# DBMS

Peyman Najafi

Quera

*2021 January*

## Database Management System (DBMS):

standard methods to store and organize data

## Types:

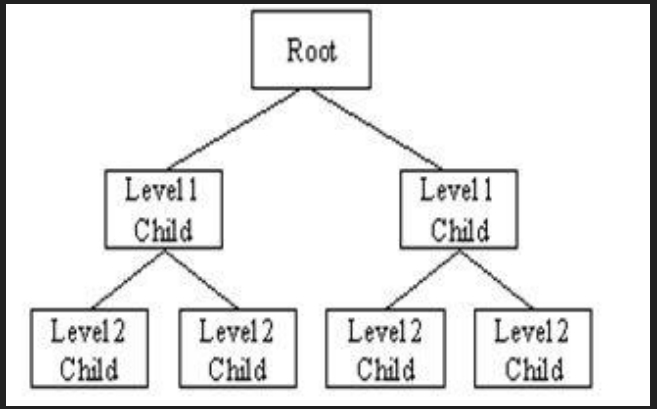
- Hierarchical databases
- Network databases
- Relational databases
- NoSQL databases
- Object-oriented databases
- Graph databases
- Document databases

# Hierarchical databases

- by IBM in 1960
- simple but inflexible
- banking and telecommunications industries
- popular examples:
  - IBM Information Management System (IMS)
  - Windows Registry
  - File systems
  - XML and XAML

# Hierarchical databases

- parent-children relationship node
- tree-like structure
- Child has only one parent
- Parent can have multiple children
- one-to-one and a one-to-many relationship



# Hierarchical databases

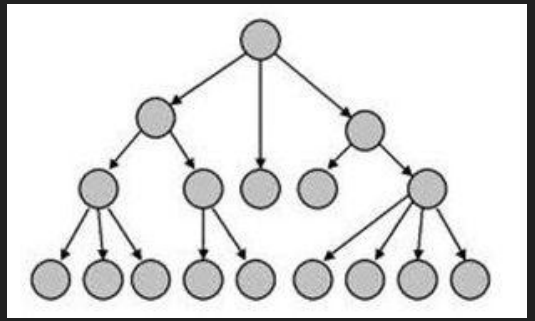
+ Data accessed and updated rapidly

- Child have only one parent

- Relationships between children are not permitted

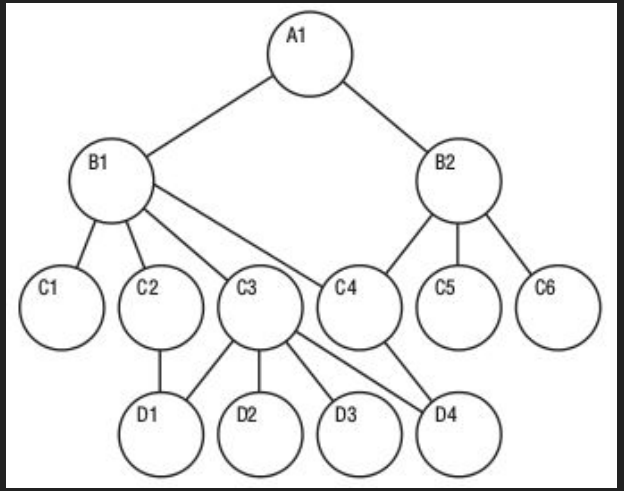
- Adding new field / record requires redefining database

- Can not support complex relationships (many-to-many)



# Network databases

- by Charles Bachman
- progression from the hierarchical database model
- is not restricted to being a hierarchy or lattice
- one node can have multiple parent
- children => members
- parents => owners



# Network databases

popular examples: IDS, IDMS, Raima, TurboIMAGE

+ Can support many-to-many relationships

- structure is quite complicated

- modify requires to understand it well (difficult to implement and maintain)

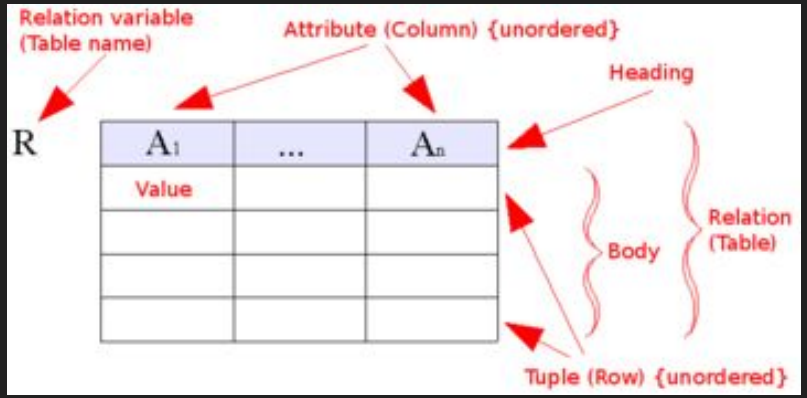
- Relationships between children are not permitted

# Relational databases

- by **Edgar Codd** in **1970**
- Based on **Relational Model**
- Using **SQL** (Structured Query Language) for querying and maintaining
- Popular examples: Postgre, Oracle, SQL Server, MySQL, SQLite, IBM DB2
- **+** Can be used with no or little training
- **+** Modify entries without specifying the entire body

# Relational model concepts

- Relation: (H, B) set
  - Header
    - Attribute
      - Name
      - Type
  - Body
    - Tuple
      - Value




- Key: unique attribute value in any tuple
- Cardinality: Tuples count
- Degre: Attributes count

# Relational databases

| Relational database term | SQL term |
| --- | --- |
| Tuple or Record | Row |
| Attribute or Field | Column |
| Relation or Base relvar | Table |
| Derived relvar | View or Result set |
| Cardinality | Rows count |
| Degre | Columns count |

# Relational databases

- Base and derived relations:
  - Base relations => stored => Tables
  - Derived relations => computing => Views or Queries
- Domain: set of possible values for a given attribute
- Constraints: *restrict the domain of an attribute*
  - Primary key
  - Foreign key
  - Stored procedures
  - Index

# Codd’s 12 rules

- Rule 0: The foundation rule
  - must be able to manage database entirely through its relational capabilities
- Rule 1: The information rule
  - All information is represented explicitly at the logical level and in exactly one way – by values in tables
- Rule 2: The guaranteed access rule
  - Each and every value is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name

# Codd’s 12 rules

- Rule 3: Systematic treatment of null values
  - Null values are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type
- Rule 4: Dynamic catalog based on the relational model
  - The database description is represented at the logical level in the same way as ordinary data

# Codd’s 12 rules

- Rule 5: The comprehensive data sublanguage rule

  supporting all of the following items:

  1. Data definition.
  2. View definition.
  3. Data manipulation (interactive and by program).
  4. Integrity constraints.
  5. Authorization.
  6. Transaction boundaries (begin, commit and rollback).

# Codd’s 12 rules

- Rule 6: The view updating rule
- Rule 7: Possible for high-level insert, update, and delete
- Rule 8: Physical data independence
- Rule 9: Logical data independence
- Rule 10: Integrity independence
- Rule 11: Distribution independence
- Rule 12: The non-subversion rule

# NoSQL databases

- not have predefined schemas
- make changes on the fly without affecting application
- Types of NoSQL Databases:
  - Column
  - Document
  - Graph
  - Key-value
  - Object databases

# NoSQL databases

- + more scalable and higher performance with high volumes of data
- + prefect candidate for rapidly changing development environments
- - no unified single model like SQL to work with NoSQL databases
- common NoSQL databases:
  - Network database
  - Graph database
  - Object database
  - Document database

# NoSQL databases

- Column Data Store (column-oriented DBMS or columnar DBMS)
  - stores data in columns, rather than rows
  - Imagine you need to list all names from a table based on an ID
  - Characteristics:
    - Keyspace (like schema in RDBMS)
    - Key Family (like table in RDBMS)
    - keyspace contains all the column families

# NoSQL databases

- Column Data Store (column-oriented DBMS or columnar DBMS)
  - Row Key: unique identifier for that row
  - Each column fields:
    - Name
    - Value
    - TimeStamp
  - Each row can contain a different number of columns
  - Each column can contain multiple rows

# NoSQL databases

- Key-Value Store
  - store dictionary type of data structure
  - Popular key-value databases
    - Redis
    - Memcache DB
    - Dynamo
    - ArangoDB
    - Berkeley DB

# NoSQL databases

10 popular NoSQL databases:

1. MongoDB
2. Elasticsearch
3. CouchDB
4. Couchbase Server
5. Amazon DocumentDB
6. CouchBase
7. ArangoDB
8. Informix
9. SAP HANA
10. Neo4j

# Object-Oriented databases

- OODBMs were created in 1980
- adds database functionality to object-oriented languages
- application and database development into a constant data model and language environment
- less code, natural data modeling, and easy maintain
- Each object contains two elements:
  1. A piece of data
  2. Methods

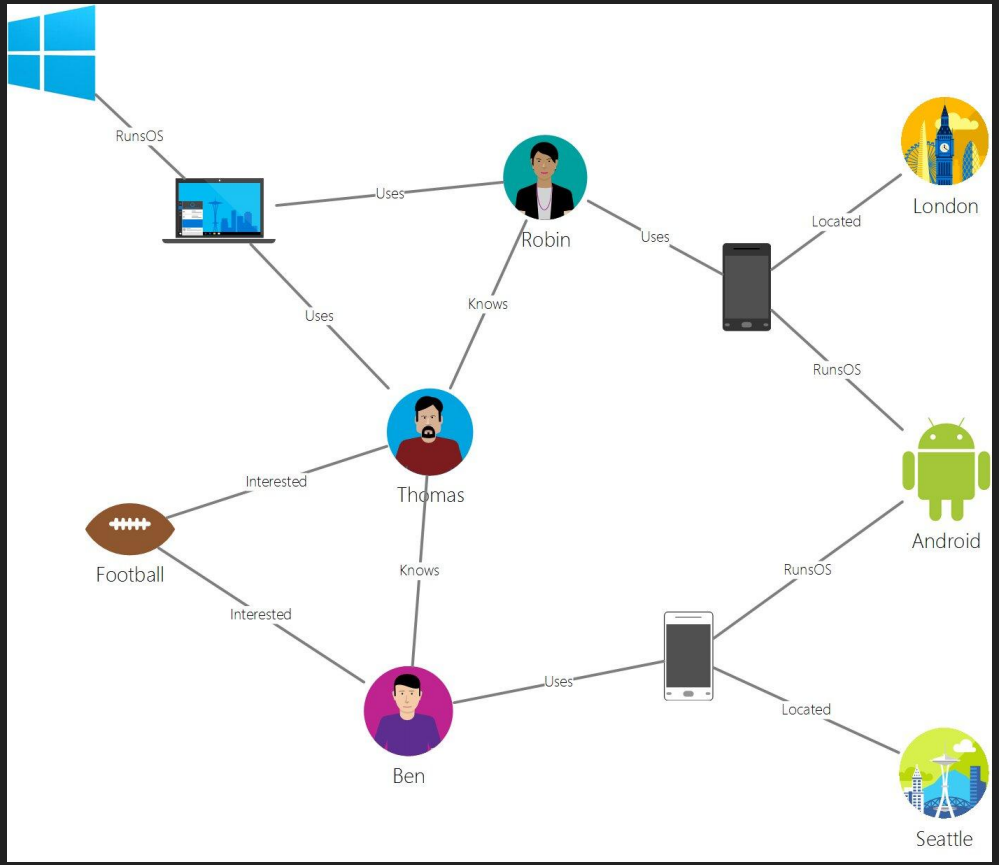
# Object-Oriented databases

- + Reading and mapping an object database data to the objects is direct without any API or OR tool
- + faster data access and better performance
- - Not many programming language support object databases
- - Not have a standard query language

# Graph databases

- are NoSQL databases
- use a graph structure for semantic queries
- data is stored in the form of nodes, edges, and properties
- **Node** represents an entity or instance (equivalent to record in relational databases)
- **Edge** represents a relationship
- **Properties** are information associated to nodes and edges

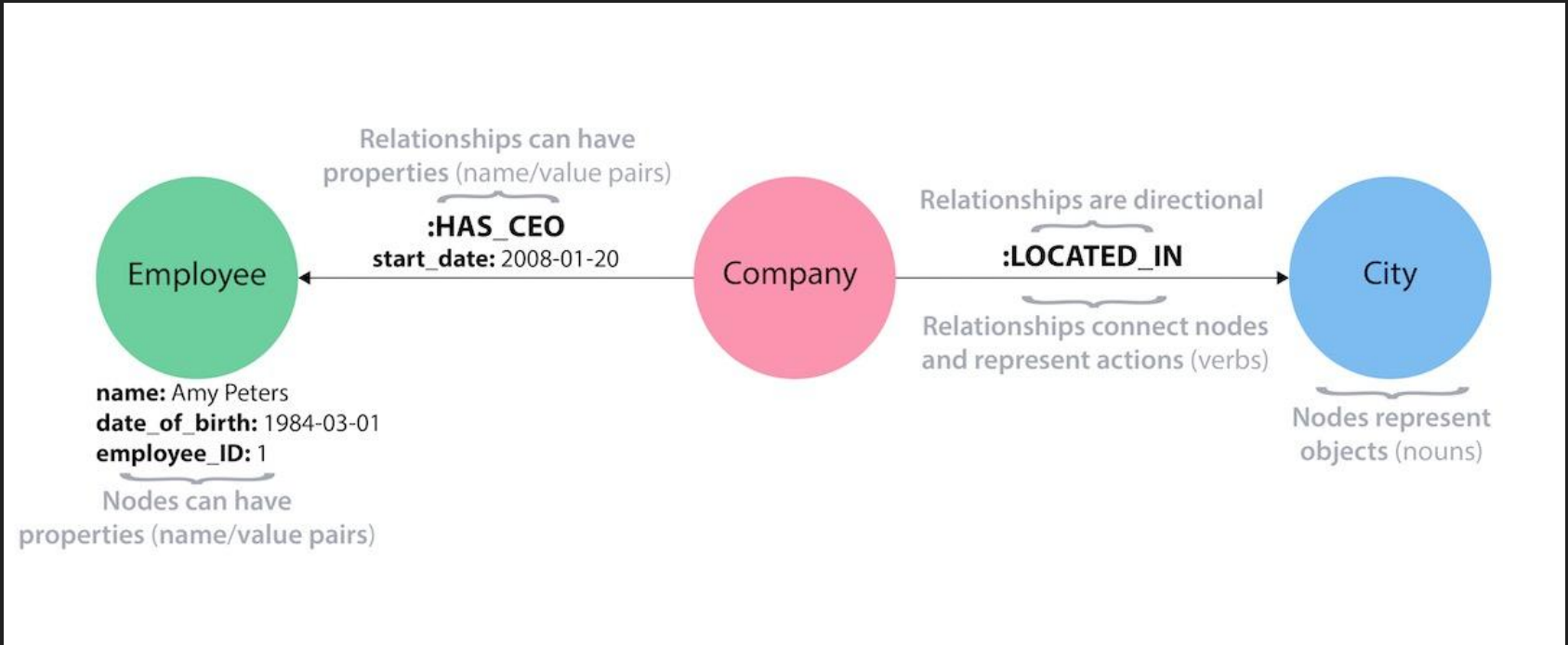
# Graph databases



# Graph databases

- Relationships can be intuitively visualized using graph
- useful for heavily inter-connected data
- similar to 1970s network model databases (but network-model databases operate at a lower level of abstraction)
- Not require a strict schema (unlike relational database)

# Graph databases



# Document databases

- are NoSQL databases
- Each document represents
  - Data
  - Relationship between other data elements
  - Attributes of data
- Store data in **Key-Value** form
- faster mechanism to store and search documents

# Resources

- DBMS types
- Hierarchical DBMS
- Network DBMS
- Relational database
- Codd's 12 rules
- NoSQL DBMS

# Thank You!

# Q&A