PSR ENGINEERING COLLEGE
SIVAKASI
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SUBJECT NAME	**: DATABASE MANAGEMENT SYSTEMS**
SUBJECT CODE	**: 191CS42**
STAFF NAME	**:AMUTHA J**
YEAR/SEM	**:II/V**

UNIT-1

IntroductiontoDatabaseManagementSystem

Asthenamesuggests,thedatabasemanagement systemconsistsoftwoparts.They are:

1. Databaseand
2. ManagementSystem

WhatisaDatabase?

Tofindoutwhatdatabaseis,wehavetostartfrom data,whichisthe basicbuildingblockof anyDBMS.

Data:Facts,figures,statisticsetc.havingnoparticularmeaning(e.g.1,ABC,19etc).
Record: Collection of related data items, e.g. in the above example the three data items had nomeaning. But if we organize them in the following way, then they collectively represent meaningfulinformation.

Roll	Name	Age
1	ABC	19

Table or **Relation**: Collection of relatedrecords.

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	28

The columns of this relation are called **Fields**, **Attributes** or **Domains**. The rows arecalled**Tuples** or **Records**.
Database:Collectionofrelatedrelations.Considerthefollowingcollectionoftables:

T1

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	28

T2

Roll	Address
1	KOL
2	DEL
3	MUM

T3 **T4**

Roll	Year
1	I
2	II
3	I

—

Year	Hostel
I	H1
II	H2

***Age*and *Hostel*attributes arein differenttables.**

Adatabasein aDBMScouldbe viewedby lotsofdifferent peoplewith differentresponsibilities.

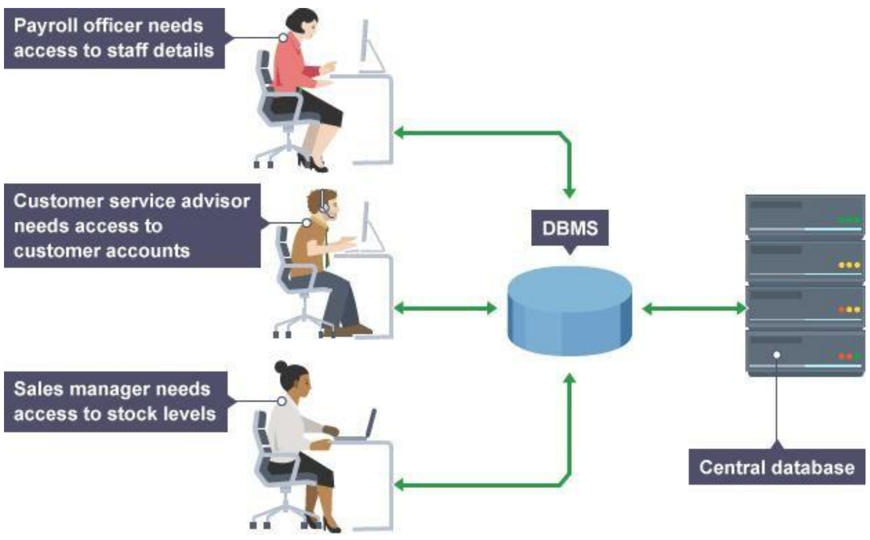

Payroll officer needs access to staff details
Customer service advisor needs access to customer accounts
Sales manager needs access to stock levels
DBMS
Central database

Figure1.1:EmpolyeesareaccessingDatathroughDBMS

Forexample,withinacompanytherearedifferentdepartments,aswellascustomers,whoeachneedto see different kinds of data. Each employee in the company will have different levels of access to thedatabasewith their own customized **front-end**application.

In a database, data is organized strictly in row and column format. The rows are called **Tuple** or**Record**. The data items within one row may belong to different data types. On the other hand, thecolumns are often called **Domain** or **Attribute**. All the data items within a single attribute are of thesamedata type.

WhatisManagementSystem?

A **database-management system** (DBMS) is a collection of interrelated data and a set of programs toaccess those data. This is a collection of related data with an implicit meaning and hence is a database.Thecollectionofdata,usuallyreferredtoasthe**database**,containsinformationrelevanttoan enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database informationthat is both *convenient* and *efficient*. By **data,**we mean known facts that can be recorded and that haveimplicitmeaning.

Database systems are designed to manage large bodies of information. Management of data involvesboth defining structures for storage of information and providing mechanisms for the manipulation ofinformation. In addition, the database system must ensure the safety of the information stored, despitesystem crashes or attempts at unauthorized access. If data are to be shared among several users, thesystemmust avoid possible anomalous results.

DatabaseManagementSystem(DBMS)andItsApplications:

A Database management system is a computerized record-keeping system. It is a repository or acontainer for collection of computerized data files. The overall purpose of DBMS is to allow he usersto define, store, retrieve and update the information contained in the database on demand. Informationcanbe anything that is of significance to an individualororganization.

Databasestouchallaspectsofourlives.Someofthemajorareasof applicationareasfollows:
1. Banking
2. Airlines
3. Universities
4. Manufacturingandselling
5. Humanresources

EnterpriseInformation
◦*Sales*:Forcustomer, product,and purchaseinformation.
◦*Accounting*:Forpayments, receipts,accountbalances, assetsand otheraccountinginformation.
◦*Humanresources*:Forinformationaboutemployees,salaries,payrolltaxes,andbenefits,andforgeneration ofpaychecks.
◦*Manufacturing*: For management of the supply chainand for tracking production of items in factories,inventories of items in warehouses and stores,and orders for items.
Online retailers: For sales data noted above plus online order tracking,generation ofrecommendationlists, and maintenanceofonline productevaluations.

◦*Banking*:Forcustomerinformation,accounts,loans, and bankingtransactions.
◦*Creditcardtransactions*: Forpurchases oncredit cardsand generationof monthlystatements.
◦*Finance*:Forstoringinformationaboutholdings,sales,andpurchasesoffinancialinstrumentssuchasstocksa ndbonds;alsoforstoringreal-timemarketdatatoenableonlinetradingbycustomersand automatedtrading bythe firm.
•*Universities*:Forstudentinformation,courseregistrations,andgrades(inadditiontostandardenterpriseinf ormation suchas human resources and accounting).
•*Airlines*:Forreservationsandscheduleinformation.Airlineswereamongthefirsttousedatabasesina geographically distributed manner.
•*Telecommunication*:Forkeepingrecordsofcallsmade,generatingmonthlybills,maintainingbalancesonp repaidcallingcards, andstoringinformationaboutthecommunicationnetworks.

PurposeofDatabaseSystems

Databasesystemsaroseinresponsetoearlymethodsofcomputerizedmanagementofcommercialdata. As an example of such methods, typical of the 1960s, consider part of a university organizationthat, among other data, keeps information about all instructors, students, departments, and courseofferings. One way to keep the information on a computer is to store it in operating system files. Toallow users to manipulate the information, the system has a number of application programs thatmanipulatethe files, including programs to:

- Addnewstudents,instructors,andcourses
- Registerstudents for courses andgenerateclass rosters
- Assigngradestostudents,computegradepointaverages(GPA),andgeneratetranscripts

Thistypical**file-processingsystem**issupportedbyaconventionaloperatingsystem.Thesystemstores permanent records in various files, and it needs different application programs to extract recordsfrom, and add records to, the appropriate files. Before database management systems (DBMSs) wereintroduced,organizationsusuallystoredinformationinsuchsystems.Keepingorganizationalinformationin a file-processing system has a numberof major disadvantages:

Data redundancy and inconsistency. Since different programmers create the files and applicationprograms over a long period, the various files are likely to have different structures and the programsmay be written in several programming languages. Moreover, the same information may be duplicatedin several places (files). For example, if a student has a double major (say, music and mathematics) theaddress and telephone number of that student may appear in a file that consists of student records ofstudents in the Music department and in a file that consists of student records of students in theMathematics department. This redundancy leads to higher storage and access cost. In addition, it maylead to **data inconsistency**; that is, the various copies of the same data may no longer agree.
Forexample,achangedstudentaddressmaybereflectedintheMusicdepartmentrecordsbutnotelsewherein the system.

Difficultyinaccessingdata.Supposethatoneoftheuniversityclerksneedstofindoutthenamesofallstudentswholivewithinaparticularpostal-codearea.Theclerkasksthedata-processingdepartmenttogeneratesuchalist.Becausethedesignersoftheoriginalsystemdidnotanticipatethisrequest,thereisnoapplicationprogramonhandtomeetit.Thereis,however,anapplicationprogramtogenerate the listof *all* students.

Dataisolation.Becausedataarescatteredinvariousfiles,andfilesmaybeindifferentformats,writingnew application programs to retrievethe appropriate data is difficult.

Integrity problems. The data values stored in the database must satisfy certain types of **consistencyconstraints**. Suppose the university maintains an account for each department, and records the balanceamount in each account. Suppose also that the university requires that the account balance of adepartment may never fall below zero. Developers enforce these constraints in the system by addingappropriate code in the various application programs. However, when new constraints are added, it isdifficulttochangetheprogramstoenforcethem.Theproblemiscompoundedwhenconstraintsinvolveseveral data items from different files.

Atomicityproblems.Acomputersystem,likeanyotherdevice,issubjecttofailure.Inmanyapplications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existedprior tothe failure.

Consideraprogramtotransfer$500fromtheaccountbalanceofdepartment*A*totheaccountbalanceofdepartment *B*.Ifasystemfailureoccursduringtheexecutionoftheprogram,itispossiblethatthe

$500 was removed from the balance of department *A* but was not credited to the balance of department*B*, resulting in an inconsistent database state. Clearly, it is essential to database consistency that eitherboththe credit and debit occur, or thatneither occur.

That is, the funds transfer must be *atomic*—it must happen in its entirety or not at all. It is difficult toensureatomicity in aconventionalfile-processing system.

Concurrent-access anomalies. For the sake of overall performance of the system and faster response,many systemsallowmultipleuserstoupdatethe datasimultaneously.Indeed,today,thelargestInternetretailersmayhavemillionsofaccessesperdaytotheirdat abyshoppers.Insuchanenvironment,interactionofconcurrentupdatesispossibleandmayresultininconsisten tdata.Considerdepartment *A*,withanaccountbalanceof$10,000.Iftwodepartmentclerksdebittheaccount balance (by say $500 and $100, respectively) of department *A* at almost exactly the same time,the result of the concurrent executions may leave the budget in an incorrect (or inconsistent) state.Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce thatvalue by the amount being withdrawn, and write the result back. If the two programs run concurrently,they may both read the value $10,000, and write back $9500 and $9900, respectively. Depending onwhichonewritesthevaluelast,theaccountbalanceofdepartment *A*maycontaineither$9500or $9900, rather than the correct value of $9400. To guard against this possibility, the system mustmaintainsomeform of supervision.
But supervision is difficult to provide because data may be accessed by many different applicationprogramsthathave not beencoordinated previously.

Security problems. Not every user of the database system should be able to access all the data. Forexample, in a university, payroll personnel need to see only that part of the database that has financialinformation. They do not need access to information about academic records. But, since applicationprogramsareaddedtothefile-processingsysteminanadhocmanner,enforcingsuchsecurityconstraintsis difficult.

These difficulties, among others, prompted the development of database systems. In what follows, weshall see the concepts and algorithms that enable database systems to solve the problems with file-processingsystems.

AdvantagesofDBMS:

Controlling of Redundancy: Data redundancy refers to the duplication of data (i.e storing same datamultiple times). In a database system, by having a centralized database and centralized control of databy the DBA the unnecessary duplication of data is avoided. It also eliminates the extra time forprocessingthe large volumeofdata. It results insaving the storagespace.

ImprovedDataSharing:DBMSallowsausertosharethedatainanynumberofapplicationprograms.

Data Integrity : Integrity means that the data in the database is accurate. Centralized control of thedata helps in permitting the administrator to define integrity constraints to the data in the database. Forexample: in customer database we can can enforce an integrity that it must accept the customer onlyfromNoidaand Meerut city.

Security : Having complete authority over the operational data, enables the DBA in ensuring that theonly mean of access to the database is through proper channels. The DBA can define authorizationchecksto becarried outwhenever access to sensitivedata is attempted.

DataConsistency: Byeliminatingdataredundancy,wegreatlyreducetheopportunitiesforinconsistency.For example:isacustomeraddressisstoredonlyonce,wecannothavedisagreementon the stored values. Also updating data values is greatly simplified when each value is stored in oneplaceonly.Finally,weavoid thewastedstoragethat resultsfromredundantdata storage.

Efficient Data Access : In a database system, the data is managed by the DBMS and all access to thedatais through the DBMS providing a key to effectivedata processing

Enforcements of Standards : With the centralized of data, DBA can establish and enforce the datastandardswhichmay includethe namingconventions, dataquality standardsetc.

Data Independence : Ina database system, the database management system provides the interfacebetween the application programs and the data. When changes are made to the data representation, themeta data obtained by the DBMS is changed but the DBMS is continues to provide the data toapplication program in the previously used way. The DBMs handles the task of transformation of datawherevernecessary.

Reduced Application Development and Maintenance Time :DBMS supports many importantfunctions that are common to many applications, accessing data stored in the DBMS, which facilitatesthequick developmentof application.

DisadvantagesofDBMS

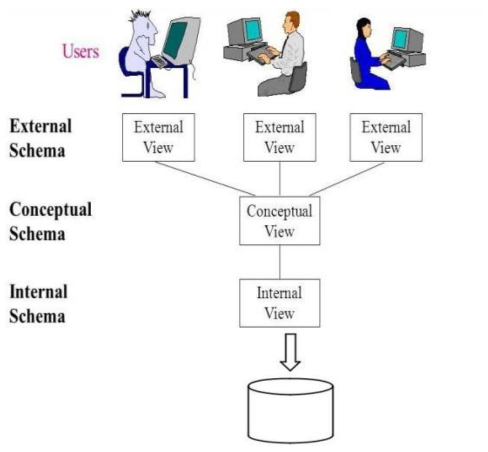
1) It is bit complex. Since it supports multiple functionality to give the user the best, the underlyingsoftware has become complex. The designers and developers should have thorough knowledgeaboutthe software to get the most out of it.
2) Becauseofitscomplexityandfunctionality,itusеslargeamountofmemory.Italsoneedslargememoryto run efficiently.
3) DBMSsystemworksonthecentralizedsystem,i.e.;alltheusersfromallovertheworldaccessthisdatabase. Henceanyfailure of the DBMS,willimpact all theusers.
4) DBMSisgeneralizedsoftware,i.e.;itiswrittenworkontheentiresystemsratherspecificone.Hencesomeo f the application willrun slow.

ViewofData

A database system is a collection of interrelated data and a set of programs that allow users to accessand modify these data. A major purpose of a database system is to provide users with an *abstract* viewofthe data.That is,the systemhides certain detailsof howthedataare storedand maintained.

DataAbstraction

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designerstousecomplexdatastructurestorepresentdatainthedatabase.Sincemanydatabase-systemusersarenotcomputertrained,developershidethecomplexityfromusersthroughseverallevelsofabstraction,to simplify users' interactionswith the system:



Database
DISK

Figure1.2: Levelsof Abstractionin aDBMS

•**Physical level (or Internal View / Schema)**: The lowest level of abstraction describes *how* the dataareactuallystored.The physicalleveldescribes complexlow-leveldata structuresindetail.

•**Logical level (or Conceptual View / Schema)**: The next-higher level of abstraction describes *what*data are stored in the database, and what relationships exist among those data. The logical level thusdescribes the entire database in terms of a small number of relatively simple structures. Althoughimplementation of the simple structures at the logical level may involve complex physical-levelstructures,theuserofthelogicalleveldoesnotneedtobeawareofthiscomplexity.Thisisreferredtoas **physicaldataindependence**.
•**View level (or External View / Schema):** The highest level of abstraction describes only part of theentire database. Even though the logical level uses simpler structures, complexity remains because ofthe variety of information stored in a large database. Many users of the database system do not needall this information; instead, they need to access only a part of the database. The view level ofabstraction exists to simplify their interaction with the system. The system may provide many viewsforthe samedatabase.
Forexample, wemaydescribe arecordas follows:

```
                    typeinstructor = record
                         ID :char (5);
                         name :char (20);
                         deptname :char (20);
                         salary:numeric(8,2);
end;
```

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

•*department*,withfields *dept_name*,*building*, and*budget*
•*course*,withfields*course_id*,*title*,*dept_name*,and *credits*
•*student*,withfields*ID*,*name*,*dept_name*,and*tot_cred*

At the physical level, an *instructor*, *department*, or *student* record can be described as a block ofconsecutivestoragelocations.

At the logical level, each such record is described by a type definition, as in the previous codesegment,and the interrelationship of theserecord types is definedaswell.

Finally,attheviewlevel,computeruserssseeasetofapplicationprogramsthathidedetailsofthedata types. At the view level, several views of the database are defined, and a database user sees someor allof these views.

InstancesandSchemas

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

Each variable has a particular value at a given instant. The values of the variables in a program at apointintimecorrespondtoan *instance*ofadatabaseschema.Databasesystemshaveseveralschemas,partitionedaccordingtothelevelsofabstraction.The **physicalschema**describesthedatabase design at the physical level, while the **logical schema** describes the database design at thelogicallevel.Adatabasemayalsohaveseveralschemasattheviewlevel,sometimescalled**subschemas**,whichdescribedifferentviewsofthedatabase.Ofthese,thelogicalschemaisbyfarthe most important, in terms of its effect on application programs, since programmers constructapplications by using the logical schema. Application programs are said to exhibit **physical dataindependence** if they do not depend on the physical schema, and thus need not be rewritten if thephysicalschema changes.

DataModels

Underlyingthestructureofadatabaseisthe**datamodel**:acollectionofconceptualtoolsfordescribingdata, datarelationships, datasemantics,andconsistencyconstraints.

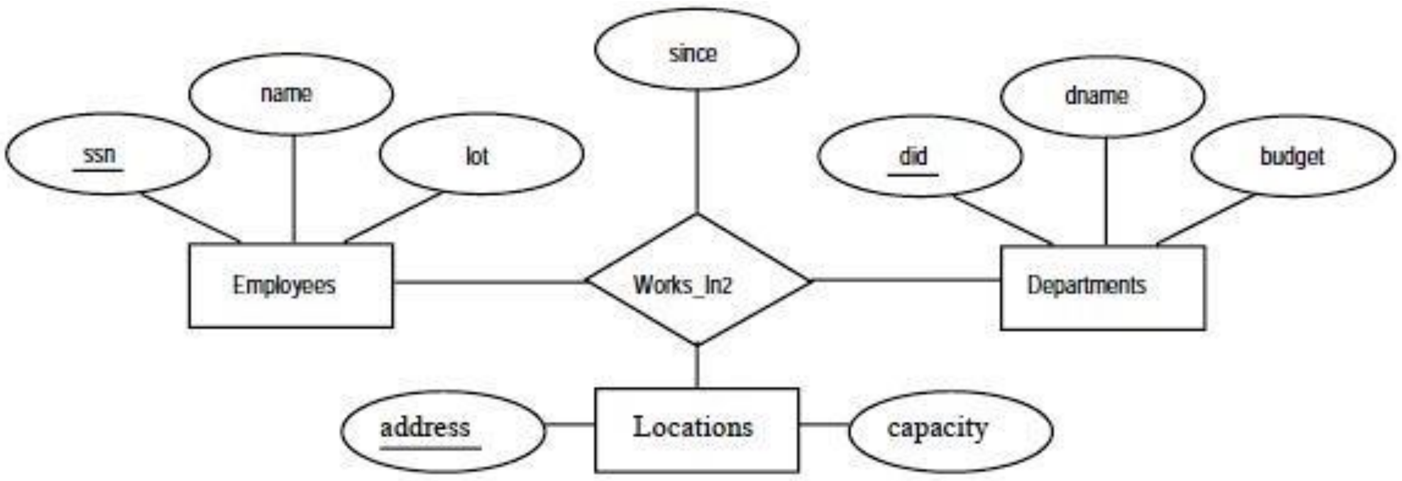
Thedata models canbe classified intofour different categories:

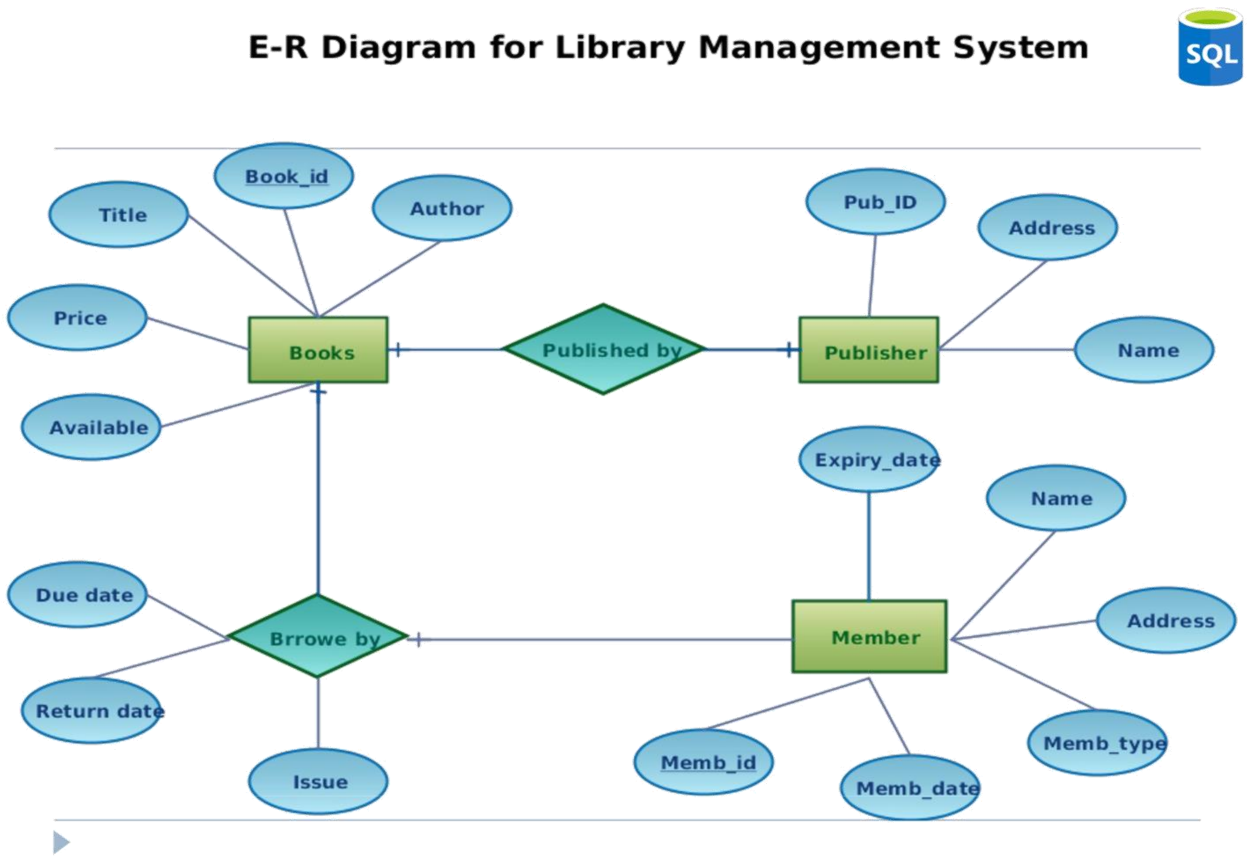
•**RelationalModel**.Therelationalmodelusesacollectionoftablestorepresentbothdataandtherelationships amongthosedata.Eachtablehasmultiplecolumns,andeachcolumnhasauniquename.Tablesarealsoknownas**relations**.Therelationalmodelisanexampleofarecord-basedmodel.

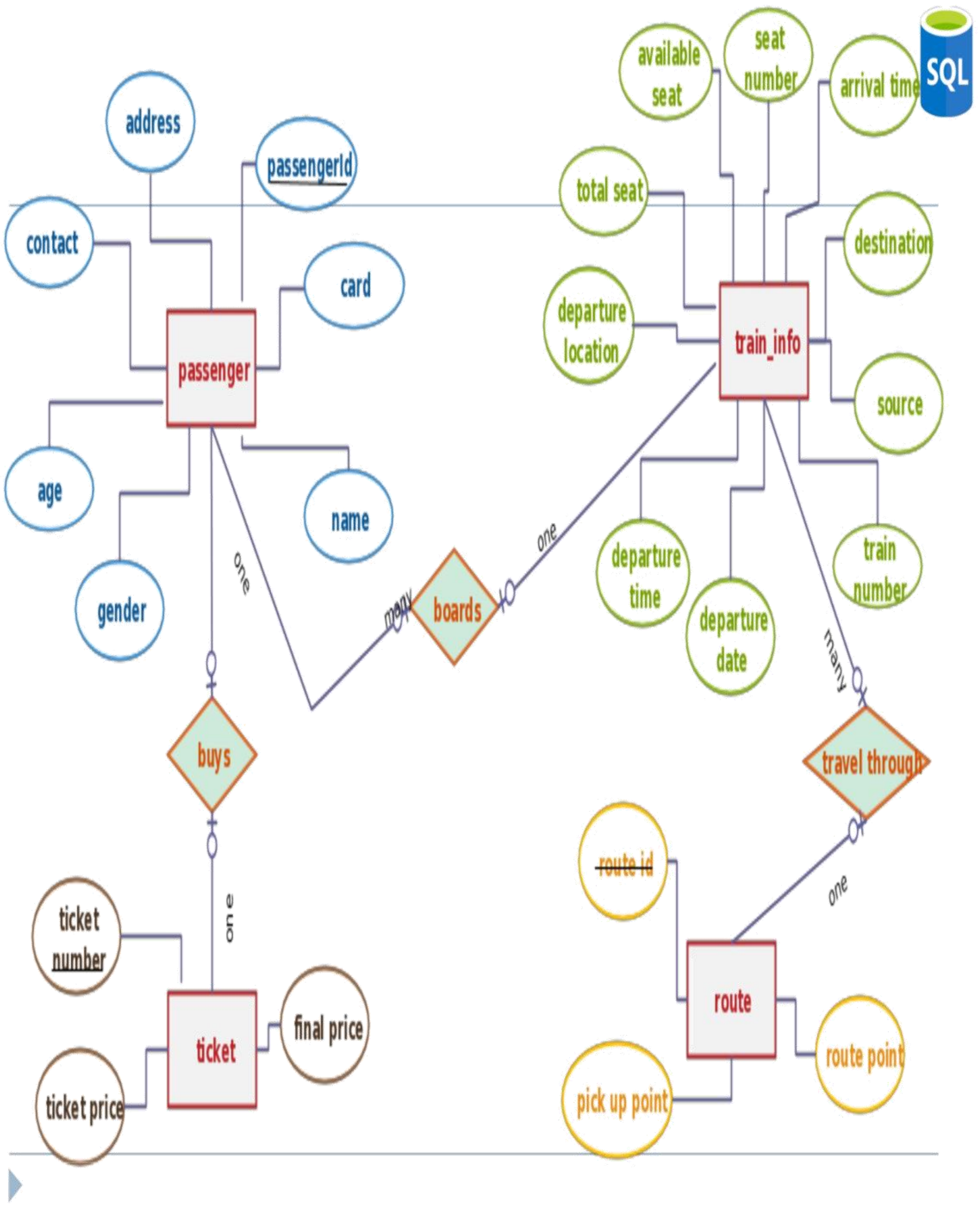
Entity-RelationshipModel.Theentity-relationship(E-R)datamodelusesacollectionofbasicobjects,called *entities*, and *relationships* among these objects.

Supposethateachdepartmenthasofficesinseverallocationsandwewanttorecordthelocationsatwhicheachemployeeworks.TheERdiagramforthisvariantofWorksIn,whichwecallWorksIn2

Example - ternary







ERModel-(RailwayBookingSystem)

ERModel-(BankingTransactionSystem)

Object-BasedDataModel.Object-orientedprogramming(especiallyinJava,C++,orC#)hasbecome the dominant software-development methodology. This led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation,methods(functions), and object identity.

Semi-structuredDataModel.Thesemi-structureddatamodelpermitsthespecificationofdatawhereindividualdataitemsofthesametypemayhavedifferentsetsofattributes.Thisisincontrasttothedatamodelsmentionedearlier,whereeverydataitemofaparticulartypemusthavethesamesetofattributes.The **ExtensibleMarkupLanguage(XML)** iswidelyusedtorepresentsemi-structureddata.

Historically,the**networkdatamodel**andthe**hierarchicaldatamodel**precededtherelationaldatamodel. Thesemodelsweretiedcloselytotheunderlyingimplementation,andcomplicatedthetaskofmodelingdata. Asaresulttheyare usedlittlenow,exceptin olddatabasecodethatis stillinserviceinsomeplaces.

DatabaseLanguages

Adatabasesystemprovidesa**data-definitionlanguage**tospecifythedatabase schema and a **data-manipulation language** to express database queries and updates. In practice,the data-definition and data-manipulation languages are not two separate languages; instead theysimplyformpartsofa singledatabaselanguage,suchasthewidelyused SQLlanguage.

Data-ManipulationLanguage

A**data-manipulationlanguage(DML)**isalanguagethatenablesuserstoaccessormanipulatedataasorganized by the appropriate data model. The types of access are:

- Retrievalofinformationstoredinthedatabase
- Insertionof newinformation intothe database
- Deletionofinformationfromthedatabase
- Modificationofinformationstoredinthedatabase

Thereare basicallytwo types:

•**ProceduralDMLs**requirea usertospecify*what*dataare neededand*how* togetthosedata.
•**DeclarativeDMLs**(alsoreferredtoas**nonproceduralDMLs**)requireausertospecify*what*dataareneeded *without* specifying how to get those data.

A **query** is a statement requesting the retrieval of information. The portion of a DML that involvesinformationretrievaliscalleda**querylanguage**.Althoughtechnicallyincorrect,itiscommonpracticeto use the terms *querylanguage* and*data-manipulation language* synonymously.

Data-DefinitionLanguage(DDL)

Wespecifyadatabaseschemabyasetofdefinitionsexpressedbyaspeciallanguagecalleda**data-definitionlanguage**(**DDL**).The DDLisalsoused tospecifyadditionalpropertiesofthedata.

•**Domain Constraints**. A domain of possible values must be associated with every attribute (forexample, integer types, character types, date/time types). Declaring an attribute to be of a particulardomain acts as a constraint on the values that it can take. Domain constraints are the most elementaryform of integrity constraint. They are tested easily by the system whenever a new data item is enteredinto thedatabase.

•**Referential Integrity**. There are cases where we wish to ensure that a value that appears in onerelation for a given set of attributes also appears in a certain set of attributes in another relation(referential integrity). For example, the department listed for each course must be one that actuallyexists. More precisely, the *dept name* value in a *course* record must appear in the *dept name* attributeofsomerecord of the *department* relation.
•**Assertions**.Anassertionisanyconditionthatthedatabasemustalwayssatisfy.Domainconstraintsandreferential-integrityconstraintsarespecialformsofassertions.However,therearemanyconstraintsthatwecannotexpressbyusingonlythesespecialforms.Forexample,"Everydepartmentmusthaveatleastfivecoursesofferedeverysemester"mustbeexpressedasanassertion..

•**Authorization**. We may want to differentiate among the users as far as the type of access they arepermitted on various data values in the database. These differentiations are expressed in terms of**authorization**,themostcommonbeing: **readauthorization**,whichallowsreading,butnotmodification, of data; **insert authorization**, which allows insertion of new data, but not modificationof existing data; **update authorization**, which allows modification, but not deletion, of data; and**deleteauthorization**,whichallowsdeletionofdata.Wemayassigntheuserall,none,oracombinationof these types of authorization.

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

DataDictionary

Wecandefineadatadictionaryasa DBMScomponentthatstoresthedefinitionofdatacharacteristics and relationships. You may recall that such "data about data" were labeled metadata.The DBMS data dictionary provides the DBMS with its self describing characteristic. In effect, thedata dictionary resembles and X-ray of the company's entire data set, and is a crucial element in thedataadministrationfunction.
Forexample,thedatadictionarytypicallystoresdescriptionsofall:

- Data elements that are define in all tables of all databases. Specifically the data dictionary storesthe name, datatypes, display formats, internal storage formats, and validation rules. The datadictionarytells wherean elementis used, by whomitis used and soon.
- Tables define in all databases. For example, the data dictionary is likely to store the name of thetablecreator, thedate of creationaccess authorizations, the number of columns,and soon.
- Indexes define for each database tables. For each index the DBMS stores at least the index nametheattributes used, thelocation, specific indexcharacteristics and thecreation date.
- Define databases: who created each database, the date of creation where the database is located, whothe DBAisand soon.
- Endusersand TheAdministrators ofthe database
- Programsthataccessthedatabaseincludingscreenformats,reportformatsapplicationformats,SQLqueries and soon.
- Accessauthorizationforallusersofalldatabases.
- Relationshipsamongdataelementswhichelementsareinvolved:whethertherelationshiparemandatory or optional, the connectivity and cardinality and soon.

DatabaseAdministratorsandDatabaseUsers
Aprimarygoalofadatabasesystemistoretrieveinformationfromandstorenewinformationinthedatabase.
DatabaseUsersandUserInterfaces

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

Naiveusers areunsophisticateduserswhointeractwiththesystembyinvokingoneoftheapplication programs that have been written previously. For example, a bank teller who needs totransfer$50 from account *A* to account *B* invokes aprogram called *transfer*.

Applicationprogrammersarecomputerprofessionalswhowriteapplicationprograms.Applicationprogra mmerscanchoosefrommanytoolstodevelopuserinterfaces.

Rapidapplicationdevelopment(RAD)toolsaretoolsthatenableanapplicationprogrammertoconstructfor msandreportswithoutwriting a program.

Sophisticated users interact with the system without writing programs. Instead, they form theirrequests in a database query language. They submit each such query to a **query processor**, whosefunction is to break down DML statements into instructions that the storage manager understands.Analystswhosubmitqueriesto exploredatain thedatabase fall inthis category.

Online analytical processing (OLAP) tools simplify analysts' tasks by letting them view summariesof data in different ways. For instance, an analyst can see total sales by region (for example,
North,South,East,andWest),orbyproduct,orbyacombinationofregionandproduct(thatis,totalsalesofeach product in each region).

DatabaseArchitecture:

The architecture of a database system is greatly influenced by the underlying computer system onwhich the database system runs. Database systems can be centralized, or client-server, where oneserver machine executes work on behalf of multiple client machines. Database systems can also bedesignedtoexploitparallelcomputerarchitectures.Distributeddatabasesspanmultiplegeographicallyse parated machines.

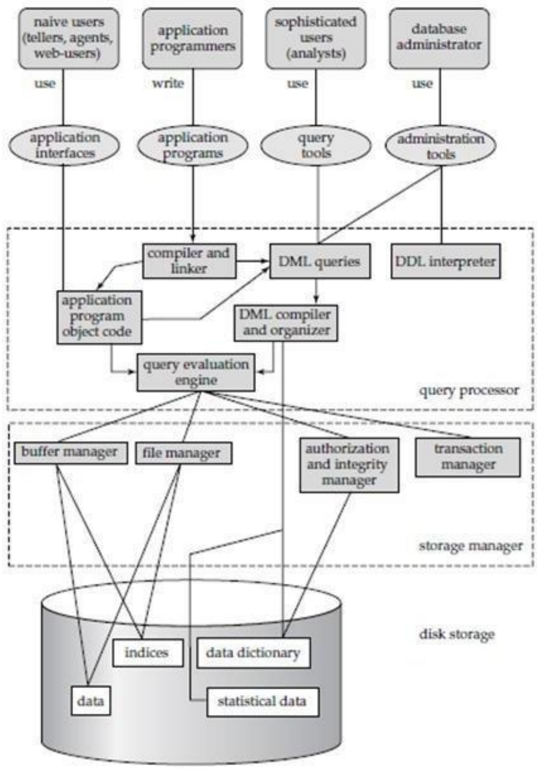


Figure1.3:DatabaseSystemArchitecture

A database system is partitioned into modules that deal with each of the responsibilities of the overallsystem. The functional components of a database system can be broadly divided into the **storagemanager**andthe**queryprocessor**components.Thestoragemanagerisimportantbecausedatabases typically require a large amount of storage space. The query processor is important because it helpsthedatabase systemsimplify and facilitateaccess to data.

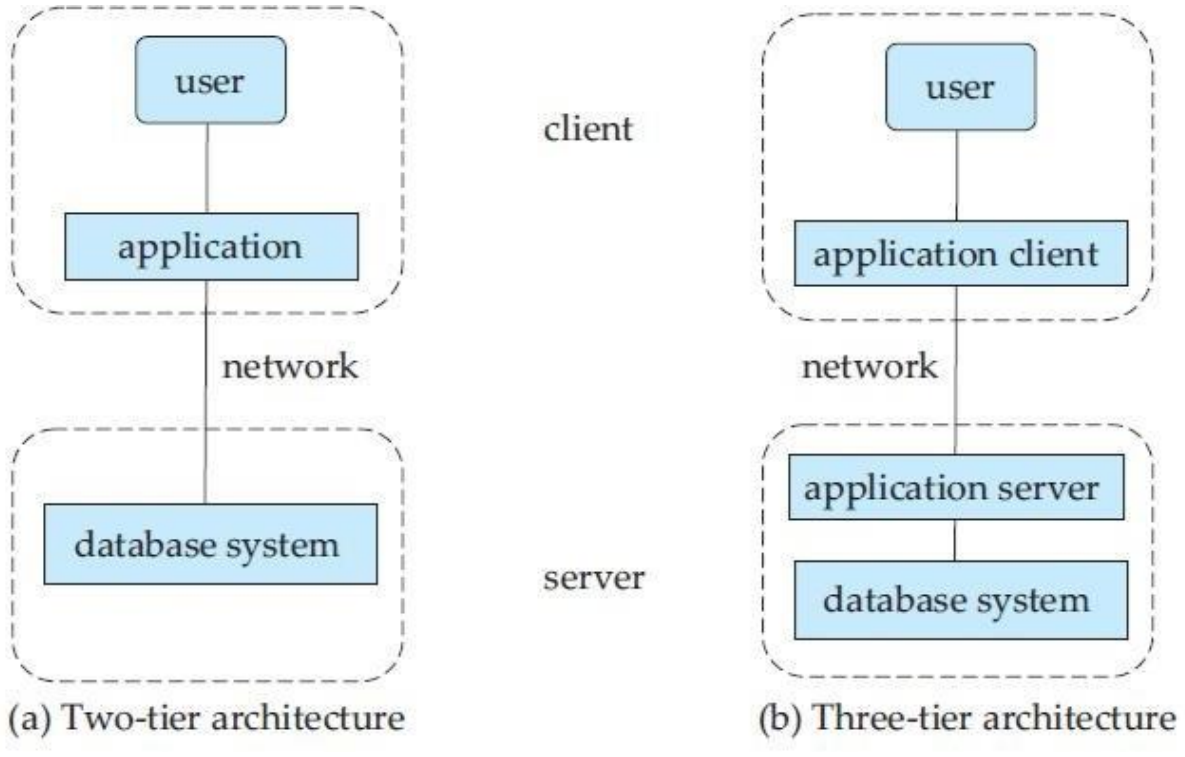


Figure1.4:Two-tierandthree-tierarchitectures.

QueryProcessor:

Thequery processorcomponents include

·**DDLinterpreter,** whichinterpretsDDLstatementsandrecordsthedefinitionsinthedatadictionary.

·**DML compiler,** which translates DML statements in a query language into an evaluation planconsistingof low-level instructions that the queryevaluationengine understands.

A query can usually be translated into any of a number of alternative evaluation plans that all give thesame result. The DML compiler also performs **query optimization**, that is, it picks the lowest costevaluationplan from among the alternatives.

Queryevaluationengine,whichexecuteslow-levelinstructionsgeneratedbytheDMLcompiler.

StorageManager:

A *storage manager* is a program module that provides the interface between the lowlevel data stored inthe databaseandtheapplicationprogramsandqueriessubmittedtothesystem.Thestoragemanageris responsible for the interaction with the file manager.

Thestoragemanagercomponentsinclude:

·**Authorizationandintegritymanager**,whichtestsforthesatisfactionofintegrityconstraintsandchecks the authority ofusers to access data.
·**Transactionmanager**,whichensuresthatthedatabaseremainsinaconsistent(correct)statedespitesystemfailures,andthatconcurrenttransactionexecutionsproceedwithoutconflicting.
·**Filemanager**,whichmanagestheallocationofspaceondiskstorageandthedatastructuresusedto representinformation stored on disk.
·**Buffermanager**,whichisresponsibleforfetchingdatafromdiskstorageintomainmemory,anddeciding whatdatatocacheinmainmemory.Thebuffermanagerisacriticalpartofthe databasesystem,sinceitenablesthedatabasetohandledatasizesthataremuchlargerthanthesizeof mainmemory.

Transaction Manager:

A **transaction** is a collection of operations that performs a single logical function in a databaseapplication. Each transaction is a unit of both atomicity and consistency. Thus, we require thattransactionsdo not violateany database-consistency constraints.

ConceptualDatabaseDesign-EntityRelationship(ER)Modeling:

DatabaseDesignTechniques

1. ERModeling(TopdownApproach)
2. Normalization(BottomUpapproach)

What is ER Modeling?

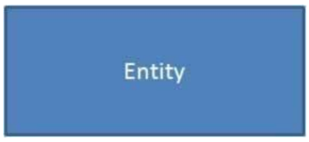
Agraphicaltechniqueforunderstandingandorganizingthedataindependentoftheactualdatabaseimplementation

Weneedto befamiliar withthe followingtermsto gofurther.

Entity

Anythingthathasanindependentexistenceandaboutwhichwecollectdata.Itisalsoknownasentitytype.

InER modeling, notationfor entity isgiven below.



Entity instance

Entity instance is a particular member of the entity

type.Example for entity instance : A particular

employee**RegularEntity**

An entity which has its own key attribute is a regular

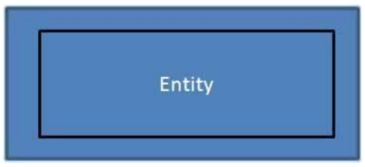
entity.Examplefor regular entity : Employee.

Weak entity

Anentitywhichdependsonotherentityforitsexistenceanddoesn'thaveanykeyattributeofitsownisaweakentity.

Exampleforaweakentity:Inaparent/childrelationship,aparentisconsideredasastrongentityandthe child is a weakentity.

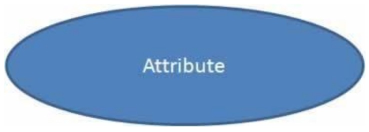
InERmodeling, notationfor weakentityis givenbelow.



Attributes

Properties/characteristics which describe entities are called

attributes.InER modeling, notation for attribute isgiven below.



DomainofAttributes

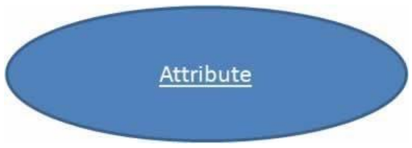
The set of possible values that an attribute can take is called the domain of the attribute. For example,theattributedaymaytakeanyvaluefromtheset{Monday,Tuesday...Friday}.Hencethissetcanbetermed as the domain of the attribute day.

Keyattribute

Theattribute(orcombinationofattributes)whichisuniqueforeveryentityinstanceiscalledkeyattribute.

E.gtheemployee_idofanemployee,pan_card_numberofapersonetc.Ifthekeyattributeconsistsof two or more attributes in combination, it is calleda composite key.

InER modeling, notation for keyattributeis given below.



Simpleattribute

If an attribute cannot be divided into simpler components, it is a simple

attribute.Examplefor simpleattribute :employee_id of an employee.

Compositeattribute

Ifanattribute canbesplitintocomponents,it iscalled acompositeattribute.

Exampleforcompositeattribute:NameoftheemployeewhichcanbesplitintoFirst_name,Middle_name,and Last_name.

SinglevaluedAttributes

If an attribute can take only a single value for each entity instance, it is a single valued

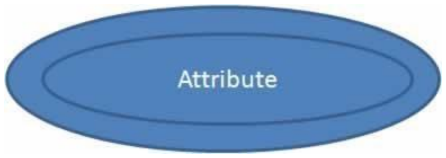
attribute.exampleforsinglevaluedattribute: ageofastudent.Itcan takeonlyonevaluefora

particularstudent.**Multi-valuedAttributes**

Ifanattributecantakemorethanonevalueforeachentityinstance,itisamulti-valuedattribute.Multi-valued

exampleformultivaluedattribute:telephonenumberofanemployee,aparticularemployeemayhavemultip le telephone numbers.

InER modeling, notationfor multi-valued attributeis given below.



StoredAttribute

AnattributewhichneedtobestoredpermanentlyisastoredattributeExamp

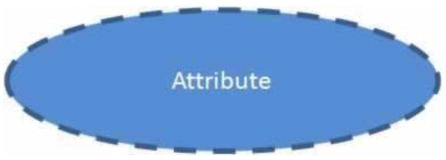
lefor storedattribute: nameofa student

DerivedAttribute

Anattributewhichcanbecalculated orderived basedonother attributesis aderivedattribute.

Examplefor derivedattribute:ageofemployeewhichcanbecalculatedfromdateofbirthandcurrentdate.

InER modeling,notation for derivedattribute isgiven below.

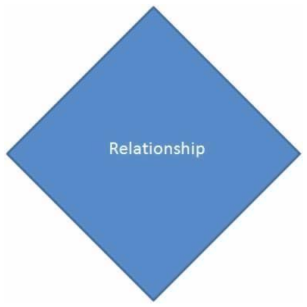


Relationships

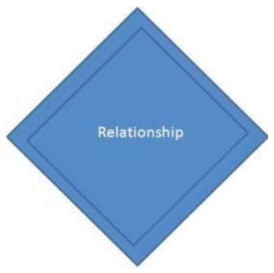
Associationsbetweenentitiesarecalledrelationships

Example:Anemployeeworksforanorganization.Here"worksfor"isarelationbetweentheentitiesemployeeand organization.

InER modeling,notation forrelationship is givenbelow.



HoweverinERModeling,ToconnectaweakEntitywithothers,youshoulduseaweakrelationshipnotation asgiven below



DegreeofaRelationship

Degree of a relationship is the number of entity types involved. The n-ary relationship is thegeneral form for degree n. Special cases are unary, binary, and ternary ,where the degree is 1, 2,and 3,respectively.

Example for unary relationship : An employee ia a manager of

anotheremployeeExampleforbinaryrelationship:Anemployeeworks-

fordepartment. Example for ternary relationship : customer purchase

itemfroma shopkeeper **Cardinalityof aRelationship**

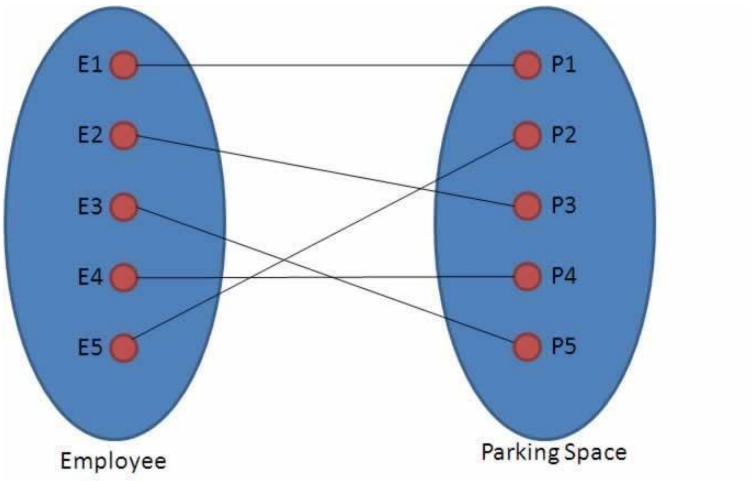
Relationshipcardinalitiesspecifyhowmanyofeachentitytypeisallowed.Relationshipscanhavefourpossible connectivities as given below.

1. Onetoone(1:1)relationship
2. Onetomany(1:N)relationship
3. Manytoone(M:1)relationship
4. Manytomany(M:N)relationship

Theminimum andmaximum valuesof thisconnectivity iscalled thecardinality of therelationship

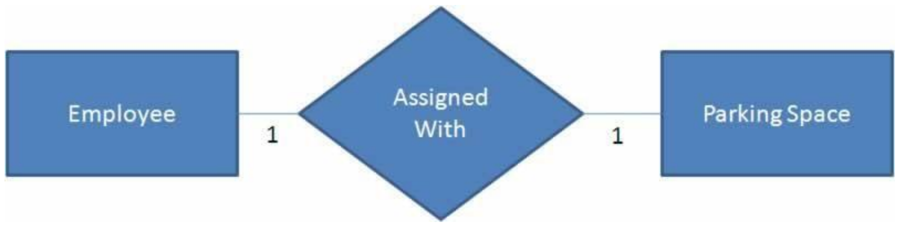
ExampleforCardinality–One-to-One(1:1)

Employeeisassignedwithaparkingspace.



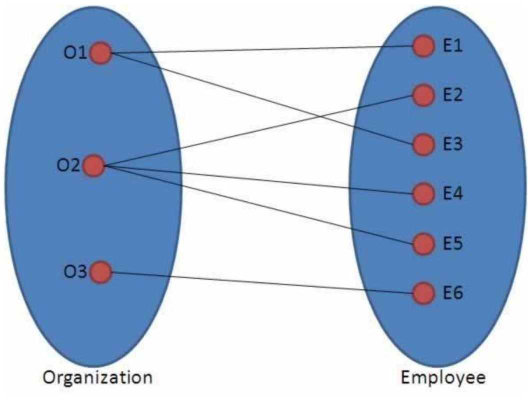
Oneemployeeisassignedwithonlyoneparkingspaceandoneparkingspaceisassignedtoonlyoneemployee.Henceitisa 1:1relationshipand cardinalityisOne-To-One(1:1)

InER modeling, this can bementionedusing notations as given below



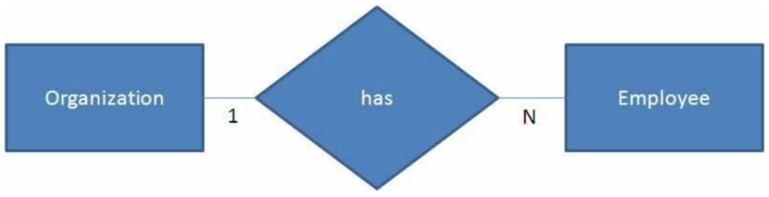
ExampleforCardinality–One-to-Many(1:N)

Organizationhasemployees



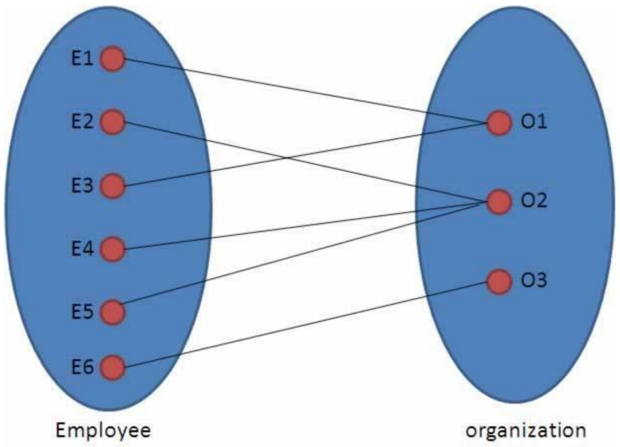
Oneorganizationcanhavemanyemployees,butoneemployeeworksinonlyoneorganization.Henceitis a1:N relationship and cardinalityis One-To-Many(1:N)

InER modeling, this can bementionedusing notations as given below



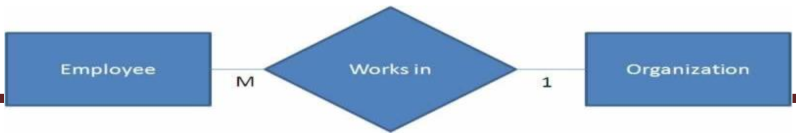
ExampleforCardinality–Many-to-One(M:1)

ItisthereverseoftheOnetoManyrelationship.employee worksinorganization



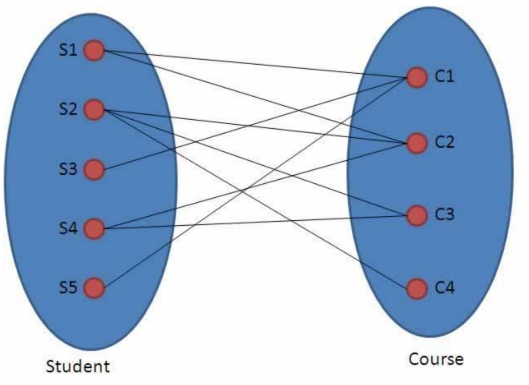
Oneemployeeworksinonlyoneorganization Butoneorganizationcanhavemanyemployees.Henceitis aM:1relationship andcardinality is Many-to-One(M :1)

InER modeling, this can bementionedusing notations as given below.



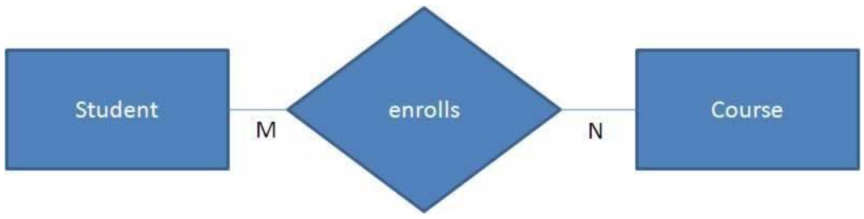
Cardinality–Many-to-Many(M:N)

Studentsenrollsforcourses



Onestudentcanenrollformanycoursesandonecoursecanbeenrolledbymanystudents.Henceitis a M:N relationshipand cardinality is Many-to-Many (M:N)

InER modeling, this can bementionedusing notations as given below



RelationshipParticipation

1. Total

Intotalparticipation,everyentityinstancewillbeconnectedthroughtherelationshiptoanotherinstanceof the other participating entity types

2. Partial

Exampleforrelationshipparticipation

Considerthe relationship -Employee is headof the department.

Here all employees will not be the head of the department. Only one employee will be the headofthedepartment.Inotherwords,onlyfewinstancesofemployeeentityparticipateintheaboverelationship. Soemployee entity'sparticipation is partialin thesaidrelationship.

Advantages and Disadvantages of ER Modeling (Merits and Demerits of ER Modeling)Advantages

1. ER Modeling is simple and easily understandable. It is represented in business users language anditcan be understood by non-technical specialist.

2. IntuitiveandhelpsinPhysicalDatabasecreation.

3. Canbegeneralizedandspecializedbasedonneeds.

4. Canhelp indatabasedesign.

5. Givesahigherleveldescriptionofthesystem.

Disadvantages

1. PhysicaldesignderivedfromE-RModelmayhavesomeamountofambiguitiesorinconsistency.

2. Sometimediagramsmayleadtomisinterpretations

RelationalModel

The relational model is today the primary data model for commercial data processing applications. Itattaineditsprimarypositionbecauseofitssimplicity,whicheasesthejoboftheprogrammer,comparedto earlier data models suchas the network modelor the hierarchical model.

StructureofRelationalDatabases:

Arelationaldatabaseconsistsofacollectionof **tables**,eachofwhichisassignedauniquename.Forexample,considerthe*instructor*tableofFigure:1.5,whichstoresinformationaboutinstructors.Thetablehasfourcolumnheaders:*ID*, *name*, *deptname*,and*salary*.Eachrowofthistablerecordsinformationabout aninstructor, consisting ofthe instructor's*ID*, *name*,*dept name*, and*salary*.

DatabaseSchema

Whenwetalkaboutadatabase,wemustdifferentiatebetweenthe**databaseschema**,whichisthelogicaldesign ofthedatabase,andthe**databaseinstance**,whichisasnapshotofthedatainthedatabaseatagiveninstantintime .Theconceptofarelationcorrespondstotheprogramming-languagenotionofavariable,whiletheconceptofa**relationschema**correspondstotheprogramming-languagenotion of type definition.

Keys

A**superkey**isasetofoneormoreattributesthat,takencollectively,allowustoidentifyuniquelyatupleintherelation.Forexample,the*ID*attributeoftherelation *instructor*issufficienttodistinguishoneinstructortuplefromanother.Thus, *ID*isasuperkey.The*name*attributeof*instructor*,ontheotherhand,isnot a superkey,because severalinstructors might havethe samename.

A superkey may contain extraneous attributes. For example, the combination of *ID* and *name* is asuperkey for the relation *instructor*. If *K* is a superkey, then so is any superset of *K*. We are ofteninterested in superkeys for which no proper subset is a superkey. Such minimal superkeys are called**candidatekeys**.

It is customary to list the primary key attributes of a relation schema before the other attributes; forexample, the *dept name* attribute of *department* is listed first, since it is the primary key. Primary keyattributes are also underlined. A relation, say *r*1, may include among its attributes the primary key ofanotherrelation, say*r*2.This attribute iscalled a **foreign key**from *r*1, referencing*r*2.

SchemaDiagrams

Adatabaseschema,alongwithprimarykeyandforeignkeydependencies,canbedepictedby **schemadiagrams**.Figure1.12showstheschemadiagramforouruniversityorganization.

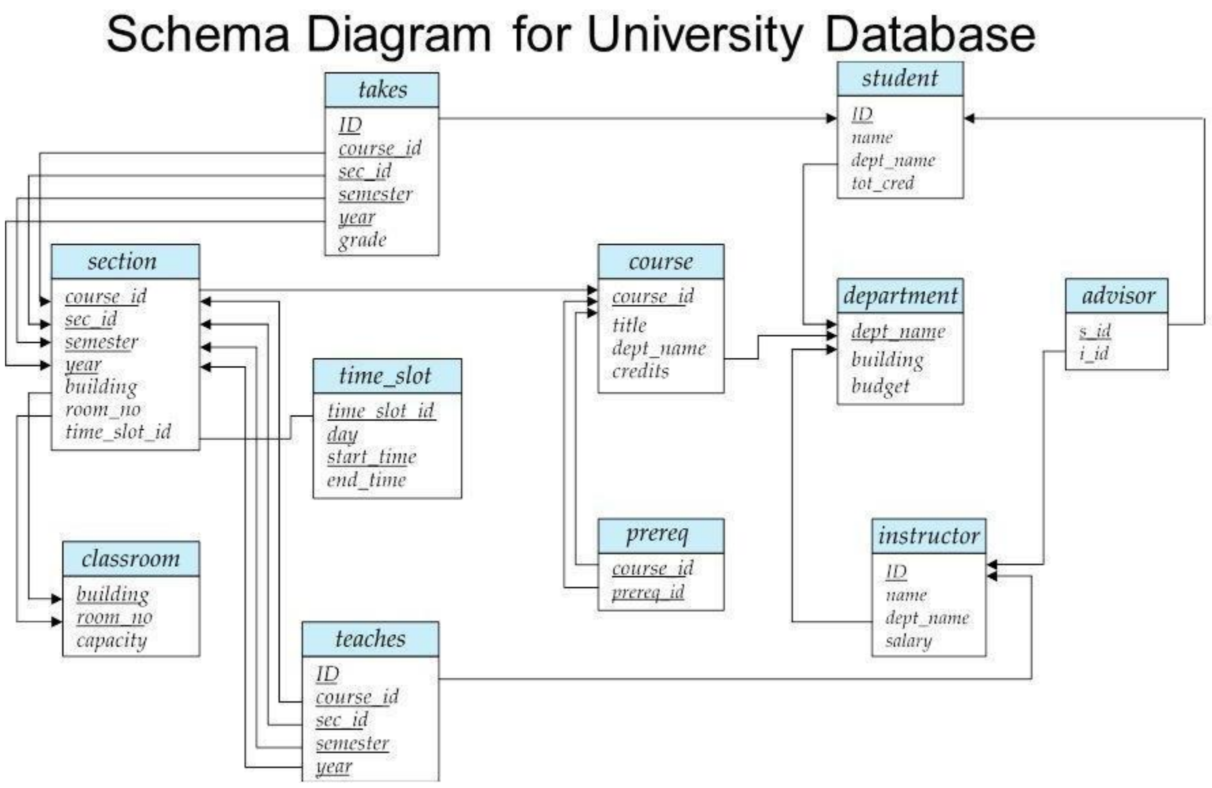


Figure1.12:Schemadiagramfortheuniversitydatabase.

Referentialintegrityconstraintsotherthanforeignkeyconstraintsarenotshownexplicitlyinschemadiagrams.We willstudyadifferentdiagrammaticrepresentationcalledtheentity-relationshipdiagram.

Relational Data Model in DBMS | Database Concepts
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 way the data are logically organized.

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 in DBMS

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Stu_Rollno, NAME,etc.

Stu_Rollno	Name

2. **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.

Stu_Rollno	Name
101	Jay
102	sri

3. **Tuple** – It is nothing but a single row of a table, which contains a single record.

Stu_Rollno	Name
101	Jay
102	sri

Number of Tuple:2

4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.

One-one

One-Many

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

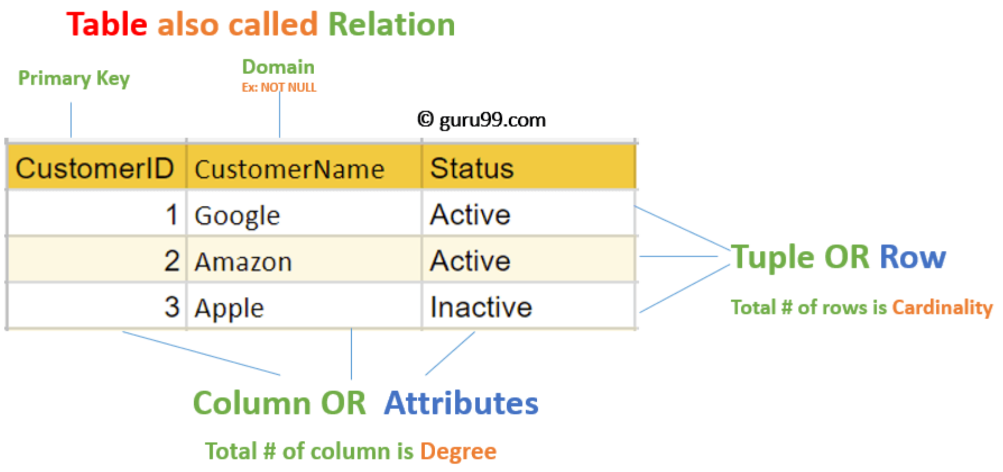


Table also called Relation
Primary Key
Domain
Ex: NOT NULL
© guru99.com
Tuple OR Row
Total # of rows is Cardinality
Column OR Attributes
Total # of column is Degree

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

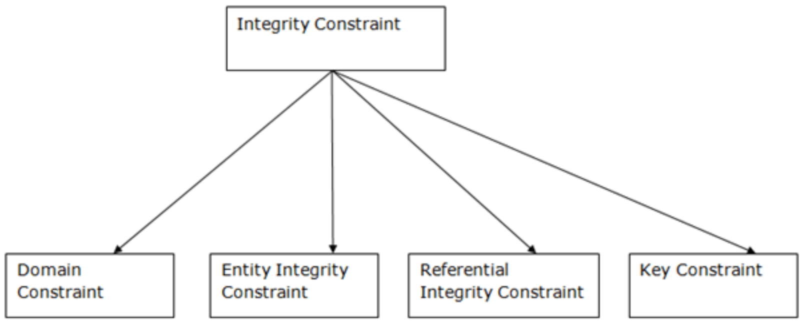
Define degree and cardinality. Also, Based upon given table write degree and cardinality.

Rollno	Name	Dept	phoneno
1.	Sri	Cse	8344289963
2.	Amutha	MECH	8344688331
3.	Jeni	EEE	8344289962
4.	jenila	ECE	8344285145
5.	kumar	BT	8458125458

Answer:
Degree is the number of attributes or columns present in a table.
Cardinality is the number of tuples or rows present in a table.
Patients Degree = 4
Cardinality = 5

Relational Integrity Constraints

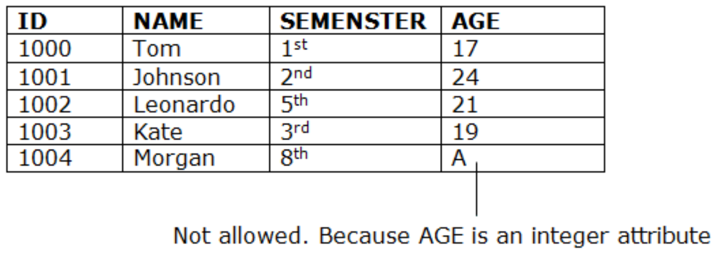
- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.



1. Domain constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example:



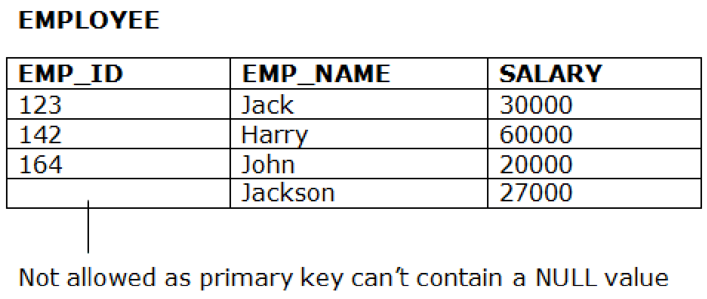
ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:



EMPLOYEE

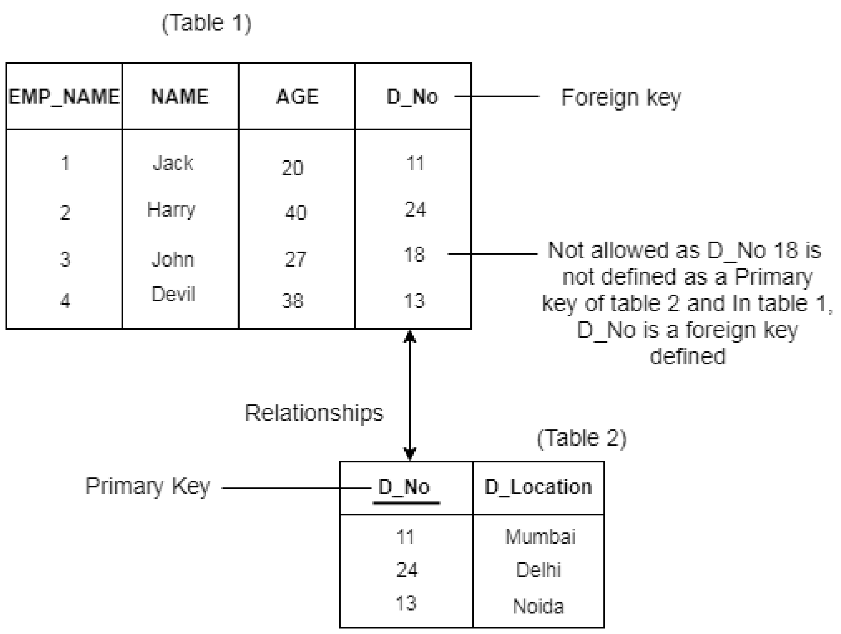
EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:


(Table 1)
Foreign key
Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined
Relationships
(Table 2)
Primary Key

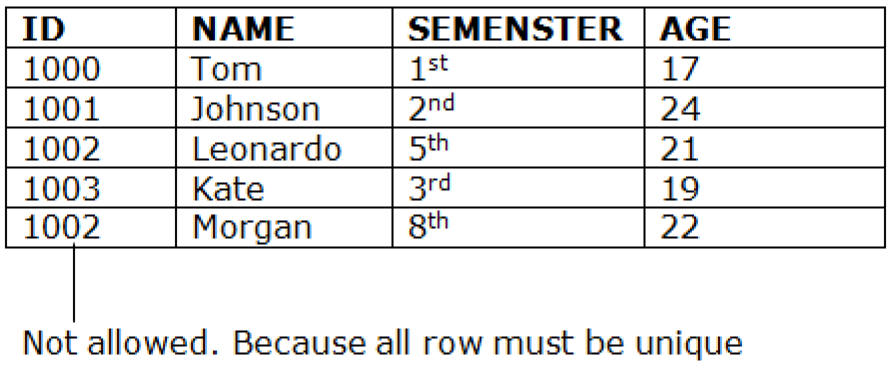
EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example:



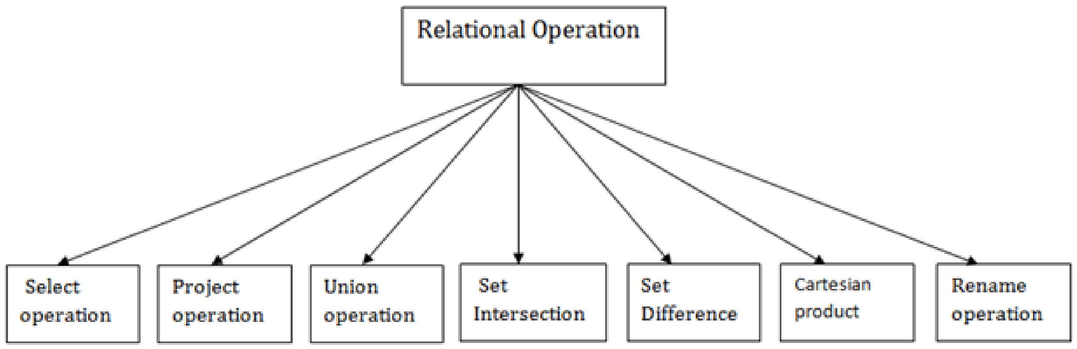
ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

Relational Algebra

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

Types of Relational operation



1. Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).

1. Notation: σ p(r) **Where:**

σ is used for selection prediction
r is used for relation
p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

For example: LOAN Relation

BRANCH_NAME	**LOAN_NO**	**AMOUNT**
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

1. σ BRANCH_NAME="perryride" (LOAN)

Output:

BRANCH_NAME	**LOAN_NO**	**AMOUNT**
Perryride	L-15	1500
Perryride	L-16	1300

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.
- It is denoted by ∏.

1. Notation: ∏ A1, A2, An (r)

Where

A1, **A2**, **A3** is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

NAME	**STREET**	**CITY**
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

1. ∏ NAME, CITY (CUSTOMER)

Output:

NAME	**CITY**
Jones	Harrison
Smith	Rye
Hays	Harrison

Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Union Operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by ∪.

1. Notation: R ∪ S

A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

Example:

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

Input:

1. ∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.

1. Notation: R ∩ S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. ∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.

- It is denoted by intersection minus (-).

1. Notation: R - S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. ∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.

1. Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

1. EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

7. Rename Operation:

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

1. ρ(STUDENT1, STUDENT)

EXTENDED E-R FEATURES

EER is a high-level data model that incorporates the extensions to the original ER model. Enhanced ERD are high level models that represent the requirements and complexities of complex database.

In addition to ER model concepts EE-R includes −

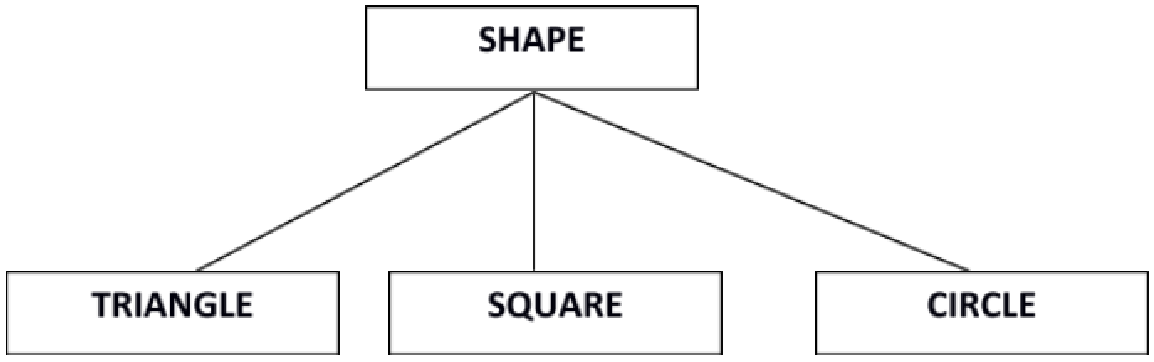
- Subclasses and Super classes.
- Specialization and Generalization.
- Category or union type.
- Aggregation.

These concepts are used to create EE-R diagrams.

Subclasses and Super class

Super class is an entity that can be divided into further subtype.

For **example** − consider Shape super class.

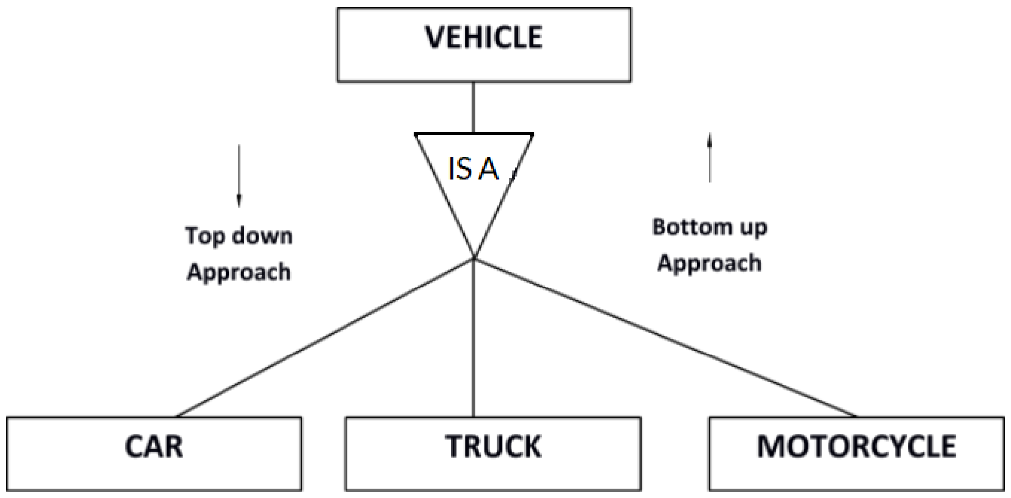


Super class shape has sub groups: Triangle, Square and Circle.

Sub classes are the group of entities with some unique attributes.Sub class inherits the properties and attributes from super class.

Specialization and Generalization

Generalization is a process of generalizing an entity which contains generalized attributes or properties of generalized entities.



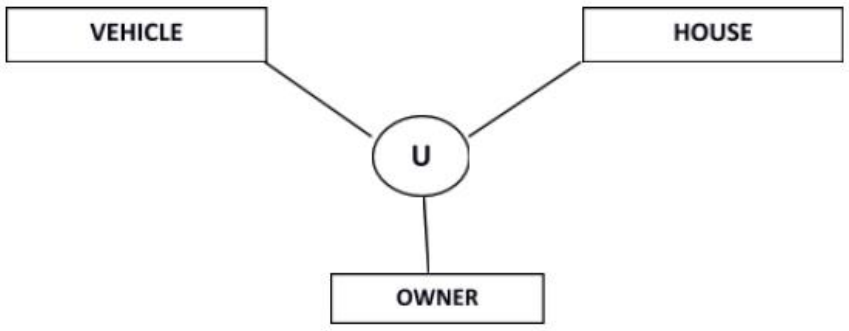
It is a Bottom up process i.e. consider we have 3 sub entities Car, Truck and Motorcycle. Now these three entities can be generalized into one super class named as Vehicle.

Specialization is a process of identifying subsets of an entity that share some different characteristic. It is a top down approach in which one entity is broken down into low level entity.

In above example Vehicle entity can be a Car, Truck or Motorcycle.

Category or Union

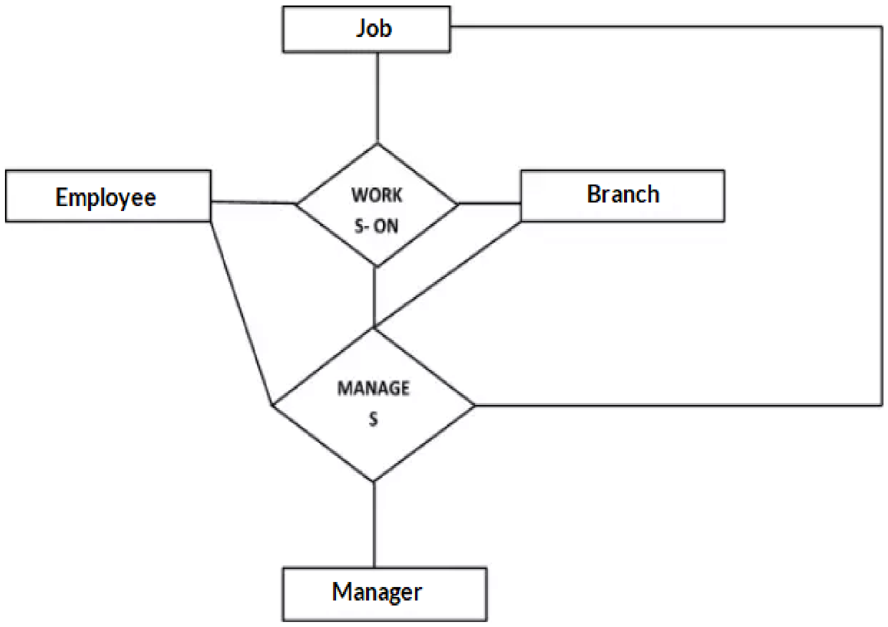
Relationship of one super or sub class with more than one super class.



Owner is the subset of two super class: Vehicle and House.

Aggregation

Represents relationship between a whole object and its component.



Consider a ternary relationship Works_On between Employee, Branch and Manager. Now the best way to model this situation is to use aggregation, So, the relationship-set, Works_On is a higher level entity-set. Such an entity-set is treated in the same manner as any other entity-set. We can create a binary relationship, Manager, between Works_On and Manager to represent who manages what tasks.

UNIT-II

Structured Query Language (SQL)

Structured Query Language is a standard Database language which is used to create, maintain and retrieve the relational database. Following are some interesting facts about SQL.

- SQL is case insensitive. But it is a recommended practice to use keywords (like SELECT, UPDATE, CREATE, etc) in capital letters and use user defined things (liked table name, column name, etc) in small letters.
- We can write comments in SQL using "–" (double hyphen) at the beginning of any line.
- SQL is the programming language for relational databases (explained below) like MySQL, Oracle, Sybase, SQL Server, Postgre, etc. Other non-relational databases (also called NoSQL) databases like MongoDB, DynamoDB, etc do not use SQL
- Although there is an ISO standard for SQL, most of the implementations slightly vary in syntax. So we may encounter queries that work in SQL Server but do not work in MySQL.
.

What is Relational Database?

Relational database means the data is stored as well as retrieved in the form of relations (tables). Table 1 shows the relational database with only one relation called **STUDENT** which stores **ROLL_NO**, **NAME**, **ADDRESS**, **PHONE** and **AGE** of students.

STUDENT

ROLL_NO	**NAME**	**ADDRESS**	**PHONE**	**AGE**
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156768971	18

TABLE 1

These are some important terminologies that are used in terms of relation.

Attribute: Attributes are the properties that define a relation. e.g.; **ROLL_NO**, **NAME** etc.

Tuple: Each row in the relation is known as tuple. The above relation contains 4 tuples, one of which is shown as:

1	RAM	DELHI	9455123451	18

Degree: The number of attributes in the relation is known as degree of the relation.
The **STUDENT** relation defined above has degree 5.

Cardinality: The number of tuples in a relation is known as cardinality. The **STUDENT** relation defined above has cardinality 4.
Column: Column represents the set of values for a particular attribute. The column **ROLL_NO** is extracted from relation STUDENT.

ROLL_NO
1
2
3
4

The queries to deal with relational database can be categories as:

Data Definition Language: It is used to define the structure of the database. e.g; CREATE TABLE, ADD COLUMN, DROP COLUMN and so on.
Data Manipulation Language: It is used to manipulate data in the relations. e.g.; INSERT, DELETE, UPDATE and so on.
Data Query Language: It is used to extract the data from the relations. e.g.; SELECT
So first we will consider the Data Query Language. A generic query to retrieve from a relational database is:

1. **SELECT** [**DISTINCT**] Attribute_List **FROM** R1,R2….RM
2. [**WHERE** condition]
3. [**GROUP BY** (Attributes)[**HAVING** condition]]
4. [**ORDER BY**(Attributes)[**DESC**]];

SQL Server Clauses

DISTINCT Clause	Retrieve unique records
FROM Clause	List tables and join information
WHERE Clause	Filter results
ORDER BY Clause	Sort query results
GROUP BY Clause	Group by one or more columns

HAVING Clause	Restrict the groups of returned rows

5.
Part of the query represented by statement 1 is compulsory if you want to retrieve from a relational database. The statements written inside [] are optional. We will look at the possible query combination on relation shown in Table 1.

Case 1: If we want to retrieve attributes **ROLL_NO** and **NAME** of all students, the query will be:
SELECT ROLL_NO, NAME **FROM** STUDENT;

ROLL_NO	**NAME**
1	RAM
2	RAMESH
3	SUJIT
4	SURESH

Case 2: If we want to retrieve **ROLL_NO** and **NAME** of the students whose **ROLL_NO** is greater than 2, the query will be:
SELECT ROLL_NO, NAME **FROM** STUDENT
WHERE ROLL_NO>2;

ROLL_NO	**NAME**
3	SUJIT
4	SURESH

CASE 3: If we want to retrieve all attributes of students, we can write * in place of writing all attributes as:
SELECT * FROM STUDENT
WHERE ROLL_NO>2;

ROLL_NO	**NAME**	**ADDRESS**	**PHONE**	**AGE**
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156768971	18

CASE 4: If we want to represent the relation in ascending order by **AGE**, we can use ORDER BY clause as:
SELECT * FROM STUDENT **ORDER BY** AGE;

ROLL_NO	**NAME**	**ADDRESS**	**PHONE**	**AGE**
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
4	SURESH	DELHI	9156768971	18
3	SUJIT	ROHTAK	9156253131	20

Note: ORDER BY **AGE** is equivalent to ORDER BY **AGE** ASC. If we want to retrieve the results in descending order of **AGE**, we can use ORDER BY **AGE** DESC.
CASE 5: If we want to retrieve distinct values of an attribute or group of attribute, DISTINCT is used as in:
SELECT DISTINCT ADDRESS **FROM** STUDENT;

ADDRESS
DELHI
GURGAON
ROHTAK

If DISTINCT is not used, DELHI will be repeated twice in result set. Before understanding GROUP BY and HAVING, we need to understand aggregations functions in SQL.

AGGRATION FUNCTIONS: Aggregation functions are used to perform mathematical operations on data values of a relation. Some of the common aggregation functions used in SQL are:

- **COUNT:** Count function is used to count the number of rows in a relation. e.g;

SELECT COUNT (PHONE) **FROM** STUDENT;

COUNT(PHONE)
4

- **SUM:** SUM function is used to add the values of an attribute in a relation. e.g;

SELECT SUM (AGE) **FROM** STUDENT;

SUM(AGE)
74

In the same way, MIN, MAX and AVG can be used. As we have seen above, all aggregation functions return only 1 row.

AVERAGE: It gives the average values of the tupples. It is also defined as sum divided by count values.
Syntax:AVG(attributename)
OR
Syntax:SUM(attributename)/COUNT(attributename)
The above mentioned syntax also retrieves the average value of tupples.

MAXIMUM:It extracts the maximum value among the set of tupples.
Syntax:MAX(attributename)

MINIMUM:It extracts the minimum value amongst the set of all the tupples.
Syntax:MIN(attributename)

GROUP BY: Group by is used to group the tuples of a relation based on an attribute or group of attribute. It is always combined with aggregation function which is computed on group. e.g.;
SELECT ADDRESS, **SUM**(AGE) **FROM** STUDENT
GROUP BY (ADDRESS);
In this query, SUM(**AGE**) will be computed but not for entire table but for each address. i.e.; sum of AGE for address DELHI(18+18=36) and similarly for other address as well. The output is:

ADDRESS	**SUM(AGE)**
DELHI	36
GURGAON	18
ROHTAK	20

If we try to execute the query given below, it will result in error because although we have computed SUM(AGE) for each address, there are more than 1 ROLL_NO for each address we have grouped. So it can't be displayed in result set. We need to use aggregate functions on columns after SELECT statement to make sense of the resulting set whenever we are using GROUP BY.

SELECT ROLL_NO, ADDRESS, **SUM**(AGE) **FROM** STUDENT
GROUP BY (ADDRESS);

NOTE: An attribute which is not a part of GROUP BY clause can't be used for selection. Any attribute which is part of GROUP BY CLAUSE can be used for selection but it is not mandatory. But we could use attributes which are not a part of the GROUP BY clause in an aggregate function.
Quiz on SQL
Article Contributed by Sonal Tuteja. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

SQL Server: DISTINCT Clause

This SQL Server tutorial explains how to use the **DISTINCT clause** in SQL Server (Transact-SQL) with syntax and examples.

Description

The SQL Server (Transact-SQL) DISTINCT clause is used to remove duplicates from the result set. The DISTINCT clause can only be used with SELECT statements.

Syntax

The syntax for the DISTINCT clause in SQL Server (Transact-SQL) is:

```
SELECT DISTINCT expressions

FROM tables

[WHERE conditions];
```

Parameters or Arguments

expressions

The columns or calculations that you wish to retrieve.

tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

WHERE conditions

Optional. The conditions that must be met for the records to be selected.

Note

- When only one expression is provided in the DISTINCT clause, the query will return the unique values for that expression.
- When more than one expression is provided in the DISTINCT clause, the query will retrieve unique combinations for the expressions listed.
- In SQL Server, the DISTINCT clause doesn't ignore NULL values. So when using the DISTINCT clause in your SQL statement, your result set will include NULL as a distinct value.

Example - With Single Expression

Let's look at the simplest SQL Server DISTINCT clause example. We can use the SQL Server DISTINCT clause to return a single field that removes the duplicates from the result set.

For example:

```
SELECT DISTINCT last_name

FROM employees

WHERE employee_id >= 50;
```

This SQL Server DISTINCT example would return all unique *last_name* values from the *employees* table where the *employee_id* is greater than or equal to 50.

Example - With Multiple Expressions

Let's look at how you might use the SQL Server DISTINCT clause to remove duplicates from more than one field in your SELECT statement.

For example:

```
SELECT DISTINCT first_name, last_name

FROM employees

WHERE employee_id >=50

ORDER BY last_name;
```

This SQL Server DISTINCT clause example would return each unique *first_name* and *last_name* combination from the *employees* table where the *employee_id* is greater than or equal to 50. The results are sorted in ascending order by *last_name*.

In this case, the DISTINCT applies to each field listed after the DISTINCT keyword, and therefore returns distinct combinations.

The SQL WHERE Clause

The `WHERE` clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

WHERE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Demo Database

Below is a selection from the "Customers" table in the Northwind sample database:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico

3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

WHERE Clause Example

The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:

WHERE Clause Example

The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:

Example

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

Number of Records: 4

ProductID	ProductName	SupplierID	CategoryID	Unit	Price

1	Chais	1	1	10 boxes x 20 bags	18
35	Steeleye Stout	16	1	24 - 12 oz bottles	18
39	Chartreuse verte	18	1	750 cc per bottle	18
76	Lakkalikööri	23	1	500 ml	18

Text Fields vs. Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes:

Example

```
SELECT * FROM Customers
WHERE CustomerID=1;
```

```
SELECT * FROM Products

WHERE Price = 18;
```

SQL Server: WHERE Clause

This SQL Server tutorial explains how to use the **WHERE clause** in SQL Server (Transact-SQL) with syntax and examples.

Description

The SQL Server (Transact-SQL) WHERE clause is used to filter the results from a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax

The syntax for the WHERE clause in SQL Server (Transact-SQL) is:

```
WHERE conditions;
```

Parameters or Arguments

conditions

The conditions that must be met for records to be selected.

Example - With Single condition

It is difficult to explain the syntax for the SQL Server WHERE clause, so let's look at some examples.

We'll start by looking at how to use the WHERE clause with only a single condition.

For example:

```
SELECT *

FROM employees

WHERE first_name = 'Jane';
```

In this SQL Server WHERE clause example, we've used the WHERE clause to filter our results from the *employees* table. The SELECT statement above would return all rows from the *employees* table where the *first_name* is 'Jane'. Because the * is used in the SELECT, all fields from the *employees* table would appear in the result set.

Example - Using AND condition

Let's look at how to use the WHERE clause with the AND condition.

For example:

```
SELECT *

FROM employees

WHERE last_name = 'Anderson'

AND employee_id >= 3000;
```

This SQL Server WHERE clause example uses the WHERE clause to define multiple conditions. In this case, this SELECT statement uses the AND condition to return all *employees* that have a *last_name* of 'Anderson' and the *employee_id* is greater than or equal to 3000.

Example - Using OR condition

Let's look at how to use the WHERE clause with the OR condition.

For example:

```
SELECT employee_id, last_name, first_name

FROM employees

WHERE last_name = 'Johnson'

OR first_name = 'Danielle';
```

This SQL Server WHERE clause example uses the WHERE clause to define multiple conditions, but instead of using the AND condition, it uses the OR condition. In this case, this SELECT statement would return all *employee_id*, *last_name*, and *first_name* values from the *employees* table where the *last_name* is 'Johnson' or the *first_name* is 'Danielle'.

Example - Combining AND & OR conditions

Let's look at how to use the WHERE clause when we combine the AND & OR conditions in a single SQL statement.

For example:

```
SELECT *

FROM employees

WHERE (state = 'California' AND last_name = 'Smith')
```

```
OR (employee_id = 82);
```

This SQL Server WHERE clause example uses the WHERE clause to define multiple conditions, but it combines the AND condition and the OR condition. This example would return all *employees* that reside in the *state* of 'California' and whose *last_name* is 'Smith' as well as all employees whose *employee_id* is equal to 82.

The parentheses determine the order that the AND and OR conditions are evaluated. Just like you learned in the order of operations in Math class!

Example - Joining Tables

Let's look at how to use the WHERE clause when we join multiple tables together.

For example:

```
SELECT employees.employee_id, contacts.last_name

FROM employees

INNER JOIN contacts

ON employees.employee_id = contacts.contact_id

WHERE employees.first_name = 'Sarah';
```

This SQL Server WHERE clause example uses the WHERE clause to join multiple tables together in a single SELECT statement. This SELECT statement would return all rows where the *first_name* in the *employees* table is 'Sarah'. And the *employees* and *contacts* tables are joined on the *employee_id* from the *employees* table and the *contact_id* from the *contacts* table.

SQL Server: ORDER BY Clause

This SQL Server tutorial explains how to use the **ORDER BY clause** in SQL Server (Transact-SQL) with syntax and examples.

Description

The SQL Server (Transact-SQL) ORDER BY clause is used to sort the records in your result set. The ORDER BY clause can only be used in SELECT statements.

Syntax

The syntax for the ORDER BY clause in SQL Server (Transact-SQL) is:

```
SELECT expressions

FROM tables

[WHERE conditions]

ORDER BY expression [ ASC | DESC ];
```

Parameters or Arguments

expressions

The columns or calculations that you wish to retrieve.

tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

WHERE conditions

Optional. The conditions that must be met for the records to be selected.

ASC

Optional. It sorts the result set in ascending order by *expression* (default, if no modifier is provider).

DESC

Optional. It sorts the result set in descending order by *expression*.

Note

- If the ASC or DESC modifier is not provided in the ORDER BY clause, the results will be sorted by *expression* in ascending order. This is equivalent to `ORDER BY expression ASC`.

Example - Sorting without using ASC/DESC attribute

The SQL Server ORDER BY clause can be used without specifying the ASC or DESC value. When this attribute is omitted from the ORDER BY clause, the sort order is defaulted to ASC or ascending order.

For example:

```
SELECT last_name

FROM employees

WHERE employee_id > 1000

ORDER BY last_name;
```

This SQL Server ORDER BY example would return all records sorted by the *last_name* field in ascending order and would be equivalent to the following ORDER BY clause:

```
SELECT last_name

FROM employees

WHERE employee_id > 1000

ORDER BY last_name ASC;
```

Most programmers omit the ASC attribute if sorting in ascending order.

Example - Sorting in descending order

When sorting your result set in descending order, you use the DESC attribute in your ORDER BY clause.

For example:

```
SELECT last_name

FROM employees

WHERE first_name = 'Sarah'

ORDER BY last_name DESC;
```

This SQL Server ORDER BY example would return all records sorted by the *last_name* field in descending order.

Example - Sorting by relative position

You can also use the SQL Server ORDER BY clause to sort by relative position in the result set, where the first field in the result set is 1. The next field is 2, and so on.

For example:

```
SELECT last_name

FROM employees

WHERE last_name = 'Anderson'

ORDER BY 1 DESC;
```

This SQL Server ORDER BY would return all records sorted by the *last_name* field in descending order, since the *last_name* field is in position #1 in the result set and would be equivalent to the following ORDER BY clause:

```
SELECT last_name

FROM employees

WHERE last_name = 'Anderson'

ORDER BY last_name DESC;
```

Example - Using both ASC and DESC attributes

When sorting your result set using the SQL Server ORDER BY clause, you can use the ASC and DESC attributes in a single SELECT statement.

For example:

```
SELECT last_name, first_name

FROM employees

WHERE last_name = 'Johnson'

ORDER BY last_name DESC, first_name ASC;
```

This SQL Server ORDER BY would return all records sorted by the *last_name* field in descending order, with a secondary sort by *first_name* in ascending order.

SQL Server: GROUP BY Clause

This SQL Server tutorial explains how to use the **GROUP BY clause** in SQL Server (Transact-SQL) with syntax and examples.

Description

The SQL Server (Transact-SQL) GROUP BY clause is used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

Syntax

The syntax for the GROUP BY clause in SQL Server (Transact-SQL) is:

```
SELECT expression1, expression2, ... expression_n,
       aggregate_function (expression)
FROM tables
[WHERE conditions]
GROUP BY expression1, expression2, ... expression_n;
```

Parameters or Arguments

expression1, expression2, ... expression_n

The expressions that are not encapsulated within an aggregate function and must be included in the GROUP BY clause.

aggregate_function

It can be a function such as SUM, COUNT, MIN, MAX, or AVG functions.

tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

WHERE conditions

Optional. The conditions that must be met for the records to be selected.

SQL Set Operation

The SQL Set operation is used to combine the two or more SQL SELECT statements.

Types of Set Operation

1. Union
2. UnionAll
3. Intersect
4. Minus

1. Union

- The SQL Union operation is used to combine the result of two or more SQL SELECT queries.
- In the union operation, all the number of datatype and columns must be same in both the tables on which UNION operation is being applied.
- The union operation eliminates the duplicate rows from its resultset.

Syntax

```
1. SELECT column_name FROM table1
2. UNION
3. SELECT column_name FROM table2;
```

Example:

The First table

ID	NAME
1	Jack
2	Harry
3	Jackson

The Second table

ID	NAME
3	Jackson

4	Stephan
5	David

Union SQL query will be:

1. SELECT * FROM First
2. UNION
3. SELECT * FROM Second;

The resultset table will look like:

ID	NAME
1	Jack
2	Harry
3	Jackson
4	Stephan
5	David

2. Union All

Union All operation is equal to the Union operation. It returns the set without removing duplication and sorting the data.

Syntax:

1. SELECT column_name FROM table1
2. UNION ALL
3. SELECT column_name FROM table2;

Example: Using the above First and Second table.

Union All query will be like:

1. SELECT * FROM First
2. UNION ALL
3. SELECT * FROM Second;

The resultset table will look like:

ID	NAME
1	Jack
2	Harry
3	Jackson
3	Jackson
4	Stephan
5	David

3. Intersect

- It is used to combine two SELECT statements. The Intersect operation returns the common rows from both the SELECT statements.
- In the Intersect operation, the number of datatype and columns must be the same.
- It has no duplicates and it arranges the data in ascending order by default.

Syntax

1. SELECT column_name FROM table1
2. INTERSECT
3. SELECT column_name FROM table2;

Example:

Using the above First and Second table.

Intersect query will be:

1. SELECT * FROM First
2. INTERSECT
3. SELECT * FROM Second;

The resultset table will look like:

ID	NAME
3	Jackson

4. Minus

- It combines the result of two SELECT statements. Minus operator is used to display the rows which are present in the first query but absent in the second query.
- It has no duplicates and data arranged in ascending order by default.

Syntax:

1. SELECT column_name FROM table1
2. MINUS
3. SELECT column_name FROM table2;

Example

Using the above First and Second table.

Minus query will be:

1. SELECT * FROM First
2. MINUS
3. SELECT * FROM Second;

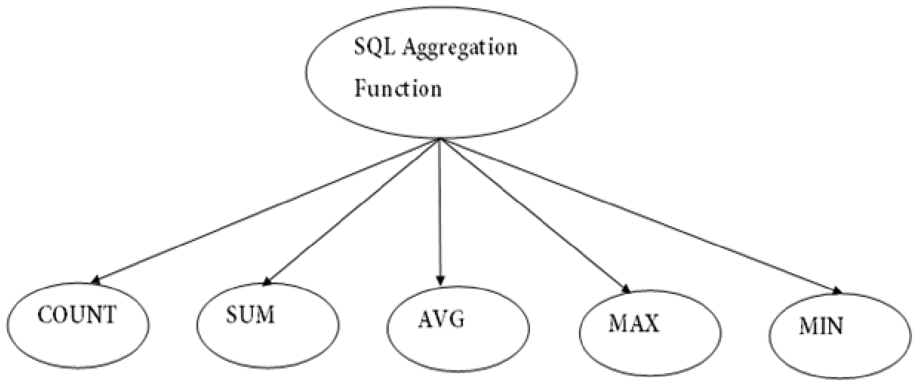
The resultset table will look like:

ID	NAME
1	Jack
2	Harry

SQL Aggregate Functions

- SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.
- It is also used to summarize the data.

Types of SQL Aggregation Function



1. COUNT FUNCTION

- COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.
- COUNT function uses the COUNT(*) that returns the count of all the rows in a specified table. COUNT(*) considers duplicate and Null.

Syntax

1. COUNT(*)
2. or
3. COUNT([ALL|DISTINCT] expression)

Sample table:

PRODUCT_MAST

PRODUCT	COMPANY	QTY	RATE	COST
Item1	Com1	2	10	20
Item2	Com2	3	25	75
Item3	Com1	2	30	60

Item4	Com3	5	10	50
Item5	Com2	2	20	40
Item6	Cpm1	3	25	75
Item7	Com1	5	30	150
Item8	Com1	3	10	30
Item9	Com2	2	25	50
Item10	Com3	4	30	120

Example: COUNT()

1. SELECT COUNT(*)
2. FROM PRODUCT_MAST;

Output:

```
10
```

Example: COUNT with WHERE

1. SELECT COUNT(*)
2. FROM PRODUCT_MAST;
3. WHERE RATE>=20;

Output:

```
7
```

Example: COUNT() with DISTINCT

1. SELECT COUNT(DISTINCT COMPANY)
2. FROM PRODUCT_MAST;

Output:

```
3
```

Example: COUNT() with GROUP BY

1. SELECT COMPANY, COUNT(*)
2. FROM PRODUCT_MAST

```
3.	GROUP BY COMPANY;
```

Output:

```
Com1     5
Com2     3
Com3     2
```

Example: COUNT() with HAVING

```
1.	SELECT COMPANY, COUNT(*)
2.	FROM PRODUCT_MAST
3.	GROUP BY COMPANY
4.	HAVING COUNT(*)>2;
```

Output:

```
Com1     5
Com2     3
```

2. SUM Function

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

Syntax

```
1.	SUM()
2.	or
3.	SUM( [ALL|DISTINCT] expression )
```

Example: SUM()

```
1.	SELECT SUM(COST)
2.	FROM PRODUCT_MAST;
```

Output:

```
670
```

Example: SUM() with WHERE

```
1.	SELECT SUM(COST)
2.	FROM PRODUCT_MAST
3.	WHERE QTY>3;
```

Output:

```
320
```

Example: SUM() with GROUP BY

```
SELECT SUM(COST)
FROM PRODUCT_MAST
WHERE QTY>3
GROUP BY COMPANY;
```

Output:

```
Com1    150
Com2    170
```

Example: SUM() with HAVING

```
SELECT COMPANY, SUM(COST)
FROM PRODUCT_MAST
GROUP BY COMPANY
HAVING SUM(COST)>=170;
```

Output:

```
Com1    335
Com3    170
```

3. AVG function

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

Syntax

1. AVG()
2. or
3. AVG([ALL|DISTINCT] expression)

Example:

```
SELECT AVG(COST)
FROM PRODUCT_MAST;
```

Output:

```
67.00
```

4. MAX Function

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

Syntax

```
1. MAX()
2. or
3. MAX( [ALL|DISTINCT] expression )
```

Example:

```
1. SELECT MAX(RATE)
2. FROM PRODUCT_MAST;
```

```
30
```

5. MIN Function

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

Syntax

```
1. MIN()
2. or
3. MIN( [ALL|DISTINCT] expression )
```

Example:

```
1. SELECT MIN(RATE)
2. FROM PRODUCT_MAST;
```

Output:

```
10
```

Null Values in DBMS

Null Values in DBMS

In this article, we will learn about Null values in DBMS.

- Special value that is supported by SQL is called as `null` which is used to represent values of attributes that are unknown or do not apply for that particular row
- For example age of a particular student is not available in the age column of student table then it is represented as null but not as zero
- It is important to know that ***null values is always different from zero value***
- A null value is used to represent ***the following different interpretations***
    - ***Value unknown*** (value exists but is not known)
    - ***Value not available*** (exists but is purposely hidden)
    - ***Attribute not applicable*** (undefined for that row)
- SQL provides special operators and functions to deal with data involving null values

Consider the sample table '*emp'*

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20
7369	SMITH	CLERK	7902	17-DEC-80	800	500	20

IS NULL operator

All ***operations upon null values present in the table must be done using this 'is null' operator* .**we cannot compare null value using the assignment operator

Example

```
select * from emp
where comm is null
```

O/P

```
4 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	–	17-NOV-81	5000	–	10
7566	JONES	MANAGER	7839	02-APR-81	2975	–	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	–	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	–	20

The details of those employees whose commission value is Null are displayed.

IS NOT NULL

```
select * from emp
where comm is not null;
```

O/P

```
3 rows selected.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	500	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	500	10
7369	SMITH	CLERK	7902	17-DEC-80	800	500	20

Details of all those employees whose Commission value is not null value are displayed.

NOT NULL Constraint

- Not all constraints prevents a column to contain null values
- Once ***not null is applied to a particular column, you cannot enter null values to that column*** and restricted to maintain only some proper value other than null
- A ***not-null constraint cannot be applied at table level***

Example

```
CREATE TABLE STUDENT
(
   ID   INT               NOT NULL,
   NAME VARCHAR (20)       NOT NULL,
   AGE  INT               NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

- In the above example, we have applied not null on three columns ID, name and age which means ***whenever a record is entered using insert statement all three columns should contain a value other than null***
- We have two other columns address and salary, ***where not null is not applied*** which means that ***you can leave the row as empty or use null value while inserting the record into the table.***

NVL() NULL Function

- Using NVL function you can ***substitute a value in the place of NULL values.***
- The substituted value then ***temporarily replaces the NULL values in your calculations or expression***. Remember that the substituted value ***only replaces the NULL value temporarily*** for the session and ***does not affect the value stored in the table.***
- Here is the syntax of NVL function.

```
NVL (exp, replacement-exp)
```

- As you can see NVL function ***takes two parameters exp and replacement exp***. ***First parameter exp can be a column name of a table or an arithmetic expression*** and the ***second parameter replacement expression will be the value which you want to substitute*** when a NULL value is encountered.
- Always remember the ***data type of both the parameters must match otherwise*** the compiler will raise an error.

Example

```
SELECT NVL (comm, 500) FROM employees
 WHERE salary>1000;
```

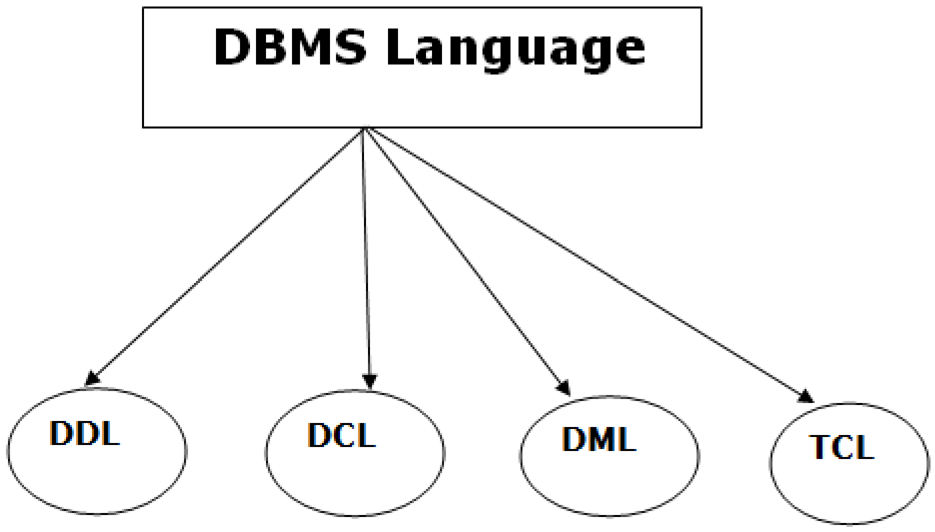
- On execution ***all the null values in the result set will get replaced by 500.***
- Similarly we can use NVL null function while performing arithmetic expression.
- Again let's take the same arithmetic expression which we used in the previous query where we added 100 to the values of commission column.

```
SELECT NVL(comm,100), NVL(comm,100)+100 FROM employees WHERE salary>1000;
```

Database Language

- A DBMS has appropriate languages and interfaces to express database queries and updates.
- Database languages can be used to read, store and update the data in the database.

Types of Database Language



Data Definition Language

Data Definition Language (DDL) statements are used to classify the database structure or schema. It is a type of language that allows the DBA or user to depict and name those entities, attributes, and relationships that are required for the application along with any associated integrity and security constraints. Here are the lists of tasks that come under DDL:

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

Data Manipulation Language

A language that offers a set of operations to support the fundamental data manipulation operations on the data held in the database. Data Manipulation Language (DML) statements are used to manage data within schema objects. Here are the lists of tasks that come under DML:

- **SELECT** - It retrieves data from a database
- **INSERT** - It inserts data into a table
- **UPDATE** - It updates existing data within a table
- **DELETE** - It deletes all records from a table, the space for the records remain
- **MERGE** - UPSERT operation (insert or update)
- **CALL** - It calls a PL/SQL or Java subprogram
- **EXPLAIN PLAN** - It explains the access path to data
- **LOCK TABLE** - It controls concurrency

Data Control Language:

There are two other forms of database sub-languages. The Data Control Language (DCL) is used to control privilege in Databases. To perform any operation in the database, such as for creating tables, sequences, or views, we need privileges. Privileges are of two types,

- **System** - creating a session, table, etc. are all types of system privilege.
- **Object** - any command or query to work on tables comes under object privilege. DCL is used to define two commands. These are:
- **Grant** - It gives user access privileges to a database.
- **Revoke** - It takes back permissions from the user.

Transaction Control Language

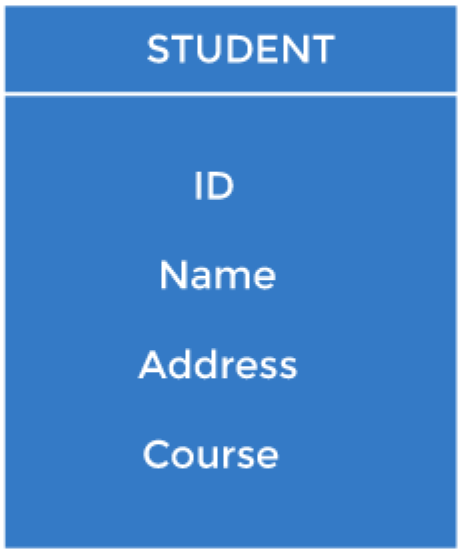
Transaction Control statements are used to run the changes made by DML statements. It allows statements to be grouped into logical transactions.

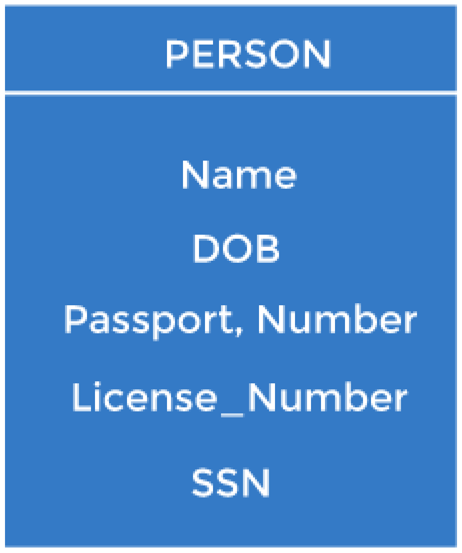
- **COMMIT** - It saves the work done
- **SAVEPOINT** - It identifies a point in a transaction to which you can later roll back
- **ROLLBACK** - It restores the database to original since the last COMMIT
- **SET TRANSACTION** - It changes the transaction options like isolation level and what rollback segment to use

Keys

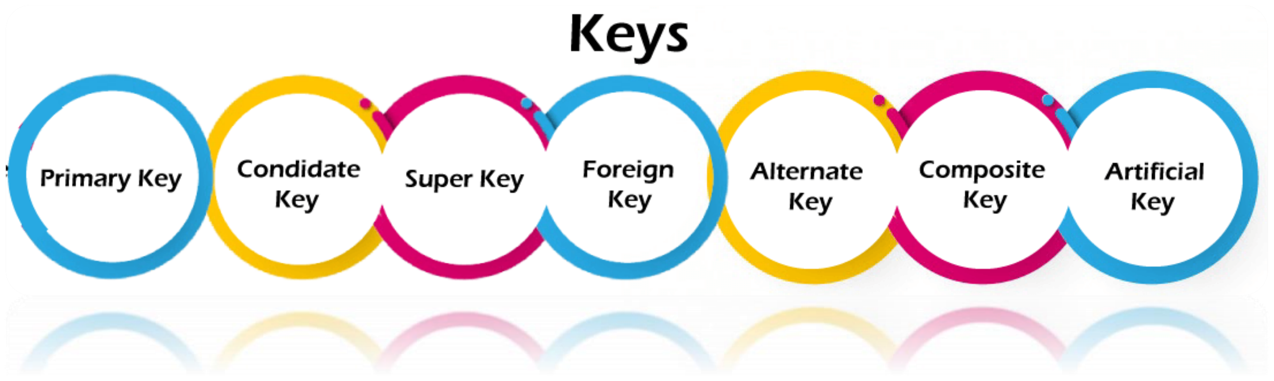
- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

For example, ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.





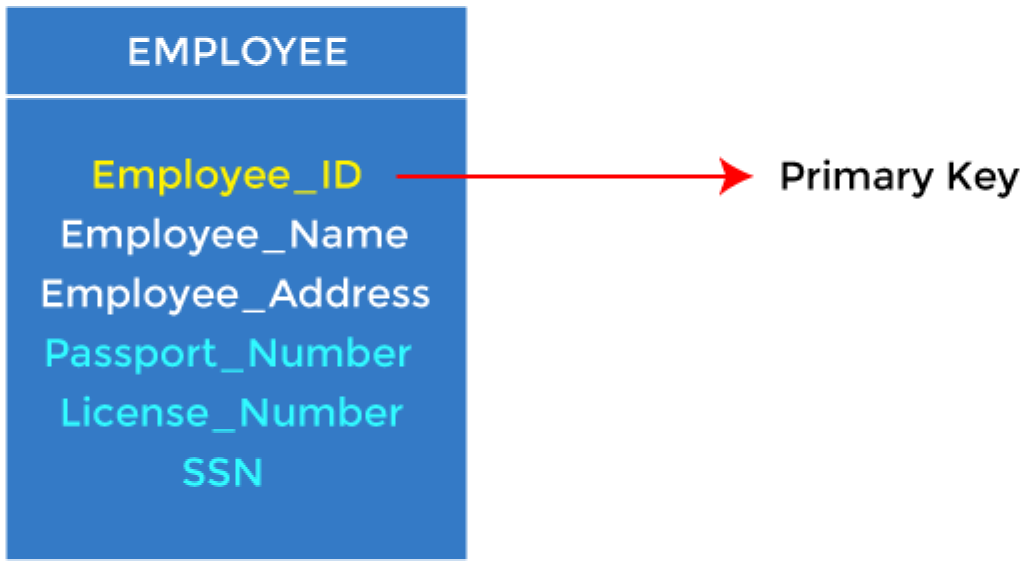
Types of keys:



1. Primary key

- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.

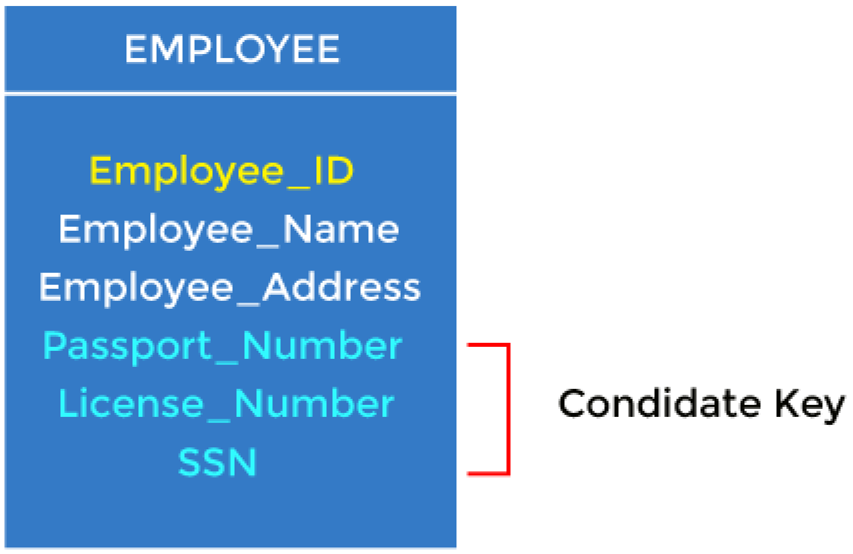
- In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.



2. Candidate key

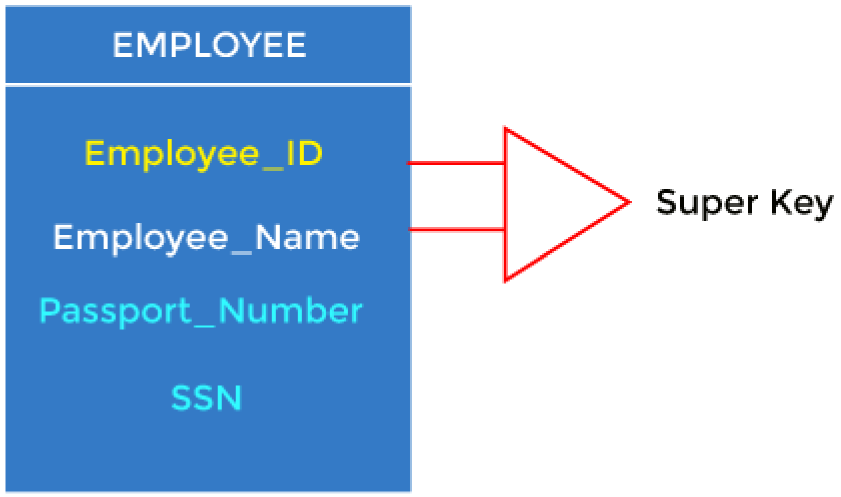
- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



3. Super Key

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.

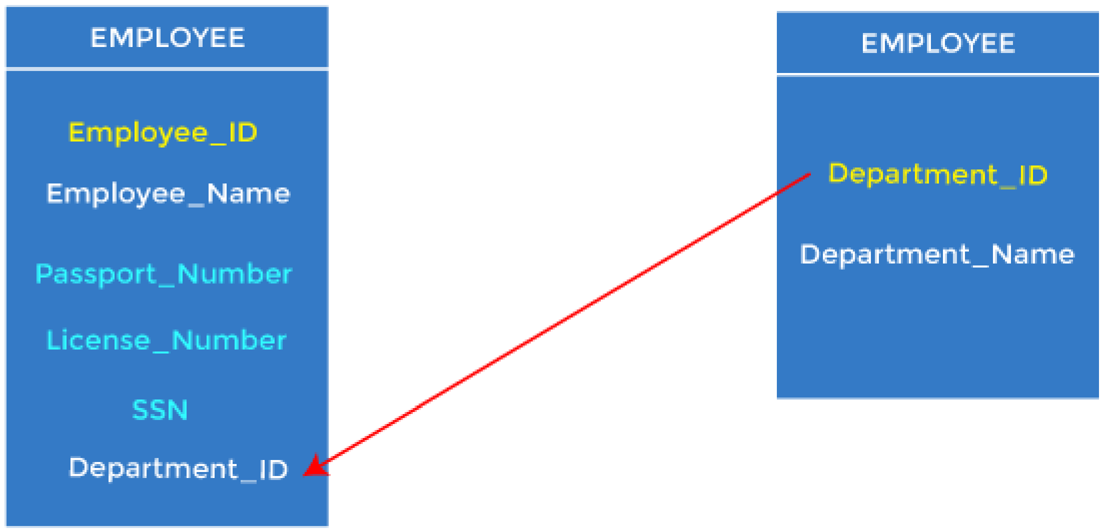


For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key

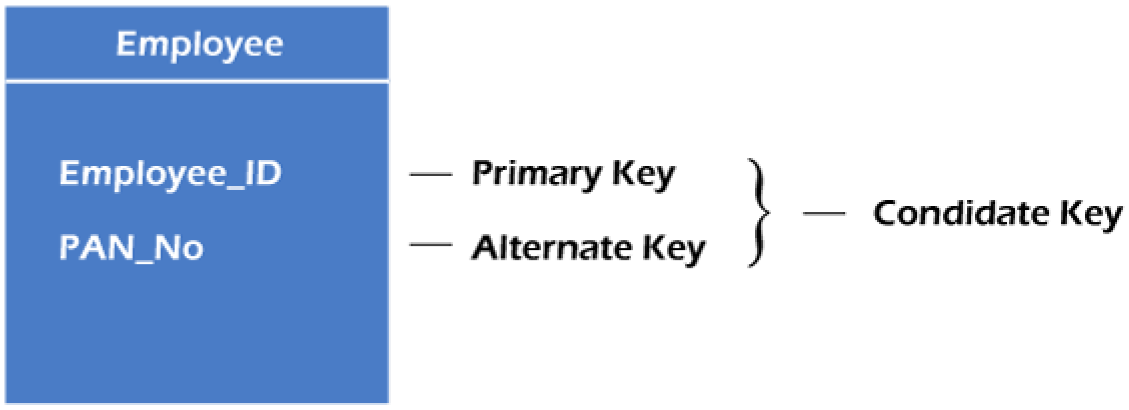
- Foreign keys are the column of the table used to point to the primary key of another table.
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



5. Alternate key

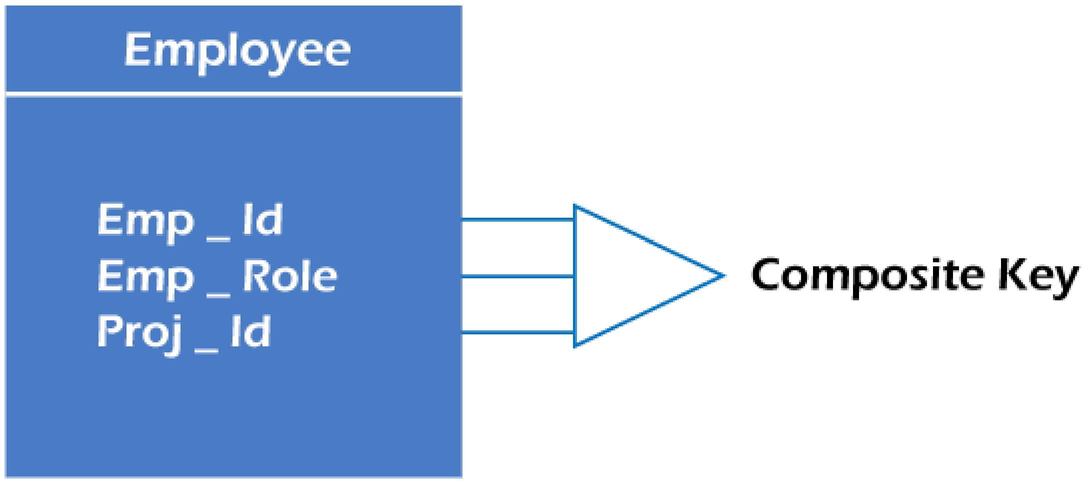
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key. **In other words,** the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

For example, employee relation has two attributes, Employee_Id and PAN_No, that act as candidate keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.

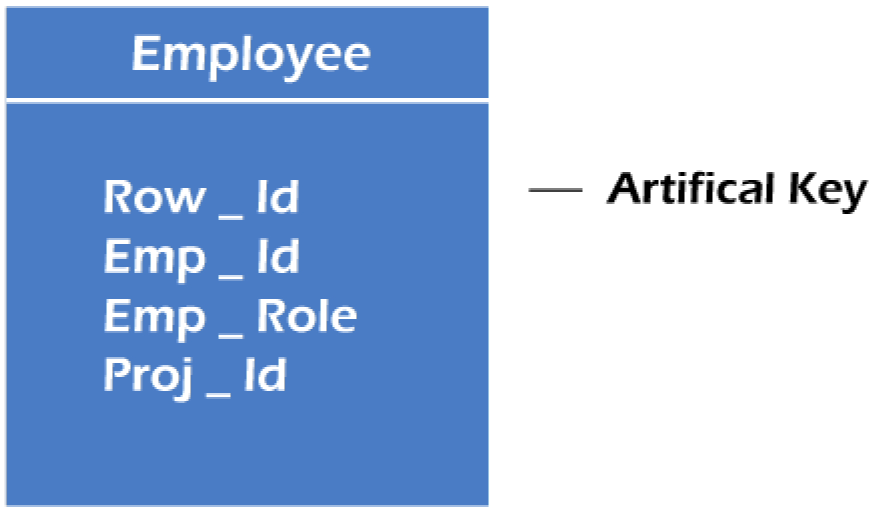


6. Composite key

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.



For example, in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.



7. Artificial key

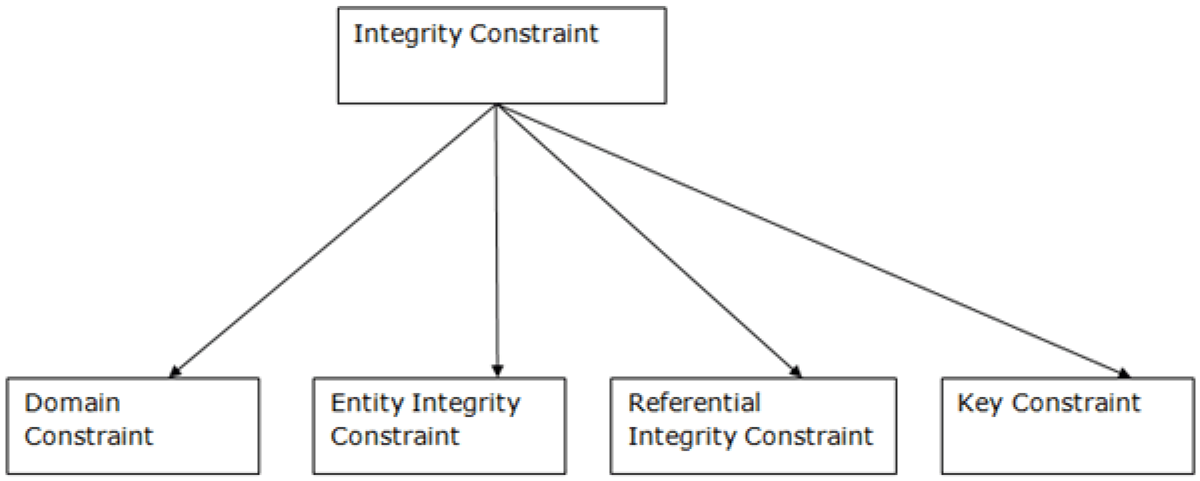
The key created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are usually numbered in a serial order.

For example, the primary key, which is composed of Emp_ID, Emp_role, and Proj_ID, is large in employee relations. So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.

Integrity Constraints

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

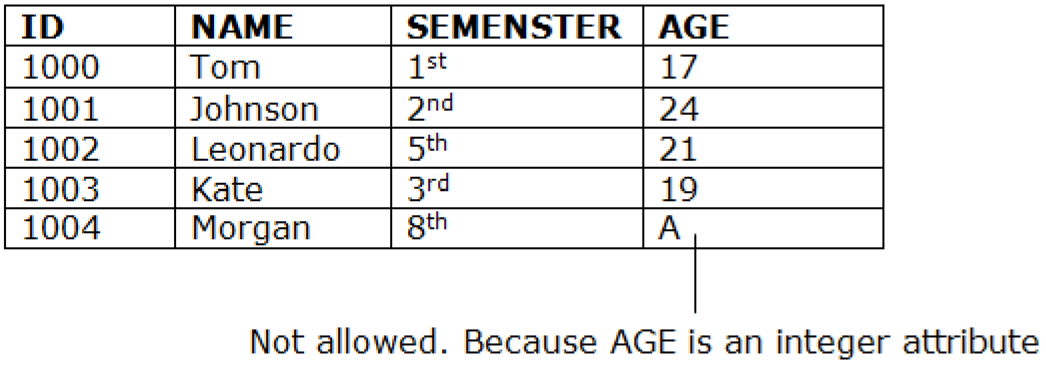
Types of Integrity Constraint



1. Domain constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example:



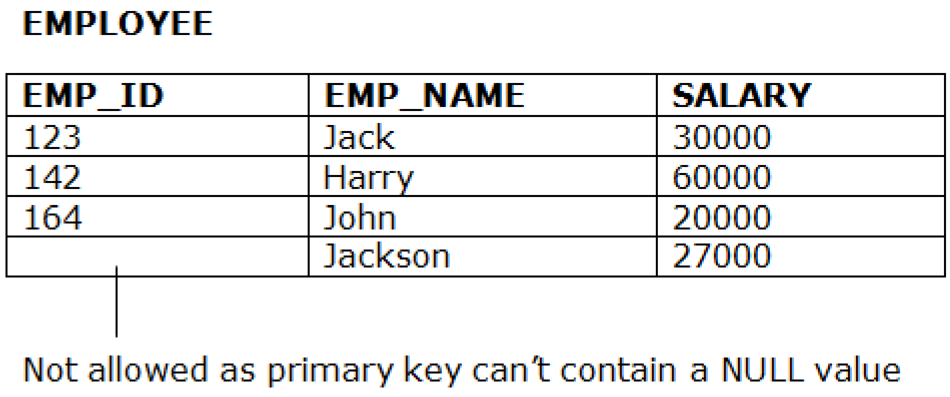
ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:



EMPLOYEE

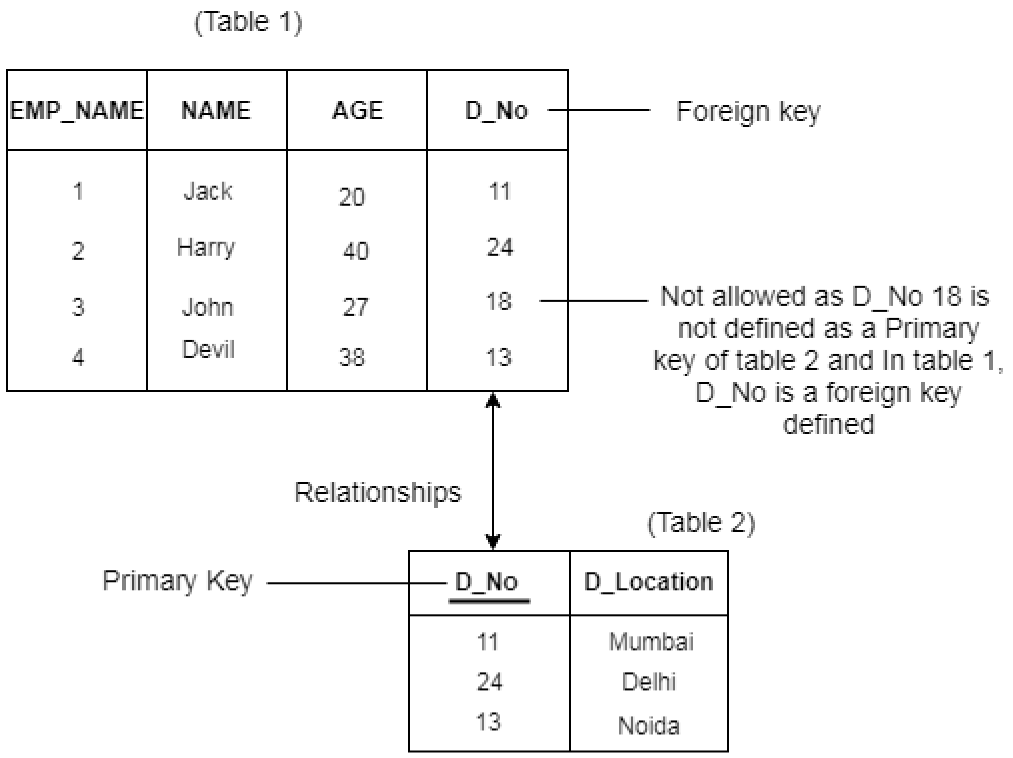
EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:


(Table 1)
Foreign key
Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined
Relationships
(Table 2)
Primary Key

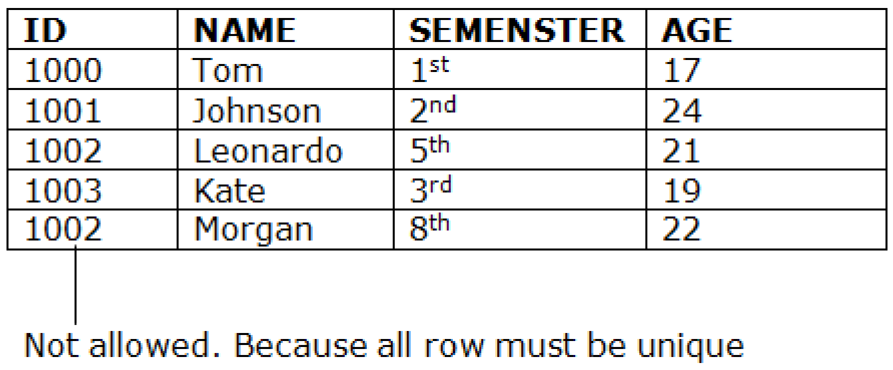
EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example:



ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

Views in SQL

- Views in SQL are considered as a virtual table. A view also contains rows and columns.
- To create the view, we can select the fields from one or more tables present in the database.
- A view can either have specific rows based on certain condition or all the rows of a table.

Sample table:

Student_Detail

STU_ID	NAME	ADDRESS
1	Stephan	Delhi
2	Kathrin	Noida
3	David	Ghaziabad
4	Alina	Gurugram

Student_Marks

STU_ID	NAME	MARKS	AGE
1	Stephan	97	19
2	Kathrin	86	21
3	David	74	18
4	Alina	90	20
5	John	96	18

1. Creating view

A view can be created using the **CREATE VIEW** statement. We can create a view from a single table or multiple tables.

Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

2. Creating View from a single table

In this example, we create a View named DetailsView from the table Student_Detail.
Query:

```
CREATE VIEW DetailsView AS  SELECT NAME, ADDRESS  FROM Student_Details  WHERE STU_ID < 4;
```

Just like table query, we can query the view to view the data.

```
SELECT * FROM DetailsView;
```

Output:

NAME	ADDRESS
Stephan	Delhi
Kathrin	Noida
David	Ghaziabad

3. Creating View from multiple tables

View from multiple tables can be created by simply include multiple tables in the SELECT statement.
In the given example, a view is created named MarksView from two tables Student_Detail and Student_Marks.
Query:

```
CREATE VIEW MarksView AS SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Stud
ent_Marks.MARKS FROM Student_Detail, Student_Mark WHERE Student_Detail.NAME = St
udent_Marks.NAME;
```

To display data of View MarksView:

```
SELECT * FROM MarksView;
```

NAME	ADDRESS	MARKS
Stephan	Delhi	97
Kathrin	Noida	86
David	Ghaziabad	74
Alina	Gurugram	90

4. Deleting View

A view can be deleted using the Drop View statement.

Syntax

DROP VIEW view_name;

Example:

If we want to delete the View **MarksView**, we can do this as:

DROP VIEW MarksView;

TRIGGER:

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

Syntax:

create trigger [trigger_name]

[before | after]

{insert | update | delete}

on [table_name]

[for each row]

[trigger_body]

Explanation of syntax:

1. create trigger [trigger_name]: Creates or replaces an existing trigger with the trigger_name.
2. [before | after]: This specifies when the trigger will be executed.
3. {insert | update | delete}: This specifies the DML operation.
4. on [table_name]: This specifies the name of the table associated with the trigger.
5. [for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.
6. [trigger_body]: This provides the operation to be performed as trigger is fired

BEFORE and AFTER of Trigger:

BEFORE triggers run the trigger action before the triggering statement is run.
AFTER triggers run the trigger action after the triggering statement is run.

Example in After trigger:

To create a row level trigger for the CUSTOMERS table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. Trigger Event appear to another table example to insert event, update event, delete:

insertion:

```
create table emp(emp_id int primary key,emp_name varchar(20) not null,age int)
create table emp_log(emp_id int,action varchar(10),atime datetime)

select*from emp
select*from emp_log
```

Trigger-Insert

```
create trigger employee_trigger_insert on emp after insert
as
begin
insert into emp_log select emp_id,'inserted',getdate() from inserted
end

insert into emp values(1,'Jay',18)
insert into emp values(2,'sri',18)
```

Emp		
1	Jay	18
2	Sri	18
Emp_log		
1	inserted	2022-03-24 10:49:25.283
2	inserted	2022-03-24 10:50:42.740

Trigger-Update

```
create trigger employee_trigger_update on emp after update
as
begin
insert into emp_log select emp_id,'updated',getdate() from inserted
end
update set emp set emp_name='jothi' where emp_id='1'
```

Emp		
1	jothi	18
2	Sri	18
Emp_log		
1	inserted	2022-03-24 10:49:25.283
2	inserted	2022-03-24 10:50:42.740
1	Updated	2022-03-25 10:50:52.640

Trigger-delete

```
create trigger employee_trigger_delete on emp after delete
as
begin
insert into emp_log select emp_id,'deleted',getdate() from deleted
end
```

delete form emp where emp_id=1

Emp		
1	jothi	18
2	Sri	18
Emp_log		
1	inserted	2022-03-24 10:49:25.283
2	inserted	2022-03-24 10:50:42.740
1	Updated	2022-03-25 10:50:52.640
1	Deleted	2022-03-26 09:40:52.640

Drop Trigger

Drop trigger employee_trigger_delete

What are Assertions?

When a constraint involves 2 (or) more tables, the table constraint mechanism is sometimes hard and results may not come as expected. To cover such situation SQL supports the creation of assertions that are constraints not associated with only one table. And an assertion statement should ensure a certain condition will always exist in the database. DBMS always checks the assertion whenever modifications are done in the corresponding table.

Syntax –

CREATE ASSERTION [assertion_name]

CHECK ([condition]);

Example –

CREATE TABLE sailors (sid int,sname varchar(20), rating int,primary key(sid),

CHECK(rating >= 1 AND rating <=10)

CHECK((select count(s.sid) from sailors s) + (select count(b.bid)from boats b)<100));

Difference between Assertions and Triggers:

S.No	Assertions	Triggers
1.	We can use Assertions when we know that the given particular condition is always true.	We can use Triggers even particular condition may or may not be true.
2.	When the SQL condition is not met then there are chances to an entire table or even Database to get locked up.	Triggers can catch errors if the condition of the query is not true.

S.No	Assertions	Triggers
3.	Assertions are not linked to specific table or event. It performs task specified or defined by the user.	It helps in maintaining the integrity constraints in the database tables, especially when the primary key and foreign key constraint are not defined.
4.	Assertions do not maintain any track of changes made in table.	Triggers maintain track of all changes occurred in table.
5.	Assertions have small syntax compared to Triggers.	They have large Syntax to indicate each and every specific of the created trigger.
6.	Modern databases do not use Assertions.	Triggers are very well used in modern databases.

Difference between Static and Dynamic SQL

Static (Embedded) SQL	Dynamic (Interactive) SQL
In Static SQL, how database will be accessed is predetermined in the embedded SQL statement.	In Dynamic SQL, how database will be accessed is determined at run time.
It is more swift and efficient.	It is less swift and efficient.
SQL statements are compiled at compile time.	SQL statements are compiled at run time.
Parsing, Validation, Optimization and Generation of application plan are done at compile time.	Parsing, Validation, Optimization and Generation of application plan are done at run time.
It is generally used for situations where data is distributed uniformly.	It is generally used for situations where data is distributed non uniformly.
EXECUTE IMMEDIATE, EXECUTE and PREPARE statements are not used.	EXECUTE IMMEDIATE, EXECUTE and PREPARE statements are used.
It is less flexible.	It is more flexible.

UNIT-3

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**.

Example:

roll_no	name	dept_name	dept_building
42	abc	CO	A4
43	pqr	IT	A3
44	xyz	CO	A4
45	xyz	IT	A3
46	mno	EC	B2
47	jkl	ME	B2

From the above table we can conclude some valid functional dependencies:

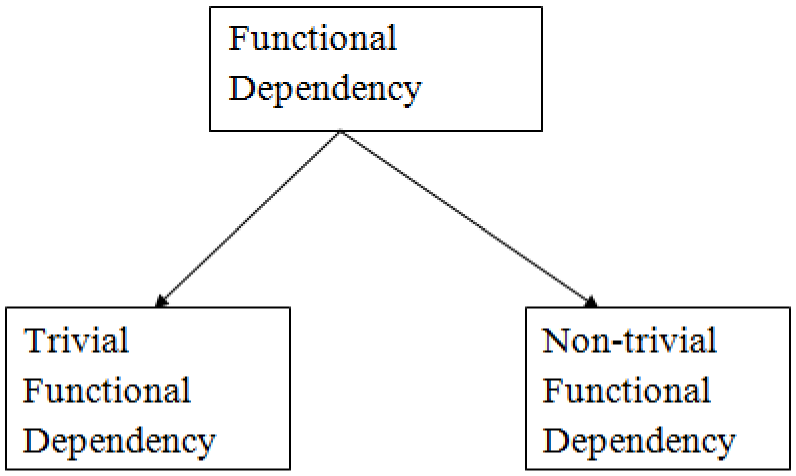
- roll_no → { name, dept_name, dept_building },→ Here, roll_no can determine values of fields name, dept_name and dept_building, hence a valid Functional dependency
- roll_no → dept_name , Since, roll_no can determine whole set of {name, dept_name, dept_building}, it can determine its subset dept_name also.
- dept_name → dept_building , Dept_name can identify the dept_building accurately, since departments with different dept_name will also have a different dept_building
- More valid functional dependencies: roll_no → name, {roll_no, name} ⇢ {dept_name, dept_building}, etc.

Here are some invalid functional dependencies:

- name → dept_name Students with the same name can have different dept_name, hence this is not a valid functional dependency.
- dept_building → dept_name There can be multiple departments in the same building, For example, in the above table departments ME and EC are in the same building B2, hence dept_building → dept_name is an invalid functional dependency.

- More invalid functional dependencies: name → roll_no, {name, dept_name} → roll_no, dept_building → roll_no, etc.

Types of Functional dependency


Functional Dependency
Trivial Functional Dependency
Non-trivial Functional Dependency

1. Trivial functional dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B

Example:

1. Consider a table with two columns Employee_Id and Employee_Name.
2. {Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as
3. Employee_Id is a subset of {Employee_Id, Employee_Name}.
4. Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are tri vial dependencies too.

2. Non-trivial functional dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A Intersection B is NULL, then A → B is called as complete non-trivial.

Example:

1. ID → Name,
2. Name → DOB

Inference Rule (IR):

- The Armstrong's axioms are the basic inference rule.
- Armstrong's axioms are used to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.

The Functional dependency has 6 types of inference rule:

1. Reflexive Rule (IR_1)

In the reflexive rule, if Y is a subset of X, then X determines Y.

1. If X ⊇ Y then X → Y

Example:

1. X = {a, b, c, d, e}
2. Y = {a, b, c}

2. Augmentation Rule (IR_2)

The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

1. If X → Y then XZ → YZ

Example:

For R(ABCD), **if** A → B then AC → BC

3. Transitive Rule (IR_3)

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

1. If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

4. Union Rule (IR_4)

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

1. If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

Proof:

1. $X \rightarrow Y$ (given)
2. $X \rightarrow Z$ (given)
3. $X \rightarrow XY$ (using IR_2 on 1 by augmentation with X. Where XX = X)
4. $XY \rightarrow YZ$ (using IR_2 on 2 by augmentation with Y)
5. $X \rightarrow YZ$ (using IR_3 on 3 and 4)

5. Decomposition Rule (IR_5)

Decomposition rule is also known as project rule. It is the reverse of union rule.

This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

1. If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

Proof:

1. $X \rightarrow YZ$ (given)
2. $YZ \rightarrow Y$ (using IR_1 Rule)
3. $X \rightarrow Y$ (using IR_3 on 1 and 2)

6. Pseudo transitive Rule (IR_6)

In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

1. If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$

Proof:

1. $X \rightarrow Y$ (given)
2. $WY \rightarrow Z$ (given)
3. $WX \rightarrow WY$ (using IR_2 on 1 by augmenting with W)
4. $WX \rightarrow Z$ (using IR_3 on 3 and 2)

UNIT-4

Concurrency Control

Concurrency Control is the management procedure that is required for controlling concurrent execution of the operations that take place on a database. But before knowing about concurrency control, we should know about concurrent execution.

Concurrent Execution in DBMS

- In a multi-user system, multiple users can access and use the same database at one time, which is known as the concurrent execution of the database. It means that the same database is executed simultaneously on a multi-user system by different users.
- While working on the database transactions, there occurs the requirement of using the database by multiple users for performing different operations, and in that case, concurrent execution of the database is performed.
- The thing is that the simultaneous execution that is performed should be done in an interleaved manner, and no operation should affect the other executing operations, thus maintaining the consistency of the database. Thus, on making the concurrent execution of the transaction operations, there occur several challenging problems that need to be solved.

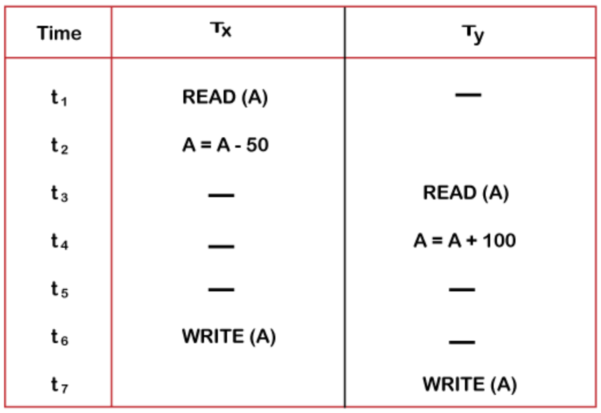
Problems with Concurrent Execution

In a database transaction, the two main operations are **READ** and **WRITE** operations. So, there is a need to manage these two operations in the concurrent execution of the transactions as if these operations are not performed in an interleaved manner, and the data may become inconsistent. So, the following problems occur with the Concurrent Execution of the operations:

Problem 1: Lost Update Problems (W - W Conflict)

The problem occurs when two different database transactions perform the read/write operations on the same database items in an interleaved manner (i.e., concurrent execution) that makes the values of the items incorrect hence making the database inconsistent.

For example:Consider the below diagram where two transactions T_X and T_Y, are performed on the same account A where the balance of account A is $300.



Time	Tx	Ty
t_1	READ (A)	—
t_2	A = A - 50	
t_3	—	READ (A)
t_4	—	A = A + 100
t_5	—	—
t_6	WRITE (A)	—
t_7		WRITE (A)

LOST UPDATE PROBLEM

○

- At time t1, transaction T_X reads the value of account A, i.e., $300 (only read).
- At time t2, transaction T_X deducts $50 from account A that becomes $250 (only deducted and not updated/write).
- Alternately, at time t3, transaction T_Y reads the value of account A that will be $300 only because T_X didn't update the value yet.
- At time t4, transaction T_Y adds $100 to account A that becomes $400 (only added but not updated/write).
- At time t6, transaction T_X writes the value of account A that will be updated as $250 only, as T_Y didn't update the value yet.
- Similarly, at time t7, transaction T_Y writes the values of account A, so it will write as done at time t4 that will be $400. It means the value written by T_X is lost, i.e., $250 is lost.

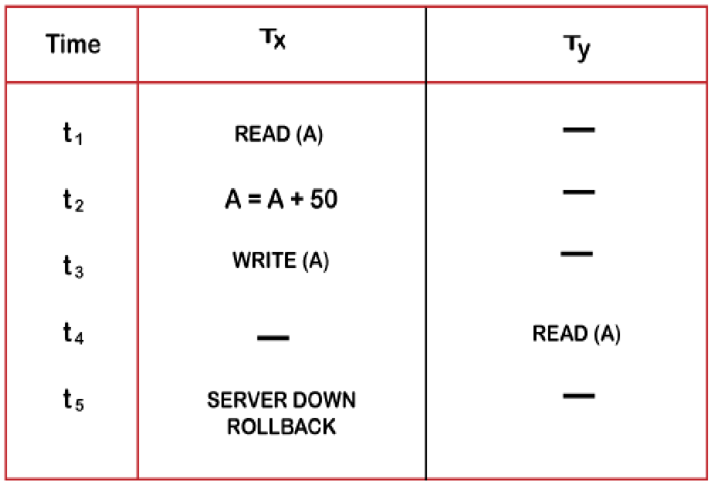
Hence data becomes incorrect, and database sets to inconsistent.

Dirty Read Problems (W-R Conflict)

The dirty read problem occurs *when one transaction updates an item of the database, and somehow the transaction fails, and before the data gets rollback, the updated database item is accessed by another transaction. There comes the Read-Write Conflict between both transactions.*

For example:

Consider two transactions T_X and T_Y in the below diagram performing read/write operations on account A where the available balance in account A is $300:



Time	T_x	T_y
t_1	READ (A)	—
t_2	A = A + 50	—
t_3	WRITE (A)	—
t_4	—	READ (A)
t_5	SERVER DOWN ROLLBACK	—

DIRTY READ PROBLEM

-
- At time t1, transaction T_X reads the value of account A, i.e., $300.
- At time t2, transaction T_X adds $50 to account A that becomes $350.

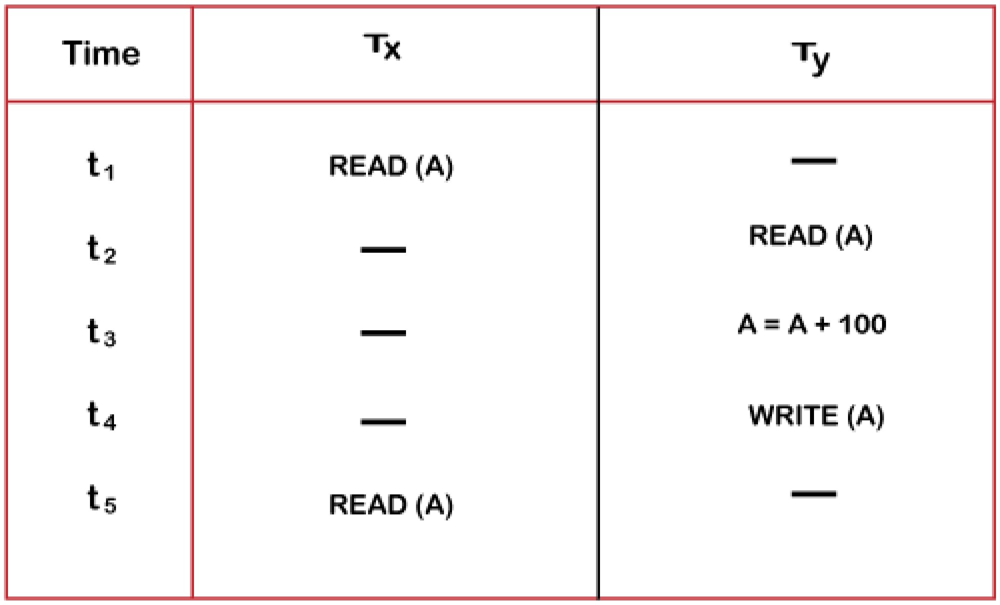
- At time t3, transaction T_X writes the updated value in account A, i.e., $350.
- Then at time t4, transaction T_Y reads account A that will be read as $350.
- Then at time t5, transaction T_X rollbacks due to server problem, and the value changes back to $300 (as initially).
- But the value for account A remains $350 for transaction T_Y as committed, which is the dirty read and therefore known as the Dirty Read Problem.

Unrepeatable Read Problem (W-R Conflict)

Also known as Inconsistent Retrievals Problem that occurs when in a transaction, two different values are read for the same database item.

For example:

Consider two transactions, T_X and T_Y, performing the read/write operations on account A, having an available balance = $300. The diagram is shown below:



Time	T_x	T_y
t_1	READ (A)	—
t_2	—	READ (A)
t_3	—	A = A + 100
t_4	—	WRITE (A)
t_5	READ (A)	—

UNREPEATABLE READ PROBLEM

- At time t1, transaction T_X reads the value from account A, i.e., $300.
- At time t2, transaction T_Y reads the value from account A, i.e., $300.
- At time t3, transaction T_Y updates the value of account A by adding $100 to the available balance, and then it becomes $400.
- At time t4, transaction T_Y writes the updated value, i.e., $400.
- After that, at time t5, transaction T_X reads the available value of account A, and that will be read as $400.

- It means that within the same transaction T_X, it reads two different values of account A, i.e., \$ 300 initially, and after updation made by transaction T_Y, it reads \$400. It is an unrepeatable read and is therefore known as the Unrepeatable read problem.

Thus, in order to maintain consistency in the database and avoid such problems that take place in concurrent execution, management is needed, and that is where the concept of Concurrency Control comes into role.

Concurrency Control

Concurrency Control is the working concept that is required for controlling and managing the concurrent execution of database operations and thus avoiding the inconsistencies in the database. Thus, for maintaining the concurrency of the database, we have the concurrency control protocols.

Concurrency Control Protocols

The concurrency control protocols ensure the *atomicity, consistency, isolation, durability* and *serializability* of the concurrent execution of the database transactions. Therefore, these protocols are categorized as:

- Lock Based Concurrency Control Protocol
- Time Stamp Concurrency Control Protocol
- Validation Based Concurrency Control Protocol

We will understand and discuss each protocol one by one in our next sections.

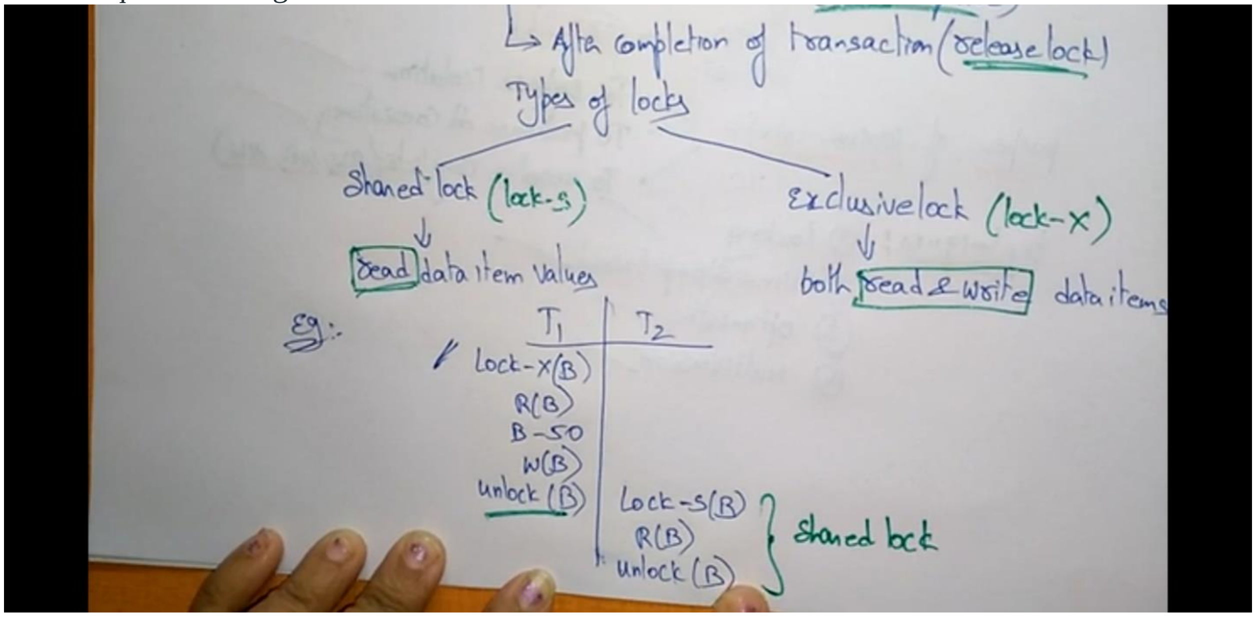
Lock-Based Protocol

In this type of protocol, any transaction cannot read or write data until it acquires an appropriate lock on it. There are two types of lock:

1. Shared Lock (S) :
- Shared lock is also called read lock, used for reading data items only.
- Shared locks support read integrity. They ensure that a record is not in process of being updated during a read-only request.
- Shared locks can also be used to prevent any kind of updates of record.
- It is denoted by Lock-S.
- S-lock is requested using Lock-S instruction.

For **example**, consider a case where initially A=100 and there are two transactions which are reading A. If one of transaction wants to update A, in that case other transaction would be reading wrong value. However, Shared lock prevents it from updating until it has finished reading.

2. Exclusive Lock (X) :
- With the Exclusive Lock, a data item can be read as well as written. Also called write lock.
- An exclusive lock prevents any other locker from obtaining any sort of a lock on the object.
- They can be owned by only one transaction at a time.

- It is denoted as Lock-X.
- X-lock is requested using Lock-X instruction.
- 

↳ After completion of transaction (release lock)

Types of locks

Shared lock (lock-s) → read data item values

Exclusive lock (lock-x) → both read & write data items

Eg:

T1	T2
Lock-X(B)	
R(B)	
B-50	
W(B)	
unlock(B)	Lock-S(B)
	R(B)
	unlock(B)

} Shared lock

Timestamp Ordering Protocol

- The Timestamp Ordering Protocol is used to order the transactions based on their Timestamps. The order of transaction is nothing but the ascending order of the transaction creation.
- The priority of the older transaction is higher that's why it executes first. To determine the timestamp of the transaction, this protocol uses system time or logical counter.
- The lock-based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But Timestamp based protocols start working as soon as a transaction is created.
- Let's assume there are two transactions T1 and T2. Suppose the transaction T1 has entered the system at 007 times and transaction T2 has entered the system at 009 times. T1 has the higher priority, so it executes first as it is entered the system first.
- The timestamp ordering protocol also maintains the timestamp of last 'read' and 'write' operation on a data.

Basic Timestamp ordering protocol works as follows:

1. Check the following condition whenever a transaction Ti issues a **Read (X)** operation:

- If W_TS(X) >TS(Ti) then the operation is rejected.
- If W_TS(X) <= TS(Ti) then the operation is executed.

- Timestamps of all the data items are updated.

Where,

TS(TI) denotes the timestamp of the transaction Ti.

R_TS(X) denotes the Read time-stamp of data-item X.

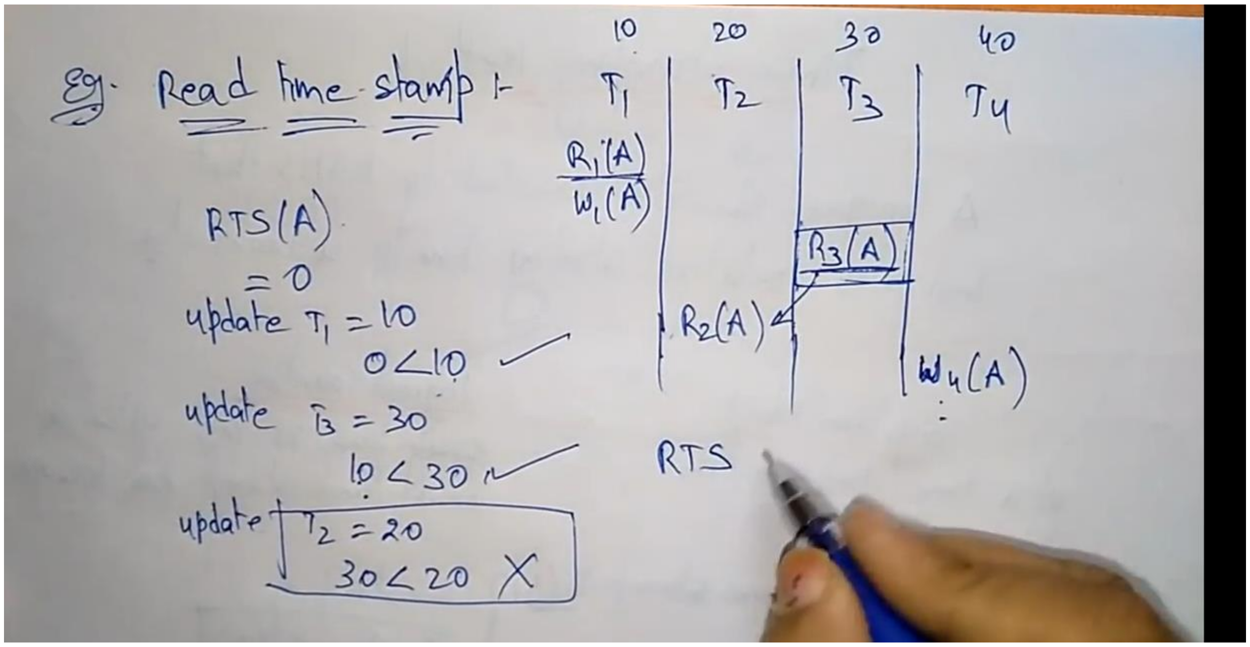
W_TS(X) denotes the Write time-stamp of data-item X.

9.00	T1→	Time Stamp Ts(1)
9.10	T2-→	Time Stamp Ts(2)

Time stamp of Ts1< Ts(2)
9.00<9.10

Read time-stamp

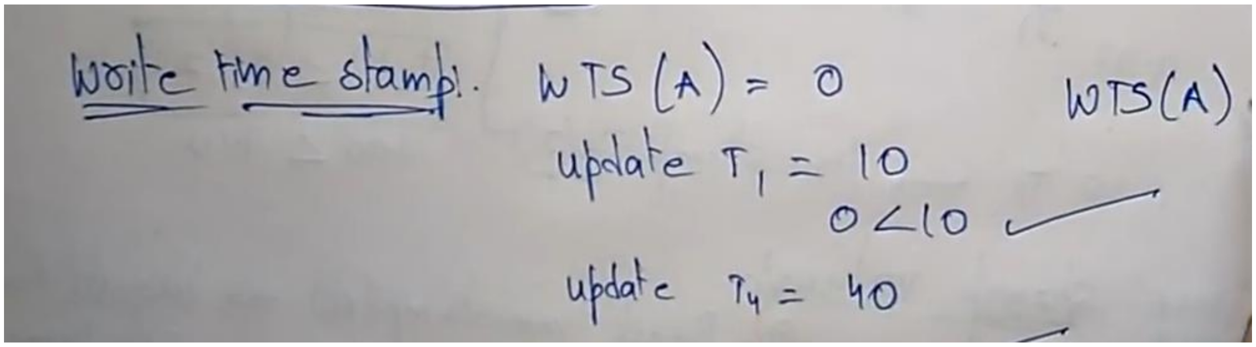
if Ts(T1) < W(A)	T1 is an older transaction then the last transaction that write the value of A{Request fails}
if Ts(T1) ≥W(A)	T1 allowed to read updated value of A



Write time Stamp:

2. Check the following condition whenever a transaction Ti issues a **Write(X)** operation:

- If TS(Ti) < R_TS(X) then the operation is rejected.
- If TS(Ti) < W_TS(X) then the operation is rejected and Ti is rolled back otherwise the operation is executed.



Validation Based Protocol: Validation phase is also known as optimistic concurrency control technique. In the validation based protocol, the transaction is executed in the following three phases:

1. **Read phase:** In this phase, the transaction T is read and executed. It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database.
2. **Validation phase:** In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability.
3. **Write phase:** If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back.

Here each phase has the following different timestamps:

Start(Ti): It contains the time when Ti started its execution.

Validation (T_i): It contains the time when Ti finishes its read phase and starts its validation phase.

Finish(Ti): It contains the time when Ti finishes its write phase.

- This protocol is used to determine the time stamp for the transaction for serialization using the time stamp of the validation phase, as it is the actual phase which determines if the transaction will commit or rollback.
- Hence TS(T) = validation(T).
- The serializability is determined during the validation process. It can't be decided in advance.
- While executing the transaction, it ensures a greater degree of concurrency and also less number of conflicts.
- Thus it contains transactions which have less number of rollbacks.

UNIT-5

Data Mining

Data mining is one of the most useful techniques that help entrepreneurs, researchers, and individuals to extract valuable information from huge sets of data. Data mining is also called ***Knowledge Discovery in Database (KDD)***. The knowledge discovery process includes Data cleaning, Data integration, Data selection, Data transformation, Data mining, Pattern evaluation, and Knowledge presentation.

Our Data mining tutorial includes all topics of Data mining such as applications, Data mining vs Machine learning, Data mining tools, Social Media Data mining, Data mining techniques, Clustering in data mining, Challenges in Data mining, etc

Data Mining

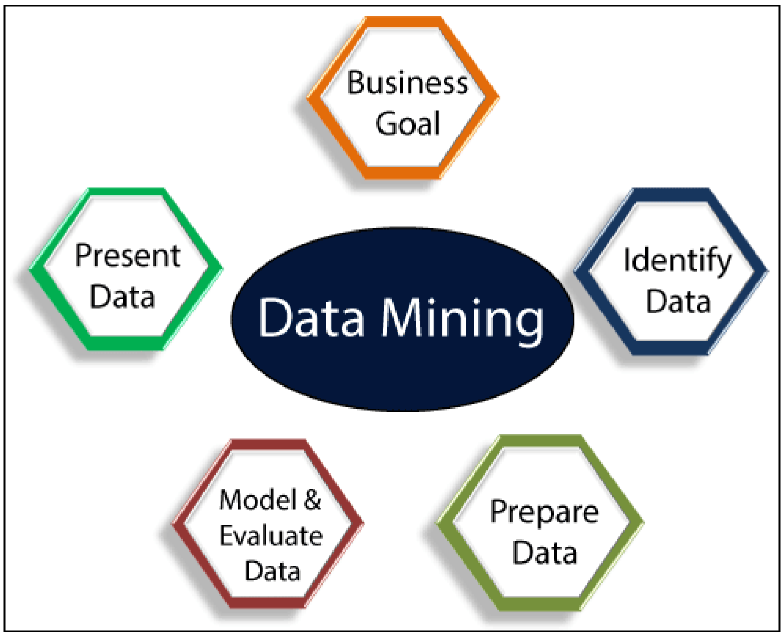
The process of extracting information to identify patterns, trends, and useful data that would allow the business to take the data-driven decision from huge sets of data is called Data Mining.

Data Mining is the process of investigating hidden patterns of information to various perspectives for categorization into useful data, which is collected and assembled in particular areas such as data warehouses, efficient analysis, data mining algorithm, helping decision making and other data requirement to eventually cost-cutting and generating revenue.

Data mining is the act of automatically searching for large stores of information to find trends and patterns that go beyond simple analysis procedures. Data mining utilizes complex mathematical algorithms for data segments and evaluates the probability of future events. Data Mining is also called Knowledge Discovery of Data (KDD).

Data Mining is a process used by organizations to extract specific data from huge databases to solve business problems. It primarily turns raw data into useful information.

Data Mining is similar to Data Science carried out by a person, in a specific situation, on a particular data set, with an objective. This process includes various types of services such as text mining, web mining, audio and video mining, pictorial data mining, and social media mining. It is done through software that is simple or highly specific. By outsourcing data mining, all the work can be done faster with low operation costs. Specialized firms can also use new technologies to collect data that is impossible to locate manually. There are tonnes of information available on various platforms, but very little knowledge is accessible. The biggest challenge is to analyze the data to extract important information that can be used to solve a problem or for company development. There are many powerful instruments and techniques available to mine data and find better insight from it.



Types of Data Mining

Data mining can be performed on the following types of data:

Relational Database:

A relational database is a collection of multiple data sets formally organized by tables, records, and columns from which data can be accessed in various ways without having to recognize the database tables. Tables convey and share information, which facilitates data searchability, reporting, and organization.

Data warehouses:

A Data Warehouse is the technology that collects the data from various sources within the organization to provide meaningful business insights. The huge amount of data comes from multiple places such as Marketing and Finance. The extracted data is utilized for analytical purposes and helps in decision- making for a business organization. The data warehouse is designed for the analysis of data rather than transaction processing.

Data Repositories:

The Data Repository generally refers to a destination for data storage. However, many IT professionals utilize the term more clearly to refer to a specific kind of setup within an IT structure. For example, a group of databases, where an organization has kept various kinds of information.

Object-Relational Database:

A combination of an object-oriented database model and relational database model is called an object-relational model. It supports Classes, Objects, Inheritance, etc.

One of the primary objectives of the Object-relational data model is to close the gap between the Relational database and the object-oriented model practices frequently utilized in many programming languages, for example, C++, Java, C#, and so on.

Transactional Database:

A transactional database refers to a database management system (DBMS) that has the potential to undo a database transaction if it is not performed appropriately. Even though this was a unique capability a very long while back, today, most of the relational database systems support transactional database activities.

Advantages of Data Mining

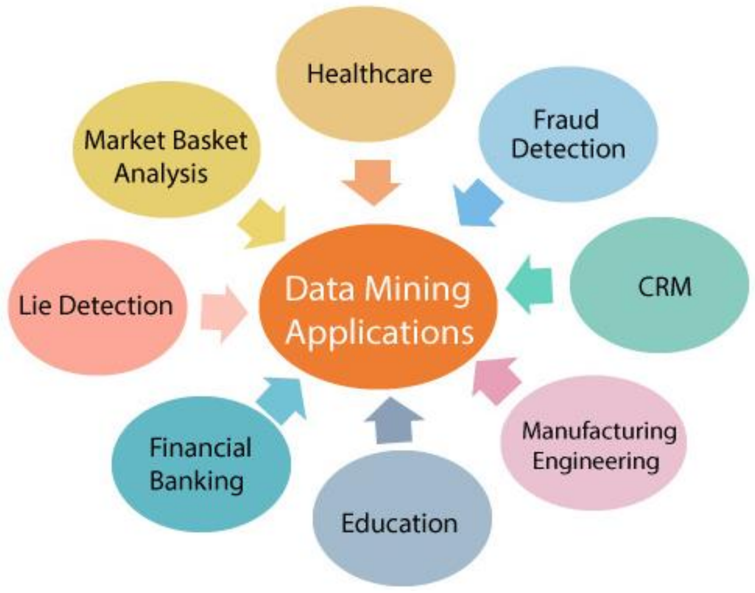
- The Data Mining technique enables organizations to obtain knowledge-based data.
- Data mining enables organizations to make lucrative modifications in operation and production.
- Compared with other statistical data applications, data mining is a cost-efficient.
- Data Mining helps the decision-making process of an organization.
- It Facilitates the automated discovery of hidden patterns as well as the prediction of trends and behaviors.
- It can be induced in the new system as well as the existing platforms.
- It is a quick process that makes it easy for new users to analyze enormous amounts of data in a short time.

Disadvantages of Data Mining

- There is a probability that the organizations may sell useful data of customers to other organizations for money. As per the report, American Express has sold credit card purchases of their customers to other organizations.
- Many data mining analytics software is difficult to operate and needs advance training to work on.
- Different data mining instruments operate in distinct ways due to the different algorithms used in their design. Therefore, the selection of the right data mining tools is a very challenging task.
- The data mining techniques are not precise, so that it may lead to severe consequences in certain conditions.

Data Mining Applications

Data Mining is primarily used by organizations with intense consumer demands- Retail, Communication, Financial, marketing company, determine price, consumer preferences, product positioning, and impact on sales, customer satisfaction, and corporate profits. Data mining enables a retailer to use point-of-sale records of customer purchases to develop products and promotions that help the organization to attract the customer.



These are the following areas where data mining is widely used:

Data Mining in Healthcare:

Data mining in healthcare has excellent potential to improve the health system. It uses data and analytics for better insights and to identify best practices that will enhance health care services and reduce costs. Analysts use data mining approaches such as Machine learning, Multi-dimensional database, Data visualization, Soft computing, and statistics. Data Mining can be used to forecast patients in each category. The procedures ensure that the patients get intensive care at the right place and at the right time. Data mining also enables healthcare insurers to recognize fraud and abuse.

Data Mining in Market Basket Analysis:

Market basket analysis is a modeling method based on a hypothesis. If you buy a specific group of products, then you are more likely to buy another group of products. This technique may enable the retailer to understand the purchase behavior of a buyer. This data may assist the retailer in understanding the requirements of the buyer and altering the store's layout accordingly. Using a different analytical comparison of results between various stores, between customers in different demographic groups can be done.

Data mining in Education:

Education data mining is a newly emerging field, concerned with developing techniques that explore knowledge from the data generated from educational Environments. EDM objectives are recognized as affirming student's future learning behavior, studying the impact of educational support, and promoting learning science. An organization can use data mining to make precise decisions and also to predict the results of the student. With the results, the institution can concentrate on what to teach and how to teach.

Data Mining in Manufacturing Engineering:

Knowledge is the best asset possessed by a manufacturing company. Data mining tools can be beneficial to find patterns in a complex manufacturing process. Data mining can be used in system-level designing to obtain the relationships between product architecture, product portfolio, and data needs of the customers. It can also be used to forecast the product development period, cost, and expectations among the other tasks.

Data Mining in CRM (Customer Relationship Management):

Customer Relationship Management (CRM) is all about obtaining and holding Customers, also enhancing customer loyalty and implementing customer-oriented strategies. To get a decent relationship with the customer, a business organization needs to collect data and analyze the data. With data mining technologies, the collected data can be used for analytics.

Data Mining in Fraud detection:

Billions of dollars are lost to the action of frauds. Traditional methods of fraud detection are a little bit time consuming and sophisticated. Data mining provides meaningful patterns and turning data into information. An ideal fraud detection system should protect the data of all the users. Supervised methods consist of a collection of sample records, and these records are classified as fraudulent or non-fraudulent. A model is constructed using this data, and the technique is made to identify whether the document is fraudulent or not.

Data Mining in Lie Detection:

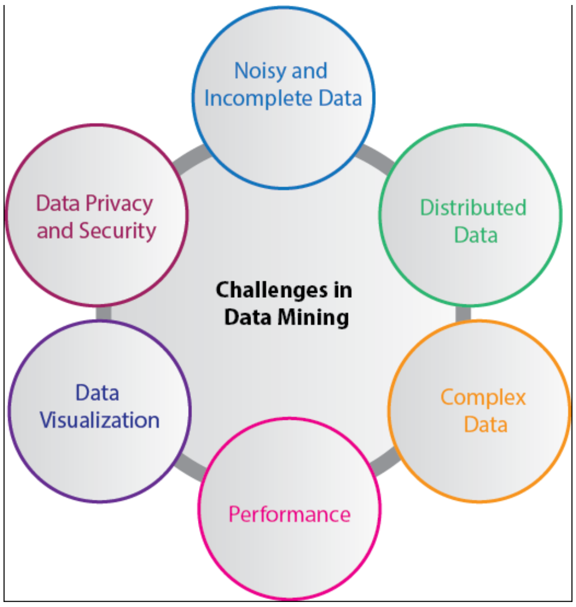
Apprehending a criminal is not a big deal, but bringing out the truth from him is a very challenging task. Law enforcement may use data mining techniques to investigate offenses, monitor suspected terrorist communications, etc. This technique includes text mining also, and it seeks meaningful patterns in data, which is usually unstructured text. The information collected from the previous investigations is compared, and a model for lie detection is constructed.

Data Mining Financial Banking:

The Digitalization of the banking system is supposed to generate an enormous amount of data with every new transaction. The data mining technique can help bankers by solving business-related problems in banking and finance by identifying trends, casualties, and correlations in business information and market costs that are not instantly evident to managers or executives because the data volume is too large or are produced too rapidly on the screen by experts. The manager may find these data for better targeting, acquiring, retaining, segmenting, and maintain a profitable customer.

Challenges of Implementation in Data mining

Although data mining is very powerful, it faces many challenges during its execution. Various challenges could be related to performance, data, methods, and techniques, etc. The process of data mining becomes effective when the challenges or problems are correctly recognized and adequately resolved.


Noisy and Incomplete Data
Distributed Data
Complex Data
Performance
Data Visualization
Data Privacy and Security
Challenges in Data Mining

Incomplete and noisy data:

The process of extracting useful data from large volumes of data is data mining. The data in the real-world is heterogeneous, incomplete, and noisy. Data in huge quantities will usually be inaccurate or unreliable. These problems may occur due to data measuring instrument or because of human errors. Suppose a retail chain collects phone numbers of customers who spend more than $ 500, and the accounting employees put the information into their system. The person may make a digit mistake when entering the phone number, which results in incorrect data. Even some customers may not be willing to disclose their phone numbers, which results in incomplete data. The data could get changed due to human or system error. All these consequences (noisy and incomplete data)makes data mining challenging.

Data Distribution:

Real-worlds data is usually stored on various platforms in a distributed computing environment. It might be in a database, individual systems, or even on the internet. Practically, It is a quite tough task to make all the data to a centralized data repository mainly due to organizational and technical concerns. For example, various regional offices may have their servers to store their data. It is not feasible to store, all the data from all the offices on a central server. Therefore, data mining requires the development of tools and algorithms that allow the mining of distributed data.

Complex Data:

Real-world data is heterogeneous, and it could be multimedia data, including audio and video, images, complex data, spatial data, time series, and so on. Managing these various types of data and extracting useful information is a tough task. Most of the time, new technologies, new tools, and methodologies would have to be refined to obtain specific information.

Performance:

The data mining system's performance relies primarily on the efficiency of algorithms and techniques used. If the designed algorithm and techniques are not up to the mark, then the efficiency of the data mining process will be affected adversely.

Data Privacy and Security:

Data mining usually leads to serious issues in terms of data security, governance, and privacy. For example, if a retailer analyzes the details of the purchased items, then it reveals data about buying habits and preferences of the customers without their permission.

Data Visualization:

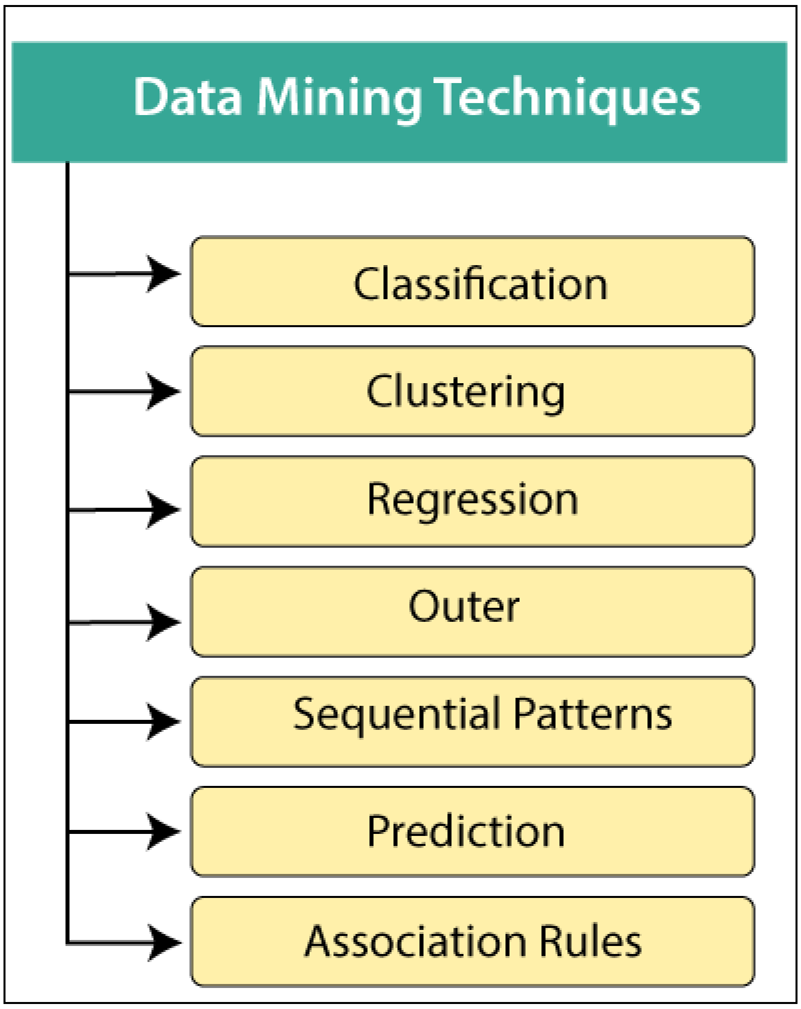
In data mining, data visualization is a very important process because it is the primary method that shows the output to the user in a presentable way. The extracted data should convey the exact meaning of what it intends to express. But many times, representing the information to the end-user in a precise and easy way is difficult. The input data and the output information being complicated, very efficient, and successful data visualization processes need to be implemented to make it successful.

Data Mining Techniques

Data mining includes the utilization of refined data analysis tools to find previously unknown, valid patterns and relationships in huge data sets. These tools can incorporate statistical models, machine learning techniques, and mathematical algorithms, such as neural networks or decision trees. Thus, data mining incorporates analysis and prediction.

Depending on various methods and technologies from the intersection of machine learning, database management, and statistics, professionals in data mining have devoted their careers to better understanding how to process and make conclusions from the huge amount of data, but what are the methods they use to make it happen?

In recent data mining projects, various major data mining techniques have been developed and used, including association, classification, clustering, prediction, sequential patterns, and regression.



1. Classification:

This technique is used to obtain important and relevant information about data and metadata. This data mining technique helps to classify data in different classes.

Data mining techniques can be classified by different criteria, as follows:

i. **Classification of Data mining frameworks as per the type of data sources mined:** This classification is as per the type of data handled. For example, multimedia, spatial data, text data, time-series data, World Wide Web, and so on..

ii. **Classification of data mining frameworks as per the database involved:** This classification based on the data model involved. For example. Object-oriented database, transactional database, relational database, and so on..

iii. **Classification of data mining frameworks as per the kind of knowledge discovered:** This classification depends on the types of knowledge discovered or data mining functionalities. For example, discrimination, classification, clustering, characterization, etc.

some frameworks tend to be extensive frameworks offering a few data mining functionalities together..

iv. **Classification of data mining frameworks according to data mining techniques used:** This classification is as per the data analysis approach utilized, such as neural networks, machine learning, genetic algorithms, visualization, statistics, data warehouse-oriented or database-oriented, etc.
The classification can also take into account, the level of user interaction involved in the data mining procedure, such as query-driven systems, autonomous systems, or interactive exploratory systems.

2. Clustering:

Clustering is a division of information into groups of connected objects. Describing the data by a few clusters mainly loses certain confine details, but accomplishes improvement. It models data by its clusters. Data modeling puts clustering from a historical point of view rooted in statistics, mathematics, and numerical analysis. From a machine learning point of view, clusters relate to hidden patterns, the search for clusters is unsupervised learning, and the subsequent framework represents a data concept. From a practical point of view, clustering plays an extraordinary job in data mining applications. For example, scientific data exploration, text mining, information retrieval, spatial database applications, CRM, Web analysis, computational biology, medical diagnostics, and much more.

In other words, we can say that Clustering analysis is a data mining technique to identify similar data. This technique helps to recognize the differences and similarities between the data. Clustering is very similar to the classification, but it involves grouping chunks of data together based on their similarities.

3. Regression:

Regression analysis is the data mining process is used to identify and analyze the relationship between variables because of the presence of the other factor. It is used to define the probability of the specific variable. Regression, primarily a form of planning and modeling. For example, we might use it to project certain costs, depending on other factors such as availability, consumer demand, and competition. Primarily it gives the exact relationship between two or more variables in the given data set.

4. Association Rules:

This data mining technique helps to discover a link between two or more items. It finds a hidden pattern in the data set.

Association rules are if-then statements that support to show the probability of interactions between data items within large data sets in different types of databases. Association rule mining has several applications and is commonly used to help sales correlations in data or medical data sets.

The way the algorithm works is that you have various data, For example, a list of grocery items that you have been buying for the last six months. It calculates a percentage of items being purchased together.

These are three major measurements technique:

- **Lift:**
  This measurement technique measures the accuracy of the confidence over how often item B is purchased.
  (Confidence) / (item B)/ (Entire dataset)
- **Support:**
  This measurement technique measures how often multiple items are purchased and compared it to the overall dataset.
  (Item A + Item B) / (Entire dataset)
- **Confidence:**
  This measurement technique measures how often item B is purchased when item A is purchased as well.
  (Item A + Item B)/ (Item A)

5. Outer detection:

This type of data mining technique relates to the observation of data items in the data set, which do not match an expected pattern or expected behavior. This technique may be used in various domains like intrusion, detection, fraud detection, etc. It is also known as Outlier Analysis or Outilier mining. The outlier is a data point that diverges too much from the rest of the dataset. The majority of the real-world datasets have an outlier. Outlier detection plays a significant role in the data mining field. Outlier detection is valuable in numerous fields like network interruption identification, credit or debit card fraud detection, detecting outlying in wireless sensor network data, etc.

6. Sequential Patterns:

The sequential pattern is a data mining technique specialized for **evaluating sequential data** to discover sequential patterns. It comprises of finding interesting subsequences in a set of sequences, where the stake of a sequence can be measured in terms of different criteria like length, occurrence frequency, etc.

In other words, this technique of data mining helps to discover or recognize similar patterns in transaction data over some time.

7. Prediction:

Prediction used a combination of other data mining techniques such as trends, clustering, classification, etc. It analyzes past events or instances in the right sequence to predict a future event.

Data Warehouse?

A Data Warehouse (DW) is a relational database that is designed for query and analysis rather than transaction processing. It includes historical data derived from transaction data from single and multiple sources.

A Data Warehouse provides integrated, enterprise-wide, historical data and focuses on providing support for decision-makers for data modeling and analysis.

A Data Warehouse is a group of data specific to the entire organization, not only to a particular group of users.

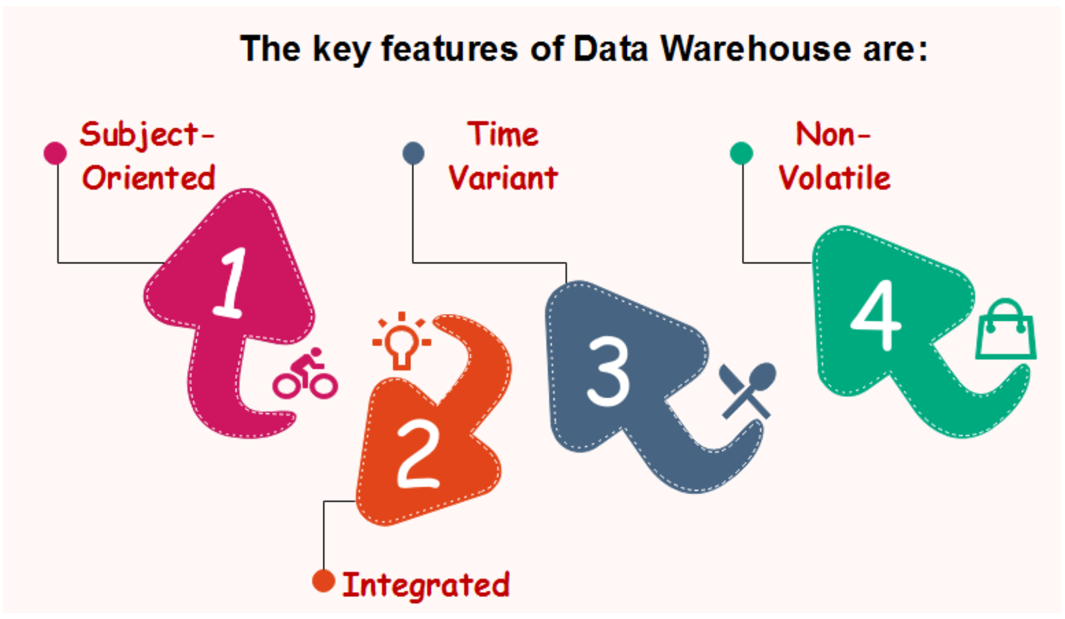
It is not used for daily operations and transaction processing but used for making decisions.

A Data Warehouse can be viewed as a data system with the following attributes:

- It is a database designed for investigative tasks, using data from various applications.
- It supports a relatively small number of clients with relatively long interactions.
- It includes current and historical data to provide a historical perspective of information.
- Its usage is read-intensive.
- It contains a few large tables.

"Data Warehouse is a subject-oriented, integrated, and time-variant store of information in support of management's decisions."

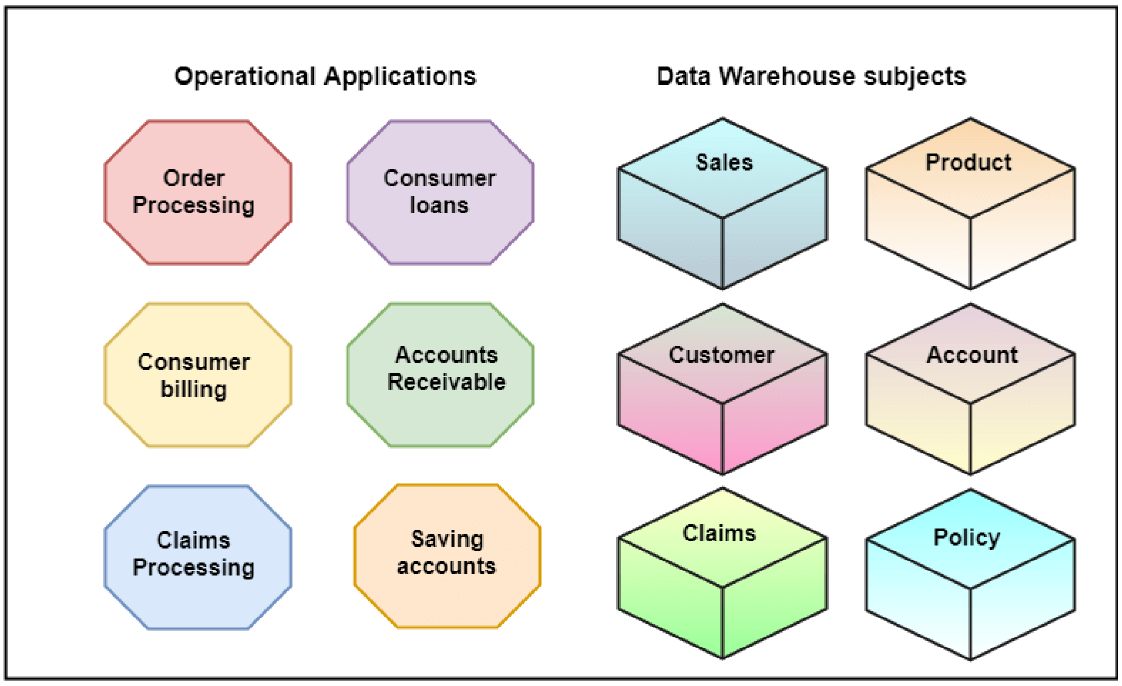
Characteristics of Data Warehouse



Subject-Oriented

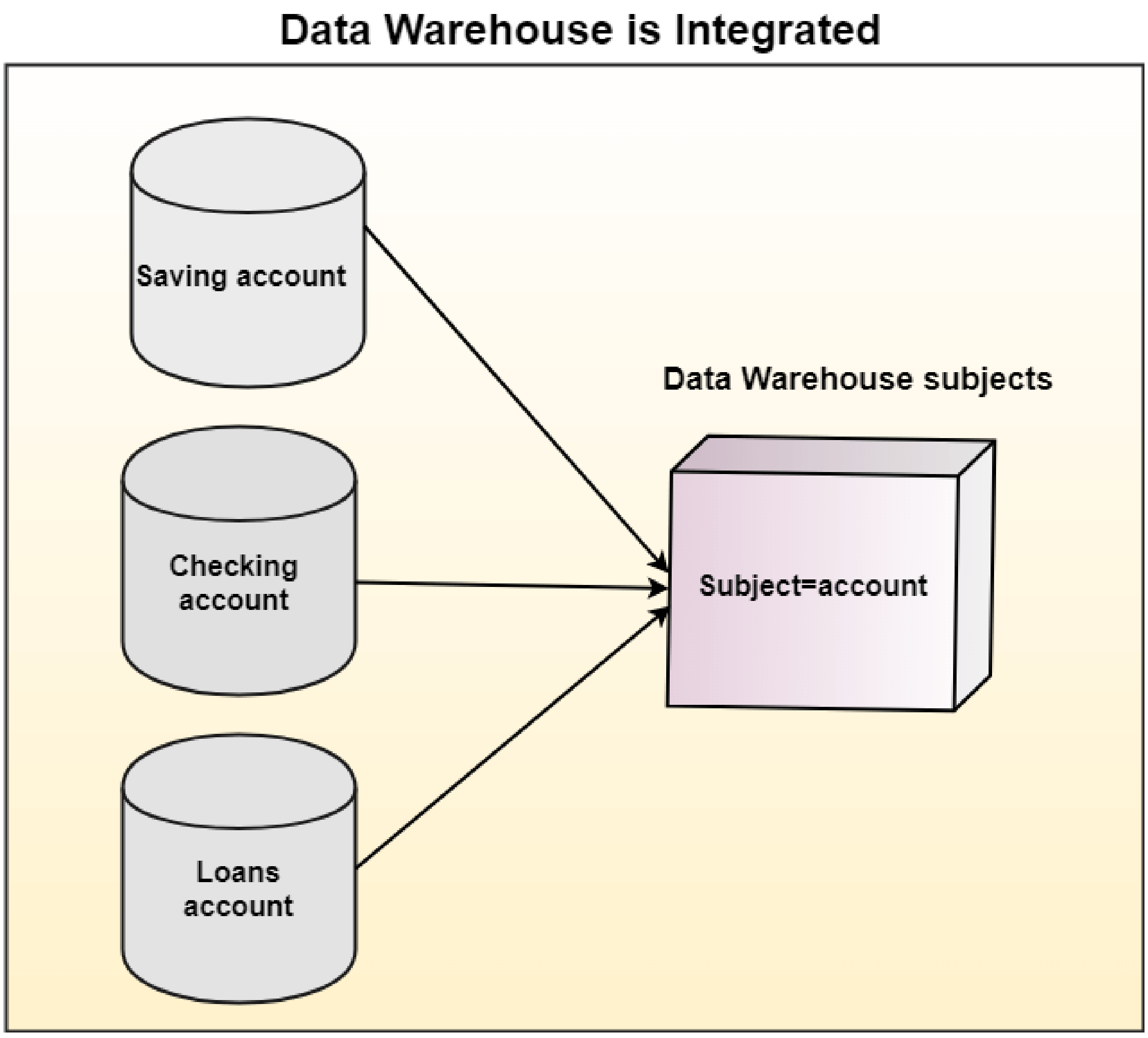
A data warehouse target on the modeling and analysis of data for decision-makers. Therefore, data warehouses typically provide a concise and straightforward view around a particular subject, such as customer, product, or sales, instead of the global organization's ongoing operations. This is done by excluding data that are not useful concerning the subject and including all data needed by the users to understand the subject.

Data Warehouse is Subject-Oriented



Integrated

A data warehouse integrates various heterogeneous data sources like RDBMS, flat files, and online transaction records. It requires performing data cleaning and integration during data warehousing to ensure consistency in naming conventions, attributes types, etc., among different data sources.

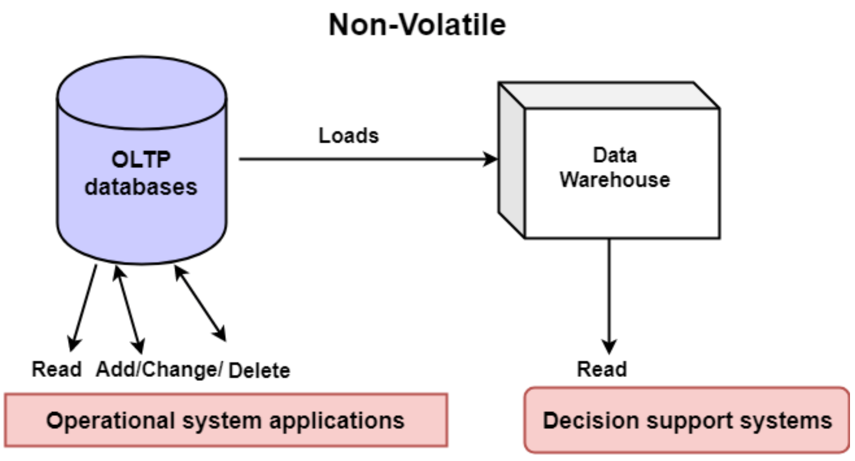


Time-Variant

Historical information is kept in a data warehouse. For example, one can retrieve files from 3 months, 6 months, 12 months, or even previous data from a data warehouse. These variations with a transactions system, where often only the most current file is kept.

Non-Volatile

The data warehouse is a physically separate data storage, which is transformed from the source operational RDBMS. The operational updates of data do not occur in the data warehouse, i.e., update, insert, and delete operations are not performed. It usually requires only two procedures in data accessing: Initial loading of data and access to data. Therefore, the DW does not require transaction processing, recovery, and concurrency capabilities, which allows for substantial speedup of data retrieval. Non-Volatile defines that once entered into the warehouse, and data should not change.



History of Data Warehouse

The idea of data warehousing came to the late 1980's when IBM researchers Barry Devlin and Paul Murphy established the "Business Data Warehouse."

In essence, the data warehousing idea was planned to support an architectural model for the flow of information from the operational system to decisional support environments. The concept attempt to address the various problems associated with the flow, mainly the high costs associated with it.

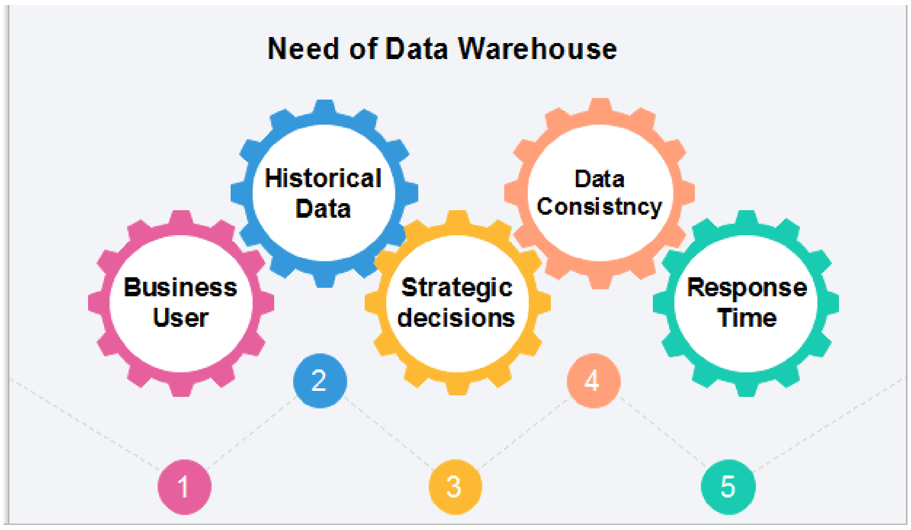
In the absence of data warehousing architecture, a vast amount of space was required to support multiple decision support environments. In large corporations, it was ordinary for various decision support environments to operate independently.

Goals of Data Warehousing

- To help reporting as well as analysis
- Maintain the organization's historical information
- Be the foundation for decision making.

Need for Data Warehouse

Data Warehouse is needed for the following reasons:



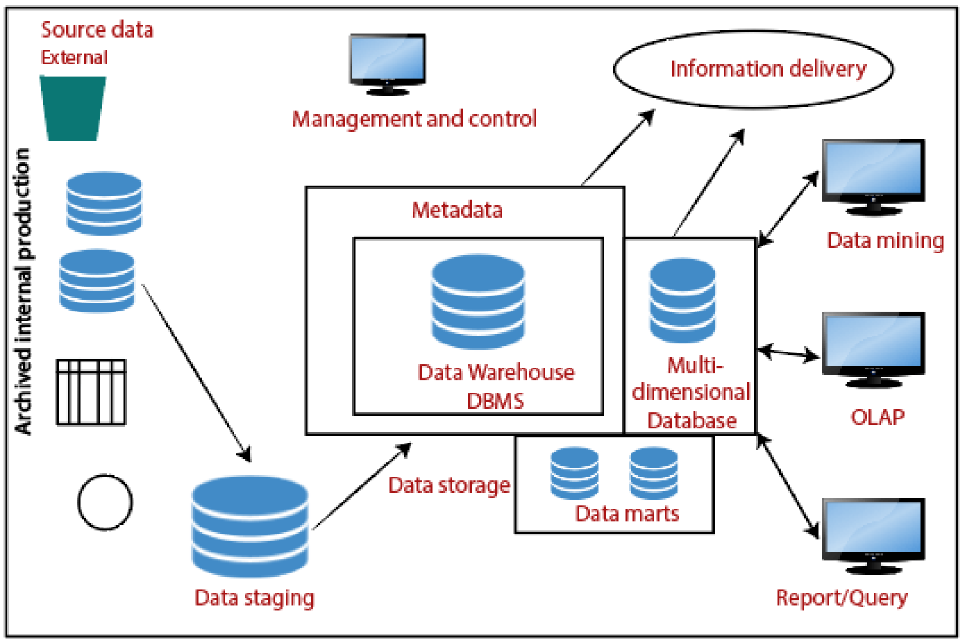
1. 1) **Business User:** Business users require a data warehouse to view summarized data from the past. Since these people are non-technical, the data may be presented to them in an elementary form.
2. 2) **Store historical data:** Data Warehouse is required to store the time variable data from the past. This input is made to be used for various purposes.
3. 3) **Make strategic decisions:** Some strategies may be depending upon the data in the data warehouse. So, data warehouse contributes to making strategic decisions.
4. 4) **For data consistency and quality:** Bringing the data from different sources at a commonplace, the user can effectively undertake to bring the uniformity and consistency in data.
5. 5) **High response time:** Data warehouse has to be ready for somewhat unexpected loads and types of queries, which demands a significant degree of flexibility and quick response time.

Benefits of Data Warehouse

1. Understand business trends and make better forecasting decisions.
2. Data Warehouses are designed to perform well enormous amounts of data.
3. The structure of data warehouses is more accessible for end-users to navigate, understand, and query.
4. Queries that would be complex in many normalized databases could be easier to build and maintain in data warehouses.
5. Data warehousing is an efficient method to manage demand for lots of information from lots of users.
6. Data warehousing provide the capabilities to analyze a large amount of historical data.

Components or Building Blocks of Data Warehouse

Architecture is the proper arrangement of the elements. We build a data warehouse with software and hardware components. To suit the requirements of our organizations, we arrange these building we may want to boost up another part with extra tools and services. All of these depends on our circumstances.



Components or Building Blocks of Data Warehouse

The figure shows the essential elements of a typical warehouse. We see the Source Data component shows on the left. The Data staging element serves as the next building block. In the middle, we see the Data Storage component that handles the data warehouses data. This element not only stores and manages the data; it also keeps track of data using the metadata repository. The Information Delivery component shows on the right consists of all the different ways of making the information from the data warehouses available to the users.

Source Data Component

Source data coming into the data warehouses may be grouped into four broad categories:

Production Data: This type of data comes from the different operating systems of the enterprise. Based on the data requirements in the data warehouse, we choose segments of the data from the various operational modes.

Internal Data: In each organization, the client keeps their "**private**" spreadsheets, reports, customer profiles, and sometimes even department databases. This is the internal data, part of which could be useful in a data warehouse.

Archived Data: Operational systems are mainly intended to run the current business. In every operational system, we periodically take the old data and store it in achieved files.

External Data: Most executives depend on information from external sources for a large percentage of the information they use. They use statistics associating to their industry produced by the external department.

Data Staging Component

After we have been extracted data from various operational systems and external sources, we have to prepare the files for storing in the data warehouse. The extracted data coming from several different sources need to be changed, converted, and made ready in a format that is relevant to be saved for querying and analysis.

We will now discuss the three primary functions that take place in the staging area.

1) Data Extraction: This method has to deal with numerous data sources. We have to employ the appropriate techniques for each data source.

2) Data Transformation: As we know, data for a data warehouse comes from many different sources. If data extraction for a data warehouse posture big challenges, data transformation present even significant challenges. We perform several individual tasks as part of data transformation.

First, we clean the data extracted from each source. Cleaning may be the correction of misspellings or may deal with providing default values for missing data elements, or elimination of duplicates when we bring in the same data from various source systems.

Standardization of data components forms a large part of data transformation. Data transformation contains many forms of combining pieces of data from different sources. We combine data from single source record or related data parts from many source records.

On the other hand, data transformation also contains purging source data that is not useful and separating outsource records into new combinations. Sorting and merging of data take place on a large scale in the data staging area. When the data transformation function ends, we have a collection of integrated data that is cleaned, standardized, and summarized.

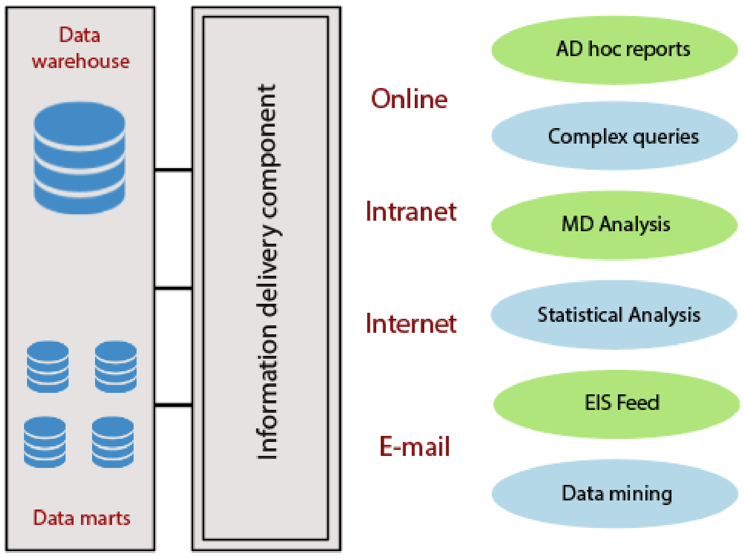
3) Data Loading: Two distinct categories of tasks form data loading functions. When we complete the structure and construction of the data warehouse and go live for the first time, we do the initial loading of the information into the data warehouse storage. The initial load moves high volumes of data using up a substantial amount of time.

Data Storage Components

Data storage for the data warehousing is a split repository. The data repositories for the operational systems generally include only the current data. Also, these data repositories include the data structured in highly normalized for fast and efficient processing.

Information Delivery Component

The information delivery element is used to enable the process of subscribing for data warehouse files and having it transferred to one or more destinations according to some customer-specified scheduling algorithm.



Information delivery component

Metadata Component

Metadata in a data warehouse is equal to the data dictionary or the data catalog in a database management system. In the data dictionary, we keep the data about the logical data structures, the data about the records and addresses, the information about the indexes, and so on.

Data Marts

It includes a subset of corporate-wide data that is of value to a specific group of users. The scope is confined to particular selected subjects. Data in a data warehouse should be a fairly current, but not mainly up to the minute, although development in the data warehouse industry has made standard and incremental data dumps more achievable. Data marts are lower than data warehouses and usually contain organization. The current trends in data warehousing are to developed a data warehouse with several smaller related data marts for particular kinds of queries and reports.

Management and Control Component

The management and control elements coordinate the services and functions within the data warehouse. These components control the data transformation and the data transfer into the data warehouse storage. On the other hand, it moderates the data delivery to the clients. Its work with the database management systems and authorizes data to be correctly saved in the repositories. It monitors the movement of information into the staging method and from there into the data warehouses storage itself.

Database	Data Warehouse
1. It is used for Online Transactional Processing (OLTP) but can be used for other objectives such as Data Warehousing. This records the data from the clients for history.	1. It is used for Online Analytical Processing (OLAP). This reads the historical information for the customers for business decisions.
2. The tables and joins are complicated since they are normalized for RDBMS. This is done to reduce redundant files and to save storage space.	2. The tables and joins are accessible since they are de-normalized. This is done to minimize the response time for analytical queries.
3. Data is dynamic	3. Data is largely static
4. **Entity:** Relational modeling procedures are used for RDBMS database design.	4. **Data:** Modeling approach are used for the Data Warehouse design.
5. Optimized for write operations.	5. Optimized for read operations.
6. Performance is low for analysis queries.	6. High performance for analytical queries.
7. The database is the place where the data is taken as a base and managed to get available fast and efficient access.	7. Data Warehouse is the place where the application data is handled for analysis and reporting objectives.

Difference between OLTP and OLAP

OLTP System

OLTP System handle with operational data. Operational data are those data contained in the operation of a particular system. Example, ATM transactions and Bank transactions, etc.

OLAP System

OLAP handle with Historical Data or Archival Data. Historical data are those data that are achieved over a long period. For example, if we collect the last 10 years information about flight reservation, the data can give us much meaningful data such as the trends in the reservation. This may provide useful information like peak time of travel, what kind of people are traveling in various classes (Economy/Business) etc.

The major difference between an OLTP and OLAP system is the amount of data analyzed in a single transaction. Whereas an OLTP manage many concurrent customers and queries touching only an individual record or limited groups of files at a time. An OLAP system must have the capability to operate on millions of files to answer a single query.

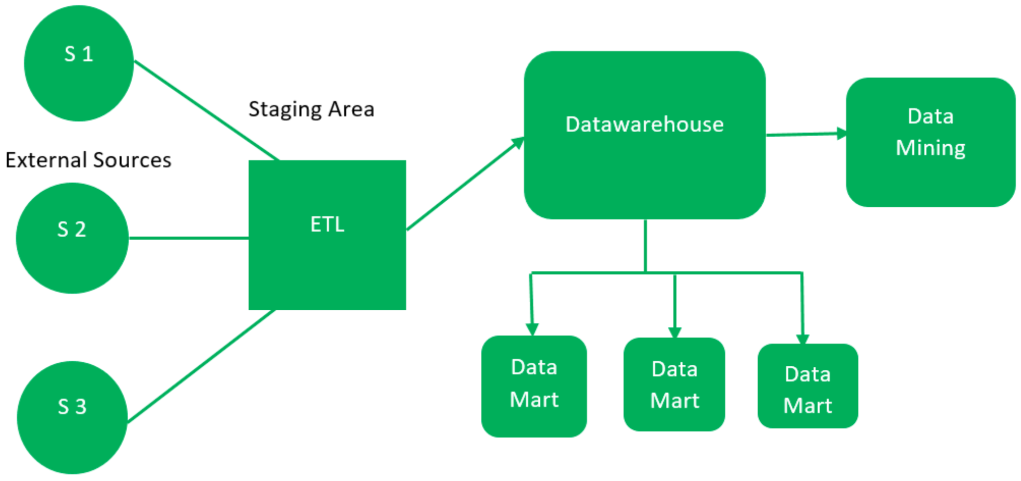
Feature	**OLTP**	**OLAP**
Characteristic	It is a system which is used to manage operational Data.	It is a system which is used to manage informational Data.
Users	Clerks, clients, and information technology professionals.	Knowledge workers, including managers, executives, and analysts.
System orientation	OLTP system is a customer-oriented, transaction, and query processing are done by clerks, clients, and information technology professionals.	OLAP system is market-oriented, knowledge workers including managers, do data analysts executive and analysts.
Data contents	OLTP system manages current data that too detailed and are used for decision making.	OLAP system manages a large amount of historical data, provides facilitates for summarization and aggregation, and stores and manages data at different levels of granularity. This information makes the data more comfortable to use in informed decision making.
Database Size	100 MB-GB	100 GB-TB

Database design	OLTP system usually uses an entity-relationship (ER) data model and application-oriented database design.	OLAP system typically uses either a star or snowflake model and subject-oriented database design.
View	OLTP system focuses primarily on the current data within an enterprise or department, without referring to historical information or data in different organizations.	OLAP system often spans multiple versions of a database schema, due to the evolutionary process of an organization. OLAP systems also deal with data that originates from various organizations, integrating information from many data stores.
Volume of data	Not very large	Because of their large volume, OLAP data are stored on multiple storage media.
Access patterns	The access patterns of an OLTP system subsist mainly of short, atomic transactions. Such a system requires concurrency control and recovery techniques.	Accesses to OLAP systems are mostly read-only methods because of these data warehouses stores historical data.
Access mode	Read/write	Mostly write
Insert and Updates	Short and fast inserts and updates proposed by end-users.	Periodic long-running batch jobs refresh the data.
Number of records accessed	Tens	Millions
Normalization	Fully Normalized	Partially Normalized
Processing Speed	Very Fast	It depends on the amount of files contained, batch data refresh, and complex query may take many hours, and query speed can be upgraded by creating indexes.

Data Warehouse Architecture

A **data-warehouse** is a heterogeneous collection of different data sources organised under a unified schema. There are 2 approaches for constructing data-warehouse: Top-down approach and Bottom-up approach are explained as below.

1. Top-down approach:



The essential components are discussed below:

1. **External Sources –**
   External source is a source from where data is collected irrespective of the type of data. Data can be structured, semi structured and unstructured as well.

2. **Stage Area –**
   Since the data, extracted from the external sources does not follow a particular format, so there is a need to validate this data to load into datawarehouse. For this purpose, it is recommended to use **ETL** tool.
   - **E(Extracted):** Data is extracted from External data source.
   - **T(Transform):** Data is transformed into the standard format.
   - **L(Load):** Data is loaded into datawarehouse after transforming it into the standard format.

3. **Data-warehouse –**
   After cleansing of data, it is stored in the datawarehouse as central repository. It actually stores the meta data and the actual data gets stored in the data marts. **Note** that datawarehouse stores the data in its purest form in this top-down approach.

4. **Data Marts –**
   Data mart is also a part of storage component. It stores the information of a particular function of an organisation which is handled by single authority. There can be as many number of data marts in an organisation depending upon the functions. We can also say that data mart contains subset of the data stored in

datawarehouse.

5. **Data Mining –**
   The practice of analysing the big data present in datawarehouse is data mining. It is used to find the hidden patterns that are present in the database or in datawarehouse with the help of algorithm of data mining.
   This approach is defined by **Inmon** as – datawarehouse as a central repository for the complete organisation and data marts are created from it after the complete datawarehouse has been created.

Advantages of Top-Down Approach –

1. Since the data marts are created from the datawarehouse, provides consistent dimensional view of data marts.

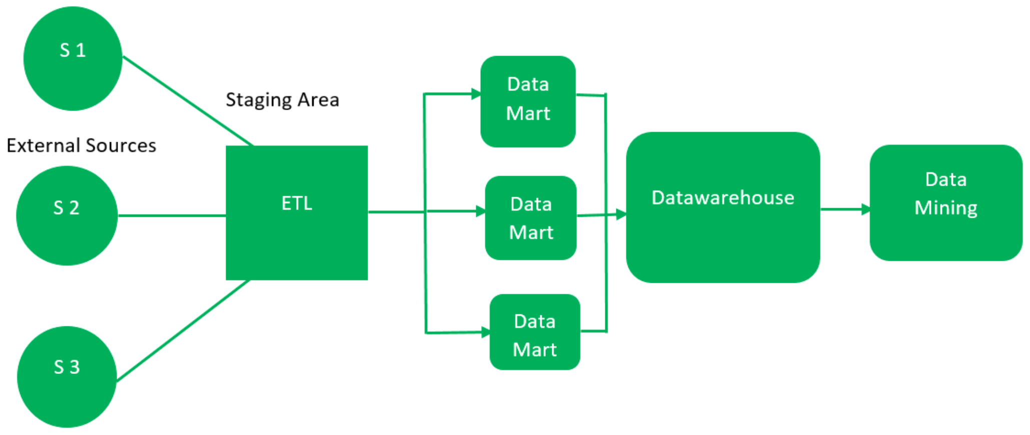
2. Also, this model is considered as the strongest model for business changes. That's why, big organisations prefer to follow this approach.

3. Creating data mart from datawarehouse is easy.

Disadvantages of Top-Down Approach –

1. The cost, time taken in designing and its maintenance is very high.

2. Bottom-up approach:



1. First, the data is extracted from external sources (same as happens in top-down approach).

2. Then, the data go through the staging area (as explained above) and loaded into data marts instead of datawarehouse. The data marts are created first and provide reporting capability. It addresses a single business area.

3. These data marts are then integrated into datawarehouse.

This approach is given by **Kinball** as – data marts are created first and provides a thin view for analyses and datawarehouse is created after complete data marts have been created.

Advantages of Bottom-Up Approach –

1. As the data marts are created first, so the reports are quickly generated.
2. We can accommodate more number of data marts here and in this way datawarehouse can be extended.
3. Also, the cost and time taken in designing this model is low comparatively.

Disadvantage of Bottom-Up Approach –

1. This model is not strong as top-down approach as dimensional view of data marts is not consistent as it is in above approach.

A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.

Features

- Databases in the collection are logically interrelated with each other. Often they represent a single logical database.
- Data is physically stored across multiple sites. Data in each site can be managed by a DBMS independent of the other sites.
- The processors in the sites are connected via a network. They do not have any multiprocessor configuration.
- A distributed database is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.

Distributed Database Management System

A distributed database management system (DDBMS) is a centralized software system that manages a distributed database in a manner as if it were all stored in a single location.

Features

- It is used to create, retrieve, update and delete distributed databases.
- It synchronizes the database periodically and provides access mechanisms by the virtue of which the distribution becomes transparent to the users.
- It ensures that the data modified at any site is universally updated.
- It is used in application areas where large volumes of data are processed and accessed by numerous users simultaneously.

- It is designed for heterogeneous database platforms.
- It maintains confidentiality and data integrity of the databases.

Factors Encouraging DDBMS

The following factors encourage moving over to DDBMS −

- **Distributed Nature of Organizational Units** − Most organizations in the current times are subdivided into multiple units that are physically distributed over the globe. Each unit requires its own set of local data. Thus, the overall database of the organization becomes distributed.
- **Need for Sharing of Data** − The multiple organizational units often need to communicate with each other and share their data and resources. This demands common databases or replicated databases that should be used in a synchronized manner.
- **Support for Both OLTP and OLAP** − Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) work upon diversified systems which may have common data. Distributed database systems aid both these processing by providing synchronized data.
- **Database Recovery** − One of the common techniques used in DDBMS is replication of data across different sites. Replication of data automatically helps in data recovery if database in any site is damaged. Users can access data from other sites while the damaged site is being reconstructed. Thus, database failure may become almost inconspicuous to users.
- **Support for Multiple Application Software** − Most organizations use a variety of application software each with its specific database support. DDBMS provides a uniform functionality for using the same data among different platforms.

Advantages of Distributed Databases

Following are the advantages of distributed databases over centralized databases.

Modular Development − If the system needs to be expanded to new locations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning. However, in distributed databases, the work simply requires adding new computers and local data to the new site and finally connecting them to the distributed system, with no interruption in current functions.

More Reliable − In case of database failures, the total system of centralized databases comes to a halt. However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance. Hence DDBMS is more reliable.

Better Response − If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response. On the other hand, in centralized systems, all queries have to pass through the central computer for processing, which increases the response time.

Lower Communication Cost − In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized. This is not feasible in centralized systems.

Adversities of Distributed Databases

Following are some of the adversities associated with distributed databases.

- **Need for complex and expensive software** − DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.
- **Processing overhead** − Even simple operations may require a large number of communications and additional calculations to provide uniformity in data across the sites.
- **Data integrity** − The need for updating data in multiple sites pose problems of data integrity.
- **Overheads for improper data distribution** − Responsiveness of queries is largely dependent upon proper data distribution. Improper data distribution often leads to very slow response to user requests.