CS403 FAVOURITE SUBJECT
SRS+DD+SQL+PHP
FINAL-VIVA
Prepared by: JUNAID MALIK



AL-Junaid Institude

Contact no: 03041659294

VU LMS HELP vulmshelp.com

LMS Handling

Subject Enrollment

1) Online classes available
HTML, CSS, Javascript
Phyton..etc
2) Solved Graded Activities
- *Assignment's*
- *Quizze's*
- *GDB's*

3) Sloved Quiz File's
4) Short Note's
5) Past paper's &
Current paper's

Final project CS619

1) SRS
(Software Requirement's Specification)
2) DD
(Design Document)
3) Test phase + viva
4) Viva preparation
5) Final Deliverable

Website Link
vulmshelp.com

Gmail
junaidfazal08@gmail.com

0304-1659294
Junaidfazal08@gmail.com

UML (Unified Modeling Language)

UML is a standard language for specifying, visualizing, constructing, and documenting the of software systems. UML was created by the Object Management Group (OMG). UML specification draft was proposed to the OMG in January 1997.

- **UML** has become an OMG standard
- **UML** is different from the other common programming languages such as C++, Java, COBOL, etc.
- **UML** is a pictorial language used to make software blueprints.
- **UML** can be described as a general-purpose visual modeling language to visualize, specify, construct, and document software system.
- **UML** diagrams are not only made for developers but also for business users, common people, and anybody interested to understand the system. The system can be a software or non-software system
- **UML** is not a programming language but tools can be used to generate code in various languages using UML diagrams.
- **UML** has a direct relation with object-oriented analysis and design.
- The goals for developing UML is to define some general-purpose modeling language, which all modelers can use and it also needs to be made simple to understand and use.

A CONCEPTUAL MODEL OF UML

A conceptual model can be defined as a model which is made of concepts and their relationships. A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other.

The conceptual model of UML can be mastered by learning the following three major elements.

- UML building blocks

vulmshelp.com

- Rules to connect the building blocks
- Common mechanisms of UML

OBJECT-ORIENTED CONCEPTS

UML is an analysis and design of object-oriented (OO).

An object contains both data and methods that control the data. The data represents the state of the object. A class describes an object and they also form a hierarchy to model the real-world system. The hierarchy is represented as inheritance. The classes can also be associated in different ways as per the requirement.

Objects are the real-world entities that exist around us and the basic concepts such as **abstraction**, **encapsulation**, **inheritance**, and **polymorphism** all can be represented using UML.

UML is powerful enough to represent all the concepts that exist in object-oriented analysis and design.

UML diagrams are representation of object-oriented concepts only.

Fundamental concepts of the object-oriented world:

- **Objects** − Objects represent an entity and the basic building block.
- **Class** − Class is the blue print of an object.
- **Abstraction** − Abstraction represents the behavior of a real-world entity.
- **Encapsulation** − Encapsulation is the mechanism of binding the data together, hiding them from outside world.
- **Inheritance** − Inheritance is the mechanism of making new classes from existing ones.
- **Polymorphism** − Is the mechanism to exists in different forms. The ability of different objects to respond in a unique way to the same message.

OO ANALYSIS AND DESIGN

vulmshelp.com

Analysis means the investigation of objects; Design means the collaboration of identified objects.

The purpose of OO analysis is to identify objects of a system to be designed. This analysis is also done for an existing system. After identifying the objects, their relationships are identified and finally the design is produced.

Purpose of OO analysis and design:

- Identifying the objects of a system.
- Identifying their relationships.
- Making a design, executable using OO languages.

Steps where the OO concepts are applied and implemented.

OO Analysis → OO Design → OO Implementation

The above three points can be described in detail as −

- **OO Analysis** - The objects should be identified with responsibilities. Responsibilities are the functions performed by the object. Each and every object has some type of responsibilities to be performed.
- **OO Design** - The objects are collaborated according to their intended association.
- **OO Implementation** - The design is implemented using OO languages such as Java, C++, etc.

Role of UML in OO Design

- **Things**
- **Relationships**
- **Diagrams**

Things:

Things are the most important building blocks of UML. Things can be −

- Structural
- Behavioral
- Grouping

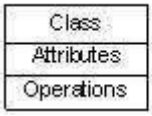
vumshelp.com

- An notational

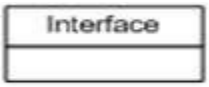
Structural

Define the **static part of the model**. They represent the physical and conceptual elements. Following are the brief descriptions of the structural things.

Class − Class represents a set of objects having similar responsibilities.



Interface − Interface defines a set of operations, which specify the responsibility of a class.



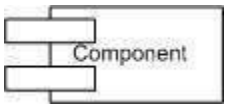
Collaboration −Collaboration defines an interaction between elements.



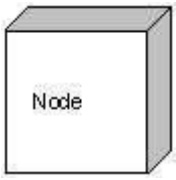
Use case −Use case represents a set of actions performed by a system for a specific goal.



Component −Component describes the physical part of a system.



Node − A node can be defined as a physical element that exists at run time.

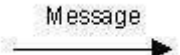


vulmshelp.com

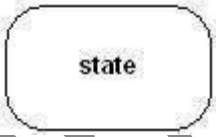
BEHAVIORAL THINGS

A behavioral thing consists of the **dynamic parts of UML model**. Following are the behavioral things

Interaction − behavior that consist of a group of messages exchanged among elements to accomplish a specific task.



State Machine − The sequence of states an object goes through in response to events. Events are external factors responsible for state change



GROUPING THINGS

A mechanism to group UML elements together. There is only one grouping thing available −

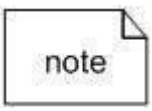
Package − Package is the only one grouping thing available for gathering structural and behavioral things.



ANNOTATIONAL THINGS

A mechanism to capture remarks, descriptions, and comments of UML model elements.

Note - It is the only one An notational thing available. A note is used to render comments, constraints, etc. of an UML element.



vulmshelp.com

Relationship:

It shows how the elements are associated with each other and this association describes the functionality of an application.

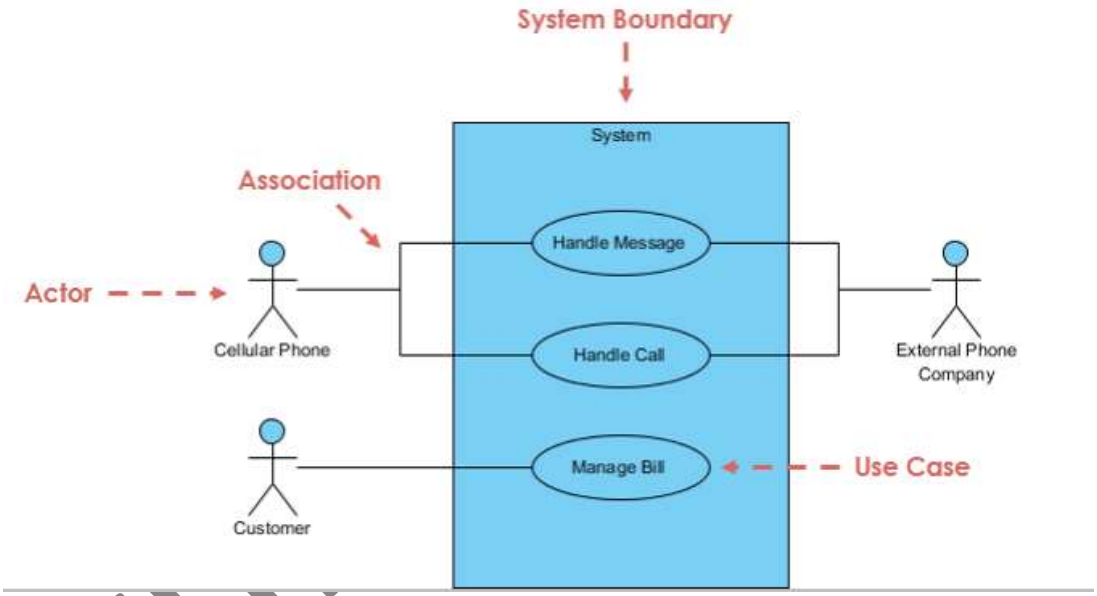
There are four kinds of relationships available.

DEPENDENCY

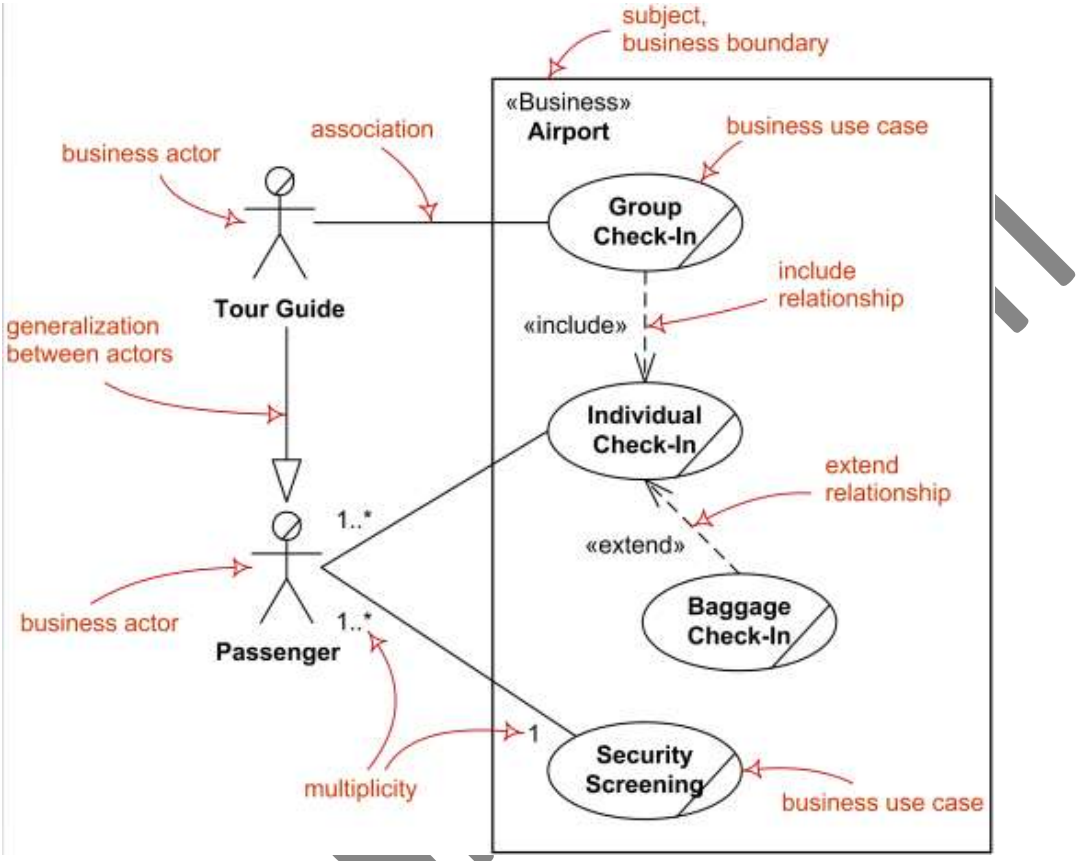
A relationship between two things in which change in one element also affects the other.

Dependency

EXAMPLE OF USE CASE DIAGRAM:



USE CASE DIAGRAM RELATIONSHIP:



Icon	Name
	Use Case
	Association
	Actor
	System
	Include
	Extend
	Dependency
	Generalization
	Realization
	Collaboration

Use Case

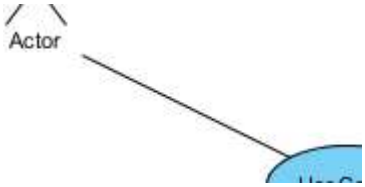


A use case represents a user goal that can be achieved by accessing the system or software application. In Visual Paradigm, you can make use of the sub-diagram feature to describe the interaction between user and system within a use case by creating a sub-sequence diagram under a use case. You can also describe the use case scenario using the Flow of Events editor.

OMG UML Specification

What is a use case in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version use case is:

Association



Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association correspond to a sequence of actions between the actor and use case in achieving the use case.

OMG UML Specification

What is an association in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version), association is:

An association describes a set of tuples whose values refer to typed instances. An i nstance of an association is called a link. A link is a tuple with one value for each e nd of the association, where each value is an instance of the type of the end.

An association specifies a semantic relationship that can occur between typed inst ances. It has at least two ends represented by properties, each of which is connecte d to the type of the end. More than one end of the association may have the same ty pe.

An end property of an association that is owned by an end class or that is a navigab le owned end of the association indicates that the association is navigable from the opposite ends; otherwise, the association is not navigable from the opposite ends.

Actor



Actors are the entities that interact with a system. Although in most cases, actors are used to represent the users of system, actors can actually be anything that needs to exchange information with the system. So, an actor may be people, computer hardware, other systems, etc.

Note that actor represents a role that a user can play but not a specific user. So, in a hospital information system, you may have doctor and patient as actors but not Dr. John, Mrs. Brown as actors.

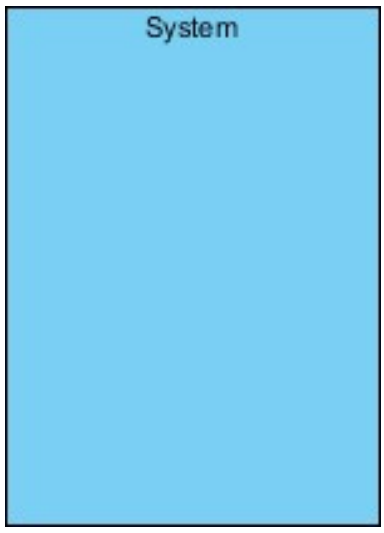
OMG UML Specification

What is an actor in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version actor is:

An actor specifies a role played by a user or any other system that interacts with th e subject. (The term "role" is used informally here and does not necessarily imply t he technical definition of that term found elsewhere in this specification.)

An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data) but which is external to the subject (i.e. in the sense that an instance of an actor is not a part of the instance of its corresponding s ubject). Actors may represent roles played by human users, external hardware, or o ther subjects. Note that an actor does not necessarily represent a specific physical e ntity but merely a particular facet (i.e."role") of some entity that is relevant to the s pecification of its associated use cases. Thus, a single physical instance may play t he role of several different actors and conversely, a given actor may be played by multiple different instances.

System

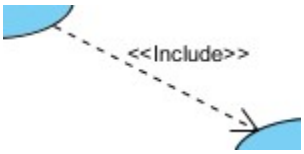


OMG UML Specification

What is a system in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version system is:

If a subject (or system boundary) is displayed, the use case ellipse is visually located inside the system boundary rectangle. Note that this does not necessarily mean that the subject classifier owns the contained use cases, but merely that the use case applies to that classifier.

Include



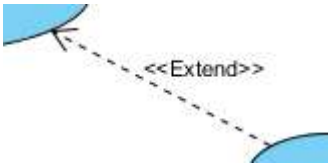
An include relationship specifies how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.

OMG UML Specification

What is an include in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1, page 604), include is:

An include relationship defines that a use case contains the behavior defined in another use case.

Extend



An extend relationship specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.

OMG UML Specification

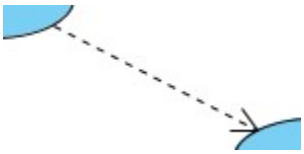
What is an extend in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version extend is:

A relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case.

This relationship specifies that the behavior of a use case may be extended by the behavior of another (usually supplementary) use case. The extension takes place at one or more specific extension points defined in the extended use case. Note, however, that the extended use case is defined independently of the extending use case and is meaningful independently of the extending use case. On the other hand, the extending use case typically defines behavior that may not necessarily be meaningful by itself. Instead, the extending use case defines a set of modular behavior increments that augment an execution of the extended use case under specific conditions.

Note that the same extending use case can extend more than one use case. Furthermore, an extending use case may itself be extended.

Dependency



A dependency relationship represents that a model element relies on another model element for specification and/or implementation.

OMG UML Specification

What is a dependency in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version dependency is:

A dependency is a relationship that signifies that a single or a set of model element s requires other model elements for their specification or implementation. This mea ns that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s).

Generalization



A generalization relationship is used to represent inheritance relationship between model elements of same type. The more specific model element share the same specification with. the more general the model element but carries more details in extra.

OMG UML Specification

What is a generalization in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version generalization is:

A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features o f the more general classifier.

Realization



A realization is a relationship between a specification and its implementation.

OMG UML Specification

What is a realization in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version realization is:

Realization is a specialized abstraction relationship between two sets of model ele ments, one representing a specification (the supplier) and the other represents an i mplementation of the latter (the client). Realization can be used to model stepwise refinement, optimizations, transformations, templates, model synthesis, framework composition, etc.

Collaboration



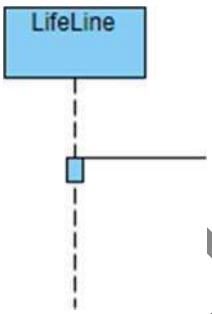
OMG UML Specification

What is a collaboration in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version collaboration is:

A collaboration describes a structure of collaborating elements (roles), each performing a specialized function, which collectively accomplish some desired functionality. Its primary purpose is to explain how a system works and, therefore, it typically only incorporates those aspects of reality that are deemed relevant to the explanation. Thus, details, such as the identity or precise class of the actual participating instances are suppressed.

ACTIVATION

An activation is represented by a thin rectangle on a lifeline) represents the period during which an element is performing an operation. The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively



Calls Message:

A call message defines a particular communication between lifelines of an interaction, which represents an invocation of operation of target lifeline.



vulmshelp.c

Return Message:

A return message defines a particular communication between lifelines of an interaction, which represents the pass of information back to the caller of a corresponded former message.



Self-Message:

A self-message defines a particular communication between lifelines of an interaction, which represents the invocation of message of the same lifeline.



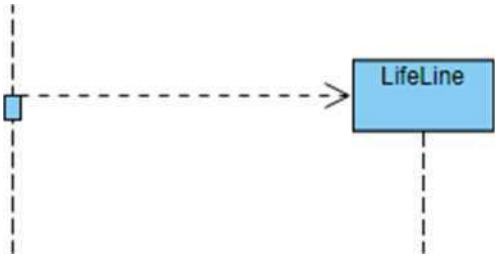
Recursive message

A recursive message defines a particular communication between lifelines of an interaction, which represents the invocation of message of the same lifeline. It's target points to an activation on top of the activation where the message was invoked from.



Create Message:

A create message defines a particular communication between lifelines of an interaction, which represents the instantiation of (target) lifeline.



vulmshelp.com

Destroy message

A destroy message defines a particular communication between lifelines of an interaction, which represents the request of destroying the lifecycle of target lifeline.



Duration message:

A duration message defines a particular communication between lifelines of an interaction, which shows the distance between two-time instants for a message invocation.



NOTE:

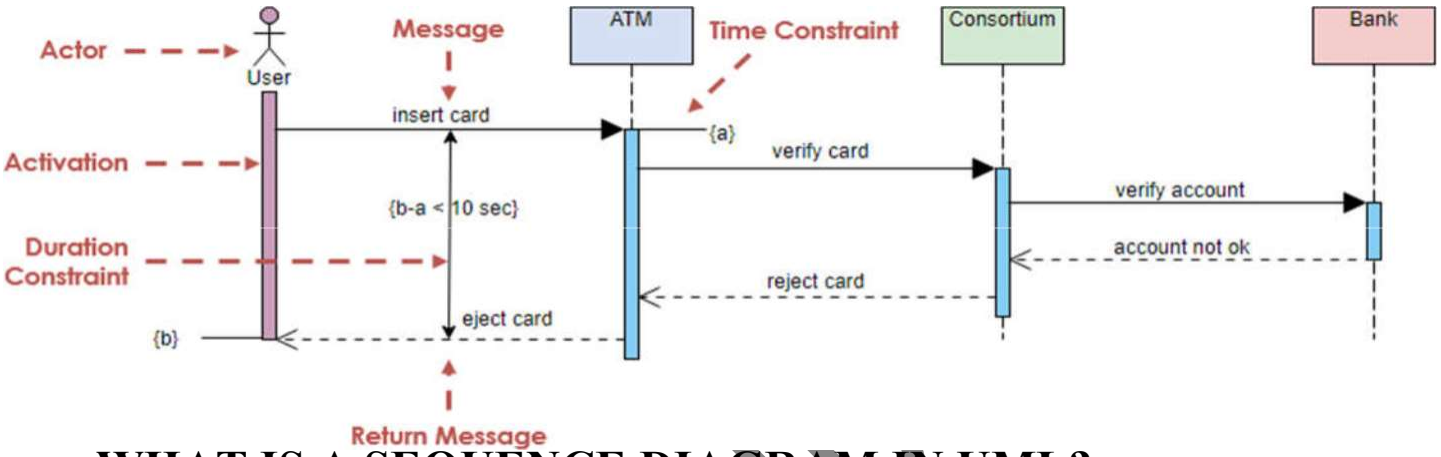
A note (comment) gives the ability to attach various remarks to elements. A comment carries no semantic force, but may contain information that is useful to a modeler.



vulmshelp.com

SEQUENCE DIAGRAM TUTORIAL

A sequence diagram describes an interaction among a set of objects participated in a collaboration (or scenario), arranged in a chronological order; it shows the objects participating in the interaction by their "lifelines" and the messages that they send to each other.



WHAT IS A SEQUENCE DIAGRAM IN UML?

UML Sequence diagrams are interaction diagrams that detail how operations are carried out. As sequence diagrams can be used to capture the interaction between objects in the context of a collaboration, one of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams.

Sequence diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

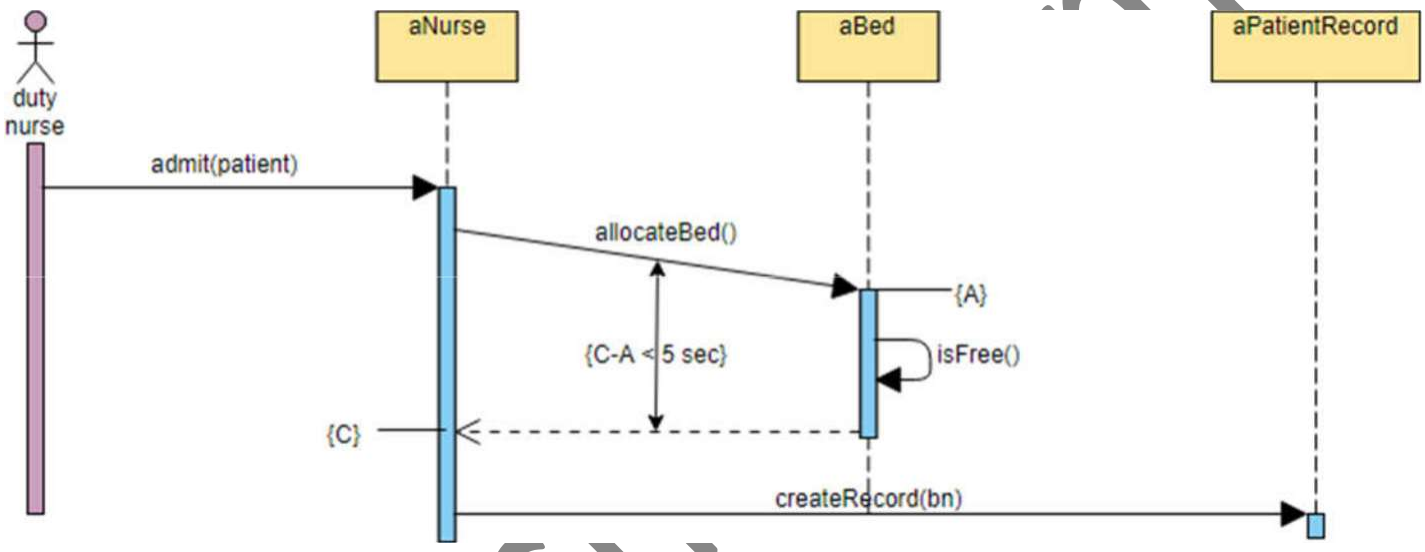
Sequence Diagrams captures interaction in different level of granularity:

1. high-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams)

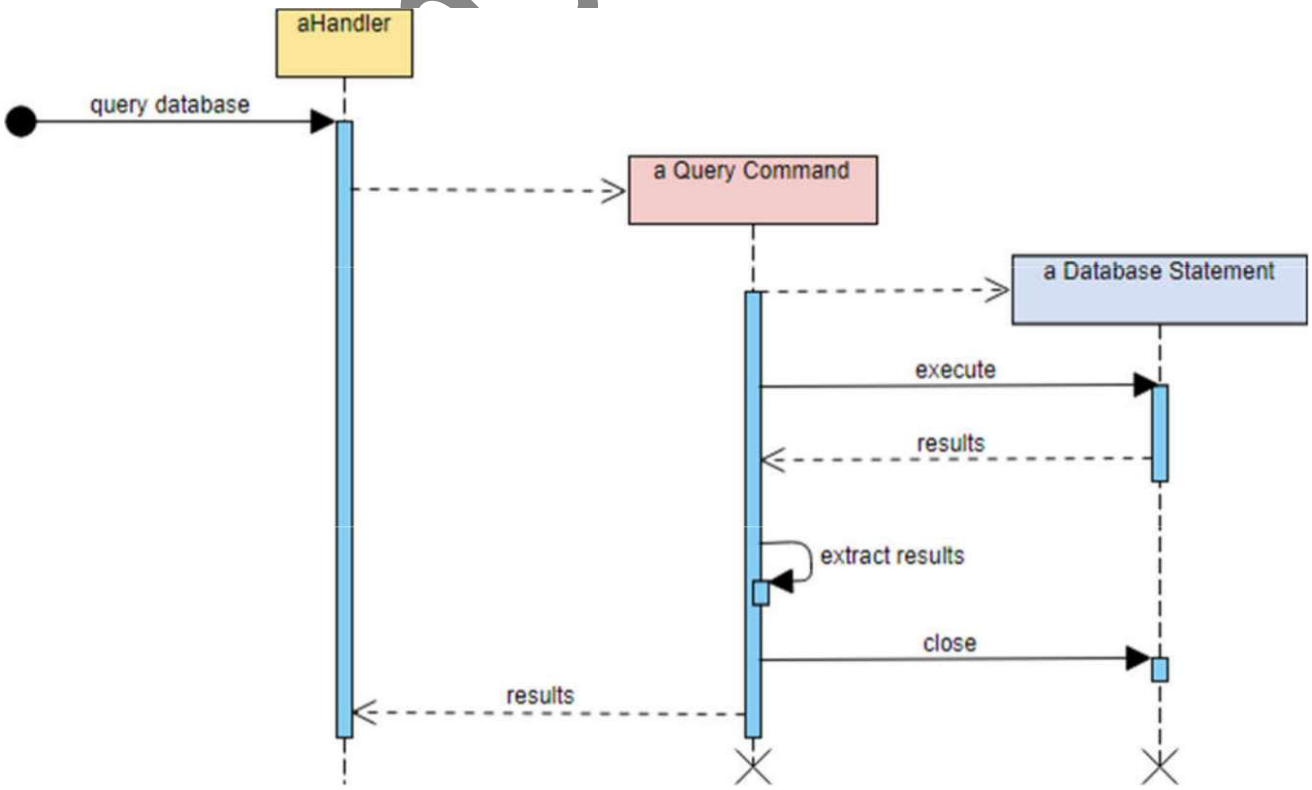
vulmshen

2. the interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
3. Represent objects interact in (Model, View / Controller) MVC pattern of software framework

The sequence diagram example below shows a patient admission process. It shows the use of timing and duration constraint.



The sequence diagram example shows how recursive message can be used in interaction modeling.



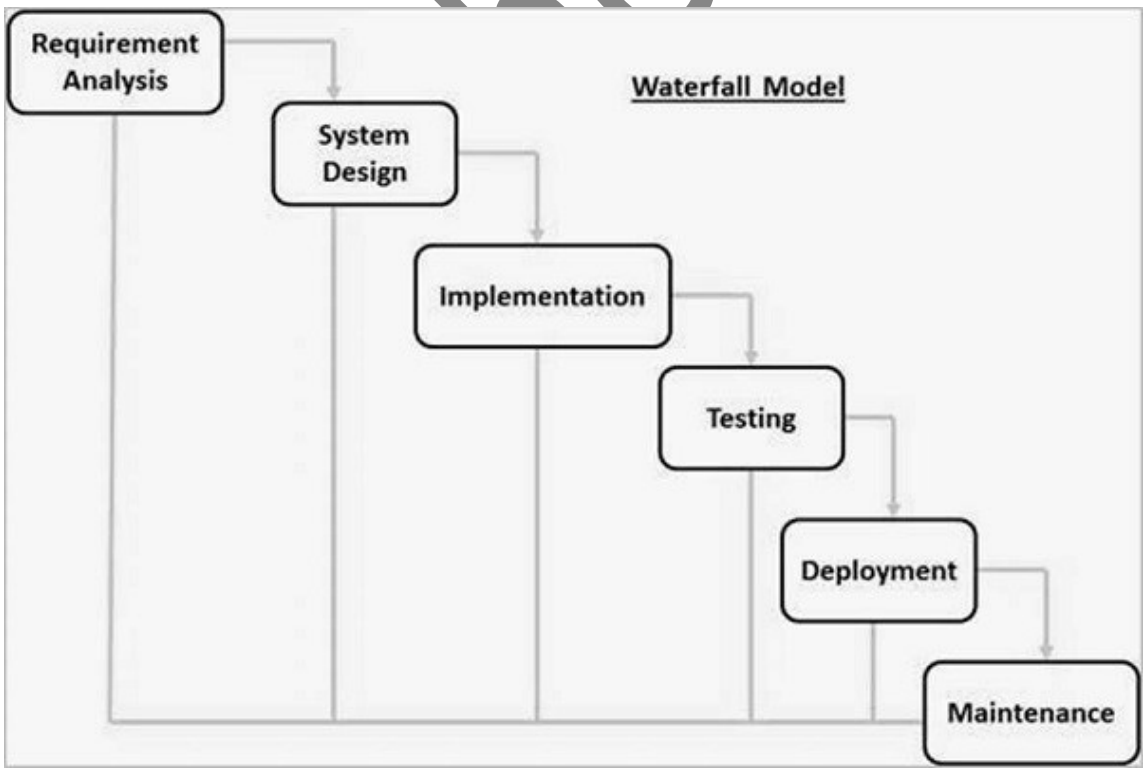
Waterfall Model

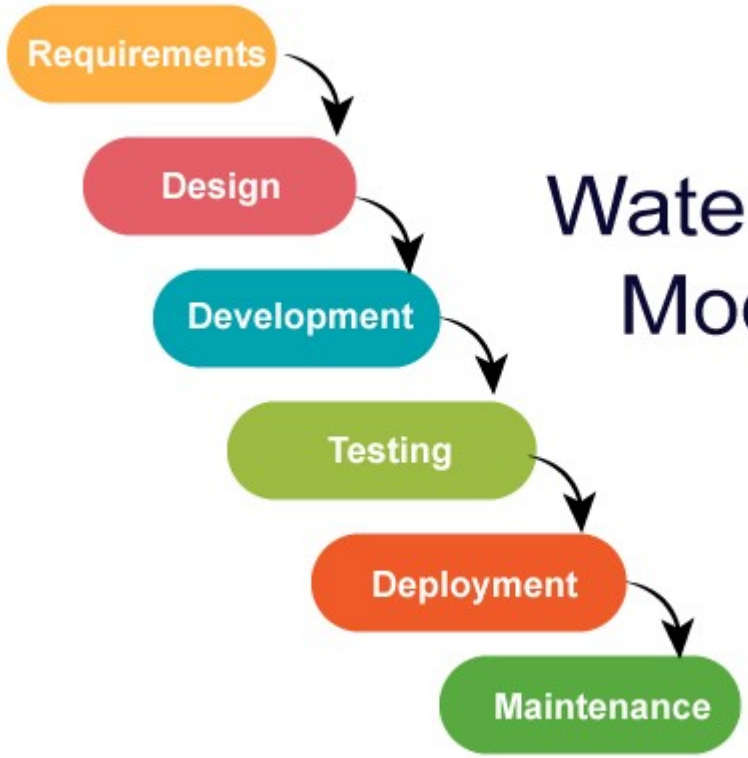
The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.





The sequential phases in Waterfall model are

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

Waterfall Model - Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model - Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development

vulmshelp.com

and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.

vulmshelp.com

- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

Introduction to DBMS

DBMS stands for **D**ata**b**ase **M**anagement **S**ystem. We can break it like this DBMS = Database + Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can **define DBMS** like this: DBMS is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.

Need and Purpose of DBMS?

Database systems are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: **Storage of data** and **retrieval of data**. The main purpose of database systems is to manage the data.

- **Data redundancy:** Data redundancy refers to the duplication of data, let's say we are managing the data of a college where a student is enrolled for two courses, the same student details in such case will be stored twice, which will take more storage than needed. Data redundancy often leads to higher storage costs and poor access time.
- **Data inconsistency:** Data redundancy leads to data inconsistency, let's take the same example that we have taken above, a student is enrolled for two courses and we have student address stored twice, now let's say student requests to change his address, if the address is changed at one place and not on all the records then this can lead to data inconsistency.

vulmshelp.com

- **No redundant data**: Redundancy removed by data normalization. No data duplication saves storage and improves access time.
- **Data Consistency and Integrity**: As we discussed earlier the root cause of data inconsistency is data redundancy, since data normalization takes care of the data redundancy, data inconsistency also been taken care of as part of it
- **Data Security**: It is easier to apply access constraints in database systems so that only authorized user is able to access the data. Each user has a different set of access thus data is secured from the issues such as identity theft, data leaks and misuse of data.
- **Privacy**: Limited access means privacy of data.
- **Easy access to data** – Database systems manages data in such a way so that the data is easily accessible with fast response times.
- **Easy recovery**: Since database systems keeps the backup of data, it is easier to do a full recovery of data in case of a failure.
- **Flexible**: Database systems are more flexible than file processing systems.

Types of DBMS Architecture

There are three types of DBMS architecture:

1. Single tier architecture
2. Two tier architecture
3. Three tier architecture

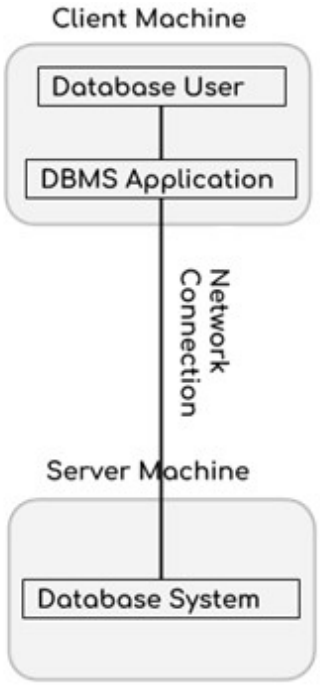
1. Single tier architecture

In this type of architecture, the database system and the DBMS application are available on the **client machine**, any request made by client doesn't require a network connection to perform the action on the database.
For example, let's say you want to fetch the records of employee from the database and the database is available on your computer system, so the request

vulmshelp.com

to fetch employee details will be done by your computer and the records will be fetched from the database by your computer as well. This type of system is generally referred as **local database system**.

2. Two tier architecture



Two-Tier architecture

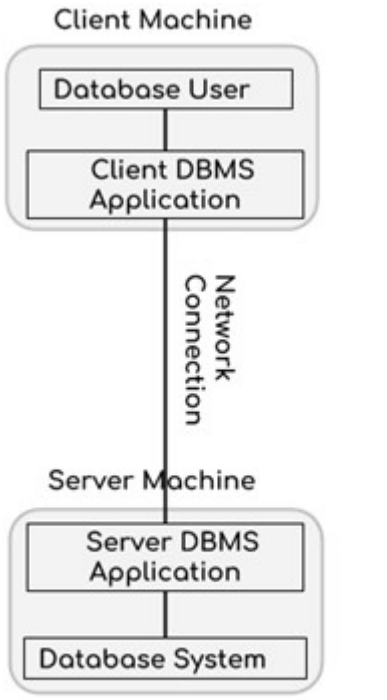
www.vuhelp.com

In **two-tier architecture**, the Database system is present at the **server machine** and the DBMS application is present at the **client machine**, these two machines are connected with each other through a **reliable network** as shown in the above diagram.

Whenever client machine makes a request to access the database present at server using a query language like SQL, the server perform the request on the database and returns the result back to the client. The application connection interface such as JDBC, ODBC are used for the interaction

between server and client.

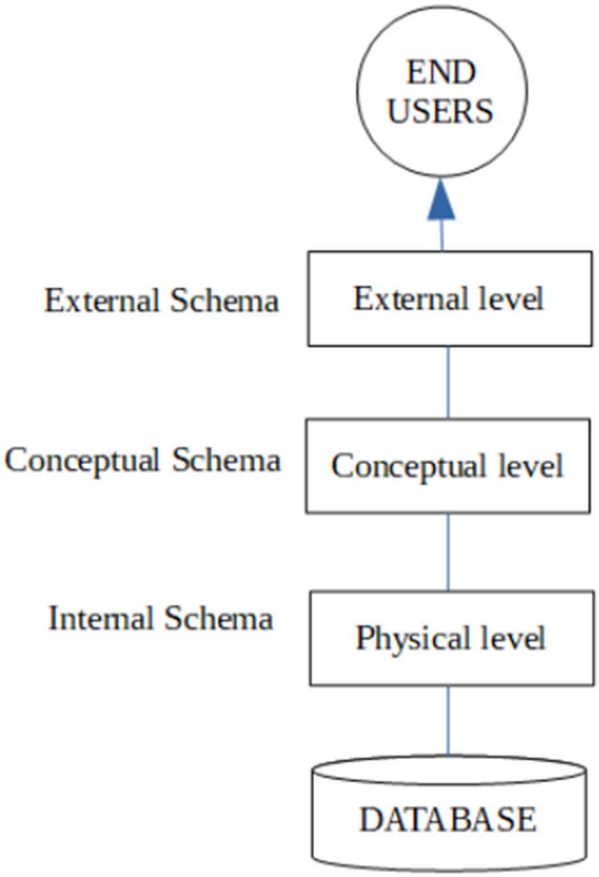
3. Three tier architecture



Three-Tier architecture

In **three-tier architecture**, another layer is present between the **client machine** and **server machine**. In this architecture, the client application doesn't communicate directly with the database systems present at the server machine, rather the client application communicates with server application and the server application internally communicates with the database system present at the server.

vulmsh



DBMS Three Level Architecture Diagram

This architecture has three levels:
1. External level
2. Conceptual level
3. Internal level

1. External level

It is also called **view level**. The reason this level is called "view" is because several users can view their desired data from this level which is internally fetched from database with the help of conceptual and internal level mapping. The user doesn't need to know the database schema details such as data structure, table definition etc. user is only concerned about data which is what returned back to the view level after it has been fetched from

vums.

database (present at the internal level).
External level is the "**top level**" of the Three Level DBMS Architecture

. Conceptual level

It is also called **logical level**. The whole design of the database such as relationship among data, schema of data etc. are described in this level. Database constraints and security are also implemented in this level of architecture. This level is maintained by DBA (database administrator).

3. Internal level

This level is also known as **physical level**. This level describes how the data is actually stored in the storage devices. This level is also responsible for allocating space to the data. This is the **lowest level of the architecture**.

Data Abstraction
The process of hiding irrelevant details from user is called data abstraction.

We have three levels of abstraction:

Physical level: This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.

Logical level: This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

View level: Highest level of data abstraction. This level describes the user interaction with database system.

Example: Let's say we are storing customer information in a customer table.

At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the

vulmshelp.com

programmers.
At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

DBMS Schema

Schema is a **structural view (design) of a database**. Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

The design of a database at physical level is called **physical schema**, how the data stored in blocks of storage is described at this level.

Design of database at logical level is called **logical schema**, programmers and database administrators work at this level.

Design of database at view level is called **view schema**. This generally describes end user interaction with database systems.

DBMS Instance

At a particular moment, the data stored in database is called the instance. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

vulmshelp.com

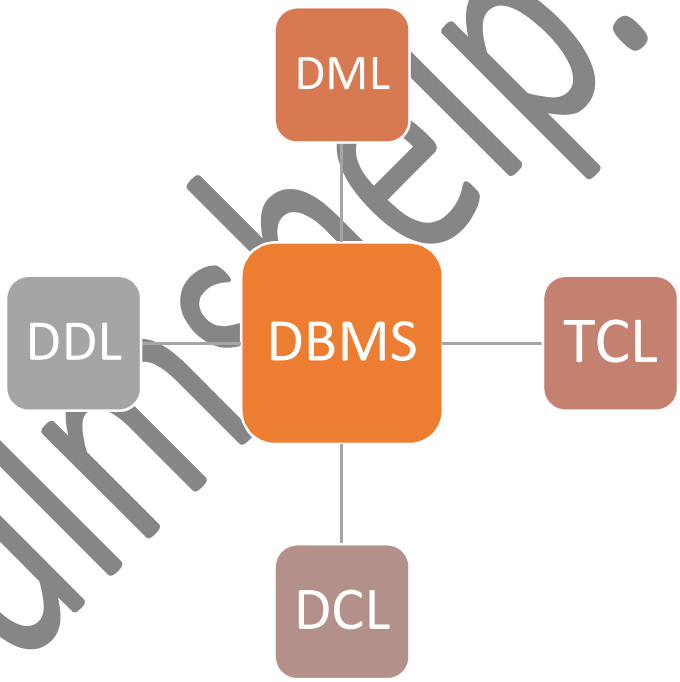
DBMS Languages

Database languages are used to **read**, **update** and **store** data in a database. There are several such languages that can be used for this purpose; one of them is **SQL** (**S**tructured **Q**uery **L**anguage).

Types of DBMS Languages:

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control language (DCL)
4. Transaction Control Language (TCL)

vulmshelp.com



Data Definition Language (**DDL**)

DDL is used for specifying the **database schema**. It is used for creating tables, schema, indexes, constraints etc.

- To create the database instance – **CREATE**

- To alter the structure of database – **ALTER / RENAME**
- To drop database instances, objects – **DROP / TRUNCATE**

DML is used for **accessing and manipulating data** in a database.

- To read records from table(s) – **SELECT**
- To insert record(s) into the table(s) – **INSERT**
- Update the data in table(s) – **UPDATE**
- Delete all the records from the table – **DELETE**
- Data Control language (**DCL**)

DCL is used for **granting and revoking user access** on a database –

- To grant access to user – **GRANT**
- To revoke access from user – **REVOKE**

Transaction Control Language (**TCL**)

The changes in the database that we made using DML commands are either performed or roll backed using TCL.

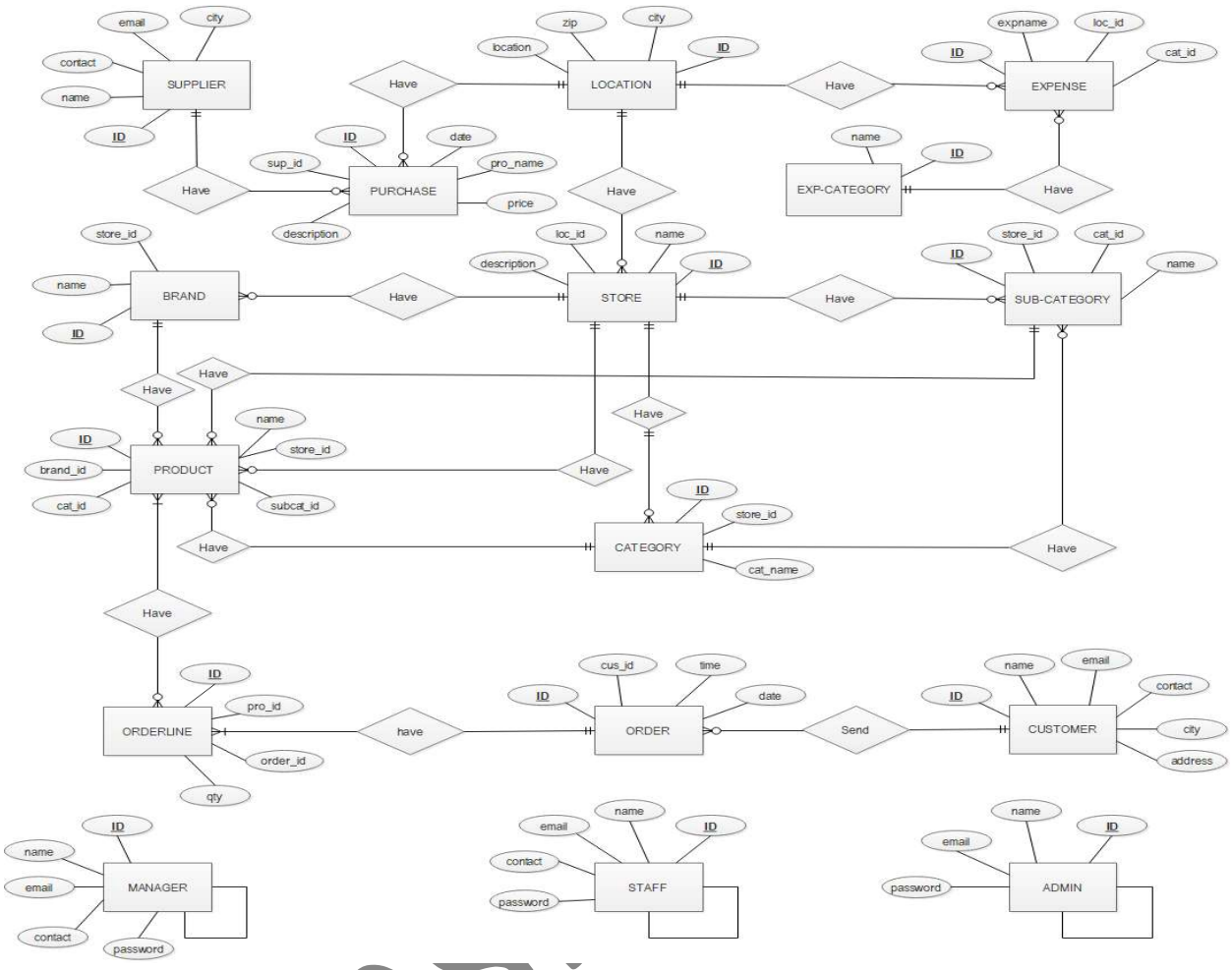
- To persist the changes made by DML commands in database – **COMMIT**
- To roll back the changes made to the database – **ROLLBACK**

Data Model is a logical structure of Database. It describes the design of database to reflect entities, attributes, relationship among data, constrains etc.

Entity Relationship Diagram – DBMS

An Entity–relationship model **(ER model)** describes the **structure of a database** with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E- R model are: **entity set** and **relationship set**.

vulmshelp.com



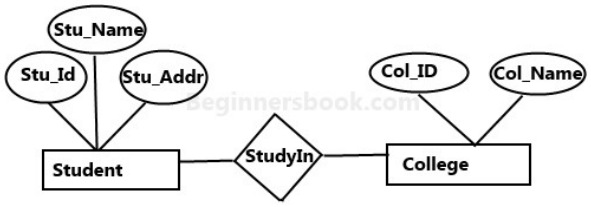
What is an Entity Relationship Diagram (ER Diagram)?

An ER diagram shows the relationship among entity sets. An **entity set is a group of similar entities** and these entities can **have attributes**. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete **logical structure of a database**.

A simple ER Diagram:

In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study

in multiple colleges at the same time. Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name.



Sample E-R Diagram

Rectangle: Represents Entity sets.

Ellipses: Attributes

Diamonds: Relationship Set

Lines: They link attributes to Entity Sets and Entity sets to Relationship Set

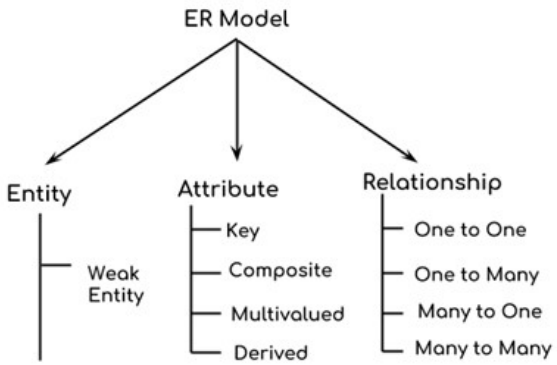
Double Ellipses: Multivalued Attributes

Dashed Ellipses: Derived Attributes

Double Rectangles: Weak Entity Sets

Double Lines: Total participation of an entity in a relationship set

Components of ER Diagram:



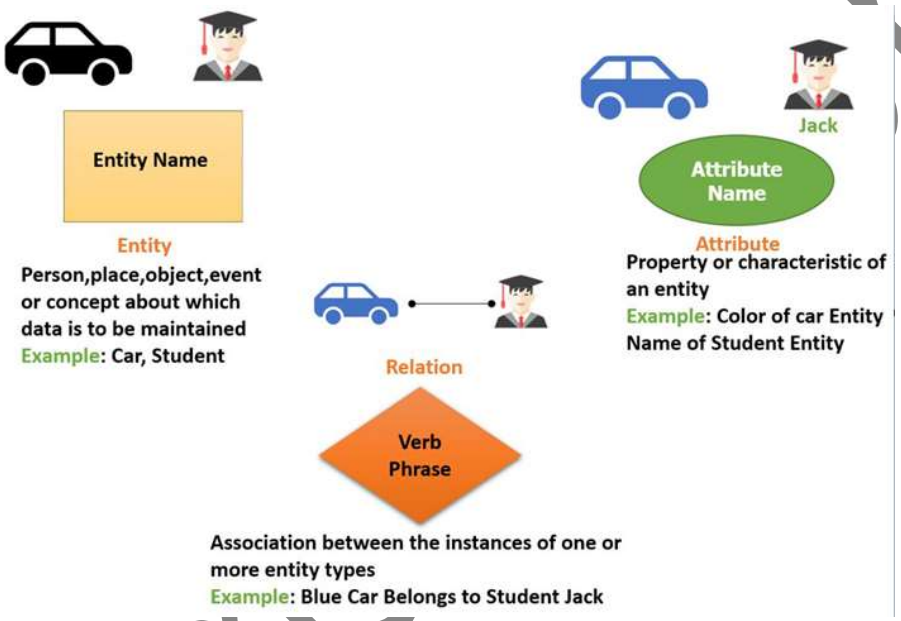
Components of ER Diagram

As shown in the above diagram, an ER diagram has three main components:

1. Entity
2. Attribute
3. Relationship

1. Entity

An entity is an **object or component of data**. An entity is represented as

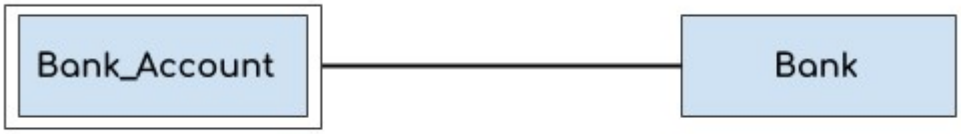


rectangle in an ER diagram.

Weak Entity:

An entity that **cannot be uniquely identified** by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a **double rectangle**.

For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.



2. Attribute

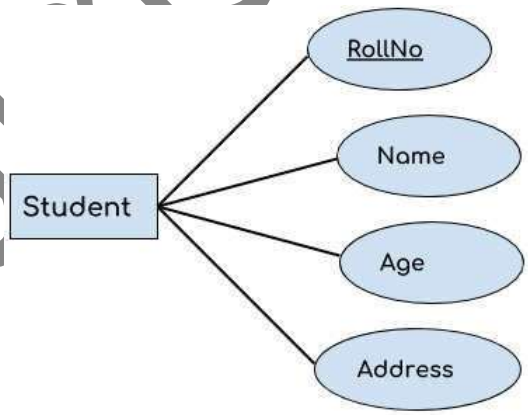
An attribute describes the **property of an entity**. An attribute is represented as **Oval/ Ellipses** in an ER diagram. There are four types of attributes:

1. Key attribute.
2. Composite attribute.
3. Multivalued attribute.
4. Derived attribute.

1. Key attribute:

A key attribute **can uniquely identify** an entity from an entity set. Key attribute is represented by oval same as other attributes however the **text of key attribute is underlined**.

For example, student roll number can uniquely identify a student from a set of students.

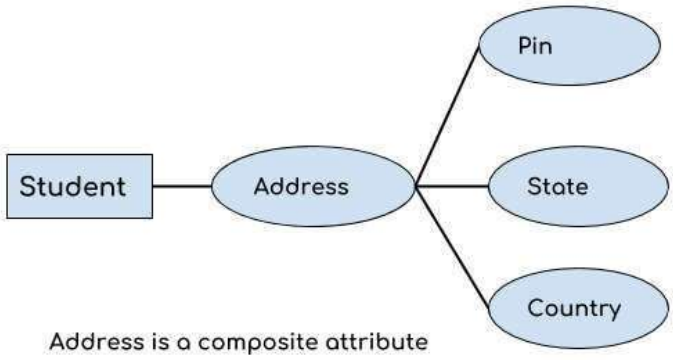


2. Composite attribute:

vulmshelp.com

An attribute that is a **combination of other attributes** is known as composite attribute.

For example, in student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.



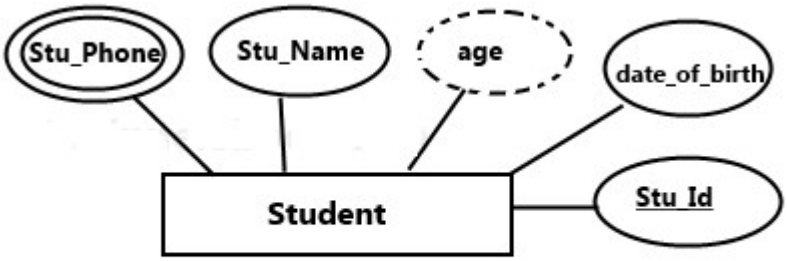
3. Multivalued attribute:

An attribute that can **hold multiple values** is known as multivalued attribute. It is represented with **double ovals** in an ER Diagram. For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

4. Derived attribute:

A derived attribute is one whose value is dynamic and **derived from another attribute**. It is represented by **dashed oval** in an ER Diagram. For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).

vulmshelp.com



Relationship

A relationship is represented by **diamond shape** in ER diagram, it shows the relationship among entities. There are four types of relationships:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

1. One to One Relationship

When a **single instance of an entity is associated with a single instance** of another entity then it is called one to one relationship. For example, a person has only one passport and a passport is given to one person.



2. One to Many Relationship

vulmshelp.com

When a **single instance of an entity is associated with more than one instances** of another entity then it is called one to many relationships. For example – a customer can place many orders but a order cannot be placed by many customers.



3. Many to One Relationship

When **more than one instances of an entity is associated with a single instance** of another entity then it is called many to one relationship. For example – many students can study in a single college but a student cannot study in many colleges at the same time.



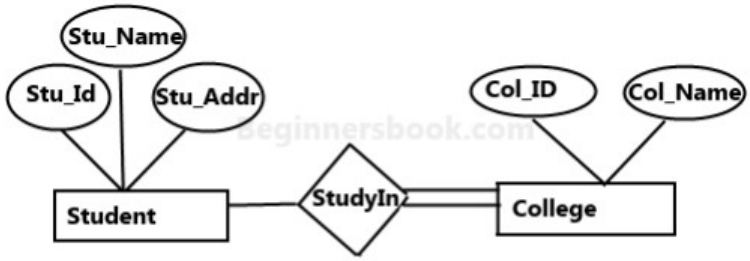
4. Many to Many Relationship

When **more than one instances of an entity is associated with more than one instances** of another entity then it is called many to many relationships. For example, a student can be assigned to many projects and a project can be assigned to many students.



Total Participation of an Entity set

A Total Participation of an entity set represents that **each entity in entity set must have at least one relationship** in a relationship set. For example: In the below diagram each college must have at-least one associated Student.



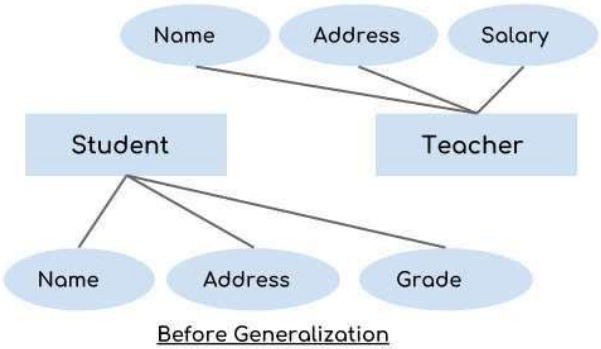
Generalization, Specialization & Aggregation

Generalization

Generalization is a process in which the **common attributes of more than one entity form a new entity**. This newly formed entity is called generalized entity.

Generalization Example

Let's say we have two entities Student and Teacher. Attributes of Entity Student are: Name, Address & Grade
Attributes of Entity Teacher are:
Name, Address & Salary

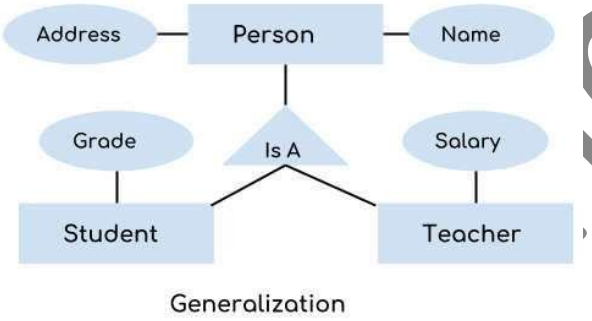


Before Generalization

These two entities have two common attributes: **Name and Address**, we can make a generalized entity with these common attributes. Let's have a look at the ER model after generalization.

The ER diagram after generalization:

We have created a new generalized entity **Person** and this entity has the common attributes of both the entities. After the generalization process the entities Student and Teacher only has the specialized attributes Grade and Salary respectively and their common attributes (Name & Address) are now associated with a new entity Person which is in the relationship with both the entities (Student & Teacher).



Generalization

Note:

1. Generalization uses **bottom-up approach** where two or more lower-level entities combine together to form a **higher-level** new entity.
2. The new generalized entity can further combine together with lower-level entity to create a further higher level generalized entity.

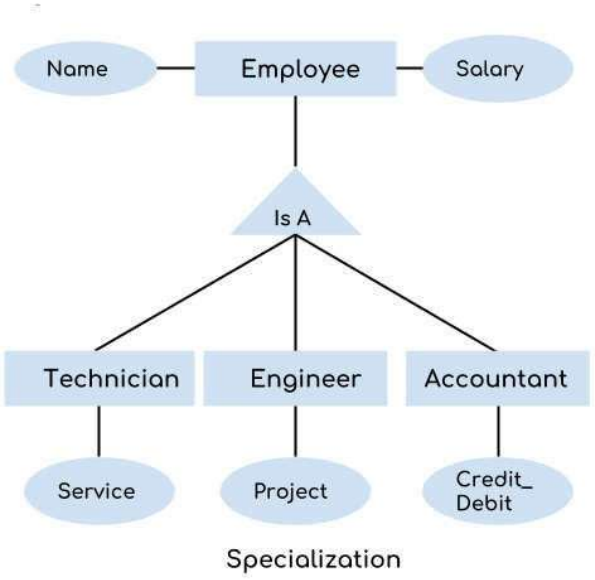
Specialization

Specialization is a process in which **an entity is divided into sub-entities**. It as a **reverse process of generalization**, in generalization two entities combine together to form a new higher-level entity. Specialization is a **top-down process**.

vulms.help.com

Specialization Example

Consider an entity employee which can be further classified as sub-entities Technician, Engineer & Accountant because these sub entities have some distinguish attributes

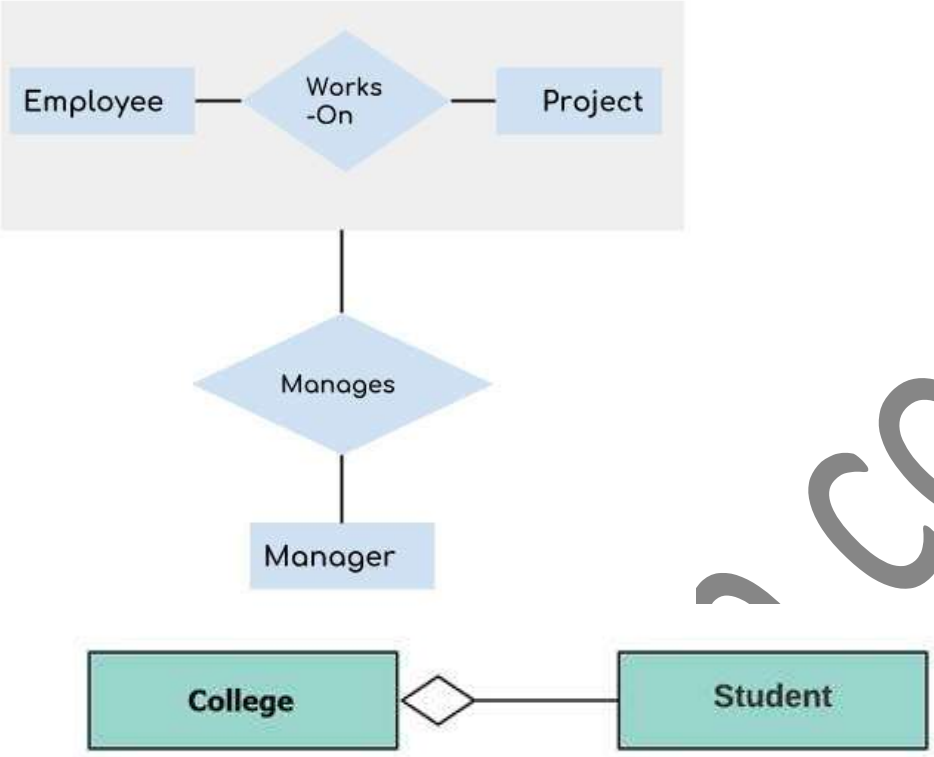


In the above diagram, we can see that we have a higher-level entity "**Employee**" which we have divided in sub entities "**Technician**", "**Engineer**" & "**Accountant**". All of these are just an employee of a company; however, their role is completely different and they have few different attributes.

Aggregation

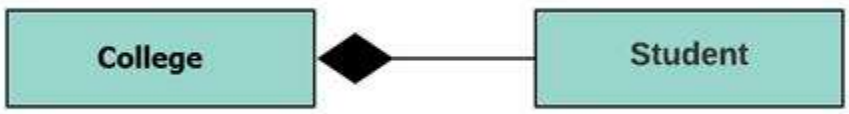
Aggregation is a process of **combining two or more entities to form a more meaningful new entity**. Aggregation is a process in which a single entity alone is not able to make sense in a relationship so the relationship of two entities acts as one entity.

vulmshelp.com



In real world, we know that a manager not only manages the employee working under them but he has to manage the project as well. In such scenario if entity "Manager" makes a "manages" relationship with either "Employee" or "Project" entity alone then it will not make any sense because he has to manage both. In these cases, **the relationship of two entities acts as one entity**. In our example, the relationship "Works-On" between "Employee" & "Project" acts as one entity that has a relationship "Manages" with the entity "Manager".

Composition:



The composition is a special type of aggregation which denotes strong ownership between two classes when one class is a part of another class.

For example, if college is composed of classes student. The college could contain many students, while each student belongs to only one college. So, if college is not functioning all the students also removed.

AGGREGATION VS. COMPOSITION

Aggregation	Composiion
Aggregation indicates a relationship where the child can exist separately from their parent class. Example: Automobile (Parent) and Car (Child). So, if you delete the Automobile, the child Car still exist.	Composition display relationship where the child will never exist independent of the parent. Example: House (parent) and Room (child). Rooms will never separate into a House.

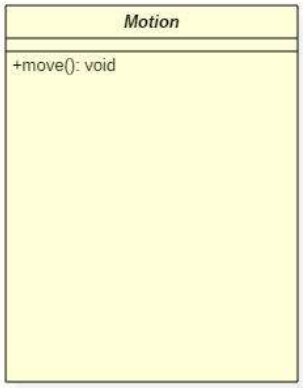
ABSTRACT CLASSES

It is a class with an operation prototype, but not the implementation. It is also possible to have an abstract class with no operations declared inside of it. An abstract is useful for identifying the functionalities across the classes. Let us consider an example of an abstract class. Suppose we have an abstract class called as a motion with a method or an operation declared inside of it. The method declared inside the abstract class is called a move ().

This abstract class method can be used by any object such as a car, an animal, robot, etc. for changing the current position. It is efficient to use this abstract class method with an object because no implementation is provided for the given function. We can use it in any way for multiple objects.

In UML, the abstract class has the same notation as that of the class. The only difference between a class and an abstract class is that the class name is strictly written in an italic font.

An abstract class cannot be initialized or instantiated.



vulmshelp.com

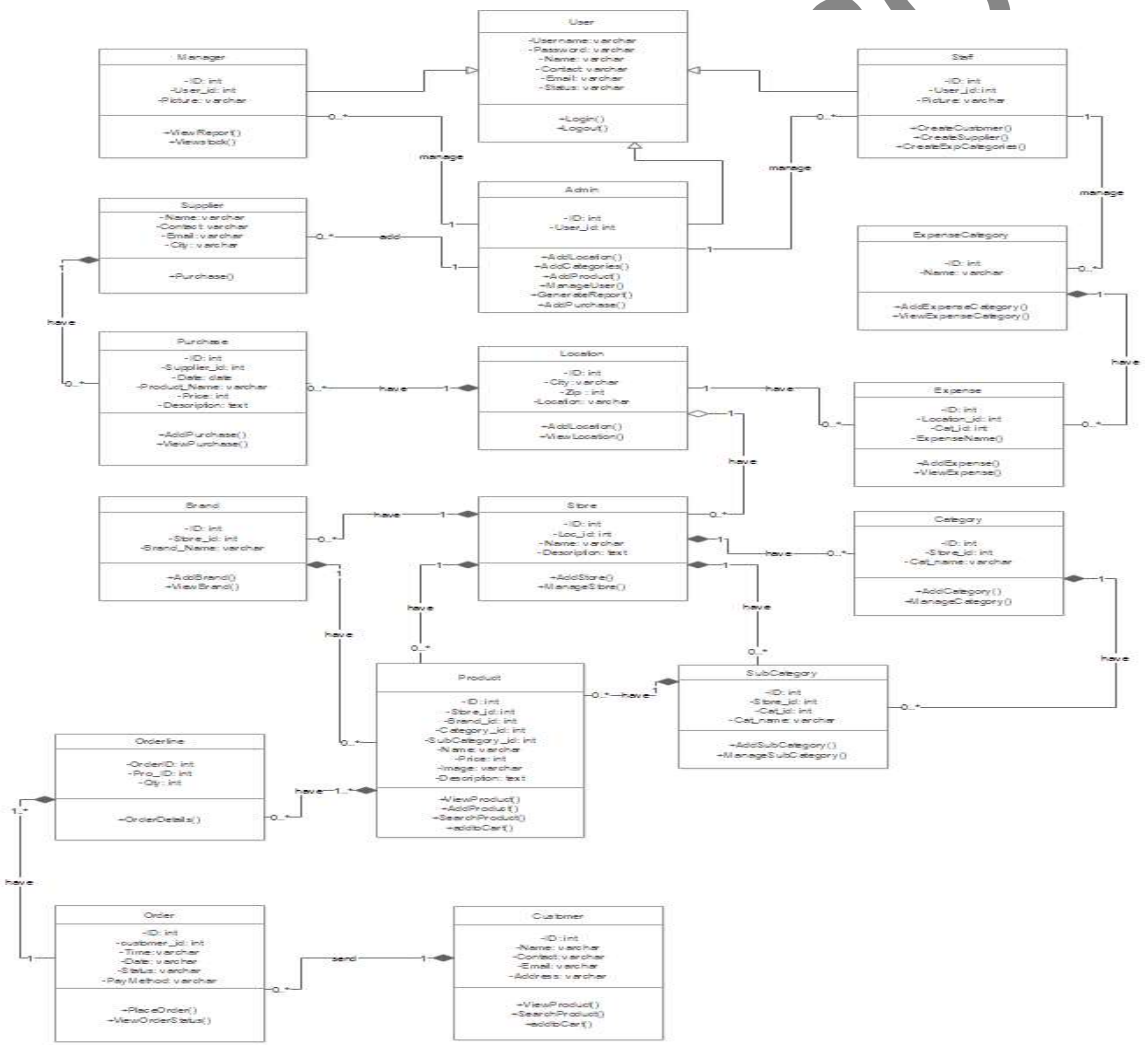
In the above abstract class notation, there is the only a single abstract method which can be used by multiple objects of classes.

EXAMPLE OF UML CLASS DIAGRAM

Creating a class diagram is a straightforward process. It does not involve many technicalities. Here, is an example:

ATMs system is very simple as customers need to press some buttons to receive cash. However, there are multiple security layers that any ATM system needs to pass. This helps to prevent fraud and provide cash or need details to banking customers.

Below given is a UML Class Diagram example:



CLASS DIAGRAM IN SOFTWARE DEVELOPMENT LIFECYCLE

Class diagrams can be used in various software development phases. It helps in modeling class diagrams in three different perspectives.

1. **Conceptual perspective:** Conceptual diagrams are describing things in the real world. You should draw a diagram that represents the concepts in the domain under study. These concepts related to class and it is always language-independent.

2. **Specification perspective:** Specification perspective describes software abstractions or components with specifications and interfaces. However, it does not give any commitment to specific implementation.

3. **Implementation perspective:** This type of class diagrams is used for implementations in a specific language or application. Implementation perspective, use for software implementation.

BEST PRACTICES OF DESIGNING OF THE CLASS DIAGRAM:

Class diagrams are the most important UML diagrams used for software application development. There are many properties which should be considered while drawing a Class Diagram. They represent various aspects of a software application.

Here, are some points which should be kept in mind while drawing a class diagram:

1. The name given to the class diagram must be meaningful. Moreover, It should describe the real aspect of the system.
2. The relationship between each element needs to be identified in advance.
3. The responsibility for every class needs to be identified.
4. For every class, minimum number of properties should be specified. Therefore, unwanted properties can easily make the diagram complicated.
5. User notes should be included whenever you need to define some aspect of the diagram. At the end of the drawing, it must be understandable for the software development team.
6. Lastly, before creating the final version, the diagram needs to be drawn on plain paper. Moreover, It should be reworked until it is ready for final submission.

vulmshelp.com

Constraints in DBMS

Constraints are used to specify **rules for the data** in a table. Constraints enforce limits to the data that can be inserted/updated/deleted from a table. The purpose of constraints is **to maintain the data integrity** during an update /delete / insert into a table.

Types of constraints

- NOT NULL
- UNIQUE
- DEFAULT
- CHECK
- Key Constraints – PRIMARY KEY, FOREIGN KEY
- Domain constraints
- Mapping constraints

NOT NULL:

NOT NULL constraint makes sure that a column **does not hold NULL value**. When we don't provide value for a particular column while inserting a record into a table, it takes NULL value by default. By specifying NULL constraint, we can be sure that a particular column(s) cannot have NULL values.

UNIQUE:

UNIQUE Constraint enforces a column or set of columns to have unique values. If a column has a unique constraint, it means that particular column **cannot have duplicate values** in a table.

DEFAULT:

The DEFAULT constraint **provides a default value to a column** when there is no value provided while inserting a record into a table.

CHECK:

This constraint is used for **specifying range of values for a particular column** of a table. When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range.

```
CREATE TABLE STUDENT (
ROLL_NO INT NOT NULL CHECK (ROLL_NO >1000) ,
```

vulmshelp.com

```
STU_NAME VARCHAR (35) NOT NULL,
STU_AGE INT NOT NULL,
 EXAM_FEE INT DEFAULT 10000,
STU_ADDRESS VARCHAR (35) UNIQUE, PRIMARY KEY
(ROLL_NO)
);
```

Key Constraints:

PRIMARY KEY:

Primary key uniquely identifies each record in a table. It must have **unique values** and **cannot contain duplicate & null value**.

FOREIGN KEY:

Foreign keys are the columns of a table **that points to the primary key of another table**. They act as a cross-reference between tables.

Domain constraints:

Each table has certain set of columns and each column allows a **same type of data**, based on its **data type**. The column does not accept values of any other data type. Domain constraints are **user defined data type** and we can define them like this:

Domain Constraint = data type + constraints (NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK

/ DEFAULT)

Mapping Constraints / Mapping Cardinality**:**

Mapping constraints can be explained in terms of mapping cardinality:

One to One: An entity of entity-set A can be associated with at most **one entity of entity-set B**

and an entity in entity-set B can be associated with at most one entity of

vulmshelp.com

entity-set A.

One to Many: An entity of entity-set A can be associated with **any number of entities of entity- set B** and an entity in entity-set B can be associated with at most one entity of entity-set A.

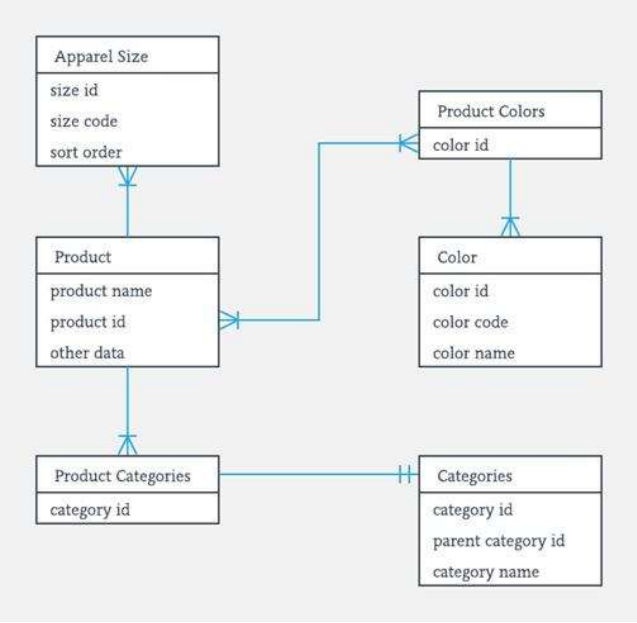
Many to One: An entity of entity-set A can be associated with at most one entity of entity-set B and an entity in entity-set B can be associated with **any number of entities of entity-set A**.

Many to Many: An entity of entity-set A can be associated with **any number of entities of entity- set B** and an entity in entity-set B can be associated with **any number of entities of entity-set A**.

HISTORY OF ER MODELS

ER diagrams are a visual tool which is helpful to represent the ER model. It was proposed by Peter Chen in 1971 to create a uniform convention which can be used for relational database and network. He aimed to use an ER model as a conceptual modeling approach.

Entity relationship diagram displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.



imshelp.com

Facts about ER Diagram Model:

- ER model allows you to draw **Database** Design
- It is an easy-to-use graphical tool for modeling data

- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities

WHY USE ER DIAGRAMS?

Here, are prime reasons for using the ER Diagram

- Helps you to define terms related to **entity relationship modeling**
- Provide a preview of how all your tables should connect, what fields are going to be on each table

- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram

RDBMS Concepts:

Cardinality – Cardinality refers to the **relationship between two tables.** Relationship can be of four types as we have already read in **ERD**.

Table – A table is a collection of data represented in rows and columns. Each table has a name in database.

Domain – A domain is a **set of permitted values for an attribute** in table. We specify domain of attribute while creating a table. For example, In the table "STUDENT", the Student_Id field has integer domain so that field cannot accept values that are not integers.

Schema – Design of a database is called the schema.

Instance – At a particular moment the data stored in database is called the instance.

vulmshelp.com

Why we need a Key?

Here are some reasons for using SQL key in the DBMS system.

- Keys help us **to identify any row of data** in a table. In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated.
- Keys allows us to **establish a relationship** between and identify the relation between tables
- Keys help us to enforce **identity and integrity** in the relationship.

Types of Keys in Database Management System

There are mainly seven different types of Keys in DBMS and each key has its different functionality:

- **Super Key -** Is a set of attributes (Group of one/more keys) that uniquely identify rows in a table.
- **Primary Key -** Is a column or group of columns that uniquely identify every row in that table.
- **Candidate Key -** Candidate Key is a super key with no repeated attributes. If a Super Key does not have any duplicate attribute, it is known as a Candidate Key.
- **Alternate Key -** Any key that is capable of being Primary Key, but at the moment is not the Primary Key, is known as an Alternate Key. It is a candidate key that has not been selected as the Primary Key.
- **Foreign Key -** Is a column that creates a relationship between two tables. Foreign Keys help us in establishing relationships with other tables. It is also called Referential Integrity.
- **Compound Key -** Is a primary key that contain two or more columns that allow us to distinctly identify a particular row. It is also known as composite key.
- **Surrogate Key-** A situation may arise where a particular table does not have a Primary Key. In this case, we use a Surrogate Key, which is an artificial key that can distinctly identify every row in the table. Surrogate Keys are used specifically when we do not have a natural primary key.

vumshelp.com

Normalization in DBMS:

1NF, 2NF, 3NF AND BCNF DBMS

Normalization is a **process of organizing data in database to avoid data redundancy**, insertion anomaly, update anomaly & deletion anomaly.
Let's discuss about anomalies first;

Anomalies in DBMS

There are three types of anomalies that occur when the database is not normalized. These are – **Insertion, update and deletion anomaly**.

Example: Suppose a manufacturing company stores the employee details in a table named **employee**

that has four attributes:

emp_id for storing employee's id,

emp_name for storing employee's name,

emp_address for storing employee's address and

emp_dept for storing the department details in which the employee works.

emp_id	emp_name	emp_addres	emp_dept
101	JUNAID	Islamabad	D001
101	MALIK	Karachi	D002
123	ALIK	Lahore	D890
166	FRAZ	Multan	D900
166	Shoaib	JP	D004

Update Anomaly: In the above table we have two rows for employee **Rick** as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will

vulmshelp.com

become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.

Insert Anomaly: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

Delete Anomaly: Suppose, if at a point of time the company closes the department **D890** then deleting the rows that are having emp_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.

To overcome these anomalies, we need to normalize the data

Normalization

Here are the most commonly used normal forms:

- First normal form(**1NF**)
- Second normal form(**2NF**)
- Third normal form(**3NF**)
- Boyce & Codd normal form (**BCNF**)

First normal form (1NF)

As per the rule of first normal form, an attribute (column) of a table **cannot hold multiple values**. It should hold only atomic values.

Example: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

vulmshelp.com

mp_id	emp_name	emp_address	emp_mobile
101	JUNAID	Islamabad	03041659294
102	MALIK	Karachi	8812121212 9900012222
103	ALISHA	MULTAN	7778881212
104	ALI	LAHORE	8123450987 9990000123

Two employees (MALIK & ALI) are having two mobile numbers so the company stored them in the same field as you can see in the table above. This table is not in 1NF as the rule says "**each attribute of a table must have atomic (single) values**", the emp_mobile values for employees MALIK & ALI violates that rule.
To make the table complies with 1NF we should have the data like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	ISLAMABAD	03041659294
102	MALIK	LAHORE	8812121212
102	MALIK	LAHORE	9900012222
103	Ron	Karachi	7778881212
104	ALI	MULTAN	8123450987
104	ALI	MULTAN	9990000123

Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)

vulmshelp.com

- No **non-prime attribute** is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

Example: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {teacher_id, subject}

Non-prime attribute: teacher_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non- prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".

To make the table complies with 2NF we can break it in two tables like this:

Teacher Detail Table

teacher_id	**teacher_ag**
111	38
222	38

vulmshelp.com

333	40

Teacher Subject Table

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as **non-prime attribute**. An attribute that is a part of one of the candidate keys is known as **prime attribute**.

Example: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:

emp_id	emp_nam	emp_zip	emp_state	emp_city	emp_distric
1001	John	282005	ISL	Islamaba	Islamabad
1002	Jon	222008	LHR	Karachi	Karachi
1006	JUNAID	282007	BWP	Multan	MULTAN
1101	Malik	292008	JP	BWP	Bahawalpur
1201	ALI	222999	MLTN	JP	Multan

vulmshelp.com

Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}
Candidate Keys: {emp_id}

Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.
Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF. To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

Employee Table:

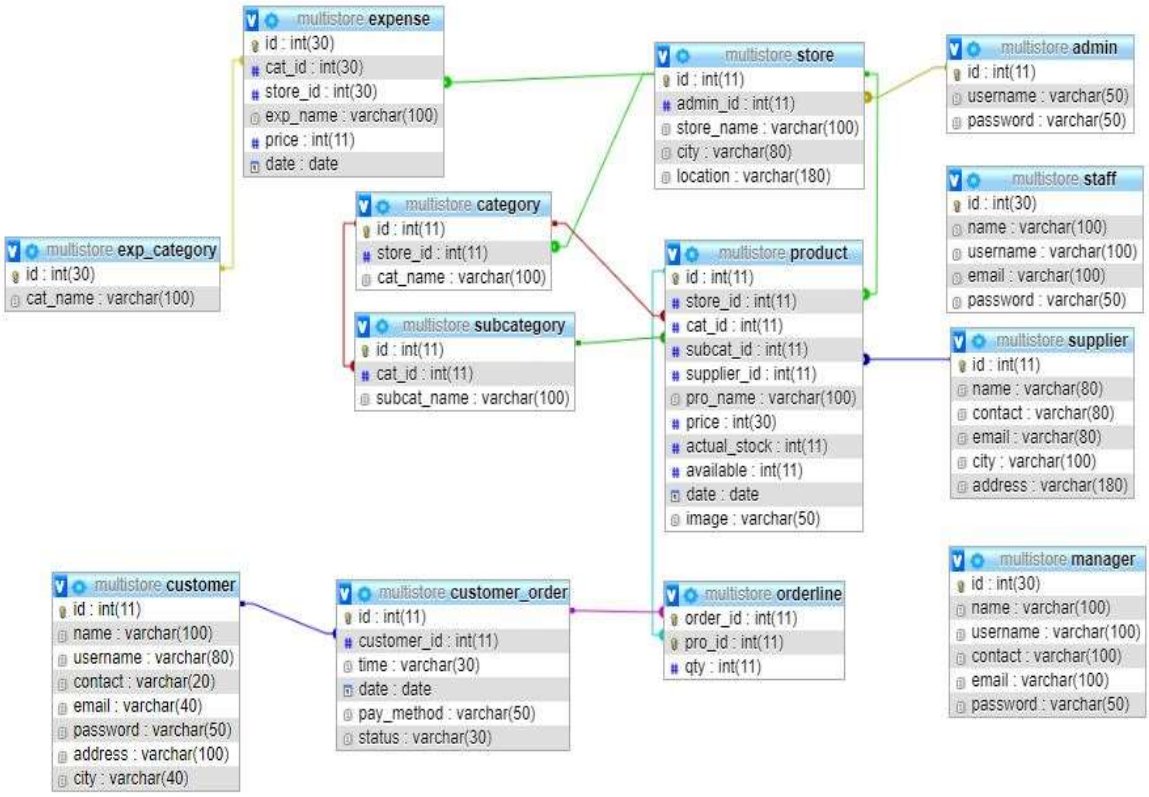
emp_i	emp_nam	emp_zi
1001	John	28200
1002	Jon	22200
1006	JUNAID	28200
1101	Malik	29200
1201	ALI	22299

Employee Zip Table:

emp_zi	emp_stat	emp_cit	emp_distri
28200	ISL	Islamaba	Islamabad
22200	LHR	Karachi	Karachi
28200	BWP	Multan	MULTAN
29200	JP	BWP	Bahawalpur
22299	MLTN	JP	Multan

vulmshelp.com

Database Diagram

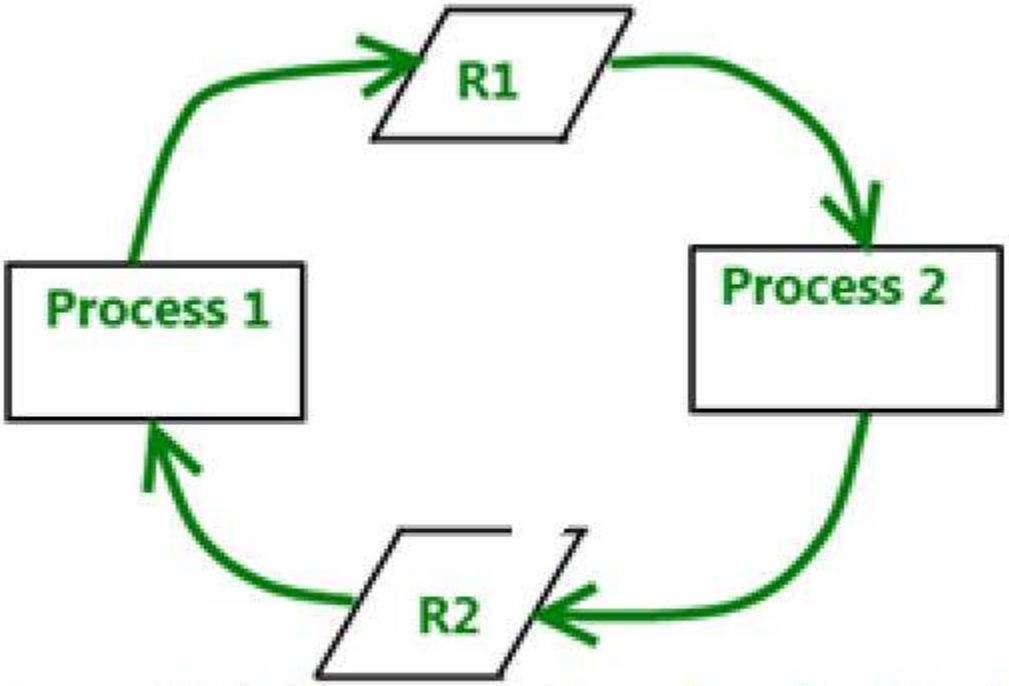


Boyce Code Normal Form (BCNF)

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Deadlock in DBMS:

A deadlock is a condition wherein two or more tasks are waiting for each other in order to be finished but none of the task is willing to give up the resources that other task needs. In this situation no task ever gets finished and is in **waiting state forever**.



Process P1 holds resource R2 and requires R1 while
Process P2 holds resource R1 and requires R2.

View of Data in DBMS

Abstraction is one of the main features of database systems. Hiding irrelevant details from user and providing abstract view of data to users, helps in easy and efficient user-database interaction. The top level of architecture is "view level". The view level provides the "**view of data**" to the users and hides the irrelevant details such as data relationship, database schema, constraints, security etc. from the user.

Concurrency Control in DBMS

When more than one transaction is running simultaneously there are chances of a conflict to occur which can leave database to an inconsistent state. To handle these conflicts, we need concurrency control in DBMS, which allows transactions to run simultaneously but handles them in such a way so that the integrity of data remains intact.

Solution of Conflicts: Locks

A lock is kind of a mechanism that ensures that the integrity of data is maintained. There are two types of a lock that can be placed while accessing the data so that the concurrent transaction cannot alter the data while we are processing it.

1. Shared Lock(S)
2. Exclusive Lock(X)

1. Shared Lock(S): Shared lock is placed when we are reading the data, multiple shared locks can be placed on the data but when a shared lock is placed no exclusive lock can be placed.

2. **Exclusive Lock(X)**: Exclusive lock is placed when we want to read and write the data. This lock allows both the read and write operation, once this lock is placed on the data no other lock (shared or Exclusive) can be placed on the data until Exclusive lock is released

What is SQL:

SQL or Structured Query Language is a critical tool for data professionals. It is undoubtedly the most important language for getting a job in the field of data analysis or data sciences.

Millions of data points are being generated every minute and raw data does not have any story to tell. After all this data gets stored in databases and professionals use SQL to extract this data for further analysis.

In this blog post, we will cover what is SQL and how does it work. SQL is fairly simple, thanks to easy syntaxes of this language. Let's dig into understanding this Structured Query Language by asking this simple question.

What is SQL and How Does it Work?

SQL is the most common language for extracting and organizing data that is stored in a relational database. A database is a table that consists of rows and columns. SQL is the language of databases. It facilitates retrieving specific information from databases that are further used for analysis. Even when the analysis is being done on another platform like Python or R, SQL

would be needed to extract the data that you need from a company's database.

The query reaches SQL server, where it compiles in three phases; Parsing, Binding and Optimization.

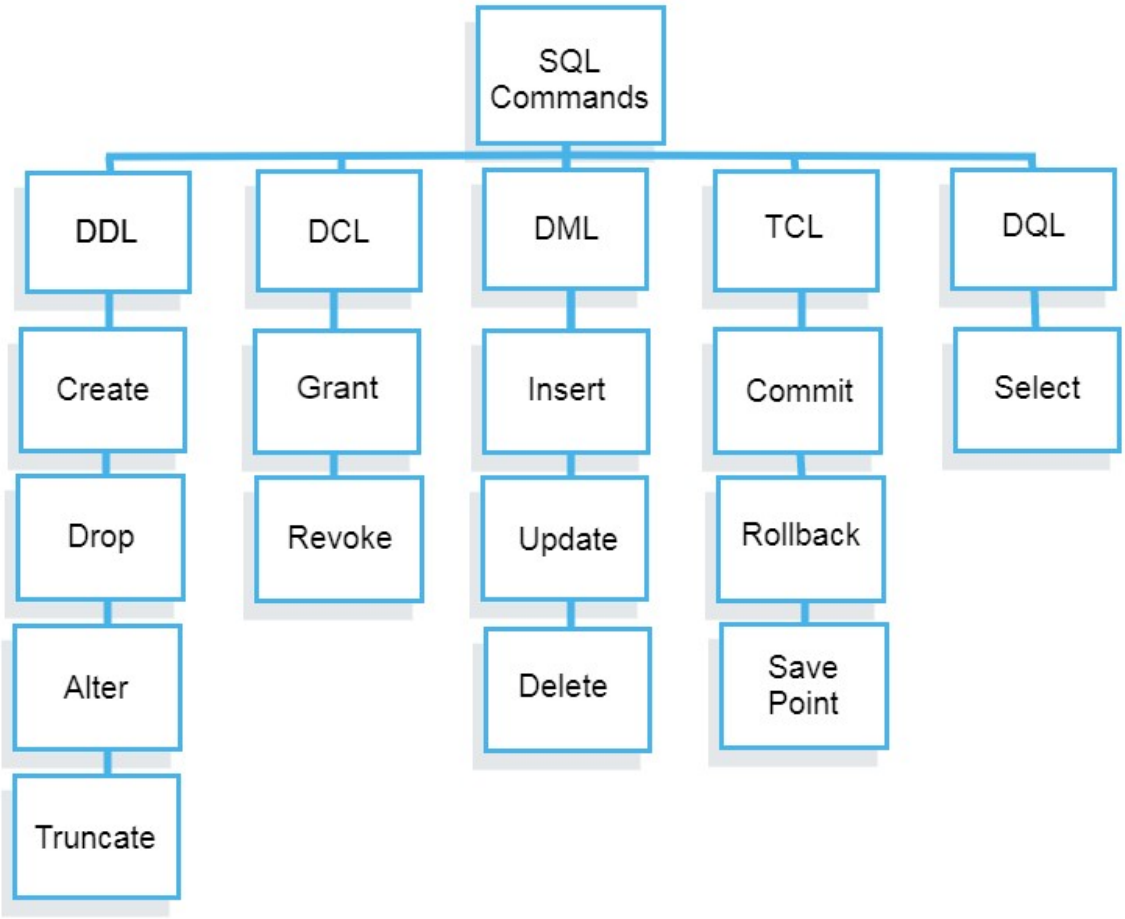
- Parsing – A process to check the syntax
- Binding – A process to check the query semantics
- Optimization – A process to generate the query execution plan

What is SQL Used for?

Now that we understand what is SQL and how does it work, let's try to see what SQL can do. This programming language has various uses for data analysts & data science professionals. It is particularly helpful because it can:

- Execute queries against a database
- Retrieve data from a database
- Insert records into a database
- Update records in a database
- Delete records from a database
- Create new databases, or new tables in a database
- Create stored procedures & views in a database
- Set permissions on tables, procedures, and views

Types of SQL Statement:



Here are some commands that we will be learning about, and later practicing as well.

Command	Description
CREATE	Creates a new table, a view of a table, or another object in the database.
ALTER	Modifies an existing database object, such as a table.

DROP	Deletes an entire table, a view of a table or other objects in the database.
SELECT	Retrieves certain records from one or more tables.
INSERT	Creates a record.
UPDATE	Modifies a record.
DELETE	Deletes a record.
GRANT	Gives a privilege to users.
REVOKE	Takes back privileges granted from users.

If you notice, the commands 'create, alter & drop' define the complete structure of data. They come under the 'Data Definition Language' set of commands. Similarly, 'select, insert, update, delete' fall under Data Manipulation Language & 'grant, revoke' come under Data Control Language set of commands.

SQL COMMANDS

SQL is a programming language for Relational Databases. It is designed over relational algebra and tuple relational calculus. SQL comes as a package with all major distributions of RDBMS.

SQL comprises both data definition and data manipulation languages. Using the data definition properties of SQL, one can design and modify database schema, whereas data manipulation properties allows SQL to store and retrieve data from database.

vulms.vlp.com

SQL Commands:

The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into groups based on their nature:

DDL - Data Definition Language:

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.
CREATE	Creates a new table, a view of a table, or other object in database

DML - Data Manipulation Language:

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

Command	Description
SELECT	Retrieves certain records from one or more tables
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

DCL - Data Control Language:

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

DQL (Data Query Language):

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it. We can define DQL as follows it is a component of SQL statement that allows getting data from the database and imposing order upon it. It includes the SELECT statement. This command allows getting the data out of the database to perform operations with it. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e. a front-end.

List of DQL:

.

Command	Description
SELECT	It is used to retrieve data from the database.

Data Definition Language:

The DDL section is used for creating database objects, such as tables. In practice, people often use a GUI for creating tables and so on, so it is less common to hand-write DDL statements than it used to be. SQL uses the following set of commands to define database schema

CREATE

Creates new databases, tables and views from RDBMS.

```
CREATE DATABASE databasename; CREATE TABLE table_name (
column1 datatype, column2 datatype, column3 datatype,
);
```

DROP

Drops commands, views, tables, and databases from RDBMS. For example,.

```
DROP DATABASE databasename; DROP TABLE table_name;
```

ALTER

Modifies database schema.

```
ALTER TABLE table_name
ADD column_name datatype;
```

Difference Between truncate, Drop and Delete

DELETE:

DELETE removes some rows if WHERE clause is used

Drop:

vulmshelp.com

Removes a table from the database. Table structures, indexes, privileges, constraints will also be removed.

Truncate:

It Removes all rows from a table, but the table structures and its columns, constraints, indexes remain.

NOTE:

For PHP MySQL NOTES Contact on **0304-1659294**

(Back-end Language)

Or

http://vulmshelp.com/

vulmshelp.com

What is a Stored Procedure?

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

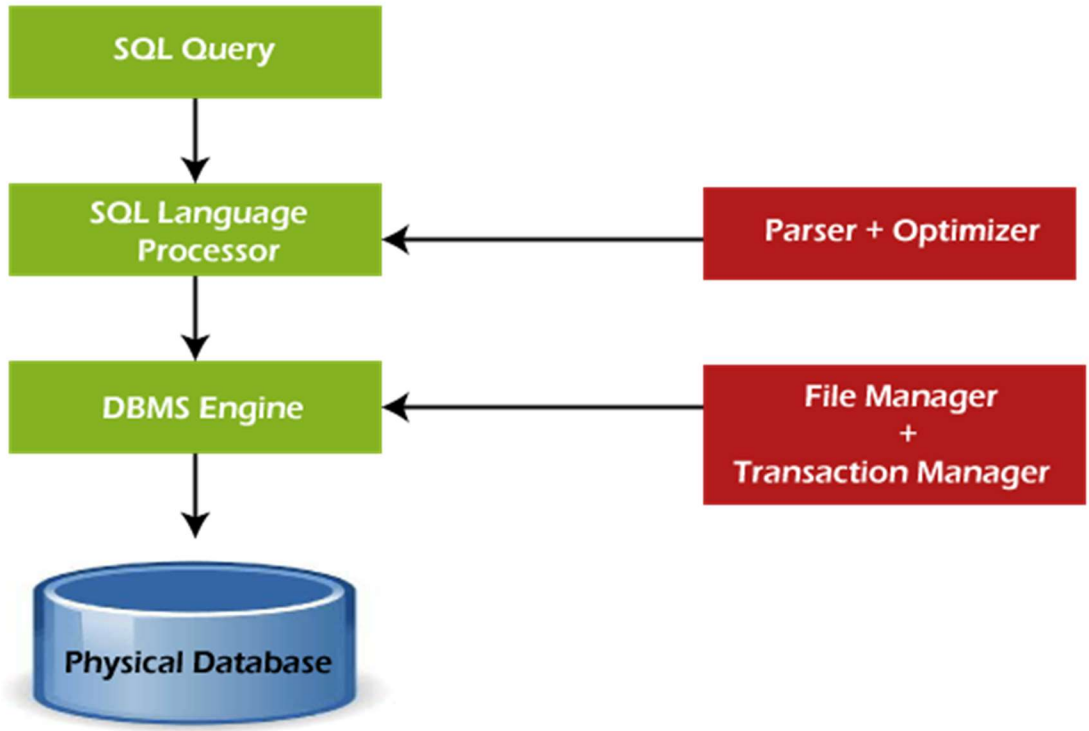
So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

It includes translations on high level Queries into low level expressions that can be used at physical level of file system, query optimization and actual execution of query to get the actual result.

SQL Process Steps

1. Getting Data (*From, Join*)
2. Row Filter (*Where*)
3. Grouping (*Group by*)
4. Group Filter (*Having*)
5. Return Expressions (*Select*)
6. Order & Paging (*Order by & Limit / Offset*)



What is a NoSQL database?

When people use the term "NoSQL database," they typically use it to refer to any non-relational database. Some say the term "NoSQL" stands for "non SQL" while others say it stands for "not only SQL." Either way, most agree that NoSQL databases are databases that store data in a format other than relational tables.

As storage costs rapidly decreased, the amount of data that applications needed to store and query increased. This data came in all shapes and sizes structured, semi-structured, and polymorphic and defining the schema in advance became nearly impossible. NoSQL databases allow developers to store huge amounts of unstructured data, giving them a lot of flexibility.

Types of NoSQL databases

Over time, four major types of NoSQL databases emerged: document databases, key-value databases, wide-column stores, and graph databases.

- Document databases store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, Booleans, arrays, or objects.
- Key-value databases are a simpler type of database where each item contains keys and values.
- Wide-column stores store data in tables, rows, and dynamic columns.
- Graph databases store data in nodes and edges. Nodes typically store information about people, places, and things, while edges store information about the relationships between the nodes.

Difference between RDBMS and NoSQL databases

While a variety of differences exist between relational database management systems (RDBMS) and NoSQL databases, one of the key differences is the way the

vulmshelp.com

data is modeled in the database. In this section, we'll work through an example of modeling the same data in a relational database and a NoSQL database. Then, we'll highlight some of the other key differences between relational databases and NoSQL databases.

RDBMS vs NoSQL: Data Modeling Example

Let's consider an example of storing information about a user and their hobbies. We need to store a user's first name, last name, cell phone number, city, and hobbies.


```
"_id": 1,
  "first_name": "JUNAID",
  "last_name": "MALIK",
  "cell": "03041659294",
  "city": "MULTAN",
  "hobbies": ["reading", "working"]
```


functional dependency

A functional dependency is a constraint that specifies the relationship between two sets of attributes where one set can accurately determine the value of other sets. It is denoted as **X → Y**, where X is a set of attributes that is capable of determining the value of Y. The attribute set on the left side of the arrow, **X** is called **Determinant**, while on the right side, **Y** is called the **Dependent**. Functional dependencies are used to mathematically express relations among database entities and are very important to understand advanced concepts in Relational Database System and understanding problems in competitive exams like Gate.

junaidhelp.com

Types of Functional dependencies in DBMS:

- Trivial functional dependency
- Non-Trivial functional dependency
- Multivalued functional dependency
- Transitive functional dependency

What is Relational Model?

Relational Model (RM) represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.
The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the

Some popular Relational Database management systems are:

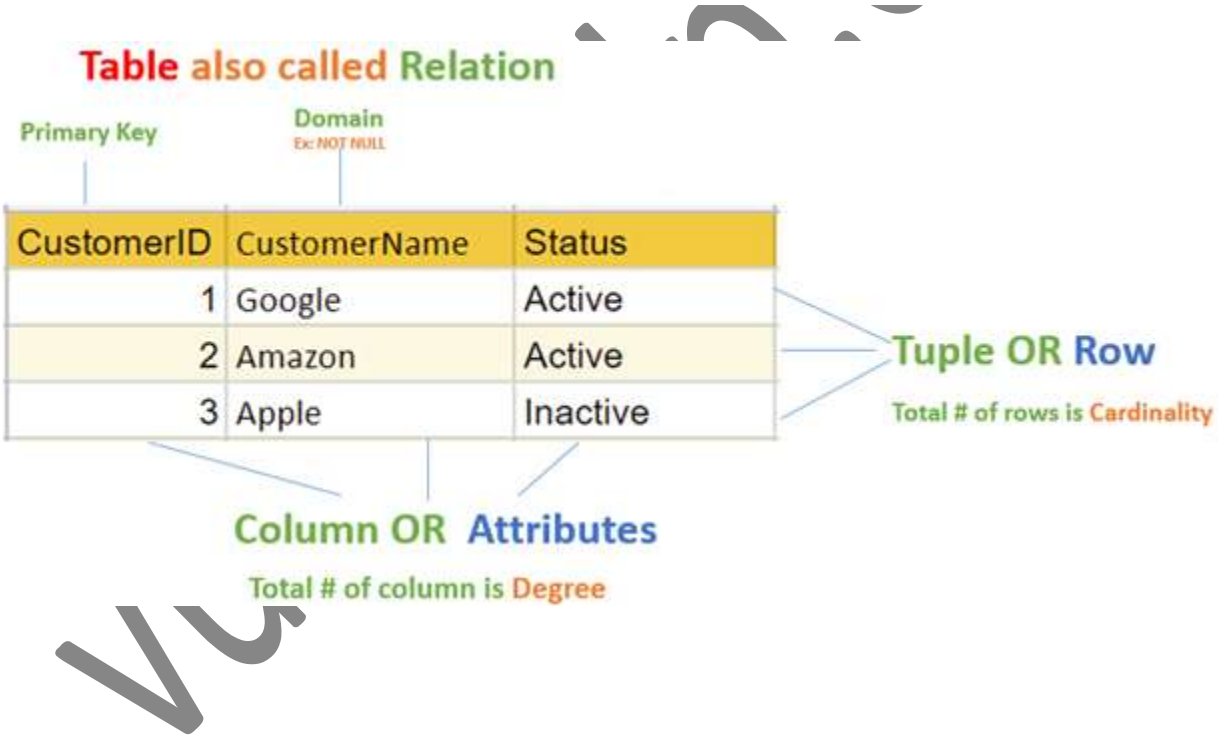
- DB2 and Informix Dynamic Server – IBM
- Oracle and RDB – Oracle
- SQL Server and Access – Microsoft

Relational Model Concepts

- **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME, etc.
- **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
- **Tuple** – It is nothing but a single row of a table, which contains a single record.
- **Relation Schema:** A relation schema represents the name of the relation with its attributes.

- **Degree:** The total number of attributes which in the relation is called the degree of the relation.
- **Cardinality:** Total number of rows present in the Table.
- **Column:** The column represents the set of values for a specific attribute.
- **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
- **Relation key** – Every row has one, two or multiple attributes, which is called relation key.
- **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain

RELATIONAL MOEL:



Relational Integrity Constraints

Relational Integrity constraints in DBMS are referred to conditions which must be present for a valid relation. These Relational constraints in DBMS are derived from the rules in the mini-world that the database represents.

There are many types of Integrity Constraints in DBMS. Constraints on the Relational database management system is mostly divided into three main categories are:

1. Domain Constraints
2. Key Constraints
3. Referential Integrity Constraints

Domain Constraints

Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.

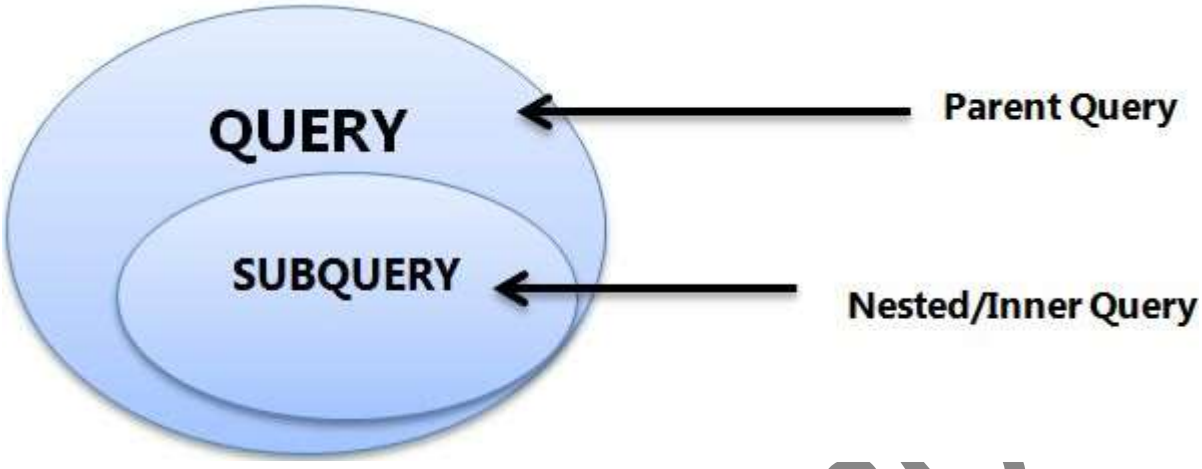
Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

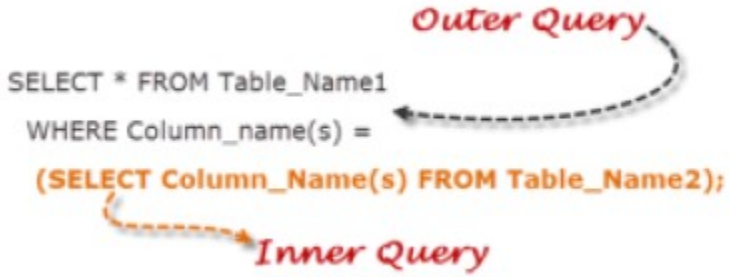
```
Create DOMAIN CustomerName
CHECK (value not NULL)
```

About Queries and Subqueries

A **query** is an operation that retrieves data from one or more tables or views. In this reference, a top-level SELECT statement is called a **query**, and a query nested within another SQL statement is called a **subquery**.

This section describes some types of queries and subqueries and how to use them. The top level of the syntax is shown in this chapter. Please refer to SELECT for the full syntax of all the clauses and the semantics of this statement.





JOIN in SQL

A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are:

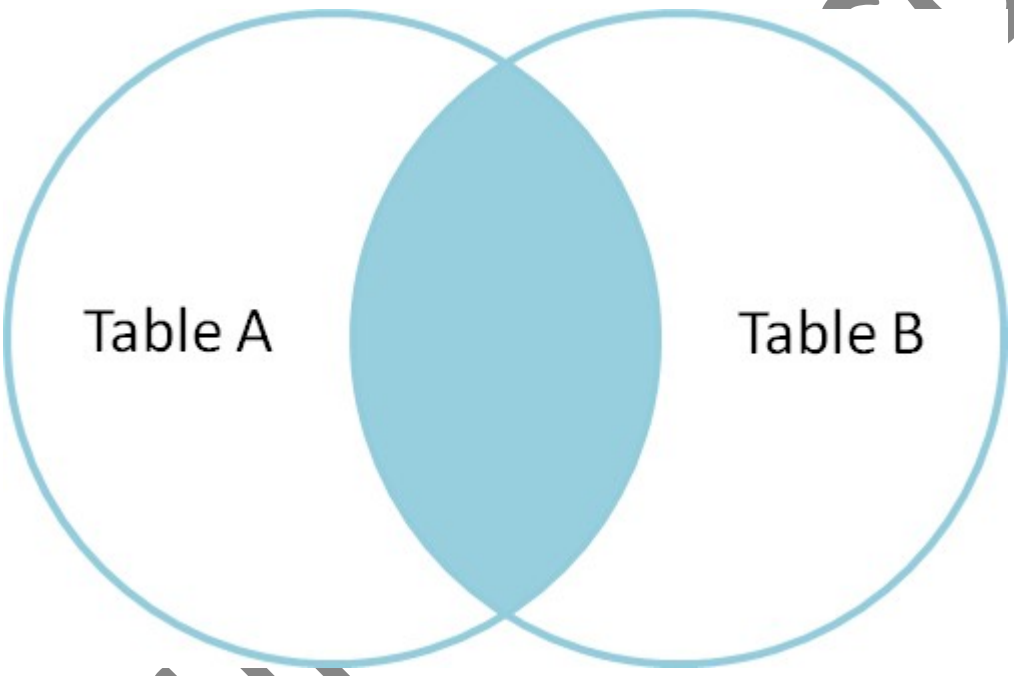
- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

INNER JOIN:

The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be same.

SYNTAX:

```
SELECT table1.column1,table1.column2,table2.column1,
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```




```
SELECT StudentCourse.COURSE_ID, Student.NAME, Student.AGE
FROM Student
INNER JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

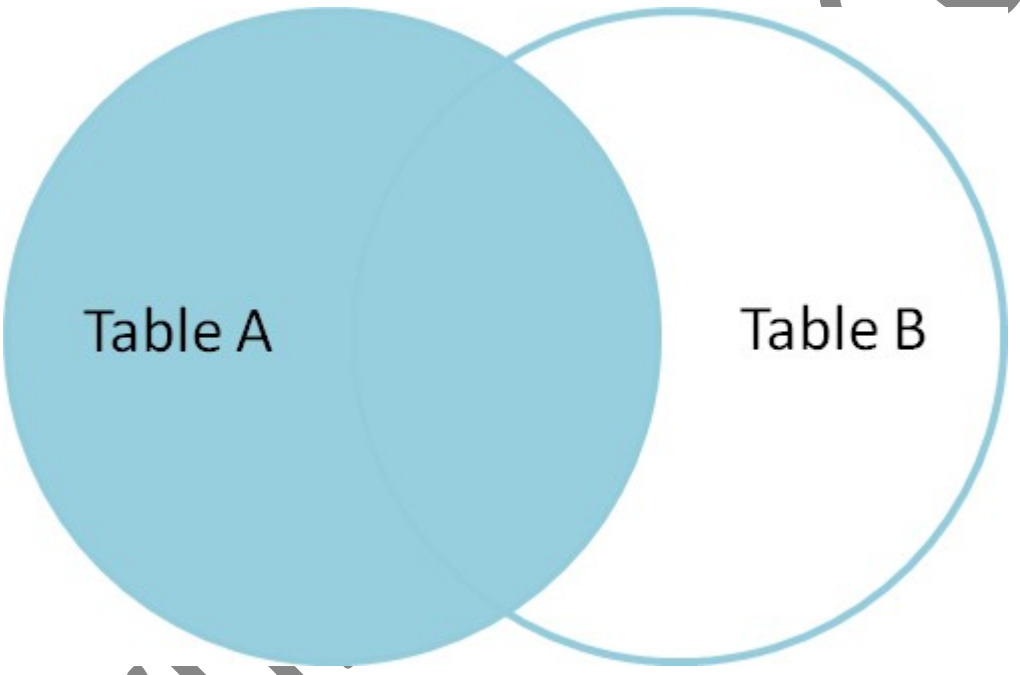
LEFT JOIN:

This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no

matching row on right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

SYNATX:

```
SELECT table1.column1,table1.column2,table2.column1,
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```



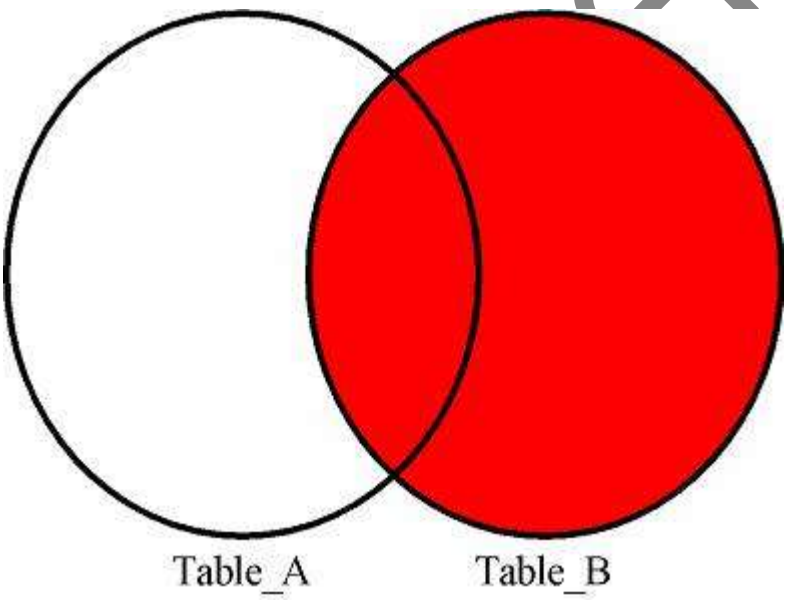

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

RIGHT JOIN:

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join. The rows for which there is no matching row on left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

SYNATX:

```
SELECT table1.column1,table1.column2,table2.column1,
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```




```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
```

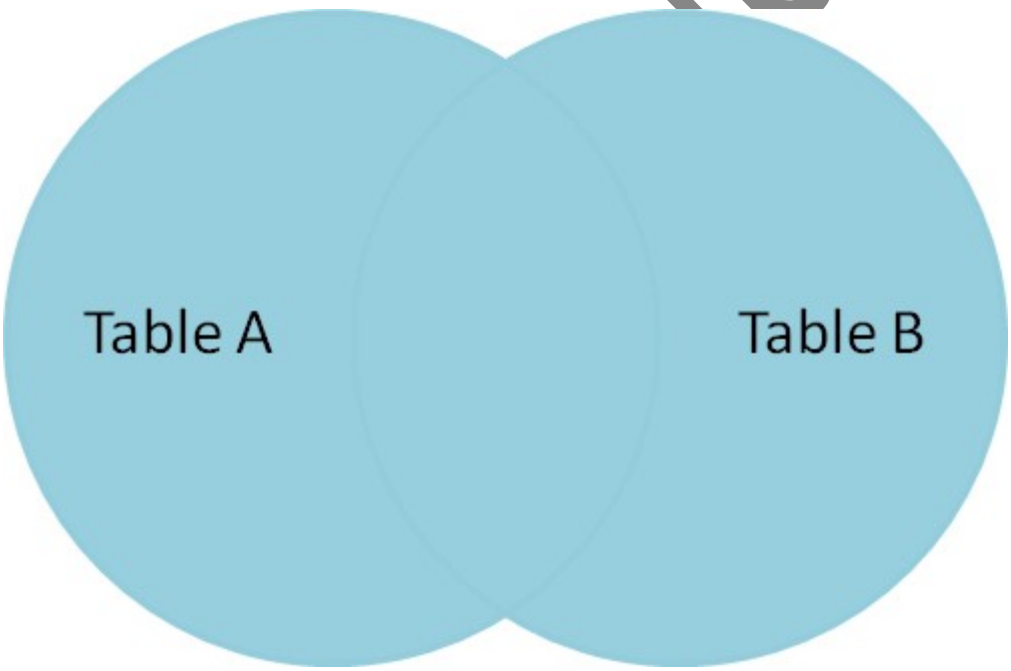
help.com

ON StudentCourse.ROLL_NO = Student.ROLL_NO;

FULL JOIN: FULL JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain *NULL* values.

SYNATX:

```
SELECT table1.column1,table1.column2,table2.column1,
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```



SELECT Student.NAME,StudentCourse.COURSE_ID

```
FROM Student
FULL JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

UNION OPERATOR

The UNION operator is used to combine the result-set of two or more SELECT statements.

- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL:

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

Note: The column names in the result-set are usually equal to the column names in the first SELECT statement.

GROUP BY & ORDER BY STATEMENT

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions COUNT(), MAX(), MIN(), SUM(), AVG() to group the result-set by one or more columns.

vulmchelp.com

The SQL ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default.

SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s);
SELECT column-list FROM table_name [WHERE condition] [ORDER BY column1, column2, .. columnN];

HAVING CLAUSE

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions. The HAVING Clause enables you to specify conditions that filter which group results appear in the results. The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause The HAVING clause must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used. The following code block has the syntax of the SELECT statement including the HAVING clause.

SELECT column1, column2 FROM table1, table2 WHERE [conditions] GROUP BY column1, column2 HAVING [conditions] ORDER BY column1, column2
OR
SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) HAVING condition ORDER BY column_name(s);

INDEXING:

Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexes are special lookup tables that the database search engine can use to speed up data retrieval. An index in a database is very similar to an index in the back of a book.

vulmihelp.com

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

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

The basic syntax of a CREATE INDEX is as follows.
CREATE INDEX index_name ON table_name;

Data Integrity in a Database

Data integrity refers to the overall accuracy, completeness, and reliability of data. It can be specified by the lack of variation between two instances or consecutive updates of a record, indicating that your information is error-free. It also corresponds to the security of data pertaining to regulatory compliance.

Data integrity is preserved by an array of error-checking and validation procedures, rules, and principles executed during the integration flow designing phase. These checks and correction procedures are based on a predefined set of business rules. For instance, the rules dictate to filter out the data with an incorrect date or time value.

The question then arises, why is it imperative to maintain data integrity in a database?

The importance of maintaining data integrity is evident when creating relationships between disparate data elements. It ensures that the data transferring from one stage to another is accurate and error-free. This way, when the information is written into the database, its trustworthiness is implied regardless of how often data is accessed.

Types of Data Integrity

Data integrity is applied in all databases can be categorized into two main types:

Physical Integrity

Protecting data against external factors, such as natural calamities, power outages, or hackers falls under the domain of physical integrity. Moreover, human faults, storage attrition, and several other problems can make it unmanageable for data operators to obtain information from a database.

Logical Integrity

It concerns the rationality of data present within the relational database. Logical integrity constraints can be categorized into four types:

Entity Integrity

It depends on the making of primary keys or exclusive values that classify data items. The purpose is to ensure that data is not recorded multiple times (i.e. each data item is unique), and the table has no null fields.

Entity integrity is a critical feature of a relational database that stores data in a tabular format, which can be interconnected and used in various ways.

Referential Integrity

It denotes a series of procedures that ensure proper and consistent storage and usage of data. Referential integrity ensures that only the required alterations, additions, or removals happen via rules implanted into the database's structure about how foreign keys are used.

These rules might include conditions that remove duplicate data records, warrant that data is precise, and/or prohibit recording data that is unsuitable.

Domain Integrity

It's an assortment of procedures that ensures the precision of every data item is maintained in a domain. Here, a domain is defined as a set of suitable values that a column is permitted to enclose.Domain integrity encompasses rules and other processes that restrict the format, type, and volume of data recorded in a database. It ensures that every column in a relational database is in a defined domain.

User-Defined Data Integrity

It comprises the rules defined by the operator to fulfill their specific requirements. At times entity, referential, and domain integrity are not enough to refine and secure data. Time in time again, particular business rules must be considered and integrated into data integrity processes to meet enterprise standards.

Regards:

Junaid Malik

BS (Computer Science)

0304-1659294

Junaidfazal08@gmail.com

For More Visit:

http://vulmshelp.com/