DATABASEMANAGEMENTSYSTEMS

UNIT-1

Introduction:

Database: Thecollectionsofrelateddataarecalledasdatabase.

Database management system: It is a collection of interrelated data and a set of programs to access those data.

FILE PROCESSING SYSTEM: It is supported by a conventional operating system. The system stores permanent records in various file. It's having some disadvantages,

1. Dataredundancyandinconsistency
2. Difficultyinaccessingdata
3. Data isolation
4. Integrityproblems
5. Atomicityproblems
6. Concurrentaccessanomalies
7. Securityproblems

DATABASEMANAGEMENTSYSTEMSAPPLICATIONS:

RailwayReservationSystem

Database is required to keep record of ticket booking, train's departure and arrival status. Also if trains get late then people get to know it through database update.

LibraryManagementSystem

Therearethousands ofbooksin thelibrary soitis verydifficulttokeeprecord of all the booksinacopyorregister.SoDBMSusedtomaintainalltheinformationrelatetobook issue dates, name of the book, author and availability of the book.

Banking

We make thousands of transactions through banks daily and we can do this without going to the bank. So how banking has become so easy that by sitting at home we can send or get money through banks. That is all possible just because of DBMS that manages all the bank transactions.

Universitiesandcolleges

Examinations are done online today and universities and colleges maintain all these records throughDBMS.Student'sregistrationsdetails,results,coursesandgradesallthe information is stored in database.

SocialMediaSites

We all are on social media websites to share our views and connect with our friends. Daily millionsofuserssignedupforthesesocialmediaaccountslikefacebook,twitter,interest andGoogleplus.Buthowalltheinformationofusersarestoredandhowwebecomeable to connect to other people, yes this all because DBMS.

Telecommunications

AnytelecommunicationcompanycannoteventhinkabouttheirbusinesswithoutDBMS. DBMSismustfor these companies to store the call details and monthly postpaid bills.

Finance

Those dayshavegone farwheninformation related tomoney wasstoredinregistersand files.Todaythetimehastotallychangedbecausetherearelotsfthingtodowithfinance like storing sales, holding information and finance statement management etc.

Military

Military keeps records of millions of soldiers and it has millions of files that should be keep securedandsafe.AsDBMSprovidesabigsecurityassurancetothemilitaryinformationso itis widely usedin militaries. One can easilysearch foralltheinformationabout anyone within seconds with the help of DBMS.

DATABASESYSTEMVSFILESYSTEM:

Sno	Database	File system
1	Dataare notredundantandconsistent	Dataare redundantandinconsistent
2	Easy toaccess	Difficulttoaccess
3	Itensuresintegrity,atomicityand security	Itdoesn'tensuresit
4	Datacanbeconcurrentlyaccessed	Concurrentaccessisnotpossible
5	Easytoretrieveappropriatedata	Difficulttoretrieveappropriatedata

Disadvantagesoffilemanagementsystems:

1. **Data Mapping and Access: -** Although all the related information's are groupedandstoredindifferentfilesinsamedirectory,thereisnomappingbetweenany two files.

2. **Data Redundancy: -** There are no methods to validate the insertion of duplicate data in file system. Decentralization of data leads to data redundancy.

3. **DataDependence:-**Itistostorethedatainflatfiles.Sothereisnodependent on files.

4. **Data inconsistency: -** It is due to decentralization of data[i.e. no updating of the data]

5. **Security:-**Eachfilecanbepasswordprotected.Butwhatifhavetogiveaccess toonly fewrecordsin the file?Forexample, userhas tobegivenaccesstoview only theirbankaccount informationin the file.Thisisverydifficultinthe file system.

6. **Integrity:**InFMSitdoesnotchecktherules.

7. **Atomicity: -** It depends on transaction which means set of actions. Transaction is done or undone is atomicity.

8. **Concurrent Access: -** In FMS, multiple users cannot access same file at sametime.

NOTE: All these disadvantages in reverse become the advantages of Database management systems.

VIEW OF DATA: The database system provides an abstract view of data to users. The system hides details/ of how the data is stored and maintained. Here we are including 2things,

Dataabstraction:Data ishiddenthroughvariouslevelsofabstractionsuchas,

a. Physical level

b. Logicallevel

c. View level

a) **Physical level**- The lowest level of abstraction describes how the data are physically stored. The physical level describes complex low level data structure.

Figure-1.1

b) **Logical level**- The next level of abstraction describes about data is stored in thedatabase,andwhatrelationshipexistsamongthosedata.Itdescribestheentiredatabase in terms of a small number of relatively simple structures.

c) **View level**- The highest level of abstraction describesonlypart ofthe entire database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

Instances and schema: The collection of information stored in the database at a particularmomentiscalledaninstanceofthedatabase.Theoveralldesignofthedatabase is called the database schema. Types of schemas are

1. **Physicalschema**-describesthedatabasedesignatthephysicallevel.
2. **Logicalschema**-describesthedatadatabasedesignatthelogicallevel.
3. **Sublevel**-describesthedatadatabasedesignattheviewlevel.

DataIndependence:

Adatabasesystemnormallycontainsalotofdatainadditiontousers'data.Forexample, itstoresdataaboutdata,knownasmetadata,tolocateandretrievedataeasily.Itis ratherdifficulttomodifyorupdateasetofmetadataonceitisstoredinthe database.But asaDBMSexpands,itneedstochangeovertimetosatisfytherequirementsoftheusers. If the entire data is dependent, it would become a tedious and highly complex job.

Metadata itself follows a layered architecture, so that when we change data at one layer, it doesnotaffectthedataatanotherlevel.Thisdataisindependentbutmappedtoeach other.
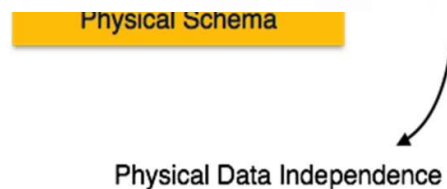
Physical Schema

Physical Data Independence
Figure-1.2

LogicalDataIndependence

Logicaldataisdataaboutdatabase,thatis,itstoresinformationabouthowdatais managedinside.Forexample,atable (relation)storedinthedatabaseandallits constraints, applied on that relation.

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

PhysicalDataIndependence

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical dataindependence is the powertochange thephysical datawithoutimpactingthe schema or logical data.

Forexample,incasewewanttochangeorupgradethestoragesystemitself−suppose we want to replace hard-disks with SSD − it should not have any impacton the logical data or schemas.

DATA MODELS: A data model is a collection of concepts that can be used to describe the structure of a database.

Itprovidesthenecessarymeanstoachievetheabstraction.

Bystructureofadatabasewemeanthedatatypes,relationshipsandconstraintsthat apply to data.

Design of database is called data model

Data models are classified intothree types:

1. **Objectbasedmodel**
    1. Objectorientedmodel
    2. E-Rmodel
    3. Functionalmodel

2. **Recordbasedmodel**
    1. Relationalmodel
    2. Networkmodel
    3. Hierarchicalmodel

3. **Physicalmodel**
    1. Unifymodel
    2. Frame model

Objectorientedmodel:

This data model is another method of representing real world objects. It considers eachobject in the world as objects and isolates it from each other. It groups its related functionalities togetherand allows inheriting its functionality to other related sub-groups.

E-R model:

In the below diagram, Entities or real world objects are represented in a rectangular box. Their attributes are represented in ovals. Primary keys of entities are underlined. All the entities are mapped using diamonds. This is one of the methods of representing ER model. There are many different forms of representation. Basically, ER model is a graphical representation of real world objects with their attributes and relationship. It makes the systemeasilyunderstandable.Thismodelisconsideredasatopdownapproachof designing a requirement.

Figure1.3
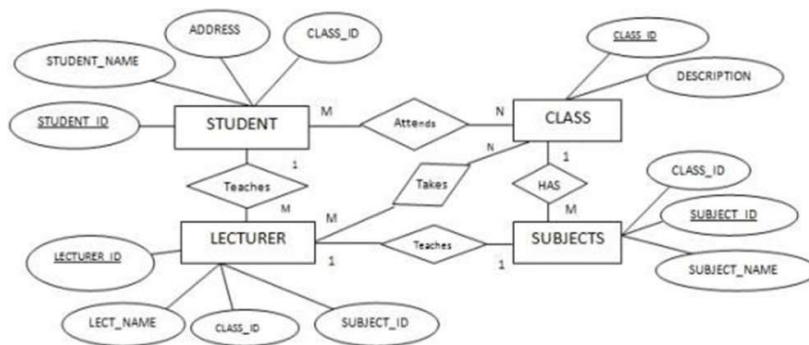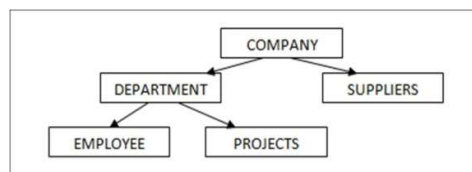
Functionalmodel:

Infunctionalmodelthedataisstoredand accessedinformoffunctions.

Hierarchicalmodel:

In this data model, the entities are represented in a hierarchical fashion. Here we identify a parententity,anditschildentity.Againwedrilldowntoidentifynextlevelofchildentity and so on. This model can be imagined as folders inside a folder.

Networkmodel:

This is the enhanced version of hierarchical data model. It is designed to address the drawbacksofthehierarchicalmodel.IthelpstoaddressM:Nrelationship.Thisdatamodel is also represented as hierarchical, but this model will not have single parent concept. Any child in the tree can have multiple parents here.
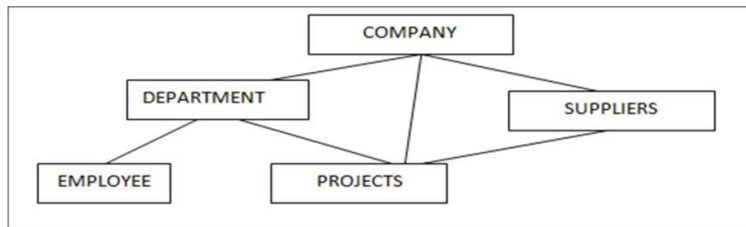

Figure-1.5

Relationalmodel:

This model is designed to overcome the drawbacks of hierarchical and network models. It is designed completely different from those two models. Those models define how they are structured in the database physically and how they are inter-related. But in the relational model, we are least bothered about how they are structured. It purely based on how the records in each table are related. It purely isolates physical structure from the logical structure. Logical structure is defines records are grouped and distributed.

EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID
100	Joseph	Clinton Town	10
101	Rose	Fraser Town	20
102	Mathew	Lakeside Village	10
103	Stewart	Troy	30
104	William	Holland	30

DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

Table1.1   Table1.2

Physicalmodel:

Physicaldatamodelrepresentthemodelwhereitdescribeshowdataarestoredin computer memory, how they are scattered and ordered in the memory, and how they would beretrievedfrommemory.Basicallyphysicaldatamodelrepresentsthedataatdatalayer orinternal layer. It represents each table, their columns and specifications, constraints like

primary key, foreign key etc. It basically represents how each tables are built and related to each other in DB.

In shortwecansayaphysicaldatamodelhas

- Tables and its specifications – table names and their columns. Columns arerepresented along with their data types and size. In addition primary key of each table is shown at the top of the column list.
- Foreign keys are used to represent the relationship between the tables. Mapping between the tables are represented using arrows between them.
- Physical data model can have denormalized structure based on the user requirement. The tables might not be in normalized forms.

DATABASEACCESSFORAPPLICATIONPROGRAMS:

➢ Data base will not interact directly it will use the software. It is called the embedded software

➢ The softwareusedissql.

➢ Thisgivesthecommandstothefrontend people(navyusers)

➢ Thepeoplewhointeractdirectlywithdatabasetheyareknownas**data base administrators** .They have all rights on data base

➢ Thesqlstatementsarecombinedwiththehostusersie:frontendiscalled**" embedded sql"**

Databaseusers:

1. Applicationprogrammers: Through application statements interacts with DML statements.

- ❖ Embeddedtohostlanguage

2. Sophisticated users: ThroughdirectSQLstatements.

3. Specialize users: Similar to sophisticated users but they write special data base programs like by PL/SQL.

4. Naive users or unsophisticated: Interactwithstandalonepermanentapplication programs

5. Data base administrators: create, resolvethedata .Grant permission to users. Storage can be checked. Backup and recovery

DBAcancheckbothtechnicaland non-technicaldetailsofthedatabase.

TRANSACTIONMANAGEMENT:

Ifthetransactionis notdoneproperlydatawilllosetheconsistency

- Eg:Bookingthe ticket

- Once the transaction is completed the reflection on data should be done immediatelyif it is not done consistency will lose.
- Eachtransactionshouldfollow the**ACID**properties

  A-Atomicity

  C-Consistency

  I-Isolation

  D-Durability

- **Atomicity**−thispropertystatesthatatransaction mustbetreatedasanatomic unit,thatis,eitherallofitsoperationsareexecutedornone.Theremustbeno state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.

- **Consistency**−thedatabasemustremaininaconsistentstateafterany

  transaction.Notransactionshouldhaveanyadverseeffectonthedataresidingin the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.

- **Durability** − the database should be durable enough to hold all its latest updateseven if the system fails or restarts. If a transaction updates a chunk of data in a databaseandcommits,thenthedatabasewillholdthemodifieddata.Ifa transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system brings back into action.

- **Isolation** − In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

Everytransactionhave5states:
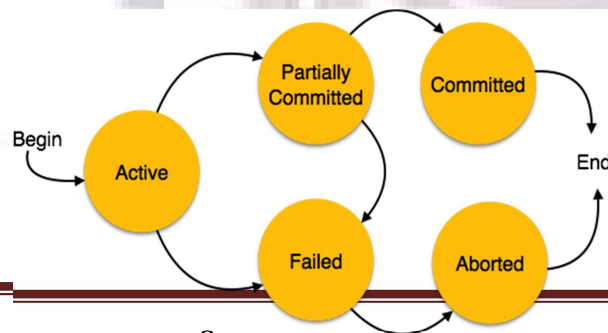
- **Active** – in this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** – When a transaction executes its final operation, it is said tobe in a partially committed state.
- **Failed**–Atransactionissaidtobeinafailedstateifanyofthechecksmadeby thedatabaserecoverysystemfails.Afailedtransaction can nolongerproceed further.
- **Aborted** – If any of the checks fails and the transaction has reached a failed state, thentherecoverymanagerrollsbackallitswriteoperationsonthedatabaseto bringthedatabasebacktoitsoriginalstatewhereitwaspriortotheexecutionof the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts –
    - o Re-startthe transaction
    - o Killthe transaction
- **Committed** – If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.
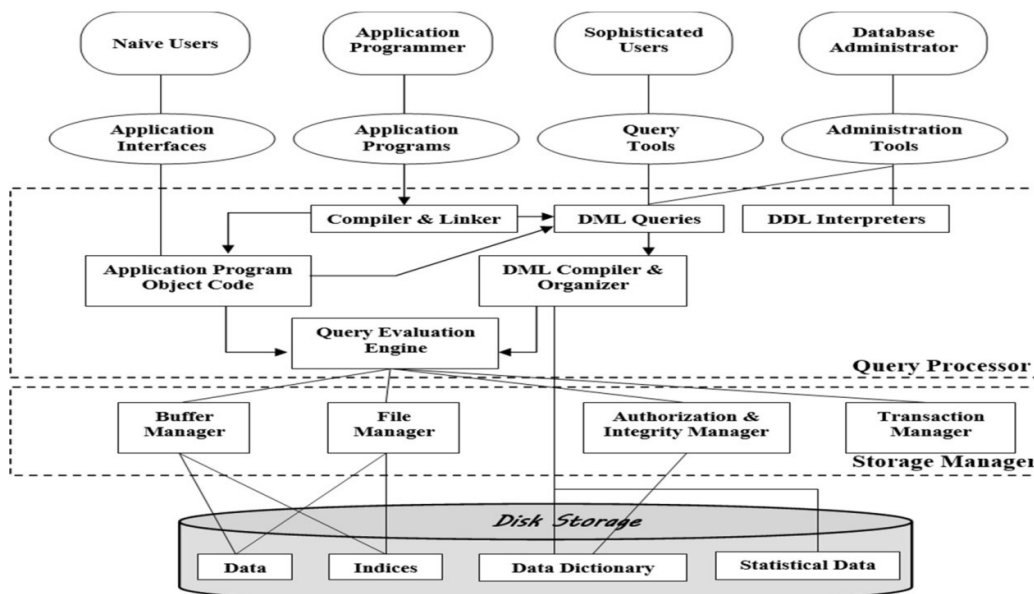
DATABASESYSTEMARCHITECTURE:

Figure: System Architecture
Figure-1.7

DBMSarchitectureismainlydividedinto2modules

1. Queryprocessor
2. Storagemanager

QUERYPROCESSOR:

Thequeryprocessorcomponentsindbmsacceptssqlcommandsgeneratedfrom varietyofusersinterfaces,producequeryevaluationplanexecutestheseplanswiththe data base and return results.

Thecomponentsare:

DMLcompiler:

ItiswhichtranslatesDMLstatementsinquerylanguage,lowlevelinstructionthatthe query evolution engine understands.

EmbeddedDMLpre-compiler:

Sqlcommandscanbeembeddedinhostlanguageapplicationprograms(c,c++,JAVA) which can be converted into DML statements by this embedded DML precompiler. The precompiler interact with the DML compiler to generate appropriate code.

DDLinterpreter:

WhichinterpretstheDDLstatementsandrecordstheminasetoftablescontaining metadata

Applicationprogramobjectcode:

It is which is low level instructions of a program written by no voice users/naive users which the query evaluation engine understands and execute them

Queryevaluationengine:

Itis whichexecuteslow levelinstructionsgeneratedbyDMLcompiler.

STORAGEMANAGER:

The storage manager components in dbms provide the interface between the physical level data stored in data base and the program on queries requesting the stored data from data base.

Thestoragemanagercomponentsare:

Authorizationandintegritymanager:

Itiswhichtestforthesatisfactionofintegrityconstraintsandcheckstheauthorityofuser to access the data.

Recoverymanager:

Itiswhichisresponsibleformaintainingalogandrestoringthesystemtoaconsistent

state,aftersystemfailure.

Transactionmanager:

Itiswhichensurethatthedatabaseremainsinconsistentstatedespiteofthesystem failure and the concurrent transaction execution proceed without conflicting the transaction. **Lock manager:** Alockisa mechanismusedtocontrolaccesstodatabaseobjects.

Two kinds of locks are commonly supported by a DBMS: – Shared locks on an object can be held by two different transactions at the same time.

Exclusive lock on an object ensures that no other transaction holds any lock on this object.

Everytransactionbeginsbyobtainingasharedlockoneachdataobjectthatitneedsto readandanexclusivelockoneachdatathatitneedstomodify,andthenreleaseallits locks after completing all actions.

Itiswhichkeepstrackofrequestsforlocksandgrantlocksondatain thedatabase.

Buffermanager:

Itiswhichisresponsibleforfetchingthedatafromdiskstorageintomainmemoryand deciding what data to cache memory.

Diskspacemanager:

Itiswhichmanagesallocationofspaceondiskstorageandthedatastructuresusedto represent the stored information on disk.

Diskstoragecomponentsare:

Datafiles:

Whichstoresthedatabaseitself   i.e.Theactualdatainthedatabase.

Datadictionary:

ItiswhichstorestheMetadataaboutthestructureofdatabase.Thedatadictionaryis used very heavily. Therefore great emphasis should be placed on developing a good design and efficient implementation of the data dictionary.

Indices:

Itiswhichprovidesthefastaccesstothe dataitemsthatholdtheparticularvalue.

Statisticaldata:

It stores the statistical data about the data in the database .This information is used by the query processor to select efficient ways and to execute user queries.

DATABASEDESIGNWITHE-RMODEL

E-Rmodel:

An **entity–relationship model (ER model)** is a systematic way of describing and defining a business process. An ER model is typically implemented as a database. The main components of E-R model are: entity set and relationship set.

HerearethegeometricshapesandtheirmeaninginanE-RDiagram–

Rectangle: RepresentsEntitysets.

Ellipses:Attributes

Diamonds: RelationshipSet

Lines: TheylinkattributestoEntitySetsandEntitysetstoRelationshipset.

Double Ellipses: Multivalued Attributes

Dashed Ellipses: Derived Attributes

Double Rectangles: Weak Entity Sets

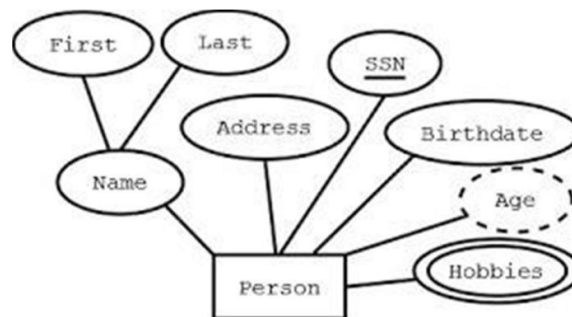**DoubleLines:**Totalparticipation ofanentityin arelationshipset

AsampleE-RDiagram:

Figure-1.8

WhatisDatabaseDesign?

DatabaseDesignisacollectionofprocessesthatfacilitatethedesigning, development, implementation and maintenance of enterprise data management systems

Ithelpsproducedatabasesystems

1. Thatmeettherequirementsoftheusers
2. Havehighperformance.

The main objectives of database designing are to produce logical and physical designs models of the proposed database system.

The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.

The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as databasemanagement systems (DBMS).

Attributes:

An **Attribute** describes a property or characterstic of an entity. For example, Name, Age, Address etc can be attributes of a Student. An attribute is represented using eclipse.
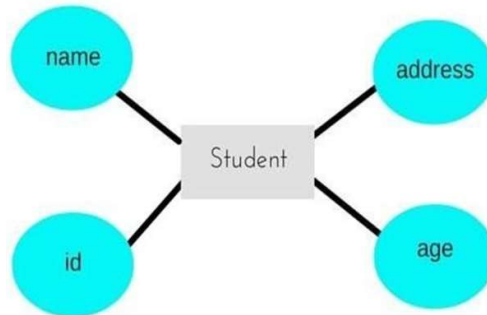
Figure-1.9

KeyAttribute:

Key attribute represents the main characterstic of an Entity. It is used to represent Primary key. Ellipse with underlying lines represent Key Attribute.
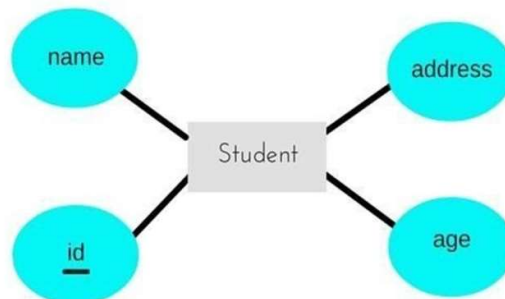
Figure-1.10

CompositeAttribute:

An attribute can also have their own attributes.Theseattributesareknownas **Composite** attribute.

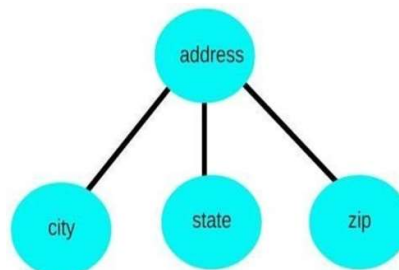Figure-1.11

DerivedAttribute:

The last category that attributes can be defined is called a derived attribute, where one attribute is calculated from another attribute.

Figure-1.12

Multivaluedattribute:

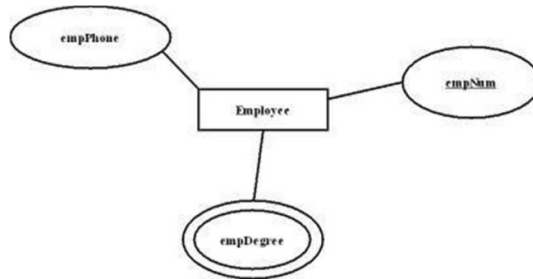multi-valued attributes usually can store multiple data in them.

Figure-1.13

DescriptiveAttribute:

.**Attributes**oftherelationshipiscalled**descriptiveattribute**.

Figure-1.14

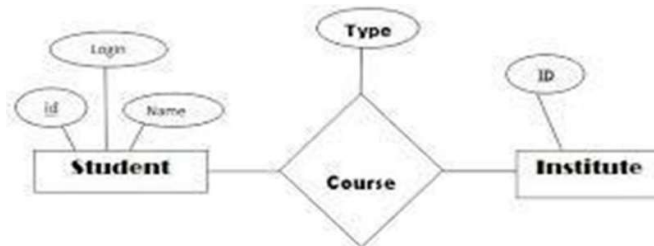DeptofIT,NRCM   VANILKUMARASST.PROF

Entities: An **entity set** is a set of entities of the same type (e.g., all persons having an account at a bank).

The entitysetwhichdoes nothave sufficientattributes to forma primarykey iscalled as Weak entity set.

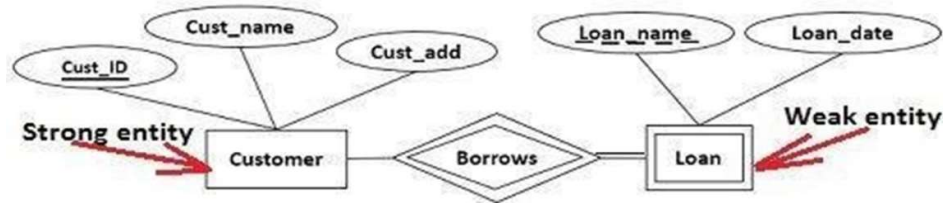AnentitysetthathasaprimarykeyiscalledasStrongentityset.

Figure-1.14

Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

BinaryRelationshipandCardinality:

A relationship where two entities are participating is called a **binary relationship**.Cardinalityisthenumberofinstanceofanentityfromarelationthatcanbeassociated with the relation.

- **One-to-one**–Whenonlyoneinstanceofanentityisassociatedwiththe

    relationship,itismarkedas'1:1'.Thefollowingimagereflectsthatonlyone instance of each entity should be associated with the relationship. It depicts one-to- one relationship.

Figure-1.15

- **One-to-many** – When more than one instance of an entity is associated with a relationship,itismarkedas'1:N'.Thefollowingimagereflectsthatonlyone instanceofentityontheleftandmorethanoneinstanceofanentityontheright can be associated with the relationship. It depicts one-to-many relationship.

- **Many-to-one** – When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instanceofanentityontheleftandonlyoneinstanceofanentityontherightcan be associated with the relationship. It depicts many-to-one relationship.

Figure-1.16

- **Many-to-many** – The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.

Figure-1.17

ParticipationConstraints

- **Total Participation** – Each entity is involved in the relationship. Total participationis represented by double lines.

- **Partial participation** –Notallentitiesareinvolvedintherelationship.Partial participation is represented by single lines.
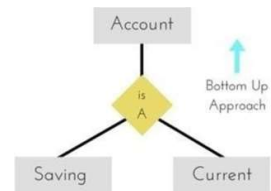

Figure-1.18

AdditionalfeaturesofERModel:

Generalization:

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

Specialization:

Specialization is opposite to Generalization. Itisatop-downapproachinwhichone higher level entity can be broken down into twolowerlevelentity.Inspecialization, some higher level entities may not have lower-level entity sets at all.
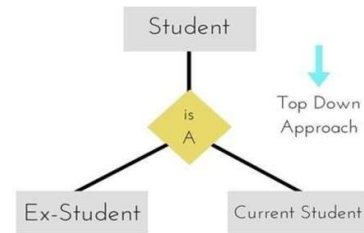
Figure-1.20

Aggregation:

Aggregation is a process when relation between two entity is treated as a single entity. Here the relation between Center and Course, is acting as an Entity in relation with Visitor.
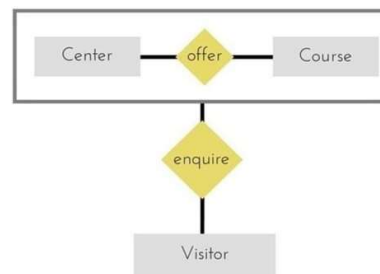
Figure-1.21

ConceptualDesign:

• Conceptual design follows requirements analysis, Yields a high-level description of data to be stored

• ERmodelpopularforconceptualdesign.Constructsareexpressive,closetotheway people think about their applications.

• Basicconstructs: *entities*, *relationships*, and *attributes* (ofentitiesandrelationships)

• Constraints

• Someadditionalconstructs: *weakentities*, *ISAhierarchies*

• Note:TherearemanyvariationsonERmodel

UNIT-II

IntroductiontoRelationalModel:

The main construct for representing data in the relational model is a **relation**. A relation consists of a **relation schema** and a **relation instance**. The relation instance is a table,andtherelationschemadescribesthecolumnheadsforthetable.Wefirstdescribe the relation schema and then the relation instance. The schema species the relation's name, thenameofeachfi**eld**(or**column**,or**attribute**)andthe**domain**ofeachfield.Adomain isreferredtoinarelationschemabythe**domainname**andhasasetofassociated **values**.

We usetheexampleofstudentinformation:

Students (*sid:*string,*name:*string,*login:*string,*age:*integer,gpa:real)

Thissays,forinstance,thatthefieldnamed*sid*hasadomainnamedstring.Thesetof values associated with domain string is the set of all character strings.We now turn to the instances of a relation. An **instance** of a relation is a set of **tuples**, also called **records**, in which each tuple has the same number of fields as the relation schema. A relationinstancecanbethoughtofasa*table*inwhicheachtupleisa*row*,andallrowshavethesame numberoffields.(Theterm*relationinstance*isoftenabbreviatedtojust*relation*,when there is no confusion with other aspects of a relation such as its schema.)

Fields(Attributes,columns)   Fieldnames

sid	name	login	age	gpa
50000	Jones	Jones@cs	19	1.8
53666	Smith	Smith@cs	21	2.0
53668	Roy	Roy@cs	18	3.2
54532	Mealey	Mealey@cs	20	3.8
54566	Rony	Rony@cs	14	3.4
53452	Guldu	Guldu@cs	12	3.3

Aninstanceofthe Studentsrelation   Tuples(Records,rows)

Table2.1

The instance *S*1 contains six tuples and has, as we expect from the schema, five fields. Note that no two rows are identical. This is a requirement of the relational model each relation is defined to be a *set* of unique tuples or rows.1. The order in which the rows are listed is not important.Thus*sid*isfield1ofStudents,*login*isfield3,andsoon.Ifthisconventionis used, the order of fields is significant.

20

Most database systems use a combination of these conventions. For example, in SQL the named fields convention is used in statements that retrieve tuples, and the ordered fields convention is commonly used when inserting tuples.

Arelationschemaspecifiesthedomainofeachfieldorcolumninthe relationinstance.

These **domain constraints** in the schema specify an important condition that we want each instanceoftherelationtosatisfy:The valuesthatappearin acolumnmustbedrawnfrom thedomainassociatedwiththatcolumn.Thus,thedomainofafieldisessentiallythe*type* ofthatfield,inprogramminglanguageterms,andrestrictsthevaluesthatcanappearin the field.

The**degree**,alsocalled**arity**,ofarelationisthenumberoffields.The**cardinality** of a relation instance is the number of tuples in it. In Figure the degree of the relation (the numberofcolumns)isfive,andthecardinalityofthisinstanceissix.A**relational database** is a collection of relations with distinct relation names. The **relational database schema** is the collection of schemas for the relations in the database.

CreatingandModifyingrelationsusingSQL:

The SQL that supports the creation, deletion, and modification of tables is called the **Data Definition Language (DDL)**. Now, we will just consider domains that are built-intypes, such as integer.

The CREATE TABLE statement is used to define a new table. To create the Students relation, we can use the following statement:

CREATETABLEStudents(sidCHAR(20),

nameCHAR(30),

loginCHAR(20), age

INTEGER,

gpaREAL)

Tuples areinserted usingtheINSERT command.Wecaninserta single tupleintothe Students table as follows:

INSERT INTOStudents(sid,name,login,age,gpa)

VALUES(53688,`Smith',`smith@ee',18,3.2)

Wecanoptionallyomitthe listofcolumnnamesintheINTOclauseandlistthevalues intheappropriateorder,butitisgoodstyletobeexplicitaboutcolumnnames.Wecan deletetuplesusingtheDELETEcommand.WecandeleteallStudentstupleswith*name* equal to Smith using the command:

DELETEFROMStudentsSWHERES.name=`Smith'

We can modify the column values in an existing row using the UPDATE command. For example, wecanincrementthe age and decrementthe gpaof the studentwith *sid* 53668:

        UPDATEStudentsS

                SETS.age =S.age+ 1, S.gpa = S.gpa - 1 WHERE

                S.sid = 53688

These examples illustrate some important points. The WHERE clause is applied first and determines which rows are to be modified. The SET clause then determines how these rows are to be modified. If the column that is being modified is also used to determine the new value,thevalueusedintheexpressionontherightsideofequals(=)isthe*old*value,that is, before the modification. To illustrate these points further,

consider the following variation of the previous query:

        UPDATE Students S

                SETS.gpa=S.gpa- 0.1 WHERE

                S.gpa >= 3.3

Ifthisqueryisappliedontheinstance*S*1ofStudentsshowninFigure,weobtainthe instance shown in following figure

sno	name	login	age	gpa
50000	Jones	Jones@cs	19	3.2
53666	Smith	Smith@cs	21	3.3
53668	Roy	Roy@cs	13	3.2
54532	Mealey	Mealey@cs	15	3.7
54566	Rony	Rony@cs	11	1.8
53452	Guldu	Guldu@cs	12	2.0

StudentsInstanceafterupdate

Table2.2

INTEGRITYCONSTRAINTSOVERRELATIONS:

        Adatabaseisonlyasgoodastheinformationstoredinit,andaDBMSmust therefore help prevent the entry of incorrect information. An **integrity constraint (IC)** is a conditionthatisspecifiedonadatabaseschema,andrestrictsthedatathatcanbestored in an instance of the database. If a database instance satisfies all the integrity constraints specified on the database schema, it is a **legal** instance. A DBMS **enforces** integrity constraints, in that itpermits only legal instances to be stored in the database.

Integrityconstraintsarespecifiedandenforcedatdifferenttimes:

1. When the DBA or end user defines adatabase schema, he or she specifies the ICs that must hold on any instance of this database.

2. Whenadatabaseapplicationisrun,theDBMSchecksforviolationsanddisallows changes to the data that violate the specified ICs. (In some situations,ratherthan disallowthechange,theDBMSmightinsteadmakesomecompensatingchangestothedatato ensure that the database instance satisfies all ICs. In any case, changes to the database are not allowed to create an instance that violates any IC.)

Manykindsofintegrityconstraintscanbespecifiedintherelationalmodel.

TYPESOFINTEGRITYCONSTRAINTS

Varioustypesofintegrityconstraintsare-

1. DomainIntegrity
2. EntityIntegrityConstraint
3. ReferentialIntegrityConstraint
4. Key Constraints

Domain Integrity- Domain integrity means the definition of a valid set of values foranattribute.Youdefinedatatype,lengthorsize,isnullvalueallowed,isthevalueunique ornotforanattribute,thedefaultvalue,therange(valuesinbetween)and/orspecific values for the attribute.

Entity Integrity Constraint- This rule states that in any database relation value of attribute of a primary key can't be null.

EXAMPLE- Consider a relation "STUDENT" Where "Stu_id" is a primary key and it must not contain any null value whereas other attributes may contain null value e.g "Branch" in the following relation contains one null value.

Stu_id	Name	Branch
113421	Ajay	IT
113564	Sandy	CSE
112453	Sujith	
114532	Rose	CIVIL
113422	Jack	IT

Table2.3

Referential Integrity Constraint-It states that if a foreign key exists in a relation then either the foreign key value must match a primary key value of some tuple in its home relation or the foreign key value must be null.

Therulesare:

1. Youcan'tdeletearecordfromaprimary tableifmatchingrecordsexistinarelatedtable.

2. Youcan'tchangeaprimarykeyvalueintheprimarytableifthatrecordhasrelated records.

3. Youcan'tenteravalueintheforeignkeyfieldoftherelatedtablethatdoesn'texistin the primary key of the primary table.

4. However, you can enter a Null value in the foreignkey, specifying that the records are unrelated.

EXAMPLE-Consider 2 relations "stu" and "stu_1" Where "Stu_id " is the primary key in the "stu" relation and foreign key in the "stu_1" relation.

Relation"stu"

Stu_id	Name	Branch
113421	Ajay	IT
113564	Sandy	CSE
112453	Sujith	ECE
114532	Rose	CIVIL
113422	Jack	IT

Table2.4

Relation"stu_1"

Stu_id	Course	Duration
113421	B.Tech	4years
113564	B.Tech	4years
112453	B.Tech	4years
114532	B.Tech	4years
113422	B.Tech	4years

Table2.5

Examples:

Rule1.Youcan'tdeleteanyoftherowsinthe"stu"relation thatarevisiblesinceallthe "stu" are in use in the "stu_1" relation.

Rule 2. You can't change any of the "Stu_id" in the "stu" relation since all the "Stu_id" arein use in the "stu_1" relation.

Rule 3. The values that you can enter in the" Stu_id" field in the "stu_1" relation must be in the" Stu_id" field in the "stu" relation.

Rule4 Youcanenteranullvalue inthe"stu_1"relationiftherecordsareunrelated.

Key Constraints- A Key Constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple.

SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.

Constraintscanbedividedintofollowingtwotypes,

- **Column level constraints:** limits only column data.Thestyle is calledInline specification.

- **Table level constraints:** limits whole table data. The style is called out of line specification

Constraintsareusedtomakesurethattheintegrityofdataismaintainedinthedatabase.

Followingarethemostusedconstraints.

- NOTNULL
- UNIQUE
- PRIMARYKEY
- FOREIGNKEY
- UNIQUE

NOTNULLConstraint:

NOTNULLconstraintrestrictsacolumnfromhavingaNULLvalue.Once**NOTNULL** constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value. One important point to note about NOT NULL constraint is that it cannot be defined at table level.

ExampleusingNOTNULLconstraint:

CREATE table Student (s_id int NOT NULL, Name varchar(60), Age int);

The above querywill declare thatthe **s_id**field of **Student** table will not take NULLvalue.

UNIQUEConstraint:

UNIQUE constraint ensures that a field or column will only have unique values. A UNIQUE constraintfieldwillnothaveduplicatedata.UNIQUEconstraintcanbeappliedatcolumn level or table level.

ExampleusingUNIQUEconstraintwhencreatingaTable(TableLevel)

CREATEt

The abovequerywilldeclare thatthe**s_id**fieldof**Student**tablewill onlyhaveunique values and won't take NULL value.

ExampleusingUNIQUEconstraintafterTableiscreated(ColumnLevel)

ALTERtal

Theabovequeryspecifiesthat**s_id**fieldof**Student**tablewillonlyhave uniquevalue.

PrimaryKeyConstraint:

Primary key constraint uniquely identifies each record in a database. A Primary Key must containuniquevalueanditmustnotcontainnullvalue.UsuallyPrimaryKeyisusedto index the data inside the table.

ExampleusingPRIMARYKEYconstraintat TableLevel:

CREATEt

TheabovecommandwillcreatesaPRIMARYKEYonthes_id.

ExampleusingPRIMARYKEYconstraintatColumnLevel:

ALTERtal

TheabovecommandwillcreatesaPRIMARYKEYonthes_id.

ForeignKeyConstraint:

FOREIGN KEY is used to relate two tables. FOREIGN KEY constraint is also used to restrict actions that would destroy links between tables. To understand FOREIGN KEY, let's see itusing two table.

Customer_detailtable:			Order_detailtable:		
C_id	Customer_name	Address	O_id	Order_name	C_id
101	Ajay	Delhi	10	Order1	101
102	Adam	Kolkata	11	Order2	102
103	Anand	Noidak	12	Order3	103
Table2.6			Table2.7		

In**Customer_Detail**table,c_idistheprimarykeywhichissetasforeignkeyin **Order_Detail**table.Thevaluethatisenteredinc_idwhichissetasforeignkeyin

Order_Detail table must be present in **Customer_Detail** table where it is set as primary key. This prevents invalid data to be inserted into c_id column of **Order_Detail** table.

Example usingFOREIGNKEYconstraintatTableLevel:

```
CREATE table Order_Detail(order_id int PRIMARY KEY,order_name varchar(60) NOT NULL,
c_id int FOREIGN KEY REFERENCES Customer_Detail(c_id));
```

In this query, c_id in table Order_Detail is made as foriegn key, which is a reference of c_id column of Customer_Detail.

Example usingFOREIGNKEYconstraintatColumnLevel:

```
ALTERtable Order_Detailadd FOREIGNKEY(c_id)REFERENCESCustomer_Detail(c_id);
```

Behaviour ofForiegnKeyColumnonDelete:

TherearetwowaystomaintintheintegrityofdatainChildtable,whenaparticularrecord isdeletedinmaintable.WhentwotablesareconnectedwithForiegnkey,andcertaindata inthemaintableisdeleted,forwhichrecordexitinchildtabletoo,thenwemusthave some mechanism to save the integrity of data in child table.

Deleting Foriegn Key

Cascade
(on Delete cascade)
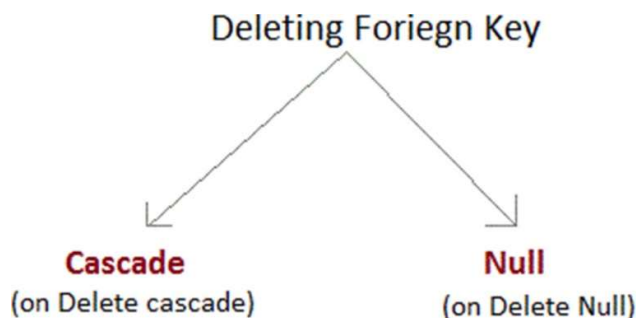
Null
(on Delete Null)

- **OnDeleteCascade:**Thiswillremovetherecordfromchildtable,ifthatvalueof foriegn key is deleted from the main table.
- **OnDeleteNull:**ThiswillsetallthevaluesinthatrecordofchildtableasNULL,for which the value of foriegn key is deleted from the main table.
- Ifwedon'tuseanyoftheabove,thenwecannotdeletedatafromthemaintablefor which data in child table exists. We will get an error if we try to do so.

ERROR:Recordinchild tableexist

CHECK Constraint:

CHECKconstraintisusedtorestrictthevalueofacolumnbetweenarange.Itperforms checkonthevalues,beforestoringthemintothedatabase.Itslikeconditionchecking before saving data into a column.

ExampleusingCHECKconstraintatTable Level:

create table Student(s_id int NOT NULL **CHECK(s_id > 0)**,

Name varchar(60) NOT NULL, Age int);

Theabovequerywillrestrictthes_idvaluetobegreaterthanzero.

ExampleusingCHECKconstraintatColumnLevel:

ALTER tableStudentaddCHECK(s_id>0);

QUERYINGRELATIONALDATA

A**relationaldatabasequery**isaquestion aboutthedata,andtheanswerconsists of a new relation containing the result. For example, we might want to find all students younger than 18 or all students enrolled in Reggae203. A **query language** is a specialized language for writing queries.

SQL is the most popular commercial query language for a relational DBMS. We now present someSQLexamplesthatillustratehoweasilyrelationscanbequeried.Considerthe instance of the Students relation shown figure . We can retrieve rows corresponding to students who are younger than 18 with the following SQL query:

SELECT*FROMStudentsS

WHERE S.age <18

The symbol *means that we retain all fields of selected tuples in the result. Tounderstand thisquery,thinkofS asavariablethattakeson thevalueofeach tuplein Students, one tupleaftertheother.Thecondition*S.age<18*intheWHEREclausespecifiesthatwewant to select only tuples in which the *age* field has a value less than 18. This query evaluates to the relation shown in below figure.

sid	name	login	age	gpa
54566	Rony	Rony@cs	14	3.4
53452	Guldu	Guldu@cs	12	3.3

Table2.8

Studentswith*age*<18onInstanceS1

This example illustrates that the domain of a field restricts the operations that are permitted onfieldvalues,inadditiontorestrictingthevaluesthatcanappearinthefield.The condition*S.age<18*involvesanarithmeticcomparisonofan*age*valuewithanintegerand is permissible because the domain of *age* is the set of integers. In addition to selecting a subset of tuples, a query can extract a subset of the fields of each selected tuple. We can computethenamesandloginsofstudentswhoareyoungerthan18withthefollowing query:

SELECTS.name,S.loginFROMStudentsSWHERES.age<18

Thesolutiontothisqueryisthefollowingfigure:

name	Login
Rony	Rony@cs
Guldu	Guldu@cs

LOGICALDATABASEDESIGN:ERTORELATIONAL

The ER model is convenient for representing an initial, high-level database design. Given an ER diagram describing a database, there is a standard approach to generating a relationaldatabaseschemathatcloselyapproximatestheERdesign.Itisdiscussedin2nd unit.

SQLAlias:

Alias is used to give an alias name to a table or a column. This is quite useful in caseoflargeorcomplexqueries.Aliasismainlyusedforgivingashortaliasnameforacolumn or a table with complex names.

SyntaxofAliasfortablenames,

SELECTcolumn-namefrom*table-name*as**alias-name**

Following isanExampleusingAlias,

SELECT * from Employee_detail as **ed**;

Alias syntax for columns will be like,

SELECT *column-name* as **alias-name** from*table-name*

Example using alias for columns,

SELECTcustomer_idas**cid**fromEmp;

Introductiontoviews:

A view is nothing more than a SQL statement that is stored in the database with a associated name. A view is actually a composition of a table in the form of apredefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which dependson thewritten SQLquery tocreate a view.

Views,wicharea typeofvirtualtablesallowuserstodothefollowing−

- Structuredatain awaythatusersorclassesofusersfindnaturalorintuitive.
- Restrictaccesstothedatainsuchawaythatusercanseeand(sometimes)modify exactly what they need and no more.
- Summarize datafromvarioustableswhichcanbeusedtogeneratereports.
    - Viewisavirtualtableorlogicaltableoritactslikeamirrorimage.
    - Viewinpracticallycontainsnodatabyitself.
    - Thetableuponwhichaviewisbasedarecalledasbase tables.
    - Viewscanbecreated asobjectviewsorrelationalviews.
    - Viewissimplyaselectstatement.

CreatingViews:

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic**CREATEVIEW**syntaxisasfollows−

```
CRE
```

Youcan includemultipletablesinyourSELECT statementin asimilarway asyouuse themin a normal SQL SELECT query.

Example:ConsidertheCUSTOMERStablehavingthefollowingrecords−

```
+......+............+......+..............+............+
|ID|NAME       |AGE|ADDRESS      |SALARY    |
+......+............+......+..............+............+
|1|Ramesh       |32|Ahmedabad|2000.00|
| 2 |Khilan    | 25|Delhi         |1500.00 |
| 3 |kaushik|23|Kota           |2000.00|
| 4 |Chaitali|25 |Mumbai        |6500.00|
| 5 |Hardik    |27|Bhopal        |8500.00|
```

Following is an example to create a viewfrom the CUSTOMERStable. This viewwould be usedto have customer name and age from the CUSTOMERS table.

```
SQL>CREATEVIEWCUSTOMERS_VIEWASSELECTname,ageFROMCUSTOMERS;
```

Now,youcanqueryCUSTOMERS_VIEWinasimilarwayasyouqueryanactualtable. Following is an example for the same.

```
SQL >SELECT*FROMCUSTOMERS_VIEW;
```

Thiswouldproducethefollowingresult.

```
+..............+......+
|name      |age|
+..............+......+
|Ramesh    |32|
|Khilan    |25|
|kaushik|23|
|Chaitali|25|
|Hardik    |27|
|Komal     |22|
|Muffy     |24|
+..............+......+
```

WITHCHECKOPTION

TheWITHCHECKOPTIONisaCREATEVIEWstatementoption.ThepurposeoftheWITH CHECK OPTION is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition.

Iftheydonotsatisfythecondition(s),theUPDATEor INSERTreturnsanerror.

The following code block has an example of creating same view CUSTOMERS_VIEW with the WITH CHECK OPTION.

```
CREATEVIEWCUSTOMERS_VIEWAS
SELECT
name,ageFROMCUSTOME
RS
WHEREageISNOTNULL
```

The WITH CHECK OPTION in this case should deny the entry of any NULL values in the view's AGE column, because the view is defined by data that does not have a NULL value in the AGE column.

ForceViewCreation

Forcekeywordisusedwhilecreatingaview.ThiskeywordforcetocreateVieweven if the table does not exist. After creating a force View if we create the base table and enter values in it, the view will be automatically updated.

SyntaxforforcedViewis,

CREATE or REPLACE *force* **view** *view_name* AS

SELECT *column_name*(s)

FROM*table_name*

WHERE*condition*

Read-OnlyView

We can create a view with read-only option to restrict access to the view.

Syntax to create a view with Read-Only Access

CREATE or REPLACE *force* **view** *view_name* AS

SELECT *column_name*(s)

FROM*table_name*

WHERE*condition*with**read-only**

The above syntax will create view for read-only purpose, we cannot Update or Insert data into read-only view. It will throw an error.

UpdatingaView

Aviewcanbeupdatedundercertainconditionswhicharegivenbelow−

- TheSELECTclausemaynotcontainthekeywordDISTINCT.
- TheSELECTclausemaynotcontainsummaryfunctions.
- TheSELECTclausemaynotcontainsetfunctions.
- TheSELECTclause maynotcontainsetoperators.
- TheSELECTclausemaynotcontainanORDERBYclause.
- TheFROMclausemaynotcontainmultipletables.
- TheWHEREclausemaynotcontainsubqueries.
- ThequerymaynotcontainGROUPBYorHAVING.
- Calculated columnsmaynotbeupdated.
- All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So, if aview satisfies all the above-mentioned rules then you can update that view. Thefollowing code block has an example to update the age of Ramesh.

SQL >UPDATECUSTOMERS_VIEW   SETAGE =35   WHEREname='Ramesh';

ThiswouldultimatelyupdatethebasetableCUSTOMERSandthesamewouldreflectinthe   view   itself. Now, try to query the base table and the SELECT statement would produce the following result.

```
+-----+-----------+------+-------------+----------+
|ID|NAME       |AGE|ADDRESS    |SALARY   |
+-----+-----------+------+-------------+----------+
|1|Ramesh      |35|Ahmedabad|2000.00|
| 2 |Khilan    | 25|Delhi        |1500.00|
| 3 |kaushik|23|Kota          |2000.00|
| 4 |Chaitali|25| Mumbai       |6500.00|
| 5 |Hardik    |27|Bhopal       |8500.00|
| 6 |Komal     |22|MP        |4500.00|
| 7 |Muffy     |24 | Indore     |10000.00|
+-----+-----------+------+-------------+----------+
```

InsertingRowsinto aView

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command.

Here,wecannotinsertrowsintheCUSTOMERS_VIEWbecausewehavenotincludedallthe

NOTNULLcolumnsinthisview,otherwiseyoucaninsertrowsinaviewinasimilarwayas you insert them in a table.

DeletingRowsfromaView

Rowsofdatacanbedeletedfromaview.ThesamerulesthatapplytotheUPDATEand INSERT commands apply to the DELETE command.

FollowingisanexampletodeletearecordhavingAGE=22.

SQL>DELETEFROMCUSTOMERS_VIEW   WHEREage=22;

ThiswouldultimatelydeletearowfromthebasetableCUSTOMERSandthesamewould reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

```
+-----+--------------+------+---------------+-------------+
|ID|NAME       |AGE|ADDRESS      |SALARY    |
+-----+--------------------+----------+----------------------------+-------------------+
```

```
1
2
3
4
5
```

DroppingViews

Obviously, where you have a view, you need a way to drop the view if it is no longer needed. The syntax is very simple and is given below −

```
DROPVIEWview_name;
```

FollowingisanexampletodroptheCUSTOMERS_VIEWfromtheCUSTOMERStable.

```
DROPVIEWCUSTOMERS_VIEW;
```

TypesofView

Therearetwotypesofview,

- SimpleView
- ComplexView

SimpleView	**ComplexView**
Createdfromonetable	Createdfromoneormoretable
Doesnotcontainfunctions	Contain functions
Doesnotcontaingroupsofdata	Containsgroupsofdata

2.6DESTROYING/ALTERINGTABLESANDVIEWS:

If we decide that we no longer need a base table and want to destroyit, we can usethe DROP TABLE command. For example, DROP TABLE Students RESTRICT destroys the StudentstableunlesssomevieworintegrityconstraintreferstoStudents;ifso,the command fails. If the keyword RESTRICT is replaced by CASCADE, Students is dropped and anyreferencingviewsorintegrityconstraintsare(recursively)droppedaswell;oneof these two keywords must always be specified. A view can be dropped using the DROP VIEW command, which is just like DROP TABLE.ALTER TABLE modifies the structure of an existing table. To add a column called *maiden-name* to Students, for example, we would use the following command:

ALTERTABLEStudentsADDCOLUMNmaiden-namechar(10)

ThedefinitionofStudentsismodifiedtoaddthiscolumn,andallexistingrowsarepadded

with*null*valuesinthiscolumn.ALTERTABLEcanalsobeusedtodeletecolumnsandtoadd

or drop integrity constraints on a table; we will not discuss these aspects of the command beyondremarkingthatdroppingcolumnsistreatedverysimilarlytodroppingtablesor views.

Relationaldatabasesystemsareexpectedtobeequippedwithaquerylanguagethatcan assist its users to query the database instances. There are two kinds of query languages − relational algebra and relational calculus proposed by E. F. Codd in 1976.

2.7RELATIONALALGEBRA:

- Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output.
- It usesoperatorsto performqueries. Anoperatorcan beeither**unary**or**binary**. They accept relations as their input and yield relations as their output.
- Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

Terminology

- Relation −asetoftuples.
- Tuple−acollectionofattributeswhichdescribesomerealworldentity.
- Attribute−arealworldroleplayedbyanameddomain.
- Domain−asetofatomic values.
- Set−amathematicaldefinitionforacollectionofobjectswhichcontainsnoduplicates.

Unaryoperations:Operatesononerelation.Theseinclude:

- *Selection*
- *Projection*
- *Renaming*

SelectionOperation(σ):

Itselectstuplesthatsatisfythegivenpredicatefroma relation.

Notation−$\sigma_p(r)$

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like − =, ≠, ≥, <, >,≤.

Forexample−

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

Output−Selectstuplesfrombookswheresubjectis 'database'.

$\sigma_{subject="database"andprice="450"oryear>"2010"}(Books)$

Output−Selectstuplesfrombookswheresubjectis'database'and'price'is450or those books published after 2010.

ProjectOperation(∏):

Itprojectscolumn(s)thatsatisfyagivenpredicate.

Notation − $\prod_{A1, A2, An}(r)$ Where A_1,A_2, $A_n$are attribute names of relation **r**.

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

Forexample−

$\prod_{subject,author}(Books)$

Selectsandprojectscolumnsnamed assubjectandauthorfromtherelationBooks.

RenameOperation(ρ):

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

Notation−$\rho_x(E)$

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

Binary operations: Operates on pairs of relation. These include:

- Union
- Set Difference
- Cartesianproduct
- Division
- Joins

UnionOperation(□):

Itperformsbinaryunionbetweentwogivenrelationsandisdefinedas−

r□s={t |t∈rort∈s}

Notation−:rUs

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

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

- **r**,and**s**musthave thesamenumberofattributes.
- Attributedomainsmustbecompatible.
- Duplicatetuplesareautomaticallyeliminated.

SetDifference(−):

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

Notation−:(**r−s)**

Findsallthetuplesthatare presentin**r**butnotin**s**.

∏author(Books)−∏author(Articles)

Output−Providesthenameofauthorswhohavewrittenbooksbutnotarticles.

CartesianProduct(X):

Combinesinformationoftwodifferentrelationsintoone.

Notation−rX s

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

r X s = { q t | q ∈ r and t ∈ s}

σauthor='tutorial'(BooksXArticles)

Output−Yieldsarelation,whichshowsallthebooksandarticles writtenbytutorial.

Division:

The division operator divides a dividend relation A of degree (means number of columns in a relation) m + n by a divisor relation B of degree n, and produces a\ resultant relation ofdegree m. It is denoted by ÷.
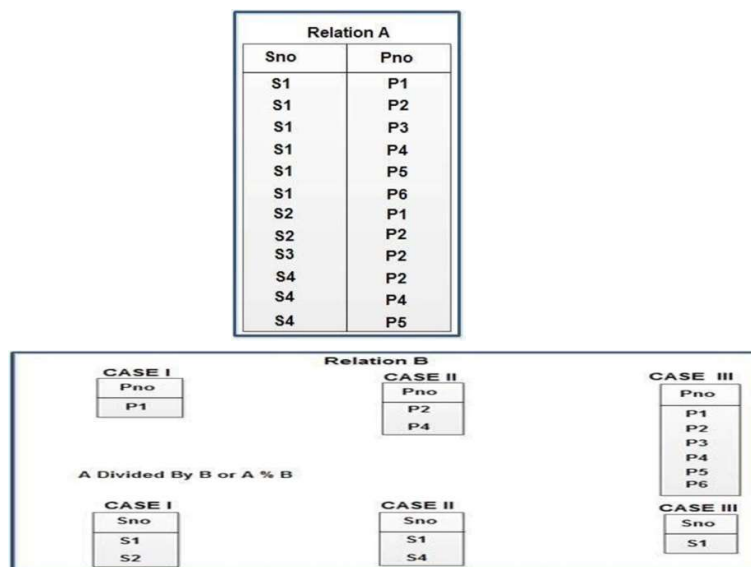
In this example dividend relation A has two attributes of Sno, Pno (of degree 2) and division relation B has only one attribute Pno (of degree 1). Then, A divide by B gives a resultant relation of degree 1. It means it has only one attribute of Sno.

A=SNO*SNO=SNO.B   =   PNO

The resultant relation has those tuples that are commonvalues of those attributes, which appears in the resultant attribute sno. *For* example, in CASE II,

      P2has Snos.--Sl,S2,S3,S4   P4hasSnos.--SI,S4

S1, S4 are the common supplier who supply both P2 and P4. So the resultant relation has tuples S1 and S4.

InCASEIII:There isonlyonesupplierS1who supplyallthepartsfromPI toP6.

Joins:

      We understand the benefits of taking a Cartesian product of two relations, which givesus all the possible tuples that are paired together. But it might not be feasible for us in certain cases to take a Cartesian product where we encounter huge relations with thousands of tuples having a considerable large number of attributes.

Join is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples fromdifferent relations, if andonly if agiven join conditionis satisfied.

Wewillbrieflydescribevariousjointypes inthefollowingsections.

Theta(θ)Join:

Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol **θ**.

Notation

$$R1 \bowtie_\theta R2$$

R1andR2arerelationshavingattributes(A1,A2,..,An)and(B1,B2,..,Bn)suchthatthe attributes don't have anything in common, that is R1 ∩ R2 = Φ.

Thetajoincanuseallkindsofcomparisonoperators.

Student

SID	Name	Std
101	Alex	10
102	Maria	11

Subjects

Class	Subject
10	Math
10	English
11	Music
11	Sports

Student_Detail−

$$STUDENT \bowtie_{Student.Std=Subject.Class} SUBJECT$$

51

Student_detail

SID	Name	Std	Class	Subject
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

Equijoin:

When Theta join uses only **equality** comparison operator, it is said to be equijoin. The above example corresponds to equijoin.

NaturalJoin(⋈):

Natural joindoesnotuse anycomparisonoperator.Itdoesnotconcatenate theway a Cartesian product does. We can perform a Natural Join only if there is at least one common attributethatexistsbetweentworelations.Inaddition,theattributesmusthavethesame name and domain.

Natural join acts on those matching attributeswherethe values ofattributesin both the relations are same.

CS01	Database	CS
ME01	Mechanics	ME
EE01	Electronics	EE

CS	Alex
ME	Maya
EE	Mira

CS	CS01	Database	Alex
ME	ME01	Mechanics	Maya
EE	EE01	Electronics	Mira

Outer Joins: ThetaJoin,Equijoin,and Natural Joinare calledinnerjoins.Aninnerjoin includes only those tuples with matching attributes and the rest are discarded in the resulting relation. Therefore, we need to use outer joins to include all the tuples from the participating relations in the resulting relation. There are three kinds of outer joins − left outer join, rightouter join, and full outer join.

LeftOuterJoin(R ⋈ S):

All the tuplesfromtheLeftrelation,R,are includedintheresulting relation.If there are tuples in R without any matching tuple in the Right relation S, then the S-attributes of the resulting relation are made NULL.

Left

A	B
100	Database
101	Mechanics
102	Electronics

Right

A	B
100	Alex
102	Maya
104	Mira

Courses ⋈HoD

A	B	C	D
100	Database	100	Alex
101	Mechanics	---	---
102	Electronics	102	Maya

RightOuterJoin:(R ⋈ S):

AllthetuplesfromtheRightrelation,S,areincludedintheresultingrelation.Ifthereare

tuplesinSwithoutanymatchingtupleinR,thentheR-attributesofresultingrelationare made NULL.

Courses ⋈HoD

A	B	C	D
100	Database	100	Alex
102	Electronics	102	Maya

DeptofIT,NRCM

FullOuterJoin:(RS)⋈

Allthetuplesfrombothparticipatingrelationsareincludedintheresultingrelation.Ifthere arenomatchingtuplesforbothrelations,theirrespectiveunmatchedattributesaremade NULL.

Courses	HoD		
A	B	C	D
100	Database	100	Alex
101	Mechanics	---	---
102	Electronics	102	Maya
---	---	104	Mira

UNIT-3

ExamplesofBasic SQLQueries:

SQL is a language which is used to interact with relational database management system. Before we learn what is a SQL and what we can do with the SQL, lets learn the basics first.

What is SQL?

1. SQL stands for Structured Query Language, which is a standardised language for interacting with RDBMS (Relational Database Management System). Some of the popular relational database example are: MySQL, Oracle, mariaDB, postgreSQL etc.

2. SQL is used to perform C.R.U.D (Create, Retrieve, Update & Delete) operations on relational databases.

3. SQL can also perform administrative tasks on database such as database security,backup, user management etc.

4. WecancreatedatabasesandtablesinsidedatabaseusingSQL.

Typesof StructuredQueryLanguage(SQL):

In the above section, we learned what we do with the database using SQL. SQL is basically combination of four different languages, they are –

DQL (DataQueryLanguage):   DQLisusedto fetchtheinformationfromthedatabase which is already stored there.

DDL(DataDefinitionLanguage):DDLisusedtodefine tableschemas.

DCL (Data Control Language): DCL is used for user & permission management. It controls the access to the database.

DML (Data Manipulation Language): DML is used for inserting, updating and deleting data from the database.

SQLCREATEDatabase:

TheSQL CREATE DATABASE statement is used to create new SQL database.

Syntax: Basic syntax of CREATE DATABASE statement is as follows:

        CREATEDATABASEDatabaseName;

Always database name should be unique within the RDBMS.

Example:

If you want to create new database <testDB>, then CREATE DATABASE statement would be as follows:

        SQL>CREATEDATABASEtestDB;

Makesureyouhaveadminprivilegebeforecreatinganydatabase.Onceadatabaseis created, you can check it in the list of databases as follows:

SQL>SHOWDATABASES;

SQLCREATETable:

Creatingabasictableinvolvesnamingthetableanddefiningitscolumnsandeach column's data type.

TheSQL CREATE TABLE statement is usedtocreate a newtable.

Syntax:

Basic syntax of CREATE TABLE statement isas follows:

    CREATETABLEtable_name(column1datatype,

                          column2datatype,

                          column3datatype,

                          .....

                          columnNdatatype,

                          PRIMARYKEY(oneormorecolumns)

                          );

CREATETABLEisthekeywordtellingthedatabasesystemwhatyouwanttodo.Inthis

case,youwanttocreateanewtable.Theuniquenameoridentifierforthetablefollows   the   CREATE

TABLE statement.

Theninbracketscomesthelistdefiningeachcolumninthetableandwhatsortofdata type it is. The

syntax becomes clearer with an example below.

A copy of an existing table can be created using a combination of the CREATE TABLE

statementandtheSELECTstatement.YoucancheckcompletedetailsatCreateTable Using another

Table.

CreateTableUsinganother Table:

AcopyofanexistingtablecanbecreatedusingacombinationoftheCREATETABLE statement and the

SELECT statement.

Thenewtablehasthesamecolumndefinitions.Allcolumnsorspecificcolumnscanbe selected.

Whenyoucreateanewtableusingexistingtable,newtablewouldbepopulatedusing existing values

in the old table.

Syntax:

The basic syntax for creating a table from another table isas follows:

    CREATE TABLE NEW_TABLE_NAME AS

        SELECT[column1,column2...columnN]

         FROMEXISTING_TABLE_NAME

        [WHERE]

Here, column1, column2...are the fields of existing table and same would be used to create fields of new table.

Example:

Following is an example, which would create a table SALARY using CUSTOMERS table and having fields customer ID and customer SALARY:

SQL>CREATETABLESALARYASSELECTID, SALARYFROMCUSTOMERS;

Thiswouldcreatenewtable SALARY,whichwouldhavethefollowingrecords:

SQL DROPorDELETETable:

The SQL DROP TABLE statement is used to remove a table definition and all data, indexes, triggers, constraints, and permission specifications for that table.

NOTE:Youhavetobecarefulwhileusingthiscommandbecauseonceatableisdeleted then all the information available in the table would also be lost forever.

Syntax:

Basic syntax of DROPTABLEstatement is as follows:

    DROP TABLE table_name;

Example:

Let us first verify CUSTOMERS table andthen we woulddelete it from the database: SQL> DESC CUSTOMERS;

```
+............+.................+.........+....+...........+..........+
|Field|Type|Null|Key|Default|Extra|
+............+.................+.........+....+...........+..........+
|ID|int(11)|NO|PRI|||
|NAME |varchar(20)|NO||| |
|AGE| int(11) |NO| |||
|ADDRESS| char(25)|YES||NULL||
|SALARY|decimal(18,2)|YES||NULL||
+............+.................+.........+....+...........+..........+
5 rows in set (0.00 sec)
```

This means CUSTOMERS table is available in the database, so let us drop it as follows:

SQL> DROP TABLE CUSTOMERS;

    QueryOK,0rowsaffected(0.01sec)

Now, if you would try DESC command, then you would get error as follows:

SQL> DESC CUSTOMERS;

    ERROR1146(42S02):Table'TEST.CUSTOMERS'doesn'texist

SQLINSERTQuery:

TheSQLINSERTINTOStatementisusedtoaddnewrowsofdatatoatableinthe database.

Syntax:

TherearetwobasicsyntaxesofINSERTINTOstatementasfollows:

> INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)]
>
> > VALUES (value1, value2, value3,...valueN);

Here,column1,column2,...columnNarethenamesofthecolumnsinthetableintowhich you want to insert data.

You may not need to specify the column(s)name in the SQL query if you are adding values forallthecolumnsofthetable.Butmakesuretheorderofthevaluesisinthesameorder as the columnsin the table. The SQL INSERT INTO syntax would be asfollows:

> INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);

Example:

Following statements would create six records in CUSTOMERS table:

> INSERTINTOCUSTOMERS(ID,NAME,AGE,ADDRESS,SALARY)
>
> > VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);
>
> INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
>
> > VALUES(2,'Khilan',25,'Delhi',1500.00);

we can create a record in CUSTOMERS table using second syntax as follows:

> INSERT INTO CUSTOMERS VALUES (7, 'Muffy', 24, 'Indore', 10000.00);

SQLSELECTQuery:

SQL SELECT Statement is used to fetch the data from a database table which returns

datain the form of result table. These result tables are called result-sets.

Syntax:

ThebasicsyntaxofSELECTstatementisasfollows:

> SELECTcolumn1,column2,columnNFROMtable_name;

Here, column1, column2...are the fields of a table whose values you want to fetch. If youwantto fetch all the fields available inthe field, thenyou canuse the following syntax:

> SELECT*FROMtable_name; Example:

Following is an example, which would fetch ID, Name and Salary fields of the customers available in CUSTOMERS table:

SQL>SELECTID,NAME,SALARYFROMCUSTOMERS;

SQLWHEREClause:

The SQL WHEREclause is used tospecifya conditionwhile fetching thedatafrom single table or joining with multiple tables.

Ifthegivenconditionissatisfied,thenonlyitreturnsspecificvaluefromthetable.You would use WHERE clause to filter the records and fetching only necessary records.

The WHEREclauseis notonlyusedinSELECTstatement,butitisalsousedinUPDATE, DELETE statement, etc., which we would examine in subsequent chapters.

Syntax:

The basic syntax of SELECT statement with WHERE clause is as follows:

    SELECT column1, column2, column FROM table_name

        WHERE[condition]

You can specify a condition using comparison or logical operators like >, <, =, LIKE, NOT etc. Below examples would make this concept clear.

SQLANDandOROperators:

The SQL AND and OR operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called conjunctive operators.

These operators provide a means to make multiple comparisons with different operators in the same SQL statement.

TheANDOperator:

The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.

Syntax:

The basic syntax of AND operator with WHERE clause is as follows:

    SELECT column1, column2, columnN

        FROMtable_name

        WHERE[condition1]AND[condition2]...AND[conditionN];

Example:

    SQL>SELECTID, NAME,SALARY

        FROM CUSTOMERS

        WHERESALARY>2000ANDage<25;

SQLUPDATEQuery:

You can use WHERE clause with UPDATE query to update selected rows, otherwise all the rows would be affected.

Syntax:

The basic syntax of UPDATE query with WHERE clause is as follows:

UPDATE table_name

SETcolumn1=value1,column2=value2……………,columnN=valueN

WHERE[condition];

You can combine N number of conditions using AND or OR operators.

Example:

Following is an example, which would update ADDRESS for a customer whose ID is 6:

SQL> UPDATE CUSTOMERS

SETADDRESS='Pune' WHERE

ID = 6;

SQLDELETEQuery

TheSQLDELETEQueryisused todeletetheexistingrecords fromatable.

Youcanuse WHEREclausewithDELETE query to deleteselected rows, otherwiseall the records would be deleted.

Syntax:

Thebasic syntax of DELETE query with WHERE clauseis as follows:

DELETE FROM table_name

WHERE[condition];

You can combine N number of conditions using AND or OR operators.

Example:

Following is an example, which would DELETE a customer, whose ID is 6:

SQL> DELETE FROM CUSTOMERS

WHEREID=6;

SQL LIKEClause:

The SQL LIKE clause is used to compare a value to similar values using wildcard operators.

There are two wildcards used in conjunction with the LIKE operator:

Thepercentsign(%) The

underscore (_)

The percent sign represents zero, one, or multiple characters. The underscore represents a single number or character. The symbols can be used in combinations

Syntax:

Thebasic syntax of %and_ is as follows:

SELECT FROM table_name

WHEREcolumnLIKE'XXXX%'

or

SELECTFROMtable_name

WHEREcolumnLIKE'%XXXX%'

or

SELECTFROMtable_name

WHEREcolumnLIKE'XXXX_'

or

SELECTFROMtable_name

WHEREcolumnLIKE'_XXXX'

or

SELECTFROMtable_name

WHEREcolumnLIKE'_XXXX_'

You can combine N number of conditions using AND or OR operators. Here, XXXX could be any numeric or string value.

Example:

SQL>SELECT*FROMCUSTOMERS

WHERESALARYLIKE'200%';

SQLORDERBYClause:

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

Syntax:

ThebasicsyntaxofORDERBYclause isasfollows:

SELECT column-listFROM table_name [WHERE

condition][ORDERBY column1, column2, .. columnN] [ASC

| DESC];

You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort, that column should be in column-list.

Example:Followingisanexample,whichwouldsorttheresultindescendingorderby NAME:

SQL> SELECT*FROMCUSTOMERSORDERBYNAMEDESC;

SQLGroupBy:

The SQL GROUP BY clause is used in collaborationwith the SELECT statement to arrange identical data into groups.

The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

Syntax:

The basic syntax of GROUP BY clause is given below. The GROUP BY clause must follow the conditions in the WHERE clause and must precede the ORDER BY clauseifoneisused.

```
SELECT column1, column2
        FROM table_name
        WHERE[conditions]
        GROUPBYcolumn1, column2
        ORDERBYcolumn1,column2
```

Example:

```
SQL> SELECT NAME, SUM(SALARY) FROM CUSTOMERS
            GROUP BY NAME;
```

SQLDistinctKeyword:

The SQLDISTINCTkeywordisusedinconjunctionwith SELECTstatementtoeliminate all the duplicate records and fetching only unique records.

Theremaybeasituationwhenyouhavemultipleduplicaterecordsinatable.While fetching such records, it makes more sense to fetch only unique records instead of fetching duplicate records.

Syntax:

The basic syntax of DISTINCT keyword to eliminate duplicate records is as follows:

```
SELECT DISTINCT column1, column2,………columnN
        FROM table_name
        WHERE[condition]
```

Example:

First, let us see how the following SELECT query returns duplicate salary records:

```
SQL> SELECT SALARY FROM CUSTOMERS
            ORDER BYSALARY;
```

SQLSORTINGResults:

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

Syntax:

The basic syntax of ORDER BY clause which would be used to sort result in ascending or descending order is as follows:

SELECTcolumn-list

FROM table_name

[WHEREcondition]

[ORDERBYcolumn1,column2,..columnN][ASC|DESC];

You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort,that column should be in column-list.

Example:

SQL>SELECT*FROMCUSTOMERS

ORDERBYNAME,SALARY;

NestedQueries andsub queries:

ASubqueryorInnerqueryoraNestedqueryisaquerywithinanotherSQLqueryand embedded within the WHERE clause.

Asubqueryisusedtoreturndatathatwillbeusedinthemainqueryasaconditionto further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

Syntax:

selectcolumnlist

fromtablename

where column operator (select column from tablename

where column operator …..));

Thereareafewrulesthatsubqueriesmustfollow:−

- Subqueriesmustbeenclosedwithinparentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY command cannot be used in a subquery, although the main query can useanORDERBY.TheGROUPBYcommandcanbeusedtoperformthesame

functionastheORDERBYina subquery.

- Subqueriesthatreturnmorethanonerowcanonlybeusedwithmultiplevalue operators such as the IN operator.
- Asubquerycannotbeimmediatelyenclosedinasetfunction.
- TheBETWEENoperatorcannotbeusedwithasubquery.However,theBETWEEN operator can be used within the subquery.

➢ Asubquerycanapartofcolumnintheselectstatementiscalled<u>SUBSET.</u>

➢ A <u>INLINE VIEW</u> is a select statement in the FROM clause of another select statement

Example:

  select * from (select deptno , count(*) emp_count

    from emp

    group by deptno) emp , dept

    wheredept.deptno = emp.deptno;

➢ A subquery in the where clause of the select statement is called <u>NESTED SUBQUERY</u>Example:

  select*fromemp

    where sal>(selectsalfromempwhereempno=7566);

Types ofsubqueries:

I. Single row subquery: The query returns only one row from the inner   select statement. The operators used in this query are >, <, =, <=, >=.

 Example:

 Display the emp details whose salary is equal to minimum salary in company.

  select ename,job,sal from emp

   wheresal=(selectmin(sal)fromemp);

II. Multiple row subquery: The query returns more than one row from inner select statement. The operators used in these queries are ANY, ALL, IN.

 Example:

 Display who are getting minimum salary of each dept

  select ename, sal from emp

   wheresalin(selectmin(sal)fromempgroupby deptno);

III. Multiple column subquery: The query returns more than one column from the inner select statement. The column comparison in a multiple column subquery can be 1) pair wise comparison   2) Non-pair wise comparison.

IV. Correlated subquery: It is another way of performing queries upon the data with simulationofjoins.Inthisinformationfromouterselectstatementparticipates

64

astheconditionintheinnerselectstatement.Theoperatorsusedinthese queries are exists, not exists.

Stepstobeperformedincorrelatedsubqueries:

1. Firsttheouterqueryisexecuted.
2. Passesthequalifiedcolumnvaluetotheinnerqueries"where"clause.
3. Then the inner query orcandidate query is executed and the result is passed is passed to the outer query where clause.
4. Depending upon the separate values the condition is qualified for the specific record.

Example: Display deptno, dept name there exist at least one employee in dept

    select d.deptno,d.dname from dept d
            whereexist(select*fromempewhered.deptno =e.deptno);

Set Comparison Operators: We have seen the set comparison operators EXISTS, IN &UNIQUE along with their negation. ALL,ANY are the arithmeticoperators (<,>, <=,>=, <>,=).

Example: Find sailors whose rating is better than some sailor called horatio

    select s.sid from sailors s where s.rating > any(select s2.rating
                        fromsailorss2wheres2.name='horatio');

Aggregative Operators: The aggregative functions are used in the subqueries like max(), min(), avg()…. It is used in the in the multiple row subqueries because in thesequeries returns more than one row for the outer query.

Example: Display who are getting the minimum salary of each department.

    select ename, sal from emp
            wheresalin(selectmin(sal)fromempgroupbydeptid);

NULLVALUES:

SQL provides a special column value called null to use null when the column value is unknown or inapplicable.

Example: create table student (sid integer constraint student_sid_nn not null,
                name char(20));

Intheaboveexamplesiddoesn'tallowthenullvaluesinthecolumn.

Impact on SQL Constructs: BooleanexpressionsariseinmanycontextsinSQL,andthe impactofnullvaluesmustberecognized.Inthepresenceofnullvalueanyrowthat

evaluatestofalseorunknowniseliminatedsothatthereisanimpactonthenested queries.
Thearithmeticoperatorsallreturnnullifoneoftheirargumentsisnull.Theaggregate operators discard the null values.

OuterJoins:

Someinterestingvariantsofthejoinoperationthat relyonnull valuescalledouterjoins.

LeftOuter Join:

Toretrievethemissingrecordsfromtheleftsidetableofthejoincondition.

Example:Displayname,deptno,deptnamefromtables select

e.ename, d.deptno, d.dname from empe, dept d

wheree.deptno=d.deptno(+);

RightOuterJoin:

To retrieve the missing records from the right side table of the join condition.

Example: Display name, dept name, dept no from tables.

select e.ename,e.deptno,d.dname   fromempe,deptd

where e.deptno(+) = d.deptno;

Full OuterJoin:

Toretrievethemissingrecordsfromthebothsidetableofthejoincondition.

ComplexIntegrityConstraintsinSQL:

Constraintsoversingletable:

Constraints can specify over a single table using table constraints by CHECK conditional expression.

Domain Constraintsanddistincttypes:

A user can define a new domain using the CREATE DOMAIN statement which uses CHECK constraints.

Triggers:

Atriggerisaprocedurethatisautomaticallyinvokedbythe DBMSin responsetospecialchangestothedatabaseand DBA. Adatabasethathasasetof associated triggers is called an ACTIVE database.

- Event: Achangestothedatabasethatactivatesthetrigger.
- Condition:Aqueryortestthatisrunwhenthe triggerisactivated.
- Action:Aprocedurethatisexecutedwhenthetriggerisactivatedandthe condition is true.

A trigger can be thought of as a daemon that monitors a database and executed when the database is modified in a way that matches the event specification

Eg: Write a trigger to display salary difference of old and new values.

```
set serveroutput on;
createorreplace triggeremp_diff
after insert or delete or update on emp        foreachrow
declare diff number;
begin
    diff := :new.sal – :old.sal;
    dbms_output.put_line('old salary': || :old.sal);
    dbms_output.put_line('new salary': || :new.sal);
    dbms_output.put_line(' salary difference': || diffl);
end;
```

Examplesofsubqueries

1. Displaythe detailsofallemployees
2. Displaythedepartinformationfromdepartmenttable
3. Display thenameandjobforalltheemployees
4. Displaythenameandsalaryforallthe employees
5. Displaytheemployeeno andtotalsalaryforalltheemployees
6. Displaytheemployeenameandannualsalaryforallemployees.
7. Displaythenamesofalltheemployeeswho areworkingindepartnumber10.
8. Display the names of all the employees who are workingas clerks anddrawing a salary more than 3000.
9. Displaythe employeenumberandnamewhoareearningcomm.
10. Displaytheemployeenumberandnamewhodonotearnanycomm.
11. Display the names of employees who are working as clerks, salesman or analyst and drawing a salary more than 3000.
12. Display the names of the employees who are working in the companyfor the past 5 years;
13. Display the list of employees who have joined the company before 30-JUN-90 or after 31-DEC-90.
14. Listtheempswhoarejoinedinthe year1981
15. ListtheempswhoarejoinedinthemonthofAug1980

16. Listtheempswhoseannulsalranging from22000and45000
17. Display the names of employees working in depart number 10 or 20 or40 or employees working as CLERKS, SALESMAN or ANALYST.
18. DisplaythenamesofemployeeswhosenamestartswithalphabetS.
19. Display the EmployeenamesforemployeeeswhosenameendswithAlphabetS.
20. Display the names of employees whose names have second alphabet A in their names.
21. Selectthenamesoftheemployeewhosenamesisexactlyfivecharactersinlength.
22. Displaythenamesoftheemployeewhoarenotworking asMANAGERS.
23. Display the names of the employee who are not working as SALESMAN OR CLERK OR ANALYST.
24. Displaythetotalnumberofemployeeworkinginthecompany.
25. Display thetotalsalarybeingpaidtoallemployees.
26. Display the maximum salary and maximum salary and average salary from emp table.
27. DisplaythemaximumsalarybeingpaidtoCLERK.
28. Displaythemaximumsalarybeingpaidtodepartnumber20.
29. Displaytheminimumsalarybeing paid toanySALESMAN.
30. DisplaytheaveragesalarydrawnbyMANAGERS.
31. DisplaythetotalsalarydrawnbyANALYSTworkingindepartnumber30
32. Display empno,ename,deptno,sal sort the output first base on name and within name by deptno and with in deptno by sal.
33. Display department numbers and total number of employees working in each department.
34. Displaythevariousjobsandtotalnumberofemployeeswithineachjobgroup.
35. Displaythe departnumbersandtotalsalaryforeachdepartment.
36. Displaythe departnumbersandmaxsalaryforeachdepartment.
37. Display thevariousjobsandtotalsalaryforeachjob
38. Displaythedepartnumberswithmorethanthreeemployeesineachdept.
39. Display the various jobs along with total salary for each of the jobs wheretotal salary is greater than 40000.
40. Display the various jobs along with total number of employees in each job. The output should contain only those jobs with more than three employees.
41. Displaythejobsfoundindepartment10and20Eliminateduplicatejobs.
42. Displaythedistinctjobsinthedepartment10.

43. Display those department whose name start with "S" while the location name ends with "K".

44. Display those employees whose salary is more than 3000 after giving 20% increment.

45. Displaythoseemployeeswhoarenotworkingunderany manager.

46. Selectcountofemployeeineachdepartmentwherecountgreaterthan3?

47. List out employee name and salary increased by 15% and expressed as whole number of Dollars?

48. Findoutthe averagesalaryandaverage totalremunerationforeachjobtype

49. Listtheempswhoseannulsalrangingfrom22000and45000

50. ListtheempswhojoinedinJanuary

51. selectsecond maxsalaryfromemp

SchemaRefinementandNormalforms

Schema:

A Schema can be defined as a complete description of database. The Specifications for database schema is provided during the databasedesignand this schema does not change frequently.

SchemaRefinement:

Schema Refinement is a process of refining the schema so as to solve the problems caused by redundantly storing the information.

Redundancy means duplication of data. Redundancy is at the root of several problems associated with relational schemas. Some of them are

➢ Redundantstorage,

➢ Insert/delete/updateanomalies.

Figure4.1

ProblemscausedbyRedundancy:

DataInconsistency

Memory Fragmentation

Anomalies:

Therearethreetypesofanomalies:

i.   update,
ii. deletionand
iii. insertionanomalies.

Letusconsideranexample:

Eachemployeeinacompanyhasadepartmentassociatedwiththemaswellasthestudent group they participate in.

Employee_ID	Name	Department	Student_Group
123	James	Accounting	Pstclub
234	B.Rech	Marketing	MarketingClub
234	B. Rech	Marketing	ManagementClub
456	Anand	CIS	TechnologyOrg.
456	Anand	CIS	Pstclub

UpdateAnomaly:

Anupdateanomalyisadatainconsistencythatresultsfromdataredundancyandapartial update.
For example, if Anand's department is an error it must be updated at least 2 times or there will be inconsistent data in the database. If the user performing the update does not realize the data is stored redundantly the update will not be done properly.

Deletionanomaly:

Adeletionanomalyisthe unintended lossofdataduetodeletionofotherdata.

For example, if the student group Pst club is disbanded and was deleted from the table above, JamesandtheAccountingdepartmentwouldceasetoexist.Thisresultsindatabaseinconsistencies and is an example of how combining information that does not really belong together into one table can cause problems.

Insertionanomaly:

An insertionanomalyistheinabilitytoadd datatothedatabaseduetoabsenceofotherdata.

For example, assume Student_Group is defined so that null values are not allowed. If a new employee is hired but not immediately assigned to a Student_Group then this employee could not be entered into the database. This results in database inconsistencies due to omission.

Update, deletion, and insertion anomalies are very undesirable in any database. Anomalies are avoided by the process of normalization.

WaystoavoidDataAnomalies:

Therearetwowaystoavoiddataanomalies.Theyare:

1. Normalization
2. Decomposition

FunctionalDependency:

AFunctionaldependencyisdefinedastherelationshipbetweentheattributesthatcorrespond to a single relation.

Afunctionaldependency(FD)hastheformX->Y(readasXfunctionallydeterminesY)   where   Xand Yaresetsofattributesinarelation R. HereXis usedto determine thevalueofY,soitis said that Y is functionally dependent on X.

Example:Astudent canhaveonlyonebirthyear:**S→B**

FunctionalDependenciesinentitysets:

Anentitysetcanhavethefollowingfunctionaldependencies:

Fully FunctionalDependency:

If x and y areattributes of an entity set in atable such that yis functionallydependent only on x, but not on any proper subset of x,then this type of dependency is called as Fully Functional Dependency.

Eg:RollNo,SubName->Marks.

PartialFunctionalDependency:

If x and y areattributes of an entity set in atable such that yis functionallydependent onlyonxandeliminationofsomeattributesfromxdoesnotaffectthedependency,then   this   type   of dependency is called as Partial Functional Dependency.

Eg:emp_id,emp_name->salary.

TransitiveFunctionalDependency:

If x,y,z are attributes of an entity set in a table such that x is functionally dependent on y and yisfunctionallydependent on z, then zwill be transitivelydependentonx through y.

Eg: Students -> Teachers

Teachers->Management

Management->Students

Reasoning aboutFD's:

Armstrong'sAxioms:

WilliamW. Armstrongestablished asetofrules whichcan beused toinfer thefunctional dependencies in a relational database (from umbc.edu - no external linking, Google Database Design UMBC):

- **Reflexivityrule**:

If A is a set of attributes, and B is a set of attributes that are completely contained in A, then A implies B.

- **Augmentationrule:**

IfAimpliesB,and Cisa setofattributes,thenifAimpliesB,thenACimpliesBC.

- **Transitivityrule:**

    IfA implies B andB implies C,thenA implies C.

These can be simplified if we also use:

- **Unionrule:**If AimpliesBandAimpliesC, theAimpliesBC.

- **Decompositionrule**:If AimpliesBCthenAimpliesBandAimpliesC.

- **Pseudotransitivityrule**:IfAimpliesBandCBimpliesD,thenACimpliesD.

Normalization:

Normalization is a relational database management system design concept which is a process of designing thedatabasestructure suchthatitminimizesthedata redundancy and alsodataanomalies.

The process of normalization includes a series of stages known as Normal Forms. Normalization rules are divided into following normal form :

1. FirstNormalForm(1NF)
2. Second NormalForm(2NF)
3. ThirdNormalForm(3NF)
4. Boyce-CoddNormalForm(BCNF)

FirstNormalForm(1NF):

Arelationis infirstNormalFormifandonlyifallunderlyingdomainscontainatomicvaluesonly. Inotherwords,arelationdoesn'thavemultivalveattributes.

Forexample:

Considera**STUDENT(Sid,Sname,Cname)**relation.

Student:

SID	Sname	Cname
S1	A	C,C++
S2	B	C++,DB
S3	A	DB

SID:PrimaryKey

⇐

DuetooccurrenceofMVA, theaboverelationisnotin1NF.

	Solution:
□	RemovalofMVAbyinsertingmorerows

Student:

SID	Sname	Cname
S1	A	C
S1	A	C++
S2	B	C++
S2	B	DB
S3	A	DB

SID:PrimaryKey

⇐ Therelationisin1NF

4.6.22NF−SecondNormalForm:

Relation**R**isin SecondNormalForm(2NF) onlyiff:

1. **R**shouldbe in1NFand
2. *R*shouldnotcontainany*PartialDependency*

PartialDependency:
Let **R** be a relational Schema and **X,Y,A** be the attribute sets over **R**.
X: Any Candidate Key
Y:ProperSubsetofCandidate Key
A:NonKeyAttribute
IfY → A exists in R, then R isnot in2 NF.
(Y →A) is a Partial dependency only if**Y:**
Proper subset of Candidate Key
A:NonPrime Attribute

Removalof PartialDependency

Ifthereisanypartialdependency,removepartiallydependentattributesfromoriginaltable, place them in a separate table along with a copy of its determinant.

Example1:Considertherelation

*Student(SID,Sname,Cname)*whichisin1NF(NoMulti-Valued-Attributes):

Student:

SID	Sname	Cname
S1	A	C
S1	A	C++
S2	B	C++
S2	B	DB
S3	A	DB

{SID,Cname}: PrimaryKey

FunctionalDependencies:

{SID,Cname}→ Sname

SID →Sname
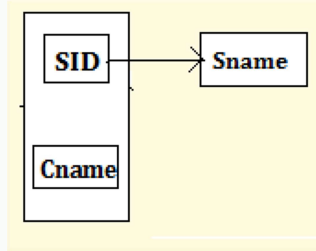
PartialDependencies:

SID→Sname {asSIDisa Proper Subset of Candidate Key {SID,Cname}.

Solution: RemovalofPartialDependencybycreatingseparatetable

SID

:Primary Key

R2:

SID	Cname
S1	C
S1	C++
S2	C++
S2	DB
S3	DB

{SID,Cname}: PrimaryKey

TheabovetworelationsR1andR2are LosslessJoinandDependencyPreserving. Theywerein2NF

*Thereislessredundancyin2NFrather thanin1NF, but2NFisnotfreefromredundancy.

3NF–ThirdNormalForm:

Let*R*be therelationalschema,*R*isin3NFonlyif:

- *R*shouldbein<mark>2NF</mark>.
- *R*shouldnotcontain*transitivedependencies.*

TransitiveDependency

Let*R*bearelationalSchemaand*X,Y,Z*betheattributesetsover*R.*

If *X* is functionally dependent on *Y (X → Y)*

and*Y* is functionally dependent on *Z (Y → Z)*

then *X* is transitive dependent on *Z (X → Z)*
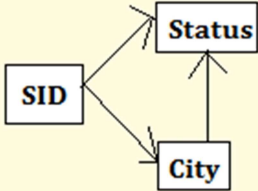
RemovalofTransitiveDependency

Ifthereisanytransitivedependencyintherelation, then

- Createaseparaterelationandcopythe dependentattributealongwithacopyofits determinant. and remove these determinants from the original table.
- Mark dependent attribute as a foreign key in the original relation and Mark dependent attribute as a Primary key in the separate relation

Exampleof3NF:

Letusconsiderthe relation**Sup_City(SID,Status,City)**:

S4	30	Delhi

Sup_City :

SID	Status	City

FDD of Sup_City :

TransitiveDependency:

SID→City{AsSID→CityandCity→Status}

Solution:

RemovalofTransitiveDependencybycreatingseparate table
□

TherelationsSCandCSarein 3NFastheydoesn'tcontainanytransitivedependencies.

4.6.4BCNF−BoyceCoddNormalForm:

Let*R*betherelationalschema,*R*isinBCNFonlyif:

- *R*shouldbein   3NF.

- Every Functional Dependency will have a Superkey on the LHS or all determinants are the superkeys.

Example:

Considerthefollowingrelationship***R(ABCD)***havingfollowingfunctionaldependencies:

F={A→BCD,BC→AD,D→B}

CandidateKeysare:
  (A)$^+$={ABCD}
  (BC)$^+$={BCAD}
  (DC)$^+$={DCBA}

Functional Dependency	IsFDinBCNFornot?	Reason?
A→BCD	Yes	Aisasuperkey
BC→AD	Yes	BCisalso asuperkey
D→A	No	Disnotsuperkey,itispartofkey

Solution:

DecompositioninBCNF:

Therelation**R(ABCD)**isdecomposedintotworelations**R1**and**R2**suchthat:

R1(A,D,C) and **R2(D,B).**

Theabovetworelations**R1**and**R2**

1. LosslessJoin
2. BCNFDecomposition
3. ButNotDependencyPreserving

RedundancyinBCNF

There will be 0% redundancy, because of Single Valued Functional Dependency. Redundancymayexistbecauseof**MultivalvedDependency.**

MultivaluedDependency

ConsiderarelationFaculty(FID,Course,Book)whichconsistsoftwomultivaluedattributes FID Course

	Book	alued attributes are independent of each other.
1	C1/C2	B1/B2
2	C1	B1

Differencesbetween3NFandBCNF:

S.NO	3NF	BCNF
1.	It concentratesonPrimaryKey	It concentratesonCandidateKey.
2.	RedundancyishighascomparedtoBCNF	0%redundancy
3.	Itmaypreserveallthedependencies	Itmay notpreservethedependencies.
4.	Adependency X → Yis allowed in 3NF ifX is a super key or Y is a part of some key.	AdependencyX→ YisallowedifXisa super key

<u>Decomposition</u>:

Decomposition is the processof dividing longer relations into smaller relations.This division is based on functional and otherdependencies which are specified by database designer.

The decomposition of a relation scheme R consists of replacing the relation schema by two or more relationschemasthateachcontainasubsetoftheattributesofRandtogetherincludeallattributesin R.

ProblemsrelatedtoDecomposition:

Decomposingarelationschemacancreatemoreproblemsunlesswehandlethemcarefully.

- Two important questions must beavoided repeatedly.They are:
    1. Do we need to decompose a relation ?
    2. Whatproblemsdoesagivendecompositioncause?
- To answer the first question, several normal forms have been proposed for relations. If a relation schema is in one of these normal forms, we know that certain kinds of problems cannot arise.
- To answer the second question, two properties of decomposition arte of particular interest :
    1. Loss-less join decomposition
    2. DependencyPreservingDecomposition

Losslessjoin Decomposition:

LossyDecomposition:

"The decomposition of relation R into R1 and R2 is **lossy** when the join of R1 and R2 does not yield the same relation as in R."

One of the disadvantages of decomposition into two or more relational schemes (or tables) is that some information is lost during retrieval of original relation or table.

ConsiderthatwehavetableSTUDENTwiththreeattributeroll_no,snameanddepartment.

STUDENT:

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

Thisrelationisdecomposedinto tworelationno_nameandname_dept:

No_name:

Roll_no	Sname
111	parimal

Sname	Dept
parimal	COMPUTER
parimal	ELECTRICAL

222	parimal

Name_dept:

In lossy decomposition ,spurious tuples are generated when a natural join is appliedto the relations in the decomposition.

stu_joined:

Roll_no	Sname	Dept
111	parimal	COMPUTER
111	parimal	ELECTRICAL
222	parimal	COMPUTER
222	parimal	ELECTRICAL

Theabovedecompositionis abaddecompositionorLossydecomposition.

LosslessJoinDecomposition:

"The decomposition of relation R into R1 and R2 is **lossless** when the join of R1 and R2 yield the same relation as in R."

A relational table is decomposed (or factored) into two or more smaller tables, in such a way that the designer can capture the precise content of theoriginal table by joining the decomposed parts. This is called lossless-join (or non-additive join) decomposition.

Thisisalsoreferredas**non-additive**decomposition.

The lossless-join decomposition is always defined with respect to a specific set F of dependencies.

Example:

Considerthatwehavetable STUDENTwiththreeattributeroll_no,snameanddepartment.

STUDENT:

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

ThisrelationisdecomposedintotworelationStu_nameandStu_dept:

Stu_name: **Stu_dept:**

Roll_no	Sname
111	parimal
222	parimal

Roll_no	Dept
111	COMPUTER
222	ELECTRICAL

 Now , when these two relations are joined on the common column 'roll_no' ,the resultant relation will look like stu_joined.

stu_joined:

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

 In lossless decomposition, no any spurious tuples are generated when a natural joined is applied to the relations in the decomposition.

Testsforlosslessjoindecomposition:

ToIdentifywhetheradecompositionislossyorlossless,itmustsatisfythefollowingconditions:

a) $R_1 \square R_2 = R$

   i. 2. $R_1 \cap R_2 \neq \Phi$ and

   ii. 3. $R_1 \cap R_2 \to R_1$ or $R_1 \cap R_2 \to R_2$

Example1:

LetR(ABC)F={A→B,A →C}decomposedintoD=R_1(AB),R_2(BC)FindwhetherDis Lossless or Lossy ?

Solution:

GivenD={AB, BC}

Step1: AB$\square$BC=ABC

Step2:AB∩BC=B   //Intersection

Step 3: B^+ = {B}   //Notasuperkey ofR_1orR_2

Therefore, the given Decomposition is lossy.

Example2:

LetR(ABCDEF)F={A→B,B→C,C→D,E→F}decomposedintoD=R_1(AB),R_2(BCD),R_3(DEF). FindwhetherD isLosslessorLossy?

Solution:

Step 1:

AB$\square$BCD$\square$ DEF=ABCDEF=R//Condition1satisfies step 2:

AB∩BCD=B

B^+={BCD}   //superkeyofR_2

⇒R_{12}(ABCD)

ABCD∩DEF=D

D^+ = {D}   //Nota superkey ofR_{12}or R_3

Therfore, the given decomposition is Lossy.

DependencyPreservingDecomposition:

Anotherpropertyofdecompositionis**DependencyPreservingDecomposition**.

Iftheoriginaltableisdecomposedintomultiplefragments,thensomehow,wesupposetoget   all   original FDs from these fragments. In other words, every dependency in original table must be preserved or say, every dependency must be satisfied by at least one decomposed table.

Let R be the original relational schema having FD set F. Let R1 and R2 having FD set F1 and F2 respectively, are the decomposed sub-relations of R. then there can be three cases:

➢ F1 U F2 = F   > Decomposition is dependency preserving.

➢ F1 U F2 is a subset of F   > Not Dependency preserving.

➢ F1 U F2 is a super set of F   > This case is not possible.

Example:

LetarelationR(A,B,C,D)andfunctionaldependency{AB–>   C,C–>   D,D–>   A}.Relation RisdecomposedintoR1(A,B,C)andR2(C,D).Checkwhetherdecompositionisdependency preserving or not.

Solution:

GivenR1(A,B,C) andR2(C,D)

LetusfindclosureofF1andF2:

To find closure of F1, consider all combination of ABC. i.e., find closure of A, B, C, AB, BC and AC. Note that ABC is not considered as it is always ABC

closure(A) = { A } // Trivial

closure(B) ={ B}//Trivial

closure(C)={C,A,D}butDcan'tbeinclosureasDis notpresentR1.

={C,A}

C-->A   //RemovingCfromrightsideasitis trivialattribute

closure(AB)={A,B,C,D}={A,B, C}

AB-->C//RemovingABfromrightsideas theseare trivialattributes

closure(BC)={B,C,D,A}={A,B,C}

BC-->A//RemovingBCfromrightsideasthesearetrivialattributes

closure(AC)={A,C,D}

AC-->D//RemovingAC fromrightsideasthesearetrivialattributes

F1{C-->A,AB-->C,BC-->A}.

SimilarlyF2{C-->D }

In theoriginalRelationDependency{AB-->C,C-->D,D-->A}.

AB-->CispresentinF1.

C-->DispresentinF2.

D-->Aisnotpreserved.

F1UF2is a subsetofF.

Therefore,thegivendecompositionis**notdependencypreserving**.

Unit-4

Transactionmanagement:

A **transaction** is a set of logically related operations. For example, you are transferring money from your bank account to your friend's account, the set of operations would be like this:

TheACIDProperties:

A transaction is a single logical unit of work which accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.

In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called ACID properties.
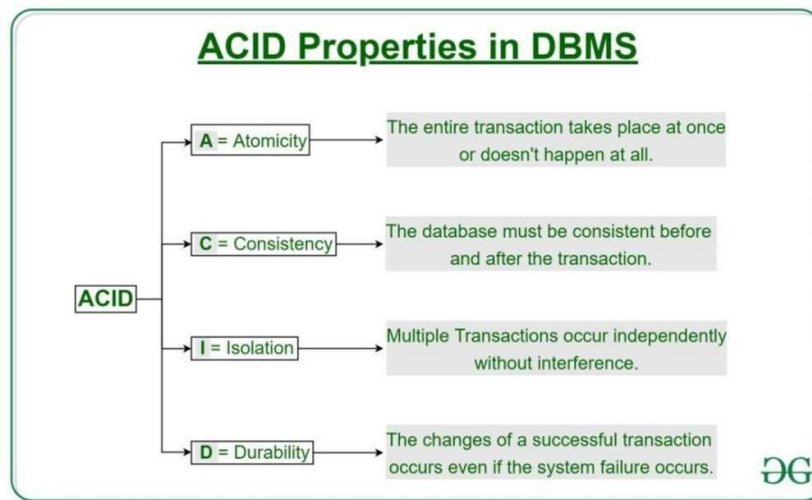
Figure5.1

Atomicity:

Bythis,wemean that eitherthe entiretransaction takesplaceat once ordoesn't happen at all. There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves the following two operations.

—Abort:Ifatransactionaborts,changesmadetodatabasearenotvisible.

—Commit:Ifatransactioncommits,changesmadearevisible.

Atomicityisalsoknownasthe'Allornothingrule'.

Consistency:

This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database. Referring to the example above,

Thetotalamountbeforeandafterthetransactionmustbemaintained. Total before T occurs = 500 + 200 = 700.

TotalafterToccurs=400+300=700.

Therefore,databaseisconsistent.InconsistencyoccursincaseT1 completesbutT2fails. As a result T is incomplete.

Isolation:

This property ensures that multiple transactions can occur concurrently without leadingto the inconsistency of database state. Transactions occur independently without interference. Changes occurringin a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed seriallyin some order.

Durability:

This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

TransactionsinDBMS:

A transaction is an action or series of actions that are being performed by a single useror application program, which reads or updates the contents of the database.

A transaction can be defined as a logical unit of work on the database. This may be an entire program, a pieceof a program, or a single command (like the SQL commands such as INSERT or UPDATE), and it may engage in any number of operations on the database. In the database context, the execution of an application program can be thought of as one or more transactions with non-database processing taking place in between.

ExampleofTransactioninDBMS:

Asimpleexampleofatransactionwillbedealingwiththe bankaccountsoftwousers,letsayKarlosandRay.A simpletransactionofmovinganamountof5000from KarlostoRayengagesmanylow-leveljobs.Astheamount ofRs.5000getstransferredfromtheKarlos'saccountto Ray'saccount,aseriesoftasksgetsperformedinthe backgroundofthescreen. Thisstraightforwardandsmalltransactionincludesseveral	Open_Acc(Karlos) OldBal=Karlos.bal NewBal=OldBal-5000 Ram.bal=NewBal CloseAccount(Karlos)

| steps:decreaseKarlos'sbankaccountfrom5000: | |

OpenAccount(Ray)   Old_Bal = Ray.bal   NewBal=OldBal+1000   Ahmed.bal = NewBal   CloseAccount(B)

we can say, the transaction involves many tasks, such as opening the account of Karlos, reading the old balance, decreasing the specific amount of 5000 from that account, saving new balance to an account of Karlos, and finally closing the transaction session.

For adding amount 5000 in Ray's account, the same sort of tasks needs to be done:

SchedulesinDBMS:

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.

Schedule − A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.

Serial Schedule − It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

TypesofSchedulesinDBMS

Schedule, as the name suggests, is a process of lining the transactions and executing them one by one. When there are multiple transactions that are running in a concurrent manner and the order of operation is needed to be set so that the operations do not overlap each other, Scheduling is brought into play and the transactions are timed accordingly.
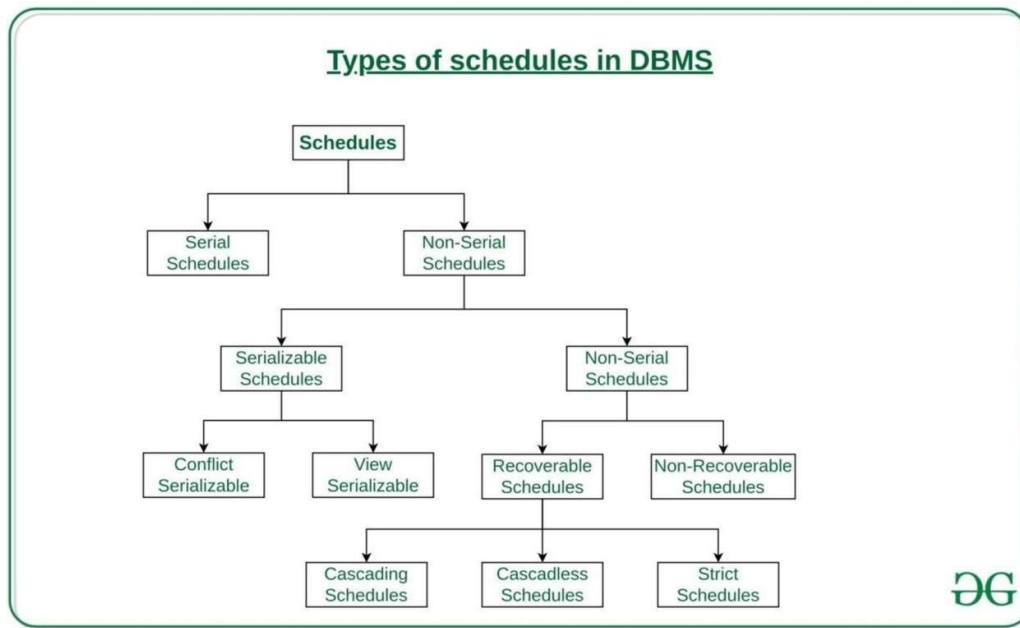
Figure5.2

SerialSchedules:

	T₁	T₂
Schedules in which the transactions are executed non-interleaved, i.e., a serial schedule is one in which no transaction startsuntilarunningtransactionhas ended are called serial schedules. **Example:**Considerthefollowingschedule involving two transactions T₁and T₂. where R(A) denotes that a read operation isperformedonsomedataitem 'A' This is a serial schedule since the transactions perform serially in the order T₁—>T₂	R(A) W(A) R(B)	W(B) R(A) R(B)

Non-SerialSchedule:

This is a type of Scheduling where the operations of multiple transactionsare interleaved. This might lead to a rise in the concurrency problem. The transactions are executed in a non-serial manner, keeping the end result correct and same as the serial schedule.

TheNon-SerialSchedulecanbedividedfurtherintoSerializableandNon-Serializable.

a. Serializable:

This isused tomaintain theconsistency of the database. Itis mainlyused in the Non-Serial scheduling to verify whether the scheduling will lead to any inconsistency or not. AserializableschedulehelpsinimprovingbothresourceutilizationandCPUthroughput. These are of two types:

i. *ConflictSerializable:*

A schedule is called conflict serializable if it can be transformed into a serial schedulebyswappingnon-conflictingoperations.Twooperationsaresaidtobe conflicting if all conditions satisfy:

- Theybelongtodifferenttransactions
- Theyoperateonthesamedataitem
- AtLeastoneofthemisawriteoperation

ii. *ViewSerializable:*

AScheduleiscalledviewserializableifitisviewequaltoaserialschedule(no overlapping transactions). A conflict schedule is a view serializable but if the serializabilitycontainsblindwrites,thentheviewserializabledoesnotconflict serializable.

b. Non-Serializable:

Thenon-serializablescheduleisdividedintotwotypes,Recoverableand Non- recoverable Schedule.

i. Recoverable Schedule:

	T₁	T₂
Schedulesinwhichtransactionscommitonly afteralltransactionswhosechangestheyread commitarecalledrecoverableschedules.if sometransactionT_jisreadingvalueupdatedor writtenbysomeothertransactionT_i,thenthe commitofT_jmustoccurafterthecommitofT_i. **Example–**Considerthefollowingschedule involvingtwotransactionsT₁andT₂. ThisisarecoverableschedulesinceT₁commits beforeT₂,thatmakesthevaluereadby T₂correct.		
	R(A)	
	W(A)	
		W(A)
		R(A)
	commit	
		commit

iii. *Non-RecoverableSchedule:*

	T₁	T₂
	R(A)	
	W(A)	
		W(A)
		R(A)
		commit
	abort	

Example: Consider the following scheduleinvolvingtwotransactions T_2 read the value of A written by T_1, and committed.T_1lateraborted,thereforethe value read by $T_2$is wrong, but since T_2committed,thisscheduleis**non-recoverable**.

ConcurrentExecutionofTransactions:

AnimportanttaskofaDBMSistoscheduleconcurrentaccessestodatasothateach usercansafelyignorethefactthatothersareaccessingthedataconcurrently.The importanceofthista.skcannotbeunderestimatedbecause adatabaseistypically sharedbyalargenumberofusers,whosubmittheirrequeststotheDBMS independentlyand simplycannotbeexpectedtodeal witharbitrarychangesbeingmade concurrently by other users. A DBMS allows users to think of their programs &'3 if they were executing inisolation,one after the other in some order chosenby the DBJ\;:IS. For example, if a program that deposits cash into an account is submitted to the DBMS at the same time as another program that debits money from the same account, either of these programs could be run first by theDBMS, but theirsteps will not be interleaved in such a way that they interfere with each other.

A locking protocol is a set of rules to be followed by each transaction (and enforced by the DBMS) to ensure that, even though actions of several transactions might be interleaved,thenet effectisidentical toexecutingall transactionsin someserial order.A lock is a mechanism used to control access to database objects. Two kinds of locks are commonly supported by a DBMS: shared locks on an object can be held by two different transactions at the same time, but an exclusive lock on an object ensures that no other transactions hold any lock on this object.

Suppose that the following locking protocol is followed: Every transaction begins by obtaining a shared lock on each data object that it needs to read and anexclusive lock on each data object that it needs to rnod~fy, then releases all its locks after completing allactions.ConsidertwotransactionsT1andT2suchthatT1wantstomodifyadata

Intuitively, if T1's request for an exclusive lock on the object is granted first, T2 cannot proceed until T1 relea..':les this lock, because T2's request for a shared lock will not be granted by the DBMS until then. Thus, all of T1's actions will be completed before any of T2's actions are initiated.

ConcurrentExecutionofDatabase:

Multiple transactions are allowed to run concurrently in the system.Concurrentexecution of database is meant by execution of database in parallel. I.e.eachtransaction must behave in isolation. This means that the concurrent execution does not result an inconsistent state. Ensuring consistency in spite of concurrent execution of transactions requires is very complex

Advantagesare:

- increasedprocessoranddiskutilization,leadingtobettertransactionthroughput
- 4 E.g. one transaction can be using the CPU while another is reading from or writing to the disk
- reduced average response time for transactions: short transactions need not wait behind long ones.

Timestamp Protocol: Timestamp is a unique identifier to identify a transaction. Timestamp can be considered to be as the transaction start time. It determines the concurrent execution such that the timestamp determines the serializability order. Time stamp holds two timestamp values:

- W-timestamp() is the largest time-stamp of any transaction that executed write() successfully.
- R-timestamp () is the largest time-stamp of any transaction that executed read() successfully.

SupposethattransactionTiissueswrite(Q).

If TS (Ti) < R-timestamp (Q), then the value of Q that Ti is producing was needed previously, and the system assumed thatthat value would never be produced. Hence, the write operation is rejected, and Ti is rolled back.

If TS (Ti) < W-timestamp (Q), then Ti is attempting to write an obsolete value of Q. Hence, this write operation is rejected, and Ti is rolled back.

Otherwise, the write operation is executed, and W-timestamp(Q) is set to TS(Ti).The timestamp-ordering protocol guarantees serializability since all the arcs inthe precedence graph are of the form. Timestamp protocolensures freedom from deadlock

ConcurrencyControl:

Concurrency control is the procedure in DBMS for managing simultaneous operations withoutconflictingwitheach another.Concurrentaccess isquiteeasyifall users arejust reading data.

Concurrency control is used to address such conflicts which mostly occur with a multi-user system. It helps you to make sure that database transactions are performed concurrently without violating the data integrity of respective databases.

WhyuseConcurrencymethod?

ReasonsforusingConcurrencycontrolmethodisDBMS:

ToapplyIsolationthroughmutualexclusionbetweenconflictingtransactions To resolve read-write and write-write conflict issues

To preserve database consistency through constantly preserving execution obstructions

Thesystemneedstocontroltheinteractionamongtheconcurrenttransactions.This control is achieved using concurrent-control schemes.

Concurrencycontrolhelpstoensureserializability

ConcurrencyControlProtocols:

Different concurrency control protocols offer different benefits between theamount ofconcurrency they allow and the amount of overhead that they impose.

- Lock-BasedProtocols
- TwoPhase
- Timestamp-BasedProtocols
- Validation-BasedProtocols

Lock-basedProtocols:

A lock is a data variable which is associated with a data item. This lock signifies that operations that can be performed on the data item. Locks help synchronize access to the database items by concurrent transactions.

All lock requests are madeto the concurrency-controlmanager. Transactions proceedonly once the lock request is granted.

BinaryLocks:ABinarylockonadataitemcaneitherlockedorunlockedstates.

Shared/exclusive:Thistypeoflockingmechanismseparatesthelocksbasedontheir

uses.Ifalockisacquiredonadataitemtoperformawriteoperation,itiscalledan exclusive lock.

1. SharedLock(S):

A shared lock is also called a Read-only lock. With the shared lock, the data item can be shared between transactions. This is because we will never have permission to update data on the data item.

For example, consider a case where two transactions are reading the account balance of a person. The database will let them read by placing a shared lock. However, if another transaction wants to update that account's balance, shared lock prevent it until the reading process is over.

2. ExclusiveLock (X):

With the Exclusive Lock, a data item can be read as well as written. This is exclusive and can't be held concurrently on the same data item. X-lock is requested using lock-x instruction. Transactions may unlock the data item after finishing the 'write' operation.

For example, when a transaction needs to update the account balance of a person. You can allows this transaction by placing X lock on it. Therefore, when the second transaction wants to read or write, exclusive lock prevent this operation.

TwoPhaseLocking(2PL)Protocol:

Two-Phase locking protocol which is also known as a 2PL protocol. Itis also called P2L. In thistypeoflockingprotocol,thetransaction should acquirealockafterit releases one of its locks.

This locking protocol divides the execution phase of a transaction into three different parts.

- Inthefirstphase,when thetransactionbeginstoexecute,itrequirespermission for the locks it needs.
- Thesecondpartiswherethetransactionobtainsallthelocks.Whenatransaction releases its first lock, the third phase starts.
- Inthisthirdphase,thetransactioncannotdemandanynewlocks.Instead,itonly releases the acquired locks.
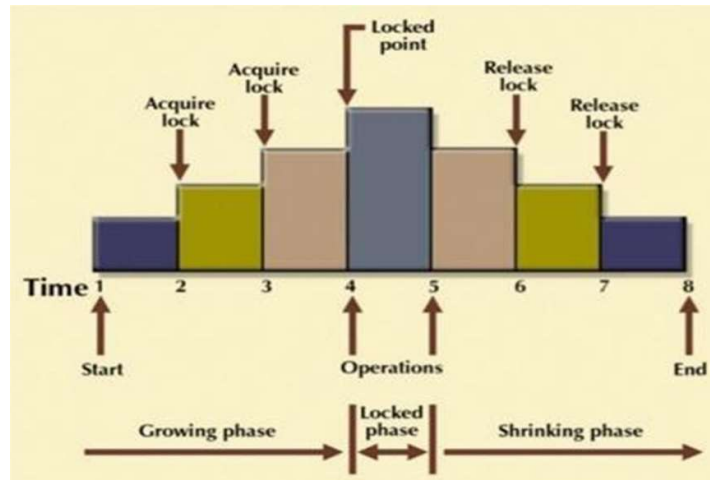
Figure5.2

The Two-Phase Locking protocol allows each transaction to make a lock or unlockrequest in two steps:

- **Growing Phase**: In this phase transaction may obtain locks but may not release any locks.
- **Shrinking Phase**: In this phase, a transaction may release locks but not obtain any new lock

It is true that the 2PL protocol offers serializability. However, it does not ensure that deadlocks do not happen.

In the above-given diagram, you can see that local and global deadlock detectors are searching for deadlocks and solve them with resuming transactions totheirinitial states.

Strict Two-Phase Locking Method:

Strict-Two phase locking system is almost similar to 2PL. The only difference is that Strict-2PLneverreleasesa lockafterusingit.It holdsallthelocksuntilthecommitpoint and releases all the locks at one go when the process is over.

Centralized2PL

In Centralized 2 PL, a single site is responsible for lock management process. It has only one lock manager for the entire DBMS.

Primarycopy 2PL

Primarycopy2PLmechanism,manylockmanagersaredistributedtodifferentsites.

Afterthat,aparticularlockmanagerisresponsibleformanagingthelockforasetof data items.

When the primary copy has been updated, the change is propagated to the slaves.

Distributed2PL

In this kind of two-phase locking mechanism, Lock managers are distributed to all sites. They are responsible for managing locksfor dataatthatsite. If no data isreplicated, it is equivalent to primary copy 2PL. Communication costs of Distributed 2PL are quite higher than primary copy 2PL

Timestamp-basedProtocols:

The timestamp-based algorithm uses a timestamp to serialize the executionof concurrent transactions. This protocol ensures that every conflicting read and write operations are executed in timestamp order. The protocol uses the **System Time or Logical Count as** a Timestamp.

The older transaction is always given priority in this method. It uses system time to determine the time stamp of the transaction. This is the most commonly used concurrency protocol.

Lock-based protocols help you to manage the order between the conflicting transactions when they will execute. Timestamp-based protocols manage conflicts as soon as an operation is created.

Example:
SupposetherearetheretransactionsT1,T2,andT3. T1

has entered the system at time 0010

T2hasenteredthesystemat0020

T3hasenteredthesystemat0030

PrioritywillbegiventotransactionT1,thentransactionT2andlastlyTransactionT3.

ConcurrencyControl:

Serializability:

- Serializabilityisaconcurrencyschemewheretheconcurrenttransactionisequivalent   to one that executes the transactions serially.
- Ascheduleisalist oftransactions.
- Serial schedule defines each transaction is executed consecutively without any interference from other transactions.

Themainobjectiveofserializabilityistofindnon-serialschedulesthatallowtransactions

toexecute concurrently without interference and producea databasestatethat could be produced by a serial execution.

ConflictSerializability:

- Conflictserializabilitydefinestwoinstructionsoftwodifferenttransactionsaccessing the same data item to perform a read/write operation.
- Itdealswithdetectingtheinstructionsthatareconflictinginanywayandspecifying the order in which the instructions should execute in case there is any conflict.
- Aconflictserializabilityariseswhenoneoftheinstructionisawriteoperation.

ThefollowingrulesareimportantinConflictSerializability:

1. Iftwotransactionsarebothread operation,thentheyarenotinconflict.

2. Ifonetransactionwantstoperformareadoperationandothertransactionwants to perform a write operation, then they are in conflict and cannot be swapped.

3. Ifboththetransactionsareforwriteoperation,thentheyareinconflict,butcanbe allowed to take place in any order, because the transactions do not read the value updated by each other.

ViewSerializability:

- Viewserializabilityistheanothertypeofserializability.
- Itcanbederivedbycreatinganotherscheduleoutofanexistingscheduleand involves the same set of transactions.

Example : Let us assume two transactions T1 and T2 that are being serialized to create two different schedules SH1 and SH2, where T1 and T2 want to access the same data item. Now there can be three scenarios

1. If inSH1,T1readstheinitialvalueof dataitem,then inSH2 , T1should readthe initial value of that same data item.

2. IfinSH2, T1writesavaluein thedataitemwhichisreadbyT2,theninSH2,T1 should write the value in the data item before T2 reads it.

3. IfinSH1,T1performsthefinal writeoperationonthat dataitem,then inSH2,T1 should perform the final write operation on that data item.

Ifaconcurrentscheduleisviewequivalenttoaserialscheduleofsametransactionthen it is said to be **View serializable.**

Recoverability:

a transaction may not execute completely due to hardware failure, system crash or software issues. In that case, we have to roll back the failed transaction. But some other transaction may alsohaveused values produced by thefailedtransaction. Sowehaveto rollbackthosetransactionsaswell.

RecoverableSchedules:

Schedules in which transactions commit only after all transactions whose changes they read commit are called recoverable schedules. In other words, if some transaction T_j is readingvalueupdatedorwrittenbysomeothertransactionT_i,thenthecommitof T_j must occur after the commit of T_i.

Example1:

S1:R1(x),W1(x),R2(x),R1(y),R2(y),

W2(x), W1(y), C1, C2;

Given schedule follows order of **Ti->Tj=>C1->C2**. Transaction T1 is executed before T2 hence there is no chances of conflict occur. R1(x) appears before W1(x) and transaction T1 is committed before T2 i.e. completion of first transaction performed first update on data item x, hence given schedule is recoverable.

Example2:

	T₁	T₂
Considerthefollowingschedule involvingtwotransactionsT₁andT₂. Thisisarecoverableschedulesince T₁commitsbeforeT₂,thatmakesthe valuereadbyT₂correct.	R(A)	
	W(A)	
		W(A)
		R(A)
	commit	
		commit

IntroductiontoLock:

Transaction processing systems usually allow multiple transactions to run concurrently. By allowing multiple transactions to run concurrently will improve the performance of the system in terms of increased throughout or improved response time, but this allows causes several complications with consistency of the data.

Alockisavariableassociatedwithadataitemthatdescribesthestatusoftheitem with respect to possible operations that can be applied to it. Generally, there is one lock for each data item in the database. Locks are used as a means of synchronizing the access by concurrent transactions to the database item.

TypesofLocks:

Severaltypesoflocksareusedinconcurrencycontrol.

BinaryLocks:

Abinarylockcanhavetwostatesorvalues:lockedandunlocked.

Abinarylockcanhavetwostatesorvalues:lockedandunlocked.

A distinct lock is associated with each database item *A.* If the value of the lock on *A* is 1, item *A* cannot be accessed by a database operation that requests the item. If the valueof the lock on *A* is 0 then item can be accessed when requested. We refer to the current valueofthelockassociatedwithitem *A* as *LOCK(A).* Therearetwooperations, lock item and unlock item are used with binary locking A transaction requests access to an item *A* by first issuing a lock *item (A)* operation. IfLOCK (A) = 1, the transaction is forced to wait. If LOCK (A) = 0 it is set to 1 (the transaction locks the item) and the transaction is allowed to access item *A.* When the transaction is through using the item,it issues an unlock *item (A)* operation, which sets *LOCK (A)* to 0 (unlocks the item) so that *A* may be accessed by other transactions. Hence binary lock enforces mutual exclusiol1 on the data item.

Share/Exclusive(forRead/Write)Locks

We should allow several transactions to access the same item *A* if they all access *A'*for reading purposes only. However, if a transaction is to write an item A, it must have exclusive access to *A.* For this purpose, a different type of lock called a multiple-mode lock is used. In this scheme there are shared/exclusive or read/write locks are used.

Sharedlock:

- o ItisalsoknownasaRead-onlylock.Inasharedlock,thedataitemcanonly read by the transaction.
- o It canbesharedbetweenthetransactionsbecausewhenthetransactionholdsa lock, then it can't update the data on the data item.

Exclusivelock:

- o Intheexclusivelock,thedataitemcan beboth readsaswell aswrittenbythe transaction.
- o Thislockisexclusive,andinthislock, multipletransactionsdo notmodifythe same data simultaneously.

DealingwithDeadlock:

Ina database,adeadlock is anunwanted situation inwhichtwoormore transactions are waiting indefinitely for one another to give up locks.

Example–letusunderstandtheconceptofDeadlockwithanexample: Suppose, Transaction T1 holds a lock on some rows in the Students table and **needs to update** some rows in the Grades table. Simultaneously, Transaction **T2 holds** locks on those very rows (Which T1 needs to update) inthe Grades table **butneeds**to update the rows in the Student table **held by Transaction T1**.

Now, themainproblemarises. Transaction T1will waitfortransaction T2togiveuplock, and similarly transaction T2 will wait for transaction T1 to give up lock. As a consequence, All activity comes to a halt and remains at a standstill forever unless the DBMS detects the deadlock and aborts one of the transactions.
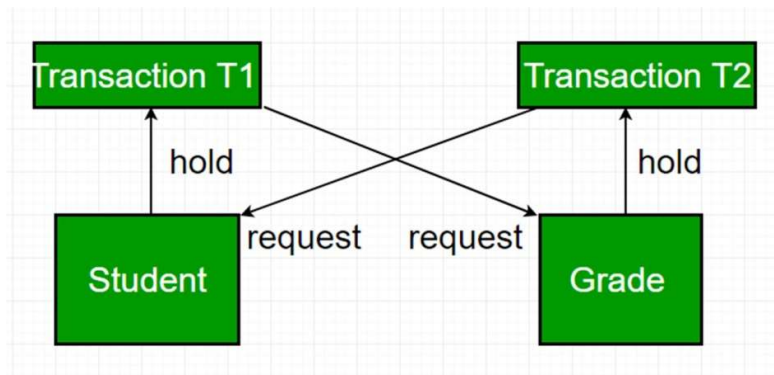
Figure5.3

Deadlockprevention–

For large database, deadlock prevention method is suitable. A deadlock canbeprevented if the resources are allocated in such a way that deadlock never occur. The DBMS analyzes the operations whether they can create deadlock situation or not, If they do, that transaction is never allowed to be executed.

Deadlockpreventionmechanismproposestwoschemes:

i. Wait-DieScheme:

In this scheme, If a transaction request for a resource that is locked byother transaction, then the DBMS simply checks the timestamp of both transactions and allows the older transaction to wait until the resource is available for execution.

Suppose, there are two transactions T1 and T2 and Let timestamp of any transaction T be TS (T). Now, If there is a lock on T2 by some other transaction and T1 is requesting for resources held by T2, then DBMS performs following actions:

Checks if TS (T1) < TS (T2) – if T1 is the older transaction and T2 has held some resource, then it allows T1 to wait until resource is available for execution. That means if a younger transaction has locked some resource and older transaction is waiting for it, thenoldertransactionisallowedwaitforittillitisavailable.IfT1isoldertransaction

andhasheldsomeresourcewithitandifT2iswaitingforit,thenT2iskilledand
restartedlatterwithrandomdelaybutwiththesametimestamp.i.e.iftheolder

transaction has held some resourceand youngertransaction waits for theresource, then younger transaction is killed and restarted with very minute delay with same timestamp. This scheme allows the older transaction to wait but kills the younger one.

WoundWaitScheme–

In this scheme, if an older transaction requests for a resource held by younger transaction, then older transaction forces younger transaction to kill the transaction and release the resource. The younger transaction is restarted with minute delay but with same timestamp. If the younger transaction is requesting a resource which is held by older one, then younger transaction is asked to wait till older releases it.

DeadlockDetection–

Whenatransactionwaitsindefinatelytoobtainalock,Thedatabasemanagememt

system should detect whether the transaction is involved in a deadlock or not.

Wait-for-graph is one of the methods for detecting the deadlock situation. This method is suitable for smaller database. In this method a graph is drawn based on the transaction and their lock on the resource. If the graph created has a closed loop or a cycle, then there is a deadlock.

FortheabovementionedscenariotheWait-Forgraphisdrawnbelow
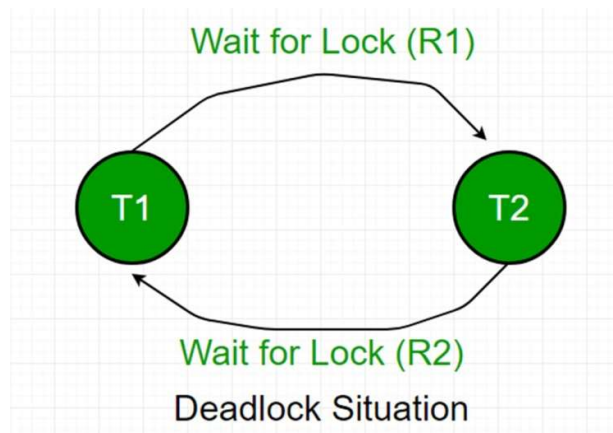
Figure5.4

Deadlock   Avoidance   –

When a database is stuck in a deadlock, It is always better to avoid the deadlock rather than restarting or aborting the database. Deadlock avoidance method is suitable for smaller database whereas deadlock prevention method is suitable for larger database. Onemethod ofavoiding deadlockisusing application consistentlogic.In theabovegiven example, Transactions that access Students and Grades should always access the tables in the same order. In this way, in the scenario described above, Transaction T1 simply waits for transaction T2 to release the lock on Grades before it begins. When transaction T2releasesthelock,TransactionT1canproceedfreely.

Anothermethodforavoidingdeadlockistoapplybothrowlevellockingmechanismand

READCOMMITTEDisolationlevel.However,Itdoesnotguaranteetoremovedeadlocks completely.

CrashRecovery:
IntroductiontoARIES:

AlgorithmforRecoveryandIsolationExploitingSemantics(ARIES)isbasedontheWrite Ahead Log (WAL) protocol. Every update operation writes a <u>log record</u>which is one of the following :

1. **Undo-onlylogrecord:**

   Onlythebeforeimageislogged.Thus,anundooperationcanbedonetoretrieve the old data.

2. **Redo-onlylogrecord:**

   Onlytheafterimageislogged.Thus,aredooperationcanbeattempted.

3. **Undo-redologrecord:**

   Bothbeforeimagesandafterimagesarelogged.

Therecoveryprocessactuallyconsistsof3phases:

1. **Analysis:**

   Therecoverysubsystemdeterminestheearliestlogrecordfromwhichthenext pass must start. It also scans the log forward from the checkpoint record to constructasnapshot of whatthesystemlookedlikeattheinstantofthecrash.

2. **Redo:**

   StartingattheearliestLSN,thelogisreadforwardandeachupdateredone.

3. **Undo:**

   Thelogisscannedbackwardandupdatescorrespondingtolosertransactionsare undone.

FailureClassification:

   Toseewheretheproblemhasoccurredwegeneralizethefailureintovarious categories, as follows:

TRANSACTIONFAILURE :

   When atransaction isfailed toexecuteorit reachesapointafterwhich itcannot be completed successfully it has to abort. This is called transaction failure. Where only few transaction or process are hurt.

Reasonfortransactionfailurecouldbe:

- **Logical errors:** where a transaction cannot complete because of it has some code error or any internal error condition
- **System errors:** where the database system itself terminates an active transaction because DBMS is not able to execute it or it has to stop because of some system condition. For example, in case of deadlock or resource unavailability systems aborts an active transaction.

SYSTEMCRASH:

There are problems, which are external to the system, which may cause the system to stop abruptly and cause the system to crash. For example interruption in power supply, failure of underlying hardware or software failure.

- Examplesmayincludeoperatingsystemerrors.

DISKFAILURE:

Inearlydaysoftechnologyevolution,itwasacommonproblemwhereharddisk drives or storage drives used to fail frequently.

- Disk failures include formation ofbad sectors, unreachability to the disk, disk head crash or any other failure, which destroys all or part of disk storage

StorageStructure:

We have already described storage system here. In brief, the storage structure can be divided in various categories:

- **Volatile storage:** As name suggests, this storage does not survive system crashes and mostly placed very closed to CPU by embedding them onto the chipset itself for examples: main memory, cache memory. They are fast but can store a small amount of information.

- **Nonvolatile storage:** Thesememories are made to survive system crashes. They are huge in data storage capacity but slower in accessibility. Examples may include, hard disks, magnetic tapes, flash memory, non-volatile (battery backed up) RAM.

RecoveryandAtomicity:

When a system crashes, it many have several transactions being executed and various files opened forthem to modifyingdataitems. Aswe know that transactionsaremadeof various operations, which are atomic in nature. But according to ACID properties of DBMS, atomicity of transactions as a whole must be maintained that is, either all operations are executed or none.

WhenDBMSrecoversfromacrashitshouldmaintainthefollowing:

- Itshouldcheckthestatesofalltransactions,whichwerebeingexecuted.
- Atransactionmaybe in themiddleofsomeoperation;DBMSmustensure the atomicity of transaction in this case.
- Itshouldcheckwhetherthetransactioncanbecompletednoworneedstobe rolled back.
- NotransactionswouldbeallowedtoleftDBMSininconsistentstate.

There are two types of techniques, which can help DBMS in recovering as well as maintaining the atomicity of transaction:

- Maintaining the logs of each transaction, and writing them onto some stable storage before actually modifying the database.
- Maintaining shadow paging, where are the changes are done on a volatile memory and later the actual database is updated.

Log-BasedRecovery

- Log is a sequence of records, which maintains the records of actionsperformedby a transaction. It is important that the logs are written prior to actual modification and stored on a stable storage media, which is failsafe.
- Logbasedrecoveryworksasfollows:
- Thelogfileiskeptonstablestoragemedia
- Whenatransactionentersthesystemandstartsexecution,itwritesa logabout it.
  <Tn,Start>
    - WhenthetransactionmodifiesanitemX,itwritelogsasfollows:
    <Tn,X,V1,V2>ItreadsTnhaschangedthevalueofX,fromV1toV2.

- When transaction finishes, it logs:

<Tn,commit>

- Database can be modified using two approaches:

1. **Deferred database modification:** All logs are written on to the stable storage and database is updated when transaction commits.

2. **Immediate database modification:** Each log follows an actual database modification. That is, database is modified immediately after every operation.

Recovery with concurrent transactions:

When more than one transactions are being executed in parallel, the logs are interleaved. At the time of recovery it would become hard for recovery system to backtrack all logs, and then start recovering. To ease this situation most modern DBMS use the concept of 'checkpoints'.

CHECKPOINT:

Keeping and maintaining logs in real time and in real environment may fill out all the memory space available in the system. At time passes log file may be too big to be handled at all. Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in storage disk. Checkpoint declares a point before which the DBMS was in consistent state and all the transactions were committed.

RECOVERY

When system with concurrent transaction crashes and recovers, it does behave in the following manner:

Figure5.5

- The recovery system reads the logs backwards from the end to the last Checkpoint.
- It maintains two lists, undo-list and redo-list.
- If the recovery system sees a log with <Tn,Start> and <Tn,Commit> or just

<Tn,Commit>, it puts the transaction in redo-list.

- If the recovery system sees a log with <Tn, Start> but no commit or abort log found, it puts the transaction in undo-list.

All transactions in undo-list are then undone and their logs are removed. All transaction in redo-list, their previous logs are removed and then redone again and log saved.

AriesAlgorithm:

ARIES (Algorithm for Recovery and Isolation Exploiting Semantics) recovery is based on the Write Ahead Logging (WAL) protocol. Every update operation writes a log record which is one of

1. Anundo-onlylogrecord:Onlythebeforeimageislogged.Thus,anundo operation can be done to retrieve the old data.
2. Aredo-onlylogrecord:Onlytheafterimageislogged.Thus,aredooperation can be attempted.
3. Anundo-redologrecord.Bothbeforeimageandafterimagesarelogged.

Every log record is assigned a uniqueand monotonicallyincreasing log sequence number (LSN). Every data page has a page LSN field that is set to the LSN of the log record corresponding to the last update on the page. WAL requires that the log record corresponding to an update make it to stable storage before the datapagecorresponding to that update is written to disk. For performance reasons, each log write is not immediately forced to disk. A log tail is maintained in main memory to buffer log writes. The log tail is flushed to disk when it gets full. A transaction cannot be declared committed until the commit log record makes it to disk.

Once in a while the recovery subsystem writes a checkpoint record to the log. The checkpoint record contains the transaction table (which gives the list of active transactions) and the dirty page table (the list of data pages in the buffer pool that have not yet made it to disk). A master log record is maintained separately, in stable storage, to store the LSN of the latest checkpoint record that made it to disk. On restart, the recovery subsystem reads the master log record to find the checkpoint's LSN, reads the checkpoint record, and starts recovery from there on.

Theactualrecoveryprocessconsistsofthreepasses:

1. Analysis: The recovery subsystem determines the earliest log record from which thenext pass must start. It also scans thelog forward from the checkpoint record toconstruct asnapshot of what thesystemlooked likeat theinstant of the crash.
2. Redo: Starting at the earliest LSN determined in pass (1) above, the log is read forward and each update redone.
3. Undo: The log is scanned backward and updates corresponding to loser transactions are undone.

UNIT–v

OverviewofIndexing:

We know that data is stored in the form of records. Every record has a key field, which helps it to be recognized uniquely.

Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database.

Indexesarecreatedusingafewdatabasecolumns.

- The first column is the **Search key** that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly.

  Note:Thedatamayormaynotbestoredinsortedorder.

- The second column is the **Data Reference** or **Pointer** which contains a set of pointers holding the address of the disk block where that particular key value can be found.
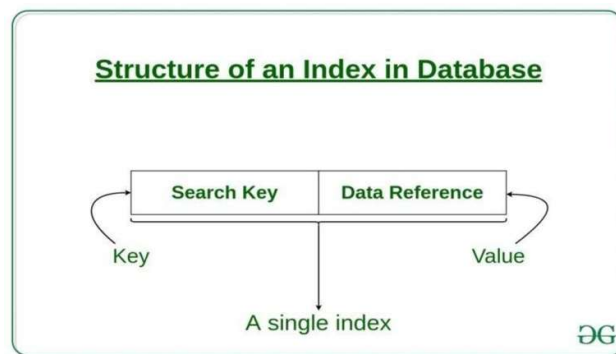
Figure6.1

Theindexinghasvariousattributes:

- **AccessTypes**:Thisreferstothetypeofaccesssuchasvaluebasedsearch, range access, etc.
- **AccessTime**:Itreferstothetimeneededtofindparticulardataelementorsetof elements.
- **InsertionTime**:Itreferstothetimetakentofindtheappropriatespaceand insert a new data.
- **DeletionTime**:Timetakentofindanitemanddeleteitaswellasupdatethe index structure.
- **SpaceOverhead**:Itreferstotheadditionalspacerequiredbytheindex.

In general,therearetwotypes offileorganization mechanismwhicharefollowedbythe indexing methods to store the data:

Sequential File Organization or Ordered Index File: In this, the indices are based on a sorted ordering of the values. These are generally fast and a more traditional type of storing mechanism. These Ordered or Sequential file organization might store the data in a dense or sparse format:

- o **DenseIndex:**
  - Foreverysearchkeyvalueinthedatafile,thereisanindexrecord.
  - This record contains the search key and also a reference to the first data record with that search key value.
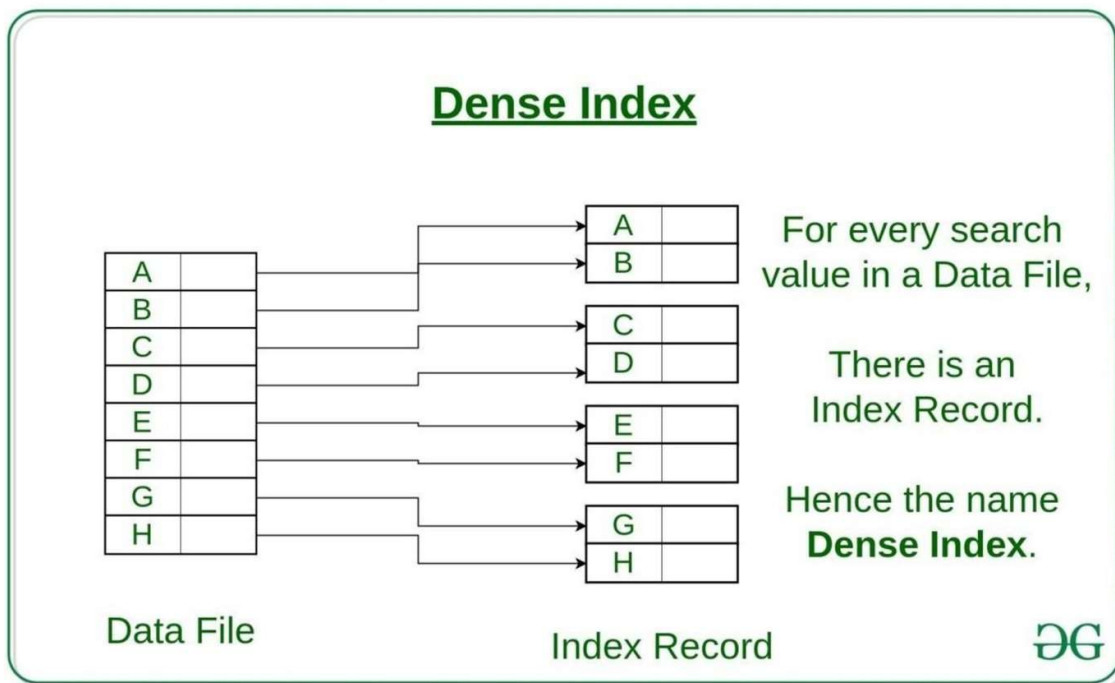
Figure6.2

SparseIndex:

- Theindexrecordappearsonlyforafewitemsinthedatafile.Eachitempointsto a block as shown.
- Tolocatearecord,wefindtheindexrecordwiththelargestsearchkey valueless than or equal to the search key value we are looking for.
- Westartat thatrecord pointedtobytheindexrecord,andproceedalongwiththe pointers in the file (that is, sequentially) until we find the desired record.
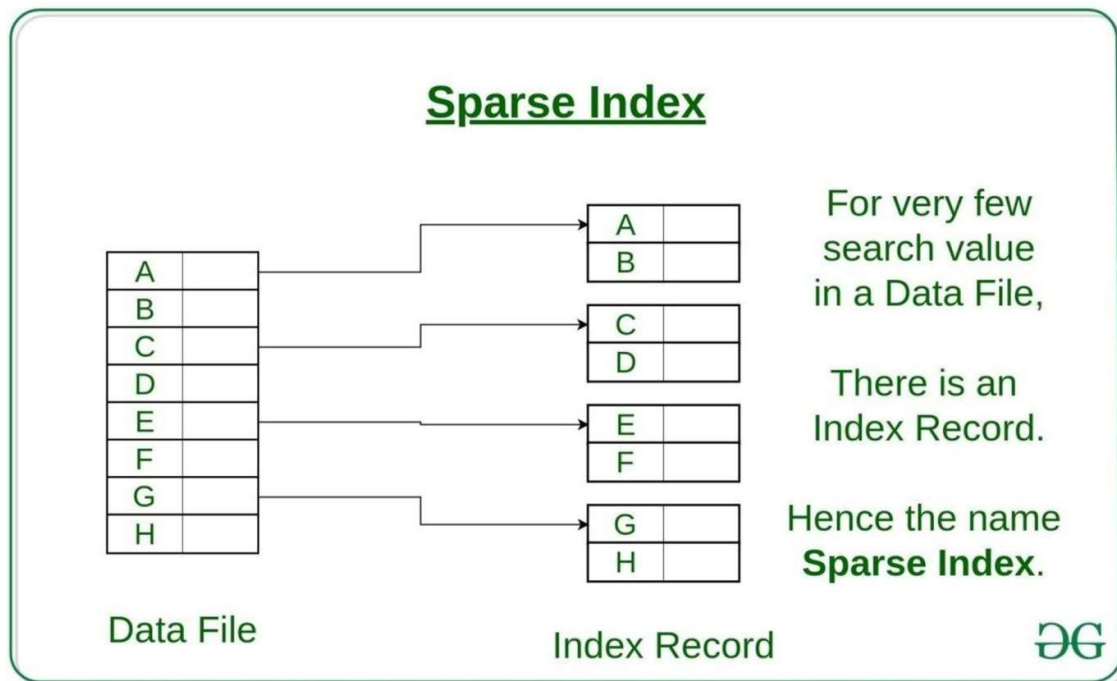
Figure6.3

Hash File organization: Indices are based on the values being distributed uniformly across a range of buckets. The buckets to which a value is assigned is determined by a function called a hash function.

Therearerimarilythreemethodsofindexing:

- ClusteredIndexing
- Non-ClusteredorSecondaryIndexing
- MultilevelIndexing

ClusteredIndexing:

When more than two records are stored in the same file these types of storing known as cluster indexing. By using the cluster indexing we can reduce the cost ofsearching reason being multiple records related to the same thing are stored at one place and it also gives the frequent joing of more than two tables(records).

Clustering index is defined on an ordered data file. The data file is ordered on a non-key field. In some cases, the index is created on non-primary key columns which may not be uniqueforeachrecord.Insuchcases,inordertoidentifytherecordsfaster,wewill grouptwoormorecolumnstogethertogettheuniquevaluesandcreateindexoutof them.Thismethodisknownastheclusteringindex.Basically,recordswithsimilar characteristicsaregroupedtogetherandindexesarecreatedforthese   groups. For example, students studying in each semester are grouped together. i.e. 1stSemester students,2ndsemester students,3rdsemester students etc are grouped.
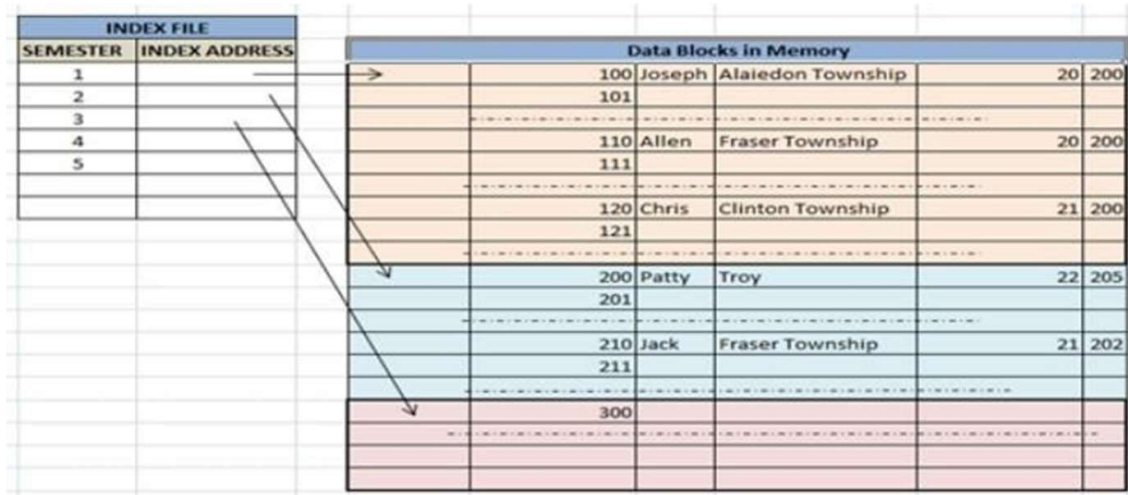
Figure6.4

Clusteredindexsortedaccordingtofirstname(Searchkey)

PrimaryIndexing:

This is a type of Clustered Indexing wherein the data is sorted according to the search key and the primary key of the database table is used to create the index. It is a default format of indexing where it induces sequential file organization. As primary keys are unique and are stored in a sorted manner, the performance of the searching operation is quite efficient.

Non-clusteredorSecondaryIndexing:

A non clustered index just tells us where the data lies, i.e. it gives us a list of virtual pointers or references to the location where the data is actually stored. Data is not physically stored in the order of the index. Instead, data is present in leaf nodes. For eg. the contents page of a book. Each entry gives us the page number or location of the information stored. The actual data here(information on each page of the book) is not organized but we have an ordered reference(contents page) to where the data points actually lie. We can have only dense ordering in the non-clustered index as sparse orderingisnotpossiblebecausedataisnotphysicallyorganizedaccordingly. It requires more time as compared to the clustered index because some amount of extra workisdonein orderto extractthedataby further followingthepointer. In thecaseof a clustered index, data is directly present in front of the index.
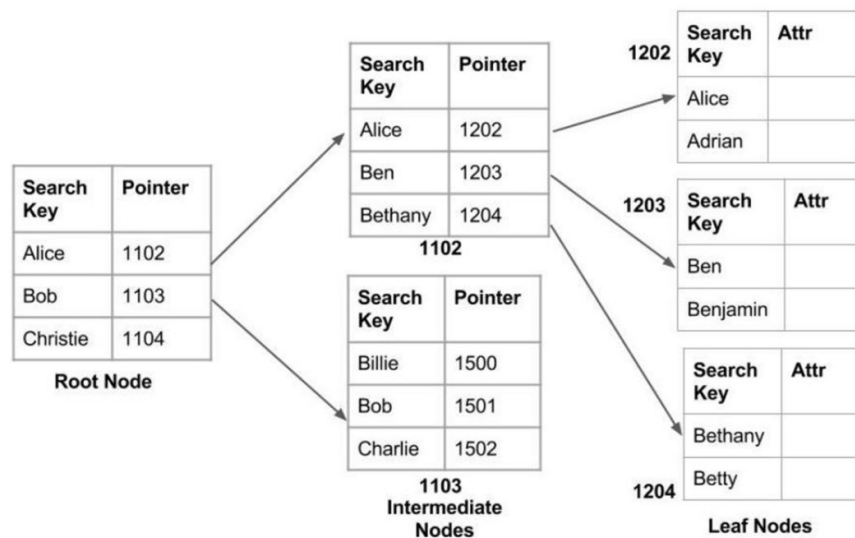
Non clustered index

Figure6.5

iii.MultilevelIndexing:

With the growth of the size of the database, indices also grow. As the index is stored in the main memory, a single-level index might become too large a size to store with multiple disk accesses. The multilevel indexing segregates the main block into various smallerblockssothat thesamecan storedin a singleblock. Theouter blocksaredivided intoinnerblockswhich inturn are pointedtothedatablocks.Thiscanbeeasilystored in the main memory with fewer overheads.
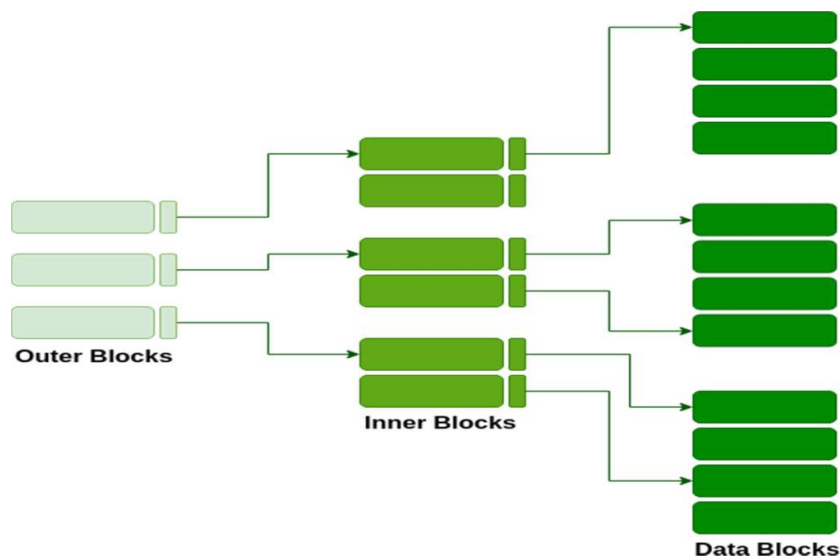
Figure6.6

TreeStructuredIndexing:

IntuitionforTreeIndexes:

Consider a file of Students recorcls sorted by *gpa.* To answer a range selection such as "Find all students with a gpa higher than 3.0," we must identify the first such student by doing a binary search of the file and then scan the file from that point on. If the file is large, the initial binary search can be quite expensive, since cost is proportional to the number of pages fetched.

One idea is to create a second file with One record per page in the original (data) file, of the form (first key on page, pointer to page), again sorted by the key attribute (which is gpa in our example). Theformat of a pageinthe second indexfileisillustratedin Figure.

Format of an Index Page
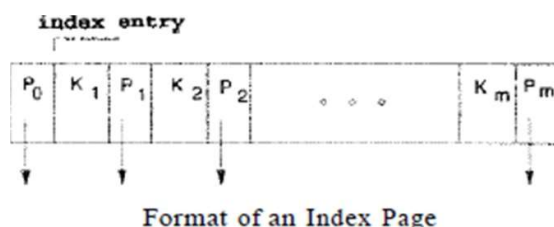
Figure6.7

We refer to pairs of the form (key, pointer) as indx entries or just entries whenthe ontext is clear. Notethat each index page contains One pointermorethan thenumber of keys --- each key serves as a separator- for the contents of the pages pointed to by the pointers to its left and right.The simple index file data structure is illustrated in Figure

One-Level Index Structure
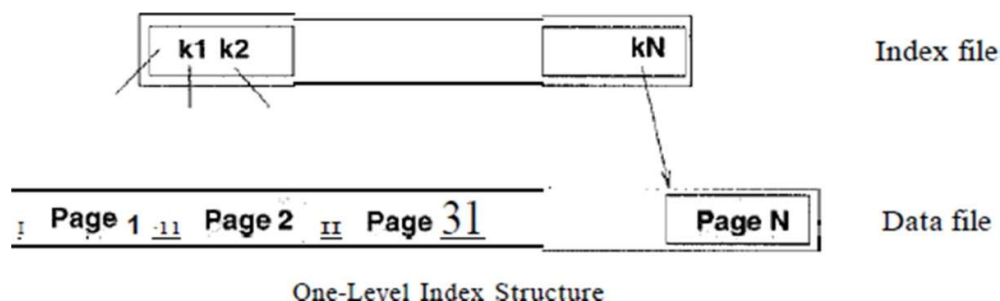
Figure6.8

We can do a binary search of the index file to identify the page containing the first key (gpa.) value that satisfies the range selection (in our example, the first student with gpa over 3.0) and follow the pointer to the page containing the firstdata. record with that key value. We can then scan the data file sequentiallyfrom that point on to retrieve other qualifying records. This example uses the index to find the first data page containing aStudentsrecord withgpagreaterthan 3.0,and thedatafileisscanned from that point on to retrieve other such Students records.

IndexedSequentialAccessMethods(ISAM):

ISAM method is an advanced sequential file organization. In this method, records are stored in the file using the primary key. An index value is generated for eachprimarykey and mapped withthe record. This indexcontains the address of the record inthe file. ThedataentriesoftheISAMindexareintheleafpagesofthetreeandadditional

overflowpageschainedtosomeleafpage.Databasesystemscarefullyorganizethe layout

of pages so that page boundaries correspond closely to the physical characteristics of the underlying storage device. The ISAM structure is completely static (exceptfortheoverflowpages,ofwhichitishoped,therewillbefew)andfacilitates such low-level optimizations. The ISAM data structure is illustrated in Figure

ISAM Index Structure
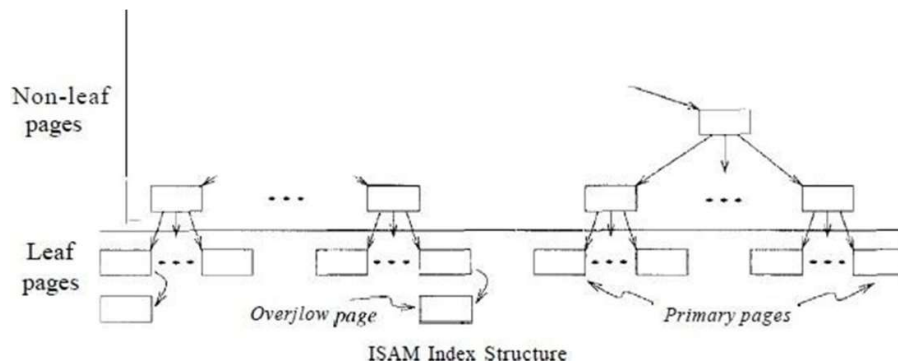
Figure6.9

Eachtreenodeisadiskpage,andallthedataresidesintheleafpages. Thisorresponds to an index that uses Alternative .

(1) fordataentries,intermsofthealternativeswecancreateanindexwithAlternative

(2) bystoringt.hedatarecordsinaseparatefileandstoring(key,rid)pairsintheleaf pages of the ISAM index.

Ifthere are several insertsto thefile subsequently,so that more entries are inserted into a leaf than will fit onto a single page, additional pages are needed becausethe index structure is static. These additional pages are allocated from an overflow area. Theallocationofpagesisillustrated	Data Pages Index Pages Overflow Pages Page Allocation in ISAM
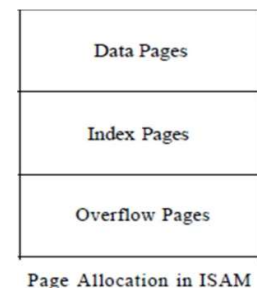
Thebasicoperationsofinsertion,deletion,andsearchareallquitestraightforward.For anequalityselectionsearch,westartattherootnodeanddeterminewhichsubtreeto

searchbycomparingthevalueinthesearchfieldofthegivenrecord withthekeyvalues in the node.

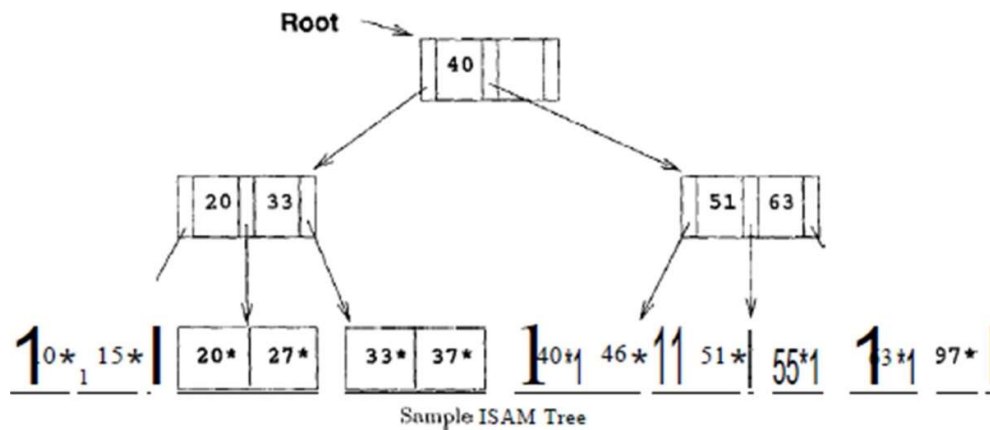ThefollowingexampleillustratestheISAMindexstructure.Considerthetreeshown below

Figure6.10

All searches begin at the root. For example, to locate a record with the key value 27, we start at the root and follow the left pointer, since 27 < 40. We then follow the middle pointer, since 20 <= 27 < 33. For a range search, we find the first qualifying data entry as for an equality selection and then retrieve primary leaf pagessequentiallyThe primary leaf pages are assumed to be allocated sequentially this assumption is reasonable because the number of such pages is known when the tree is created and does not change subsequently under inserts and deletes-and so no 'next leaf page' pointers are needed.
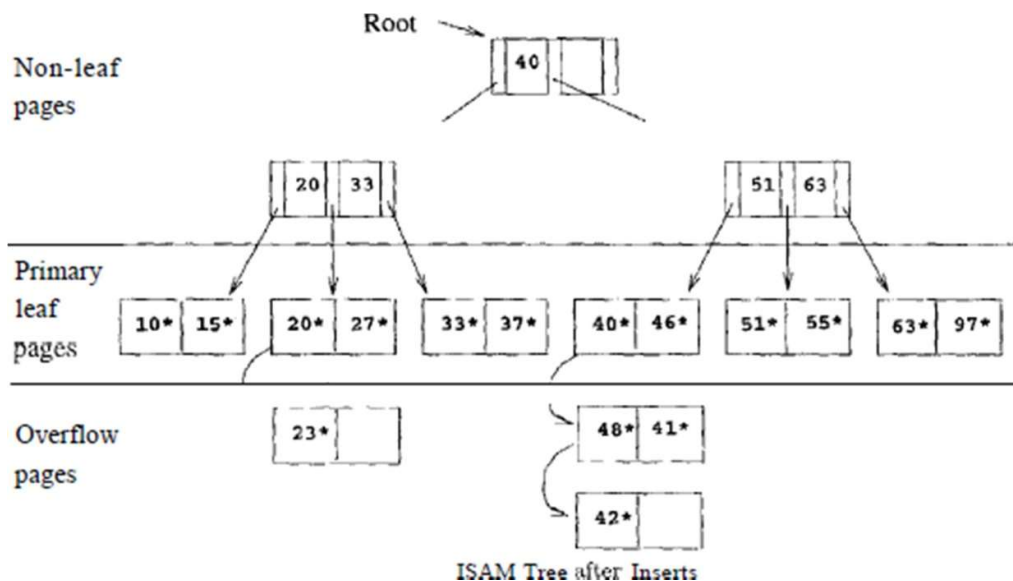
Figure6.11

We assume that each leaf page can contain two entries. If we now insert a record with keyvalue23,theentry23*belongsinthesecconddatapage,whichalreadycontains

20* and 27* and has no more space. We deal with this situation by adding an overflow page and putting 23* in the overflow page. Chains of overflow pages can easily develop. For instance, inserting 48*, 41* and 42* leads to an overflow chain of two pages. all these insertions shown in Figure 6.11

The deletion of an entry h is handled by simply removing the entry. If this entry is on an overflow page and the overflow page becomes empty, the page can be removed. If the entry is on a primary page and deletion makes the primary page empty, the simplest approach is to simply leave the empty primary pageas it is; it serves as a placeholder for future insertions. Thus, the number of primary leaf pages is fixed atfile creationtime.

B+Trees:

IntroductionofB+Tree:

To implement dynamic multilevel indexing, B-tree and B+ tree are generally employed. B+tree eliminatesthedrawbackofstoringthedatapointerbystoringdatapointersonly at the leaf nodes of the tree. Thus, the structure of leafnodesof a B+ tree is quite different from the structure of internal nodes of the B tree.

B+Tree:

TheB+treeisabalancedbinarysearchtree.Itfollowsamulti-levelindexformat. IntheB+tree,leafnodesdenoteactualdatapointers.B+treeensuresthatallleaf nodes remain at the same height.
IntheB+tree,theleafnodesarelinkedusingalinklist.Therefore,aB+treecan support random access as well as sequential access.

StructureofB+Tree:

IntheB+tree,everyleafnodeisatequal distancefromtherootnode.TheB+treeisof the order n where n is fixed for every B+ tree.
Itcontainsaninternalnodeandleafnode.

Figure6.12

Internalnode:

AninternalnodeoftheB+treecancontainatleastn/2recordpointersexcepttheroot node. At most, an internal node of the tree contains n pointers.

Leafnode:

TheleafnodeoftheB+treecancontainatleastn/2recordpointersandn/2keyvalues. At most, a leaf node contains n record pointer and n key values.

EveryleafnodeoftheB+treecontainsoneblockpointerPtopointtonextleafnode.

SearchingarecordinB+Tree:

Supposewehavetosearch55inthebelowB+treestructure. First,wewillfetchforthe intermediary node which will direct to the leaf node that can contain a record for 55.

So, in the intermediary node, we will find a branch between 50 and 75 nodes. Thenat the end, we will be redirected to the third leaf node. Here DBMS will performasequential search to find 55.
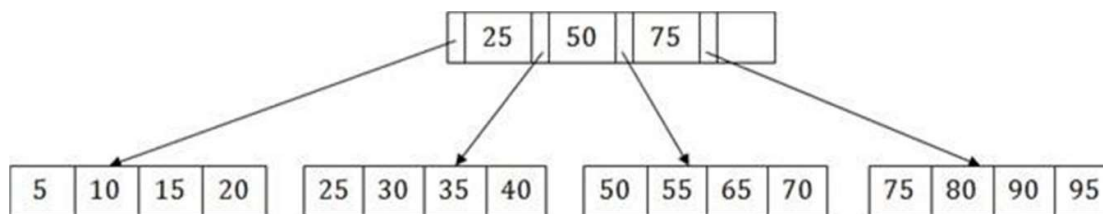
Figure6.13

B+TreeInsertion:

Suppose we want to insert a record 60 in the below structure. It will go to the 3rd leaf node after 55. It is a balanced tree, and a leaf node of this tree is already full, so we cannot insert 60 there.

In this case, we have to split the leaf node, so that it can be inserted into tree without affecting the fill factor, balance and order.
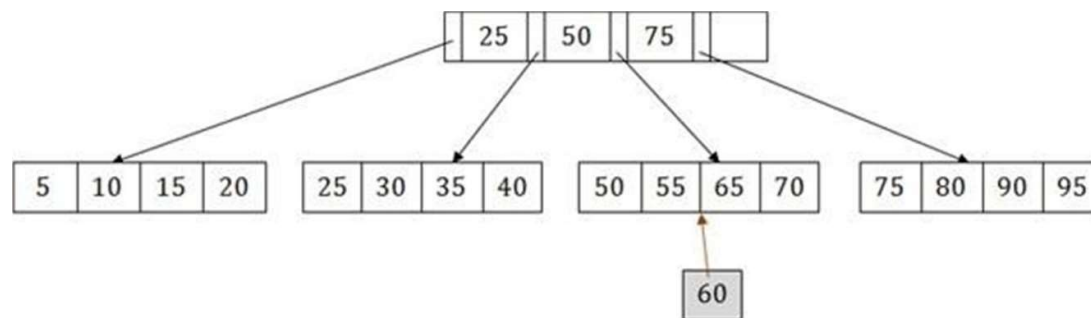
Figure6.14

The 3rd leafnodehas thevalues(50,55, 60, 65,70) anditscurrent root node is50.We will split the leaf node of the tree in the middle so that its balance is not altered. So we can group (50, 55) and (60, 65, 70) into 2 leaf nodes.

If these two has to be leaf nodes, the intermediate node cannot branch from 50. It should have 60 added to it, and then we can have pointers to a new leaf node.
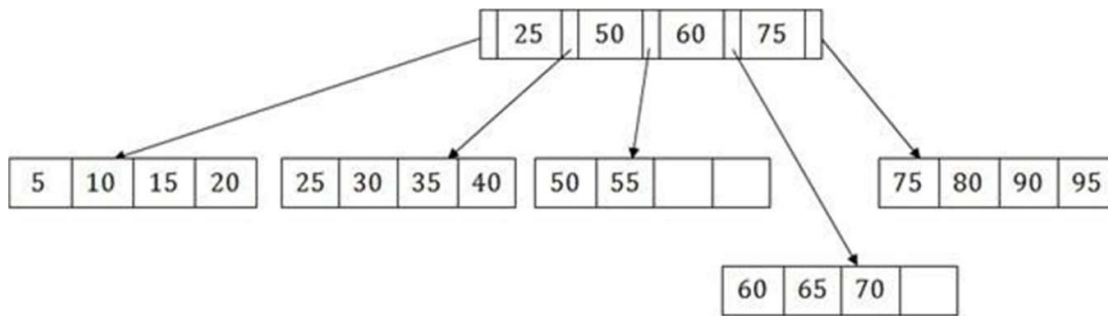

Figure6.15

Thisishowwecaninsertanentrywhenthereisoverflow.Inanormalscenario,itis very easy to find the node where it fits and then place it in that leaf node.

B+TreeDeletion:

Suppose wewant todelete60fromtheaboveexample.In this case,we haveto remove

60 from the intermediate node as well as from the 4th leaf node too. If we remove it from the intermediate node, then the tree will not satisfy the rule of the B+ tree. So we need to modify it to have a balanced tree.

After deleting node60fromaboveB+treeandre-arrangingthenodes,itwillshowas follows:
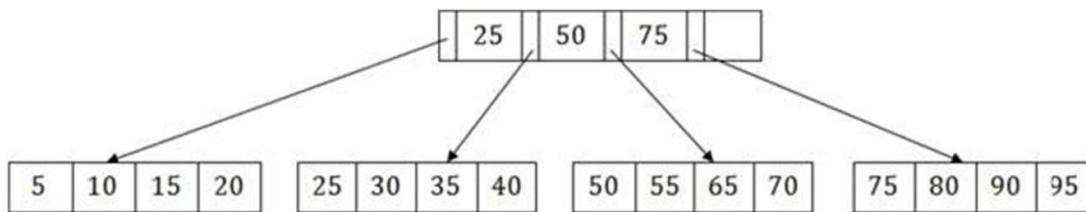
Figure6.16

Advantages:

- Automatically Adjust the nodes to fit the new record. Similarly it re-organizes the nodes in the case of delete, if required. Hence it does not alter the definitionof B+ tree.
- Nofiledegradationproblem

HashBasedIndexing:

Introduction:

In DBMS, hashing is a technique to directly search the location of desired data onthe disk without using index structure. Data is stored in the form of data blocks whose address is generated by applying a hash function in the memory location where these records are stored known as a data block or data bucket.

* Hashingmapsasearchkeydirectlytothepidofthecontainingpage/page- overflow chain
* Doesn'trequireintermediatepagefetchesforinternal"steeringnodes"oftree- based indices
* Hash-basedindexesarebestforequalityselections.Theydonotsupportefficient range searches.
* Staticanddynamichashingtechniquesexistwithtrade-offssimilartoISAMvs. B+ trees.

ImportantTerminologies usinginHashing:

Here,areimportantterminologieswhichareusedinHashing:

* **Databucket**–Databucketsarememorylocationswheretherecordsarestored. It is also known as Unit Of Storage.
* **Key**:ADBMSkeyisanattributeorsetofanattributewhichhelpsyoutoidentify arow(tuple) in arelation(table).Thisallowsyou tofind therelationship between two tables.
* **Hashfunction**:Ahash function,isamappingfunctionwhichmapsallthesetof search keys to the address where actual records are placed.
* **Linear Probing** – Linear probing is a fixed interval between probes. In this method,thenextavailabledatablockisusedtoenterthenewrecord,insteadof overwriting on the older record.
* **Quadratic probing**- It helps you to determine the new bucket address. It helps youtoaddIntervalbetweenprobesbyaddingtheconsecutiveoutputofquadratic polynomial to starting value given by the original computation.
* **Hashindex**–Itisanaddressofthedatablock.Ahashfunctioncouldbea simple mathematical function to even a complex mathematical function.
* **DoubleHashing**–Doublehashingisacomputerprogrammingmethodusedin hash tables to resolve the issues of has a collision.
* **BucketOverflow**:Theconditionofbucket-overflowiscalledcollision.Thisisa fatal stage for any static has to function.

TherearemainlytwotypesofSQLhashingmethods:

1. StaticHashing
2. DynamicHashing

StaticHashing

In the static hashing, the resultant data bucket address will always remain the same.

Therefore, if you generate an address forsay **Student_ID=10** usinghashing

function**mod(3)**,theresultantbucketaddresswill alwaysbe**1**.So,youwillnotseeany change in the bucket address.

Therefore,inthisstatichashingmethod,thenumberofdatabucketsinmemoryalways remains constant.

Hence in this static hashing, the number of data buckets in memory remains constant throughout. In this example, we will have five data buckets in the memory used to store the data.
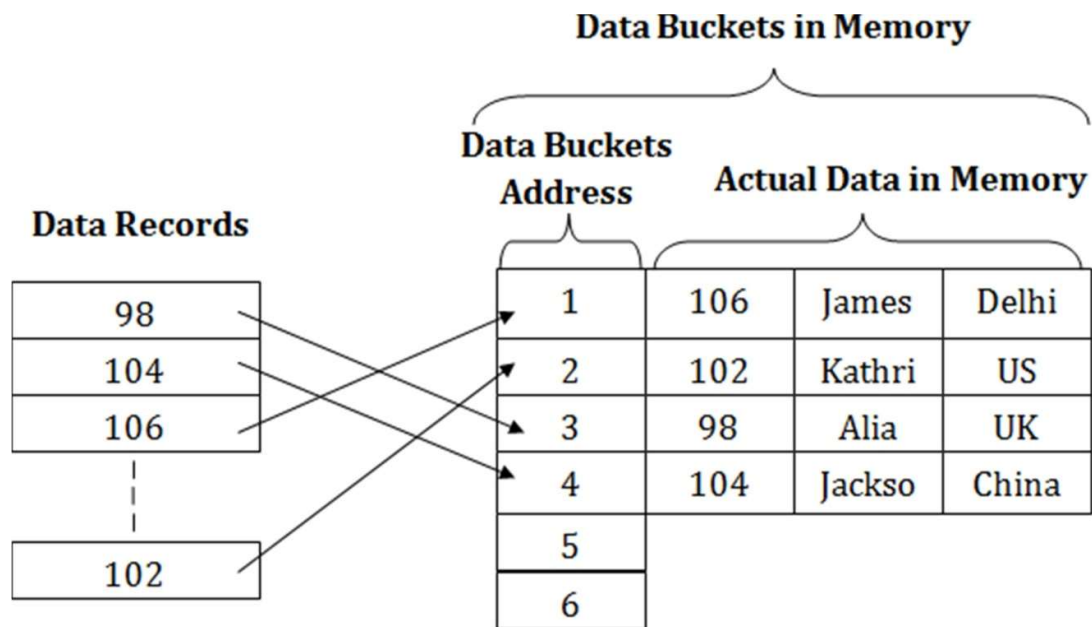
Figure6.17

StaticHashFunctions

- **Inserting a record**: When a new record requires to be inserted into the table, you can generate an address for the new record using its hash key. When the address is generated, the record is automatically stored in that location.

- **Searching**: Whenyou needtoretrievetherecord,the samehashfunctionshould be helpful to retrieve the address of the bucket where data should be stored.

- **Delete a record**: Using the hash function, you can first fetch the record which is you wants to delete. Then you can remove the records for that address in memory.

Statichashingisfurtherdividedinto:

1. *Openhashing*
2. *Closehashing.*

1. Open Hashing: In Open hashing method, Instead of overwriting older one the next available data block is used to enter the new record, This method is also known as linear probing.

For example: suppose R3 is a new address which needs to be inserted, the hash function generates address as 112 for R3. But the generated address is already full. So the system searches next available data bucket, 113 and assigns R3 to it.
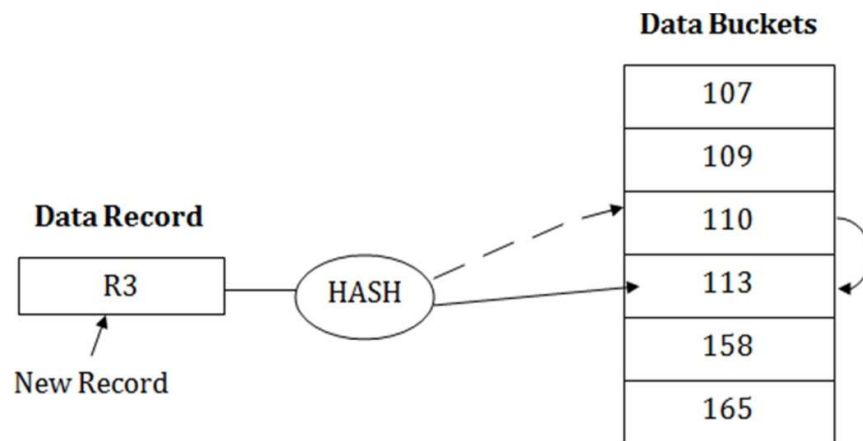
Data Buckets

107
109
110
113
158
165

Data Record

R3

New Record

HASH

Figure6.18

2. CloseHashing:

When buckets are full, then a new data bucket is allocated for the same hash result and is linked after the previous one. This mechanism is known as **Overflow chaining**.

Forexample: Suppose R3 is a new address which needs to be inserted into the table, the hash function generates address as 110 for it. But this bucket is full to store the new data. In this case, a new bucket is inserted at the end of 110 buckets and is linked to it.
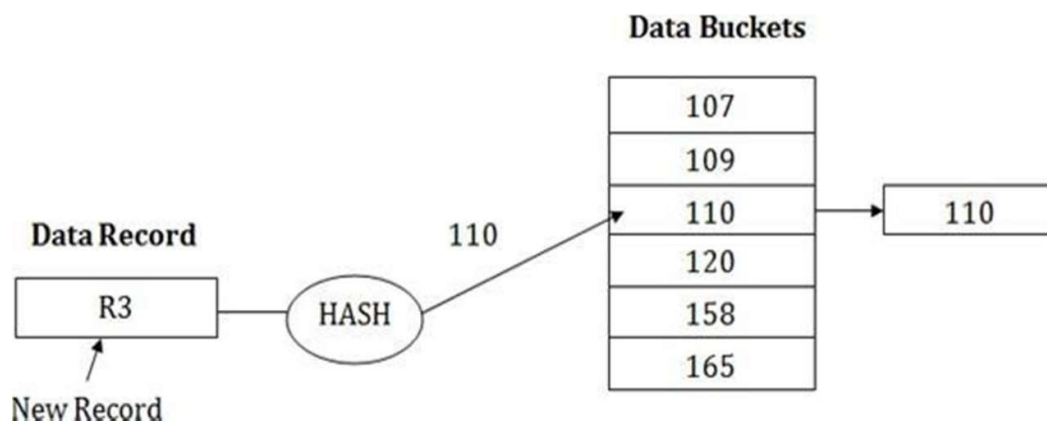
Data Buckets

107
109
110
120
158
165

110

Data Record

R3

New Record

HASH

110

Figure6.19

6.3.2DynamicHashing:

o Thedynamichashingmethodisusedtoovercometheproblemsofstatichashing like bucket overflow.

o Inthismethod,databucketsgroworshrinkastherecordsincreasesor decreases. This method is also known as Extendable hashing method.

o Thismethodmakeshashingdynamic,i.e.,itallowsinsertionordeletionwithout resulting in poor performance.

Howtosearchakey

o First,calculatethehashaddressofthekey.
o Checkhowmanybitsareusedinthedirectory,andthesebitsarecalledasi.
o Taketheleastsignificantibitsofthe hashaddress.Thisgivesanindexofthe directory.
o Nowusingtheindex,gotothedirectoryandfindbucketaddresswherethe record might be.

Howtoinsertanewrecord

o Firstly, you havetofollowthe same procedurefor retrieval, ending upin some bucket.
o Ifthereisstillspaceinthatbucket,thenplacetherecordinit.
o Ifthebucketisfull,thenwewillsplitthebucketandredistributetherecords.

Forexample:

Considerthefollowinggroupingofkeysintobuckets,dependingontheprefixoftheir hash address:
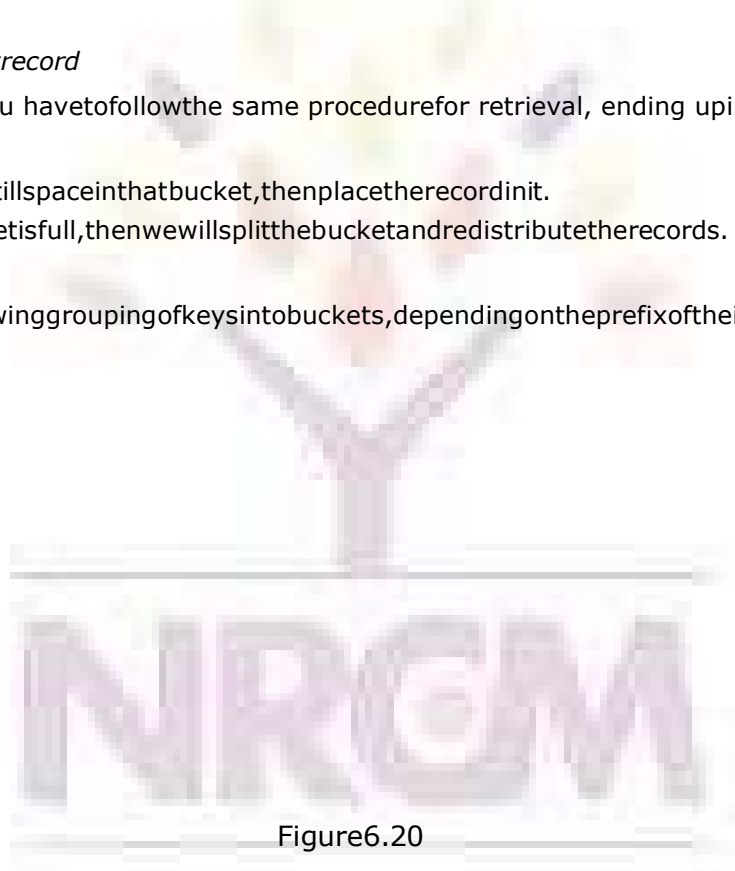
Figure6.20

Thelasttwobitsof2and4are00. Soitwillgo intobucketB0.Thelasttwobits of 5 and 6 are 01, so it will go into bucket B1. The last two bits of 1 and 3 are 10,soitwillgo intobucketB2.Thelasttwobitsof7are11,soitwillgointoB3.
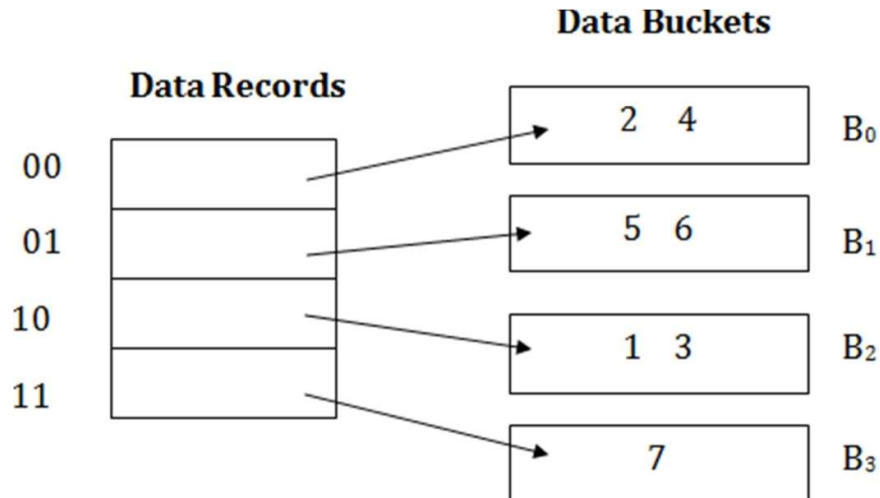
Figure6.21

Insertkey9withhashaddress10001intotheabovestructure:

o Sincekey9hashashaddress10001,itmustgointothefirstbucket.Butbucket B1 is full, so it will get split.

o Thesplittingwillseparate5,9 from6 sincelast threebitsof5,9are001,soit will go into bucket B1, and the last three bits of 6 are 101, so it will go into bucket B5.

o Keys2and4arestillinB0.TherecordinB0pointedbythe000and100entry because last two bits of both the entry are 00.

o Keys1and3arestillinB2.TherecordinB2pointedbythe010and110entry because last two bits of both the entry are 10.

o Key7arestillinB3.TherecordinB3pointedbythe111and011 entrybecause last two bits of both the entry are 11.
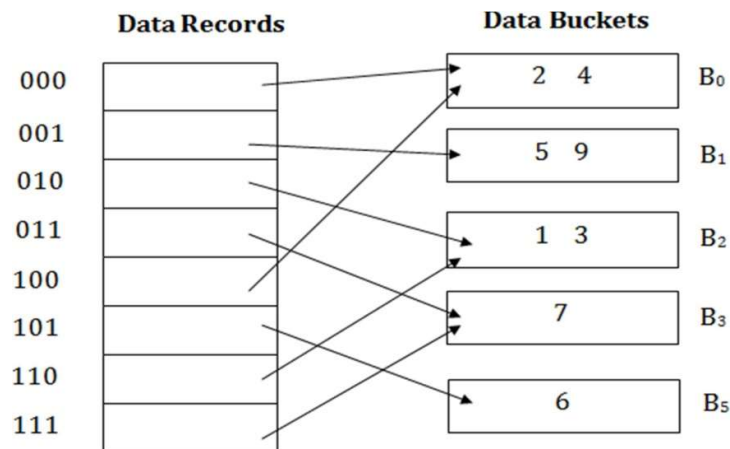
Advantagesofdynamichashing:

- o Inthismethod,theperformancedoesnotdecreaseasthedatagrowsinthe system. It simply increases the size of memory to accommodate the data.
- o Inthismethod,memoryiswellutilizedasitgrowsandshrinkswiththedata. There will not be any unused memory lying.
- o Thismethodisgood forthedynamicdatabasewheredatagrowsandshrinks frequently.

6.3.3LinearHashing:

Linear Hashing is a dynamic hashing technique, like Extendible Hashing, adjusting gracefully to inserts and deletes. In contrast to Extendible Hashing, it does not require a directory, deals naturally with collisions, and offers a lot of flexibility with respect to the timing of bucket splits.

If the data distribution is very skewed, however, overflow chains could cause Linear Hashing performance to be worse than that of Extendible Hashing.

The scheme utilizes a family of hash functions h0, h1, h2, ... , with the property that each function's range is twice that of its predecessor.

That is, if $h_i$maps a data entry into one of M buckets, h_{i+1}maps a data entry into one of 2M buckets.

The idea is best understood in terms of **rounds** of splitting. During round number *Level*, only hash functions *hLevel* and *hLevel+1* are in use. The buckets in the file at the beginning of the round are split, one by one from the first to the last bucket, thereby doubling the number of buckets.

Buckets during a Round in Linear Hashing
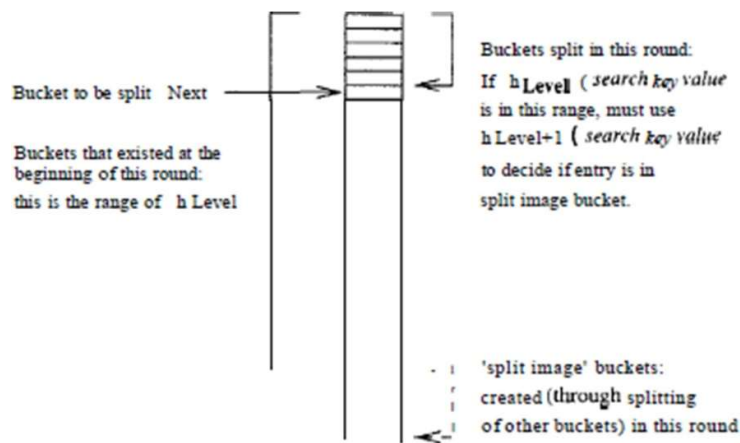
UnlikeExtendible Hashing, when an insert triggers a split, the bucket intowhich the data entry is inserted is not necessarily the bucket that is split. An overflow page is added to store the newly inserted data entry as in Static Hashing.

WenowdescribeLinearHashinginmoredetail.

A counter *Level* is used to indicate the current round number and is initialized to 0. The bucket to split is denoted by *Next* and is initially bucket. We denote the number of buckets in the file at the beginning of round *Level* by *NLevel*.

We can easily verify that N*Level* = N * *2Level*. Let the number of buckets at the beginning of round 0, denoted by No, be N.

Whenever a split is triggered the *Next* bucket is split, and hash function *hLevel+l* redistributes entries between this bucket (say bucket number b) and its split image the split image is therefore bucket number *b+ NLevel*.

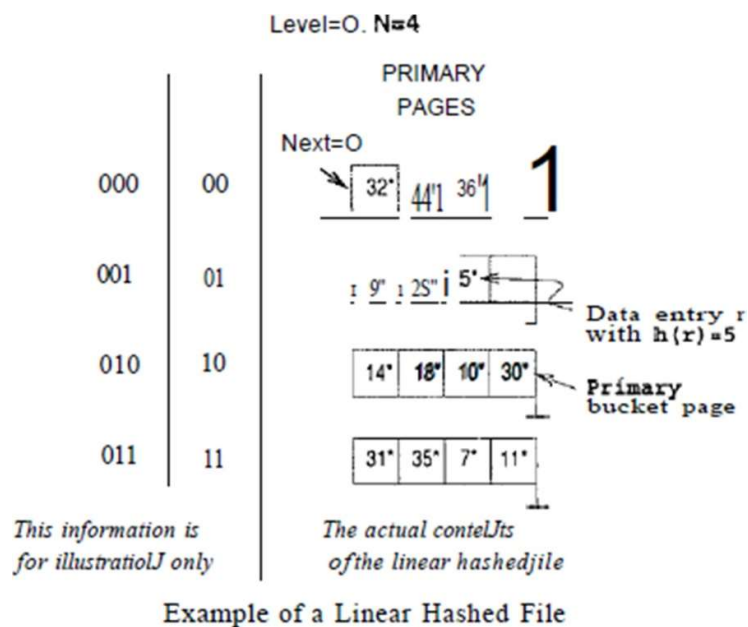AftersplittingabuckFet,thevalueofNextisincrementedby1.Intheexamplefile,

Example of a Linear Hashed File

Figure6.24

dataentry43* triggersasplit.Thefileaftercompletingtheinsertionisshowninthe figure.

Atany timein.the middleofa round*Level*,allbucketsabovebucketNext have been split, and the file contains buckets that are their split images, as illustrated.

Buckets *Next* through *NLevel* have not yet been split. If we use *hLevel* on a data entry andobtainanumberbintherangeNextthrough*NLevel*,thedataentrybelongsto

bucketb.
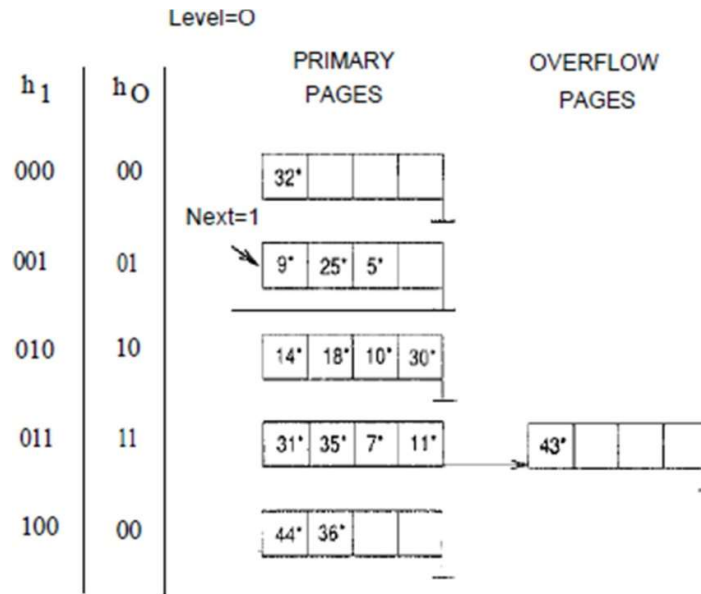
After Inserting Record r with h(T) = 43

Figure6.25

Notallinsertionstriggerasplit,ofcourse.Ifweinsert37*intothefiletheappropriate buckethasspaceforthenewdataentry.ThefileaftertheinsertionisshowninFigure
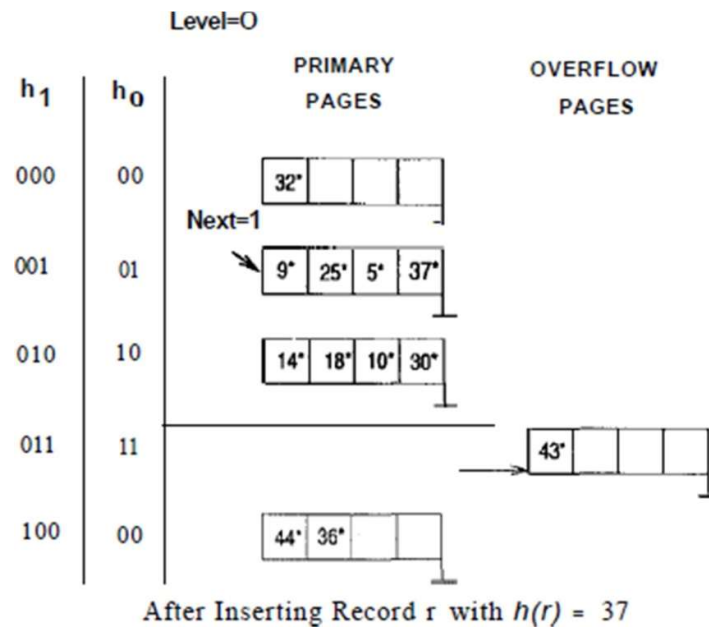

After Inserting Record r with h(r) = 37

Figure6.26

Sometimes the bucket pointed to by Next (the current candidate for splitting) is full, and a new data entry should be inserted in this bucket. In this case, a split is triggered, of course, but we do not need a new overflow bucket. This situation is illustrated by inserting 29* into the file.

When Next is equal to NLevel - 1 and a split is triggered, we split the last of the buckets present in the file at the beginning of round Level.

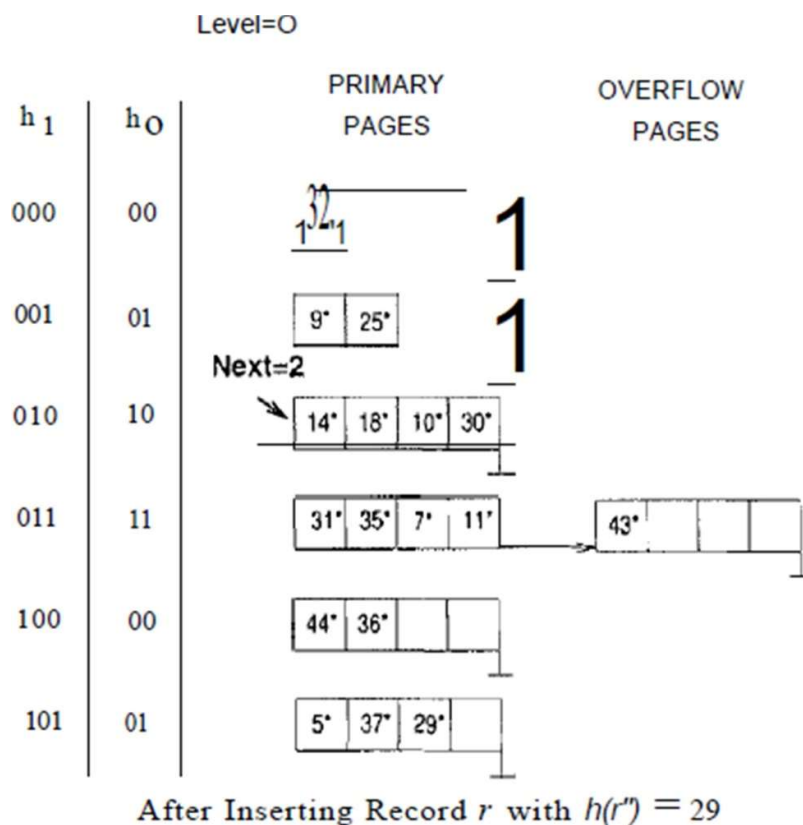

After Inserting Record r with $h(r'') = 29$

Figure6.27

The number of buckets after the split is twice the number at the beginning of the round, and we start a new round with Level incremented by 1 and Next reset toO.Incrementing Level amounts to doubling the effective range into which keys are hashed.

'We not discuss deletion in detail, but it is essentially the inverse of insertion. If the last bucket in the file is empty, it can be removed and Next can be decremented.

If we wish, we can combine the last bucket with its split image even whenit is not empty, using some criterion to trigger this merging in essentially the same way.

Extendablevs.Linearhashing:

To understand the relationship between Linear Hashing and Extendible Hashing, imagine that we also have a directory in Linear Hashingwith elements 0 to N - 1. The first splitis at bucket 0, and so we add directory element N.

we may imagine that the entire directory has been doubled at this point; however, because element1 is the same as element N + 1, element 2 is the same as element N+2, and so on, we can avoid the actual copying for the rest of the directory.

The second split occurs at bucket 1; now directory element N + 1 becomes significant and is added. At the end of the round, all the original N buckets are split, and the directory is doubled in size.

The directory analogy is useful for understanding the ideas behind Extendible and Linear Hashing.

However, the directory structure can beavoided for Linear Hashing by allocating primary bucket pages consecutively, which would allow us to locate the page for bucket i by a simple offset calculation.

For uniform distributions, this implementation of LinearHashing has a lower averagecost for equality selections.

A different implementation of Linear Hashing,in which adirectory is actually maintained, offers the flexibility of not allocating one page per bucket; null directory elements can be used as in Extendible Hashing. However, this implementation introduces the overhead of a directory level and could prove costly for large, uniformly distributed files.