

Database Management System

Unit I

General Introduction to database System

Syllabus

Database –DBMS distinction, Approaches to building a database. Data models, Three-schema architecture of a database, Challenges in building a DBMS, Various components of a DBMS, ER data model, SQL, PL/SQL Concept.

Unit II

Syllabus

Relational Data Model, Concept of relations, Schema-instance distinction, keys, referential integrity and foreign keys, relational algebra operators, Tuple Relation calculus, domain relational calculus.

Unit III

Syllabus

Physical and logical Hierarchy, concept of index, B- tree, hash index, function index, bitmap index, concept of Functional dependency, Normalization, Business data analytics, tool and techniques for business data analytics.

Database Management System

Unit I

Data

Data are raw or isolated facts from which the required information is produced. Data are distinct piece of information, usually formatted in a special way. Data can exist in a variety of forms that have meaning in the user's environment such as number or text on a piece of paper, bits stored in computer's memory or as facts stored in a person's mind.

Information

Data and information are closely related and are often used interchangeably. Information is processed, organized or summarized data. It may be defined as collection of related data that when put together, communicated meaningful and useful message to a recipient who uses it, to make decision or to interpret the data to get the meaning.

Metadata

Metadata is the data about the data. It is also called the system catalog, which is the self-describing a nature of the database that provides program-data independence. The metadata is the data that describe objects in the database and make easier for those objects to be accessed or manipulated. It describes the database structure, constraints, applications, authorization, and size of data types and so on.

Data Item or Field

A data item is the smallest unit of the data that's has meaning to its user. It is traditionally called a field or data element. It is an occurrence of the smallest unit of named data. It represented in the database by a value. Name, telephone number, bill amount are a few example of data.

Records

A record is a collection of logically related fields or data items, with each field possessing a fixed number of bytes and having a fixed data type. A record consists of values for each field. It is an occurrence of a named collection of zero, one or more than one data items or aggregates.

Files

Database Management System

A file is a collection of related sequence of records. All records in a file are of the same record type. If every record in the file has exactly the same size the file is said to be made up of **fixed-length records**. If different records in the file have different size the file is said to be made of **variable-length records**.

Database

A database is defined as collection of logically related data stored together that is designed to meet the information needs of an organization. A *database* is a collection of information that is organized so that it can easily be accessed, managed, and updated. In one view, *databases* can be classified according to types of content: bibliographic, full-text, numeric, and images.

Disadvantages of File Processing System

Data Redundancy

Data Redundancy means same information is duplicated in several files. This makes data redundancy.

Data Inconsistency

Data Inconsistency means different copies of the same data are not matching. That means different versions of same basic data are existing. This occurs as the result of update operations that are not updating the same data stored at different places. Example: Address Information of a customer is recorded differently in different files.

Difficulty in Accessing Data

It is not easy to retrieve information using a conventional file processing system. Convenient and efficient information retrieval is almost impossible using conventional file processing system.

Data Isolation

Data are scattered in various files, and the files may be in different format, writing new application program to retrieve data is difficult.

Integrity Problems

The data values may need to satisfy some integrity constraints. For example the balance field Value must be greater than 5000. We have to handle this through

Database Management System

program code in file processing systems. But in database we can declare the integrity constraints along with definition itself.

Atomicity Problem

It is difficult to ensure atomicity in file processing system. For example transferring \$100 from Account A to account Biff a failure occurs during execution there could be situation like \$100 is deducted from Account A and not credited in Account B.

Concurrent Access anomalies

If multiple users are updating the same data simultaneously it will result in inconsistent data state. In file processing system it is very difficult to handle this using program code. This results in concurrent access anomalies.

Security Problems

Enforcing Security Constraints in file processing system is very difficult as the application programs are added to the system in an ad-hoc manner.

Database Management System (DBMS)

Database management system (DBMS) is *system* software for creating and *managing databases*. The *DBMS* provides users and programmers with a systematic way to create, retrieve, update and *manage* data. **Database** is a collection of related data and data is a collection of facts and figures that can be processed to produce information. Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks. A **database management system** stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

Characteristics

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics –

Database Management System

Real-world entity – A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

Relation-based tables – DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.

Isolation of data and application – A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.

Less redundancy – DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.

Consistency – Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.

Query Language – DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.

ACID Properties – DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.

Multuser and Concurrent Access – DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.

Multiple views – DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

Database Management System

Security – Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features.

Advantages and Disadvantages of Database Management System (DBMS)

Advantages of Database Management System:

The DBMS has a number of advantages as compared to traditional computer file processing approach. The DBA must keep in mind these benefits or capabilities during designing databases, coordinating and monitoring the DBMS.

The major *advantages of DBMS* are described below.

1. Controlling Data Redundancy:

In non-database systems (traditional computer file processing), each application program has its own files. In this case, the duplicated copies of the same data are created at many places. In DBMS, all the data of an organization is integrated into a single database. The data is recorded at only one place in the database and it is not duplicated.

2. Data Consistency:

By controlling the data redundancy, the data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value (new value of item) is immediately available to all users.

If the DBMS has reduced redundancy to a minimum level, the database system enforces consistency. It means that when a data item appears more than once in the database and is updated, the DBMS automatically updates each occurrence of a data item in the database.

3. Data Sharing:

In DBMS, data can be shared by authorized users of the organization. The DBA manages the data and gives rights to users to access the data. Many users can be authorized to access the same set of information simultaneously. The remote users

Database Management System

can also share same data. Similarly, the data of same database can be shared between different application programs.

4. Data Integration:

In DBMS, data in database is stored in tables. A single database contains multiple tables and relationships can be created between tables (or associated data entities). This makes easy to retrieve and update data.

5. Integrity Constraints:

Integrity constraints or consistency rules can be applied to database so that the correct data can be entered into database. The constraints may be applied to data item within a single record or they may be applied to relationships between records.

There are also some standard constraints that are intrinsic in most of the DBMSs. These are;

Constraint Name	Description
PRIMARY KEY	Designates a column or combination of columns as Primary Key and therefore, values of columns cannot be repeated or left blank.
FOREIGN KEY	Relates one table with another table.
UNIQUE	Specifies that values of a column or combination of columns cannot be repeated.
NOT NULL	Specifies that a column cannot contain empty values.
CHECK	Specifies a condition which each row of a table must satisfy.

6. Data Security:

Data security is the protection of the database from unauthorized users. Only the authorized persons are allowed to access the database. Some of the users may be allowed to access only a part of database i.e., the data that is related to them or related to their department. Mostly, the DBA or head of a department can access all the data in the database. Some users may be permitted only to retrieve data, whereas others are allowed to retrieve as well as to update data. The database access is controlled by the DBA. He creates the accounts of users and gives rights to

Database Management System

access the database. Typically, users or group of users are given usernames protected by passwords.

7. Data Atomicity:

A transaction in commercial databases is referred to as atomic unit of work. For example, when you purchase something from a point of sale (POS) terminal, a number of tasks are performed such as;

- Company stock is updated.
- Amount is added in company's account.
- Sales person's commission increases etc.

All these tasks collectively are called an atomic unit of work or transaction. These tasks must be completed in all; otherwise partially completed tasks are rolled back. Thus through DBMS, it is ensured that only consistent data exists within the database.

8. Database Access Language:

Most of the DBMSs provide SQL as standard database access language. It is used to access data from multiple tables of a database.

9. Development of Application:

The cost and time for developing new applications is also reduced. The DBMS provides tools that can be used to develop application programs. For example, some wizards are available to generate Forms and Reports. Stored procedures (stored on server side) also reduce the size of application programs.

10. Creating Forms:

Form is very important object of DBMS. You can create Forms very easily and quickly in DBMS, once a Form is created, it can be used many times and it can be modified very easily. The created Forms are also saved along with database and behave like a software component.

11. Report Writers:

Most of the DBMSs provide the report writer tools used to create reports. The users can create reports very easily and quickly. Once a report is created, it can be used many times and it can be modified very easily. The created reports are also saved along with database and behave like a software component.

12. Control over Concurrency:

Database Management System

In a computer file-based system, if two users are allowed to access data simultaneously, it is possible that they will interfere with each other. For example, if both users attempt to perform update operation on the same record, then one may overwrite the values recorded by the other. Most DBMSs have sub-systems to control the concurrency so that transactions are always recorded" with accuracy.

13. Backup and Recovery Procedures:

In a computer file-based system, the user creates the backup of data regularly to protect the valuable data from damaging due to failures to the computer system or application program. It is a time consuming method, if volume of data is large. Most of the DBMSs provide the 'backup and recovery' sub-systems that automatically create the backup of data and restore data if required. For example, if the computer system fails in the middle (or end) of an update operation of the program, the recovery sub-system is responsible for making sure that the database is restored to the state it was in before the program started executing.

14. Data Independence:

The separation of data structure of database from the application program that is used to access data from database is called data independence. In DBMS, database and application programs are separated from each other. The DBMS sits in between them. You can easily change the structure of database without modifying the application program. For example you can modify the size or data type of a data items (fields of a database table).

15. Advanced Capabilities:

DBMS also provides advance capabilities for online access and reporting of data through Internet. Today, most of the database systems are online. The database technology is used in conjunction with Internet technology to access data on the web servers.

Disadvantages of Database Management System (DBMS):

Although there are many advantages but the DBMS may also have some minor *disadvantages*. These are:

1. Cost of Hardware & Software:

A processor with high speed of data processing and memory of large size is required to run the DBMS software. It means that you have to upgrade the hardware used for file-based system. Similarly, DBMS software is also Very costly.

Database Management System

2. Cost of Data Conversion:

When a computer file-based system is replaced with a database system, the data stored into data file must be converted to database files. It is difficult and time consuming method to convert data of data files into database. You have to hire DBA (or database designer) and system designer along with application programmers;

3. Cost of Staff Training:

Most DBMSs are often complex systems so the training for users to use the DBMS is required. Training is required at all levels, including programming, application development, and database administration. The organization has to pay a lot of amount on the training of staff to run the DBMS.

4. Appointing Technical Staff:

The trained technical persons such as database administrator and application programmers etc are required to handle the DBMS. You have to pay handsome salaries to these persons. Therefore, the system cost increases.

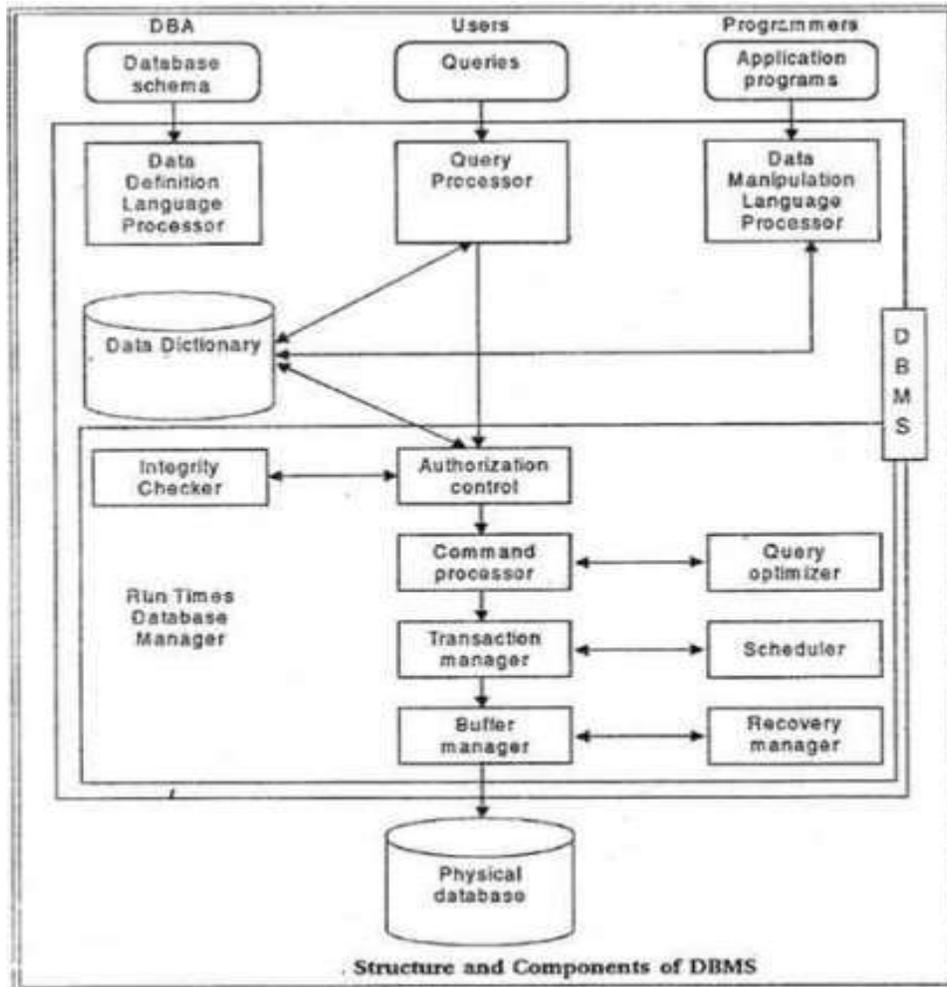
5. Database Failures:

In most of the organizations, all data is integrated into a single database. If database is corrupted due to power failure or it is corrupted on the storage media, then our valuable data may be lost or whole system stops.

Structure of DBMS

A typical structure of a DBMS with its components and relationships between them is show. The DBMS software is partitioned into several modules. Each module or component is assigned a specific operation to perform. Some of the functions of the DBMS are supported by operating systems (OS) to provide basic services and DBMS is built on top of it. The physical data and system catalo are stored on a physical disk. Access to the disk is controlled primarily by as, which schedules disk input/output.

Database Management System



As show, conceptually, following logical steps are followed while executing users to request to access the database system:

- Users issue a query using particular database language, for example, SQL commands.
- The passes query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for the evaluating the query.
- The DBMS accepts the users SQL commands and analyses them.
- The DBMS produces query evaluation plans, that is, the external schema for the user, the corresponding external/conceptual mapping, the conceptual

Database Management System

schema, the conceptual/internal mapping, and the storage structure definition. Thus, an evaluation\ plan is a blueprint for evaluating a query.

- The DBMS executes these plans against the physical database and returns the answers to the user.

Using components such as transaction manager, buffer manager, and recovery manager, the DBMS supports concurrency and recovery.

Components of a DBMS

The DBMS accepts the SQL commands generated from a variety of user interfaces, produces query evaluation plans, executes these plans against the database, and returns the answers. As shown, the major software modules or components of DBMS are as follows:

- (i) **Query processor:** The query processor transforms user queries into a series of low level instructions. It is used to interpret the online user's query and convert it into an efficient series of operations in a form capable of being sent to the run time data manager for execution. The query processor uses the data dictionary to find the structure of the relevant portion of the database and uses this **information** in modifying the query and preparing an optimal plan to access the database.
- (ii) **Run time database manager:** Run time database manager is the central software component of the DBMS, which interfaces with user-submitted application programs and queries. It handles database access at run time. It converts operations in user's queries coming. Directly via the query processor or indirectly via an application program from the user's logical view to a physical file system. It accepts queries and examines the external and conceptual schemas to determine what conceptual records are required to satisfy the user's request. It enforces constraints to maintain the consistency and integrity of the data, as well as its security. It also performs backing and recovery operations. Run time database manager is sometimes referred to as the database control system and has the following components:
 - **Authorization control:** The authorization control module checks the authorization of users in terms of various privileges to users.

Database Management System

- **Command processor:** The command processor processes the queries passed by authorization control module.
 - **Integrity checker:** It checks the integrity constraints so that only valid data can be entered into the database.
 - **Query optimizer:** The query optimizers determine an optimal strategy for the query execution.
 - **Transaction manager:** The transaction manager ensures that the transaction properties should be maintained by the system.
 - **Scheduler:** It provides an environment in which multiple users can work on same piece of data at the same time in other words it supports concurrency.
- (iii) **Data Manager:** The data manager is responsible for the actual handling of data in the database. It provides recovery to the system which that system should be able to recover the data after some failure. It includes Recovery manager and Buffer manager. The buffer manager is responsible for the transfer of data between the main **memory** and secondary storage. It is also referred as the cache manger.
- (iv) **DML Processor:** Using a DML compiler the DML processor converts the DML statement embedded in an application program into standard function calls in the host language. The DML compiler converts the DML statements written in a host programming language into object code for database access. The DML processor must interact with the query processor to generate the appropriate code.
- (v) **DLL Processor:** Using a DDL compiler the DDL processor converts the DLL statements into a set of tables containing metadata. This table contain the metadata concerning the database and are in a form that can be used by other components of the DBMS. The DLL compiler processes schema definition, specified in the DDL and stores description of the schema in the DBMS system catalo.

Codd's 12 Rules

Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.

These rules can be applied on any database system that manages stored data using only its relational capabilities. This is a foundation rule, which acts as a base for all the other rules.

Database Management System

Rule 1: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following – data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule

Database Management System

A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: Logical Data Independence

The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rules to apply.

Rule 10: Integrity Independence

A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

Rule 12: Non-Subversion Rule

If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

Data Model

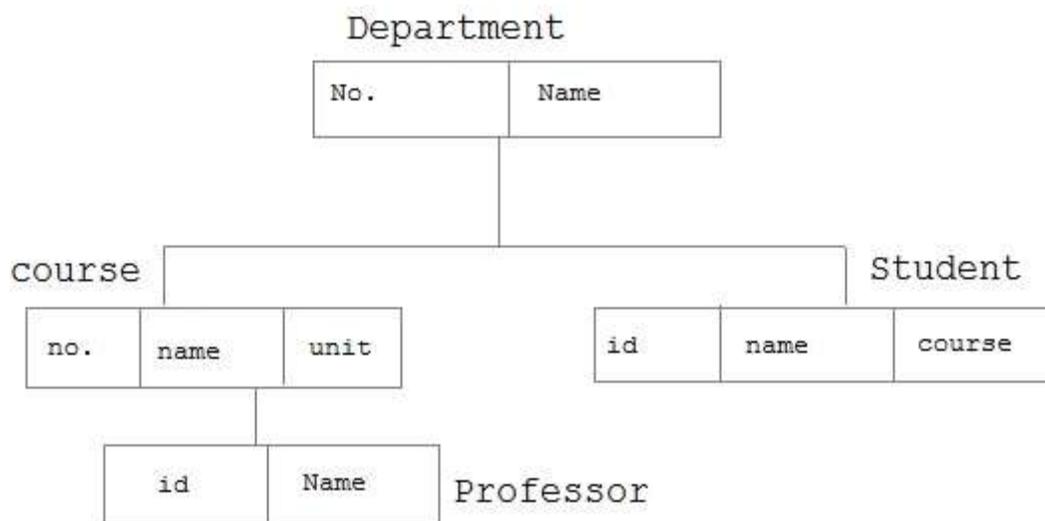
A data model is an abstraction process that concentrate essential and inherent aspects of the organisation's applications while ignore superfluous or accidental details

Database Management System

Hierarchical Database Model

Hierarchical Database model is one of the oldest database models, dating from late 1950s. One of the first hierarchical databases Information Management System (IMS) was developed jointly by North American Rockwell Company and IBM. This model is like a structure of a tree with the records forming the nodes and fields forming the branches of the tree

The hierarchical data model is represented by an upside-down tree. The perceives the hierarchical database as a hierarchy of segment. A segment is the equivalent of file system's record type. In a hierarchical data model, the relationship between the files or records forms a hierarchy. The hierarchical database is a collection of record that is perceives as organised to conform to the upside-down tree structure



Advantages

1. **Simplicity**

Data naturally have hierarchical relationship in most of the practical situations. Therefore, it is easier to view data arranged in manner. This makes this type of database more suitable for the purpose.

2. **Security**

These database systems can enforce varying degree of security feature unlike flat-file system.

3. **Database Integrity**

Because of its inherent parent-child structure, database integrity is highly promoted in these systems.

4. **Efficiency:** The hierarchical database model is a very efficient, one when the database contains a large number of I: N relationships (one-to-many relationships) and when the users require large number of transactions, using data whose relationships are fixed.

5. **Data sharing:** because all data are held in a common database data sharing become practical.

6. **Data Independence:** The DBMS creates an environment in which data independence can be maintained. This substantially decreases the programming efforts and program maintenance.

7. **Available expertise:** Due to a large number of available installed mainframe computer base expertise programmers were available.

8. **Tried business applications:** There was a large amount of tried-and- true business applications available within the mainframe environment.

Disadvantages

1. **Complexity of Implementation:** The actual implementation of a hierarchical database depends on the physical storage of data. This makes the implementation complicated.

2. **Difficulty in Management:** The movement of a data segment from one location to another cause all the accessing programs to be modified making database management a complex affair.

3. **Complexity of Programming:** Programming a hierarchical database is relatively complex because the programmers must know the physical path of the data items.

4. **Poor Portability:** The database is not easily portable mainly because there is little or no standard existing for these types of database.

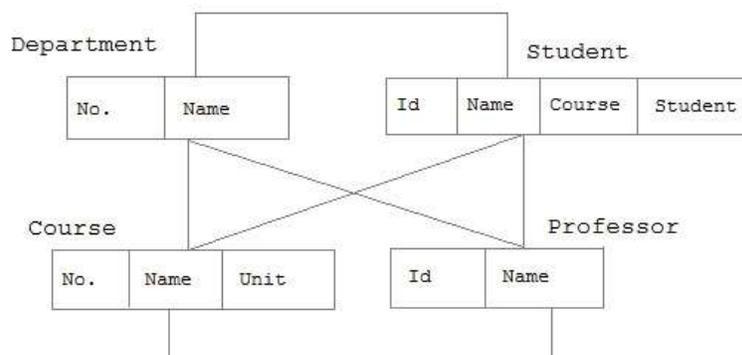
Database Management System

5. **Database Management Problems:** If you make any changes in the database structure of a hierarchical database, then you need to make the necessary changes in all the application programs that access the database. Thus, maintaining the database and the applications can become very difficult.
6. **Inflexibility:** A hierarchical database lacks flexibility. The changes in the new relation or segments often yield very complex system management tasks.
7. **No standards:** there is no precise set of standard concepts nor the does the implementation of model confirm to a specific standard in a hierarchical data model.

Network Data Model

The Network model replaces the hierarchical tree with a graph thus allowing more general connections among the nodes. The main difference of the network model from the hierarchical model, is its ability to handle many to many (N:N) relations. In other words, it allows a record to have more than one parent. Suppose an employee works for two departments. The strict hierarchical arrangement is not possible here and the tree becomes a more generalized graph - a network. The network model was evolved to specifically handle non-hierarchical relationships. As shown below data can belong to more than one parent. Note that there are lateral connections as well as top-down connections. A network structure thus allows 1:1 (one: one), 1: M (one: many), M: M (many: many) relationships among entities.

In network database terminology, a relationship is a set. Each set is made up of at least two types of records: an owner record (equivalent to parent in the hierarchical model) and a member record (similar to the child record in the hierarchical model).



Advantages and Disadvantages of Network Model

The Network model retains almost all the advantages of the hierarchical model while eliminating some of its shortcomings.

The main advantages of the network model are:

1. **Conceptual simplicity:** Just like the hierarchical model, the network model IS also conceptually simple and easy to design.
2. **Capability to handle more relationship types:** The network model can handle the one to- many (1:N) and many to many (N:N) relationships, which is a real help in modelling the real life situations.
3. **Ease of data access:** The data access is easier and flexible than the hierarchical model.
4. **Data Integrity:** The network model does not allow a member to exist without an owner. Thus, a user must first define the owner record and then the member record. This ensures the data integrity.
5. **Data independence:** The network model is better than the hierarchical model in isolating the programs from the complex physical storage details.
6. **Database Standards:** One of the major drawbacks of the hierarchical model was the non-availability of universal standards for database design and modelling. The network model is based on the standards formulated by the DBTG and augmented by ANSI/SP ARC (American National Standards Institute/Standards Planning and Requirements Committee) in the 1970s. All the network database management systems conformed to these standards. These standards included a Data Definition Language (DDL) and the Data Manipulation Language (DML), thus greatly enhancing database administration and portability.

Disadvantages of Network Model

Even though the network database model was significantly better than the hierarchical database model, it also had many drawbacks. Some of them are:

Database Management System

- 1. System complexity:** All the records are maintained using pointers and hence the whole database structure becomes very complex.
- 2. Operational Anomalies:** As discussed earlier, network model's insertion, deletion and updating operations of any record require large number of pointer adjustments, which makes its implementation very complex and complicated.
- 3. Absence of structural independence:** Since the data access method in the network database model is a navigational system, making structural changes to the database is very difficult in most cases and impossible in some cases. If changes are made to the database structure then all the application programs need to be modified before they can access data. Thus, even though the network database model succeeds in achieving data independence, it still fails to achieve structural independence.
- 4. Not User Friendly:** The network data model is not a design for user-friendly and is a highly skill-oriented system.

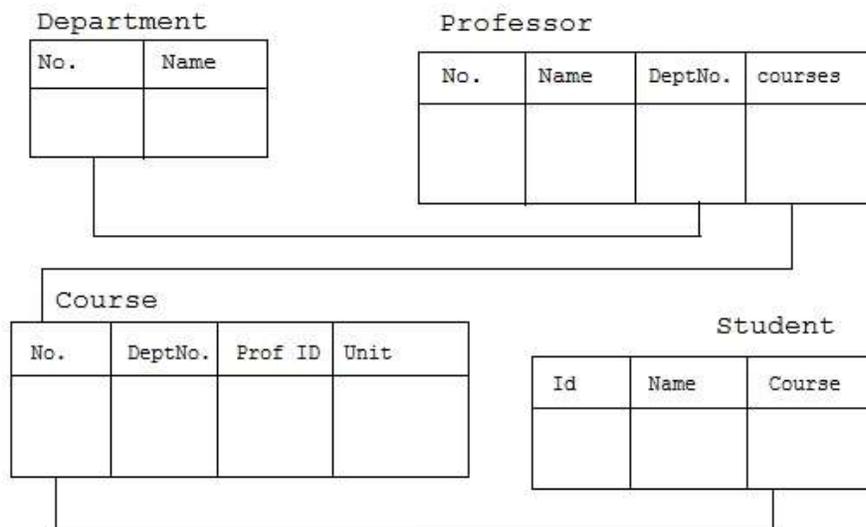
Relational Data Model

Relational model stores data in the form of tables. This concept purposed by Dr. E.F. Codd, a researcher of IBM in the year 1960s. The relational model consists of three major components:

1. The set of relations and set of domains that defines the way data can be represented (data structure).
2. Integrity rules that define the procedure to protect the data (data integrity).
3. The operations that can be performed on data (data manipulation).

A relational model database is defined as a database that allows you to group its data items into one or more independent tables that can be related to one another by using fields common to each related table.

Database Management System



Advantages and Disadvantages of Relational Model

The major advantages of the relational model are:

- 1. Structural independence:** In relational model, changes in the database structure do not affect the data access. When it is possible to make change to the database structure without affecting the DBMS's capability to access data, we can say that structural independence has been achieved. So, relational database model has structural independence.
- 2. Conceptual simplicity:** We have seen that both the hierarchical and the network database model were conceptually simple. But the relational database model is even simpler at the conceptual level. Since the relational data model frees the designer from the physical data storage details, the designers can concentrate on the logical view of the database.
- 3. Design, implementation, maintenance and usage ease:** The relational database model achieves both data independence and structure independence making the database design, maintenance, administration and usage much easier than the other models.
- 4. Ad hoc query capability:** The presence of very powerful, flexible and easy-to-use query capability is one of the main reasons for the immense popularity of

the relational database model. The query language of the relational database models structured query language or SQL makes ad hoc queries a reality. SQL is a fourth generation language (4GL). A 4 GL allows the user to specify what must be done without specifying how it must be done. So, using SQL the users can specify what information they want and leave the details of how to get the information to the database.

Disadvantages of Relational Model

The relational model's disadvantages are very minor as compared to the advantages and their capabilities far outweigh the shortcomings. Also, the drawbacks of the relational database systems could be avoided if proper corrective measures are taken. The drawbacks are not because of the shortcomings in the database model, but the way it is being implemented.

Some of the disadvantages are:

- 1. Hardware overheads:** Relational database system hides the implementation complexities and the physical data storage details from the users. For doing this, i.e. for making things easier for the users, the relational database systems need more powerful hardware **computers** and data **storage devices**. So, the RDBMS needs powerful machines to run smoothly. But, as the processing power of modern computers is increasing at an exponential rate and in today's scenario, the need for more processing power is no longer a very big issue.
- 2. Ease of design can lead to bad design:** The relational database is an easy to design and use. The users need not know the complex details of physical data storage. They need not know how the data is actually stored to access it. This ease of design and use can lead to the development and implementation of very poorly designed database management systems. Since the database is efficient, these design inefficiencies will not come to light when the database is designed and when there is only a small amount of data. As the database grows, the poorly designed databases will slow the system down and will result in performance degradation and data corruption.
- 3. 'Information island' phenomenon:** As we have said before, the relational database systems are easy to implement and use. This will create a situation where too many people or departments will create their own databases and applications.

Database Management System

Difference between DBMS and RDBMS

Although DBMS and RDBMS both are used to store information in physical database but there are some remarkable differences between them.

The main differences between DBMS and RDBMS are given below:

No.	DBMS	RDBMS
1)	DBMS applications store data as file.	RDBMS applications store data in a tabular form.
2)	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3)	Normalization is not present in DBMS.	Normalization is present in RDBMS.
4)	DBMS does not apply any security with regards to data manipulation.	RDBMS defines the integrity constraint for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.
5)	DBMS uses file system to store data, so there will be no relation between the tables.	in RDBMS, data values are stored in the form of tables, so a relationship between these data values will be stored in the form of a table as well.
6)	DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7)	DBMS does not support distributed database.	RDBMS supports distributed database.
8)	DBMS is meant to be for small organization and deal with	RDBMS is designed to handle large amount of data. it supports multiple

Database Management System

	small data. it supports single user.	users.
9)	Examples of DBMS are file systems, xml etc.	Example of RDBMS are mysql, postgre, sql server, oracle etc.

ER (Entity Relationship) Data Model

There are two techniques used for the purpose of data base designing from the system requirements. These are:

- Top down Approach known as Entity-Relationship Modelling
- Bottom Up approach known as Normalization.

It is a graphical technique, which is used to convert the requirement of the system to graphical representation, so that it can become well understandable. It also provides the framework for designing of database.

The Entity-Relationship (ER) model was originally proposed by Peter in 1976 as a way to unify the network and relational database views. Simply stated, the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram, which is used to visually represent data objects. For the database designer, the utility of the ER model is:

- It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.
- It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.
- In addition, the model can be used as a design plan by the database developer to implement a data model in specific database management software.

Advantages and Disadvantages of E-R Data Model

Following are advantages of an E-R Model:

- **Straightforward relation representation:** Having designed an E-R diagram for a database application, the relational representation of the database model becomes relatively straightforward.
- **Easy conversion for E-R to other data model:** Conversion from E-R diagram to a network or hierarchical data model can easily be accomplished.
- **Graphical representation for better understanding:** An E-R model gives graphical and diagrammatical representation of various entities, its attributes and relationships between entities. This in turn helps in the clear understanding of the data structure and in minimizing redundancy and other problems.

Disadvantages of E-R Data Model

Following are disadvantages of an E-R Model:

- **No industry standard for notation:** there is an industry standard notation for developing an E-R diagram.
- **Popular for high-level design:** The E-R data model is especially popular for high level.

DBA Responsibilities

Database Management System

- Installation, configuration and upgrading of software and related products.
- Establish and maintain sound backup and recovery policies and procedures.
- Take care of the Database design and implementation.
- Implement and maintain database security (create and maintain users and roles, assign privileges).
- Database tuning and performance monitoring.
- Application tuning and performance monitoring.
- Setup and maintain documentation and standards.
- Plan growth and changes (capacity planning).
- Work as part of a team and provide 7×24 supports when required.
- Do general technical trouble shooting and give consultation to development teams.
- Technical support.
- ITIL Skill set requirement (Problem Management/Incident Management/Chain Management etc)

Types of DBA

1. **Administrative DBA** – Work on maintaining the server and keeping it running. Concerned with backups, security, patches, replication, etc. Things that concern the actual server software.
2. **Development DBA** – works on building queries, stored procedures, etc. that meet business needs. This is the equivalent of the programmer. You primarily write T-SQL.
3. **Architect** – Design schemas. Build tables, FKs, PKs, etc. Work to build a structure that meets the business needs in general. The design is then used by developers and development DBAs to implement the actual application.
4. **Data Warehouse DBA** – Newer role, but responsible for merging data from multiple sources into a data warehouse. May have to design warehouse, but cleans, standardizes, and scrubs data before loading. In SQL Server, this DBA would use DTS heavily.
5. **OLAP DBA** – Builds multi-dimensional cubes for decision support or OLAP systems. The primary language in SQL Server is MDX, not SQL here

Approaches to building a Database

1. **Determine the purpose of the database** - This helps prepare for the remaining steps.

Database Management System

- 2. Find and organize the information required** - Gather all of the types of information to record in the database, such as product name and order number.
- 3. Divide the information into tables** - Divide information items into major entities or subjects, such as Products or Orders. Each subject then becomes a table.
- 4. Turn information items into columns** - Decide what information needs to be stored in each table. Each item becomes a field, and is displayed as a column in the table. For example, an Employees table might include fields such as Last Name and Hire Date.
- 5. Specify primary keys** - Choose each table's primary key. The primary key is a column, or a set of columns, that is used to uniquely identify each row. An example might be Product ID or Order ID.
- 6. Set up the table relationships** - Look at each table and decide how the data in one table is related to the data in other tables. Add fields to tables or create new tables to clarify the relationships, as necessary.
- 7. Refine the design** - Analyze the design for errors. Create tables and add a few records of sample data. Check if results come from the tables as expected. Make adjustments to the design, as needed.
- 8. Apply the [normalization rules](#)** - Apply the data normalization rules to see if tables are structured correctly. Make adjustments to the tables

An alternative approach to the traditional file processing system is the modern concept, known as the database approach. A database is an organised collection of records and files which are related to each other. In a database system, a common pool of data can be shared by a number of applications as it is data and program independent. Thus unlike a file processing system, data redundancy and data inconsistency in the database system approach are minimized. In this approach the user is free from the detailed and complicated task of keeping up with the physical

Database Management System

structure of data. A clear-cut distinction between traditional file system and database system is depicted by the following diagram

A database is organised in such a way that a computer program can quickly select the desired piece of data. A database can further be defined as,

- it is a collection of interrelated data stored together without harmful or unnecessary redundancy.
- Stores data independent of programs and any changes in data storage structure or access strategy do not require changes in accessing programs or queries.
- Serves multiple applications in which each user has its own view of data. The data is protected from unauthorized access by security mechanism and concurrent access to data is provided with recovery mechanism.

Broadly, the objectives of the database approach are to make information access easy, fast, relatively inexpensive and flexible for the user. The specific objectives may be listed as follows :

1. Controlled data redundancy
2. Enhanced data consistency
3. Data independence
4. Application independence
5. Ease of use
6. Recovery from failure

Three Schema Architecture of a Database

A DBMS is a collection of interrelated files and a set of programs that allow several users to access and modify these files. 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 is stored and maintained. We can imagine that the whole database system is divided into levels. The generalised architecture of a database system is called the ANSI/SPARC (American National Standards Institute - Standards Planning and Requirements Committee) model.

The objective of the three-level architecture is to separate the users' view,

- **It allows independent customized user views:** Each user should be able to access the same data, but have a different customized view of the data. These should be independent: changes to one view should not affect others.

Database Management System

- **It hides the physical storage details from users:** Users should not have to deal with physical database storage details.
- The database administrator should be able to change the database storage structures without affecting the users' views.
- The internal structure of the database should be unaffected by changes to the physical aspects of the storage: For example, a changeover to a new disk.

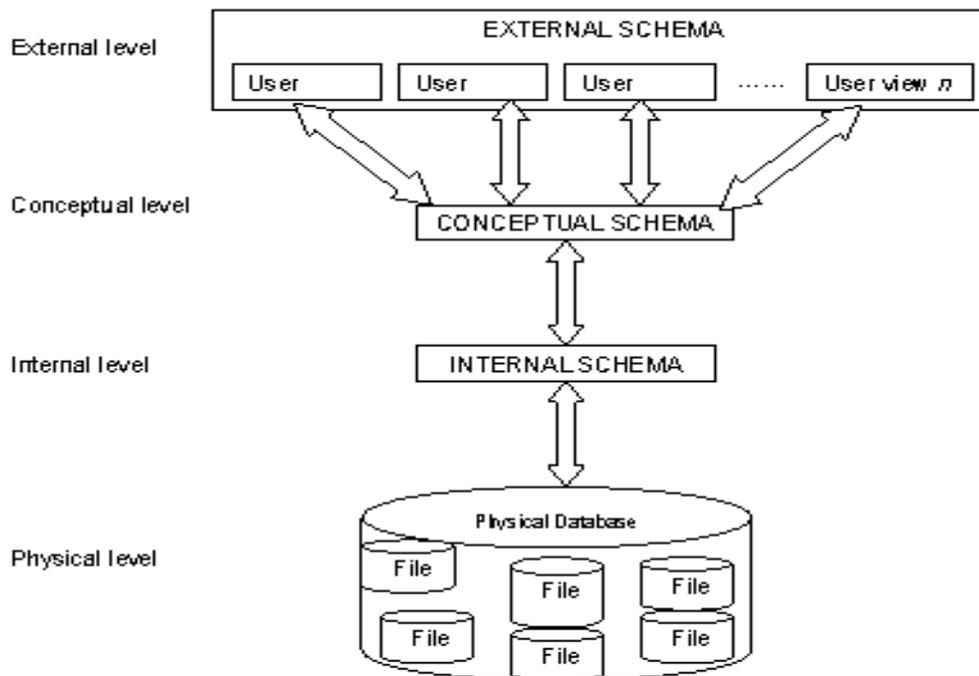


Fig. Three-tier database architecture

Three levels are:

- External level
- Conceptual level
- Internal level

External level:

The external level is the user's view of the database and closest to the users. This level describes that part of the database that is relevant to the user. Most of the users of database are not concerned with all the information contained in the database. Instead, they need only a part of the database relevant to them. For example, even though the bank database stores a lot more information, an account holder would be interested only in the account details such as the current balance

Database Management System

and the transactions made. They may not need the rest of the information stored in the account holder's database. An external schema describes each external view. The external schema consists of the definition of the logical records and the relationships in the external view.

Conceptual level:

Conceptual level is the middle level of the three-tier architecture. At this level of database abstraction, all the database entities and relationships among them are included. Conceptual level provides the community view of the database and describes what data is stored in the database and the relationships among the data. One conceptual view represents the entire database of an organization. It is a complete view of the data requirements of the organization that is independent of any storage consideration. The conceptual schema defines conceptual view. It is also called the logical schema. There is only one conceptual schema per database. The figure shows the conceptual view record of a data base.

Internal level or physical level:

The lowest level of abstraction is the internal level. It is the one closest to physical storage device. This level is also termed as physical level, because it describes how data are actually stored on the storage medium such as hard disk, magnetic tape etc. This level indicates how the data will be stored in the database and describe the data structures, file structures and access methods to be used by the database. The internal schema defines the internal level. The internal schema contains the definition of the stored record, the methods of representing the data fields and accessed methods used. The figure shows the internal view record of a database.

Challenges in building a DBMS

- 1) **Data redundancy and consistency:-** Different files have different formats of programs written in different programming languages by different users. So the same information may be duplicated in several files. It may lead to data inconsistency. If a customer changes his address, then it may be reflected in one copy of data but not in the other.
- 2) **Difficulty in accessing data:-** The file system environment does not allow needed data to be retrieved in a convenient and efficient manner.

Database Management System

- 3) **Data isolation:-** Data is scattered in various files; so it gets isolated because file may be in different formats.
- 4) **Integrity problems:-** Data values stored in the database must satisfy consistency constraints. Problem occurs when constraints involve several data items from different files.
- 5) **Atomicity problems:-** If failure occurs, data must be stored to constant state that existed prior to failure. For example, if in a bank account, a person abc is transferring Rs 5000 to the account of pqr, and abc has withdrawn the money but before it gets deposited to the pqr's account, the system failure occurs, then Rs5000 should be deposited back to abc's bank account.
- 6) **Concurrent access anomalies:-** Many systems allow multiple users to update data simultaneously. Concurrent updates should not result in inconsistent data.
- 7) **Security problems:-** Not every user of the database system should be able to access all data. Data base should be protected from access by unauthorized users.

Data Independence

We can define two types of data independence:

1. Logical data independence:

It is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), or to reduce the database (by removing a record type or data item). In the latter case, external schemas that refer only to the remaining data should not be affected. Only the view definition and the mappings need be changed in a DBMS that supports logical data independence. Application programs that reference the external schema constructs must work as before, after the conceptual schema undergoes a logical reorganization. Changes to constraints can be applied also to the conceptual schema without affecting the external schemas or application programs.

2. Physical data independence:

It is the capacity to change the internal schema without having to change the conceptual (or external) schemas. Changes to the internal schema may be needed because some physical files had to be reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema. Whenever we have a multiple level DBMS, its catalog must be expanded to include information on how to map requests and data among the various levels. The DBMS uses additional software to accomplish these mappings by referring to the mapping information in the catalog. Data independence is accomplished because, when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.

Data Abstraction:

Major purpose of DMBS is to provide users with abstract view of data i.e. the system hides certain details of how the data are stored and maintained.

Since database system users are not computer trained, developers hide the complexity from users through **3 levels of abstraction**, to simplify user's interaction with the system.

1) Physical level of data abstraction:

This is the lowest level of abstraction which describes how data are actually stored.

2) Logical level of data abstraction:

This level hides what data are actually stored in the database and what relationship exists among them.

3) View Level of data abstraction:

View provides security mechanism to prevent user from accessing certain parts of database.

Database Management System

SQL

SQL (Structured Query Language) is a standard interactive and programming **language** for getting information from and updating a database. Although **SQL** is both an ANSI and an ISO standard, many database products support **SQL** with proprietary extensions to the standard **language**.

Different types of database languages

A database system provides two different types of languages: One to specify the database schema and other to express database queries and updates.

Data-definition Languages (DDL): A database schema is specified by a set of definitions expressed by a special language (DDL). The result of compilation of DDL statements is a set of tables that is stored in a special file called data dictionary or data directory.

A data dictionary is a file that contains metadata that is data about data. This file is consulted before actual data are read or modified in the database system.

The storage structure and access methods used by the database system are specified by a set of definitions in a special type of DDL called as data storage and definition language. The result of compilation of these definitions is a set of instruction to specify the implementation details of the database schemas-Details are usually hidden from the user. Create table command is used for creation of the table. Suppose we wish to create table account having columns account_no, branch, and amount, then we have to write the following query;

```
create table account
(account_no number,
branch varchar2(50),
account number);
```

In this example, table account is created on which various modification operations can be performed.

Data Definition Language (DDL) : Statements are used to define the database structure or schema.

Some

examples:

Database Management System

- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

Data manipulation language (DML): By data manipulation, we mean

1. The retrieval of information stored in the database
2. The insertion of new information into the database
3. Deletion of information from the database.
4. The modification of information stored in the database.

At the physical level, we must define algorithms that allow efficient access to data. At higher levels of abstraction, we emphasize ease of use. The goal is to provide efficient human interaction system.

A data manipulation language (DML) is a language that enables users to access or manipulate data organised by the appropriate data model. There are basically two types:

a) **Procedural DMLs** requires a user to specify what data are needed and how to get those data.

b) **Non procedural DMLs** require a user to specify what data are needed without specifying how to get those data.

Non procedural DMLs are usually easier to learn and use than are procedural DMLs. However since a user does not have to specify, how to get those data, these languages may generate code that may not be as that efficient as that produced by procedural languages.

For example: The simplest insert statement is a request to insert one tuple. Suppose that we wish to insert the fact that there is an account A-9732 at a perryridge branch and that it has a balance of \$1200. The query is

Data Manipulation Language (DML) : Statements are used for managing data within schema objects.

Some examples:

Database Management System

- SELECT - Retrieve data from the a database
- INSERT - Insert data into a table
- UPDATE - Updates existing data within a table
- DELETE - deletes all records from a table, the space for the records remain
- MERGE - UPSERT operation (insert or update)
- CALL - Call a PL/SQL or Java subprogram
- EXPLAIN PLAN - explain access path to data
- LOCK TABLE - control concurrency

Data Control Language (DCL)

Some examples:

- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with the GRANT command

Transaction Control (TCL) : Statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

Some examples:

- COMMIT - save work done
- SAVEPOINT - identify a point in a transaction to which you can later roll back
- ROLLBACK - restore database to original since the last COMMIT
- SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use

Assertion

Assertions - An assertion is a piece of SQL which makes sure a condition is satisfied or it stops action being taken on a **database object**. It could mean locking out the whole table or even the whole database.

To make matters more confusing - a trigger could be used to enforce a check constraint and in some DBs can take the place of an assertion (by allowing you to run code un-related to the table being modified). A common mistake for beginners is to use a check constraint when a trigger is required or a trigger when a check constraint is required.

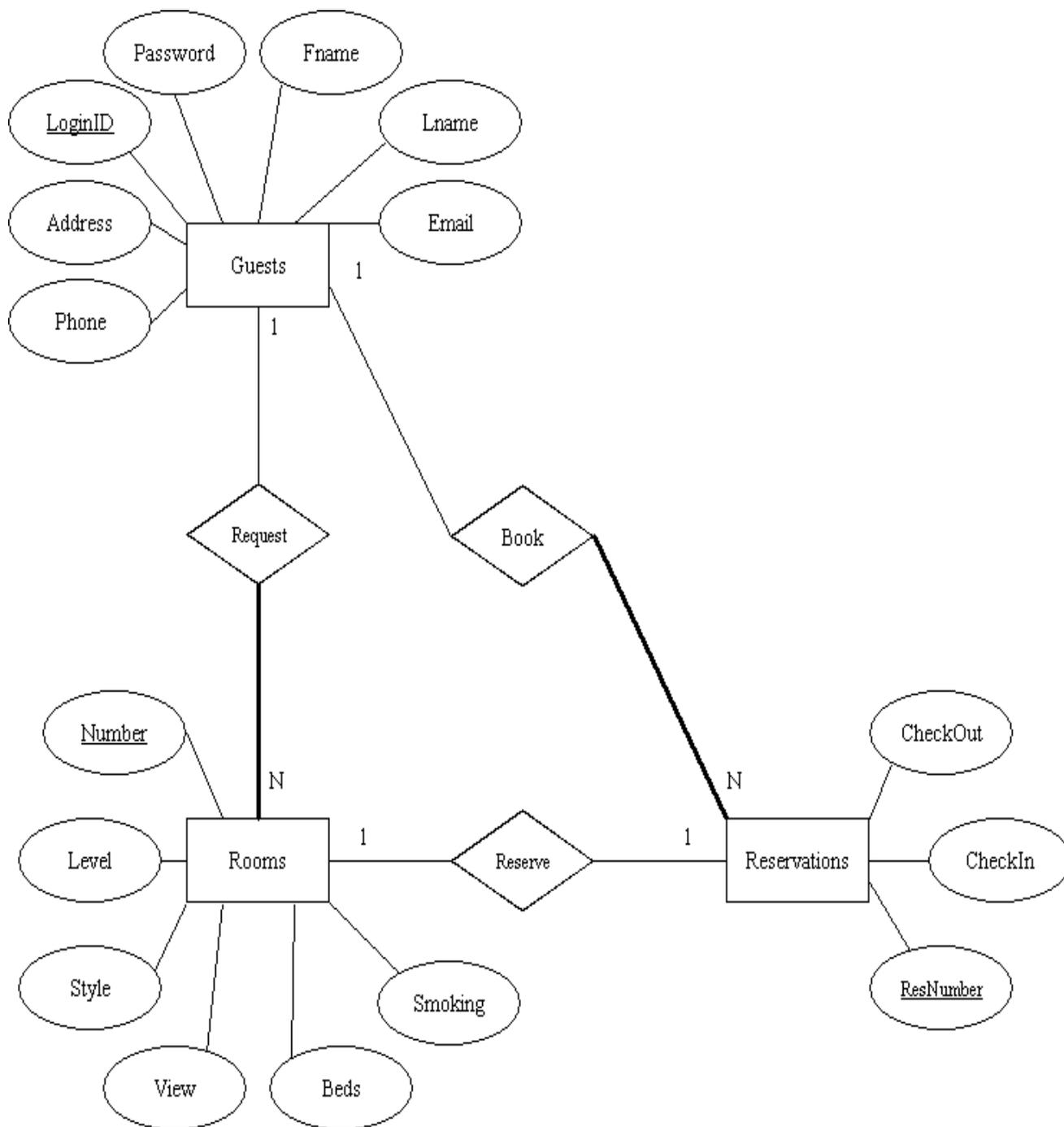
An example: All new customers opening an account must have a balance of \$100; however, once the account is opened their balance can fall below that amount. In this case you have

Database Management System

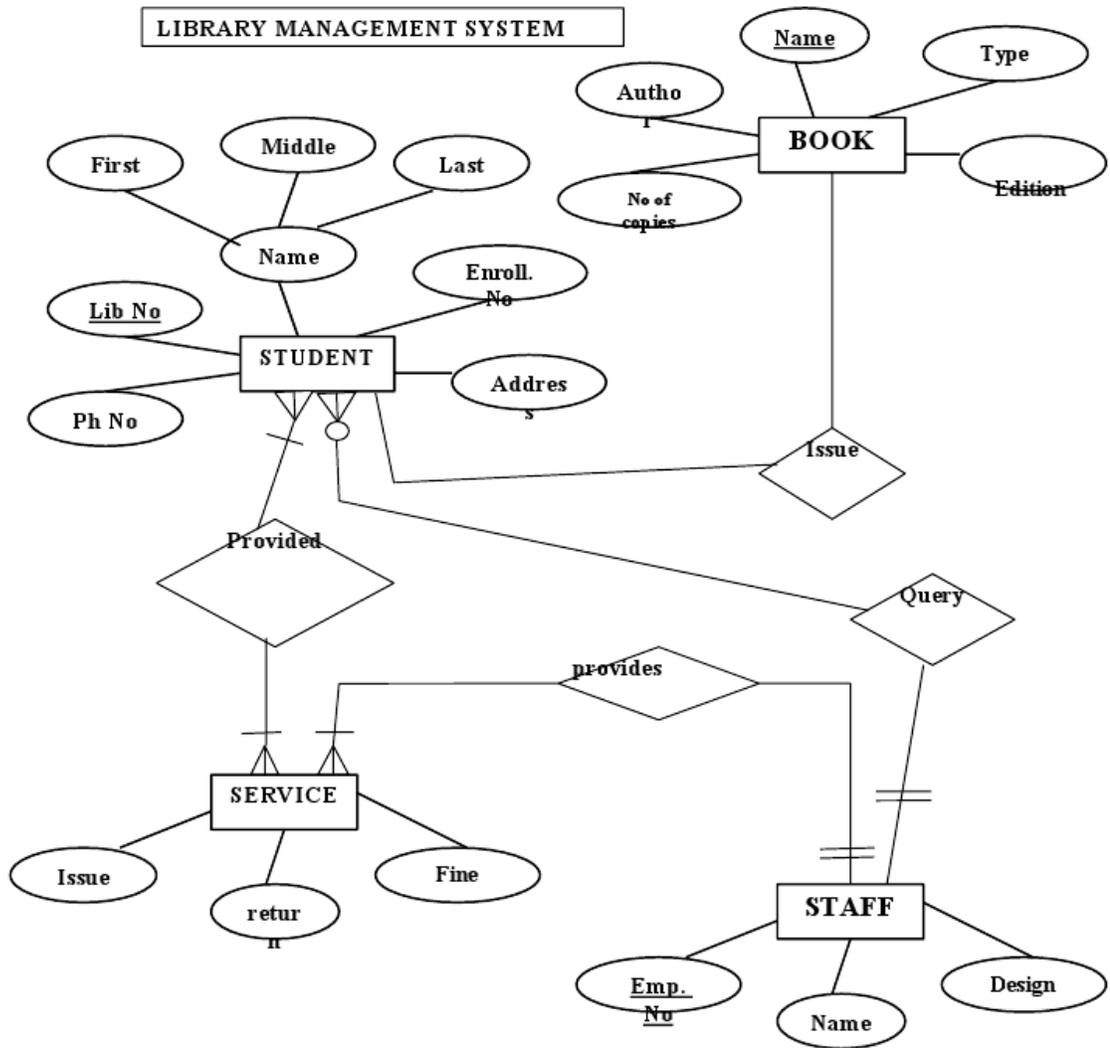
to use a trigger because you only want the condition evaluated when a new record is inserted.

ER Diagrams

1. Hotel Management



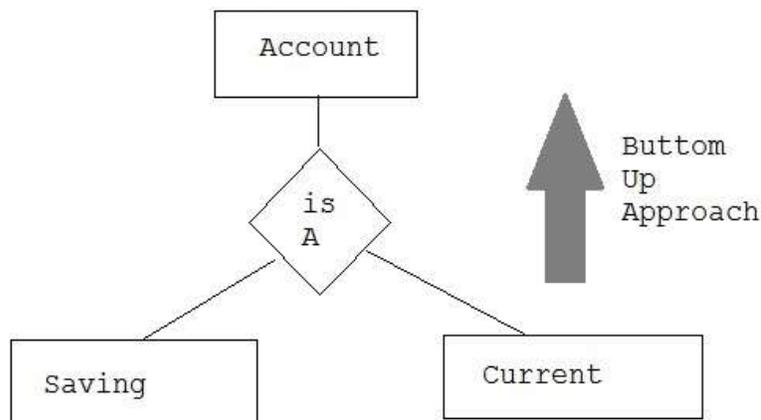
2. Library Management



Database Management System

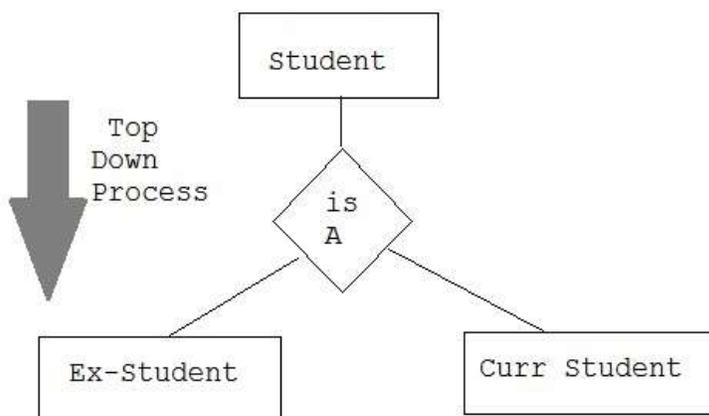
Generalization

Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entity to make further higher level entity.



Specialization

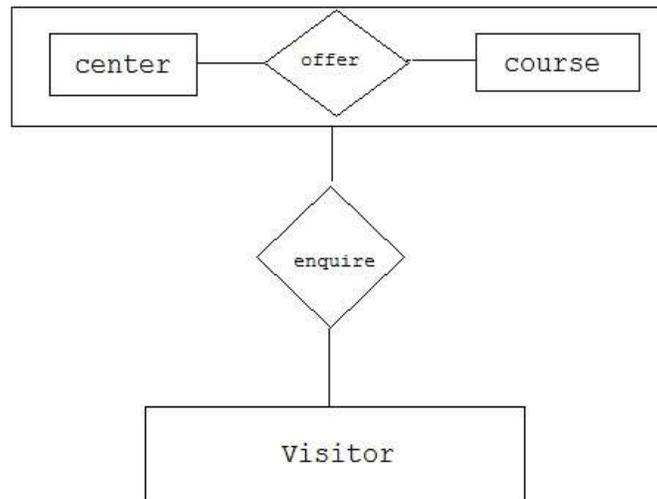
Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, some higher level entities may not have lower-level entity sets at all.



Database Management System

Aggregation

Aggregation is a process when relation between two entity is treated as a single entity. Here the relation between Center and Course, is acting as an Entity in relation with Visitor.



VIEW

A view is a "virtual table" in the database whose contents are defined by a query.

The tables of a database define the structure and organization of its data. However, SQL also lets you look at the stored data in other ways by defining alternative views of the data. A view is a SQL query that is permanently stored in the database and assigned a name. The results of the stored query are "visible" through the view, and SQL lets you access these query results as if they were, in fact, a "real" table in the database.

Views are an important part of SQL, for several reasons:

- Views let you tailor the appearance of a database so that different users see it from different perspectives.
- Views let you restrict access to data, allowing different users to see only certain rows or certain columns of a table.
- Views simplify database access by presenting the structure of the stored data in the way that is most natural for each user.

Advantages of VIEW

Views provide a variety of benefits and can be useful in many different types of databases. In a personal computer database, views are usually a convenience, defined to simplify

Database Management System

database requests. In a production database installation, views play a central role in defining the structure of the database for its users and enforcing its security. Views provide these major benefits:

- **Security:** Each user can be given permission to access the database only through a small set of views that contain the specific data the user is authorized to see, thus restricting the user's access to stored data.
- **Query simplicity:** A view can draw data from several different tables and present it as a single table, turning multi-table queries into single-table queries against the view.
- **Structural simplicity:** Views can give a user a "personalized" view of the database structure, presenting the database as a set of virtual tables that make sense for that user.
- **Insulation from change:** A view can present a consistent, unchanged image of the structure of the database, even if the underlying source tables are split, restructured, or renamed.
- **Data integrity:** If data is accessed and entered through a view, the DBMS can automatically check the data to ensure that it meets specified integrity constraints.

Disadvantages of VIEW

While views provide substantial advantages, there are also two major disadvantages to using a view instead of a real table:

Performance: Views create the appearance of a table, but the DBMS must still translate queries against the view into queries against the underlying source tables. If the view is defined by a complex, multi-table query, then even a simple query against the view becomes a complicated join, and it may take a long time to complete.

Update restrictions: When a user tries to update rows of a view, the DBMS must translate the request into an update on rows of the underlying source tables. This is possible for simple views, but more complex views cannot be updated; they are "read-only."

Unit II

STORAGE HIERARCHIES

The collection of data that makes up a computerized database must be stored physically on some computer storage medium. The DBMS software can then retrieve, update, and process this data as needed. Computer storage media form a storage hierarchy that includes two main categories:

- **Primary storage:** This category includes storage media that can be operated on directly by the computer central processing unit (CPU), such as the computer main memory and smaller but faster cache memories. Primary storage usually provides fast access to data but is of limited storage capacity.
- **Secondary storage:** This category includes magnetic disks, optical disks, and tapes. These devices usually have a larger capacity, cost less, and provide slower access to data than do primary storage devices. Data in secondary storage cannot be processed directly by the CPU; it must first be copied into primary storage.

Memory Hierarchies and Storage Devices

In a modern computer system data resides and is transported throughout a hierarchy of storage media. The highest speed memory is the most expensive and is therefore available with the least capacity. The lowest-speed memory is tape storage, which is essentially available in indefinite storage capacity. At the primary storage level, the memory hierarchy includes at the most expensive end cache memory, which is a static RAM (Random Access Memory). Cache memory is typically used by the CPU to speed up execution of programs. The next level of primary storage is DRAM (Dynamic RAM), which provides the main work area for the CPU for keeping programs and data and is popularly called main memory. The advantage of DRAM is its low cost, which continues to decrease; the drawback is its volatility and lower speed compared with static RAM. At the secondary storage level, the hierarchy includes magnetic disks, as well as mass storage in the form of CD-ROM (Compact Disk–Read-Only Memory) devices, and finally tapes at the least expensive end of the hierarchy. The storage capacity is measured in kilobytes (Kbyte or 1000 bytes), megabytes (Mbyte or 1 million bytes), gigabytes (Gbyte or 1 billion bytes), and even terabytes (1000 Gbytes). Programs reside and execute in DRAM. Generally, large permanent databases reside on secondary storage, and portions of the database are read into and written from buffers in main memory as needed. Now that personal computers and workstations have tens of megabytes of data in DRAM, it is becoming possible to load a large fraction of the database into main memory. In some cases, entire databases can be kept in main memory (with a backup copy on magnetic disk), leading to main memory databases; these are particularly useful in real-time applications that require extremely fast response times. An example is telephone switching applications, which store databases that contain routing and line information in main memory.

Database Management System

Between DRAM and magnetic disk storage, another form of memory, flash memory, is becoming common, particularly because it is nonvolatile. Flash memories are high-density, high performance memories using EEPROM (Electrically Erasable Programmable Read-Only Memory) technology. The advantage of flash memory is the fast access speed; the disadvantage is that an entire block must be erased and written over at a time.

CD-ROM disks store data optically and are read by a laser. CD-ROMs contain prerecorded data that cannot be overwritten. WORM (Write-Once-Read-Many) disks are a form of optical storage used for archiving data; they allow data to be written once and read any number of times without the possibility of erasing. They hold about half a gigabyte of data per disk and last much longer than magnetic disks. Optical juke box memories use an array of CD-ROM platters, which are loaded onto drives on demand. Although optical juke boxes have capacities in the hundreds of gigabytes, their retrieval times are in the hundreds of milliseconds, quite a bit slower than magnetic disks. This type of storage has not become as popular as it was expected to be because of the rapid decrease in cost and increase in capacities of magnetic disks. The DVD (Digital Video Disk) is a recent standard for optical disks allowing four to fifteen gigabytes of storage per disk. Finally, magnetic tapes are used for archiving and backup storage of data. Tape jukeboxes—which contain a bank of tapes that are catalogued and can be automatically loaded onto tape drives—are becoming popular as tertiary storage to hold terabytes of data.

RDBMS Concepts

A **Relational Database management System** (RDBMS) is a database management system based on relational model introduced by E.F Codd. In relational model, data is represented in terms of tuples (rows).

RDBMS is used to manage Relational database. **Relational database** is a collection of organized set of tables from which data can be accessed easily. Relational Database is most commonly used database. It consists of number of tables and each table has its own primary key.

What is Table ?

In Relational database, a **table** is a collection of data elements organised in terms of rows and columns. A table is also considered as convenient representation of **relations**. But a

Database Management System

table can have duplicate tuples while a **relation** cannot have duplicate tuples. Table is the most simplest form of data storage. Below is an example of Employee table.

ID	Name	Age	Salary
1	Adam	34	13000
2	Alex	28	15000
3	Stuart	20	18000
4	Ross	42	19020

What is a Record ?

A single entry in a table is called a **Record** or **Row**. A **Record** in a table represents set of related data. For example, the above **Employee** table has 4 records. Following is an example of single record.

1	Adam	34	13000
---	------	----	-------

What is Field ?

A table consists of several records(row), each record can be broken into several smaller entities known as **Fields**. The above **Employee** table consist of four fields, **ID**, **Name**, **Age** and **Salary**.

Database Management System

What is a Column ?

In **Relational** table, a column is a set of value of a particular type. The term **Attribute** is also used to represent a column. For example, in Employee table, Name is a column that represent names of employee.

Name
Adam
Alex
Stuart
Ross

Database Keys

Keys are very important part of Relational database. They are used to establish and identify relation between tables. They also ensure that each record within a table can be uniquely identified by combination of one or more fields within a table.

Super Key

Super Key is defined as a set of attributes within a table that uniquely identifies each record within a table. Super Key is a superset of Candidate key.

Candidate Key

Candidate keys are defined as the set of fields from which primary key can be selected. It is an attribute or set of attribute that can act as a primary key for a table to uniquely identify each record in that table.

Primary Key

Primary key is a candidate key that is most appropriate to become main key of the table. It is a key that uniquely identify each record in a table.

Primary Key



s_id	S_name	age	course	address

Composite Key

Key that consist of two or more attributes that uniquely identify an entity occurrence is called **Composite key**. But any attribute that makes up the **Composite key** is not a simple key in its own.

Composite Key



cust_id	order_id	sale_detail

Secondary or Alternative key

The candidate key which are not selected for primary key are known as secondary keys or alternative keys

Database Management System

Non-key Attribute

Non-key attributes are attributes other than **candidate key** attributes in a table.

Non-prime Attribute

Non-prime Attributes are attributes other than **Primary attribute**.

Database Schemes and Database Instances

Independent from the database model it is important to differentiate between the description of the database and the database itself. The description of the database is called database scheme or also metadata. The database scheme is defined during the database design process and changes very rarely afterwards.

The actual content of the database, the data, changes often over the years. A database state at a specific time defined through the currently existing content and relationship and their attributes is called a database instance

Analogy

	Real World	Database												
Scheme	 Plan for a Standard-House	<table border="1" style="margin: auto;"> <thead> <tr> <th>P-ID</th> <th>Name</th> <th>Prenome</th> </tr> </thead> <tbody> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> </tbody> </table> Template for a Table	P-ID	Name	Prenome									
P-ID	Name	Prenome												
Instances	 Built Standard-Houses	<table border="1" style="margin: auto;"> <thead> <tr> <th>P-ID</th> <th>Name</th> <th>Prenome</th> </tr> </thead> <tbody> <tr> <td>102356</td> <td>Smith</td> <td>John</td> </tr> <tr> <td>102357</td> <td>Potter</td> <td>Harry</td> </tr> <tr> <td>523646</td> <td>Wood</td> <td>Lucinda</td> </tr> </tbody> </table> Data-filled Tables	P-ID	Name	Prenome	102356	Smith	John	102357	Potter	Harry	523646	Wood	Lucinda
P-ID	Name	Prenome												
102356	Smith	John												
102357	Potter	Harry												
523646	Wood	Lucinda												

Analogy Database Schemes and Building Plans

The following illustration shows that a database scheme could be looked at like a template or building plan for one or several database instances.

Analogy Database Schemes and Building Plans

When designing a database it is differentiated between two levels of abstraction and their respective data schemes, the conceptual and the logical data scheme.

Database Management System

Conceptual Data Scheme:

A conceptual data scheme is a system independent data description. That means that it is independent from the database or computer systems used.

Logical Data Scheme:

A logical data scheme describes the data in a data definition language DDL of a specific database management system.

The conceptual data scheme orients itself exclusively by the database application and therefore by the real world. It does not consider any data technical infrastructure like DBMS or computer systems, which are eventually employed. Entity relationship diagrams and relations are tools for the development of a conceptual scheme.

When designing a database the conceptual data scheme is derived from the logical data scheme (see unit Relational Database Design). This derivation results in a logical data scheme for one specific application and one specific DBMS. A DB-Development System converts then the logical scheme directly into instructions for the DBMS.

Relational Algebra

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows –

- Select
- Project
- Union
- Set different
- Cartesian product
- Rename

Database Management System

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation – $\sigma_p(r)$

Where σ stands for selection predicate and r stands for relation. p is propositional logic formula which may use connectors like **and**, **or**, and **not**. These terms may use relational operators like – =, \neq , \geq , $<$, $>$, \leq .

For example –

$\sigma_{\text{subject} = \text{"database"}}(\text{Books})$

Output – Selects tuples from books where subject is 'database'.

$\sigma_{\text{subject} = \text{"database"} \text{ and } \text{price} = \text{"450"}}(\text{Books})$

Output – Selects tuples from books where subject is 'database' and 'price' is 450.

$\sigma_{\text{subject} = \text{"database"} \text{ and } \text{price} = \text{"450"} \text{ or } \text{year} > \text{"2010"}}(\text{Books})$

Output – Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

Project Operation (Π)

It projects column(s) that satisfy a given predicate.

Notation – $\Pi_{A_1, A_2, A_n}(r)$

Where A_1, A_2, A_n are attribute names of relation r .

Duplicate rows are automatically eliminated, as relation is a set.

For example –

$\Pi_{\text{subject, author}}(\text{Books})$

Selects and projects columns named as subject and author from the relation Books.

Database Management System

Union Operation (\cup)

It performs binary union between two given relations and is defined as –

$$r \cup s = \{ t \mid t \in r \text{ or } t \in s \}$$

Notion – $r \cup s$

Where r and s are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold –

- r , and s must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

$$\Pi_{\text{author}}(\text{Books}) \cup \Pi_{\text{author}}(\text{Articles})$$

Output – Projects the names of the authors who have either written a book or an article or both.

Set Difference ($-$)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation – $r - s$

Finds all the tuples that are present in r but not in s .

$$\Pi_{\text{author}}(\text{Books}) - \Pi_{\text{author}}(\text{Articles})$$

Output – Provides the name of authors who have written books but not articles.

Cartesian Product (\times)

Combines information of two different relations into one.

Notation – $r \times s$

Where r and s are relations and their output will be defined as –

$r \times s = \{ q \ t \mid q \in r \text{ and } t \in s \}$

$\Pi_{\text{author} = \text{'tutorialspoint'}}(\text{Books} \times \text{Articles})$

Output – Yields a relation, which shows all the books and articles written by tutorialspoint.

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** ρ .

Notation – $\rho_x(E)$

Where the result of expression **E** is saved with name of **x**.

Additional operations are –

- Set intersection
- Assignment
- Natural join

Relational Calculus

In contrast to Relational Algebra, Relational Calculus is a non-procedural query language, that is, it tells what to do but never explains how to do it.

Relational calculus exists in two forms –

Tuple Relational Calculus (TRC)

Filtering variable ranges over tuples

Notation – $\{T \mid \text{Condition}\}$

Returns all tuples **T** that satisfies a condition.

For example –

Database Management System

$\{ T.name \mid Author(T) \text{ AND } T.article = 'database' \}$

Output – Returns tuples with 'name' from Author who has written article on 'database'.

TRC can be quantified. We can use Existential (\exists) and Universal Quantifiers (\forall).

For example –

$\{ R \mid \exists T \in Authors(T.article='database' \text{ AND } R.name=T.name) \}$

Output – The above query will yield the same result as the previous one.

Domain Relational Calculus (DRC)

In DRC, the filtering variable uses the domain of attributes instead of entire tuple values (as done in TRC, mentioned above).

Notation –

$\{ a_1, a_2, a_3, \dots, a_n \mid P(a_1, a_2, a_3, \dots, a_n) \}$

Where a_1, a_2 are attributes and **P** stands for formulae built by inner attributes.

For example –

$\{ \langle article, page, subject \rangle \mid \in \text{TutorialsPoint} \wedge subject = 'database' \}$

Output – Yields Article, Page, and Subject from the relation TutorialsPoint, where subject is database.

Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators.

The expression power of Tuple Relation Calculus and Domain Relation Calculus is equivalent to Relational Algebra.

Indexing

Database Management System

Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing in database systems is similar to what we see in books.

Indexing is defined based on its indexing attributes. Indexing can be of the following types –

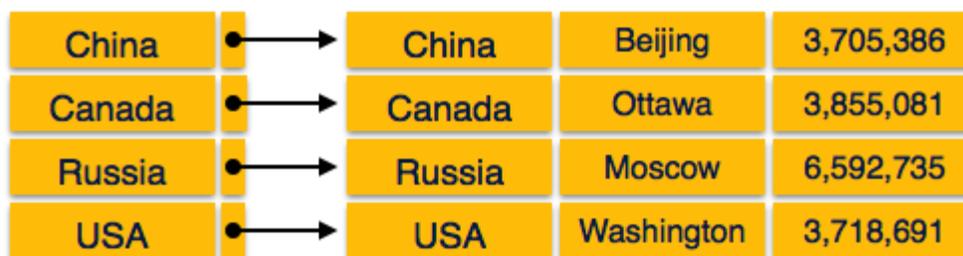
- **Primary Index** – Primary index is defined on an ordered data file. The data file is ordered on a **key field**. The key field is generally the primary key of the relation.
- **Secondary Index** – Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
- **Clustering Index** – Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

Ordered Indexing is of two types –

- Dense Index
- Sparse Index

Dense Index

In dense index, there is an index record for every search key value in the database. This makes searching faster but requires more space to store index records itself. Index records contain search key value and a pointer to the actual record on the disk.



Sparse Index

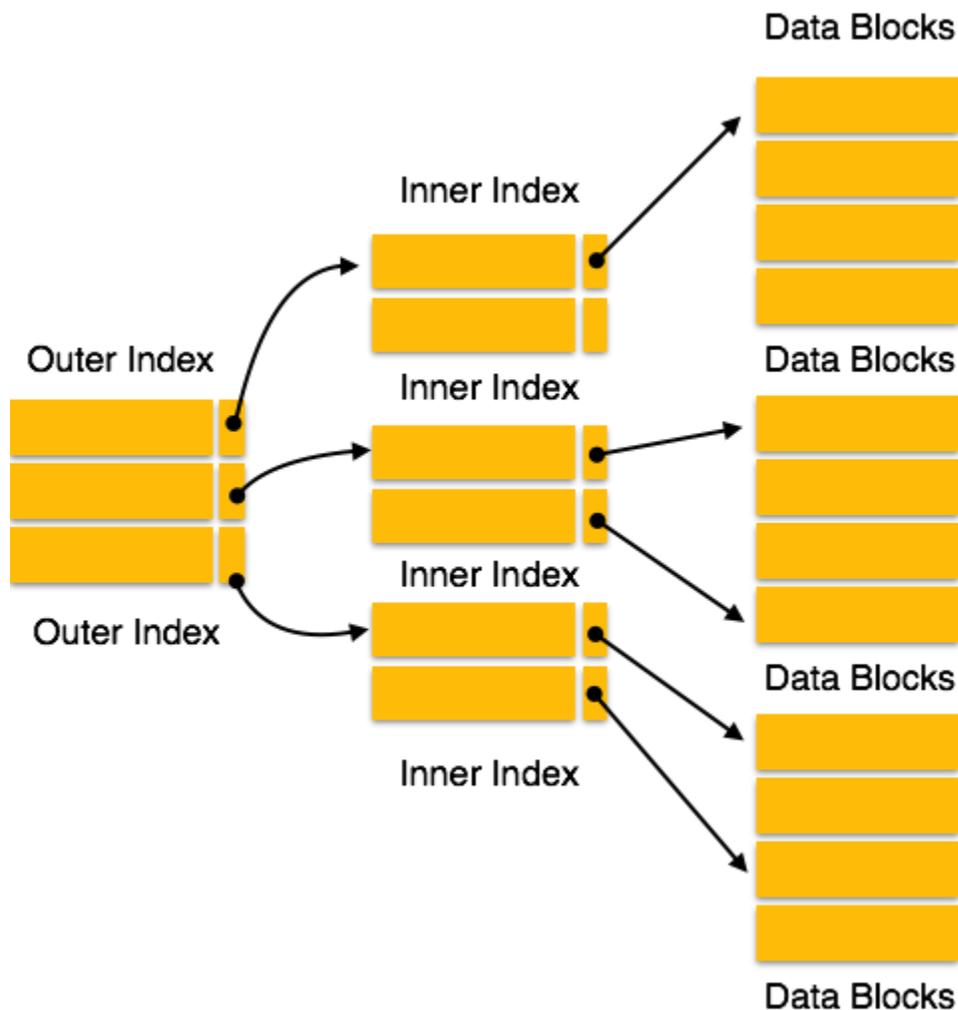
In sparse index, index records are not created for every search key. An index record here contains a search key and an actual pointer to the data on the disk. To search a record, we first proceed by index record and reach at the actual location of the data. If the data we are looking for is not where we directly reach by following the index, then the system starts sequential search until the desired data is found.

Database Management System

China	→	China	Beijing	3,705,386
Russia	→	Canada	Ottawa	3,855,081
USA	→	Russia	Moscow	6,592,735
	→	USA	Washington	3,718,691

Multilevel Index

Index records comprise search-key values and data pointers. Multilevel index is stored on the disk along with the actual database files. As the size of the database grows, so does the size of the indices. There is an immense need to keep the index records in the main memory so as to speed up the search operations. If single-level index is used, then a large size index cannot be kept in memory which leads to multiple disk accesses.



Database Management System

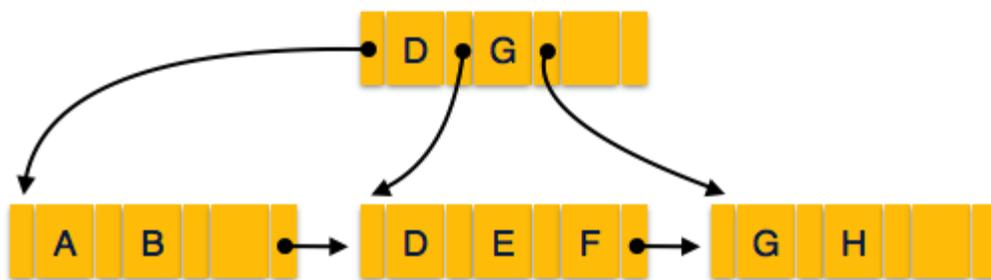
Multi-level Index helps in breaking down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block, which can easily be accommodated anywhere in the main memory.

B+ Tree

A B+ tree is a balanced binary search tree that follows a multi-level index format. The leaf nodes of a B+ tree denote actual data pointers. B+ tree ensures that all leaf nodes remain at the same height, thus balanced. Additionally, the leaf nodes are linked using a link list; therefore, a B+ tree can support random access as well as sequential access.

Structure of B+ Tree

Every leaf node is at equal distance from the root node. A B+ tree is of the order n where n is fixed for every B+ tree.



Internal nodes –

- Internal (non-leaf) nodes contain at least $\lceil n/2 \rceil$ pointers, except the root node.
- At most, an internal node can contain n pointers.

Leaf nodes –

- Leaf nodes contain at least $\lceil n/2 \rceil$ record pointers and $\lceil n/2 \rceil$ key values.
- At most, a leaf node can contain n record pointers and n key values.
- Every leaf node contains one block pointer **P** to point to next leaf node and forms a linked list.

B+ Tree Insertion

- B+ trees are filled from bottom and each entry is done at the leaf node.
- If a leaf node overflows –
 - Split node into two parts.

Database Management System

- Partition at $i = \lfloor (m+1)/2 \rfloor$.
- First i entries are stored in one node.
- Rest of the entries ($i+1$ onwards) are moved to a new node.
- i^{th} key is duplicated at the parent of the leaf.
- If a non-leaf node overflows –
 - Split node into two parts.
 - Partition the node at $i = \lfloor (m+1)/2 \rfloor$.
 - Entries up to i are kept in one node.
 - Rest of the entries are moved to a new node.

B+ Tree Deletion

- B+ tree entries are deleted at the leaf nodes.
- The target entry is searched and deleted.
 - If it is an internal node, delete and replace with the entry from the left position.
- After deletion, underflow is tested,
 - If underflow occurs, distribute the entries from the nodes left to it.
- If distribution is not possible from left, then
 - Distribute from the nodes right to it.
- If distribution is not possible from left or from right, then
 - Merge the node with left and right to it.

UNIT III

Normalization of Database

Database Normalisation is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anamolies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

Problem Without Normalization

Without Normalization, it becomes difficult to handle and update the database, without facing data loss. Insertion, Updation and Deletion Anamolies are very frequent if Database is not Normalized. To understand these anomalies let us take an example of **Student** table.

S_id	S_Name	S_Address	Subject_opted
401	Adam	Noida	Bio
402	Alex	Panipat	Maths
403	Stuart	Jammu	Maths
404	Adam	Noida	Physics

- **Updation Anamoly** : To update address of a student who occurs twice or more than twice in a table, we will have to update **S_Address** column in all the rows, else data will become inconsistent.

Database Management System

- **Insertion Anamoly** : Suppose for a new admission, we have a Student id(S_id), name and address of a student but if student has not opted for any subjects yet then we have to insert **NULL** there, leading to Insertion Anamoly.
- **Deletion Anamoly** : If (S_id) 401 has only one subject and temporarily he drops it, when we delete that row, entire student record will be deleted along with it.

Normalization Rule

Normalization rule are divided into following normal form.

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF

First Normal Form (1NF)

As per First Normal Form, no two Rows of data must contain repeating group of information i.e each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

The **Primary key** is usually a single column, but sometimes more than one column can be combined to create a single primary key. For example consider a table which is not in First normal form

Student Table :

Student	Age	Subject
Adam	15	Biology, Maths
Alex	14	Maths

Database Management System

Stuart	17	Maths
--------	----	-------

In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be :

Student	Age	Subject
Adam	15	Biology
Adam	15	Maths
Alex	14	Maths
Stuart	17	Maths

Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

Second Normal Form (2NF)

As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails **Second normal form**.

In example of First Normal Form there are two rows for Adam, to include multiple subjects that he has opted for. While this is searchable, and follows First normal form, it is an inefficient use of space. Also in the above Table in First Normal Form, while the candidate key is {**Student, Subject**}, **Age** of Student only depends on Student column, which is incorrect as per Second Normal Form. To achieve second normal form, it would be helpful to split out the subjects into an independent table, and match them up using the student names as foreign keys.

New Student Table following 2NF will be :

Database Management System

Student	Age
Adam	15
Alex	14
Stuart	17

In Student Table the candidate key will be **Student** column, because all other column i.e **Age** is dependent on it.

New Subject Table introduced for 2NF will be :

Student	Subject
Adam	Biology
Adam	Maths
Alex	Maths
Stuart	Maths

In Subject Table the candidate key will be {**Student, Subject**} column. Now, both the above tables qualifies for Second Normal Form and will never suffer from Update Anomalies. Although there are a few complex cases in which table in Second Normal Form suffers Update Anomalies, and to handle those scenarios Third Normal Form is there.

Third Normal Form (3NF)

Third Normal form applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute. So this *transitive functional dependency* should be removed from the table and also the table must be in **Second Normal form**. For example, consider a table with following fields.

Student_Detail Table :

Student_id	Student_name	DOB	Street	city	State	Zip
------------	--------------	-----	--------	------	-------	-----

Database Management System

In this table Student_id is Primary key, but street, city and state depends upon Zip. The dependency between zip and other fields is called **transitive dependency**. Hence to apply 3NF, we need to move the street, city and state to new table, with **Zip** as primary key.

New Student_Detail Table :

Student_id	Student_name	DOB	Zip
------------	--------------	-----	-----

Address Table :

Zip	Street	city	state
-----	--------	------	-------

The advantage of removing transitive dependency is,

- Amount of data duplication is reduced.
- Data integrity achieved.

Boyce and Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form
- and, for each functional dependency (X -> Y), X should be a super Key.

Database Management System

Consider the following relationship : **R (A,B,C,D)**

and following dependencies :

A -> BCD

BC -> AD

D -> B

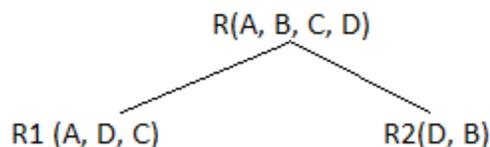
Above relationship is already in 3rd NF. Keys are **A** and **BC**.

Hence, in the functional dependency, **A -> BCD**, A is the super key.

in second relation, **BC -> AD**, BC is also a key.

but in, **D -> B**, D is not a key.

Hence we can break our relationship R into two relationships **R1** and **R2**.



Breaking, table into two tables, one with A, D and C while the other with D and B.

INTEGRITY CONSTRAINTS

To preserve the consistency and correctness of its stored data, a relational DBMS typically imposes one or more data integrity constraints. These constraints restrict the data values that can be inserted into the database or created by a database update. Several different types of data integrity constraints are commonly found in relational databases, including:

Required data: Some columns in a database must contain a valid data value in every row; they are not allowed to contain missing or NULL values. In the sample database, every order must have an associated customer who placed the order. The DBMS can be asked to prevent NULL values in this column.

Validity checking: Every column in a database has a domain, a set of data values that are legal for that column. The DBMS can be asked to prevent other data values in these columns.

Entity integrity: The primary key of a table must contain a unique value in each row, which is different from the values in all other rows. Duplicate values are illegal, because they wouldn't allow the database to distinguish one entity from another. The DBMS can be asked to enforce this unique values constraint.

Database Management System

Referential integrity: A foreign key in a relational database links each row in the child table containing the foreign key to the row of the parent table containing the matching primary key value. The DBMS can be asked to enforce this foreign key/primary key constraint.

Other data relationships: The real-world situation modeled by a database will often have additional constraints that govern the legal data values that may appear in the database. The DBMS can be asked to check modifications to the tables to make sure that their values are constrained in this way.

Business rules: Updates to a database may be constrained by business rules governing the real-world transactions that are represented by the updates.

Consistency: Many real-world transactions cause multiple updates to a database. The DBMS can be asked to enforce this type of consistency rule or to support applications that implement such rules.

Data Analysis

Analysis of data is a process of inspecting, cleaning, transforming, and modeling **data** with the goal of discovering useful **information**, suggesting conclusions, and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains.

Data mining is a particular data analysis technique that focuses on modeling and knowledge discovery for predictive rather than purely descriptive purposes. **Business intelligence** covers data analysis that relies heavily on aggregation, focusing on business information. In **statistical applications**, some people divide data analysis into **descriptive statistics**, **exploratory data analysis (EDA)**, and **confirmatory data analysis (CDA)**. EDA focuses on discovering new features in the data and CDA on confirming or falsifying existing hypotheses. **Predictive analytics** focuses on application of statistical models for predictive forecasting or classification, while **text analytics** applies statistical, linguistic, and structural techniques to extract and classify information from textual sources, a species of **unstructured data**. All are varieties of data analysis.

Data integration is a precursor to data analysis, and data analysis is closely linked to **data visualization** and data dissemination. The term *data analysis* is sometimes used as a synonym for **data modeling**.

Database Management System

Business Analysis Technique

Data analysis is a **process** for obtaining raw data and converting it into information useful for decision-making by users. Data is collected and analyzed to answer questions, test hypotheses or disprove theories

Data requirements

The data necessary as inputs to the analysis are specified based upon the requirements of those directing the analysis or customers who will use the finished product of the analysis. The general type of entity upon which the data will be collected is referred to as an experimental unit (e.g., a person or population of people). Specific variables regarding a population (e.g., age and income) may be specified and obtained. Data may be numerical or categorical (i.e., a text label for numbers).¹

Data collection

Data is collected from a variety of sources. The requirements may be communicated by analysts to custodians of the data, such as information technology personnel within an organization. The data may also be collected from sensors in the environment, such as traffic cameras, satellites, recording devices, etc. It may also be obtained through interviews, downloads from online sources, or reading documentation.

Data processing

Data initially obtained must be processed or organized for analysis. For instance, this may involve placing data into rows and columns in a table format for further analysis, such as within a spreadsheet or statistical software

Data cleaning

Once processed and organized, the data may be incomplete, contain duplicates, or contain errors. The need for data cleaning will arise from problems in the way that data is entered and stored. Data cleaning is the process of preventing and correcting these errors. Common tasks include record matching, deduplication, and column segmentation. Such data problems can also be identified through a variety of analytical techniques. For example, with financial information, the totals for particular variables may be compared against separately published numbers believed to be reliable. Unusual amounts above or below

Database Management System

pre-determined thresholds may also be reviewed. There are several types of data cleaning that depend on the type of data. Quantitative data methods for outlier detection can be used to get rid of likely incorrectly entered data. Textual data spellcheckers can be used to lessen the amount of mistyped words, but it is harder to tell if the words themselves are correct

Exploratory data analysis

Once the data is cleaned, it can be analyzed. Analysts may apply a variety of techniques referred to as **exploratory data analysis** to begin understanding the messages contained in the data. The process of exploration may result in additional data cleaning or additional requests for data, so these activities may be iterative in nature. **Descriptive statistics** such as the average or median may be generated to help understand the data. **Data visualization** may also be used to examine the data in graphical format, to obtain additional insight regarding the messages within the data.^l

Modeling and algorithms

Mathematical formulas or models called **algorithms** may be applied to the data to identify relationships among the variables, such as **correlation** or **causation**. In general terms, models may be developed to evaluate a particular variable in the data based on other variable(s) in the data, with some residual error depending on model accuracy (i.e., $\text{Data} = \text{Model} + \text{Error}$).

Inferential statistics includes techniques to measure relationships between particular variables. For example, **regression analysis** may be used to model whether a change in advertising (independent variable X) explains the variation in sales (dependent variable Y). In mathematical terms, Y (sales) is a function of X (advertising). It may be described as $Y = aX + b + \text{error}$, where the model is designed such that a and b minimize the error when the model predicts Y for a given range of values of X. Analysts may attempt to build models that are descriptive of the data to simplify analysis and communicate results

Database Management System

Data product

A data product is a computer application that takes data inputs and generates outputs, feeding them back into the environment. It may be based on a model or algorithm. An example is an application that analyzes data about customer purchasing history and recommends other purchases the customer might enjoy

Communication

Once the data is analyzed, it may be reported in many formats to the users of the analysis to support their requirements. The users may have feedback, which results in additional analysis. As such, much of the analytical cycle is iterative.^[3]

When determining how to communicate the results, the analyst may consider [data visualization](#) techniques to help clearly and efficiently communicate the message to the audience. Data visualization uses [information displays](#) such as tables and charts to help communicate key messages contained in the data. Tables are helpful to a user who might lookup specific numbers, while charts (e.g., bar charts or line charts) may help explain the quantitative messages contained in the data.

