

DATABASE MANAGEMENT SYSTEM - I

Categorisation of DBMS systems. Network, Hierarchical and Relational database. Application of DBMS system. Entity relationship charts. Relational database Management systems (RDBMS). Why to use them and where. Data manipulation language (DML) and Data control Language(DCL). Security consideration in DBMS, performance improvement in databases.

DATA BASE MANAGEMENT SYSTEM - II

Relational database - Advance concepts, Introduction to ORACLE / INGRESS or a similar RDBMS on a multi user environment. Structure Query Language (SQL). Form design on an advanced RDBMS. Report generation. Query by Example (QBE)and Report by form. Accessing RDBMS Using management. Security consideration.

Data are characteristics or information, usually numerical, that are collected through observation.^[1] In a more technical sense, data is a set of values of qualitative or quantitative variables about one or more persons or objects, while a **datum** (singular of data) is a single value of a single variable

the quantities, characters, or symbols on which operations are performed by a computer, which may be stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media.

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:

- Enterprise Information
 - Sales: For customer, product, and purchase information.

- Accounting: For payments, receipts, account balances, assets and other accounting information.
- Human resources: For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.
- Manufacturing: For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.
- Online retailers: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.

- Banking and Finance

- Banking: For customer information, accounts, loans, and banking transactions.
- Credit card transactions: For purchases on credit cards and generation of monthly statements.
- Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.

- Universities

For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).

- Airlines:

For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.

- Telecommunication:

For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

The Internet revolution of the late 1990s sharply increased direct user access to databases. Organizations converted many of their phone interfaces to databases into Web interfaces, and made a variety of services and information available online. For instance, when you access an online bookstore and browse a book or music collection, you are accessing data stored in a database. When you enter an order online, your order is stored in a database. When you access a bank Web site and retrieve your bank balance and transaction information, the information is retrieved from the bank's database system

database system vendors like Oracle are among the largest software companies in the world, and database systems form an important part of the product line of Microsoft and IBM.

Purpose of Database Systems

Database systems arose in response to early methods of computerized management of commercial data. As an example of such methods, typical of the 1960s, consider part of a university organization that, among other data, keeps information about all instructors, students, departments, and course offerings. 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 programs to:

- Add new students, instructors, and courses
- Register students for courses and generate class rosters
- Assign grades to students, compute grade point averages (GPA), and generate transcripts

System programmers wrote these application programs to meet the needs of the university

New application programs are added to the system as the need arises. For example, suppose that a university decides to create a new major (say, computer science). As a result, the university creates a new department and creates new permanent files (or adds information to existing files) to record information about all the instructors in the department, students in that major, course offerings, degree requirements, etc. The university may have to write new application programs to deal with rules specific to the new major. New application programs may also have to be written to handle new rules in the university. Thus, as time goes by, the system acquires more files and more application programs. This typical 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) were introduced, 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 structures and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). For example, if a student has a double major (say, music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematics department. 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 student address may be reflected in the Music department records but not elsewhere in the system.

- Difficulty in accessing data.

Suppose that one of the university clerks needs to find out the names of all students who live within a particular postal-code area. The clerk 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 students. The university clerk has now two choices: either obtain the list of all students and extract the needed information manually or ask a 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 clerk needs to trim that list to include only those students who have taken at least 60 credit hours. As expected, a program to generate such a list does not exist. Again, the clerk has the preceding two options, neither of which is satisfactory. The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

- 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. Suppose the university maintains an account for each department, and records the balance amount in each account. Suppose also that the university requires that the account balance of a department may never fall below zero. 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 device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the database be restored to the consistent state that existed prior to the failure. Consider a program to transfer \$500 from the account balance of department A to the account balance of department B. If a system failure occurs during the execution of the program, it is possible that the \$500 was removed from the balance of department A but was not credited to the balance of department 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. Indeed, today, the largest Internet retailers may have millions of accesses per day to their data by shoppers. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data. Consider department A, with an account balance of \$10,000. If two department clerks debit the account balance (by say \$500 and \$100, respectively) of department A at almost exactly the same time, the result of the concurrent executions may leave the budget 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 \$10,000, and write back \$9500 and \$9900, respectively. Depending on which one writes the value last, the account balance of department A may contain either \$9500 or \$9900, rather than the correct value of \$9400. 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. As another example, suppose a registration program maintains a count of students registered for a course, in order to enforce limits on the number of students registered. When a student registers, the program reads the current count for the courses, verifies that the count is not already at the limit, adds one to the count, and stores the count back in the database. Suppose two students register concurrently, with the count at (say) 39. The two program executions may both read the value 39, and both would then write back 40, leading to an incorrect increase of only 1, even though two students successfully registered for the course and the count should be 41. Furthermore, suppose the course registration limit was 40; in the above case both students would be able to register, leading to a violation of the limit of 40 students.

- Security problems.

Not every user of the database system should be able to access all the data. For example, in a university, payroll personnel need to see only that part of the database that has financial information. They do not need access to information about academic records. But, since application programs are added to the file-processing