Chapter 1: Introduction

- Purpose of Database Systems
- View of Data
- Data Models
- Data Definition Language
- Data Manipulation Language
- Transaction Management
- Storage Management
- Database Administrator
- Database Users
- Overall System Structure

Database Management System (DBMS)

- Collection of interrelated data
- Set of programs to access the data
- DBMS contains information about a particular enterprise
- DBMS provides an environment that it both *convenient* and *efficient* to use

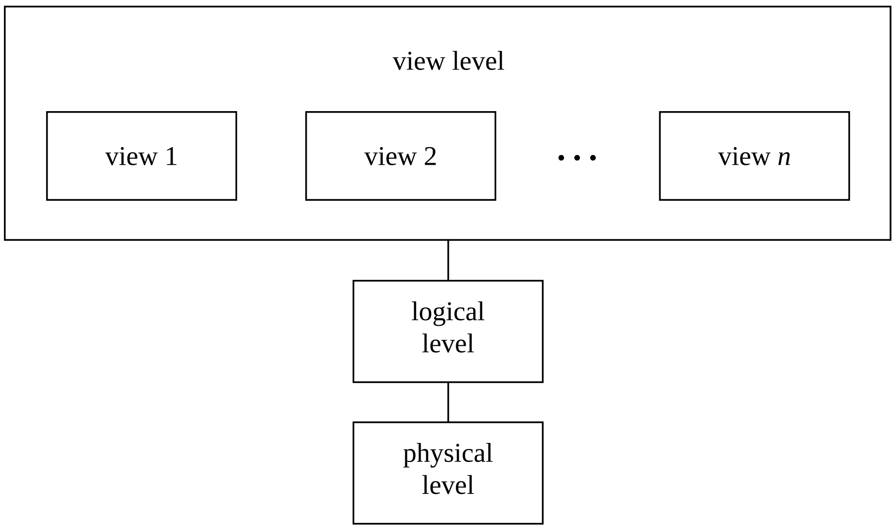
Purpose of Database Systems

Database management systems were developed to handle the following difficulties of typical file-processing systems supported by conventional operating systems.

- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation – multiple files and formats
- Integrity problems
- Atomicity of updates
- Concurrent access by multiple users
- Security problems

View of Data

An architecture for a database system



Levels of Abstraction

- Physical level: describes how a record (e.g., *customer*) is stored.
- Logical level: describes data stored in database, and the relationships among the data.

```
type customer = record
                    name : string;
                    street : string;
                    city : integer;
                end;
```

- View level: application programs hide details of data types. Views can also hide information (e.g. salary) for security purposes.

Instances and Schemas

- Similar to types and variables in programming languages
- Schema – the logical structure of the database (e.g., set of customers and accounts and the relationship between them)
- Instance – the actual content of the database at a particular point in time

Data Independence

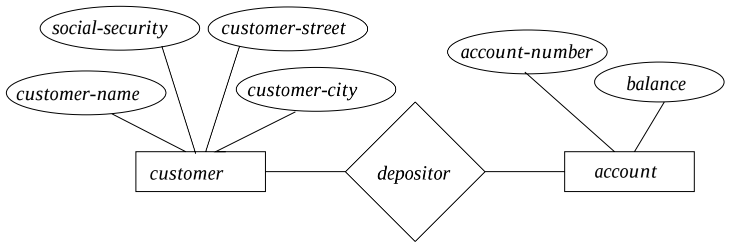
- Ability to modify a schema definition in one level without affecting a schema definition in the next higher level.
- The interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.
- Two levels of data independence:
  - Physical data independence
  - Logical data independence

Data Models

- A collection of tools for describing:
  - data
  - data relationships
  - data semantics
  - data constraints
- Object-based logical models
  - entity-relationship model
  - object-oriented model
  - semantic model
  - functional model
- Record-based logical models
  - relational model (e.g., SQL/DS, DB2)
  - network model
  - hierarchical model (e.g., IMS)

Entity-Relationship Model

Example of entity-relationship model



Relational Model

Example of tabular data in the relational model:

customer-name	*social-security*	*customer-street*	*customer-city*	*account-number*
Johnson	192-83-7465	Alma	Palo Alto	A-101
Smith	019-28-3746	North	Rye	A-215
Johnson	192-83-7465	Alma	Palo Alto	A-201
Jones	321-12-3123	Main	Harrison	A-217
Smith	019-28-3746	North	Rye	A-201

account-number	*balance*
A-101	500
A-201	900
A-215	700
A-217	750

Data Definition Language (DDL)

- Specification notation for defining the database schema
- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
- *Data storage and definition* language – special type of DDL in which the storage structure and access methods used by the database system are specified

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
- Two classes of languages
  - Procedural – user specifies what data is required and how to get those data
  - Nonprocedural – user specifies what data is required without specifying how to get those data

Transaction Management

- A *transaction* is a collection of operations that performs a single logical function in a database application
- Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g. power failures and operating system crashes) and transaction failures.
- Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Storage Management

- A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible for the following tasks:
  - interaction with the file manager
  - efficient storing, retrieving, and updating of data

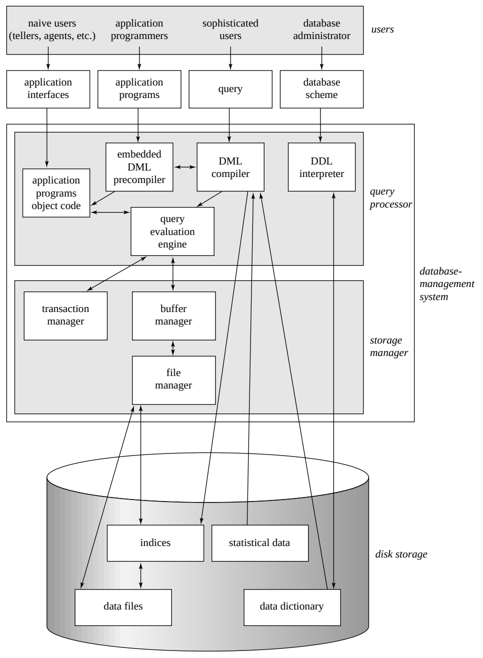
Database Administrator

- Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs.
- Database administrator's duties include:
  - Schema definition
  - Storage structure and access method definition
  - Schema and physical organization modification
  - Granting user authority to access the database
  - Specifying integrity constraints
  - Acting as liaison with users
  - Monitoring performance and responding to changes in requirements

Database Users

- Users are differentiated by the way they expect to interact with the system
- Application programmers – interact with system through DML calls
- Sophisticated users – form requests in a database query language
- Specialized users – write specialized database applications that do not fit into the traditional data processing framework
- Naive users – invoke one of the permanent application programs that have been written previously

Overall System Structure


naive users (tellers, agents, etc.)
application programmers
sophisticated users
database administrator
users
application interfaces
application programs
query
database scheme
embedded DML precompiler
DML compiler
DDL interpreter
application programs object code
query evaluation engine
query processor
database-management system
transaction manager
buffer manager
file manager
storage manager
indices
statistical data
data files
data dictionary
disk storage