DATABASE PROGRAMMING WITH SQL SERVER

DATABASE FUNDAMENTALS

Before learning **SQL Server** and its **T-SQL execution** of the **SQL language**, it's important that you understand some basic concepts about databases and database technologies. Whether you know it or not, you use databases all the time. Each time you select a name from your **email address book**, or make a search on an Internet search web site, you are using a database. Even when you use your **ATM card** at a **cash machine**, you are using databases for **PIN verification** and **balance checking**. However, even though we all **use databases all the time**, there is still a lot of confusion over the term database. Different people use **similar database terms** to mean **different things**. Before you can use **SQL** to work with an **SQL Server database**, you need to be familiar with the concepts and terms that apply to **database systems**. In particular, you need to understand what a **relational database** is.

Database is a collection of **logically related information** that is organized, stored and electronically accessed while a **Database management system (DBMS)** is a **software suit** that enables users to create, manage, maintain and control access to databases. **People** regularly use the term **database** to mean the **database management system (DBMS)** which is the **database software** they use to create **databases**. A **database** is a container created and manipulated by the **(DBMS) database management system. People** never access databases directly. We always use a **DBMS software** to accesses the database file. An example is **Microsoft Access** which is a **DBMS software** used to generate **Access databases.** A **database** can be used **simultaneously** by many departments or users and it's usually a **shared corporate resource.** A **database** also contains a description of the **operational data**, that is the **meta-data** or the '**data about data'** known as a **system catalog** or **data dictionary**. A **database management system (DBMS)** also enables users to **write queries** for a database to perform the required actions like **retrieving data**, **modifying data**, **deleting data** etc. A **DBMS** also support database **tables** to store data in **rows** (**records** or **tuples**) and **columns** (**fields** or **attributes**). A **relational database management system** (**RDBMS**) is a type of **DBMS** that stores data in the form of related tables based on the relational model.

~~A **database model** is more of a concept than a physical object that is used to create database tables.~~ A **database** is a **structured object** that consists of **data** and **metadata**, with **metadata** being the structured part. **Data** in a database is the actual stored **descriptive information** i.e., all the names and addresses of customers. **Metadata** describes the structure applied by a database to the **customer data.** Therefore, **metadata** is the customer table definition which has the **fields for the names, addresses** etc., **lengths of each of the fields**, and the **datatypes**. A **datatype** restricts values in fields, i.e., allowing only **date** or **number**). A database is represented graphically by a cylindrical disk, as shown here. A database is stored and executed on a database server computer.

WHAT IS DATABASE TABLE?

DBMS databases use tables to store information. A **table** is a structured file that can store related data entries of a specific category represented by **columns** and **rows**. The data stored in a table is of a **one type of data** or one list. You can never store **a list of students** and **a list of fees balance** in the

same database table. If you do so, accurate information access and retrieval becomes difficult. Rather, you must create **two tables**, one for **a list of students** and the other for **a list of fees balance**. Every table in a database has a name that identifies it. That name should always be unique and descriptive, meaning no other table in that database can have the same name.

ID	NAME	AGE	COURSE
1	Ajeet	24	B.Tech
2	aryan	20	C.A
3	Mahesh	21	BCA
4	Ratan	22	MCA
5	Vimal	26	BSC

Student's Table

Tables have characteristics and properties that define how data is stored. These are information about what data may be stored, how it is broken up, how individual pieces of information are named etc. This set of information that describes a table is known as **a schema**. A **schema** are the characteristics and properties that define how data is stored by database tables. **Schemas** are used to describe specific tables within a database, the relationship between tables if any and the entire database. All in all, a schema is Information about the database and table layout and properties.

WHAT IS A ROW OR A RECORD IN A DATABASE TABLE?

A **row** is a collection of related data fields in a database table. A **row** of a table is also called **record**. A **row** has related information of each individual entry in a database table. For example: our table above contains 5 records. Let's have a look at one example of a **record** or **row** in our student's table.

1	Ajeet	24	B.Tech

A Record

WHAT IS A COLUMN IN A DATABASE TABLE?

A **column** is a collection of vertical fields that contains a specific type of data in a database table. **Every column** in a database table grid contains a particular piece of data. In the above student's database table, for example, one column contains the student's **Identification Number**, another contains the student's **Name**, another contains the **age**, and another contains the **course** all stored in their own specific columns. It's very important to correctly break information into separate columns such as, **ID**, **NAME**, **AGE** and **COURSE.** By doing this, it becomes very possible to **sort** or **filter data** by specific columns (e.g., to find all students in a course or of specific age).

NAME
Ajeet
aryan
Mahesh
Ratan
Vimal

A Column

Every table column in a database has an associated datatype. A **datatype** defines what type of data a particular column can contain. E.g., if the column is to contain a number (perhaps the age of students), the datatype would be numeric. If the column were to contain dates, text, notes, currency amounts, and so on, the appropriate datatype would be used to specify this. **Every table column** has an associated datatype that **restricts** (or **allows**) specific data in a particular column. **Datatypes** restrict the type of

data that can be stored in a column (for example, preventing the entry of alphabetical characters into a numeric field). Datatypes also help sort data correctly, and they play an important role in optimizing disk usage. As such, special attention must be given to picking the right datatype when tables are created.

WHAT IS A FIELD IN A DATABASE TABLE?

A **field** is the smallest single piece of data in a database table. Every data field in a database table can define the characteristics of its data as alphabetical characters, numbers, date and or time etc.

A Field

WHAT IS A NULL VALUE IN A DATABASE TABLE?

A **NULL value** of the table specifies that the field has been left blank during record creation. It is totally different from the value filled with zero or a field that contains space.

WHAT IS A PRIMARY KEY IN A DATABASE TABLE?

A **primary Key** is a field or column that uniquely identifies a specific record or row in a database table. Every record or row in a database table should have some field or column that uniquely identifies it. A **database table** containing students may use a student's Identification Number field or column for this purpose. An employee table may use an employee ID or the employee Social Security number column. A **primary key** is used to refer to a specific row or record. Without a primary key, updating or deleting specific rows in a table becomes extremely difficult because there is no guaranteed safe way to refer to just the rows to be affected. Any **field** or **column** in a database table can be set as the primary key, as long as it meets the following conditions: ●No two rows can have the same primary key value. ●Every row must have a primary key value (primary key columns may not allow NULL values). There is another very important type of key called **a foreign key**, but I'll get to that later.

WHY USE A DATABASE?

The following are some of the reasons why you would use databases:

- **Compactness:** Databases help you maintain large amounts of data and thus completely replace voluminous paper files.
- **Speed:** Searches for a particular piece of data or information in a database are much faster than sorting through piles of paper.
- **Less drudgery:** It is a dull work to maintain files by hand; using a database completely eliminates such maintenance.
- **Currency:** Database systems can easily be updated and so provide accurate information all the time and on demand.

DATABASE MANAGEMENT SYSTEMS (DBMS)

Database management system (DBMS) is a software suit or a collection of programs that aids users to create, manage, maintain and control access to databases. **DBMSs** internally consist of a collection of programs each executing different tasks related to database management. Some **DBMS** examples, My SQL, Oracle, System 2000, MS Access, My SQL server, etc.

ADVANTAGES OF DATABASE MANAGEMENT SYSTEMS (DBMSs)

- DBMS offers a variety of techniques to store and retrieve data.
- DBMS serves as an efficient handler to balance the needs of applications using the same data.
- DBMS offers similar administration procedures for data.
- Application programmers are never exposed to details of data representation and storage.
- DBMS use various powerful functions to store and retrieve data efficiently.
- DBMS offers offer Data Integrity and Security.
- DBMS offers integrity constraints to get a high level of protection against barred access to data.
- DBMS schedules concurrent access to the data in such a way that only one user can access the same data at a time.
- Reduced Application Development Time
- **Controls database redundancy**: It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing**: In DBMS, an authorized users of a company can share data among many users.
- Easy Maintenance: is easily maintainable due to the centralized nature of the database system.
- Reduced time: It reduces development time and maintenance need.
- Backup: It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- Multiple user interface: It provides different types of user interfaces like graphical user interfaces, application program interfaces

DISADVANTAGE OF DATABASE MANAGEMENT SYSTEMS (DBMS)

DBMS may offer plenty of advantages but, it has certain flaws-

- Cost of Hardware and Software of a DBMS is quite high which increases the budget of business.
- Most DBMS are often difficult systems, so training users to use the DBMS is required.
- In some organizations, all data is integrated into a single database which can be damaged because of electric failure or a database is corrupted on the storage media.
- Use of the same program at a time by many users sometimes lead to the loss of some data.
- DBMS can't perform sophisticated calculations.
- Size: It occupies a large space of disks and large memory to run them efficiently.

DBMS ALLOW USERS TO DO THE FOLLOWING TASKS:

- Data Definition: It is used for creation, modification, and removal of definition that defines the organization of data in the database.
- Data Updating: It is used for insertion, modification, and deletion of the data in the database.
- Data Retrieval: It is used to retrieve the data from the database which can be used by applications for various purposes.
- User Administration: It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

CHARACTERISTICS OF DATABASE MANAGEMENT SYSTEMS (DBMS)

- DBMS uses a digital repository established on a server to store and manage the information.
- DBMS can provide a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.

- DBMS contains **ACID properties** which maintain data in a healthy state in case of failure.
- DBMS can reduce the complex or hard relationships between table data.
- DBMS is used to support manipulation and processing of data.
- DBMS is used to provide data security at all levels.
- DBMS can view databases from different viewpoints according to the requirements of the user.

EXAMPLES OF HOW DATABASE SYSTEMS ARE USED

- **PURCHASES FROM SUPERMARKETS:** When you buy goods from your local supermarket, it's likely that a **database** is accessed. A **checkout assistant** uses a **bar code reader** to scan each of your purchases. That **bar code reader** is linked to an application program that uses the bar code to find the price of the item from the **products database**. The application program then reduces the number of such items in stock and displays the price on the **cash register.** If the **reorder level** falls below a specified threshold, the **database system** may automatically place an order to obtain more stocks of that particular item.

- **PURCHASES USING YOUR CREDIT CARDS:** When you purchase goods using your credit card, the assistant usually checks to find out if you have **sufficient credit left to make the purchase**. The check may be carried out by telephone or it may be done automatically by a **card reader** linked to a computer system. In either case, there is **a database** somewhere that holds information about the **purchases** that you've made using **your credit card**. To check your credit, there is a database application program that uses your credit card number to check that the price of the goods you wish to buy, together with the sum of the purchases you have made, is within your credit limit. When the purchase is confirmed, the details of your purchase are then added to a database. The application program also accesses the database to check that the credit card is not on the list of stolen or lost cards before authorizing the purchase. There are other database application programs that send out monthly statements to each cardholder and credit accounts when payment is received.

- **BOOKING A HOLIDAY AT, THE TRAVEL AGENTS:** When you make inquiries about a holiday, the travel agent may access several databases containing holiday and flight details. When you book a holiday, a **database system** makes all the booking arrangements. **Database system** ensure that two different agents don't book the same holiday or overbook the seats on the flight. E.g., if there is only one seat left on the flight from **Uganda** to **New York** and two agents try to reserve the last seat at the same time, the **database system** has to recognize this situation and only allows one booking to proceed, and also inform the other agent that there are now no seats available. The travel agent may have another, usually separate, database for invoicing.

- **USING A LIBRARY:** When you visit any **library**, there is **a database** with all the books particulars in that library, details of the readers, made reservations etc. There will be a computerized index that allows readers to find a book based on **its title**, or **its authors**, or **its subject area**, or **it's ISBN**. A **database system** handles reservations to allow a reader to reserve a book and to be informed by post when a book is available. A database system also sends out reminders to borrowers who have failed to return books on the due date. Typically, the database system will have a bar code reader, similar to that used by the supermarket described earlier, which is used to keep track of books coming in and going out of the library.

- **RENTING A VIDEO:** When you rent a video from a **video rental company**, you will find that the

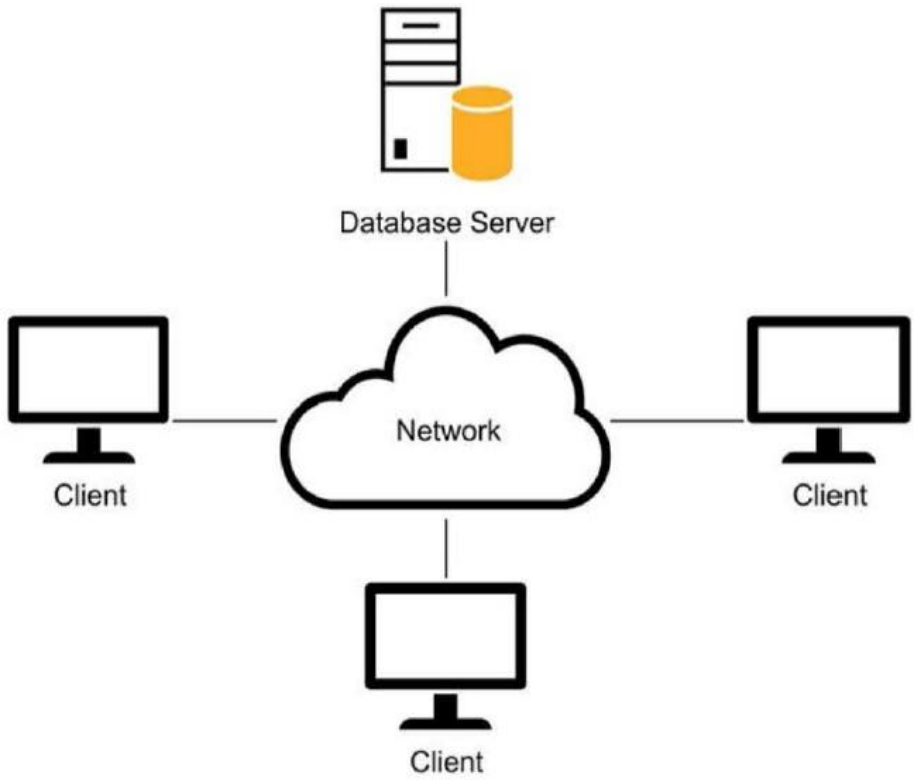
company maintains **a database** consisting of the **video titles** that it stocks, details on the copies it has for **each title**, whether the copy is available for rent or whether it is currently on loan, details of its members (the renters) and which videos they are currently renting and date they are returned. The **database** may even store more detailed data on each video, e.g., **its director** and its **actors**.

●**USING THE INTERNET:** Many **websites** on the Internet are driven by database applications. E.g. you may visit an **online bookstore** that allows you to browse and buy books, like **Amazon.com.** It allows you to browse books in different categories, such as **computing** or **management**, or it may allow you to browse books by author name. In either case, there is a database on the organization's **Web server** that has book details, availability, shipping information, stock levels, and on-order data. Book details include **book titles**, **ISBNs**, **authors**, **prices, sales histories, publishers, reviews**, and in-depth descriptions. The database allows books to be **cross-referenced**: e.g., a book may be listed under several categories, i.e., **computing**, programming languages, bestsellers, and indorsed titles. **Cross-referencing** also allows Amazon to give data on other books that are ordered along with the title you are interested in. ●**BANKING:** For customer data, account activities, payments, deposits, loans, etc. ●**AIRLINES:** For reservations and schedule information. ●**UNIVERSITIES:** For students' data, course registrations, colleges and grades. ●**TELECOMMUNICATION:** It keep call records, monthly bills, maintaining balances etc. ●**SALES:** Use for storing customer, product & sales information.

AN INTRODUCTION TO CLIENT / SERVER SYSTEMS

1.THE HARDWARE COMPONENTS OF A CLIENT/SERVER SYSTEM

Clients are the computers that access shared resources and services from the server. A **network** includes the cabling, communication lines, network interface cards, hubs, routers, and other components that connect clients and the server. The **server**, usually referred to as the **database server**, is a computer with high processing speed, internal memory (RAM), and disk storage to store the files and databases of a system and offer services to the clients. A **server** is a high-powered computer, but it can also be a midrange system like **an IBM Power System** or **a Unix system**, or even **a mainframe system**. To **back up** the files of a client/server system, a server usually has a disk drive or some other form of offline storage. It often has one or more printers or specialized devices that can be shared by the users of the system. And it can provide programs or services like e-mail that can be accessed by all the users of the system. In a simple client/server system, clients and the server are part of a local area

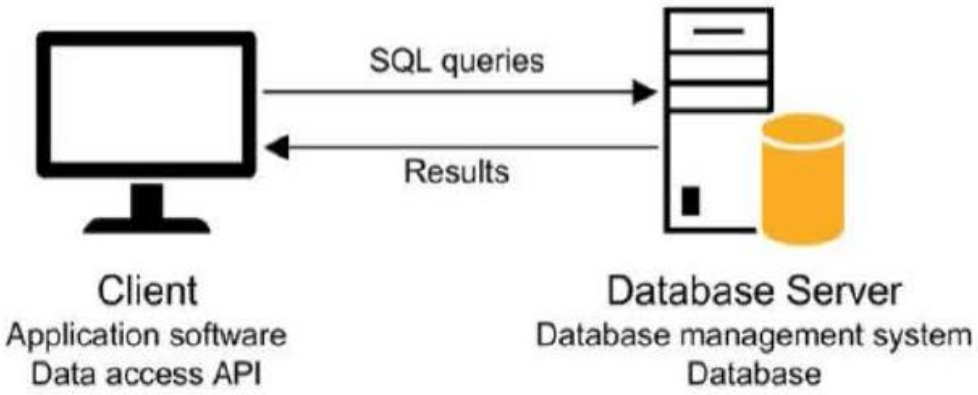


A Client/Server System

network (LAN). But, two or more LANs that reside at separate geographical locations can be linked as part of a larger network i.e., a wide area network (WAN). In addition, individual systems or networks can be connected over the Internet.

2.THE SOFTWARE COMPONENTS OF A CLIENT/SERVER SYSTEM.

In addition to a network operating system that manages the functions of the network, the server requires a database management system (DBMS) like **Microsoft SQL Server** or **Oracle**. This DBMS manages the databases that are stored on the server.



In contrast to a server, clients require application software to perform useful work. This can be a bought software package like a financial accounting package, or it can be custom software that's developed for a specific application. Although the application software is run on the client, it uses data that's stored on the server. To do that, it uses a data access **API** (application programming interface) e.g., **ADO.NET**. Since the method you use to work with an API depends on the programming language and the API you're using, Instead, you'll learn about the standard language called **SQL**, or **Structured Query Language**, that allows any software application to communicate with any **DBMS**. (In conversation, SQL is pronounced as either **S-Q-L** or **sequel**.) Once the software for both the client computer and server is installed, the client communicates with the server via **SQL queries** (or just **queries**) that are passed to the **DBMS** through the **API**. When the client computer sends a query to the **database management system (DBMS)**, the **DBMS** interprets the query and sends the results back to the client computer. As you can see in the figure provided, the processing done by **a client/server system** is divided between the client computers and the server computer. Here, the **DBMS** on the server processes the requests made by the application running on the client computer.

Server Software

- To store and manage the databases of the client/server system, each server needs a database management system (DBMS) e.g., **Microsoft SQL Server**.
- The processing that's done by the DBMS is typically referred to as back-end processing, and the database server is referred to as the **back end**.

Client Software

- The application software does the work that the user wants to do. This type of software can be purchased or developed.
- The data access **API** (application programming interface) provides the interface between the application program and the DBMS. The current Microsoft API is **ADO.NET**, which can communicate directly with **SQL Server**. Older APIs required a data access model, such as ADO or DAO, plus a driver, such as OLE DB or ODBC.
- The processing that's done by the client software is typically referred to as front-end

processing, and the client is typically referred to as the front end.

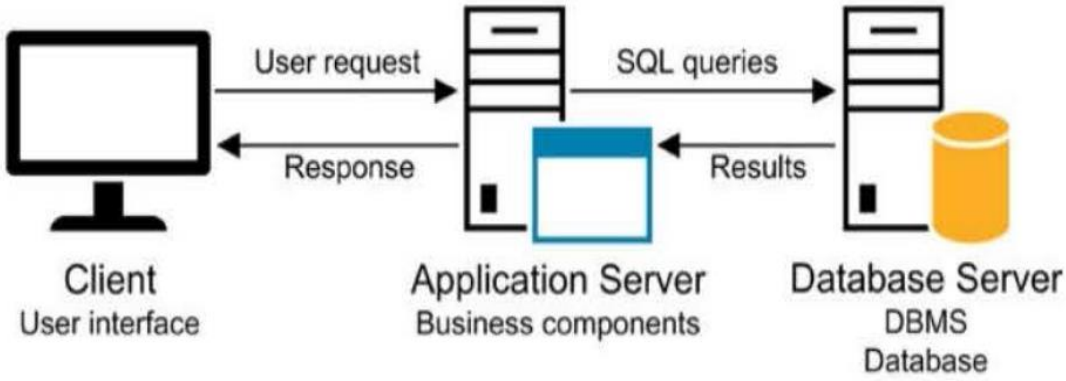
The SQL Interface

- The application software communicates with the DBMS by sending SQL queries through the data access **API**. When the DBMS receives a query, it provides a service like returning the requested data (the query results) to the client.
- SQL stands for Structured Query Language, which is the standard language for working with a relational database.

3.OTHER CLIENT/SERVER SYSTEM ARCHITECTURES.

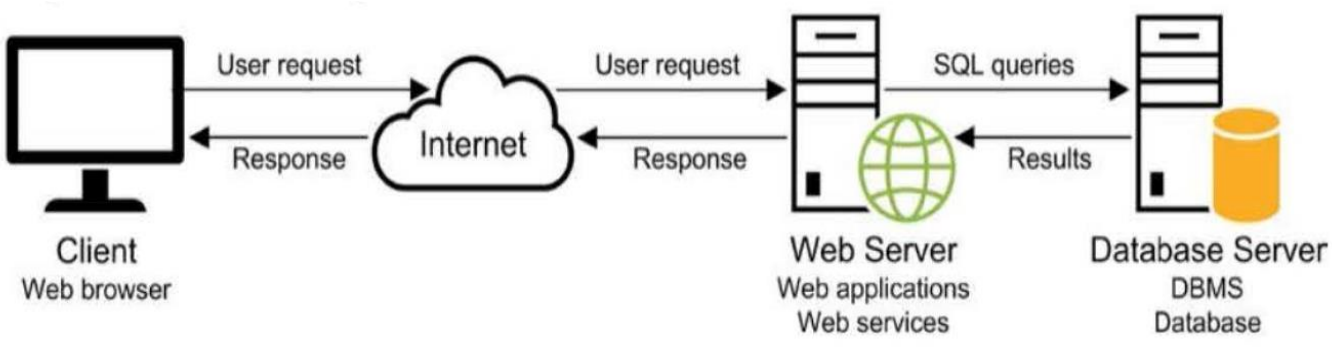
In its simplest form, **a client/server system** involves of a **single database server** and one or more clients. Many client/server systems today, though, include other servers. In figure below, you can see two client/server systems that include an additional server between the clients and the database server.

The diagram below is for a simple **Windows-based system**. With this system, only the user interface for an application runs on the client computer. The rest of the processing that's done by the application is stored in one or more business components on **the application server**. The client computer sends requests to the application server for processing. If the request involves accessing data in a database, the application server formulates the appropriate query and passes it on to the database server. The results of the query are then sent back to the application server, which processes the results and sends the appropriate response back to the client.



A Windows-based System that uses an Application Server

Similar processing is also done by the **web-based system**, as shown by the second example. Here, a **web browser** running on **client computers** is used to send requests to a **web application running** on a web server somewhere on the Internet. The web application, in turn, can use the web services to perform some of its processing. Then, the web application or web service can pass requests for data on to the database server.



A Simple Web-based System

In a **Windows-based system**, e.g., **business components** can be distributed over several application servers, and those components can communicate with databases on many database servers. Similarly, **web applications and services** in a web-based system can also be distributed over numerous web servers that access numerous database servers. In most cases, it's not necessary for you to know how a system is configured to use **SQL**. You should know that client/server systems aren't the only systems that support **SQL**. E.g., **old mainframe systems** and **newer thin client systems** also use **SQL**. Unlike client/server systems, most of the processing for these types of systems is done by a **mainframe** or **another high-powered machine**. Computers that are linked to the system do little or no work.

DESCRIPTION

- In addition to a database server and clients, a client/server system can include additional servers, such as application servers and web servers.
- Application servers are typically used to store business components that do part of the processing of the application. In particular, these components are used to process database requests from the user interface running on the client.
- Web servers are used to store web applications and web services. Web applications are applications that are designed to run on a web server. Web services are like business components, except that, like web applications, they are designed to run on a web server.
- In a web-based system, a web browser running on a client sends a request to a web server over the Internet. Then, the web server processes the request and passes any requests for data on to the database server.
- More complex system architectures can include two or more application servers, web servers, and database servers.

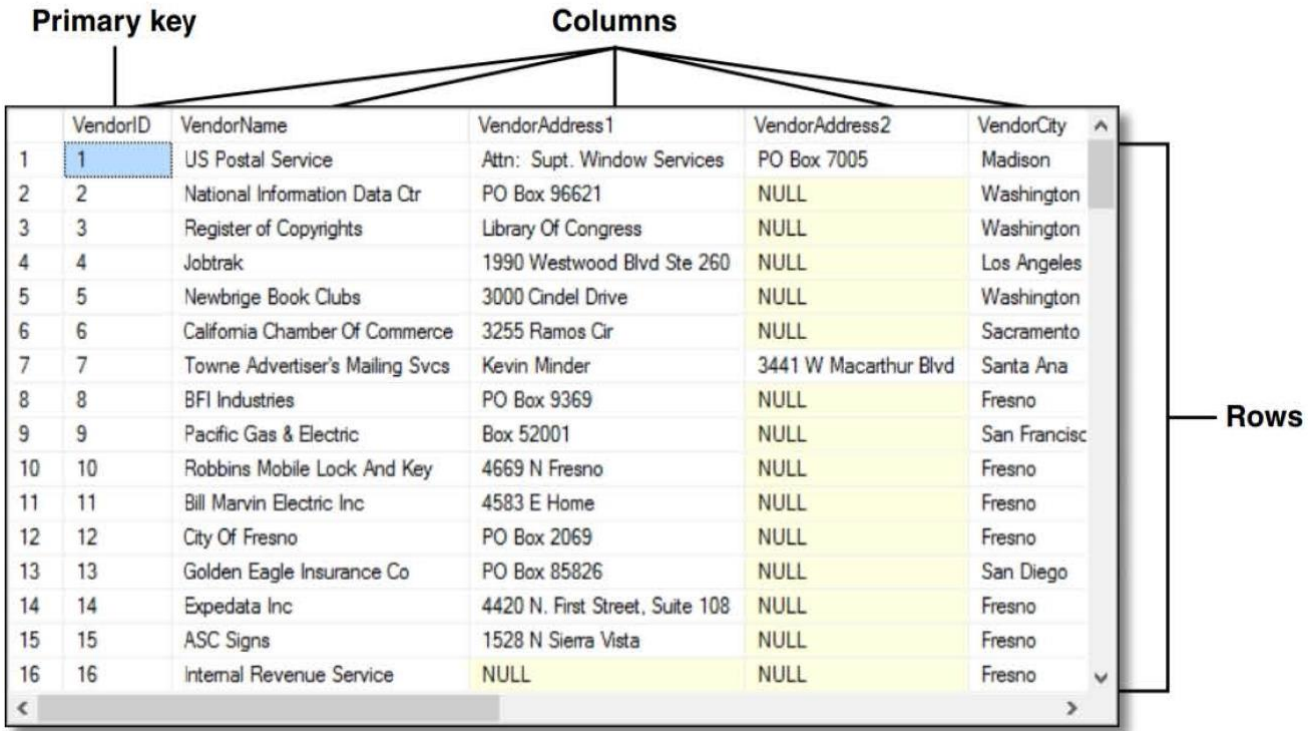
AN INTRODUCTION TO THE RELATIONAL DATABASE MODEL

In **1970**, **Dr. E. F. Codd** developed a model for a new type of database called a relational database. This database type eliminated some of the problems that were associated with standard files and other database designs. By using the relational model, you can reduce data redundancy, which saves disk storage and leads to efficient data retrieval. You can similarly view and manipulate data in a way that is both intuitive and efficient. At the moment, relational databases are the **de facto** standard for database applications.

HOW A DATABASE TABLE IS ORGANIZED

The model for a **relational database** state that data is stored in one or more tables. It also states that each table can be seen as a two-dimensional matrix consisting of rows and columns. This is illustrated by the relational table below. Each row in this table contains data about a single vendor. In practice, the rows and columns of a relational database table are regularly referred to by the more traditional terms, **records** and **fields**. We shall use the terms **rows** and **columns** since these are the terms used by SQL Server. In general, each table is **modeled** after a **real-world entity** such as **a vendor** or an invoice. Then, the **columns** of the table represent the attributes of the entity such as name, address, and phone number. And each **row** of a table represents one instance of an entity. A **value** is stored in a field at the intersection of each row and column, sometimes called a **cell**. If a table contains one or more columns that uniquely identify each row in the table, you can define these columns as the **primary key** of the table. For example, the primary key of the **Vendors table** in the figure below is the **VendoriD column**. In this example, the primary key consists of a single column. But a primary key can also consist of two

or more columns, in which case it's called a **composite primary key**. In addition to primary keys, some database management systems let you define additional keys that uniquely identify each row in a table. If, for example, the **VendorName column** in the **Vendors table** contains unique data, it can be defined as a non-primary key. In **SQL Server**, this is called a **unique key**. **Indexes** provide an efficient way of accessing the rows in a table based on the values in one or more columns. Since applications typically access rows in a table by referring to their key values, an index is automatically created for each key you define.



	VendorID	VendorName	VendorAddress1	VendorAddress2	VendorCity
1	1	US Postal Service	Attn: Supt. Window Services	PO Box 7005	Madison
2	2	National Information Data Ctr	PO Box 96621	NULL	Washington
3	3	Register of Copyrights	Library Of Congress	NULL	Washington
4	4	Jobtrak	1990 Westwood Blvd Ste 260	NULL	Los Angeles
5	5	Newbrige Book Clubs	3000 Cindel Drive	NULL	Washington
6	6	California Chamber Of Commerce	3255 Ramos Cir	NULL	Sacramento
7	7	Towne Advertiser's Mailing Svcs	Kevin Minder	3441 W Macarthur Blvd	Santa Ana
8	8	BFI Industries	PO Box 9369	NULL	Fresno
9	9	Pacific Gas & Electric	Box 52001	NULL	San Francisc
10	10	Robbins Mobile Lock And Key	4669 N Fresno	NULL	Fresno
11	11	Bill Marvin Electric Inc	4583 E Home	NULL	Fresno
12	12	City Of Fresno	PO Box 2069	NULL	Fresno
13	13	Golden Eagle Insurance Co	PO Box 85826	NULL	San Diego
14	14	Expedata Inc	4420 N. First Street, Suite 108	NULL	Fresno
15	15	ASC Signs	1528 N Sierra Vista	NULL	Fresno
16	16	Internal Revenue Service	NULL	NULL	Fresno

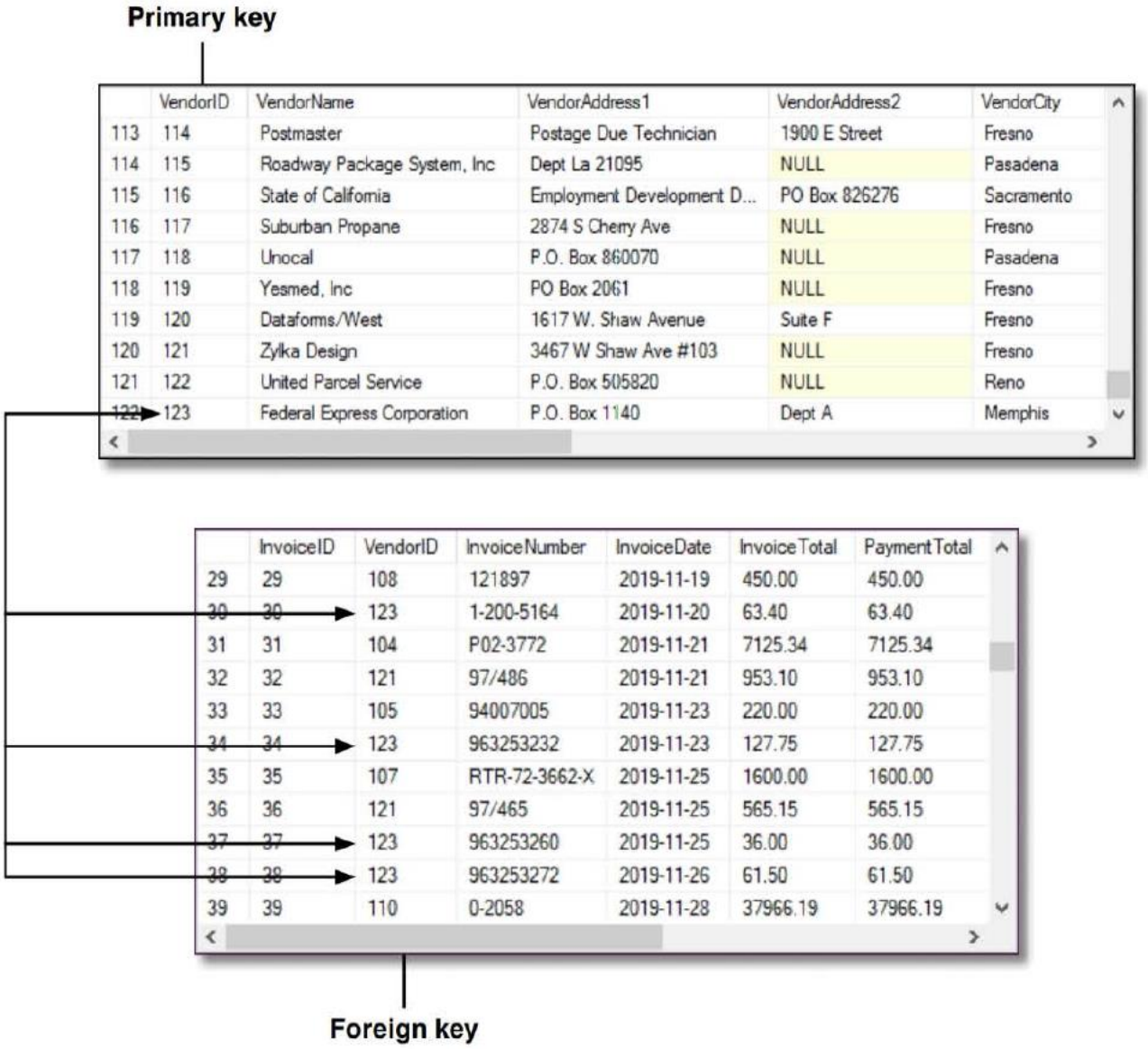
GENERAL CONCEPTS

- A **relational database** consists of related tables with relationships. Tables consist of rows and columns, which can also be referred to as records and fields.
- A **table** is typically modeled after a real-world entity, such as an invoice or a vendor.
- A **column** represents a particular attribute of the entity, such as a vendor's address.
- A **row** holds a set of values for a single instance of the entity, such as a vendor.
- The intersection of a row and a column is sometimes called a **cell**. A **cell** stores a single value.
- Most tables have **a primary key** that uniquely identifies each row in the table. A primary key is a single column, but it can also consist of two or more columns. If a primary key uses two or more columns, it's called a **composite primary key**.
- In addition to primary keys, some database management systems let you define one or more non-primary keys. In SQL Server, these keys are called unique keys. Like a primary key, a non-primary key uniquely identifies each row in the table.
- A table can also be defined with one or more indexes. An index provides an efficient way to access data from a table based on the values in specific columns. An index is automatically created for a table's primary and non-primary keys.

HOW TABLES IN A RELATIONAL DATABASE ARE RELATED

The tables in a **relational database** can be related to other tables by values in specific columns. The two tables shown in the figure below illustrate this relationship concept. Here, each row in the Vendors table is related to one or more rows in the Invoices table. This is called a **one-to-many relationship**.

Typically, relationships exist between the primary key in one table and the foreign key in another table. The **foreign key** is simply one or more columns in a table that refer to a primary key in another table. In **SQL Server**, relationships can also exist between a **unique key** in one table and a **foreign key** in another table. Although one-to-many relationships are the most common, two tables can also have a **one-to-one** or **many-to-many relationship**. If a table has a one-to-one relationship with another table, the data in the two tables could be stored in a single table. But, it's often useful to store large objects such as images, sound, and videos in a separate table. Then, you can join the two tables with the one-to-one relationship only when the large objects are needed. By contrast, a many-to-many relationship is usually applied by using an intermediate table that has a one-to-many relationship with the two tables in the many-to-many relationship.



	VendorID	VendorName	VendorAddress1	VendorAddress2	VendorCity
113	114	Postmaster	Postage Due Technician	1900 E Street	Fresno
114	115	Roadway Package System, Inc	Dept La 21095	NULL	Pasadena
115	116	State of California	Employment Development D...	PO Box 826276	Sacramento
116	117	Suburban Propane	2874 S Cherry Ave	NULL	Fresno
117	118	Unocal	P.O. Box 860070	NULL	Pasadena
118	119	Yesmed, Inc	PO Box 2061	NULL	Fresno
119	120	Dataforms/West	1617 W. Shaw Avenue	Suite F	Fresno
120	121	Zylka Design	3467 W Shaw Ave #103	NULL	Fresno
121	122	United Parcel Service	P.O. Box 505820	NULL	Reno
122	123	Federal Express Corporation	P.O. Box 1140	Dept A	Memphis

	InvoiceID	VendorID	InvoiceNumber	InvoiceDate	InvoiceTotal	PaymentTotal
29	29	108	121897	2019-11-19	450.00	450.00
30	30	123	1-200-5164	2019-11-20	63.40	63.40
31	31	104	P02-3772	2019-11-21	7125.34	7125.34
32	32	121	97/486	2019-11-21	953.10	953.10
33	33	105	94007005	2019-11-23	220.00	220.00
34	34	123	963253232	2019-11-23	127.75	127.75
35	35	107	RTR-72-3662-X	2019-11-25	1600.00	1600.00
36	36	121	97/465	2019-11-25	565.15	565.15
37	37	123	963253260	2019-11-25	36.00	36.00
38	38	123	963253272	2019-11-26	61.50	61.50
39	39	110	0-2058	2019-11-28	37966.19	37966.19

A **Relational Database Management System (RDBMS)** is a collection of software programs and capabilities that enables users to create, maintain, modify, and manipulate a relational database. Most commercial **RDBMSs** use **SQL**. The most basic **RDBMS** functions are related to create, read, update and delete operations, collectively known as the CRUD cycle.

Features of RDBMS:

- An RDBMS is easily accessible using SQL commands.
- An RDBMS provides full data independence.
- The basic unit of data storage in a relational database is called a table.
- A table consists of tuples/rows/records and each record has one or more columns used to store values.
- In RDBMS, we can use conditional operations such as joins and restrictions.
- An RDBMS enables data sharing between users.
- Also at the same time, you can ensure consistency of data across multiple tables by using integrity constraints.
- An RDBMS minimizes the redundancy of data.

Features of SQL:

- High performance
- High availability
- Easy to learn and use
- Robust transactional support
- Functionally complete
- Highly secure
- Comprehensive application development
- Management ease

Advantages of RDBMS:

- Support for a very large database.
- Automatic optimization of searching (when possible).
- RDBMS has a simple view of the database that conforms to much of the data used in businesses.
- RDBMS uses Structured Query Language.
- Easy extendibility, as new data may be added without modifying existing records this is also known as scalability.
- RDBMS has data security which is critical when data sharing is based on privacy.
- RDBMS defines how the data is organized and how the relations among them are associated.
- It defines the entities and relationships among them. It contains a descriptive detail of the database.

Disadvantages of RDBMS:

- No support for complex objects such as documents, video, images.
- Often poor support for storage of complex objects.
- Still no efficient and effective integrated support.

- 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

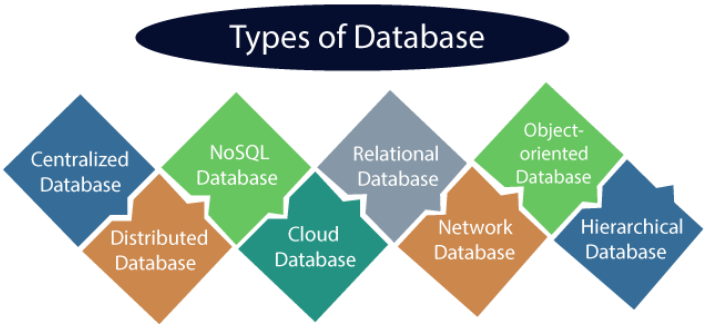
interrelated to each other within the database.

- 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.

DIFFERENCES BETWEEN A DBMS AND AN RDBMS

CATEGORIES OF DATABASES

There are various types of databases used for storing different varieties of data:



1.CENTRALIZED DATABASE: It is the type of database that actually stores information in a centralized database system. It allows users to access the stored data from different locations over numerous applications. These applications have the authentication process to allow users access data securely. An example of such a database is a **central Library** with a **central database** of every college library or university.

Advantages of Centralized Database

- It has decreased the risk of data management, i.e., manipulation of data will not affect core data.
- Data consistency is maintained as it manages data in a central repository.
- It provides better data quality, which enables organizations to establish data standards.
- It is less costly because fewer vendors are required to handle the data sets.

Disadvantages of Centralized Database

- The size of a centralized database is large, which increases the response time for fetching data.
- It is not easy to update such an extensive database system.
- If any server failure occurs, entire data will be lost, which could be a huge loss.

2.DISTRIBUTED DATABASE: In a distributed system, data is distributed among different database systems of an organization. These **database systems** are linked thru communication links. Such links help **end-users** to easily access the data. **Examples** are **Apache Cassandra**, **HBase**, **Ignite**, etc. We can sub-divide the distributed database system in: **Homogeneous DDB:** These are database systems which execute on the same operating system and use the same application process and use the same hardware devices. **Heterogeneous DDB:** Are database systems which execute on different operating systems under different application procedures, and carries different hardware devices.

Advantages of Distributed Database

- Modular development is possible in a distributed database, i.e., the system can be expanded by including new computers and connecting them to the distributed system.
- One server failure will not affect the entire data set.

3.RELATIONAL DATABASE: Is a database based on the **relational data model**, which stores data in the form of **rows**(**tuple**) and **columns**(**attributes**), and together forms a **table**(**relation**). A **relational database** uses SQL for storing, manipulating, as well as maintaining the data. **E.F. Codd** invented the database in **1970**. Each table in the database carries a key that makes the data unique from others.

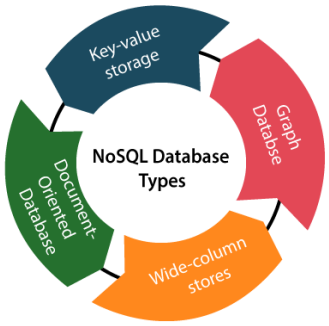
Examples of Relational databases are **MySQL**, **Microsoft SQL Server**, **Oracle**, etc.

PROPERTIES OF RELATIONAL DATABASE

There are four commonly known properties of a relational model known as **ACID** properties, where:

- **A** means **Atomicity:** This makes sure that **a data operation** will complete either with success or with failure. It follows the 'all or nothing' strategy. A transaction will either be committed or abort.
- **C** means **Consistency:** If we perform any operation over the data, its value before and after the operation should be preserved. E.g., the account balance before and after a transaction should be correct, i.e., it should remain conserved.
- **I** means **Isolation:** There can be **concurrent users** for accessing data at the same time from the database. Therefore, isolation between the data should remain isolated. For example, when multiple transactions occur at the same time, one transaction effects should not be visible to the other transactions in the database.
- **D** means **Durability:** It ensures that once it completes the operation and commits the data, data changes should remain permanent.

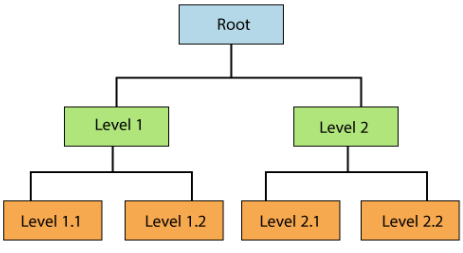
4.NOSQL DATABASE: Non-SQL or **Not Only SQL** is a type of database used for storing a wide range of data sets. It is not a relational database as it stores data not only in a **tabular format** but in several different ways. It appeared when the demand for building to modern applications increased. Therefore, it presented a wide variety of database technologies in response to the demands. We can further subdivide a NoSQL database into four:



- **Key-value storage:** It's the simplest type of database storage where it stores every single item as **a key** (or **attribute name**) holding its value, together.
- **Document-oriented Database:** A type of database used to store data as JSON-like document. It helps developers to store data by using the same document-model format as that used in the application code.
- **Graph Database:** It's used to store very huge amounts of data in a graph-like structure. Usually, social networking websites use the graph database.
- **Wide-column stores:** It's similar to the data represented in relational databases. Here, data is stored in large columns together, instead of storing in rows.

5.CLOUD DATABASE: Is a type of database where data is truly stored in a virtual environment and executes over cloud computing platforms. It provides users with various cloud computing services (**SaaS**, **PaaS**, **IaaS**, etc.) for accessing the database. There are numerous cloud platforms, but the best options are: **Amazon Web Services**(AWS), **Microsoft Azure**, **Kamatera**, **PhonixNAP**, **ScienceSoft, Google Cloud SQL** etc.

6.OBJECT-ORIENTED DATABASES: This is a database that use the object-based data model approach for storing data in a database system. The data is represented and stored as objects which are very similar to the objects used in the object-oriented programming language.



Hierarchical Database

7.HIERARCHICAL DATABASES: It's a database that stores information in form of a parent-children relationship nodes. It typically organizes the data in a tree-like structure. Data get stored in the form of records that are connected via links. Each child record in the

tree will has only one parent. On the other hand, each parent record can have multiple child records.

8.NETWORK DATABASES: It is a database category that follows the network data model. Data representation is in the form of nodes connected through links between them. It allows each record to have many children and parent nodes to form a general graph structure.