AIIQ 3211 DATABASE CONSTRUCTION AND MANAGEMENT

Mr. Jackson Alunga

jalungar@gmail.com

Database

- A database is a collection of related fields grouped into records.
- Fields are the smallest unit of data and are useless by themselves. All fields related to a particular subject form records.
- Records are used to represent all the information pertaining to one person, place, topic, or thing.
- The conglomeration of records forms the database.

Database

- A collection of related data with reduced/controlled redundancy
- Represents some part of the real world (miniworld).
  - Changes in miniworld reflected in database
- Designed, built and populated for a specific purpose – for an intended group of users and some application in which the users are interested

Data:

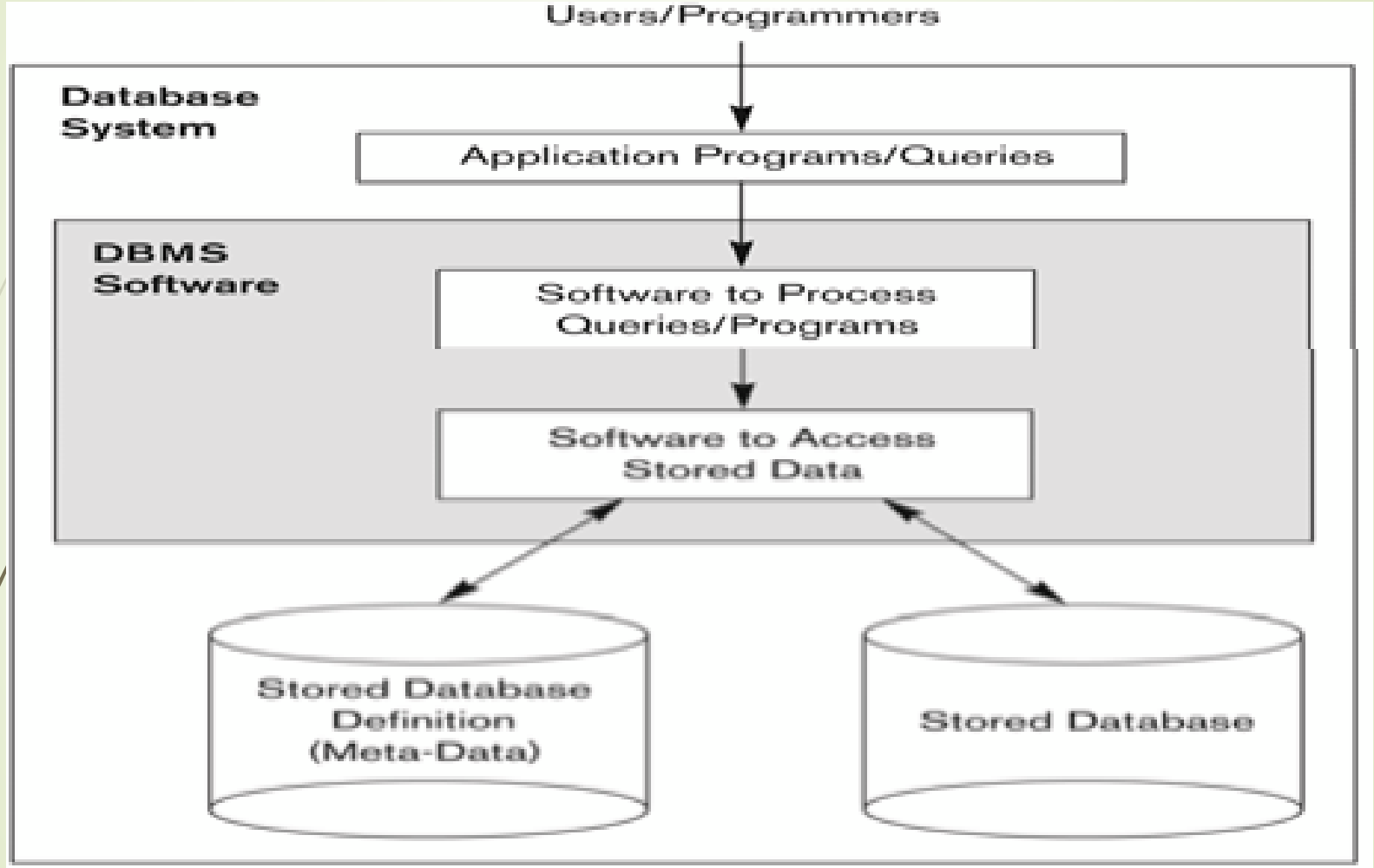
- Known facts that can be recorded and have an implicit meaning
  - e.g. names, addresses, telephone etc. about clients

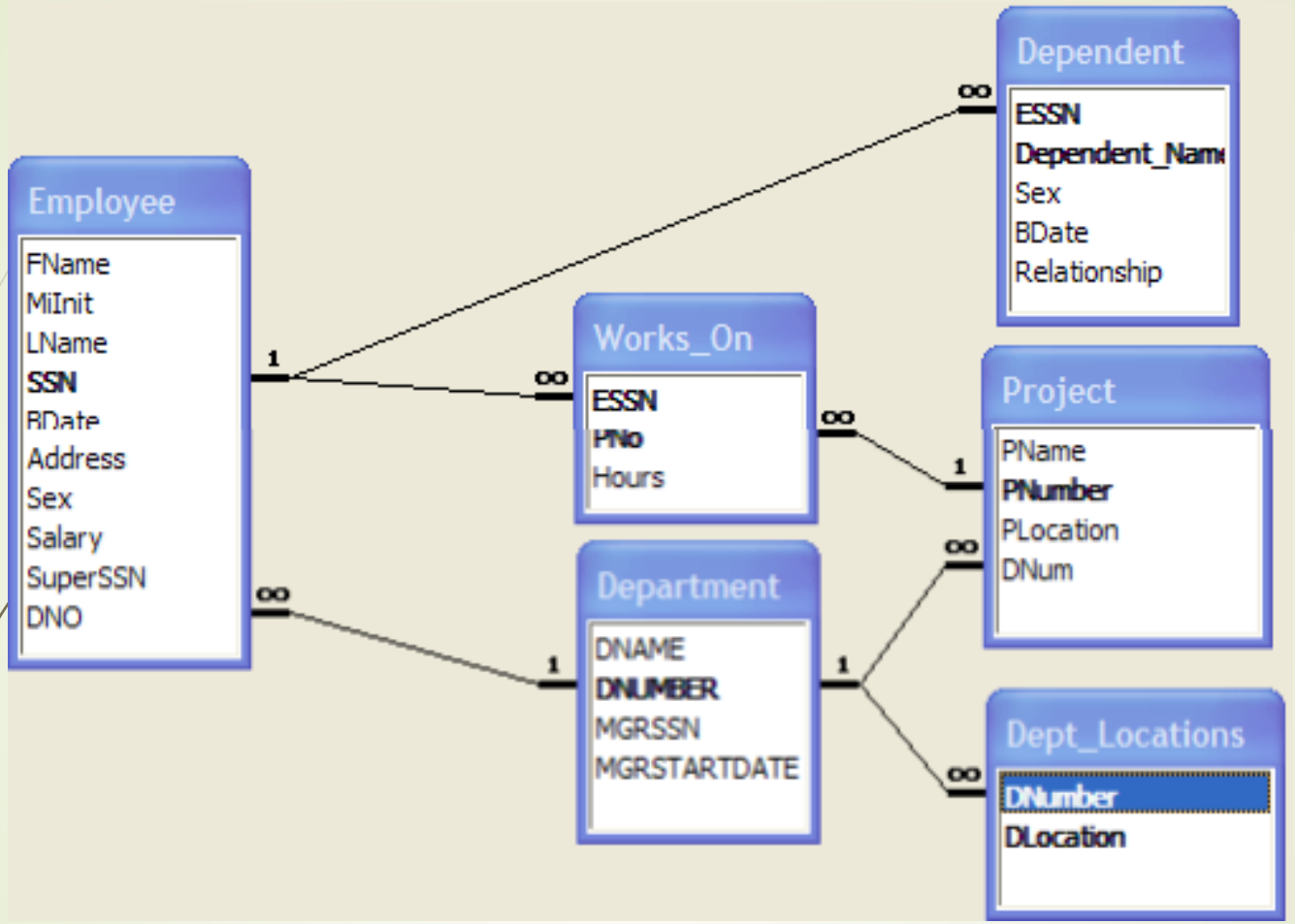
Database Management System (DBMS):

- A software package/ system to facilitate the creation and maintenance of a computerized database.
  - E.g. MSSQL, MySql, Access, Oracle

Database System:

- The DBMS software together with the data itself. (Sometimes the definition includes applications that manipulate the database)


Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database


Dependent
ESSN
Dependent_Nam
Sex
BDate
Relationship
Employee
FName
MiInit
LName
SSN
BDate
Address
Sex
Salary
SuperSSN
DNO
Works_On
ESSN
PNo
Hours
Project
PName
PNumber
PLocation
DNum
Department
DNAME
DNUMBER
MGRSSN
MGRSTARTDATE
Dept_Locations
DNumber
DLocation
1
∞

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Examples of database applications

- Banking systems – manage deposits, withdrawals, loan issuance and repayment etc.
- Hotel/Airline Reservation systems
- A computerized library catalog
- Point of Sale System
- Purchases from the supermarket
- Purchases using your credit card
- Booking a holiday at the travel agents
- Using the local library
- Taking out insurance
- Renting a video
- Using the Internet
- Studying at university

Recent Advances In Database Applications

- Multimedia Databases:
  - Store pictures, video clips, sound messages etc.
- Geographic Information Systems (GIS):
  - Store and analyze maps, weather data, satellite images
- Data Warehouses:
  - Used to extract and analyze info from very large DBs for decision making
- Real-time and Active Databases:
  - Such as are used in controlling industrial and manufacturing processes
  - Designed to handle workload that is constantly changing e.g stock exchanges
- WWW databases
  - People constantly upload files

File-Based Systems

- Collection of application programs that perform services for the end users (e.g. reports).
- Each program defines and manages its own data.
  - Each user defines and implements the files needed for a specific application as part of programming the application.

File-Based Systems

- For example, one user, the grade reporting office, may keep a file on students and their grades.
  - Programs to print a student's transcript and to enter new grades into the file are implemented.
- A second user, the accounting office, may keep track of students' fees and their payments.
  - Although both users are interested in data about students, each user maintains separate files—and programs to manipulate these files—because each requires some data not available from the other user's files.
- This redundancy in defining and storing data results in wasted storage space and in redundant efforts to maintain common data up-to-date.

FILE BASED PROCESSING

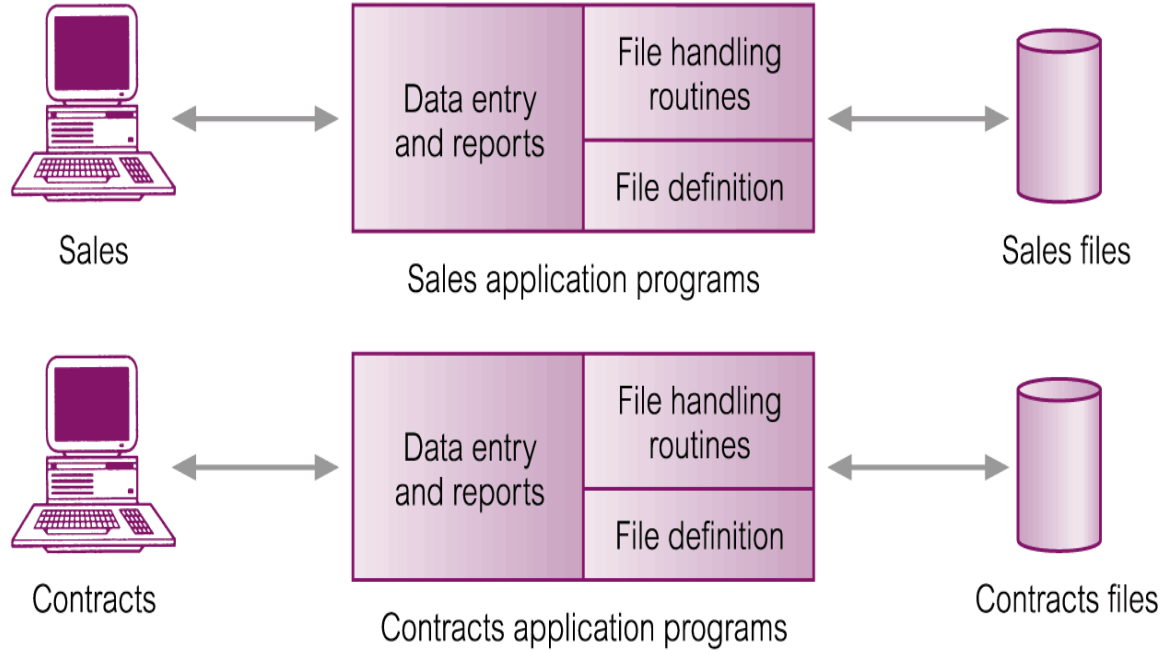


Figure 1.5
File-based processing.

Limitations of File-Based Approach

- Separation and isolation of data
  - Each program maintains its own set of data.
  - Users of one program may be unaware of potentially useful data held by other programs.
- Duplication of data
  - Same data is held by different programs.
  - Wasted space and potentially different values and/or different formats for the same item

Limitations of File-Based Approach

- Data dependence
  - File structure is defined in the program code.
- Incompatible file formats
  - Programs are written in different languages, and so cannot easily access each other's files.

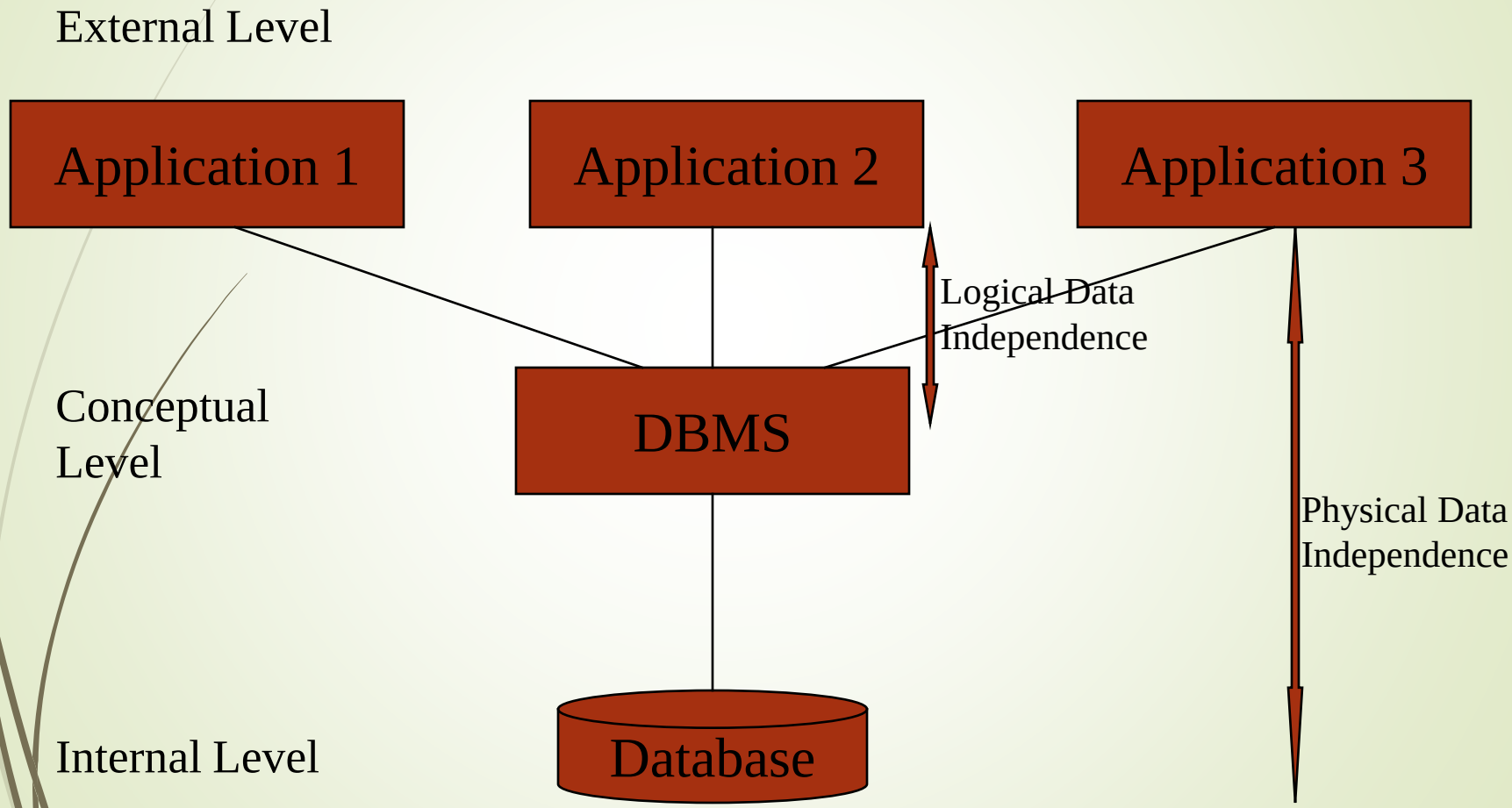
Database Approach

- 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).

Database Approach Languages

- Data definition language (DDL).
  - Permits specification of data types, structures and any data constraints.
  - All specifications are stored in the database.
- Data manipulation language (DML).
  - General enquiry facility (query language) of the data.
  - Insert, Update, Delete and retrieve data from the database

Database Architecture



External level

- Describes the part of the database that a particular user group is interested in
  - hides the rest of the database from that user group.
- The end-user's view of the data environment (the typical employee in sales, finance, marketing etc.)
- Each business unit uses a data subset of the overall data
  - end users view their data subset as separate/external to other units
- The perspective (external schema) may share certain data elements with other business units' perspectives

Conceptual level

- Describes the structure of the whole database for a community of users.
  - Represents a global view of the entire database.
  - A representation of data as viewed by the entire organization i.e. an integration of all external views to form a conceptual schema
- Hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
- Intended to be software and hardware independent.
  - Doesn’t depend on any particular DBMS
  - Doesn’t depend on the hardware to be used

Internal/Physical level

- Describes the physical storage structure of the database.
  - describes the complete details of data storage and access paths for the database i.e. how data is to be saved on storage media: tapes, disks etc
- Requires definition of physical storage devices and required access methods i.e. it is both h/w and s/w dependent
- Represents the database as ‘seen’ by the DBMS
  - places the data in such a format that it is only readable by the DBMS
  - A result of the designer matching the conceptual model’s characteristics and constraints to those supported by the selected implementation DBMS

Logical data independence

- The capacity to change the conceptual schema without having to change external schemas or application programs.
  - expand the database by adding a record type or data item
  - reduce the database by removing a record type or data item
  - The external schemas that refer only to the remaining data should not be affected.

Logical data independence

- Only the view definition and the mappings need be changed in a DBMS that supports logical data independence.
  - Application programs that reference the external schema constructs must work as before, after the conceptual schema undergoes a logical reorganization.
  - Changes to constraints can be applied also to the conceptual schema without affecting the external schemas or application programs.

Physical data independence

- The capacity to change the internal schema without having to change the conceptual (or external) schemas.
- Reorganizing physical files -for example, by creating additional access structures
  - to improve the performance of retrieval or update.
  - If the same data as before remains in the database, we should not have to change the conceptual schema. For example, providing an access path to improve retrieval of SECTION records

Changes May Include

- Using new storage devices.
- Using different data structures.
- Switching from one access method to another.
- Using different file organizations or storage structures.
- Modifying indexes.

A change to the internal schema, such as using different file organization or storage structures, storage devices, or indexing strategy, should be possible without having to change the conceptual or external schemas.

- The objective of the three-level architecture is to separate each user's view of the database from the way the database is physically represented.
- There are several reasons why this separation is desirable:
  - Each user should be able to access the same data, but have a different customized view of the data.
  - Each user should be able to change the way he or she views the data, and this change should not affect other users.
  - Users should not have to deal directly with physical database storage details, such as indexing
  - A user's interaction with the database should be independent of storage considerations.

- The Database Administrator (DBA) should be able to change the database storage structures without affecting the users' views.
- The internal structure of the database should be unaffected by changes to the physical aspects of storage, such as the changeover to a new storage device.
- The DBA should be able to change the conceptual structure of the database without affecting all users.

Characteristics of database Approach

- insulation of programs and data (program-data and program-operation independence);
- support of multiple user views
- Self-Describing in Nature; use of a catalog to store the database description
- Sharing of Data and Multiuser Transaction Processing
- Data Abstraction

Characteristics of database Approach

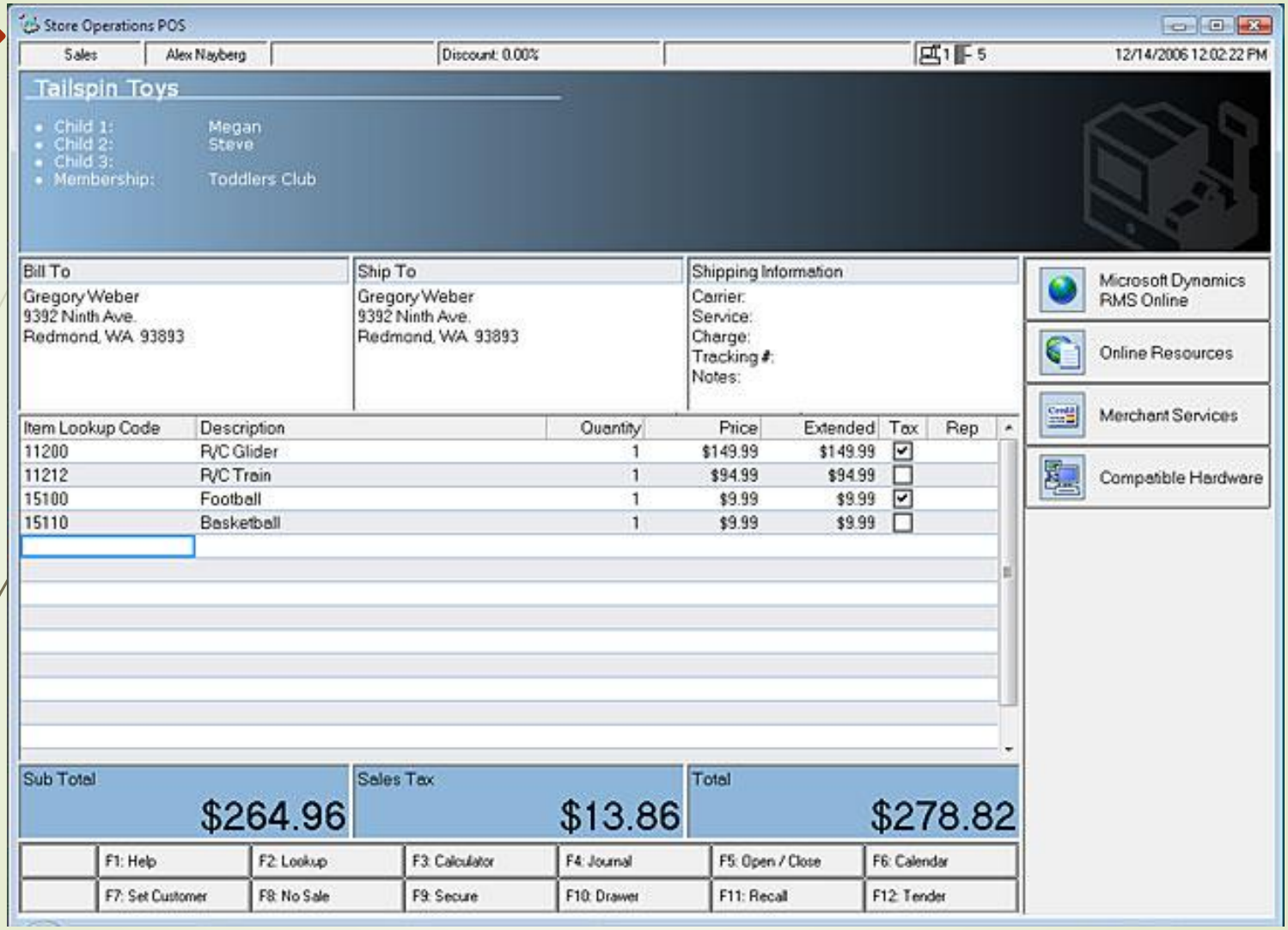
- Self-Describing in Nature
  - database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints.
  - This definition is stored in the system catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data.
  - The information stored in the catalog is called *meta-data,* and it describes the structure of the primary database
    - The catalog is used by all users who need information about the database structure.
  - The DBMS software must work equally well with any number of database applications—for example, a university database, a banking database, or a company database—as long as the database definition is stored in the catalog.

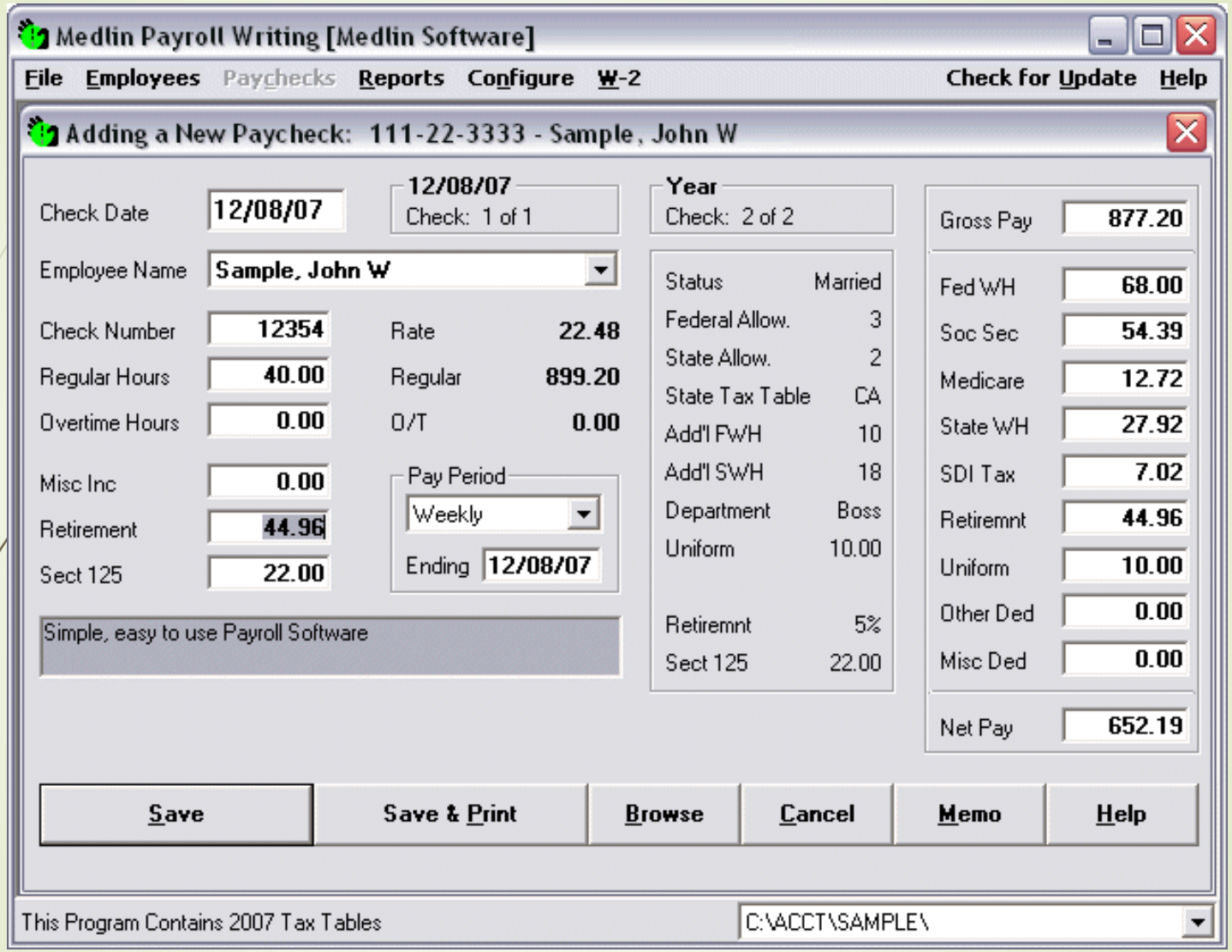
Insulation between Programs and Data

- In traditional file processing, the structure of data files is embedded in the access programs, so any changes to the structure of a file may require changing all programs that access this file.
- By contrast, DBMS supports Program-data independence.
  - access programs do not require such changes in most cases.
  - The structure of data files is stored in the DBMS catalog separately from the access programs.
  - For example, a file access program may be written in such a way that it can access only STUDENT records
  - If we want to add another piece of data to each STUDENT record, say the Birthdate, such a program will no longer work and must be changed.
  - By contrast, in a DBMS environment, we just need to change the description of STUDENT records in the catalog to reflect the inclusion of the new data item Birthdate; no programs are changed.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

Support of multiple views of the data:

- A database has many users,
  - each requires a different perspective or view of the database.
- Each user may see a different view of the database, which describes only the data of interest to that user.
- A view may be a subset of the database


Store Operations POS
Sales
Alex Nayberg
Discount: 0.00%
1
5
12/14/2006 12:02:22 PM
Tailspin Toys
Child 1: Megan
Child 2: Steve
Child 3:
Membership: Toddlers Club
Bill To
Gregory Weber
9392 Ninth Ave.
Redmond, WA 93893
Ship To
Gregory Weber
9392 Ninth Ave.
Redmond, WA 93893
Shipping Information
Carrier:
Service:
Charge:
Tracking #:
Notes:
Microsoft Dynamics RMS Online
Online Resources
Merchant Services
Compatible Hardware
Item Lookup Code | Description | Quantity | Price | Extended | Tax | Rep
11200 | R/C Glider | 1 | $149.99 | $149.99 | ☑
11212 | R/C Train | 1 | $94.99 | $94.99 | ☐
15100 | Football | 1 | $9.99 | $9.99 | ☑
15110 | Basketball | 1 | $9.99 | $9.99 | ☐
Sub Total
$264.96
Sales Tax
$13.86
Total
$278.82
F1: Help
F2: Lookup
F3: Calculator
F4: Journal
F5: Open / Close
F6: Calendar
F7: Set Customer
F8: No Sale
F9: Secure
F10: Drawer
F11: Recall
F12: Tender


Medlin Payroll Writing [Medlin Software]
File Employees Paychecks Reports Configure W-2 Check for Update Help
Adding a New Paycheck: 111-22-3333 - Sample, John W
Check Date 12/08/07
12/08/07
Check: 1 of 1
Year
Check: 2 of 2
Employee Name Sample, John W
Check Number 12354
Rate 22.48
Regular Hours 40.00
Regular 899.20
Overtime Hours 0.00
O/T 0.00
Misc Inc 0.00
Retirement 44.96
Sect 125 22.00
Pay Period
Weekly
Ending 12/08/07
Simple, easy to use Payroll Software
Status Married
Federal Allow. 3
State Allow. 2
State Tax Table CA
Add'l FWH 10
Add'l SWH 18
Department Boss
Uniform 10.00
Retiremnt 5%
Sect 125 22.00
Gross Pay 877.20
Fed WH 68.00
Soc Sec 54.39
Medicare 12.72
State WH 27.92
SDI Tax 7.02
Retiremnt 44.96
Uniform 10.00
Other Ded 0.00
Misc Ded 0.00
Net Pay 652.19
Save
Save & Print
Browse
Cancel
Memo
Help
This Program Contains 2007 Tax Tables
C:\ACCT\SAMPLE\

Sharing of Data and Multiuser Transaction Processing

- A multiuser DBMS, must allow multiple users to access the database at the same time.
  - This is essential if data for multiple applications is to be integrated and maintained in a single database.
  - The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct.
  - For example, when several reservation clerks try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one clerk at a time for assignment to a passenger.
- A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly.

Data Abstraction

- a.k.a the ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee) Architecture
- A data model is used to hide storage details and present the users with a conceptual view of the database.
- Programs refer to the data model constructs rather than data storage details
  - For the system to be usable, it must retrieve data efficiently.
  - The need for efficiency has led designers to use complex data structures to represent data in the database.
  - Since many database-systems users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:

Levels of data abstraction

- Physical Level : The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.
- Logical Level : Middle Level, Database administrators, who must decide what information to keep in the database,
  - what data are stored in the database,
  - what relationships exist among those data.

Levels of Abstraction

- The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. use the logical level of abstraction.
- View Level : The highest level of abstraction
- describes only part of the entire database.
- Users of the database system do not need the entire database information; instead, they need to access only a part of the database.
- The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

Data Abstraction

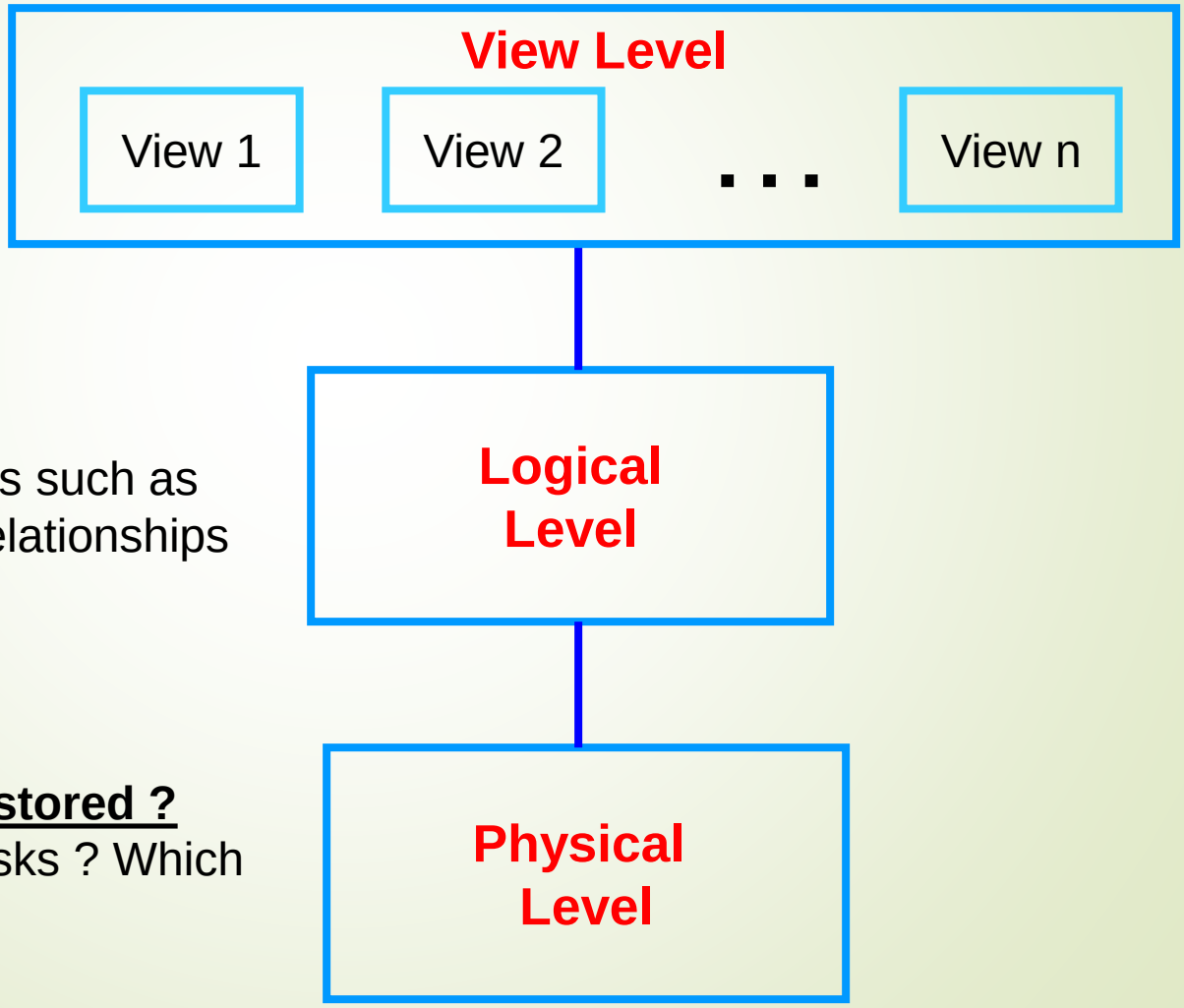
What data users and application programs see ?

What data is stored ?

describe data properties such as data semantics, data relationships

How data is actually stored ?

e.g. are we using disks ? Which file system ?



Advantages of DBMS

- Control of data redundancy
- Data consistency
- More information from the same amount of data
- Sharing of data
- Improved data integrity
- Improved security
- Enforcement of standards
- Economy of scale

1. Controlling Redundancy

- In traditional software development utilizing file processing, every user group maintains its own files for handling its data-processing applications.
  - For example, consider the UNIVERSITY database
  - two groups of users might be the course registration personnel and the accounting office.
  - In the traditional approach, each group independently keeps files on students.
  - The accounting office also keeps data on registration and related billing information, whereas the registration office keeps track of student courses and grades. Much of the data is stored twice: once in the files of each user group.
  - This redundancy in storing the same data multiple times leads to several problems.
    - The need to perform a single logical update multiple times; leads to duplication of effort
    - Storage space is wasted when the same data is stored repeatedly,
    - files that represent the same data may become inconsistent-because an update is applied to some of the files but not to others.
  - In the database approach, the views of different user groups are integrated during database design

2. Restricting Unauthorized Access

- When multiple users share a database, it is likely that some users will not be authorized to access all information in the database.
  - For example, financial data is often considered confidential, and hence only authorized persons are allowed to access such data.
  - In addition, some users may be permitted only to retrieve data, whereas others are allowed both to retrieve and to update.
- Type of access operation update—must also be controlled.
- A DBMS provides a security and authorization subsystem, for the DBA to create accounts and to specify account restrictions.

3. Providing Persistent Storage for Program Objects and Data Structures

- Providing Storage Structures (e.g. indexes) for efficient Query Processing
- Databases can be used to provide persistent storage for program objects and data structures.
  - This is one of the main reasons for the emergence of the object-oriented database systems.
  - Programming languages typically have complex data structures,
  - Objects are persistent, even after the termination of program execution and can later be directly retrieved by another program.
- The persistent storage of program objects and data structures is an important function of database systems.
- Traditional database systems often suffered from the so-called impedance mismatch problem, since the data structures provided by the DBMS were incompatible with the programming language's data structures.
  - Object-oriented database systems typically offer data structure compatibility with one or more object-oriented programming languages.

4. Providing Multiple User Interfaces

- Users vary in levels of technical knowledge
- DBMS should provide a variety of user interfaces.
  - These include query languages for casual users;
  - Programming language interfaces for application programmers;
  - forms and command codes for parametric users;
  - menu-driven interfaces and natural language interfaces for stand-alone users.

Benefits of views

- Views reduce complexity by letting users see the data in the way they want to see it.
- Views provide a level of security. Views can be set up to exclude data that some users should not see.
  - For example, we could create a view that allows a branch manager and the Payroll Department to see all staff data, including salary details, and a second view that other staff would use that excludes salary details.

Benefit of Views

- Views provide a mechanism to customize the appearance of the database.
  - For example, the Contracts Department may wish to call the monthly rent field (rent) by the more obvious name, Monthly Rent.
- A view helps provide the program-data independence.
  - even if fields are added or removed from a file, and these fields are not required by the view, the view is not affected by this change

5. Enforcing Integrity Constraints

- Most database applications have certain integrity constraints that must hold for the data.
- specifying a data type for each data item.
  - For example, data item within each student record must be an integer between 1 and 5 or that the value of Name must be a string of no more than 30 alphabetic characters.
- It is the database designers' responsibility to identify integrity constraints during database design.
- A data item may be entered erroneously and still satisfy the specified integrity constraints.
  - For example, if a student receives a grade of A but a grade of C is entered in the database, the DBMS cannot discover this error automatically, because C is a valid value for the Grade data type.
  - Such data entry errors can only be discovered manually (when the student receives the grade and complains) and corrected later by updating the database.
  - However, a grade of Z can be rejected automatically by the DBMS, because Z is not a valid value for the Grade data type.

6. Representing Complex Relationships Among Data

- A database may include numerous varieties of data that are interrelated in many ways.
- Each section record is related to one course record as well as to a number of GRADE_REPORT records—one for each student who completed that section.
- A DBMS must have the capability to represent a variety of complex relationships among the data as well as to retrieve and update related data easily and efficiently.

7. Permitting Inferring and Actions Using Rules

- Some database systems provide capabilities for defining deduction rules for inferring new information from the stored database facts.

8. Providing Backup and Recovery

- A DBMS must provide facilities for recovering from hardware or software failures.
- The backup and recovery subsystem of the DBMS is responsible for recovery.
  - For example, if the computer system fails in the middle of a complex update program, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the program started executing.
  - Alternatively, the recovery subsystem could ensure that the program is resumed from the point at which it was interrupted so that its full effect is recorded in the database.

Disadvantages of DBMSs

- Complexity
- Size
- Cost of DBMS
- Additional hardware costs
- Cost of conversion
- Performance
- Higher impact of a failure

Motivations for evolution of Database Management systems

- Commercialization of relational systems
- SQL becomes ``intergalactic standard".
- Various players in research, industry and both scrambling to standardize the "next thing"

Components of DBMS Environment

- Hardware
  - Can range from a PC to a network of computers.
- Software
  - DBMS, operating system, network software (if necessary) and also the application programs.
- Data
  - Used by the organization and a description of this data called the schema.

Components of DBMS Environment

- Procedures
  - Instructions and rules that should be applied to the design and use of the database and DBMS.
  - Back up copies
  - Handling Failures
- People
  - Users

Types of users

- Database Designers - designs conceptual and logical database
- Application Developers - writes application programs that use the database
- Data and Database Administrator
- End - user - interacts with the system from an on-line terminal by using Query Languages etc.

Database System Utilities

- Loading data stored in files into a database. Includes data conversion tools.
- Backup utilities
- Reorganizing database file structures.
- Report generation utilities.
- Performance monitoring utilities.

DBMS Interfaces

- Menu based
- Forms-based, designed for naïve users
- Graphics-based
  - (Point and Click, Drag and Drop, etc.)
- Natural language: requests in written English

Users may be divided into

- Those who actually use and control the database content, and those who design, develop and maintain database applications (called "Actors on the Scene"), and •
- Those who design and develop the DBMS software and related tools, and the computer systems operators (called "Workers Behind the Scene").

Actors on the scene

- **Database administrators:**
  - Responsible for authorizing access to the database, for coordinating and monitoring its use,
  - acquiring software and hardware resources,
  - controlling its use and monitoring efficiency of operations.
- **Database Designers:**
  - Responsible to define the content, the structure, the constraints, and functions or transactions against the database.
  - They must communicate with the end-users and understand their needs.

System Analysts and Applications Programmers (S/W Engineers)

- Analysts Determine requirements of end-users and develop specifications for the required 'canned transactions'
- Programmers implement the specifications as programs.
- Both should have familiarity with capabilities provided by DBMS

End-users:

- They use the data for queries, reports and some of them update the database content.
- End-users can be categorized into:
  - **Casual**: access database occasionally when needed
  - **Naïve or Parametric:** they make up a large section of the end-user population.
    - Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.

Workers Behind The Scene

- **DBMS System Designers:**
  - Persons who design and implement DBMS modules and interfaces as a software package
  - Very complex work
- **Tool Developers:**
  - Design and Implement tools i.e. software packages that facilitate database system design and use e.g. performance monitoring, natural language interface, graphical interfaces, simulation, test data generation etc.
- **Operators and Maintenance Personnel**
  - Responsible for actual running of hardware and software environment for the database system

When Not to Use a DBMS

- Main inhibitors (costs) of using a DBMS:

– High initial investment and possible need for additional hardware.

– Overhead for providing generality, security, concurrency control, recovery, and integrity functions.

When a DBMS may be unnecessary:

- If the database and applications are simple, well defined, and not expected to change.
- If there are stringent real-time requirements that may not be met because of DBMS overhead.
- If access to data by multiple users is not required.