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 Systems

- 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

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., 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.

View of Data

An architecture for a database system

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

- **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

**Entity Relationship Model (Cont.)**

- 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

<table>
<thead>
<tr>
<th>Customer-id</th>
<th>customer-name</th>
<th>customer-street</th>
<th>customer-city</th>
<th>account-number</th>
</tr>
</thead>
<tbody>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>Alma</td>
<td>Palo Alto</td>
<td>A-101</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
<td>A-215</td>
</tr>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>Alma</td>
<td>Palo Alto</td>
<td>A-201</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
<td>A-217</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
<td>A-201</td>
</tr>
</tbody>
</table>

**Attributes**

**A Sample Relational Database**

<table>
<thead>
<tr>
<th>customer-id</th>
<th>customer-name</th>
<th>customer-street</th>
<th>customer-city</th>
</tr>
</thead>
<tbody>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>4 North St.</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>3 Main St.</td>
<td>Rye</td>
</tr>
<tr>
<td>677-90-0911</td>
<td>Hayes</td>
<td>123 Putnam Ave.</td>
<td>Harrison</td>
</tr>
<tr>
<td>182-73-6991</td>
<td>Turner</td>
<td>100 Main St.</td>
<td>Stanford</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>175 Park Ave.</td>
<td>Harrison</td>
</tr>
<tr>
<td>336-66-9999</td>
<td>Lindsey</td>
<td>72 North St.</td>
<td>Pittsfield</td>
</tr>
</tbody>
</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**

- 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 allows 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
  - Naive 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**

- **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”

**Application Architectures**

- [Diagram showing two-tier and three-tier architectures]