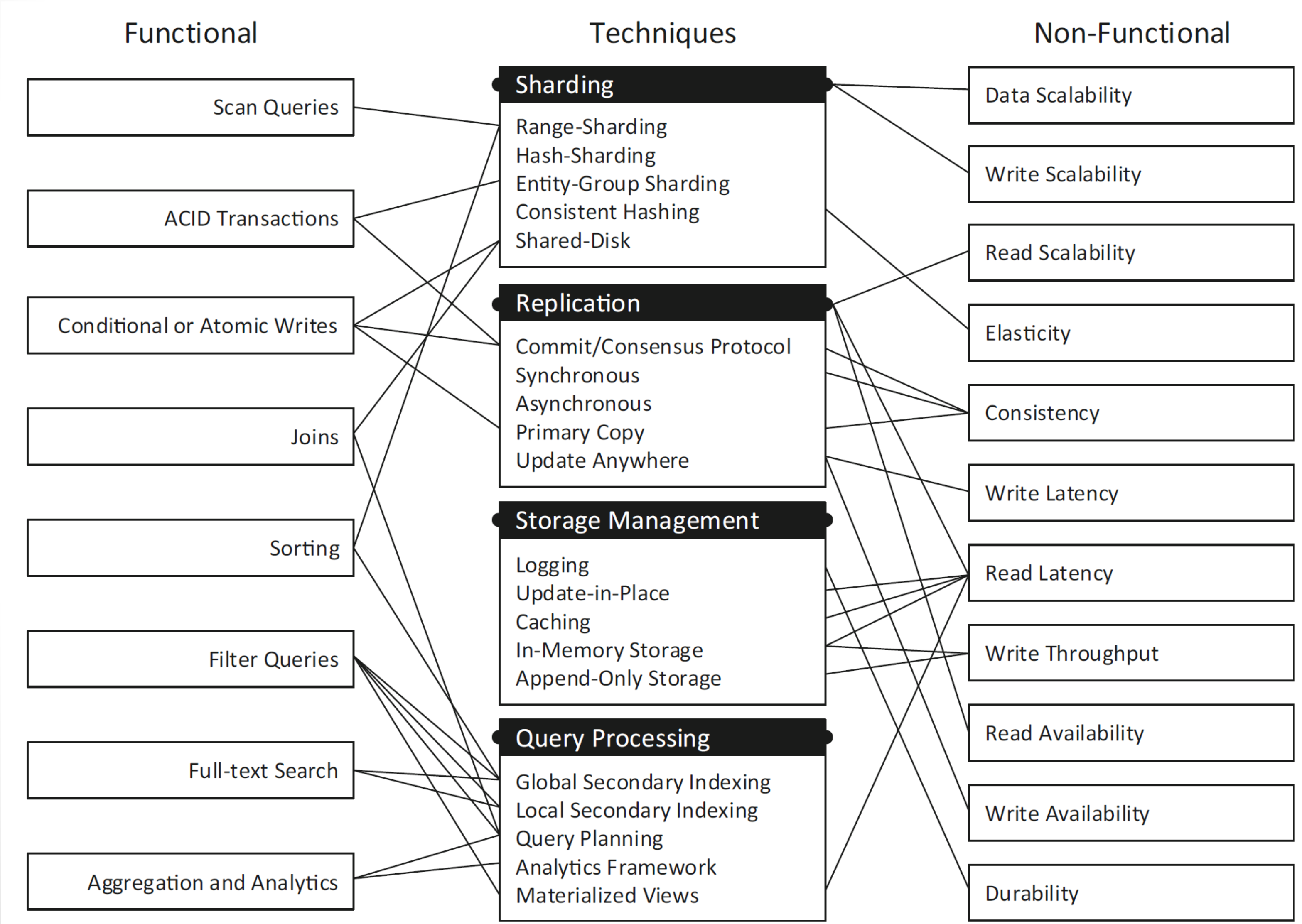
HP2

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

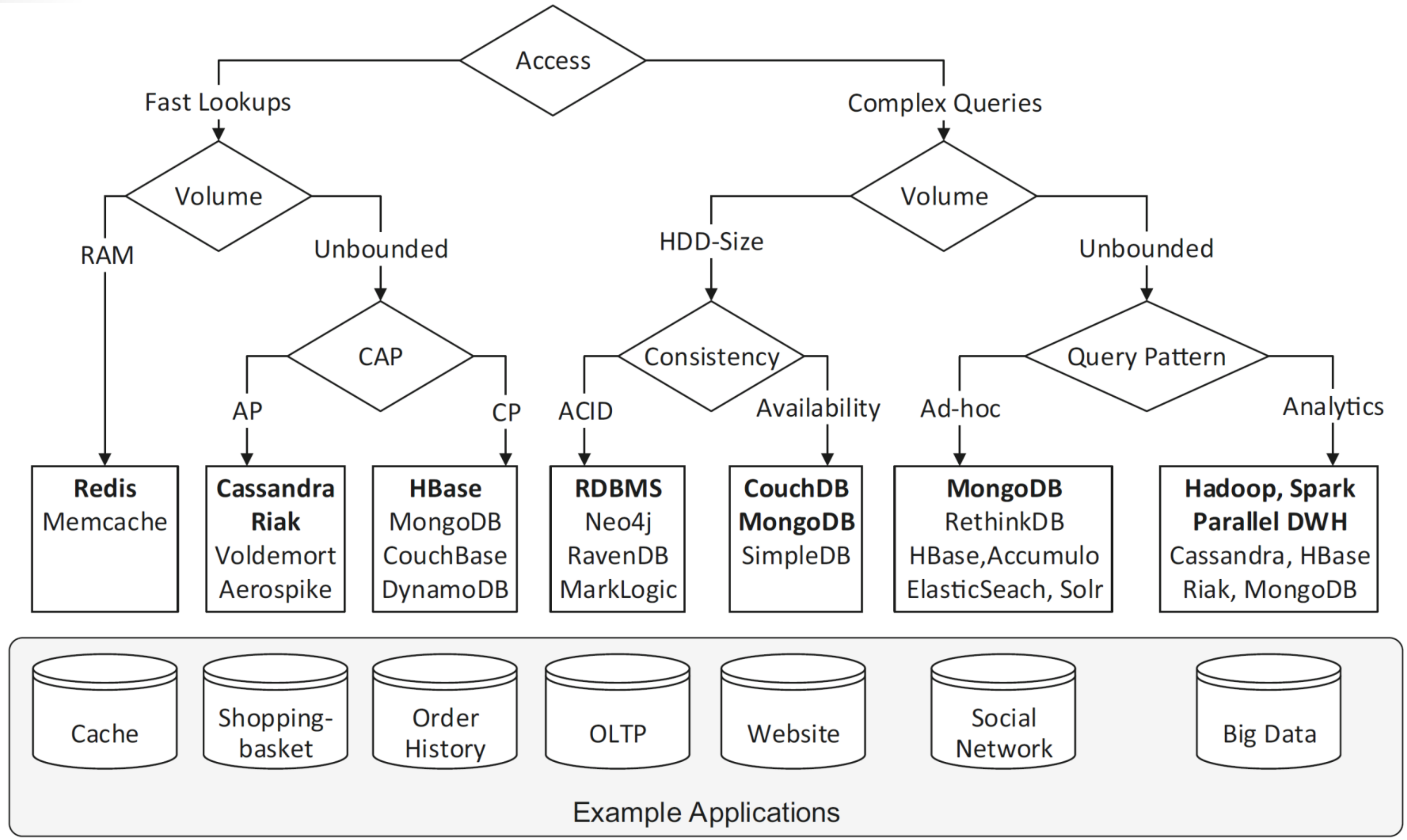
Sharding

- Sharding is a database architecture pattern related to horizontal partitioning: the practice of separating one table's rows into multiple different tables, known as partitions.
- Each partition has the same schema and columns, but also entirely different rows. Likewise, the data held in each is unique and independent of the data held in other partitions.
- Sharding can be hash-based, range-based, or dictionary-based

Database Techniques



Database Selection



Any Questions?

Your time is limited, don't waste it living someone else's life

Steve Jobs, Stanford University speech, 2005