24/12/11

Database and Database user.

CH-1

Database :

A DB is collection of related data. Data means known facts that can be recorded and have implicit meaning.

Eg : Ms access , Oracle, Sybase DB2 Etc.

A DB has the following properties :

→ A DB represent some aspects of real world.

→ A DB is logical co-herent collection of data and some inherent meaning. A random set of data cannot be refer has a dB

→ A db is design built popular [adding data to the db] with data for a specific purpose.

→ it has intended group of users.

DBMS : A DBMS is a collection of program [software] that Enables user to create and maintain a db hence dbms is a general purpose software that facilitates process of defining, constructing and manupulating db for various application.

→ Defining a db :

Defining a db involves specifying data type, structure, constraints for the data to be stored in the db.

→ Constructing the db :

It is a process of storing data itself on source storage medium that is controlled by DBMS.

→ Manupulating DB :

It involves function such as Querying the db updating base and generating the reports from the data.

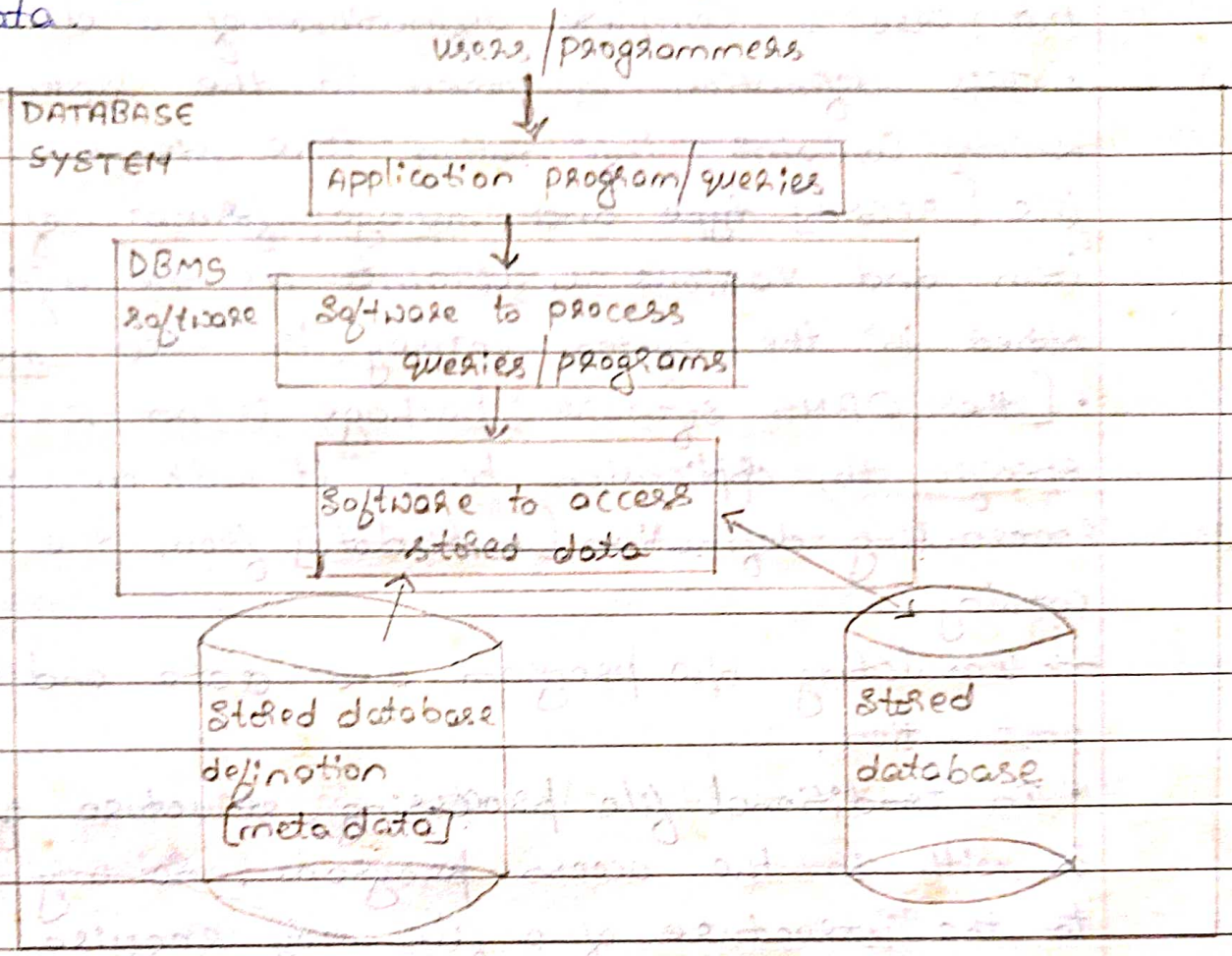


fig : database system Environment

→ uses queries the data base using application program.

→ This query is picked up the DBMS software which is going to convert the query required format using this information the software can access the database and fetch the information.

→ Software which is used to access stored data firstly refers to the defination about the db [metadata] present in the Catalog using this defination system it going fetch the data represent in the data base.

Characteristic of db approach :

→ Self describing nature of db system.

- The db system contain not only the db itself but also a complete defination of a db.
- This defination is stored in the system catalog which contains information like structure of each file [record] type and storage format of each data item and various constraints on the information stored in the system catalog is called metadata.
- [the DBMS software package is not written for specific db application hence it work on any db by extracting defination [metadata] from the system catalog.

→ Insulating b/w program and data and data abstraction

- In traditional file processing, structure of datafile is with in the access program. So any changes to the structure of a file may require changing all program that access the file.
- In contrast DBMS program donot require such changes since the structure of datafiles is stored in DBMS catalog seperate from access program this property is called as program data independence
- An object oriented db users ~~the~~ define the operation on the data has a part of db defination
- User program can operate on the data by invoking operations through interface [program name and parameter implimented list] regards of the operation are [any changes in the implementation is not going to effect] this is called as program operation indepence.

28/12/11 The characteristics that allows program data

independence and program operation independence is known as data abstraction.

→ Support of multiple views of data

DB has many users Each may require different view of db. A view may be a subset of db or it may contain virtual data that is derived from the db but it is not Explicitly stored.

→ Sharing of data and multiusers transaction processing

A multiuser DBMS must allow the multiple uses to access the db at the same time this is Essential if the data for multiple application is to be integrated and maintain in a single db. DBMS must include Concurrency control software so that several users trying to update the same data, do it in a Controlled manner so that the result is correct and Consistent.

→ difference b/w db approach and file oriented approach.

DB	File
• A single repository of data is maintain that is defined once and accessed by various users	• Each user defines and implements the files needed for specific application as a part of programming
• A db system Contains not only the db itself but also complete defination of db structure and Constraints	• data defination is typically a part of application program itself hence this programs work with only one specific db whose structure is declare within the application program
• The structure of data file	

is stored in DBMS catalog separately from access program	• The structure of data file is embedded [present] in the access program so any changes in structure of the file may require changes in all programs that access the file.
• A db typically has many users each may require diff views of the data hence it supports multiple views of the data	• This doesn't support multiple views of the data.
• DBMS allows sharing of data & multiuser transaction processing that is it allows multiple users to access the db at the sametime	• It doesn't allow multiple users to access the file at the sametime

**

29/12/11

Actors on the scene :

The people whose job involve day to day use of large db

→ Db administrator [DBA] : In a db environment the primary resource is the db itself and secondary resource is the DBMS & related softwares

• Administrating this resources is the responsibility of DBA

• DBA is a person or a group of person who are responsible for overall development of any project

• He is the chief administrator to oversee and manage all resources of the db

Date / /20

- He/she is responsible for authorized access to db, for co-ordinating and monitoring its use and for acquiring software and hardware resources as needed
- He is also responsible for system performance and security aspects. He is responsible for poor system response.

→ Data base designers :

They are responsible for identifying data to be stored in the db and for choosing appropriate structure to represent and to store the data. these task are undertaken before the db is actually implemented. It is the responsibility of db designers to Communicate with db users in order to understand their requirement and to come up with the design that meets the user requirement

→ End users :

They are the people who job require access to the db for quering, updating and generating the report.

There are several Entities in End user they are

* casual End user : They occasionally access the db but they may need different information Each time. They use sophisticated query language to specify there request.

Eg : middle or high level manager.

* Naive or parametric End user : There main job is to Constantly quering and ~~jobing~~ updating the db using standard types of queries and updates called canned transaction

That are carefully programmed and tested.
Eg : Bank tellers check account balance
Reservation clerks for airlines, hotels check the availability for giving the request and make reservation.

* Sophisticated End users : It includes engineers, scientists, business analysts who thoroly familiarize themselve with the facilities of DBMS so as to implement their application to meet their complex requirement

* Stand alone users : They maintain personal db by using readymade program packages that provide easy to use menu or GUI
Eg : users of tax package that stores variety of personal financial data for tax purpose

→ * System analist and application program [SE] :
System analists determine the requirement of the End users especially navie and parametric End users and develop specification for the transactions that meets the requirements
Application programmers implement this specification as program, then they test, debug, document and maintain the transaction.

* Workers behind the scene :
Workers those who work to maintain the db system environment but they are not interested in the db

→ DBMS system and designers & implementers
They are the persons who design and implement a DBMS module and interfaces

as a s/w package. A DBMS is a complex s/w system it consists of many components and modules

→ Tool developers :

They are the persons who design and implement tools. Tools are s/w packages that help in improving the performance of the db. They are optional packages and are purchased separately

→ Operators and maintainence personnel :

They are system administration personnel who are responsible for actually running and maintainance of h/w & s/w environment for the DBMS.

30/12/11 Advantages of using DBMS :

1) Controlling redundancy

In file oriented approach every group of users maintain their own file for handling data processing application. This redundancy is storing same data multiple times may led to several problems like wastage of storage space, inconsistent data and so on. but in DBMS the redundancy is avoided but controlled redundancy is allowed which may be helpful in improving the performance of query.

→ Restricting the unauthorized access :

The DBMS should provide security and authorization sub system such that only the users

who are given authorities by DBA can only use db.

→ providing persistance storage for program objects and data structure.

programming languages have complex data structure like class defination in C++. The value of this program variables ~~variable~~ are discarded once the program terminates unless the program explicitly stores it in a file, when the need arises to provide this data to a programmer there is a need for conversion from file format to program variable structure.

But OO db are compatable with programming language such as C++, java, .

The DBMS software automatically performes necessary conversions hence complex object can be stored permenently in OO [such an object is said to persistant]

→ providing multiple user interface :

31/12/11 DBMS should provide a variety of interface they include query languages called casual end users, programming language interface for application programmers forms and command course for paramatric users and menu driven interfaces & natural language interface for standalone users

→ Enforcing integrity constraints :

most db applications have certain integrity constraint that must be followed for the data. DBMS should provide capability for defining & enforcing these constraints

Eg : specifying datatype for each data item

Date / /20

→ Providing backup and recovery :
A DBMS must provide facilities for the recovery from h/w & s/w failure. The backup & recovery subsystem of DBMS is responsible for recovery

→ Representing complex relationship among the data :
DBMS must have the capability to represent a variety of complex relationship among the data as well as to retrieve & update related data Easily & Efficiently

Implication of db approach [benefits of DBMS] :

→ Reduced application development time : once the db is up & running less time is required to create new application using DBMS facilities. Developement time used in DBMS is Estimated to be 1/6 th to 1/4 th of time required using traditional file system.

→ potential for Enforcing standards : The db approach permits DBA to define & Enforce standards among db users in large organisation

→ Flexibility : It may be necessary to change structure of the DB as the requirements changes. DBMS allows certain types of changes to the structure of db without affecting structure of stored data & Existing applications

→ Availability of upto date information : A DBMS makes the db available to all users as soon

as one users updates, it is applied to its db. All other users can immediately see this update.

→ Economics of scale : A DBMS approach allows consolidation of data & application this enables the whole organisation to ~~impressed~~ invest in more powerful process of storage devices & so on rather than having each department purchase its own equipments this reduces overall cost of operation and management

<u>When not to use DBMS</u> :

There are few situation in which DBMS is not advisable to use they are

* DB is very simple
* Multiple user access to the db is not required
* DB doesn't change for long time
* The overhead cost of using DBMS are due to the following
  → high initial investment in h/w & s/w & training.
  → overhead of providing security, concurrency control, recovery and integrity function.

<u>Application of DBMS</u> :

→ Scientific : It stores large amount of data resulting from scientific experiments [such as high energy physics & mapping of human gene]

→ Images : Storage & retrival of images from scanned news to satellite photograph image

and images from medical procedures such as X-rays [MRI]

→ videos : storage and retrival of videos such as movies or video clips from news or personal digital Camera

→ Data mining : ~~which~~ Data mining application that analysis large amount of data, searching for the ~~upperbounds~~ occurance of specific pattern.

→ spatial application : app that store data such as weather information.

→ Time series : app that store information such as Economic data at regular point in time

2/1/12

Data Base System Concepts and architecture

CH-2

Data model is a collection of concepts that can be used to describe the structure of the db. The structure of db means datatypes, relationship & the constraints that hold on data.

Catagories of data model

→ Highlevel or conceptual data model :

It provides the concepts that are closed to the way many users perceive they use concepts such as Entity, attribute and relationship.

An Entity represent an real world object or conceptual.

Eg : Employee, project.

An attribute represents some property of an Entity

Eg : for Employee Entity ID, name, address are some of the attribute.

A relationship among two or more Entity is an association among two or more Entity.

Eg : works on relationship b/w an Employee and project.

→ low level or physical data model :

It describe how data is stored in computer memory by representing information, record ordering and access path such as record format which makes search for a particular db record efficient.

→ Representational or implimentational data model :
→ This provides concepts that may be understood by End users and also some information regarding the organization of data within the computer.
→ Representational data model hide some detail of data storage but can be implimented on a computer system in a direct way.

→ Network or hierarchical model :
They were used widely in the passed
→ Record based data model : In this data model data is represented using record structure.
→ object based data model : These are high level data implimentation data model that are closed to conceptual data model.

Schema, instances and data based states :

Schema : The description of data base is called db schema which is specified using db design and is not excepted to change frequently. most data model have certain convention for displaying schemas has diagram and is called 'as schema diagram.

STUDENT

Name	Studentnumber	class	major

COURSE

Course name	Coursenumber	Credit hours	dept

GRAD - REPORT

			grade

Each object in the schema such as student is called schema construct.
The actual data is the db may change frequently the data in db at the particular time is called as db state or snapshot or instance or occurances in the db.
The schema is also called as intension of the db. the db state snapshot or instance is called Extension.

DBMS architecture or 3 schema architecture.

There are 3 levels they are :

→ internal
→ Conceptual
→ External.

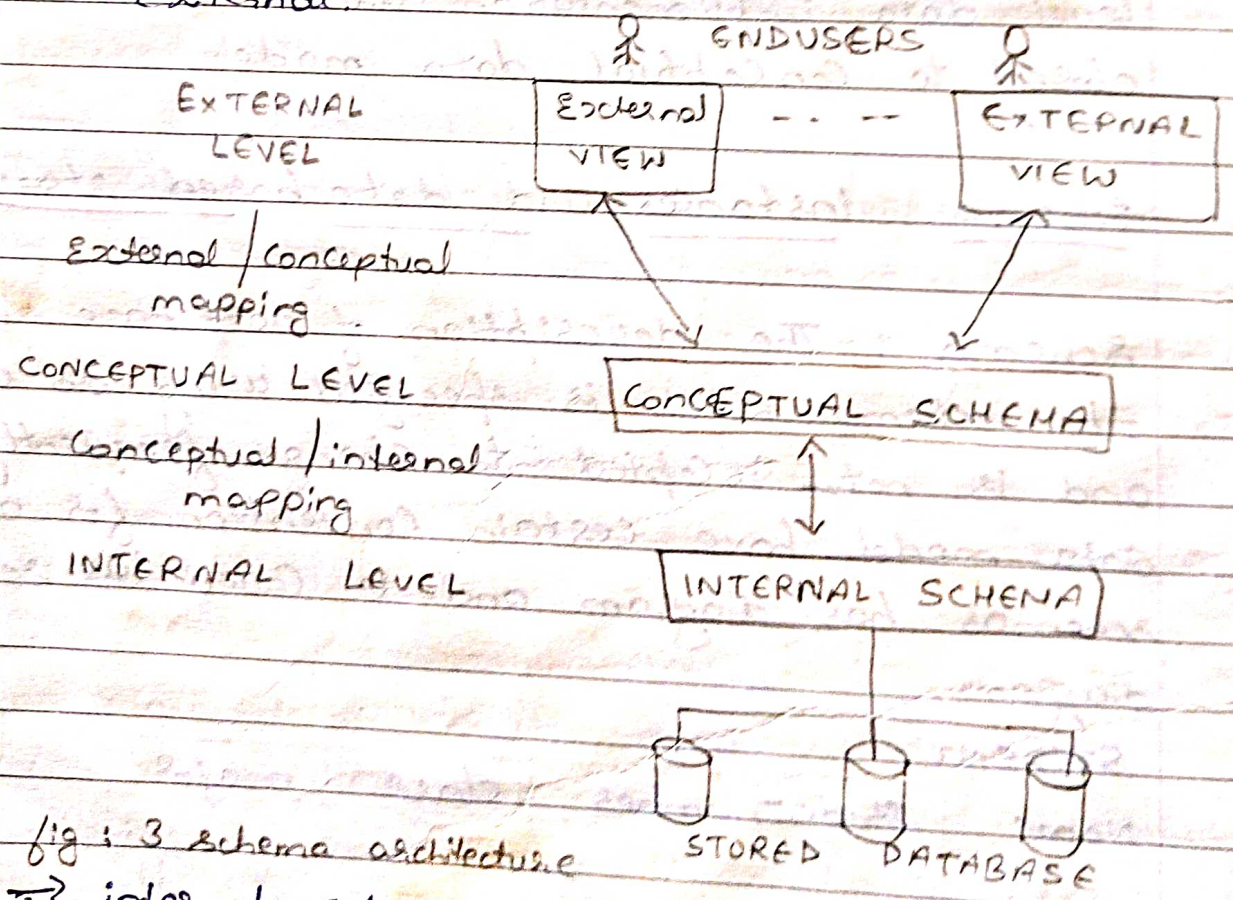


fig : 3 schema architecture

→ internal schema : It describe the physical storage structure of db it uses physical data model and describe the storage structure and access path.

Date / /20

→ Conceptual : It describe the structure of whole db to the user it hides the details of the physical storage structure & describes Entity, datatype, relationships, user operations and Constraints.

It uses high level data model and implimential data model.

→ External schema : Each External schema describes a part of the db or subset of db that the particular user group is intrested it hides the rest of the db from the user group

A high data model and implimentational data model can be used at this level.

users come in the External level their request must be transfered from External schema to Conceptual schema from their to internal schema process of transfering request b/w level is called mapping.

Data independence : Data independence is defined as capacity to change the schema at one level of db system with out effecting the schema's at other level.

→ logical data independence : It is the capacity to change conceptual schema without changing External schema.

Conceptual schema will be change to Expand data bases by adding the data items (or) reduce the db by remaining the data item.

→ physical data independence : It is the capacity to change internal schema without having to

change conceptual or External schema changes to the internal schema [physical level] may be required to improve the performance of retrival, updata operations on the db.

5/1/12 <u>DBMS languages</u> : There are 6 types of languages

→ <u>Data defination language [DDL]</u> :

It is used by the database administrator and db designers to define conceptual and internal schema.

The DBMS will have DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema construct and to store the schema description in the DBMS catalog.

→ <u>Storage defination language [SDL]</u> :

[In DBMS where a clear seperation is maintained b/w the conceptual and internal level DDL is use to specify the conceptual schema only].

Storage defination language is use to specify the internal schema.

→ <u>view defination language [VDL]</u> :

It is use to specify user views and their mapping to conceptual schema [view contains subset of actual data in the db]

VDL comes under external schema.

→ <u>Data Manupulation language [DML]</u> :

Fetching or retrieving from the database is known as Querying

Bhaskar Date / /20

Data manupulation includes retrival, insertion deletion and modification of data. DML use for this purpose.

DML is classified into,

* high level or non-procedural DML
* low level or procedural DML

High level DML are entered in the command form they normally can extract more than one record, it is also called as set-at-time DML.

Low level DML must be embedded in high level programming languages like C, C++ etc. It can extract individual records at a time it is called as record-at-time DML

→ Structure Query Language [SQL] :

It is a db language which represents a combination of DDL, VDL, DML as well as statements for constraints specification and schema Evolution.

→ Host language : whenever a DML commands at embedded in general purpose programming language, that language is called as host language.

DBMS interface :

The following are the interfaces provided by the DBMS to interact with the stored db they are :

→ Menu based interface

→ form based interface

→ GUI based interface
→ Natural language interface
→ Interface for parametric End users
→ Interface for DBA.

→ Menu based interface :
DBMS s/w supports different types of menu based interfaces which includes pop-up menu, pull-down menu etc.

→ Form based interface :
A form based interface displace a form to Each users. users can fill out all of the form Entries to insert new data or they fill out only Certain Entries in which case the DBMS will retrieve matching data for the remaining Entries

9/1/12 → GUI based interface :
A GUI displays the schema to the user in diagrametic form. The user can specify queries by manupulating diagrams it also supports mouse interaction

→ Natural Language Interface [NLI]:
Here the queries will be Entered in a natural language like English and will be converted internally as a query language [this interface is useful for those users who don't have knowledge in query language]

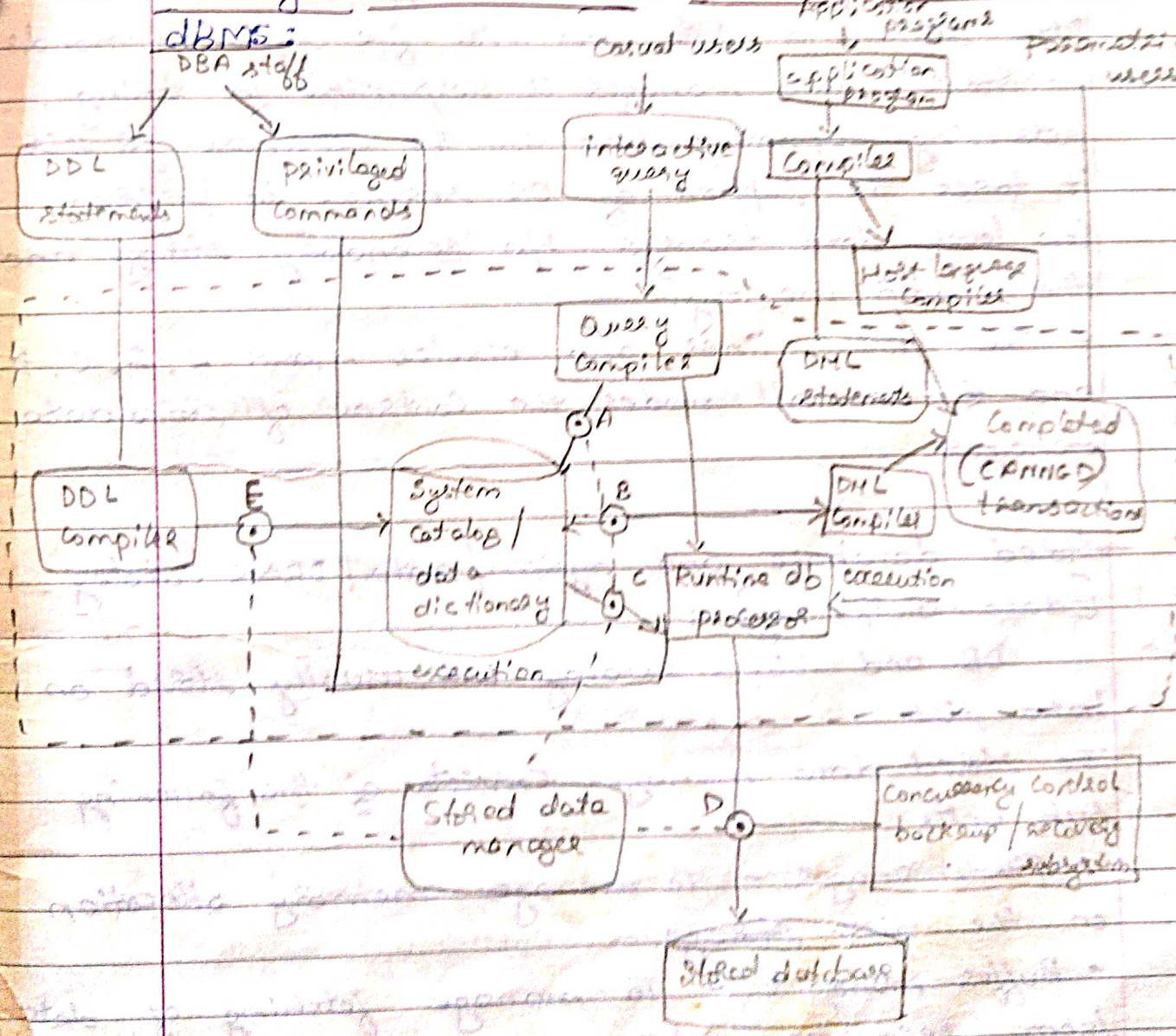
→ Interface for parametric End users :
parametric users such as clerks uses the

db very frequently to perform small set of operation. The small set of abbreviated commands are provided for minimizing the number of keystrokes required for each request.

→ Interface for DBA :

Most db system contains privileged commands that can be used only by DBA. This includes commands for creating accounts, granting accounts authorization, changing the schema & so on.

* DB system Environment or components modules of dbms :



Component modules of DBMS :

The following are different components of DBMS

→ Query compiler
→ DDL compiler
→ DML compiler
→ pre compiler
→ Stored data manager
→ System / data / DBMS catalog [data dictionary]
→ Run time db processor
→ Stored db

→ Stored data manager :

• It controls access to the dBMS information stored on the disk. Data manager converts various DML statements into low level file system command
• Stored data manager is responsible for storing, retrieving, updating data in the database for this it takes the support of OS services to perform low level data transfer b/w secondary storage and the main memory.

The dotted lines and circles mark A, B, C, D and E are all under the control of stored data manager.

→ Stored database and system / data / DBMS catalog [data dictionary] :

DB and DBMS catalog are usually stored on hard disk

The stored data manager consists of the following sub units

• File manager - To manage memory allocation on the disk storage
• Buffer manager - To manage fetching of data from secondary storage unit into main memory

• Transaction manager - To manage concurrent transaction processing and to Ensure Consistancy
• Authorization and integrity manager - To manage integrity Constraints and to provide authentication and authorization for users to access data in db

→ Run time data processor :
It performs run time db access and receives retrieve and update Commands and performs these operations on db at run time.

→ DDL Compiler :
It performs translation of DDL statements into schema defination and stores schema description called metadata in a DBMS Catalog

→ DBA's staff :
They Execute 2 types of commands that is
• DDL statements • privilaged Commands
These statements are Compiled by DDL Commands

12/1/12 → SQL Query Compiler :
It performs translation of high level query into db access codes and then generates Calls to the run time processor for Executing the Code

→ Casual End users :
Casual End user issue high level queries ~~and~~ it is processed by query Compiler

→ pre Compiler :
It will identify SQL or DML statements

Embedded in a host language application program, It seperates SQL or DML statements from host language

→ DML Compiler :

The SQL statements Extracted by precompiler and submitted to DML compiler for compilation into object code for db access. This translation of command issues by parametric End users using canned transaction are finally processed by runtime processor to retrieve, update or processed data in the db.

→ Application programmers :

Application programmes written by application programmer are first passed through pre compiler to seperate host language and DML or SQL statements. Host language is compiled by host language compiler. DML or SQL statements are compiled by DML Compiler

→ Concurrency control & backup recovery :

Concurrency control is required whenever a db is accessed by more than one user concurrently [same time or simultaneously]

Backup and recovery ~~take~~ ~~p~~ techniques are used to protect db from h/w and s/w failure

Database System utilities :

→ loading :

Loading utility is use to load existing data file such as text file, sequential files into DB. usually the current format of the data file and the design [target] data base file structure are specified to the utility

utility s/w automatically reformate the data and store it in the database. Such tools are also called as Conversion tools.

→ Backup :

A backup utility creates a backup [copy] of the database usually by dumping the entire database on to the tape. Backup can be use to restore the database incase of failures.

→ file reorganization :

This utility can be used to reorganize db file into different file organization to improve performance.

→ performance monitoring :

Such utility monitors the database usage and provides statistics [about the performance] to the DBA. The DBA uses the statistic in ~~maintai~~ making decisions such as whether or not to reorganise files to improve performance

Classification of DBMS :

DBMS can be classified based on the foll category; they are

→ Based on data model
→ Based on number of users supported by the system
→ Based on number of sites over which the db is distributed
→ Based on cost
→ Based on type of access path options for storing files

→ Based on data model :

DBMS can be categorized as relational, object, object relational, n/w, hierarchial.

Relational model :

It represents db as the collection of relations [tables]

Object model :

The object data model defines a db in terms of objects there properties and operations

Relational object oriented data model :

Relational DBMS have been extending these models to incorporate object database concepts and other capabilities. They are also called as extended relational systems.

13/1/12 N/w model :

n/w model represents data as record time

Hierarchial data model :

It represents data as hierarchical tree structure

→ Based on number of users supported by the system

- Single user system :

It supports only one user at a time and is mostly used with personal computers.

- Multi user system :

They include majority of DBMS, supports multiple users concurrently

→ Based on number of sites over which the db is distributed

- Centralized DBMS :

In this data is stored at a single computer site

- Distributed DBMS :

In this actual db and the DBMS s/w are distributed over many sites connected by computer n/w

[same copy of DBMS & db are stored in multiple system]

→ Based on cost :

The DBMS is classified based on cost. The majority of the DBMS packages cost b/w $10,000 and $100,000. A few data packages cause more than $100,000.

→ Based on types of access path option for a storing files :

- General purpose :

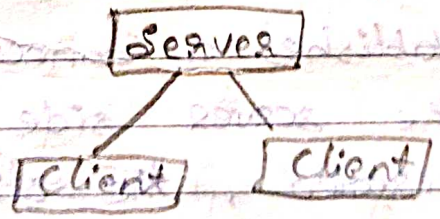
It can be used for any application

- Special purpose DBMS :

The performance is primary consideration special purpose DBMS is used and it is also used to built special application.

client - server architecture :

In general a client-server architecture they are server & client



A client is a user machine that provides user interface capabilities and local processing when a client requires additional functionality, such as database access, that does not exist on client, it connects to a server that provides the needed functionality.

A server is a machine that can provide services to the client machine such as file access, printing or database access

There are 2 DBMS architecture on this client - server architecture.

1) 2 tier
2) 3 tier

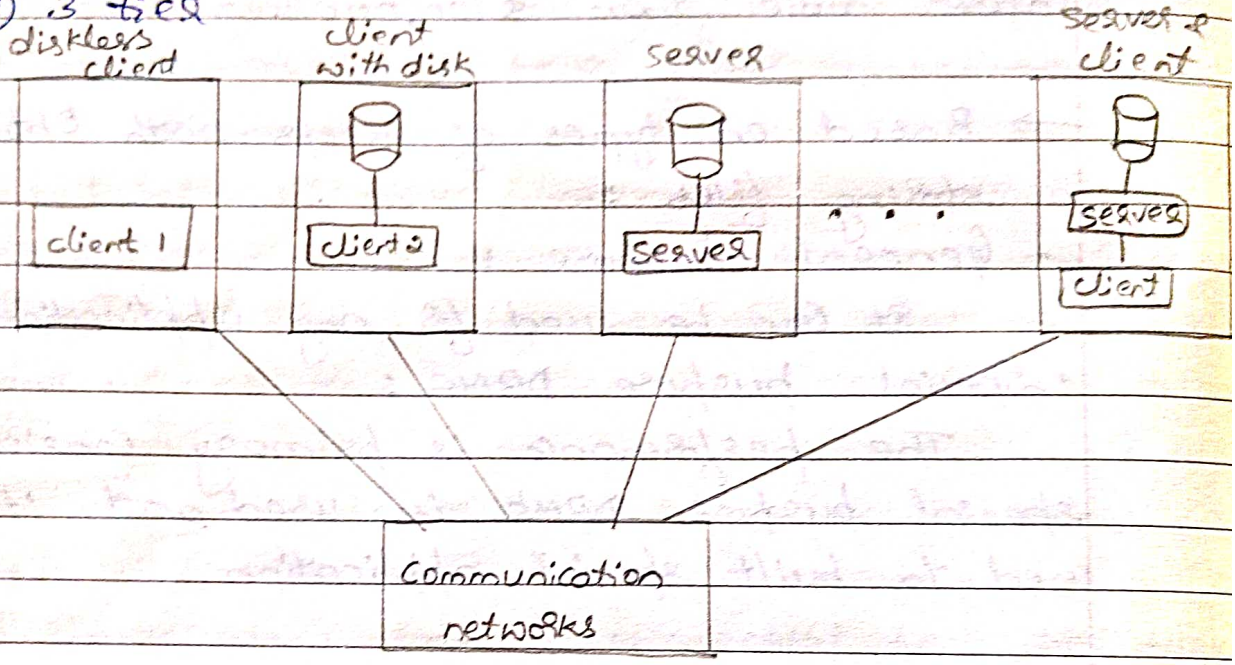
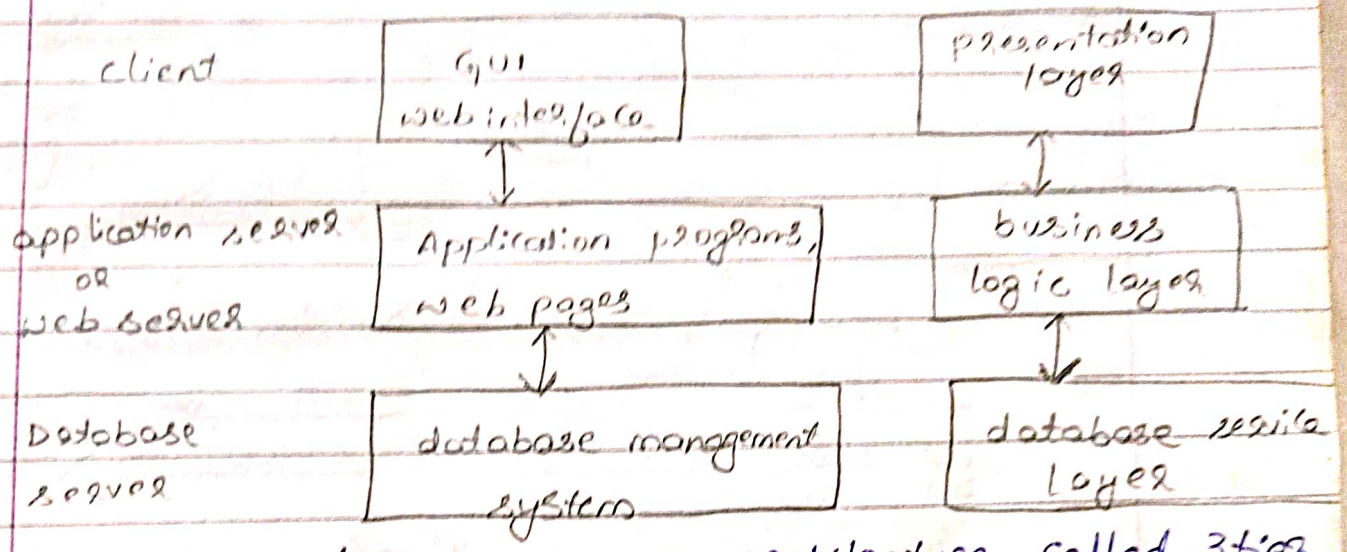


fig: two tier client server architecture

In two tier client server architecture the user interface program & application program can run on the client side, when DBMS access is required the program establishes a connections to the DBMS which is on the server side, once the connection is created, the client program can communicate with the DBMS. A standard called ODBC provides an application program interface [API], which allows client side program to call the DBMS, Most DBMS vendors provide ODBC driver for their system, A client program can actually connects several DBMS and send query & transaction requests using the ODBC API which are then processed at the server side

Any query results are send back to the client program which can process are display the results. The figure shows the 2 tier client server architecture.

Three - tier client server architecture :



Many web application use an architecture called 3 tier architecture, which adds an intermediate layer b/w the client & the database server as shown in fig. This intermediate layer or middle tier is sometimes called application server & sometimes the web server depending on the application. This server plays an intermediate role by storing business rule (procedures or constraints) that are used to access data from the database server.

Clients contain GUI interface and some additional application specific business rules. The intermediate server accept request from the client, processes the request & ~~da~~ sends database commands to the database server and then act as a conduit for passessing processed data from database server to the client. Thus the user interface, application rules & data access act as the 3 tiers.

17/1/12

DATA MODELLING USING ENTITY RELATIONSHIP [ER] MODEL

CH-3

Data modelling using Entity relationship model :-

* → Main phases of database design -

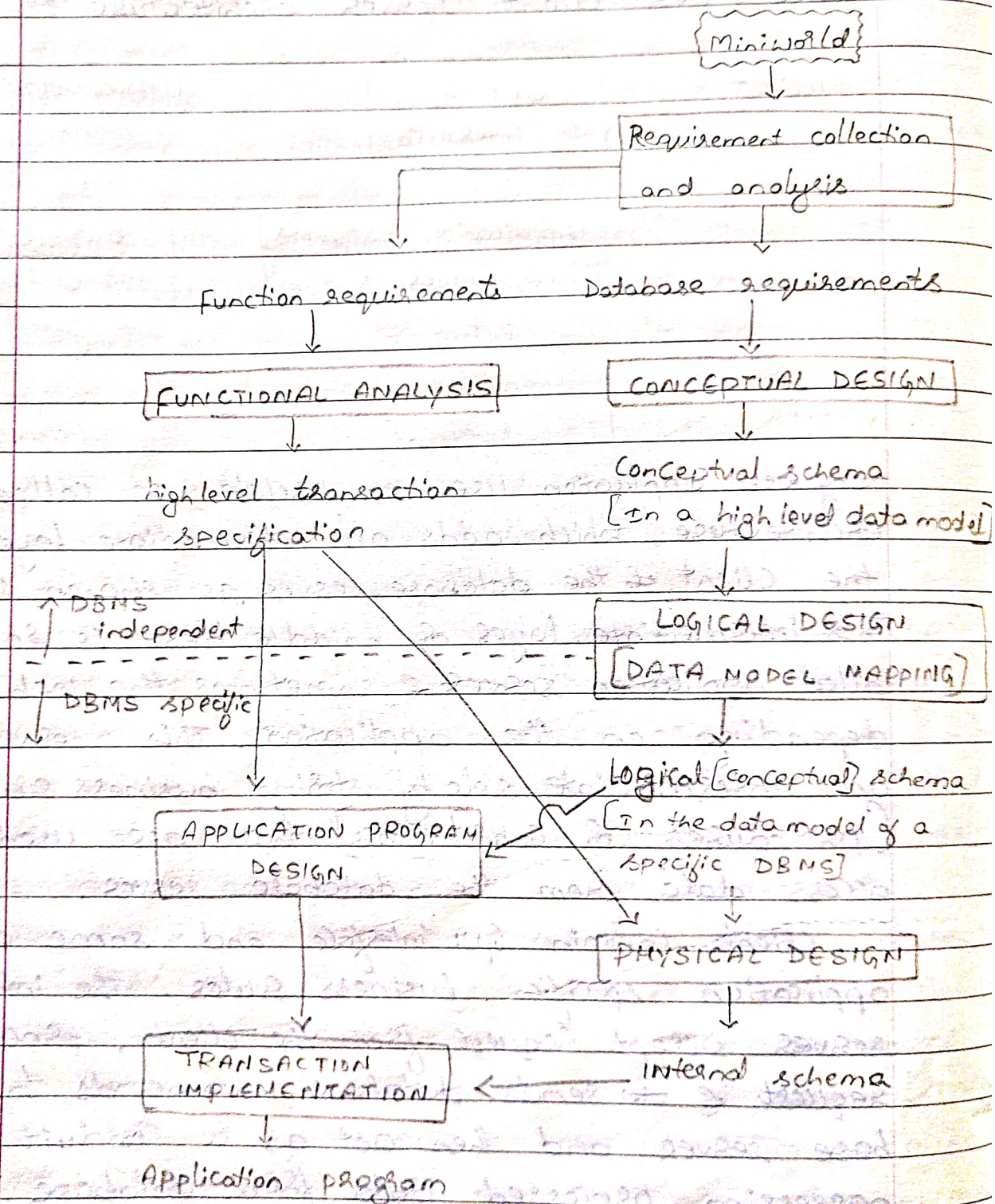


fig : main phases of database design

The steps followed in designing a database are :

1) Requirements collections and analysis :

All the ~~requiremen~~ the database designers interview prospective database users to understand and document these data requirements. These result of the step is a ~~re~~ return set of users' requirement. In parallel to this information regarding user defined operations or transactions that will be applied to the database to retrieve and update database will be collected the result of this is a function requirement.

Requirements are classified into

a) Database requirement :

To identify the database tables

b) Functional requirement :

To identify the functions required in the applications.

2) Conceptual design :

Once all the requirements have been collected and analysed the next step is to create conceptual schema. The conceptual schema is a brief description of database requirements of the user it includes description of Entity types, relationships and constraints.

3) Logical design :

In this step the actual implementation of the database is done using commerical DBMS like oracle, sybase. This is an implementation level of database model.

4) physical design :

During this phase internal storage structure, access path and file organization are specified

5) functional analysis :

This step is Executed in parallel with the database analysis. Analysis is done for the requirements collected for transactions and other applications applied to the database

6) Application program design :

During this stage design is developed for the transactions identified in the requirement stage

7) Transaction implementation :

During this stage functions or transactions are implemented using high level language [modification to the conceptual schema can be done if some functional requirements cannot be implemented]

<u>Entities and attributes</u> :

Entyties -

An Entity is a real world object with an independent Existance. An Entity may be an object with

physical Existence - Eg : person, car, house

Conceptual Existence - Eg : Company, job, course

Attributes -

Attributes are the particular properties that describe an Entity

Eg : An Employee Entity can have attributes like

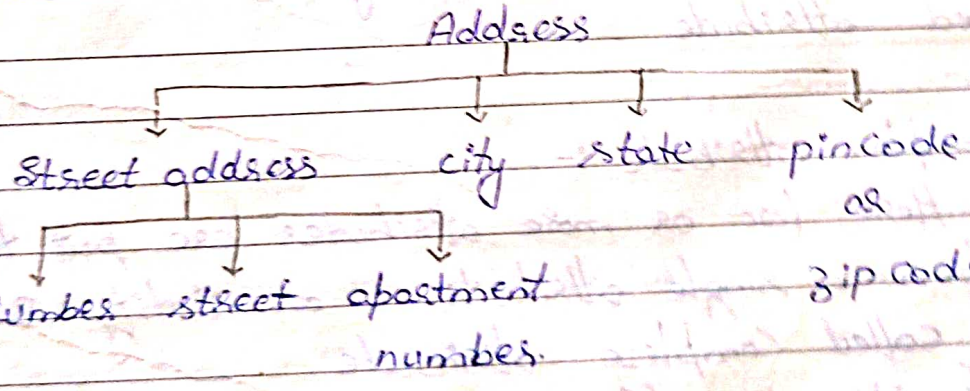
name, age, address, salary.

Attribute values :
A particular Entity will have a value for Each of its attributes. This is called as attribute values.

* Types of attributes in ER model :

19/1/12 1) Simple attribute :-
Simple attributes are the attributes which are not further divisible.
Eg : Emp ID, Emp age

2) Composite attribute :-
Attributes that can be divided into smaller sub parts are called composite attributes
Eg : The address attribute of an Employee can be divided into street address, city, state, zip code



3) Single value attribute :-
Most attributes have a single value for a particular Entity. Such attribute are called single value attribute
Eg : age attribute of Entity person.

4) Multivalued attribute :-

Certain attributes can have a set of values for the same Entity such attributes are called multi valued attributes.

Eg: college degree attribute of a person Entity one person may not have any degree and another person may 2 or more degrees

5) Stored and derived attribute :-

In some cases 2 or more attribute values are related

Eg : age and date of birth attribute of a person for a particular person Entity value of the age can be determined [found] from current date and value of the attribute date of birth stored in the person Entity.

The age attribute is called as <u>derived attribute</u> and it is derivable from date of birth attribute which is stored in the person Entity and it is called as <u>stored attribute</u>

6) Complex attribute :-

Here two or more attributes are put together to form a single attribute in an Entity such attrib are called Complex attribute

Eg : { address phone ({ phone (area code, ph no)},
address (street address (number, street, apartment no),
city, state, zipcode))}

7) Null valued attribute :-

In some cases a particular Entity may not have applicable values for an attribute
May be in the following situations

Date / /20

a) The value for an attribute Exists but the value is not known.
Eg : age or height attribute of a person
b) The value for an attribute may or ~~not may~~ not Exists.
Eg : Contact number attribute of a person.

Entity type, Entity set, value set :-

Entity type :-

Entity type defines a collection [or set] of Entity that have same attribute. Each Entity type in database is described by its name & its attributes

Entity set :-

The collection of all Entities of a particular Entity type in the database at any point in time is called as Entity set or Extension.

Entity type name : EMPLOYEE COMPANY

attribute : Name, age, salary Name, headquarters, president

ENTITY SET : (Extension)	e_1 Maheshkumar, 55, 80k	c_1 CDAC, Hyderabad, mahesh
	e_2 Rishab Govind, 40, 30k	c_2 CDOT, bangalore, musty
	e_3 Ajaykumar, 25, 20k	
	⋮	⋮

value set of an attribute or domain :-

An attribute of an Entity may be associated

with a set of values called value set or domain of values, which specify the set of values that may be assigned to that attribute.

Eg: If the range of age allowed for Employees is b/w 18 to 60, the value set of the age attribute of Employee Entity is specify to be the set of integer, number b/w 18 to 60.

Key attribute of an Entity type :-

It is an attribute whose values are distinct [unique] for Each individual Entity [instance]. It is used to distinguish one Entity [one instance] from another in an Entity set.

Eg: In an Employee Entity set Emp ID can be Choosen has the key attribute since it has distinct values

Sometimes several attributes together form a key that is Combination of attribute values must be distinct or unique for Each Entity.

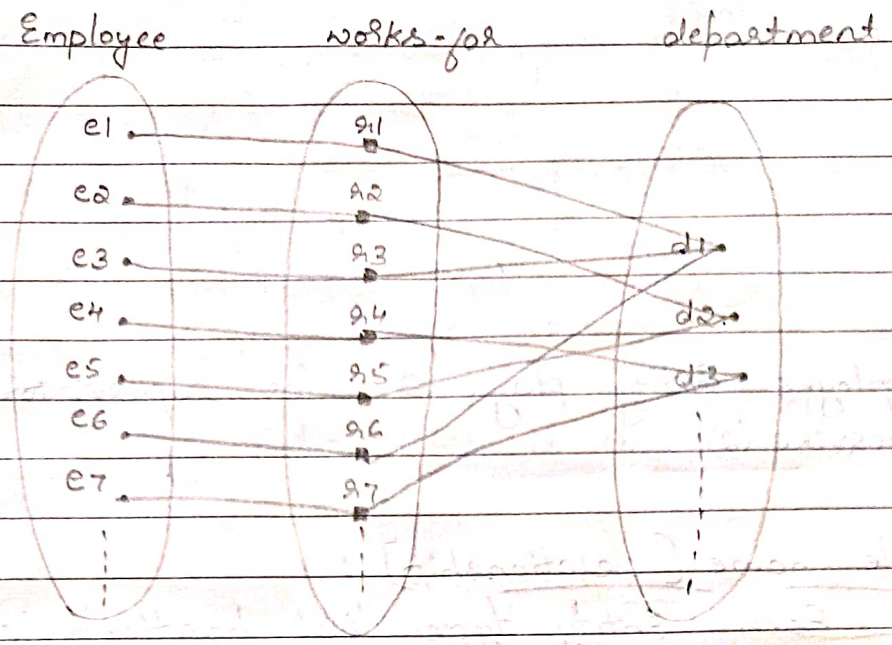
Relationship :-

Representation of how an Entity in an Entity set is related with an Entity of another Entity type is called relationship

Eg: "works-for" is the relationship b/w Employee Entity type and department Entity type

Relationship types :-

A relationship type R among n Entity types E1, E2 ---- En defines a set of association or a relationship set among Entities from these Entity types



Relationship set :-

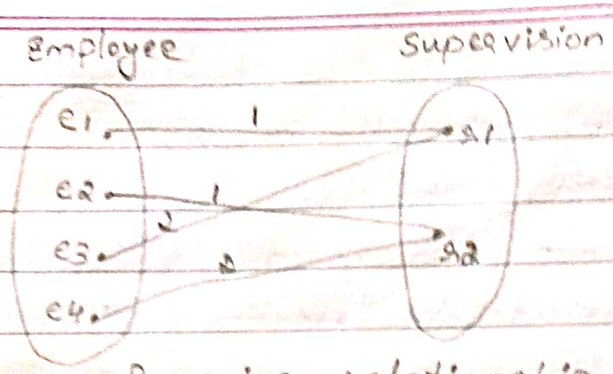
Relationship set R is a set of relationship instances (R_i) where Each r_i is associated with n individual Entities [$E_1, E_2 \ldots E_n$].
[In relationship set ~~are~~ R, R_1, R_2 ... r_n are relationship instances]

Relationship degree :-

The degree of the relationship type is the number of participating Entity types. Hence the works-for relationship is of degree 2. A relationship type of degree 2 is called binary relationship and of 3 is called ternary relation
Eg for ternary relationship :
supplier, parts, project Entities participate in supply relationship type. The relationship degree is 3.

Recursive relationship :-

In some cases the same Entity type participates more than once in relationship type in different roles such type is called recursive relationship



Recursive relationship

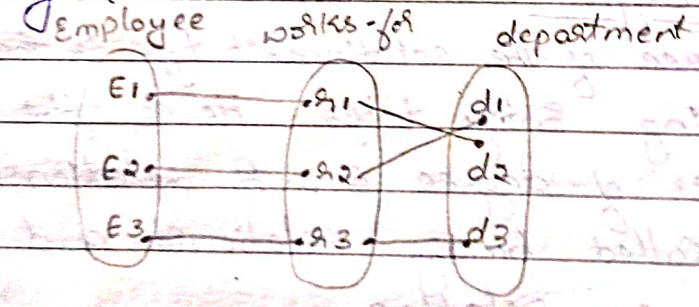
Employee Entity plays 2 roles supervisor (1), supervise (2)

23/1/12 <u>Role name [relationship]</u> :

Each Entity type that participates in a relationship type plays a particular role in the relationship

The role name signifies the role played by participating Entity in Each relationship instance

The works-for relationship type, Employee plays the role of employee or worker and DEPARTMENT plays the role of department or Employer



<u>Constraints on relationship type</u> :

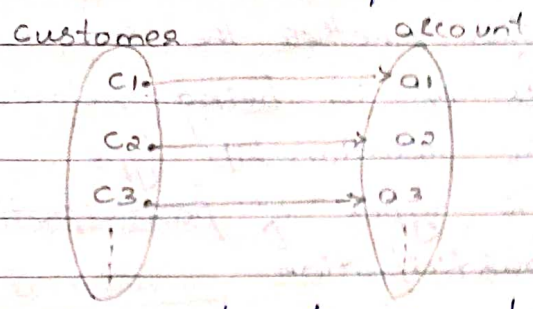
There are 2 main types of relationship Constraints

1) Cardinality ratio
2) participation constraints

** 1) Cardinality ratios for binary relationships [relationship types] : The cardinality ratio for a binary relationship specifies the no of relationship instances that an Entity can participate in.

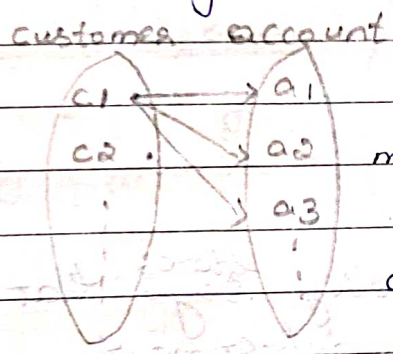
Date / /20

a) one to one relationship :



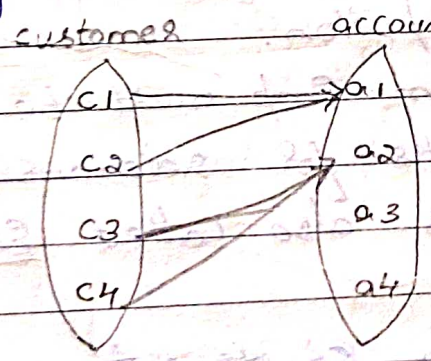
customer in a bank can have only one SB account this relationship is called one to one relationship.

24/1/12 b) one to many relationship :



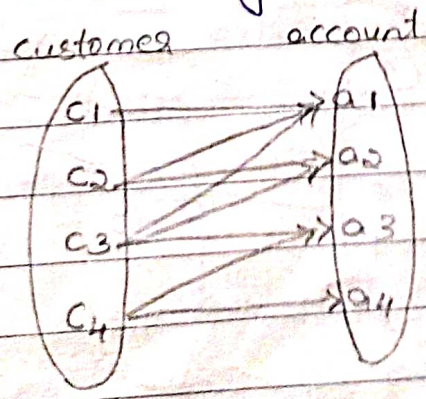
customer in a bank can open more than 1 account say for Eg : 1 SB account, 1 FD account, 1 RD account.

c) Many to one relationship :



Many customers join together & open a joint account. This is an Eg for many to one relationship.

d) Many to many relationship :



Many customers join together & open many joint accounts. This is an Eg for many to many relationship.

Participation Constraints :

It specifies whether the Existence of one Entity depends on its being related to an another Entity via the relationship type.

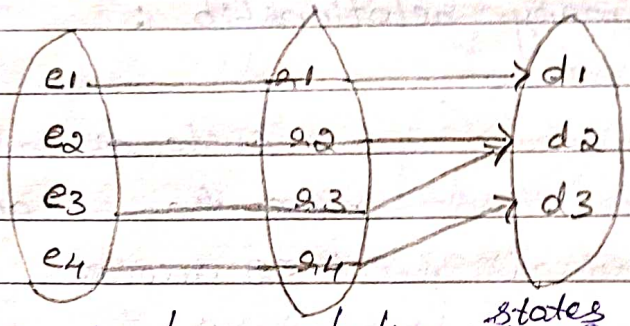
There are 2 types of participation constraints :

a) Total participation

b) partial participation.

a) Total participation constraints :

Employee works-for department



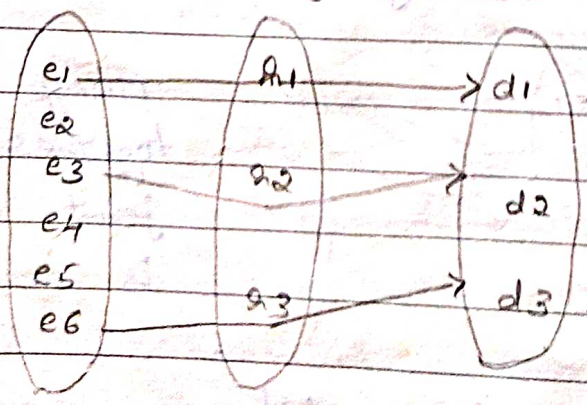
Eg : If a company policy ~~says~~ states that Every Emp must work for a department, the partici-pation of Emp in works for relationship is called total participation.

Every Entity from Emp must be related to dpt. Entity via works-for relationship.

Total participation is also called <u>Existence depen</u>

b) partial participation Constraints :

Employee manages department



25/1/12 The participation of all Entity in a relationship is

not compulsory is called partial participation.
Eg : In "Manages" relationship Every Employee doesn't manage a department.

Weak Entity type :
Entity type that do not have key attribute of these are are called weak Entity types.

Strong Entity type :
Entity type that have key attribute of these are are called strong Entity type
Eg : Employee.
weak Entity type always depends on a strong Entity type.
Strong Entity is also called as identifying Entity or owner Entity or parent Entity or dominent Entity.

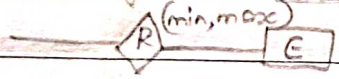
Identifying relationship :
The relationship type that relates weak Entity type to its owner or strong Entity type is called as identifying relationship.

Partial key :
A weak Entity type has a partial key which is the set of attributes that can uniquely identify weak Entities that are related to the same owner Entity.
Eg : In dependent table the attribute dependent_name is a partial key.

Attributes of relationship type :
Relationship type can also have attributes similar to Entity types.

Eg : To record numbers of hours per week that an Employee works-on a particular project, an attribute "Hours" can be included for the relationship type "works-on".

** ER diagram and naming conventions:

	Symbol		meaning
1)		–	Entity
2)		–	weak Entity
3)		–	Relationship
4)		–	Identifying relationship
5)		–	attribute
6)		–	key attribute
7)		–	multivalued
8)		–	Composite attributes.
9)		–	Derived attributes.
10)		–	Total participation of E2 in R
11)		–	Cardinality ratio 1:N for E1 : E2 in R
12)		–	Structural Constraint (min, max) on participation of E in R.

ER diagram for Company database:

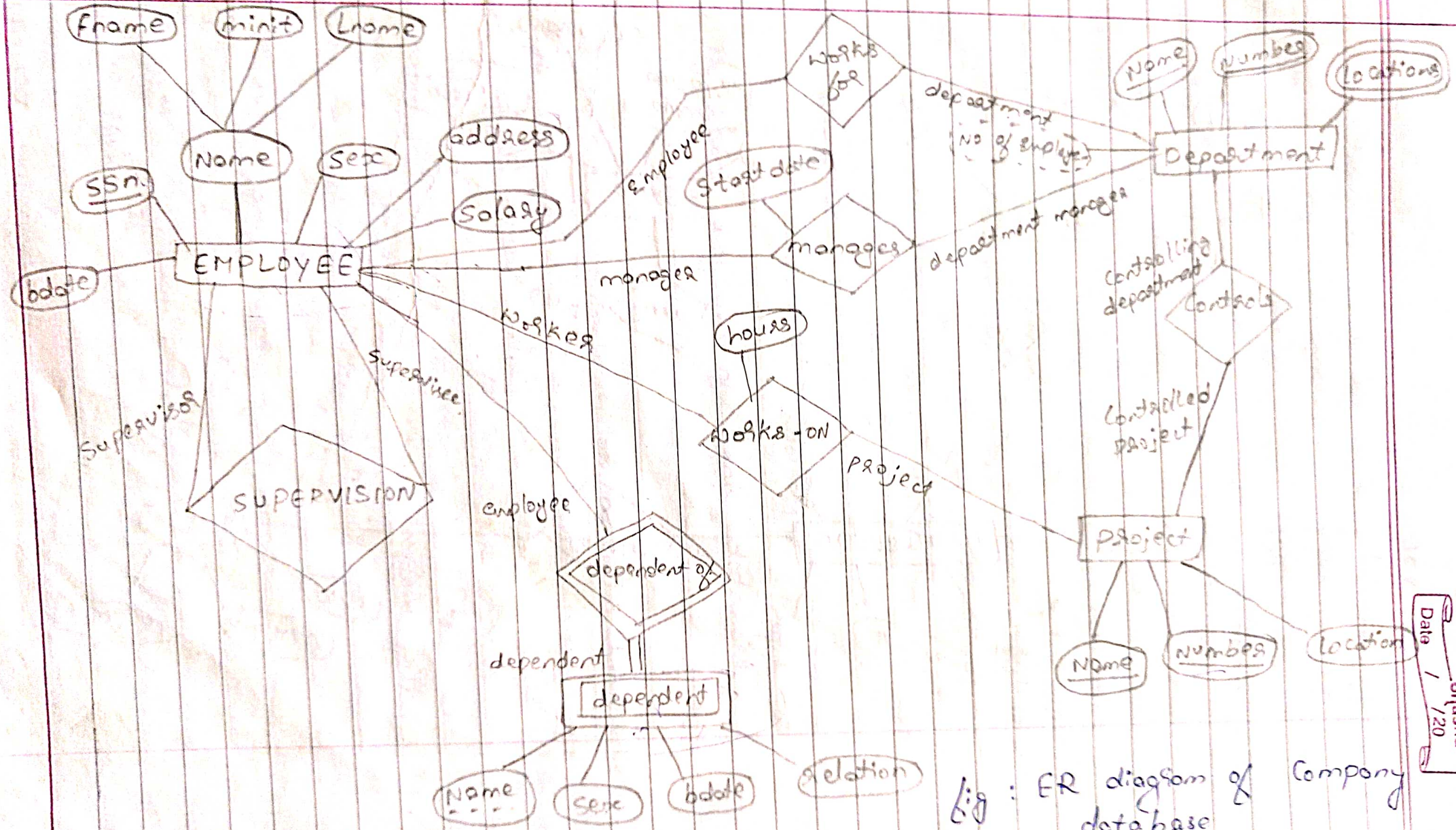


fig : ER diagram of Company database

Date / / 20 Bhaskar

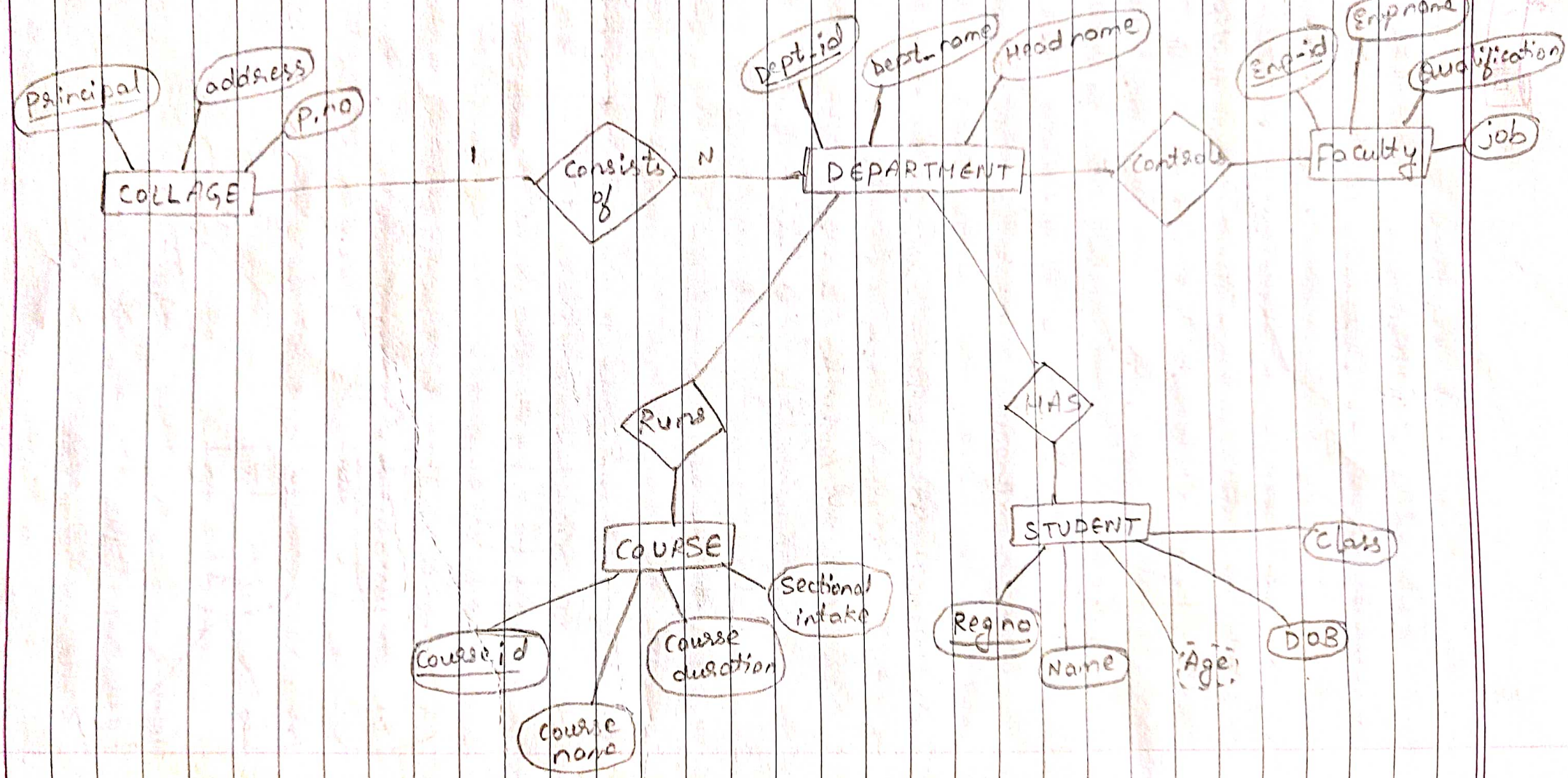


fig: ER diagram of collage database

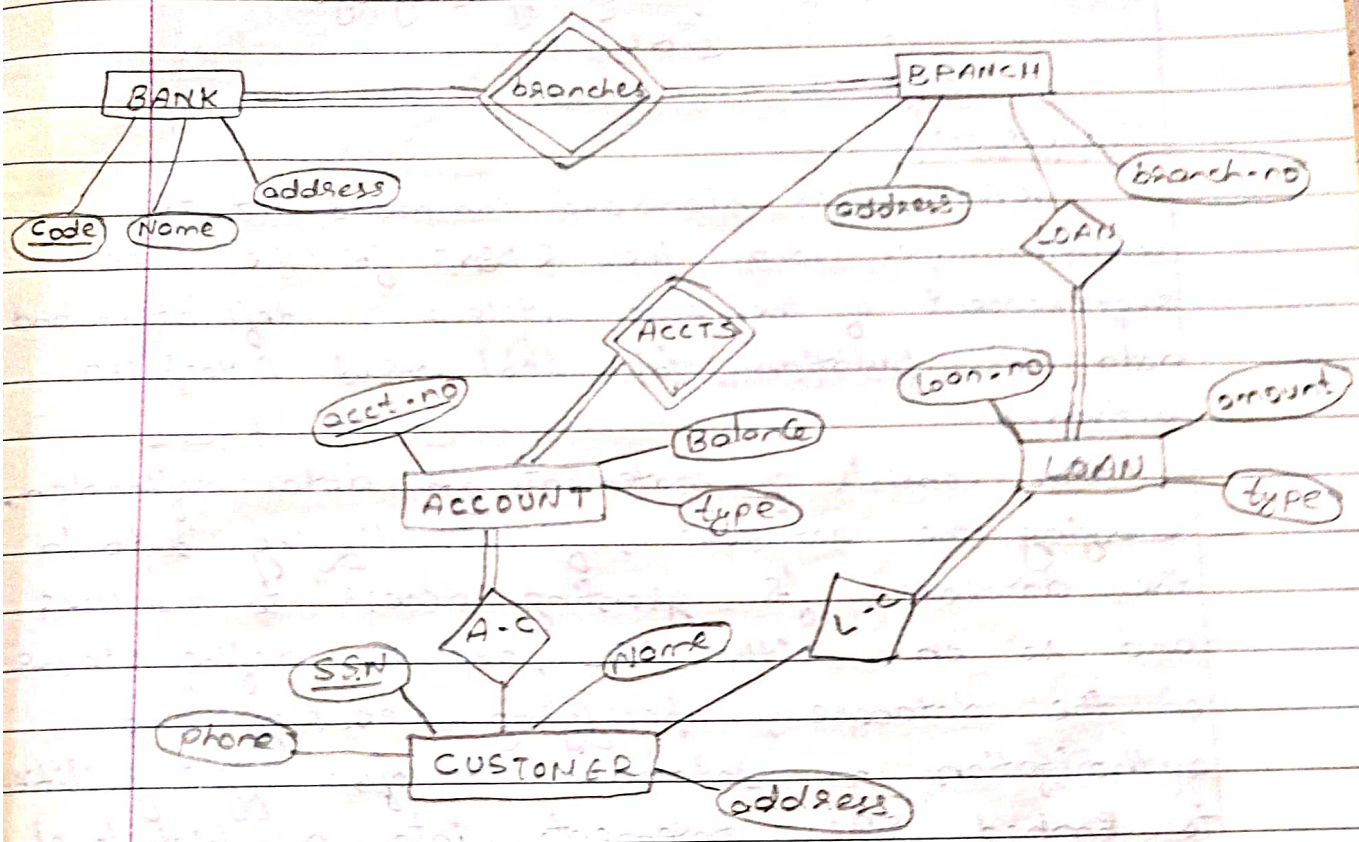


fig : ER diagram of bank database

Question

1. Describe the naming convention used for ER diagram
2. Define the following
   1) Entity 2) attribute 3) attribute values 4) composite 5) entity set
3. Design a ER diagram for company database
4. Explain create table command in SQL
5. How to create a view in SQL explain with an Eg.
6. What is diff b/w where & having clause
7. Consider the following table
   1) works (pname, cname, salary)
   2) located in (cname, city)
   3) Lives (pname, street, city)
   4) Manager (pname, mgr-name)

Relation is also known as table
In RDBMS attributes is known as column, tupple as rows

30/1/12

Structured Query Language [SQL]

CH-6

Introduction :

SQL offers standard command set to communicate with the RDBMS for the user requirement of the data retrieval, defination and data manupulation for real world Enterprise activities

SQL consists of statements for data definition, quesying the database, for modifying data in the database, for granting access to the user and so on, with SQL, one can define views on the database, specifying security and authorization on database it also offer facilities to Embed SQL statements into a high level programming language such as COBOL, C, C++, java & so on

Schema and Catalog concepts in SQL :

An SQL schema is identified by a schema name and includes authorization identifier to indicate the user who owns the schema, as well as descriptors for Each Element in the schema. Schema Elements includes tables, Constraints, views and other constraints that describe schema.

A schema is created via Create schema statement

Eg : following schema student creates a schema called Company. Company owned by the user JSMITH.

Syntax : [create schema <schemaname> authorization username]

create schema Company authorization JSMITH

< > → varying
[ ] → not compulsory
{ } → repeat

Date / /20

A catalog always contains a special schema called information schema which provides information on all the elements of the schema in the catalog to authorized users

Create table command:

Create table command is use to specify a new relation (table) by giving it a name, specifying its attributes & constraints.

Syntax:

CREATE TABLE <table name> (<column name>
<column type>
[<attribute constraints>],
{,<column name> <column type> [<attribute constraint>]}
[<table constraint> {,<table constraint>}])

Eg:

create table Employee (Empno number(6) primary key,
Empname varchar(10) not null,
Salary decimal (8,2));

6/2/12 <u>Datatypes in SQL</u>:

Different datatypes available are -

1) Numeric - It includes integer numbers of various sizes and real numbers of various precisions. They can be declared as

Decimal (i, j) or dec (i, j) or numeric (i, j)

where i → is the precision, total number of decimal digits

j → is the scale, number of digits after decimal point. Default value for scale is zero.

2) Character string : Datatype are Either fixed that is char(n) or character(n) or of varying length that is varchar(n) where n → is the maximum number of characters

3) Bit string : Datatype are Either fixed or varying length
bit(n) → fixed
bit varying (n) → varying

4) Date : Data type has 10 positions and its Components are year, month & day, in the form YYYY - mm - dd

5) Time : The time datatype has atleast 8 position with the component hours, min, sec in the form hh : mm : ss

Specifying constraints and default values in SQL :
1) Not null - A Constraint not null should always be specified for the primary key attribute of each relation as well as for any other attributes whose values ~~fo~~ are required not to be null.

7/2/12 2) Default - Default value can be defined for an attribute by appending the clause default to an attribute defination. Default value is included in a new tupple, if an Explicit value is not provided to the attribute.

3) primary key - primary key clause specifies one or more attribute that make up the primary key of a relation (table) should be unique and cannot hold null values

Date / /20

4) unique - The unique clause specifies that an attribute cannot hold duplicate values. It is an alternate key also called as secondary key.

5) Referential integrity constraint - It is specified b/w 2 relations (table). It says that a tupple that refers to another relations must refer to an existing tupple in that relation. It is specified ~~by~~ via ~~primary~~ foreign key clause

6) check - check clause is also called tuppled based constraint bcoz they apply to each tupple individually and are checked whenever a tupple is inserted or modified. check clause verifies that the attribute value is within the specified range.

Eg:

```
create table Emp( name varchar(10) not null,
                  ssn int primary key,
                  mgrssn ~~Dno~~ int default '555',
                  Contact no varchar(10) unique,
                  foreign key(Dno) reference dep(dnum)
                  Dno int check(Dno>0 and Dno<5));
```

Drop schema and drop table command:

If a whole schema is not needed any more drop schema command can be used

The drop behaviour options are -

1) Cascade
2) Restrict

Cascade - To remove the Company database schema and all its tables, domains and other Elements the Cascade option is used.
Eg : Drop schema Company Cascade ;
The schema will be removed Even if it Contains some Elements

Restrict - If the restrict option is choosen the schema is droped only if it has no Elements in it.
Eg : Drop schema Company restrict ;

Drop table Command :
If a base relation (table) within a schema is not needed any longer, the relation and its defination can be deleted using drop table Command.
There are 2 option Cascade and restrict -
If Restrict option is choosen the table is drop only if it not reference in any Constraint [foreign key defination in another relation] or views. Eg : Drop table dependent restrict ;
If cascade option is choosen all such Constraints and views that references the table are drop automatically from the schema along with the table itself.
Eg : Drop table dependent Cascade ;

Alter table Command :
Defination of the base table can be changed by using alter table command. The possible alter table action includes adding or droping a Column

changing the Column defination and adding or dropping table Constraints.

Employee

Name	Id	Address
Shashade	1	# 89, 4th cross, 5th main jayalakshmipuram
veena	2	# 201, 2nd cross, 4th main, Saraswathipuram

To add an attribute dno to the above table

Alter table Employee add dno int ;

To drop a column address

Alter table Employee drop address Cascade ;

General Syntax :

Alter table <table name> ADD <Column name> <column type>

Alter table <table name> MODIFY <column name> <column type>

Alter table <table name> DROP <Column name> <Constraint>

Basic Queries in SQL :

General form select - from - where clause is as follows

SELECT <attribute list>

FROM <table list>

WHERE <Condition>

where, <attribute list> - is a list of ~~list~~ attribute names whose values are to be retrieved by the Query.

<table list> - is a list of relation (table) names require to process the Query.

<Condition> - is a conditional expression that identifies the tupples to be retrieve by the Query

Eg : Retrieve the birthdate and address of the Employee whose name is John Smith.

Select Bdate, address
from Employee
where fname = 'John' and lname = 'Smith';

Dealing with ambiguous attribute names and renaming (aliasing):

In SQL the same name can be used for two or more attributes as long as the attributes are in different relations.

If this is the case we must use attribute name with the relation name (table) to prevent ambiguity this is done by prefixing the tablename to the attribute name and separating the two by period (dot)

~~for Each Employee~~ 1) Retrieve the name and address of all the Employee who works on research department.

Select fname, lname, address
from Employee ~~as e~~, department ~~as d~~
where dname = 'research' and
Employee.dno = department.dno

2) for Each Employee retrieve Employees fname and lname and fname, lname of his/her immediate supervisor.

Select E.fname, E.lname, S.fname, S.lname
from Employee as E, Employee as S
where E.superssn = S.ssn;

unspecified where clause and a use asterisk (*)

A missing where clause indicates there is no condition to select the tupple

Eg: Select all Employees of SSN

Select SSN
from Employee ;

To retrieve <u>all</u> attribute values of the selected tupple there is no need to list all the attribute names explicitly in SQL instead it can be specified using asterisk (*) which means all the attribute.

Eg : Retrieve all the information of all Employee who belong to deptno 5.

```
Select *
from Employee
where dno = 5 ;
```

Questions :

1. Write a note on SQL
2. Write a note on Constraints in SQL
3. Explain drop command with its options
4. How can ambiguity in attribute name from different table can be solved. Explain
5. Explain with an Example
   i) Like ii) b/w
6. Explain Complete SQL syntax
7. Explain nested & correlated queries
8. Explain different datatypes supported by SQL
9. Write the SQL queries for the following.
   i) List of name of people who work for the Company Wipro along with the cities they live in
   ii) find the people who work for the company Infosys with salary more than 50000
   iii) find the name of persons who live in & work in same city

8/2/22 Table as a set in the SQL:

SQL usually leads the table not as a set but as a multiset such that duplicate tupples can appear more than once in a table and also in the result of a query

If duplicate tupples has to be eliminated from the result of an SQL query the keyword DISTINCT should be used in the select clause, meaning only distinct tupples should remain in the result.

In general query with select distinct elements duplicate values whereas a query with select all does not

Eg: Select all salary values

```
Select salary          (or)    Select all salary
from Employee;                 from Employee;
```

Retrieve distinct salary values from the Emp table

```
Select distinct salary
from Employee;
```

union, intersection and set difference operation:

There is a set union operation [UNION], set difference [Except] and set intersection [INTERSECT] operation

☆ Substring Comparision and arithmetic operators:

Substring Comparision allows comparision condition on only part of character string using LIKE Comparision operator. partial strings are specified using 2 reserved characters that is

% → replaces arbitrary number of character [any no of characters]

Date / /20

_ → The underscore symbol replaces a single character.

Eg : Retrieve all the employees who reside in saraswathipuram

```
Select fname, lname
from Employee
where address LIKE '%saraswathipuram%'
```

Arithmetic operator : arithmetic operation are supported by SQL + → addition, - → sub, * → multiplication, / → division.

Eg : Show the resulting salary if every Employee is given a 10% raise.

```
Select fname, lname, 1.1 * salary
from Employee ;
```

100+10 = 110/100 = 1.1

Between : An other comparision operator that can be used in b/w to check the given attribute value to be in the specified range

Eg : Retrieve all Employee in dno 5 whose salary is b/w 30000 and 40000

```
Select *
from Employee
where dno = 5 and (salary BETWEEN 30000 and 40000)
```

Order by clause : SQL allows user to order the tupples in the resultent of a query by the values of attributes using order by clause

Ascending order is by default & can specify Asc and if many attributes have to be arranged in descending order then we have to explicitly specify desc after the attribute name in order by clause

List all the Emp working for dep 5 order by alphabetically by lname

ascending

```
select *
from Employee
where Dno = 5
order by lname
```

descending

```
select *
from Employee
where dno = 5
order by lname desc
```

Aggregate functions and grouping:

Some built in functions like count, sum, max, min and avg are used as aggregate functions.

The functions sum, max, min and avg are applied to a set & multiset of numeric values. These aggregate functions can be used in select clause & in the having clause of any query.

1) Count : returns the no of tuples & values as specified in the query.

2) min : returns the minimum value of the specified attribute in a relation.

3) max : returns the maximum value of the specified attribute in a relation.

4) Sum : returns the sum of values of specified attribute in a relation.

5) avg : returns the average of values of the specified attribute in a relation

Eg : Employee

Ename	age	salary	dno
abc	23	10000	1
efg	24	20000	2
hij	25	30000	1
klm	23	15000	2
nop	22	17000	1

Eg : Retrieve the no of tuples in the Employee table, also the min and max of age, sum of salary and avg of salary

```
select count(*), min(age), max(age),
        sum(salary), avg(salary)
from Employee ;
```

Group by clause :

The group by clause specifies grouping attributes which should also appear in select clause so that the result of grouping function is more understandable (that is the function applied on a grouped attribute appears along with the value of grouping attribute)

Here tuples with same value for specified attributes are grouped to perform some function usually groupby clause is used with aggregate functions.

Eg :

To retrieve sum of salaries of all Employees working for dno 1 and 2

```
select sum(salary), dno
from Employee
group by dno ;
```

having clause is same as where condition clause but it specifies condition on groups of tuples being selected rather than on individual tuple.

Eg : To retrieve sum of salaries of all Employees working for dno 1 and 2 and having sum of salary greater than 50000

```
select sum(salary), dno
```

from Employee
group by dno
having sum (salary) > 50000 ;

Complete SQL syntax :

A query in SQL can consist of upto six clauses but only select and from clauses are compulsory.

select <attribute and function list>
from <table list>
[where <condition>]
[group by <grouping attributes>]
[having <group condition>]
[order by <attribute list>]

→ The select clause list the attributes or functions to be retrieved

→ The from clause specifies the names of one or more tables from which the desired column values are to be retrieved.

→ The where clause specifies the condition for selection of tuples from the tables, including join conditions if needed

→ The group by specifies grouping attributes and mostly used with aggregate functions.

→ Having specifies condition on the groups being selected rather than on individual tuple.

→ order by specifies an order for displaying the result of the query.

<u>More complex SQL queries :</u>

Nested queries : Some queries require that

existing values in the database to fetched and then used in a comparision condition such queries can be formulated by using nested queries which are complete select from where blocks within where clause of another query. That other query is called outer query.

[Refer to the lab 8a]

Correlated nested queries :

Whenever a condition in the where clause of a nested query references some attribute of a relation declared in the outer query, the two queries are said to be correlated

The exists and not exists functions in SQL :

These functions in SQL is used to check whether the result of a correlated nested query is empty or not.

Exists returns true if there is atleast one tuple in the result of the query and returns false otherwise

Not exists returns true if there are no tuples in the result of the query and it returns false otherwise

Eg : for exists and correlated queries

Retrieve the name of each Employee who has a dependent with the same first name and same gender as the Employee

```
select E.fname, E.lname
from Employee as E
where exists (select *
              from dependent
```

where E.ssn = E.ssn and
E.gender = gender and
E.fname = dependent_name);

b) Retsieve the names of Employees who have no dependents

```
Select fname, lname
from Employee
where notexists (select *
                 from dependent
                 where ssn = essn);
```

Explicit sets and nulls in SQL :

It is possible to use an Explicit set of values in the "WHERE" clause rather than a nested query. Such a set is enclosed in parenthesis.

Eg : Retsieve the SSN of all Employees who work on project numbers 1, 2 or 3

```
Select DISTINCT ESSN
from works-on
where pno in (1, 2, 3);
```

SQL allows queries that check whether a value is NULL rather than using = or ≠ to compare an attribute to NULL, SQL uses IS or IS NOT.

Eg : Retsieve the names of the Employees who do have supervisors

```
Select fname, lname
from Employee
where super ssn is null;
```

Bhaskar
Date / /20

Renaming attributes :

It is possible to rename any attribute that appears in the result of a query by adding the qualifier as followed by the desired new name. Hence the as constraint can be used to alias both attribute and relation names and it can be used in both select and from clauses

Eg : To retrieve all Employees and their supervisors name

```
Select E.lname as Employee-name,
       S.lname as Supervisor-name
from Employee as E, Employee as S
where E.superssn = S.SSN;
```

Insert, delete and update statements in SQL

Insert Command :

Insert command is used to add a single tuple to a relation. We must specify the relation name and a list of values for the tuple.

The values should be listed in the same order in which the attributes were specified in the create table command

SYNTAX :

1) Insert into tablename values ('atr1', 'atr2' ...);

2) Insert into tablename (attr1, attr2 ...)
   values ('val1', 'val2', 'val3' ...);

Insert into tablename values ('fattr1', 'fattr2' ...);

Delete statement :

The delete statement can be used to remove one or more existing tuples from a given relation or table. A single delete statement can delete all rows of a table if where condition clause is not specified. If where clause is specified only selected tuples can be deleted

Syntax : delete from table_name
[where <condition>] ;

Eg : a) delete from Employee where id = '2' ; ⇒ only a row with id = 2 will be deleted

b) delete from Employee ; ⇒ all rows will be deleted

update statement :

The update statement can be used to modify attribute values of one or more selected tuples. At a time more than one column value can be updated using where condition clause the tuples to be updated can be choosen. An additional set clause specifies the attributes to be modified and their new values

Syntax :

update table_name
set column1 = value / expression,
[column2 = value / expression] - - - - -
where <condition> ;

views in SQL :

view in SQL is a single table derived

from one & more other tables. A view does not exist in physical form, it is considered as a virtual table i.e., only definition of view can be stored but not the view contents.

Specification of views in SQL :

The command to specify a view is create view. The view is given a name, a list of attribute names and a query to specify contents of view

Syntax :

create view viewname as Query the existing tables ;

Eg : create view v1 as select from student ;

If we don't need view anymore we can use <u>drop view</u> command to delete it

Eg : drop view v1 ;

view updation :

1) A view with a single defining table is updatable, if view attributes contain primary key of base relation

2) views defined on multiple tables using joins are generally not updatable

3) views defined using grouping and aggregate functions are not updatable

Advantages of creating a view :

→ views offer additional level of security on base tables by restricting access to pre determined set of rows and columns of that

→ views hide complexity since, views can be created by joining attributes of 2 or more tables.

→ views permit data to be presented in different position that of a base table.

2/10

Relational datamodel and Relational database constraints

CH = 4

Relational data model :

Relational model represents the database has a collections of relations
In relational model terminology a row is called a tuple, a column header is called an attribute and the table is called a relation. The Datatype describing the types of value that can appear in Each column is called a domain

Domains : Domain 'D' is a set of atomic values by atomic we mean all the values should be of same type

A name will be specified to Each and Every domain and is called as a domain names

Relational Schema :

Relational schema is specified as R and is denoted by R (A1, A2 ..... An), is made up of relations name R and list of attributes A1, A2 ... An. Each attributes 'Ai' belongs to the corresponding domain 'Di'. D is called the domain of Ai and is denoted by Dom (Ai).

A relation schema is used to describe a relation ; R is called the name of this relation The degree of relation. The relational schema remains constant for a long time but the data stored inside a relation changes with respect to time. At any instance of time, the available data inside a relation is called as database instance.

The degree of relation is the number of attributes of a relations
cardinality ratio of a relation refers to the number of rows presents in a relation.

Relation : A relation is denoted by r. Each and Every relations will be having a relational schem 'R', then relationed can also be denoted by r(R).

In general a relation is a collections of set of "n-tuple" and is denoted by

$$r(R) = \{ t_1, t_2 \ldots t_n \}$$

where Each and Every tuple is an ordered list of n values so $t = \langle v_1, v_2, v_3 \ldots v_n \rangle$

14/2/18 Relational constraints : It specifies the value of an attributes. It includes datatype for the field, field length assertion and additional built in constraints like primary key, not null Etc. All these put together specifies the value that can be stored for an attributes

Key constraints :

a) Super key : A relation is defined as a set of tupples by defination all Elements of a set are distinct hence all tupples in a relation must also be distinct. This means that no 2 tupples can have the same combination of values for all these attributes. The "subset of attributes" of a relation schema 'R' with the property that no 2 tupples can have the same combinations of values is called as a super key. It is denoted by 'SK'. In other words 'SK' should satisy the

following condition that is $t_1[SK] \neq t_2[SK]$

b) Minimal super key : A key of relations 'r' is a super key with additional property that removing any attributes 'A' from 'SK' lives a set of attributes which is not a super key of the relation 'r'. Hence a key can also be called as minimal super key.

c) Candidate key : A relation schema can have more than one key fields then each of these key field is called as candidate key.

Eg : CAR

Regno	model	Engine no	Date of manufacture	...

In the above schema registration no and Engine no are the key field and hence they are called candidate keys.

d) primary key : primary key is a key field with has unique values throughout the table. If we have several candidate key we need to identify one candidate key as primary key use the convention that the attributes that form the primary key of a relation schema is underlined

e) not null : It is an additional constraints for a field the constraints not null specified for an attributes make compulsory to enter a value for that fields.

Integrity constraints : There are 2 integrity constraints they are

1) Entity integrity or primary key integrity
2) Referential integrity or foreign key integrity

1) Entity integrity : It states that no primary key

can be null. This is booz the primary key value is used to distinguish one Entity from another Entity in a relation. Having null value for the primary key field may cause problem to distinguish b/w rows

12/2/12 2) Referential integrity : It is specified b/w 2 relations. The referential integrity constraints says that a tupple that refers to another relation, must refers to an Existing tupples in that relations

Eg :

pk

Sno	Sname	Status	city
S1	x	20	MYS
S2	Y	30	BAN

P

PK

Pno	Pname	color	weight
P1	A	red	20
P2	B	Blue	30

SP

FK FK

Sno	Pno	qty
S1	P1	200.
S1	P2	300
S2	P2	200

pk → primary key
fk → foreign key

In the above example 'Sno' attributes in 'sp' relation must match with any one tupple for 'Sno' attribute in 's' relation

The above concept is the foreign key concept of the relation.

The condition for a foreign key b/w 2 relation schema can be specified as below

1) The attributes in 'fk' have the same domain as the primary key attribute 'pk' of a relation schema R2

2) A value of 'fk' in tuple 'T1' in relation 'R1' must occur as a value of 'pk' for some tuple 'T2' in relation 'R2' that is $t_1(fk) = t_2(pk)$

update operations : List and Explain different update operation on a relation (or)
2) Explain the Constraints applicable on Each of the update operations

There are 3 basic update operations on relations they are
1) Insert 2) Delete 3) Modify operation.

1) Insert operation :

The insert operations provide a list of attribute value for a new tupple 'T' that is to be inserted into a relation 'R'
It is used to insert one or more tuples to a relation
Always a new records will be inserted at the End of the relation.

Syntax :
insert into <table name> values <list of values> ; (or)
insert into <table name> <attributes list> values <list of values> ;

The Constraints that are checked during insert operations are
a) domain Constraints : To check the voliations of datatype, length and assertion constraints
b) Entity integrity : If the field is a primary key field then the system checks for unique value for the field and also checks for whether null value is Entered
c) Referential integrity : It is voliated if the 'FK' of relation 'r1' refers to a tuples in relation 'R2' that does not exists

/2/12 2) Delete : It is used to delete one or more tuples in a relation. we need to specify a condition to delete a particular tuples in the relation

Syntax :
Delete from <table name> where <condition> ;

if we donot specify any condition then all the rows in the relations will be deleted

The delete operation can violate referential integrity that is if the tuples being deleted is referenced by a foreign key of another relations

3) Modify or update operation : It is used to update the value of one or more attributes of a tuple in a relation condition is satisfied to limit the update operation to some specified tuple ; If no condition is specified then updation will be executed for all the existing tuples of the relations

The update operation checks domain constraints, Entity integrity and referential integrity.

Syntax :

update <tablename> set <field name = value>
where <condition> ;

Bhaskar Date / /20

18/2/12

The Relational Algebra

CH - 5

Algebraic ~~rela~~ operations on relations is known as "relational algebra operation."

Each of these operation takes one or more relation as its operands and produces another relation as its relation

The relational algebraic operations are classified into 2 types

1) Traditional set of operations.
2) Specific (special) set of operations

1) Traditional set of operations : It is also called as "set theoretic operation". It contains algebraic operations that are the standard mathematical operations on sets.

The traditional set of operations are -

a) union
b) intersect
c) Set difference (minus)
d) ~~Ext~~ Extended cartiesian product.

21/2/12 a) union : It is denoted by the symbol 'U'. If R and S are two relations, then union of these two relations is represented by "R union S" or "R U S".

The result is a relation that includes all tuples that are Either in 'R' or 'S' or both. Duplicate tuples are Eliminated from the resultant relations.

S	Sno	Sname	Status	city
	S1	X	30	Mys
	S2	Y	20	Ban
	S3	Z	30	Mys
	S4	P	40	Ban

Sp	Sno	Pno	city
	S1	P1	200
	S1	P2	500
	S2	P1	600
	S3	P2	400

Query :

Select Sno from S where city = 'Mys'
union
Select Sno from SP where pno = 'p1';

S1 S1 o/p

S3 S2

Sno
S1
S2
S3

Intersection : It is denoted by the symbol '∩'. If 'R' and 'S' are two relations then intersection of these two relation is represented by "R intersect S" or "R ∩ S".

The result of intersect operation is a relation which includes all the tuples that are in both 'R' and 'S'.

Query :

Select Sno from S where city = 'Mys'
intersect
Select Sno from SP where pno = 'p1';

S1 S1 o/p

S3 S2

Sno
S1

Set difference [minus] : It is denoted by the symbol "−". If R and S are two relations then difference b/w them is denoted by 'R minus S' or 'R − S'

The result of minus operation is a relation that includes all tuples that are presents only in R but not is S.

Eg : 1) Student − instructor
2) Instructor − student.

1)

Bhaskar Date / /20

Student

Fn	Ln
Susan	Yoo
Ramesh	Shah
Johnny	Kower
Barbara	Jones
Amy	Ford
Jimmy	Wong
Ernest	Gillent

Instructor

Fname	Lname
John	Smith
Ricardo	Boune
Susan	Yao
Francies	Johnson
Ramesh	Shah

Student - instructor

Johny	Kower
barbara	James
Amy	ford
Jimmy	Wong
Ernest	Gillent

Instructor - student

John	Smith
Ricardo	Bourne
Francies	Johnson

4) Extended cartesian products : It is denoted by the symbol 'x'. If R and S are two relations then cartesian product b/w these two is denoted by '<u>R times S</u>' or '<u>R x S</u>'.

The result of this operation is a relation which is a collections of all possible combinations tuples represent in 'R' concatenated with all tuples of 'S' relation.

4/2/12 Eg1:

R

Sno
S1
S3

S

Sno
S1
S2

R x S =

Sno	Sno
S1	S1
S1	S2
S3	S1
S3	S2

Eg2 :

R

id	name
1	x
2	Y

S

Branch
CS
mech

R x S :

id	name	branch
1	x	CS
1	x	mech
2	Y	CS
2	Y	mech

Special Relational algebra operations : These set of operations are not present in mathematical algebraic set of operations. The special relational operations are applied only on relations.

The special set of operators include

1) select
2) project
3) Join
4) Division

1) <u>Select [σ]</u> : The select operation is used to select subset of tuples from the relation that specifies a selection Condition.

It is denoted by $\sigma_{\langle selection\ Condition \rangle}(R)$ Table name

Eg1: Get student information from "stud" relation who are studying in CS branch

```
Select *
from stud
where branch = 'CS';
```
SQL Format

$\sigma_{branch = 'CS'}(stud)$ → Special relation format

Eg 2: $\sigma_{\langle dno = 4 \text{ AND } salary > 2500 \rangle \text{ OR } \langle Dno = 5 \text{ AND } Salary > 3000 \rangle}(Emp)$

2) <u>project</u> (π or pi) : The select operation selects some of the rows from the table while discarding other rows.

The project operation on the other hand select certain Columns from the table and discards the other Column

Syntax :

$\pi_{\langle attribute\ list\ (or)\ list\ of\ columns \rangle}(R)$

Eg : Select Id and name field from stud relation

$\pi_{\langle Id, name \rangle}(stud)$ → algebraic expression

Select Id, name from stud; → SQL Statement

3) <u>Join ($\bowtie$)</u> : Join operation is used to combine related tuples from two relations into a single tuples (or)

Join operations combine cartesian products and select operations.

It is denoted by $\bowtie$

~~Eg~~ : If R and S are 2 relations on which we are applying join operations with attributes are of $R(A1, A2 \ldots An)$ and $S(B1, B2 \ldots Bn)$. The resultent relations will be having 'm+n' attributes. The general form of a join operation is denoted by

$$R \bowtie_{\langle \text{join Condition} \rangle} S$$

Eg :

R =

a	b	c
1	2	3
4	5	6
7	8	9

S =

d	e
3	6
7	8

Rxs =

a	b	c	d	e
1	2	3	3	6
1	2	3	7	8
4	5	6	3	6
4	5	6	7	8
7	8	9	3	6
7	8	9	7	8

then

$$R \bowtie_{\langle b>d \rangle} S$$

a	b	c	d	e
4	5	6	3	6
7	8	9	3	6
7	8	9	7	8

The Condition is any one of the Comparision operator like $=, >, >=, <, <=, \neq$. The join operations with a general join Condition is called as theta join (or) θ.

25/2/12 Equi $\overset{(=)}{}$ join : produces all combinations of tuples from 'R' and 'S' that satisfies a join conditions with only Equality Comparision

R

a	b	c
1	2	3
4	5	6
7	8	9

S

d	e
4	5
8	6
1	3

$R \bowtie S$
$(a+b)=(d+e)$

a	b	c	d	e
4	5	6	4	5

$R \times S =$

a	b	c	d	e
1	2	3	4	5
1	2	3	8	6
1	2	3	1	3
4	5	6	4	5
4	5	6	8	6
4	5	6	1	3
7	8	9	4	5
7	8	9	8	6
7	8	9	1	3

$R \bowtie S$
$c+d=10$

a	b	c	d	e
4	5	6	4	5
7	8	9	1	3

$R \bowtie S$
$b=d$

a	b	c	d	e
7	8	9	8	6

Natural join (*) : It is basically ~~has~~ as Equi join followed by removal of duplicate attributes. In natural join no Condition is specified

Eg :

If we want we can do cartesian product before natural join

Bhaskar Date / /20

let R

a	b	c
1	2	3
4	5	6
7	8	9

let S

b	c	d
2	3	6
5	6	9
8	9	1

$R \overset{(*)}{\bowtie} S$

a	b	c	d
1	2	3	6
4	5	6	9
7	8	9	1

It is denoted by (*)

'b' and 'c' are the attributes which are present both in relation 'R' as well as 'S'. Then the natural join conditions will be R.b = S.b, R.c = S.c

Sequence of operation and renames of operations:

Apply several relational algebra operations one after the other. Either we can write the operation as a single relational algebra expression by nesting the operations or we can apply one operation at a time and create intermediate result relation.

Eg 1: Select salary, name of all Employees who are working in CS department

SQL statement : Select salary, name
from Employee
where dept = 'CS';

Relational algebra : 1) $x \leftarrow \sigma_{branch = 'CS'}(Emp)$

2) $result \leftarrow \pi_{name, salary}(x)$

where 'x' is the intermediate resultant relation. Further sequence of operations can be join together into a single statement.

Eg 2 : To retrieve the fname, Lname and salary of all Employees who work in deptno 5

$$\pi_{fname,\ lname,\ salary}\left(\sigma_{dno=5}(Emp)\right)$$

OR

1) $x \leftarrow \sigma_{dno=5}(Emp)$

2) $Result \leftarrow \pi_{fname,\ lname,\ salary}(x)$

27/2/12 Rename operator (ρ (rho)) : which can rename Either the relation name or the attribute names or both in a manner similar to the way we define select and project.

The general rename operation is denoted by

$$\rho_S(B1, B2 \ldots Bn)(R)$$

(OR)

$$\rho(B1, B2 \ldots Bn)(R)$$

where the symbol ρ(rho) is used to denote the rename operator. S is the relation name and B1, B2 ... Bn are the new attributes name.

A Complete set of relational algebra operation :

what do you mean by a Complete set of relation algebra operation ?

The following set of relation ~~of~~ algebraic operation is called as Complete set of relational algebraic operation

$$\{\cup, \pi, -, \times, \sigma\}$$

That is any of other relational algebra operation can be expressed as a sequence of operation from this set. The operator Eliminated are

$$\{\cap, \bowtie, \div\}$$

Eg : The intersect operations can be Expressed using union and minus as given below

Date / /20

$R \cap S = (R \cup S) - (R-S) \cup (S-R)$

let $R = \{1, 2, 3, 4, 5\}$

let $S = \{4, 5, 7, 9\}$

$R \cap S = \{1, 2, 3, 4, 5, 7, 9\} - \{1, 2, 3\} \cup \{7, 9\}$

$= \{1, 2, 3, 4, 5, 7, 9\} - \{1, 2, 3, 7, 9\}$

$= \{4, 5\}$

Strictly speaking intersect operation is not required but it is used to reduce the complex operation.

Similarly join operations can be expressed as cartesian products followed by select ~~condition~~ operation

$$R \bowtie_{condition} S = \sigma_{condition}(R \times S)$$

* **Division** : The division operation is a special kind of operation it is denoted by $\div$. If R and S are 2 relations then the division of 'R by S' is denoted by '$R \div S$'

Division operation can be expressed as a sequence of operations using π, $\times$ and $-$

$T_1 \leftarrow \pi_y(R)$

$T_2 \leftarrow \pi_y((S \times T_1) - R)$

$T \leftarrow T_1 - T_2$

where y is the list of attributes of resultant relation. In other words the list of attributes which are present in first relation but not in second relation

let R

A	B	C	D
a	b	t	g
c	d	x	a
c	g	r	t
c	d	p	q
s	s	s	y
y	s	x	a
y	s	t	c

let S

C	D
x	a
p	q

$R \div S =$

A	B
c	d
y	s

x0
cd
ys
sy

P2
cd
ys

x0 & P2 having (cd, ys)

Eg 2 :

m	n
c	D
4	3
8	1
	7

C	D
4	3
4	1
4	7
8	3
8	1
8	7

Division is opposite to the Cartiesion product

m
c
4
8

$O \div m =$

n
D
3
1
7

28/2/12 <u>Additional Relational operation</u> :

Some database request cannot be executed with the help of basic relational algebraic operation. These are executed with the help of additional relational operators. Commonly used additional relational operators are aggregate functions and grouping

Eg: To compute the total salary of Employee or to Compute total number of Employee etc from Emp relation, cannot be executed using basic relational operations. Hence aggregate functions like sum, count, min, max and avg are used

Another Common type of request involves grouping the tuples in a relation by the value of some of their attributes and then applying an aggregating functions independently to Each group

Bhaskar
Date / /20

Eg : Group Employee tuples by dno so that each group includes the tuples for Employees working in the same dept.

We can define an aggregate function operation using the symbol $\mathcal{F}$ (~~pro~~ Script F)

$\langle$grouping attribute$\rangle$ $\mathcal{F}$ $\langle$function list$\rangle$ (R)

where grouping attributes is a list of attributes of the relation specified in R and function list is a list of {(function)(attribute)} pair. In each such pair function is one of the allowed function such as sum, min, max, avg, count and attribute is an attribute of the relation specified by R.

The resulting relation as the grouping attribute plus one attributes for each elements in the function list

2/3/12 Outer join : The outer join operation is an extension to join operation where in it will take care to add non-matching tuples and their corresponding attributes in resulting relation

Eg : R1

Name	job
a	stenographer
b	engineer
c	doctor
d	lecturer
e	manager

R2

Name	skill set
a	Singing
b	dancing
e	anchoring
f	writing
g	swimming
h	reading
i	painting.

i) left outer join ($\bowtie$) : The left outer join operation denoted by R1 $\bowtie$ R2

yields new relation comprising all tuples of relation R1 (left) and fill up the null values for attributes of R2 for every non matching tuples of R1 in R2

Eg : R1 ⟕ R2

Name	job	skill set
a	Stenographer	singing
b	Eng	dancing
c	doctor	NULL
d	lecturer	NULL
e	manager	anchoring

2) Right Outer join (⟖) : The right outer join operation is denoted by R1 ⟖ R2 yields new relation comprising all tuples of relation R2 (right) and fillup the null values for attributes of R1 for every non matching tuples of R2 in R1.

Eg : R1 ⟖ R2

Name	job	Skill set
a	Steno	Singing
b	Eng	dancing
e	manager	anchoring
f	Null	writing
g	Null	Swimming
h	Null	reading
i	Null	painting

Full outer join (R1 ⟗ R2) : The full outer join operation denoted by R1 ⟗ R2 yields new relation comprising of all tuples of relation R1 as well as R2 and fillup the null values for respective attributes of R1 and R2. when Either of them finds no matching tuples

Eg : R1 ⋈ R2

Name	Job	Skill set
a	stenо	singing
b	eng	dancing
c	doctor	null
d	manager	null
e	~~null~~ lecturer	anchoring
f	Null	writing
g	null	swimming
h	Null	Reading
i	Null	painting.

3/3/12

Introduction to transaction processing Concepts and theories CH-8

What is <u>transaction</u>? Series of operations is known as transaction

The concept of transaction a logic unit of database processing. That includes one or more data access operation. The data access operations are insertion, deletion, updation Etc.

Transaction processing System :

Write a note on TPS :

Transaction processing System are with large database and hundreds of concurrent user, That are executing a database transaction.

Eg : Reservation systems, banking system, credit and processing, stock markets.

They require high availaibility and fast response time for hundreds of current users

The transaction processing support ACID properties

A - Atomicity

C - Consistancy

I - Isolation

D - Durability.

Write a note on Concurrent transaction.

Compare single user system with multiuser system.

Define current transaction processing.

What do you mean by parallel processing.

Classifying a database system is according to number of user, who can use the systems concurrently [at the same time]

If the system supports only one user to operate on its database then that system is called as single user system

Date / /20

Eg : MicroComputer

If the system supports more than one user to operates on its database then that system is called as multiuser system.

Eg : reservation system

Multiprogramming : It allows the computer to execute more than one program concurrently. Eventhough the system has only one micro processor the OS which supports multiprogramming concepts [like linux, unix, windows etc] can execute more than one program concurrently

1/3/12 Multiprogrammed OS executes some commands from one process and suspends that process and execute command from other process and so on. A process is resumed at the point where it was suspended whenever it gets its terms to use the cpu again. Hence concurrent execution of process is actually interleaved processing. figure shows 2 process a and b executing concurrently in an interleaved fashion. (fig a)

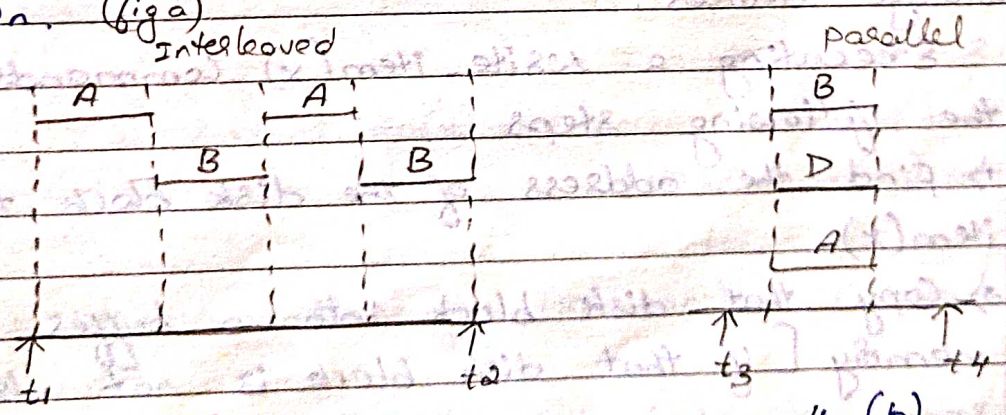


fig (a) fig (b)

If the computer system has multiple h/w processors (cpu's) is called parallel processing, It can execute more than one program at the same instance of time. The above diagram (fig b) is an example for parallel processing system where 2 programs are

executed by a different microprograms parallely

Transaction read operation :

If the database operation in the transaction do not update the database, but only retrieves the data then the transaction is called read only transaction.

The read transaction is written as "read_item(x)". Executing read_item(x) includes the following steps

1) Find the address of the disk block that contain item(x)

2) Copy that disk block into a buffer in main memory [if that disk block is not already in some main memory buffer]

3) Copy item(x) from the buffer to the program variable named 'x'

Write operation :

The write transaction is written as write_item(x) writes a value of program variable into secondary storage devices.

Executing a write_item(x) commands include the following steps

1) Find the address of the disk block that contain item(x)

2) Copy that disk block into a buffer in main memory [if that disk block is not already in some main memory buffer]

3) Copy item(x) from the program variable named 'x' into ~~the~~ its correct location in the buffer

4) Store the updated block from the buffer back to disk [Either immediately or at some later point in time]

Date / /20

why Concurrency Control is needed ?
1) write a note on Concurrency Control is needed.
2) List and Explain the problem that occurs bcoz of Concurrent transaction.

Several problems occurs when Concurrent transaction are executed in unconditional manner. The problems are :

1) lost update problem
2) Dirty read problem
3) Incorrect summary problem
4) Repeated read problem.

13/12 1) Lost update problem : This problem occurs when 2 transaction access the same data item and having these operations interleaved, in such a way that the final value of the data item is inconsistent.

Eg : let $x = 80$, $m = 5$, $n = 4$

T1	T2
80 → Read_item(x)	
75 → $x = x - m$;	
	Read_item(x) ← 80
	$x = x + n$ ← 84
75 → write_item(x)	
	write_item(x) ← 84

Here the updation of transaction T1 is lost.

2) Dirty ~~(or)~~ read problem (or) temporary updation problem : This problem occurs when one transaction update the item and then fails. The second transaction access the same item after the completions of write operation of first transaction

and before the first transaction failure. This causes inconsistent result. This is because the first transaction revokes (undo) all its operation after its failure.

Eg : let $x = 80$, $m = 5$, $n = 4$.

T_1	T_2
Read_item(x);	
x = x - m ;	
write_item(x) ;	
	Read_item(x);
	x = x + n ;
	write_item(x);

T_1 failure and revokes T_1.

3) Incorrect summary problem : If one transaction is calculating an aggregate summary functions on a number of records, while other transaction is updating some of these record. The aggregate function may calculate some value before they are updated and other after they are updated.

Eg : let T_1 & T_2 are transactions.

4) Repeated read problem :

T_1	T_2
	Sum = 0
	read_item(A)
	Sum = Sum + A
	⋮
Read_item(x)	
x = x - N ;	
write_item(x);	
	Read_item(x)
	Sum = Sum + x ;

read_item (x)
Sum = Sum + Y;

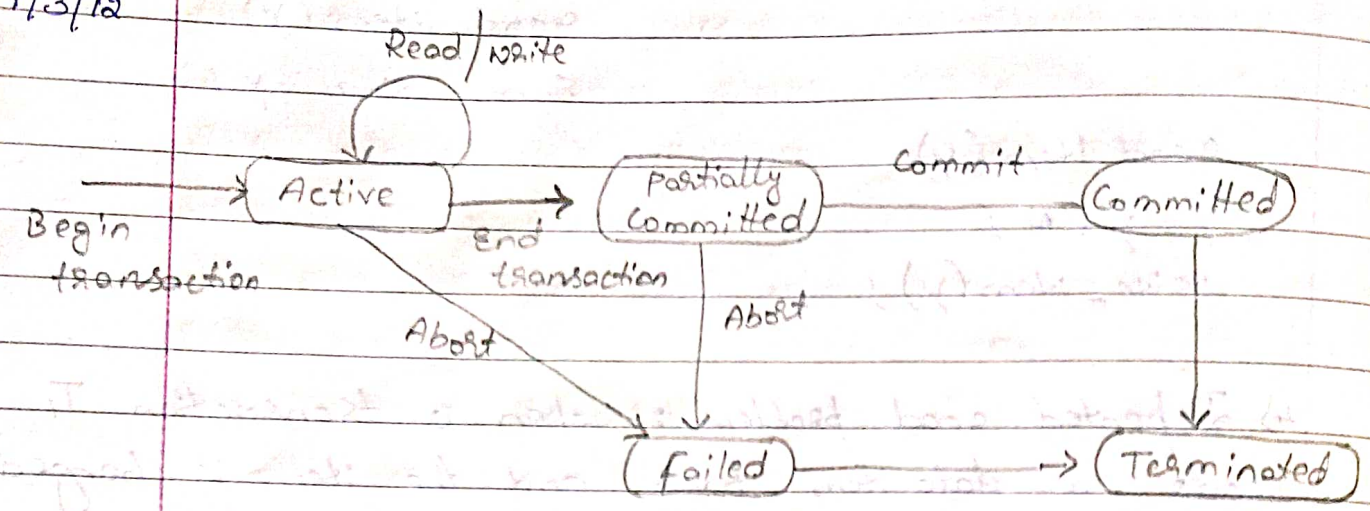
read_item (Y)
Y = y + N;
write_item (y);

4) Repeated read problem : when a transaction T_1 reads a data item twice and the item is changed by another transaction T_2 b/w the 2 read operations of T_1 inconsistent result occurs at transaction T_1.

* why recovery is needed :
Explain different failures during transaction processing
Discuss why recovery is needed.
when a transaction fails recovery is needed beoz of the following problem :
1) Computer failure :- A h/w, s/w or network Error occurs in the Computer System during transaction execution
2) Transaction Error :- Some operations in the transaction may cause it to fail, such as integer overflow or division by zero.
3) Concurrency Control Enforcement :- The Concurrency Control may decide to abort (cancel) the transaction to be started later, if otherwise causes deadlock.
4) Disk failure :- Some disk block may loose there data because of read/write head crash.
5) physical problem :- This refers to the list of problems that include fire, theft etc.

* with a state transition diagram illustrate the states of transaction Execution:

9/3/12



A transaction is an automic unit of work that is Either Completed in its Entirity or not done at all.

For recovery purpose the system needs to keep track of when the transaction starts, terminates and committ or abort hence the recovery manager keep track of the following operations

1) Begin transaction : This marks the beginning of transaction Execution

2) Read/write : These specifies the read/write operations on the database items that are executed as part of a transaction

3) End transaction : This specifies that read/write operation have ended and marks the End of transaction execution at this point the transaction manager need to check whether the changes introduced by the transaction can be permanently applied to the database or the transaction has to be aborted

4) partially committed : when the transaction ends it moves to partially committed state

5) Committ transactions : This signals (indicates) the successfull End of transaction

6) Roll back or abort : This indicates that the transaction Ended unsuccessfully so

that the changes applied to the database must be undone

The system log :

To recover from failures that affects the transaction, the system maintain a log [DBMS journal] to keep track of all transaction operation that affects the ~~do~~ values of database item. The log is kept on disk so it is not affected by any type of failure except for disk or catastrophic. In addition the log is periodically backed up in a tape devices

The following are the list of entries called "log records" that are written to log table. In this entry 'T' refers to an unique transaction Id generated automatically by the system

1) [Start_Transaction, T] - Indicates that transaction with Id T as started

2) [~~Read~~ write_item, T, x, oldvalue, new value] - Indicates that transaction 'T' has changed the value of dataitem 'x' from its oldvalue to a newvalue

3) [Read_item, T, x] - indicates that transaction 'T' has read the value of database item x

4) [commit, T] - indicates that transaction 'T' has completed successfully and it affects can be transfer to secondary memory.

5) [abort, T] - indicates that transaction 'T' has failed

10/3/12 what do you mean by commit point ?

A transaction 'T' reaches its commit point when all its operation that access the database

has been executed successfully and the effect of all the transaction operation are recorded in the log table beyond the commit point the transaction is said to be committed

In any system failure, transaction manager looks for log table for all transaction 'T' that has both [start - transaction, T], [commit, T]. If both are available then transaction is said to be completed.

If [start - transaction, T] is available and [commit, T] is not available then the transaction manager decides that the transaction is not completed and hence roll back all the transaction operations

What are the <u>desirable properties of a transaction</u>? Explain.

Any transaction should causes 4 properties they are called as "ACID" properties.

The ACID properties are

1) Atomicity [A]
2) Consistency [C]
3) Isolation [I]
4) Durability [D]

Atomicity : A transaction is an atomic unit of work which is completed entirely or not done at all

Consistency : A transaction is consistency preserving if its complete execution takes the database from one consistent state to another consistent state

Isolation : A transaction should appear as is being executed in isolation from other transaction.

Bhaskar
Date / /20

In other words the Execution of one transaction should not effect any other transaction executing concurrently

Durability: The changes applied to the database by a transaction must persist in the database. These changes must not be lost bcoz of any failure

10/3/12

Concurrency Control Techniques

CH-9

* Locking techniques for Concurrency Control :

1) How do you control concurrency by locking techniques?

one of the popular techniques used to control Concurrent Execution of transaction is based on the Concept of locking the data item

A lock is a variable associated a dataitem. That describes the status of the item with respect to possible operation that can be applied to it. Generally there is one lock for Each dataitem in database

* Types of lock :

Several types of locks are used in Concurrency Control

1) Binary lock
2) Shared / Exclusive lock (or) read/write lock

** 1) Binary lock : A binary lock can have 2 states or values they are locked and unlocked (1 or 0). A distinct lock is associated with Each data_item. when a data_item is locked the status indicator holds a value 1. The status indicator holds 0 if the data_item is unlocked.

Two operation lock_item(x) and unlock_item(x) are used with binary locking.

13/3/12 → lock_item(x) : Now the data_item(x) is locked and it can be used by only by the transaction by which it is locked.

→ unlocked_item(x) : Now the data_item(x) is free and it can be used by any transaction by locking it.

The binary locking scheme should obey the following rules :

- A transaction 'T' must issue the operation lock_item(x) before any read_item(x) or write_item(x) operations are performed in 'T'
- A transaction 'T' must issue the operation unlock_item(x) after any all read_item(x) and write_item(x) operations are completed in 'T'
- A transaction 'T' will not issue a lock_item(x) operation if it already holds the lock on item(x)
- A transaction 'T' will not issue an unlock_item(x) operation unless it already holds the lock on item(x)

2) Shared or Exclusive lock : This locking system allows several transactions for access the same data_item(x) for reading purpose only however if the transaction is write on item(x) then the data_items can be locked only by one transaction

This locking method has 3 locking operations they are
→ read_lock(x)
→ write_lock(x)
→ unlock(x)

→ read_lock(x) : It is also called shared lock because other transactions are allowed to read the item

→ write_lock(x) : It is also called Exclusive lock because a single transactions Exclusively holds the lock on the item

→ unlock(x) : when unlock(x) is applied on a data_item the lock held by the data_item will be released

when we used the shared / Exclusive locking scheme the system must Ensure the following rules -

- A transaction 'T' must issue the operation write_lock(x) or read_lock(x) before any read_item(x) operation is performed in 'T'.
- A transaction 'T' must issue the operation write_lock(x) before any write_item(x) operation is performed in 'T'
- A transaction 'T' will not issue a read_lock(x) operation if it already holds a read_lock or a write_lock on item(x)
- A transaction 'T' will not issue a write_lock(x) operation if it already holds a read_lock or a write_lock on item(x)

Conversions of lock:

Sometimes it is desirable to convert the lock from one locked state to another

Eg: It is possible for a transaction 'T' to issue a read_lock(x) and then later on to upgrade the lock by issuing a write_lock(x) operation. This is called "upgradation" of lock.

It is also possible for a transaction 'T' to issue a write_lock(x) and then downgrade to read_lock(x) without any restrictions [degradation]

<u>Guaranteeing Serializability by 2 phase locking</u>:

What do you mean by 2 phase locking (2PL)? Explain

A transaction 'T' is said to follow the 2 phase locking protocol, if all locking operation [read_lock or write_lock] if all locking operation precede the unlock operation in the transaction such a transactions can be divided into 2 phases

1) Expanding or growing [first phase]
2) Shrinking [2 phase]

3/12 1) Expanding or growing phase : During which new locks on items can be acquired but none can be released

2) Shrinking phase : During which existing locks can be released, but no new locks can be acquired

If lock conversion is allowed then upgrading of locks [from read-locked to write-locked] must be done during the expanding phase and downloading of locks [from write-locked to read-locked] must be done in the shrinking phase

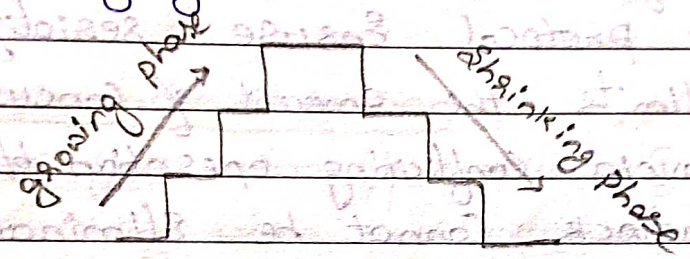


fig : 2PL

for example, consider 2 transaction T1 and T2 as shown below.

	T1 [2PL]	T2 (Not 2PL)
Growing	write_lock (B)	read_lock (A)
	Read_item (B)	Read_item (A)
	B = B + 20	unlock (A)
	write_item (B)	read_lock (B)
	write_lock (A)	Read_item (B)
	Read_item (A)	unlock (B)
	A = A - 10	display (A, B)
	write_item (A)	
Shrinking	unlock (B)	
	unlock (A)	

In T1 we put an write_lock on B [data item], so we can read or write B then we put on

write-lock on A and thus we can read or write to A finally we unlock both B and A in that order so T_1 is 2 phase

On the other ~~needs~~ hand T_2 is not in 2 phase this is bcoz T_2 puts a read-lock on data item A so read operations can be performed then it unlocks A. Then it locks B in read mode and it reads it then it unlocks B finally T_2 tries to display the result. This schedule of T_2 is not a 2 phase as it release lock on A and locks B immediately

A 2PL protocol means it should acquires all locks fst and then release them.

The 2PL protocol Ensure serializability bcoz

→ 2PL limits the amount of concurrency.

→ 2PL avoid conflicting operation b/w 2 transaction

Deadlock cannot be Eliminated using 2PL

Conservative, strict and rigorous 2phase locking:

There are a number of variations in 2PL. The technique just described is known as a basic 2PL.

A variation known as Conservation 2PL [static 2PL] requires a transaction to lock all the items it accesses before the transaction begin Executions, by predeclaring its read-set and write-set

The Read-set of a transaction is the set of all items that the transactions reads.

The write-set is the set of all items that it writes

If any of the predeclared items needed cannot be locked, the transaction does not lock any item instead it waits untill all the items are available for locking

Conservating 2PL is a deadlock free protocol

The most popular variation of 2PL is strict 2PL. In this variation a transaction T does not release any of its Exclusive [write] lock until after it commits or abort hence no other transactions can read or write an item that is written by T unless T has committed, leading to strict schedule for recoverability. Strict 2PL is not deadlock free.

A more restrictive variation of strict 2PL is rigorous 2PL. In this variation a transaction T does not release any of its locks [Exclusive & shared] until after its commits or aborts and so it is Easier to implement than strict 2PL

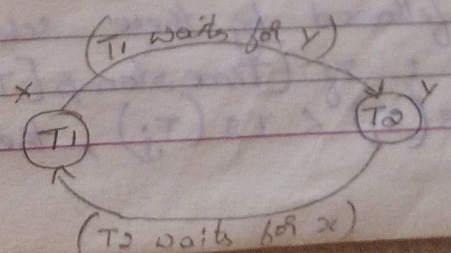
19/3/12 **Deadlock and starvation:**

What do you mean by deadlock? Explain with Example

Deadlock occurs when Each transaction 'T' in a set of two or more transaction is waiting for some item that is locked by some other transaction [T1] T' in the set hence Each transaction in the set is on a waiting queue, waiting for one of the other transaction in the set to release the lock on an item. Consider the following transaction

T1	T2
write_lock(x)	
read_item(x)	
	write_lock(y)
	read_item(y)
read_lock(y)	
	read_lock(x)

→ deadlock occurs.



Here, T1 is holding x and is waiting for y, which is locked by T2.
Similarly T2 is holding y and is waiting for x, which is locked by T1 and hence a deadlock

Deadlock preventation protocol :

The following are the protocol used to prevent deadlock –

1) By using Conservative 2PL [C2PL] - where every transaction lock all the item it needs in advance if any ~~one~~ of the item can't be obtain none of the items are locked rather the transaction waits and tries again to lock all the items it needs.

2) Concurrency : A second protocol which also limits Concurrency involves ordering all the items in the database and making sure that a transaction that needs several items will lock them according to that order. This requires that the programmer involvement in making sure that their is no deadlock. If their is a deadlock user has to break the deadlock cycle.

3) Time stamp phased protocol : This protocol works on a principle that " we must have a prior knowledge about the order in which the transactions will be access ".

Two schemes that prevent deadlocks are called
a) wait - die b) wound - wait

Suppose the transaction T_i tries to lock an item x but is not able to because x is locked by some other transaction T_j with a conflicting lock. The rules followed by these scheme are as follow

a) wait - die : If [Time stamp [TS]]
$TS(T_i) < TS(T_j)$, then [T_i older than T_j]

T_i is allowed to wait otherwise [T_i younger than T_j] abort [T_j]. T_i dies and restart it later with the same Time stamp [TS]

b) wound_wait : if $TS(T_i) < TS(T_j)$ then T_i older than T_j abort T_j [T_i wounds T_j and restart it later with the same timestamp otherwise [T_i younger than T_j] T_i is allowed to wait.

In wait_die scheme an older transaction is allowed to wait on a younger transaction whereas a younger transaction requesting an item held by an older transaction is aborted and restarted.

The wound_wait approaches does the opposite a younger transaction is allowed wait on an older one where as older transaction is requesting an item held by an younger transaction preempts the younger transaction by aborting it.

Starvation : Another problems may occur when we use locking is starvation, which occurs when a transaction cannot proceed for an indefinite period of time, while other transaction in the system will continue normally.

one solution for starvation is to have a fair waiting scheme such as using first come first served queue : Transactions are Enabled to lock an item in the order in which the originally requested the lock

20/3/12 Database Recovery Techniques CH-10

Recovery Concepts :

Recovery from transaction failure means that the database restored to the most recent consistent state just before the time of failure. To do this the system must keep information about the changes that were applied to the data items by the various transaction. This information is typically kept in the system log.

The failure in the transaction can be classified as follows

1) Catastrophic failure
2) non - catastrophic failure

1) Catastrophic failure : If there is Extensive damage to a wide portions of the database due to catastrophic failure such as disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstruct a more current state by reapplying or redoing the operation of committed transaction from the backed up log, upto the time of failure

2) Non - catastrophic failure : when the database is not physically damaged but as become inconsistent due to non - catastrophic failure such as concurrency control Enforcement, transactions Error Etc.

Several techniques are available to recover from non - catastrophic failures the 2 popular techniques are

a) Immediate update
b) Deferred update

a) Immediate update : This algorithm/method/
29/3/12 technique is also called as undo/redo algorithm.
In this algorithm if the transaction fails after recording some changes in the database but before reading its commit point, the Effect of its operation on the database must be undone [that is the transaction must be rollbacked]
If the transaction fails after reaching the commits points then redo operation is Executed.

b) Deferred update : It is also known as no undo/redo algorithm. In this algorithm if the transactions fails before reaching the Commit points then the transactions can be neglected this is because the transactions will not be having any Effect on the databases in the diskblock hence undo operation is not needed [no undo]
If the transaction fails after writing the Commit statement in the system log then it is necessary to redo the operation is by Extracting the informations from the system log.

Caching/buffering of the disk blocks :
The recovery process is closely related
30/3/12 with OS functions.
During the process of transactions one or more disk pages that includes the data item

to be updated are cached into main memory buffers and then updated in the main memory before being written back to the disk.

A directory for cache is used to keep track of which database items are in the buffers when the DBMS requests action on some ~~times~~ items first it checks the cache directory to determine whether the disk page containing the item is in the cache. If it is not, the item must be located on disk and then appropriate disk page are copied into the cache.

If there is no space in the main memory then page replacement algorithm will be used [such as Least Recently used and First In First Out].

Each buffer in the cache is associated with a bit called dirty bit which indicates whether or not the buffer has been modified.

When the buffer's contents are sent back to the secondary memory, the content must be written back to the corresponding disk page only if the dirty bit is one.

Two main methods are employed while sending the buffer back to the disk

1) Inplace updating - writes the buffer back to the same original disk location, thus overwriting the old value of any changed data items on disk

2) Shadowing : writes the updated buffer at a different disk location so multiple version of data items can be maintained

Date / /20

In general the old value of the data item before updating is called before image [BFIM] and the new value after updating is called [AFIM] the after image.

In shadowing both BFIM and AFIM can be kept on disk hence it is not strictly necessary to maintain a log for recovery

write - ahead logging :

~~Ith~~ In case of inplace updating recovery mechanism must ensure that the BFIM of the data is recorded in the appropriate log entry and that log entry is flushed to disk before the BFIM is overwritten with the AFIM in the database on disk so this process is known as write ahead logging.

steal/ no-steal approach :

If a cache page updated by a transaction cannot be written to disk before transactions commits this is called no-steal approach.

If it allows writing an updated buffer before the transaction commits is called steal approach.

force/ no-force approach :

If all pages updated by a transaction are immediately written to disk when the transaction commits it is called force approach otherwise it is called no-force approach.

2/~~4~~/10 Check points in the system log :

A [checkpoint] records is written into the log

periodically at that point when the system writes out to the database on disk all DBMS buffer that have been be modified

checkpoint records indicates that ~~that~~ the data have been updated on the secondary storage

checkpoints consists of the following actions

1) Suspend Execution of transaction temporarily
2) Force write all main memory buffers that have been modified to disk
3) write a checkpoint record to the log and force write the log to disk.
4) Resume Executing transaction.

* Fuzzy checkpointing: In this technique the system can resume transaction processing after the checkpoint record is written to the log without having to wait for step2 to finish however until step2 is completed, the previous checkpoint record should remain valid, To accomplish this the system maintains a pointer to the valid checkpoint which continues to points to the previous checkpoint record in the log once step2 is concluded that pointer is changed to point to the new checkpoints in the log.

Transaction rollback:

If a transactions fails for whatever reasons ~~records~~ ~~for~~ after updating the database it may be necessary to rollback the transaction.

If any data items values have been changed by the transactions and written to the database, they must be restored to their

previous values [BFIM].

The undo type log entry are used to restore the old values of data items that must be rolled back.

Cascading rollback:

If a transaction T is rollback, any transaction S that has, in the interim read the value of some data item (x) written by T must also be rolled back. Similarly once S is rolledback any transaction R that has read the value of some dataitem (Y) written by S must also be rolledback and so on. This phenomenon is called cascading rolled back.

Eg:

T1	T2	T3
read_item(A)	read_item(B)	read_item(C)
Read_item(D)	write_item(B)	write_item(B)
write_item(D)	read_item(D)	read_item(A)
	write_item(D)	write_item(A)

3/4/12

Functional dependency and Normalization for relational database

CH-7

Informal design ~~for~~ guidelines for relational schema

1) Semantic of the attributes
2) Redundant information in tuples and update ~~re~~ anomalies
3) Reducing the null values in tuples
4) ~~This~~ disallowing the possibilities of generating spurious tuples

1) Semantic of the attributes (relation):

Semantics : How the attributes values in a tuple relate to one another

Guidelines : design a relation schema so that it is easy to Explain its meaning. Do not Combine attributes from multiple Entity type and relationship types into a single relation intituly if a relation schema Corresponds to one Entity type or one relationship type, it is straight forward to interpret and to Explain its meaning otherwise if a relation Corresponds to a mixture of multiple Entities and relationships, semantic ambiguities will result and the relations Cannot be Easily Explained.

2) Redundant information in tuples and update anomalies :

one goal of schema design is to minimize the storage space used by the base relation

anomalis - Errors

Bhaskar
Date / /20

Grouping attributes into relationschema as a significant effect on storage space
Guidelines :

Design the base relation schema so that no insertion, deletion or modifications anomalies are present in the relation. If any anomalis are present note them clearly and make sure that the programs that update the database will operate correctly

3) Null values in tuples :

As far as possible, avoid placing attributes in a base relation whose values may frequently be null. If null are unavoidable, make sure that they apply in Exceptional cases only and do not apply to majority of tuples in the relation

4) Generations of spurious tuples :

Design relation schema so that they can be joined with Equality Condition on attributes that are Either primary key or foreign key in a way that guaranties that no spurious guarented.

Avoids the relation that contains matching attributes that are not [foreign key or primary key] combinations because joining on such attributes may produce spurious tuples

5/4/12 Anomalies : Errors or inconsistencies that may result when user attempts to update a table that contain redundant data.

Define 3 types of anomalies [update anomalies]

1) Insertion anomaly : means that some data cannot be inserted in the database

2) update anomaly : means we have data redundancy in the database and to make any modification we have to change all copies of the redundent data or else the database will contain incorrect data

3) deletion anomaly : means deleting some data cause other information to be lost

Functional dependency :

A functional dependency denoted by $x \rightarrow y$ b/w 2 sets of attributes x and y that are sub sets of R specifies a constraints on the possible tuples that can form a relation state r of R.

The constraints is that for any 2 tuple t_1 and t_2 in r that have $t_1[x] = t_2[x]$, they must also have $t_1[y] = t_2[y]$

Inference rules for functional dependency :

1) IR1 reflexive rule :

if $x \supseteq y$, then $x \rightarrow y$

2) IR2 augmentation rule :

$\{x \rightarrow y\} \models xz \rightarrow yz$

3) IR3 transitive rule :

$\{x \rightarrow y, y \rightarrow z\} \models x \rightarrow z$

4) IR4 decomposition, (or) projective rule :

$\{x \rightarrow yz\} \models x \rightarrow y$

5) IR5 union, or additive rule :

$\{x \rightarrow y, x \rightarrow z\} \models x \rightarrow yz$

6) IR6 pseudotransitive rule :
$\{x \rightarrow y, wy \rightarrow z\} \models wx \rightarrow z$

Normalization :

Normalization is the process of converting unnormalized table into normalized version.
Normalization was first proposed by "CODD"
normalizations define set of rules to be followed

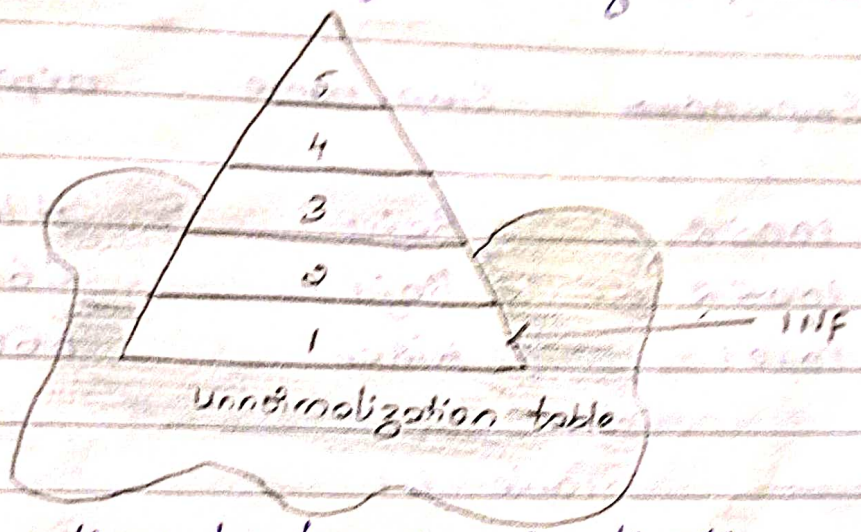


fig : levels of normalization

First normal form or 1NF :

The rules for 1NF are -
1) Every columns must be atomic
2) It cannot be decomposed into 2 or more subcolumns [eg : address fields of a person]
3) you cannot have multivalued columns
4) Every row and column can have only one value

Ex : Emp

Emp_no	Emp_name	dept_no
10052	Raju	A10, C66
10101	Kiran	D60

The above table violates INF.
It can be converted to satisfy INF as given below. There are 3 main techniques to achieve INF

1) Seperate tuples will be introduced for the repeating values.

Table 1 will be converted into INF using the first technique as given below.

Emp-name	Emp-name	dept-no
10052	Raju	A10
10052	Raju	C66
10101	Kiran	C60

Table 1.1

Table 1.1 is in INF but the above technique has redundancy problem.

2) If The maximum number of values for a column is known then the multivalued columns will be split as given in the Example below

Emp-no	Emp-name	dept-no1	dept-no2
10052	Raju	A10	C66
10101	Kiran	C60	Null

Table 1.2

The table 1.2 is in INF but it introduce lot

of null values

3) In the third method, the table will be split to Eliminate unnormalized information

Emp

Emp_num	Emp_name
10052	Raju
10101	Kiran

Emp_dept

Emp_num	dept_no
10052	A10
10052	C66
10101	D60

out of the above 3 solutions the third solution is superior it does not suffer from redundancy and null value and hence it is usefull popularly

12 Second normal form or 2NF :

2NF is based on the Concept of full functional dependency.

It says that Every non key attributes must depend on the Entire primary key, not on part of a Composite key. If a database has only single field primary key it is automatically in 2NF

Eg : Emp_dept

Emp_num	dept_no	dept_name
10052	A10	Accountancy
10072	A10	Accountancy
10074	D60	Computing

← depends on part of pk

In the above table the primary key is a composite key on Emp_num and dept_no, but the value of dept_name depends only on dept_no not on the entire primary key hence the above table voilates the 2NF.

The above tables can be normalized as given below

Emp_dept

Emp_num	dept_no
10052	A10
10072	A10
10074	D60

primary key

dept

dept_no	dept_name
A10	Accountancy
D60	Computing

primary key.

Third normal form or 3NF :

In 3NF a non key field cannot dependent on another non key field. If it is dependent on another non key attributes then the table is said to be violating 3NF

Eg : dept

dept_no	dept_name	mgr_Emp_num	mgr_name
A10	Accounting	10073	Johson
D60	Computing	10089	white
m80	marketing	10035	Dumant

primary key

depend on non key field

In the above table the non-key attributes mgr-name depends on another non key field mgr-Emp-num and hence the table voilates 3NF.

The dependency specified above is called ~~transicude~~ transitive dependency.
The above tables can be converted into 3NF as given below.

dept

dept_no	dept_name	mgr_Emp_num
A10	accounting	10073
D60	Computing	10089
M80	marketing	10036

Emp

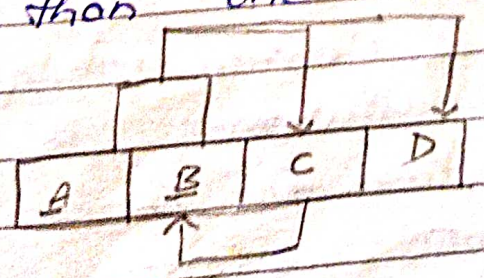
mgr_Emp_num	mgr_name
10073	johson
10089	white
10035	Dumont

Boyce-Codd-normal form [BCNF]:

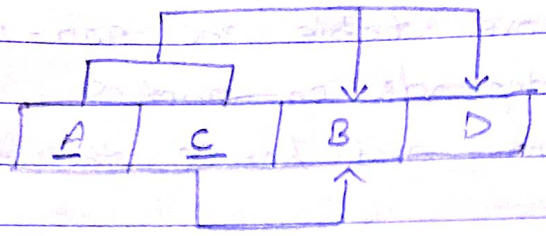
A special form of 3NF in which every determinant is a Candidate key. A table that is in BCNF must be in 3NF

A determinant is any attributes whose value determines other value within a row it is called as an determinant

If a table contain only one Candidate key the 3NF and the BCNF are Equivalent. putting this proposition another way, BCNF can be voilated only if the table contain more than one Candidate key

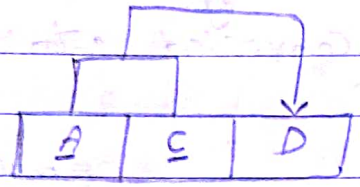


3NF Not BCNF

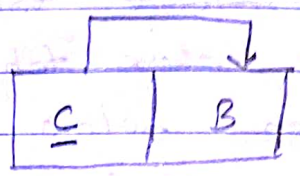


~~Part~~ 2NF

partial dependency.



3NF & BCNF



3NF & BCNF

Bhaskar Date / /20

CH - 8

Schedule : [History]

when a transactions are Executing Concurrently in an interleaved fashion, then the order of Execution of operation from the various transaction is known as a schedule

A schedule is a list of actions / operations [read, write, Commit, abort] from a set of transaction which must consists of all the instructions of those transaction.

A schedule or history S of 'n' transactions T1, T2, --- Tn is an ordering of the operations of the transactions subject to the Constraints that, for Each transaction T_i that participates in S, the operations of T_i in S must appear in the same order in which they occur in T_i

4/12 Types of schedules :

There are 5 types of schedule -

1) Complete schedule
2) Serial schedule
3) non serial schedule
4) Equivalent schedule
5) Serializable schedule.

1) Complete Schedule :

A schedule that Contain Either an abort or a Commit for Each transaction whose actions are listed in it is called as a Complete schedule

A complete schedule must contain all the actions of every transaction that appears in it.

2) Serial schedule :

T_1	T_2
$A = A * 10$;	
$B = B - 10$;	
	$A = A / 100$;
	$B = B * 4.06$;

Schedule 1

In a serial schedule only one transaction at a time is active - the commit or abort of the active transaction initiates execution of the next transaction. no interleaving occurs in a serial schedule.

If the action of different transactions are not interleaved that is transactions are executed from start to finish then the schedule is called as a serial schedule.

Eg : The above schedule 1 T_1 is followed by T_2 and hence is a serial schedules.

3) Non - serial schedule :

Schedule 2

T_1	T_2
$A = A * 10$;	
	$A = A / 100$
$B = B - 10$	
	$B = B * 4.06$

Bhaskar
Date / /20

A schedules in which the operation from a set of concurrent transactions are interleaved is called as non - serial schedule.

4) Equivalent schedule : Two schedules are said to be Equivalent schedules if they produce same result on any database states. It means that for any database state, the effect on the set of objects in the database, of executing the first schedule is indentical to the Effect of executing the second schedule.

Equivalent schedule always produce indentical result

for Eg : schedule C is an Equivalent schedule to schedule A and schedule B

Schedule A

T1	T2
A = A * 100	
B = B - 10	
	A = A/100
	B = B * 4.06

Schedule B

T1	T2
	A = A/100
	B = B*4.06
A = A * 100	
B = B - 10	

Schedule C

T1	T2
A = A * 100	
	A = A/100
B = B - 10	
	B = B * 4.06

5) Serializable Schedule :

A non serial schedule that is Equivalent to some serial Execution of transaction is known as serializable schedule

for Eg : schedule C is a serializable schedules which is Equivalent to schedules and schedule B

Conflict :

Two operations is a schedule are said to conflict if they satisfy all 3 of the following Conditions.

1) They belong to different transactions
2) they access the same item(x)
3) atleast one of the operation is a write_item(x)

Recoverable Schedule & non recoverable :

In a schedule if we cannot undo all the actions of an aborted transaction we say such a schedule is unrecoverable

A recoverable schedule is one in which transaction commit only after all transaction whose changes they read commit, recoverable schedules are allowed in DBMS.

In a recoverable schedule no committed transactions ever needs to be rolled back however it is possible for a phenomenon known as cascading rollback or cascading abort to occur, where an

Date / /20

uncommitted transaction has to be rolled back bcoz it reads an item from a transaction that failed

ng clause.
, update
dency & inference rules
INF, 2NF.
n.
ransaction
s
ect.