**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**DATABASE MANAGEMENT SYSTEM**

**LAB ASSIGNMENT#1**

**Submitted by:**

Shreesha Pokharel

013BSCCSIT038

**Submitted to:**

|  |  |
| --- | --- |
| **Er. Sanjay Kumar Yadav** |  |

Lecturer

Department of Computer Science

Date of submission: 24th July, 2015

**PURPOSE OF DATABASE SYSTEM:**

A **database management system** (DBMS) is a software tool that makes it possible to organize data in a database. The ultimate purpose of a database management system is to store and transform data into information to support making decisions.

DBMS makes it possible for users to create, edit and update data in database files. Once created, the DBMS makes it possible to store and retrieve data from those database files.

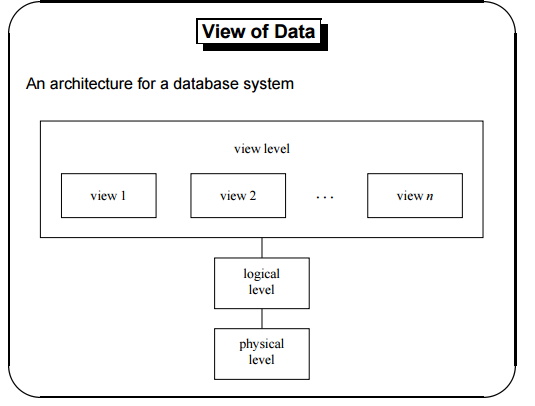
More specifically, a DBMS provides the following functions:

* **Concurrency:** concurrent access (meaning 'at the same time') to the same database by multiple users
* **Security:** security rules to determine access rights of users
* **Backup and recovery**: processes to back-up the data regularly and recover data if a problem occurs
* **Integrity:** database structure and rules improve the integrity of the data
* **Data descriptions:** a data dictionary provides a description of the data

Within an organization, the development of the database is typically controlled by **database administrators (DBAs)** and other specialists. This ensures the database structure is efficient and reliable.

Database administrators also control access and security aspects. For example, different people within an organization use databases in different ways. Some employees may simply want to view the data and perform basic analysis. Other employees are actively involved in adding data to the database or updating existing data. This means that the database administrator needs to set the user permissions.

**VIEW OF DATA:**

****

**DATABASE LANGUAGES:**

Database language is a generic term referring to a class of languages used for defining and accessing databases. A particular database language will be associated with a particular database management system. There are two distinct classes of database language: those that do not provide complete programming facilities and are designed to be used in association with some general-purpose programming language (the host language), and those that do provide complete programming facilities (database programming languages). Some products adopting the former approach seek to minimize host-language programming by the provision of fourth-generation language (4GL) facilities.   
  
A database language must provide for both logical-schema specification and modification (data description) and for retrieval and update (data manipulation). In some cases, particularly products derived from the CODASYL network database standard, these aspects are treated distinctly as the data description language (DDL) and the data manipulation language (DML). Modification to the storage schema is also generally separately provided.

**Data description language (DDL)**

* Specification notation for defining the database schema
* DDL compiler generates a set of tables stored in a data dictionary
* Data dictionary contains metadata (i.e., data about data)
* Data storage and definition language – special type of DDL in which the storage structure and access methods used by the database system are specified

**Data manipulation language (DML)**

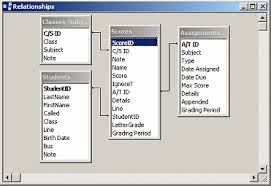
* Language for accessing and manipulating the data organized by the appropriate data model
* Two classes of languages
  + Procedural – user specifies what data is required and how to get those data
  + Nonprocedural – user specifies what data is required without specifying how to get those data

**RELATIONAL DATABASES:**

A **relational database** is a digital database whose organization is based on the relational model of data, as proposed by E.F. Codd in 1970.[[1]](https://en.wikipedia.org/wiki/Relational_database#cite_note-1) This model organizes data into one or more tables (or "relations") of rows and columns, with a unique key for each row. Generally, each entity type described in a database has its own table, the rows representing instances of that type of entity and the columns representing values attributed to that instance. Because each row in a table has its own unique key, rows in a table can be linked to rows in other tables by storing the unique key of the row to which it should be linked (where such unique key is known as a "foreign key"). Codd showed that data relationships of arbitrary complexity can be represented using this simple set of concepts.

Prior to the advent of this model, databases were usually hierarchical, and each tended to be organized with a unique mix of indexes, chains, and pointers. The simplicity of the relational model led to it soon becoming the predominant type of database.

The various software systems used to maintain relational databases are known as Relational Database Management Systems (RDBMS).



**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

**OBJECT-BASED AND SEMI STRUCTURED DATABASE:**

**Object-based database:** An **object database** (also **object-oriented database management system**) is a database management system in which information is represented in the form of objects as used in object-oriented programming. Object databases are different from relational databases which are table-oriented. Object-relational databases are a hybrid of both approaches.

**Semi structured database:** The **semi**-**structured** model is a **database** model where there is no separation between the **data** and the schema, and the amount of **structure** used depends on the purpose. The advantages of this model are the following: It can represent the information of some **data** sources that cannot be constrained by schema.

**DATA STORAGE AND QUERYING:**

**Data storage:** Database storage is the container of the physical materialization of a database. It comprises the *internal* (physical) *level* in the database architecture. It also contains all the information needed (e.g., metadata, "data about the data", and internal data structures) to reconstruct the *conceptual level* and *external level* from the internal level when needed. Putting data into permanent storage is generally the responsibility of the database engine a.k.a. "storage engine". Though typically accessed by a DBMS through the underlying operating system (and often utilizing the operating systems' file systems as intermediates for storage layout), storage properties and configuration setting are extremely important for the efficient operation of the DBMS, and thus are closely maintained by database administrators. A DBMS, while in operation, always has its database residing in several types of storage (e.g., memory and external storage). The database data and the additional needed information, possibly in very large amounts, are coded into bits. Data typically reside in the storage in structures that look completely different from the way the data look in the conceptual and external levels, but in ways that attempt to optimize (the best possible) these levels' reconstruction when needed by users and programs, as well as for computing additional types of needed information from the data (e.g., when querying the database).

Some DBMSs support specifying which character encoding was used to store data, so multiple encodings can be used in the same database.

Various low-level database storage structures are used by the storage engine to serialize the data model so it can be written to the medium of choice. Techniques such as indexing may be used to improve performance. Conventional storage is row-oriented, but there are also column-oriented and correlation databases.

**Querying:**  A query is a request for information from a database. There are three general methods for posing queries:

* **Choosing parameters from a menu**: In this method, the database system presents a list of parameters from which you can choose. This is perhaps the easiest way to pose a query because the menus guide you, but it is also the least flexible.
* **Query by example (QBE):** In this method, the system presents a blank record and lets you specify the fields and values that define the query.
* **Query language:** Many database systems require you to make requests for information in the form of a stylized query that must be written in a special query language. This is the most complex method because it forces you to learn a specialized language, but it is also the most powerful.

**TRANSACTION MANAGEMENT:**

A **transaction** symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database. Transactions in a database environment have two main purposes:

1. To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status.
2. To provide isolation between programs accessing a database concurrently. If this isolation is not provided, the program's outcome is possibly erroneous.

A database transaction, by definition, must be atomic, consistent, isolated and durable. Database practitioners often refer to these properties of database transactions using the acronym ACID.

Transactions provide an "all-or-nothing" proposition, stating that each work-unit performed in a database must either complete in its entirety or have no effect whatsoever. Further, the system must isolate each transaction from other transactions, results must conform to existing constraints in the database, and transactions that complete successfully must get written to durable storage.

**DATABASE ARCHITECTURE:**

Database architecture focuses on the design, development, implementation and maintenance of computer programs that store and organize information for businesses, agencies and institutions. A database architect develops and implements software to meet the needs of users. Several types of databases, including relational or multimedia, may be created. Additionally, database architects may use one of several languages to create databases, such as structured query language (SQL).

**DATABASE USERS AND ADMINISTRATORS:**

**Database administrators:**

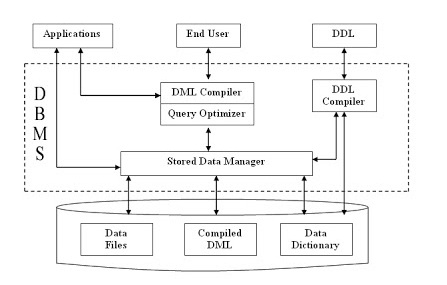
* Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise’s information resources and needs.
* Database administrator’s duties include:
  + Schema definition
  + Storage structure and access method definition
  + Schema and physical organization modification
  + Granting user authority to access the database
  + Specifying integrity constraints
  + Acting as liaison with users
  + Monitoring performance and responding to changes in requirements

**Database users:**

Users are differentiated by the way they expect to interact with the system

* Application programmers – interact with system through DML calls
* Sophisticated users – form requests in a database query language
* Specialized users – write specialized database applications that do not fit into the traditional data processing framework
* Naive users – invoke one of the permanent application programs that have been written previously

**OVERALL STRUCTURE:**

****

**HISTORY OF DATABASE STRUCTURE:**

Following the technology progress in the areas of processors, computer memory, computer storage and computer networks, the sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. The development of database technology can be divided into three eras based on data model or structure: navigational, SQL/relational, and post-relational.

The two main early navigational data models were the hierarchical model, epitomized by IBM's IMS system and the CODASYL model (network model), implemented in a number of products such as IDMS.

The relational model, first proposed in 1970 by Edgar F. Codd, departed from this tradition by insisting that applications should search for data by content, rather than by following links. The relational model employs sets of ledger-style tables, each used for a different type of entity. Only in the mid-1980s did computing hardware become powerful enough to allow the wide deployment of relational systems (DBMSs plus applications). By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2015 they remain dominant: Oracle, mySQL and SQL server are the top DBMS. The dominant database language, standardized SQL for the relational model, has influenced database languages for other data models.

Object databases were developed in the 1980s to overcome the inconvenience of object-relational impedance mismatch, which led to the coining of the term "post-relational" and also the development of hybrid object-relational databases.

The next generation of post-relational databases in the late 2000s became known as NoSQL databases, introducing fast key-value stores and document-oriented databases. A competing "next generation" known as NewSQL databases attempted new implementations that retained the relational/SQL model while aiming to match the high performance of NoSQL compared to commercially available relational DBMSs.