→ KORTH & 3 text Book
→ Navathe

5 Jan 2019

DataBase Engineering

Intro to DBS

* Data : Data are known facts that can be recorded and have implicit meaning

⇒ DataBase (DB) : collection of related data relevant to a particular enterprise.

⇒ DBMS : is a collection of interrelated data and a set of program to access those data.

⇒ File vs DB system

file syst.

→ It leads to data redundancy and inconsistency which results in higher storage ~~cost~~ and access cost.

→ Causes difficulty in accessing data efficiently

→ Integrity problems are created becoz of consistency constraints

→ Atomicity is difficult to ensure

→ concurrent access ~~accessanamo~~ anamolies occurs becoz data may be accsed uncoordinated application programs.

→ Security constraints are difficult to ensure as applications are added in an adhoc manner.

problem with file system

1. data redundancy & inconsistency
2. difficulty in ~~reading~~ accessing data
3. Isolation problem
4. Integration problem
5. Atomicity problem
6. Security problem
7. concurrency anamolies problem.

9 Jan 18

Data Abstraction

to hide the complexity of the system from the end users the developers use three levels of abstraction :-

(i) physical level

It describes how the data is actually stored.

(ii) logical level - It describes what data is stored and inter-relationship b/w them.

(iii) The view-level It provides many views of the same data base alongwith security and restricted access.

Instance & schema :

The schema is the overall design and logical structures of the data base. The schema is seldy changed.

The instance is a collection of information stored in the data base at any particular ~~movement~~ moment and it also called extension

Data Models

is a collection of conceptual tools for describing data, the relationship semantics and constraints

(1) Relational Model / Record Based Model

It represents data and the relationship b/w them using a collection of tables having fixed format, records and attributes.

eg oracle DB. MySQL server

(ii) Entity Relationship (ER) model

It is based on a collection of object called entities and the relationship b/w them.

(iii) Object Based data model

It is an extension of ER model with the concept of objects, methods and encapsulation.

(iv) Semi-Structured data Model

It is used to represent individual data item of the same type that may have different set of attributes.

relational dB system use SQL to Query and manipulate data.

10/01/2018.

⇒ Naive Users: are unsophisticated user who interact with ~~interact~~ DB using an application.

⇒ Application Programmers:
They develop user interfaces using RAD tools

⇒ Sophisticated Users:
They interact with the system without writting programs using a query language and they are internal to the organisation.

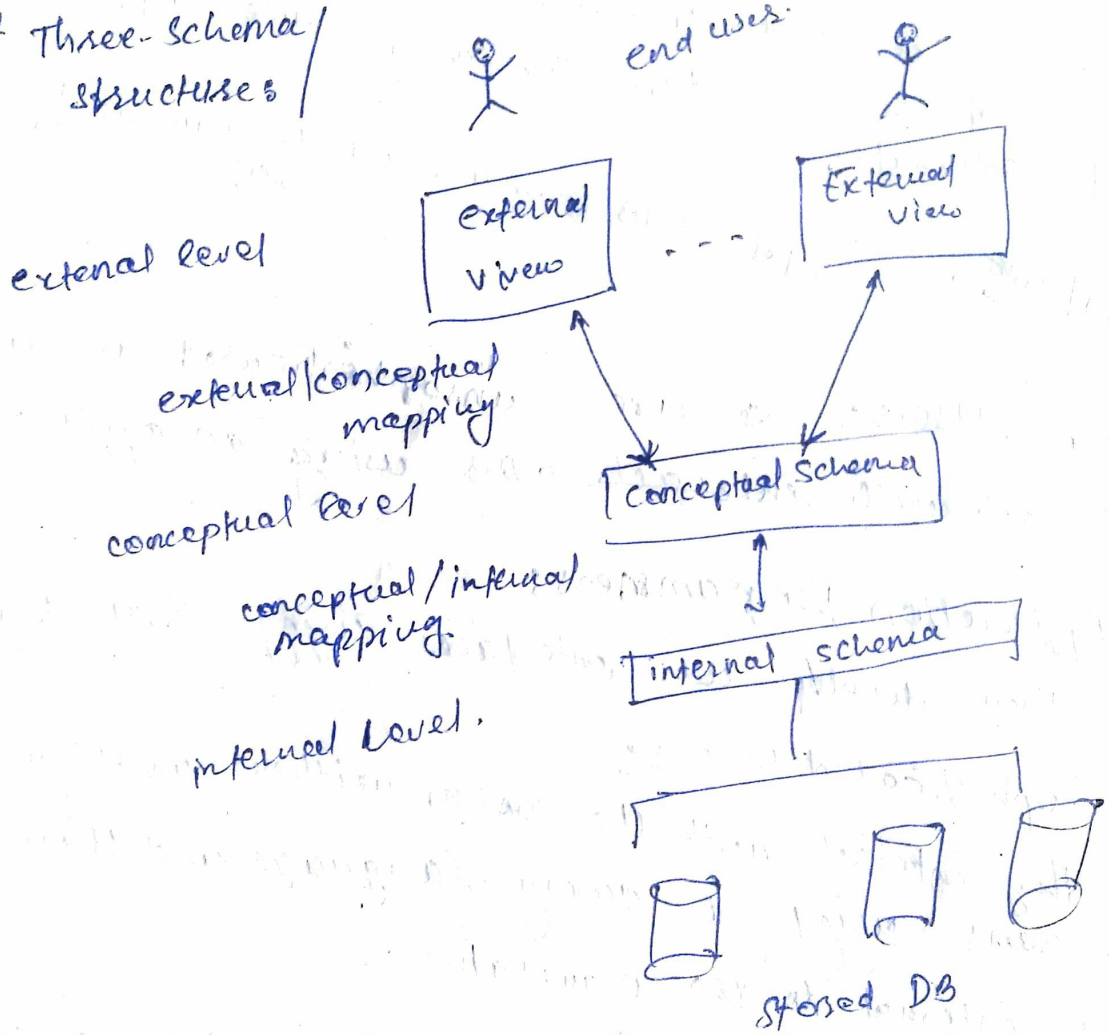
⇒ Specialised User: write special data base applicatio like knowledge based system and expert system

Function of DBA (DB administrator):

- It defines the storage structure and access method.
- It reorganises the schema to reflect changes as impsove performance.

- Granting of authorization to users for data access
- Routine Maintaince like, taking periodic backup to remote servers and disks
- The DBA ensures enough disks space for normal operation and monitors jobs for system performance.

Three-Schema Structures:



⇒ External level: It describes the part of the data base that a particular user is interested and hide the rest.

⇒ Conceptual Schema: It describes what data is present and stored in the DB and the relationships b/w them

⇒ internal/physical level:
Describes, how the data is stored using physical storage structures.

12/01/18

Data Independency: The three schema architecture implements data abstraction which leads to data independency. It is the capacity to change the DB schema at one level without having the to change to schema at next higher level.

Logical DI: is the ability to change the conceptual schema without having to change the external schema.

Physical DI: ability to change the physical schema without having to change the conceptual schema.

Classification of DBMS:

Based on Data Models

① Heirarchical Model
② Network model

Based on no. of users
(i) single users
(ii) multi users

Based on the no. of sites:
i) centralized
All data is located in one server.
(ii) Distributed
The data and the DBMS is distributed over many sites using communication network
(iii) Homogeneous: Uses the same set of S/w in multiple location.

(iv) Heterogenous: In heterogeneous syst the DBMs has diff^nt set of s/w, os and h/w at multiple locations.

Based on Domain:

(i) General Purpose DBMs:
e.g; Domain specific ERP.

(ii) Special Purpose DBMS
this syst have to support large no. of concurrent transactions without ~~accessing~~ accessive delays.
also called OLTP syst
Online Transaction processing.

Q. Create an instance & schema for the following.

Customer (cust_name, cust_id, address)
Bank (name, IFSC code, location)
Loan (loan no, Bank name, cust_id).

Customer

cust_name	cust_id	address.

Bank

name	IFSC code	location.

Loan

loan no.	Bank name	Cust_id

Customer

cust_name	cust_id	address
A	1	1/2
B	2	1/3

Loan

loan no.	Bank name	Cust_id
011	BA	1
012	BC	2

Bank

name	IFSC code	location
BA	BLD	ABC
BC	B11	BCD

16/01/18

ER Modeling: It describes data as entities attribute and represents the logical relationships structure of the database.

Entity: is an object or thing in real world which has independent existence.

e.g; student → concrete/physical

e.g; file, room. ← Abstract/conceptual

EntitySet: It is a set of entities of the same type that share the same properties or attributes. The individual entity of a set are called extensions on that set

Attributes: are descriptive properties of an entity and the set of permitted values for each attribute is called domain set.

Types of Attributes: i) Simple (Atomic) Vs Composite

Atomic Attributes are those that can not be divided into subparts and composite attributes can be divided into subparts.

Single value Vs Multivalue

Single valued attributes have a single value for a particular entity and Multi valued attributes have a set of values for a particular entities.

stored vs derived attributes.

the value of a ~~is~~ derived attributes can be obtained from value of other stored attributes.

eg; DOB is stored
Age is derived.

NULL attributes

NULL attributes takes a null value when a particular entities does not have a value for it.

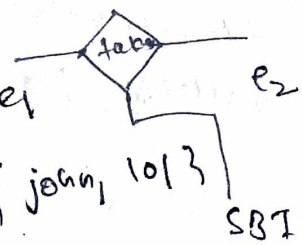
complex attributes: composition of composite and multi valued.

Relationship: It is an association among several entities and relationship set ~~said~~ is a mathematical relation among $n(\geq 2)$ entities taken from entity set

$\{ (e_1, e_2, e_3 \ldots e_n) / e_1 \in E_1, e_2 \in E_2 \ldots e_n \in E_n$

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

E_1	E_2
Customer	Loan
John	101



{ john, 101 }
SBI

{ John, 101, SBI } - takes loan

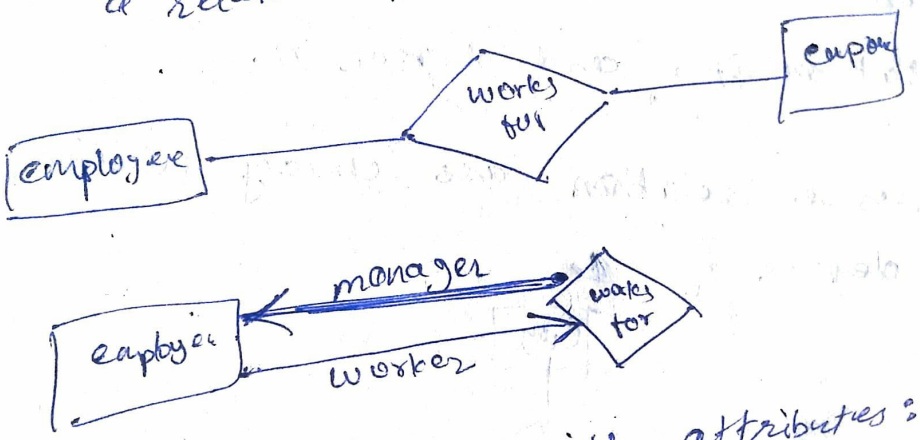
the association b/w entity sets is known as participation of the entity sets where ~~e1, e2, ... en~~ $E_1, E_2, \ldots E_n$ participates in a relationship R and relationship instance represents an association b/w the named entities.

17/1/18.

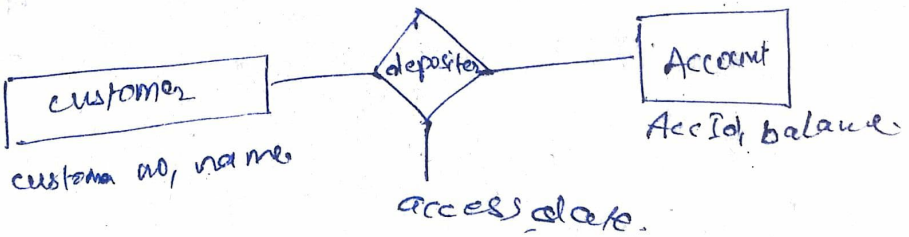
Recursive Relationships:—

when a entity sets of a relationship set is not distinct then the entities role is defined

→ Role is the f^n that an entity plays in a relationship



→ Relationship sets with attributes:
A relationship may have attributes called discriptive attributes.



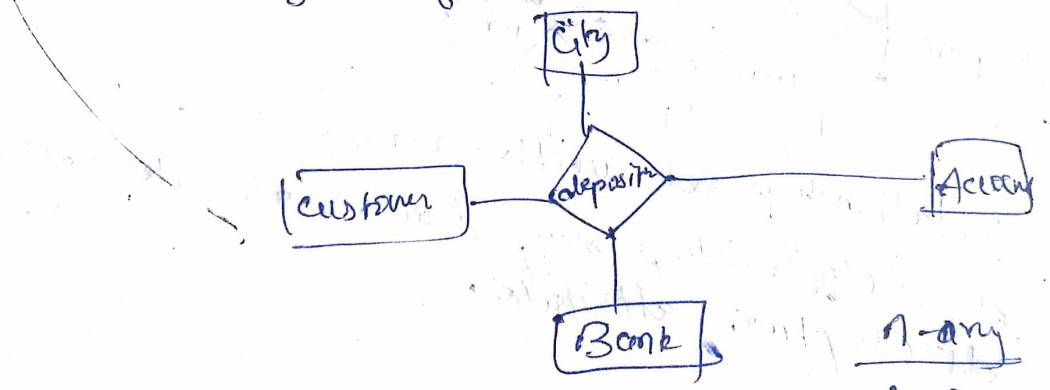
Degree of Relationship set :— the no. of entity sets that participate in a relationship set is called the degree of the relationship set.

A relationship set that involves two entity sets is called binary relationship and has a degree of two.

if the no. of entity set is 3, the relationship is called ternary and the degree is 3.

if the no. of entity set is greater than three ($n > 3$) it is called n-ary relationship. and degree n.

* Recursive relation are unary relation with a degree of 1.



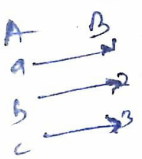
1-ary
2-ary
degree-4.

constraints:
→ Mapping Cardinality
→ key constraint
→ participation constraint.

~~Constraints~~ constraints are the limitation to which the contents of the relationship must confirm.

⇒ Mapping cardinality also called cardinality ratios express the no. of entities to which another entity can be associated within a relationship set.

1 to 1: an entity in A is associated with atmost 1 entity in B and vice-versa.

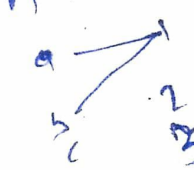


eg; persons driving license
persons ~~per~~ passport.

1 to many:
an entity in A is associated with any no. of entities in B but an entity in B is associated with max^m one entity in A.

eg person phone no.

Many to 1: An entity in A is associated with at most one entity in B but an entity in B can be associated with any no. of ~~ass~~ entities in A.



Many-Many: An entity in A is associated with any no. of entity in B and vice-versa.

Key Constraints:

A key allows to identify a set of attributes such that they can uniquely identify an entity where there ~~can be~~ cant be two entity e_1 and e_2 where

$e_1(a_1) = e_2(a_1)$

$e_1(a_2) = e_2(a_2)$

$e_1(a_3) = e_2(a_3)$

Superkey: If an entity set is a combination of one or more attributes whose values uniquely determines each entity of the set.

19/1/18

Candite key: If an entity set ~~of a~~ is a minimal super key for which no proper subset is a super key. There might exist several candidates keys for a single table.

Keys for Relationship sets:

$$pk(R) = pk_{E_1} \cup pk_{E_2} \cup pk_{E_3} \cup \dots \cup pk_{E_n}$$

Employee = {ename, empno, telno, deptname, deptno, address)

Dept (dname, jid, dno)

job (jname, jid, dno)

superkey:

{ename, telno} {jid}, {empno}, {dlocation, dname},

{dno}, {jid, jname}.

∴ ~~pk(R)~~ pk(R) = {empno, jid, dno},

(3) participation constraints:

if every entity in E, participates in at least one relationship in R. The participation is called total. But if some entities in E participate in relationship R then the participation is partial.

WEEK ENTITY SETS

An entity set that does not have a primary key and is existance dependent on an identified entity set is called the week entity set.

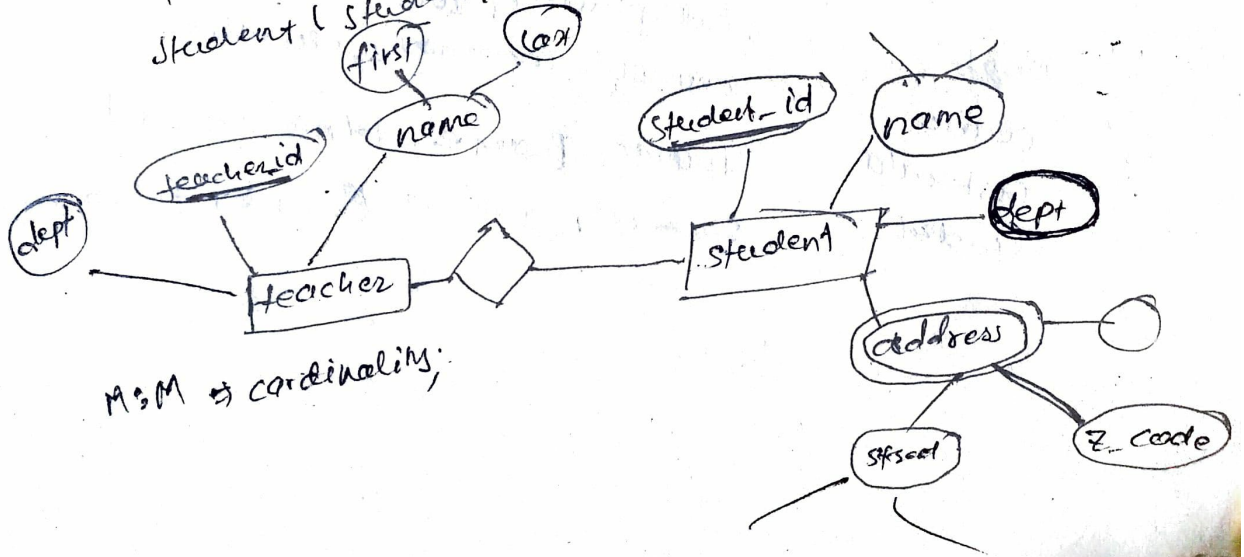
The primary key of a week entity set is formed by the strong entity set on which it is existance dependent and the discriminator of week entity set.

The discriminator is the attribute or set of attributes that distinguishes among the entity of the week entity set.

example:- (consider a teacher student relationship

Teacher(teacher_id, name, dept)

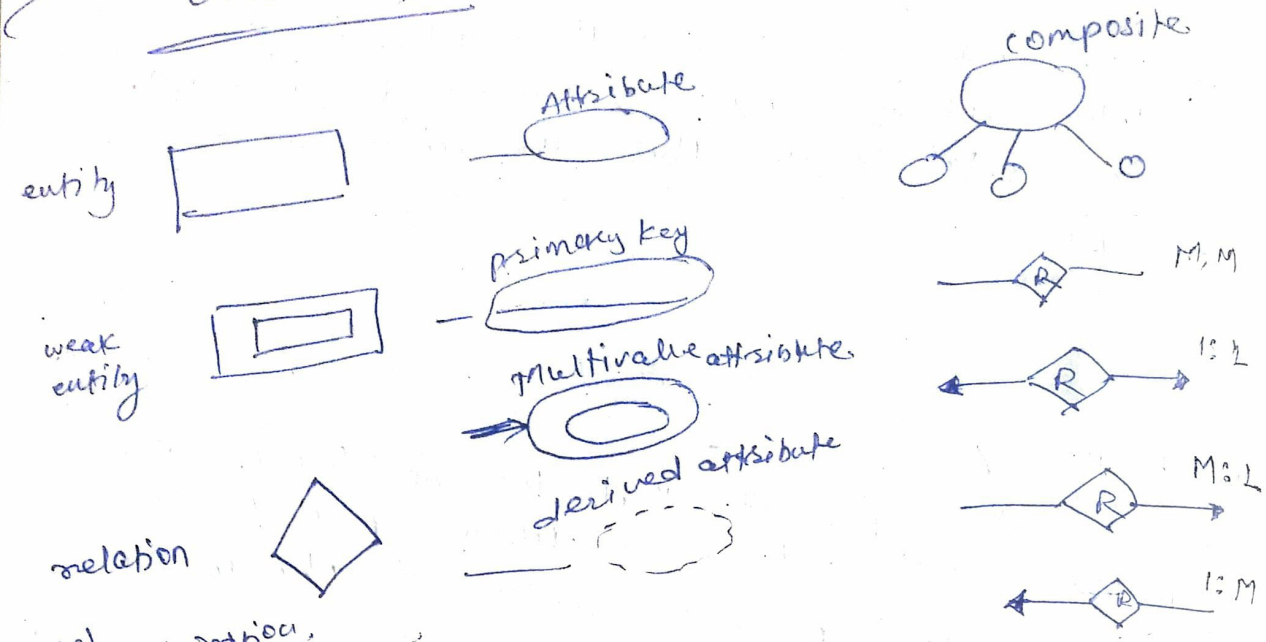
Student (student_id, name, dept, address).



M:M → cardinality;

23/01/18

ER notation



Developing ER diagram

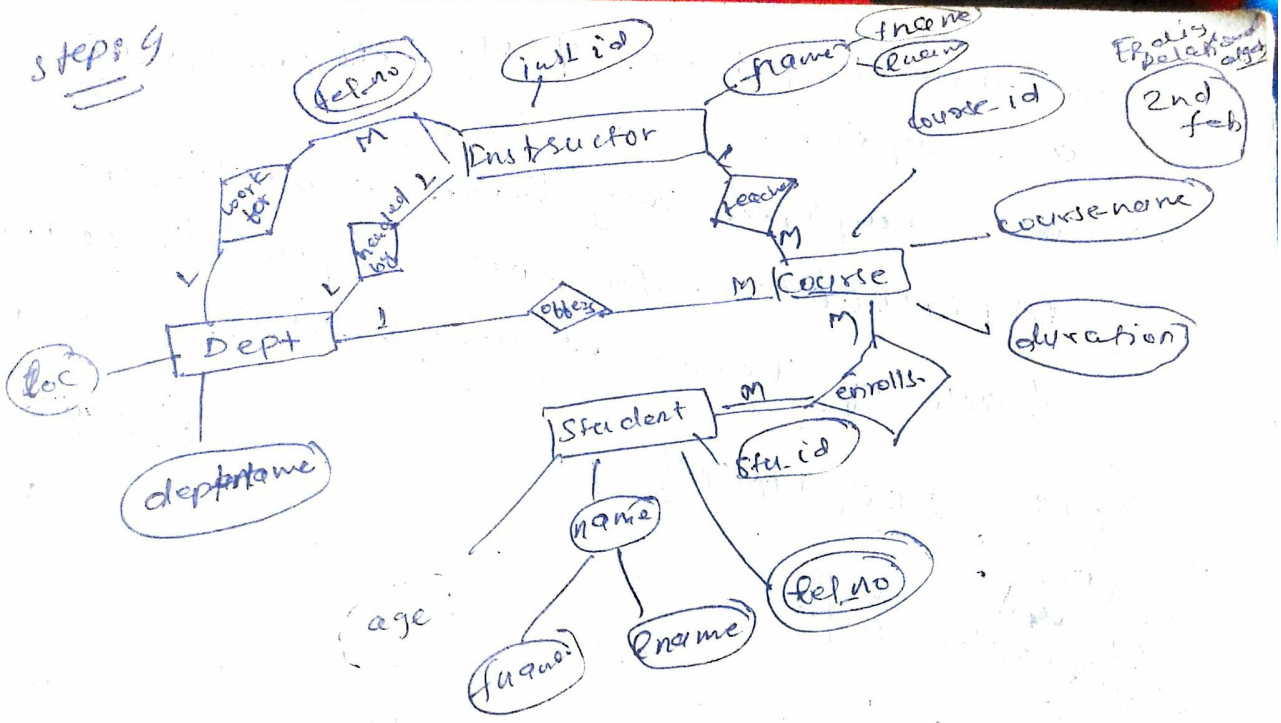
Step 1: Identify the entities

Step 2: " " relationship

Step 3: find the attributes:

entity	Attributes
Dept	deptname, HOC
course	courseid, coursename, duration
instructor	fname, lname, telno, inst_id
student	stu_id, fname, lname, telno, age.

Steps 4



Company DB

Requirements of the company:

-> The company is organised into DEPARTMENTS. each department has a name, number, location and an employee who manage the department. we keep track of the start date of the department manager.

-> Each department controls a number of PROJECTS. Each projects has a name, number and is located at a single location

-> We store each ~~employee~~ EMPLOYEE's social security number, address, salary, name and birth date

-> Each employee work for one department but may work on several projects.

-> We keep track of the number of hours ~~course~~ per week that an employee currently works on each project

→ We also keep track of the direct supervisor of each employee.

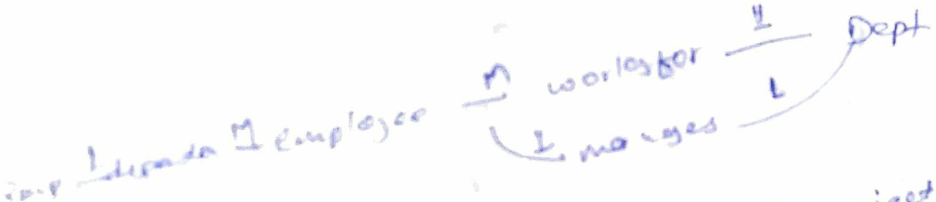
→ each employee may have a number of DEPENDENTS. For each dependent, we keep track of their name, address, birth date

Steps 1: DEPATMENT, PROJECT, EMPLOYEE, DEPENDENT
↓ weak entity.

→ company
→ department
manager

Steps 2:

deamptment	1 : M	projects.	controlls
employee	1 : 1	. department	works for
employee	1 : M	project	works on.



Dept 1 controlls M project.
employee 1 works on M project.
emp 1 supervises 1 emp.

30/1/18

construction of ER Diagram :—

Extended features of ER diagram:

1. Specialization ⇒ ▽

it is a process of designating subgroup within an entity set that are distinctive from each other in the set. The subgrouping become the lower level entity set where the lower level entity set is called subclass and the higher level entity set called the superclass.

2. Generalisation & specialisation

→ bottom up process → top down process

→ focuses on similarity b/w attributes → focuses on difference b/w the attributes

→

3. Attribute inheritance - a lower level entity set inherits all the attributes and relationship of the higherlevel entity set to which it belongs alongwith distinct feature within a particular level.

4. Design constraints ⇒

→ condition defined or user defined

→ also known as attribute define

→ membership is evaluated on the basis of wheater or not an entity satisfies an explict condn or predicting

Usere defined? the data base administration assign a particular entity into a given lower level entity set.

Disjoint:
This constraint requires that an entity belongs to no more than one lower level entity set and the overlapping constraint requires that same entity may belongs more than one lower level entity set within a single generalization

→ completeness constraints specifies whether or not an entity in the higher level entity set must belong to at least one of the lower level entity set within a generalization.

Total Constraints: Each higher level entity set must belong to a lower level entity set

Partial: is some higher level entity may not belong to any lower level entity set.

Aggregation: is an abstraction through which relationship are treated as higher level entities where it is convinient to express among relationship among relationship.

31/1/18

1. Data Structures
2. Data Integrity
3. Data Manipulation

1. Data structure

→ Relation: It is a representation of data and relationship in a 2D table

→ Tuple → It is a particular row a relation.

→ ~~the~~ attribute is a named column of a relation.

→ Domain: defines the kind of data or the set of all possible values that an attribute may contain.

→ Degree = is the no. of attribute in a relation.

→ Cardinality is the no of tuple a relation contain.

Data Integrity
→ keys
→ constraints
→ ER conversion into schemes.

Data Integrity: It is enforced through relational keys and constraints, which ensure that the data is accurate and allows the requested operation to be performed on the data.

keys:

① Candidate key of a relation are those attribute which have the property of uniqueness and irreducibility.

uniqueness → at any instance of time no two tuple are duplicate of each other for that attribute

→ if a candidate key is composite key then no individual attribute of the candidate key is unique.

② Superkey: Must satisfy the uniqueness property but may not satisfy the irreducibility property

③ Primary key: An attribute or set of attribute that uniquely identifies specific instances of an entity. It must satisfy the following properties.

→ It must not contain null values.
→ the value of the primary key should not change.
→ The primary key must be visible to anyone who can create, read or delete a record.

4. Alternate key: Are those keys from the list of candidate keys which are not selected as primary key.

5. Foreign key: Are attributes of a table which refer to the primary key of some other table and also known as referential integrity constraints.

C-1

a set of attributes f_k in relation schema R1 is a foreign key of R1 that references relation R2 if it satisfies the following two conditions.

(i) the attribute f_k has the same domain as the primary key attribute P_k

(ii) The value of f_k in a tuple T_1 of $s: r_1(R_1)$ either occurs as a value of P_k of some tuple T_2 of $r_2(R_2)$ is null

Q.2. A university Registrars office maintains data about the following entities.

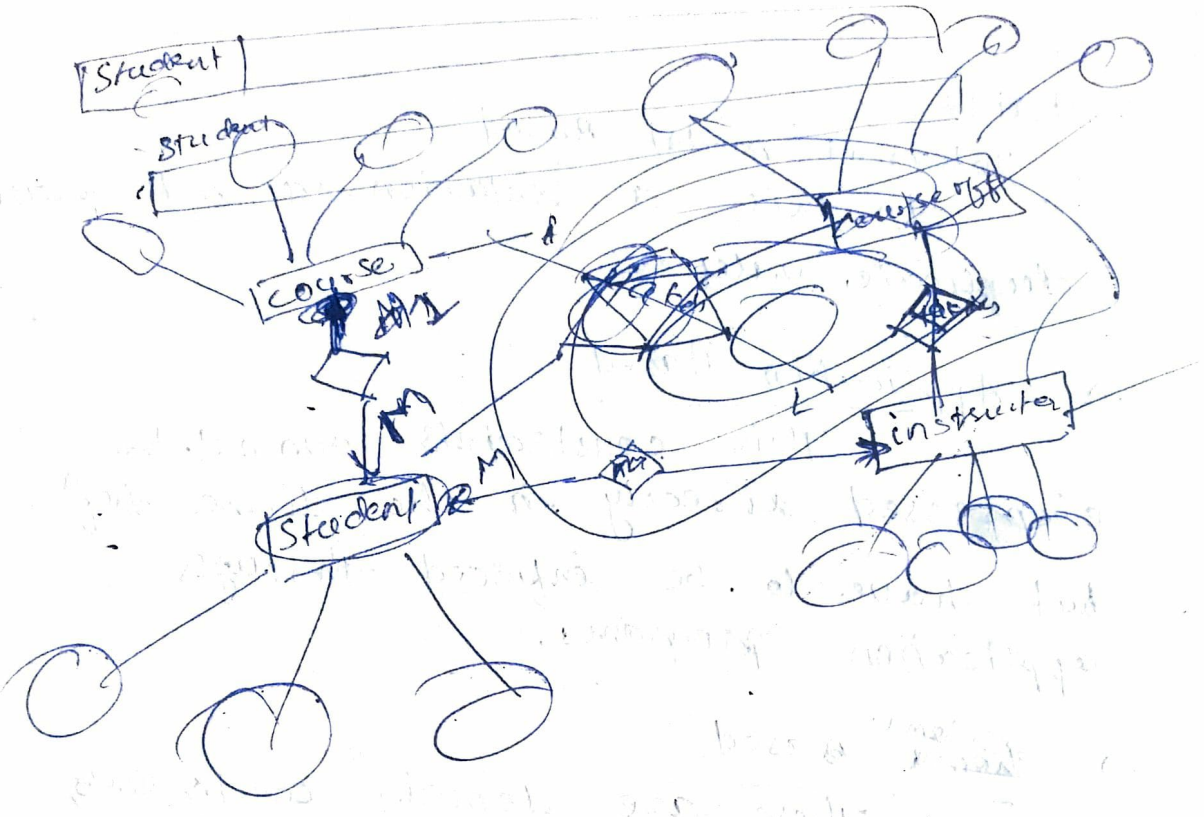
→ courses, including number, title, credits, syllabus, and prerequisites.

→ course offering, including course number, year, semester, section number, instructor(s), timing and classroom

→ Students, including student-id, name and program.

→ instructor, including identification number, name, department, and title.

→ Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled, instructors teaching courses for must be appropriately modeled


Student
Student
course
course
instructor
Student

6/2/18

Constraints:

→ Inhasent model Based
In this, a relation can not have duplicate values.

→ Application Based
This constraints can not be expressed directly in the schema def^n but have to be enforced through application programs.

→ Schema Based
These are domain constraints, keyconstraints, not_NULL constraints and refrenceial integrity constraints.

Reduction of ER diag. to Relation schema.

(1) Mapping strong Entity sets

(ii) Mapping a composite attribute
composite attribute only their simple component in the resulting schema.

iii) Mapping a multivalued attributes
it becomes a seperate table with a foreign key taken from the superior entity.

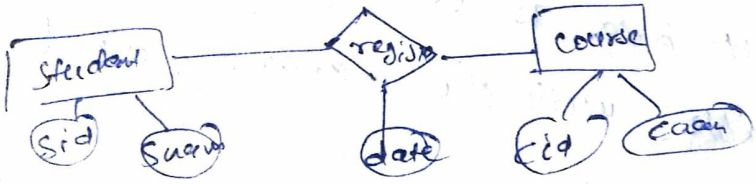
iv) Mapping a weak entity set

It become a seprate table with a foreign key taken from the strong entity set. The primary key of the weak entity set is composed of the discriminator of the weak entity set and the primary key of the strong entity set. On which it is existance dependent.

Representing Relationship set as table / schema.

(i) 1-M (one-many)
The primary key of the 'one' side become the foreign key of the 'many' side

(ii) M-M
Create a new table for the relationship set with the primary key of the two entities as its own primary key.



student
| sid | sname |

course
| courseid | cname |

registration
| sid | cid | date |
pk.

: Binary (1:1) relationship

total-partial.

The pk of the partial side become the foreign key (fk) of the total side.

⇒ The fk of the total side will use a synoname for the pk of the partial side.

total - total or partial - partial

when there is total-total or partial-2 participation on both side, the pk of the either of the entity becomes the fk of the other entity.

7/2/18

- Mapping on unary

1 : M relationship

the resulting table has the primary key attribute as the foreign key of the same table with the mandatory use of synonyms.

convesion of ternary relationship :-

mapping Extended ER features:

for the super type of each

(ii) assign to the subtype those attribute which are unique to each subtype.

⇒ assign to the subtype, the primary key of the super type table

Properties of a relation:—

(1) A rel^n is defined as a set of tuples where no preference is given for the logical ordering of element in a relation

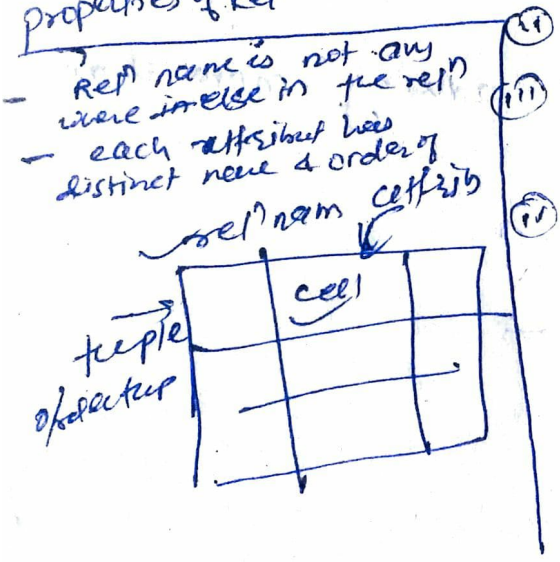
(2) Ordering of values within a tuple must be maintained

(3) Avoid storing null values as much as possible as they lead to

(1) wastage of memory
problem of join 'open'
difficulty in using aggregate function.
lead to multiple interpretations.
like, attribute does not apply
attribute value not recorded
or is unknown.

Properties of Rel^n

- Rel^n name is not any where in rel^n
- each attribut has distinct name & order of



Data Manipulation

need update delete more

→ Relational Algebra
→ Relational calculus

→ It is a procedural lang. used to tell the db system how to build a new rel^n from one or more existing rel^n and it forms the basis of a query lang.

① Select (σ): it is a unary opes^n

notation; $\sigma_p(r)$

the operator selects tuples that satisfy a given selection predicate p and is defined as,

$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$

Select * from loan where branch = 'BBSR'

$\sigma_{}$ (loan)
branch = 'BBSR'

eg $\sigma_{amt>1200 \wedge branch='BBSR'}$ (loan);

→ connectives are used for adding more than one predicates where

and ∧
or ∨
not ¬

③ project (Π):

In the projection oper^n the result is defined as a relation of K columns obtained by erasing the columns that are not listed and eliminating the duplicate rows from the result.

$$\Pi_{A_1, A_2, \ldots A_k}(r)$$

$\{A_1, A_2, \ldots$ Attribute name
$r \rightarrow$ relation name.

e.g.; 1. select the ename & salary of all the employee.

$$\Pi_{ename, sal}(emp)$$

2. $\Pi_{ename, sal}(\sigma_{sal > 2000 \wedge deptno = 10}(emp))$

④ Rename operation : allows to name the results of a relational algebra expression and also individual column.

$\Rightarrow \rho_x(E)$

⑤ Union Oper^n : The union operation takes two relation r and s as input and gives the result p where $p = r \cup s$ which has tuple drawn from r and s such that they are either in r or s

notation $r \cup s$

defined as: $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

⑤ Set Difference Oper: takes two relation r and s as input and results in an relation where the tuple belongs to r but not s.

⑥ Set - Intersection Operation : takes two rel^n as input and results in relation where the tuple must belong to r and s.

$r \cap s = r - (r - s)$

set of those elements which are only present in r and not in s.
→ this is also called set-difference oprn.

Assumptions of Unions, Intersection and set Difference

→ r and s must have the same arity
→ the attribute must be compatable i.e; the domain of the ith attribute of r must be same as the domain of ith attribute of s.

⇒ Properties of Union, Intersection and set Difference

→ both Union and Intersection are commutative operation
→ Both (∪, ∩) can be treated as associative and is applicable two any number of relation.
→ the set difference operation is not commutative.

⑦ Cartesian product : take two input relation r and s and result in a relation r×s containing tuple t∈r and q∈s

disadvantages: overloads the oracle engine

→ Not a relevent method
→ Results in spurious tuple

Banking example

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, amount)
depositor (customer_name, account_number)
loan (loan_number, branch_name, amount).

e.g) Find all loans of over $1200

$\sigma_{amount > 1200}$ (loan)

→ Find the loan number for each loan of an amount greater than 1200

$\Pi_{loan\_number}(\sigma_{amount > 1200}(loans))$

→ Find the names of all customers who have a loan, an account, or both from the bank

$\Pi_{customer\_name}(borrower) \cup \Pi_{customer\_name}(depositor)$

Find the names of all customers who have loan at the Perryridge branch.

$\Pi_{customer\_name}(\sigma_{branch\_name = Perryridge}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$

q. Find the names of the students whose CGPA is greater than 9.0.

Natural join : is a binary operation that allows to combine certain selections and a cartesian products into one operation.

symbol ⇒ (⋈)

steps: 1: Cartesian products of arguments

i.e; r ⋈ s

2. selection forcing ~~enq~~ equality on those attributes that appear in both rel^n schema.

3. Removes duplicates rows.

e.g $R = (A, B, C, D)$ $S = (E, B, D)$

- Result schema = (A, B, C, D, E)

r ⋈ s is defined

$$\Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \wedge r.D = s.D} (r \times s))$$

Q. Find the names of all customer who have a loan at the bank and the loan amount

Extended Relational-Algebra operation

① Generalized projection: extend the projection operation by allowing arithmatic function to be used in the projection list.

$$\Pi_{F_1, F_2, \ldots, F_n}(E)$$

20/2/18

② Aggregate fn

$$G_1, G_2, \ldots G_n \mathcal{G}_{f_1(A_1), f_2(A_2), \ldots f_n(A_n)}(E)$$

E → stands for any relation

$G_1, G_2, \ldots G_n$ → is a list for grouping the expression and can be left empty.

$f_1 \ldots f_n$ → aggregate fn's

$A_1 \ldots A_2$ → are attributes.

$\mathcal{G}$ → caligraphic G.

eg; Find the sum of the salary from emp table job wise.

$$job\ \mathcal{G}_{sum(sal)}(emp)$$

3) Outer join : Outer joins the extension of the join operation that avoid loss of information. It computes the join and then add the tuple from one relation that does not match the tuple in the other relation to the result of the join.

⟕ → Left outer join

⟖ → Right "

⟗ → full outer join.

Theta join(θ): it uses any of the comparision operators after performing the cartesian product.

$$R \underset{\theta}{\bowtie} S = \sigma_{\theta}(R \times S)$$

eg;

This Page is Blank

① data Retrieval Component
② data manipulation language (dml)
③ data definition language (ddl)
④ Transaction control language. (TCL)

① DRC : select
② DML : Insert, update, delete, merge
③ DDL : CARD/T
create, Alter, update, rename, drop, truncate

④ TCL:
commit | Grant
Rollback | revoke.

1. DS
- Relation
- tuple
- attribute
- domain
- degree
- cardinality

2. Data Integrity
Keys, Constraints, ER Rel schema
1. Candidate key (c) — uniqueness, irredundibility
2. Super key (s) — not null
3. Primary key (P) — visible, not changable
4. Alternate key (A) — C − P
5. Foreign key (F)
also called Referential Integrity constraints.

3. Data Manipulation
Rel^n Algebra
Rel^n Calculus

- Transaction: a unit of program execution that access or modify the data item.
properties of transaction: A, C, I, D.

21/2/18

Query Processor :

i) DDL Interpretor : It interprets DDL statements and records them in the data dictionary

ii) DML compiler & Organiser : It translates DML statements into low level instr that the query evaluation engine understan

iii) Query evaluation engine : executes low level instruction, generated by DML compiler.

iv) Embeded DML pre-compiler : It converts DML statements embeded in an application program into normal procedure call in the host language. (like C, C++, COBOL)

Storage Manager :

i) Buffer Manager : fetches, data from disk storage to main memory and decide what data to cache in the main memory

ii) File Manager : Manages allocation of of disk space and manage data structure used to represent information stored on disks.

iii) Transaction Manager. ensures DB inconsistency and concurrent transaction are executed without any conflict or system failure.

Authorization and Integrity Manager: Checks user to access data and ensures integrity constra

iv) Disk storage: Data contains the actual data files → data dictionary stores the meta data and the shed schema of the DB.

→ Indices provide faster access to data item.

→ statistical data provides data for reporting and pre fetching frequently used data.

Conversion of ER model to RDM scheme

23/2/2018

Steps of data Base design

(i) characterise the data needs of the perspective user and prepare user's requirement specification.

(ii) Use a data model and translate the requirement to a conceptual schema which provides the detailed overview of the enterprise

(iii) Examine the schema to remove incosistency and redundancy

(iv) Decide how to group attributes to form table using ER models and normalisation

(v) Review the schema to ensure that it meets the functional requirement.

(vi) Implementation involves two design phases

- logical design:
  High level schema is converted to data models.
- physical design:
  The physical aspects like, file organisation and internal storage structure are specified.

Modification of the DB

- Deletion
- Insertion
- Updation

- Deletion: Removes tuples from the DB but does not display the result.

$r \leftarrow r - E$ — r: relation (table name); E: relational algebra Expression

e.g; delete all account records from perryridge branch.

$account \leftarrow account - \sigma_{branch\_name = "perryridge"}(account)$

e.g; delete all loan records with amount in the range of 0-50.

$loan \leftarrow loan - \sigma_{amt \geq 0 \text{ or } amt \leq 50}(loan)$

- Insertion: to insert data into a rel^n a tuple is specified to be inserted or a query is specified whose results is the set of tuples to be inserted.

$r \leftarrow r \cup E$

e.g; insert information in the DB specifying that Smith has $1200 in account A-973 at the perryridge branch.

$account \leftarrow account \cup \{("A-973", "perryridge", 1200)\}$

$deposit_2 \leftarrow deposit_2 \cup \{("Smith", "A-973")\}$

e.g; insert, Alex staying in 25 downtown and city RedRoad and the loan no is B427.

$customer \leftarrow customer \cup \{("Alex", "25 Downtown", "Red Road")\}$

$Borrower \leftarrow borrower \cup \{("Alex", "B427")\}$.

- Updation: is used to change a value in the tuple without changing the other value of the tuple.

$$r \leftarrow \Pi_{F_1, F_2, \ldots F_e}(r)$$

e.g; make interest payment by increasing all accountbalances by 5% (0.05).

$$account \leftarrow \Pi_{account\_no,\ branch\_name,\ account\_bal \times 1.05}(account)$$

e.g; Pay all account with balance over $10000 6 percent interest and pay all other 5 percent

$$account \leftarrow \Pi_{account\_num,\ branch\_name,\ acc\_bal \times 1.06}(\sigma_{acc\_bal > 10000}(account))$$

$$\cup$$

$$account \leftarrow \Pi_{acc\_num,\ branch\_name,\ acc\_bal \times 1.05}(\sigma_{acc\_bal \leq 10000}(account))$$

e.g; Give all employees of First Bank Corporation a 10 percent salary raise.

$$works \leftarrow \Pi_{per\_name,\ company\_name,\ salary \times 1.1}(\sigma_{company\_name = "FBC"}(work))$$

e.g; Find the name of supplier who supply red part

$$\Pi_{sname}(supplier \bowtie (catalog \bowtie (\sigma_{color = "Red"}(Parts))))$$

27-2-18

Relational Algebra

1. It is a procedural language
2. It specifies how to open the query results
3. Describe the order in which operations have to be performed
4. It is closed to programming language
5. It is difficult for the use of non experts

Relational Calculus

1. declarative lang.
2. specify what result we have to obtained
3. Does not specify order of operation
4. closed to natural lang.
5. easy for the use of non-experts.

* Relational Calculus (TRC, DRC, QBE,)

i) TRC (Tuple Relational Calculus)

- It is a non-procedural declarative lang. which describe the desired information without giving a specific procedure to obtain that information

$\{t \mid P(t)\}$

projection, condition.

t is the set of all tuples such that the predicate P is true for t.

t - tuple variable

can be classified as
- free variable or
- bound variable

- a t variable is said to be free unless it is quantified by $\exists$, $\forall$. (i.e; without any $\exists$, $\forall$)
- Any tuple variable with $\forall$ / $\exists$ is condition is called bound variable.

(variable used in cond^n called tuple variable).

Q. Select those employee from employee table whose salary is greater than 10K.

$\{ t \mid employee(t) \text{ and } (\wedge) t.salary > 10K \}$

Q. select all tuple from employee table who work for department id = 10

$\{ t \mid employee(t) \wedge t.dno = 10 \}$

* Select the fname and lname from employee where salary is greater than 5K.

$\{ t.fname, t.lname \mid employee(t) \wedge t.salary > 5K \}$

Q. select the birthdate (b_date) and address of those employee where fname is John and lname is Smith.

$\{ t.b\_date, t.address \mid employee(t) \wedge t.fname = 'John' \wedge t.lname = 'Smith' \}$

2) DRC (Domain Relational Calculus):

- It uses a list of attributes to be selected from the relation rather than whole tuple and is also a declarative language.

Notation:

$$\{ \langle x_1, x_2, \ldots, x_n \rangle \mid p(x_1, x_2, \ldots x_n) \}$$

$x_1 - x_n \rightarrow$ domain variable

$P \rightarrow$ formula

- the result is a set of tuples $x_1 - x_n$ that make the DRC formula true.

3) QBE (Query By Example)

- It is an abstraction between the user and the actual query language where a parser is used to convert the user actions into DML command. It provide a simple GUI to enter queries in terms of the data and the command the user wants to perform.

Employee (fname, lname, address, dno, dnumber)

Department (Dname, dnumber)

Q. find the name and address of all employees who work for research department

R.A:

Π

e.fname,

28/2/18

A Relational DB Design Guidlines

-A. semantics of the relational attributes.
Each tuple in a rel should represent one entity and the attributes of other entities should not be mixed in the same rel. Only foreign keys should be used to refer to other entities.

-B. Reduce the redundant values in the tuples

Minimise the storage space used by the base relation and design a schema which does not suffer from insert, update and delete anamoly.

eg: examples of delete, insert, update anamoly.

problems:
i) insertion anamolies
ii) Deletion anamolies
iii) updation anamolies.

[C] Reducing null values in tuples.

Avoid placing attributes in a base relation whose values may frequently be null as null values lead to wastage of storage space, problems with join operation and mis-interpretation of attributes.

problem: wastage of storge spce, difficulty in applying aggregate fn (sum, count)

[D] Disallowing the possibility of generating spurious tuples by designing schema that can be joined with equality constraints

Functional Dependencies (F.D)

def^{n}:- Is used to specify formal measures of the goodness of the relationship design.

- A set of attributes X, functionally determine Y if the value of X uniquely determines Y

: $X \longrightarrow Y$

- for any two tuples, t1 and t2 in any rel^{n} instance r(R): if $t1[X] = t2[X]$

then $t1[Y] = t2[Y]$

eg; dos cage

t1 ||

t2

$X \longrightarrow Y$

e.g.; examples of FD constraints:

eg project no. determines project name and location

pnumber ⟶ {pname, plocation}.

{ssn, projectno} ⟶ hours

Properties of f^n^al dependencies :

- the set of attribute X is called the LHS of the f^n^al dependence (F.D) and the set of attributes Y is called RHS of F.D.
- It is defined explicitly by someone who knows the semantic of the relation.
- New F.Ds can be ~~involved~~ ~~inferred~~ derived from the existing ones.
- F.D's are used to find the primary key
- if K is a key of relation R - then K functionally determines all the attributes of the relation.
- It specifies a constraint on all relationship instances.

Inference Rules (IRs)

1. If $X \supseteq Y$ then $X \rightarrow Y$ (Reflexive Rule)
2. $\{X \rightarrow Y\}$ then $XZ \rightarrow YZ$ Augmentation Rule
3. $\{X \rightarrow Y, Y \rightarrow Z\}$ then $X \rightarrow Z$ Transitive Rule

} Armstrong Axioms

4. $\{X \rightarrow YZ\}$ then $X \rightarrow Y$ Decomposition Rule
5. $\{X \rightarrow Y, X \rightarrow Z\}$ then $X \rightarrow YZ$ Union rule.
6. $\{X \rightarrow Y, WY \rightarrow Z\}$ then $WX \rightarrow Z$ Pseudotransitive rule

e.g.) Reflexive Rule

$X \rightarrow$ partno (NT203)

$Y \rightarrow$ partid (203)

$Y \subseteq X$

As per Rule

partno $\rightarrow$ partid

Augmentation rule

SSN $\rightarrow$ dept

SSN, ename $\rightarrow$ dept, ename

Transitive Rule

SSN $\rightarrow$ ename

ename $\rightarrow$ email id

SSN $\rightarrow$ email id

e.g.) Decomposition

SSN $\rightarrow$ ename, telno

then SSN $\rightarrow$ ename

SSN $\rightarrow$ telno

Union rule.

SSN $\rightarrow$ ename

SSN $\rightarrow$ telno

SSN $\rightarrow$ ename, telno

pseudotransitive:

SSN $\rightarrow$ ename

dept, ename $\rightarrow$ manager

~~ename, SSN~~ $\rightarrow$

ESN, dept $\rightarrow$ manager

Mer
Armstrong Axioms are said to be sound & complete. sound, because given set of F.Ds on any rel^n R, if we can infer a new set of FD's from IR_1, IR_2, IR_3 it will hold good in every relationship instances of R.
and complete because, the closure of f (f. of's 3) can be determined from using only IR 1, 2, 3.

eg; 1. $x^+ = x$;

2. Repeat
for each f.D $Y \rightarrow Z$ in f
do
if $x^+ \supseteq Y$, then $x^+ = x^+ \cup Z$;
until (There are no changes to x^+);

closure of f or f^+
: f is a set of F.D, the set of all & Dependencies that include f as well as the new dependencies logically derived from F is called closure of f. or f^+

eg; $F = \{A \rightarrow B, B \rightarrow C\}$

∴ using Transitive Rule we get new depency

$A \rightarrow C$

∴ $f^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

∴ f^+ is the closure of f or f^+.

ie R is a rel^n with attributes X, Y

for $X \to Y$

e.g; $R = \langle A, B, C, D, E, F, G, H, I, J \rangle$

$F = \{ AB \to C$
$A \to DE$
$B \to F$
$F \to GH$
$D \to IJ \}$

$AB^+ \to \{ A, B, C, D, E, F, G, H, I, J \}$
$A^+ \to \{ A, D, E, I, J \}$
$B^+ \to \{ B, F, G, H \}$
$F^+ \to \{ F, G, H \}$
$D^+ \to \{ D, I, J \}$

Equivalence of F.D's using cover

e.g;

$F:$
$A \to C$
$AC \to D$
$E \to AD$
$E \to H$

$G:$
$A \to CD$
$E \to AH$

closure of F using G.

$(A)^+ = A, C, D$
$(AC)^+ = ACD$
$(E)^+ \to EAHCD$

all dependency of F can be also performed in closure of F using G.

$\therefore F \subseteq G$

closure of G using F.

$(A)^+ = ACD$
$(E)^+ \to EHADC$

$\therefore$ all dependency of F can be also performed in closure of G using F

$\therefore G \subseteq F$

$\therefore F = G.$

F equivalent to G.

Minimal set of functional Dependency.

→ A minimal / canonical cover of a set of FD is a minimal set of f.Ds that is equivalent to F.

→ set of $(\alpha \to \beta)$ where α, β is set of attributes then redunes arises due to i) α or ii) β or iii) $\alpha \to \beta$ the redundans.

→ minimal cover gives the reduced

→ finding minimal cover / cannonical cover

e.g., given, $R(w, x, y, z)$

$F: \{ x \to w$
$wz \to x$
$y \to wxz \}$.

① Decompose the dependency

$x \to w$
$wz \to x$
$wz \to y$
$wy \to w$
$y \to x$
$y \to z$

ii) remove the dependency which is not useful.

$x \to w \to x^+ \to x\,w$ } not same so this dependency is essential

$x^+ \to x$

$wz \to x$

$(wz)^+ \to w\,z\,x\,y$] same ∴ so this is not need

$wz \to y$

$wz^+ \to w\,z\,y\,x$]

$y \to w$

$(wz)^+ \to w\,z\,y\,x$] not same so, essential

$y \to x$

$(wz)^+ \to wz$

$y \to z$.

$y^+ \to y\,w\,x\,z$] same

$y^+ \to y\,x\,z\,w$

$y^+ \to y\,x\,z\,w$]

$y^+ \to y\,z$

$y^+ \to y\,z$]

$y^+ \to y\,x\,w$

So essential dependencies are

for $wz \rightarrow y$

~~x~~ $\rightarrow y$

$x \rightarrow w$

$wz \rightarrow y$

$y \rightarrow x$

$y \rightarrow z$

$wz^+ \rightarrow wzyx$

not same — $w^+ \rightarrow w$ (~~ignoring z~~)

no same — $z^+ \rightarrow z$ (ignoring w)

So they can not be ignored.

e.g; $F = \{ A \rightarrow BC$

$B \rightarrow C$

$AB \rightarrow C$

$AC \rightarrow D \}$

$A \rightarrow B$

~~$A \rightarrow C$~~

$B \rightarrow C$

~~$AB \rightarrow C$~~

$AC \rightarrow D$

$A^+ \rightarrow ABCD$

$A^+ \rightarrow ACD$

$A^+ \rightarrow ACBD$

$A^+ \rightarrow ABCD$

$B^+ \rightarrow BC$

$B^+ \rightarrow B$

$AB^+ \rightarrow ABCD$

$AB^+ \rightarrow ABCD$

$AC^+ \rightarrow ACDB$

$AC^+ \rightarrow ACBD$

$A \rightarrow B$

$B \rightarrow C$

$AC \rightarrow D$

$AC \rightarrow D$

$(AC)^+ \rightarrow ACBD$

eq $A^+ \rightarrow ABCD$

$C^+ \rightarrow C$

AC replaced next A+

$\therefore$ $A \rightarrow B$

$B \rightarrow C$

$A \rightarrow D$.

Normalisation

1NF
2NF
3NF
BCNF (Boyce and Codd NF) } based on F.D

4NF } based on mvd (multivalued dependency.
5NF

→ ① at first normalise then decomposition then check these two property
i) dependency preservation
ii) lossless join property
(non-additive join)

i) dependency preservation:
it ensures that each F.D's is represented in some individual resulting rel^n after decomposition

ii) lossless join property
guarantee that spurious tuple generation problem does not occur at after decomposition

1. 1NF: it allows only single atomic value for an attribute (i.e; no multivalued attribute is allowed).
e.g) Dept

Dname	Dnumber	dlocation
CS	10	L1, L2
IT	20	L3
MCA	30	L4

soln

Dnumber	Dlocation
10	L1
10	L2
20	L3
30	L4

Dname	Dnumber
CS	10
IT	20
MCA	30

ii) 2NF

- ~~fully base~~
- Based on fully functional dependency

→ prime attribute (the attribut which is part of candidate key (at least one))
→ Non-prime attribute (" " " not part of any key)

fully functional dependency:

A f.D $x \to y$ is called fully f.D if due to removal of any attribute from x the dependency does not hold any more.

But if the f.D $x \to y$ still holds after removal of some attribute from x then it is called Partial f.D

Partial dependency: if proper subset of candidate key determines non-prime attribut, it is called P.D.

" A rel^n schema R is called in 2NF, if every non-prime attribute of R is not partially dependent on key of R "

A Rel^n is in 2nf iff
- It is in 1NF
- It has no Partial dependency.

13/3/18

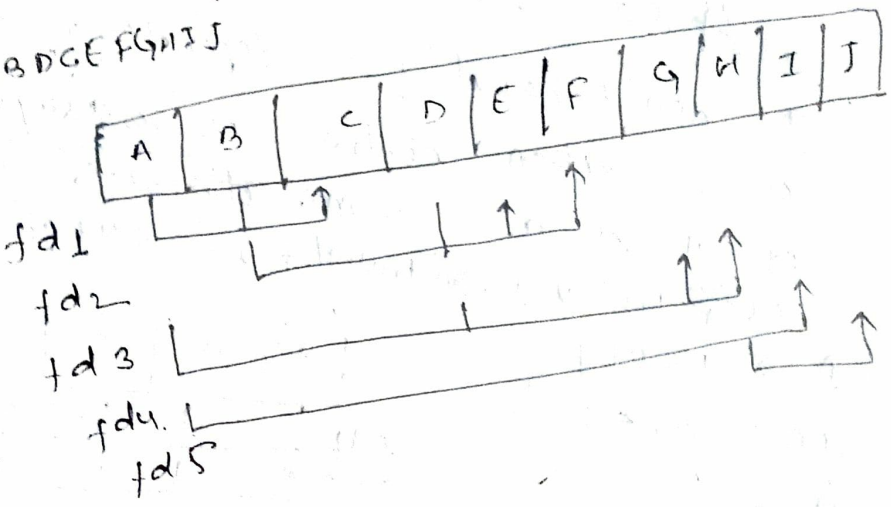
AB determines C

AB → C
BD → EF
AD → GH
A → I
H → J

$(AB)^+ \longrightarrow ABCI$
$(BD)^+ \longrightarrow BDEF$
$(AD)^+ \longrightarrow ADGHIJ$
$(A)^+ \longrightarrow AI$
$(H)^+ \longrightarrow I$

$(ABD)^+ \longrightarrow ABDCEFGHIJ$

↓ prime as well as key for the relation



The Goals of Normalization

i) satisfy the attribute preservation property

ii) normalization must satisfy the dependency preservation property.

iii) satisfy the loseless join property or non additive join property.

*** goals of normalization.

~~Properties~~

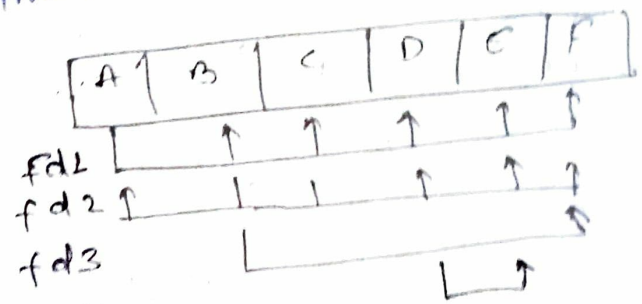
satisfy :- Attribute preservation
- dependency preservation
- lossless join property (or additive join).

non-trivial depency: if functional dependency $x \to y$ holds true when y is not a subset of x then it is called a non-trivial f'al dependency. (i.e., $x \to y$ but $y \not\subset x$)

3NF: 3rd Normal form
: Based on transitivity property.

A rel^n are is 3NF if, whenever a non-trivial f'al dependency $x \to y$ holds in R then either x is a super key of R or y is a prime attribute. of R.
(i.e., for every non-trivial FD one of the above cond^n should occur)

$A \to BCDEF$
$BC \to ADEF$
$B \to F$
$D \to E$



$\{A\}$, $\{BC\}$ → prime

$(A)^+ \longrightarrow ABCDEF$
$(BC)^+ \longrightarrow ABCDEF$
$(B)^+ \longrightarrow BF$
$(D)^+ \longrightarrow DE$

$ABC \longrightarrow PA$ → prime attribute
$DEF \longrightarrow NPA$
↓
non-PA

14/3/18

a b c d e f g h i j

1 2 3 4 5

AB → C
A → DE
B → F
F → GH
D → IJ

$(AB)^+$ → ~~ABC~~ CdeFGHIJAB

$(A)^+$ → ~~AIJ~~ E DEIJA.

$(B)^+$ → ~~BGH~~ ~~GHIJ~~ FGHB

$(F)^+$ → GHF

$(D)^+$ → IJD.

2NF

R_1 (a b c)

R_2 (a d e i j)

R_3 (b f g h)

R_{21} (a d e)

R_{22} (d i j)

R_{31} (b f)

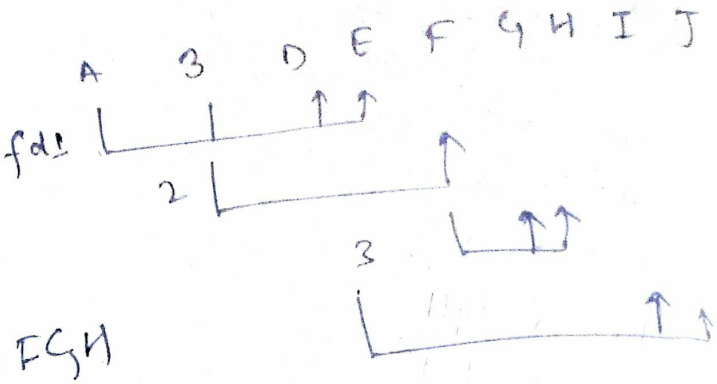
R_{32} (f g h)

1. $AB \rightarrow DE$

$B \rightarrow f$

$f \rightarrow GH$

$D \rightarrow IJ$



$(AB)^+ \longrightarrow DEIJ\ AB\ FGH$

$(B)^+ \longrightarrow FGHB$

$(f)^+ \longrightarrow GHf$

$(D)^+ \rightarrow IJD$

2. $AB \rightarrow C$

$(BD) \rightarrow EF$

$(AD) \rightarrow GH$

$(A) \rightarrow I$

$(H) \rightarrow J$

$(AB)^+ \longrightarrow ABCI$

$(BD)^+ \longrightarrow BDEf$

$(AD)^+ \longrightarrow ADGHIJ$

$(A)^+ \longrightarrow AI$

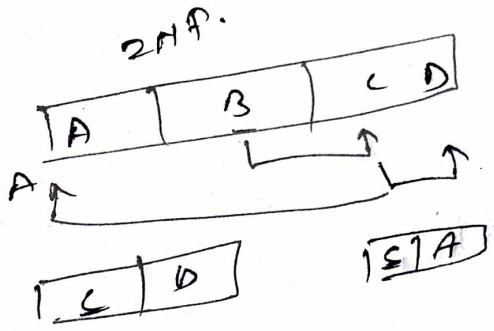
$(H)^+ \longrightarrow HJ$

BCNF (Boyce Codd Normal form):
does not guarantee about dependency preservation property

- A Relⁿ R is said to be in BCNF if there is a F.D $x \to y$ then x is a Sk. of R. (i.e; for every non-trivial F.D $x-y$, x is a super key).

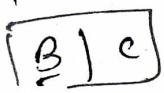
e.g; $F = \{ B \to C, C \to A, C \to D \}$

Sk $B^+ \to BCAD$



3NF & BCNF

R1



20/3/18

→ Multi value Dependency

A MVD $x \twoheadrightarrow y$ specified on a rel^n r, specifies the constraint that if two tuples t_1 and t_2 exist such that $t_1[x] = t_2[x]$ then these must exist tuples t_3 and t_4 such that → $t_1[x] = t_2[x] = t_3[x] = t_4[x]$,

→ $t_1[y] = t_3[y]$ & $t_3[y] = t_4[y]$

→ $t_1[z] = t_4[z]$ & $t_2[z] = t_3[z]$

4NF

A rel^n R is said to be 4NF w.r.t a set of dependencies f iff for every non-trivial MVD $x \to y$; x is a super key of R.

TRANSACTION :

It is a unit of program execution that accesses and possibly updates various data items.

- Transactions are written in HLL basically SQL and Java.

i) Transaction concept:
has a propery called
ACID property:

A → Atomicity
C → consistancy
I → Isolation
D → Durability

→ Atomicity ensures that all the steps of the transaction are executed or none is executed.
- It is handled by transaction management componet.

→ Consistancy : ensures that the execution of the transaction does not change the consistancy of a DB and it is manged by the application programmer.

→ Isolation : ensures that when multiple transaction are executing. con-currently each transcation must be unaware of other con-currently executing transactions and handled con-currency controlled component.

→ Durability : ensures that once the transcation complets successfully, the changes it has made to the DB persists even if there is syst. failure.
- and managed by the recovry maugement componenf.

28/3/18

Concurrency control : A DB must provide a mechanism that will ensure all possible schedules are conflict or view serializable.
- recoverable
- Preferably cascade-less schedules

techniques:
- Lock-Based protocols
- Timestamp based "

lock-Based: A lock is a mechanism to control concurrent access to a data item.

1. X-mode (exclusive) lock: data can be both read and written
   Lock-X(A)
2. Shared(S) mode lock: data item can only be read.
   Lock-S(A)

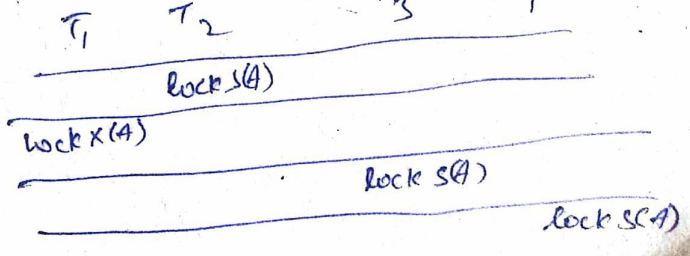
A: data item name

* locks are managed by concurrency control manager.

A trans can be granted or lock on a specific data item if the requested lock is compatible with the locks already existing on the data item by either trans.
- If a lock cant be granted then the transaction is made to wait until the locks are released

	S	X
S	true	false
X	false	false

Disadvantage of lock-Based:
→ dead lock : the situation in which two or more trans are waiting for one another to release locks.
→ starvation :



if a trans^n T_i has a shared mode lock on datat item A and another trans^n T_j request an exclusive mode lock on same data item. Then T_j has to wait until T_i releases the lock. Meanwhile, if a third trans^n T_k request a shared mode lock on the same data item A, then T_k is granted access and T_j has to wait further.

Sol^n to starvation: when a trans^n request an incompatable lock then it is put on the ready queue and the other trans^n requesting compatable lock are also put on the same queue below the transcation requesting incompatiable lock

Two phase locking protocol: simply ensure serializability but not cascadeless scheduling

phase1: Growing phase

→ trans^n may obtain locks

→ trans^n may not release any locks

phase2: Shrinking phase

Transcation may release the lock but it wont obtain new locks

Strict 2-phase locking protocol:

the transn must hold all its x-lock till it commits all aborts

Rigorous Two phase locking protocols:

it holds all the lock till it commit all aborts.

30/3/18

⇒ lock conversions :

in the 1st phase of lock-conversion :-

growing phase
- can acquire lock-S on any data item
- " " lock-X " "
- convert a lock S → X (upgrade)

shrinking phase
- can release lock-S
- " " lock-X
- downgrade X ⟶ S

→ lock point : point in the schedule where the transaction has obtained its final lock (i.e) the end of the growing phase and before the begning of the shrinking phase.

DEADLOCK :

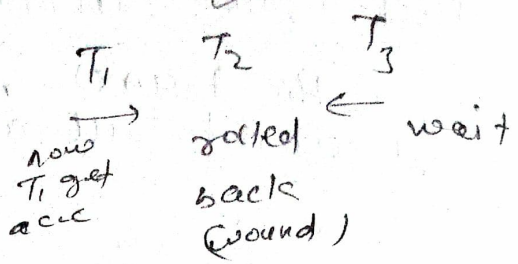
- prevention
- detection
- Recovery

-prevention:

i) wait-die scheme
(non-preemptive)

	T_1	T_2	T_3
wait	✓		rolled back
continue		✓	

2) wait-wound scheme (pre-emptive)



- Detection of deadlock:-

This protocol will initially allocate system to enter to a deadlock state and try to recover from it.

wait-for-graph

pair, $G = (V, E)$

an edge is created when a trans^n T_i requests an data item from trans^n T_j. The edge is removed when T_j releases the data item.

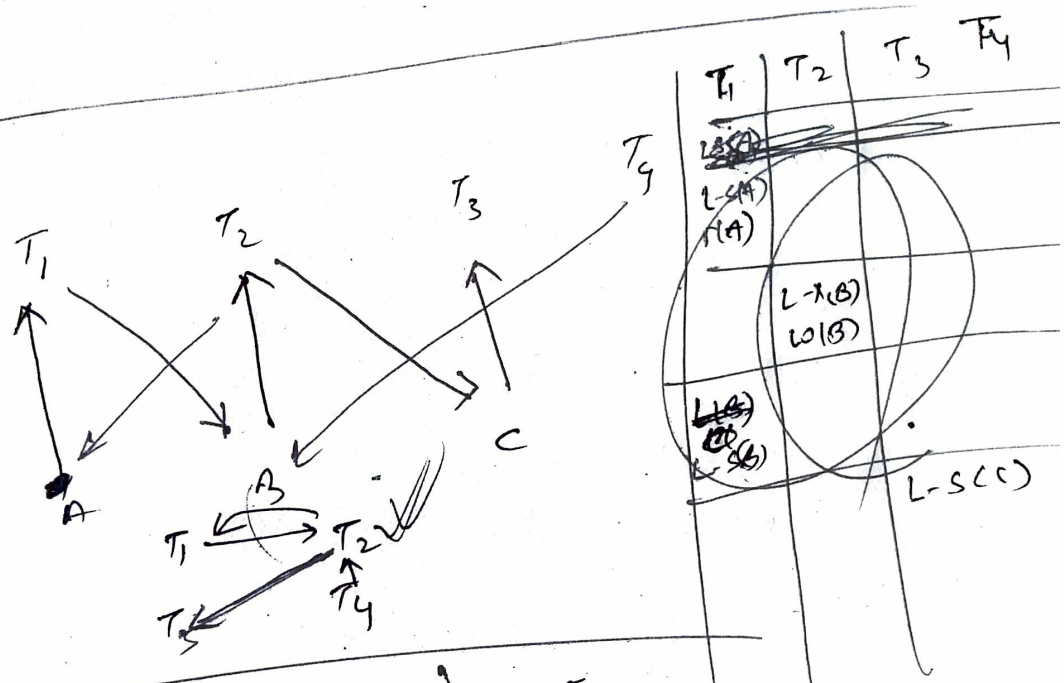
Dead lock Recovery: with one or more trans^n is rolled back to break the cycle in the wait-for graph. Actions taken are:-

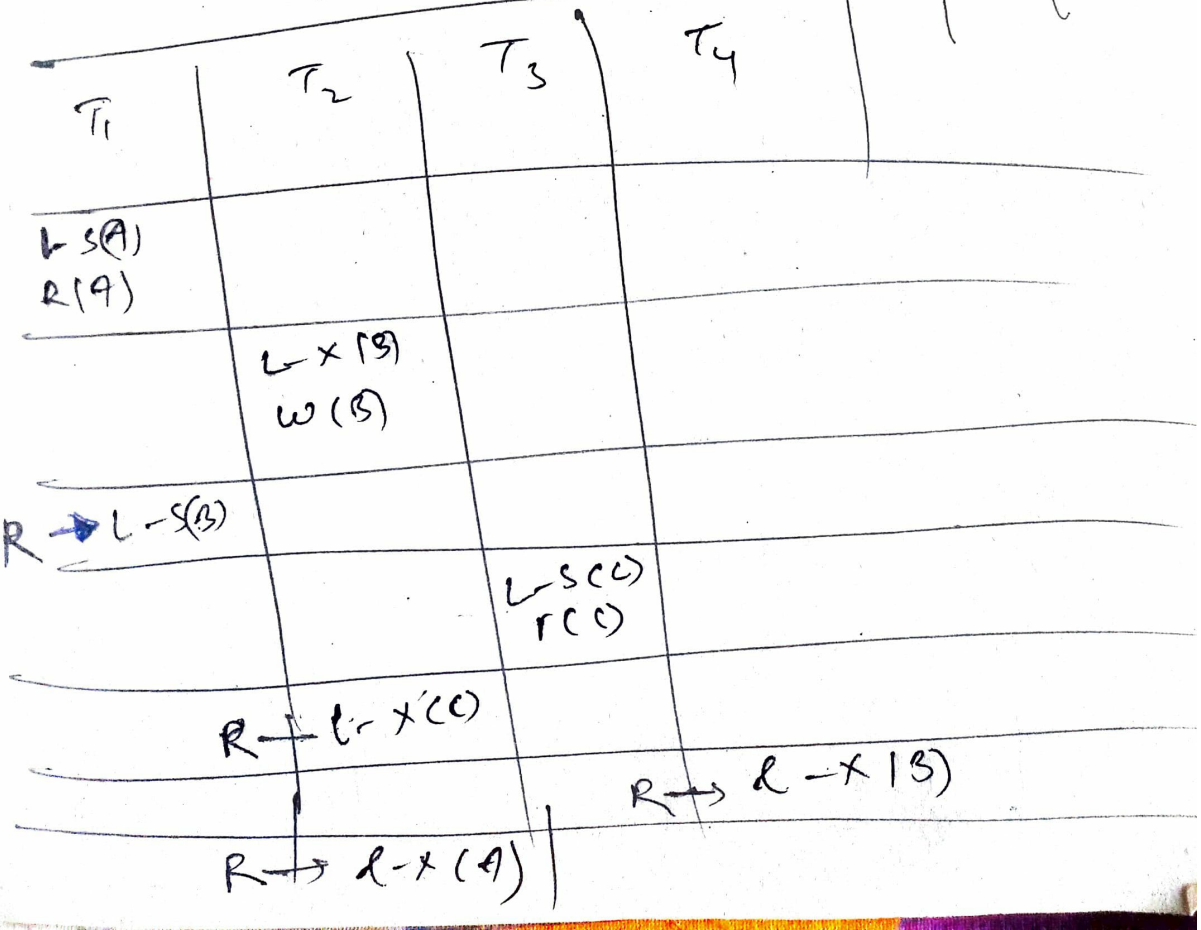
1). Selection of victim (i.e; which trans^n will be rolled back)
- the selection of victim will be done on following criteria:-
  - a trans^n which has computed for lesser time
  - the one which has used less no. of locks.
  - the one which has least dependency with other transaction.

2) Roll back:-
   i) complete roll back (Restart the transaction again)
   ii) Partial roll back i.e; roll back

as for as necessary to break the deed lock.

3) Starvation: if the same trans^n is selected for victim repeatedly then it is said to be starved. To resolve starvation, there must be a threshold value which indicates no. of times a particular trans^n can be selected as victim.





Many Blank Pages Later...