Motivation and basic concepts

What is a database?

Omnipresence of databases

Databases are in the core of almost all computer systems

- library catalogs, booking systems, ERP-systems
- multi-medial databases
- geographical information systems

Basic operations on databases

- Create
- Read
- Update
- Delete

Database - definition

A database is an organized collection of related data:

1. which represents aspects of the part of real world, called the **universe of discourse**
2. data has some **inherent meaning**
3. database is designed to be used for a **specific purpose**

Database is an organized collection of inter-related data that models an aspect of the real-world

Data storage requirements

- scalability
- performance even with big data sets
- ensuring data integrity
- concurrent access
- structural changes need to be easy to implement
- data protection

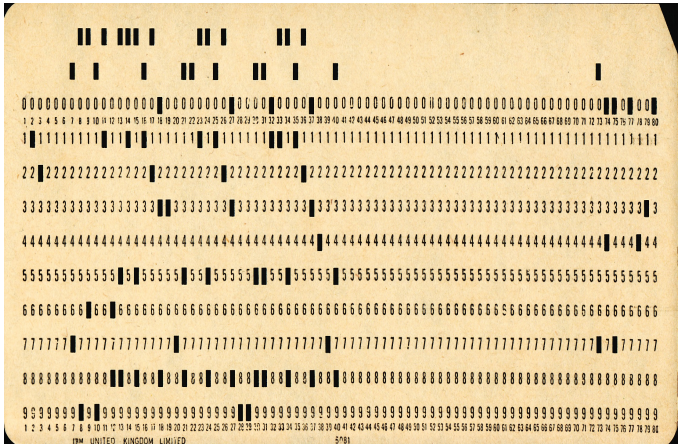
Data storage requirements 2

- Massive
- Persistent
- Safe
- Multiuser
- Convenient
- Efficient
- Reliable

Historical development of data stores

1950s punched cards and magnetic tapes

- access is only sequential
- general sensitivity of mediums
- little capacity
  - capacity of a punch card 60 - 100 Bytes



Historical development of data stores

1960s and 1970s: hard drives

– saving data in logical units
– direct access
– average access times
– data manipulation significantly easier

Question

Which functions should have an application for storing data of customers?

How would you implement these functions?

Historically first approaches

1. traditional **file processing** approach (flat file applications)
    - each user has necessary files for its application (there can be distinguished different departments) such as:
        - accounting office
        - grade reporting office
        - …
2. **database** approach
    - single repository

File processing approach (storing data as a list)

StudentInfo

Student name	Class	Course num	Course name	Department
John Brown	1	CS1310	Database systems	CS
Christine Smith	2	MATH245	Discrete Mathematics	MATH
Christine Smith	2	CS1310	Database systems	CS
Leslie Connor	1	CS2450	C++ programming	CS

Problems with using lists (anomalies):

- **Insertion problems**
  - inserting a new course with no students
- **Deletion problems**
  - if one deletes the row with Leslie Connor in our example than the data about the course “C++ programming” will also be lost
- **Update problems**
  - updating the name of a course requires updating it on every place

Flat file application

– each entity has a separate file and the application working with that file

Issues

– someone overwrites the year of students class with an invalid string

– the lack of centralized application - everyone can change the file in some strange ways.

– inserting new data, enrolling students on a particular course

– finding a particular record (scanning all records)?

– creating a new application that uses the same database in a new programming language

– two threads try to write to the same file at the same time

Flat file storage

- databases in 60s and 70s
- tight coupling between physical and logical level
- knowing which queries would be executed before designing the database is necessary
- each user implements files needed for a specific software application
- data definition is a part of the application programs
- changes of the structure often require changing of all programs

File based data storage

Issues with storing data in simple lists

- redundancy and inconsistency
- limited access
- problems with multi-user access
- data loss in crashes
- integrity breach
- security problems
- development costs

File based data storage lacks in flexibility.

Example: Part of the university database

The StudentInfo table from the previous page is split into different tables

– What is the difference between this and the previous case?

Student

Stud_num	Student name	Class
1	John Brown	1
2	Christine Smith	2
3	Leslie Connor	1

Course

Course num	Course name	Department
CS1310	Database systems	CS
MATH245	Discrete Mathematics	MATH
CS2450	C++ programming	CS

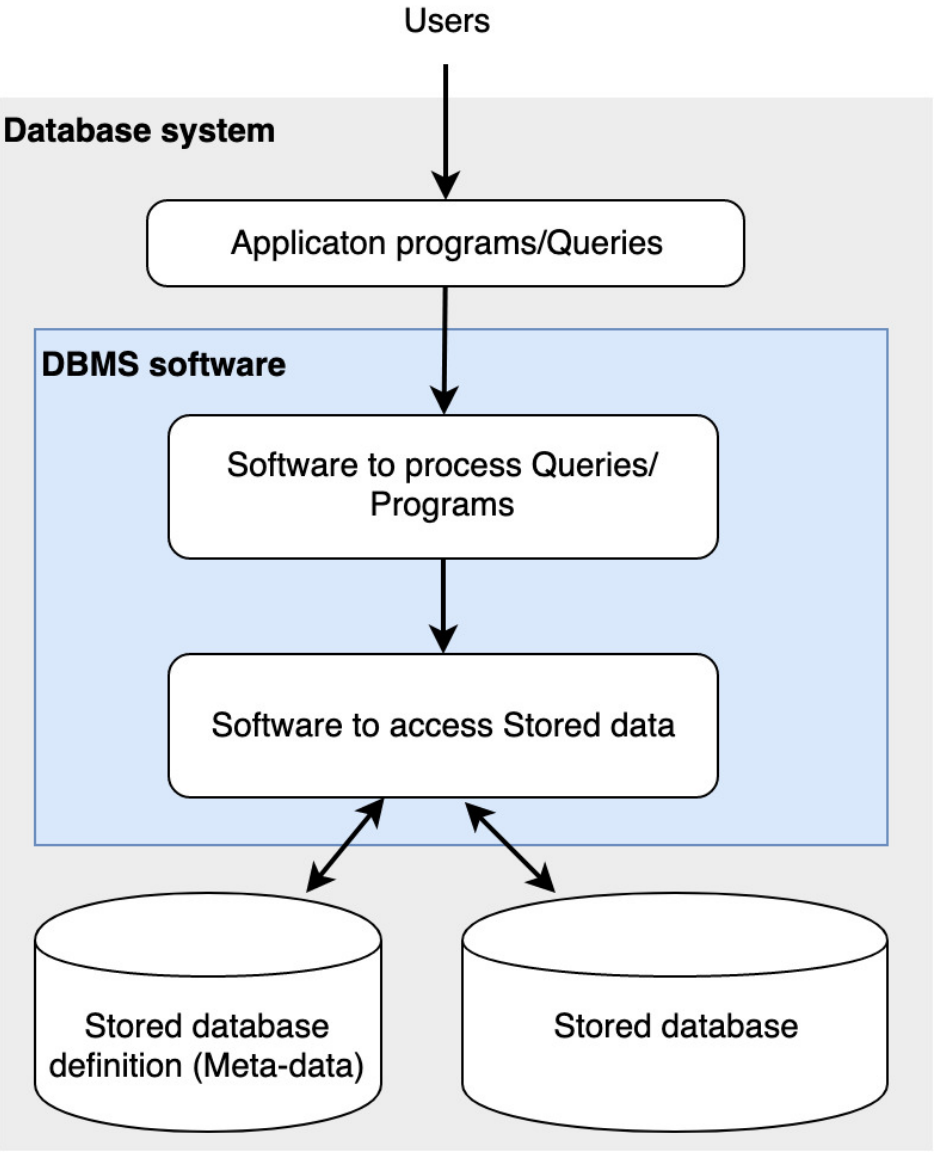
Enrolled

Stud_num	Course num
1	CS1310
2	MATH245
2	CS1310
3	CS2450

Database functions according to Codd

– Integration
– Operations: CRUD
– Catalog (Data dictionary / Meta-data)
– User views
– Consistency
– Data protection
– Transactions
– Synchronization
– Security

Database system environment



Database system environment

- Database
- **Database Management System (DBMS)**
  - is a *general-purpose software* that enables defining, constructing, manipulating and sharing databases
  - querying, update, administration of databases
  - allows users to interact with more databases
  - examples:
    - PostgreSQL
    - Oracle database
    - MSSQL
    - MySQL
    - DB2
    - …
  - Metadata - database definition and information about data is stored by DBMS
- Applications - they access the data in databases exclusively using DBMS

Database approach

1. Self describing nature of a database system
   – use of a catalog to store the database description (meta-data)
2. Insulation between programs and data, and data abstraction
   – **data abstraction** - using *data model*
      – **program-data independence**
      – **program-operation independence**
3. Support of multiple views of the data
   – different users require different perspectives of the database
4. Sharing of data and multiuser transaction processing - concurrency control
   – online transaction processing (OLTP)
   – online analytical processing (OLAP)

Key people

- database administrators
  - authorizing access
  - monitoring database use
- database designers
  - understanding requirements and choosing appropriate structures
  - creating design to meet requirements
- end users
  - querying, updating, generating report
  - parametric (naive) users - use standard transactions
    - canned transactions
  - casual users - need different information and use a query language
  - sophisticated users
- system analysts and application programmers
- database system designers and implementers
- tool developers

Basic ideas of the database concept

– controlling redundancy
– restricting unauthorized access
– providing persistent storage for program objects
– providing storage structures for efficient query processing
– providing backup and recovery
– providing multiple user interfaces
– representing complex relationships among data
– enforcing integrity constraints
– permitting interfaces and actions using rules

Situations inadequate for DBMS

Sometimes DBMS represents overhead for the system

- embedded systems with limited storage capacity
- systems with with no multiple-user access
- many computer-aided design(CAD) tools
- communication and switching systems
- GIS systems

Modern Database system architecture

Basic client/server DBMS architecture

- client module - designed to work on user workstations
  - application programs
  - user-friendly interfaces
- server module - handles data storage, access, search and other functions

Data abstraction and data model

Data abstraction is generally a way to **suppress** details of data organization and storage to **highlight essential features** for an improved understanding of data

Data model - a collection of concepts used for **describing the structure** of a database in order to achieve data abstraction

- data model includes
  - description of data types, relations and constraints over data
  - description of basic operation for specifying retrieval and updates on the database
- Data model (database model) is also related to the work of Edgar Codd
- **database model** is a type of data model which describes how a database is structured and used

Categories of data models

1. Conceptual (high-level) data models
   - entity-relationship model
   - object-data model
2. Representational data models
   - **Relational data model**
   - Relational Algebra
3. Physical (low-level) data models
   describe haw data is stored as files
   - record formats, record orderings, access paths
     - index is an example of the access path

Database Models (Data Models)

- **Relational**
- Kay/Value
  - Redis
- Graph
  - Neo4J
- Document
  - MongoDb
- Column-family (HBase, Big Table)
- Array/Matrix
- Hierarchical -IBM IMS (IBM Information Management System)
- Network
- …

Database schema

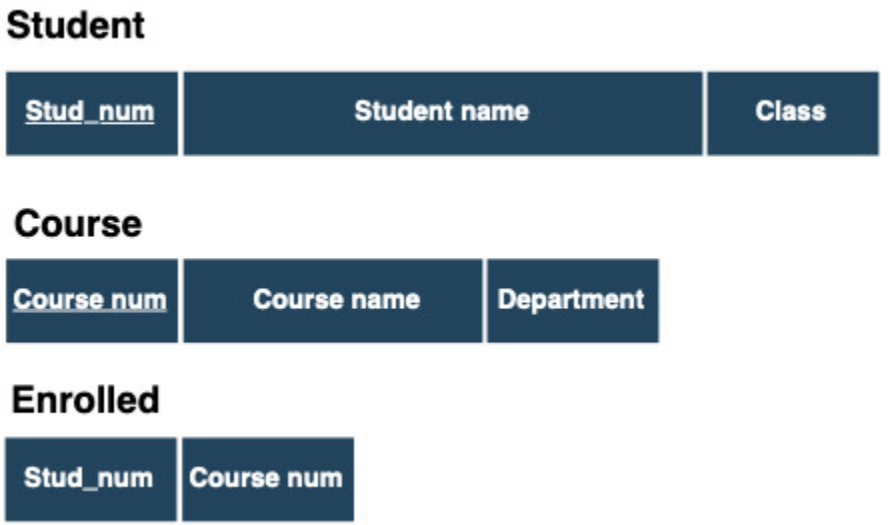
Data models differentiate between

– description of the database (**database schema**)

– database itself

Database schema is the description of a database which is specified during database design and is not expected to change frequently.

Database schemas are mostly represented using **schema diagrams**



An object of the schema is called a **schema construct** (Student, Course)

Database state

Database state (snapshot)

– represents the data in the database at a particular moment in time

– is also called current state of **instances** (occurrences)

An instance represent an individual entity (record)

Student

Stud_num	Student name	Class
1	John Brown	1
2	Christine Smith	2
3	Leslie Connor	1

Instance

It is important to distinguish between database schema and database state

DBMS and database schema

DBMS system has a role to

- maintain every database state as a **valid state**
- store schema constructs together with constraints

Database schema is also called **intesion**

- it doesn't change so often

Database state is called **extension** of the schema

- it is changed constantly

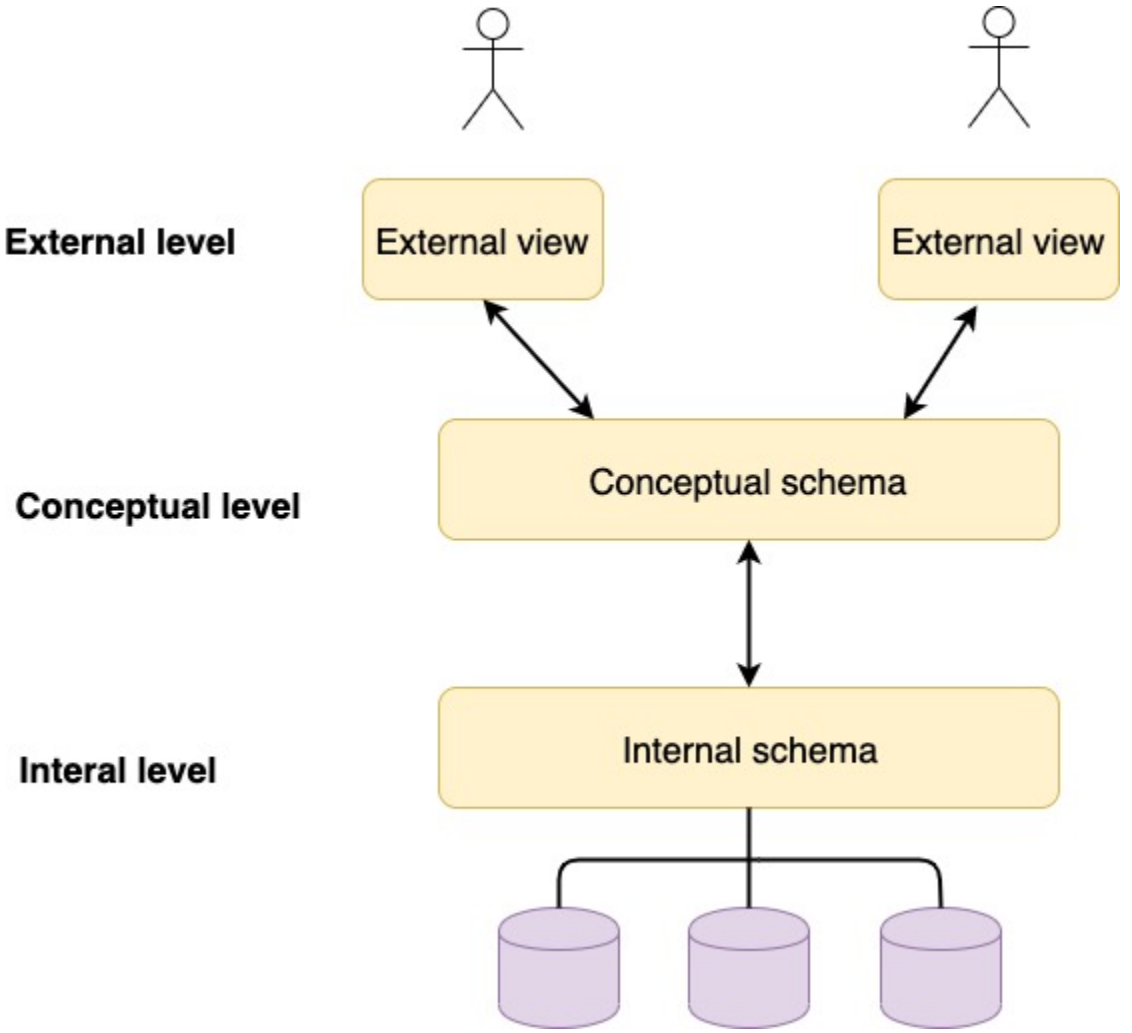
ANSI/SPARC architecture (Three-Schema Architecture)

separates user applications from the physical database

- **internal level**
  - internal schema describes the physical storage structure
  - internal schema uses physical data model to describe details
- **conceptual level**
  - conceptual schema describes entities, relationships, data types, user operations and constraints
  - described by representational data models
- **external (view) level**
  - external schemas (user views) describe the part of the database for a particular user group (typically used representational data models)

Most DBMS support three-schema to some extent

ANSI/SPARC architecture (Three-Schema Architecture)



Data independence

Data independence is a *capacity to change* the schema at one level of a database system without having to change schema at the next higher level

- **Logical data independence**
  - capacity to change the conceptual schema without changing external schemas or application programs
  - examples: adding or removing constraints, record types or data items
- **Physical data independence**
  - capacity to change the internal schema without having to change the conceptual schema
  - examples: reorganizing physical files (creating additional access structures)
  - capacity to change

Logical independence is harder to achieve

- **mappings** between layers - to achieve data independence

Database languages

Data Definition Language (DDL)

- DBMS use DDL to design both conceptual and external schemas
- DBMS have a DDL compiler
- used by the DBA and database designers

Data Manipulation Language (DML)

- used to manipulate the database (insertion, retrieval, deletion and modification of data)

Current DBMS use practically the same language

- used for conceptual schema definition, view definition and data manipulation
- combination of DDL, VDL, DML, constraint specification, schema evolution and other features

Data Manipulation Language (DML)

DML in DBMS can be:

- high level
  - such as **Structured Query Language (SQL)**
    SELECT * FROM Student WHERE Class = "1"
  - entered interactively or embedded in a general-purpose programming language
- low level (procedural)
  - must be embedded in a general-purpose programming language
  - retrieves individual objects from database and process them separately
  - commands GET UNIQUE , GET NEXT, …

Query languages in general

Query languages can be categorized as:

1. Non-Procedural (declarative) languages
   - Relational-Calculus (formal query language)
     - {s | Student(s) ^ s.Class = “1**}
     - proposed first by Codd in 1970s
     - SQL
2. Procedural
   - Relational Algebra (formal)

Updates are also considered as a part of a query language in DBMS

DBMS interfaces

Different database users use different ways to interact with a database

- **Menu-Based Interfaces for Web clients**
  - parametric (naive users) use mostly user-friendly interfaces
  - helpful not to memorize specific commands
- **Form-based interfaces**
  - display forms with form field that should be filled and are than matched in DBMS to retrieve matching data
- **Graphical user interfaces**
  - to display schema in diagrammatic form
- Specific interfaces for parametric users
- Natural language interfaces
- other interfaces
  - speech input and output, …

Classification of DBMSs

Main criterion used for the classification is the data model

- Relational Database Management Systems (RDBMS)
  - **relational data model**
- object DBMSs
  - **object data model**
  - not widespread
- Object-relational DBMSs
  - more data models: relational, object, object-relational, hierarchical,
- Native XML DBMSs -
- Hierarchical DBMSs
  - **hierarchical data model**
  - example: IMS (IBM)
- Network DBMSs
  - **network data model**
  - examples: IMAGE (Hewlett-Packard), SUPRA (Cincom)

Review questions

- What is a data model?
- What is the difference between a database schema and a database state?
- What is difference between logical and physical data independence?
- Define the term database management system?
- What are disadvantages of the traditional file processing storage?
- In which scenarios features of DBMS are not desirable?
- Explain components of the database system architecture?