National University of Singapore
CS2102: Database System

Final Assessment AY2023/24 Sem 2

Instructions

1. Please read the **Instructions** carefully.
2. This assessment paper contains **12 (TWELVE)** questions and comprises of **8 (EIGHT)** printed pages.
3. The total mark for the assessment is **100 Points**.
4. **Answer ALL questions**.
5. All answers are to be written in the space provided in the answer sheet.
   - Any answer **NOT** in the space provided will **NOT** be graded.
6. This is a **CLOSED-BOOK** assessment
   - You are allowed **1 (ONE)** A4-sized double-sided cheatsheet.
   - Calculators are **NOT** allowed.
7. All codes are run on PostgreSQL 16.
8. We will use the *shorthand* notation for the functional dependencies.
   - Instead of writing $\{A, B\} \rightarrow \{C, D\}$, we will write $AB \rightarrow CD$.
   - You may use the same notation on your answer.
9. The duration of the assessment is **2 (TWO)** hours.

Marks

Section	Questions	Points
A	1 - 4	42
B	5 - 6	14
C	7 - 12	44
	Total	**100**

Good Luck!

A. Relational Model 42 Points

For the next **3 (THREE)** questions, consider the relations $R_1(A, B)$, $R_2(C, D)$, $R_3(A, C, E)$, and $R_4(A, C, F)$. Furthermore, consider the *natural join* of all the relations above. We have the following set of functional dependencies that holds on the ER diagram **AND** on the natural join of all the relations above.

$$\Sigma = \{A \rightarrow B,\ C \rightarrow D,\ A \rightarrow CE,\ C \rightarrow AF\}$$

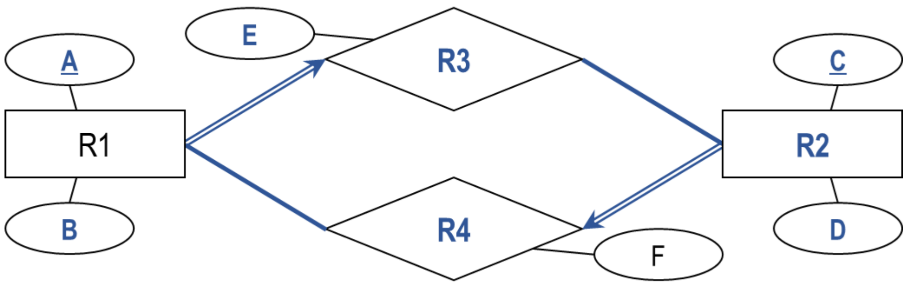
You may assume that the following foreign key constraints are also satisfied:

$(R_3.A) \rightsquigarrow (R_1.A)$ $(R_4.A) \rightsquigarrow (R_1.A)$

$(R_3.C) \rightsquigarrow (R_2.C)$ $(R_4.C) \rightsquigarrow (R_2.C)$

1. (12 points) Complete the ER diagram that corresponds to the relations above and satisfies the functional dependencies as well as foreign key constraints.

Comments:



Requirements:

- A and C are primary keys of their respective relations (*due to $A \rightarrow B$ and $C \rightarrow D$ respectively*), so they need to be underlined.
- R_2, R_3, and R_4 can only be in the respective rectangle and diamond. Due to the presence of the given attributes and/or relation name, there are no alternative possibilities.
- The attributes A to E are attached to the correct construct.
- We accept both double-line and single-line versions for all connections (*except on attributes*).
  - Connection between R_1 and R_3 must have arrow due to $A \rightarrow CE$.
  - Connection between R_2 and R_4 must have arrow due to $C \rightarrow AF$.

Note: because the functional dependencies imply that $A \rightarrow CF$ and $C \rightarrow AE$, we also allow arrow from R_2 to R_3 and from R_1 to R_4.

2. (10 points) Complete the schema that corresponds to the relations above and satisfies the functional dependencies as well as foreign key constraints. Additionally, ensure that none of the attributes can be `NULL`.

Continue on the next page

Comments:

The following is the only alternative (*with minor allowances for some redundant constraint such as* `NOT NULL` and/or `UNIQUE` *in addition to* `PRIMARY KEY`). Recap that A is the primary key of R_3 due to $A \rightarrow CE$ and C is the primary key of R_4 due to $C \rightarrow AF$.

```
CREATE TABLE R1 (
  A  INT  PRIMARY KEY,
  B  INT  NOT NULL
);
```

```
CREATE TABLE R2 (
  C  INT  PRIMARY KEY,
  D  INT  NOT NULL
);
```

```
CREATE TABLE R3 (
  A  INT  PRIMARY KEY
    REFERENCES R1 (A),
  C  INT  NOT NULL
    REFERENCES R2 (C),
  E  INT  NOT NULL
);
```

```
CREATE TABLE R4 (
  A  INT  NOT NULL
    REFERENCES R1 (A),
  C  INT  PRIMARY KEY
    REFERENCES R2 (C),
  F  INT  NOT NULL
);
```

3. (8 points) Select **ALL** queries that *may* produce duplicate, namely, multiple rows that have the same value on each attributes. We assume that the relational instance satisfies the constraints in the schema (*including the constraint that none of the attributes can be* `NULL`) as well as all the functional dependencies and foreign key constraints. You may assume that all queries are valid queries. We will use the `monospace` typeface `R1` instead of R_1 in our queries.

(A) Query #1

```
SELECT R1.A, R2.C
FROM   R1, R2;
```

(✓) Query #2

```
SELECT R1.B, R2.D
FROM   R1 NATURAL JOIN R2;
```

(C) Query #3

```
SELECT R2.C, R3.E
FROM   R2 NATURAL JOIN R3;
```

(D) Query #4

```
SELECT R2.C, R4.F
FROM   R2, R4
WHERE  R4.C = R2.C;
```

(✓) Query #5

```
SELECT R3.E, R4.F
FROM   R3 NATURAL JOIN R4;
```

Continue on the next page

Comments:

The basic reasoning is as follows. Note that we assume the functional dependencies is satisfied. So, $A \rightarrow C$ and $C \rightarrow A$ are satisfied all the time in all relations. This means that we cannot have the following:

- In `R3`, we cannot have `{(1, 10, 2), (2, 10, 2)}` as it would violate $C \rightarrow A$.
- In `R4`, we cannot have `{(10, 1, 2), (10, 2, 2)}` as it would violate $A \rightarrow C$.

Given that preliminary, we can explain the behaviour of the queries as follows.

(A) **Query #1:** Since `R1.A` is unique in `R1` and `R2.C` is unique in `R2`, the combination of both is unique in `R1` $\times$ `R2`.

(B) **Query #2:** A simple counterexample is

- `R1` = `{(1, 1), (2, 1)}`
- `R2` = `{(10, 1), (20, 1)}`

Since there are no common attributes, `R1` $\bowtie$ `R2` is equivalent to `R1` $\times$ `R2`. Query result is `{(1, 1), (1, 1), (1, 1), (1, 1)}`.

(C) **Query #3:** Assuming that all functional dependencies are satisfied, we know that $A \rightarrow C$ and $C \rightarrow A$. Additionally, `R2` $\bowtie$ `R3` is lossless-join due to $C \rightarrow \text{Attr}(R_2)$. Since $A \rightarrow E$, we also have $C \rightarrow E$ and so there will be no duplicate rows.

(D) **Query #4:** `C` is unique in both `R2` and `R4`, so will be unique here too since we only keep when `R4.C = R2.C`.

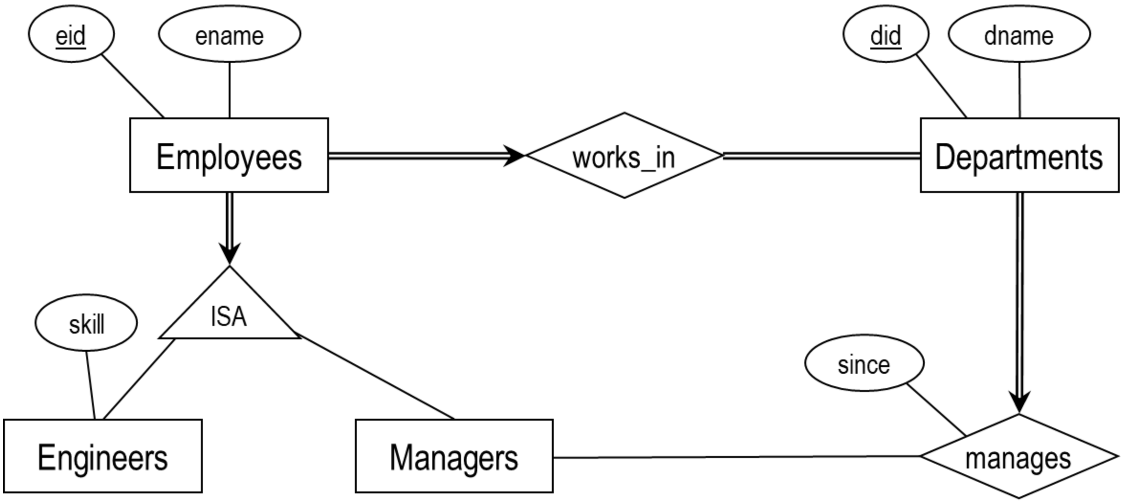
(E) Simple counterexample is

- `R3` = `{(1, 10, 1), (2, 20, 1)}`
- `R2` = `{(1, 10, 2), (2, 20, 2)}`

Query result is `{(1, 2), (1, 2)}`.

For each of the counterexample above, note that we satisfy all constraints including the functional dependencies.

4. (12 points) Consider the following ER diagram.



Further consider the following translation to schema. We will be using **VIEW**.

```
CREATE TABLE Departments (
  did   INT  PRIMARY KEY,
  dname TEXT NOT NULL
);

CREATE TABLE Engineers (
  eid    INT  PRIMARY KEY,
  ename  TEXT NOT NULL,
  skill  TEXT NOT NULL,
  did    INT  NOT NULL REFERENCES Departments (did)
);

CREATE TABLE Managers (
  eid    INT  PRIMARY KEY,
  ename  TEXT NOT NULL,
  did    INT  NOT NULL REFERENCES Departments (did)
);

CREATE VIEW Employees AS (
  SELECT eid, ename FROM Engineers
  UNION
  SELECT eid, ename FROM Managers
);

CREATE TABLE manages (
  did   INT  PRIMARY KEY REFERENCES Departments (did),
  eid   INT  NOT NULL REFERENCES Managers (eid),
  since DATE NOT NULL
);
```

Continue on the next page

Select **ALL** statements that are true about the relational schema.

(✓) It enforces cardinality constraint on `Employees` with respect to `works_in`.
(✓) It enforces total participation constraint on `Employees` with respect to `works_in`.
(✓) It enforces cardinality constraint on `Departments` with respect to `manages`.
(D) It enforces total participation constraint on `Departments` with respect to `manages`.
(E) It enforces total participation constraint on `Departments` with respect to `works_in`.
(✓) It enforces covering constraint of the ISA.
(G) It enforces overlap constraint of the ISA.
(H) It enforces the functional dependency $\{\texttt{eid}\} \rightarrow \{\texttt{ename}\}$.

Comments:

(A) To see this, note that both `Engineers` and `Managers` can only work in at most one `Departments` due to the use of `did` as an attribute. Since there are no other kinds of `Employees` due to the use of `VIEW`, this constraint is satisfied.

(B) Similar to above but due to the use of `NOT NULL` constraint.

(C) In `manages`, `did` is the primary key.

(D) There may be an entry for a particular `did` in `Departments` but not in `manages`.

(E) There may be an entry for a particular `did` in `Departments` but no `Engineers` or `Managers` references it.

(F) There are no other kinds of `Employees`, we either insert into `Engineers` or insert into `Managers`.

(G) We may have the same `eid` in both `Engineers` and `Managers`.

(H) We may have the same `eid` in both `Engineers` and `Managers` but with different `ename`.

Continue on the next page

B. Stored Procedures and Triggers 14 Points

For the next **2 (TWO)** questions, we will consider the following table `Scores` on the right. The data types of `sid` and `name` are *TEXT*, while the data type of `score` is *INT*.

sid	name	score
s1	Alice	50
s2	Bob	60
s3	Cathy	70
s4	David	80
s5	Eric	90

5. (6 points) Consider the `test_func` function below.

```
CREATE OR REPLACE FUNCTION test_func()
RETURNS INT AS $func$
DECLARE
  curs CURSOR for (SELECT * FROM Scores ORDER BY score desc);
  r RECORD;
  sum_score INT;
  cnt1 INT;
  cnt2 INT;
BEGIN
  sum_score := -1;
  cnt1 := 0;
  cnt2 := 0;

  OPEN curs;
  LOOP
    FETCH curs INTO r;
    EXIT WHEN NOT FOUND;
    cnt1 := cnt1 + 1;

    IF (sum_score = -1) THEN
      sum_score := r.score;
    ELSE
      sum_score := sum_score + r.score;
      IF (r.score < sum_score / cnt1) THEN
        cnt2 := cnt2 + 1;
      END IF;
    END IF;
  END LOOP;

  CLOSE curs;
  RETURN cnt2;
END;
$func$ LANGUAGE plpgsql;
```

Continue on the next page

Suppose that we execute the following query.

```
SELECT * FROM test_func();
```

What will be the result of the query? In your answer, you may omit the column name of the query result.

Comments:

We have the following run. In the table below, the first set of variables are at the start of the loop and the second set is at the end of the loop. *pre* indicates the value before the loop and *post* indicates the value at the end of the loop.

Iter	r.score	sum_score	cnt1	cnt2	sum_score	cnt1	cnt2
pre	-	-1	0	0	-	-	-
1	90	-1	0	0	90	1	0
2	80	90	1	0	170	2	1
3	70	170	2	1	240	3	2
4	60	240	3	2	300	4	3
5	50	300	4	3	350	5	4
post	-	350	5	4	-	-	-

The answer is `cnt2` at the bottom, which is 4.

6. (8 points) Consider the `scores_check_func` function below.

```
CREATE OR REPLACE FUNCTION scores_check_func()
RETURNS TRIGGER AS $func$
DECLARE
  cnt1 INT;
  cnt2 INT;
BEGIN
  SELECT COUNT(*) INTO cnt1
  FROM   Scores;

  SELECT COUNT(*) INTO cnt2
  FROM   Scores
  WHERE  score >= 70;

  IF (cnt2 * 2 < cnt1) THEN
    RAISE EXCEPTION 'You are too mean!';
  END IF;

  RETURN OLD;
END;
$func$ LANGUAGE plpgsql;
```

Continue on the next page

Suppose that we create a trigger `scores_check_trigger` based on the function above, and then we execute the following transaction.

```
BEGIN TRANSACTION;
  DELETE FROM Scores WHERE sid = 's4';
  DELETE FROM Scores WHERE sid = 's5';
  INSERT INTO Scores VALUES ('s6', 'Fred', 95);
COMMIT;
```

After the execution of the transaction above, we find that the table `Scores` contains **4 (FOUR)** rows as shown on the right.

Provide **2 (TWO)** different **VALID** definitions of `scores_check_trigger` such that both definitions allow the above scenario. Note that some blank(s) in the answer sheet can be empty and each blank can be filled with multiple keywords. However, you are **NOT** allowed to use the `WHEN` keyword.

sid	name	score
s1	Alice	50
s2	Bob	60
s3	Cathy	70
s6	Fred	95

Comments:

We either use `BEFORE` trigger or we use `DEFERRABLE` trigger.

```
CREATE TRIGGER scores_check_trigger
BEFORE DELETE ON Scores
FOR EACH ROW
EXECUTE FUNCTION scores_check_func();
```

```
CREATE CONSTRAINT TRIGGER scores_check_trigger
AFTER DELETE ON Scores
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW
EXECUTE FUNCTION scores_check_func();
```

NOTE:

- Adding checks on `INSERT` or `UPDATE` does not make the trigger different as the `DELETE` trigger is still the same.
  - Recap that the keyword `DELETE` is already in the answer sheet.
- The keyword `CONSTRAINT` is required for `AFTER` trigger.
  - The keyword `CONSTRAINT` is not penalized for `BEFORE` trigger as we did not have an example showing this.
- The `AFTER` trigger cannot be `INITIALLY IMMEDIATE` as the transaction did not defer the trigger.

Continue on the next page

C. Functional Dependencies and Normal Form 44 Points

For the next **2 (TWO)** questions, consider the following relation $R(A, B, C, D, E, F)$ with the following set of functional dependencies. Note that we will use the *shorthand* notation for the functional dependencies. Instead of writing $\{A, B\} \rightarrow \{C, D\}$, we will write $AB \rightarrow CD$. You may use the same notation on your answer.

$$\Sigma = \{A \rightarrow D,\ A \rightarrow E,\ BC \rightarrow A,\ BE \rightarrow A,\ BF \rightarrow A,\\ CD \rightarrow A,\ D \rightarrow B,\ D \rightarrow E,\ E \rightarrow C,\ F \rightarrow B,\ F \rightarrow C\}$$

7. (4 points) Suppose we decompose R into $R_1(A, B, C, D)$ and $R_2(C, D, E, F)$. Is this a lossless join decomposition? Briefly justify your answer.

> **Comments:**
>
> **Yes, it is a lossless join decomposition.**
>
> Working
>
> - The common attributes of R_1 and R_2 are $\{C, D\}$.
> - $\{C, D\}^+ = \{A, B, C, D, E\}$.
> - Therefore, $\{C, D\}$ is the superkey of R_1.
>
> You may use Chase algorithm (`https://en.wikipedia.org/wiki/Chase_(algorithm)`).

8. (6 points) Suppose we decompose R into $R_1(A, B, C, D)$ and $R_2(C, D, E, F)$. Is this a dependency-preserving decomposition? If not, please identify **ALL** functional dependencies in Σ that are **NOT** preserved. If yes, please identify the projections of Σ on both R_1 and R_2 respectively.

> **Comments:**
>
> **Yes, it is a dependency preserving decomposition.**
>
> On R_1, we have the following closures.
>
> - $\{A\}^+ = \{A, B, C, D\}$
> - $\{B\}^+ = \{B\}$
> - $\{C\}^+ = \{C\}$
> - $\{D\}^+ = \{A, B, C, D\}$
> - $\{B, C\}^+ = \{A, B, C, D\}$
>
> So we have the following projection:
>
> $$\{\\ A \rightarrow BCD,\\ D \rightarrow ABC,\\ BC \rightarrow AD\\ \}$$
>
> many others including $AB \rightarrow CD$, *etc* can be derived from this.

On R_2, we have the following closures.

- $\{C\}^+ = \{C\}$
- $\{D\}^+ = \{C, D, E\}$
- $\{E\}^+ = \{C, E\}$
- $\{F\}^+ = \{C, D, E, F\}$
- $\{C, D\}^+ = \{C, D, E\}$
- $\{C, E\}^+ = \{C, E\}$
- $\{D, E\}^+ = \{C, D, E\}$
- $\{C, D, E\}^+ = \{C, D, E\}$

So we have the following projection:

$$\{ \\ D \to CE, \\ E \to C, \\ F \to CDE, \\ CD \to E, \\ DE \to C \\ \}$$

the rest can be derived from this.

Simply selecting from Σ all the functional dependencies that are relevant is *insufficient* because it will miss some functional dependencies such as $A \to C$. If you missed $A \to C$, your answer will not be equivalent to the solution. Additionally, the projection should not contain any attributes not in the relation (*e.g.*, $A \to E$).

Let $\Sigma^+_{R_1}$ and $\Sigma^+_{R_2}$ be the solution and Σ_{R_1} and Σ_{R_2} be your answer. We require $\Sigma^+_{R_1} \equiv \Sigma_{R_1}$ and $\Sigma^+_{R_2} \equiv \Sigma_{R_2}$. But note that you may still arrive at the same conclusion (*i.e.*, $(\Sigma_{R_1} \cup \Sigma_{R_2}) \equiv \Sigma$) while not satisfying the required equivalence above.

For the remainder of the papers, each question will have their own set of functional dependencies Σ.

9. (4 points) Consider the following relation $R(A, B, C, D, E, F)$ with the following set of functional dependencies. Note that we will use the *shorthand* notation for the functional dependencies. Instead of writing $\{A, B\} \to \{C, D\}$, we will write $AB \to CD$. You may use the same notation on your answer.

$$\Sigma = \{A \to B,\ BC \to A,\ D \to E,\ E \to D,\ CF \to B,\\ BF \to E,\ B \to C,\ EF \to A,\ DE \to C,\ EF \to BC\}$$

Identify **ALL** keys of R with respect to Σ. Briefly justify your answer.

Comments:

Since F does not appear on the right hand side of any FD, F must be in every key.

- $\{A, F\}^+ = \{A, B, C, D, E, F\}$
- $\{B, F\}^+ = \{A, B, C, D, E, F\}$
- $\{C, F\}^+ = \{A, B, C, D, E, F\}$
- $\{D, F\}^+ = \{A, B, C, D, E, F\}$
- $\{E, F\}^+ = \{A, B, C, D, E, F\}$

All 3 attributes containing F will include one of the keys above. So the keys are $\{A, F\}$, $\{B, F\}$, $\{C, F\}$, {D,F},{E,F}.

10. (10 points) Consider the following relation $R(A, B, C, D, E, F)$ with the following set of functional dependencies. Note that we will use the *shorthand* notation for the functional dependencies. Instead of writing $\{A, B\} \rightarrow \{C, D\}$, we will write $AB \rightarrow CD$. You may use the same notation on your answer.

$$\Sigma = \{A \rightarrow F,\ AD \rightarrow E,\ AE \rightarrow C,\ AF \rightarrow B,\ B \rightarrow A,$$
$$BC \rightarrow A,\ C \rightarrow DE,\ CD \rightarrow B,\ DE \rightarrow A,\ E \rightarrow F,\ F \rightarrow D,\ BF \rightarrow C\}$$

Is R in BCNF with respect to Σ? If yes, briefly justify your answer. If not, derive a BCNF decomposition of R using **ONLY** the decomposition algorithm introduced in the CS2102 lectures. Show your steps.

Comments:

Since $\{F\}^+ = \{D, F\}$, we have a violation as $\{F\}$ is not a superkey. Using $F \rightarrow DF$ as a violation, we can decompose R into:

- $R_1(D, F)$
- $R_2(A, B, C, E, F)$

Notice that R_1 only has two attributes, so it is in BCNF. For R_2, we can check all closures:

- $\{A\}^+ = \{A, B, C, D, E, F\}$ so all FD with A on left hand side will not violate.
- $\{B\}^+ = \{A, B, C, D, E, F\}$ so all FD with B on left hand side will not violate.
- $\{C\}^+ = \{A, B, C, D, E, F\}$ so all FD with C on left hand side will not violate.
- $\{E\}^+ = \{A, B, C, D, E, F\}$ so all FD with E on left hand side will not violate.
- $\{F\}^+ = \{F\}$ this is trivial and does not violate.

Note that any FD with two or more attributes as left hand side will not violate as the left hand side will be a superkey. So there is no violation of BCNF on R_2 and it is in BCNF.

NOTE: $F \rightarrow D$ is the only violation. So this is the only solution.

Additionally, if there is no mention of closure of all single attribute, then mentioning that R_2 is in BCNF because $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, and $\{E\}$ are keys is a leap in reasoning, and will be penalized.

Continue on the next page

11. (12 points) Consider the following relation $R(A, B, C, D, E, F)$ with the following set of functional dependencies. Note that we will use the *shorthand* notation for the functional dependencies. Instead of writing $\{A, B\} \rightarrow \{C, D\}$, we will write $AB \rightarrow CD$. You may use the same notation on your answer.

$$\Sigma = \{A \rightarrow BCD,\ B \rightarrow C,\ E \rightarrow F,\ BD \rightarrow EF,\ EF \rightarrow D,\ AE \rightarrow DF\}$$

Is R in 3NF with respect to Σ? If yes, briefly justify your answer. If not, derive a 3NF decomposition of R using **only** the 3NF decomposition (*i.e.*, synthesis) algorithm introduced in the CS2102 lectures. Show your steps for deriving minimal basis.

Comments:

A does not appear on the right hand side of any FD, so A must be in all key of R. Since $\{A\}^+ = \{A, B, C, D, E, F\}$, we know that $\{A\}$ is the only key. In that case, $B \rightarrow C$ violates 3NF so the table is not in 3NF.

Deriving Minimal Basis

Step 1: Singleton RHS
$\Sigma = \{A \rightarrow B, A \rightarrow C, A \rightarrow D, B \rightarrow C, E \rightarrow F, BD \rightarrow E, BD \rightarrow F, EF \rightarrow D, AE \rightarrow D,$
$AE \rightarrow F\}$

Step 2: Remove Redundant Attributes from LHS
$\Sigma = \{A \rightarrow B, A \rightarrow C, A \rightarrow D, B \rightarrow C, E \rightarrow F, BD \rightarrow E, BD \rightarrow F, E \rightarrow D, A \rightarrow D, E \rightarrow F\}$
Remove duplicates:
$\Sigma = \{A \rightarrow B, A \rightarrow C, A \rightarrow D, B \rightarrow C, E \rightarrow F, BD \rightarrow E, BD \rightarrow F, E \rightarrow D\}$

Step 3: Remove Redundant FDs
$\Sigma = \{A \rightarrow B, A \rightarrow D, B \rightarrow C, E \rightarrow F, BD \rightarrow E, E \rightarrow D\}$

Step 4: Combine FDs with the same LHS
$\Sigma = \{A \rightarrow BD, B \rightarrow C, E \rightarrow DF, BD \rightarrow E\}$

Step 5: Removed Subsumed Table(s)
$\{R_1(A, B, D), R_2(B, C), R_3(D, E, F), R_4(B, D, E)\}$

Note: This is the only minimal cover of the set of FD using our algorithm. However, there are two possibilities for Step 2.

- $\Sigma = \{A \rightarrow B, A \rightarrow C, A \rightarrow D, B \rightarrow C, E \rightarrow F, BD \rightarrow E, BD \rightarrow F, E \rightarrow D\}$
- $\Sigma = \{A \rightarrow B, A \rightarrow C, A \rightarrow D, B \rightarrow C, E \rightarrow F, BD \rightarrow E, BD \rightarrow F, E \rightarrow D, A \rightarrow F\}$
  - $A \rightarrow F$ comes from $AE \rightarrow F$ in which the attribute E is redundant and removed.

As a side note, you can check that $\Sigma = \{A \rightarrow B, \underline{A \rightarrow E}, B \rightarrow C, E \rightarrow F, BD \rightarrow E, E \rightarrow D\}$ is also a minimal basis, but cannot be derived by our algorithm.

Continue on the next page

12. (8 points) Consider the following relation $R(A, B, C, D)$. We do not know what the set of functional dependencies is yet, that is what we want to find out. We know that the set of functional dependencies Σ satisfies the following conditions simultaneously:

- R is in 3NF **BUT NOT** in BCNF with respect to Σ.
- If we derive a BCNF decomposition of R using **ONLY** the BCNF decomposition algorithm introduced in CS2102 lectures, we will obtain the following decomposition:

$$R_1(A, B), \quad R_2(B, C), \quad R_3(C, D)$$

Find Σ. Briefly justify your answer.

Comments:

Answer: $\{\ B \rightarrow A, C \rightarrow D, AD \rightarrow BC\ \}$

From here, we can work backwards for the brief justification of the answer.

- The keys are $\{A, B\}$, $\{B, C\}$, $\{A, D\}$, $\{B, D\}$.
  - Therefore $B \rightarrow A$ violates the BCNF property of R.
- The prime attributes are $\{A, B, C, D\}$.
  - Therefore, no violation of 3NF property as all attributes in R are prime attributes.
- Using $B \rightarrow A$ as a violation, we decompose R into
  - $R_1(A, B)$ *(in BCNF)*
  - $R_x(B, C, D)$ *(not in BCNF)*
    - Using $C \rightarrow D$ as a violation, we decompose $R_x(B, C, D)$ into
      - $R_2(B, C)$ *(in BCNF)*
      - $R_3(C, D)$ *(in BCNF)*

quod erat demonstrandum

– End of Paper –