CSE2004 – Database Management Systems

Text Books :

1.R. Elmasri & S. B. Navathe, **Fundamentals of Database Systems**, Addison Wesley, 7 th Edition, 2015

2.Raghu Ramakrishnan, **Database Management Systems**,Mcgraw-Hill,4th edition,2015

Reference Book

3.A. Silberschatz, H. F. Korth & S. Sudershan, **Database System Concepts**, McGraw Hill, 6th Edition 2010
4.Thomas Connolly, Carolyn Begg,” **Database Systems : A Practical Approach to Design, Implementation and Management**”,6th Edition,2012

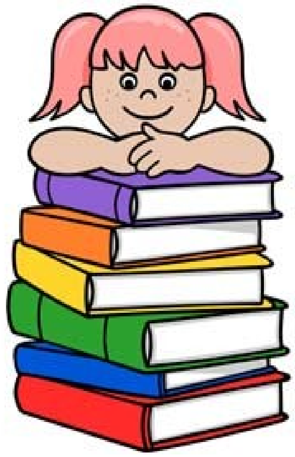
5.Shashank Tiwari ,“Professional NoSql”,Wiley ,2011

CSE2004 – Database Management Systems

VIT® UNIVERSITY (Estd. u/s 3 of UGC Act 1956) www.vit.ac.in Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Unit – 1 : DATABASE SYSTEMS CONCEPTS AND ARCHITECTURE

- ✓ History and motivation for database systems
- ✓ Characteristics of database approach
- ✓ Actors on the scene
- ✓ Workers behind the scene
- ✓ Advantages of using DBMS approach,
- ✓ Data Models, Schemas, and Instances
- ✓ Three-Schema Architecture and Data Independence
- ✓ The Database System Environment
- ✓ Centralized and Client/Server Architectures for DBMSs
- ✓ Classification of database management systems

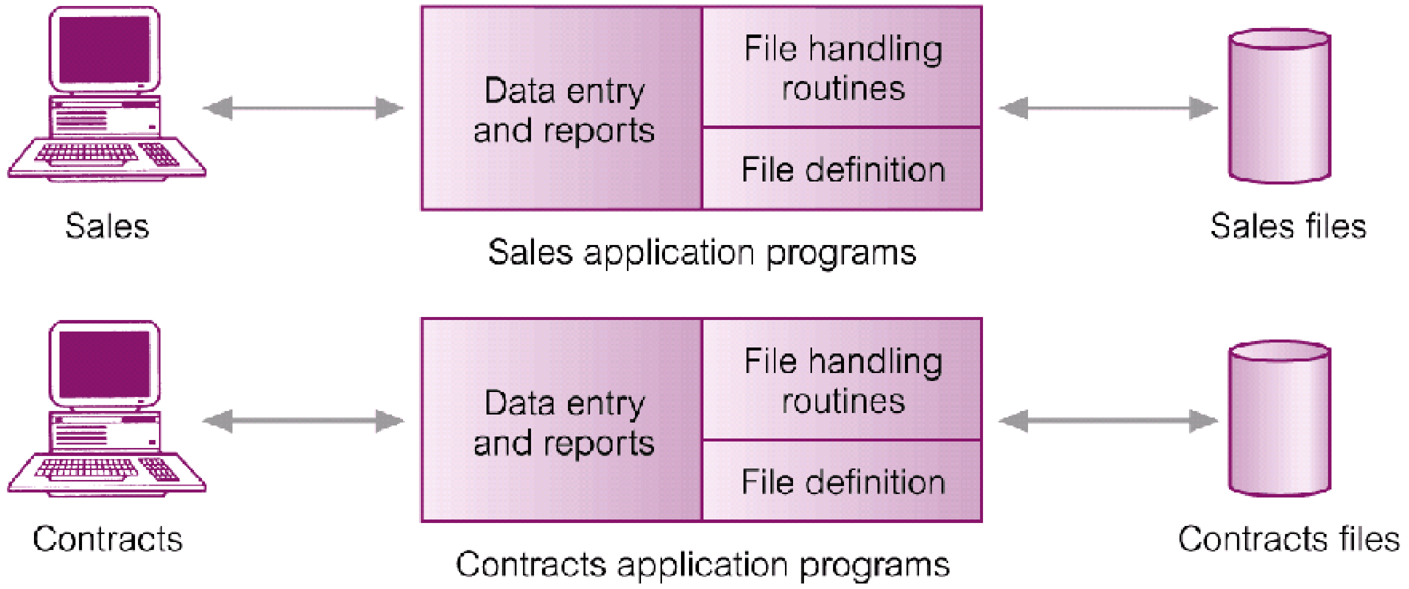


Outline

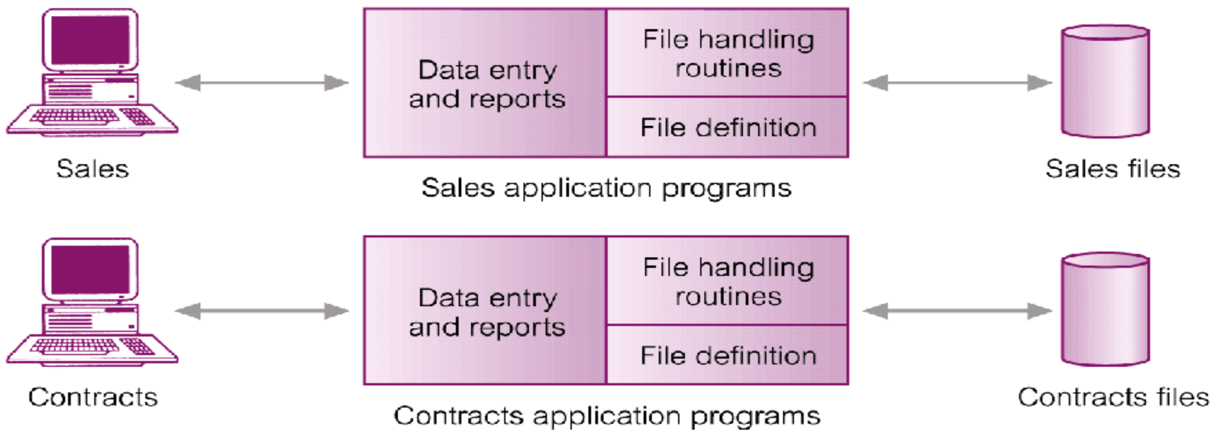
- Introduction
  - File based approach
  - Database approach
  - Nuts and Bolts
- Database systems concepts
  - Data model
  - Three schema architecture – Data independence
  - Database schema – state – instance
  - DBMS languages
  - Classification of DBMS
  - Database users

Introduction to File Based Approach

Data is stored in one or more separate computer files
Data is then processed by computer programs - **applications**



Introduction to File Based Approach



Problems/Limitations
- Data Redundancy
- Data Inconsistency

Sales Files

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)

PrivateOwner (ownerNo, fName, lName, address, telNo)

Client (clientNo, fName, lName, address, telNo, prefType, maxRent)

Contracts Files

Lease (leaseNo, propertyNo, clientNo, rent, paymentMethod, deposit, paid, rentStart, rentFinish, duration)

PropertyForRent (propertyNo, street, city, postcode, rent)

Client (clientNo, fName, lName, address, telNo)

Introduction to File Based Approach

Shared File Approach

- ✓ Data (files) is shared between different applications
- ✓ Data redundancy problem is alleviated
- ✓ Data inconsistency problem across different versions of the same file is solved
- ✓ Other problems:
  - Rigid data structure: If applications have to share files, the file structure that suits one application might not suit another
  - Physical data dependency: If the structure of the data file needs to be changed in some way, this alteration will need to be reflected in all application programs that use that data file
  - No support of concurrency control: While a data file is being processed by one application, the file will not be available for other applications or for ad hoc queries

Introduction to Database Approach

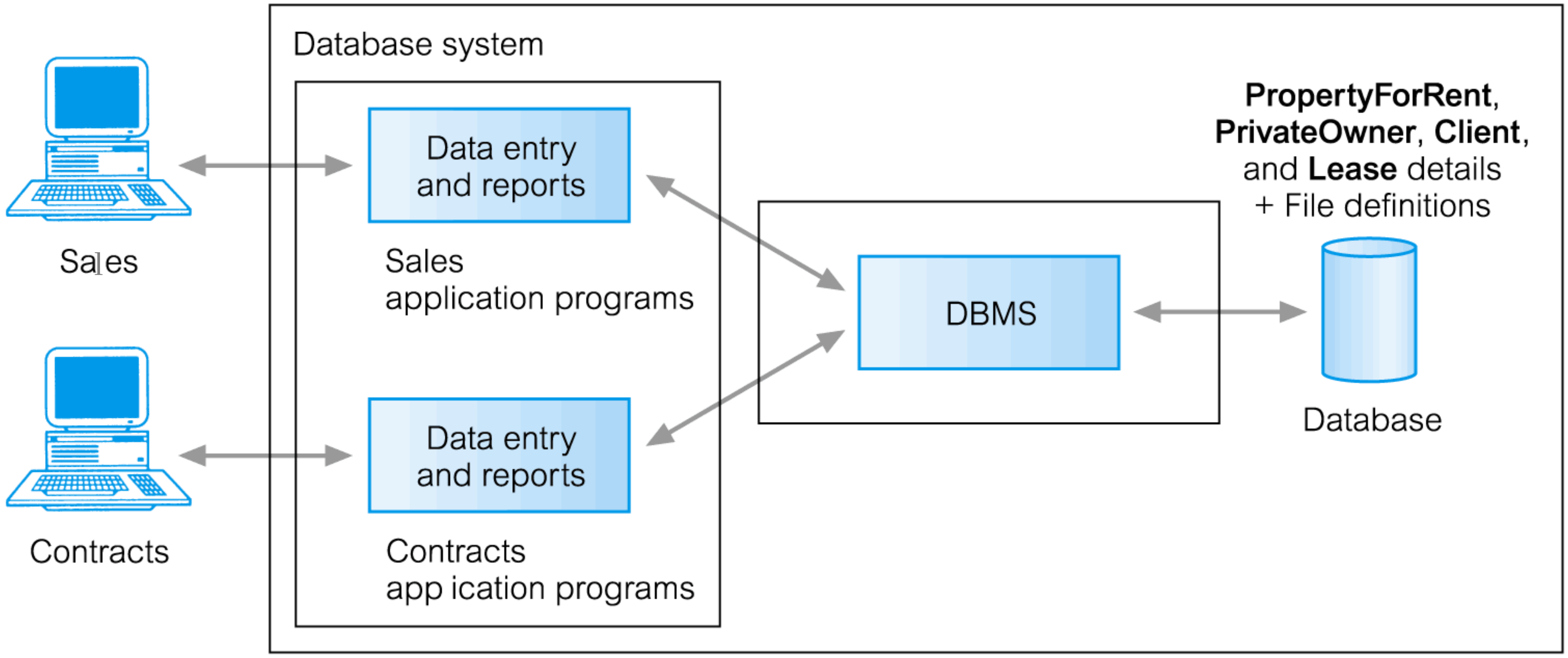
Arose because:

- ✓ Definition of data was embedded in application programs, rather than being stored separately and independently
- ✓ No control over access and manipulation of data beyond that imposed by application programs

Result:

The Database and Database Management System (DBMS).

Introduction to Database Approach


Database system
Data entry and reports
Sales application programs
Data entry and reports
Contracts app ication programs
Sales
Contracts
DBMS
PropertyForRent, PrivateOwner, Client, and Lease details + File definitions
Database

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)
Lease (leaseNo, propertyNo, clientNo, paymentMethod, deposit, paid, rentStart, rentF nish)

Introduction to Database Approach

Database: A collection of related data.

Data: Known facts that can be recorded and have an implicit meaning.

Mini-world: Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.

Database Management System (DBMS): A software package/ system to facilitate the creation and maintenance of a computerized database.

Database System: The DBMS software together with the data itself. Sometimes, the applications are also included.

Introduction to Database Approach

Example of Database (UNIVERSITY)

Mini-world for the example: Part of a UNIVERSITY environment.

Some mini-world *entities*:

STUDENTs , COURSEs, SECTIONs (of COURSEs) (academic)DEPARTMENTs, INSTRUCTORs

Some mini-world *relationships*:

- SECTIONs *are of* specific COURSEs
- STUDENTs *take* SECTIONs
- COURSEs *have* prerequisite COURSEs
- INSTRUCTORs *teach* SECTIONs
- COURSEs *are offered by* DEPARTMENTs
- STUDENTs *major in* DEPARTMENTs

Note: The above could be expressed in the E-R data model.

Characteristics of Database Approach

- ✓ Self-describing nature of a database system:
- ✓ Insulation between programs and data: (**program-data independence**)
  - ✓ Allows changing data storage structures and operations without having to change the DBMS access programs.

- ✓ Data Abstraction: A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.
- ✓ Support of multiple views of the data
- ✓ Sharing of data and multiuser transaction processing

Introduction to Data Models

Data Model: A set of concepts to describe the *structure* of a database, and certain *constraints* that the database should obey.

Data Model Operations: Operations for specifying database retrievals and updates by referring to the concepts of the data model.

Operations on the data model may include

- *Basic operations* and
- *User-defined operations*.

Categories of Data Models

Conceptual (**high-level**, **semantic**) data models: Provide concepts that are close to the way many users *perceive* data.

(Also called **entity-based** or **object-based** data models.)

Physical (**low-level**, **internal**) data models: Provide concepts that describe details of how data is stored in the computer.

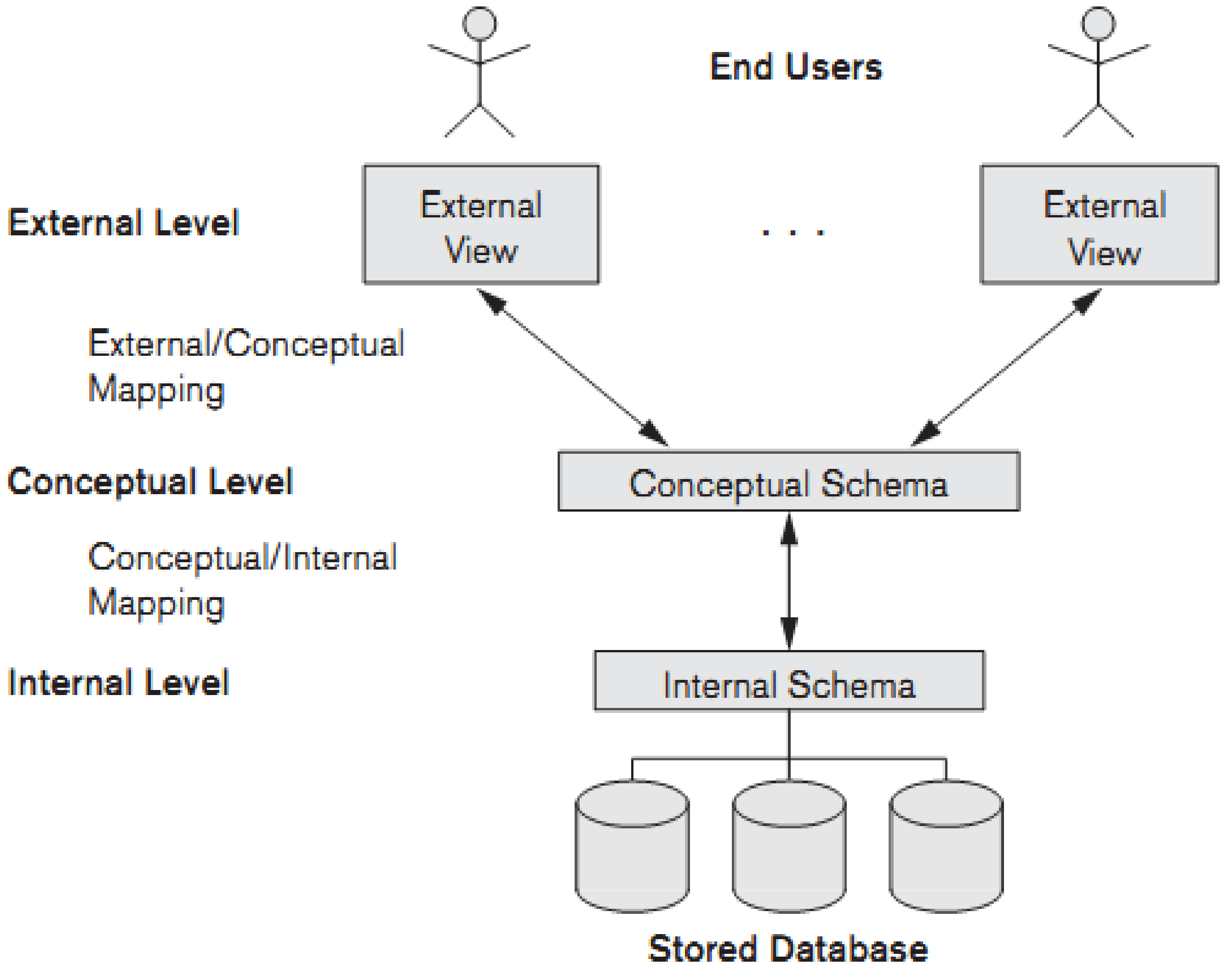
Implementation (**representational**) data models: Provide concepts that fall between the above two, balancing user views with some computer storage details.

Three Scheme Architectures

- Proposed to support DBMS characteristics of:
  - **Program-data independence**.
  - Support of **multiple views** of the data.

- **Objectives of Three-Schema Architecture**
  - All users should be able to access same data
  - A user's view is immune to changes made in other views
  - Users should not need to know physical database storage details
  - DBA should be able to change database storage structures without affecting the users' views
  - Internal structure of database should be unaffected by changes to physical aspects of storage
  - DBA should be able to change conceptual structure of database without affecting all users

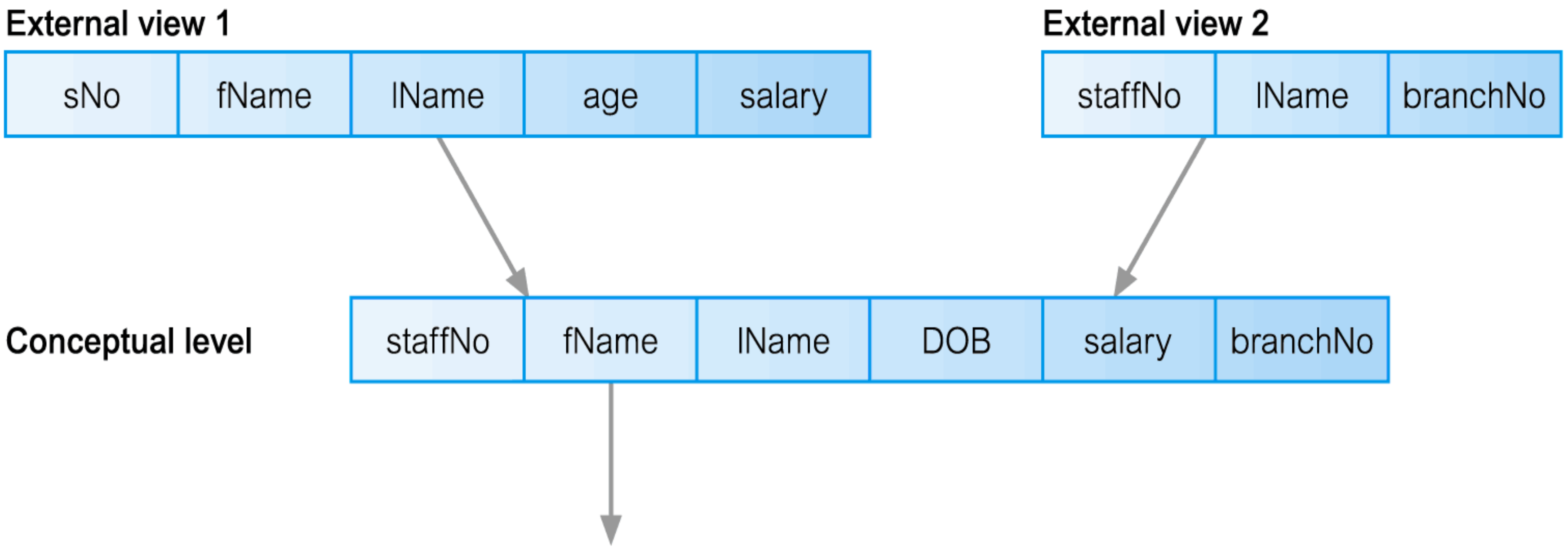
Three Scheme Architectures



Three Scheme Architectures

- Defines DBMS schemas at *three levels*:
  - **Internal schema :** describes physical storage structures and access paths. Typically uses a *physical* data model.
  - **Conceptual schema** : describes the structure and constraints for the *whole* database for a community of users. Uses a *conceptual* or an *implementation* data model.
  - **External schemas** : describes the various user views. Usually uses the same data model as the conceptual level.

Three Scheme Architectures



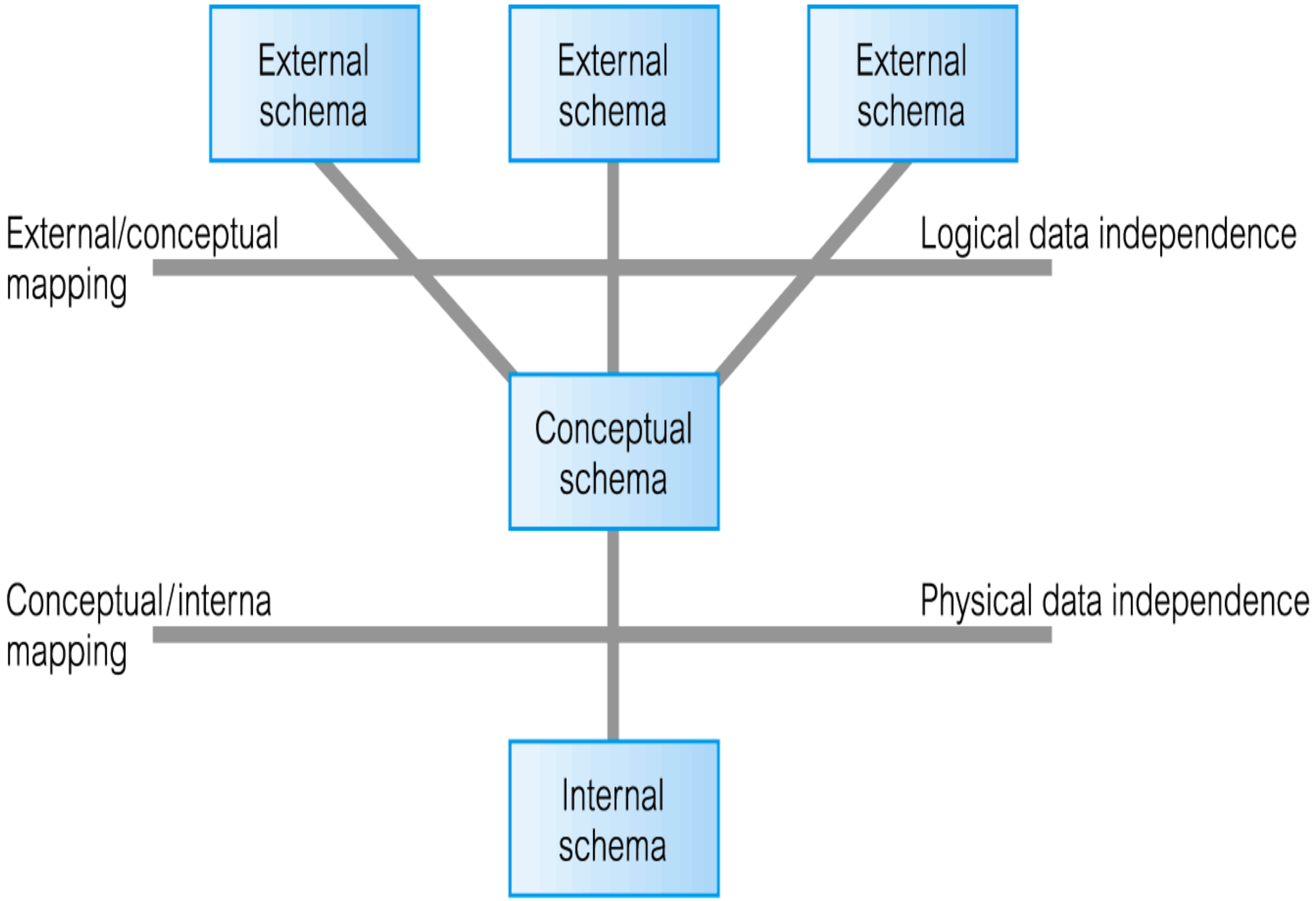
Internal level

```
struct STAFF {
      int staffNo;
      int branchNo;
      char fName [15];
      char lName [15];
      struct date dateOfBirth;
      float salary;
      struct STAFF *next;              /* pointer to next Staff record */
};
index staffNo; index branchNo;         /* define indexes for staff */
```

Data Independence

- Data Independence is the capacity to change the schema at one level of a database system without having to change the schema at the next higher level
  - ✓ **Logical Data Independence**: Change *conceptual schema* without having to change *external schemas* and their application programs.
  - ✓ **Physical Data Independence**: Change *internal schema* without having to change *conceptual schema*.

Data Independence


External schema
External schema
External schema
External/conceptual mapping
Logical data independence
Conceptual schema
Conceptual/interna mapping
Physical data independence
Internal schema

Historical Development and Database Technology

- **Early Database Applications:** Hierarchical and Network Models (in mid 1960's).
- **Relational Model based Systems:** Researched and experimented with in IBM and the universities (in 1970).
- **Object-oriented applications:** OODBMSs (in late 1980's and early 1990's)
- **Data on the Web and E-commerce Applications:** using new standards like XML (eXtended Markup Language).

Schema Verses Instance

- **Database Schema**: The *description* of a database.
- **Schema Diagram**: A diagrammatic display of (some aspects of) a database schema.
- **Schema Construct**: A component of the schema or an object within the schema, e.g., STUDENT, COURSE.
- **Database Instance**: The actual data stored in a database at a *particular moment in time*. Also called **database state** (or **occurrence**).

Database Schema and Database State

- **Database State:** Refers to the content of a database at a moment in time.
- **Initial Database State:** Refers to the database when it is loaded
- **Valid State:** A state that satisfies the structure and constraints of the database.
- **Distinction**
  - ✓ The **database schema** changes *very infrequently*.
  - ✓ The **database state** changes *every time the database is updated.*
  - ✓ **Schema** is also called **intension**, whereas **state** is called **extension**.

DBMS Languages

- **Data Definition Language (DDL)** allows the DBA or user to describe and name entities, attributes, and relationships required for the application plus any associated integrity and security constraints
- **Data Manipulation Language (DML)** provides basic data manipulation operations on data held in the database
- **Data Control Language (DCL)** defines activities that are not in the categories of those for the DDL and DML, such as granting privileges to users, and defining when proposed changes to a databases should be irrevocably made.
- **Low Level or Procedural DML:** allow user to tell system exactly how to manipulate data (e.g., Network and hierarchical DMLs)
- **High Level** or **Non-procedural DML**(declarative language): allow user to state what data is needed rather than how it is to be retrieved (e.g., SQL, QBE)

DBMS Classification

- **Based on the data model used**:
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational.
- **Other classifications:**
  - Single-user (typically used with micro- computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)

History of Data Models

- Relational Model: proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82. Now in several commercial products (DB2, ORACLE, SQL Server, SYBASE, INFORMIX).

- Network Model: the first one to be implemented by Honeywell in 1964-65 (IDS System). Adopted heavily due to the support by CODASYL (CODASYL - DBTG report of 1971). Later implemented in a large variety of systems - IDMS (Cullinet - now CA), DMS 1100 (Unisys), IMAGE (H.P.), VAX -DBMS (Digital Equipment Corp.).

- Hierarchical Data Model: implemented in a joint effort by IBM and North American Rockwell around 1965. Resulted in the IMS family of systems. The most popular model. Other system based on this model: System 2k (SAS inc.)

History of Data Models

- Object-oriented Data Model(s): several models have been proposed for implementing in a database system. One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE). Additionally, systems like $O_{2,}$ ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).

- Object-Relational Models: Most Recent Trend. Started with Informix Universal Server. Exemplified in the latest versions of Oracle-10i, DB2, and SQL Server etc. systems.

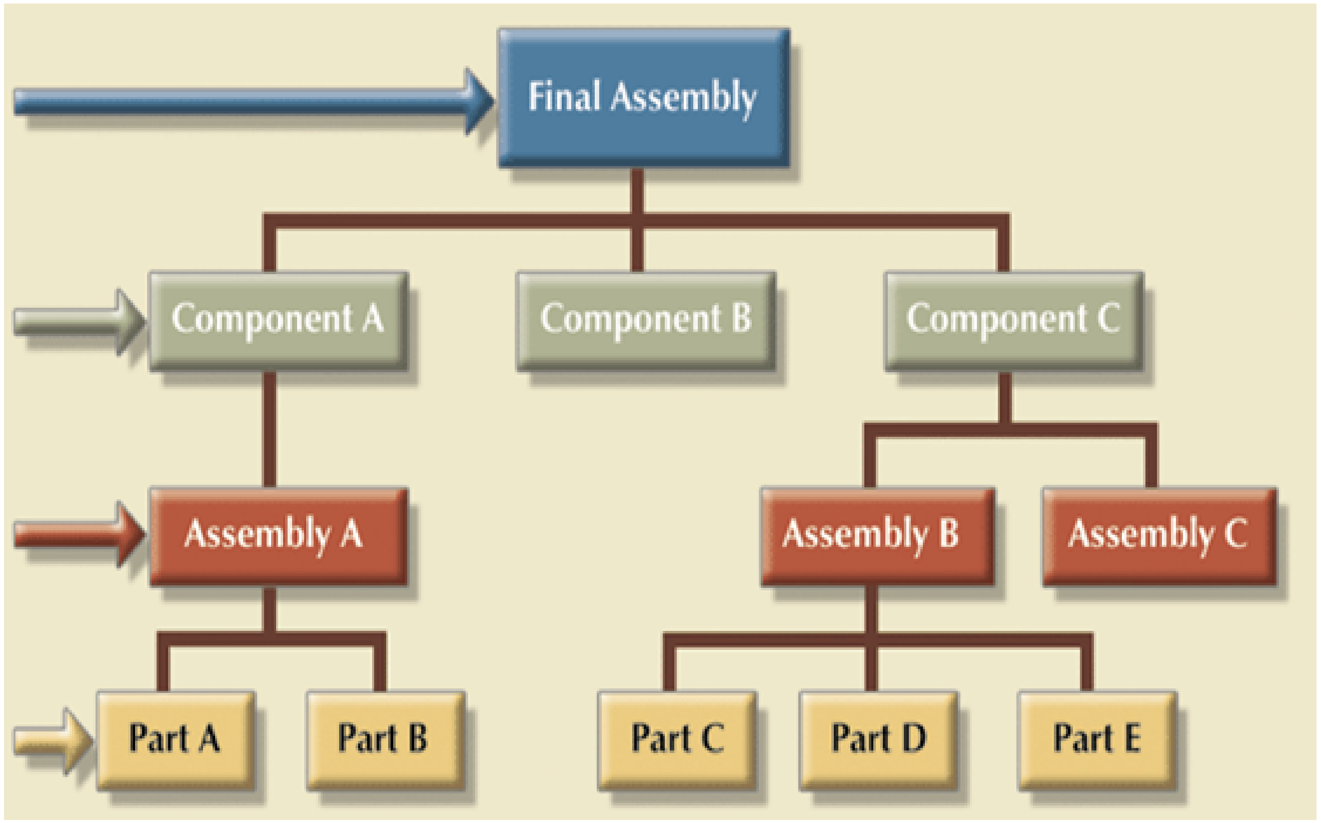
Hierarchical Model

Root Segment

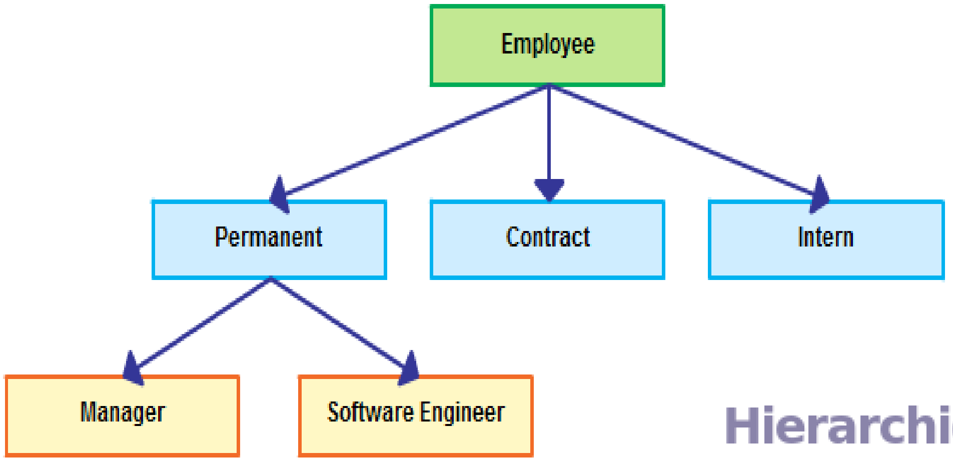
Level-1 Segment
(Root Children)

Level-2 Segment
(Level-1 Children)

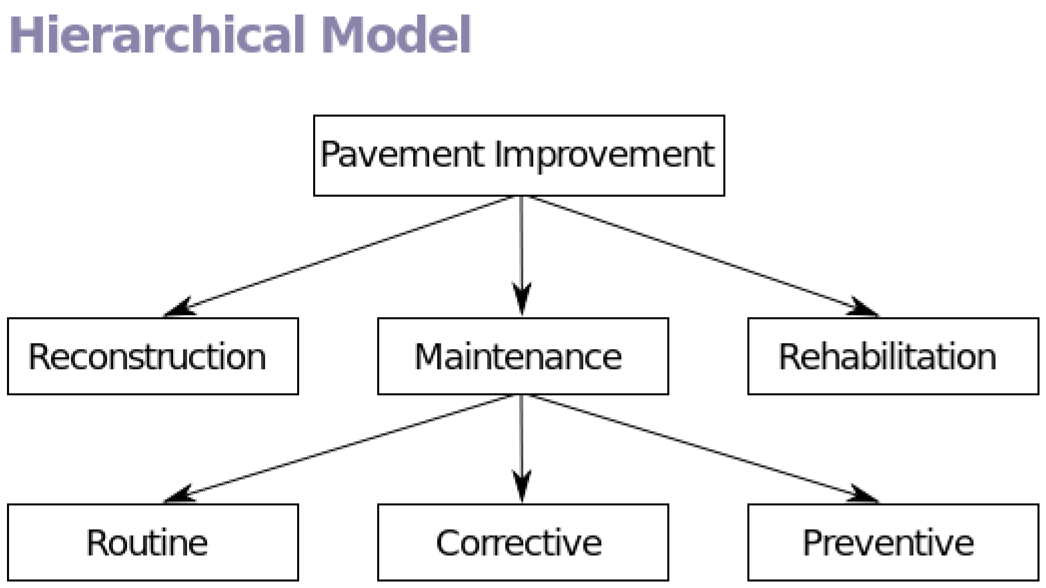
Level-3 Segment
(Level-2 Children)



Hierarchical Model



Example-1



Hierarchical Model

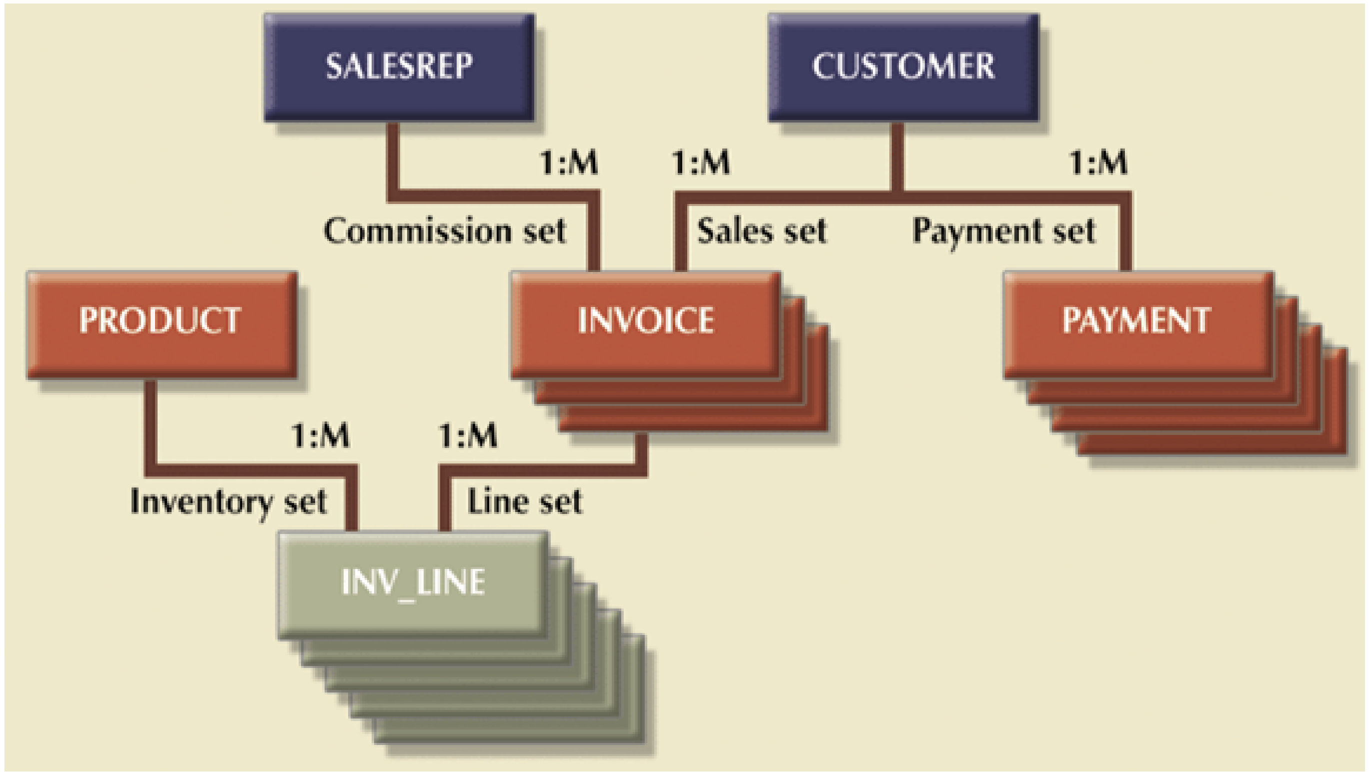
ADVANTAGES:

- Hierarchical Model is simple to construct and operate
- Corresponds to a number of natural hierarchically organized domains - e.g., assemblies in manufacturing, personnel organization in companies
- Language is simple; uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT etc.

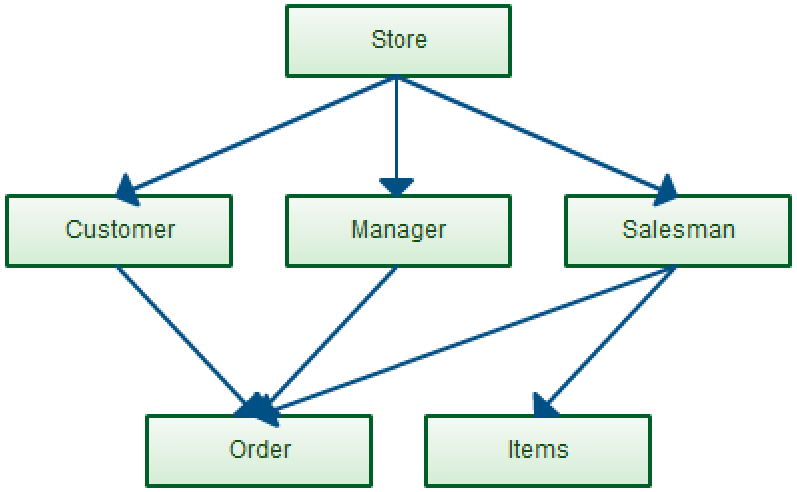
DISADVANTAGES:

- Navigational and procedural nature of processing
- Database is visualized as a linear arrangement of records
- Little scope for "query optimization"

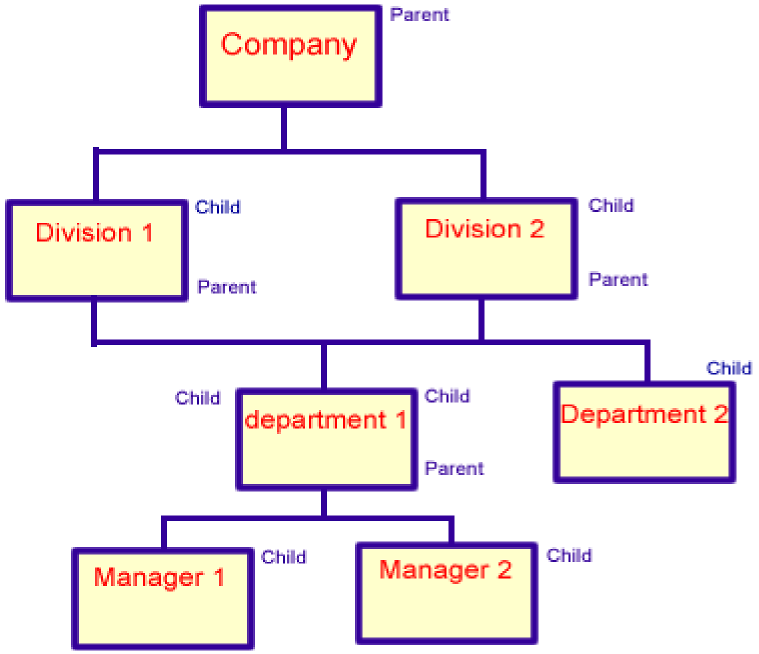
Network Model



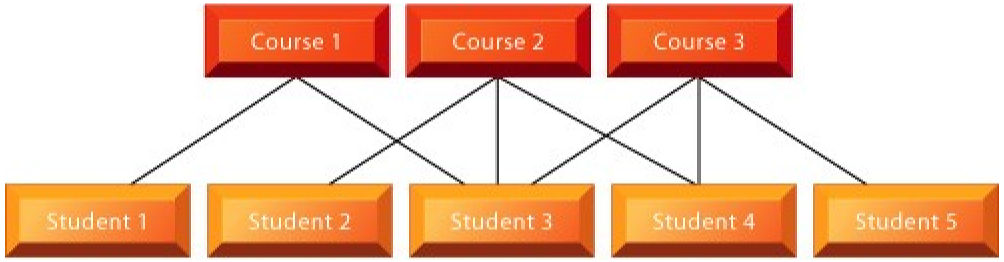
Network Model



Example-1



Example-2



Example-3

Network Model

ADVANTAGES:

- It is able to model complex relationships and represents semantics of add/delete on the relationships.
- Can handle most situations for modeling using record types and relationship types.
- Language is navigational; uses constructs like FIND, FIND member, FIND owner, FIND NEXT within set, GET etc. Programmers can do optimal navigation through the database.

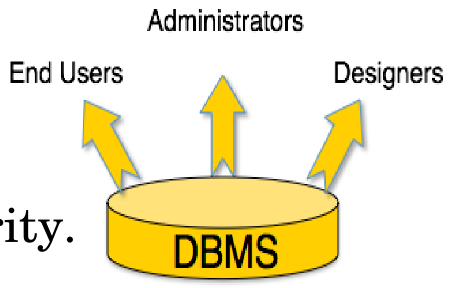
DISADVANTAGES:

- Navigational and procedural nature of processing
- Database contains a complex array of pointers that thread through a set of records.
- Little scope for automated "query optimization”

DBMS Users (Actors on Scene)

Administrators – maintains the DBMS and are responsible for

- Administrating the database.
- Its usage and by whom it should be used.
- Create access profiles for users and
- Apply limitations to maintain isolation and force security.
- System license, required tools, and
- Other software and hardware related maintenance.



Designers – Group of people who actually work on the designing part of the database. They look on what data should be kept and in what format.
They identify and design the whole set of entities, relations, constraints, and views

End Users – End users are those who actually uses the benefits of DBMS. They can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts

DBMS Interfaces

- Stand-alone query language interfaces.
- Programmer interfaces for embedding DML in programming languages:
  - ✓ Pre-compiler Approach
  - ✓ Procedure (Subroutine) Call Approach
- User-friendly interfaces:
  - ✓ Menu-based, popular for browsing on the web
  - ✓ Forms-based, designed for users
  - ✓ Graphics-based (Point and Click, Drag and Drop etc.)
  - ✓ Natural language: requests in written English
  - ✓ Combinations of the above
- Interfaces for the DBA:
  - ✓ Creating accounts, granting authorizations
  - ✓ Setting system parameters
  - ✓ Changing schemas or access path

Database System Utilities/Tools

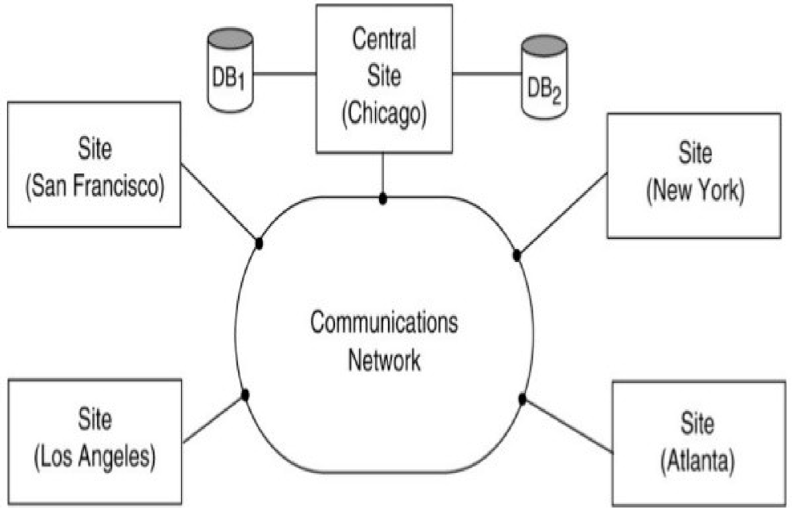
- To perform certain functions such as:
  - ✓ *Loading* data stored in files into a database. Includes data conversion tools.
  - ✓ *Backing up* the database periodically on tape.
  - ✓ *Reorganizing* database file structures.
  - ✓ *Report generation* utilities.
  - ✓ *Performance monitoring* utilities.
  - ✓ Other functions, such as *sorting*, *user monitoring*, *data compression*, etc.

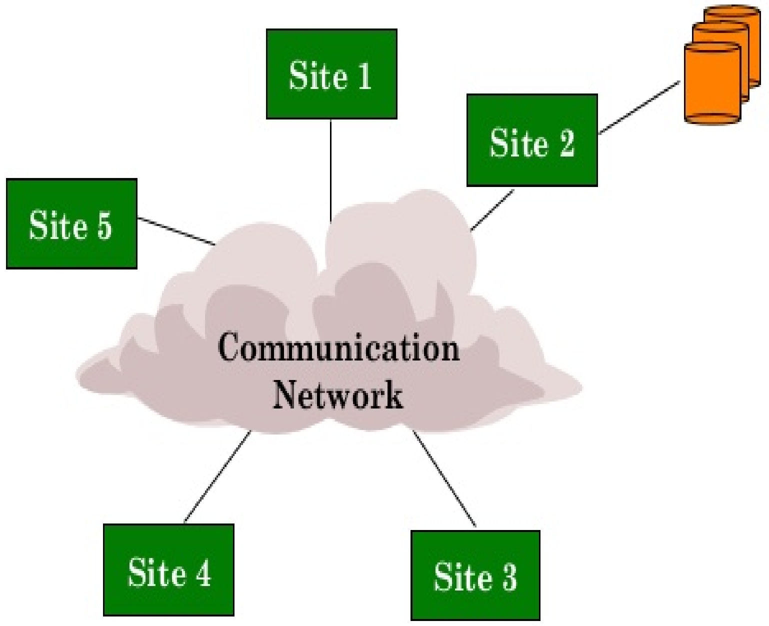
Database System Utilities/Tools

- **Data dictionary / repository**:
    - Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.
    - *Active* data dictionary is accessed by DBMS software and users/DBA.
    - *Passive* data dictionary is accessed by users/DBA only.

- **Application Development Environments and CASE (computer-aided software engineering) tools:**
    - Examples – Power builder (Sybase), Builder (Borland)

Centralized Database





Centralized Database

- A **centralised database** (**CDB**) is located, stored, and maintained in a single location (**Desktop/server CPU or mainframe computer**).
- It would be used by an organisation (e.g. a business company) or an institution (e.g. a university.)
- Users access a centralised database through a computer network which is able to give them access to the central CPU, which in turn maintains to the database itself.

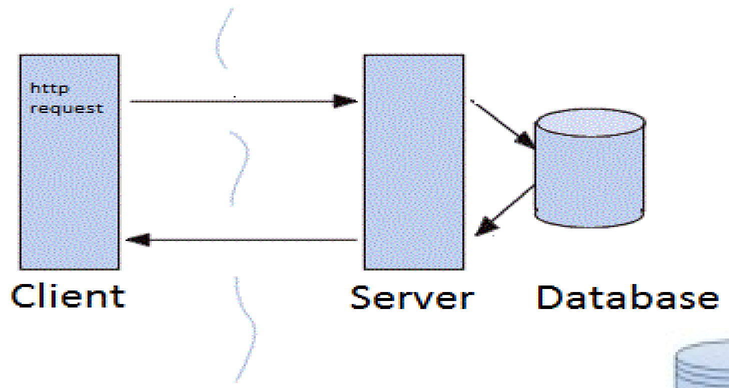
Advantages of CDB

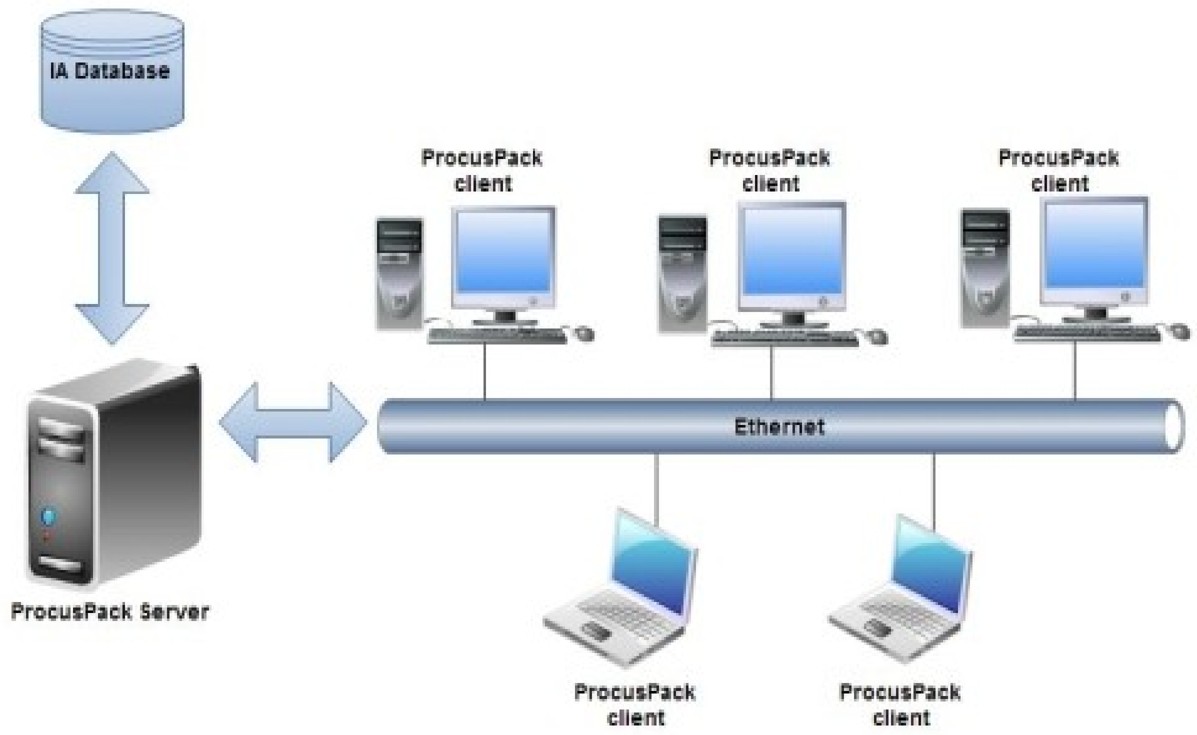
- Data integrity is maximised and data redundancy is minimised.
- Bigger data security
- Better data preservation than other types of databases due to often-included fault-tolerant setup.
- Easier for using by the end-user due to the simplicity of having a single database design.
- Generally easier data portability and database administration.
- More cost effective in terms of labour, power supply.
- Maintenance costs are all minimised.
- Data kept in the same location is easier to be changed, re-organised, mirrored, or analysed.
- All the information can be accessed at the same time from the same location.[9]
- Updates to any given set of data are immediately received by every end-user.

Disadvantages of CDB

- Centralised databases are highly dependent on network connectivity.
- Bottlenecks can occur as a result of high traffic.
- Limited access by more than one person to the same set of data as there is only one copy of it and it is maintained in a single location. This can lead to major decreases in the general efficiency of the system.
- If there is no fault-tolerant setup and hardware failure occurs, all the data within the database will be lost.
- Since there minimal to none data redundancy, if a set of data is unexpectedly lost it is very hard to retrieve it back, and in most cases it would have to be done manually.

Client/Server Data Model





Client/Server Data Model

- The **client–server model** is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.
- Clients and Servers communicate over a computer network on separate hardware, but both client and server may reside in the same system.
- A server host runs one or more server programs which share their resources with clients.
- A client does not share any of its resources, but requests a server's content or service function.
- Clients therefore initiate communication sessions with servers which await incoming requests.
- Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

Merits and DeMerits of Client/Server Model

Merits

- Centralization – Access, Resources, and Data Security are controlled through server
- Scalability – Any element can be upgraded when needed
- Flexibiltiy – New Technology can be easily integrated into the system
- Interoperabilty – All components work together.

DeMerits

- Dependability – When Servers goes down, operations will cease
- Lack of Mature Tools - To administrate
- Lack of Scalability – Network OS are not vary scalable.
- Higher than anticipated Cost.
- Network Congestion.