ST. XAVIER'S COLLEGE

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**DATABASE MANAGEMENT SYSTEM**

**LAB ASSIGNMENT # 1**

**SUBMITTED BY:**

Abhishek Tamrakar  
013BSCCSIT003

4th Semester

**SUBMITTED TO:**

|  |  |
| --- | --- |
| **Er. Sanjay Kumar Yadav**  Supervisor,  Department of Computer Science  St. Xavier’s College |  |

**Date of Submission:** 24thJuly 2015

1. **PURPOSE OF DATABASE SYSTEM**

File processing system was used to manage information. It stores data in various files of different application programs to extract or insert data to appropriate file. File processing system has several drawbacks due to which database management system is required. Database management system removes problems found in file processing system. Some major problems of file processing systems are:

1. Data redundancy and inconsistency

In file processing system, different programmer creates files and writes application programs to access it. After a long period of time files may exist with different formats and application programs may written in many different programming languages. Moreover, same information may be duplicated in several files. We have to pay for higher storage and access cost for such redundancy. It may leads database in inconsistent state because update made in one file may reflected in one file but it may not reflected in another files where same information exist in another files.

1. Difficulty in accessing data

In file processing system, we cannot easily access required data stored in particular file. For each new task we have to write a new application program. File processing system cannot allow data to be retrieve in convenient and efficient manner.

1. Data isolation

Since data are scatter in different files and data may stored in different format, so it is difficult to write program to retrieve appropriate data.

1. Integrity problem

In database, we required to enforce certain type consistency constraints to ensure the database correctness or to enforce certain business rules. It is in fact called integrity constraints (e.g. account balance > 0), integrity of database need not to be violated. In file processing system, integrity constraint becomes the part of application program. Programmer need to write appropriate code to enforce it. When new constraints are required to add or change existing one, it is difficult to change program to enforce it.

1. Atomicity problem

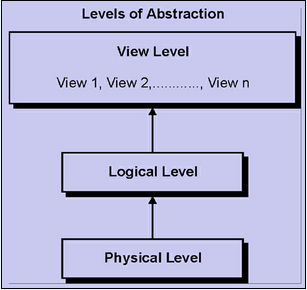
Failures may lead database in an inconsistent state with partial updates. For example, failure occurs while transferring fund from account A to B. There would be the case that certain amount from account A is retrieved and it is updated but failure occurs just before it is deposited to account B, such case may lead database in inconsistent state.

1. Concurrent access problem

Concurrent accessed increase the overall performance of system providing fast response time but uncontrolled concurrent accesses can lead inconsistencies in system. File processing system allow concurrent access but it is unable to coordinate different application programs so database may lead in inconsistent state. E.g. two people reading a balance and updating it at the same time

1. **VIEW OF DATA**

It is the highest level of abstraction. It describes only part of the entire database. It simplifies interaction with the system. It allows database system to provide many views for the same database. That is it allows each user/application to get different perspective of the database.



**INSTANCES AND SCHEMAS**

Similar to types and variables in programming languages

* Schema – the logical structure of the database
* e.g., the database consists of information about a set of customers and accounts and the relationship between them)
* Analogous to type information of a variable in a program
* Physical schema: database design at the physical level
* Logical schema: database design at the logical level
* Instance – the actual content of the database at a particular point in time
* Analogous to the value of a variable
* Physical Data Independence – the ability to modify the physical schema without changing the logical schema
* Applications depend on the logical schema

In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

1. **DATABASE LANGUAGE**

Data definition language used to specify database scheme. For example, following DDL statement in SQL defines account relation.

create table account

(

account\_no char(2), balance integer

)

The execution of above DDL statement creates table account. Moreover, it updates special set of tables called data dictionary or data directory. Data dictionary contains Meta data that is data about data. For example table containing tables’ information like table name, owner, created date, modified date etc refers data dictionary and contain information are example of Meta data. Data definition language also allows to define storage structure and access methods for database system, such special set of DDL statement called data storage and definition language.

**Data manipulation language**

Data manipulation language allow database user to access (query) and manipulate data. That is, DML is responsible for

• Retrieval of information from the database

• Insertion of new information into the database

• Deletion of information in the database

• Modification of information in the database

DML established communication between user and database.

There are two types of DML

(a) Procedural DML: user required to specify what data are needed and how they get those data.

(b) Nonprocedural (Declarative) DML: user only required to what data needed without specifying how to get those data.

Declarative DMLs are usually easier to learn and use than procedural DMLs. However, since a user does not have to specify how to get data, the database system has to figure out an efficient means of accessing data. The DML component of SQL is nonprocedural. A query is statement requesting the retrieval of information. Special set of DML which only use to retrieve information from database called query language. Example: Select customer\_name from customer where customer.customer\_id=’c001’. This query retrieves those rows from table customer where the customer\_id=c01

1. **RELATIONAL DATABASE**

Object based logical model describe data at the logical and view levels. It has flexible structuring capabilities. It allows specifying data constraints explicitly. Under object-based logical model there are sever data models

• Entity-relationship model

• Object-oriented model

**Entity Relationship Model**

E-R model describes the design of database in terms of entities and relationship among them. An entity is a “thing” or “object” in real world that are distinguishable from other objects. An entity is describes by a set of attributes.

For example

• Attributes account\_number and balance may describe entity “account”.

• Attributes customer\_id, customer\_name, customer\_city may describe entity “customer”.

A relationship is an association among several entities. For example, a depositor relationship associates a customer with each account he or she has.

The set of all entities of same type called entity set and similarly set of all relationship of the same type called relationship set.

E-R model graphically express overall logical structure of a database by an E-R diagram. Components of E-R diagram are as follows

Rectangles: represent entity sets

Ellipses: represent attributes

Diamonds: represent relationships among entity sets lines: link attributes to entity sets and entity sets to relationships.

**Object oriented model**

Object oriented data model is extension to E-R model with the notion of encapsulation, methods (functions) and object identity. It is based on collection of objects, like the E-R model. An object contains values stored in instance variables within the object. These values are themselves objects. That is, objects can contain objects to an arbitrarily deep level of nesting. An object also contains bodies of code that operate on the object. Objects that contain the same types of values and the same methods are grouped into classes. The only way in which one object can access the data of another object is by invoking the method of that other object. This is called sending a message to the object. Internal parts of the object, the instance variables and method code, are not visible externally.

Example: Consider an object representing a bank account.

• The object may contain instance variables account\_number and balance.

• The object may contain a method pay-interest which adds interest to the balance.

Unlike entities in the E-R model, each object has its own unique identity. It is independent to the values it contains. Two objects containing the same values are distinct. Distinction is maintained in physical level by assigning distinct object identifier.

1. **DATABASE DESIGN**

**Database design** is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and view. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).

The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

* Determine the relationships between the different data elements.
* Superimpose a logical structure upon the data on the basis of these relationships

1. **DATA STORAGE AND QUERYING**

The database manager is a program module which provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system. Since database required lots of storage space so it must be stored on disks. Data need to move between disk and main memory as needed.

Since the goal of database system is to simplify and facilitate access to data providing optimal performance as far as possible. So the database manager module is responsible for

• Interaction with the file manager: responsible to translate DML statements into low-level file system commands for storing, retrieving and updating data in the database.

• Integrity enforcement: responsible to check any updates in the database do not violate consistency constraints.(e.g. no bank account balance below $25).

• Security enforcement: responsible to ensure that users only have access to information they are permitted to see.

• Backup and recovery: Detecting failures due to power failure, disk crash, software errors, etc., and restoring the database to its state before the failure.

• Concurrency control: responsible to preserving data consistency when there are concurrent users.

**Querying**

Queries are one of the things that make databases so powerful. A "query" refers to the action of retrieving data from your database. Usually, you will be selective with how much data you want returned. If you have a lot of data in your database, you probably don't want to see everything. More likely, you'll only want to see data that fits a certain criteria.

For example, you might only want to see how many individuals in your database live in a given city. Or you might only want to see which individuals have registered with your database within a given time period.

As with many other tasks, you can query a database either programmatically or via a user interface.

The 2nd query only returns records where the value in the "First Name" column equals "Homer". Therefore, if only one individual in our database had the name "Homer", that person's record would be shown.

Something like this:



Results of a database query

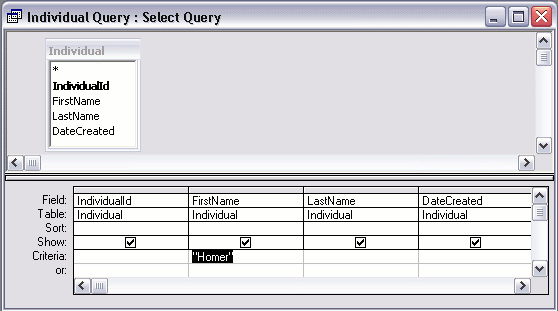
SQL is a powerful language and the above statement is very simple. You can use SQL to choose which columns you want to display, you could add further criteria, and you can even query multiple tables at the same time. If you're interested in learning more about SQL, be sure to check out our SQL tutorial after you've finished this one!

Option 2: User Interface

You might find the user interface easier to generate your queries, especially if they are complex.

Database management systems usually offer a "design view" for your queries. Design view enables you to pick and choose which columns you want to display and what criteria you'd like to use to filter the data.

Here's an example of design view in Microsoft Access:



Database query - design view

When using design view, the database system actually uses SQL (behind the scenes) to generate the query.

1. **TRANSACTION MANAGEMENT**

A transaction is a collection of operations that performs a single logical function in a database application

■ Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

■ Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

1. **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.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3-tier Architecture

3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.



* **Database (Data) Tier** − at this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
* **Application (Middle) Tier** − at this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
* **User (Presentation) Tier** − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

1. **DATABASE USERS AND ADMINISTRATORS**

**Database Users**

There are four different types of database users; they are differentiated according to their interaction with the system. Moreover, there are different types of user interfaces for different types of users.

* 1. Naïve Users:

Naïve users are unsophisticated users who interact with the system by invoking one of the application programs that are already written. For example, banks teller who needs to transfer fund from one account to another invoking a program called transfer. This program asks the teller for the amount of money to be transferred, and account to which the money is to be transferred. The typical user interface for the native user is a form interface, where user can fill appropriate fields of the form. Native users may also simply read reports generated from the database.

* 1. Application programmers:

Application programmers are computer professional who write application programs. Application programmers may choose any programming tool to develop user interfaces. They can also used RAD tools that enable an application programmer to construct forms and reports without writing the program. There are also special type of programming languages that combine imperative control structures (e.g. for loops, while loops and if-then-else statements) with the statements of data manipulation language. These languages are sometimes called fourth generation languages. It often includes special features to facilitate the generation of forms and display data on the screen. Most major commercial database system includes a fourth generation language.

* 1. Sophisticated users:

Sophisticated user interact with system without writing programs but they requests by writing queries in database using DML query language. This query goes to query processor and it converted into instructions for the database manager module.

* 1. Specialized users:

Specialized users are responsible to write special database application programs it could be computer-aided design systems, knowledge based and expert systems that store data with complex data types (e.g. graphics data, audio/video data).

**Database Administrator**

The database administrator is a person having central control over data and programs accessing that data. Database administrator has the following responsibility:

• Schema definition: responsible for the creation of original database schema. So DBA is responsible to write data definition statements in DDL.

• Storage structure and access method definition: DBA is responsible to write a set of definitions to define storage and access method using storage and access.

• Schema and physical organization modification: DBA is responsible for modification of schema and to reflect the changes in schema or to improve the performance physical organization may need to be change.

• Granting authorization for data access: DBA is responsible to grant different types of authorization for data access to various users.

• Routine maintenance: periodically backing up the database ensuring enough free disk space available for normal operations and upgrading disk space as required. Monitoring jobs running on the database and ensuring that performance is not degraded too much.

1. **OVERALL STRUCTURE**

The functional component of the database system is divided into storage manager and query processor component.

**Storage Manager**

Storage manager is a program module that provides interface between the low level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The storage manager various DML statements into low level file system command. And it is responsible for storing, retrieving, and updating data in the database.

Storage manager consist following components:

Authorization and integrity manager: responsible to ensure integrity constraint does not violate and checks the authority of users to access data.

Transaction Manager: responsible to ensure database remain inconsistent state even system failure occurs. It is also responsible to manage concurrent transactions so that they could not conflict, which also helps to ensure consistency of database.

File Manager: responsible to manage the allocation of space on disk storage and the data structures used to represent information stored on disk.

Buffer Manager: responsible for fetching data from disk storage into main memory, and decides what data to cache in main memory. The storage manager implements several data structure for physical system implementation:

Data files: stores database itself,

Data dictionary: stores Meta data about structure of database, in particular schema of database.

Indices: provides fast access to data items that holds particular values.

**Query processor**

The query processor is responsible to simplify and facilitate access data. It is responsible to translate updates and queries written in nonprocedural language at the logical level, into an efficient sequence of operations at the physical level. The query processor component includes the following components: DDL interpreter: responsible to interprets DDL statements and records the definitions in the data dictionary. DML Compiler: responsible to translate DML statements in a query language into low level instructions that query evaluation engine understands. Query is generally translated into no. of alternative evaluation plans that produce the same result. It is also responsible for query optimization; it required to select the lowest cost evaluation plan among the alternatives Query evaluation: responsible to execute low level instruction generated by DML compiler.

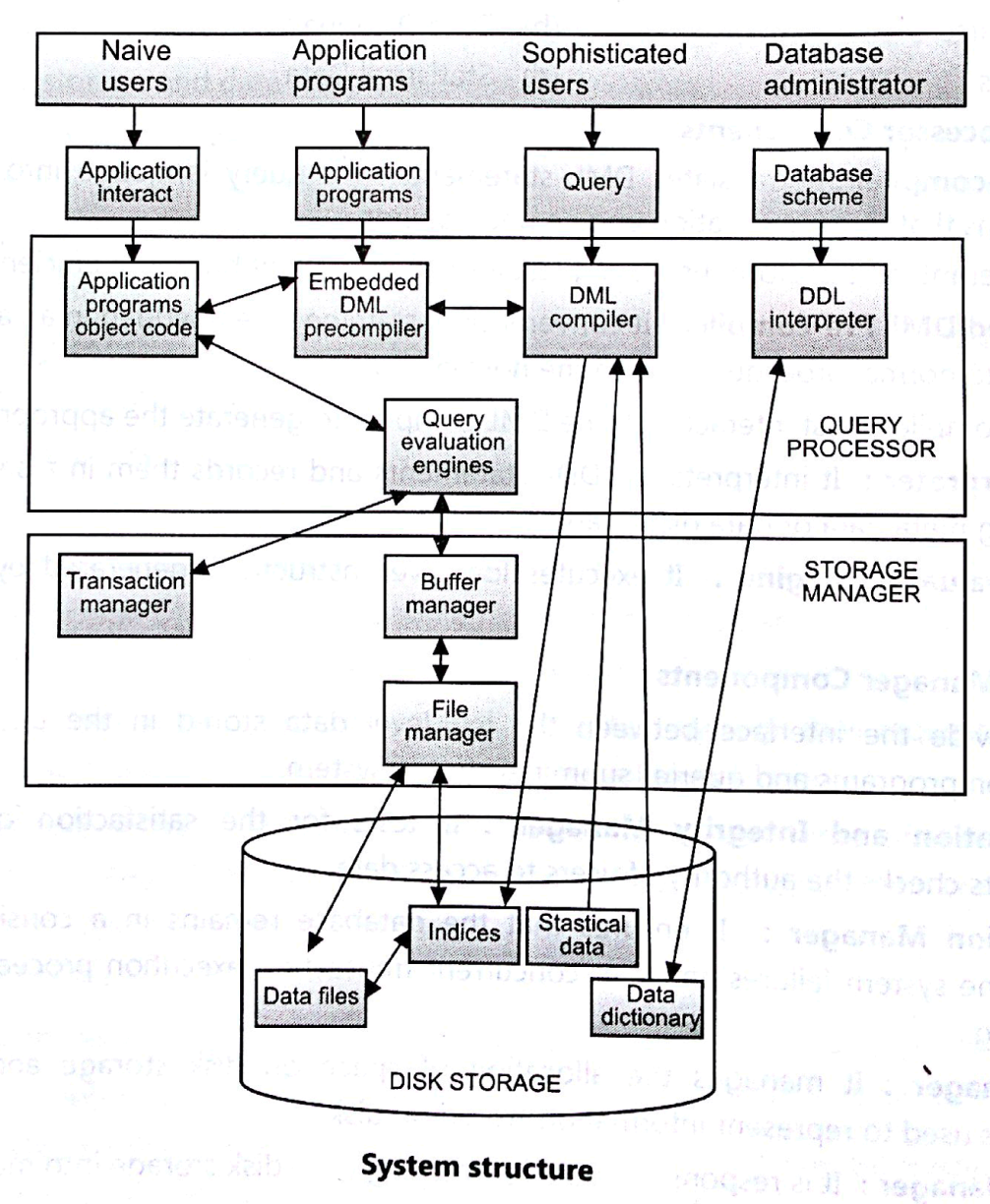
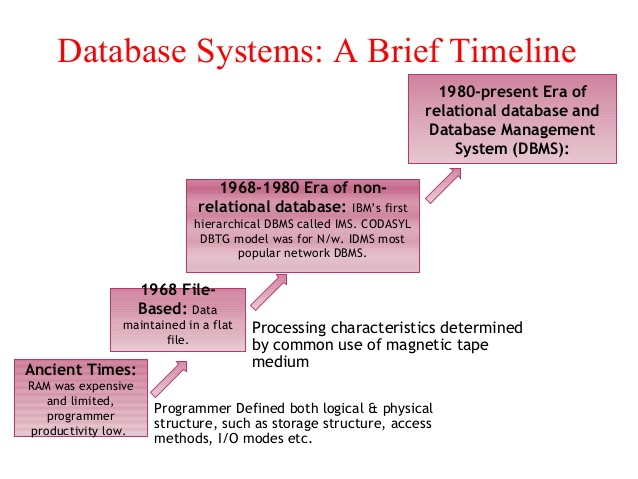


FIG: OVERALL STRUCTURE OF DBMS

1. **HISTORY OF DATABASE SYSTEM**



A DBMS (Database management system) is used to create and maintain the structure of a database, and then to enter, manipulate and retrieve the data it stores. Creating an efficient database design is the key to effectively using a database to support an organization's business operations. A database management system (DBMS) is a computer software that manages databases, it may use any of a variety of database models, such as the hierarchical DBMS, network DBMS and relational DBMS. In large systems, a DBMS allows users and other software to store and retrieve data in a structured way.

**History**

1960's-1970's: The emergence of the first type of DBMS, the hierarchical DBMS. IBM had the first model, developed on IBM 360 and their (DBMS) was called IMS, originally it was written for the Apollo program. This type of DBMS was based on binary trees, where the shape was like a tree and relations were only limited between parent and child records. The benefits were numerous; less redundant data, data independence, security and integrity, which all lead to efficient searches. Nonetheless; there were some disadvantages such as; complex implementation, was hard to manage because of the absence of standards, which made it harder to handle many relationships.

1970's-1980's: The emergence of the network DBMS. Charles Bachmann developed first DBMS at Honeywell, Integrated Data Store (IDS) then a group called CODASYL who is responsible for the creation of COBOL, had that system standardized. However; the CODASYL group invented what they call the "CODASYL APPROACH. Based on that approach many systems using network DBMS were developed for business use) in this model, each record can have multiple parents in comparison with one in the hierarchical DBMS. It is made of sets of relationships where a set represents a one to much relationship between the owner and the member. The main and unfortunate disadvantage was that the System was complex and there was difficulty in design and maintenance, it is believed that the Lack of structural independence was the main cause.

1980's- 1990's: The emergence of the relational DBMS on the hands of Edgar Codd. He worked at IBM, and he was unhappy with the navigational model of the CODASYL APPROACH. To him, a tool for searching, such as a search facility was very useful, and it was absent. In 1970, he proposed a new approach to database construction, which made the creation of a Relational DBMS intended for Large Shared Data Banks, possible and easy to grab. Moreover; this was a new system for entering data and working with big databases, where the idea was to use a table of records. All tables will be then linked by either one to one relationships, one too many, or many too many. When elements took space and were not useful, it was easy to remove them from the original table, and all the other "entries" in other tables linked to this record were removed. Worth mentioning, is that two initial projects were launched, the R program at IBM, and INGRES program at the University of California. In 1985, the object oriented DBMS was developed, but it did not have any booming commercial profit because of the high unjustified costs to change systems, and format. In 1990, the DBMS took on a new object oriented approach joint with relational DBMS. In this approach, text, multimedia, internet and web use in conjunction with DBMS were available and possible.

Past and present: In the early years of computing, a punch card was used in unit record machines for input, data storage and processing this data. Data was entered offline and for both data, and computer programs input. This input method is similar to voting machines nowadays . This was the only method, where it was fast to enter data, and retrieve it, but not to manipulate or edit it.

After that era, there was the introduction of the file type entries for data, then the DBMS as hierarchical, network, and relational.