Unit-1 Introduction of Database

- Purpose of database system
- View of data
- Database languages
- Transaction management
- Database administrator
- System structure

Data:

Data is commonly defined as raw facts or observation, typically about physical phenomena or business transactions*. For example of data would be the marks obtained by students in different subjects. Data can be in any form-numerical, textual, graphical, image, sound, video etc.*

The following figure shows the hierarchy of data storage.

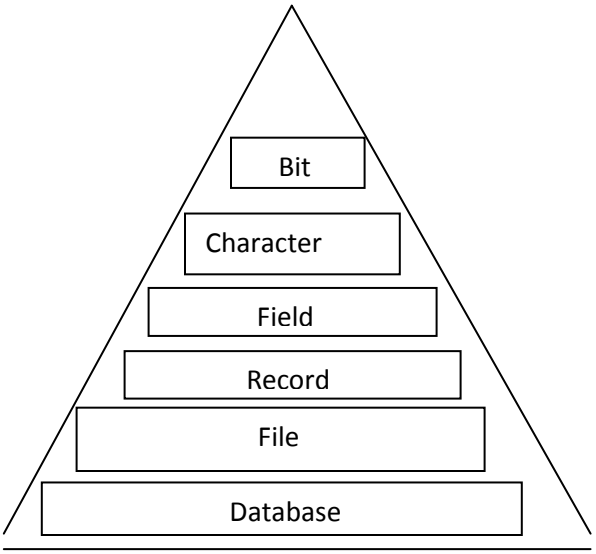


Fig:-Data storage hierarchy

Information:

Information is defined as refined or processed data that has been transformed into meaningful and useful form for specific users. *For example, after processing the marks obtained by student it transformed into information, which is meaningful and from which we can decide which student stood first, second and so forth. Information comes from data and takes the form of table, graphs, diagrams etc.*

Database:

*A **database-management system** (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the **database**, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.*

Database systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results.

Database System Applications

Databases are widely used. Here are some representative applications:

- *Banking: For customer information, accounts, and loans, and banking transactions.*
- *Airlines: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner—terminals situated around the world accessed the central database system through phone lines and other data networks.*
- *Universities: For student information, course registrations, and grades. Credit card transactions: For purchases on credit cards and generation of monthly statements.*
- *Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.*

- *Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.*
- *Sales: For customer, product, and purchase information.*
- *Manufacturing: For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.*
- *Human resources: For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.*

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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

Figure 1.2
A database that stores student and course information.

TRANSCRIPT

(a)

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

COURSE_PREREQUISITES

(b)

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

Figure 1.5
Two views derived from the database in Figure 1.2. (a) The TRANSCRIPT view. (b) The COURSE_PREREQUISITES view.

Figure 1.6
Redundant storage of Student_name and Course_name in GRADE_REPORT. (a) Consistent data. (b) Inconsistent record.

(a)

GRADE_REPORT

Student_number	Student_name	Section_identifier	Course_number	Grade
17	Smith	112	MATH2410	B
17	Smith	119	CS1310	C
8	Brown	85	MATH2410	A
8	Brown	92	CS1310	A
8	Brown	102	CS3320	B
8	Brown	135	CS3380	A

(b)

GRADE_REPORT

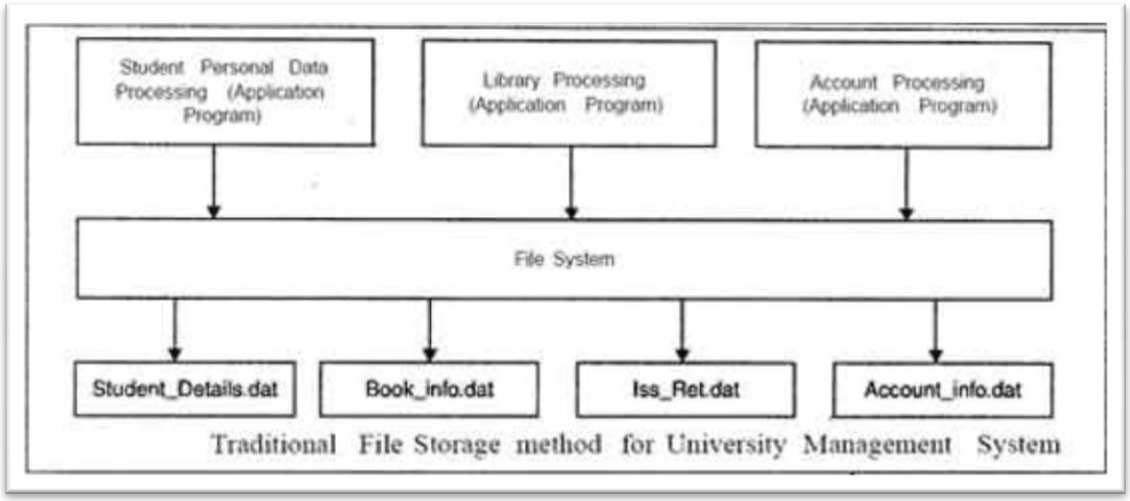
Student_number	Student_name	Section_identifier	Course_number	Grade
17	Brown	112	MATH2410	B

Database Systems versus File Systems

Consider part of a savings-bank enterprise that keeps information about all customers and savings accounts. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information, the- system has a number of application programs that manipulate the files ,including

- *A program to debit or credit an account*
- *A program to add a new account*
- *A program to find the balance of an account*
- *A program to generate monthly statements*

file-processing system *is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Before database management systems (DBMSs) came along, organizations usually stored information in such systems.*



Keeping organizational information in a file-processing system has a number of major disadvantages:

•***Data redundancy and inconsistency***:-*Since different programmers create the files and application programs over a long period, the various files are likely to have different formats and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). For example, the address and telephone number of a particular customer may appear in a file that consists of savings-account records and in a file that consists of checking-account records. This redundancy leads to higher storage and access cost. In addition, it may lead to* ***data inconsistency****; that is, the various copies of the same data may no longer agree. For example, a changed customer address may be reflected in savings-account records but not elsewhere in the system.*

General office	Library	Hostel	Account office
Rollno	Rollno	Rollno	Rollno
Name	Name	Name	Name
Class	Class	Class	Class
Father_Name	Address	Father_Name	Address
Date of birth	Date_of _birth	Date_of_birth	Phone_No
Address	Phone_No	Address	Fee
Phone_No	No_of_books_issued	Phone_No	Installments
Previous_Record	Fine	Mess_Bill	Discount
Attendance	etc.	Room No	Balance
Marks		etc.	Total
etc.			etc.

- ***Difficulty in accessing data****:-Suppose that one of the bank officers needs to find out the names of all customers who live within a particular postal-code area. The officer asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of all customers. The bank officer has now two choices: either obtain the list of all customers and extract the needed information manually or ask a system programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same officer needs to trim that list to include only those customers who have an account balance of $10,000 or more. As expected, a program to generate such a list does not exist. Again, the officer has the preceding two options, neither of which is satisfactory.*

- ***Data isolation****. Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.*

- ***Integrity problems****. The data values stored in the database must satisfy certain types of* ***consistency constraints****. For example, the balance of a bank account may never fall below a prescribed amount (say, $25). Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.*

- **Atomicity problems**. A computer system, like any other mechanical or electrical device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure. Consider a program to transfer $50 from account A to account B. If a system failure occurs during the execution of the program, it is possible that the $50 was removed from account A but was not credited to account B, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur. That is, the funds transfer must be atomic—it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

- **Concurrent-access anomalies**. For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data. Consider bank account *A*, containing $500. If two customers withdraw funds (say $50 and $100 respectively) from account *A* at about the same time, the result of the concurrent executions may leave the account in an incorrect (or inconsistent) state. Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value $500, and write back $450 and $400, respectively. Depending on which one writes the value last, the account may contain either $450 or $400, rather than the correct value of $350. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated *previously.*

- **Security problems**. Not every user of the database system should be able to access all the data. For example, in a banking system, payroll personnel need to see only that part of the database that has information about the various bank employees. They do not need access to information about customer accounts. But, since application programs are added to the system in an ad hoc manner, enforcing such security constraints is difficult.

Characteristics of data in database:

The data in a database should have the following features:

- **Shared:** Data should be sharable among different users and applications.
- **Persistence:** Data should exist permanently in the database. Changes in the database must not be lost because of any failure.
- **Validity/Integrity/Correctness:** It should maintain the integrity so that there is always correct data in the database.
- **Security:** Data should be protected from unauthorized access.
- **Non-redundancy:** Data should not be repeated.
- **Consistency:** A consistent state of the database satisfies all the constraints specified in the database. Data in a database is consistent if any changes in the database take the database from one consistent state to another.
- **Independence:** The three levels in the schema should be independent of each other so that the changes in the schema in one level should not affect the other levels.

Purpose of DBMS (Functions of DBMS):

The benefits of using DBMS are:

- ***To reduce redundancy:*** *Repeating of the same information in database is called redundancy of data which leads to several problems such as wastage of space, duplication effort for entering data and inconsistency. When DBMS is used and database is created, redundancy is minimized.*
- ***To avoid inconsistency:*** *The database is said to be inconsistent if various copies of the same data may no longer agree. For example, a changed customer address may be reflected in saving account but not elsewhere in the system. By using DBMS we can avoid inconsistency.*
- ***To share data:*** *The data in the database can be shared among many users and applications. The data requirements of new applications may be satisfied without having to create any new stored files.*
    - ***To provide support for transactions:*** *A transaction is a sequence of database operations that represents a logical unit of work. It accesses a database and transforms it from one state to another. A transaction can update a record, delete one, modify a set of records etc. when the DBMS does a 'commit', the changes made by the transaction are made permanent. We can roll back the transaction to undo the effects of transaction.*
    - ***To maintain integrity:*** *Most database applications have certain integrity constraints that must hold for the data. A DBMS provides capabilities for defining and enforcing these constraints. For example, the value of roll number field of each student in student database should be unique for each student. It is a type of rule. Such a rule is enforced using constraint at the time of creation of database.*
    - ***To enforce security:*** *Not every user of the database system should be able to access all data. Different checks can be established for each type of access (retrieve, modify, delete, etc) to each piece of information in the database.*
    - ***To provide efficient backup and recovery:*** *Provide facilities for recovering from software and hardware failures to restore database to previous consistent state.*
    - ***To Concurrent Access Database:*** *Concurrent access means access to the same data simultaneously by more than one user. The same data may be used by many users for the purpose reading at the same time. But when a user tries to modify a data, there should be a concurrency control mechanism to avoid the inconsistency of data. A DBMS provides facilities for these operations.*

Disadvantages of DBMS

- ***Problem associated with centralization:*** *Centralization increases the security problems.*
- ***Cost of software:*** *Today's there are several softwares which are very costly. Hence from economic point of view it is the drawback.*
- ***Cost of hardware:****To support various software some upgraded hardware components are needed.Hence from economic point of view it is the drawback.*
- ***Complexity of backup and recovery:****DBMS provides the centralization of the data, which requires the adequate backups of data.*
- *Overhead for providing generality, security, recovery, integrity, and concurrency control.*

- *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.*

Differences bet^{n} DBMS and file processing system:

DBMS	File processing system
1. *A Database Management System (DBMS) is a collection of interrelated data and the set of programs to access those data.*	1. *A flat file system stores data in a plain text file.A flat file is a file that contains records, and in which each record is specified in a single line.*
2. *Data redundancy problem is not found.*	2. *Data redundancy problem exist.*
3. *Data inconsistency does not exist.*	3. *Data inconsistency may exist.*
4. *Accessing data from database is easier.*	4. *Accessing data from database is comparatively difficult.*
5. *The problem of data isolations is not found.*	5. *Here data are scattered in various files and formats so data isolation problem exist.*
6. *Atomicity and integrating problems are not found.*	6. *Here these problems are found.*
7. *Data are more secure.*	7. *Data are less secure.*
8. *Concurrent access and crash recovery.*	8. *Here there is no concurrent access and no recovery.*

DataBase User And Administration

Database users:

Database users are the one who really use and take the benefits of database. There will be different types of users depending on their need and way of accessing the database.

1. ***Native Users*** *– these are the users who use the existing application to interact with the database. For example, online library system, ticket booking systems, ATMs etc which has existing application and users use them to interact with the database to fulfill their requests.*

2. ***Application Programmers –*** *They are the developers who interact with the database by means of DML queries. These DML queries are written in the application programs like C, C++, JAVA, Pascal etc. These queries are converted into object code to communicate with the database. For example, writing a C program to generate the report of employees who are working in particular department will involve a query to fetch the data from database. It will include a embedded SQL query in the C Program.*

3. ***Sophisticated Users –*** *They are database developers, who write SQL queries to select/insert/delete/update data. They do not use any application or programs to request the database. They directly interact with the database by means of query language like SQL. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirement. In short, we can say this category includes designers and developers of DBMS and SQL.*

4. ***Specialized Users –*** *These are also sophisticated users, but they write special database application programs. They are the developers who develop the complex programs to the requirement. They are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, knowledge base and expert systems, systems that store data with complex data types (graphics data and audio data), and environment-modeling systems.*

Database Administrator:

The person who has such central control over the system is called the database administrator (DBA). The function of the DBA includes the following:

1. ***Schema definition**: The DBA creates the original database schema by writing a set of definitions that is translated by the DDL compiler to a set of tables that is stored permanently in the data dictionary.*

2. ***Schema and physical-organization modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs to the organization, or to alter the physical organization to improve performance.*

3. ***Granting the authorization of data access**: The granting of different types of privileges to the database users so that all the users are not able to all data.*

4. ***Integrity-constraint specifications**: The data values stored in the database must satisfy certain consistency constraints. The database administrator must specify such a constraint explicitly.*

5. ***Routine maintenance:** Routine maintenance includes periodic backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding, Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required etc.*

Data Abstraction

Data Abstraction:

The system hides certain details of how the data are stored and maintained and such view is an abstract view.

- ✓ *The Database System provides users with an abstract view of the data.*

***The database designers use the complex data structure to represent the data in the database and developer hides the complexity from user from several level of abstraction such as physical level, logical level, and view level. This process is called data abstraction**.*

Levels of Data Abstraction:

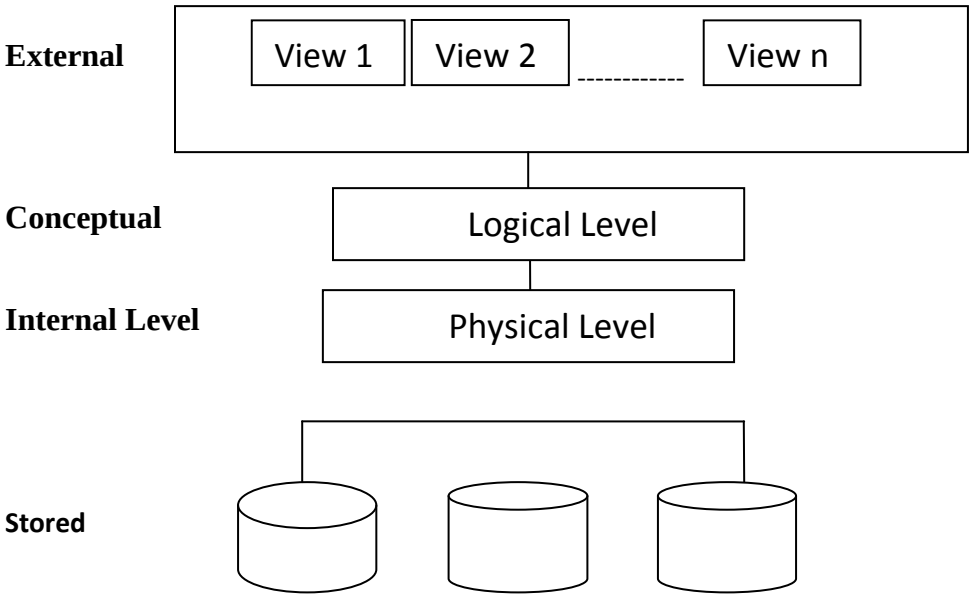


Fig: Three levels of data abstraction

Physical level

- ✓ *It is the lowest level of abstractions describes how the data are actually stored.*
- ✓ *The physical level describes complex low level data structure in details.*
- ✓ *At this level records such as customer, account can be described as a block of consecutive storage location (e.g. byte, word)*
- ✓ *The database system hides many of the lowest level storage details from database programmer. Database administrator may be aware of certain details of the physical organization of the data.*

Logical level

- ✓ *It is the next higher level of data abstraction which describes what data are stored in the database, and what relationships exist among those data.*
- ✓ *At the logical level , each record is described by a type definition*
- ✓ *Programmers and database administrator works at this level of abstraction.*

View level

- ✓ ***It is the highest level of abstraction describes only a part of the database and hides some information to the user. This level describes the user interaction with database system.***
- ✓ ***At view level, computer users see a set of application programs that hide details of data types. Similarly at the view level several views of the database are defined and database user see only these views.***
- ✓ ***Views also provides the security mechanism to prevent users from accessing certain parts of the database (that is views can also hide information (such as an employee's salary) for security purposes.)***

Example: **Let's say we are storing customer information in a customer table. At physical level these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers**.

At the logical level these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

At view level, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

Instances , Schema And Data Independence

Instance (Database State):

The collection of information stored in the database at a particular moment is called an instance of the database. *It is the actual content of the database at a particular point in time*

✓ *The term instance is also applied to individual database components, e.g. record instance, table instance, entity instance***.**

Initial Database State

Refers to the database state when it is initially loaded into the system.

Valid State

A state that satisfies the structure and constraints of the database.

Schema:

The overall design of the database which is not expected to change frequently is called database schema. *Simply, the database schema is the logical structure of the database.*

- ✓ *The concept of database schema and instances can be understood by analogy to a program written in a programming language*
- ✓ *A database schema corresponds to the variable declaration and the values of the variables in a program at a point in time correspond to an instance of a database.*
- ✓ *The database systems have several schemas and partitioned according to the level of abstraction such as physical and logical schema.*

STUDENT

Name	Student-number	Class	Major

Fig: Schema diagram for Student

Note:

- ✓ The ***database schema*** changes very infrequently.
- ✓ The ***database state*** changes every time the database is updated.

Data Independence:

The three schema architecture further explains the concept of data independence, the capacity to change the schema at one level without having to change the schema at the next higher level.

- ➢ **Logical Data Independence**
- ➢ **Physical Data Independence**

Logical Data Independence:

The capacity to change the conceptual schema without having to change the external schemas and their associated application programs is called logical data independence. When modification is done to the conceptual schema (i.e tables) the mapping called "external mapping" is changes automatically by DBMS.

Physical Data Independence:

The capacity to change the internal schema without having to change the conceptual schema is called physical data independence. When a schema at a lower level is changed, only the ***mappings*** *between this schema and higher-level schemas need to be changed in a DBMS. This mapping is called "logical mapping".*

For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance.

Examples of changes under Physical Data Independence

Due to Physical independence, any of the below change will not affect the conceptual layer.

- **Using a new storage device like Hard Drive or Magnetic Tapes**
- **Modifying the file organization technique in the Database**
- **Switching to different data structures.**
- **Changing the access method.**
- **Modifying indexes.**
- **Changes to compression techniques or hashing algorithms.**
- **Change of Location of Database from say C drive to D Drive**

Examples of changes under Logical Data Independence

Due to Logical independence, any of the below change will not affect the external layer.

1. **Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs**
2. **Merging two records into one**
3. **Breaking an existing record into two or more records**

Data Model

Database Models:

Data model is a collection of tools for describing data, data relationships, data semantics and data constraints. The database model refers the way for organizing and structuring the data in the database. Traditionally, there are different database models which are used to design and develop the database of the organization.

1. **Entity- Relationship Model**
2. **Object oriented Model**
3. **Relational Model**
4. **Hierarchical model**
5. **Network Model**
6. **Object Relational Data Model**

Object based data

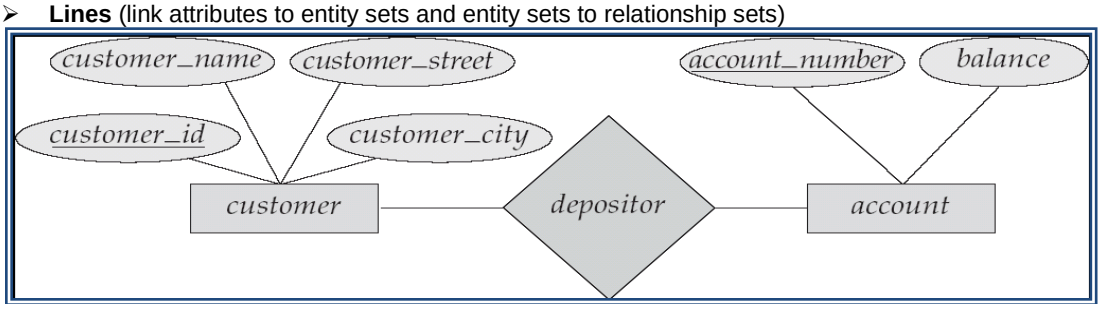
Record based data model

Entity- Relationship Model:

The E-R data models is based on a perception of real world that consist of a collection of basic objects called entities and relationship among these objects. In an E-R model a **database** can be modeled as a collection of **entities, and relationship** among entities.

Overall logical structure of a database can be expressed graphically by E-R diagram. The basic components of this diagram are:

- **Rectangles** (represent entity sets)
- **Ellipses** (represent attributes)
- **Diamonds** (represent relationship sets among entity sets)
- **Lines** (link attributes to entity sets and entity sets to relationship sets)



2. Relational Model:

It is the current favorite model. The relational model is a lower level model that uses a collection of tables to represent both data and relationships among those data. Each table has multiple columns, and each column has a unique name. Each table corresponds to an entity set or relationship set, and each row represents an instance of that entity set or relationship set. Structured query language (SQL) is used to manipulate data stored in tables.

3. Object oriented data model:

The object oriented data model is based on object-oriented programming paradigm. It is based on the concept of encapsulating the data and the functions that operate on those data in a single unit called an object. The internal parts of objects are not visible externally.
Here one object communicates with other objects by sending message.

4. Hierarchical data model:

In a hierarchical data model, the data elements are linked in the form of an inverted tree structure with the root at the top and the branches formed below. Below the single root data element are subordinate elements, each of which, in turn, has one or more other elements. There is a parent child relationship among the data elements of a hierarchical database. There may be many child elements under each parent element, but there can be only one parent element for any child element. The branches in the tree are not connected.

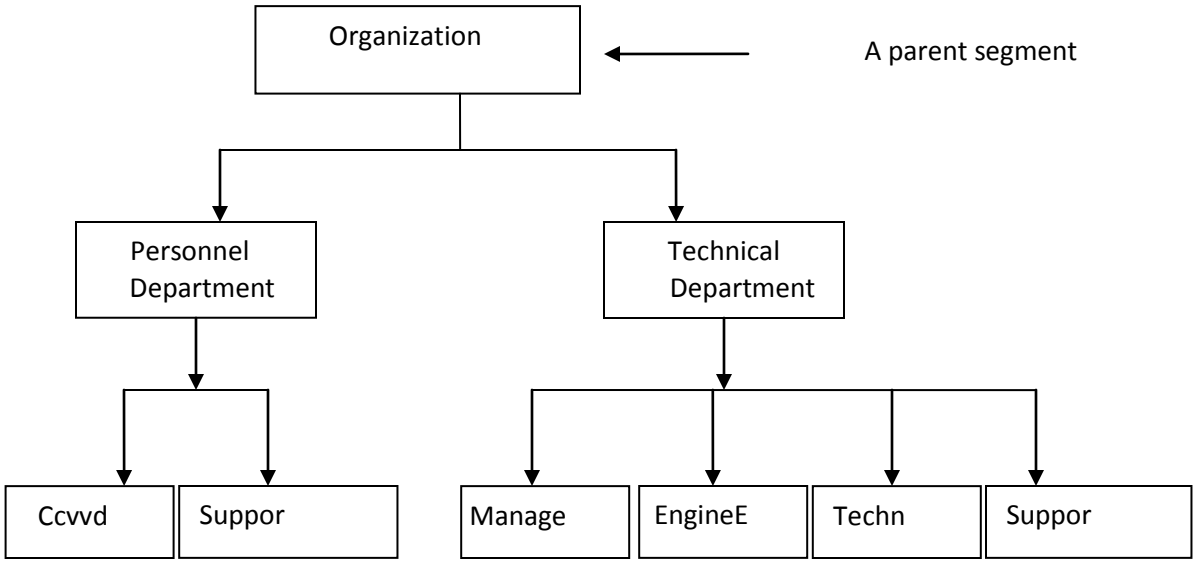
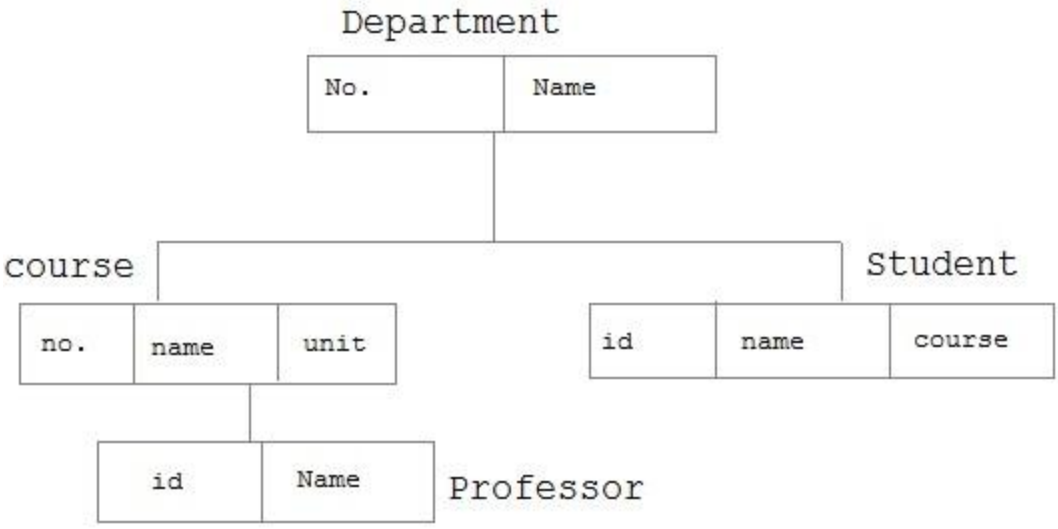


Fig: An example of hierarchical database model



The main limitation of this structure is that it does not support flexible data access, because data can be accessed only by following the path down the tree structure.

Advantages of hierarchical database model

- It is the easiest model of database.
- It is secure model as nobody can modify the child without consulting to its parent.
- Searching is fast.
- Very efficient in handling one- to- many relationship.

Disadvantages hierarchical database model

- It is old fashion, outdated database model
- Modification and addition of child without consulting its parent is impossible.
- Cannot handle many- to- many relationships.
- Increase redundancy.
- It does not support flexible data access, because data can be accessed only by following the path down the tree structure.

5. Network Data Model:

A network data model is an extension of the hierarchical database structure. In this model also, the data elements of a database are organized in the form of parent-child relationships and all types of relationships among the data elements must be determined when the database is first designed. In a network database, a child data element can have more than one parent element or no parent at all. Moreover, in this type of database, the database management system permitsthe extraction of the needed information from any data element in the database structure, instead of s3tarting from the root data element.

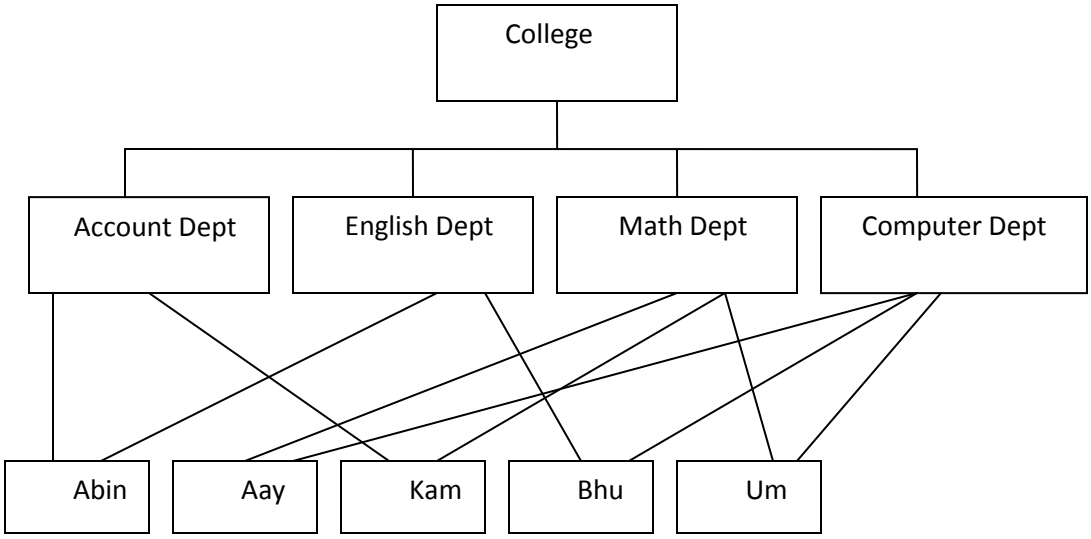


Fig: An example of a network database

Advantages of network database model

- More flexible than hierarchical database because it accept many to many relationship.
- Searching is faster because of multidirectional pointers.
- Promotes database integrity
- Data independence

Disadvantages of network database model

- Less secure than hierarchical as it is open to all.
- Need long program to handle the relationship.
- Pointer is used in this database and that increased the overhead of storages

Transaction management

Transaction Management:

Collections of operations that form a single logical unit of work are called transactions. For example, a transfer of funds from a checking account to a savings account is a single operation from the customer's standpoint; within the database system, however, it consists of several operations. A database system must ensure proper execution of transactions despite failures—either the entire transaction executes, or none of it does.

Consider the transaction,

```
A = A – 50;
Write (A);
Read (B);
B = B+50;
Write (B);
```

To ensure integrity of the data, we require that the database system maintain the following properties of the transactions:

- ✓ **Atomicity:** Either all operations of the transaction are reflected properly in the database or none are. Suppose during the execution of above transaction a failure occurred after the write (A) operation but before

write (B) operation. Then the values of amount reflected in database will be 950 and 2000. The system destroyed 50 as a result of failure.

- ✓ **Consistency:** Database must be in correct state before and after execution of the transaction. The consistency requirement here is that sum of A and B be unchanged by the execution of transaction. Without the consistency requirement, money could be created or destroyed by a transaction.

- ✓ **Isolation:** Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions T_i and T_j it appears to T_i that either T_j finished execution before T_i started, or T_j started execution after T_i finished. Thus, each transaction is unaware of other transactions executing concurrently in the system.

- ✓ **Durability:** After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

These properties are called ACID properties, with acronym derived from the first letters of the above four properties.

Transactions access data using two operations:

- ✓ ***Read(x):*** *Which transfers the data item x from the database to a local buffer belonging to the transaction that executed the read operation.*
- ✓ ***Write(x):*** *Which transfers the data item x from the local buffer of the transaction that executed to the database.*

DBMS-Architecture

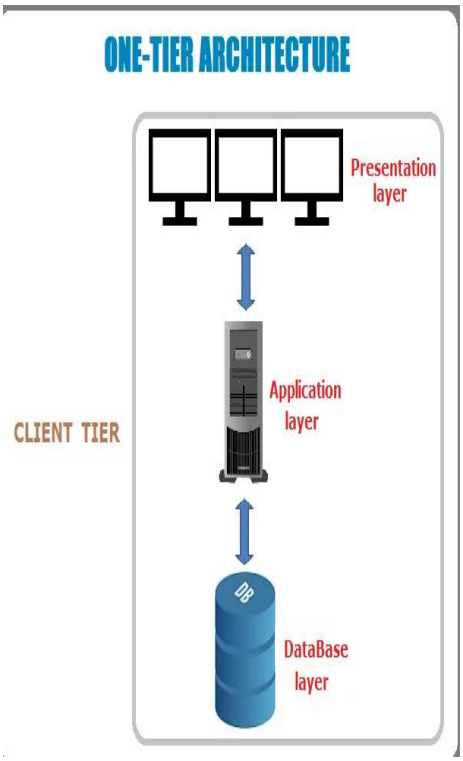
Database Architecture:

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent ***n*** *modules, which can be independently modified, altered, changed, or replaced. It helps in design , development, implementation and maintenance of a database.*

1.**One-tier Architecture:**

Simplest of database where the client , server and database all reside on the same machine or computer.

One tier architecture has all the layers such as Presentation, Business, Data Access layers in a single software package. Applications which handles all the three tiers such as MP3 player, MS Office are come under one tier application. The data is stored in the local system or a shared drive.



2.Two-tier client-Server Architecture:

A two-tier architecture is a database architecture where,

1.Presentation layer runs on client.(PC,Mobile)

2.Data is stored in a server.

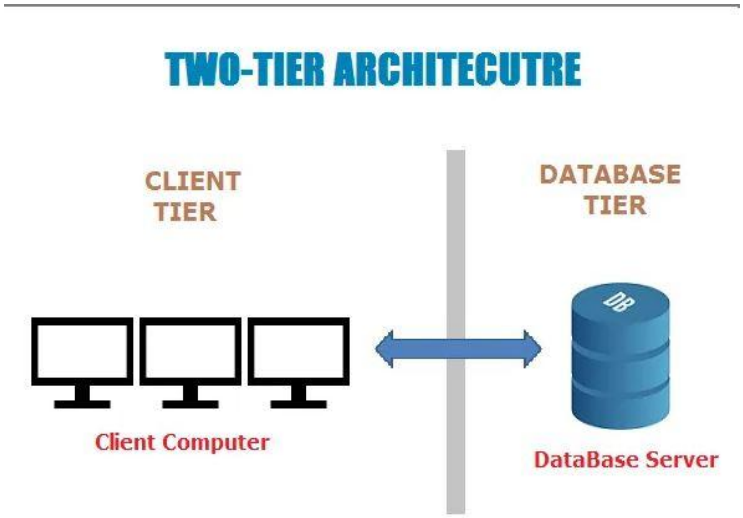
-Clients handle presentation and application layers and server system handles database layer.

- Clients directly interact with the database.

-An interface called ODBC(Open database Connectivity) provides an API that allow client side program to call the DBMS.

-Direct Communication and faster Communication .

-2-tier architecture provides added security to the dbms as it is not exposed to the end user directly.



3.Three-tier Architecture:

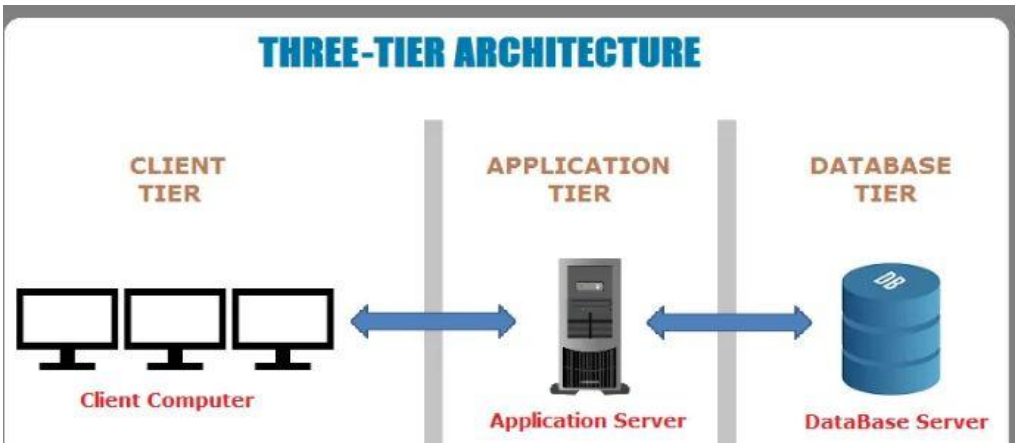
-Extensions of 2-tier architecture.3-tier architecture has following layers .

1.Presentation layer

2.Application layer

3.Database server

- Client system handles Presentation layer, Application server handles Application layer and Server system handles Database layer.



-Application layer between the users and the dbms is responsible for communicating users request to the dbms system and send response from dbms to the server.

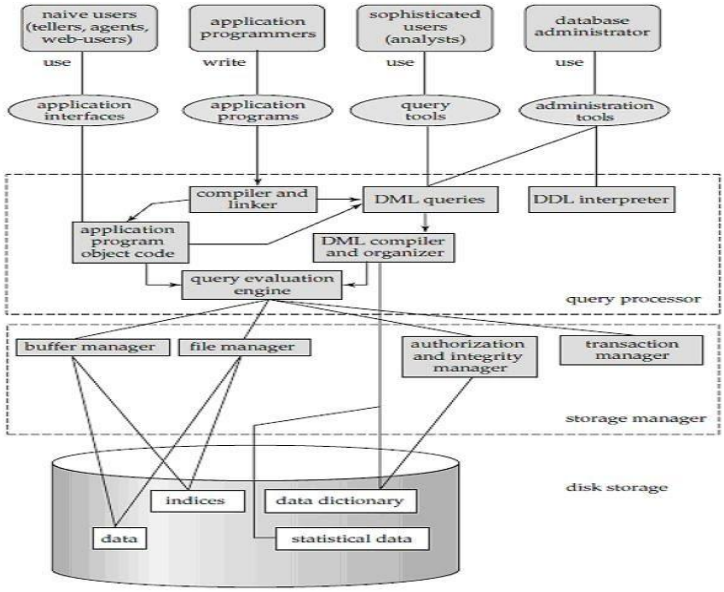
-Application layer(business logic layer) also processes functional logic, Constraint and rules before passing data to the user or down to the dbms.

-Used for web application

Database System Structure &Components

Database Structure:

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the **storage manager** and the **query processor** components. The storage manager is important because databases typically require a large amount of storage space. The query processor is important because it helps the database system simplify and facilitate access to data.



Query Processor

The query processor components include:

-**DDL interpreter,** which interprets DDL statements and records the definitions in the data diction

-**DML compiler,** which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs **query optimization**, that is, it picks the lowest cost evaluation plan from among the alternatives.

-**Query evaluation engine,** which executes low-level instructions generated by the DML compiler.

Storage Manager

A storage manager is a program module that provides the interface between the lowlevel data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The raw data are stored on the disk using the file

system, which is usually provided by a conventional operating system. The storage manager translates the various DML statements into low-level file-system commands. Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

The storage manager components include:

- **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
- **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
- **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
- **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

Components

The database management system can be divided into five major components, they are:

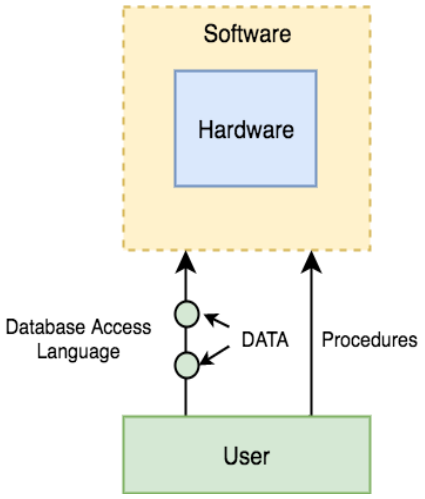
1.Hardware

2.Software

3.Data

4.Procedures

5.Database Access Language



```
36
37    @PostMapping(path="/add") // Map ONLY POST Requests
38    public @ResponseBody String addNewUser (@RequestBody User1 user1)
39        {
40      // @ResponseBody means the returned String is the response, not a view name
41      // @RequestParam means it is a parameter from the GET or POST request
42
43
44
45      user1Repository.save(user1);
46      return "save";
47    }
48    @PostMapping(path="/save")
49      public User1 saveOrUpdate(@RequestBody User1 user1){
50          return user1Repository.save(user1);
51
52      }
53
54
55
56
57    @GetMapping(path="/all")
58    public @ResponseBody Iterable<User1> getAllUsers() {
59      // This returns a JSON or XML with the users
60      return user1Repository.findAll();
61    }
62    @DeleteMapping(path="delete/{id}")
63     public void  deleteById(@PathVariable int id){
64        //int uid=Integer.parseInt(id);
```

Fig:application programmer

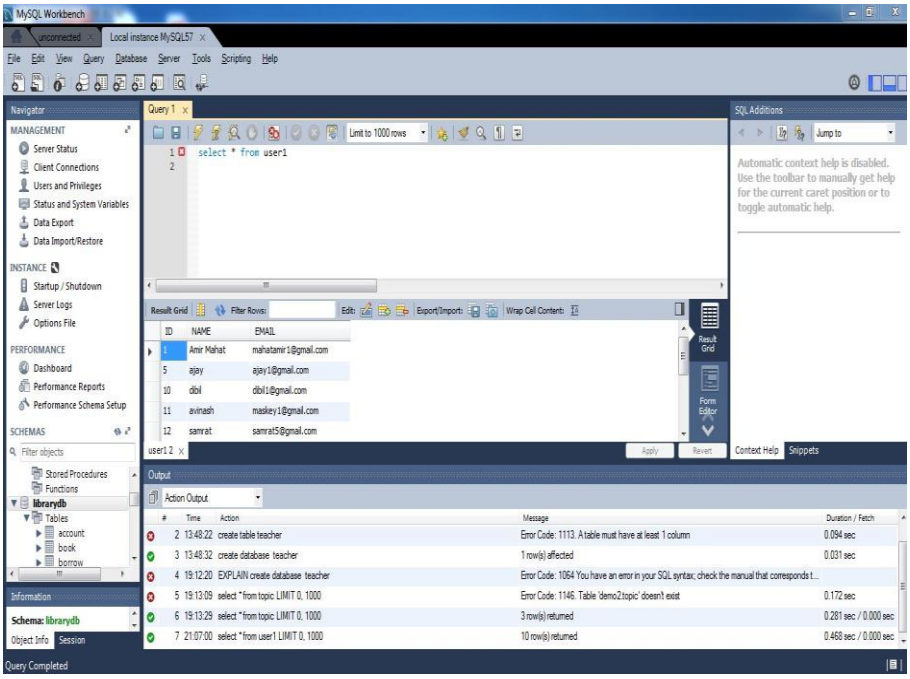


Fig:sophisticated users

Data Dictionary

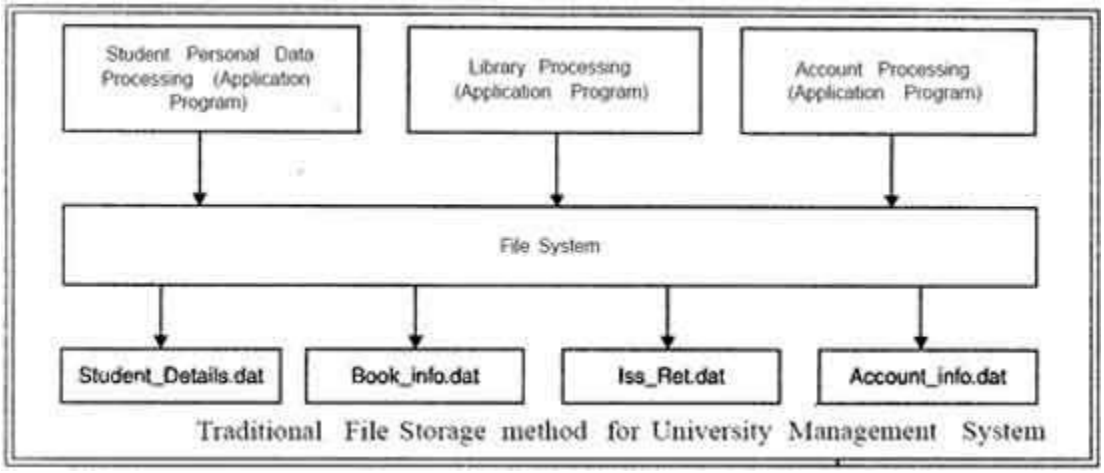
Data Dictionary outlining a Database on Driver Details in NSW

Field Name	Data Type	Data Format	Field Size	Description	Example
License ID	Integer	NNNNNN	6	Unique number ID for all drivers	12345
Surname	Text		20	Surname for Driver	Jones
First Name	Text		20	First Name for Driver	Arnold
Address	Text		50	First Name for Driver	11 Rocky st Como 2233
Phone No.	Text		10	License holders contact number	0400111222
D.O.B	Date / Time	DD/MM/YYYY	10	Drivers Date of Birth	08/05/1956

Sample Student File

Student ID	First Name	Last Name	Address	City	State	Postal Code	E-mail Address	Date Admitted	Major	Photo
2295	Milton	Brewer	54 Lucy Court	Charlestown	IN	46176		6/10/2010	EE	mbrewer.jpg
3876	Louella	Drake	33 Timmons Place	Bonner	IN	45208	lou@world.com	8/9/2010	BIO	ldrake.jpg
3928	Adelbert	Ruiz	99 Tenth Street	Sheldon	IN	46033		10/8/2010	CT	aruiz.jpg
2872	Benjamin	Tu	2204 Elm Court	Rowley	IN	46167	tu@indi.net	11/6/2010	GEN	btu.jpg

records key field fields fields



Traditional File Storage method for University Management System