[639] Database Management Systems (DBMS)

Meenakshi Syamkumar

Learning Objectives

Differentiate	the three types of data
Understand	the fundamentals of Database Management Systems (DBMS)
Define	schema along with its abstraction types
Identify	various types of DBMS
Differentiate	OLAP and OLTP systems
Identify	primary keys and foreign keys

Types of data

- Structured data
- Semi-structured data
- Unstructured / raw data data

Types of data: Structured

- High degree of organization and indexing
- Data conforms to a strict schema
  - predefined, tabular format
- Data relationships are defined
- Easy to query, search, aggregate, and analyze
- Examples:
  - Tables in databases
  - Spreadsheets

Types of data: Semi-Structured

- Some structure in data but not rigid, implicit, and irregular
- Flexible schema
  - contains tags, markers to separate semantic elements
  - enforces hierarchy of fields within the data
- Easy to store and parse compared to unstructured data
- Examples:
  - JSON
  - HTML
  - XML
  - Email

Types of data: Unstructured

- Data does not have any specific format or pre-defined structure
- Storage
  - objects
  - file systems
- Hard to query data but extremely valuable data
- Examples:
  - Text documents: word files, PDFs
  - Multimedia: images, audio, and video files
  - Social media: posts

DBMS

What is DBMS?

- Systematic way to store, retrieve, and manage data
- Ensures consistency, integrity, and security
- Key features:
  - Data storage and retrieval: organized storage (ex: tables, key-value stores, …)
  - Data security: authorized access
  - Concurrency: multiple simultaneous users
  - Data integrity: accuracy and consistency
  - Backup and recovery: data protection and restoration

Types of DBMS

Types of DBMS

- Relational DBMS (RDBMS):
  - Data storage: tables with predefined schema
  - Use case: transactional systems
  - Examples: MySQL, PostgreSQL, Oracle, Microsoft
- NoSQL DBMS:
  - Data storage:
    - key-value store (ex: Redis, DynamoDB)
    - document store (ex: MongoDB, CouchDB)
    - column-family store (ex: Cassandra)
    - graph databases (ex: Neo4j)
  - Use case: Big data, real-time applications

Types of DBMS

- Object-Oriented DBMS (OODBMS):
  - Data storage: objects (data and behavior)
  - Use case: complex data models, multimedia applications
  - Examples: db4o, ObjectDB
- Graph DBMS:
  - Data storage: graphs
  - Use case: social networks, recommendation engines
  - Example: Neo4j, Amazon Neptune
- In-Memory DBMS (IMDBMS):
  - Data storage: RAM
  - Use case: high-frequency trading, gaming
  - Example: Redis, VoltDB

Types of DBMS

- Columnar DBMS:
  - Data storage: columns instead of rows
  - Use case: data warehousing, large-scale analytics (aka big data)
  - Examples: Amazon Redshift, Snowflake
- Hierarchical DBMS (LEGACY):
  - Data storage: tree
  - Use case: legacy telecommunication systems
  - Example: IBM Information Management System and RDM mobile
- Network DBMS (LEGACY):
  - Data storage: many-to-many relationships
  - Use case: legacy banking systems
  - Example: Integrated Data Store (IDS)

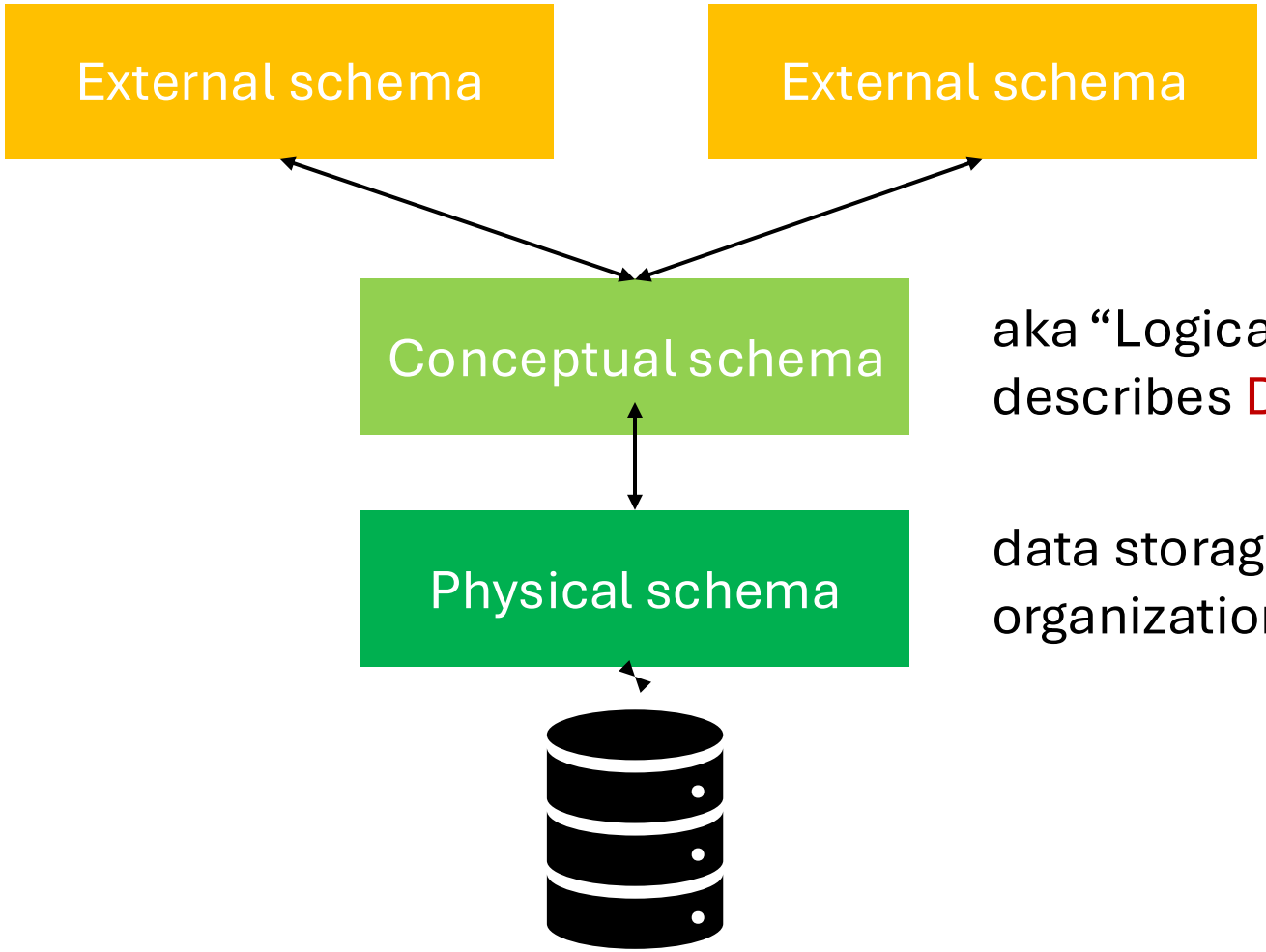
Schema

What is schema?

- Organizational blueprint
- Key aspects:
  - Structure and type definition
  - Relationships: primary keys, foreign keys
  - Constraints: unique, not null, default values
  - Indexes: fast searches, and sorts
  - Views: subset of data
  - Stored procedures: predefined queries

Levels of abstraction

application visibility



aka “High-level schema” describes high-level design

aka “Logical schema” describes Data Model

data storage details, file organization, indexing

OLTP vs OLAP

OLTP (Online Transaction Processing)

- Managing and processing transactional data (row-based)
- Example transactions: updates, inserts, and deletes
- Data Integrity:
  - ACID (Atomicity, Consistency, Isolation, Durability)
- Examples scenarios: banking systems, e-commerce, airline reservation systems

OLAP (Online Analytics Processing)

- Managing and processing data for analytics (column-based)
- Example transactions: aggregations, data mining
- Data Integrity:
  - very low support
- Examples scenarios: business intelligence, data warehousing

RDBMS

RDBMS example

- Consider building a learning management system (**LMS**):
  - Students
  - Courses
  - Professors

  } *Entities*

  - Who enrolls to what
  - Who teaches what

  } *Relationships*

Modeling the LMS: External schema

- External schema (Views)
  - Entities
    - Students (sid: *string*, name: *string*, gpa: *float*)
    - Courses (cid: *string*, cname: *string*, credits: *int*)
    - Enrolled (sid: *string*, cid: *string*, grade: *string*)
    - Professors (fid: *string*, name: *string*, cid: *string*, did: *string*), …
  - Relationships:
    - *Students* enroll in *Courses*
    - *Professors* teach *Courses*
    - *Professors* belong to *Departments*
    - …

Modeling the LMS: Logical schema

- Logical schema
  - Students (sid: *string*, name: *string*, gpa: *float*)
  - Courses (cid: *string*, cname: *string*, credits: *int*)
  - Enrolled (sid: *string*, cid: *string*, grade: *string*)
  - Professors (fid: *string*, name: *string*, cid: *string*, did: *string*)
  - …

Modeling the LMS: Logical schema

sid	name	gpa
101	Alice	3.2
123	Bob	3.8

Students

cid	cname	credits
544	Big Data	4
639	Data Mgmt	2

Courses

Relationships

sid	cid	grade
123	544	A
101	639	A

Enrolled

fid	name	cid	did
356	Tyler	544	22
439	Meena	639	22

Professors

Modeling the LMS: Logical schema

sid	name	gpa
101	Alice	3.2
123	Bob	3.8

Students

cid	cname	credits
544	Big Data	4
639	Data Mgmt	2

Courses

Primary keys

sid	cid	grade
123	544	A
101	639	A

Enrolled

fid	name	cid	did
356	Tyler	544	22
439	Meena	639	22

Professors

Modeling the LMS: External schema

sid	name	gpa
101	Alice	3.2
123	Bob	3.8

Students

cid	cname	credits
544	Big Data	4
639	Data Mgmt	2

Courses

Foreign keys

sid	cid	grade
123	544	A
101	639	A

Enrolled

fid	name	cid	did
356	Tyler	544	22
439	Meena	639	22

Professors

Professors table schema

- Table name: Professors
- Column names: fid, name, cid, did
- Column data types: fid: *string*, name: *string*, cid: *string,* did*: string*
- Column constraints*:*
  - Primary key*:* column or set of columns that uniquely identifies rows (ex: fid)
  - Foreign key*:* relates one table to another (ex: cid, did)
  - *NULL:* column can have empty values
  - *NOT NULL:* column cannot have empty values

Data independence

- Applications do not need to worry about how the data is structured and stored
- Logical data independence:
  - protection from changes in the logical structure of the data
  - not worrying about:
    - can we add a new entity or attribute without rewriting the application
- Physical data independence:
  - protection from physical layout changes
  - not worrying about:
    - which disks are the data stored in?
    - what type of indexing is used?