ST. XAVIER'S COLLEGE

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database management system**

**LAB ASSIGNMENT # 2**

**SUBMITTED BY:**

Bishal Pandey  
013BSCCSIT016

4rd Semester

**SUBMITTED TO:**

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

**Date of Submission:** 27thjuly 2015

1. **Simplified database system**

A DBMS (database management system) is a collection of programs that enables users to create and maintain database. The DBMS is a common purpose software system that facilitates the process of constructing, defining, manipulating and sharing databases among various users as well as applications. Defining a database state the database involves specifying the constraints, data types and structures of the data to be stored in the database. The descriptive information is as well stored in the database in the form database catalogue or dictionary- it is called meta-data.

* Manipulating the data comprises the querying the database to retrieve the specific data.
* An application program accesses the database through transferring the quarries or requests for data to DBMS.
* The significant function provided by the DBMS includes protecting the database and maintain the database.

Fig: Simplified data base environment

1. **Approaches to management data**

**2.1 Database Approach**

With database systems, it is possible to hold the facts relating to parts of the organization on an integrated set of computer files: a database. The ‘organization’ in this context could be the whole business or, more likely, a part of it, such as a division or department. The various computer applications can use this as the data source. If the functions change, the data on the database will probably still be appropriate. If the facts change, then the database can be amended without redesigning the application systems. There is thus an element of data independence between the database and the applications that use it. The hardware and software can also be changed in order to reflect developments in the technology without requiring substantial changes in the application systems or the structure of the database

The database provides a major data resource for the organization. It is an important asset of most organizations – just imagine if a company lost its customer or accounts data. The motivations for companies choosing a database approach for their data handling are many and include:

* Increase data shareability
* Increase data integrity
* Increase the speed in implementing applications
* Ease data access by programmers and users
* Increase data independence ·
* Reduce program maintenance ·
* Provide a management view of the organization
* Improve the standards of the systems developers.

**File system approach**

Increase the speed in implementing applications · Ease data access by programmers and users · Increase data independence · Reduce program maintenance · Provide a management view of the organization · Improve the standards of the systems developers.

In a [computer](http://searchwindowsserver.techtarget.com/definition/computer), a file system (sometimes written *filesystem*) is the way in which [files](http://searchexchange.techtarget.com/definition/file) are named and where they are placed logically for storage and retrieval.

For example, DOS, Windows, OS/2,Macintosh and Unix-based operating systems (OSes) all have file systems in which files are placed somewhere in a hierarchical (tree) structure. A file is placed in a directory (*folder* in Windows) or subdirectory at the desired place in the tree structure.

File systems specify conventions for naming files, including the maximum number of characters in a name, which characters can be used and, in some systems, how long the file name suffix can be. A file system also includes a format for specifying the path to a file through the structure of directories.

1. **Database vs filesystem approach**

Technically, both of them support the basic features necessary for data access. For example, both of them:

* Ensure data is managed to ensure its integrity and quality
* Allow shared access by a community of users
* Use well-defined schema for data-access
* Support a query language

But file-systems seriously lack some of the critical features necessary for managing data. Lets take a look at some of these features.

**Transaction support**  
Atomic transactions guarantee complete failure or success of an operation. This is especially needed when there is concurrent access to the same data-set. This is one of the basic features provided by all databases.

But most file-systems don't have this feature. Only the lesser known file-systems, such as [Transactional NTFS(TxF)](http://en.wikipedia.org/wiki/Transactional_NTFS), [Sun ZFS](http://en.wikipedia.org/wiki/ZFS), and [Veritas VxFS](http://en.wikipedia.org/wiki/Veritas_File_System), support this feature. Most of the popular opensource file-systems (including ext3, xfs, reiserfs) are not even POSIX compliant.

**Fast Indexing**  
Databases allow indexing based on any attribute or data-property (i.e. SQL columns). This helps fast retrieval of data, based on the indexed attribute. This functionality is not offered by most file-systems, i.e. you can't quickly access "all files created after 2PM today."

The desktop search tools like Google desktop or MAC spotlight offer this functionality. But for this, they have to scan and index the complete file-system and store the information in a internal relational-database.

**Snapshots**  
Snapshot is a point-in-time copy/view of the data. Snapshots are needed for backup applications, which need consistent point-in-time copies of data.

The transactional and journaling capabilities enable most of the databases to offer snapshots without shopping access to the data. Most file-systems however, don't provide this feature (ZFS and VxFS being only exceptions). The backup softwares have to either depend on running application or underlying storage for snapshots.

**Clustering**  
Advanced databases like Oracle (and now MySQL) also offer clustering capabilities. The "**g**" in "Oracle 11**g**" actually stands for "*grid*" or clustering capability. MySQL offers shared-nothing clusters using synchronous replication. This helps the databases scale up and support larger & more-fault tolerant production environments.

File systems still don't support this option. :(  The only exceptions are Veritas CFS and GFS (Open Source).

**Replication**  
Replication is commodity with databases and form the basis for disaster-recovery plans. File-systems still have to evolve to handle it.

**Relational View of Data**  
File systems store files and other objects only as a stream of bytes, and have little or no information about the data stored in the files. Such file systems also provide only a single way of organizing the files, namely via directories and file names. The associated attributes are also limited in number, e.g. type, size, author, creation time, etc. This does not help in managing related data, as disparate items do not have any relationships defined.

* 1. **Data abstraction**

Data abstraction is the reduction of a particular body of data to a simplified representation of the whole.

[Abstraction](http://whatis.techtarget.com/definition/abstraction), in general, is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics. As in abstract art, the representation is likely to be one potential abstraction of a number of possibilities. A database abstraction layer, for example, is one of a number of such possibilities.

Data abstraction is usually the first step in database design. A complete [database](http://searchsqlserver.techtarget.com/definition/database) is much too complex a system to be developed without first creating a simplified framework. Data abstraction makes it possible for the developer to start from essential elements -- data abstractions -- and incrementally add data detail to create the final system.

* 1. **Reliability**

Main aspects of reliability: -

correctness: the system works in accordance with the corresponding specifications Some kind of software testing methods ( ) can be used. There is no appropriate tool to proof the correctness of a complex software system.

- availability: fraction of the time that the system meets its specification

Usually the requirements of the correctness and of the availability cannot be met at the same time. Increasing correctness decreasing availability Increasing availability Þ decreasing correctness Correctness oriented applications (banking, simulations) Availability oriented applications ( tourist information system) Trade-off problem : which of the two aspects should get higher priority? 2-Phase Commit example: A distributed transaction is performed. Some of the participating nodes sent a 'ready to commit' message to the coordinator. The network connection is getting now down to a site which did not give an answer yet. What should the coordinator do? - waiting for answer : high correctness, low availability - take a decision : high availability, low correctness .

* 1. **efficiency /performance**

In this section we discuss some of the more "general" methods that can be used for database performance evaluation. The word "general" is binding to systems, meaning that the approaches mentioned here are generally true for "systems" with a special focus on database systems.

According to , performance analysis of database systems serve two basic purposes:

1. For the evaluation of the best configuration and operating environment of a single database system, and

2. Studying two or more database systems and providing a systematic comparision of the systems.

1. **Three layer architecture**

Three-tier architecture allows any one of the three tiers to be upgraded or replaced independently. The user interface is implemented on a desktop PC and uses a standard graphical user interface with different modules running on the application server. The relational database management system on the database server contains the computer data storage logic. The middle tiers are usually multitiered.

The three tiers in a three-tier architecture are:

1. Presentation Tier: Occupies the top level and displays information related to services available on a website. This tier communicates with other tiers by sending results to the browser and other tiers in the network.
2. Application Tier: Also called the middle tier, logic tier, business logic or logic tier, this tier is pulled from the presentation tier. It controls application functionality by performing detailed processing.
3. Data Tier: Houses database servers where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic.

1. **Advantage and disadvantage of DBMS**

**Advantage of DBMS :**  
The DBMS serves as the intermediary between the user and the database. The database structure itself is stored as a collection of files, and the only way to access the data in those files is through the DBMS. The DBMS receives all application requests and translates them into the complex operations required to fulfill those requests. The DBMS hides much of the database’s internal complexity from the application programs and users.  
**The different advantages of DBMS are as follows.**  
1. **Improved data sharing.**  
The DBMS helps create an environment in which end users have better access to more and better-managed data. Such access makes it possible for end users to respond quickly to changes in their environment.  
**2. Improved data security.**  
The more users access the data, the greater the risks of data security breaches. Corporations invest considerable amounts of time, effort, and money to ensure that corporate data are used properly. A DBMS provides a framework for better enforcement of data privacy and security policies.  
**3. Better data integration.**  
 Wider access to well-managed data promotes an integrated view of the organization’s operations and a clearer view of the big picture. It becomes much easier to see how actions in one segment of the company affect other segments.  
**4. Minimized data inconsistency.**  
 Data inconsistency exists when different versions of the same data appear in different places. For example, data inconsistency exists when a company’s sales department stores a sales representative’s name as “Bill Brown” and the company’s personnel department stores that same person’s name as “William G. Brown,” or when the company’s regional sales office shows the price of a product as $45.95 and its national sales office shows the same product’s price as $43.95. The probability of data inconsistency is greatly reduced in a properly designed database.

**5. Improved data access.**  
The DBMS makes it possible to produce quick answers to ad hoc queries. From a database perspective, a query is a specific request issued to the DBMS for data manipulation—for example, to read or update the data. Simply put, a query is a question, and an ad hoc query is a spur-of-the-moment question. The DBMS sends back an answer (called the query result set) to the application. For example, end users, when dealing with large amounts of sales data, might want quick answers to questions (ad hoc queries) such as:  
- What was the dollar volume of sales by product during the past six months?  
- What is the sales bonus figure for each of our salespeople during the past three months?  
- How many of our customers have credit balances of $3,000 or more?

**6.Improved decision making.**  
Better-managed data and improved data access make it possible to generate better-quality information, on which better decisions are based. The quality of the information generated depends on the quality of the underlying data. Data quality is a comprehensive approach to promoting the accuracy, validity, and timeliness of the data. While the DBMS does not guarantee data quality, it provides a framework to facilitate data quality initiatives.

**7.Increased end-user productivity.**  
The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.  
  
**Disadvantages of Database:**  
  
Although the database system yields considerable advantages over previous data management approaches, database systems do carry significant disadvantages. For example:  
**1. Increased costs.**  
Database systems require sophisticated hardware and software and highly skilled personnel. The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial. Training, licensing, and regulation compliance costs are often overlooked when database systems are implemented.  
**2. Management complexity.**  
Database systems interface with many different technologies and have a significant impact on a company’s resources and culture. The changes introduced by the adoption of a database system must be properly managed to ensure that they help advance the company’s objectives. Given the fact that database systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.  
**3. Maintaining currency.**  
To maximize the efficiency of the database system, you must keep your system current. Therefore, you must perform frequent updates and apply the latest patches and security measures to all components. Because database technology advances rapidly, personnel training costs tend to be significant. Vendor dependence. Given the heavy investment in technology and personnel training, companies might be reluctant to change database vendors. As a consequence, vendors are less likely to offer pricing point advantages to existing customers, and those customers might be limited in their choice of database system components.  
**4. Frequent upgrade/replacement cycles.**  
DBMS vendors frequently upgrade their products by adding new functionality. Such new features often come bundled in new upgrade versions of the software. Some of these versions require hardware upgrades. Not only do the upgrades themselves cost money, but it also costs money to train database users and administrators to properly use and manage the new features.

**6. Drawback of using file system approach:**

1. Data Security

The data stored in the flat file(s) can be easily accessible and hence it is not secure.

Example: Consider an online banking application where we store the account related information of all customers in flat files. A customer will have access only to his account related details. However from a flat file, it is difficult to put such constraints. It is a big security issue.

1. Data Redundancy

In this storage model, the same information may get duplicated in two or more files. This may lead to to higher storage and access cost. it also may lead to data inconsistency.

For Example, assume the same data is repeated in two or more files. If a change is made to data stored in one file, other files also needs to be change accordingly.

Example: Assume employee details such as firstname, lastname, emailid are stored in employee\_details file and employee\_salary file. If a change needs to be made to emailid, both employee\_details file and emplyee\_salary file need to be updated otherwise it will lead to inconsistent data.

However, it is possible to design file systems with minimal redundancy. Also note that Data redundancy is sometimes preferred.

Example: Assume employee details such as firstname, lastname, emailid are stored only in employee\_details file and not in employee\_salary file. If we need to access an employee salary along with firstname of the employee, we have to retrieve details from two files. This would mean an increased overhead.

1. Data Isolation

Data Isolation means that all the related data is not available in one file. Usually the data is scattered in various files having different formats. Hence writing new application programs to retrieve the appropriate data is difficult.

1. Program/Data Dependence

In traditional file approach, application programs are closely dependent on the files in which data is stored. If we make any changes in the physical format of the file(s), like addition of a data field , etc, all application programs needs to be changed accordingly. Consequently, for each of the application programs that a programmer writes or maintains, the programmer must be concerned with data management. There is no centralized execution of the data management functions. Data management is scattered among all the application programs.

Example: Consider the banking system. An employee\_salary file exists which has details about the salary of employees. An employee\_salary record is described by

employee\_id

firstname

lastname

salary\_amount

An application program is available to display all the details about the salary of all employees. Assume a new data field, the date\_of\_joining is added to the employee\_salary file. Since the application program depends on the file, it also needs to be altered.

If the physical format of the employee\_salary file for example the field delimiter, record delimiter, etc. are changed, it necessitates that the application program which depends on it, also be altered.

1. Lack of Flexibility

The traditional systems are able to retrieve information for predetermined requests for data. If we need unanticipated data, huge programming effort is needed to make the information available, provided the information is there in the files. By the time the information is made available, it may no longer be required or useful.

Example : Consider a software application which is able to generate employee salary report. Assume that all the data is stored in flat files. Suppose we now have a requirement to retrieve all the employee details whose salary is greater than Rs.10000. It is not easy to generate such on-demand reports and lot of time is needed for application developers to modify the application to meet such requirements.

1. Concurrent Access Anomalies

Many traditional systems allow multiple users to access and update the same piece of data simultaneously. However this concurrent updates may result in inconsistent data. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult because data may be accessed by many different application programs and these application programs may not have been coordinated previously.

Example : Consider a personal information system which has the data of all employees. Now there may be an employee updating his address details in the system and at the same time, an administrator may be taking a report containing the data of all employees. This is called concurrent access. Since the employee's address is being updated at the same time, there is a possibility of the administrator reading an incorrect address.

These difficulties lead to the development of database systems.