Intelligent Data Management with SQL Server

Intelligent Data Management with SQL Server

APTECH LIMITED
Contact E-mail: ov-support@onlinevarsity.com

Edition 1 - 2020

Onlinevarsity

ANYTIME | ANYWHERE

PREFACE

SQL Server is a client-server based Relational Database Management System (RDBMS) from Microsoft. It provides an enterprise-level data management platform for an organization. SQL Server includes numerous features and tools that make it an outstanding database and data analysis platform. It is also targeted for large-scale Online Transaction Processing (OLTP), data warehousing, and e-commerce applications. One of the key features of SQL Server is that it is now available on the cloud too.

The book begins with an introduction to RDBMS concepts and moves on to introduce SQL Server 2019. The book then covers various SQL Server topics such as data types, usage of Transact-SQL, and database objects such as indexes, stored procedures, functions, and so on. The book also introduces Azure SQL and cloud databases. The book describes transactions, programming elements with Transact-SQL, and finally troubleshooting errors with error handling techniques.

The book also explores SQL Server 2019 new features and enhancements. These include features such as Big Data clusters, PolyBase, Query Store, Stretch Database, and In-Memory enhancements. Besides these, you will also learn about the improved Performance Tools and Transact-SQL enhancements.

The knowledge and information in this book is the result of the concentrated effort of the Design Team, which is continuously striving to bring to you the latest, the best and the most relevant subject matter in Information Technology. As a part of Aptech's quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends and learner requirements.

Onlinevarsity

Onlinevarsity App

for Android devices

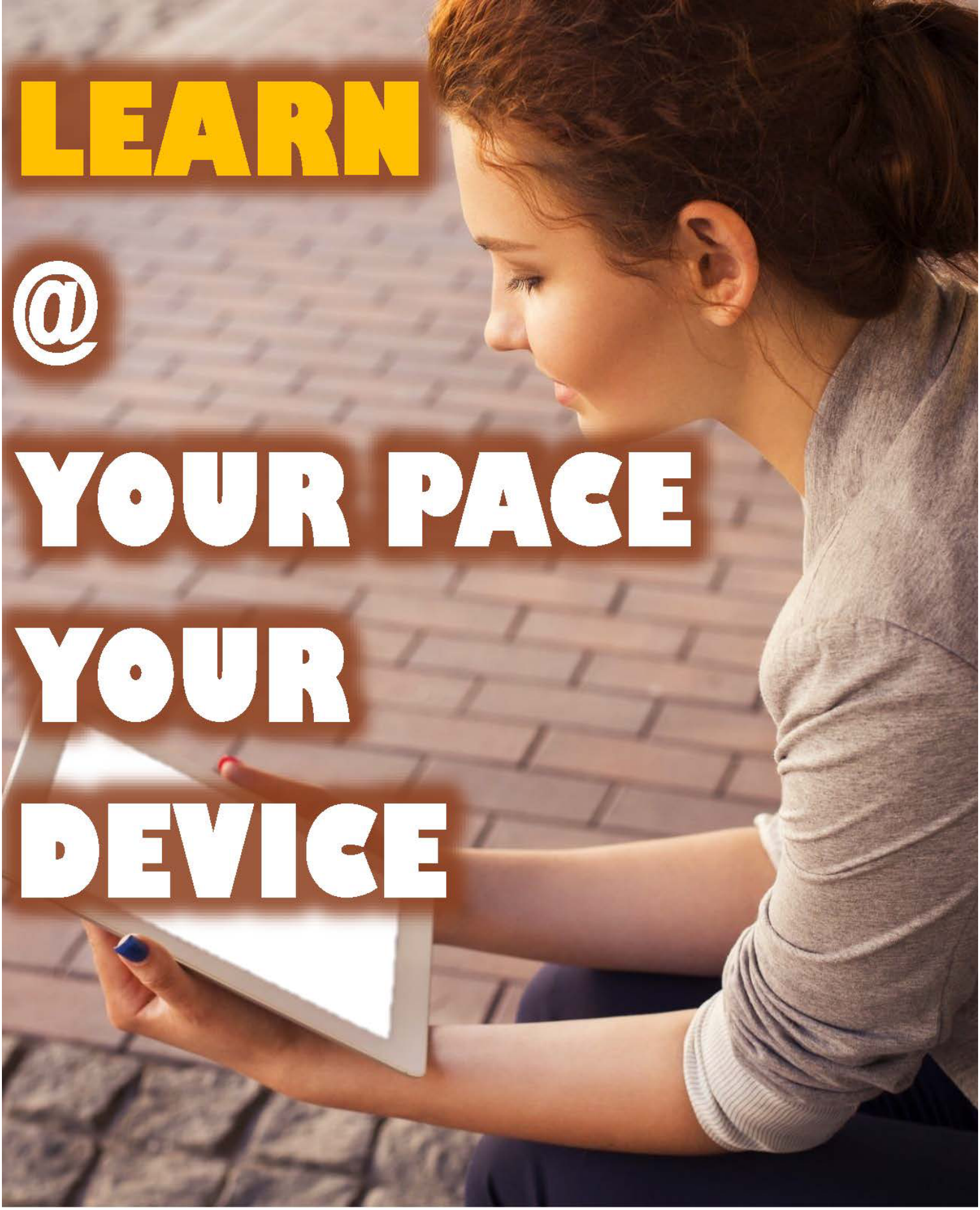
Download from Google Play Store

Onlinevarsity

INDUSTRY BEST PRACTICES

SYNC WITH THE INDUSTRY

Contents



Onlinevarsity

Session - 1
RDBMS Concepts

Welcome to the Session, **RDBMS Concepts**.

This session deals with the concepts related to databases and database management systems, explores various database models, and introduces the concept of an RDBMS.

In this session, you will learn to:

- Explain the concept of data and database
- Describe the approaches to data management
- Define a Database Management System (DBMS) and list its benefits
- Explain the different database models
- Define and explain RDBMS
- Describe entities and tables and list the characteristics of tables
- List the differences between a DBMS and an RDBMS

1.1 Introduction

Organizations often maintain large amounts of data, which are generated as a result of day-to-day operations. A database is an organized form of such data. It may consist of one or more related data items called records. Think of a database as a data collection to which different questions can be asked. For example, 'What are the phone numbers and addresses of the five nearest post offices?' or 'Do we have any books in our library that deal with health food? If so, on which shelves are they located?' or 'Show me the personnel records and sales figures of five best-performing sales people for the current quarter, but their address details are not required to be shown'.

1.2 Data and Database

Data means information and it is the most important component in any work that is done. In the day-to-day activity, either existing data is used or more data is generated. When this data is gathered and analyzed, it yields information. It can be any information such as information about the vehicle, sports, airways, and so on. For example, a sport magazine journalist (who is a soccer enthusiast) gathers the score (data) of Germany's performance in 10 world cup matches. These scores constitute data. When this data is compared with the data of 10 world cup matches played by Brazil, the journalist can obtain information as to which country has a team that plays better soccer.

Information helps to foresee and plan events. Intelligent interpretation of data yields information. In the world of business, to be able to predict an event and plan for it could save time and money. Consider an example, where a car manufacturing company is planning its annual purchase of certain parts of the car, which has to be imported since it is not locally available. If data of the purchase of these parts for the last

five years is available, the company heads can actually compile information about the total amount of parts imported. Based on these findings, a production plan can be prepared. Therefore, information is a key-planning factor.

A database is a collection of data. Some like to think of a database as an organized mechanism that has the capability of storing information. This information can be retrieved by the user in an effective and efficient manner.

A phone book is a database. The data contained consists of individuals' names, addresses, and telephone numbers. These listings are in alphabetical order or indexed. This allows the user to reference a particular local resident with ease. Ultimately, this data is stored in a database somewhere on a computer. As people move to different cities or states, entries may have to be added or removed from the phone book. Likewise, entries will have to be modified for people changing names, addresses, or telephone numbers, and so on.

Figure 1.1 illustrates the concept of a database.

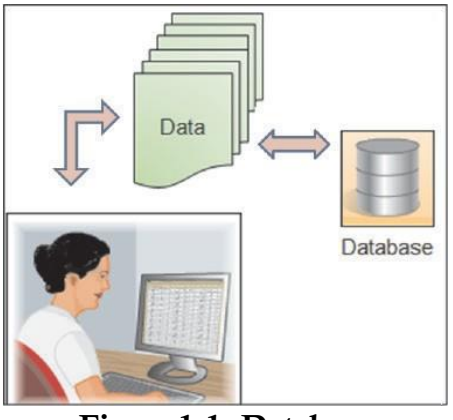


Figure 1.1: Database

Thus, a database is a collection of data that is organized such that its contents can be easily accessed, managed, and updated.

1.3 Data Management

Data management deals with managing large amount of information, which involves both the storage of information and the provision of mechanisms for the manipulation of information. In addition, the system should also provide the safety of the information stored under various circumstances, such as multiple user access and so on.

The two different approaches of managing data are file-based systems and database systems.

1.3.1 File-based Systems

Storage of large amounts of data has always been a matter of huge concern. In early days, file-based systems were used. In this system, data was stored in discrete files and a collection of such files was stored on a computer. These could be accessed by a computer operator. Files of archived data were called tables because they looked like tables used in traditional file keeping. Rows in the table were called records and columns were called fields.
Conventionally, before the database systems evolved, data in software systems was stored in flat files.

An example of the file-based system is illustrated in table 1.1.

First Name	Last Name	Address	Phone
Eric	David	ericd@eff.org	213-456-0987
Selena	Sol	selena@eff.org	987-765-4321
Jordan	Lim	nadroj@otherdomain.com	222-3456-123

Table 1.1: File-based System

- **Disadvantages of File-based Systems**

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

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

- **Unanticipated queries**
  In a file-based system, handling sudden/ad-hoc queries can be difficult, since it requires changes in the existing programs. For example, the bank officer must generate a list of all the customers who have an account balance of $20,000 or more. The bank officer has two choices: either obtain the list of all customers and have the required information extracted manually, or hire a system programmer to design the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and several days later, the officer must trim that list to include only those customers who have opened their account one year ago. As the program to generate such a list does not exist, it leads to a difficulty in accessing the data.

- **Data isolation**
  Data are scattered in various files, and files may be in a different format. Though data used by different programs in the application may be related, they reside as isolated data files.

- **Concurrent access anomalies**
  In large multi-user systems, the same file or record may have to be accessed by multiple users simultaneously. Handling this in a file-based system is difficult.

- **Security problems**
  In data-intensive applications, security of data is a major concern. Users should be given access only to required data and not to the whole database.

For example, in a banking system, payroll personnel may have to view only that part of the database that has information about various bank employees. They do not require access to information about customer accounts. Since application programs are added to the system in an ad-hoc manner, it is difficult to enforce such security constraints. In a file-based system, this can be handled only by additional programming in each application.

- **Integrity problems**
  In any application, there will be certain data integrity rules, which must be maintained. These could be in the form of certain conditions/constraints on the elements of the data records. In the savings bank application, one such integrity rule could be 'Customer ID, which is the unique identifier for a customer record, should not be empty'. There can be several such integrity rules. In a file-based system, all these rules must be explicitly programmed in the application program.

Though all these are common issues of concern to any data-intensive application, each application had to handle all these problems on its own. The application programmer must not only be concerned about implementing the application business rules but also, about handling these common issues.

1.3.2 Database Systems

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

Databases are used to store data in an efficient and organized manner. A database allows quick and easy management of data. For example, a company may maintain details of its employees in various databases. At any point of time, data can be retrieved from the database, new data can be added into the databases and data can be searched based on some criteria in these databases.

Data storage can be achieved even using simple manual files. For instance, a college has to maintain information about teachers, students, subjects, and examinations.

Details of the teachers can be maintained in a Staff Register and details of the students could be entered in a Student Register and so forth. However, data stored in this form is not permanent. Records in such manual files can only be maintained for a few months or few years. The registers or files are bulky, consume a lot of space, and hence, cannot be kept for many years.

Instead of this, if the same data was stored using database system, it could be more permanent and long-lasting.

➢ **Advantages of database systems**

Information or data can be permanently stored in the form of computerized databases. A database system is advantageous because it provides a centralized control over the data.

Some of the benefits of using such a centralized database system are as follows:

- **The amount of redundancy in the stored data can be reduced**
  In an organization, several departments often store the same data. Maintaining a centralized database helps the same data to be accessed by many departments. Thus, duplication of data or 'data

redundancy' can be reduced.

- **No more inconsistencies in data**
  When data is duplicated across several departments, any modifications to the data have to be reflected across all departments. Sometimes, this can lead to inconsistency in the data. As there is a central database, it is possible for one person to take up the task of updating the data on a regular basis. Consider that Mr. Larry Finner, an employee of an organization is promoted as a Senior Manager from Manager.

  In such a case, there is just one record in the database that must be changed. As a result, data inconsistency is reduced.

- **The stored data can be shared**
  A central database can be located on a server, which can be shared by several users. In this way, all users can access the common and updated information all the time.

- **Standards can be set and followed**
  A central control ensures that a certain standard in the representation of data can be set and followed. For example, the name of an employee has to be represented as 'Mr. Larry Finner'. This representation can be broken down into the following components:
  - A title (Mr.)
  - First name (Larry)
  - Last name (Finner)

  It is certain that all the names stored in the database will follow the same format if the standards are set in this manner.

- **Data Integrity can be maintained**
  Data integrity refers to the accuracy of data in the database. For example, when an employee resigns and leaves the organization, consider that the Accounts department has updated its database and the HR department has not updated its records. The data in the company's records is hence, inaccurate.

  Centralized control of the database helps in avoiding these errors. It is certain that if a record is deleted from one table, its linked record in the other table is also deleted.

- **Security of data can be implemented**
  In a central database system, the privilege of modifying the database is not given to everyone. This right is given only to one person who has full control over the database. This person is called as Database Administrator or DBA. The DBA can implement security by placing restrictions on the data. Based on the permissions granted to them, the users can add, modify, or query data.

1.4 Database Management System (DBMS)

A DBMS can be defined as a collection of related records and a set of programs that access and manipulate these records. A DBMS enables the user to enter, store, and manage data. The main problem with the earlier DBMS packages was that the data was stored in the flat file format. So, the information about different objects was maintained separately in different physical files. Hence, the relations between these objects, if any, had to be maintained in a separate physical file. Thus, a single package would consist of too many files and vast functionalities to integrate them into a single system.

A solution to these problems came in the form of a centralized database system. In a centralized database system, the database is stored in the central location. Everybody can have access to the data stored in a central location from their machine. For example, a large central database system would contain all the data pertaining to the employees. The Accounts and the HR department would access the data required using suitable programs. These programs or the entire application would reside on individual computer terminals.

A Database is a collection of interrelated data, and a DBMS is a set of programs used to add or modify this data. Thus, a DBMS is a set of software programs that allow databases to be defined, constructed, and manipulated.

A DBMS provides an environment that is both convenient and efficient to use when there is a large volume of data and many transactions to be processed. Different categories of DBMS can be used, ranging from small systems that run on personal computers to huge systems that run on mainframes.

Examples of database applications include the following:

- Computerized library systems
- Automated teller machines
- Flight reservation systems
- Computerized parts inventory systems

From a technical standpoint, DBMS products can differ widely. Different DBMS support different query languages, although there is a semi-standardized query language called Structured Query Language (SQL).

Sophisticated languages for managing database systems are called Fourth Generation Language (4GLs). The information from a database can be presented in a variety of formats. Most DBMS include a report writer program that enables the user to output data in the form of a report. Many DBMSs also include a graphics component that enables the user to output information in the form of graphs and charts.

It is not necessary to use general-purpose DBMS for implementing a computerized database. The users can write their own set of programs to create and maintain the database, in effect creating their own special-purpose DBMS software. The database and the software together are called a database system.

The end user accesses the database system through application programs and queries. The DBMS software enables the user to process the queries and programs placed by the end user. The software accesses the data from the database.

Figure 1.2 illustrates a database system.

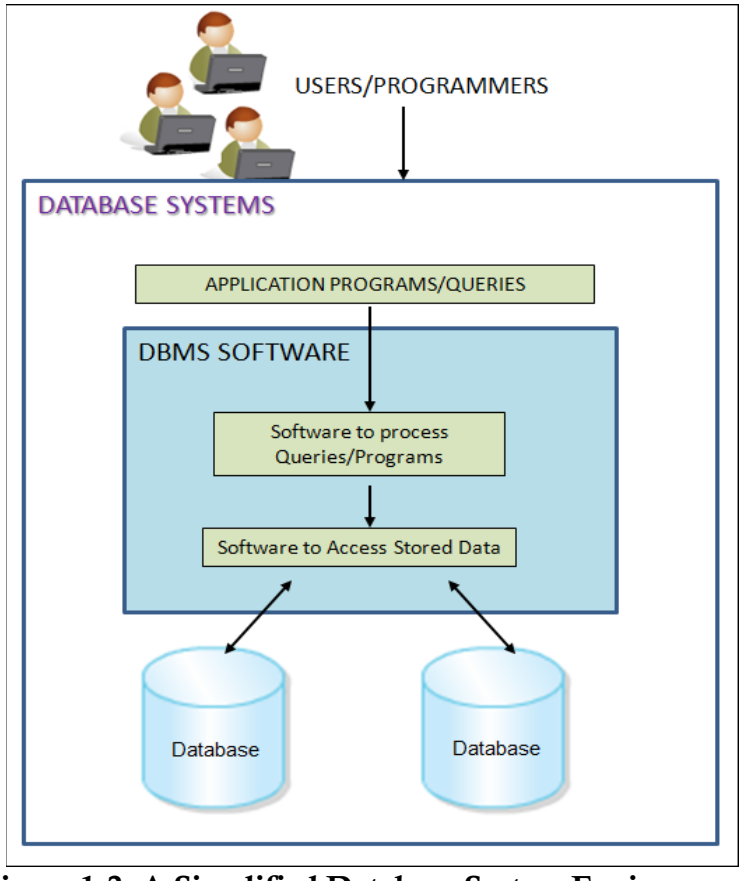


Figure 1.2: **A Simplified Database System Environment**

1.4.1 Benefits of DBMS

A DBMS is responsible for processing data and converting it into information. For this purpose, the database has to be manipulated, which includes querying the database to retrieve specific data, updating the database, and finally, generating reports.

These reports are the source of information, which is, processed data. A DBMS is also responsible for data security and integrity.

The benefits of a typical DBMS are as follows:

- **Data storage**
  The programs required for physically storing data, handled by a DBMS, is done by creating complex data structures, and the process is called data storage management.

- **Data definition**
  A DBMS provides functions to define the structure of the data in the application. These include defining and modifying the record structure, the type and size of fields, and various constraints/conditions to be satisfied by the data in each field.
- **Data manipulation**
  Once the data structure is defined, data must be inserted, modified, or deleted. The functions, which perform these operations, are also part of a DBMS. These functions can handle planned and unplanned data manipulation requirements. Planned queries are those, which form part of the application.

Unplanned queries are ad-hoc queries, which are performed as and when required.

- **Data security and integrity**
  Data security is of utmost importance when there are multiple users accessing the database. It is required for keeping a check over data access by users. The security rules specify, which user has access to the database, what data elements the user has access to, and the data operations that the user can perform.

  Data in the database should contain as few errors as possible. For example, the employee number for adding a new employee should not be left blank. Telephone number should contain only numbers. Such checks are taken care of by a DBMS.

  Thus, the DBMS contains functions, which handle the security and integrity of data in the application. These can be easily invoked by the application and hence, the application programmer does not have to code these functions in the programs.

- **Data recovery and concurrency**
  Recovery of data after a system failure and concurrent access of records by multiple users are also handled by a DBMS.

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

- **Multi-user access control**
  At any point of time, more than one user can access the same data. A DBMS takes care of the sharing of data among multiple users, and maintains data integrity.

- **Database access languages and Application Programming Interfaces (APIs)**
  The query language of a DBMS implements data access. SQL is the most commonly used query language. A query language is a non-procedural language, where the user must request what is required and does not have to specify how it is to be done. Some procedural languages such as C, Visual Basic, Pascal, and others provide data access to programmers.

1.5 Database Models

Databases can be differentiated based on functions and model of the data. A data model describes a container for storing data, and the process of storing and retrieving data from that container. The analysis and design of data models has been the basis of the evolution of databases. Each model has evolved from the previous one.

1.5.1 Flat File Data Model

In this model, the database consists of only one table or file. This model is used for simple databases - for example, to store the roll numbers, names, subjects, and marks of a group of students. This model cannot handle very complex data. It can cause redundancy when data is repeated more than once.
Table 1.2 depicts the structure of a flat file database.

Roll Number	FirstName	LastName	Subject	Marks
45	Jones	Bill	Maths	84
45	Jones	Bill	Science	75
50	Mary	Mathew	Science	80

Table 1.2: **Structure of Flat File Data Model**

1.5.2 Hierarchical Data Model

In the Hierarchical Model, different records are inter-related through hierarchical or tree-like structures. In this model, relationships are thought of in terms of children and parents. A parent record can have several children, but a child can have only one parent. To find data stored in this model, the user must know the structure of the tree.

The Windows Registry is an example of a hierarchical database storing configuration settings and options on Microsoft Windows operating systems.

Figure 1.3 illustrates an example of a hierarchical representation.

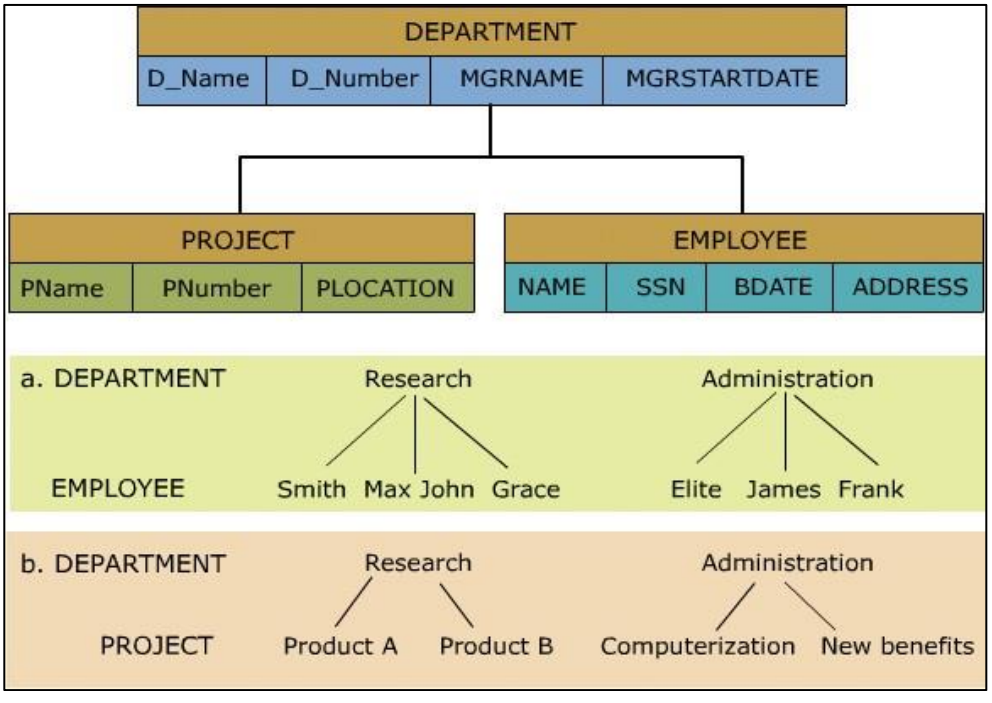


Figure 1.3: Example of a Hierarchical Model

Within the hierarchical model, Department is perceived as the parent of the segment. The tables, Project and Employee, are children. A path that traces the parent segments beginning from the left, defines the tree. This ordered sequencing of segments tracing the hierarchical structure is called the hierarchical path.

It is clear from the figure that in a single department, there can be many employees and a department can have many projects.

- **Advantages of the hierarchical model**
  The advantages of a hierarchical model are as follows:
  - Data is held in a common database so data sharing becomes easier, and security is provided and enforced by a DBMS.
  - Data independence is provided by a DBMS, which reduces the effort and costs in maintaining the program.

  This model is very efficient when a database contains a large volume of data. For example, a bank's customer account system fits the hierarchical model well because each customer's account is subject to a number of transactions.

1.5.3 Network Data Model

This model is similar to the Hierarchical Data Model. The hierarchical model is actually a subset of the network model. However, instead of using a single-parent tree hierarchy, the network model uses set theory to provide a tree-like hierarchy with the exception that child tables were allowed to have more than one parent.

In the network model, data is stored in sets, instead of the hierarchical tree format. This solves the problem of data redundancy. The set theory of the network model does not use a single-parent tree hierarchy. It allows a child to have more than one parent. Thus, the records are physically linked through linked-lists. Integrated Database Management System (IDMS) from Computer Associates International Inc. and Raima Database Manager (RDM) Server by Raima Inc. are examples of a Network DBMS.

The network model together with the hierarchical data model was a major data model for implementing numerous commercial DBMS. The network model structures and language constructs were defined by Conference on Data Systems Language (CODASYL).

For every database, a definition of the database name, record type for each record, and the components that make up those records is stored. This is called its network schema. A portion of the database as seen by the application's programs that actually produce the desired information from the data contained in the database is called sub-schema. It allows application programs to access the required data from the database.

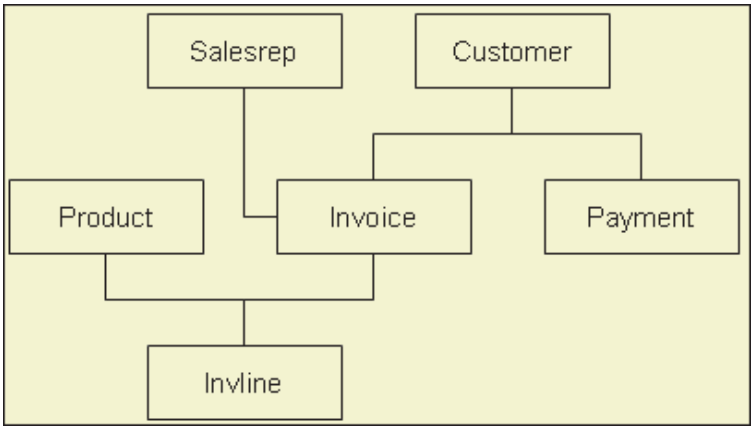


Figure 1.4: Network Model

The network model shown in figure 1.4 illustrates a series of one-to-many relationships, as follows:

1. A sales representative might have written many Invoice tickets, but each Invoice is written by a single Sales representative (Salesrep).
2. A Customer might have made purchases on different occasions. A Customer may have many Invoice tickets, but each Invoice belongs only to a single customer.
3. An Invoice ticket may have many Invoice lines (Invline), but each Invline is found on a single Invoice ticket.
4. A Product may appear in several different Invline, but each Invline contains only a single Product.

Components of the language used with network models are as follows:

Advantages of network model

Advantages of such a structure are specified as follows:

- The relationships are easier to implement in the network database model than in the hierarchical model.
- This model enforces database integrity.
- This model achieves sufficient data independence.

Disadvantages of network model

Disadvantages are specified as follows:

- The databases in this model are difficult to design.
- The programmer has to be very familiar with the internal structures to access the database.
- The model provides a navigational data access environment. Hence, to move from A to E in the sequence A-B-C-D-E, the user has to move through B, C, and D to get to E.

This model is difficult to implement and maintain. Computer programmers, rather than end users, utilize this model.

1.5.4 Relational Data Model

As demand for information grew and more sophisticated databases and applications were required, database design, management, and use became too cumbersome. The lack of query facility took a lot of time of the programmers to produce even the simplest reports. This led to the development of what came to be called the Relational Model database.

The term 'Relation' is derived from the set theory of mathematics. In the Relational Model, unlike the Hierarchical and Network models, there are no physical links. All data is maintained in the form of tables consisting of rows and columns. Data in two tables is related through common columns and not physical links. Operators are provided for operating on rows in tables.

Popular relational DBMSs are Oracle, Sybase, DB2, Microsoft SQL Server, and so on.

This model represents the database as a collection of relations. In this model's terminology, a row is called a tuple, a column, an attribute, and the table is called a relation. The list of values applicable to a particular field is called domain. It is possible for several attributes to have the same domain. The number of attributes of a relation is called degree of the relation. The number of tuples determines the cardinality of the relation.

In order to understand the relational model, consider tables 1.3 and 1.4.

Roll Number	Student Name
1	Sam Reiner
2	John Parkinson
3	Jenny Smith
4	Lisa Hayes
5	Penny Walker
6	Peter Jordan
7	Joe Wong

Table 1.3: Students Table

Roll Number	Marks Obtained
1	34
2	87
3	45
4	90
5	36
6	65
7	89

Table 1.4: Marks Table

The **`Students`** table displays the **`Roll Number`** and the **`Student Name`**, and the **`Marks`** table displays the **`Roll Number`** and **`Marks`** obtained by the students. Now, two steps must be carried out for students who have scored more than 50. First, locate the roll numbers of those who have scored more than 50 from the **`Marks`** table. Second, their names have to be located in the **`Students`** table by matching the roll number.

The result will be as shown in table 1.5.

Roll Number	Student Name	Marks Obtained
4	Lisa	90
6	Peter	65
7	Joe	89

Table 1.5: Displaying Student Names and Marks

It was possible to get this information because of two facts: First, there is a column common to both the tables - **`Roll Number`**. Second, based on this column, the records from the two different tables could be matched and the required information could be obtained.

In a relational model, data is stored in tables. A table in a database has a unique name that identifies its contents. Each table can be defined as an intersection of rows and columns.

Advantages of relational model

- The relational database model gives programmers time to concentrate on logical view of the database rather than being bothered about physical view. One of the reasons for popularity of relational databases is querying flexibility. Most relational databases use Structured Query Language (SQL). An RDBMS uses SQL to translate user query into technical code required to retrieve requested data. Relational model is so easy to handle that even untrained people find it easy to generate handy reports and queries, without giving much thought to the requirement to design a proper database.

Disadvantages of relational model

- Though the model hides all complexities of system, it tends to be slower than other database systems.
- As compared to all other models, relational data model is the most popular and widely used.

As compared to all other models, the relational data model is the most popular and widely used.

1.6 Relational Database Management System (RDBMS)

The Relational Model is an attempt to simplify database structures. It represents all data in the database as simple row-column tables of data values. An RDBMS is a software program that helps to create, maintain, and manipulate a relational database. A relational database is a database divided into logical units called tables, where tables are related to one another within the database.

Tables are related in a relational database, allowing adequate data to be retrieved in a single query (although the desired data may exist in more than one table). By having common keys, or fields, among relational database tables, data from multiple tables can be joined to form one large resultset.

Figure 1.5 shows two tables related to one another through a common key (data value) in a relational database.

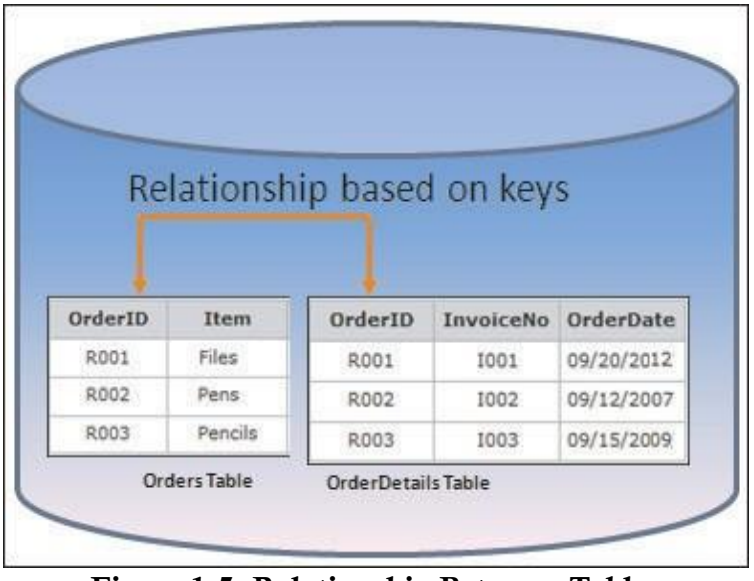


Figure 1.5: Relationship Between Tables

Thus, a relational database is a database structured on the relational model. The basic characteristic of a relational model is that in a relational model, data is stored in relations. To understand relations, consider the following example.

The **`Capitals`** table shown in table 1.6 displays a list of countries and their capitals, and the **`Currency`** table shown in table 1.7 displays the countries and the local currencies used by them.

Country	Capital
Greece	Athens
Italy	Rome
USA	Washington
China	Beijing
Japan	Tokyo
Australia	Sydney
France	Paris

Table 1.6: Capitals

Country	Currency
Greece	Drachma
Italy	Lira
USA	Dollar
China	Renminbi (Yuan)
Japan	Yen
Australia	Australian Dollar
France	Francs

Table 1.7: Currency

Both the tables have a common column, that is, the **`Country`** column. Now, if the user wants to display the information about the currency used in Rome, first find the name of the country to which Rome belongs. This information can be retrieved from table 1.6. Next, that country should be looked up in table

1.7 to find out the currency.

It is possible to get this information because it is possible to establish a relation between the two tables through a common column called **`Country`**.

1.6.1 Terms Related to RDBMS

There are certain terms that are mostly used in an RDBMS. These are described as follows:

- Data is presented as a collection of relations.
- Each relation is depicted as a table.
- Columns are attributes.
- Rows ('tuples') represent entities.
- Every table has a set of attributes that are taken together as a 'key' (technically, a 'superkey'), which uniquely identifies each entity.

For example, a company might have an **`Employee`** table with a row for each employee. What attributes might be interesting for such a table? This will depend on the application and the type of use the data will be put to, and is determined at database design time.
Consider the scenario of a company maintaining customer and order information for products being sold and customer-order details for a specific month, such as, August.

The tables 1.8, 1.9, 1.10, and 1.11 are used to illustrate this scenario. These tables depict tuples and attributes in the form of rows and columns. Various terms related to these tables are given in table 1.12.

Cust_No	Cust_Name	Phone No
002	David Gordon	0231-5466356
003	Prince Fernandes	0221-5762382
003	Charles Yale	0321-8734723
002	Ryan Ford	0241-2343444
005	Bruce Smith	0241-8472198

Table 1.8: Customer

Item_No	Description	Price
HW1	Power Supply	4000
HW2	Keyboard	2000
HW3	Mouse	800
SW1	Office Suite	15000
SW2	Payroll Software	8000

Table 1.9: **Items**

Ord_No	Item_No	Qty
101	HW3	50
101	SW1	150
102	HW2	10
103	HW3	50
104	HW2	25
104	HW3	100
105	SW1	100

Table 1.10: Order_Details

Ord_No	Ord_Date	Cust_No
101	02-08-12	002
102	11-08-12	003
103	21-08-12	003
104	28-08-12	002
105	30-08-12	005

Table 1.11: Order_August

Term	Meaning	Example from the Scenario
Relation	A table	Order_August, Order_Details, Customer and Items
Tuple	A row or a record in a relation	A row from Customer relation is a Customer tuple
Attribute	A field or a column in a relation	Ord_Date, Item_No, Cust_Name, and so on
Cardinality of a relation	The number of tuples in a relation	Cardinality of Order_Details relation is 7
Degree of a relation	The number of attributes in a relation	Degree of Customer relation is 3
Domain of an attribute	The set of all values that can be taken by the attribute	Domain of Qty in Order_Details is the set of all values which can represent quantity of an ordered item
Primary Key of a relation	An attribute or a combination of attributes that uniquely defines each tuple in a relation	Primary Key of Customer relation is Cust_No Ord_No and Item_No combination form the primary key of Order_Details
Foreign Key	An attribute or a combination of attributes in one relation R1 that indicates the relationship of R1 with another relation R2 The foreign key attributes in R1 must contain values matching with those of the values in R2	Cust_No in Order_August relation is a foreign key creating reference from Order_August to Customer. This is required to indicate the relationship between orders in Order_August and Customer

Table 1.12: **Terms Related to Tables**

1.6.2 RDBMS Users

The primary goal of a database system is to provide an environment for retrieving information from and storing new information into the database.

For a small personal database, one person typically defines the constructs and manipulates the database.

However, many persons are involved in the design, use, and maintenance of a large database with a few hundred users.

Database Administrator (DBA)

- Is a person who collects information that will be stored in database.
- Administering these resources is responsibility of DBA. DBA is also responsible for authorizing access to the database, for coordinating and monitoring its use, and for acquiring software and hardware resources as required. DBA is accountable for problems such as breach of security or poor system response time.

Database Designer

- Are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. It is the responsibility of database designers to communicate with all prospective database users, in order to understand their requirements, and to come up with a design that meets the requirements.

System Analysts and Application Programmers

- Determine requirements of end users, and develop specifications for pre- determined transactions that meet these requirements. Application Programmers implement these specifications as programs; then, they test, debug, document, and maintain these pre-determined transactions.

DBMS Designers and Implementers

- Design and implement DBMS modules and interfaces as a software package. A DBMS is a complex software system that consists of many components or modules, including modules for implementing the catalog, query language, interface processors, data access, and security. A DBMS must interface with other system software such as the operating system and compilers for various programming languages.

End User

- Invokes an application to interact with the system, or writes a query for easy retrieval, modification, or deletion of data.

1.7 Entities and Tables

The components of an RDBMS are entities and tables, which will be explained in this section.

1.7.1 Entity

An entity is a person, place, thing, object, event, or even a concept, which can be distinctly identified. For example, the entities in a university are students, faculty members, and courses.

Each entity has certain characteristics known as attributes. For example, the student entity might include attributes such as student number, name, and grade. Each attribute should be named appropriately.

A grouping of related entities becomes an entity set. Each entity set is given a name. The name of the entity set reflects the contents. Thus, the attributes of all the students of the university will be stored in an entity set called **`Student`**.

1.7.2 Tables and their Characteristics

The access and manipulation of data is facilitated by the creation of data relationships based on a construct known as a table. A table contains a group of related entities that is an entity set. The terms entity set and table are often used interchangeably. A table is also called a relation. The rows are known as tuples. The columns are known as attributes. Figure 1.6 highlights the characteristics of a table.

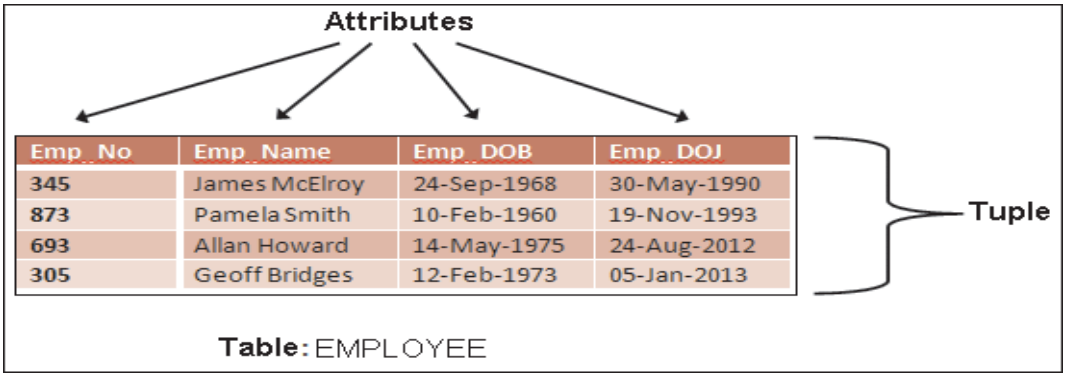


Figure 1.6: Characteristics of a Table

The characteristics of a table are as follows:

- A two-dimensional structure composed of rows and columns is perceived as a table.
- Each tuple represents a single entity within the entity set.
- Each column has a distinct name.
- Each row/column intersection represents a single data value.
- Each table must have a key known as primary key that uniquely identifies each row.
- All values in a column must conform to the same data format. For example, if the attribute is assigned a decimal data format, all values in the column representing that attribute must be in decimals.
- Each column has a specific range of values known as the attribute domain.
- Each row carries information describing one entity occurrence.
- The order of the rows and columns is immaterial in a DBMS.

1.8 Differences between a DBMS and an RDBMS

Differences between a DBMS and an RDBMS are listed in table 1.13.

DBMS	RDBMS
It does not require to have data in tabular structure nor does it enforce tabular relationships between data items.	In an RDBMS, tabular structure is a must and table relationships are enforced by the system. These relationships enable the user to apply and manage business rules with minimal coding.
Small amount of data can be stored and retrieved.	An RDBMS can store and retrieve large amount of data.
A DBMS is less secure than an RDBMS.	An RDBMS is more secure than a DBMS.
It is a single user system.	It is a multi-user system.
Most DBMSs do not support client/server architecture.	It supports client/server architecture.

Table 1.13: Difference between DBMS and RDBMS

In an RDBMS, a relation is given more importance. Thus, the tables in an RDBMS are dependent and the user can establish various integrity constraints on these tables so that the ultimate data used by the user remains correct. In case of a DBMS, entities are given more importance and there is no relation stablished among these entities.

1.9 Check Your Progress

1. The_____data model allows a child node to have more than one parent.

(A)	Flat File	(C)	Network
(B)	Hierarchical	(D)	Relational

2. _____ is used to administer permissions on the databases and database objects.

(A)	Data Definition Language (DDL)	(C)	Sub-schema
(B)	Data Manipulation Language (DML)	(D)	Data Control Language (DCL)

3. In the relational model terminology, a row is called a_____, a column an_____, and a table a _____.

(A)	attribute, tuple, relation	(C)	attribute, relation, tuple
(B)	tuple, attribute, relation	(D)	row, column, tuple

4. A_____can be defined as a collection of related records and a set of programs that access and manipulate these records.

(A)	Database Management System	(C)	Data Management
(B)	Relational Database Management System	(D)	Network Model

5. A_____describes a container for storing data and the process of storing and retrieving data from that container.

(A)	Network model	(C)	Data model
(B)	Flat File model	(D)	Relational model

1.9.1 Answers

1.	C
2.	D
3.	B
4.	B
5.	C

- A database is a collection of related data stored in the form of a table.
- A data model describes a container for storing data and the process of storing and retrieving data from that container.
- A DBMS is a collection of programs that enables the user to store, modify, and extract information from a database.
- A Relational Database Management System (RDBMS) is a suite of software programs for creating, maintaining, modifying, and manipulating a relational database.
- A relational database is divided into logical units called tables. These logical units are interrelated to each other within the database.
- The main components of an RDBMS are entities and tables.
- In an RDBMS, a relation is given more importance, whereas, in case of a DBMS, entities are given more importance and there is no relation established among these entities.

1. Create a PowerPoint presentation highlighting in brief what is a DBMS, different database models, and key features of an RDBMS.

Session - 2
Entity-Relationship (E-R) Model and Normalization

Welcome to the Session, **Entity-Relationship (E-R) Model and Normalization**.

This session talks about Data Modeling, the E-R model, its components, symbols, diagrams, relationships, Data Normalization, and Relational Operators.

In this session, you will learn to:
- Define and describe data modeling
- Identify and describe the components of the E-R model
- Identify relationships that can be formed between entities
- Explain E-R diagrams and their use
- Describe an E-R diagram, the symbols used for drawing, and show various relationships
- Describe various Normal Forms
- Outline uses of different Relational Operators

2.1 Introduction

A data model is a group of conceptual tools that describes data, its relationships, and semantics. It also consists of the consistency constraints that the data adheres to. The Entity-Relationship, Relational, Network, and Hierarchical models are examples of data models. The development of every database begins with the basic step of analyzing its data in order to determine the data model that would best represent it. Once this step is completed, the data model is applied to the data.

2.2 Data Modeling

The process of applying an appropriate data model to the data, in order to organize and structure it, is called data modeling.

Data modeling is as essential to database development as are planning and designing to any project development. Building a database without a data model is similar to developing a project without its plans and design. Data models help database developers to define the relational tables, primary and foreign keys, stored procedures, and triggers required in the database.

Data modeling can be broken down into the following three broad steps:

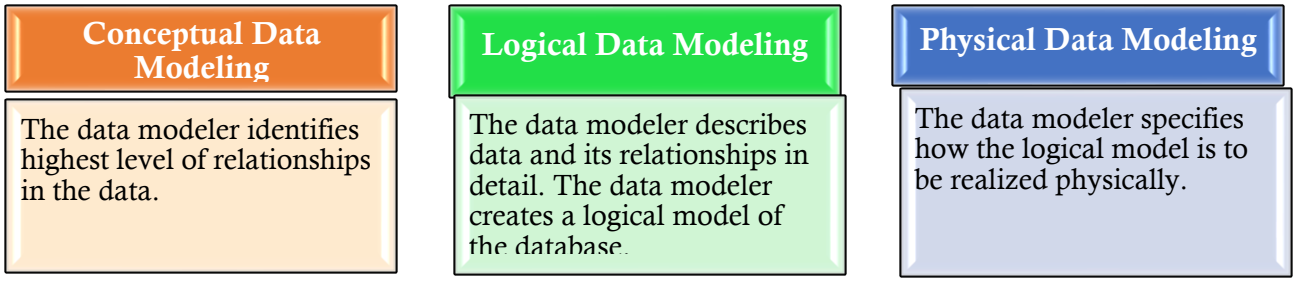


Figure 2.1 exhibits various steps involved in data modeling.

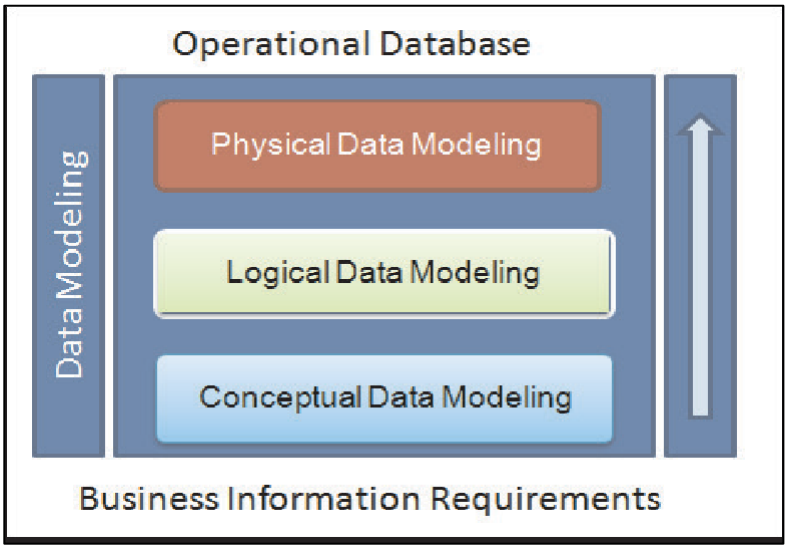


Figure 2.1: **Data Modeling Steps**

2.3 Entity-Relationship (E-R) Model

Data models can be classified into three different groups:

The Entity-Relationship (E-R) model belongs to the first classification.

The model is based on a simple idea. Data can be perceived as real-world objects called entities and the relationships that exist between them. For example, the data about employees working for an organization can be perceived as a collection of employees and a collection of various departments that form the organization. Both employee and department are real-world objects. An employee belongs to a department. Thus, the relation 'belongs to' links an employee to a particular department.

The employee-department relation can be modeled as shown in figure 2.2.

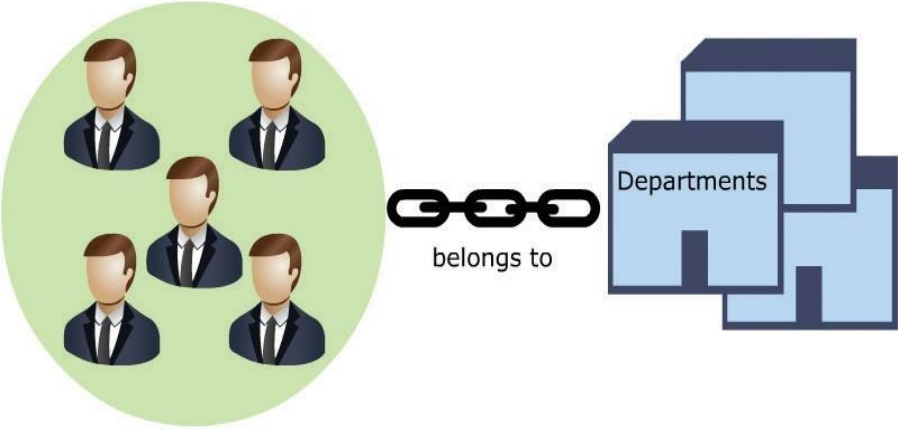


Figure 2.2: E-R Model Depiction of an Organization

An E-R model consists of five basic components. They are as follows:

Entity	An entity is a real-world object that exists physically and is distinguishable from other objects. For example, employee, department, student, customer, vehicle, and account are entities.
Relationship	A relationship is an association or bond that exists between one or more entities. For example, belongs to, owns, works for, saves in, purchased, and so on.
Attributes	Attributes are features that an entity has. Attributes help distinguish every entity from another. For example, the attributes of a student would be **roll_number**, **name**, **stream**, **semester**, and so on. The attributes of a car would be **registration_number**, **model**, **manufacturer**, **color**, **price**, **owner**, and so on.
Entity Set	An entity set is the collection of similar entities. For example, the employees of an organization collectively form an entity set called employee entity set.
Relationship Set	A collection of similar relationships between two or more entity sets is called a relationship set. For example, employees work in a particular department. The set of all 'work in' relations that exists between the employees and the department is called the 'work in' relationship set.

Various E-R model components can be seen in figure 2.3.

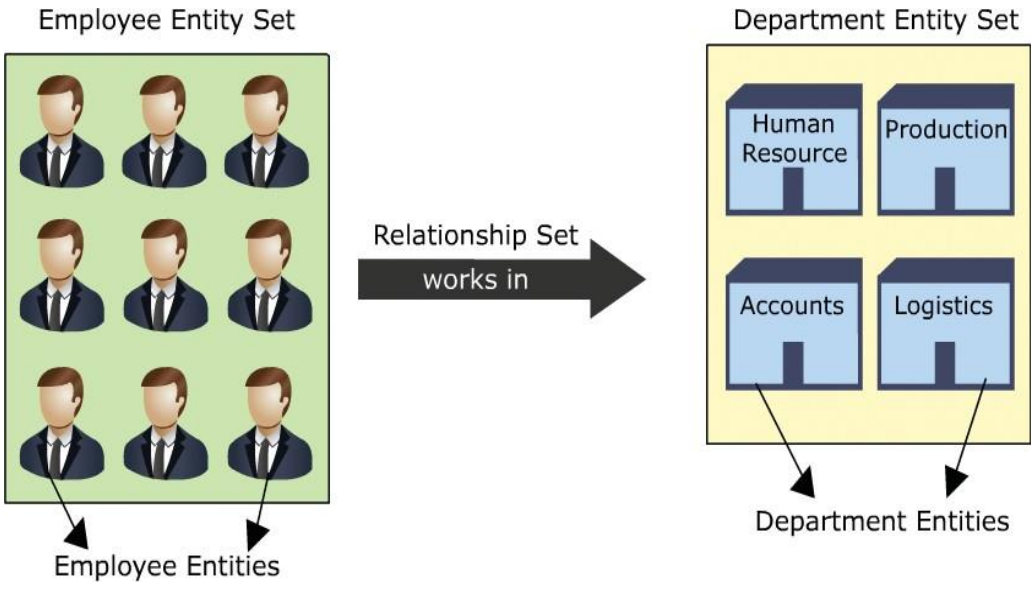


Figure 2.3: Components of the E-R Model

Relationships associate one or more entities and can be of three types. They are as follows:

- **Self-relationships**

  Relationships between entities of the same entity set are called self-relationships. For example, a manager and his team member, both belong to the employee entity set. The team member works for the manager. Thus, the relation, 'works for', exists between two different employee entities of the same employee entity set.

  The relationship can be seen in figure 2.4.

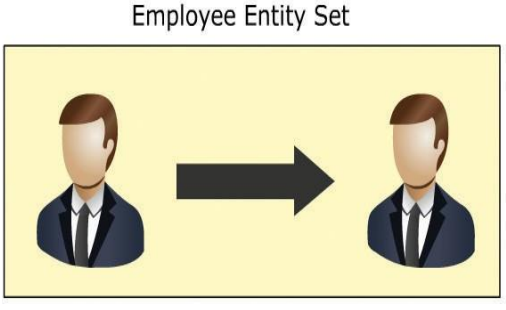


Figure 2.4: **Self-Relationship**

- **Binary relationships**

  Relationships that exist between entities of two different entity sets are called binary relationships. For example, an employee belongs to a department. The relation exists between two different entities, which belong to two different entity sets. The employee entity belongs to an employee entity set. The department entity belongs to a department entity set.

The relationship can be seen in figure 2.5.

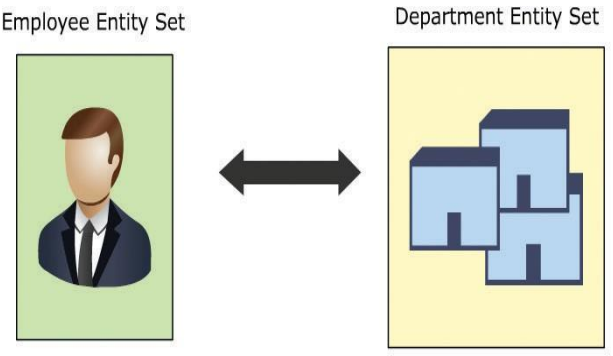


Figure 2.5: **Binary Relationship**

- **Ternary relationships**

  Relationships that exist between three entities of different entity sets are called ternary relationships. For example, an employee works in the accounts department at the regional branch. The relation, 'works' exists between all three, the employee, the department, and the location.

  The relationship can be seen in figure 2.6.

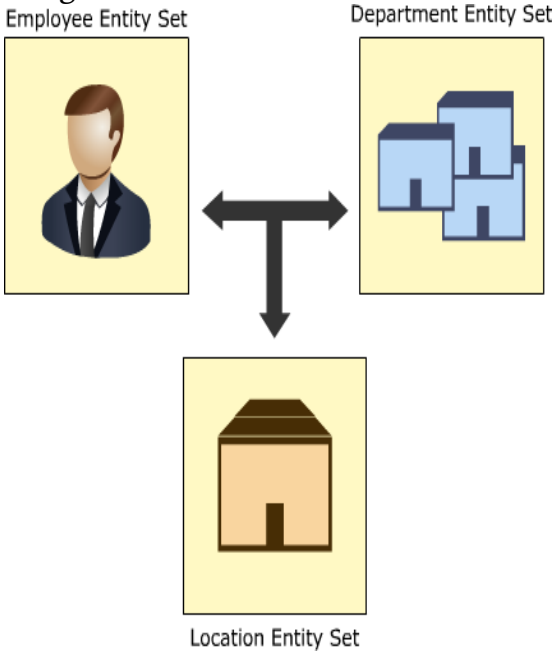


Figure 2.6: Ternary Relationship

Relationships can also be classified as per mapping cardinalities. Different mapping cardinalities are as follows:

One-to-One

This kind of mapping exists when an entity of one entity set can be associated with only one entity of another set. Consider the relationship between a vehicle and its registration. Every vehicle has a unique registration. No two vehicles can have the same registration details. The relation is one-to-one, that is, one

vehicle-one registration. The mapping cardinality can be seen in figure 2.7.

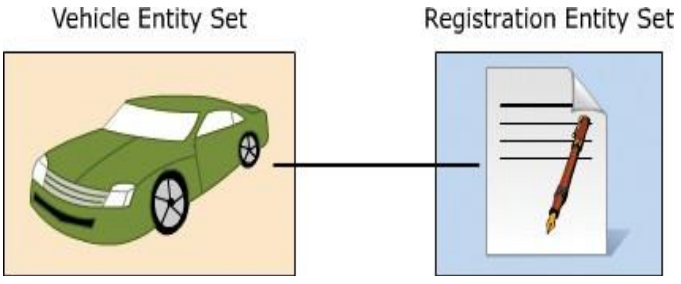


Figure 2.7: One-to-One Mapping Cardinality

One-to- Many

This kind of mapping exists when an entity of one set can be associated with more than one entity of another entity set.

Consider the relation between a customer and the customer's vehicles. A customer can have more than one vehicle. Therefore, the mapping is a one to many mapping, that is, one customer - one or more vehicles. The mapping cardinality can be seen in figure 2.8.

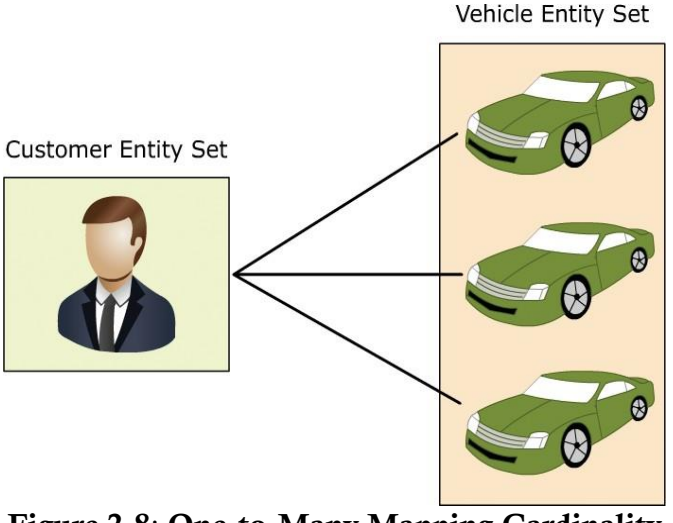


Figure 2.8: **One-to-Many Mapping Cardinality**

Many-to-One

This kind of mapping exists when many entities of one set is associated with an entity of another set. This association is done irrespective of whether the latter entity is already associated to other or more entities of the former entity set.

Consider the relation between a vehicle and its manufacturer. Every vehicle has only one manufacturing company or coalition associated to it under the relation, 'manufactured by', but the same company or coalition can manufacture more than one kind of vehicle.

The mapping can be seen in figure 2.9.

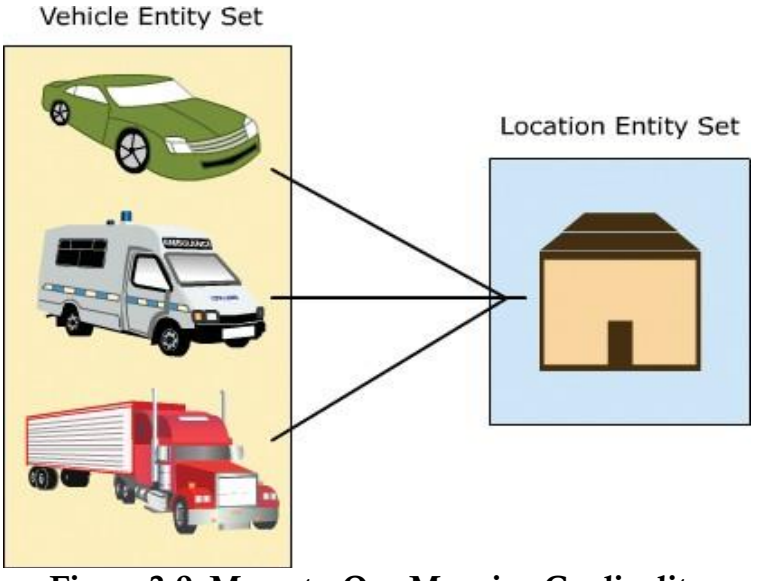


Figure 2.9: **Many-to-One Mapping Cardinality**

Many-to-Many

This kind of mapping exists when any number of entities of one set can be associated with any number of entities of the other entity set.

Consider the relation between a bank's customer and the customer's accounts. A customer can have more than one account and an account can have more than one customer associated with it in case it is a joint account or similar. Therefore, the mapping is many-to-many, that is, one or more customers associated with one or more accounts.

The mapping cardinality can be seen in figure 2.10.

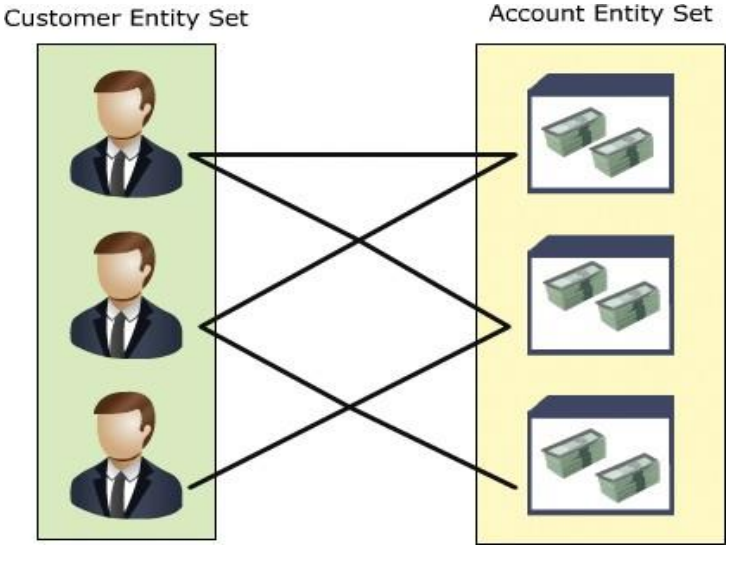


Figure 2.10: Many-to-Many Mapping Cardinality

Some additional concepts in the E-R model are as follows:

Primary keys

A primary key is an attribute that can uniquely define an entity in an entity set. Consider table 2.1 containing the details of students in a school.

Enrollment_Number	Name	Grade	Division
786	Ashley	Seven	B
957	Joseph	Five	A
1011	Kelly	One	A

Table 2.1: Student Details

In a school, every student has a unique **`enrollment number`** (such as **`enrollment_number`** in table 2.1), which is unique to the student. Any student can be identified based on the enrollment number. Thus, the attribute **`enrollment_number`** plays the role of the primary key in the **`Student Details`** table.

Weak entity sets

Entity sets that do not have enough attributes to establish a primary key are called weak entity sets.

Strong entity sets

Entity sets that have enough attributes to establish a primary key are called strong entity sets. Consider the scenario of an educational institution where at the end of each semester, students are required to complete and submit a set of assignments. The teacher keeps track of the assignments submitted by the students. Now, an assignment and a student can be considered as two separate entities. The assignment entity is described by the attributes **assignment_number**and **subject**. The student entity is described by **roll_number**, **name**, and **semester**. The assignment entities can be grouped to form an assignment entity set and the student entities can be grouped to form a student entity set. The entity sets are associated by the relation 'submitted by'. This relation is depicted in figure 2.11.

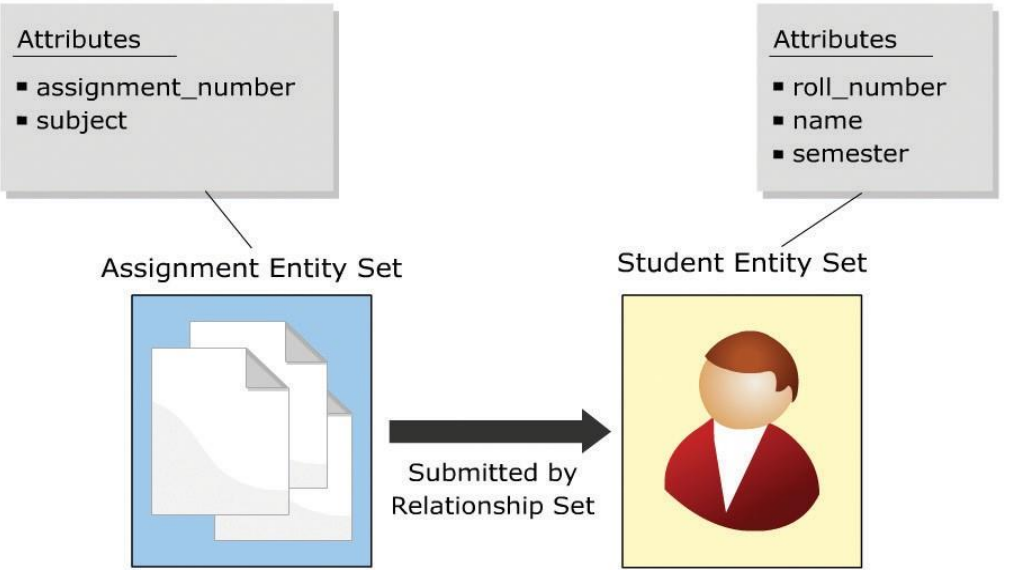


Figure 2.11: Assignment Student Relation

The attributes, **`assignment_number`** and **`subject`**, are not enough to identify an assignment entity uniquely. The **`roll_number`** attribute alone is enough to uniquely identify any student entity. Therefore, **`roll_number`** is a primary key for the student entity set. The assignment entity set is a weak entity set since it lacks a primary key. The student entity set is a strong entity set due to the presence of the **`roll_number`** attribute.

2.3.1 Entity-Relationship Diagrams

The E-R diagram is a graphical representation of the E-R model. The E-R diagram, with the help of various symbols, effectively represents various components of the E-R model.

The symbols used for various components can be seen in table 2.2.

Component	Symbol	Example
Entity	Entity	Student
Weak Entity	Weak Entity	Assignments
Attribute	Attribute	Roll_num

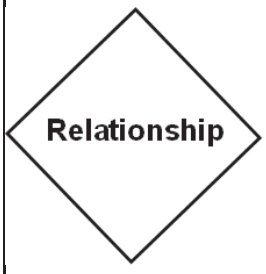
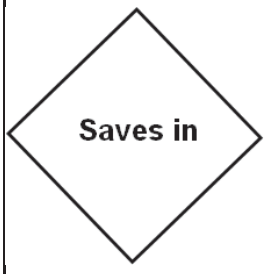
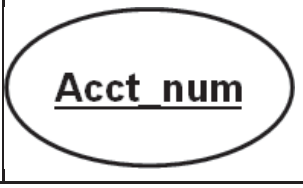
Component	Symbol	Example
Relationship		
Key Attribute		

Table 2.2: E-R Diagram Symbols

Attributes in the E-R model can be further classified as follows:

- **Multi-valued**

A multi-valued attribute is illustrated with a double-line ellipse, which has more than one value for at least one instance of its entity. This attribute may have upper and lower bounds specified for any individual entity value.

The telephone attribute of an individual may have one or more values, that is, an individual can have one or more telephone numbers. Hence, the telephone attribute is a multi-valued attribute.

The symbol and example of a multi-valued attribute can be seen in figure 2.12.

Figure 2.12: **Symbol and Example of Multi-valued Attribute**

- **Composite**

A composite attribute may itself contain two or more attributes, which represent basic attributes having independent meanings of their own.

The address attribute is usually a composite attribute, composed of attributes such as street, area, and so on. The symbol and example of a composite attribute can be seen in figure 2.13.

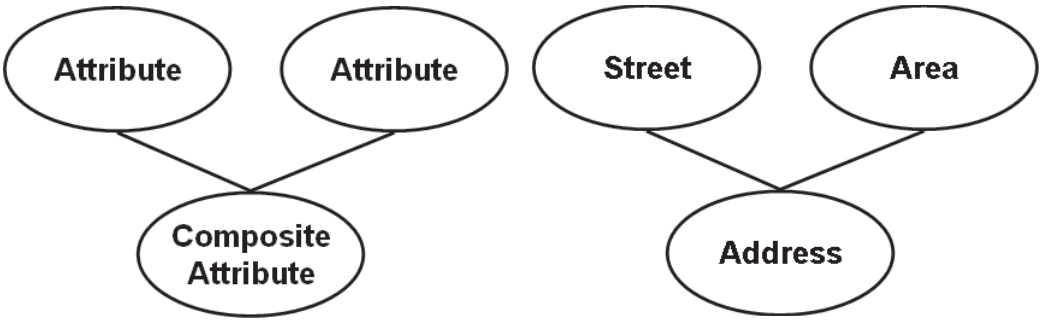


Figure 2.13: Symbol and Example of Composite Attribute

- **Derived**

Derived attributes are attributes whose value is entirely dependent on another attribute and are indicated by dashed ellipses.

The age attribute of a person is the best example for derived attributes. For a particular person entity, the age of a person can be determined from the current date and the person's birth date. The symbol and example of a derived attribute can be seen in figure 2.14.

Figure 2.14: Symbol and Example of Derived Attribute

Steps to construct an E-R diagram are as follows:

1. Gather all the data that must be modeled.
2. Identify data that can be modeled as real-world entities.
3. Identify the attributes for each entity.
4. Sort entity sets as weak or strong entity sets.
5. Sort entity attributes as key attributes, multi-valued attributes, composite attributes, derived attributes, and so on.
6. Identify the relations between the different entities.
7. Using different symbols draw the entities, their attributes, and their relationships. Use appropriate symbols while drawing attributes.

Consider the scenario of a bank, with customers and accounts. The E-R diagram for the scenario can be constructed as follows:

1. Step 1: **Gather data**

The bank is a collection of accounts used by customers to save money.

2. Step 2: **Identify entities**

Customer

Account

3. Step 3: **Identify the attributes**

Customer: customer_name, customer_address, customer_contact

Account: account_number, account_owner, balance_amount

4. **Step 4**: **Sort entity sets**

Customer entity set: weak entity set

Account entity set: strong entity set

5. **Step 5**: **Sort attributes**

Customer entity set: customer_address - composite, customer_contact - multi-valued

Account entity set: account_number → primary key, account_owner – multi-valued

6. **Step 6**: **Identify relations**

A customer 'saves in' an account. The relation is 'saves in'.

7. **Step 7**: Draw diagram using symbols

Figure 2.15 shows the E-R diagram for the bank.

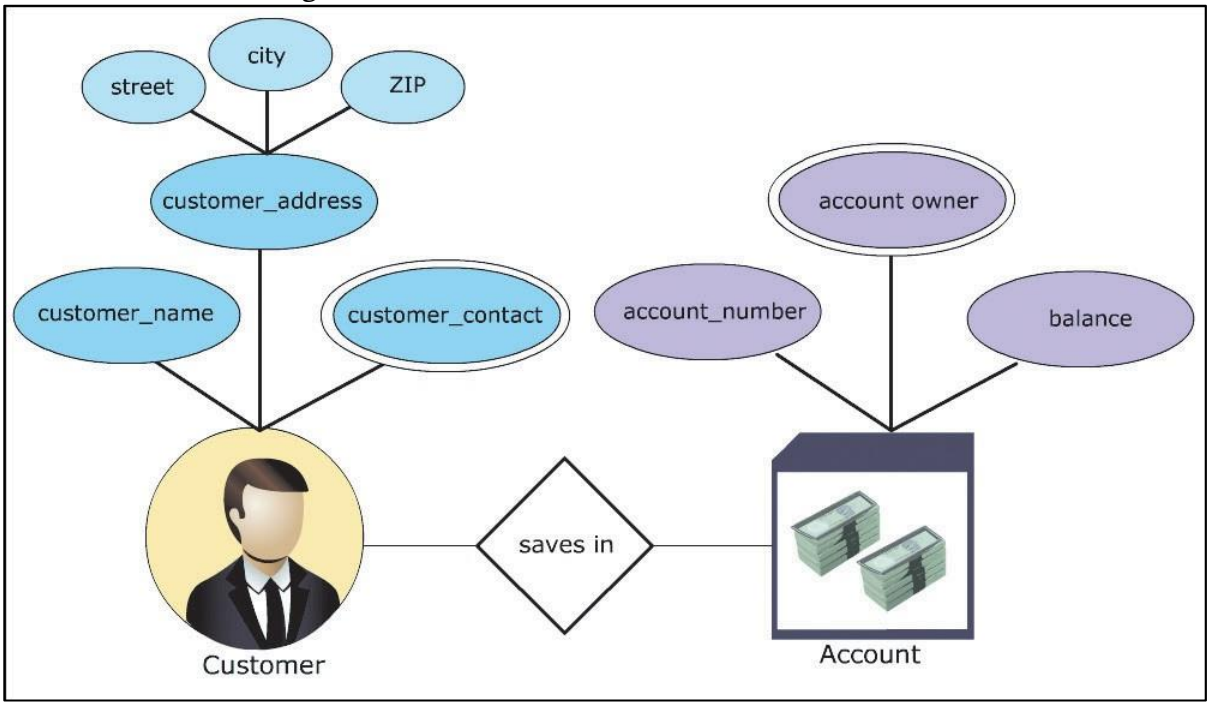


Figure 2.15: **E-R Diagram for the Bank**

2.4 Normalization

Initially, all databases are characterized by large number of columns and records. This approach has certain drawbacks. Consider the following details of the employees in a department. Table 2.3 consists of the employee details as well as the details of the project they are working on.

Emp_No	Project_Id	Project_Name	Emp_Name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John	A	20,000
168	113	BLUE STAR	James	B	15,000
263	113	BLUE STAR	Andrew	C	10,000
109	124	MAGNUM	Bob	C	10,000

Table 2.3: **Department Employee Details**

Repetition anomaly

The data such as **Project_Id**, **Project_Name**, **Grade**, and **Salary** repeat many times. This repetition hampers both, performance during retrieval of data and the storage capacity. This repetition of data is called the repetition anomaly.

The repetition is shown in table 2.4 with the help of shaded cells.

Emp_No	Project_Id	Project_Name	Emp_Name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John	A	20,000
168	113	BLUE STAR	James	B	15,000
263	113	BLUE STAR	Andrew	C	10,000
109	124	MAGNUM	Bob	C	10,000

Table 2.4: Department Employee with Insertion Anomaly

Insertion anomaly

Suppose the department recruits a new employee named **Ann**. Now, consider that **Ann** has not been assigned any project. Insertion of her details in the table would leave columns **Project_Id** and **Project_Name** empty. Leaving columns blank could lead to problems later. Anomalies created by such insertions are called insertion anomalies. The anomaly can be seen in table 2.5.

Emp_No	Project_Id	Project_Name	Emp_Name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John	A	20,000
168	113	BLUE STAR	James	B	15,000
263	113	BLUE STAR	Andrew	C	10,000
109	124	MAGNUM	Bob	C	10,000
195	-	-	Ann	C	10,000

Table 2.5: **Department Employee Details**

Deletion anomaly

Suppose, Bob is relieved from the project MAGNUM. Deleting the record deletes Bob's **Emp_No**, **Grade**, and **Salary** details too. This loss of data is harmful as all of Bob's personal details are also lost as seen in the table 2.6. This kind of loss of data due to deletion is called deletion anomaly. The anomaly can be seen in table 2.6.

Emp_No	Project_Id	Project_Name	Emp_Name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John Smith	A	20,000
168	113	BLUE STAR	James Kilber	B	15,000
263	113	BLUE STAR	Andrew Murray	C	10,000

Table 2.6: **Employee Project Details**

Updating anomaly

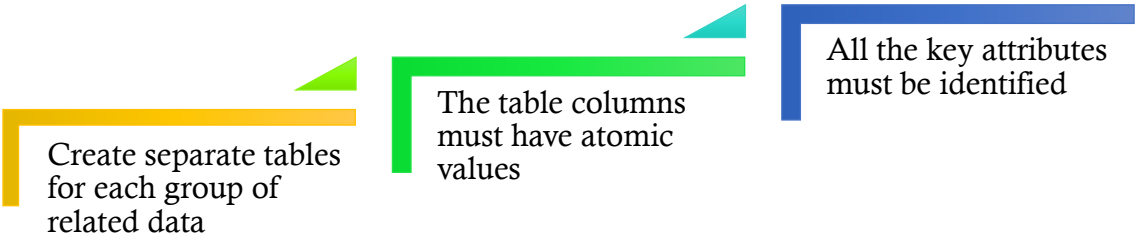
Suppose John was given a hike in **`Salary`** or John was demoted. The change in John's **`Salary`** or **`Grade`** must be reflected in all projects John works for. This problem in updating all the occurrences is called updating anomaly.

The **`Department Employee Details`** table is called an unnormalized table. These drawbacks lead to the need for normalization.

Normalization is the process of removing unwanted redundancy and dependencies. Initially, Codd (1972) presented three normal forms (1NF, 2NF, and 3NF), all based on dependencies among the attributes of a relation. The fourth and fifth normal forms are based on multi-value and join dependencies and were proposed later.

2.4.1 First Normal Form

In order to achieve the first normal form, following steps must be performed:



Consider the **`Employee Project Details`** table shown in table 2.7.

Emp_No	Project_Id	Project_Name	Emp_Name	Grade	Salary
142	113, 124	BLUE STAR, MAGNUM	John	A	20,000
168	113	BLUE STAR	James	B	15,000
263	113	BLUE STAR	Andrew	C	10,000
109	124	MAGNUM	Bob	C	10,000

Table 2.7: Employee Project Details

The table has data related to projects and employees. The table must be split into two tables, that is, a **`Project Details`** table and an **`Employee Details`** table. The table columns, **`Project_Id`** and **`Project_Name`**, have multiple values. The data must be split over different rows. The resultant tables

are **Project Details** and **Employee Details** as shown in tables 2.8 and 2.9.

Project_Id	Project_Name
113	BLUE STAR
124	MAGNUM

Table 2.8: Project Details

Emp_No	Emp_Name	Grade	Salary
142	John	A	20,000
168	James	B	15,000
263	Andrew	C	10,000
109	Bob	C	10,000

Table 2.9: **Employee Details**

The **Project_Id** attribute is the primary key for the **Project Details** table.

The **Emp_No** attribute is the primary key for the **Employee Details** table. Therefore, in first normal form, the initial **Employee Project Details** table has been reduced to the **Project Details** and **Employee Details** tables.

2.4.2 Second Normal Form

Tables are said to be in second normal form if:

- They meet the requirements of the first normal form
- There are no partial dependencies in the tables
- The tables are related through foreign keys

Partial dependency means a non-key attribute should not be partially dependent on more than one key attribute. The **Project Details** and **Employee Details** tables do not exhibit any partial dependencies. The **Project_Name** is dependent only on **Project_Id** and **Emp_Name**, **Grade**, and **Salary** are dependent only on **Emp_No**. The tables also need to be related through foreign keys. A third table, named **Employee Project Details**, is created with only two columns, **Project_Id** and **Emp_No**.

So, the project and employee details tables on conversion to second normal form generates tables **Project Details**, **Employee Details**, and **Employee Project Details** as shown in tables 2.10, 2.11, and 2.12.

Project_Id	Project_Name
113	BLUE STAR
124	MAGNUM

Table 2.10: Project Details After Conversion to Second Normal Form

Emp_No	Emp_Name	Grade	Salary
142	John	A	20,000
168	James	B	15,000
263	Andrew	C	10,000

Emp_No	Emp_Name	Grade	Salary
109	Bob	C	10,000

Table 2.11: **Employee Details After Conversion to Second Normal Form**

Emp_No	Project_Id
142	113
142	124
168	113
263	113
109	124

Table 2.12: Employee Project Details After Conversion to Second Normal Form

The attributes, **`Emp_no`** and **`Project_id`**, of the **`Employee Project Details`** table combine together to form the primary key. Such primary keys are called composite primary keys.

2.4.3 Third Normal Form

To achieve the third normal form:

- The tables should meet the requirements of the second normal form
- The tables should not have transitive dependencies in them

The **`Project Details`**, **`Employee Details`**, and **`Employee Project Details`** tables are in second normal form. If an attribute can be determined by another non-key attribute, it is called a transitive dependency. To make it simpler, every non-key attribute should be determined by the key attribute only. If a non-key attribute can be determined by another non-key attribute, it must put into another table.

On observing the different tables, it is seen that the **`Project Details`** and **`Employee Project Details`** tables do not exhibit any such transitive dependencies. The non-key attributes are totally determined by the key attributes. **`Project_Name`** is only determined by **`Project_Id`**. On further scrutinizing the **`Employee Details`** table, a certain inconsistency is seen. The attribute **`Salary`** is determined by the attribute **`Grade`** and not the key attribute **`Emp_No`**. Thus, this transitive dependency must be removed.

The **`Employee Details`** table can be split into the **`Employee Details`** and **`Grade Salary Details`** tables as shown in tables 2.13 and 2.14.

Emp_No	Emp_Name	Grade
142	John	A
168	James	B
263	Andrew	C
109	Bob	C

Table 2.13: **Employee Details After Conversion**

Grade	Salary
A	20,000
B	15,000
C	10,000

Table 2.14: **Grade Salary Details Table**

Thus, at the end of the three normalization stages, the initial **`Employee ProjectDetails`** table has been reduced to the **`Project Details`**, **`Employee Project Details`**, **`Employee Details`**, and **`Grade Salary Details`** tables as shown in tables 2.15, 2.16, 2.17, and 2.18.

Project_Id	Project_Name
113	BLUE STAR
124	MAGNUM

Table 2.15: **Project Details After Conversion to Third Normal Form**

Emp_No	Project_Id
142	113
142	124
168	113
263	113
109	124

Table 2.16: **Employee Project Details After Conversion to Third Normal Form**

Emp_No	Emp_Name	Grade
142	John	A
168	James	B
263	Andrew	C
109	Bob	C

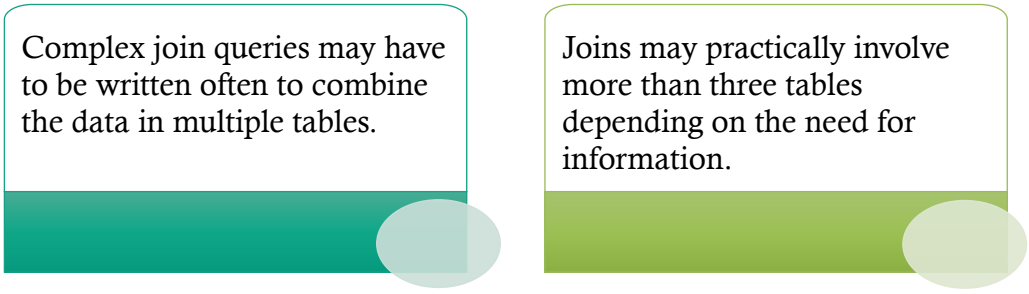
Table 2.17: **Employee Details After Conversion to Third Normal Form**

Grade	Salary
A	20,000
B	15,000
C	10,000

Table 2.18: **Grade Salary Details After Conversion to Third Normal Form**

2.4.4 Denormalization

By normalizing a database, redundancy is reduced. This, in turn, reduces the storage requirements for the database and ensures data integrity. However, it has some drawbacks. They are as follows:



If such joins are used very often, the performance of the database will become very poor.
The CPU time required to solve such queries will be very large too. In such cases, storing a few fields redundantly can be ignored to increase the performance of the database. The databases that possess such minor redundancies in order to increase performance are called denormalized databases and the process of doing so is called denormalization.

2.5 Relational Operators

The relational model is based on the solid foundation of Relational Algebra. Relational Algebra consists of a collection of operators that operate on relations. Each operator takes one or two relations as its input and produces a new relation as its output.

Consider the **`Branch Reserve Details`** table as shown in table 2.19.

Branch	Branch_Id	Reserve (Billion €)
London	BS-01	9.2
London	BS-02	10
Paris	BS-03	15
Los Angeles	BS-04	50
Washington	BS-05	30

Table 2.19: Branch Reserve Details

SELECT

The `SELECT` operator is used to extract data that satisfies a given condition. The lowercase Greek letter sigma, 'σ', is used to denote selection. A select operation, on the **`Branch Reserve Details`** table, to display the details of the branches in London would result in table 2.20.

Branch	Branch_Id	Reserve (Billion €)
London	BS-01	9.2
London	BS-02	10

Table 2.20: Details of Branches in London

A selection on the **Branch Reserve Details** table to display branches with reserve greater than 20 billion Euros would result in table 2.21.

Branch	Branch_Id	Reserve (Billion €)
Los Angeles	BS-04	50
Washington	BS-05	30

Table 2.21: Details of Branches with Reserves Greater Than 20 Billion Euros

PROJECT

The PROJECT operator is used to project certain details of a relational table. The PROJECT operator only displays the required details leaving out certain columns. The PROJECT operator is denoted by the Greek letter pi, 'Π'. Assume that only the **Branch_Id** and **Reserve** amounts need to bedisplayed.

A project operation to do the same, on the **Branch Reserve Details** table, would result in table 2.22.

Branch_Id	Reserve (Billion €)
BS-01	9.2
BS-02	10
BS-03	15
BS-04	50
BS-05	30

Table 2.22: Resultant Table with Branch_Id and Reserve Amounts

PRODUCT

The PRODUCT operator, denoted by 'x' helps combine information from two relational tables. Consider table 2.23.

Branch_Id	Loan Amount (Billion €)
BS-01	0.56
BS-02	0.84

Table 2.23: **Branch Loan Details**

The product operation on the **Branch Reserve Details** and **Branch Loan Details** tables would result in table 2.24.

Branch	Branch_Id	Reserve (Billion €)	Loan Amount (Billion €)
London	BS-01	9.2	0.56
London	BS-01	9.2	0.84
London	BS-02	10	0.56
London	BS-02	10	0.84
Paris	BS-03	15	0.56
Paris	BS-03	15	0.84
Los Angeles	BS-04	50	0.56
Los Angeles	BS-04	50	0.84

Branch	Branch_Id	Reserve (Billion €)	Loan Amount (Billion €)
Washington	BS-05	30	0.56
Washington	BS-05	30	0.84

Table 2.24: Product of Branch Reserve Details and Branch Loan Details

The product operation combines each record from the first table with all the records in the second table, somewhat generating all possible combinations between the table records.

UNION

Suppose an official of the bank with the data given in tables 2.19 and 2.23 wanted to know which branches had reserves below 20 billion Euros or loans. The resultant table would consist of branches with either reserves below 20 billion Euros or loans or both.

This is similar to the union of two sets of data; first, set of branches with reserve less than 20 billion Euros and second, branches with loans. Branches with both, reserves below 20 billion Euros and loans would be displayed only once. The `UNION` operator does just that, it collects the data from the different tables and presents a unified version of the complete data. The union operation is represented by the symbol, 'U'. The union of the **`Branch Reserve Details`** and **`Branch Loan Details`** tables would generate table 2.25.

Branch	Branch_Id
London	BS-01
London	BS-02
Paris	BS-03

Table 2.25: Unified Representation of Branches with Less Reserves or Loans

INTERSECT

Suppose the same official after seeing this data wanted to know which of these branches had both low reserves and loans too. The answer would be the intersect relational operation. The `INTERSECT` operator generates data that holds true in all the tables it is applied on. It is based on the intersection set theory and is represented by the '∩' symbol. The result of the intersection of the **`BranchReserveDetails`** and **`Branch Loan Details`** tables would be a list of branches that have both reserves below 20 billion Euros and loans in their account. The resultant table generated is table 2.26.

Branch	Branch_Id
London	BS-01
London	BS-02

Table 2.26: Branches with Low Reserves and Loans

DIFFERENCE

If the same official now wanted the list of branches that had low reserves but no loans, then the official would have to use the difference operation. The `DIFFERENCE` operator, symbolized as '-', generates data from different tables too, but it generates data that holds true in one table and not the other. Thus, the branch would have to have low reserves and no loans to be displayed.

Table 2.27 is the result generated.

Branch	Branch_Id
Paris	BS-03

Table 2.27: Branches with Low Reserves but No Loans

JOIN

The `JOIN` operation is an enhancement to the product operation. It allows a selection to be performed on the product of tables. For example, if the reserve values and loan amounts of branches with low reserves and loan values was needed, the product of the **`Branch Reserve Details`** and **`Branch Loan Details`** would be required. Once the product of tables 2.19 and 2.23 would be generated, only those branches would be listed which have both reserves below 20 billion Euros and loans.

Table 2.28 is generated as a result of the `JOIN` operation.

Branch	Branch_Id	Reserve (Billion €)	Loan Amount (Billion €)
London	BS-01	9.2	0.56
London	BS-02	10	0.84

Table 2.28: Detailed List of Branches with Low Reserve and Loans

DIVIDE

Suppose an official wanted to see the branch names and reserves of all the branches that had loans. This process can be made very easy by using the `DIVIDE` operator. All that the official must do is divide the **`Branch Reserve Details`** table (shown earlier in table 2.19) by the list of branches, that is, the **`Branch Id`** column of the **`Branch Loan Details`** table (shown earlier in table 2.23). Table 2.29 is the result generated.

Branch	Reserve (Billion €)
London	9.2
London	10

Table 2.29: Resultant Table of Division Operation

Note that the attributes of the divisor table should always be a subset of the dividend table. The resultant table would always be void of the attributes of the divisor table and the records not matching the records in the divisor table.

2.6 Check Your Progress

1. One or more attributes that can uniquely define an entity from an entity set is called a _____ key.

(A)	Primary	(C)	Alternate
(B)	Foreign	(D)	Super

2. An attribute that contains two or more attribute values in it is called a ____attribute.

(A)	Derived	(C)	Multi-valued
(B)	Composite	(D)	Network

3. Transitive dependence is eliminated in the__normal form.

(A)	First	(C)	Third
(B)	Second	(D)	Fourth

4. Which one of these operations is further enhanced in the Product operation?

(A)	Divide	(C)	Difference
(B)	Intersection	(D)	Join

5. Which of the following are the basic components of an E-R model?
   a. Entity
   b. Relationship
   c. Attributes
   d. Relationship Chart
   e. Relationship Set

(A)	a, b, c	(C)	a, c, e
(B)	a, d, c	(D)	a, b, c, e

2.6.1 Answers

1.	A
2.	B
3.	C
4.	D
5.	D

- Data modeling is the process of applying an appropriate data model to the data at hand.
- E-R model views the real-world as a set of basic objects and relationships among them.
- Entity, attributes, entity set, relationships, and relationship sets form the five basic components of E-R model.
- Mapping cardinalities express the number of entities that an entity is associated with.
- The process of removing redundant data from the tables of a relational database is called normalization.
- Relational Algebra consists of a collection of operators that help retrieve data from the relational databases.
- SELECT, PRODUCT, UNION, and DIVIDE are some of the relational algebra operators.

1. **Dynamic Data Solutions Ltd.** is a popular company based in **Chicago**, **Illinois** providing database solutions to its clients. The company is in the process of conducting recruitment interviews in different technical institutes for upcoming projects. The candidates interested in database development can attend the campus interviews. Based on their performance, the candidates are provided short-term training in any DBMS of their choice and recruited for database development in the company.

   Consider yourself as a student of second year Computer Science who has participated in the campus interview. You have been provided with a set of interview questions as follows:

   a) Explain in brief the meaning of data and database.
   b) List the two different approaches for data management.
   c) Explain the benefits of a DBMS.
   d) List the different database models used in the industry.
   e) Explain the RDBMS model in brief and differentiate between DBMS and RDBMS.

2. Champs Online is an online learning company that provides computer education to kids. The company has basic courses in subjects of Computer Science such as fundamentals of computers, databases, programming languages, and so on. The company has organized their data in a manner similar to this:

Student ID	Student Name	Course ID	Course Title	Fees	Marks
S001	Rob Martin	C001	Programming in Java	5000	50
S002	Maria Stevens	C001	Programming in Java	5000	70
S003	Clark Hood	C002	Networking Fundamentals	6000	80
S002	Maria Stevens	C002	Networking Fundamentals	6000	75
S003	Clark Hood	C003	Database Management	7000	68
S001	Rob Martin	C003	Database Management	7000	79
S002	Maria Stevens	C003	Database Management	7000	90

   a) Identify the problem in the given table.
   b) Provide a solution to resolve the problem.

(**Hint**: Use the concept of Normalization.)

Session - 3
Introduction to SQL Server 2019

Welcome to the Session, **Introduction to SQL Server 2019**. This session introduces SQL Server 2019. It explains the basic architecture of SQL Server 2019 and lists the versions and editions of SQL Server. It also explains the role and structure of SQL Server along with the new features added in SQL Server 2019. Finally, the session explains the process to connect to SQL Server instances, create and organize script files, and execute Transact-SQL queries.

In this session, you will learn to:

- Describe an overview of SQL Server 2019
- Describe basic architecture and version history of SQL Server 2019
- Outline the process of connecting to SQL Server instances
- Define databases and list the key features of AdventureWorks2019 sample database
- Explain the components of SQL Server Management Studio GUI
- Explain script file creation and organization
- Explain the process to execute Transact-SQL queries

3.1 Introduction to SQL Server 2019

SQL Server is an RDBMS developed by Microsoft. It provides an enterprise-level data management platform for an organization. SQL Server includes numerous features and tools that make it an outstanding database and data analysis platform. It is also targeted for large-scale Online Transactional Processing (OLTP), data warehousing, and e-commerce applications.

Recent versions of SQL Server have brought a revolutionary change in areas such as speedy transactions, higher security, more profound insights, and the latest hybrid cloud. It enhances mission-essential capabilities of in-memory operations.

Microsoft launched latest version of this product, SQL Server 2019, on November 2019. SQL Server 2019 provides industry leading security, performance and intelligence over your data, regardless of whether it is structured or unstructured. SQL Server 2019 provides support for Big Data Clusters in SQL Server. Existing SQL Server tools such as database engine, SQL Server Analysis Services, SQL Server Machine Learning Services, SQL Server on Linux, and SQL Server Master Data Services have been enhanced in SQL Server 2019. Using SQL Server 2019 not only helps an organization to store and manage huge amount of information, but also to protect and utilize this data at different locations as required.

3.2 Features and Benefits of SQL Server

Some of the key features and benefits of SQL Server in general include:

Strong Security	Better Performance	Multiple Editions and Pricing Models	Simple and Easy Installation Process
Policy-Based Management to detect security policies that are non-compliant. This feature allows only authorized personnel access to the database. Security audits and events can be written automatically to log files.	SQL Server has built-in transparent data compression feature along with encryption. SQL Server provides access control coupled with efficient permission management tools. It also offers an enhanced performance when it comes to data collection.	Microsoft has made available different editions of SQL Server for different kinds of users. These are also priced accordingly. Thus, from hobbyists to professional developers to enterprise users, there is an edition suitable for each one.	SQL Server is simple to install with a one-click installation procedure and readable GUI having easy instructions for the layman.

3.3 Basic Architecture of SQL Server 2019

There are various components that form a part of SQL Server 2019. All the components come together to form the basic architecture of SQL Server 2019. These components can be represented under three major heads that are shown in figure 3.1.

Figure 3.1: **Architecture of SQL Server**

Tools

There are a number of tools that are provided in SQL Server 2019 for development and query management of a database. The SQL Server Installation Center must be used to install SQL Server program features and tools. Features can also be modified or removed using SQL Server Installation Center. Table 3.1 lists different tools available for SQL Server 2019.

Tool	Description
SQL Server Management Studio (SSMS)	One of the most important tools available in SQL Server 2019 is SSMS. SSMS is a GUI-based application provided with SQL Server 2019 that helps to create databases, database objects, query data, and manage the overall working of SQL Server.
SQLCMD	SQLCMD is a command-line tool that can be used in place of SSMS. It performs similar functions as SSMS, but in command format only.
SQL Server Installation Center	The SQL Server Installation Center tool can also be used to add, remove, and modify SQL Server programs.
SQL Server Configuration Manager	SQL Server Configuration Manager is used by database administrators to manage features of SQL software installed in client machines. This tool is not available to all users. It can be used to configure services, server protocols, client protocols, client aliases, and so on.
SQL Server Profiler	SQL Server Profiler is used to monitor an instance of the Database Engine or Analysis Services.
SQL Server Data Tools (SSDT)	SSDT is an Integrated Development Environment (IDE) used for Business Intelligence Components. It helps to design the database using a tool named Microsoft Visual Studio.
Connectivity Tools	Connectivity tools include DB-Library, Open Database Connectivity (ODBC), Object Linking and Embedding Database (OLE DB), and so on. These tools are used to communicate between the clients, servers, and network libraries.

Table 3.1: **Different Tools in SQL Server 2019**

Services

There are various services that are executed on a computer running SQL Server. These services run along with the other Windows services and can be viewed in the task manager.

Some of the SQL Server 2019 services are as follows:

SQL Server Database Engine

Database Engine is a core service that is used for storing, processing, and securing data. It is also used for replication, full-text search, and Data Quality Services (DQS). It contains tools for managing relational and eXtensible Markup Language (XML) data.

SQL Server Analysis Services (SSAS)

Analysis Services contain tools that help to create and manage Online Analytical Processing (OLAP). This is used for personal, team, and corporate business intelligence purposes. Analysis services are also used in data mining applications.

SQL Server Reporting Services (SSRS)

Reporting Services help to create, manage, publish, and deploy reports. These reports can be in tabular, matrix, graphical, or free-form format. Report applications can also be created using Reporting Services.

SQL Server Integration Services (SSIS)	Integration Services are used for moving, copying, and transforming data using different graphical tools and programmable objects. The DQS component is also included in Integration Services. Integration services help to build high-performance data integration solutions.
SQL Server Master Data Services	Master Data Services (MDS) are used for master data management. MDS is used for analysis, managing, and reporting information such as hierarchies, granular security, transactions, business rules, and so on.

Instances

All the programs and resource allocations are saved in an instance. An instance can include memory, configuration files, and CPU. Multiple instances can be used for different users in SQL Server 2019. Even though many instances may be present on a single computer, they do not affect the working of other instances. This means that all instances work in isolation. Each instance can be customized as per the requirement. Even permissions for each instance can be granted on individual basis. The resources can also be allocated to the instance accordingly, for example, the number of databases allowed.

In other words, instances can be called as a bigger container that contains sub-containers in the form of databases, security options, server objects, and so on.

3.4 Version History of SQL Server

The first version of SQL Server was released in the year 1989. After this, there have been new versions released almost every year, with the latest one being SQL Server 2019. Figure 3.2 depicts the journey of the flagship database product by Microsoft from 1989 to the present.

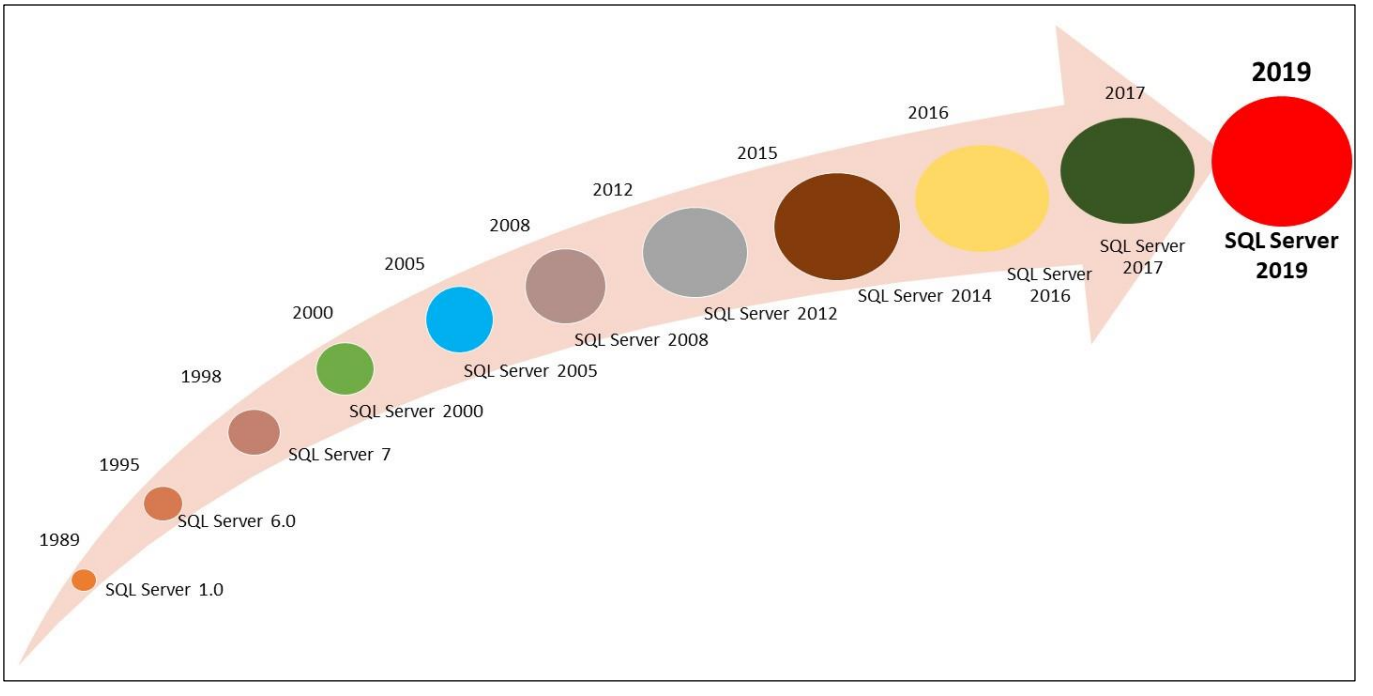


Figure 3.2: Version History of SQL Server

3.5 *Editions of SQL Server*

Based on database requirements, an organization can choose from any of the following editions of SQL Server 2019 that have been released.

The main editions of SQL Server 2019 are as follows:

Express/Web Edition
Free to use and provides an entry-level database for basic Web and mobile apps

Key Features

- Offers up to 16 cores of CPU for compute capacity
- Up to 64 GB of memory for buffer pool
- In-memory OLTP and Columnstore
- End-to-end encryption with secure enclaves
- Support for Linux and Windows containers
- UTF-8 character encoding
- Data classification and auditing

Standard/Web Edition
Provides full featured database for medium tier applications

Key Features

- Offers up to 24 cores of CPU for compute capacity
- Up to 128 GB of memory for buffer pool
- Supports automatic intelligent database tuning
- Azure Data Studio with notebook support
- Supports Big Data Clusters
- Supports Data virtualization by means of PolyBase
- Improved in-memory performance

In addition to Express/Web features

Enterprise/Web Edition
provides full featured database for top tier applications

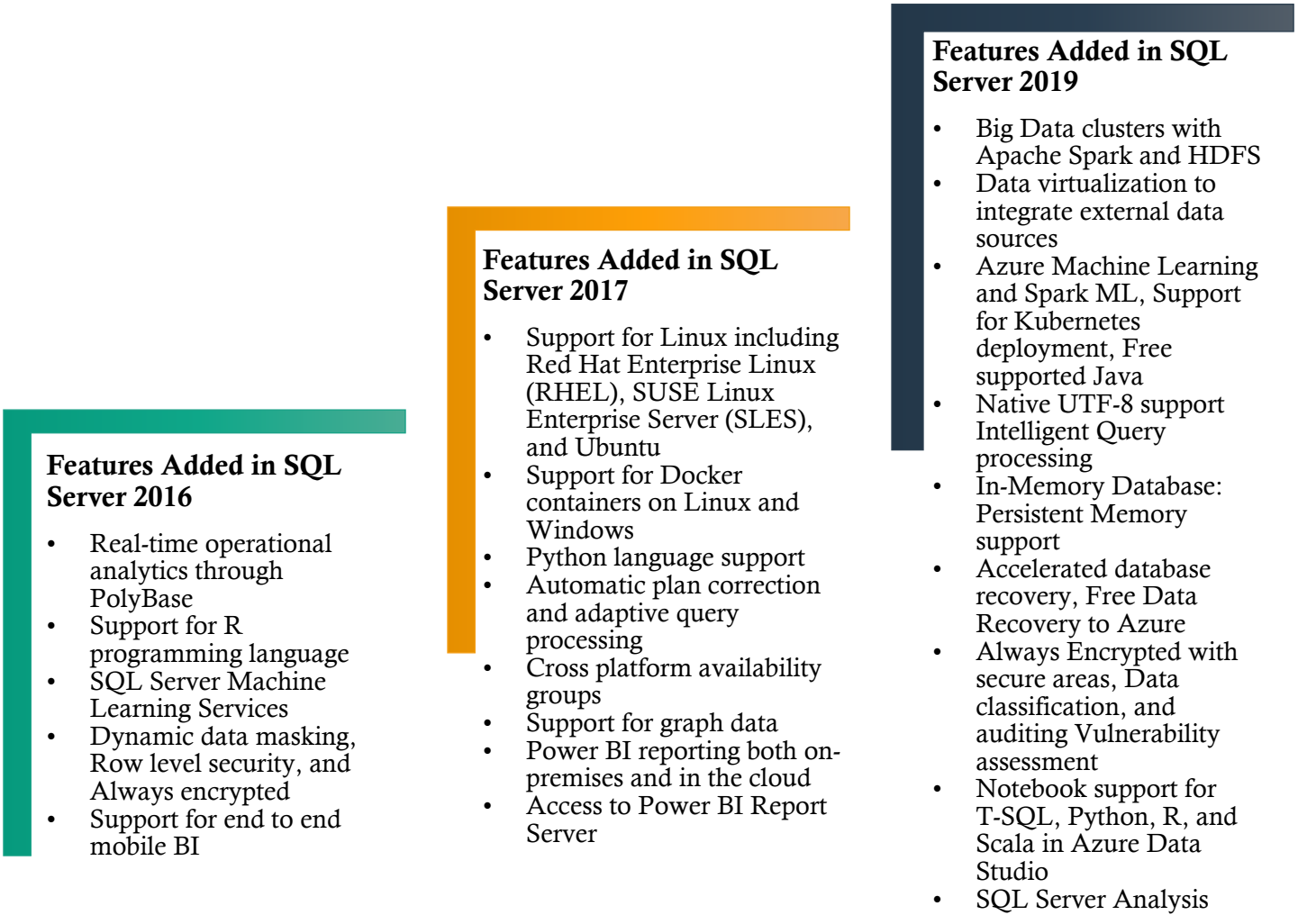
Key Features

- Supports unlimited cores of CPU
- Provides unlimited memory for buffer pool
- Industry-leading performance with unmatched scalability
- Unlimited virtualization benefits
- Access to Power Business Intelligence (BI) Report Server

In addition to Standard and Express/Web features

Developer Edition
Free to use and includes all features of Enterprise edition, licensed for use as a development and test database in a non-production environment.

With each subsequent version, enhancements are made and new features are added.



3.6 Connecting to SQL Server Instances

SQL Server Management Studio (SSMS) is used to connect to SQL Server instances. SSMS is a tool used for creating, querying, and managing the databases. In order to use SSMS, you must first launch it and then, connect to SQL Server 2019 Database Engine by specifying server information and login credentials. The login credentials will include username and password.

Detailed steps to connect to SQL Server instance are as follows:

1. Locate the Microsoft SQL Server Management Studio tool on the list of programs on **Start** menu and start the tool.

2. In the **Connect to Server** dialog box, select the **Server type** as **Database Engine**.

3. Type the **Server name**.

4. Select either **Windows Authentication** or **SQL Server Authentication**, provide the required **Login** and **Password**, and click **Connect**.

Figure 3.3 shows the **Connect to Server** dialog box.

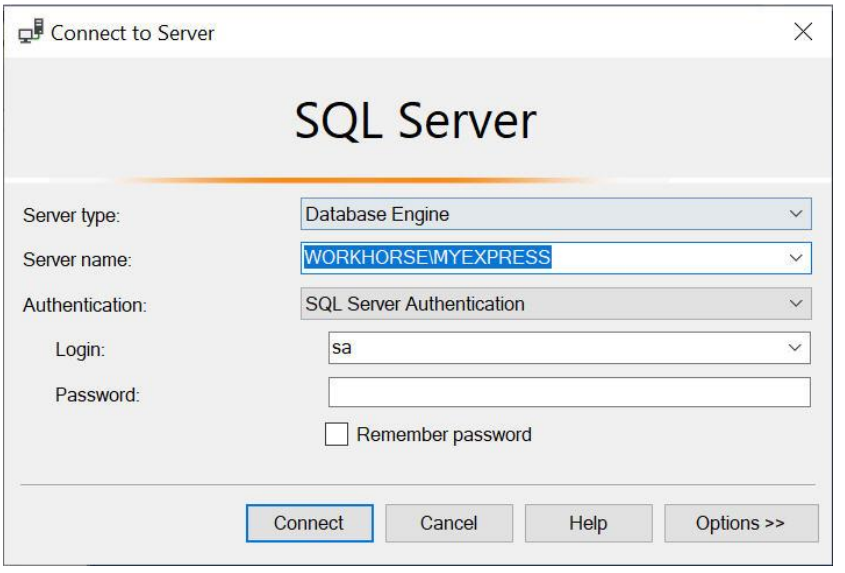


Figure 3.3: **Connect to Server Dialog Box**

Note - The two authentication methods provided by SQL Server are SQL Server Authentication and Windows Authentication. SQL Server Authentication requires a user account for login and password. Hence, multiple user accounts can access the information using their respective usernames and passwords. With Windows Authentication, the operating system credentials can be used to log in to the SQL Server database. This will work only on a single machine and cannot be used in any other computer.

Figure 3.4 shows the SSMS window.

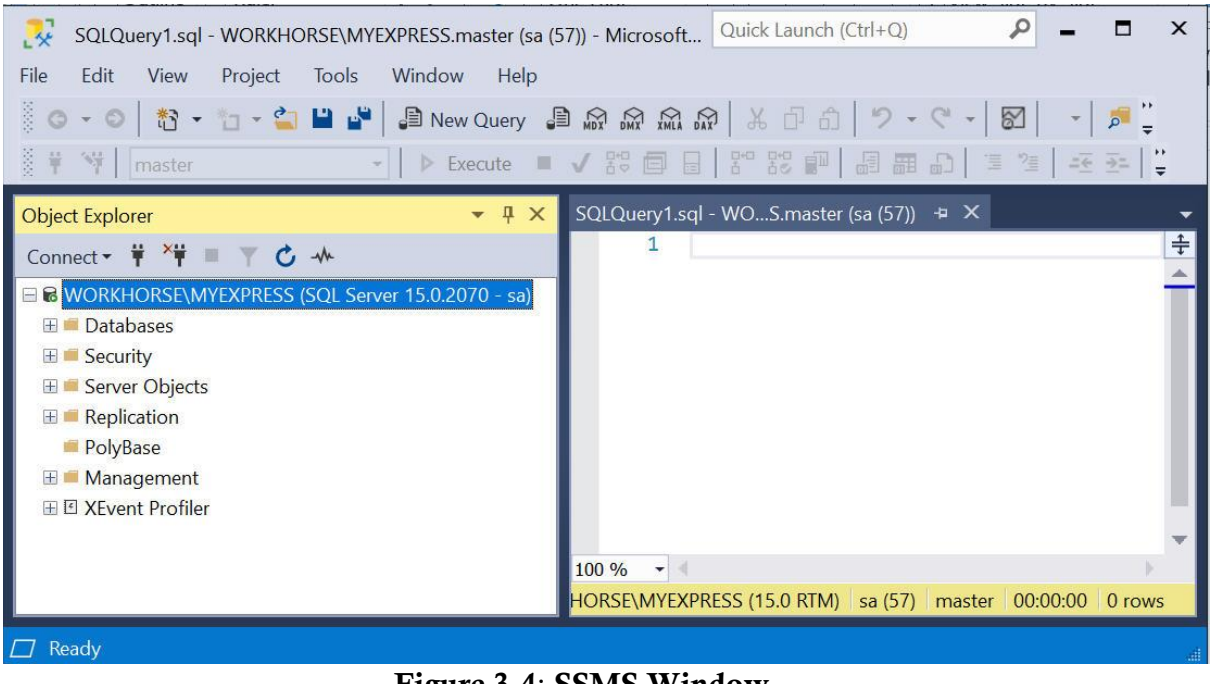


Figure 3.4: **SSMS Window**

3.7 Introduction to Databases

A database is a collection of data stored in data files on a disk or some removable medium. A database consists of data files to hold actual data.

An SQL Server database is made up of collection of tables that stores sets of specific structured data. A table includes a set of rows (also called as records or tuples) and columns (also called as attributes). Each column in the table is intended to store a specific type of information, for example, dates, names, currency amounts, and numbers.

SQL Server 2019 supports three kinds of databases, which are as follows:

3.7.1 System Databases

SQL Server uses system databases to support different parts of the DBMS. Each database has a specific role and stores job information that requires to be carried out by SQL Server. The system databases store data in tables, which contain the views, stored procedures, and other database objects. They also have associated database files (for example, .mdf and .ldf files) that are physically located on the SQL Server machine.

Using SQL Server 2019, users can create their own databases, also called user-defined databases, and work with them. The purpose of these databases is to store user data.

Table 3.2 shows the system databases that are supported by SQL Server 2019.

Database	Description
master	The database records all system-level information of an instance of SQL Server.
msdb	The database is used by SQL Server Agent for scheduling database alerts and various jobs.
model	The database is used as a template for all databases to be created on the particular instance of SQL Server 2019.
resource	The database is a read-only database. It contains system objects included with SQL Server 2019.
tempdb	The database holds temporary objects or intermediate result sets.

Table 3.2: **System Databases**

3.7.2 User-defined Databases

Using SQL Server 2019, users can create their own databases, also called user-defined databases, and work with them. The purpose of these databases is to store user data.

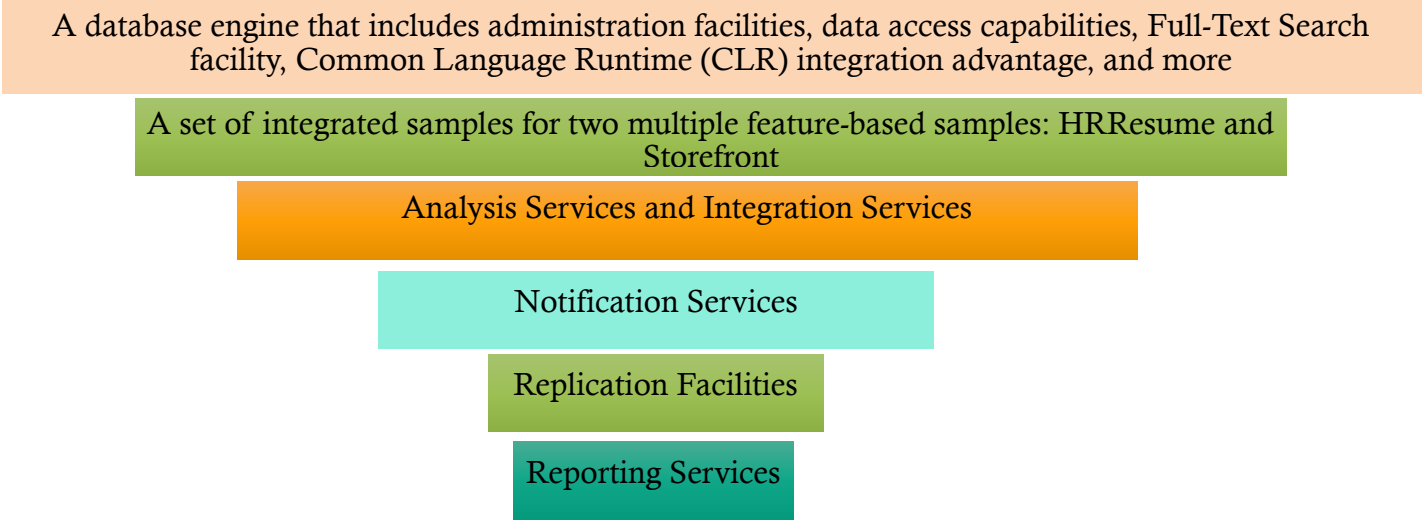
3.7.3 AdventureWorks2019 Sample Database

The sample database, `AdventureWorks`, was introduced from SQL Server 2005 onwards. This database demonstrates use of new features introduced in SQL Server. A fictitious company called Adventure Works Cycles is created as a scenario in this database. Adventure Works Cycles is a large, multinational manufacturing company. The company manufactures and sells metal and composite bicycles to North American, European, and Asian commercial markets. In SQL Server 2019, a new version of the sample database `AdventureWorks2019` is used. A readymade script is available from Microsoft for developers to install this database and populate it with fictitious data.

The `AdventureWorks2019` database schema covers many functional areas for a fictitious bicycle manufacturer. These areas include:

- Customer/sales force automation and analysis
- Human resources
- Purchasing/Vendor Electronic Data Interchange
- Manufacturing work flow

The database comprises several features. Some of its key features are as follows:



The sample database consists of these parts:

- AdventureWorks2019: Sample OLTP database
- AdventureWorks2019DW: Sample Data warehouse
- AdventureWorks2019AS: Sample Analysis Services database

3.8 Understanding the SSMS User Interface

SSMS shows several menu options, toolbars, and panes. The pane on the left seen in figure 3.4 is called Object Explorer. It displays all the objects in the server in a tabular format and providers a user interface to manage them. Capabilities of Object Explorer vary slightly depending on the type of server.

3.8.1 Role and Structure of Object Explorer in SQL Server

The structure of **Object Explorer** in SQL Server 2019 is shown in figure 3.5.

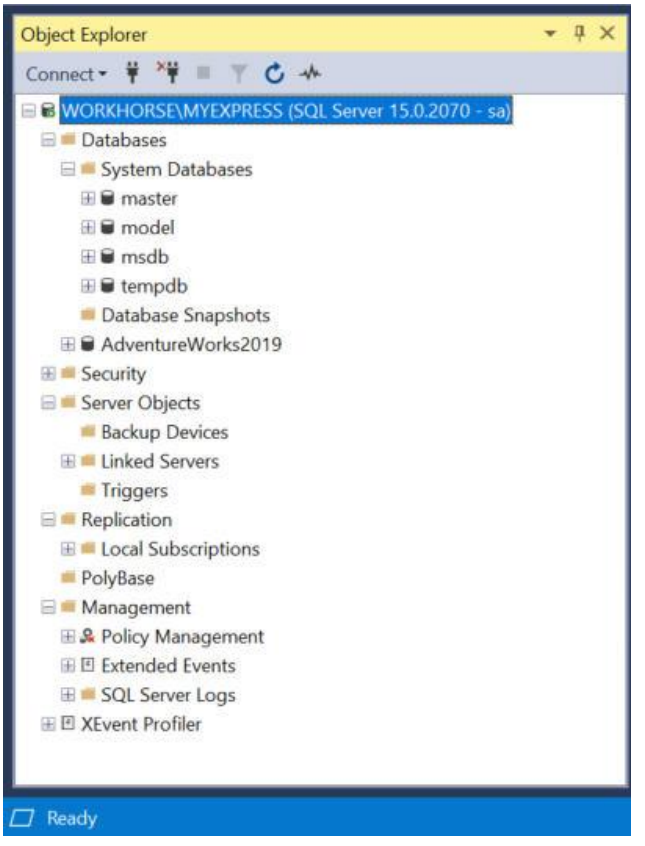
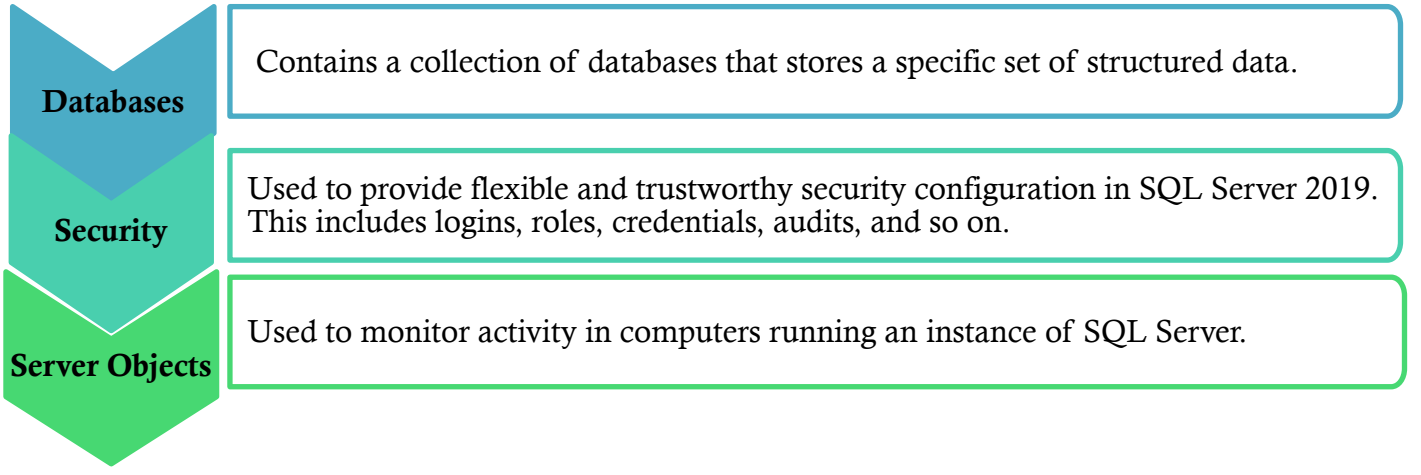
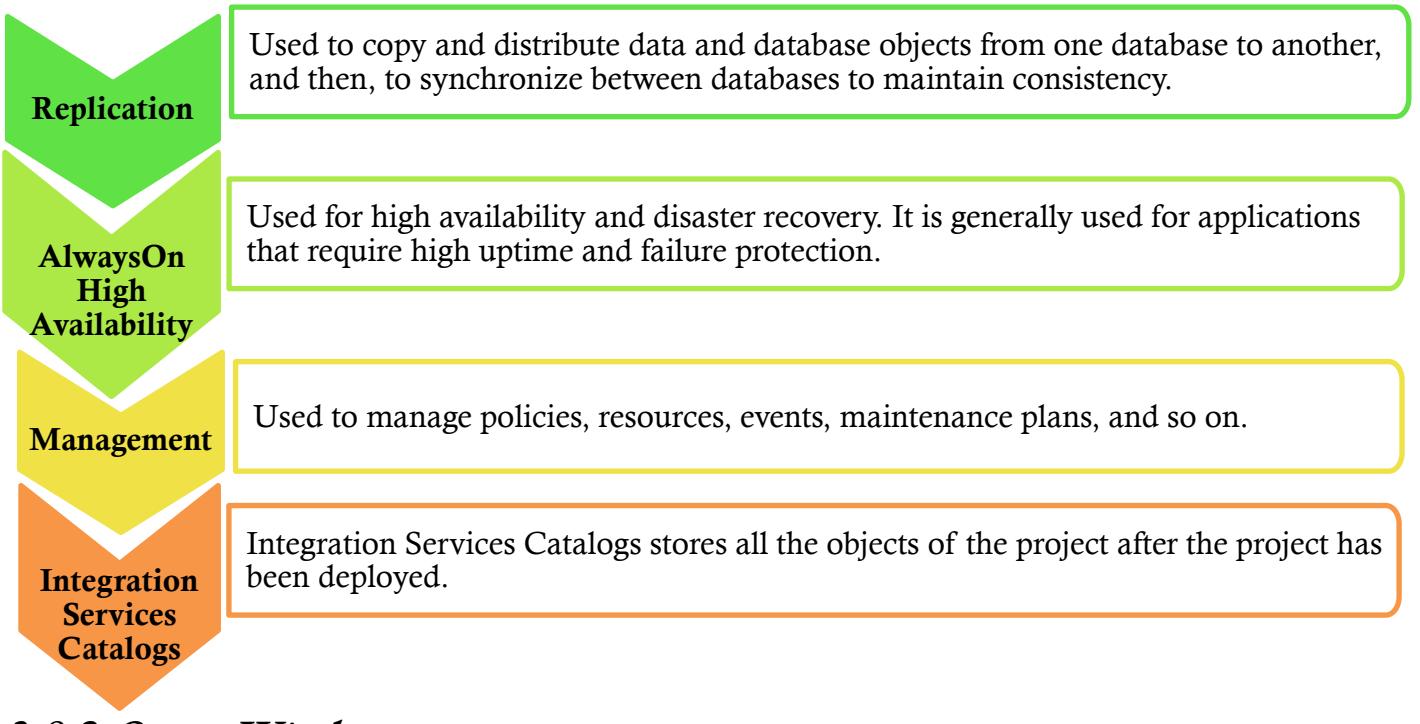


Figure 3.5: Object Explorer

The structure includes databases, security, server objects, replications, and may also show features such as AlwaysOn High Availability, Management, Integration Services Catalogs, and so on. Object Explorer can be accessed through SSMS by connecting to the database server.

Various components in the **Object Explorer** are as follows:





3.8.2 Query Window

Query window is the area where you can type Transact-SQL (T-SQL) queries. Results of your queries also appear in this window. Figure 3.6 shows an example query in the query window.

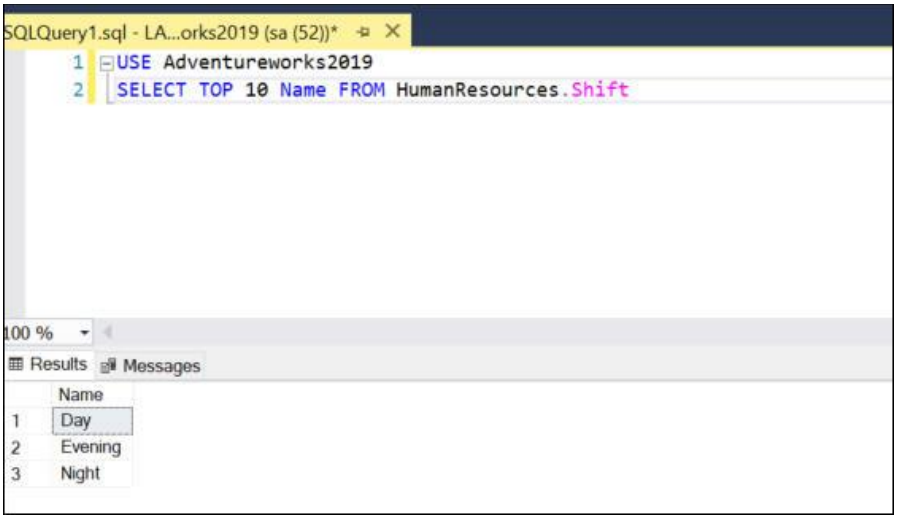


Figure 3.6: Query Window

3.9 Creating and Organizing Script Files

Script files are files that contain a set of SQL commands. A script file can contain one or more SQL statements. The script files are stored in .sql format in SQL Server.

Cconceptual layers in which the script files can be organized are shown in figure 3.7.

Figure 3.7: **Conceptual Layers**

A solution is a file in which all the projects in SQL Server are saved. This acts as a top-most node in the hierarchy. The solution file is stored as a text file with `.ssmssln` extension. A project comes under a solution node. There can be more than one project in SQL Server. All the data related to database connection metadata and other miscellaneous files are stored under a project. It is stored as a text file with `.ssmssqlproj` extension. Script files are the core files in which the queries are developed and executed. The scripts have a `.sql` extension.

Code Snippet 1 depicts a sample script. This script can be saved as **InsertData.sql**.

Code Snippet 1:

```
USE [AdventureWorks2019]
GO
INSERT INTO [Person].[Person]
           ([BusinessEntityID]
           ,[PersonType]
           ,[NameStyle]
           ,[Title]
           ,[FirstName]
           ,[MiddleName]
           ,[LastName]
           ,[EmailPromotion]
           ,[ModifiedDate])
      VALUES(21907
           ,'EM'
           ,0
           ,'Mr.'
           ,'John'
           ,'Gareth'
           ,'Hopkins'
           ,0
           ,'2020-10-10')
GO
```

3.10 Transact-SQL Queries

Queries typed in Transact-SQL and saved as .sql files can be executed directly in the SSMS query window.

Steps to execute Transact-SQL queries are as follows:

1. In the query window, select the code to be executed.

2. On the **SSMS** toolbar, click **Execute**.
   OR
   On the **Query** menu, click **Execute**.
   OR
   Press **F5** or **Alt+X** or **Ctrl+E**.

Refer to figure 3.6 to view the execution of a sample query.

Query results can be displayed in three different formats. The three formats available are grid, text, and file view.

3.11 Check Your Progress

1. Which of the following is a command-line tool in SQL Server?

(A)	SSMS	(C)	SSMSCMD
(B)	SQLCMD	(D)	SQLSSMS

2. Which of these components of Object Explorer is used to monitor activity in computers running an instance of SQL Server?

(A)	Server Objects	(C)	Replication
(B)	Security	(D)	Integration Services

3. What does SSDT stand for?

(A)	SQL Server Deterministic Tools	(C)	SQL Server Diagnostic Tools
(B)	SQL Server Data Tools	(D)	SQL Server Database Tracking

4. Which of the following statements about the tools in SQL Server 2019 are true?
   a. The SQL Server Installation Center tool can be used to add, remove, and modify SQL Server programs.
   b. SQLCMD is an IDE used for Business Intelligence Components. It helps to design the database using Visual Studio.
   c. SQL Server Profiler is used to monitor an instance of the Database Engine or Analysis Services.
   d. SQL Server Installation Center is an application provided with SQL Server 2019 that helps to develop databases, query data, and manage the overall working of SQL Server.

(A)	a and c	(C)	b, c, and d
(B)	a, b, and c	(D)	c and d

5. Which of the following are true about these statements?

Statement 1: Script files are the core files in which the queries are developed and executed.
Statement 2: Script files are files that contain a set of SQL commands.

(A)	Both Statement 1 and 2	(C)	Only Statement 2
(B)	Only Statement 1	(D)	Neither of them

3.11.1 Answers

1.	B
2.	A
3.	B
4.	C
5.	A

- Basic architecture of SQL Server 2019 includes tools, services, and instances.
- Three major editions of SQL Server are Express, Standard, and Enterprise.
- The structure of Object Explorer includes databases, security, server objects, replications, AlwaysOn High Availability, Management, Integration Services Catalogs, and so on.
- SSMS is used to connect to SQL Server instances. SSMS is a tool used for developing, querying, and managing the databases.
- Script files should be stored in .sql format in SQL Server 2019.
- Queries typed in Transact-SQL and saved as .sql files can be executed directly into the SSMS query window.

1. Start SSMS and connect to the Database Engine of SQL Server 2019. Install the AdventureWorks2019 database. Right-click the table **`HumanResources.Employee`** and click **Script Table as → CREATE TO → New Query Window**. Then, right-click again and click **Select Top 1000 Rows** option.

2. Repeat this for the tables **`HumanResources.EmployeeDepartmentHistory`**, **`Person.Person`**, and **`Production.Product`**.

Session - 4
Transact-SQL

Welcome to the Session, **Transact-SQL**. This session explains Transact-SQL and the different categories of Transact-SQL statements. It also explains various data types and elements supported by Transact-SQL. Finally, the session explains set theory, predicate logic, and the logical order of operators in the SELECT statement.

In this session, you will learn to:

- Explain Transact-SQL
- List different categories of Transact-SQL statements
- Explain various data types supported by Transact-SQL
- Explain Transact-SQL language elements
- Explain sets and predicate logic
- Describe logical order of operators in the SELECT statement

4.1 Introduction

SQL is the universal language used in the database world. Most modern RDBMS products use some type of SQL dialect as their primary query language. SQL can be used to create or destroy objects such as tables on the database server and to manipulate those objects, such as adding data into them or retrieving data from them.

Transact-SQL is Microsoft's implementation of the standard SQL. Usually referred to as T-SQL, this language implements a standardized way to communicate to the database. The Transact-SQL language is an enhancement to SQL, the American National Standards Institute (ANSI) standard relational database language. It provides a comprehensive language that supports defining tables, inserting, deleting, updating, and accessing the data in the table.

4.2 Transact-SQL

Transact-SQL is a powerful language offering features such as data types, temporary objects, and extended stored procedures. Scrollable cursors, conditional processing, transaction control, and exception and error-handling are also some of the features which are supported by Transact-SQL.

The Transact-SQL language in SQL Server 2019 provides improved performance, increased functionality, and enhanced features. Enhancements include scalar functions, paging, sequences, meta-data discovery, and better error handling support.

Note: All the queries in this session will make use of the **`AdventureWorks2019`** sample database.