SIKSHA 'O' ANUSANDHAN

DEEMED TO BE UNIVERSITY

Admission Batch: 2019 Session:2022

Theory Assignment 01

Introduction to Databases (CSE 3151)

Submitted by

Name: **Bali Babu Chauhan**

Registration No.: 19410121182

Branch: **CSE**

Semester: **6TH** Section: **M**

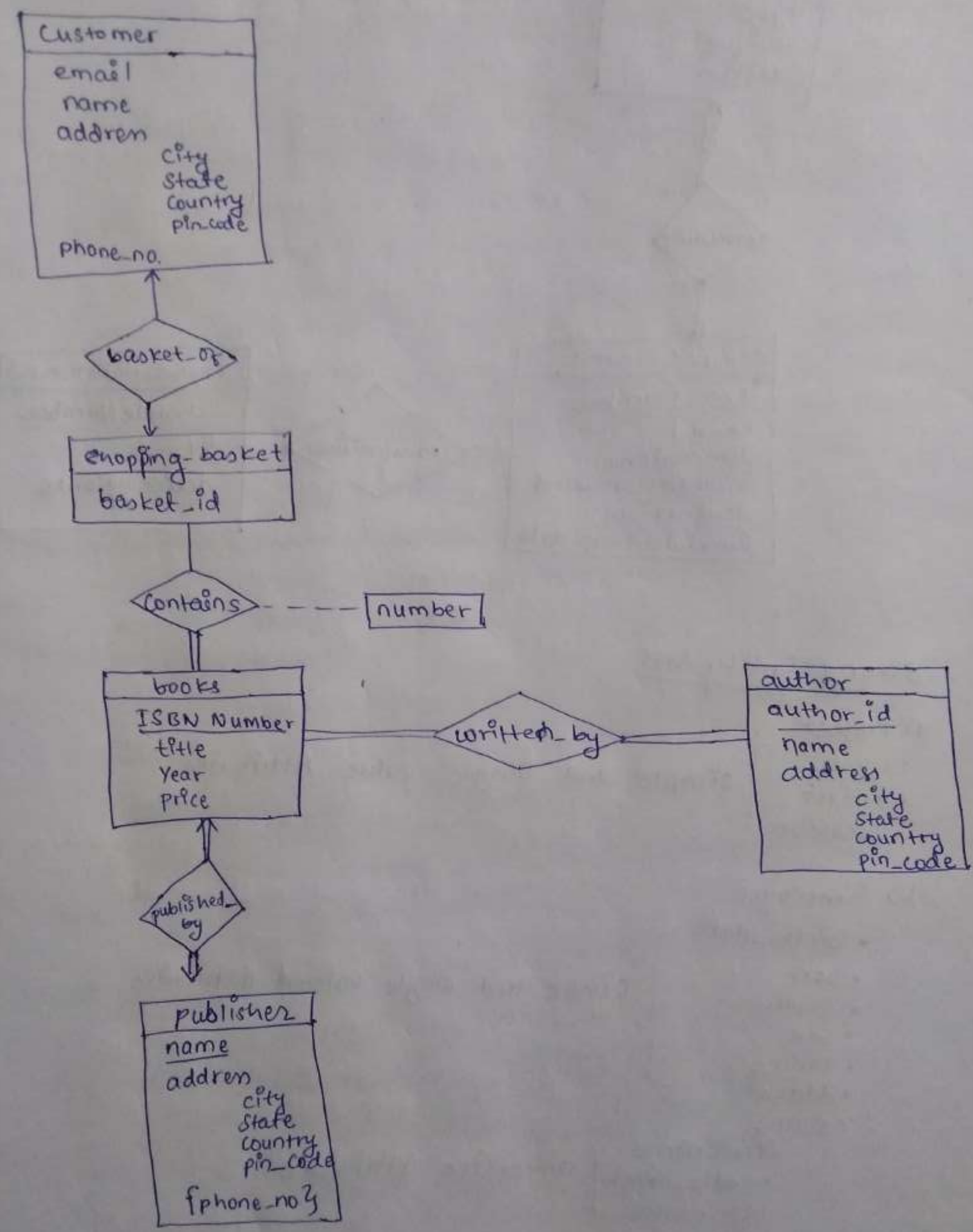
Department of Computer Science & Engineering

Faculty of Engineering & Technology (ITER)

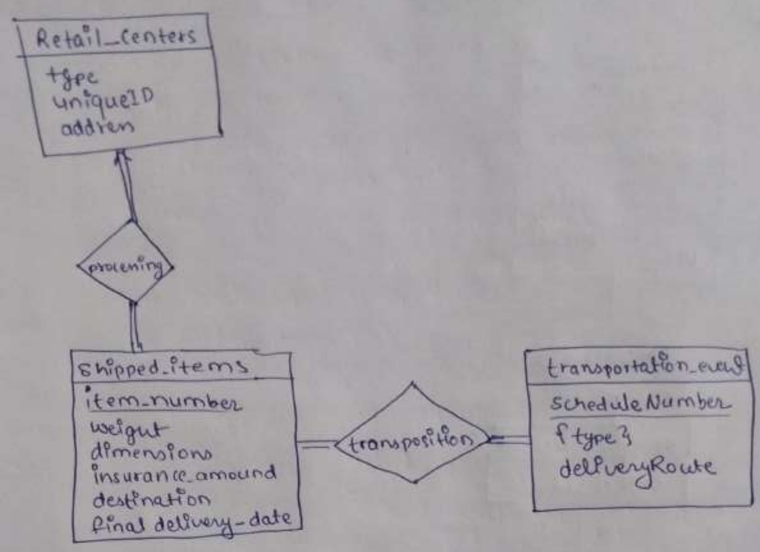
Jagamohan Nagar, Jagamara, Bhubaneswar, Odisha - 751030

Assignment-1

Q1.



Q2.



Q3.

Strong Entity Sets

a) Project
- name
- budget
- Location

simple and single value Attributes

b) Employee
- birth_date
- ssn
- status
- sex
- salary
- addren

simple and single valued Attributes

- name
  - first_name
  - middle initial
  - last_name

composite Attribute

c) Department
- no_of_employees → Derived Attribute
- Locations → Multivalued Attribute
- name → simple and single valued Attribute

Weak Entity Sets

a) Department

- name
- sex
- birth_date
- relationship

] simple and single valued Attributes

Relationship Sets

a) Assigned to

- Entity sets - Project to Department
- Mapping cardinality - many to one
- Participation constraint - Total to partial
- Degree - 2

b) Supervision

- Entity sets - Employee to Employee
- Mapping cardinality - One to Many
- Participation constraint - partial to Partial
- Degree - 2

c) Employed

- Entity sets - Employee to Department
- Mapping cardinality - Many to One
- Participation constraint - Total to Total
- Degree - 2
- Descriptive Attribute - Start date

d) Manages

- Entity sets - Employee to Department
- Mapping cardinality - One to One
- Participation constraint - partial to Total
- Degree - 2

e) Works

- Entity sets - Employee to Project
- Mapping cardinality - Many to Many
- participation constraint - Total to Total
- Degree - 2
- Descriptive Attributes - start-date, hours.

f) Identifying Relationship Set

Dependents of'

- Entity Sets - Employee to Department
- Mapping cardinality - one to Many
- Participation Constraint - Partial to Total
- Degree - 2

Qu.

Step-1: Mapping of Strong Entity sets:

* project (pname, budget, location)

* employee (ssn, birth_data, status, sex, salary, address, ename)

* department (dname, locations, number_of_employees)

Step-2: Mapping of weak Entity sets:

dependent (ssn, name, sex, birth-date, relationship)

Step-3: Mapping of Relationship sets:

* assigned to (pname, dname) → Many to One

supervision (supervisee, supervisor) → Recursive

* employed (ssn, dname, start-date) → Many to One

manages (ssn, dname, start-date) → One to One

works (ssn, pname, start_date, hours) → Many to Many

* There will be no table in Relational model for the identifying relationship set 'dependents of'.

Step-4: Mapping of Many to One and Total

project (pname, budget, location, dname)

* employee (ssn, birth_date, status, sex, salary, address, ename, dname, start_date)

Step-5: Mapping of multivalued attributes

departments (dname, number_of_employees)

departments (dname, locations)

Step-6: Mapping of composite Attributes

employee (ssn, birth_date, status, sex, salary, address, first_name, middle_initial, last_name, dname, start_date)

Final set of Tables:-

- dependent (ssn, name, sex, birth_date, relationship)
- supervision (supervisee, supervisor)
  - ↳ Foreign keys refering employee.
- manages (ssn, dname, start_date)
  - ↳ Foreign key refering department
  - → foreign key refering employee
- works (ssn, pname, start_date, hours)
  - ↳ Foreign key refering project
  - → Foreign key refering employee
- project (pname, budget, location, dname)
  - ↳ Foreign key refering department
- department (dname, number_of_employees)
- departments (dname, location)
  - ↳ Foreign key refering department
- employee (ssn, birth_date, status, sex, salary, address, first_name, middle_initial, last_name, dname, start_date)
  - ↓ Foreign key refering department

Q5.

Step-1: Mapping of strong entity sets

× Customer (cid, cname, phone, address, DOB, age)

× Loan (Lno, Lamount)

× account (accno, balance)

branch (brid, brname, city, assets)

Step-2: Mapping of weak entity sets

Payment (Lno, pno, pdate, pamount)

Step-3: Mapping of Relationship set

borrower (cid, Lno) → Many to Many

depositor (cid, accno, access_date) → Many to Many

× Loan_branch (Lno, brid) → Many to One

× Account_branch (accno, brid) → Many to One

* There will be no table in relational model for the identifying relationship set 'Loan-Payment'.

Step-4: Mapping of Many to One and Total

Loan (Lno, Lamount, brid)

account (accno, balance, brid)

Step-5: Mapping of Multivalued Attribute

* Customer (cid, cname, addren, DOB, age)
  Customers(cid, phone)

Step-6: Mapping of composite Attribute

Customer (cid, cname, city, state, pincode, DOB, age)

Step-7: Mapping of Generalization

Here, the given generalisation is total and disjoint.

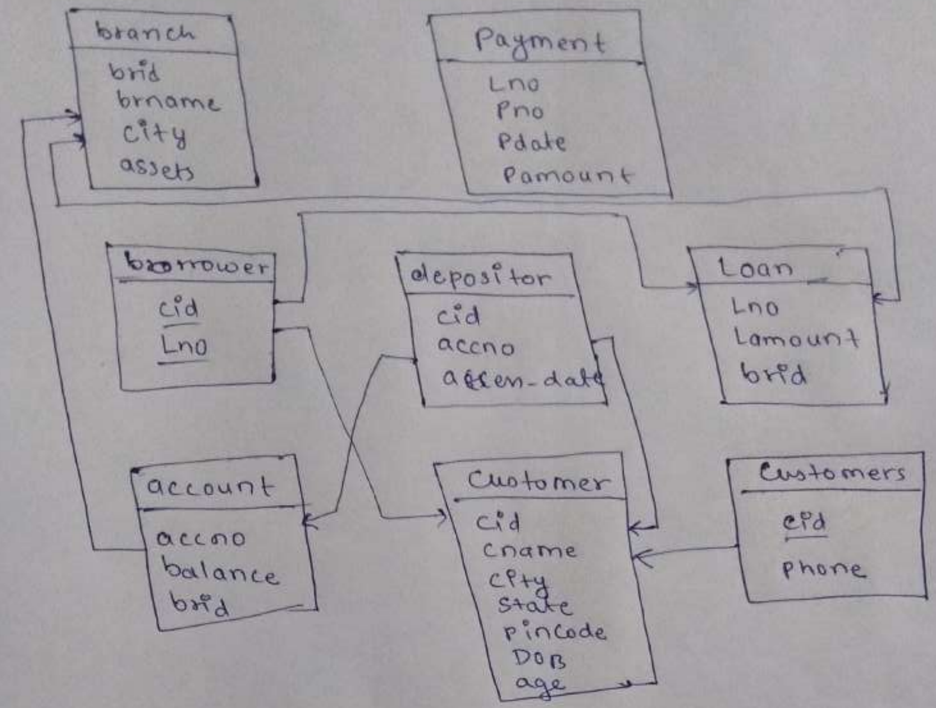
saving account(accno, balance, interest_rate)

Checking account(accno, balance, overdraft_amount)

Final set of tables:

- branch (brid, brname, city, assets)
- payment (Lno. pno., pdate, payment)
- browser (cid, Lno)
  - Lno → Foreign key refering Loan
  - cid → Foreign key refering customer
- depositor (cid, accno, accen_date)
  - accno → Foreign key refering account
  - cid → Foreign key refering customer
- Loan (Lno, Lamount, brid)
  - brid → Foreign key refering branch
- Account (accno, balance, brid)
  - brid → Foreign key refering branch
- customer (cid, cname, city, state, pincode, DOB, age)
- customer (cid, phone)
  - cid → Foreign key refering customer.
- saving account (accno, balance, interest_rate)
- checking account (accno, balance, overdraft_amount)

06. Schema Diagram



Q7. person (driver_id, name, address)

Car (license, model, year)

accident (report_number, date, location)

owns (driver_id, License)

↳ Foreign key refering car

↳ Foreign key refering person

Participated (report_number, License, driver_id, ~~damage_~~ damage_amount)

Foreign key refering accident

Foreign key refering car

Foreign key refering Person

Name: Bali Babu Chauhan

Redg. no. :- 1941012182

Branch/sec :- CSE-M

Subject :- Introduction to Database

Assignment - 2

1.

let Student_mark = R , regd = A , name = B , course_id = C, title = D , grade = E

$F = \{A \rightarrow B,\ C \rightarrow D,\ AC \rightarrow E\}$

partial dependency 2NF × (prime → Non prime)

C. key

$AC = [AC]^+ = \{ACEBD\}$

prime attributes : A, C

non-prime at. : B,D,E

So, highest order currently = 1NF

It doesn't satisfy 2NF currently because C→D has partial dependency and also A→B also have partial dependency. Thus, R is not in 2NF

For 2NF, we have to decompose it into $R_1(ACE), R_2(CD), R_3(AB)$

$R_1(ACE)$	$R_2(CD)$	$R_3(AB)$
$AC \rightarrow E$	$C \rightarrow D$	$A \rightarrow B$
key = AC	key = C	key = A
$[AC]^+ = \{ACE\}$	$[C]^+ = \{CD\}$	$[A]^+ = \{AB\}$

* Now checking properties of Decomposition:

* $R_1 \cap R_2 = C = [C]^+ = \{CD\} = R_2$

→ Loss-Less Decomposition

* $\rightarrow$ $R_2 \cap R_3 = A = [A]^+ = \{AB\} = R_3$

$\Rightarrow$ Loss-less Decomposition

The decomposition is Dependency Preserving Decomposition because the decomposition set has all the functional dependeny present in the actual relation.

* Now Checking for BCNF

$\rightarrow$ Since all the relation after decomposition (R_1, R_2, R_3) have direct functional dependency as the key for that relation.
Thus, it satisfy BCNF.
As it satisfy BCNF, it also statisfy all the lower order, i.e, 3NF and 2NF

Hence, it satisfy all the properties of decomposition.

$D = \{R_1(ACE), R_2(CD), R_3(AB)\}$ - 2NF

2

let BOOK = R, Title = A, Author = B, Catalog_no = C, Publisher = D, Year = E, Price = F

$FD = \{AB \rightarrow CF, C \rightarrow A, \underbrace{C \rightarrow D}_{P.D}, \underbrace{C \rightarrow E}_{P.D}\}$

× 2NF × 3NF   × 2NF × 3NF

Candidate Key:

① $[BC]^+ = \{BCADEF\}$

② $[AB]^+ = \{ABCFDE\}$

Now, checking for 3NF: There are two dependencies, i.e, $C \rightarrow D$ and $C \rightarrow E$, there both dependencies are having partial dependency, this is why it does not satisfy 2NF.

$\Rightarrow$ It does not satisfy 3NF too.

Now, decompose $R(ABCDEF)$ into $R_1(ABCF)$ and $R_2(CDE)$

$R_1(ABCF)$	$R_2(CDE)$
$AB \rightarrow CF$	$C \rightarrow D$
$C \rightarrow A$	$C \rightarrow E$
key = AB	key = C
$[AB]^+ = \{ABCF\}$	$[C]^+ = \{CDE\}$

* Now checking properties of decomposition:

* → $R_1 \cap R_2 = C = [C]^+ = \{CDE\} = R_2$

  ⇒ Loss-Less join decomposition

→ The decomposition is Dependency preserving because the decomposition set $D = \{R_1(ABCF), R_2(CDE)\}$ has all the functional dependency present in R(ABCDE)

* Checking for 3NF

After decomposition, the function dependencies $C \to D$, $C \to E$ change, to the form super key → Non-prime, which comes under 3NF, Thus no more issues present in the decomposed relation.

Thus, the decomposition is now in 3NF.

$D = \{R_1(ABCF), R_2(CDE)\}$

3)

R(ABCD)

a) $F = \{AB \to CD, C \to A, D \to B\}$

C.Key, 1) $[AB]^+ = \{ABCD\}$

2) $[CB]^+ = \{CBAD\}$

3) $[AD]^+ = \{ADBC\}$

4) $[CD]^+ = \{ABCD\}$

Here, $C \to A$ and $D \to B$ does not satisfy the properties of BCNF, i.e., C, D not S.K. thus it is not in BCNF.

b) BCNF decomposition of the schema is not desirable because here all A, B, C, D are prime attributes, i.e, all the attributes are prime attributes. if we divide in any relations (decompose it), then we can not satisfy the Dependency Preserving Decomposition property of decomposition. Thus, 3NF is the highest relation possible.

4.

a) R(ABCDEG)

FD = {A → BCD, BC → DE, B → D, D → A}

Proving [AG] is a super key for that, we have to use armstrong's axioms, ie. finding the closure.

$[AG]^+ = \{ABCDEFG\}$

Since AG determines all the attributes of relation R, thats why AG i.e a super key.

b) for finding minimal cover, we have to do the five steps :-

Step-1: Split function dependency set in a single attribute.

$$F = \left\{ \begin{array}{l} A \to B, \\ A \to C, \\ \cancel{A \to D} \end{array} \quad \begin{array}{l} \cancel{BC \to D}, \\ BC \to E \end{array} \quad , B \to D, \quad D \to A \right\}$$

Step-2: Remove trivial function dependeny of the above step (since, no trial function dependency, no change)

Step-3: Remove extraneous attribute over every determinant at step: 2.

We can see that we can directly get D from B→D, thus, C in BC→D is an extraneous attribute, so delete C in BC→D and this is changed to B→D.

Step-4: Remove Redundant functional dependency from Step-3

① A→B, B→D, given we don't need to have A→D, thus remove A→D. now

② B→D, B→D (one from step-3) are present twice. Thus remove one.

Step-5: Merge all functional dependency of Step-4 if determinants are same :-

$F_m = \{A \to BC, BC \to E, B \to D, D \to A\}$

Now, for BCNF, we can see in the FD that A is not a SK, BC is not a SK, B is not SK and also D is not a SK. Thus, the FD set is needed to be decomposed.

R_1(ABCDE)	R_2(AG)
$A \to BCD$, $BC \to DE$, $B \to D$, $D \to A$	

C.key: $[A]^+ = \{ABCDE\}$
$[D]^+ = \{DABCE\}$
$[B]^+ = \{BDACE\}$
S.K = $[BC]^+ = \{BCDEA\}$

Now, this decomposition satisfy Dependency preserving decomposition because all FD present in R is present in decomposed relation.

Now, $R_1 \cap R_2 = A = [A]^+ = \{ABCDE\} = R_1$
$\Rightarrow$ Loss-less join decomposition.

Since, now BC is also SK, and A, B, D is CK. for R_1, we can say that the decomposition satisfy BCNF.

Hence, BCNF decomposition for R is
R_1(ABCDE) and R_2(AG)

5. Relational Schema:

Employee (ename, ecity, state)
Works (ename, companyname, salary)
Company (companyname, city)
Manages (ename, managername)

a) Relation Algebra:-

$$\left[\pi_{ename} \left(\sigma_{\substack{Works.salary > 60000 \\ Works.companyname = "TCS"}} (Employee \bowtie Works) \right) \right] \cup$$

$$\left[\pi_{e\text{-}name} \left(\sigma_{\substack{Works.Salary > 60000 \\ Works.companyname = "Wipro"}} (Employee \bowtie Works) \right) \right]$$

c) 3NF decomposition based on canonical cover.

Candidate key:

[Skb = Super key]

1> $[AG]^+ = \{AGBCDE\}$ $[BC]^+ = \{BCADE\}$

2) $[BG]^+ = \{BGACDE\}$ $[A]^+ = \{ABCDE\}$

3. $[DG]^+ = \{DGABCE\}$

$B \to D$, $BC \to E$ and $A \to BC$, dependency is not satisfying 3NF ~~the~~ form because

* B is prime (it must be SK) but only B is not SK.
* for $BC \to E$, BC is not SK, thus we can say that it is also not in 3NF.
* $A \to BC$, A is prime but not SK, thus it is not in 3NF.

Decomposing into $R_1(ABCDE)$ and $R_2(AG)$.

$R_1(ABCDE)$	$R_2(AG)$
$A \to BC$, $B \to D$	
$BC \to E$, $D \to A$	

C.key: $A = [A]^+ = \{ABCDE\}$

$B = [B]^+ = \{BDACE\}$

$D = [D]^+ = \{DABCE\}$

S.K.: $[BC]^+ = \{BCDEA\}$

All the relation in $R(ABCDEG)$ present in $R_1(ABCDE)$ and $R_2(AG)$. Thus, it is dependency preserving decomposition.

$R_1 \cap R_2 = A = [A]^+ = \{ABCDE\} = R_1$

$\rightarrow$ loss-less join decomposition

Since BC is SK, it does not have an issue in 3NF.

$\Rightarrow$ The decomposed relation is in 3NF.

d) BCNF decomposition of R using original function dependency:

Candidate key

$[AG]^+ = \{ABCDEG\}$ $[BC]^+ = \{ABCDE\}$

$[BG]^+ = \{ABCDEG\}$ $[A]^+ = \{ABCDE\}$

$[DG]^+ = \{ABCDEG\}$ $[B]^+ = \{ABCDE\}$

$[D]^+ = \{ABCDE\}$

{FD $\to$ function dep.
{SK $\to$ super key

Tuple - relational Calculus:

$\{ t \mid t \in Employee\ \exists s \in works\ (t[e.name] = s.[e.name] \wedge$

$s[salary] > 60000 \wedge s[companyname] = "wipro" \vee "TCS" \}$

Domain relational Calculus:

$\{ <ename> \mid \exists <ecity, state> (<ename, company\text{-}name, salary>$

$\in works) \wedge salary > 60000 \wedge company\text{-}name = "TCS" \wedge "wipro" \}$

b) Relational Algebra

$$\pi_{salary\ name} \left(\sigma_{company \times city = "DELHI"} (works \bowtie Company) \right)$$

Tuple Relational Calculus

$\{ t \mid \exists s \in company\ (t[company\text{-}name] = s[company\text{-}name]) \wedge$

$s[city] = "DELHI" \}$

Domain Relational calculus

$\{ <ename, salary> \exists <company\_name> (<company\_name, city>$

$\in Company) \wedge City = "DELHI" \}$

c) Relational Algebra

$$\pi_{salary} \left(\sigma_{\substack{company.city = "MUMBAI" \wedge \\ Manages.manager\ name = "JOHN"}} (works \bowtie Company \bowtie Manages) \right)$$

Tuple Relational Calculus:-

$\{ <salary> \exists <ename, company\text{-}name> (<company\_name, city> \in$

$Company) (<ename, managername> \in Manges) \wedge (city = "MUMBAI")$

$\wedge (managername = "JOHN") \}$

6.

* Schedule S

→ S: R1(A), R2(A), R1(B), R2(B), R3(B), W1(A), W2(B)

Ri(A) represents read operation by Transaction Ti on A
Wi(B) represents write operation by Transaction Ti on B

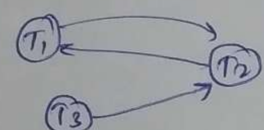
Representing schedule in Tabular form:

T_1	T_2	T_3
R(A)		
	R(A)	
R(B)		
	R(B)	
		R(B)
W(A)		
	W(B)	

Conflict pairs

R—W
W—R
W—W

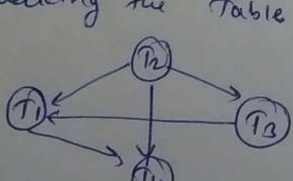
* Checking the operations and drawing the precedence graph [if no conflict, cut the operation in graph (a way of making checked)]



*Loop/cycle check:
We can see that two loops
i) $T_2 - T_1 - T_2$
ii) $T_1 - T_2 - T_1$
are present in the precedence graph. Hence, S is not a conflict serializable schedule.

7.

Checking the Table operations and drawing the precedence graph



Loop/cycle check:
Here we can check that there is no loop or cycle present
→ The graph is conflict serializable
→ Graph is serializable
→ Graph is consistent

Serializable sequence ⇒ $\boxed{T_2 \rightarrow T_3 \rightarrow T_1 \rightarrow T_4}$

Recoverability:
There is a presence of dirty read in transactions T_1, T_3 and T_4 suppose just commit, T_2 rollsback due to an error but T_3 and T_1's a write (X) had a dirty read from T_2's READ(X) and T_3 & T_1 has already commited. Thus, now they cannot Rollback, thus the making the schedule not recoverable.