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 is both *convenient* and *efficient* to use.
- Database Applications:
  - ★ Banking: all transactions
  - ★ Airlines: reservations, schedules
  - ★ Universities: registration, grades
  - ★ Sales: customers, products, purchases
  - ★ Manufacturing: production, inventory, orders, supply chain
  - ★ Human resources: employee records, salaries, tax deductions
- Databases touch all aspects of our lives

Purpose of Database System

- In the early days, database applications were built on top of file systems
- Drawbacks of using file systems to store data:
  - ★ Data redundancy and inconsistency
    - ✔ Multiple file formats, duplication of information in different files
  - ★ Difficulty in accessing data
    - ✔ Need to write a new program to carry out each new task
  - ★ Data isolation — multiple files and formats
  - ★ Integrity problems
    - ✔ Integrity constraints (e.g. account balance > 0) become part of program code
    - ✔ Hard to add new constraints or change existing ones

Purpose of Database Systems (Cont.)

- Drawbacks of using file systems (cont.)
  - ★ Atomicity of updates
    - ✔ Failures may leave database in an inconsistent state with partial updates carried out
    - ✔ E.g. transfer of funds from one account to another should either complete or not happen at all
  - ★ Concurrent access by multiple users
    - ✔ Concurrent accessed needed for performance
    - ✔ Uncontrolled concurrent accesses can lead to inconsistencies
      - – E.g. two people reading a balance and updating it at the same time
  - ★ Security problems
- Database systems offer solutions to all the above problems

Levels of Abstraction

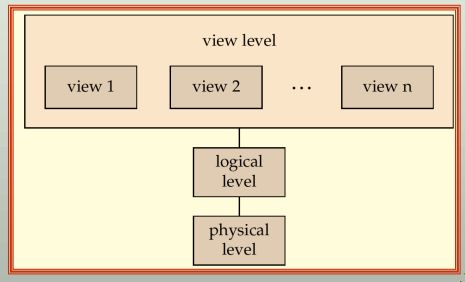
- Physical level describes how a record (e.g., customer) is stored.
- Logical level: describes data stored in database, and the relationships among the data.

```
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.

View of Data

An architecture for a database system



Instances and Schemas

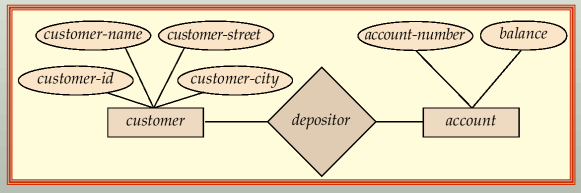
- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - ★ e.g., the database consists of information about a set of customers and accounts and the relationship between them)
  - ★ Analogous to type information of a variable in a program
  - ★ **Physical schema**: database design at the physical level
  - ★ **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
  - ★ Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - ★ Applications depend on the logical schema
  - ★ In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Models

- A collection of tools for describing
  - ★ data
  - ★ data relationships
  - ★ data semantics
  - ★ data constraints
- Entity-Relationship model
- Relational model
- Other models:
  - ★ object-oriented model
  - ★ semi-structured data models
  - ★ Older models: network model and hierarchical model

Entity-Relationship Model

Example of schema in the entity-relationship model



Entity Relationship Model (Cont.)

- E-R model of real world
  - Entities (objects)
    - E.g. customers, accounts, bank branch
  - Relationships between entities
    - E.g. Account A-101 is held by customer Johnson
    - Relationship set *depositor* associates customers with accounts
- Widely used for database design
  - Database design in E-R model usually converted to design in the relational model (coming up next) which is used for storage and processing

Relational Model

- Example of tabular data in the relational model

Attributes

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

A Sample Relational Database

customer-id	*customer-name*	*customer-street*	*customer-city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	*account-number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Data Definition Language (DDL)

- Specification notation for defining the database schema
  - E.g.
    create table *account* (
    account-number **char**(10),
    balance **integer**)
- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
  - database schema
  - Data *storage and definition* language
    - language in which the storage structure and access methods used by the database system are specified
    - Usually an extension of the data definition language

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- 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
- SQL is the most widely used query language

SQL

- SQL: widely used non-procedural language
  - E.g. find the name of the customer with customer-id 192-83-7465
    select *customer.customer-name*
    from *customer*
    where *customer.customer-id* = '192-83-7465'
  - E.g. find the balances of all accounts held by the customer with customer-id 192-83-7465
    select *account.balance*
    from *depositor*, *account*
    where *depositor.customer-id* = '192-83-7465' **and**
    depositor.account-number = *account.account-number*
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g. ODBC/JDBC) which allow SQL queries to be sent to a database

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
- Naïve users – invoke one of the permanent application programs that have been written previously
  - E.g. people accessing database over the web, bank tellers, clerical staff

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

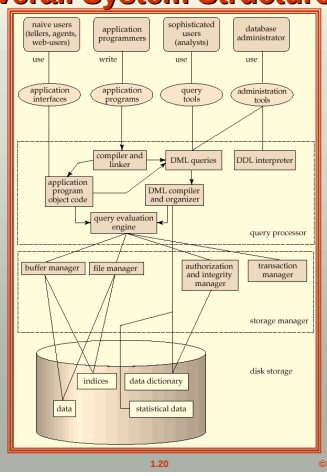
Transaction Management

- 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.

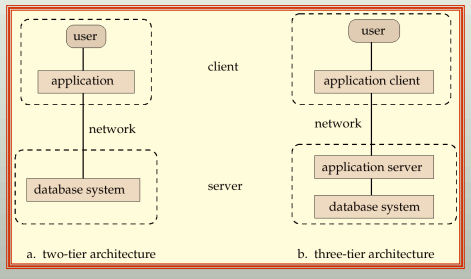
Storage Management

- 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 to the following tasks:
  - ★ interaction with the file manager
  - ★ efficient storing, retrieving and updating of data

Overall System Structure



Application Architectures



- **Two-tier architecture**: E.g. client programs using ODBC/JDBC to communicate with a database
- **Three-tier architecture**: E.g. web-based applications, and applications built using "middleware"