Introduction

Database System

- Computerized record-keeping system
  - Maintain information
  - Make information available on demand
- Must be convenient and efficient for storage and retrieval of data
- Must provide safety mechanisms
  - against system failure/crash
  - against nefarious users
  - against erroneous updates

Functions of a database management system

- Add new (empty) files to database
- Insert new data into existing files
- Retrieve data from existing files
- Update data in existing files
- Delete data from existing files
- Remove existing files (empty or not) from the database

A database system must account for

- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation
- Concurrent access anomalies
- Security problems
- Integrity problems

Jargon

- Relational table, or table
- Logical record, or record
- Fields of a logical record

The Database **cellar**

Bin	Wine	Producer	Year	Bottles	Ready	Comments
2	Chardonnay	Buena Vista	88	1	91	
3	Chardonnay	Louis Martini	89	5	90	
6	Chardonnay	Chappellet	87	4	91	Thanksgiving
11	Jo. Riesling	Jekel	89	10	92	
12	Jo. Riesling	Buena Vista	87	1	92	Late Harvest
16	Jo. Riesling	Sattui	87	1	89	very dry
21	Fume Blanc	Ch. St. Jean	88	4	91	Robt. Young
22	Fume Blanc	Robt. Mondavi	87	2	90	
25	Wh. Burgundy	Mirassou	86	6	89	
30	Gewurztraminer	Buena Vista	87	3	90	
43	Cab. Sauvignon	Robt. Mondavi	82	12	92	
50	Pinot Noir	Mirassou	82	3	90	Harvest
51	Pinot Noir	Ch. St. Jean	86	2	92	
64	Zinfandel	Mirassou	84	9	92	Anniversary
72	Gamay	Robt. Mondavi	85	2	90	

Retrieval:

```
select wine, bin, producer
from cellar
where ready = 91;
```

Result:

Wine	Bin	Producer
Chardonnay	2	Buena Vista
Chardonnay	6	Chappellet
Fume Blanc	21	Ch. St. Jean

Inserting new data:

```
insert
into cellar
values (53,'Pinot Noir','Saintsbury',87,1,93,'for Xmas');
```

Updating existing data:

```
update cellar
set bottles = 4
where bin = 3;
```

Deleting existing data:

```
delete
from cellar
where bin = 2;
```

Data Abstraction

Abstract view of the data

- Physical level – Block of consecutive storage locations

```
STORED_EMP   LENGTH=18
  PREFIX     TYPE=BYTE(6), OFFSET=0
  EMP#       TYPE=BYTE(6), OFFSET=6, INDEX=EMPX
  DEPT#      TYPE=BYTE(4), OFFSET=12
  PAY        TYPE=FULLWORD, OFFSET=16
```

- Conceptual level – Type definition (record)

```
EMPLOYEE
  EMPLOYEE_NUMBER    CHARACTER (6)
  DEPARTMENT_NUMBER  CHARACTER (4)
  SALARY             NUMERIC (5)
```

- View level – Visible part of records

PL/1	COBOL
`DCL 1 EMPP,`	`01 EMPC.`
`    2 EMP# CHAR(6),`	`   02 EMPNO PIC X(6).`
`    2 SAL FIXED BIN(31);`	`   02 DEPTNO PIC X(4).`

<u>Components of a Database System</u>

- Data
  - Small data ⇒ Single-user systems
  - Large data ⇒ Multi-user systems
    - Integrated data
      - Several distinct data files are unified
      - Any redundancy between data files must be eliminated
      - Consider the field `phones` in the data files `books` and `phones`
    - Shared data
      - Individual pieces of data can be shared among several users
      - *Concurrent* access
    - Any given user will be concerned with only a certain subset of the database
  - Persistent data
- Hardware
  - Secondary storage volumes
  - Processors and associated main memory
- Software
  - Database manager / DBMS
    - Most important component of the system
    - Handles all access requests from users
    - Shields database users from hardware-level details
  - Application development tools
  - Utilities
  - Design aids
  - Report writers

- Users
  - Application programmers
    - Write application programs using a high level language
    - Applications support end-users
  - End users
    - Interact with the system using on-line terminals
    - Can access the database using the application program or through a user interface, such as *interactive query language processor* (part of the database)
      - Forms-driven interface
      - Command-driven interface
  - Database administrator
    - The "boss"

Advantages of Database Systems

- Compactness
- Speed
- Less drudgery
- Current information
- Centralized control of data
  - Reduction in redundancy
  - Less inconsistency
    - Propagating updates – Maintains database consistency
  - Data sharing
  - Standards enforcement
    - Data interchange
    - Data naming
    - Documentation
  - Security restrictions
  - Data integrity
  - Balancing conflicting requirements

Data Models

Collection of conceptual tools to describe data, data relationships, data semantics, and consistency constraints

- Object-based Logical Models
  - Describe data at the conceptual and view levels
  - Provide flexible structuring capabilities
  - Allow explicit specification of data constraints
  - Entity-Relationship Model
    - Also known as E-R Model
    - *Entities*

      - collection of basic objects
      - distinguishable from each other by *attributes*
    - *Relationship*
      - association between the entities
    - *Entity set*
    - *Relationship set*
    - *Mapping cardinalities*
      - Instrument to represent constraints
      - Number of entities to which another entity can be associated via a relationship set
    - *E-R diagram*
      - Rectangles – entity sets
      - Ellipses – attributes
      - Diamonds – relationships among entity sets
      - Lines – link attributes to entity sets, and entity sets to relationships
    - Example – A bank
      - customer – name, street, city
      - account – number, balance
      - relationship – customer $\leftrightarrow$ account
  - Object-Oriented Model
    - Based on a collection of objects and methods
    - Object
      - Contains values stored in *instance variables*
      - Values themselves are objects
    - Method
      - Code to operate on the objects
    - Class
      - Type definition for objects
      - Group of objects containing same type of values
    - Message passing
      - Accessing of data belonging to different object
      - Internal and external part of object
    - Example – Bank
      - Object customer – instances name, street, and city
      - Object account – instances number and balance
      - Method pay_interest – compute and add interest to balance

- Record-based Logical Models
  - Describe data at the conceptual and view levels
  - Database structured in fixed-format records
  - Simplify physical level implementation
  - No direct representation of code in the database
  - Network Model
    - Data represented by collection of records
    - Relationships represented by *links* or pointers
    - Records organized as collections of arbitrary graphs
  - Hierarchical Model

- Similar to network model
- Records organized as collections of trees
- Commercial example – IMS

– Relational Model

- Dominant trend in the marketplace at present
- Collection of tables
- No pointers or links
- Relates records by the values they contain
- *Relational operators* generate new tables from the old ones
- Example data

Name	Street	City	Number
Lowery	Maple	Queens	900
Shiver	North	Bronx	556
Shiver	North	Bronx	647
Hodges	Sidehill	Brooklyn	801
Hodges	Sidehill	Brooklyn	647

Number	Balance
900	55
556	100000
647	105366
801	10533

Instances and Schemes

- Scheme
  - Overall design of the database
  - Storage area or type declarations
  - Physical level – Physical scheme
  - Conceptual level – Conceptual scheme
  - View level – Subscheme
- Instance
  - Collection of information at a particular moment
  - Values associated with the database
- Stored field
  - Smallest named unit of stored data
  - Database contains many *occurrences* (or *instances*) of each of several types of stored field
- Stored record
  - Named collection of several related stored fields
- Stored file
  - Named collection of all occurrences of one type of stored record

Data Independence

- In older systems, knowledge of data organization and access techniques was built into the application logic and code
  - Application involving "employee" file, indexed on "employee_number" field
  - Data dependent application will be aware of existence of index and the file sequence defined by index
  - Internal structure of application is built around this information

  - Impossible to change the storage structure or access technique without altering the application (like changing indexed file to a hash-addressed file)
- Why is data dependence bad?
  - Different applications need different view of the same data
  - DBA must have freedom to change storage structure or access technique in response to changing requirements, without having to modify existing applications
- Data independence hides implementation details from the users
- Immunity of applications to change in storage structure and access technique
- Distinction between stored record and logical record
- Physical data independence
  - Ability to modify the physical scheme without causing application programs to be rewritten
  - Binary storage vs `PICTURE` storage
- Logical data independence
  - Ability to modify the conceptual scheme without causing application programs to be rewritten

Data Definition Language (DDL)

- Set of definitions for database scheme
- Compilation of DDL statements results in *data dictionary*
- Data dictionary
  - contains *metadata* – data about data
  - consulted before accessing actual data

Data Manipulation Language (DML)

- Data manipulation
  - Retrieval of information stored in the database
  - Insertion of new information into the database
  - Deletion of information from the database
  - Modification of data stored in the database
- DML enables users to access or manipulate data
- Procedure DML
  - User should specify *what* data is needed and *how* to get it
- Nonprocedural DML
  - User should specify *what* data is needed but *not how* to get it
  - May not be as efficient as procedural DML
- *Query*
  - Statement to request retrieval of data

Database Administration

- Centralized control managed by data administrator and database administrator
- Data administrator must
  - be a senior management person
  - understand the data
  - understand organization needs
  - decides the data to be placed in the database
  - establishes policies to maintain and access data (data security policy)
- Database administrator
  - technical professional
  - oversees a staff of system programmers
  - creates the actual database
    * defines scheme using DDL
    * creates the data dictionary
    * defines storage structures and access methods
    * modifies scheme and physical organization definitions
  - implements technical controls for policy decisions
    * grants authorization for data access
  - ensures adequate performance of the system
    * defines the system structure for integrity constraints

Database Manager

- Data must be structured to minimize the need to move data between disk and memory
- Program module to provide interface between application programs and low-level data
- Interaction with file manager
  - Translate various query language commands to low-level file manager commands
- Intergrity enforcement
  - Check validity of updates against criterion specified by database administrator
- Security enforcement
  - Restrict access to users
- Backup and recovery
  - Detect hardware failures and restore database state
- Concurrency control
  - Maintain consistency of data by controlling interaction between concurrent users

Database Users

- Application programmers
- Sophisticated users
- Specialized users
- Naive users