Databases (Key: 1644)

Data Models

(NoSQL & OOP paradigm).

---------- Arellanes Conde Esteban ----------

Contents

1 Homework: Data Models 4

2 Object-Oriented Model 4

2.0.1 Features: 4

2.1 Advantages: 4

2.2 Disadvantages: 5

2.3 Use Cases: 5

3 NoSQL Model 6

4 Types of NoSQL Databases: 6

4.1 Key-Value Model 6

4.2 Document Model 6

4.3 Graph Model 7

4.4 Columnar Model 7

5 Entity-Relationship (E-R) Model **7**

5.1 Features: 7

6 References **10**

1 Homework: Data Models

Investigate the OO (object oriented) model object and NoSQL models.

- Object Oriented Model
- NoSQL Model (Key, Value, Documentals, Graphs)

2 Object-Oriented Model

The object-oriented model is based on representing data in the form of objects, similar to object-oriented programming. Each object stores data in the form of attributes and methods that allow manipulation of said data.

2.0.1 Features:

- Data is encapsulated in objects.
- Maintains class hierarchy and inheritance.
- Allows modeling of complex data relationships.
- Supports abstraction, encapsulation, and polymorphism.

2.1 Advantages:

- Natural integration with object-oriented languages like Java, Python, and C++.
- Facilitates code reuse and system maintenance.
- Handles complex data structures without the need for normalization.

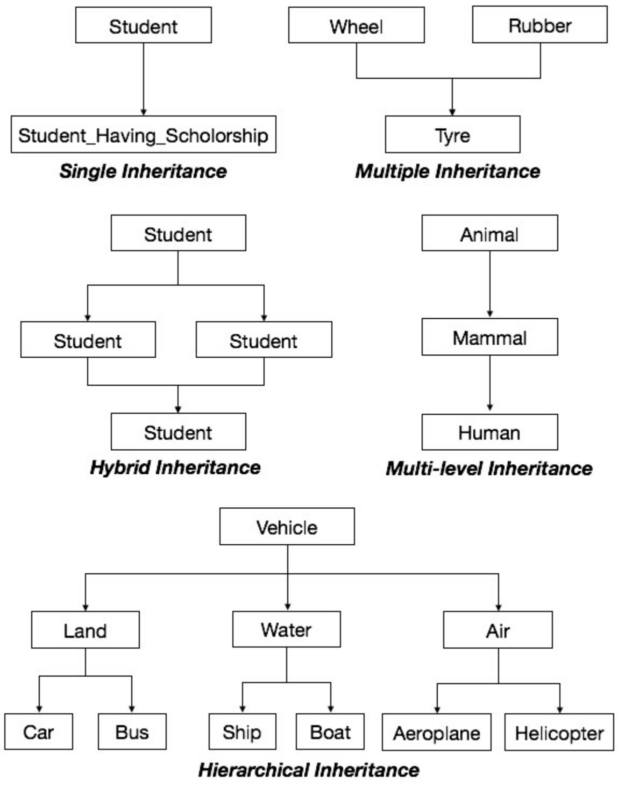

Figure 1: Types of Inheritance

2.2 Disadvantages:

- Can be slower compared to relational models due to object overhead.
- Not ideal for applications requiring intensive queries on large volumes of data.

2.3 Use Cases:

- Can be slower compared to relational models due to object overhead.
- Not ideal for applications requiring intensive queries on large volumes of data.
- CAD (computer-aided design) applications.

- Artificial intelligence systems and simulations.
- Business management software with complex data structures.

3 NoSQL Model

The NoSQL model encompasses several database subcategories designed to offer flexibility and scalability in data storage. They are characterized by not following the traditional relational model and can be more suitable for large volumes of distributed data.

4 Types of NoSQL Databases:

4.1 Key-Value Model

- Stores data in key-value pairs, similar to a dictionary in programming.
- **Advantages:** High scalability, fast searches, and simple storage.
- **Disadvantages:** Difficult to manage relationships between data.
- **Use Cases:** Data caching (Redis), session storage.

4.2 Document Model

Data is stored in JSON, BSON, or XML documents, allowing flexible structures.

- **Advantages:** High flexibility, horizontal scalability, and ease of handling semi-structured data.
- **Disadvantages:** Can be inefficient for very complex queries.
- **Use Cases:** Web and mobile applications, content management systems (MongoDB, CouchDB).

4.3 Graph Model

Represents data in nodes and edges, facilitating relationship modeling.

- **Advantages:** High efficiency in complex relationship queries.
- **Disadvantages:** Not optimal for massive write operations.
- **Use Cases:** Social networks, recommendation systems (Neo4j, ArangoDB).

4.4 Columnar Model

Stores data in columns instead of rows, optimizing analytical queries.

- **Advantages:** Fast access to large volumes of data.
- **Disadvantages:** Not suitable for highly relational transactions.
- **Use Cases:** Big Data and analytical systems (Cassandra, HBase).

5 Entity-Relationship (E-R) Model

The entity-relationship model is a conceptual approach for modeling relational databases. It represents data through entities (objects) and relationships between them.

5.1 Features:

Use of E-R diagrams to represent data structure. Clear definition of entities, attributes, and relationships. Primarily used in relational databases.

- **Advantages:**
  - Facilitates database design and normalization.

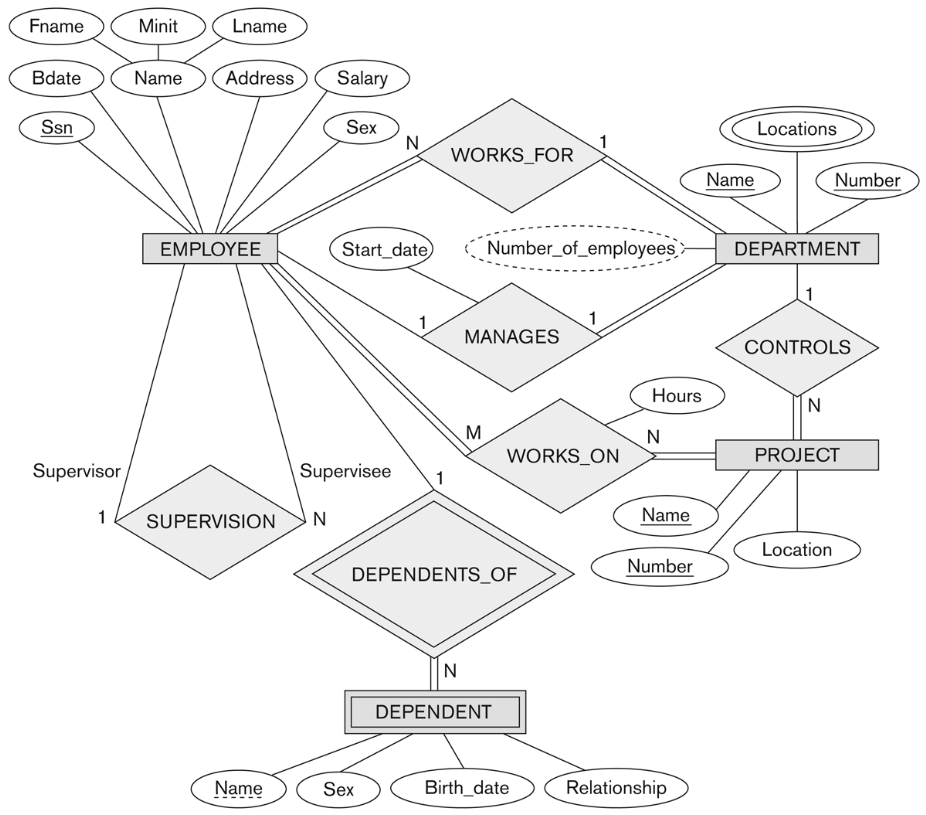

Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Figure 2: E-R Model Example

- Well-defined structure, ensuring data integrity.
- Compatible with SQL and relational databases.

- **Disadvantages:**
  - Can be complex for modeling unstructured data.
  - Not ideal for systems that require horizontal scalability.

- **Use Cases:**
  - Business and management applications.
  - Banking and inventory systems.
  - Applications requiring ACID transactions.

This document provides an overview of the main data models and their applications in different scenarios. Depending on the type of application, the most suitable model will be chosen.

6 References

Bibliographic References and Web Sites consulted:

* **[1] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th ed. Boston, MA, USA: Pearson, 2016.**
  `https://github.com/mariush2/tdt4145/blob/master/FundamentalsofDatabaseSystems(7thedition).pdf`

* **IEEE guide to software requirements specifications - IEEE Std 830-1984**
  `https://chumpolm.files.wordpress.com/2018/09/ieee-std-830-1984.pdf`