LECTURE NOTES ON
DATABASE MANAGEMENT SYSTEMS

IV Semester

DATABASE MANAGEMENT SYSTEMS

Objectives:

- To Understand the basic concepts and the applications of database systems
- To Master the basics of SQL and construct queries using SQL
- To understand the relational database design principles
- To become familiar with the basic issues of transaction processing and concurrency control
- To become familiar with database storage structures and access techniques

UNIT I:

Database System Applications, Purpose of Database Systems, View of Data – Data Abstraction –Instances and Schemas – Data Models – the ER Model – Relational Model – Other Models – Database Languages – DDL – DML – database Access for applications Programs – Database Users and Administrator – Transaction Management – Database Architecture – Storage Manager – the Query Processor.
Introduction to the Relational Model – Structure – Database Schema, Keys – Schema Diagrams.
Database design and ER diagrams – ER Model - Entities, Attributes and Entity sets – Relationships and Relationship sets – ER Design Issues – Concept Design – Conceptual Design with relevant Examples. Relational Query Languages, Relational Operations.

Introduction to Database Management System

Database management system consists of two parts. They are:

1. Database and
2. Management System

What is a Database?

Database: Collection of related relations or Tables. Consider the following collection of tables. In a database, data is organized strictly in row and column format. The rows are called Tuple or Record. The data items within one row may belong to different data types. On the other hand, the columns are often called Domain or Attribute. All the data items within a single attribute are of the same data type.

What is Data?

Data is a representation of Facts, figures, statistics etc called as a data item having no particular meaning. Eg. 100, Krishna, 19

What is a Record?

Record is defined as a Collection of related data items.

Eg: In the above example, the three data items convey no meaning by itself. But if we organize them in some manner, then they collectively represent a meaningful information.

Roll Number	Name	Age
100	Krishna	19

Roll Number	Name	Age
100	Krishna	19
120	Jaya	21

124	John	20

What is a Table or Relation?

It is a collection of related records. The columns of the relation are called as Fields, Attributes or Domains. The rows are called as Tuples or Records.

Roll Number	Address
100	KOL
120	DEL
124	MUM

Table1

Roll Number	Name	Age
100	Krishna	19
120	Jaya	21
124	John	20

Table2

Roll Number	Year
100	2000
120	2001
124	2001

Table T3

Year	Hostel
2000	H1
2001	H2

Table T4

Data items Roll Number, Address, Age , Year and Hostel are called as Attributes in T1,T2,T3 and T4 tables.

Figure 1.1: Organizational DBMS

For example, Any organization consists of many departments and each department has employees who require different types of data for their activities everyday. Each user will have different levels of access to the database with their own customized front-end application.

What is Data Base Management System?

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.
- Management of data in DBMS involves both defining structures for storage of data items and providing mechanisms for the manipulation of data items.
- The database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.

Database Management System (DBMS) and Its Applications:

A Database management system is a computerized record-keeping system. It is a repository or a container for collection of computerized data files. The overall purpose of DBMS is to allow the users to define, store, retrieve and update the information contained in the database on demand. Information can be anything that is of significance to an individual or organization.

Major areas of application are as follows:

- Banking
- Airlines
- Universities
- Manufacturing and selling
- Human resources

Need of Database Systems(DBMS)

Consider a university that has a requirement of storing and maintaining data of its instructors, students, departments, and course offerings using a file processing system. This type of 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.

File processing System : Disadvantages

1. Data redundancy and inconsistency. Files created by users are likely to have different structures accessed by several programming languages. The same information may be duplicated in several places (files). For example, if a student has registered for two courses (say, Physics and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Physics 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 copies of the same data may not exist same in all the files. For example, a changed student address may be reflected in the Physics department records but not elsewhere in the system.

2. Difficulty in accessing data. Suppose that one of the university new Faculty has a requirement to generate a report of the names of all students who live within a particular postal-code area. Then to generate a list of students from different files without a program will be a very challenging Task.

3. Data isolation: As data is scattered in various files, and files are in different formats, writing a new application program to retrieve the appropriate data is difficult.

4. Data Integrity problems: The data values stored in the database must satisfy certain types of consistency constraints. Suppose that the university has a new rule that the account balance of a department may never fall below zero. when such new constraints are added, the

problem is compounded as it involves several programs and data files to be verified before enforcing the new rule in the program.

5. Atomicity problems: In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.

   Eg: Consider 500 Rs was removed from the account balance of department A but was not credited to the balance of department B, resulting in an inconsistent database state.. 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.

6. Concurrent-access anomalies: Consider department AA, with an account balance of 5000 Rs. If two department employees debit the account balance (by say 500 Rs and 100 Rs, respectively) of department AA 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 5000 Rs, and write back 4500 Rs and 4900 Rs respectively. Depending on which one writes the value last, the account balance of department AA may contain either 4500Rs or 4900 Rs rather than the correct value of 4400Rs. 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.

7. 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 system in an ad hoc manner, enforcing such security constraints is difficult.

These Challenges prompted the development of database systems. In what follows, we shall see the concepts and algorithms that enable database systems to solve the problems with file- processing systems.

Advantages of DBMS:

1. Controlling of Redundancy: Data redundancy refers to the duplication of data (i.e storing same data multiple times). In a database system, by having a centralized database and centralized control of data the unnecessary duplication of data is avoided. It also eliminates the extra time for processing the large volume of data. It results in saving the storage space.
2. Improved Data Sharing : DBMS uses user authentication and user privileges control mechanism for sharing the data.
3. Data Integrity: Integrity means that the data in the database is accurate. Centralized control of the data helps in permitting the administrator to define integrity constraints to the data in the database.

    a. For example: in customer database we can enforce an integrity that it must accept the customer only from Hubli and Bengaluru city.

4. Security: The DBMS has the facility to define authorization checks to be carried out whenever access to sensitive data is attempted.
5. Data Consistency: By eliminating data redundancy, we greatly reduce the opportunities for inconsistency. For example is a customer address is stored only once, we cannot have disagreement on the stored values. Also updating data values is greatly simplified when each value is stored in one place only. Finally, we avoid the wasted storage that results from redundant data storage.
6. Efficient Data Access: In a database system, the data is managed by the DBMS and all access to the data is through the DBMS providing a key to effective data processing
7. Enforcements of Standards: With the centralized of data, DBA can establish and enforce the data standards which may include the naming conventions, data quality standards etc.
8. Data Independence: Ina database system, the database management system provides the interface between the application programs and the data. When changes are made to the data representation, the meta data obtained by the DBMS is changed but the DBMS is continuing to provide the data to application program in the previously used way. The DBMs handles the task of transformation of data wherever necessary.
9. Reduced Application Development and Maintenance Time: DBMS supports many important functions that are common to many applications, accessing data stored in the DBMS, which facilitates the quick development of application.

Disadvantages of DBMS:

1. It is bit complex. Since it supports multiple functionality to give the user the best, the underlying software has become complex. The designers and developers should have thorough knowledge about the software to get the most out of it.
2. it uses large amount of memory. It also needs large memory to run efficiently.
3. DBMS system works on the centralized system, i.e.; all the users from all over the world access this database. Hence any failure of the DBMS, will impact all the users.
4. DBMS is generalized software, i.e.; it is written to work on the entire systems rather specific one. Hence some of the application will run slow.

Views of Data:

A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data. A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.

DBMS : Levels of Abstraction:

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many database-system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:

Physical level (or Internal View / Schema): The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.

Logical level (or Conceptual View / Schema): The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as physical data independence.

View level (or External View / Schema): The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

For example, we may describe a record as follows:

```
     type instructor = record
                    ID : char (5);
                    name : char (20);
                    dept name : char (20);
                    salary : numeric (8,2);
    end;
```

This code defines a new record type called instructor with four fields. Each field has a name and a type associated with it. A university organization may have several such record types, including

- department, with fields dept_name, building, and budget
- course, with fields course_id, title, dept_name, and credits
- student, with fields ID, name, dept_name, and tot_cred

1. At the physical level, an instructor, department, or student record can be described as a block of consecutive storage locations.
2. At the logical level, each such record is described by a type definition, as in the previous code segment, and the interrelationship of these record types is defined as well.
3. Finally, at the view level, computer users see a set of application programs that hide details of the data types. At the view level, several views of the database are defined, and a database user sees some or all of these views.

Instances and Schemas

Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an instance of the database. The overall design of the database is called the database schema. Schemas are changed infrequently, if at all. The concept of database schemas and instances can be understood by analogy to a program written in a programming language.

Each variable has a particular value at a given instant. The values of the variables in a program at a point in time correspond to an instance of a database schema. Database systems have several schemas, partitioned according to the levels of abstraction. The physical schema describes the database design at the physical level, while the logical schema describes the database design at the logical level. A database may also have several schemas at the view level, sometimes called subschemas, which describe different views of the database. Of these, the logical schema is by far the most important, in terms of its effect on application programs, since programmers construct applications by using the logical schema. Application programs are said to exhibit physical data independence if they do not depend on the physical schema, and thus need not be rewritten if the physical schema changes.

Data Models

Underlying the structure of a database is the data model: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
The data models can be classified into four different categories:

Relational Model. The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations. The relational model is an example of a record-based model.

Entity-Relationship Model. The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects.
Suppose that each department has offices in several locations and we want to record the locations at which each employee works.

Object-Based Data Model. Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led to the development of an object- oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity.

Semi-structured Data Model. The semi-structured data model permits the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The Extensible Markup Language (XML) is widely used to represent semi- structured data.

Historically, the network data model and the hierarchical data model preceded the relational data model. These models were tied closely to the underlying implementation, and complicated the task of modeling data.
As a result they are used little now, except in old database code that is still in service in some places.

Database Languages

A database system provides a data-definition language to specify the database schema and a data-manipulation language to express database queries and updates. In practice, the data-definition and data-manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language.

Data-Manipulation Language

A data-manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:

- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information from the database
- Modification of information stored in the database

There are basically two types:

1. Procedural DMLs require a user to specify what data are needed and how to get those data.
2. Declarative DMLs (also referred to as nonprocedural DMLs) require a user to specify what data are needed without specifying how to get those data.

A query is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a query language. Although technically incorrect, it is common practice to use the terms query language and data-manipulation language synonymously.

Data-Definition Language (DDL)

We specify a database schema by a set of definitions expressed by a special language called a data-definition language (DDL). The DDL is also used to specify additional properties of the data.

- Domain Constraints. A domain of possible values must be associated with every attribute (for example, integer types, character types, date/time types). Declaring an attribute to be of a particular domain acts as a constraint on the values that it can take. Domain constraints are the most elementary form of integrity constraint. They are tested easily by the system whenever a new data item is entered into the database.

- Referential Integrity. There are cases where we wish to ensure that a value that appears in one relation for a given set of attributes also appears in a certain set of attributes in another relation (referential integrity). For example, the department listed for each course must be one that actually exists. More precisely, the dept name value in a course record must appear in the dept name attribute of some record of the department relation.
- Assertions. An assertion is any condition that the database must always satisfy. Domain constraints and referential-integrity constraints are special forms of assertions. However, there are many constraints that we cannot express by using only these special forms. For example, “Every department must have at least five courses offered every semester” must be expressed as an assertion.
- Authorization. We may want to differentiate among the users as far as the type of access they are permitted on various data values in the database. These differentiations are expressed in terms of authorization, the most common being: read authorization, which allows reading, but not modification, of data; insert authorization, which allows insertion of new data, but not modification of existing data; update authorization, which allows modification, but not deletion, of data; and delete authorization, which allows

deletion of data. We may assign the user all, none, or a combination of these types of authorization.

The DDL, just like any other programming language, gets as input some instructions (statements) and generates some output. The output of the DDL is placed in the data dictionary, which contains metadata—that is, data about data.

Data Dictionary

Data dictionary is defined as a DBMS component that stores the definition of data characteristics and relationships. The DBMS data dictionary provides the DBMS with its self-describing characteristic.

For example, the data dictionary typically stores descriptions of all:

- Data elements that are defined in all tables of all databases. Specifically, the data dictionary stores the name, datatypes, display formats, internal storage formats, and validation rules.
- The data dictionary tells where an element is used, by whom it is used and so on.
- Data dictionary is likely to store the name of the table creator, the date of creation access authorizations, the number of columns, and so on.
- For each index the DBMS stores at least, the index name the attributes used, the location, specific index characteristics and the creation date.
- Who created each database, the date of creation where the database is located, who the DBA is and so on.
- End users and The Administrators of the data base Programs that access the database including screen formats, report formats application formats, SQL queries and so on.
- Access authorization for all users of all databases.
- Relationships among data elements which elements are involved: whether the relationship are mandatory or optional, the connectivity and cardinality and so on.

Database Administrators and Database Users

A primary goal of a database system is to retrieve information from and store new information in the database.

Database Users and User Interfaces

There are four different types of database-system users, differentiated by the way they expect to interact with the system. Different types of user interfaces have been designed for the different types of users.

Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. For example, a bank teller who needs to transfer $50 from account A to account B invokes a program called transfer.

Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.

Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language. They submit each such query to a query processor; whose function is to break down DML statements into instructions that the storage manager understands. Analysts who submit queries to explore data in the database fall in this category.

Online analytical processing (OLAP) tools simplify analysts' tasks by letting them view summaries of data in different ways. For instance, an analyst can see total sales by region (for example, North, South, East, and West), or by product, or by a combination of region and product (that is, total sales of each product in each region).

Database System Architecture:

The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs. Database systems can be centralized, or client-server, where one server machine executes work on behalf of multiple client machines. Database systems can also be designed to exploit parallel computer architectures. Distributed databases span multiple geographically separated machines.
A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components. The storage manager is important because databases typically require a large amount of storage space. The query processor is important because it helps the database system simplify and facilitate access to data.

Query Processor: Query evaluation engine, which executes low-level instructions generated by the DML compiler.

The query processor components include
• DDL interpreter, which interprets DDL statements and records the definitions in the data dictionary.
• DML compiler, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands. A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs query optimization, that is, it picks the lowest cost evaluation plan from among the alternatives.

Storage Manager:
A storage manager is a program module that provides the 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 components include:
• Authorization and integrity manager: tests for the satisfaction of integrity constraints and checks the authority of users to access data.
• Transaction manager: ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
• File manager: manages 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 deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

Transaction Manager:

A transaction is a collection of operations that performs a single logical function in a database application. Each transaction is a unit of both atomicity and consistency. Thus, we require that transactions do not violate any database-consistency constraints.

Conceptual Database Design - Entity Relationship (ER) Modeling:

Database Design Techniques

- ER Modeling (Top down Approach)
- Normalization (Bottom Up approach