GANAPATI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(POLYTECHNIC)

Mathasahi , Jagatpur , Cuttack - 754200, Odisha

"DATA BASE MANAGEMENT SYSTEM"

Prepared by:

Mr Ananta Bikram Basudev Mohapatra.

CONTENTS

Module-1:

LECTURE-1:IntroductiontoData

Introduction:

In computerized information system data are the basic resource of the organization. So, proper organization and management for data is required for organization to run smoothly. Database management system deals the knowledge of how data stored and managed on a computerized information system. In any organization, it requires accurate and reliable data for better decision making, ensuring privacy of data and controlling data efficiently.

The examples include deposit and/or withdrawal from a bank, hotel, airline or railway reservation, purchase items from supermarkets in all cases, a database is accessed.

Whatis data?

Data are the known facts or figures that have implicit meaning. It can also be defined as it is the representation of facts, concepts or instructions in a formal manner, which is suitable for understanding and processing. Data can be represented in alphabets (A-Z, a-z), digits (0-9) and using special characters (+,-.#,$, etc)

e.g:25,“ajit”etc.

Information:

Informationistheprocesseddataonwhichdecisionsandactionsarebased.Informationcanbe defined as the organized and classified data to provide meaningful values.

Eg: “TheageofRaviis25”

File:

Fileisa collectionofrelateddata storedinsecondarymemory.

FileOrientedApproach:

The traditional file oriented approach to information processing each application has a separate master file and its own set of personal file. In file oriented approach the program dependent on the files and files dependent upon the programs.

Disadvantagesoffileorientedapproach:

1) **Dataredundancyandinconsistency:**
   The same information may be written in several files. This redundancy leads to higher storage and access cost. It may lead data inconsistencythat is the various copies ofthe same data may present at multiple places for example a changed customer address may be reflected in single file but not else where in the system.

2) **Difficultyinaccessingdata:**
   Theconventional file processing system do not allow data to be retrieved in a convenient and efficient manner according to user choice.

3) **Dataisolation:**
   Because data are scattered in various files and files may be in different formats withnew application programs to retrieve the appropriate data is difficult.

4) **IntegrityProblems:**

Developers enforce data validation in the system by adding appropriate code in the various application program. How ever when new constraints are added, it is difficult to change the programs to enforce them.

5) **Atomicity:**
Itisdifficult to ensureatomicityina fileprocessingsystemwhentransactionfailureoccurs due to power failure, networking problems etc. (atomicity: either all operations of the transaction are reflected properly in the database or non are)

6) **Concurrentaccess:**
Inthe file processing systemit isnot possible to accessthe same file for transactionat same the time.

7) **Securityproblems:**
There is no security provided in file processing system to securethedatafrom unauthoriz- ed user access.

LECTURE-2: DBMS

Database:

Adatabase is organized collection of related dataof an organization stored in formatted way which is shared by multiple users.

Themainfeatureofdatainadatabaseare:

1. Itmustbewellorganized
2. Itisrelated
3. Itisaccessibleinalogicalorderwithoutanydifficulty
4. Itisstoredonlyonce

For example consider the roll no, name, address ofa student stored in a student file. It is collection of related data with an implicit meaning. Data in the database may be persistent, integrated and shared.

Persistent:

Ifdataisremovedfromdatabasedueto someexplicitrequest fromuserto remove.

Integrated:

Adatabase can be a collection of data from differentfiles and when any redundancy among those files are removed from database is said to be integrated data.

SharingData:

Thedatastoredinthedatabasecanbeshared bymultipleusers simultaneouslywithout affectingthe correctness of data.

WhyDatabase:

Inordertoovercomethe limitationofa file system, anewapproachwasrequired. Henceadatabase approachemerged.Adatabase isapersistent collectionoflogicallyrelateddata.The initialattempts were to provide a centralized collection of data.Adatabase has a self describing nature. It contains not only the data sharing and integration of data of an organization in a single database.

Asmall database can be handledmanually butfor a large database and havingmultiple users itis difficult to maintain it. In that case a computerized database is useful.
Theadvantagesofdatabasesystemovertraditional,paper basedmethodsofrecordkeeping are:

- **Compactness**:Noneed forlargeamountofpaper files
- **Speed**:Themachinecanretrieveandmodifythedatamorefaster waythenhumanbeing
- **Lessdrudgery**:Muchofthemaintenanceoffilesbyhandis eliminated
- **Accuracy:** Accurate, up-to-date information is fetched as per requirement ofthe user at any time.

DatabaseManagementSystem(DBMS):

Adatabase management systemconsists ofcollection ofrelated data and refers to aset ofprograms for defining, creation, maintenance and manipulation of a database.

FunctionofDBMS:

1. **Definingdatabaseschema**:itmustgivefacilityfordefiningthedatabasestructurealso specifies access rights to authorized users.
2. **Manipulation of the database:** The dbms must have functions like insertion ofrecord into database, updation of data, deletion of data, retrieval of data
3. **Sharing of database:** The DBMS must share data items for multiple users by maintaining consistency of data.
4. **Protectionofdatabase:**Itmustprotectthedatabaseagainstunauthorizedusers.
5. **Databaserecovery:**IfforanyreasonthesystemfailsDBMSmustfacilitatedatabase recovery.

Advantages of DBMS:

Reductionofredundancies:
Centralized control of data by the DBA avoids unnecessary duplication of data and effectively reduces the total amount of data storage required avoiding duplication in the elimination of the inconsistencies that tend to be present in redundant data files.

SharingofData:
A database allows the sharing of data under its control by any number of application programs or users.

DataIntegrity:
Data integrity means that the data contained in the database is both accurate and consistent. Therefore data values being entered for storage could be checked to ensure that they fall with in a specified range and are of the correct format.

DataSecurity:
The DBAwho has the ultimate responsibility for the data in the dbms can ensure that proper access procedures are followed including proper authentication to access to the DataBase System and additional check before permitting access to sensitive data.

ConflictResolution:
DBA resolve the conflict on requirements of various user and applications. The DBA chooses the best file structure and access method to get optionalperformance for the application.

DataIndependence:
Data independence is usually considered from two points of views; physically data independence and logical data independence.

Physical Data Independenceallows changes in the physical storage devices ororganization ofthe files to be made without requiring changes in the conceptual view or any of the external views and hence in the application programs using the data base.

Logical Data Independence indicatesthat the conceptualschema can be changed without affecting the existing external schema or any application program.

DisadvantageofDBMS:

1. DBMSsoftwareandhardware(networkinginstallation)cost ishigh
2. The processing overhead by the dbms for implementation of security, integrity and sharing of the data.
3. Centralizeddatabasecontrol
4. Setupofthedatabasesystemrequires moreknowledge,money,skills, and time.
5. Thecomplexityofthedatabasemayresultinpoorperformance.

LECTURE-3:3levelArchitectureofDBMS

Database Basics:

Data Item:
Thedataitemisalsocalledasfieldindataprocessingandisthesmallestunitofdatathathas meaning to its users.
Eg:“e101”,”sumit”

Entitiesand attributes:
Anentityis athing orobject intherealworld thatisdistinguishable fromallotherobjects Eg: Bank, employee, student

Attributesarepropertiesarepropertiesofanentity. Eg:
Empcode, ename, rolno, name

Logicaldataandphysicaldata:
Logicaldataarethedataforthetablecreatedbyuser inprimarymemory. Physical datarefers to the datastored in the secondary memory.

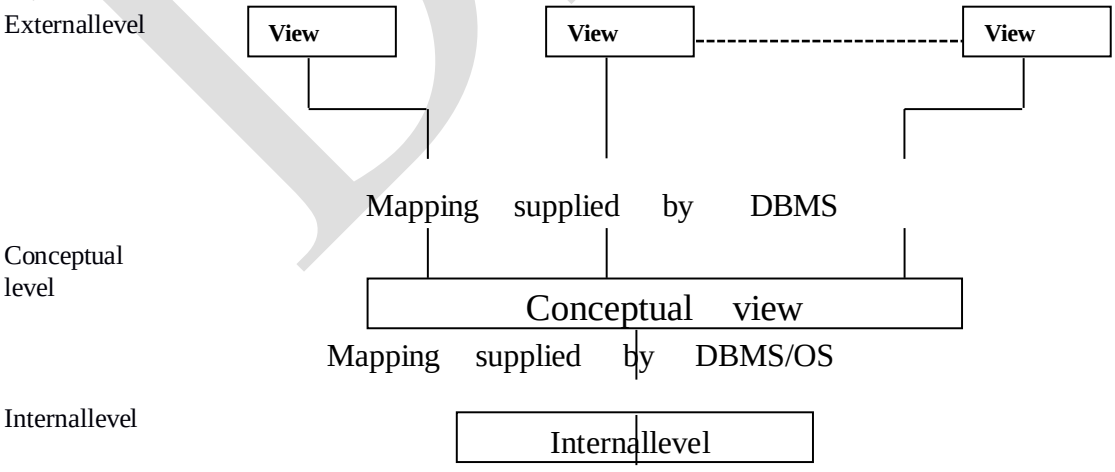
Schemaandsub-schema:
Aschema is a logicaldatabasedescription and is drawnas achart ofthetypesofdatathat areused. It gives the names of the entities andattributes and specifythe relationships betweenthem.

Adatabaseschemaincludessuchinformationas:

- Characteristicsofdataitemssuchasentitiesand attributes.
- Logicalstructuresandrelationshipsamongthesedataitems.
- Formatfor storagerepresentation.
- Integrityparameterssuchasphysicalauthorizationandbackuppolicies.

A *subschema*is derived schema derivedfrom existing schema as per the user requirement. Theremay be more then one subschema create for a single conceptual schema.

ThreeLevelArchitectureofDBMS:



Adatabasemanagementsystemthatprovidesthreelevelofdataissaidtofollowthree-level architecture .

- Externallevel
- Conceptuallevel
- Internallevel

ExternalLevel:
Theexternal level is at the highest level of database abstraction . At this level, there will be many viewsdefine fordifferent usersrequirement.Aviewwilldescribeonlyasubset ofthedatabase.Any number of user views may exist for a given global schema(coneptual schema).

For example, each student has different view of the time table. the view of a student of BTech(CSE) is different from the view of the student of Btech (ECE). Thus this level of abstraction is concerned with different categories of users.
Eachexternalviewisdescribed by meansofaschemacalledsubschema.

ConceptualLevel:
Atthis level of database abstraction all the databaseentities and the relationships among them are included. One conceptual view represents the entire database. This conceptual view is defined bythe conceptual schema.

The conceptual schema hides the details of physical storage structures and concentrate ondescribing entities, data types, relationships, user operations and constraints.

Itdescribes all the recordsand relationships included in the conceptual view. There is only one conceptual schema per database. It includes feature that specify the checks to relation data consistency and integrity.

Internallevel:
It is the lowestlevelofabstractionclosest to the physicalstoragemethod used. It indicates howthe datawillbe storedand describes the data structures and accessmethods to be used by the database. The internal view is expressed by internal schema.

Thefollowingaspectsareconsideredatthis level:

1. Storageallocatione.g:B-tree,hashing
2. Accesspathseg.specificationofprimaryand secondarykeys,indexesetc
3. Miscellaneous eg. Data compression and encryptiontechniques, optimization ofthe internal structures.

DatabaseUsers:

NaiveUsers:
Users who need not be awareofthe presence ofthe database systemoranyother systemsupporting their usage are considered naïve users.Auserofanautomatic teller machine fallsonthiscategory.

OnlineUsers:
These are users who may communicate with the database directly via an online terminal or indirectlyvia a user interface and application program. These users are aware of the database system and also know the data manipulation language system.

ApplicationProgrammers:
Professional programmers who are responsible for developing application programs or user interfaces utilized by the naïve and online user falls into this category.

DatabaseAdministration:
Apersonwho hascentralcontrolover thesystemiscalleddatabaseadministrator. The function of DBA are :

1. CreationandmodificationofconceptualSchemadefinition
2. Implementationofstoragestructureand accessmethod.
3. Schemaandphysicalorganizationmodifications.
4. Grantingofauthorizationfor dataaccess.
5. Integrityconstraints specification.
6. Executeimmediate recoveryprocedure incase of failures
7. Ensure physicalsecuritytodatabase

Databaselanguage:

1) **Datadefinitionlanguage(DDL):**
   DDL is used to define database objects .The conceptual schema is specified by a set of definitions expressed by this language. It also gives some details about how to implement this schema in the physical devices used to store the data. This definition includes all the entity sets and their associated attributes and their relationships. The result of DDL statements will bea set oftables that are stored in special file called data dictionary.

2) **DataManipulationLanguage(DML):**
   ADML is a language that enables users to access or manipulate data stored in the database. Datamanipulation involves retrievalofdatafromthe database, insertionofnew dataintothe database and deletion of data or modification of existing data.

   There are basicallytwotypesofDML:
   - **Procedural**:Whichrequiresauser tospecifywhatdataisneededandhowtogetit.
   - **Non-Procedural:**whichrequiresausertospecifywhatdataisneededwithout specifying how to get it.

3) **DataControlLanguage(DCL):**
   This language enables user to grant authorization and canceling authorization of database objects.

LECTURE-4:Elementsof DBMS

Elementsof DBMS:

DMLPre-Compiler:
It converts DML statements embedded in an application program to normal procedure calls in the host language. The pre-complier must interact with the query processor in order to generate the appropriate code.

DDLCompiler:
The DDLcompiler convertsthe data definitionstatements into asetoftables. These tables contains information concerning the database and are in a formthat can be used byother components of the dbms.

File Manager:
File manager manages the allocation of space on disk storage and the data structure used to represent information stored on disk.

Database Manager:
Adatabase manager is a program module which provides the interface between the low level data stored in the database and the application programs and queries submitted to the system.

Theresponsibilitiesofdatabasemanagerare:

1. **Interaction with File Manager**:The datais storedonthe disk using the file systemwhich is provided by operating system. The database manager translate the different DML statements into low-levelfile systemcommands sothe database manager is responsible for the actual storing, retrieving and updating of data in the database.

2. **Integrity Enforcement:** The data values stored in the database must satisfy certain constraints (eg: the age of a person can't be less then zero). These constraints are specified byDBA. Datamanager checkstheconstraintsand ifit satisfiesthenit storesthedatainthe database.

3. **Security Enforcement:** Data manager checks the security measures for database from unauthorized users.

4. **Backup and Recovery:** Database manager detects the failures occur due to different causes (like disk failure, power failure, deadlock, software error) and restores the database to original state of the database.

5. **Concurrency Control:** When several users access the same database file simultaneously, there may be possibilities of data inconsistency. It is responsible of database manager to control the problems occur for concurrent transactions.

QueryProcessor:
The queryprocessorusedtointerprettoonlineuser's queryandconvertitintoanefficientseriesof operationsinaformcapableofbeingsenttothedatamanagerforexecution.Thequeryprocessor

usesthedatadictionarytofindthedetailsofdatafileandusingthisinformationitcreatequery plan/access plan to execute the query.

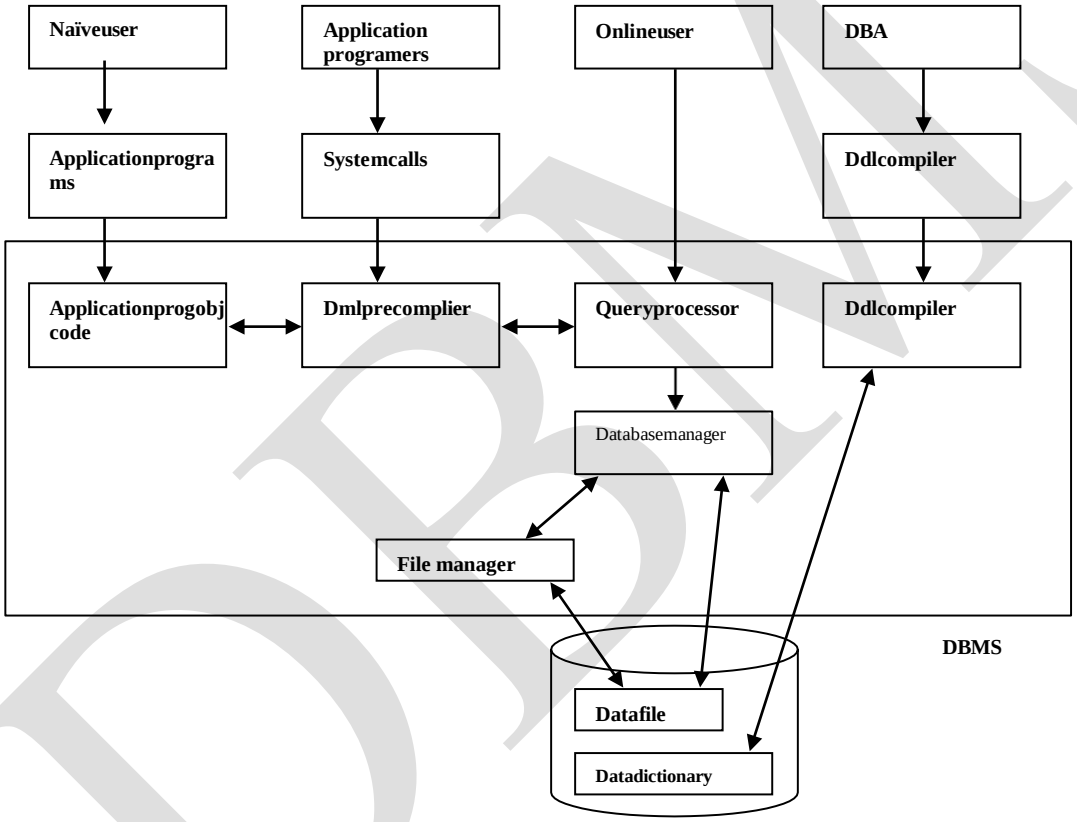
DataDictionary:

Datadictionaryisthetablewhichcontainstheinformationaboutdatabaseobjects.Itcontains information like

1. external,conceptualandinternaldatabasedescription
2. descriptionofentities,attributesaswellasmeaningofdata elements
3. synonyms, authorizationandsecuritycodes
4. databaseauthorization

Thedatastored inthedatadictionary iscalled*meta data*.

DBMSSTRUCTURE:



Que: ListfoursignificantdifferencesbetweenaFile-ProcessingSystemandaDBMS.

Ans: Some major differences between a database management systemand a file-processing system are:

- Both systems contain a collection of data and a set of programs which access that data. A database management system coordinates both the physical and the logical access to thedata, whereas a file-processing system coordinates only the physical access.

- A database management system reduces the amount of data duplication by ensuring that a physical piece of data is available to all programs authorized to have access to it, where as data written by one program in a file-processing system may not be readable by another program.

- A database management system is designed to allow flexible access to data (i.e., queries), whereas a file-processing system is designed to allow predetermined access to data (i.e., compiled programs).

- Adatabase management system is designed to coordinate multiple users accessing the same data at the same time. A file-processing system is usually designed to allow one or more programs to access different data files at the same time. In a file-processing system, a file can be accessed by two programs concurrently only if both programs have read-only access to the file.

Que: Explainthedifferencebetweenphysicalandlogicaldata independence.

Ans:

- Physical data independence is the ability to modify the physical scheme without making it necessary to rewrite application programs. Such modifications include changing from unblocked to blocked record storage, or from sequential to random access files.

- Logicaldata independence is the abilityto modifythe conceptualscheme without making it necessaryto rewrite application programs. Such a modification might be adding a field to a record; an application program's view hides this change from the program.

Que: List five responsibilities of a database management system. For each responsibility, explain the problems that would arise if the responsibility were not discharged.

Ans: Ageneralpurposedatabasemanager(DBM) has five responsibilities:

a. interactionwiththefilemanager.
b. integrityenforcement.
c. securityenforcement.
d. backupand recovery.
e. concurrencycontrol.

If these responsibilities were not met by a given DBM (and the text points out that sometimes a responsibility is omitted by design, such as concurrency control on a single-user DBM for a micro computer) the following problems can occur, respectively:

a. No DBM can do without this, if there is no file manager interaction then nothing stored inthe files can be retrieved.

b. Consistency constraints may not be satisfied, when account balances could go below the minimum allowed, employees could earn too much overtime (e.g.,hours > 80) or, airline pilots may fly more hours than allowed by law.

c. Unauthorizedusersmayaccessthedatabase,orusersauthorizedtoaccesspartofthe

database may be able to access parts of the database for which they lack authority. For example, a high school student could get access to national defense secret codes, or employees could find out what their supervisors earn.

d. Data could be lost permanently, rather than at least being available in a consistent state that existed prior to a failure.

e. Consistencyconstraints may be violated when intgrityconstraints failed ina transaction. For example, incorrect bank balances might be reflected due to simultaneous withdrawals and deposits, and so on.

Que. Whatarefivemainfunctionsofadatabaseadministrator?

Ans: Fivemainfunctionsofadatabaseadministrator are:

- Tocreatetheschemedefinition
- Todefinethestoragestructureandaccess methods
- Tomodifytheschemeand/orphysicalorganizationwhennecessary
- Tograntauthorizationfordataaccess
- Tospecifyintegrityconstraints

Que: Listsixmajorstepsthatyouwouldtakeinsettingupadatabaseforaparticularenterprise.

Ans: Sixmajorstepsinsettingupadatabaseforaparticularenterpriseare:

- Definethehighlevelrequirementsoftheenterprise(thisstepgeneratesadocument known as the system requirements specification.)
- Defineamodelcontainingallappropriate typesofdata anddatarelationships.
- Definetheintegrityconstraintsonthe data.
- Definethephysicallevel.
- For each known problem to be solved on a regular basis (e.g., tasks to be carried out by clerks or Web users) define a user interface to carry out the task, and write the necessary application programs to implement the userinterface.
- Create/initializethedatabase.

EXERCISE:

1. Whatisdatabasemanagementsystem?
2. Whatarethedisadvantageoffileprocessingsystem?
3. Stateadvantageanddisadvantageofdatabasemanagementsystem.
4. Whataredifferenttypesofdatabaseusers?
5. Whatisdatadictionaryandwhatareitscontents?
6. WhatarethefunctionsofDBA?
7. Whatarethedifferentdatabaselanguages?Explainwithexample.
8. Explainthethreelayerarchitectureof DBMS.
9. Differentiatebetweenphysicaldataindependenceandlogicaldataindependence.
10. Explainthefunctionsofdatabasemanager.
11. Explainmetadata.

LECTURE-5:ER-MODEL

DataModel:

The data model describes the structure of a database. It is a collection of conceptual tools for describing data, data relationships and consistency constraints and various types of data modelssuch as

1. Objectbasedlogicalmodel
2. Recordbased logical model
3. Physicalmodel

Typesofdatamodel:

1. Objectbasedlogicalmodel
    a. ER-model
    b. Functionalmodel
    c. Objectoriented model
    d. Semanticmodel
2. Recordbased logical model
    a. Hierarchicaldatabasemodel
    b. Networkmodel
    c. Relationalmodel
3. Physicalmodel

Entity RelationshipModel(ERModel)

The entity-relationship data model perceives the real world as consisting of basic objects, called entities and relationships among these objects. It was developed to facilitate database design by allowing specification of an enterprise schema which represents the overall logical structure of a data base.

MainFeaturesofER-MODEL:

- Entityrelationshipmodelisa high levelconceptualmodel
- It allows us to describe the data involvedin a real world enterprisein terms of objects and their relationships.
- Itiswidelyused todevelop an initialdesignofa database
- It provides a set of useful concepts that make it convenient for a developer to move from a basic set of information to a detailed and description of information that can be easily implemented in a database system
- Itdescribesdataasacollectionofentities,relationshipsand attributes.

Basic Concepts:

TheE-Rdatamodelemploysthreebasicnotions:entitysets,relationshipsetsandattributes.

Entity Sets:

Anentity is a “thing” or “object” in the realworld that is distinguishable fromall other objects. For example, each person in an enterprise is an entity. An entity has a set properties and the values for some set of properties mayuniquely identify an entity. BOOK is entityand its properties (called as attributes) bookcode, booktitle, price etc.

Anentitysetisasetofentities ofthesame type thatshare thesame properties,orattributes.Theset

ofallpersonswhoarecustomersatagiven bank.

Attributes:
Anentityisrepresentedbyasetofattributes.Attributesaredescriptivepropertiespossessedby each member of an entity set.

Customer is an entity and its attributes are **customerid, custmername, custaddress** etc.

Anattributeasused intheE-Rmodel,canbecharacterized bythe followingattributetypes.

a) **SimpleandCompositeAttribute:**
Simple attributes are the attributes which can't be divided into sub parts, e.g. customerid, empno
Composite attributes are the attributes which can be divided into subparts, e.g. name consisting of first name, middle name, last name and address consisting of city, pincode, state.

b) **Single-ValuedandMulti-ValuedAttribute:**
The attribute having unique value is single –valued attribute, e.g. empno, customerid, regdno etc.
The attribute having more than one value is multi-valued attribute, eg: phone-no, dependent name, vehicle.

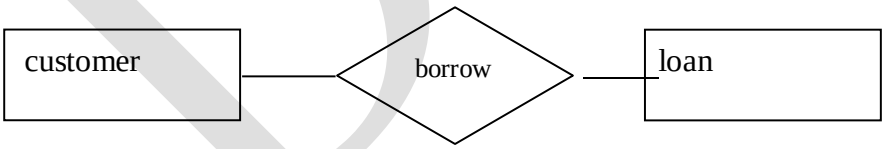
c) **DerivedAttribute:**
The values for this type of attribute can be derived from the values of existing attributes, e.g. age which can be derived from currentdate – birthdate and experience_in_year can be calculated as currentdate-joindate.

d) **NULLValuedAttribute:**
TheattributevaluewhichisnotknowntouseriscalledNULLvalued attribute.

RelationshipSets:
Arelationship is an association among severalentities.Arelationship set is a set ofrelationships of the same type. Formally, it is a mathematical relation on n>=2 entity sets. If E_1, E_2…$E_n$are entity sets, then a relation ship set R is a subset of

$$\{(e_1,e_2,\ldots e_n)|e_1 \epsilon E_1, e_2 \epsilon E_2 .., e_n \epsilon E_n\}$$

where $(e_1,e_2,\ldots e_n)$ is a relation ship.

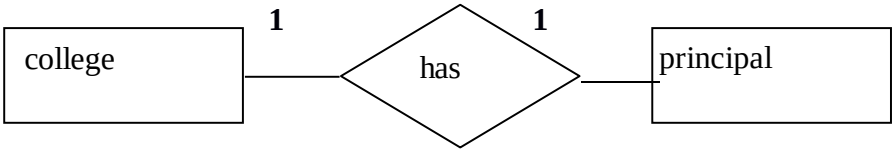


Consider the two entitysets customer and loan. We define the relationship set borrow to denote the association between customers and the bank loans that the customers have.

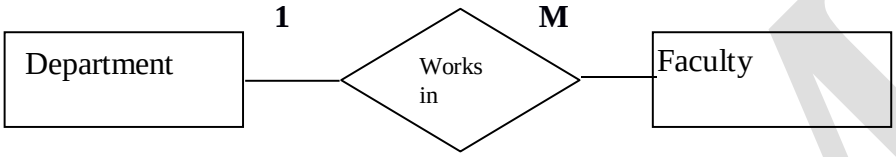
Mapping Cardinalities:
Mapping cardinalities or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set. Mapping cardinalities are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than twoentity sets. For a binary relationship set R between entity setsAand B, the mapping

cardinalitiesmustbeoneofthe following:

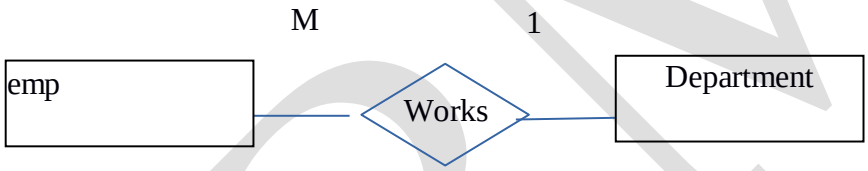
1. OnetoOne:
Anentityin A isassociatedwithatmostoneentityinB,andanentityinBisassociatedwithat most one entity in A.
Eg:relationshipbetweencollegeandprincipal



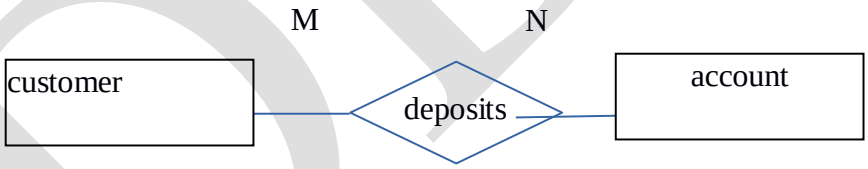
2. Oneto Many:
AnentityinAisassociated withany number ofentities in B.Anentityin B isassociated withatthe most one entity in A.
Eg:Relationshipbetweendepartmentandfaculty



3. ManytoOne:
AnentityinAisassociatedwithatmostoneentityinB.AnentityinBisassociatedwithany number in A.



4. ManytoMany:
EntitiesinAand Bareassociated withany numberofentitiesfromeachother.



Moreabout EntitiesandRelationship:

RecursiveRelationships:
When the same entity type participates more than once in a relationship type in different roles, the relationship types are calledrecursive relationships.

Participation Constraints:
Theparticipationconstraintsspecifywhethertheexistenceofanyentitydependsonitsbeing related to another entity via the relationship. There are two types of participation constraints

a) Total : When all the entities from an entity set participate in a relationship type, is called total participation. For example, the participation of the entity set student on the relationship set must 'opts'is said to be total because everystudent enrolled must opt for a course.

b) Partial: When it is not necessary for all the entities from an entity set to particapte in a relationship type, it is called partial participation. For example, the participation of the entity set student in 'represents'is partial, since not everystudent in a class is a class representative.

WeakEntity:
Entitytypes that do not containanykeyattribute, and hence can not be identified independentlyare called weak entity types. Aweak entity can be identified by uniquely only by considering some of its attributes in conjunction with the primary key attribute of another entity, which is called the identifying owner entity.

Generallyapartialkeyisattachedtoaweakentitytypethatisusedforuniqueidentificationof weak entities related to a particular owner type. The following restrictions must hold:

- Theownerentitysetandtheweakentitysetmustparticipateinonetomayrelationshipset. Thisrelationshipsetiscalledtheidentifyingrelationshipsetoftheweakentityset.
- Theweakentitysetmusthavetotalparticipationintheidentifyingrelationship.

Example:
Consider the entity type Dependent related to Employee entity, which is used to keep track of the dependents of each employee. The attributes of Dependents are: name, birthdate, sex and relationship. Each employee entityset is said to its own the dependent entities that are related to it. However, not that the 'Dependent'entitydoes not exist of its own, it is dependent onthe Employee entity.

Keys:

SuperKey:
Asuperkeyisaset ofoneormoreattributesthattakencollectively, allowusto identifyuniquelyan entity in the entityset. For example , customer-id, (cname, customer-id), (cname, telno)

CandidateKey:
InarelationR,acandidatekeyforRisasubsetofthesetofattributesofR,whichhavethe following properties:

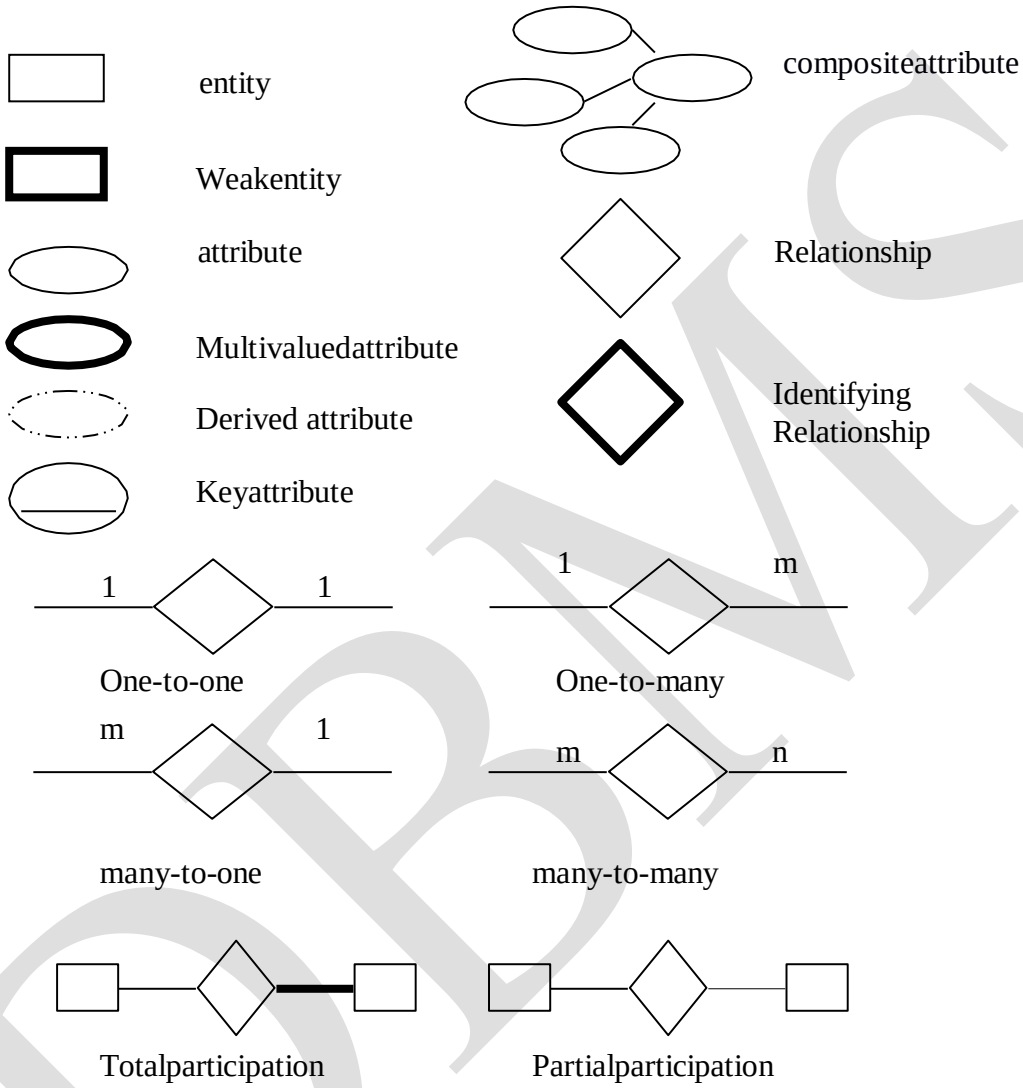
1. *Uniqueness:N*otwodistincttuplesinR havethesamevaluesforthecandidatekey
2. *Irreducible:*Noproper subsetof the candidate key has the uniqueness property thatis the candidate key. Eg: (cname,telno)

PrimaryKey:
The primarykey is the candidate keythat is chosen bythe database designer as the principal means ofidentifying entities within anentityset. The remaining candidate keys ifany, are called *Alternate Key*.

LECTURE-6:ER-DIAGRAM:

TheoverallogicalstructureofadatabaseusingER-modelgraphicallywiththehelpofanER- diagram.

SymbolsuseER-diagram:



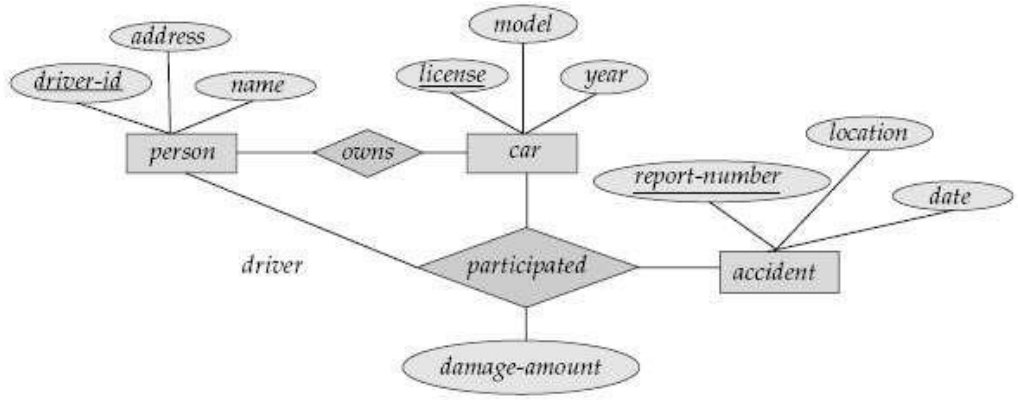


Figure 2.1 E-R diagram for a Car-insurance company.

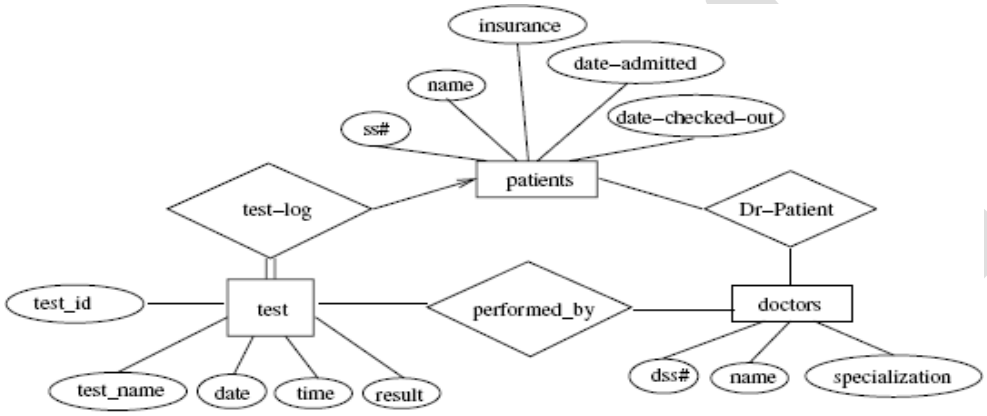


Figure 2.2 E-R diagram for a hospital.

AUniveristyregistrar'sofficemaintainsdataaboutthefollowing entities:

(a) Course,includeingnumber,title,credits,syllabusandprereqisites

(b) courseoffering,includingcoursenumber,year,semester,sectionnumber,instructortimings, and class room

(c) Studentsincludingstudent-id,nameandprogram

(d) Instructors,includingidentificationnumber,name,departmentandtitle

further,theenrollment ofstudents incoursesand gradesawarded to studentsineach coursetheyare enrolled for must be appropriate modeled.

Construct an E-R diagram for the registrar's office. Document all assumptions that you may makeabout the mapping constratints

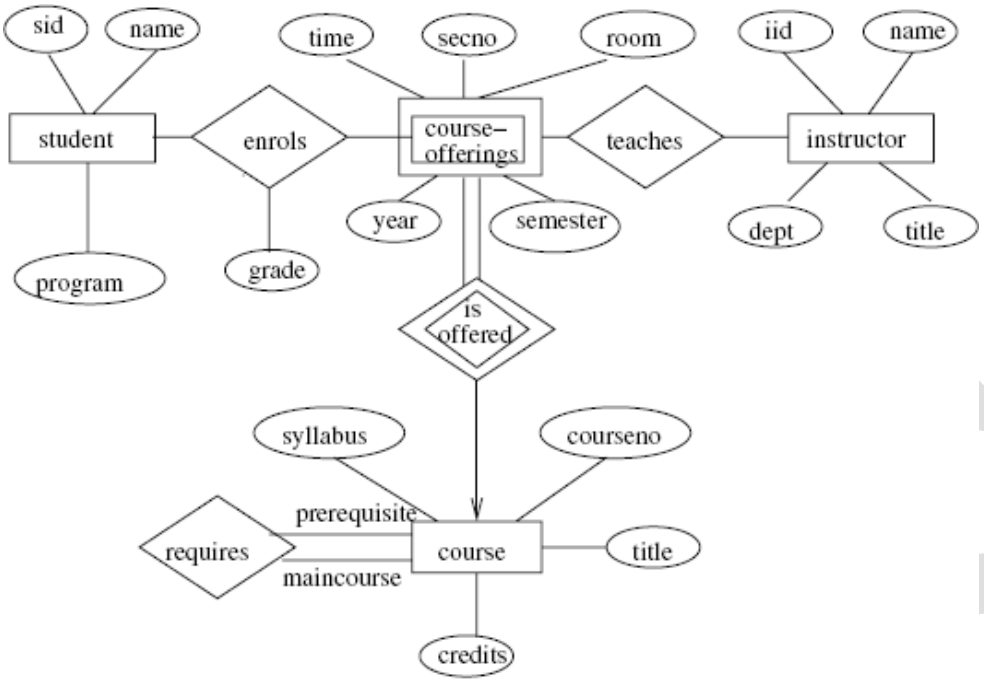


Figure 2.3 E-R diagram for a university.

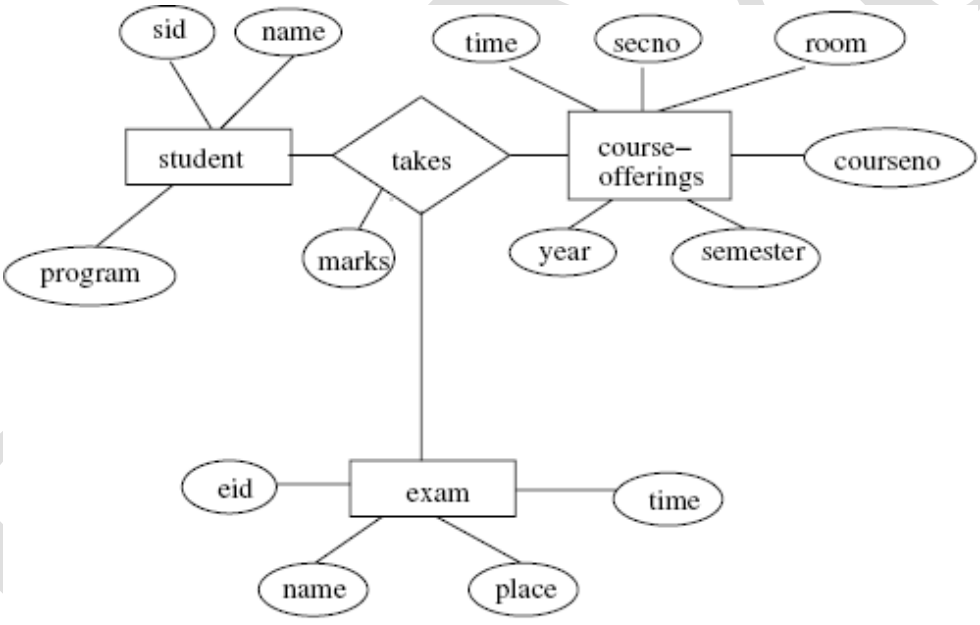


Figure 2.4 E-R diagram for marks database.

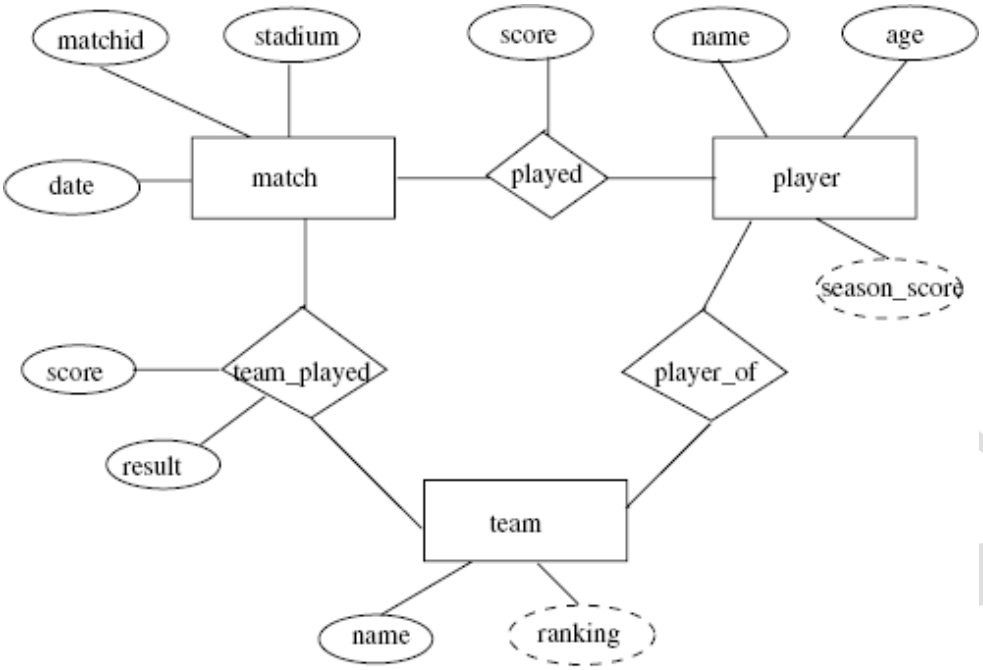


Figure 2.7 E-R diagram for all teams statistics.

Cosidet a universitydatabase forthe scheduling ofclass rooms for finalexams. This database could be modeled asthe single entityset exam, withattributescourse-name,section-number,room-number and time,Alternatively, one or more additionalentitysetswould be defined, along withrelationship sets to replae some of the attributes of the exam entity set, as

- coursewithattributesname,departmentand c-number
- sectionwithattributess-numberandenrollmentanddependentasaweakentityseton course
- roomwithattributesr-number,capacityandbuilding

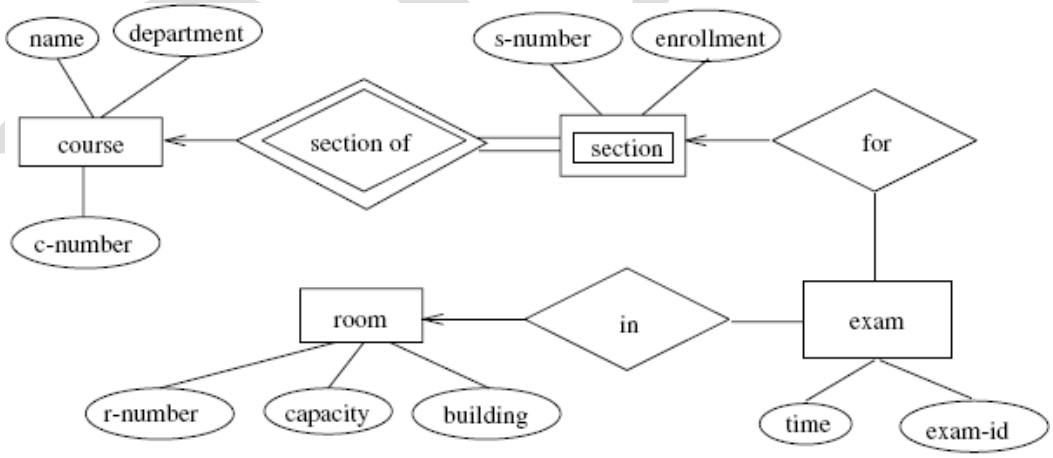


Figure 2.12 E-R diagram for exam scheduling.

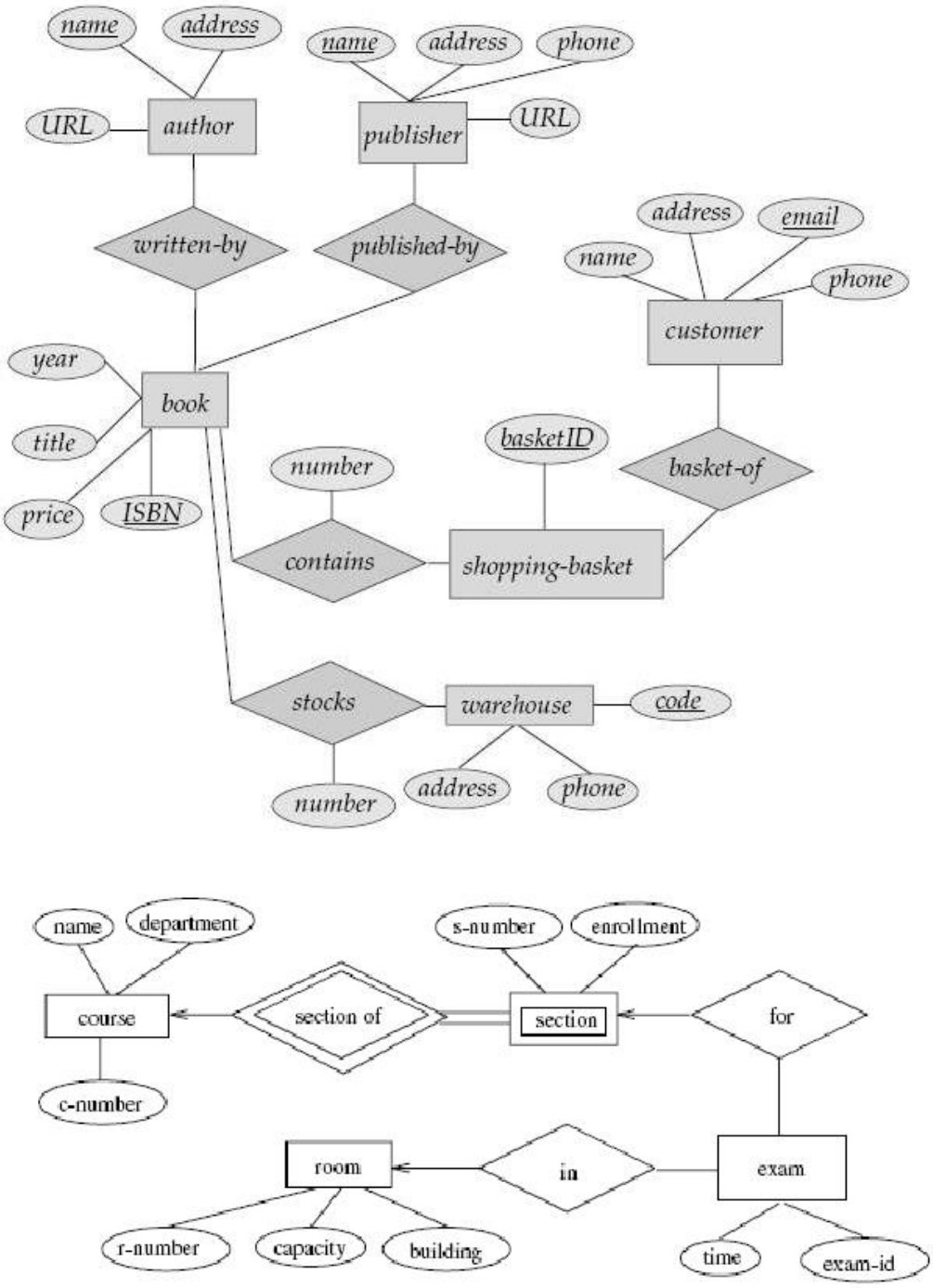


Figure 2.12 E-R diagram for exam scheduling.

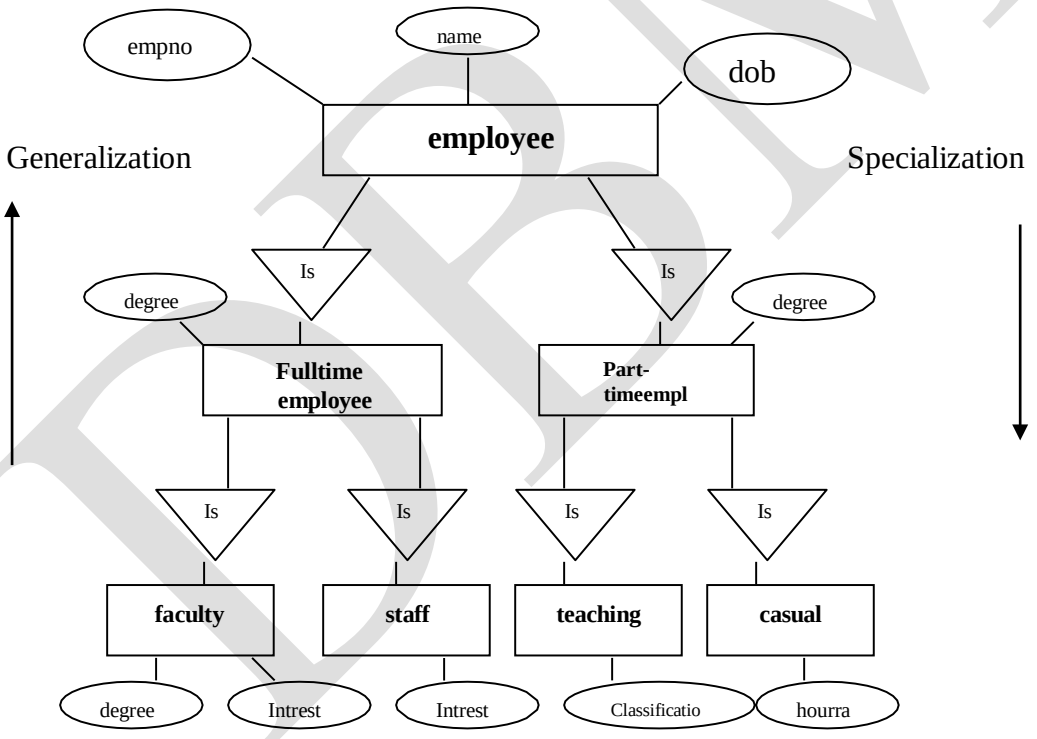
LECTURE-7:AdvancedER-Diagram:

Abstraction is the simplification mechanism used to hide superfluous details of a set of objects. It allowsoneto concentrateonthepropertiesthat areofinterest totheapplication.Therearetwomain abstraction mechanism used to model information:

Generalizationandspecialization:

Generalization is the abstracting process of viewing set of objects as a single general class by concentrating on the general characteristics of the constituent sets while suppressing or ignoring their differences. It is the union of a number of lower-levelentity types for the purpose of producing a higher-levelentitytype. For instance, student is a generalization of graduate or undergraduate, full-time or part-time students. Similarly, employee is generalization of the classesof objects cook, waiter, and cashier. Generalization is an IS_A relationship; therefore, manager IS_AN employee, cook IS_AN employee, waiter IS_AN employee, and so forth.

Specialization is the abstracting process of introducing new characteristics to an existing class of objects to create one or more new classes of objects. This involves taking a higher-level, and using additional characteristics, generating lower-level entities. The lower-level entities also inherits the, characteristics of the higher-level entity. In applying the characteristics size to car we can create a full-size, mid-size, compact orsubcompact car.Specialization maybe seenasthe reverse processof generalization addition specific properties are introduced at a lower level in a hierarchyof objects.



EMPLOYEE(**empno**,name,dob)
FULL_TIME_EMPLOYEE(**empno**,salary)
PART_TIME_EMPLOYEE(**empno**,type)

Faculty(**empno**,degree,intrest)
Staff(**empno**,hour-rate)
Teaching (**empno**,stipend)

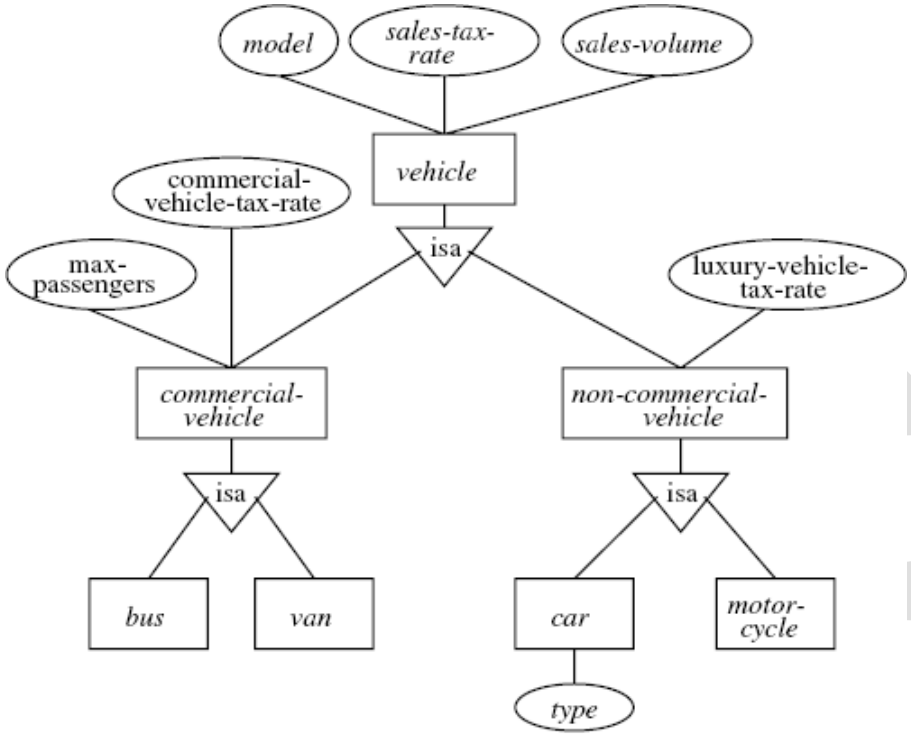
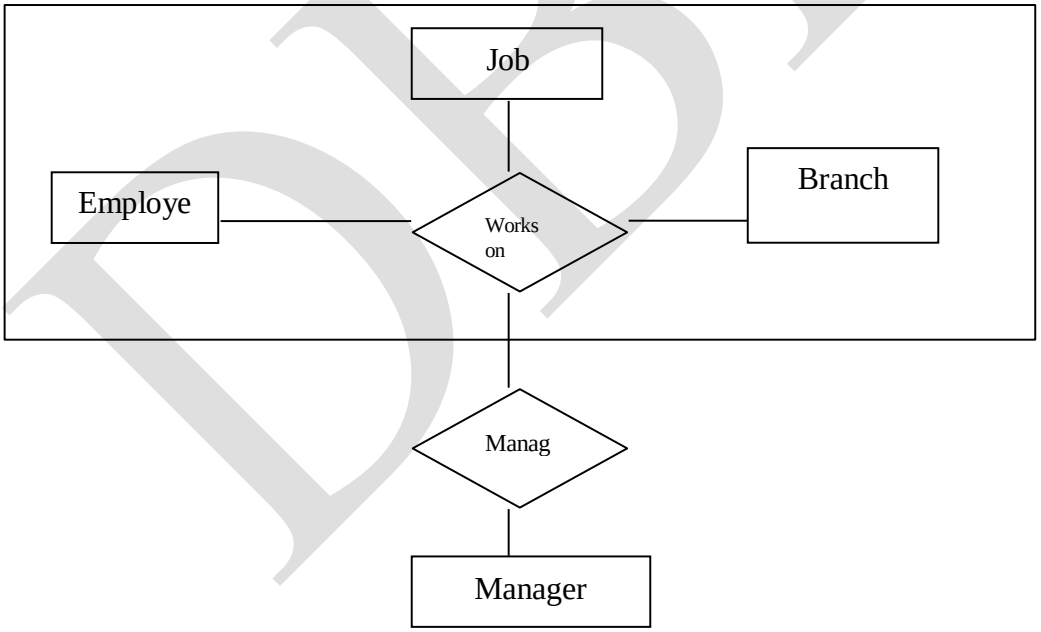


Figure 2.19 E-R diagram of motor-vehicle sales company.

Aggregation:

Aggregation is the process of compiling information on an object, there by abstracting a higherlevel object. The entity person is derived by aggregating the characteristics ofname, address, ssn. Another formoftheaggregation isabstracting arelationship objectsand viewing therelationship as an object.



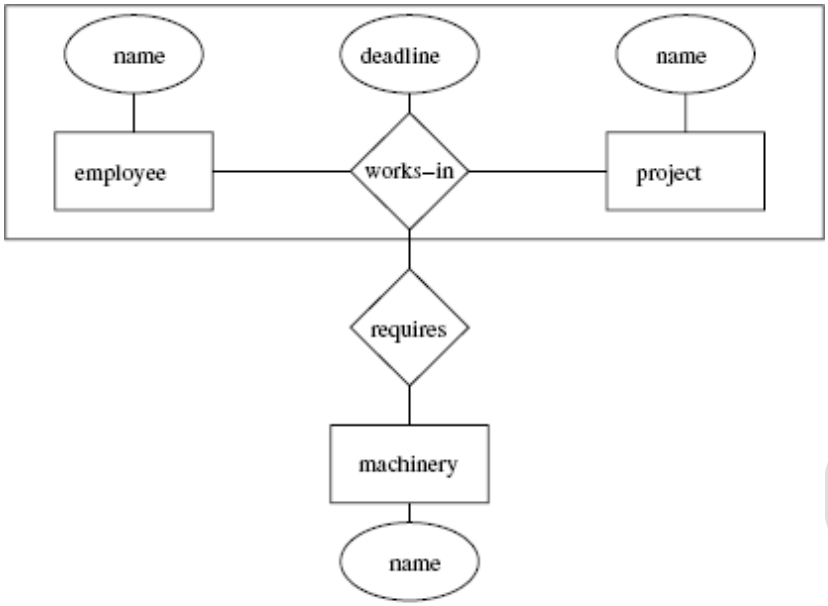


Figure 2.8 E-R diagram Example 1 of aggregation.

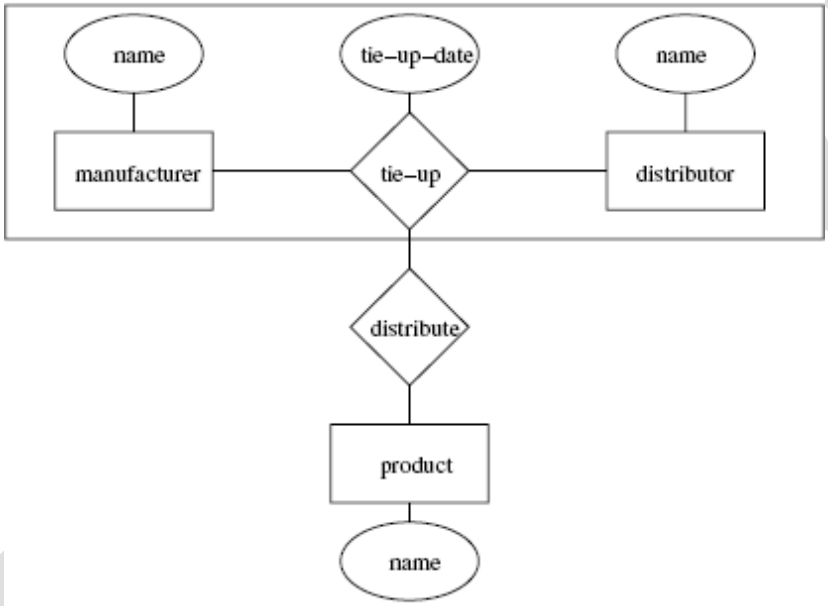
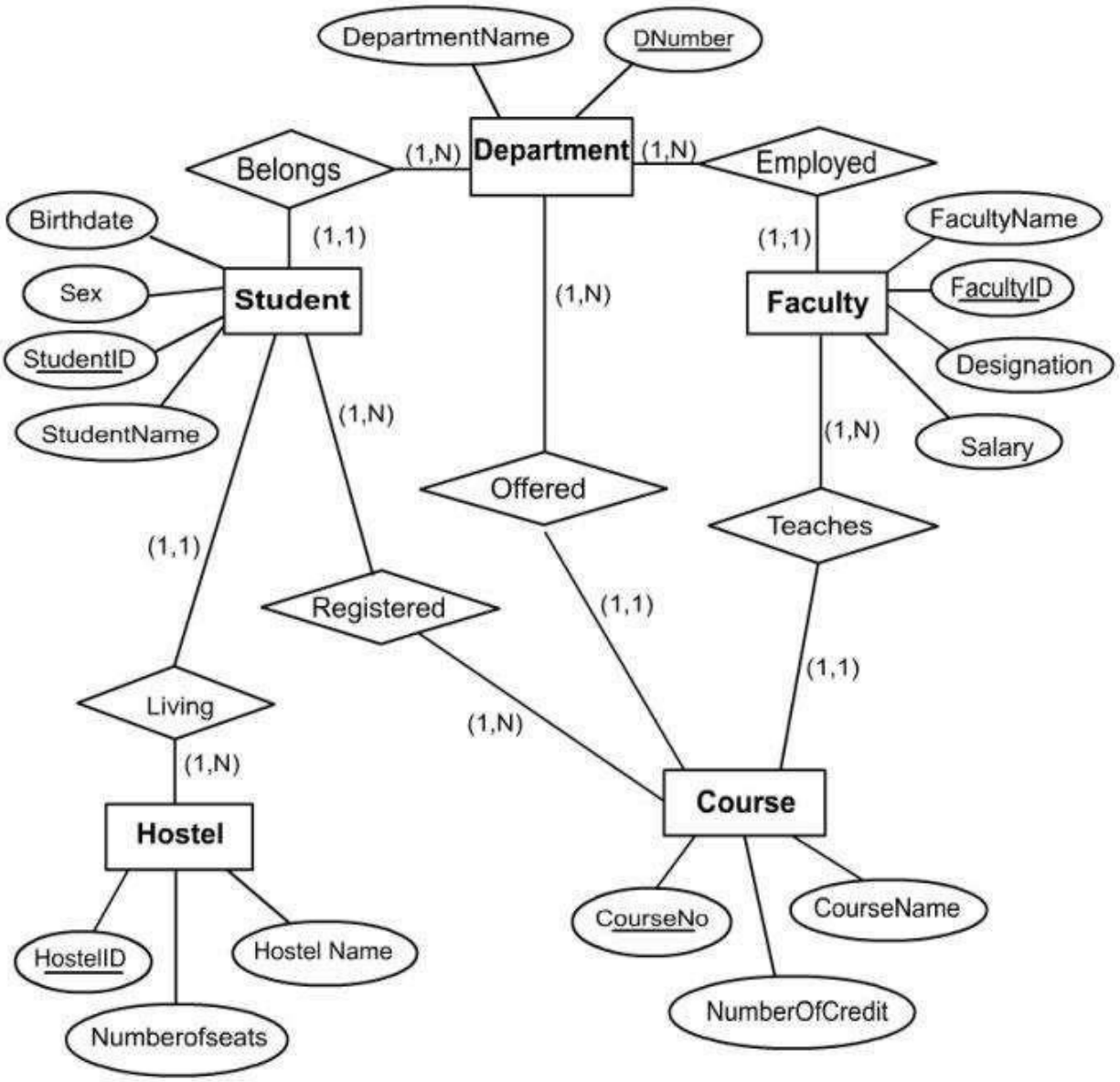


Figure 2.9 E-R diagram Example 2 of aggregation.

ER-DiagramForCollegeDatabase



LECTURE-8: ConversionofER-DiagramtoRelationalDatabase

ConversionofEntitySets:

1. For each strong entity type E in the ER diagram, we create a relation R containing all the single attributes ofE. The primarykeyofthe relationR will be one ofthe keyattributeofR.

   STUDENT(rollno (primary key),name, address)
   FACULTY(id(primary key),name ,address, salary)
   COURSE(course-id,(primarykey),course_name,duration)
   DEPARTMENT(dno(primary key),dname)

2. For each weak entity type W in the ER diagram, we create another relation R that contains allsimple attributes ofW. IfE is anowner entityofWthenkeyattribute ofE is also include In R. This key attribute of R is set as a foreign key attribute of R. Now the combination of primary key attribute of owner entity type and partial key of the weak entity type will form the key of the weak entity type

   GUARDIAN((rollno,name)(primarykey),address,relationship)

ConversionofRelationshipSets:

BinaryRelationships:

- **One-to-One Relationship:**
  For each 1:1 relationship type R in the ER-diagram involving two entities E1 and E2 we choose one of entities(say E1) preferably with total participation and add primary key attributeofanother E asaforeignkeyattribute inthetableofentity(E1).Wewillalso include all the simple attributes of relationship type R in E1 if any, For example, the department relationship has been extended tp include head-id and attribute of the relationship.
  DEPARTMENT(D_NO,D_NAME,HEAD_ID,DATE_FROM)

- **One-to-Many Relationship:**
  For each1:NrelationshiptypeR involvingtwo entitiesE1andE2, we identifytheentitytype (say E1) at the N-side of the relationship type R and include primary key of the entity on the other side of the relation (sayE2) as a foreign key attribute inthe table of E1. We include all simple attribute (or simple components ofa composite attribute ofR (ifany) in the table E1)

  Forexample:
  The worksinrelationship betweenthe DEPARTMENTand FACULTY. Forthisrelationship choose the entity at N side, i.e, FACULTY and add primary key attribute of another entity DEPARTMENT i.e., DNO as a foreignkeyattribute inFACULTY.

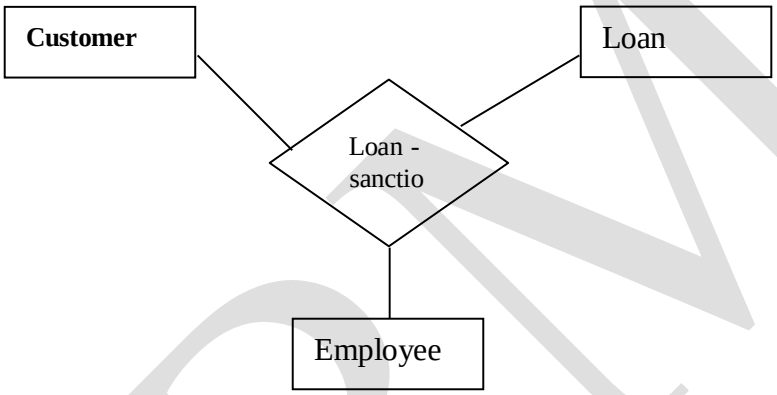
  FACULTY(CONTAINSWORKS_INRELATIOSHIP)
  (ID, NAME, ADDRESS, BASIC_SAL, DNO)

- **Many-to-ManyRelationship:**
  For each M:N relationship type R, we create a new table (say S) to represent R, we also include the primary key attributes of both the participating entity types as a foreign key attribute in S.Any simple attributes ofthe M:N relationship type (or simple components as a composite attribute) is also included as attributes of S.

Forexample:
The M:N relationship taught-by between entities COURSE and FACULTY should be represented as a new table. The structure of the table will include primary key ofCOURSE and primary key of FACULTY entities.

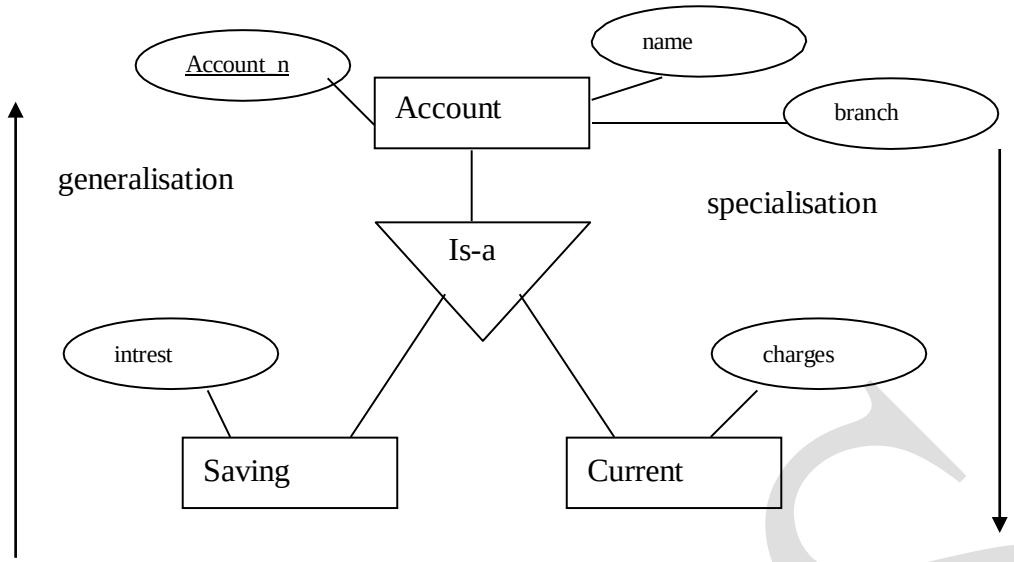
TAUGHT-BY (ID (primary key of FACULTY table), course-id (primary key of COURSE table)

- **N-aryRelationship:**
  For each N-ary relationship type R where n>2, we create a new table S to represent R, We include as foreign key attributes in S the primary keys of the relations that represent the participatingentitytypes.Wealso includeanysimpleattributesoftheN-aryrelationshiptype (or simple components of complete attribute) as attributes of S. The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types.



LOAN-SANCTION(cusomer-id, loanno,empno,sancdate,loan_amount)

- **Multi-ValuedAttributes:**
  For each multivalued attribute 'A', we create a new relation R that includes an attribute corresponding to plus the primary key attributes k of the relation that represents the entity type or relationship that has as an attribute. The primary key of R is then combination of A and k.
  Forexample, ifaSTUDENTentityhasrollno, nameandphone numberwherephone number is a multivalued attribute then we will create table PHONE (rollno, phoneno) where primary key is the combination. In the STUDENT table we need not have phone number, instead if can be simply (rollno, name) only.
  PHONE(rollno,phoneno)



- **ConvertingGeneralisation/SpecificationHierarchytoTables:**
  A simpleruleforconversionmay betodecomposeall thespecializedentitiesintotablein case theyare disjoint, for example, for the figure we can create the three tables as:
  Account(account_no,name,branch,balance)
  Saving_Account (account-no, intrest)
  Current_Account (account-no, charges)

LECTURE-9:RecordBasedLogicalModel

HierarchicalModel:

- Ahierarchicaldatabaseconsistsofacollectionof*records*whichareconnectedtoone another through *links.*
- Arecord isa collectionoffields,eachofwhichcontainsonlyone datavalue.
- Alinkisanassociationbetweenpreciselytwo records.
- The hierarchical model differs from the network model in that the records are organized as collections of trees rather than as arbitrary graphs.

Tree-StructureDiagrams:

- Theschemafor ahierarchicaldatabaseconsistsof
  - *boxes,*whichcorrespondtorecord types
  - *lines,*whichcorrespondto links
- Recordtypesareorganizedintheformofa*rootedtree.*
  - Nocyclesintheunderlyinggraph.
  - Relationshipsformedinthegraphmustbesuchthatonly one-to-manyorone-to-onerelationshipsexistbetweenaparentanda child.

Databaseschemaisrepresented asacollectionoftree-structurediagrams.

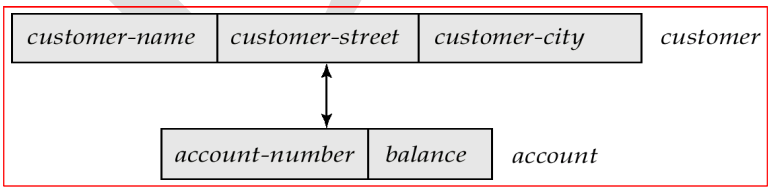
- *single*instanceofadatabasetree
- The rootofthistree is a dummynode
- The childrenofthatnodeareactualinstances oftheappropriaterecordtype

WhentransformingE-Rdiagramsto correspondingtree-structurediagrams, we must ensurethat the resulting diagrams are in the form of rooted trees.

SingleRelationships:

- ExampleofE-Rdiagramwithtwoentitysets,*customer*and*account,*relatedthrougha binary, one-to-many relationship *depositor.*
- Correspondingtree-structurediagramhas
  - therecordtype*customer*withthreefields: *customer-name,customer-street,*and *customer-city.*
  - therecordtype*account* withtwofields:*account-number*and*balance*
  - thelink*depositor,*withanarrowpointingto*customer*

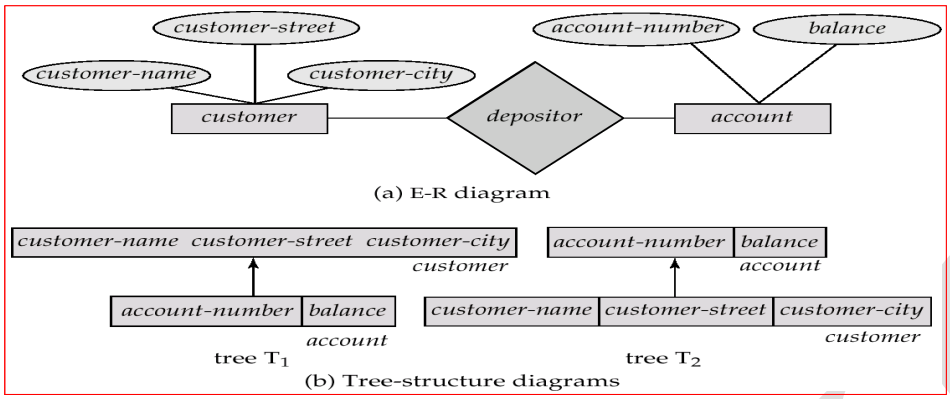
- Iftherelationship*depositor*isonetoone,thenthelink*depositor*hastwoarrows.



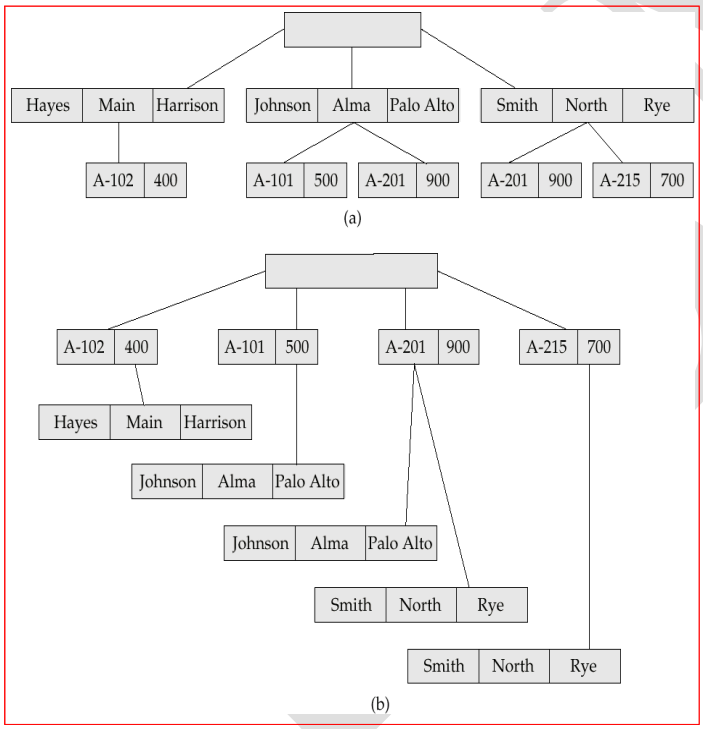
- Onlyone-to-manyandone-to-onerelationshipscanbedirectlyrepresentedinthehierarchical mode.

TransformingMany-To-Many Relationships:

- Must consider thetype ofqueriesexpected and the degreeto whichthe database schema fits the given E-R diagram.
- Inallversionsofthistransformation,theunderlyingdatabasetree(ortrees)willhave replicated records.



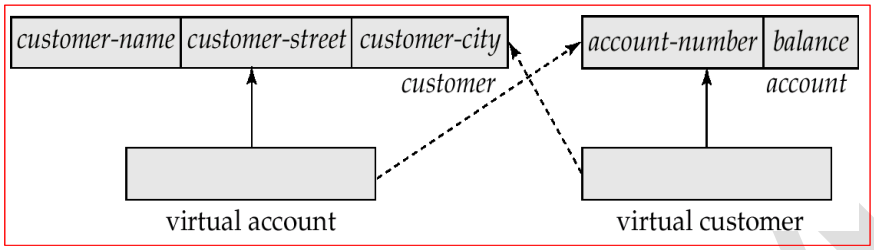
- Createtwotree-structurediagrams,*T*1,withtheroot*customer*,and*T*2,withtheroot *account.*
- In*T*1,create*depositor*,amany-to-onelink from*account*to*customer.*
- In*T*2,create*account-customer*,amany-to-one link from*customer*to*account.*



VirtualRecords:

- Formany-to-manyrelationships,recordreplicationisnecessarytopreservethetree- structure organization of the database.
- Datainconsistencymayresultwhenupdating takes place
- Wasteofspaceisunavoidable

- *Virtualrecord*—containsnodatavalue,onlyalogicalpointertoaparticularphysical record.
- Whenarecordistobereplicatedinseveral databasetrees,asinglecopy of thatrecordis kept in one of the trees and all other records are replaced with a virtual record.
- Let*R*bearecordtypethatisreplicatedin*T*1,*T*2,...,*Tn*.Createanewvirtualrecordtype *virtual-R*andreplace*R*ineachofthe*n*–1treeswitha recordoftype*virtual-R.*
- Eliminate data replication in the following diagram ;create *virtual-customer* and*virtual-account.*
- Replace*account*with*virtual-account*inthefirsttree,andreplace*customer*with*virtual- customer* in the second tree.
- Addadashedlinefrom*virtual-customer*to*customer*,andfrom*virtual-account*to*account,* tospecifythe associationbetweena virtualrecordanditscorrespondingphysicalrecord.



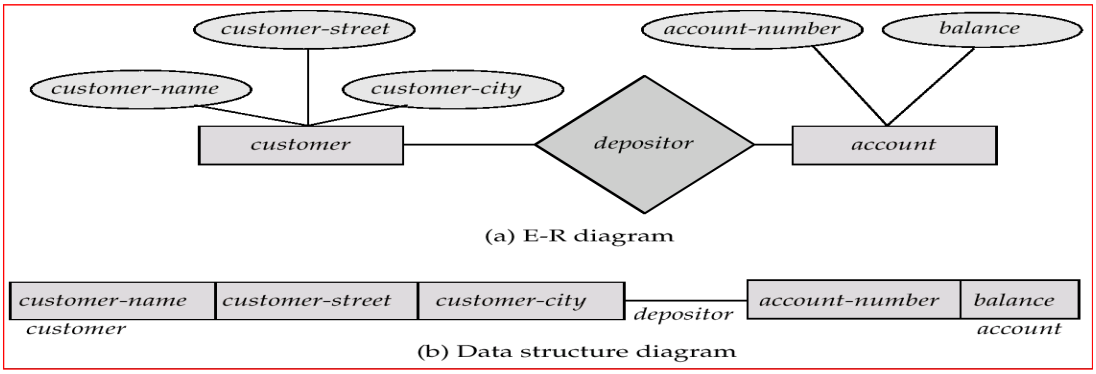
NetworkModel:

- Data arerepresentedbycollectionsof*records.*
  - similartoanentityinthe E-Rmodel
  - Recordsandtheirfieldsarerepresentedas*record type*
- type *customer*=record
  *customer-name:*string;
  *customer-street:*string;
  *customer-city:*string;
  type *account*= recordtype
  *account-number:*integer;
  *balance:*integer;
- end
  end
- Relationshipsamong dataarerepresented by*links*
  - similartoarestricted(binary)formofanE-Rrelationship
  - restrictions on links depend on whether the relationship is many-to-many, many-to-one, or one-to-one.

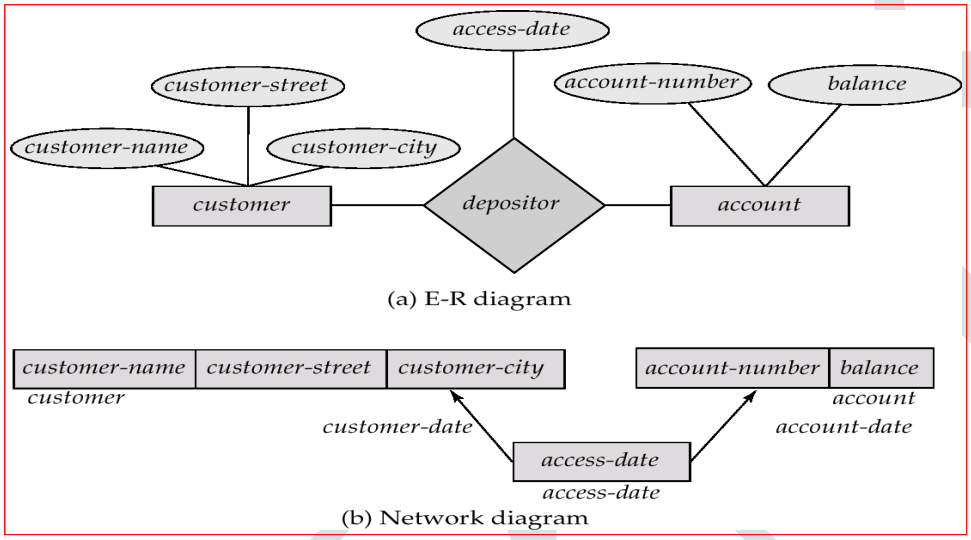
Data-StructureDiagrams:

- Schemarepresentingthe designofa networkdatabase.
- Adata-structurediagramconsistsoftwobasic components:
  - Boxes,whichcorrespondto recordtypes.
  - Lines,whichcorrespondtolinks.
- Specifiesthe overalllogicalstructureofthe database.

ForeveryE-Rdiagram,thereisacorresponding data-structurediagram.

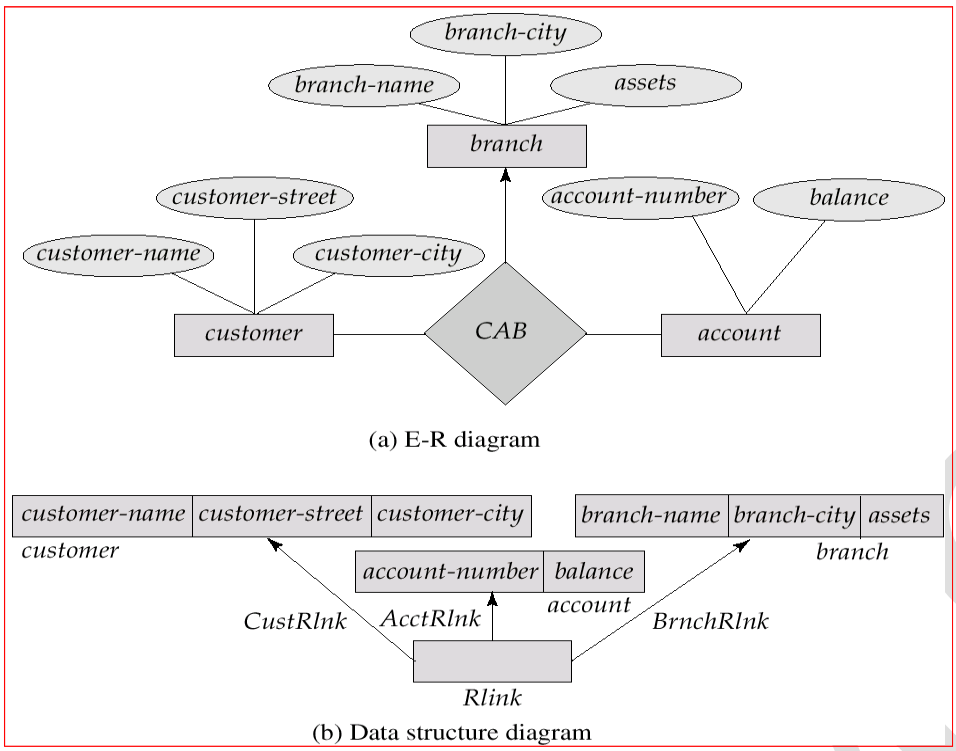


Since a link cannot contain any data value, represent an E-R relationship with attributes with a new record type and links**.**



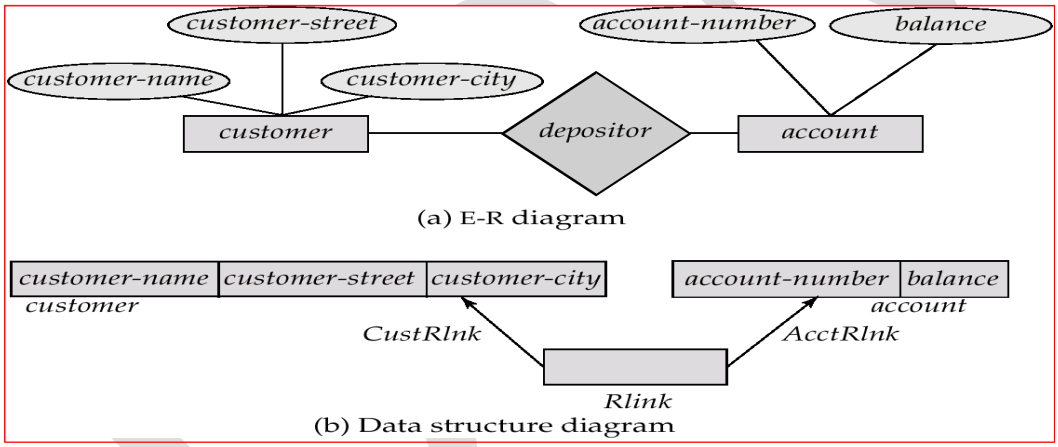
Torepresentan E-R relationship of degree 3 or higher,connect the participating record typesthrough a new record type that is linked directlyto each of the original record types.

1. Replace entitysets*account*,*customer*,and*branch*withrecordtypes*account*,*customer*,and *branch*, respectively.
2. Createanewrecordtype*Rlink*(referred toasa*dummy*recordtype).
3. Createthefollowingmany-to-one links:
   - *CustRlink* from*Rlink* recordtype to*customer*record type
   - *AcctRlnk*from*Rlink*recordtype to*account*record type
   - *BrncRlnk*from*Rlink*recordtype to*branch*recordtype



TheDBTGCODASYLModel:

- Alllinksare treatedasmany-to-one relationships.
- To model many-to-many relationships, a record type is defined to represent the relationship and two links are used.



DBTGSets:

- Thestructureconsistingoftworecordtypesthatarelinkedtogetherisreferredtointhe DBTG model as a *DBTG set*
- In eachDBTG set, one recordtype isdesignated asthe *owner*,and the other isdesignated as the *member*, of the set.
- EachDBTGsetcanhaveanynumber of*setoccurrences*(actualinstancesoflinked records).
- Since many-to-many links are disallowed, each set occurrence has precisely one owner, and has zero or more member records.
- Nomemberrecordofasetcanparticipateinmorethanoneoccurrenceofthesetatany point.
- Amemberrecordcanparticipatesimultaneouslyinseveralsetoccurrencesof*different* DBTGsets.

LECTURE-10:RELATIONALMODEL

RELATIONALMODEL

Relationalmodelissimplemodelinwhichdatabaseisrepresentedasacollectionof"relations" whereeachrelationis representedbytwo-dimensionaltable.

account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

The relational model was founded by E. F. Coddof the IBM in 1972. The basic concept in the relational model is that of a relation.

Properties:

- It is column homogeneous. In other words, in any given column of a table, all items are of the same kind.
- Eachitem isa simplenumberora characterstring. Thatis a tablemustbeinfirstnormal form.
- Allrowsofatablearedistinct.
- Theordering ofrowswith inatableisimmaterial.
- Thecolumnofatableareassigneddistinctnamesandtheorderingofthesecolumnsis immaterial.

Domain, attributestuplesandrelational:

Tuple:

Eachrowinatablerepresentsarecordandiscalledatuple.A tablecontaining'n'attributesina recordiscalledn-tuple.

Attributes:

The name of each column in a table is used to interpret its meaning and is called an attribute.Each table is called a relation. In the above table, account_number, branch name, balance are the attributes.

Domain:

A domainisasetofvaluesthatcanbegiventoanattributes.Soeveryattributeinatablehasa specific domain. Values to these attributes can not be assigned outside their domains.

Relation:

Arelationconsist of

- **Relationalschema**
- **Relationinstance**

RelationalSchema:

Arelationalschemaspecifiestherelation'sname,itsattributesandthedomainof eachattribute.If

R is the name of a relation and A1, A2,…An is a list of attributes representing R then R(A1,A2,…,An) is called a Relational Schema. Each attribute in this relational schema takes avalue from some specific domain called domain(Ai).

Example:

PERSON(PERSON_ID:INTEGER,NAME:STRING, AGE:INTEGER, ADDRESS:STRING)

Total number of attributes in a relation denotes the degree of a relation since the PERSON relation scheme contains four attributes, so this relation is of degree 4.

RelationInstance:

Arelational instance denoted as r is a collection of tuples for a given relational schema at a specific point of time.

Arelationstaterto therelationsschemaR(A1,A2…,An) also denotedbyr(R) isaset ofn-tuples

R{t1,t2,…tm}

Whereeach n-tupleisanordered listofn values

T=<v1,v2,….vn>

Whereeachvibelongstodomain(Ai) or containsnullvalues.

Therelationschema isalso called‘intension’andtherelationstateisalsocalled‘extension’.

Eg:Relationschemafor Student

STUDENT(rollno:string,name:string,city:string,age:integer)

Relationinstance:

Student:

Rollno	Name	City	Age
101	Sujit	Bam	23
102	kunal	bbsr	22

Keys:

Superkey:

Asuperkeyisanattributeorasetofattributesusedto identifytherecordsuniquelyinarelation. For example, customer-id, (cname, customer-id), (cname,telno)

Candidatekey:

Super keys of a relation can contain extra attributes. Candidate keys are minimal super keys. i.e, such a keycontains no extraneous attribute.An attribute is called extraneous ifeven after removing it from the key, makes the remaining attributes still has the properties of a key(atribute represents entire table).

InarelationR,acandidatekeyforRisasubsetofthesetofattributesofR,whichhavethe following properties:

- *Uniqueness:* notwodistincttuplesinR havethesamevalues for thecandidatekey
- *Irreducible:* Nopropersubset ofthecandidatekeyhasthe *uniquenesspropertythatisthecandidatekey.*
- *Acandidatekey'svaluesmustexist.Itcan'tbenull.*
- *Thevaluesofacandidatekeymustbestable.Itsvaluecannotchangeoutsidethe*

controlofthesystem.

Eg:(cname,telno)

Primarykey:

The primary key is the candidate key that is chosen by the database designer as the principal means of identifying entities with in an entity set. The remaining candidate keys if any arecalled ***alternate key*.**

LECTURE-11:CONSTRAINTS

RELATIONALCONSTRAINTS:

Therearethreetypesofconstraintsonrelationaldatabase thatinclude

- DOMAINCONSTRAINTS
- KEYCONSTRAINTS
- INTEGRITYCONSTRAINTS

DOMAINCONSTRAINTS:

Itspecifies thateach attribute in a relation an atomic value from the correspondingdomains. Thedata types associated with commercial RDBMS domains include:

- Standardnumericdatatypes for integer
- Realnumbers
- Characters
- Fixedlengthstringsandvariablelengthstrings

Thus, domain constraints specifies the condition that we to put on each instance of the relation. So the values that appear in each column must be drawn from the domain associated with that column.

Rollno	Name	City	Age
101	Sujit	Bam	23
102	kunal	bbsr	22

KeyConstraints:

This constraints states that the key attribute valuein each tuplemsutbe unique .i.e, no two tuples contain the same value for the key attribute.(null values can allowed)
Emp(empcode,name,address).hereempcodecanbeunique

IntegrityCONSTRAINTS:

Therearetwotypes ofintegrityconstraints:

- EntityIntegrityConstraints
- ReferentialIntegrityconstraints

EntityIntegrity Constraints:

Itstatesthatno primarykeyvaluecanbe nulland unique. This isbecausetheprimarykeyisusedto identify individual tuple in the relation. So we will not be able to identify the records uniquely containing null values for the primary key attributes. This constraint is specified on one individual relation.

ReferentialIntegrityConstraints:

It statesthat thetuple inone relationthat refers toanother relation must refer to anexisting tuple in that relation. This constraints is specified on two relations. If a column is declared as foreign key that must be primary key of another table.

Department(deptcode, dname)
Herethedeptcodeistheprimarykey.

Emp(*empcode*,name,city,*deptcode*).
Herethedeptcodeis foreignkey.

CODD'SRULES

Rule1:Theinformation Rule.

"All information in a relationaldata base is represented explicitlyat the logical leveland in exactly one way - by values in tables."

Everythingwithinthedatabaseexists intablesandisaccessedviatableaccessroutines.

Rule2: GuaranteedaccessRule.

"Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primarykey value and column name."

To access any data-item you specify which column within which table it exists, there is no reading of characters 10 to 20 of a 255 byte string.

Rule3:Systematictreatmentofnullvalues.

"Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of datatype."

If data does not exist or does not apply then a value of NULL is applied, this is understood by the RDBMS as meaning non-applicable data.

Rule4:Dynamicon-linecatalogbased ontherelationalmodel.

"The data base description is represented at the logical level in the same way as-ordinary data, so that authorized users canapplythe same relational language to its interrogationas theyapply to the regular data."

The Data Dictionary is held within the RDBMS, thus there is no-need for off-line volumes to tell you the structure of the database.

Rule5:Comprehensivedatasub-languageRule.

"A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and that is comprehensive in supporting all the following items

- DataDefinition
- ViewDefinition
- DataManipulation(Interactiveandbyprogram).
- IntegrityConstraints
- Authorization.

Every RDBMS should provide alanguage toallow the user toquery the contents of the RDBMSand also manipulate the contents of the RDBMS.

Rule6: .ViewupdatingRule

"Allviewsthataretheoreticallyupdateablearealso updateablebythesystem."

NotonlycantheusermodifyData,butsocantheRDBMS whentheuserisnotlogged-in.

Rule7 :High-levelinsert,updateanddelete.

"The capabilityofhandling a base relationora derived relationas a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data."

The user shouldbe able tomodify several tables by modifying the view towhich they actas base tables.

Rule8 :Physicaldataindependence.

"Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods."

Theuser shouldnot beawareofwhereoruponwhichmediadata-filesarestored

Rule 9 :Logicaldataindependence.

"Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit un-impairment are made to the basetables."

User programs and the user should not be aware of any changes to the structure of the tables (such as the addition of extra columns).

Rule10: Integrityindependence.

"Integrity constraints specific to a particular relational data base must be definable in the relational data sub-language and storable in the catalog, not in the application programs."

Ifa column only accepts certain values, then it is the RDBMS which enforces these constraints and not the user program, this means that an invalid value can never be entered into this column, whilst if the constraints were enforced via programs there is always a chance that a buggyprogram might allow incorrect values into the system.

Rule11:Distributionindependence.

"ArelationalDBMShasdistributionindependence."

The RDBMS mayspread across more thanone systemand across several networks, however to the end-user the tables should appear no different to those that are local.

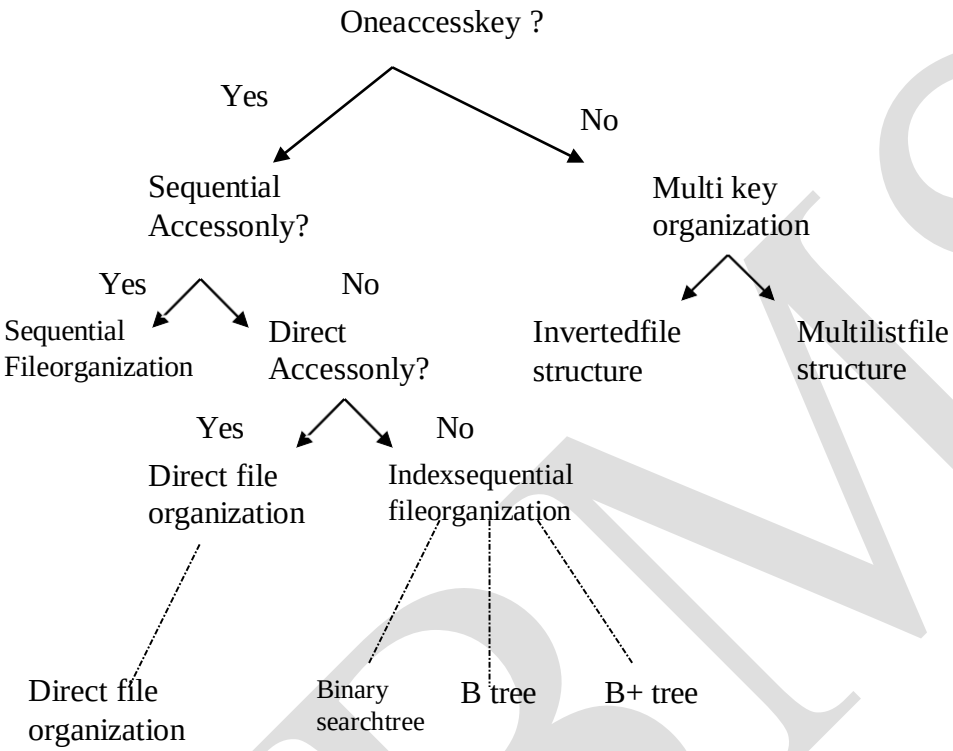
Rule12:Non-subversion Rule.

"If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrityRules and constraints expressed in the higher levelrelational language (multiple-records-at-a-time)."

LECTURE-12:FILEORGANISATION

FILEORGANISATIONANDITSTYPES:

Afile organization is a technique to organize data in the secondary memory. File organization is a wayofarranging the records in a file when the file is stored onthe disk. Data files are organized so as to facilitate access to records and to ensure their efficient storage.ADBMS supports several file organization techniques.



(Fileorganizationtechniques)

Heapfiles(unorderedfile)

Basicallythese files are unordered file. It is the simplest and most basic type. These files consist of randomlyordered records.The records willhave no particular order.The operations we canperform onthe records are insert , retrieve and delete. The features of the heap file organization are:

- Newrecordscanbeinsertedinanyemptyspace thatcanaccommodatethem.
- Whenold records aredeleted, theoccupied space becomes emptyand available for anynew insertion.
- Ifupdatedrecordsgrow,they mayneedtoberelocated toanewemptyspace.Thisneedsto keep a listofemptyspace.

Advantageofheapfiles:

1. Thisisasimplefileorganizationmethod
2. Insertionissomehow efficient
3. Goodforbulk-lading datainto atable.
4. Bestiffilescansarecommonorinsertionsarefrequent

Disadvantagesofheapfiles:

1. Retrievalrequiresalinearsearchandis inefficient

2. Deletioncanresultinunusedspace/needforreorganization

Sequentialfileorganization:

The most basic way to organize the collection of records in a file is to use sequential organization. Records ofthe file are stored in sequence bythe primarykey field values/ Theyare accessible only intheorder storedi.e, intheprimarykeyorder.Thiskind isoffileorganizationworkswellfor tasks which need to access nearly every record in a file. Eg. Payroll..

In a sequentially organized file records are written consecutively when the file is created and must be accessed consecutively when the file later used for input.

Asequential file maintains the records in the logicalsequence of its primarykey values. Sequential file are inefficient for random access. And files can be stored on devices like magnetic tape that allow sequential access. As records are in sorted order it will use binary search technique to search for a record.

Advantagesofsequentialfileorganization:

- Itisfastandefficientwhendealingwithlargevolumesof datathatneedtobeprocessed periodically(batch system)

Disadvantagesofsequentialfileorganization:

- Requires that all new transactions be sorted into the proper sequence for sequential accessprocessing
- Locating,storing,modifying,deletingoraddingrecordsinthefilerequirerearrangingthe file/
- Thismethodistooslowtohandleapplicationrequiringimmediateupdatingorresponses.

Indexedsequentialfileorganization:

It organized the file like a large dictionary, i.e, records are stored in order of the key, but an index is kept which also permits a type of direct access. The records are stored sequentially by primary key values and there is an index built over the primarykey field.

An index is a set of index value, address pairs. Indexing associates a set of objects to a set pf orderable quantities, that are usually smaller in number or their properties. Thus an index is a mechanism for faster search.Although the indices and the data blocks are kept physically, they are logically distinct.

Asequential file that is indexed on its primary key is called an index sequential file. The index allows for random access to records, while the sequential storage of the records of the file provides easy access to the sequential records. An addition feature of this file system is the overflow area.The overflow area providesadditionalspace for recordadditionwithout the need to create.

Advantage ofISAM indexes:

1. Because the whole structure is ordered to a large extent, partial (like ty%) andrange(between 12 and 29) based retrievals can often benefit from the use of this type of index.
2. ISAM isgoodforstatictablesbecausethereareusuallyfewerindexlevelsthanB-tree
3. Ingeneraltherearefewer diskI/Osrequiredtoaccessdata,providedthereisnooverflow.

DisadvantageofISAMindexes:

1. ISAMisstillnotasquickassomeinhashfile organization
2. Overflowcanbearealprobleminhighlyvolatiletable.

Hashedfileorganization:

Hashing is the most common form of purely random access to a file or database. It is also used to access columns that do not have a index as an optimization technique. Hash functions calculate the address of the page in which the record is tobe stored based on one ormore fields in the record. The records in a hash file appear randomly distributed across the available space. It requires some hashing algorithmand the technique. Hashing algorithmconverts a primarykey value into a record address.

Advantageofhashedfileorganization:

1. Insertionorsearchonhashkeyisfast.
2. Bestifequalitysearch isneeded onhash key

Disadvantageofhashedfile organization:

1. Itisacomplexfileorganization method
2. searchisslow
3. Itsuffers fromdiskspaceoverhead
4. Unbalanced bucketsdegrade performance
5. Rangesearchisslow

LECTURE-13:INDEX

TypesofIndexes:

- Indexingmechanismsusedtospeedupaccesstodesired data.
  - E.g.,authorcatalogin library
- **SearchKey**-attributetoset ofattributesused tolookup recordsinafile.
- An**indexfile**consists ofrecords(called**indexentries**)oftheform
- Indexfilesaretypicallymuchsmaller thantheoriginalfile
- Twobasickindsofindices:
  - **Orderedindices:**searchkeysarestoredinsortedorder
  - **Hash indices:**search keys are distributed uniformlyacross "buckets" using a "hash function".

OrderedIndices

Indexingtechniquesevaluatedonbasisof:

1. Inan**orderedindex,**indexentriesarestoredsortedonthesearchkeyvalue. E.g., author catalog in library.
2. **Primaryindex:** inasequentiallyorderedfile, the indexwhosesearchkeyspecifiesthe sequential order of the file.
   a. Alsocalled**clusteringindex**
   b. Thesearchkeyofaprimary index isusually butnot necessarilytheprimarykey.
3. **Secondaryindex**:anindexwhosesearchkeyspecifiesanorderdifferent fromthe sequential order of the file.Also called non-clusteringindex**.**
4. Index-sequentialfile**:** orderedsequentialfilewithaprimaryindex.

There aretwotypes oforderedindicesthatwecanuse:

Denseindex:

An index record appears for every search-key value in the file. In a dense primary index, the index record contains the search-key value and a pointer to the firstdata record with thatsearch-key value. The rest ofthe records with the same search key-value would be stored sequentially after the first record, since, because the index is a primary one, records are sorted on the same search key. Dense index implementations may store a list of pointers to all records with the same search-key value; doing so is not essential for primary indices.

Sparseindex:

An index record appears for only some of the search-key values. As is true in dense indices, each index record contains a search-key value and a pointer to the first data record with that search-key value. To locate arecord, we find the indexentrywiththe largest search-key value that isless than or equal to the search-key value for which we are looking. We start at the record pointed to by that index entry, and follow the pointers in the file until we find the desired record.

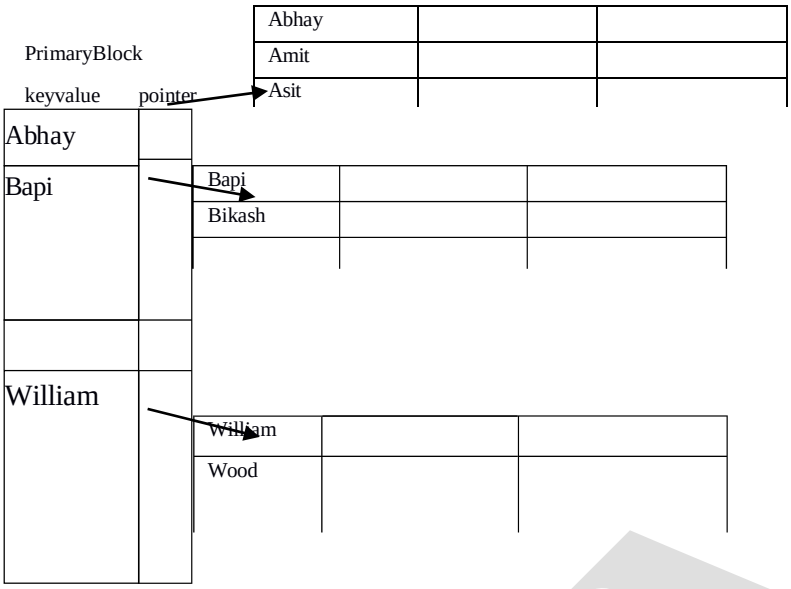
PrimaryIndexes:

Aprimary index is an ordered file whose records are of fixed length with two fields. The first field is ofthe same datatypes asthe ordering keyfiled ofthe data file and the second field is a pointerto adisk block- ablockaddress.Theorderingkeyfield iscalledtheprimarykeyofthedatafile.There

is one index entry in the index file for each block in the data file. Each index entryhas the value of the primary key field for the first record in a block and a pointer to other block as its two fields values.

The first recordofeachblockofthedatafile isknownasanchor recordoftheblock.

Primary index is an example of non dense index



(Primaryindexesontheorderingkeyfield)

A major problem with primary index as with any ordered file- is insertion and deletion of records. With a primary Index, the problem is compounded because if we attempt to insert a record in its correct position inthe data file, we not onlyhave to move recordsto make space forthe new record but also have to change some index entries because moving records willchange the anchor records of some blocks.

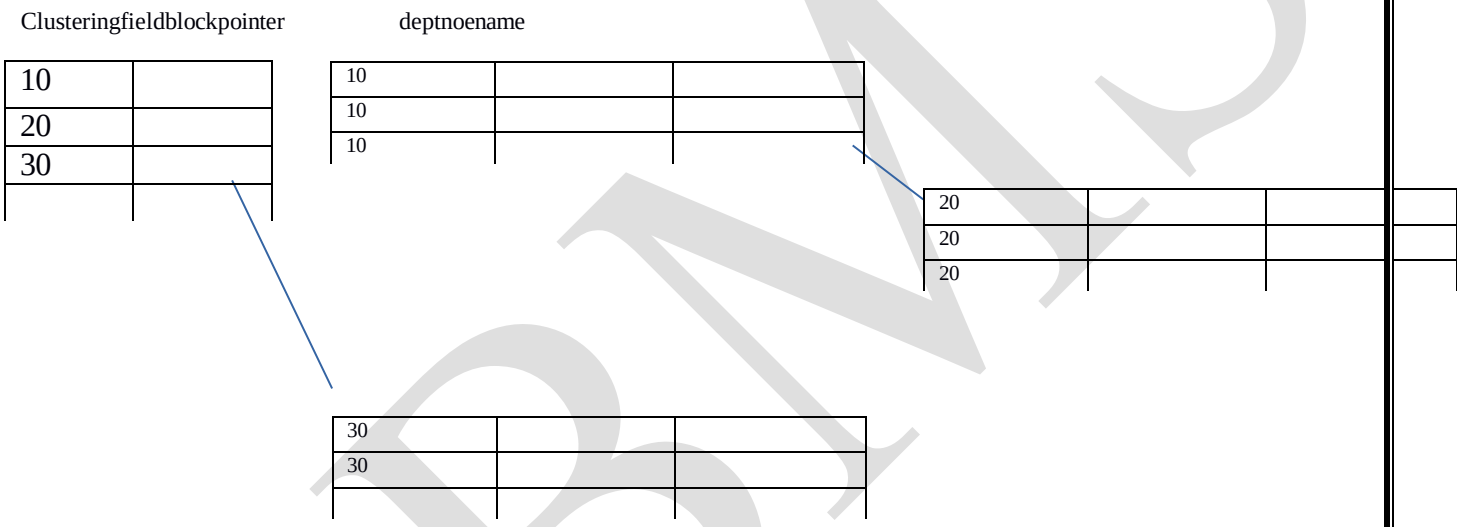
LECTURE-14:ClusteringIndex

ClusteringIndexes:

If the records of a file are physically ordered on a non key field that does not have a distinct value for each record, that filed is called the clustering filed of the file. We can create a different type of index called clustering index to speed up retrieval of records that have the same value for the clustering field. This differs froma primary index, whichrequires that the ordering field ofthe data file have a distinct value for each record.

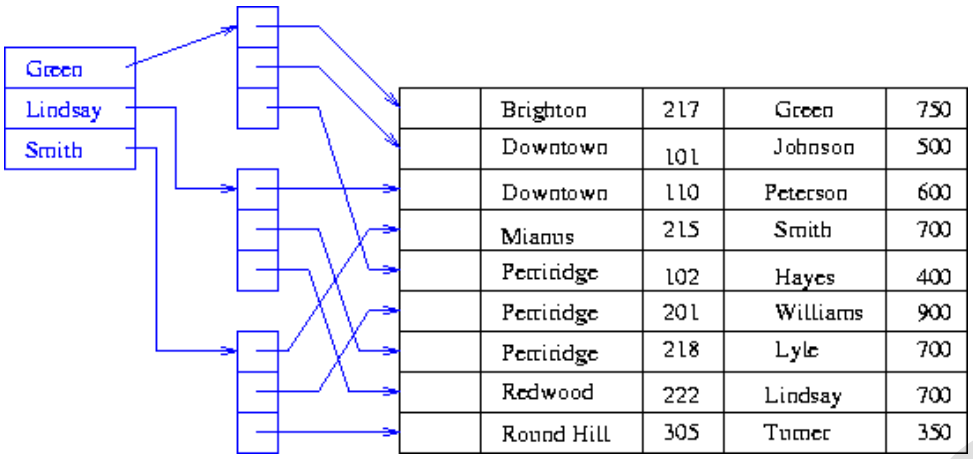
Aclustering index isalso anorderedfile withtwofields, the first field isofthe same type as the clustering field of the data file and the second field is block pointer. There is one entryin the clustering index for eachdistinct value ofthe clustering field, containing that value and a pointer to the first block in the data file that has a record with that value for its clustering field.

Aclusteringindexisanother exampleofnondenseindex.



SecondaryIndexes:

A secondary index alsois an ordered file with two fields and as in the other indexes, the second filed isa pointerto a disk block.The first field isofthe same type assome nonordering field ofthe data file. The field on whichthe secondary index is constructed is called an indexing field of the file. Whether its values are distinct for every record or not.



Therecanbemanysecondaryindexesandhenceindexingfields forthesamefile.

PrimaryandSecondaryIndices:

Secondaryindices havetobe dense.

1. Indicesoffersubstantialbenefitswhensearchingforrecords.
2. Whenafileismodified,everyindexonthefilemustbeupdated,Updatingindices imposes overhead on database modification.
3. Sequential scan using primary index is efficient, but a sequential scan using a secondary index is expensive
    a. eachrecordaccessmayfetchanewblockfromdisk

LECTURE-15:B$^+$TreeIndex

B+-TreeIndexFiles:

B+-treeindicesareanalternativetoindexed-sequentialfiles.

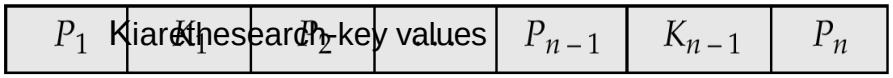
1. Disadvantage of indexed-sequential files: performance degrades as file grows, since many overflow blocks get created.Periodic reorganizationof entire file is required.
2. Advantage of B+-tree index files:automatically reorganizes itself with small, local, changes, inthe face ofinsertions and deletions.Reorganizationofentire file is not required to maintain performance.
3. DisadvantageofB+-trees: extrainsertionand deletionoverhead,spaceoverhead.
4. AdvantagesofB+-treesoutweighdisadvantages,andtheyareusedextensively.

AB+-tree is a rooted tree satisfying the following properties:

1. Allpathsfromroottoleafareofthesamelength
2. Eachnode thatisnotarootoraleafhasbetween[n/2]and$n$children.
3. Aleafnodehasbetween[(n–1)/2]and$n$–1values
4. Specialcases:
   a. Iftherootisnotaleaf,ithasatleast2children.
   b. Ifthe root is a leaf(thatis, there are no other nodes in the tree), itcan have between 0 and (n–1) values.

B$^+$-TreeNodeStructure

Typicalnode



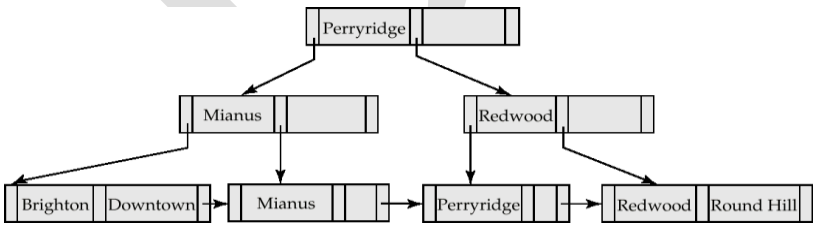
Kiarethesearch-key values

Piarepointerstochildren(fornon-leafnodes)orpointers to records or buckets of records (for leaf nodes).

Thesearch-keysinanodeareordered

$K1<K2<K3<. . . <Kn–1$

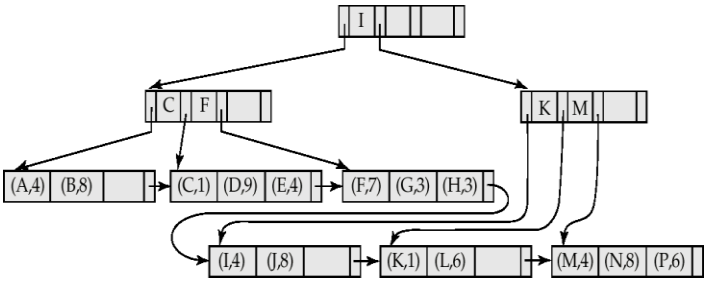
ExampleofaB+-tree



B+-TreeFileOrganization

1. Index file degradation problem is solvedby using B+-Treeindices.Data file degradationproblem is solved by using B+-Tree File Organization.
2. TheleafnodesinaB+-treefileorganizationstorerecords,instead ofpointers.
3. Since records are larger thanpointers, the maximumnumber ofrecords that canbe stored in a leaf node is less than the number of pointers in a non leaf node.

4. Leafnodesarestillrequired tobehalffull.
5. Insertion and deletion are handled in the same way as insertion and deletion of entries in a B+-tree index.



B-TreeIndexFiles

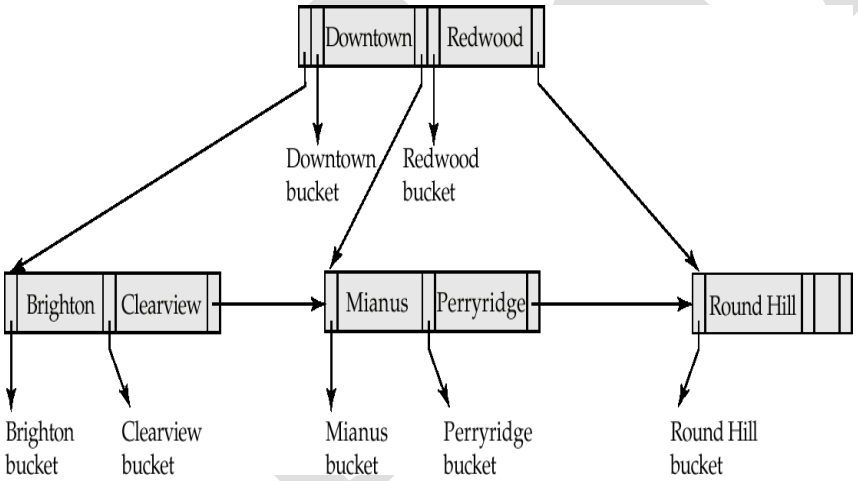
1. SimilartoB+-tree,butB-treeallowssearch-keyvaluestoappearonlyonce;eliminates redundant storage of search keys.
2. Search keys in non leaf nodes appear nowhere else in the B-tree; an additional pointer field for each search key in a non leaf node must be included.
3. GeneralizedB-treeleafnode

(a)

(b)

4. Nonleafnode– pointers*Bi*arethebucketorfilerecord pointers.

B-TreeIndexFileExample:



AdvantagesofB-Treeindices:

- Mayuselesstree nodesthana correspondingB+-Tree.
- Sometimespossibletofindsearch-keyvaluebeforereachingleafnode.

DisadvantagesofB-Treeindices:

- Onlysmallfractionofallsearch-keyvaluesare foundearly
- Non-leafnodesarelarger,sofan-outisreduced. Thus,B-Treestypicallyhave greater depth than corresponding B+-Tree

- Insertionanddeletionmorecomplicatedthanin B+-Trees
- ImplementationisharderthanB+-Trees.

Typically,advantagesofB-Treesdonotoutweighdisadvantages.

LECTURE-16:HashFileOrganization

HashFileOrganization

In a **hash file organization**, we obtain the address of the disk block containing a desired record directly by computing a function on the search-key value of the record.In our description of hashing, weshalluse the term**bucket** to denote aunit ofstoragethat canstoreone or more records. Abucket is typically a disk block, but could be chosen to be smaller or larger than a disk block.

Formally, let *K* denote the set of all search-key values, and let *B* denote the set of all bucket addresses.A**hash function** *h* is a function from *K* to *B*. Let *h* denote a hash function.

To insert a record with search key *Ki*, we compute *h*(*Ki*), which gives the address of the bucket for that record.Assume for now that there is space in the bucket to storethe record.Then, the record is stored in that bucket.

To perform a lookup on a search-key value *Ki*, we simply compute *h*(*Ki*), then search the bucket with that address. Suppose that two search keys, *K*5 and *K*7, have the same hash value; that is,*h*(*K*5) = *h*(*K*7). If we perform a lookup on *K*5, the bucket *h*(*K*5) contains records with search-key values *K*5 and records with search key values *K*7. Thus, we have to check the search-key value of every record in the bucket to verify that the record is one that we want. Deletion is equally straightforward. If the search-key value of the record to be deleted is *Ki*, we compute *h*(*Ki*), then search the corresponding bucket for that record, and delete the record from the bucket.

Hash Indices

Hashing can be used not only for file organization, but also for index-structure creation. A **hash index** organizes the search keys, with their associated pointers, into a hash file structure. We construct a hash index as follows. We apply a hash function on a search key to identify a bucket, and storethe keyand its associated pointers in the bucket (or in overflow buckets). Figureshows a secondary hash index on the *account* file, for the search key *account-number*. The hash function in the figure computes the sum of the digits of the account number modulo 7. The hashindex has sevenbuckets, eachofsize2 bucket sizes). Oneofthebuckets hasthreekeys mappedto it, so it has an overflow bucket. In this example, *account-number* is a primary key for *account*, so each search key has onlyone associated pointer. In general, multiple pointers can be associated with each key.

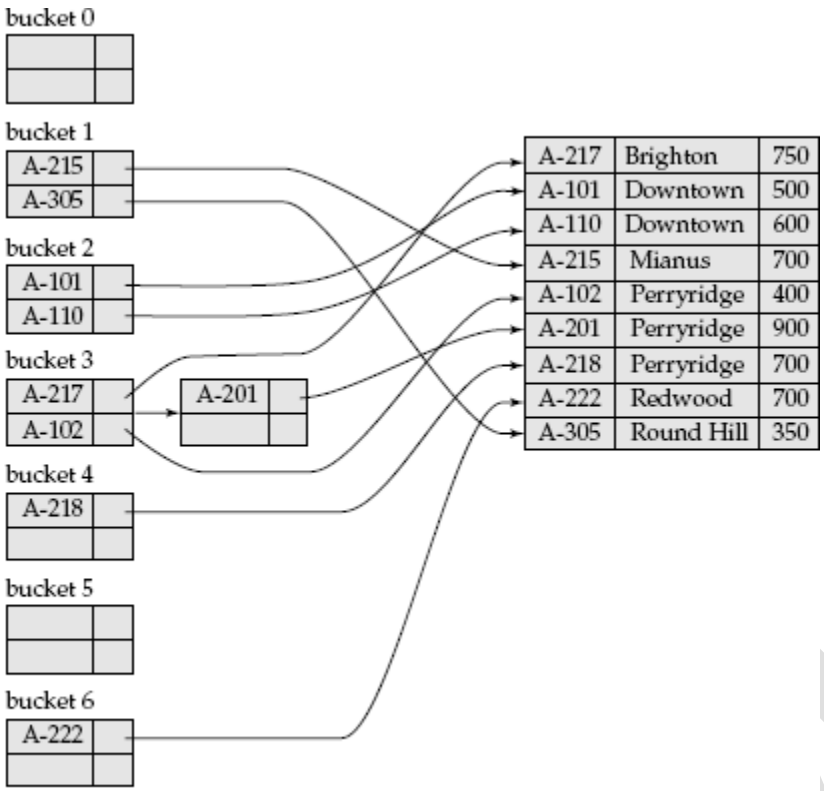


Figure 12.23 Hash index on search key *account-number* of *account* file.

We use the term*hash index* to denote hash file structures as wellas secondaryhash indices. Strictly speaking, hash indices are only secondary index structures. A hash index is never needed as a primaryindexstructure, since, ifa file itselfisorganized by hashing, there is no need for a separate hash index structure on it. However, since hash file organization provides the same direct access to records that indexing provides, we pretend that a file organized by hashing also has a primary hash index on it.

Advantages of hashing:

1. Exactkeymatchesareextremelyquick
2. Hashing is verygood long keys, or those with multiple columns, provided the complete key value is provided for the query
3. This organization usually allows for the allocation of disk space so a good deal of disk management is possible
4. Nodiskspaceisusedbythis indexingmethod

Disadvantagesof hashing:

1. It becomes difficult to predict overflow because the working of the hashing algorithm will not be visible to the DBA
2. Nosortingofdataoccurseither physicallyorlogicallysosequentialaccessispoor
3. Thisorganizationusuallytakesalotofdiskspacetoensurethatnooverflowoccurs.

LECTURE-17:QueryProcessing

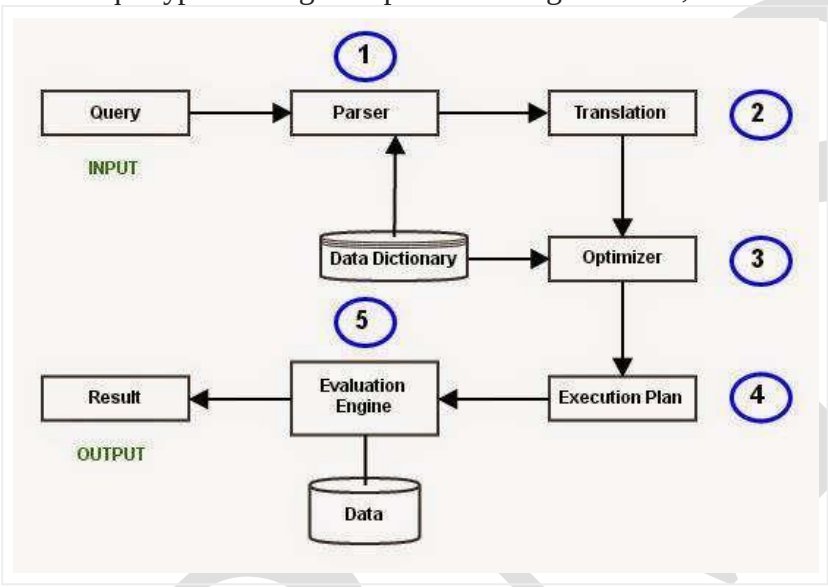
QueryProcessing

Query Processing would mean the entire process or activity which involves query translation into low level instructions, queryoptimization to save resources, cost estimation or evaluation ofquery, and extraction of data from the database.

Goal: To find an efficient QueryExecution Plan for a given SQLquery which would minimize the cost considerably, especially time.

Cost Factors: Disk accesses [which typically consumes time], read/write operations [whichtypically needs resources such as memory/RAM].

Themajorsteps involvedinqueryprocessingaredepictedinthefigurebelow;



Let usdiscussthewholeprocesswithanexample. Let usconsider thefollowingtworelationsasthe example tables for our discussion;

Employee(Eno, Ename, Phone)
Proj_Assigned(Eno,Proj_No,Role,DOP)

where,

Eno is Employee number,
EnameisEmployeename,
Proj_No isProjectNumber inwhichanemployeeisassigned, Role is
the role of an employee in a project,
DOPisdurationoftheprojectin months.

Withthisinformation,letuswriteaquery tofindthelistof allemployeeswhoareworkingina project which is more than 10 months old.

```
SELECTEname
FROMEmployee,Proj_Assigned
WHEREEmployee.Eno =Proj_Assigned.Eno ANDDOP>10;
```

Input:

A query writtenin SQLisgiven asinputto thequery processor.For our case,letusconsiderthe SQL query written above.

Step1:Parsing

Inthisstep,theparserofthequeryprocessormodulechecksthesyntaxofthequery,theuser's privilegestoexecutethequery,thetablenamesandattributenames,etc.Thecorrecttablenames,

attributenamesandtheprivilegeoftheuserscanbetakenfromthesystemcatalog(data dictionary).

Step2:Translation

If wehavewrittenavalidquery,thenitisconvertedfromhighlevellanguageSQLtolowlevel instruction in Relational Algebra.

Forexample,ourSQLquerycanbeconvertedinto aRelationalAlgebraequivalentas follows;

$$\pi_{Ename}(\sigma_{DOP>10 \wedge Employee.Eno=Proj\_Assigned.Eno}(EmployeeXProf\_Assigned))$$

Step3:Optimizer

Optimizer uses the statistical data stored as part of data dictionary. The statistical data are information about the size of the table, the length of records, the indexes created on the table, etc. Optimizer also checks for the conditions and conditional attributes which are parts of the query.

Step4:ExecutionPlan

A query can be expressed in many ways. The query processor module, at this stage, using the information collected in step 3 to find different relational algebra expressions that are equivalentand return the result of the one which we have written already.

Forourexample,thequerywritteninRelationalalgebracanalsobe writtenasthe one givenbelow;

$$\pi_{Ename}(Employee \bowtie_{Eno}(\sigma_{DOP>10}(Prof\_Assigned)))$$

So far, we have got two execution plans. Only condition is that both plans should give the same result.

Step5:Evaluation

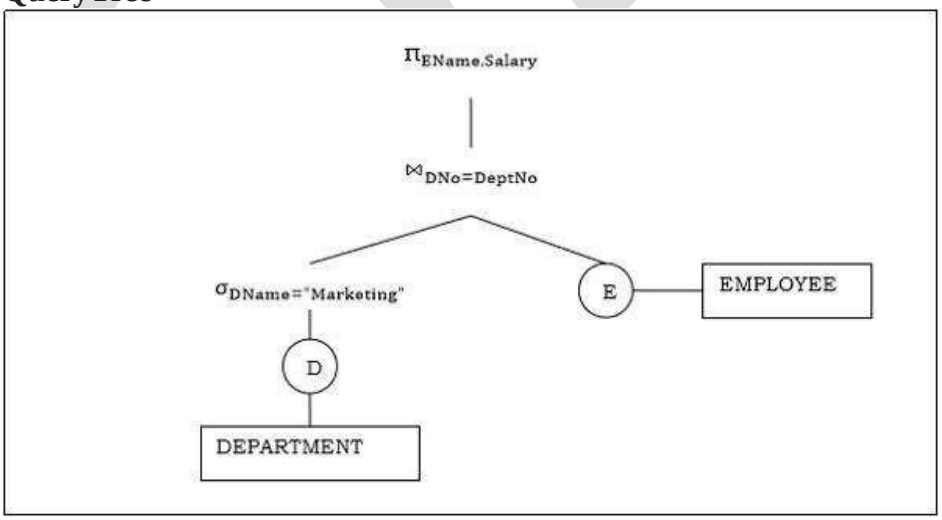
Though we got many execution plans constructed through statistical data, though they return same result (obvious), they differ in terms of Time consumption to execute the query, or the Space required executing the query. Hence, it is mandatory choose one plan which obviously consumes less cost.

At this stage, we choose one execution plan of the severalwe have developed. This Execution plan accesses data from the database to give the final result.

In our example, the second plan may be good. In the first plan, we join two relations (costly operation) thenapplythe condition(conditions are considered as filters) onthe joined relation. This consumes more time as well as space.

In the second plan, we filter one of the tables (Proj_Assigned) and the result is joined with the Employee table. This join may need to compare less number of records. Hence, the second plan is the best (with the information known, not always).

QueryTree



Usedinqueryrepresentationusedinparsing.

QueryOptimization:

A single query can be executed through different algorithms or re-written in different forms and structures. Hence, the questionofqueryoptimizationcomes intothe picture –Whichofthese forms or pathways is the most optimal? The queryoptimizer attemptsto determine the most efficient way to execute a given query by considering the possible query plans.

Therearebroadly twowaysa query canbeoptimized:

1. Analyze and transform equivalent relational expressions: Try to minimize the tuple and column counts of the intermediate and final query processes (discussed here).
2. Using different algorithms for each operation: These underlying algorithms determine how tuplesareaccessed fromthedatastructurestheyarestoredin, indexing, hashing, dataretrieval and hence influence the number of disk and block accesses (discussed in queryprocessing).

Analyzeandtransformequivalentrelationalexpressions

Here, we shall talk about generating minimal equivalent expressions. To analyze equivalent expression, listed are a set of equivalence rules. These generate equivalent expressions for a query written in relational algebra. To optimize a query, we must convert the query into its equivalent form as long as an equivalence rule is satisfied.

1. **Conjunctive selection operations can be written as a sequence of individual selections. This is called a sigma-cascade.**

   Explanation: Applyingconditionintersectionisexpensive.Instead,filterouttuples

   satisfying condition(inner selection) and then apply condition(outer selection) to the then resulting fewer tuples. This leaves us with less tuples to process the second time. Thiscan be extended for two or more intersecting selections. Since we are breaking a single condition into a series of selections or cascades, it is called a "cascade".

2. **Selectioniscommutative.**

   Explanation:condition is commutative in nature. This means, it does not matter whether we applyfirst orfirst. Inpractice, it is better and more optimalto applythat selection first which yields a fewer number of tuples. This saves time on our outer selection.

3. **All following projections can be omitted, only the first projection is required. This is called a pi-cascade.**

   Explanation: A cascade or a series of projections is meaningless. This is because in the end, we are only selecting those columns which are specified in the last, or the outermost projection. Hence, it is better to collapse all the projections into just one i.e. the outermost projection.

4. **SelectionsonCartesian Productscanbere-writtenasTheta Joins.**

   - **Equivalence1**

   Explanation: The cross product operation is known to be very expensive. This is because it matches each tuple of E1 (total m tuples) with each tuple of E2 (total ntuples). This yields m*n entries. If we apply a selection operation after that, we would have to scan through m*n entries to find the suitable tuples which satisfy the condition

   . Instead of doing all of this, it is more optimal to use the Theta Join, a join specifically designed to select only those entries in the cross product which satisfy the Theta condition, without evaluating the entire cross product first.

   - **Equivalence2**

   Explanation: Theta Join radically decreases the number of resulting tuples, so if we applyanintersectionofboththejoinconditionsi.e.andintothe Theta Joinitself,

wegetfewerscanstodo.Ontheotherhand,a conditionoutsideunnecessarily increases the tuples to scan.

5. **ThetaJoinsarecommutative.**

Explanation: Theta Joins are commutative, and the query processing time depends to some extent whichtable is used as the outerloop and whichone is used as the inner loopduring the join process (based on the indexing structures and blocks).

6. **Joinoperationsare associative.**

- **NaturalJoin**

Explanation: Joins are all commutative as wellas associative,so onemustjoin those two tables first which yield less number of entries, and then apply the other join.

- **Theta Join**

Explanation: ThetaJoinsareassociativeintheabovemanner,where involves attributes from only E2 and E3.

7. **Selectionoperationcanbe distributed.**

- **Equivalence1**

Explanation: Applying a selection after doing the Theta Join causes all the tuples returned by the Theta Join to be monitored after the join. If this selection contains attributes from only E1, it is better to apply this selection to E1 (hence resulting in a fewer number of tuples) and then join it with E2.

- **Equivalence2**

Explanation: Thiscanbeextendedtotwoselectionconditions,and,where

Theta1 contains the attributes of only E1 andcontains attributes of only E2. Hence, we can individually apply the selection criteria before joining, to drastically reduce the number of tuples joined.

8. **Projection distributesovertheTheta Join.**

- **Equivalence1**

Explanation: The idea discussed for selection can be used for projection as well. Here, if L1 is a projection that involves columns of only E1, and L2 another projection that involves the columns ofonlyE2, then it is betterto individuallyapplythe projections on both the tables before joining. This leaves us with a fewer number of columns on either side, hence contributing to an easier join.

- **Equivalence2**

Explanation: Here, when applying projections L1 and L2 on the join, where L1contains columns of only E1 and L2 contains columns of only E2, we can introduce another column E3 (which is common between both the tables). Then, we can apply projections L1 and L2 on E1 and E2 respectively, along with the added column L3. L3 enables us to do the join.

9. **UnionandIntersectionare commutative.**

Explanation: Unionandintersectionarebothdistributive;wecanencloseanytablesin parantheses according to requirement and ease of access.

10. **UnionandIntersectionare associative.**

Explanation: Unionandintersectionarebothdistributive;wecanencloseanytablesin parantheses according to requirement and ease of access.

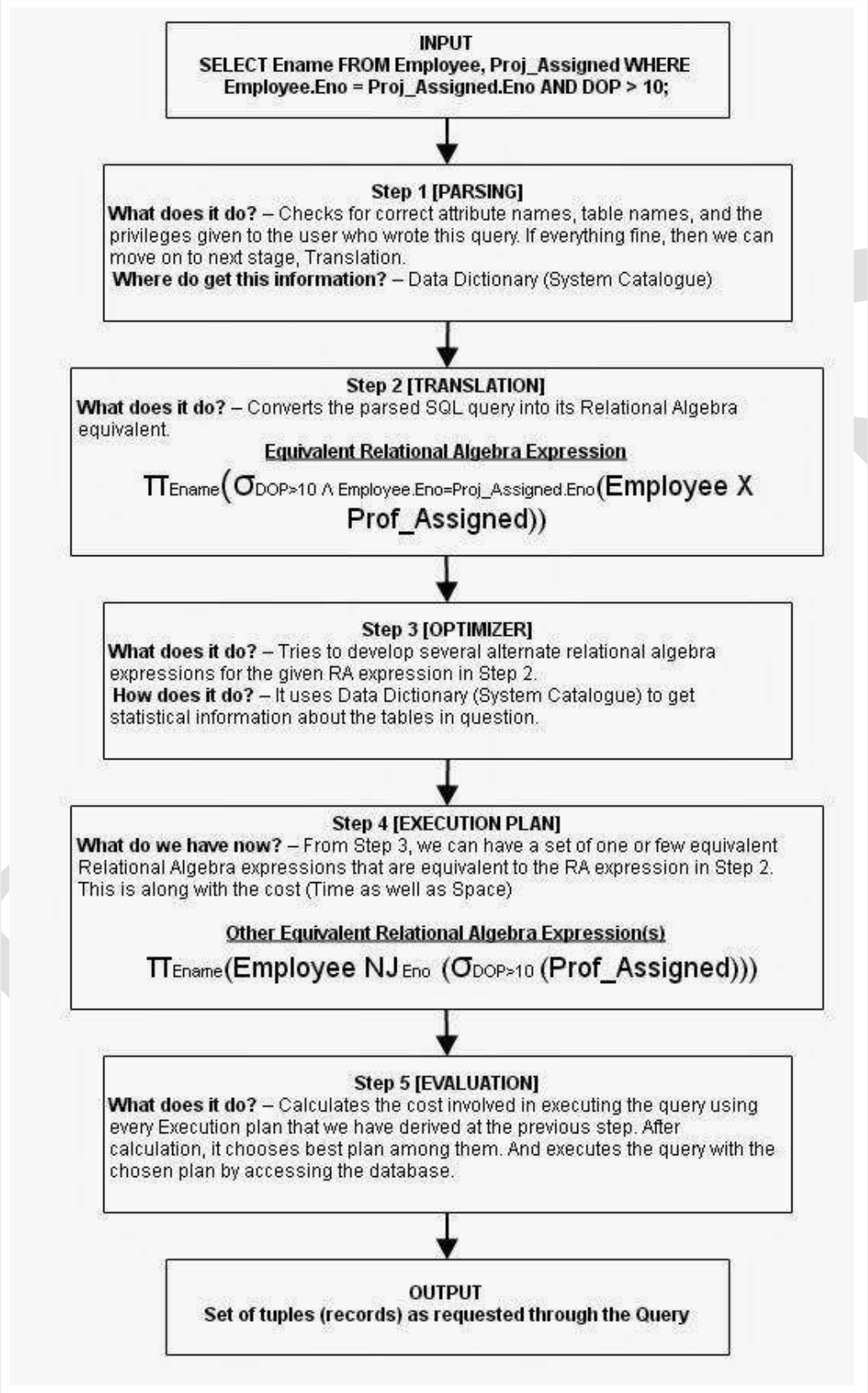
11. **Selectionoperationdistributesovertheunion,intersection,anddifferenceoperations.**

Explanation: Insetdifference,weknowthatonlythosetuplesareshownwhichbelongto

table E1 and do not belong to table E2. So, applying a selection condition on the entire set difference is equivalent to applying the selection condition on the individual tables and then applying set difference. This willreduce the number ofcomparisons inthe set difference step.

12. **Projectionoperation distributesovertheunionoperation.**

Explanation: Applying individual projections before computing the union of E1 and E2 is more optimal than the left expression, i.e. applying projection after the union step.



LECTURE-18:EvaluationofExpressions

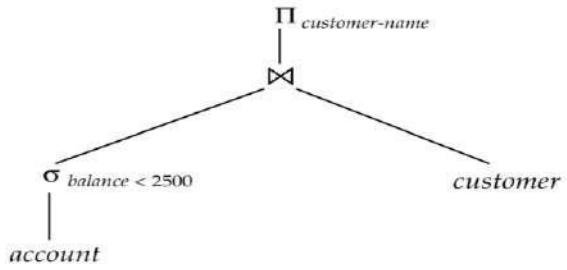
Evaluation of Expressions

- Sofar:wehaveseenalgorithmsforindividualoperations
- Alternativesforevaluatinganentireexpressiontree
  - **Materialization**:generate results ofan expression whose inputs are relations or are already computed, **materialize** (store) it on disk.Repeat.
  - **Pipelining**: passontuplestoparentoperationsevenasanoperationisbeingexecuted
- Westudyabovealternativesinmoredetail

Materialization

- **Materializedevaluation:**evaluate one operation ata time,startingatthelowest-level.Use intermediate results materialized into temporary relations to evaluate next-level operations.
- E.g.,infigurebelow,computeandstore

  then compute the store its join with *customer,* and finally compute the projections on *customer-name.*



- Materializedevaluationisalwaysapplicable
- Costofwriting resultstodisk and reading thembackcan bequitehigh
  - Ourcostformulasforoperationsignore costofwritingresultstodisk,so
    - Overallcost=Sumofcostsofindividualoperations+ costofwritingintermediateresultstodisk
- Doublebuffering:usetwooutputbuffersforeach operation,when oneisfull writeittodisk while the other is getting filled
  - Allowsoverlapofdiskwriteswithcomputationandreducesexecutiontime

Pipelining

- **Pipelined evaluation :**evaluate several operations simultaneously, passing the results of oneoperation on to the next.
  - E.g., in previous expression tree, don't store result of instead, pass tuples directly to the join..Similarly, don't store result of join, pass tuples directlyto projection.
- Muchcheaper thanmaterialization:noneedtostoreatemporaryrelationtodisk.
- Pipeliningmaynotalways bepossible–e.g.,sort,hash-join.
- Forpipeliningtobeeffective,useevaluationalgorithmsthatgenerateoutputtuplesevenas tuples are received for inputs to the operation.
- Pipelinescanbeexecutedintwoways:**demanddriven** and**producerdriven**
- In**demanddriven**or **lazy**evaluation
  - systemrepeatedlyrequestsnexttuplefromtopleveloperation
  - Each operation requestsnext tuple from children operations as required, in order to output its next tuple

- Inbetweencalls,operationhastomaintain"**state**"soitknowswhattoreturnnext
- Eachoperationisimplementedasan**iterator**implementingthefollowingoperations
  - open()
    - E.g. file scan: initialize file scan, store pointer to beginning of file as state
    - E.g.merge join: sort relationsand storepointerstobeginningofsorted relations as state
  - next()
    - E.g.forfilescan:Outputnexttuple,andadvanceandstorefilepointer
    - E.g.formerge join:continuewithmerge fromearlierstatetill next output tuple is found.Save pointers as iterator state.
  - close()

- Inproduce-drivenor **eager**pipelining
  - Operatorsproducetupleseagerlyandpassthemuptotheirparents
    - Buffermaintainedbetweenoperators,childputstuplesinbuffer,parent removes tuples from buffer
    - ifbuffer is full, child waitstillthere isspace in the buffer, and thengenerates more tuples
  - System schedules operations that have space in output buffer and can process more input tuples

EvaluationAlgorithmsforPipelining

- Somealgorithmsarenotabletooutputresultsevenastheygetinputtuples
  - E.g.mergejoin,orhashjoin
  - These resultinintermediate resultsbeingwrittentodiskandthenreadbackalways
- Algorithm variants are possible to generate (at least some) results on the fly, as input tuples are read in
  - E.g. hybrid hash join generates output tuples even as probe relation tuples in the in-memory partition (partition 0) are read in
  - **Pipelined join technique**: Hybrid hash join, modified to buffer partition 0 tuples of both relations in-memory, reading themas they become available, and output results of any matches between partition 0 tuples
    - When a new $r_0$tuple is found, match it with existing $s_0$tuples, output matches, and save it in r_0
    - Symmetricallyfors$_0$tuples

ComplexJoins

- Joininvolvingthreerelations:*loan depositor customer*
- **Strategy1.** Compute*depositor customer;*useresulttocompute*loan* (*depositor customer*)
- **Strategy2.**Computer *loan depositor*first, andthenjointheresultwith*customer.*
- **Strategy3.**Performthe pairofjoinsatonce.Buildandindexon*loan*for*loan-number,*andon *customer*for*customer-name.*
  - For each tuple *t* in *depositor,*look up the corresponding tuples in *customer* and thecorresponding tuples in *loan.*
  - Each tupleof*deposit*isexaminedexactlyonce.
- Strategy 3 combines two operations into one special-purpose operation that is more efficient than implementing two joins of two relations.

Module-2:LECTURE-19

RelationalAlgebra:

Basicoperations:

1. *Selection* (σ) Selectsasubset ofrowsfromrelation.
2. *Projection*(π)Selectsasubsetofcolumnsfromrelation.
3. *Cross-product*(×)Allowsusto combinetworelations.
4. *Set-difference*()Tuplesinrelation.1,butnotinrelationn.2.
5. *Union*(U)Tuples inreln.1andinreln. 2.
6. *Rename*(ρ)UsenewnamefortheTablesorfields.

Additionaloperations:

7. Intersection(∩),*Join*(⋈),Division(÷):Notessential,but(very!) useful.

Sinceeachoperationreturnsarelation, operationscanbe*composed*!(Algebrais "closed".)

Projection

- Deletesattributesthatarenotinprojection list.
- Schemaofresult containsexactlythe fields intheprojectionlist, withthesamenamesthat they had in the (only) input relation. (Unary Operation)
- Projectionoperatorhastoeliminate duplicates!(as itreturnsarelationwhichisaset)
  - Note:realsystemstypicallydon't do duplicateeliminationunlesstheuserexplicitly asks for it.(Duplicate values may be representing different real world entityor relationship).

    Example:ConsidertheBOOKtable:

Acc-No	Title	Author
100	"DBMS"	"Silbershatz"
200	"DBMS"	"Ramanuj"
300	"COMPILER"	"Silbershatz"
400	"COMPILER"	"Ullman"
500	"OS"	"Sudarshan"
600	"DBMS"	"Silbershatz"

π_{Title}(BOOK)=

Title
"DBMS"
"COMPILER"
"OS"

Selection

- Selectsrowsthatsatisfy*selection condition.*
- Noduplicatesinresult
- *Schema*ofresultidenticaltoschemaof(only)inputrelation.
- *Result* relationcanbethe *input* foranotherrelationalalgebraoperation!(*Operator composition.*)

Example:Fortheexamplegivenabove:

$\sigma_{Acc\text{-}no>300}$(BOOK)=

Acc-No	Title	Author
400	"COMPILER"	"Ullman"
500	"OS"	"Sudarshan"

$\sigma_{Title="DBMS"}(BOOK)=$

Acc-No	Title	Author
100	"DBMS"	"Silbershatz"
200	"DBMS"	"Ramanuj"
600	"DBMS"	"Silbershatz"

$\pi_{Acc\text{-}no}(\sigma_{Title="DBMS"}(BOOK))=$

Acc-No
100
200
600

Union,Intersection,Set-Difference

- Alloftheseoperationstaketwo inputrelations,whichmustbe*union-compatible*:
  - Samenumber offields.
  - Corresponding'fieldshavethesame type.
- What isthe*schema*ofresult?

  Consider:

Borrower

Cust-name	Loan-no
Ram	L-13
Shyam	L-30
Suleman	L-42

Depositor

Cust-name	Acc-no
Suleman	A-100
Radheshyam	A-300
Ram	A-401

Listofcustomerswhoareeitherborroweror depositoratbank=$\pi_{Cust\text{-}name}$(Borrower)U $\pi_{Cust\text{-}name}$(Depositor)=

Cust-name
Ram
Shyam
Suleman
Radeshyam

Customerswho arebothborrowersanddepositors= $\pi_{Cust\text{-}name}$(Borrower) ∩ $\pi_{Cust\text{-}name}$(Depositor)=

Cust-name
Ram
Suleman

Customerswhoareborrowersbutnotdepositors=$\pi_{Cust\text{-}name}$(Borrower) □ $\pi_{Cust\text{-}name}$ (Depositor)=

Cust-name
Shyam

Cartesian-ProductorCross-Product(S1×R1)

- EachrowofS1is pairedwitheachrowof R1.
- *Resultschema*hasonefieldperfieldofS1andR1, withfieldnames`inherited'ifpossible.
- Considertheborrowerandloantablesasfollows:

Borrower:

Cust-name	**Loan-no**
Ram	L-13
Shyam	L-30
Suleman	L-42

Loan:

Loan-no	**Amount**
L-13	1000
L-30	20000
L-42	40000

Crossproduct ofBorrowerandLoan,Borrower×Loan=

Borrower.Cust-name	**Borrower.Loan-no**	**Loan.Loan-no**	**Loan.Amount**
Ram	L-13	L-13	1000
Ram	L-13	L-30	20000
Ram	L-13	L-42	40000
Shyam	L-30	L-13	1000
Shyam	L-30	L-30	20000
Shyam	L-30	L-42	40000
Suleman	L-42	L-13	1000
Suleman	L-42	L-30	20000
Suleman	L-42	L-42	40000

Therenameoperationcanbeusedtorenamethe fieldsto avoidconfusionwhentwo fieldnamesare same in two participating tables:

For example the statement, $\rho_{\text{Loan-borrower(Cust-name,Loan-No-1, Loan-No-2,Amount)}}$(Borrower × Loan) results into- Anew Table named Loan-borrower is created where it has four fields which are renamed as Cust-name, Loan-No-1, Loan-No-2 and Amount and the rows contains the same data as the cross product of Borrower and Loan.

Loan-borrower:

Cust-name	**Loan-No-1**	**Loan-No-2**	**Amount**
Ram	L-13	L-13	1000
Ram	L-13	L-30	20000
Ram	L-13	L-42	40000
Shyam	L-30	L-13	1000
Shyam	L-30	L-30	20000
Shyam	L-30	L-42	40000
Suleman	L-42	L-13	1000
Suleman	L-42	L-30	20000
Suleman	L-42	L-42	40000

RenameOperation:

Itcanbeusedintwoways:

- $\rho_x(E)$ returnthe resultofexpressionEinthetablenamed*x*.
- $\rho_{x(A_1,A_2,\ldots,A_n)}(E)$ returntheresultofexpressionE inthetable named *x*withtheattributes renamed to $A_1, A_2, \ldots, A_n$.
- It'sbenefit canbeunderstoodbythesolutionofthequery"Findthe largest accountbalance in the bank"

Itcanbesolvedbyfollowing steps:

- Findout therelationofthosebalanceswhicharenotlargest.
- ConsiderCartesionproductofAccountwithitselfi.e.Account×Account
- ComparethebalancesoffirstAccounttablewithbalancesofsecondAccounttable inthe product.
- Forthatweshouldrenameoneoftheaccounttablebysomeothernametoavoidthe confusion

Itcanbedonebyfollowingoperation

$\Pi_{Account.balance}(\sigma_{Account.balance<d.balance}(Account \times \rho_d(Account))$

- Sotheaboverelationcontainsthebalanceswhicharenotlargest.
- Subtractthisrelationfromtherelationcontainingallthe balances i.e. $\Pi_{balance}(Account)$. So

the final statement for solving above query is

$\Pi_{balance}(Account) - \Pi_{Account.balance}(\sigma_{Account.balance<d.balance}(Account \times \rho_d(Account))$

LECTURE-20

AdditionalOperations

NaturalJoin($S_1 \bowtie R_1$)

- FormsCartesianproduct ofitstwo arguments, performsselection forcing equalityon those attributes that appear in both relations
- For example consider Borrower and Loan relations, the natural join between them $\text{Borrower} \bowtie \text{Loan}$ willautomaticallyperformthe selectiononthe table returned by Borrower × Loan which force equality on the attribute that appear in both Borrower and Loan i.e. Loan-no and also will have only one of the column named Loan-No.
- That means $\text{Borrower} \bowtie \text{Loan}$ = $\sigma_{Borrower.Loan\text{-}no=Loan.Loan\text{-}no}$(Borrower×Loan).
- Thetablereturnedfromthiswillbeasfollows:

Eliminaterowsthatdoesnotsatisfytheselectioncriteria"$\sigma_{Borrower.Loan\text{-}no=Loan.Loan\text{-}no}$"fromBorrower ×Loan=

Borrower.Cust-name	Borrower.Loan-no	Loan.Loan-no	Loan.Amount
Ram	L-13	L-13	1000
~~Ram~~	~~L-13~~	~~L-30~~	~~20000~~
~~Ram~~	~~L-13~~	~~L-42~~	~~40000~~
~~Shyam~~	~~L-30~~	~~L-13~~	~~1000~~
Shyam	L-30	L-30	20000
~~Shyam~~	~~L-30~~	~~L-42~~	~~40000~~
~~Suleman~~	~~L-42~~	~~L-13~~	~~1000~~
~~Suleman~~	~~L-42~~	~~L-30~~	~~20000~~
Suleman	L-42	L-42	40000

Andwillremoveoneofthecolumn namedLoan-no.

- i.e. $\text{Borrower} \bowtie \text{Loan}$ =

Cust-name	Loan-no	Amount
Ram	L-13	1000
Shyam	L-30	20000
Suleman	L-42	40000

DivisionOperation:

- denotedby÷ isusedfor queriesthatincludethephrase"for all".
- Forexample"Findcustomerswhohasanaccountinallbranchesinbranchcity Agra". This query can be solved by following statement.

$\Pi_{Customer\text{-}name.branch\text{-}name}(\text{Depositor} \bowtie Account) \div \Pi_{branch\text{-}name}(\sigma_{Branch\text{-}city="Agra"}(Branch)$

- Thedivisionoperationscanbespecifiedbyusingonlybasicoperationsasfollows:

Let r(R)ands(S)begivenrelationsforschemaRandSwith $S \subseteq R$ r ÷ s =

$\Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r))$

LECTURE-21

TupleRelationalCalculus

Relationalalgebra isanexampleofprocedurallanguagewhiletuplerelationalcalculus isa nonprocedural query language.
Aqueryisspecifiedas:
{t| P(t)},i.eitisthesetofalltuplestsuchthatpredicatePistruefort.

The formulaP(t)is formedusingatomswhichusestherelations, tuplesofrelationsand fieldsof tuples and following symbols
$\in$(belongs to),<,>,$\leq$,$\geq$,$\neq$,=, (comparison operators)
Theseatomscanthenbeused toformformulaswithfollowing symbols
$\forall$ (universal qualifier generally called "for all")
$\exists$ (existential qualifier generally called "there exists")
$\wedge$ (and),$\vee$ (or),$\neg$(not)
Forexample: herearesomequeriesand awaytoexpressthemusing tuple calculus:

- Findthebranch-name,loan-numberandamount forloansoverRs1200.
  $\{t \mid t \in Loan \wedge t[amount] > 1200\}$.

- Findtheloannumberforeachloanofanamountgreaterthat Rs1200.
  $\{t \mid \exists s \in Loan(t[loan\text{-}number] = s[loan\text{-}number] \wedge s[amount] > 1200\}$

- Findthenamesofallthecustomerswhohavealoan fromtheSadarbranch.
  $\{t \mid \exists s \in Borrower (t[customer\text{-}name] = s[customer\text{-}name] \wedge$
  $\exists u \in Loan (u[loan\text{-}number] = s[loan\text{-}number$
  $\wedge u[branch\text{-}name] = "Sadar"))\}$

- Findallcustomerswhohavealoan,anaccount,orbothatthebank
  $\{t \mid \exists s \in Borrower (t[customer\text{-}name] = s[customer\text{-}name])$
  $\vee \exists u \in Depositor (t[customer\text{-}name] = u[customer\text{-}name])\}$

- Findonlythosecustomerswhohavebothanaccountandaloan.
  $\{t \mid \exists s \in Borrower (t[customer\text{-}name] = s[customer\text{-}name])$
  $\wedge \exists u \in Depositor (t[customer\text{-}name] = u[customer\text{-}name])\}$

- Findallcustomerswhohaveanaccountbutdonothaveloan.
  $\{t \mid \exists u \in Depositor (t[customer\text{-}name] = u[customer\text{-}name]) \wedge$
  $\neg \exists s \in Borrower (t[customer\text{-}name] = s[customer\text{-}name])\}$

- Findallcustomerswhohaveanaccountatallbranches locatedinAgra
  $\{t \mid \forall w \in Branch(w[branch\text{-}city] = "Agra" \Rightarrow$
  $\exists s \in Depositor (t[customer\text{-}name] = s[customer\text{-}name]$
  $\wedge \exists u \in Account (u[account\text{-}number] = s[account\text{-}number]$
  $\wedge u[branch\text{-}name] = w[branch\text{-}name]))\}$

DomainRelationalCalculus

1. Domainrelationalcalculusisanothernonproceduralanguageforexpressingdatabase queries.
2. Aqueryisspecifiedas:
   $\{<x_1,x_2,\ldots,x_n>|P(x_1,x_2,\ldots,x_n)\}$ where $x_1,x_2,\ldots,x_n$ representsdomainvariables.Prepresent a predicate formula as in tuple calculus

- Sincethedomainvariablesarereferredinplaceoftuplestheformuladoesn'treferthefields oftuplesrathertheyreferthe domainvariables.
- Forexamplethequeries indomaincalculusarementionedas follows:
  - Findthebranch-name,loan-numberandamountforloansoverRs1200.
    $\{<b, l, a>| <b, l, a> \in Loan \wedge a > 1200\}$.
  - Findtheloannumberforeachloanofanamountgreaterthat Rs1200.
    $\{< l >| \exists\ b,a(<b, l, a> \in Loan \wedge a >1200\}$
  - Find the names ofallthe customers who have a loan fromthe Sadar branch and find the loan amount
    $\{< c, a > | \exists\ l(< c, l > \in Borrower$
    $\exists\ b(< b, l, a >\in Loan \wedge b="Sadar"))\}$
  - Findnamesofallcustomerswhohavealoan,anaccount,orbothattheSadar Branch
    $\{<c>| \exists\ l(< c, l > \in Borrower \wedge \exists\ b, a(<b, l, a> \in Loan \wedge b ="Sadar"))$
    $\vee \exists\ a(<c, a> \in Depositor \wedge \exists\ b, n(<b, a, n> \in Account \wedge b ="Sadar"))\}$
  - Findonlythosecustomerswhohavebothanaccountandaloan.
    $\{<c>| \exists\ l(<c, l> \in Borrower) \wedge \exists\ a(<c, a> \in Depositor)\}$
  - Findallcustomerswhohaveanaccountbutdonothaveloan.
    $\{t| \exists\ a(<c, a> \in Depositor) \wedge \neg \exists\ l(<c, l> \in Borrower)\}$
  - Findallcustomerswhohaveanaccountatallbranches locatedinAgra
    $\{<c> | \ \forall\ x, y, z(<x, y, z> \in Branch) \wedge y = "Agra" \Rightarrow$

OuterJoin.

Outerjoinoperationisanextensionofjoinoperationtodealwithmissinginformation

- Supposethatwehavefollowingrelationalschemas:
  Employee(employee-name, street, city)
  Fulltime-works(employee-name,branch-name,salary)
  Asnapshot of these relations is as follows:

Employee:

employee-name	street	city
Ram	M GRoad	Agra
Shyam	New Mandi Road	Mathura
Suleman	BhagatSingh Road	Aligarh

Fulltime-works

employee-name	branch-name	salary
Ram	Sadar	30000
Shyam	SanjayPlace	20000
Rehman	Dayalbagh	40000

Supposewewant completeinformationofthefulltimeemployees.

- The natural join ($Employee \bowtie Fulltime\text{-}works$)will result into the loss of information for Suleman and Rehman because they don't have record in both the tables (left and right relation). The outer join will solve the problem.
- Threeformsofouterjoin:
  - **Left outer join($\supset\bowtie$):**the tuples which doesn't match while doing natural join from left relation are also added in the result putting null values in missing field of right relation.
  - **Right outer join($\bowtie\subset$):**the tuples which doesn't match while natural join from right relation are also added in the result putting null values in missing field of left relation.
  - **Full outer join($\supset\bowtie\subset$):** include both of the left and right outer joins i.e. adds the tuples which did not match either in left relation or right relation and put null inplace of missing values.
- Theresultforthreeformsofouterjoinareasfollows:

Leftjoin: $Employee \supset\bowtie Fulltime\text{-}works_{=}$

employee-name	**street**	**City**	**branch-name**	**salary**
Ram	M GRoad	Agra	Sadar	30000
Shyam	New Mandi Road	Mathura	Sanjay Place	20000
Suleman	BhagatSingh Road	Aligarh	Null	Null

Rightjoin: $Employee \bowtie\subset Fulltime\text{-}works_{=}$

employee-name	**street**	**city**	**branch-name**	**salary**
Ram	M GRoad	Agra	Sadar	30000
Shyam	New Mandi Road	Mathura	Sanjay Place	20000
Rehman	null	null	Dayalbagh	40000

Fulljoin: $Employee \supset\bowtie\subset Fulltime\text{-}works_{=}$

employee-name	**street**	**city**	**branch-name**	**salary**
Ram	M GRoad	Agra	Sadar	30000
Shyam	New Mandi Road	Mathura	Sanjay Place	20000
Suleman	BhagatSingh Road	Aligarh	null	null
Rehman	null	null	Dayalbagh	40000

DBMS

LECTURE-22

StructuredQueryLanguage(SQL)

Introduction

- Commercialdatabasesystemsuse moreuserfriendlylanguageto specifythequeries. SQL is
- the most influential commercially marketed product language.
- Othercommerciallyused languagesareQBE,Quel,andDatalog.

BasicStructure

- ThebasicstructureofanSQLconsistsofthreeclauses:**select,from**and **where.**
- **select:** it correspondstotheprojectionoperationofrelationalalgebra.Usedtolist the attributes desired in the result.
- **from**:correspondstotheCartesianproductoperationofrelationalalgebra.Usedtolist the relationstobescannedintheevaluationoftheexpression
- **where**:correspondstotheselectionpredicateoftherelationalalgebra. Itconsistsofa predicate involving attributes of the relations that appear in the **from** clause.
- AtypicalSQLqueryhastheform:

  select $A_1, A_2, \ldots, A_n$ **from** $r_1, r_2, \ldots, r_m$ **where** P
  - A_i representsanattribute
  - r_j representsa relation
  - Pisa predicate
  - Itisequivalenttofollowingrelationalalgebraexpression:
  - $\Pi_{A_1, A_2, \ldots, A_n}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m))$

[Note:** The words marked in dark in this text work as keywords in SQL language. For example "select", "from" and "where" intheaboveparagraphareshowninboldfonttoindicatethatthey are keywords**]

SelectClause

Letusseesomesimplequeriesanduseof**select**clausetoexpresstheminSQL.

- Findthenamesofall branchesintheLoanrelation

  select branch-name
  from Loan
- Bydefault the **select** clause includesduplicatevalues. Ifwewanttoforcetheeliminationof duplicates the **distinct** keyword is used as follows:

  select distinct branch-name
  from Loan
- The**all**keywordcanbeusedtospecify*explicitly* thatduplicatesarenotremoved. Evenif we not use **all** it means the same so we don't require **all** to use in **select** clause.

  select all branch-name
  from Loan
- Theasterisk"*"canbeusedtodenote"allattributes".ThefollowingSQLstatementwill selectandallthe attributesofLoan.

  select *
  from Loan
- The arithmetic expressions involving operators, +, -, *, and / are also allowed in **select** clause.The followingstatement willreturntheamountmultipliedby100 fortherowsin Loan table.

selectbranch-name,loan-number,amount*10**from**Loan.

WhereClause

- Findallloannumbers forloans madeat“Sadar”branchwithloanamountsgreaterthanRs 1200.

  selectloan-number
  fromLoan
  wherebranch-name=“Sadar”**and**amount>1200
- **where**clauseusesuseslogivalconnectives**and**,**or**, and**not**
- operandsofthelogicalconnectivescanbeexpressionsinvolvingthecomparisonoperators <,<=,>,>=,=,and<>.
- **between**canbeused tosimplifythecomparisons

  selectloan-number
  fromLoan
  whereamountbetween90000and100000

FromClause

- The**from**clausebyitselfdefinesaCartesianproductoftherelations inthe clause.
- Whenanattribute ispresent inmorethanonerelationtheycanbereferredas *relation-name.attribute-name* to avoid the ambiguity.
- Forallcustomerswhohaveloanfromthebank,findtheirnamesandloannumbers

  selectdistinctcustomer-name,Borrower.loan-number
  fromBorrower,Loan
  whereBorrower.loan-number=Loan.loan-number

TheRenameOperation

- Usedfor renamingbothrelationsbothrelationsand attributesin SQL
- Use**as**clause:old-name**as**new-name
- Findthenamesandloannumbersofthecustomers whohavealoanatthe“Sadar” branch.

  selectdistinctcustomer-name,borrower.loan-number**as**loan-id
  fromBorrower,Loan
  whereBorrower.loan-number=Loan.loan-number**and**
  branch-name= “Sadar”

  wecannowrefertheloan-numberinsteadbythenameloan-id.
- Forallcustomerswhohavealoanfromthebank,findtheirnamesandloan-numbers.

  selectdistinctcustomer-name,T.loan-number
  fromBorrower**as**T,Loan**as**S
  whereT.loan-number=S.loan-number
- Findthenamesofallbranchesthathaveassetsgreaterthanatleastonebranch locatedin “Mathura”.

  selectdistinctT.branch-name
  frombranch**as**T,branch**as**S
  whereT.assets>S.assets**and**S.branch-city= “Mathura”

StringOperation

- Twospecialcharactersareusedforpatternmatching instrings:
  - Percent(%):The%charactermatchesanysubstring
  - Underscore(_):The _ charactermatchesanycharacter
- “%Mandi”:willmatchwiththestringsendingwith“Mandi” viz. “RajaKimandi”,“Peepal Mandi”
- “_ _ _”matchesanystring ofthree characters.
- Findthenamesofallcustomerswhosestreetaddress includesthesubstring“Main”

selectcustomer-name
fromCustomer
wherecustomer-streetlike"%Main%"

SetOperations

- **union,intersectandexcept**operationsaresetoperationsavailableinSQL.
- Relations participating in any of the set operation mustbe compatible; i.e.they musthave the same set of attributes.
- UnionOperation:
  - Findallcustomershavinga loan,anaccount,orbothatthebank
    (**select** customer-name**from** Depositor)
    union
    (**select**customer-name **from**Borrower)
    It will automatically eliminate duplicates.
  - Ifwewanttoretainduplicates **unionall**canbeused
    (**select** customer-name**from** Depositor)
    unionall
    (**select**customer-name**from**Borrower)
- IntersectOperation
  - Findallcustomerswho havebothanaccount andaloanatthebank
    (**select** customer-name**from** Depositor)
    intersect
    (**select**customer-name **from**Borrower)
  - If we want to retail all the duplicates
    (**select**customer-name**from**Depositor)
    intersect all
    (**select**customer-name**from**Borrower)
- ExceptOpeartion
  - Findallcustomerswho haveanaccount butno loanatthebank
    (**select** customer-name **from** Depositor)
    except
    (**select**customer-name**from**Borrower)
  - Ifwe want to retain the duplicates:
    (**select**customer-name**from**Depositor)
    except all
    (**select**customer-name**from**Borrower)

AggregateFunctions

- Aggregate functionsarethose functionswhichtakeacollectionofvaluesas input andreturn a single value.
- SQLoffers5builtinaggregatefunctions-
  - Average:**avg**
  - Minimum:**min**
  - Maximum:**max**
  - Total:**sum**
  - Count:**count**
- Theinput to **sum** and **avg** must be a collection of numbers but others may have collections of non-numeric data types as input as well
- FindtheaverageaccountbalanceattheSadar branch
  selectavg(balance)

fromAccount
wherebranch-name="Sadar"

Theresultwillbeatablewhichcontainssinglecell(onerowandonecolumn)having numerical value corresponding to average balance of all account at sadar branch.

- **groupby**clauseisusedtoformgroups,tupleswiththesamevalueonallattributesinthe **groupby**clauseareplacedinonegroup.
- Findtheaverageaccountbalanceateachbranch

  selectbranch-name,**avg**(balance)
  fromAccount
  groupbybranch-name
- Bydefaulttheaggregatefunctionsincludethe duplicates.
- **distinct**keywordisused toeliminateduplicates inanaggregate functions:
- Findthenumberofdepositorsfor eachbranch

  selectbranch-name,**count**(**distinct**customer-name)
  fromDepositor,Account
  whereDepositor.account-number=Account.account-number
  groupbybranch-name
- **having**clauseisusedtostateconditionthatappliestogroupsratherthantuples.
- Findtheaverageaccountbalanceateachbranchwhereaverageaccountbalanceismore than Rs. 1200

  selectbranch-name,**avg**(balance)
  fromAccount
  groupbybranch-name
  havingavg(balance) >1200
- Countthenumber oftuplesinCustomertable

  select count(*)
  fromCustomer
- SQLdoesn'tallow**distinct**with**count**(*)
- When**where**and**having**arebothpresentina statement**where**isappliedbefore **having**.

LECTURE-23

NestedSubqueries

Asubqueryisa**select-from-where**expressionthatisnestedwithinanother query.

Set Membership

The**in**and **not in**connectivesareusedforthistypeofsubquery.

"Findallcustomerswhohavebothaloanandanaccountatthebank",thisquerycanbewritten usingnested subqueryformasfollows

selectdistinctcustomer-name
fromBorrower
wherecustomer-name**in**(**select**customer-name
fromDepositor)

- Selectthenamesofcustomerswhohavealoanatthebank,andwhosenamesareneither "Smith"nor "Jones"

selectdistinctcustomer-name
fromBorrower
wherecustomer-name**notin**("Smith","Jones")

SetComparison

Find the names of all branches thathave assets greater than those of at least one branch located in Mathura

selectbranch-name
fromBranch
whereasstets>**some**(**select** assets
fromBranch
wherebranch-city="Mathura")

1. Apartfrom>**some**otherscomparisoncould be<**some**,<=**some**,>=**some**,=**some**,< >**some.**
2. Findthenames of allbranches thathave assets greater than thatof each branchlocatedin Mathura

selectbranch-name
fromBranch
whereasstets>**all**(**select**assets
fromBranch
wherebranch-city= "Mathura")

- Apartfrom>**all**otherscomparisoncouldbe <**all**,<= **all**,>=**all**,=**all**, <>**all.**

Views

InSQL**createview**commandisusedtodefineaviewasfollows:

create viewv as<queryexpression>
where<queryexpression>isanylegalqueryexpressionandv istheviewname.

- Theviewconsistingofbranchnamesandthenamesofcustomerswho haveeitheran account or a loan at the branch. This can be defined as follows:

createviewAll-customer**as**
(**select**branch-name,customer-name
fromDepositor,Account
whereDepositor.account-number=account.account-number)
union

(**select**branch-name,customer-name
fromBorrower,Loan
whereBorrower.loan-number=Loan.loan-number)

- Theattributesnames maybespecifiedexplicitlywithinaset ofroundbracket afterthename of view.
- Theviewnamesmaybeusedasrelationsinsubsequent queries.Usingtheview Allcustomer

FindallcustomersofSadarbranch
selectcustomer-name
fromAll-customer
wherebranch-name= "Sadar"

- A**create-view**clausecreatesaview definition inthedatabasewhichstaysuntila command - **dropview***view-name*- isexecuted.

ModificationofDatabase

Deletion

- InSQLwecandeleteonlywholetupleand notthevaluesonanyparticular

attributes. The command is as follows:

deletefromr**where**P.
wherePisapredicateandr isa relation.

- **delete**command operatesononlyonerelationatatime.Examplesareasfollows:
- Delete alltuplesfromtheLoanrelation

deletefromLoan

- DeleteallottheSmith'saccountrecord

deletefromDepositor
wherecustomer-name="Smith"

- Deleteallloanswith loanamountsbetweenRs1300 and Rs1500.
  deletefromLoan
  whereamount**between**1300**and** 1500
- Deletetherecordsofallaccountswith balancesbelowtheaverageat the bank

deletefromAccount
wherebalance<(**selectavg**(balance)
from Account)

Insertion

InSQLweeither specifyatupleto beinsertedorwriteaquerywhoseresult isa set of tuples to be inserted. Examples are as follows:
Insert anaccountofaccount numberA-9732attheSadarbranchhaving balance of Rs 1200
insertintoAccount
values("Sadar","A-9732",1200)
thevaluesarespecified intheorderinwhichthecorrespondingattributesare listed in the relation schema.
SQLallowstheattributesto bespecifiedaspartofthe **insert**statement
insert into Account(account-number, branch-name, balance)
values("A-9732", "Sadar", 1200)
insertintoAccount(branch-name,account-number,balance)

values(“Sadar”,“A-9732”,1200)

Provide for all loan customers of the Sadar branch a new Rs 200 saving account foreachloanaccounttheyhave.Where loan-numberserveastheaccount number for these accounts.
insertintoAccount
selectbranch-name,loan-number,200
fromLoan
wherebranch-name=“Sadar”

Updates

Used to change a value in a tuple without changing all values in the tuple. Supposethat annual interest paymentsare being made, and allbalancesare to be increased by 5 percent.
updateAccount
setbalance=balance*1.05
SupposethataccountswithbalancesoverRs10000 receive6percent interest, whereas all others receive 5 percent.
updateAccount
setbalance=balance*1.06
where balance > 10000
update Account
setbalance=balance*1.05
wherebalance<=10000

DataDefinitionLanguage

Data Types in SQL

char(n): fixed lengthcharacterstring,lengthn.
varchar(n):variablelengthcharacterstring,maximumlengthn.
int:aninteger.
smallint:asmallinteger.
numeric(p,d):fixedpoint number,pdigits(plusasign),anddofthepdigitsare to right of the decimal point.
real,doubleprecision:floatingpointanddoubleprecisionnumbers.
float(n):afloatingpointnumber,precisionat leastndigits.
date:calendardate;fourdigits foryear,twoformonthandtwofordayofmonth.
time:timeofdaynhoursminutesandseconds.

Domainscan bedefined as
createdomainperson-name**char**(20).
thedomainnameperson-namecanbeusedto definethetypeofanattribute just like built-in domain.

SchemaDefinition in SQL
createtablecommandisusedtodefinerelations.
createtabler(A1D1,A2D2,…,AnDn,
<integrityconstraint1>,
…,
<integrityconstraintk>)

whererisrelation name,eachAiisthenameofattribute,Diisthedomaintypeof

valuesofAi.SeveraltypesofintegrityconstraintsareavailabletodefineinSQL.

IntegrityConstraintswhichareallowedinSQLare

primarykey(Aj1,Aj2,…,Ajm)
and
check(P)wherePisthepredicate.

drop table command is used to remove relations from database.
altertablecommand isusedto addattributestoanexistingrelation **alter table** r **add** A D
itwilladdattributeAofdomaintypeDinrelationr.
altertabler **drop**A
itwillremovetheattributeAofrelationr.

LECTURE-24

IntegrityConstraints

- IntegrityConstraintsguard againstaccidentaldamagetothedatabase.
- Integrityconstraintsare predicatespertainingtothe database.
- **DomainConstraints:**
- PredicatesdefinedonthedomainsareDomainconstraints.
- Simplest Domainconstraintsaredefinedbydefiningstandarddatatypesoftheattributes like Integer, Double, Float, etc.
- Wecandefinedomainsby**createdomain** clausealsowecandefinetheconstraintsonsuch domainsasfollows:

  create domainhourly-wage **numeric**(5,2)
  constraintwage-value-testcheck(value>=4.00)
- Sowecanusehourly-wageasdatatypeforanyattributewhere DBMS willautomatically allow onlyvaluesgreaterthanorequalto4.00.
- OtherexamplesfordefiningDomainconstraintsareasfollows:

  createdomainaccount-number**char**(10)
  constraintaccount-number-null-test**check(valuenotnull)**
  create domain account-type **char**(10)
  constraintaccount-type-test
  check(valuein("Checking","Saving"))

Byusingthe laterdomainoftwo abovetheDBMS willallowonlyvalues foranyattributehaving type as account-type i.e. Checking and Saving.

- **ReferentialIntegrity:**
- **Foreign Key**:Iftwo table R and S are related to each other, K1 and K2 are primarykeys of the two relations also K1 is one of the attribute in S. Suppose we want that every row in S must have a corresponding row inR,thenwedefine the K1 inS as **foreign key**. Example in ouroriginaldatabase oflibrarywe had a table for relationBORROWEDBY, containing two fields Card No. and Acc. No. . Every row of BORROWEDBY relation must have corresponding row in USER Table having same Card No. and a row in BOOK table having sameAcc. No..Thenwe willdefine the Card No. andAcc. No. inBORROWEDBYrelation as foreign keys.
- Inother waywe cansaythat everyrowofBORROWEDBYrelation must refer to some row in BOOK and also in USER tables.
- Suchreferentialrequirementinonetableto anothertableiscalled ReferentialIntegrity.

LECTURE-25

QuerybyExample(QBE)

Query by Example (QBE)is amethod ofquerycreation thatallows the user tosearch for documents based on an example in the form of a selected text string or in the form of a document name or a list ofdocuments. Because the QBE systemformulates the actualquery, QBE is easier to learn than formal query languages, such as the standard Structured Query Language (SQL), while still enabling powerful searches.

Selectionsin QBE

QBEusesskeletontablestorepresenttablenameandfieldnameslike:

Tablename	Field1	Field2	…..

Forselection,P operatoralongwithvariablename/constantnameisusedtodisplayoneormore fields.

Example1:

Considertherelation:student(name,roll,marks)

The following query can be represented as:

SQL:selectnamefromstudentwheremarks>50;

student	name	roll	marks
	P.X		>50

HereX isa constant;alternativelywecanuse _X asavariable.

Example2:

Fortherelationgiven above

Thefollowing querycanberepresented as:

SQL:select*fromstudentwhere marks>50and marks<80;

student	name	roll	marks
P.			_X

CONDITION
_X>50∧_X<80

Wecanuseconditionboxtorepresentcomplexconditions.

Example3:

Fortherelationgiven above

Thefollowing querycanberepresented as:

SQL:selectname, rollfromstudentwheremarks<50ormarks>80;

student	name	roll	marks
	P.A	P.B	<50
	P.A	P.B	>80

ORoperationretrievesresultsinmultiplerows.

Example4:

(JoinsinQBE)

Considerthefollowingtables:

Student (name, roll, marks)

Grades (roll, grade)

Thefollowing querycanberepresented as:

SQL:select s.name,g.gradefromStudents,Gradesgwheres.roll=g.rollands.marks>50; Uses two skeleton tables:

Student	name	roll	marks
	P.A	_X	>50

And

Grades	roll	grade
	_X	P.B

Insertionsin QBE:

Uses operator I.onthetable.
Example:Consider thefollowingqueryonStudenttable
SQL:insertintostudentvalues('abc',10,60);

Student	name	roll	marks
I.	abc	10	60

Multipleinsertionscanberepresented byseparaterowsinskeletontable.

DeletionsinQBE:

UsesoperatorD.onthetable.
Example:ConsiderthefollowingqueryonStudenttable SQL:
delete from student where marks=0;

Student	name	roll	marks
D.			0

Multipledeletionscanberepresentedbyseparaterowsinskeletontable. Deletions without any condition can truncate the entire table.

UpdationinQBE:

UsesoperatorU.onthetable.
Example:ConsiderthefollowingqueryonStudenttable SQL:
update student set mark=50 where roll=40;

Student	name	roll	marks
U.			50
		40	

LECTURE-26

RELATIONALDATABASEDEGIN

Database design is a process in which you create a logical data model for a database, which store data of a company. It is performed after initial database study phase in the database life cycle. You use normalization technique to create the logical data model for a database and eliminate data redundancy.

Normalization also allows you to organize data efficiently in a database and reduce anomalies during dataoperation.Variousnormal forms, suchas first, second and third canbe applied to create a logical data model for a database. The second and third normal forms are based on partial dependencyand transitivitydependency. Partialdependencyoccurs when a rowoftable is uniquely identified by one column that is a part of a primary key. A transitivity dependency occurs when a non keycolumn is uniquely identified by values in another non-keycolumn of a table.

DatabaseDesignProcess:

Wecanidentifysixmainphasesofthedatabasedesignprocess:

1. Requirementcollectionandanalysis
2. Conceptualdatabase design
3. Choice ofaDBMS
4. Datamodelmapping(logicaldatabasedesign)
5. Physicaldatabasedesign
6. Databasesystemimplementationand tuning

1. **RequirementCollectionandAnalysis**
   Before we can effectively design a data base we must know and analyze the expectation of the users and the intended uses of the database in as much as detail.

2. **ConceptualDatabaseDesign**
   Thegoalforthisphaseistoproduceaconceptualschemaforthedatabasethatis independent of a specific DBMS.
   - Weoftenuseahigh leveldatamodelsuchER-modelduring thisphase
   - We specifyas manyofknowndatabase applicationontransactions as possible using a notation that is independent of any specific dbms.
   - Often the dbms choiceis already madefor the organization theintent of conceptual designstill to keep , it as free as possible from implementation consideration.

3. **Choice ofaDBMS**
   The choice of dbms is governed by a no. of factors some technical other economic and still other concerned with the politics of the organization.
   Theeconomicsandorganizationalfactorsthatofferthechoiceofthedbmsare:
   Softwarecost,maintenance cost,hardwarecost,databasecreationandconversioncost, personnel cost, training cost, operating cost.

4. **Datamodelmapping(logicaldatabasedesign)**
   Duringthisphase,wemaptheconceptualschemafromthehighleveldatamodelusedon

phase2intoadatamodelofthechoice dbms.

5. **Physicaldatabsedesign**
   During this phase we design the specification for the database in terms of physical storagestructure ,record placement and indexes.
6. **Databasesystemimplementationandtuning**
   Duringthisphase,thedatabaseandapplicationprogramsareimplemented,testedand eventually deployed for service.

InformalGuidelinesforRelationDesign

Want to keep the semantics ofthe relation attributes clear.The information in a tuple should represent exactlyone fact oranentity.Thehiddenorburiedentitiesarewhat wewantto discover and eliminate.

- Designarelationschemasothatitiseasytoexplainitsmeaning.
- Do notcombineattributesfrommultipleentitytypesandrelationshiptypesinto asingle relation. Use a view if you want to present a simpler layout to the end user.
- Arelationschemashouldcorrespond toonentitytypeorrelationship type.
- Minimizeredundantinformationintuples,thusreducingupdate anomalies
- Ifanomaliesarepresent,trytodecomposetherelationintotwoormoreto representthe separate facts, or document the anomalies well for management in the applications programs.

Minimizetheuseof**null**values.Nulls havemultipleinterpretations:

- Theattributedoesnotapplyto thistuple
- Theattributevalueisunknown
- Theattributevalue isabsent
- Theattributevaluemightrepresentanactualvalue

Ifnullsare likely(non-applicable) thenconsider decompositionoftherelationintotwoor more relations that hold only the non-null valued tuples.

- Donotpermitthecreationofspurioustuples

Toomuchdecompositionofrelationsintosmaller onesmayalso loseinformationorgenerate erroneous information

- Besurethatrelationscanbe logicallyjoinedusingnaturaljoinandtheresult doesn't generate relationships that don't exist

FunctionalDependencies

FD'sareconstraintsonwell-formedrelationsandrepresent aformalismonthe infrastructureof relation.

Definition: A*functionaldependency*(FD)onarelationschema**R**isaconstraint***X*→*Y*,**where*X* and*Y*are subsetsofattributesof**R.**

Definition:anFD isarelationshipbetweenanattribute"Y"andadeterminant (1ormoreother attributes) "X"suchthat for agivenvalueofadeterminantthevalueoftheattributeisuniquely defined.

- Xisa determinant
- XdeterminesY
- Yis functionallydependentonX
- X→Y
- X→YistrivialifY⊆X

Definition:AnFD ***X→Y***is*satisfied*inan instance**r**of**R**iffor<u>every</u>pairoftuples,*t*and s:if*t* andsagreeonallattributesin *X*thentheymustagree onallattributesin*Y*

Akeyconstraint isa specialkind offunctionaldependency: allattributesofrelationoccuronthe right-hand side of the FD:

- *SSN→ SSN,Name,Address*

ExampleFunctionalDependencies

LetRbe
NewStudent(***stuId,lastName,major,credits,status,socSecNo***)

FDs in R include

- ***{stuId}→{lastName}***,butnotthereverse
- ***{stuId}→{lastName,major,credits,status,socSecNo,stuId}***
- ***{socSecNo}→{stuId,lastName,major,credits,status,socSecNo}***
- ***{credits}→{status}***,butnot***{status}→{credits}***

ZipCode→AddressCity

- 16652isHuntingdon'sZIP

ArtistName→BirthYear

- Picassowasbornin1881

Autobrand→Manufacturer,Enginetype

- PontiacisbuiltbyGeneralMotorswithgasoline engine

Author,Title→PublDate

- Shakespeare'sHamletwaspublishedin1600

TrivialFunctionalDependency

TheFDX→Yis***trivial*** ifset {Y}isasubset ofset{X}

Examples: IfAand B are attributes of R,

- {A}→{A}
- {A,B}→{A}
- {A,B}→{B}
- {A,B}→{A,B}

arealltrivial FDs and will not contributeto theevaluationof normalization.

FDAxioms

Understanding: FunctionalDependenciesarerecognized byanalysisoftherealworld;no automation or algorithm. Finding or recognizing them are the database designer's task.

FDmanipulations:

- **Soundness**--noincorrectFD'saregenerated
- **Completeness**-- allFD'scanbe generated

AxiomName	Axiom	Example
Reflexivity	ifaissetofattributes,b⊆a, thena→b	*SSN,Name→ SSN*
Augmentation	ifa→ bholdsandc isasetof attributes, then ca→cb	*SSN→Name*then *SSN,Phone→Name,Phone*
Transitivity	ifa→bholdsandb→cholds, then a→ c holds	*SSN→Zip*and*Zip→City*then *SSN→City*
UnionorAdditivity*	ifa→banda→cholsthen a→ bc holds	*SSN→Name*and*SSN→Zip*then*SSN→Name,Zip*
Decompositionor Projectivity*	ifa→bc holdsthena→b and a → c holds	*SSN→Name,Zip*then*SSN→Name*and*SSN→Zip*
Pseudotransitivity*	ifa→bandcb→dholdthen ac → d holds	*Address→Project*and*Project,Date→Amount*then *Address,Date→Amount*
(NOTE)	*ab→cdoesNOTimplya→c and b → c*	

*Armstrong'sAxioms(basicaxioms)

LECTURE-27

CLOSUREOFASETOFFUNCTIONALDEPEDENCIES

GivenarelationalschemaR,afunctionaldependenciesfonRislogicallyimpliedbyasetof functionaldependencies F on R if everyrelation instance r(R) that satisfies F also satisfies f.

TheclosureofF,denotedbyF^+, isthesetofallfunctionaldependencieslogicallyimplied byF. The closure of F can be found by using a collection of rules called **Armstrong axioms.**
Reflexivityrule: IfA*isasetofattributesandB issubsetorequaltoA, then*A→Bholds.
Augmentation rule: IfA→B holds and C is a set of attributes, then CA→CB holds
Transitivity rule: IfA→B holds and B→C holds, thenA→C holds.
Unionrule: IfA→BholdsandA→CthenA→BCholds
Decompositionrule: IfA→BC holds, thenA→B holdsandA→Cholds.
Pseudotransitivityrule: IfA→BholdsandBC→Dholds,thenAC→Dholds.

SupposewearegivenarelationschemaR=(A,B,C,G,H,I)andthesetoffunctiondependencies {A→B,A→C,CG→H,CG→I,B→H}

WelistseveralmembersofF^+here:

1. A→H,sinceA→BandB→Hhold, weapplythetransitivityrule.
2. CG→HI.SinceCG→HandCG→I ,theunionruleimpliesthatCG→HI
3. AG→I,sinceA→CandCG→I,thepseudotransitivityruleimpliesthatAG→Iholds

AlgorithmofcomputeF+:

TocomputetheclosureofasetoffunctionaldependenciesF:

F+=F

repeat

foreachfunctionaldependency*f*in*F+*

applyreflexivityand augmentationruleson*f*

addtheresultingfunctionaldependenciesto*F+*

foreachpairoffunctionaldependencies*f*1and *f*2in*F+*

if*f*1 and*f*2canbecombined usingtransitivity

thenaddtheresultingfunctionaldependencyto*F+*

until*F+* doesnot change anyfurther

large.

LECTURE-28

LOSSLESS DECOMPOSITION

Adecompositionofa relationscheme R<S,F> intothe relationschemes Ri(1<=i<=n) is said to be a lossless join decomposition or simply lossless if for everyrelation R that satisfies the FDs in F, the natural join of the projections or R gives the original relation R, i.e,

$R=\pi_{R1}(R) \bowtie \pi_{R2}(R) \bowtie \ldots\ldots \bowtie \pi_{Rn}(R)$

IfR is subset of $\pi_{R1}(R) \bowtie \pi_{R2}(R) \bowtie \ldots\ldots \bowtie \pi_{Rn}(R)$

Then the decomposition is called lossy.

DEPEDENCYPRSERVATION:

GivenarelationschemeR<S,F>whereFistheassociatedsetof functional dependencieson the attributesinS,R isdecomposed intothe relationschemesR1,R2,…Rnwiththe fdsF1,F2…Fn,then this decomposition ofR is dependency preserving ifthe closure ofF'(where F'=F1 U F2 U … Fn)

Example:

LetR(A,B,C)ANDF={A→B}. ThenthedecompositionofRintoR1(A,B)andR2(A,C)islossless becausetheFD{A→B}iscontainedinR1 andthecommonattributeAisakeyofR1.

Example:

Let R(A,B,C) AND F={A→B}. Then the decomposition of R into R1(A,B) and R2(B,C) is not lossless because thecommonattribute B does not functionallydetermine eitherAor C. i.e, it is not a key of R1 or R 2.

Example:

Let R(A,B,C,D) and F={A→B,A→C, C→D,}. Then the decomposition of R into R1(A,B,C) with the FD F1={ A→B , A→C }and R2(C,D) with FD F2={ C→D} . In this decomposition all the original FDs can be logically derived from F1 and F2, hence the decomposition is dependency preserving also . the common attribute C forms a keyof R2. The decomposition is lossless.

Example:

Let R(A,B,C,D) and F={A→B, A→C, A→D,}. Then the decomposition ofR into R1(A,B,D) with the FD F1={A→B ,A→D }and R2(B,C) withFD F2={ } is lossy because the commonattribute B is not a candidate key of either R1 and R2 .

Inaddition,thefdsA→Cisnotimpliedbyanyfds R1orR2.Thusthedecompositionisnot dependencypreserving.

Fullfunctionaldependency:

Givena relationalscheme R and anFD X→Y,Yis fullyfunctionaldependent onXifthere is no Z, where Z is a proper subset ofX such that Z→Y.The dependency X→Yis left reduced, there being no extraneous attributes attributes in the left hand side of the dependency.

Partialdependency:

GivenarelationdependenciesFdefinedontheattributesofRandKasacandidatekey,ifXisa propersubset ofKandifF|=X→A,thenAissaid tobepartialdependent onK

Primeattributeandnonprime attribute:

A attribute A in a relation scheme R is a **prime attribute** or simply **prime** if A is part of any candidate key of the relation. If A is not a part of any candidate key of R, A is called a nonprime attribute or simply **non prime** .

Trivialfunctionaldependency:
AFDX→Yissaid to beatrivialfunctionaldependency ifYissubsetofX.

LECTURE-29

Normalization

While designing a database out of an entity–relationship model, the main problem existing in that "raw" database is redundancy. Redundancy is storing the same data item in more one place. A redundancy creates several problems like the following:

1. Extrastoragespace:storingthesamedatainmanyplacestakeslargeamountofdiskspace.
2. Enteringsamedatamorethanonceduring datainsertion.
3. Deletingdatafrommorethanoneplaceduring deletion.
4. Modifyingdatainmorethanoneplace.
5. Anomaliesmayoccurinthedatabaseifinsertion,deletion,modificationetcarenodone properly. It creates inconsistency and unreliability in the database.

To solve this problem, the "raw" database needs to be normalized. This is a step bystep process of removing different kinds of redundancy and anomaly at each step. At each step a specific rule is followed to remove specific kind of impurity in order to give the database a slim and clean look.

Un-NormalizedForm(UNF)

If a table contains non-atomic values at each row, it is said to be in UNF. An **atomic value** is something that can not be further decomposed. A **non-atomic value**, as the name suggests, can be further decomposed and simplified. Consider the following table:

Emp-Id	Emp-Name	Month	Sales	Bank-Id	Bank-Name
E01	AA	Jan	1000	B01	SBI
		Feb	1200		
		Mar	850		
E02	BB	Jan	2200	B02	UTI
		Feb	2500		
E03	CC	Jan	1700	B01	SBI
		Feb	1800		
		Mar	1850		
		Apr	1725		

Inthe sample table above, there are multiple occurrences ofrows under eachkeyEmp-Id.Although considered to be the primary key, Emp-Id cannot give us the unique identification facility for any single row. Further, each primarykeypoints to a variable length record (3 for E01, 2 for E02 and 4 for E03).

FirstNormalForm(1NF)

A relation is said to be in 1NF if it contains no non-atomic values and each row can provide a unique combination of values. The above table in UNF can be processed to create the following table in 1NF.

Emp-Id	Emp-Name	Month	Sales	Bank-Id	Bank-Name
E01	AA	Jan	1000	B01	SBI
E01	AA	Feb	1200	B01	SBI
E01	AA	Mar	850	B01	SBI
E02	BB	Jan	2200	B02	UTI
E02	BB	Feb	2500	B02	UTI
E03	CC	Jan	1700	B01	SBI
E03	CC	Feb	1800	B01	SBI
E03	CC	Mar	1850	B01	SBI
E03	CC	Apr	1725	B01	SBI

As you can see now, each row contains unique combination of values. Unlike in UNF, this relation contains only atomic values, i.e. the rows can not be further decomposed, so the relation is now in 1NF.

SecondNormalForm(2NF)

Arelation is said to be in2NF f if it is alreadyin1NFand eachand everyattribute fullydepends on the primary key of the relation. Speaking inversely, if a table has some attributes which is not dependant on the primary key of that table, then it is not in 2NF.

Let us explain. Emp-Id is the primary key of the above relation. Emp-Name, Month, Sales and Bank-Name all depend upon Emp-Id. But the attribute Bank-Name depends on Bank-Id, which is not the primary key of the table. So the table is in 1NF, but not in 2NF. If this position can be removed into another related relation, it would come to 2NF.

Emp-Id	Emp-Name	Month	Sales	Bank-Id
E01	AA	JAN	1000	B01
E01	AA	FEB	1200	B01
E01	AA	MAR	850	B01
E02	BB	JAN	2200	B02
E02	BB	FEB	2500	B02
E03	CC	JAN	1700	B01
E03	CC	FEB	1800	B01
E03	CC	MAR	1850	B01
E03	CC	APR	1726	B01

Bank-Id	Bank-Name
B01	SBI
B02	UTI

After removing the portion into another relation we store lesser amount of data in two relations without any loss information. There is also a significant reduction in redundancy.

ThirdNormalForm(3NF)

Arelation is said to be in 3NF, if it is already in 2NF and there exists no **transitive dependency** in thatrelation. Speaking inversely, ifatablecontainstransitivedependency, thenit isnot in3NF, and the table must be split to bring it into 3NF.

Whatisatransitivedependency?Withinarelationifwesee
A→B[BdependsonA]
And
B→C [CdependsonB]
Thenwemayderive
A→C[C dependsonA]

Suchderiveddependencies holdwellinmostofthesituations. Forexampleifwehave
Roll→Marks
And
Marks→ Grade
Thenwemaysafelyderive
Roll→Grade.

Thisthirddependencywasnotoriginallyspecifiedbutwehavederivedit.

Thederiveddependencyiscalledatransitivedependencywhensuchdependencybecomes improbable. For example we have been given
Roll→City
And
City→STDCode

If we try to derive Roll → STDCode it becomes a transitive dependency, because obviously the STDCode of a city cannot depend on the roll number issued by a school or college. In such a case the relation should be broken into two, each containing one of these two dependencies:
Roll→City
And
City→STDcode

LECTURE-30

Boyce-CodeNormalForm(BCNF)

A relationship is said to be in BCNF if it is already in 3NF and the left hand side of every dependency is a candidate key.Arelation which is in 3NF is almost always in BCNF. These could besamesituationwhena3NFrelationmaynot beinBCNFthe followingconditionsare foundtrue.

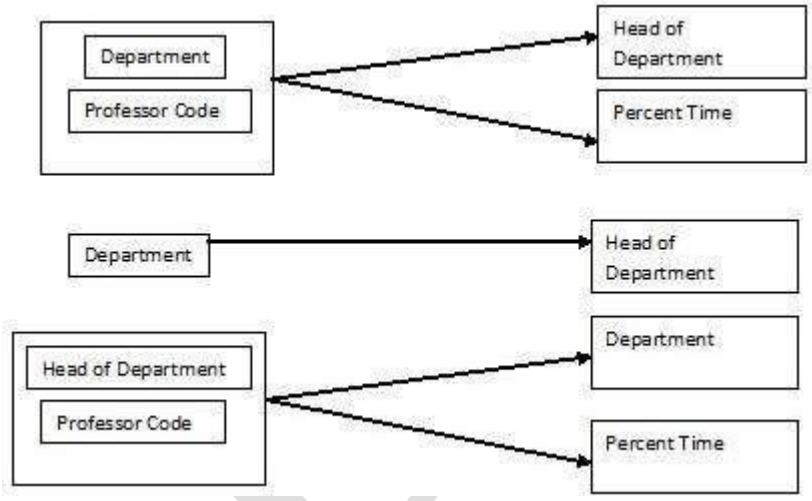
1. Thecandidatekeysarecomposite.
2. Therearemorethanonecandidatekeysintherelation.
3. Therearesomecommonattributesintherelation.

ProfessorCode	Department	Head ofDept.	PercentTime
P1	Physics	Ghosh	50
P1	Mathematics	Krishnan	50
P2	Chemistry	Rao	25
P2	Physics	Ghosh	75
P3	Mathematics	Krishnan	100

Consider,asanexample,theaboverelation.Itisassumedthat:

1. Aprofessor canworkinmorethanone department
2. Thepercentageofthetimehespendsineachdepartmentis given.
3. Eachdepartmenthasonlyone HeadofDepartment.

Therelationdiagramfortheaboverelationisgivenasthefollowing:



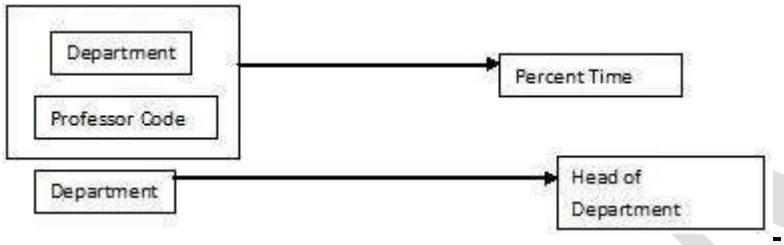
The given relation is in 3NF. Observe, however, that the names of Dept. and Head of Dept. are duplicated. Further, if Professor P2 resigns, rows 3 and 4 are deleted. We lose the information that Rao is the Head of Department of Chemistry.

Thenormalizationoftherelation isdonebycreating anewrelation for Dept. and HeadofDept. and deleting Head of Dept. form the given relation. The normalized relations are shown in thefollowing.

ProfessorCode	Department	PercentTime
P1	Physics	50
P1	Mathematics	50
P2	Chemistry	25
P2	Physics	75
P3	Mathematics	100

Department	Head ofDept.
Physics	Ghosh
Mathematics	Krishnan
Chemistry	Rao

Seethedependencydiagramsforthesenewrelations.



FourthNormalForm (4NF)

Whenattributes ina relation have multi-valued dependency, further Normalizationto 4NFand 5NF are required. Let us first find out what multi-valued dependency is.

A**multi-valueddependency**isatypicalkindofdependencyinwhicheachandeveryattribute within a relation depends upon the other, yet none of them is a unique primarykey.

We will illustrate this with an example. Consider a vendor supplying many items to many projects in an organization. The following are the assumptions:

1. Avendoriscapableofsupplyingmany items.
2. Aproject usesmanyitems.
3. Avendorsuppliestomanyprojects.
4. Anitemmaybesuppliedby manyvendors.

Amultivalueddependencyexistshere becausealltheattributesdependupontheotherand yet none of them is a primary key having unique value.

VendorCode	Item Code	ProjectNo.
V1	I1	P1
V1	I2	P1
V1	I1	P3
V1	I2	P3
V2	I2	P1
V2	I3	P1

V3	I1	P2
V3	I1	P3

Thegivenrelationhasanumber ofproblems.Forexample:

1. IfvendorV1hastosupplyto project P2, buttheitemisnot yet decided, thenarowwitha blank for item code has to be introduced.
2. The informationaboutitemI1isstoredtwicefor vendorV3.

Observe that the relation given is in 3NF and also in BCNF. It still has the problem mentioned above. The problem is reduced by expressing this relation as two relations in the Fourth Normal Form (4NF). Arelation is in 4NF if it has no more than one independent multi valued dependency or one independent multi valued dependency with a functional dependency.

The table can be expressed as the two 4NF relations given as following. The fact that vendors are capable of supplying certain items and that they are assigned to supply for some projects in independently specified in the 4NF relation.

Vendor-Supply

VendorCode	ItemCode
V1	I1
V1	I2
V2	I2
V2	I3
V3	I1

Vendor-Project

VendorCode	ProjectNo.
V1	P1
V1	P3
V2	P1
V3	P2

FifthNormalForm(5NF)

These relations still have a problem. While defining the 4NF we mentioned that all the attributes depend upon each other. While creating the two tables in the 4NF, although we have preserved the dependenciesbetweenVendorCodeandItemcode inthe first tableandVendorCodeandItemcode in thesecond table, wehave lost therelationship betweenItemCodeand Project No. Iftherewerea primary key then this loss of dependency would not have occurred. In order to revive this relationship we must add a new table like the following. Please note that during the entire processof normalization, this is the only step where a new table is created by joining two attributes, rather than splitting them into separate tables.

ProjectNo.	Item Code
P1	11
P1	12
P2	11
P3	11
P3	13

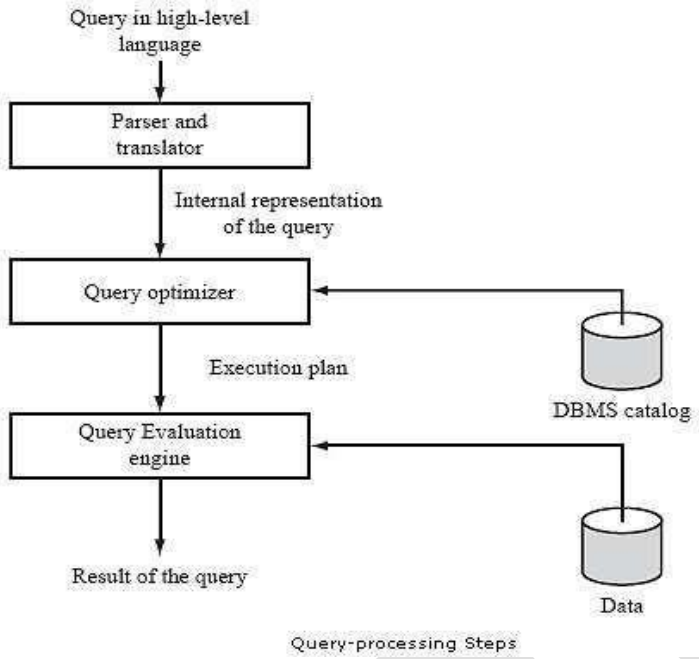
Letusfinallysummarizethenormalizationstepswehavediscussedsofar.

Input Relation	Transformation	Output Relation
All Relations	Eliminatevariablelengthrecord.Removemulti-attributelines intable.	1NF
1NF Relation	Removedependencyofnon-keyattributes onpartofamulti-attribute key.	2NF
2NF	Removedependencyofnon-keyattributesonothernon-keyattributes.	3NF
3NF	Removedependencyofanattributeofamultiattributekeyonan attributeofanother(overlapping)multi-attributekey.	BCNF
BCNF	Removemorethanoneindependentmulti-valueddependencyfrom relationbysplitting relation.	4NF
4NF	Addonerelationrelatingattributeswithmulti-valueddependency.	

LECTURE-31

QUERYPROCESSING

Query processing includes translation ofhigh-level queries into low-level expressions that can be used at the physical level of the file system, query optimization and actual execution of the query to get the result. It is a three-stepprocessthat consistsof parsing and translation,optimizationand executionof the query submitted by the user.



Query-processing Steps

Aquery is processedinfourgeneralsteps:

1. ScanningandParsing
2. QueryOptimizationorplanningtheexecutionstrategy
3. QueryCodeGenerator(interpretedorcompiled)
4. Executionintheruntimedatabaseprocessor

1. ScanningandParsing

When a query is first submitted (via anapplicationsprogram), it must be scanned and parsed to determine if the query consists of appropriate syntax.
Scanningis theprocessofconvertingthequerytext intoatokenizedrepresentation.
The tokenized representation is more compact and is suitable for processing bythe parser.
This representationmay be in a tree form.
The **Parser**checksthetokenizedrepresentationforcorrectsyntax.
Inthis stage, checks are made to determine if columns andtables identified inthe query exist inthe database and if the query has been formed correctly with the appropriatekeywordsand structure.
Ifthe querypassesthe parsingchecks,thenitispassedontothe QueryOptimizer.

2. QueryOptimizationorPlanning theExecutionStrategy

Foranygivenquery,theremaybeanumberofdifferentwaystoexecuteit.
Eachoperationinthequery(SELECT,JOIN,etc.)canbeimplementedusing oneormoredifferent *AccessRoutines.*

For example, anaccess routine that employs anindex to retrieve some rows would be more efficient that an access routine that performs a full table scan.
The goal ofthe **query optimizer** is to finda *reasonably efficient* strategy for executing the query (not quite what the name implies) using the access routines.
Optimizationtypicallytakesoneof twoforms:*HeuristicOptimization*or*CostBasedOptimization*In**Heuristic Optimization**,the query execution is refined based on*heuristic rules* for reordering the individual operations.
With **Cost Based Optimization**,the overall cost of executing the queryis systematically reduced by estimating the costs of executing several different execution plans.

3. QueryCodeGenerator(interpretedorcompiled)

Once the query optimizer has determined the execution plan(the specific ordering of access routines), the code generator writes out the actual access routines to be executed.
With aninteractive session, the query code isinterpreted and passed directly tothe runtime database processor for execution.
Itis alsopossibleto*compile*theaccessroutinesandstorethemforlaterexecution.

4. Executionintheruntimedatabaseprocessor

At this point, the query has been scanned,parsed, planned and (possibly)compiled. The runtime database processor then executes theaccess routines against the database. The results are returned to the applicationthat made the query in the first place.
Anyruntimeerrorsarealsoreturned.

Lecture-32

QueryOptimization

To enable the system toachieve (or improve) acceptable performance bychoosing a better (if not the best) strategy during the processof a query. One of the great strengths to the relationaldatabase.

AutomaticOptimizationvs.HumanProgrammer

1. Agood automatic optimizer will have a wealth of information available toit that human programmers typically do not have.
2. An automatic optimizer can easily reprocess the original relational request when the organization ofthe database is changed. For a human programmer, reorganization would involve rewriting the program.
3. The optimizer is a program, and therefore is capable of considering literally hundreds of different implementation strategies for a given request, which is much more thana human programmer can.
4. The optimizerisavailabletoawiderangeofusers,inanefficientandcost-effectivemanner.

TheOptimizationProcess

1. Castthequeryintosomeinternalrepresentation,suchasaquerytreestructure.
2. Converttheinternalrepresentationtocanonicalform.

*Asubset (sayC) of a set of queries (sayQ) is said to be a set of canonicalformsfor Q if and only if every query Q is equivalent to just one query in C.

During this step, some optimization is already achieved bytransforming the internal representation to a better canonical form.

Possibleimprovements

a. Doingtherestrictions(selects)beforethe join.
b. Reducethe amount of comparisons by converting a restriction condition to an equivalent condition in **conjunctivenormal form-** that is, a condition consistingof a set of restrictions that areANDed together, where each restriction in turn consists of a set of simple comparisons connected only by OR's.
c. Asequenceofrestrictions(selects)beforethejoin.
d. Inasequenceofprojections,all butthelastcanbeignored.
e. Arestrictionofprojectionisequivalenttoaprojectionofarestriction.
f. Others

3. Choosecandidatelow-levelproceduresbyevaluatethe transformedquery.

*Access path selection: Consider the queryexpression as a series of basic operations (join, restriction, etc.), then the optimizer choose from a set of pre-defined, low-level implementation procedures. These procedures may involve the user of primary key, foreign key or indexes and other information about the database.

4. Generate query plans and choose the cheapest by constructing a set of candidate query plans first, then choose the best plan. To pick the best plan can be achieved by assigning cost to each given plan. The costs is computed according to the number of disk I/O's involved.

Module-
3:Lecture-33

Transaction

Atransactionisthesmallest unitofoperationdoneonadatabase. It can

be basically of two types:

→Read Transaction

→WriteTransaction

3.2.ACIDPropertiesoftransaction:

Atomicity:(allornothing)

Atransaction is said to be atomic if a transaction always executes all its actions in one step or not executes anyactions at allIt means either allor none of the transactions operations are performed.

Consistency:(Noviolationofintegrityconstraints)

Atransaction must preserve the *consistency* of a database after the execution. The DBMS assumes that this property holds for each transaction. Ensuring this property of a transaction is the responsibility of the user.

Isolation:(concurrentchangesinvisibles)

The transactions must behave as if they are executed in isolation. It means that if several transactions are executed concurrentlythe results must be same as if they were executed serially in some order. The data used during the execution of a transaction cannot be used by a second transaction until the first one is completed.

Durability:(committedupdatepersist)

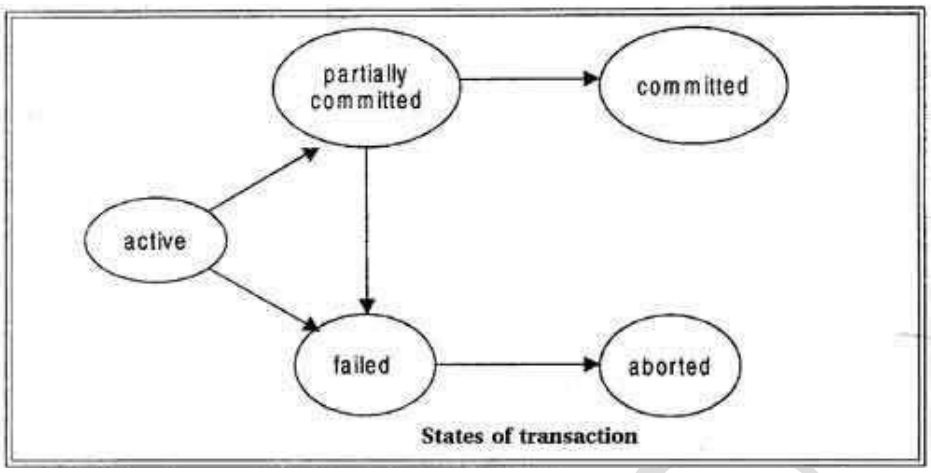
Theeffectofcompletedor committedtransactions shouldpersist evenafter acrash. It meansoncea transactioncommits, thesystemmust guaranteethattheresult ofitsoperationswillnever be lost, in spite of subsequent failures.

3.3. Statesofatransaction:

Atransactionmustbeinoneofthefollowingstates:

- **Active**:theinitialstate,thetransactionstays inthis statewhileitisexecuting.
- **Partiallycommitted**:afterthefinalstatementhas beenexecuted.
- **Failed**:whenthenormalexecutioncannolongerproceed.
- **Aborted**: after the transaction has beenrolled back and the database has beenrestoredto its state prior to the start of the transaction.
- **Committed**:aftersuccessfulcompletion.

Thestatediagramcorrespondingtoatransactionisshowninfigure.



We saythat atransactionhascommittedonlyifit hasenteredthecommittedstate.Similarly,wesay that a transaction has aborted only if it has entered the aborted state. Atransaction is said to have terminated if has either committed or aborted.

A transaction starts in the active state. When it finishes its final statement, it enters the partially committedstate.Atthispoint,thetransactionhas completed itsexecution, but it is stillpossiblethat it may have to be aborted, since the actual output may still be temporarily hiding in main memory and thus a hardware failure may preclude its successful completion

The database systemthen writes out enough information to disk that, even in the event of a failure, the updates performed by the transaction can be recreated when the systemrestarts after the failure. When the last of this information is written out, the transaction enters the committed state.

Problemsdue toconcurrency:

1) Lostupdateproblem

2) DirtyRead problem

3) Incorrectsummaryproblem

1. The lost updateproblem:Asecondtransactionwritesasecondvalueofadata-item(datum) ontop ofa first value written bya first concurrent transaction, and the first value is lost to other transactions running concurrently which need, by their precedence, to read the first value. The transactions that have read the wrong value end with incorrect results.
2. Thedirtyreadproblem:Transactionsreadavalue writtenbyatransactionthat hasbeenlater aborted. This value disappears fromthe database upon abort, and should not have been read by any transaction ("dirty read"). The reading transactions end with incorrect results.
3. The incorrect summaryproblem: While one transaction takes a summaryover the values of allthe instances ofa repeated data-item, a second transactionupdates some instances ofthat data-item. The resulting summarydoes not reflect a correct result for any(usually neededforcorrectness)precedenceorderbetweenthetwotransactions(ifone isexecutedbeforethe other), but rather some random result, depending on the timing of the updates, and whether certain update results have been included in the summary or not.

ConcurrencyControlTechniques:

1) Locking

2) Timestampordering

3) Multi-versionconcurrencycontrol

4) Optimisticconcurrencycontrol

Locking(2-PhaseLocking)

2PLprotocollocksareapplied andremovedintwophases:

1. Expandingphase:locksareacquiredandnolocksarereleased.
2. Shrinkingphase:locksarereleasedandnolocksareacquired.

Locks areoftwotypes:

1. BinaryLock
2. Share/Exclusive(Read/Write)Lock

→**BinaryLock**

Abinarylockcanhave 2Statesorvalues

- Locked(or1)and
- Unlocked (or0)

Werepresentthecurrentstate(orvalue)ofthelockassociatedwithdataitemXasLOCK(X).

OperationsusedwithBinary Locking

1. **lock_item:** Atransactionrequest accesstoanitembyfirst issuinga lock_item(X) operation.
    - **IfLOCK(X)=1 orL(X):** thetransactionisforcedtowait.
    - **IfLOCK(X)=0orU(x):** it issetto 1(thetransaction**locks**the item)andthe transaction is a load to access item X.
2. **unlock_item:** Afterusingthedataitemthetransactionissuesanoperation **unlock(X),** which sets the operation **LOCK(X) to 0** i.e. unlocks the data item so that X may be accessed by another transactions.

TransactionRulesforBinaryLocks

Everytransactionmustobeythefollowingrules:

→AtransactionT mustissuethelock(X)operation beforeanyread(X)orwrite(X)operationsinT.
→AtransactionT mustissuetheunlock(X)operationafterallread(X)andwrite(X)operations inT.
→IfatransactionTalreadyholdsthelock onitemX,thenTwillnotissuealock(X)operation.

→Ifa transactiondoesnotholdsthelockon itemX,thenTwillnotissueanunlock(X)operation.

```
lock_item (X):
  B: if LOCK (X)=0  (* item is unlocked *)
     then LOCK (X)←1  (* lock the item *)
     else begin
       wait  (until lock (X)=0  and
         the lock manager wakes up the transaction);
       go to B
       end;
```

```
unlock_item (X):
  LOCK (X)←0;  (* unlock the item *)
  if any transactions are waiting
   then wakeup one of the waiting transactions;
```

→Shared/ReadandExclusive/WriteLock

The binarylock is too restrictive for data items because at most onetransactioncanhold ona given itemwhether the transaction is reading or writing. To improve it we have shared and exclusivelocks inwhichmorethanonetransactioncanaccessthesame itemfor readingpurposes.i.e.the read operations on the same item by different transactions are not conflicting.

Inthistypesoflock,systemsupportstwokindsoflock:

- Exclusive(orWrite)Locks and
- Shared(or Read)Locks.

SharedLocks

IfatransactionTihas lockedthedataitemAinsharedmode,thenarequest fromanother transaction Tj on A for :

- **WriteoperationonA:**Denied.TransactionTjhastowaituntiltransactionTiunlockA.
- **Read operationonA:**Allowed.

ExclusiveLocks

IfatransactionTihas lockedadataitema inexclusive modethenrequest fromsomeanother transaction Tj for

- **Read operationonA:** Denied
- **Writeoperation onA:**Denied

OperationsUsedwithSharedandExclusiveLocks

1. Read_lock(A)or s(A)
2. Write_lock(A)orX(A)
3. Unlock(X)orU(A)

ImplementationofSharedandExclusiveLocks

Sharedandexclusive locksareimplementedusing4fields:

1. Data_item_name
2. LOCK
3. NumberofRecords and
4. Locking_transaction(s)

Againtosavespace, itemsthat arenot inthe locktableareconsideredto beunlocked.Thesystem maintains onlythose records for the items that are currently locked in the lock table.

ValueofLOCK(A):ReadLockedorWriteLocked

- **IfLOCK(A)=write-locked–**The valueoflockingtransactionisasingletransactionthat holds the exclusive(write) Lock on A.
- **IfLOCK(A)=read-locked –**Thevalueoflockingtransactionisa listofoneormore transactions that hold the Shared(read) onA.

TransactionRulesforSharedandExclusiveLocks

Everytransactionmustobeythefollowingrules:

1. AtransactionT mustissuetheoperation **s(A)orread_lock(A)orx(A)orwrite_lock(A)** beforeanyread(A)operationisperformedinT.
2. AtransactionTmust issuetheoperationx(A)orwrite_lock(A)beforeanywrite(A) operation is performed in T.
3. Aftercompletionofallread(A)and write(A)operationsinT,atransactionTmust issuean unlock(A) operation.
4. Ifatransactionalreadyholdsaread(shared) lockorawrite(exclusive) lockonitemA, then T will not issue an unlock(A) operation.
5. Atransactionthat already holdsa lockonitemA, isallowedtoconvertthe lockfromone locked state to another under certain conditions.
    - **UpgradingtheLockbyIssuingawrite_lock(A)OperationorConversionof read_lock() to write_lock() :**
        - **Case1–WhenConversionNotPossible:**AtransactionTwillnot issuea write_lock(A) operation if it already holds a read (shared) lock or write (exclusive) lock on item A.
        - **Case2–WhenConversionPossible:** IfTistheonlytransactionholdinga read lock onAat the time it issues the write_lock(A)operation,thelockcanbeupgraded;
    - **DowngradingtheLockbyIssuingaread_lock(A)orConversionofwrite_lock() to read_lock() :**
      AtransactionTdowngradefromthewrite locktoareadlockbyacquiringthe write_lock(A) or x(A), then the read_lock(A) or s(A) and then releasing the write_lock(A) or x(A).

read_lock (X):

```
B: if LOCK (X)='unlocked"
   then begin LOCK (X)←  "read-locked";
              no_of_reads(X)←  1
        end
   else if LOCK(X)='read-locked"
        then no_of_reads(X)←  no_of_reads(X) + 1
        else begin wait (until LOCK (X)="unlocked' and
                    the lock manager wakes up the transaction);
             go to B
             end;
```

write_lock (X):

```
B: if LOCK (X)='unlocked"
   then LOCK (X)←  "write-locked"
   else begin
        wait (until LOCK(X)="unlocked" and
         the lock manager wakes up the transaction);
        go to B
        end;
```

unlock_item (X):

```
if LOCK (X)="write-locked"
   then begin LOCK (X)←  "unlocked;"
        wakeup one of the waiting transactions, if any
        end
   else if LOCK(X)='read-locked"
        then begin
             no_of_reads(X)←  no_of_reads(X) – 1;
             if no_of_reads(X)=0
               then begin LOCK (X)="unlocked";
                    wakeup one of the waiting transactions, if any
                    end
             end;
```

Lecture-34

Problemsduetolocking

1) deadlock
2) starvation

Deadlock:

Itisanindefinitewait situationinwhichaseriesoftransactionswait foreachotherforunknown amount of time, it obeys the following conditions:

→Hold&Wait

→NoPreemption

→MutualExclusion

→CircularWait

Example:

For example, assume a set oftransactions {T_0, T_1, T_2, ...,T_n}. T_0needs a resource X to complete its task.ResourceX isheld byT_1, andT_1iswaiting foraresourceY,which isheld byT_2.T_2iswaiting for resource Z, which is held byT_0. Thus, allthe processes wait for each other to release resources. In this situation, none of the processes can finish their task. This situation is known as a deadlock.

Starvation:

Itisasituationinwhichatransactionhas lockedadatabase forunfair means&allother transactions are in indefinite waiting.

Starvation	**Deadlock**
Starvationhappensifsametransactionis always chosen as victim.	Adeadlock is a condition in which two or more transaction is waiting for each other.
It occursif the waiting schemeforlocked itemsinunfair,givingpriorityto some transactionsoverothers.	A situationwheretwoormoretransactionsareunableto proceedbecauseeachiswaitingforoneoftheothertodo something.
Starvationisalso knownaslivedlock.	Deadlockisalsoknownascircularwaiting.
Avoidance: ->switch priorities so that every thread has a chance to have high priority. ->UseFIFOorderamongcompeting request.	Avoidance: ->Acquire locksarepredefinedorder. ->Acquirelocksatoncebeforestarting.
Itmeansthattransactiongoesinastate where transaction never progress.	Itisasituationwheretransactionsarewaitingforeach other.

TimestampOrdering

Thetimestamp-orderingprotocolensuresserializabilityamongtransactionsintheirconflictingread and write operations. This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.

- ThetimestampoftransactionT$_i$isdenoted asTS(T$_i$).
- Readtime-stampofdata-itemXisdenotedbyR-timestamp(X).
- Writetime-stampofdata-itemXisdenotedbyW-timestamp(X).

Timestamporderingprotocolworksasfollows−

- **IfatransactionTiissuesaread(X)operation−**
  - IfTS(Ti)<W-timestamp(X)
    - Operationrejected.
  - IfTS(Ti)>=W-timestamp(X)
    - Operationexecuted.
  - Alldata-itemtimestamps updated.
- **IfatransactionTiissuesawrite(X)operation−**
  - IfTS(Ti)<R-timestamp(X)
    - Operationrejected.
  - IfTS(Ti)<W-timestamp(X)
    - OperationrejectedandTirolledback.
  - Otherwise,operationexecuted.

Avoidingdeadlock:

Amajordisadvantageoflocking isdeadlockwhichcanbeavoidedusingtimestamporderingas follows:

Therearetwoalgorithms fordeadlockavoidance.

- Wait/Die
- Wound/Wait

Here isthetablerepresentationofresourceallocationforeachalgorithm. Bothofthesealgorithms takeprocessage intoconsiderationwhiledeterminingthebest possiblewayofresourceallocation for deadlock avoidance.

	Wait/Die	Wound/Wait
Olderprocessneedsaresourceheldbyyounger process	**Olderprocess**waits	**Youngerprocess**dies
Youngerprocessneedsaresourceheldbyolder process	**Youngerprocess** dies	**Youngerprocess** waits

Wait-DieScheme

Inthisscheme, ifatransactionrequeststolockaresource(dataitem), whichisalreadyheldwitha conflicting lock by another transaction, then one of the two possibilities may occur −

- If $TS(T_i) < TS(T_j)$ – that is T_i, which is requesting a conflicting lock, is older than T_j – then T_i is allowed to wait until the data-item is available.
- If $TS(T_i) > TS(t_j)$ – that is T_i is younger than T_j – then T_i dies. T_i is restarted later with a random delay but with the same timestamp.

This scheme allows the older transaction to wait but kills the younger one.

Wound-Wait Scheme

In this scheme, if a transaction requests to lock a resource (data item), which is already held with conflicting lock by some another transaction, one of the two possibilities may occur −

- If $TS(T_i) < TS(T_j)$, then T_i forces T_j to be rolled back – that is T_i wounds T_j. T_j is restarted later with a random delay but with the same timestamp.
- If $TS(T_i) > TS(T_j)$, then T_i is forced to wait until the resource is available.

This scheme, allows the younger transaction to wait; but when an older transaction requests an item held by a younger one, the older transaction forces the younger one to abort and release the item.

In both the cases, the transaction that enters the system at a later stage is aborted.

Multi-version Concurrency Control Techniques:

This concurrency control technique keeps the old values of a data item when the item is updated. These are known as multiversion concurrency control, because several versions (values) of an item are maintained.

→When a transaction requires access to an item, an *appropriate* version is chosen to maintain the serializability of the currently executing schedule, if possible. The idea is that some read operations that would be rejected in other techniques can still be accepted by reading an *older version* of the item to maintain serializability. When a transaction writes an item, it writes a *new version* and the old version of the item is retained. Some multiversion concurrency control algorithms use the concept of view serializability rather than conflict serializability.

→An obvious drawback of multiversion techniques is that more storage is needed to maintain multiple versions of the database items. However, older versions may have to be maintained anyway—for example, for recovery purposes. In addition, some database applications require older versions to be kept to maintain a history of the evolution of data item values.

→The extreme case is a *temporal database*, which keeps track of all changes and the times at which they occurred. In such cases, there is no additional storage penalty for multiversion techniques, since older versions are already maintained.

Lecture-35

MultiversionTechniqueBasedonTimestampOrdering

Inthismethod,severalversions,,...,ofeachdataitem*X*aremaintained.For*eachversion,*the value of version and the following two timestamps are kept:

1. read_TS:Thereadtimestampofisthelargestofallthetimestampsoftransactionsthathave successfully read version .
2. write_TS:Thewritetimestampofisthetimestampofthetransactionthatwrotethevalueof version

Whenever a transaction T is allowed to execute a write_item(*X*) operation, a new version of item *X* is created, with both the write_TS and the read_TS set to TS(T). Correspondingly, when a transactionT is allowed to readthe value ofversion *Xi*,the value ofread_TS() is set tothe larger of the current read_TS() and TS(T).

Toensureserializability, thefollowingtworulesare used:

1. IftransactionT issues a write_item(*X*)operation, and version *i*of*X* hasthe highest write_TS()of allversions of*X* that is also *less than or equalto*TS(T), and read_TS() >TS(T),thenabort and roll back transaction T; otherwise, create a new version of *X* with read_TS() = write_TS() = TS(T).

2. If transaction T issues a read_item(*X*) operation, find the version *i* of *X* that has the highest write_TS() of all versions of *X* that is also *less than or equal to* TS(T); then return the value of to transaction T, and set the value of read_TS() to the larger of TS(T) and the current read_TS().

As we can see in case 2, a read_item(*X*) is always successful, since itfinds the appropriate version to read based on the write_TS of the various existing versions of *X*. In case 1, however, transaction T may be aborted and rolled back. This happens if T is attempting to write a version of *X* thatshould have been read by another transaction T whose timestamp is read_TS(); however, T has already readversion Xi, which was written by the transaction with timestamp equal towrite_TS(). If this conflict occurs, T is rolled back; otherwise, a new version of *X*, written by transaction T, is created. Notice that, if T is rolled back, cascading rollback may occur. Hence, to ensure recoverability, a transaction T should not be allowed to commit until after all the transactions that have written some version that T has read have committed.

MultiversionTwo-PhaseLockingUsingCertifyLocks

In this multiple-mode locking scheme, there are *three locking modes* for an item: read, write, and certify, instead of just the two modes (read, write). Hence, the state of LOCK(*X*) for an item *X* can be one of read-locked, write-locked, certify-locked, or unlocked.

In the standard locking scheme, once a transaction obtains a write lock on an item, no other transactions canaccess that item. The idea behind multiversion2PL is to allow other transactions T to read an item *X* while a single transaction T holds a write lock on *X*. This is accomplished by allowing *two versions* for each item *X*; one version must always have been written by some committed transaction. The second version *X* is created when a transaction T acquires a write lock on the item. Other transactions can continue to read the *committed version* of *X* while T holds the writelock.TransactionTcanwritethevalueof*X*asneeded,withoutaffectingthevalueofthe

committed version*X*.However,once Tisreadyto commit, it must obtaina certify lock onallitems that it currently holds write locks on before it can commit. The certify lock is not compatible with read locks, so the transaction may have to delay its commit until all its write-locked items are released by any reading transactions in order to obtain the certify locks. Once the certify locks— which are exclusive locks—are acquired, the committed version *X* of the data item is set to thevalue of version *X*, version *X* is discarded, and the certify locks are then released. In this multiversion 2PL scheme, reads can proceed concurrently with a single write operation—an arrangement not permitted under the standard 2PL schemes.

3. 6.4.OptimisticConcurrencyControlTechniques:

Basic idea:alltransactionsconsistofthree phases:

1. Read.Here,allwritesaretoprivatestorage(shadowcopies).
2. Validation.Makesurenoconflictshaveoccurred.
3. Write.IfValidationwassuccessful, makewritespublic.(Ifnot,abort!)

Useful in the following cases:

1. Alltransactionsarereaders.
2. Lotsoftransactions, eachaccessing/modifyingonlyasmallamountofdata, largetotal amount of data.
3. Fractionoftransactionexecutioninwhichconflicts"reallytakeplace" issmallcomparedto total pathlength.

TheValidationPhase

- Goal:toguaranteethatonlyserializableschedulesresult.
- Technique:actuallyfindanequivalent serializableschedule.Inparticular,

1. AssigneachtransactionaTNduringexecution.
2. Ensurethat ifyouruntransactions inorderinducedby"<"onTNs, yougetanequivalent serial schedule.

SupposeTN(Ti)<TN(Tj).Thenifoneofthefollowingthreeconditionsholds,it’sserializable:

1. TicompletesitswritephasebeforeTjstartsitsread phase.
2. WS(Ti) intersect RS(Tj)=*emptyset*andTicompletesitswritephasebeforeTjstarts its write phase.
3. WS(Ti)intersectRS(Tj)=WS(Ti)*intersect*WS(Tj)=*emptyset*andTicompletesits read phase before Tj completes its read phase.

Recovery

TypesofFailure

Failures maybe

Transaction	Causedbyerrorswithinthetransactionprocesses.
System	Caused byfailureofnetworkoroperatingsystemorphysicalthreatstothe system as a whole.
Media	Failureofhard disk,outofmemoryerrors,outofdiskspace errors.

ReasonsforFailure

Failuremaybecausedbyanumber ofthings.

ASystemCrash	Ahardware,softwareornetworkerrorcausesthetransactionto fail.
TransactionorSystem error	Someoperationinthetransactionmaycausethe failureortheusermay interrupt the transaction.
LocalErrorsor Exceptions	Conditionsoccurduringthetransactionthat resultsintransaction cancellation.
ConcurrencyControl Enforcement	Severaltransactionsmaybe indeadlocksothetransactionmaybeaborted to be restarted later.
DiskFailure	ReadWriteerroronthephysicaldisk.
PhysicalProblems	Thiscanbeanyrangeofphysicalproblems, suchaspowerfailure, mounting wrong disk or tape by operator, wiring problems etc
CatastropheSituations	Largescalethreatstothesystemandthedataforexample fire,cyclone, security breaches etc.

Transaction errors, system errors, system crashes, concurrency problems and local errors or exceptions are the more common causes of system failure.The system mustbe able to recoverfrom such failures without loss of data.

Log-BasedRecovery

COMMIT
Signalsthe successfulendofatransaction
• Anychangesmadebythetransactionshouldbe saved
• Thesechangesarenowvisibletoothertransactions

ROLLBACK
Signalsthe unsuccessfulendofatransaction
• Anychangesmadebythetransactionshouldbe undone
• Itisnowas ifthetransactionnever existed

Log-BasedRecovery

The most widely used structure for recording database modifications is the *log*. The log is a sequence of *log records* and maintains a history of all update activities in the database. There are several types of log records.

An*updatelogrecord*describesasingledatabasewrite:

- **Transactionsidentifier.**
- **Data-item identifier.**
- **Old value.**
- **Newvalue.**

Other special log records exist to record significant events during transaction processing, such asthe start of a transaction and the commit or abort of a transaction. We denote the various types of log records as:

- <Tistart>.TransactionTihasstarted.
- <Ti,Xj, V1, V2> Transaction Tihas performeda write on data item Xj.Xj hadvalue V1 before write, and will have value V2 after the write.
- <Ticommit>TransactionTihascommitted.
- <Tiabort>TransactionTihasaborted.

Whenever a transaction performs a write, it is essential that the log record for that write be created before the database is modified. Once a log record exists, we can output the modification that has alreadybeenoutputtothedatabase.Also wehavetheabilityto *undo*amodificationthat hasalready been output to the database, by using the old-value field in the log records.

For log records to be useful for recovery from system and disk failures, the log must reside onstable storage. However, since the log contains a complete record of all database activity, the volume of data stored in the log may become unreasonable large.

DeferredDatabaseModification

The deferred-modification technique ensures transaction atomicity by recording all database modifications inthe log, but deferringallwriteoperationsofatransactionuntilthetransaction partially commits (i.e., once the final action of the transaction has been executed). Then the information in the logs is used to execute the deferred writes. Ifthe systemcrashes or ifthe transaction aborts, then the information in the logs is ignored.

LetT0betransactionthattransfers$50 fromaccountAto account B: T0:

```
read (A);
A:=A-50;
Write(A);
Read(B);
B:=B+50;
Write(B).
```

ImmediateDatabaseModification

The immediate-update technique allows database modifications to be output to the database while thetransactionisstillintheactivestate.These modificationsarecalled *uncommittedmodifications*. In the event of a crash or transaction failure, the system must use the old-value field of the log records to restore the modified data items.

Transactions T0 and T1 executed one after the other in the order T0 followed byT1. The portionofthe logcontainingtherelevant informationconcerningthesetwotransactionsappearsin the following,

PortionofthesystemlogcorrespondingtoT0andT1

```
<T0start>
        <T0,A,1000,950>
        <T0,B,2000,2050 >
<T0commit>
<T1 start >
<T1,C,700,600>
<T0commit>
```

Checkpoints

When a system failure occurs, we must consult the log to determine those transactions that need to be redone and those that need to be undone. Rather than reprocessing the entire log, which is time-consuming and much of it unnecessary, we can use *checkpoints*:

- Outputontostablestorageallthelogrecordscurrentlyresidinginmainmemory.
- Outputtothediskallmodified buffer blocks.
- Outputontostablestoragealog record,**<checkpoint>**.

Lecture-36

Serializability:

When several concurrent transactions are trying to access the same data item, the instructions within these concurrent transactions must be ordered in some way so as there are no problem in accessing and releasing the shared data item. There are two aspects of serializability which are described here:

ConflictSerializability

Two instructions oftwo different transactions may want to accessthe same data item in orderto perform a read/write operation. Conflict Serializability deals with detecting whether the instructions are conflicting in any way, and specifying the order in which these two instructions will be executed in case there is any conflict. A **conflict** arises if at least one (or both) of the instructions is a write operation. The following rules are important in Conflict Serializability:

1. If two instructions of the two concurrent transactions are both for read operation, then theyare not in conflict, and can be allowed to take place in anyorder.
2. If one of the instructions wants to perform a read operation and the other instruction wants to perform a write operation, then they are in conflict, hence their ordering is important. Ifthe read instruction is performed first, then it reads the old value ofthe data item and after the reading is over, the new value of the data item is written. It the write instruction is performed first, then updates the data itemwith the new value and the read instruction reads the newly updated value.
3. If both the transactions are for write operation, then they are in conflict but can be allowed to take place in anyorder, because the transaction do not read the value updated by each other. However, the value that persists in the data itemafter the schedule is over is the one written by the instruction that performed the last write.

It mayhappenthat wemaywant to executethe same set oftransaction ina different schedule on another day. Keeping in mind these rules, we may sometimes alter partsofone schedule (S1) to create another schedule (S2) by swapping only the non-conflicting parts of the first schedule.The conflicting parts cannot be swapped in this way because the ordering of the conflicting instructions is important and cannot be changed in any other schedule that is derived from the first. Ifthese two schedules are made ofthe same set oftransactions, then both S1 and S2 would yield the same result if the conflict resolution rules are maintained while creating the new schedule. In that case the schedule S1 and S2 would be called **Conflict Equivalent**.

ViewSerializability:

This is another type of serializability that can be derived by creating another schedule out of an existing schedule, involving the same set of transactions. These two schedules would be called View Serializable if the following rules are followed while creating the second schedule out of the first. Let us consider that the transactions T1 and T2 are being serialized to create two different schedules S1 and S2 which we want to be **View Equivalent** and bothT1 and T2 wants to access the same data item.

1. If inS1, T1reads theinitial valueof thedataitem,thenin S2also, T1shouldreadthe initial value of that same data item.
2. IfinS1,T1writesavalueinthedataitemwhichisreadbyT2,theninS2also,T1 should write the value in the data item before T2 reads it.
3. IfinS1,T1performsthefinalwriteoperationonthatdataitem,theninS2also,T1 should perform the final write operation on that data item.

Exceptinthesethreecases,anyalterationcanbepossiblewhilecreatingS2bymodifying S1.

Lecture-37

ObjectOriented Databases

Object oriented databases are also called Object Database Management Systems (ODBMS). Object databases storeobjects ratherthandata suchas integers, strings orrealnumbers. Objects are used in object oriented languages such as Smalltalk, C++, Java, and others. Objectsbasically consist of the following:

- Attributes - Attributes are data which defines the characteristics of an object. This data may be simple such as integers, strings, and real numbers or it may be a reference to a complex object.
- Methods - Methods define the behavior of an object and are what was formally called procedures or functions.

Thereforeobjectscontainbothexecutablecodeanddata

ObjectPersistence

With traditional databases, data manipulated by the application is transient and data in the database is persisted (Stored on a permanent storage device). In object databases, theapplication can manipulate both transient and persisted data.

WhentoUseObjectDatabases

Object databases should be used whenthere is complexdata and/or complexdata relationships. This includes a many to many object relationship. Object databases should not be used when there would be few join tables and there are large volumes of simple transactional data.

Objectdatabasesworkwellwith:

- CAS Applications (CASE-computer aided software engineering,CAD-computer aided design, CAM-computer aided manufacture)
- MultimediaApplications
- Objectprojectsthatchangeovertime.
- Commerce

ObjectDatabaseAdvantagesoverRDBMS

- Objectsdon't requireassemblyanddisassemblysavingcodingtimeandexecutiontimeto assemble or disassemble objects.
- Reducedpaging
- Easiernavigation
- Better concurrencycontrol-Ahierarchyofobjectsmaybelocked.
- Datamodelisbasedonthe realworld.
- Workswellfordistributedarchitectures.
- Lesscoderequiredwhenapplicationsareobjectoriented.

ObjectDatabaseDisadvantagescomparedtoRDBMS

- Lowerefficiencywhendataissimpleandrelationshipsare simple.
- Relationaltables aresimpler.
- Latebindingmayslowaccessspeed.
- MoreusertoolsexistforRDBMS.
- StandardsforRDBMSaremorestable.

- Support forRDBMSis morecertainandchangeisless likelytoberequired.

HowDataisStored

Twobasicmethodsareusedtostoreobjectsbydifferentdatabasevendors.

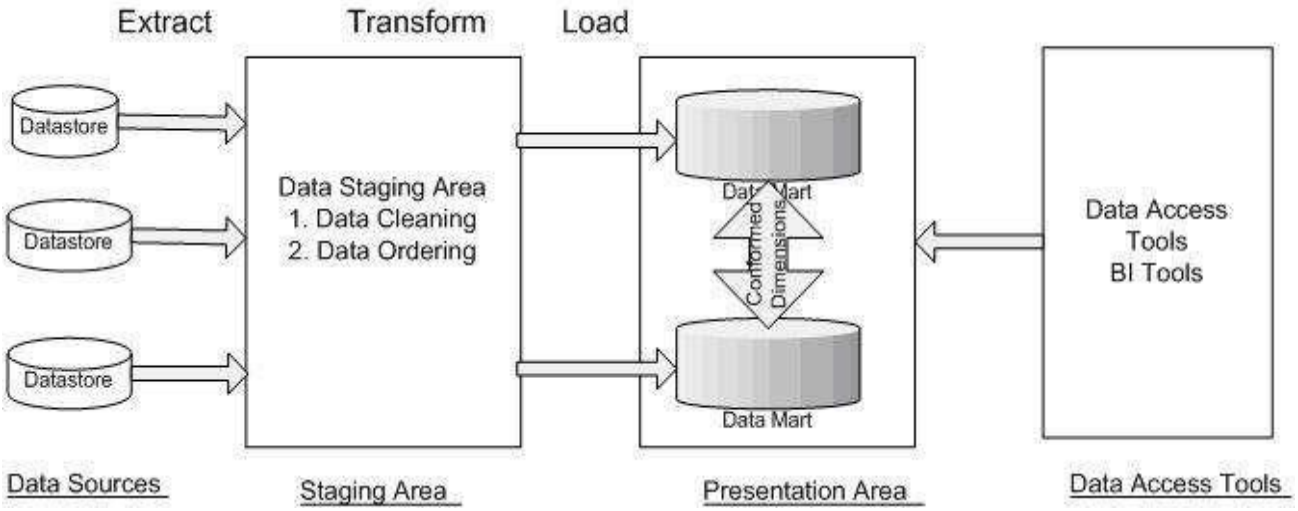
- Eachobject hasauniqueIDand isdefinedasasubclassofabaseclass, using inheritanceto determine attributes.
- Virtualmemorymapping isused forobjectstorageand management.

DataWarehouse

- **A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process.**
- **Subject-Oriented**: A data warehouse can be used to analyze a particular subject area. For example, "sales" can be a particular subject.
- **Integrated**: Adata warehouse integrates data from multiple data sources. For example, source A and source B may have different ways of identifying a product, but in a data warehouse, there will be only a single way of identifying a product.
- **Time-Variant**: Historical data is kept in a data warehouse. For example, one canretrieve data from 3 months, 6 months, 12 months, or even older data from a data warehouse. This contrasts with a transactions system, where often only the most recent data is kept. For example, a transaction system may hold the most recent address of a customer, where a data warehouse can hold all addresses associated with a customer.
- **Non-volatile**: Once data is in the data warehouse, it will not change. So, historical datain a data warehouse should never be altered.

DataWarehousingArchitecture&Components

Followingdiagramdepictsdifferent componentsofDataWarehousearchitecture.



OperationalSourceSystem

It'sthetraditionalOLTPsystemswhichstorestransactiondataoftheorganizationsbusiness.Its generally used one record at any time not necessarily stores history of the organizations information's.Operationalsourcesystemsgenerallynot usedforreporting likedatawarehouse.

DataStagingArea

Datastagingarea isthestorageareaaswellassetofETLprocessthatextract datafromsource system. It is everything between source systems and Data warehouse.

Datastagingareneverbeused forreportingpurpose.Dataisextractedfromsourcesystemand stored, cleansed, transformed in staging area to load into data warehouse.

StagingarenotnecessarilytheDBMS.Itcould beflat filesalso.Stagingareacanbe structuredlike normalized source systems. It totally depends on choice and need of development process.

DataPresentationArea

Data presentation area is generally called as data warehouse. It's the place where cleaned, transformeddataisstoredinadimensionallystructuredwarehouseand madeavailable foranalysis purpose

DataAccessTools

oncedataisavailable inpresentationarea it isaccessedusingdataaccesstoolslikeBusiness Objects.

Schema:
(Starschema)

The star schema architecture is the simplest data warehouse schema. It is called a star schema because the diagram resembles a star, with points radiating froma center. The center ofthe star consists of fact table and the points ofthe star arethe dimensiontables. Usuallythe fact tables in a star schema are inthird normalform(3NF) whereas dimensionaltables are de-normalized. Despite thefactthatthestarschema isthesimplest architecture,it ismost commonlyusednowadaysand is recommended by Oracle.

→FactTables

Afact table typically has two types of columns: foreign keys to dimension tables and measures thosethatcontainnumeric facts.Afacttablecancontainfact'sdataondetailoraggregatedlevel.

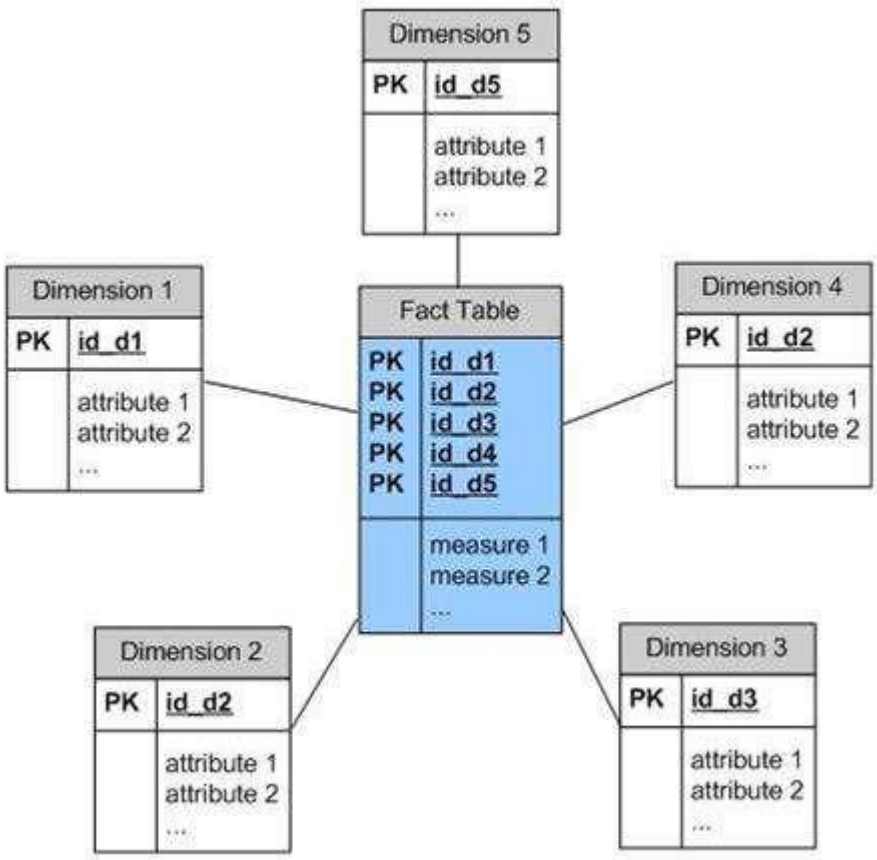
→DimensionTables

Adimension is a structure usually composed of one or more hierarchies that categorizes data. If a dimensionhasn't gotahierarchiesand levelsit iscalled **flatdimensionorlist**.Theprimarykeysof each of the dimension tables are part of the composite primary keyof the fact table. Dimensional attributes help to describe the dimensional value. They are normally descriptive, textual values. Dimensiontablesaregenerallysmallin sizethen fact table.

Typicalfacttablesstoredataaboutsaleswhiledimensiontablesdataaboutgeographic region(markets, cities) , clients, products, times, channels.

Themaincharacteristicsofstar schema:

1. Simplestructure->easytounderstandschema
2. Greatqueryeffectives ->smallnumber oftablesto join
3. Relativelylongtimeofloadingdataintodimensiontables ->de-normalization, redundancy data caused that size of the table could be large.
4. Themostcommonly usedinthedatawarehouseimplementations->widely supported by a large number of business intelligence tools



DataMining:

→Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from manydifferentdimensionsorangles,categorizeit,andsummarizetherelationshipsidentified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

→While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries.Severaltypesofanalyticalsoftwareareavailable:statistical, machine learning,and neural networks. Generally, any of four types of relationships are sought:

- **Classes**: Stored data is used to locate data in predetermined groups. For example, a restaurant chaincould mine customer purchase data to determine whencustomersvisit and what theytypicallyorder.Thisinformationcouldbeusedtoincreasetrafficbyhavingdaily specials.
- **Clusters**: Data items are grouped according to logical relationships or consumer preferences.Forexample,datacanbeminedto identifymarket segmentsorconsumer affinities.
- **Associations**:Datacanbe minedto identifyassociations.The beer-diaperexample isan example of associative mining.

- **Sequentialpatterns**:Dataisminedtoanticipatebehaviorpatternsandtrends.Forexample, an outdoor equipment retailer could predict the likelihood of a backpack being purchased based on a consumer's purchase of sleeping bags and hiking shoes.

Dataminingconsistsoffive majorelements:

- Extract,transform,andloadtransactiondataontothedatawarehouse system.
- Storeandmanagethedatainamultidimensionaldatabasesystem.
- Providedataaccesstobusinessanalystsandinformationtechnologyprofessionals.
- Analyze thedatabyapplicationsoftware.
- Presentthedatainauseful format,suchasagraphor table.

Techniquesusedindatamining:

- Association

  Associationisoneofthebest-knowndataminingtechnique. In association, apatternis discoveredbasedonarelationship betweenitemsin thesametransaction.That'sisthe reason why association technique is also known as *relation technique*. The association technique is used in *market basket analysis* to identify a set of products that customers frequently purchase together.

  Retailers are using association technique to research customer's buying habits. Based on historicalsaledata,retailersmight findoutthat customersalwaysbuycrispswhentheybuy beers, and, therefore, theycan put beers and crisps next to each other to save time for customer and increase sales.

- Classification

  Classification is a classic data mining technique based on machine learning. Basically, classification is used to classify each item in a set ofdata into one ofa predefined set of classes or groups. Classification method makes use of mathematical techniques such as decision trees, linear programming, neural network and statistics. In classification, we developthesoftwarethatcanlearnhowto classifythedataitemsinto groups.Forexample, we can applyclassification in the application that "given allrecords ofemployees who left thecompany, predict who willprobablyleavethecompanyina futureperiod."Inthiscase, we divide the records of employees into two groups that named "leave" and "stay".And then we can ask our data mining software to classifythe employees into separate groups.

- Clustering

  Clustering is a data mining technique that makes a meaningful or useful cluster of objects which have similar characteristics using the automatic technique. The clustering technique defines the classes and puts objects in each class, while in the classification techniques, objectsareassigned into predefined classes.To maketheconcept clearer, wecantakebook management in the library as an example. In a library, there is a wide range of books on varioustopicsavailable.Thechallenge ishowtokeepthosebooksinawaythat readerscan take several books on a particular topic without hassle. By using the clustering technique, we cankeep booksthat have some kinds ofsimilarities inone cluster or one shelfand label it witha meaningfulname. Ifreaderswanttograbbooksinthattopic, theywouldonlyhave to go to that shelf instead of looking for the entire library.

- Prediction

  The prediction, as its name implied, is one ofa data mining techniques that discovers the relationship between independent variables and relationship between dependent and independent variables.Forinstance,thepredictionanalysistechniquecanbeused inthesale topredict profit forthefutureifweconsiderthesale isanindependent variable, profit could be a dependent variable. Then based on the historicalsale and profit data, we can draw a fitted regression curve that is used for profit prediction.

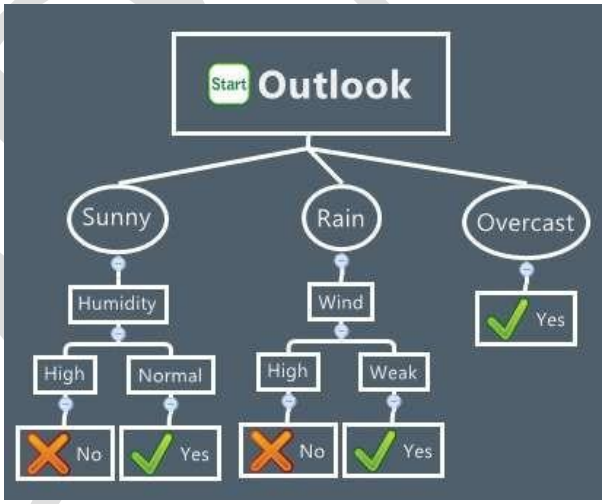
- SequentialPatterns

  Sequential patterns analysis is one of data mining technique that seeks to discover or identifysimilar patterns, regular eventsortrends intransactiondataover abusinessperiod.

  In sales, with historical transaction data, businesses can identify a set of items that customersbuytogetherdifferent timesinayear.Thenbusinessescanusethisinformationto recommend customers buy it with better deals based on their purchasing frequency in the past.

- Decisiontrees

  TheAdecision tree is one of the most common used data mining techniques because its modeliseasyto understand for users.Indecisiontreetechnique, therootofthedecisiontree is a simple question or condition that has multiple answers. Each answer then leads to a set of questions or conditions that help us determine the data so that we can make the final decision based on it. For example, We use the following decision tree to determine whether or not to play tennis:



  Startingattherootnode,iftheoutlookisovercastthenweshoulddefinitelyplaytennis. Ifit is rainy, we should onlyplay tennis if the wind is the week.And if itis sunnythen we should play tennis in case the humidity is normal.

Lecture-38

ParallelDatabase

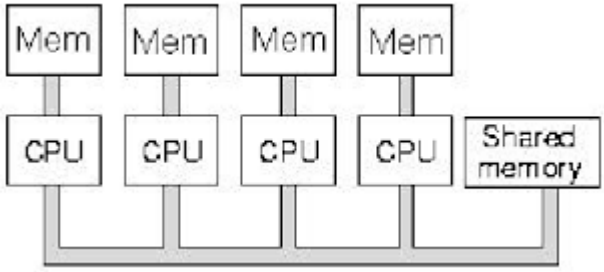
A**paralleldatabase** systemperformsparalleloperations, suchas loadingdata,building indexes and evaluating queries.

Paralleldatabasescanbe roughlydividedinto twogroups,

a) Multiprocessorarchitecture:

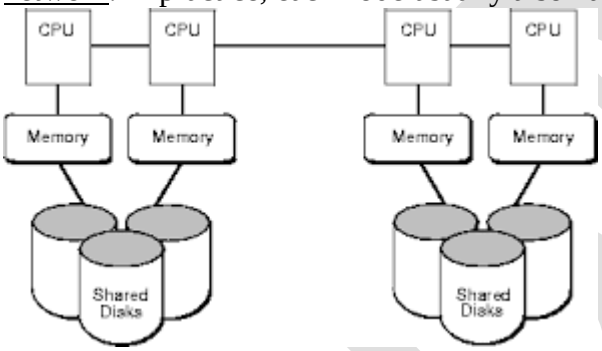
→Sharedmemoryarchitecture

Where multipleprocessorssharethemainmemoryspace.



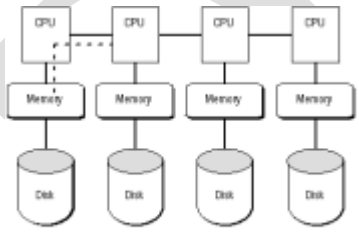
→Shareddiskarchitecture

Whereeachnodehasitsownmainmemory, but allnodessharemassstorage,usuallya storage area network. In practice, each node usually also has multiple processors.



→Sharednothingarchitecture

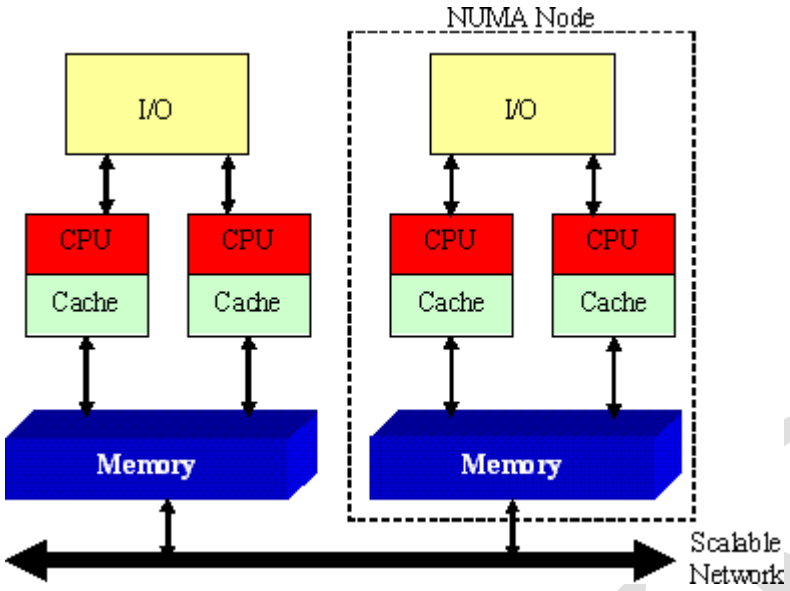
Whereeachnodehas itsownmassstorageaswellasmainmemory.



Shared-Nothing Architecture

b) Theotherarchitecturegroupiscalledhybridarchitecture, which includes:

- Non-UniformMemoryArchitecture(NUMA), whichinvolvesthenon-uniformmemory access.



- Cluster(sharednothing+shareddisk:SAN/NAS),whichisformed byagroupofconnected computers.

DistributedDatabase:

→Acentralized distributed database management system (DDBMS) manages the database as if it were all stored on the same computer. The DDBMS synchronizes all the data periodically and, in cases where multiple users must access the same data, ensures that updates and deletes performed on the data at one location will be automatically reflected in the data stored elsewhere.

→The users and administrators of a distributed system, should, with proper implementation, interact with the system as if the system was centralized. This transparency allows for the functionality desired in such a structured system without special programming requirements, allowing for any number of local and/or remote tables to be accessed at a given time across the network.

ThedifferenttypesoftransparencysoughtafterinaDDBMSare

1. datadistributiontransparency,
2. heterogeneitytransparency,
3. transactiontransparency,and
4. performancetransparency.

→Data distribution transparency requires that the user of the database should not have to know how the data is fragmented (fragmentation transparency), know where the data they access is actually located (location transparency), or be aware of whether multiple copies of the data exist (replication transparency).

→Heterogeneity transparency requires that the user should not be aware of the fact that they are using a different DBMS if they access data from a remote site. The user should be able to use the same language that they would normally use at their regular access point and the DDBMS should handle query language translation if needed.

→TransactiontransparencyrequiresthattheDDBMSguarantee thatconcurrenttransactionsdo not interferewitheachother(concurrencytransparency)andthat it must also handledatabaserecovery

(recoverytransparency).

→PerformancetransparencymandatesthattheDDBMSshouldhaveacomparablelevelof performance to a centralized DBMS. Queryoptimizers can be used to speed up response time.

TypesofDDBDesign
Non-Partitioned,Non-Replicated
Partitioned, Non-Replicated
Non-Partitioned,Replicated
Partitioned, Replicated

Advantages of DDBMS's

- Reflectsorganizationalstructure
- Improvedshareability
- Improvedavailability
- Improvedreliability
- Improvedperformance
- Dataarelocatednearestthegreatestdemandsiteandaredispersedtomatchbusiness requirements.
- FasterDataAccessbecauseusersonlywork withalocallystoredsubsetofthe data.
- Fasterdataprocessingbecausethedataisprocessedatseveraldifferentsites.
-GrowthFacilitation:Newsitescanbeaddedwithout compromisingtheoperationsofothersites.
-Improvedcommunicationsbecauselocalsitesaresmallerandclosertocustomers.
- Reduced operating costs: It ismore cost-effective toadd workstations toa network rather than update a mainframe system.
- UserFriendlyinterfaceequippedwithaneasy-to-useGUI.
- Lessinstancesofsingle-pointfailurebecausedataandworkloadaredistributedamongother workstations.
- Processorindependence:Theenduserisabletoaccessanyavailablecopyofdata.

Disadvantagesof DDBMS
- IncreasedCost
-Integritycontrol moredifficult,
-Lackofstandards,
-Databasedesignmore complex.
- Complexityofmanagement andcontrol.Applications must recognizedatalocationandtheymust be able to stitch together data from various sites.
- Technologically difficult: Data integrity, transaction management, concurrencycontrol, security, backup, recovery, query optimization, access path selection are all issues that must be addressed and resolved
- Securitylapseshave increasedinstanceswhendataareinmultiplelocations.
- Lack of standards due to the absence of communication protocols can make the processing and distribution of data difficult.
- Increased storage and infrastructure requirements because multiple copies of data are required at various separate locations which would require more disk space.
- Increasedcostsduetothe highercomplexityoftraining.
- Requires duplicate infrastructure (personnel, software and licensing, physical location/environment) and these can sometimes offset any operational savings.