COMP2400/6240 - Relational Databases Assignment 2 (Database Theory)

Name: Razeen Wasif

UID: u7283652

Solution 1:

Entity types:

- Employees:
    - Employee number (key)
    - Administrator (sub entity) – work type (multivalued attribute): remote, onsite, hybrid
    - Distributors (sub entity) – phone number (key)
    - Technicians (sub entity) – has qualification level.
        - Constraint on Specialisation – total. Every entity in the superclass must be a member of at least one subclass.
- Customer:
    - Credit card number (key)
    - Home address
- Scooter:
    - Brand
    - Manufacture year
    - ID (key)
- Ride:
    - Fare price
    - Unique reference no. (key)
    - Scooter id (key)
    - Start time
    - End time
    - Origin
    - Destination

Relationship types:

- Moves:
    - Distributor moves Scooter. Assume 1 Distributors can move around N many Scooters. Every distributor must move around scooters (total) so every scooter must be moved (total).

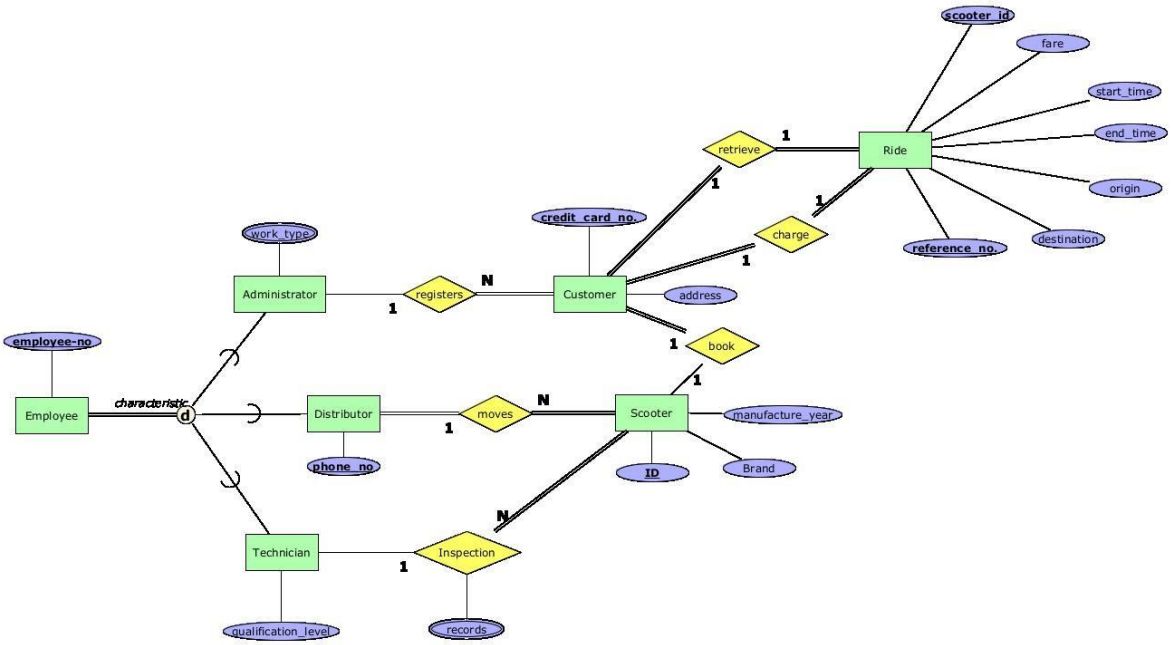
- Inspection:
    - Technician inspects Scooters. Assume 1 Technician can inspect multiple Scooters. It says each technician inspects scooters which would mean one technician inspects a scooter at a time. More than one technician doesn't inspect the same scooter at the same time.
    - Assume not all technician inspects scooter (partial) depending on their qualification. But all scooters must be inspected (total).
    - Has attribute: records which contain date and feedback

- Register:
    - Admin registers Customer's cred no. and address.
    - Assume 1 admin can register many customers
    - Assume Not every admin has to register customers. (Doesn't say each administrator registers customers).
    - Assume Every customer must be registered.

- Retrieve:
    - Customer retrieve Ride info. 1:1 because every ride is booked by one customer.

- Book:
    - Scooter booked by one customer. 1:1
    - Every customer must book a scooter (total) or else they wouldn't be a customer
    - Not every scooter will be booked by a customer (partial)

- Charged:
    - Ride charges Customer. 1:1
    - Every ride charge customer (total)
    - Every customer is charged after ride (total)



Cardinality ratios:

- Every customer must be registered, and every administrator may not register customers:
  Administrator – 1 – registers – N – Customer
- Every distributor moves scooters, and every scooter must be moved by a distributor:
  Distributor – 1 – moves – N – Scooter
- A technician may inspect scooters and every scooter must be inspected:
  Technician – 1 – inspection – N – Scooter
- A customer must book a scooter and not every scooter must be booked by customer:
  Customer – 1 – books – 1 – Scooter
- A customer must be able to retrieve ride info and every ride info can be retrieved by customer:
  Customer – 1 – retrieve – 1 – Ride
- Every customer gets charged for ride, and every ride charges customer:
  Customer – 1 – charge – 1 – Ride

Requirements that cannot be captured:

Work_type. Administrators have three different work type (onsite, remote and hybrid)

Records. Technician records the date and feedback of every inspection.

Solution 2:

R = {A, B, C, D, E}

set Σ of FDs:

• AB → C • BC → A • C → DE • DE → B

2.1. The Candidate keys (CK) of R:

$ABCDE^+$ = ABCDE (all attributes is always a superkey)

Remove an attribute from the superkey such that the closure is still a superkey.

Discard C because AB → C

$ABDE^+$ = ABDE

= ABCDE (by AB → C)

Discard DE because C → DE and since AB → C, therefore AB → DE

AB^+ = AB

= ABC (by AB → C)

= ABCDE (by C → DE)

No more attributes can be removed. Check whether a proper subset of AB is a key.

A^+ = A

B^+ = B

Proper subsets of AB are not superkeys; therefore, **AB** is a candidate key.

A and B are prime attributes.

Prime attributes: {A , B}

If prime attributes are present in the right-hand side of any functional dependency, then there will be more candidate keys in the relation. In this case:

First case: • BC → A Prime attribute A is present on the right-hand side

Replace A in CK with BC to get BC. Now check proper subset of BC:

B^+ = B

C^+ = CDE (by C → DE)

= CBDE (by DE → B)

= ABCDE (by BC → A)

Closure of C is a superkey therefore BC is not candidate key. But **C** is a candidate key because no proper subset of C is a superkey.

Second case: • DE → B Prime attribute B is present on the right-hand side

Replace B in CK with DE to get ADE. Now check proper subset of ADE:

We know closure of A is not superkey.

$D^+ = D$

$E^+ = E$

$AD^+ = AD$

$AE^+ = AE$

$DE^+ = BDE$ (by DE → B)

Proper subsets of ADE are not superkeys; therefore, **ADE** is a candidate key

Prime attributes: {A , B, C, D, E}

Third case: • AB → C prime attribute C is present in the right-hand side

Replace candidate key C with AB. We already know AB is candidate key.

Fourth case: C → D Prime attribute D is present on the right-hand side

Replace D with C to get ACE and check proper subset of ACE:

$A^+ = A$

$E^+ = E$

$C^+ = CDE$ (by C → DE)

= CBDE (by DE → B)

= ABCDE (by BC → A)

ACE is not a candidate key

Fifth case: C → E Prime attribute E is present on the right-hand side

Replace E with C to get ADC and check proper subset of ADC:

$C^+ = CDE$ (by C → DE)

= CBDE (by DE → B)

= ABCDE (by BC → A)

ADC is not CK.

Therefore, the Candidate Keys are: AB, ADE, and C.

2.2 Minimal cover of • AB → C • BC → A • C → DE • DE → B

Step 1:

Split the FD's such that right-hand side only have one attribute

• AB → C • BC → A • C → D • C → E • DE → B

Step 2:

Remove redundant attributes

From • C → D • C → E • DE → B it can be seen that C can imply B (C → B).

Since C → B, BC → A can be reduced to C → A. Therefore, we have:

• AB → C • C → A • C → D • C → E • DE → B

Step 3:

Remove redundant functional dependencies

For AB → C, Not considering AB → C :

AB^+ = AB (does not include C, C cannot be removed)

For C → A, Not considering C → A:

C^+ = CDEB (does not include A, A cannot be removed)

For C → D, Not considering C → D:

C^+ = CAE (does not include D, D cannot be removed)

For C → E, Not considering C → E:

C^+ = CAD (does not include E, E cannot be removed)

For DE → B, Not considering DE → B:

DE^+ = DE (does not include B, B cannot be removed)

Minimal Cover: { AB → C , C → A, C → D, C → E, DE → B }

= { AB → C , C → ADE, DE → B }

2.3 Are Σ and Σ_1 equivalent or not?

R = {A, B, C, D, E}

set Σ of FDs:

• AB → C • BC → A • C → DE • DE → B

set Σ_1 of FDs:

• AB → CDE • C → AB • DE → B

Step 1. Check whether all FDs of Σ are present in Σ_1:

- DE → B in Σ is present in Σ_1
- AB → C is not directly present in Σ_1 so we'll try derive it in Σ_1:
  - AB^+ = ABCDE. AB can functionally determine A,B,C,D, and E so AB→ C will also hold in Σ_1.
- BC → A is not directly present in Σ_1 so we'll try derive it in Σ_1:
  - BC^+ = BCADE. BC can functionally determine A,B,C,D, and E so BC → A will also hold in Σ_1.
- C → DE is not directly present in Σ_1 so we'll try derive it in Σ_1:
  - C^+ = CABDE. C can functionally determine A,B,C,D, and E so C → DE will also hold in Σ_1.

As all FDs in Σ also hold in Σ_1, Σ_1 is a subset of Σ

Step 2. Check whether all FDs of Σ_1 are present in Σ:

- DE → B in Σ_1 is present in Σ
- AB → CDE is not directly present in Σ, so we'll try derive it in Σ:
  - AB^+ = ABCDE. AB can functionally determine A,B,C,D, and E so AB→ CDE will also hold in Σ.
- C → AB is not directly present in Σ, so we'll try derive it in Σ:
  - C^+ = CDEBA. C can functionally determine A,B,C,D, and E so C → AB will also hold in Σ.

As all FDs in Σ_1 also hold in Σ, Σ is a subset of Σ_1

Since both Σ and Σ_1 are subsets of each other, $\Sigma = \Sigma_1$. The two FD sets are equivalent.

Solution 3:

Appointment={Patient, GP, Date, Time, Clinic, Room}

set Σ of FDs:

• Patient, Clinic, Date → Time

• Patient → GP

• GP → Clinic

• Clinic, Date, Time, Room → Patient

• Patient, Date, Time → Clinic, Room

Let:

Patient = P

GP = G

Date = D

Time = T

Clinic = C

Room = R

For a relation to be in BCNF, for each non-trivial FD X → Y, X must be a super key.

In the FD's Patient → GP and GP → Clinic, Patient and GP are not superkeys.

$Patient^+$ = Patient, GP, Clinic

GP^+ = GP, Clinic

Therefore, Appointment is not in BCNF.

BCNF decomposition for Appointment:

Start the Algorithm:

Check which FDs violate BCNF in R:

P → G and G → C violates BCNF because P and G are not a superkey.

Create two sub schemas:

XY and (R – Y) where Y is the violating FD

(R-Y) = (PGDTCR – GC)

R1 (PGC) and R2 (PDTR)

Calculate F1 and F2 from F:

$P^+ = \cancel{P}GC$ $PG^+ = \cancel{PG}C$ $P^+ = \cancel{PGC}$ $PD^+ = \cancel{PGCD}TR$ $DR^+ = \cancel{DR}$

$G^+ = \cancel{G}C$ $PC^+ = \cancel{PC}G$ $D^+ = \cancel{D}$ $PT^+ = \cancel{PGCT}$ $TR^+ = \cancel{TR}$

$C^+ = C$ $GC^+ = \cancel{GC}$ $T^+ = \cancel{T}$ $PR^+ = \cancel{PGCR}$

$R^+ = \cancel{R}$ $DT^+ = \cancel{DT}$

F1 : P → G, G → C

F2 : PD → TR

PD is a key in R2, so PDTR is in BCNF

F1 violates BCNF so decompose R1(PGC): R11 (PG) and R12 (GC)

Calculate F11 and F12 from F1:

$P^+ = \cancel{P}G$ $C^+ = \cancel{C}$

$G^+ = \cancel{GC}$ $G^+ = \cancel{G}C$

F11: P → G. P is a key

F12: G → C . G is a key

R11 and R12 is in BCNF.

Final decomposition: (PDTR) (PG) (GC)

Is it dependency preserving:

(F11 U F12) U F2 = F must be true.

CDTR → P from F cannot be derived in (F11 U F12) U F2 therefore the decomposition is not dependency preserving.

Solution 4:

4.1 [a] List the phone numbers of students who studied COMP2400 in 'S2 2020' and became a tutor for COMP2400 in 'S2 2021'.

If status is active for COMP2400 in S2 2021 in Enrol, then assume to be student, not tutor.

```
# List the phone numbers of students who studied COMP2400 in 'S2 2020' and became a tutor for
# COMP2400 in 'S2 2021'.
# if active then they are student, not tutor.
select Student_join_Tutor.Phone
from Student_join_Tutor
where Student_join_Tutor.CourseNo="COMP2400" and Student_join_Tutor.Semester="s2 2021" and Student_join_Tutor.Phone not in
(select Student_join_Enrol.Phone
from Student_join_Enrol
where Student_join_Enrol.CourseNo="COMP2400" and Student_join_Enrol.Status_="active" and Student_join_Enrol.Semester="S2 2020");

create table Student_join_Tutor as
(select *
from Students natural join Tutors
where Students.SID=Tutors.TID);

create table Student_join_Enrol as
(select *
from Students natural join Enrol
where Students.SID=Enrol.SID);
```

Using Relational Algebra:

R1: $\pi_{\text{phone}}(\sigma_{\text{Student.SID=Tutor.TID} \wedge \text{CourseNo="COMP2400"} \wedge \text{Semester="S2 2021"}}(\text{Student} \bowtie_{\text{Student.SID=Tutor.TID}} \text{Tutor}))$

R2: $\pi_{\text{phone}}(\sigma_{\text{CourseNo="COMP2400"} \wedge \text{Status="active"} \wedge \text{Semester="S2 2021"}}(\text{Student} \bowtie_{\text{Enrol.SID=Student.SID}} \text{Enrol}))$

Result : $\pi_{\text{phone}}(R1 - R2)$.

[b] List the TIDs of tutors who had tutored exactly one course in 'S2 2021'.

```
# [b] List the TIDs of tutors who had tutored exactly one course in 'S2 2021'.
create table Tutors_join_Course as
(select *
from Tutors natural join Course
where Tutors.CourseNo=Course.CourseNo and Tutors.Semester=Course.Semester);

select Tutors_join_Course.TID
from Tutors_join_Course
where Tutors_join_Course.TID not in
(select TID
from Tutors_join_Course
group by TID
having count(TID) > 1);
```

R1: $\pi_{\text{TID}}(\text{Tutor} \bowtie_{\text{Tutors.CourseNo=Course.CourseNo} \wedge \text{Tutors.Semester=Course.Semester}} \text{Course})$

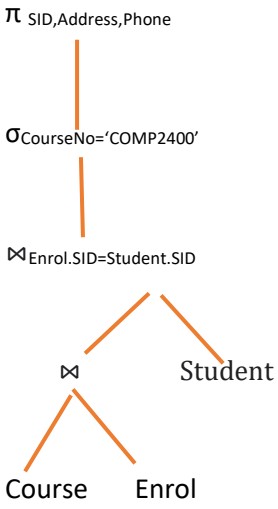
R2: $\pi_{\text{TID}}(\sigma_{\text{count(TID)} > 1}(\text{Tutor} \bowtie_{\text{Tutors.CourseNo=Course.CourseNo} \wedge \text{Tutors.Semester=Course.Semester}} \text{Course}))$

Result : $\pi_{\text{TID}}(R1 - R2)$.

4.2 Optimise the following relational algebra query (Your marks will depend on how well you optimise the query in your solution). Additionally, draw the query trees of the query before and after your optimisation.

$\pi_{SID,Address,Phone}(\sigma_{CourseNo='COMP2400'}((Course \bowtie Enrol) \bowtie_{Enrol.SID=Student.SID} Student))$

Tree before optimisation:



Optimisation:

$\pi_{SID,Address,Phone}((Course \bowtie Enrol) \bowtie_{Enrol.SID=Student.SID \wedge CourseNo='COMP2400'} Student)$

Tree after optimisation: