UMD DATA605: Big Data Systems

Lesson 1.3: Data Models

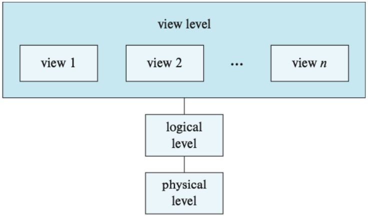
Instructor: Dr. GP Saggese, gsaggese@umd.edu

Data Models

- **Data modeling**
  - Represents and captures structure and properties of real-world entities
  - Abstraction: real-world $\rightarrow$ **representation**
- **Data model**
  - Describes how data is *represented* (e.g., relational, key-value) and *accessed* (e.g., insert operations, query)
  - Schema in a DB describes a specific data collection using a data model
- **Why need data model?**
  - Know data structure to write general-purpose code
  - Share data across programs, organizations, systems
  - Integrate information from multiple sources
  - Preprocess data for efficient access (e.g., building an index)

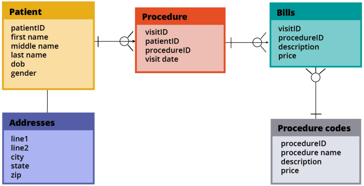
Multiple Layers of Data Modeling

- **Physical layer**
  - How is the data physically stored
  - How to represent complex data structures (e.g., B-trees for indexing)
- **Logical layer**
  - Entities
  - Attributes
  - Type of information stored
  - Relationships among the above
- **Views**
  - Restrict information flow
  - Security and/or ease-of-use



Data Models: Logical Layer

- **Modeling constructs**
  - Concepts to represent data structure
  - E.g.,
    - Entity types
    - Entity attributes
    - Relationships between entities
    - Relationships between attributes
- **Integrity constraints**
  - Ensure data integrity
    - Avoid errors and inconsistencies
    - E.g., field can't be empty, must be an integer
- **Manipulation constructs**
  - E.g., insert, update, delete data



Data Independence

- **Logical data independence**
  - Change data representation without altering programs
  - E.g., API abstracting backend
- **Physical data independence**
  - Change data layout on disk without altering programs
    - Index data
    - Partition/distribute/replicate data
    - Compress data
    - Sort data

Examples of Data Models

- **Some examples of data models**
  - Relational model (SQL)
  - Entity-relationship (ER) model
  - XML
  - Object-oriented (OO)
  - Object-relational
  - RDF
  - Property graph
- **Serialization formats** as data models
  - CSV
  - Parquet
  - JSON
  - Protocol Buffer
  - Avro/Thrift
  - Python Pickle

Good Data Models

- **Good data model** should be:
  - Expressive
    - Capture real-world data
  - Easy to use
  - Perform well
- **Trade-off between characteristics**
  - E.g., more powerful models
    - Represent more datasets
    - Harder to use/query
    - Less efficient (e.g., more memory, time)
- **Evolution of data models** captures data structure
  - Structured data $\rightarrow$ Relational DBs
  - Semi-structured web data $\rightarrow$ XML, JSON
  - Unstructured data $\rightarrow$ NoSQL DBs

A Brief History of Databases (Early 1960s)

- **1960s: Early beginning**
  - Computers become attractive technology
  - Enterprises adopt computers
  - Applications use own data stores
    - Each application has its own format
    - Data unavailable to other programs
- **Database**: term for "shared data banks" by multiple applications
  - Define data format
  - Store as "data dictionary" (schema)
  - Implement "database management" software to access data
- **Issues**
  - How to write data dictionaries?
  - How to access data?
  - Who controls the data?
    - E.g., integrity, security, privacy concerns

A Brief History of Databases (1960s)

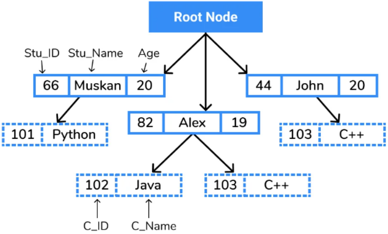
- **1960s, Hierarchical and Network Model**
  - Connect records of different types
  - Example: connect accounts with customers
  - Network model aimed for generality and flexibility
- IBM's IMS Hierarchical Database (1966)
  - Designed for Apollo space program
  - Predates hard disks
  - Used by over 95% of top Fortune 1000 companies
  - Processes 50 billion transactions daily, manages 15 million GBs of data
- **Cons**
  - Exposed too much internal data (structures/pointers)
  - Leaky abstraction

Relational, Hierarchical, Network Model

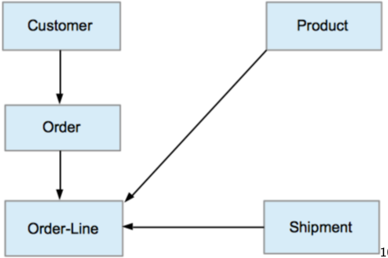
- **Relational model**
  - Data as tuples in relations
  - SQL

Customer ID	Tax ID	Name	Address	[More fields...]
1234567890	555-5512222	Ramesh	323 Southern Avenue	...
2223344556	555-5523232	Adam	1200 Main Street	...
3334445563	555-5533323	Shweta	871 Rani Jhansi Road	...
4232342432	555-5325523	Sarfaraz	123 Maulana Azad Sarani	...

- **Hierarchical model**
  - Tree-like structure
    - One parent, many children
    - Connected through links
  - XML DBs resurgence in 1990s



- **Network model**
  - Graph organization
    - Multiple parents and children
  - Graph DBs resurgence in 2010s

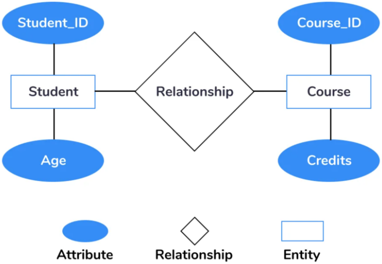


A Brief History of Databases (1970s)

- **1970s: Relational model**
  - Set theory, first-order predicate logic
    - Ted Codd developed the Relational Model
  - Elegant, formal model
    - Provided data independence
    - Users didn't worry about data storage, processing
  - High-level query language
    - SQL based on relational algebra
  - Notion of normal forms
    - Reason about data and relations
    - Remove redundancies
- **Influential projects**
  - INGRES (UC Berkeley), System R (IBM)
  - Ignored IMS compatibility
- **Debates**:
  - Relational Model vs Network Model proponents

Entity-Relationship Model

- **Entity-Relationship Model**
  - Proposed in 1976 by Peter Chen
- Describes knowledge as:
  - **Entities**: Physical or logical objects, “Nouns”
  - **Relationships**: Connections between entities, “Verbs”
- Map ER model to relational DB
  - Entities, relationships $\rightarrow$ tables



A Brief History of Databases (1980s)

- **1980s: Relational model acceptance**
  - SQL standard due to IBM's backing
  - Enhanced relational model
    - Set-valued attributes, aggregation
- **Late 80's**
  - Object-oriented DBs
    - Store objects, not tables
    - Overcome *impedance mismatch* between languages and databases
  - Object-relational DBs
    - User-defined types
    - Combine object-oriented benefits with relational model
  - No expressive difference from pure relational model

Object-Oriented

- OOP is a data model
  - Object behavior described through data (fields) and code (methods)
- **Composition**
  - `has-a` relationships
  - E.g., `Employee` class has an `Address` class
- **Inheritance**
  - `is-a` relationships
  - E.g., `Employee` class derives from `Person` class
- **Polymorphism**
  - Code executed depends on the class of the object
  - One interface, many implementations
  - E.g., `draw()` method on a `Circle` vs `Square` object, both descending from `Shape` class
- **Encapsulation**
  - E.g., private vs public fields/members
  - Prevents external code from accessing inner workings of an object

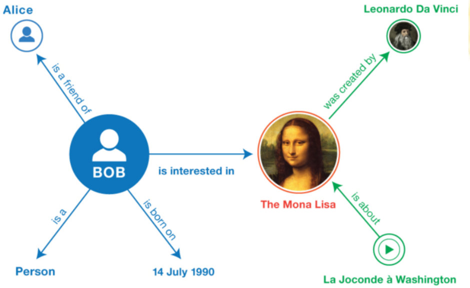
A Brief History of Databases (1990s)

- **Late 90's-today**
- Web/Internet emerges
- XML: eXtensible Markup Language
  - For *semi-structured* data
  - Tree-like structure
  - Flexible schema

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <CATALOG>
    <CD>
      <TITLE>Empire Burlesque</TITLE>
      <ARTIST>Bob Dylan</ARTIST>
      <COUNTRY>USA</COUNTRY>
      <COMPANY>Columbia</COMPANY>
      <PRICE>10.90</PRICE>
      <YEAR>1985</YEAR>
    </CD>
    <CD>
      <TITLE>Hide your heart</TITLE>
      <ARTIST>Bonnie Tyler</ARTIST>
      <COUNTRY>UK</COUNTRY>
      <COMPANY>CBS Records</COMPANY>
      <PRICE>9.90</PRICE>
      <YEAR>1988</YEAR>
    </CD>
    ...
```

Resource Description Framework

- Aka RDF
- **(subject, predicate, object) triple**
- E.g.,
  - `Subject`=sky
  - `Predicate`=has-the-color
  - `Object`=blue
- Maps to a labeled, directed multi-graph
  - More general than a tree
- Stored in:
  - Relational DBs
  - Dedicated "triple-stores" DBs



SCIENCE ACADEMY

Property Graph Model

- **Graph**:
  - Vertices and edges
  - Properties for each edge and vertex
- **Stored in**:
  - Relational DBs
  - Graph DBs