An Introduction to Databases and Database Management Systems.

Introduction

An important aspect of most every business is record keeping. In our information society, this has become an important aspect of business, and much of the world's computing power is dedicated to maintaining and using databases.
Databases of all kinds pervade almost every business. All kinds of data, from emails and contact information to financial data and records of sales, are stored in some form of a database. The quest is on for meaningful storage of less-structured information, such as subject knowledge.

What is a Database?

A brief definition might be:

- A STORE OF INFORMATION,
- HELD OVER A PERIOD OF TIME,
- IN COMPUTER-READABLE FORM.

DBMS – A Database is a collection of interrelated data and a Database Management System is a set of programs to use and/or modify this data.

DBMS functions

Data definition

This includes describing:

- FILES
- RECORD STRUCTURES
- FIELD NAMES, TYPES and SIZES
- RELATIONSHIPS between records of different types
- Extra information to make searching efficient, e.g. INDEXES.

Data entry and validation

Validation may include:

- TYPE CHECKING
- RANGE CHECKING
- CONSISTENCY CHECKING

In an interactive data entry system, errors should be detected immediately - some can be prevented altogether by keyboard monitoring - and recovery and re-entry permitted.

Updating

Updating involves:

- Record INSERTION
- Record MODIFICATION
- Record DELETION.

At the same time any back-ground data such as indexes or pointers from one record to another must be changed to maintain consistency. Updating may take place interactively, or by submission of a file of transaction records; handling these may require a program of some kind to be written, either in a conventional programming language (a host language, e.g. COBOL or C) or in a language supplied by the DBMS for constructing command files.

Data retrieval on the basis of selection criteria

For this purpose most systems provide a QUERY LANGUAGE with which the characteristics of the required records may be specified. Query languages differ enormously in power and sophistication but a standard which is becoming increasingly common is based on the so-called RELATIONAL operations. These allow:

- selection of records on the basis of particular field values.
- selection of particular fields from records to be displayed.
- linking together records from two different files on the basis of matching field values.

Arbitrary combinations of these operators on the files making up a database can answer a very large number of queries without requiring users to go into one record at a time processing.

Report definition

Most systems provide facilities for describing how summary reports from the database are to be created and laid out on paper. These may include obtaining:

- COUNTS
- TOTALS
- AVERAGES
- MAXIMUM and MINIMUM values

over particular CONTROL FIELDS. Also specification of PAGE and LINE LAYOUT, HEADINGS, PAGE-NUMBERING, and other narrative to make the report comprehensible.

Security.

This has several aspects:

- Ensuring that only those authorized to do so can see and modify the data, generally by some extension of the password principle.
- Ensuring the consistency of the database where many users are accessing and up-dating it simultaneously.

- Ensuring the existence and INTEGRITY of the database after hardware or software failure. At the very least this involves making provision for back-up and re-loading.

Approaches to Data Management

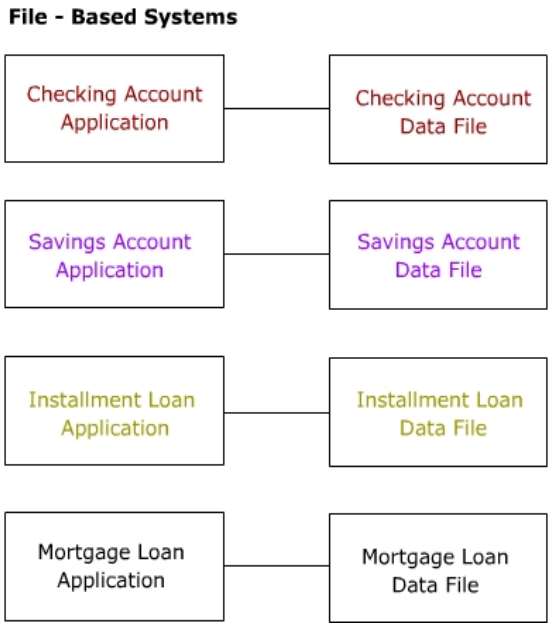
- **File-Based Systems**

  Conventionally, before the Database systems evolved, data in software systems was stored in and represented using flat files.

- **Database Systems**

  Database Systems evolved in the late 1960s to address common issues in applications handling large volumes of data which are also data intensive. Some of these issues could be traced back to the following disadvantages of File-based systems.

Drawbacks of File-Based Systems



As shown in the figure, in a file-based system, different programs in the same application may be interacting with different private data files. There is no system enforcing any standardized control on the organization and structure of these data files.

- **Data Redundancy and Inconsistency**

  Since data resides in different private data files, there are chances of redundancy and resulting inconsistency. For example, in the above example shown, the same customer can have a savings account as well as a mortgage loan. Here the customer details may be duplicated since the programs for the two functions store their corresponding data in two different data files. This gives rise to redundancy in the customer's data. Since the same data is stored in two files, inconsistency arises if a change made in the data in one file is not reflected in the other.

- **Unanticipated Queries**

  In a file-based system, handling sudden/ad-hoc queries can be difficult, since it requires changes in the existing programs.

- **Data Isolation**

  Though data used by different programs in the application may be related, they reside in isolated data files.

- **Concurrent Access Anomali**es

  In large multi-user systems the same file or record may need to be accessed by multiple users simultaneously. Handling this in a file-based systems is difficult.

- **Security Problems**

  In data-intensive applications, security of data is a major concern. Users should be given access only to required data and not the whole database. In a file-based system, this can be handled only by additional programming in each application.

- **Integrity Problems**

  In any application, there will be certain data integrity rules which needs to be maintained. These could be in the form of certain conditions/constraints on the elements of the data records. In the savings bank application, one such integrity rule could be "Customer ID, which is the unique identifier for a customer record, should be non-empty". There can be several such integrity rules. In a file-based system, all these rules need to be explicitly programmed in the application program.

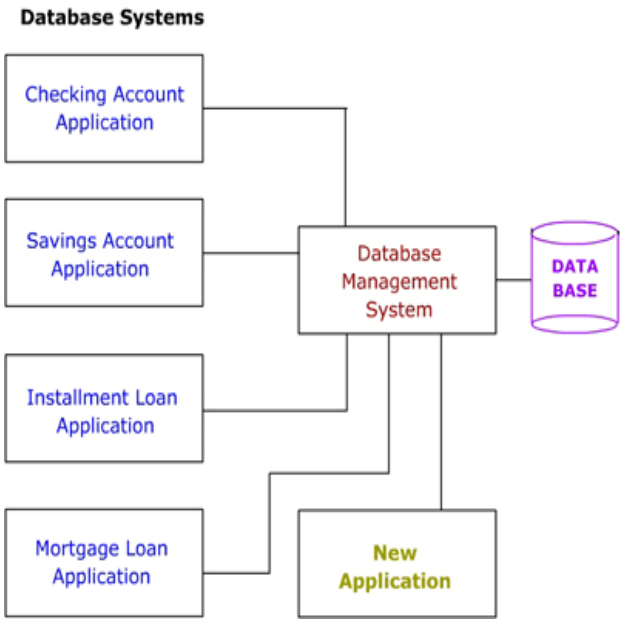
It may be noted that, we are not trying to say that handling the above issues like concurrent access, security, integrity problems, etc., is not possible in a file-based system. The real issue was that, though all these are common issues of concern to any data-intensive application, each application had to handle all these problems on its own. The application programmer needs to bother not only about implementing the application business rules but also about handling these common issues.

Advantages of Database Systems

Why Use a DBMS?

- **Data independence and efficient access.**
- **Reduced application development time.**
- **Data integrity and security.**
- **Uniform data administration.**
- **Concurrent access, recovery from crashes.**

As shown in the figure, the DBMS is a central system which provides a common interface between the data and the various front-end programs in the application. It also provides a central location for the whole data in the application to reside.



Due to its centralized nature, the database system can overcome the disadvantages of the file-based system as discussed below.

- **Minimal Data Redundancy**
  Since the whole data resides in one central database, the various programs in the application can access data in different data files. Hence data present in one file need not be duplicated in another. This reduces data redundancy. However, this does not mean all redundancy can be eliminated. There could be business or technical reasons for having some amount of redundancy. Any such redundancy should be carefully controlled and the DBMS should be aware of it.

- **Data Consistency**
  Reduced data redundancy leads to better data consistency.

- **Data Integration**
  Since related data is stored in one single database, enforcing data integrity is much easier. Moreover, the functions in the DBMS can be used to enforce the integrity rules with minimum programming in the application programs.

- **Data Sharing**
  Related data can be shared across programs since the data is stored in a centralized manner. Even new applications can be developed to operate against the same data.

- **Enforcement of Standards**
  Enforcing standards in the organization and structure of data files is required and also easy in a Database System, since it is one single set of programs which is always interacting with the data files.

- **Application Development Ease**
  The application programmer need not build the functions for handling issues like concurrent access, security, data integrity, etc. The programmer only needs to implement the application business rules. This brings in application development

ease. Adding additional functional modules is also easier than in file-based systems.

- **Better Controls**
  Better controls can be achieved due to the centralized nature of the system.

- **Data Independence**
  The architecture of the DBMS can be viewed as a 3-level system comprising the following:
  - The internal or the physical level where the data resides.
  - The conceptual level which is the level of the DBMS functions
  - The external level which is the level of the application programs or the end user.
  Data Independence is isolating an upper level from the changes in the organization or structure of a lower level. For example, if changes in the file organization of a data file do not demand for changes in the functions in the DBMS or in the application programs, data independence is achieved. Thus Data Independence can be defined as immunity of applications to change in physical representation and access technique. The provision of data independence is a major objective for database systems.

- **Reduced Maintenance**
  Maintenance is less and easy, again, due to the centralized nature of the system.

Functions of a DBMS

The functions performed by a typical DBMS are the following:

- **Data Definition**

  The DBMS provides functions to define the structure of the data in the application. These include defining and modifying the record structure, the type and size of fields and the various constraints/conditions to be satisfied by the data in each field.

- **Data Manipulation**

  Once the data structure is defined, data needs to be inserted, modified or deleted. The functions which perform these operations are also part of the DBMS. These function can handle planned and unplanned data manipulation needs. Planned queries are those which form part of the application. Unplanned queries are ad-hoc queries which are performed on a need basis.

- **Data Security & Integrity**

  The DBMS contains functions which handle the security and integrity of data in the application. These can be easily invoked by the application and hence the application programmer need not code these functions in his/her programs.

- **Data Recovery & Concurrency**

  Recovery of data after a system failure and concurrent access of records by multiple users are also handled by the DBMS.

- **Data Dictionary Maintenance**

  Maintaining the Data Dictionary which contains the data definition of the application is also one of the functions of a DBMS.

- **Performance**

  Optimizing the performance of the queries is one of the important functions of a DBMS. Hence the DBMS has a set of programs forming the Query Optimizer which evaluates the different implementations of a query and chooses the best among them.

Thus the DBMS provides an environment that is both convenient and efficient to use when there is a large volume of data and many transactions to be processed.

Role of the Database Administrator

Typically there are three types of users for a DBMS. They are :

1. The End User who uses the application. Ultimately, this is the user who actually puts the data in the system into use in business. This user need not know anything about the organization of data in the physical level. She also need not be aware of the complete data in the system. She needs to have access and knowledge of only the data she is using.
2. The Application Programmer who develops the application programs. She has more knowledge about the data and its structure since she has manipulate the data using her programs. She also need not have access and knowledge of the complete data in the system.
3. The Database Administrator (DBA) who is like the super-user of the system. The role of the DBA is very important and is defined by the following functions.

- **Defining the Schema**

  The DBA defines the schema which contains the structure of the data in the application. The DBA determines what data needs to be present in the system ad how this data has to be represented and organized.

- **Liaising with Users**

  The DBA needs to interact continuously with the users to understand the data in the system and its use.

- **Defining Security & Integrity Checks**

  The DBA finds about the access restrictions to be defined and defines security checks accordingly. Data Integrity checks are also defined by the DBA.

- **Defining Backup / Recovery Procedures**

  The DBA also defines procedures for backup and recovery. Defining backup procedures includes specifying what data is to backed up, the periodicity of taking backups and also the medium and storage place for the backup data.

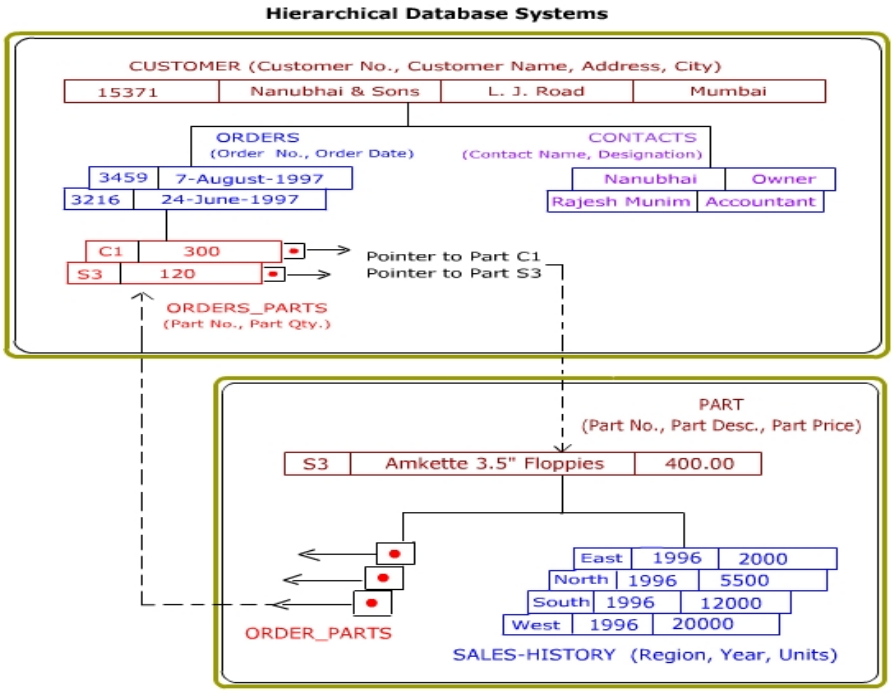
- **Monitoring Performance**

  The DBA has to continuously monitor the performance of the queries and take measures to optimize all the queries in the application.

Types of Database Systems

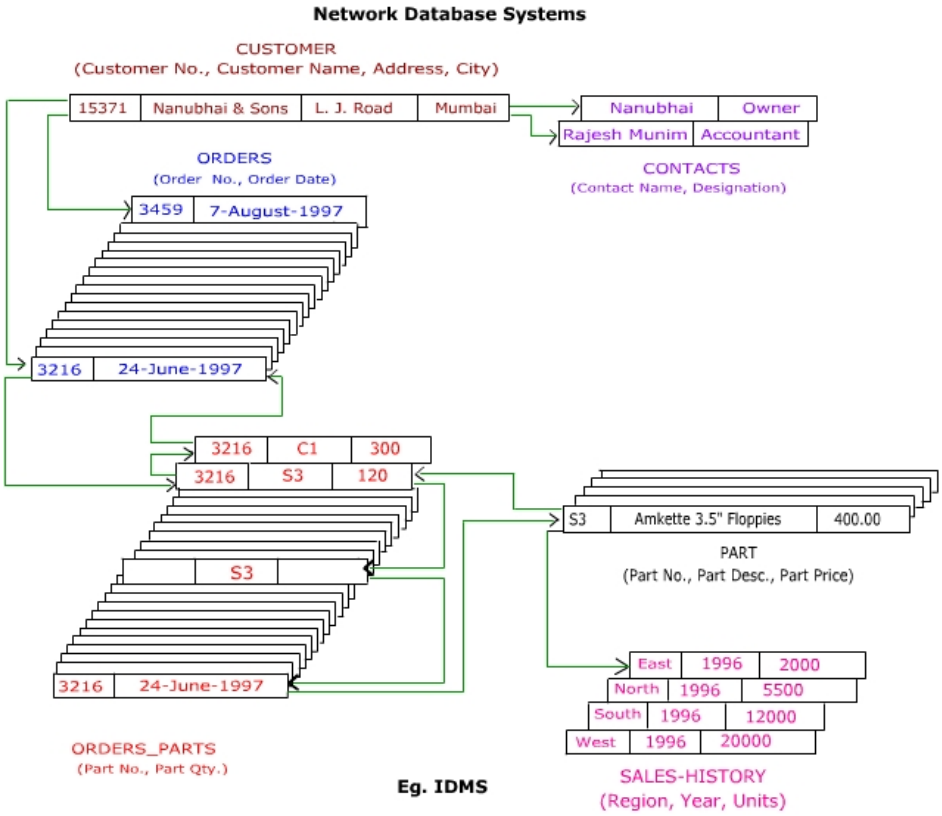
Database Systems can be catagorised according to the data structures and operators they present to the user. The oldest systems fall into inverted list, hierarchic and network systems. These are the pre-relational models.

- In the **Hierarchical Model**, different records are inter-related through hierarchical or tree-like structures. A parent record can have several children, but a child can have only one parent. In the figure, there are two hierarchies shown - the first storing the relations between CUSTOMER, ORDERS, CONTACTS and ORDER_PARTS and the second showing the relation between PARTS, ORDER_PARTS and SALES_HISTORY. The many-to-many relationship is implemented through the ORDER_PARTS segment which occurs in both the hierarchies. In practice, only one tree stores the ORDER_PARTS segment, while the other has a logical pointer to this segment. IMS (Information Management System) of IBM is an example of a Hierarchical DBMS.



Eg. Information Management System (IMS)

- In the **Network Model**, a parent can have several children and a child can also have many parent records. Records are physically linked through linked-lists. IDMS from Computer Associates International Inc. is an example of a Network DBMS.



Eg. IDMS

- In the **Relational Model**, unlike the Hierarchical and Network models, there are no physical links. All data is maintained in the form of tables consisting of rows and columns. Data in two tables is related through common columns and not physical links or pointers. Operators are provided for operating on rows in tables. Unlike the other two type of DBMS, there is no need to traverse pointers in the Relational DBMS. This makes querying much more easier in a Relational DBMS than in the the Hierarchical or Network DBMS. This, in fact, is a major reason for the relational model to become more programmer friendly and much more dominant and popular in both industrial and academic scenarios. Oracle, Sybase, DB2, Ingres, Informix, MS-SQL Server are few of the popular Relational DBMSs.

  CUSTOMER

CUST. NO.	CUSTOMER NAME	ADDRESS	CITY
15371	Nanubhai & Sons	L. J. Road	Mumbai
...	...	...	...
...	...	...	...
...	...	...	...

-

CONTACTS

CUST.NO.	CONTACT	DESIGNATION
15371	Nanubhai	Owner
15371	Rajesh Munim	Accountant
...	...	...
...	...	...

ORDERS

ORDER NO.	ORDER DATE	CUSTOMER NO.
3216	24-June-1997	15371
...	...	...
...	...	...
...	...	...

PARTS

PARTS NO.	PARTS DESC	PART PRICE
S3	Amkette 3.5" Floppies	400.00
...	...	...
...	...	...
...	...	...

ORDERS-PARTS

ORDER NO.	PART NO.	QUANTITY
3216	C1	300
3216	S3	120
...	...	...
...	...	...

-

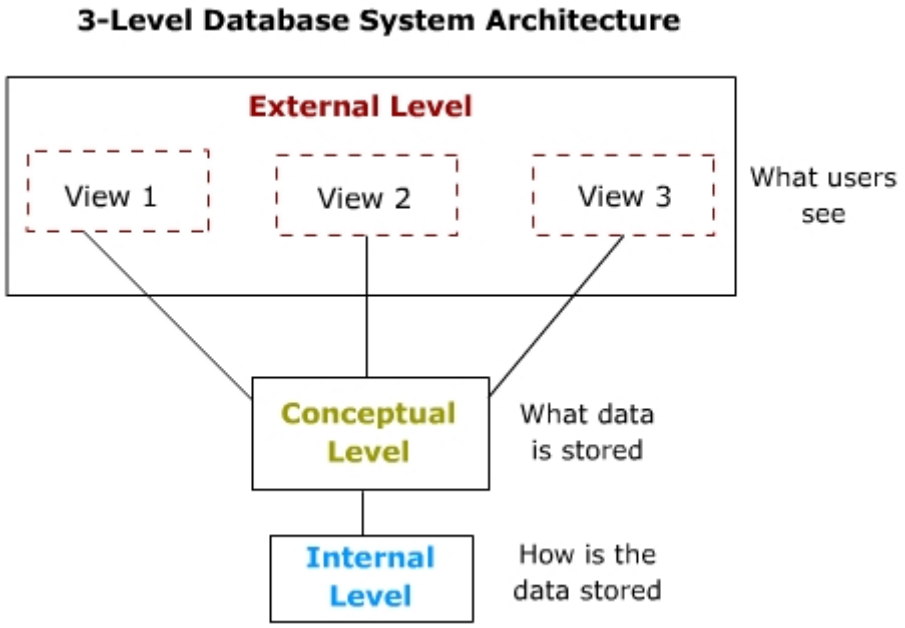
SALES-HISTORY

PART NO.	REGION	YEAR	UNITS
S3	East	1996	2000
S3	North	1996	5500

S3	South	1996	12000
S3	West	1996	20000

The recent developments in the area have shown up in the form of certain object and object/relational DBMS products. Examples of such systems are GemStone and Versant ODBMS. Research has also proceeded on to a variety of other schemes including the multi-dimensional approach and the logic-based approach.

3-Level Database System Architecture



- The External Level represents the collection of views available to different end-users.
- The Conceptual level is the representation of the entre information content of the database.
- The Internal level is the physical level which shows how the data data is stored, what are the representation of the fields etc.