CS220P Section 2 Assignment 2

Name: Chuqi Wang; studentID: 79167724; NetID: chuqiw4

Name: Zhihang Fang; studentID: 26827566; NetID: zhihangf

Date: October 28th, 2023

Part 1:

A.

Given R(A, B, C, D, E, G), F1 = {ABC → D, BC → EA, BCE → G },
F2 = {A→B, C→AD, AE→CG, BC → C} and
F3 = {AC → B, BC → D, BD → E, AE → G, ED → A, DA → C}

- For functional dependencies of F1, the closure of attribute set are shown below:

  $(ABC)^+$ = {ABCDEG} = R; $(BC)^+$= {BCEADG}=R; $(BCE)^+$ ={BCEGAD}=R

  Therefore, relation R with set F1 is in BCNF.

- For functional dependencies set F2, the closure is shown below:

  $(A)^+$= {AB} ≠ R, so A isn't a super key of R.

  Therefore, relation R with set F2 is not in BCNF.

- For functional dependencies set F3, the closures are show below:

  $(AC)^+$= {ACBDEG}=R; $(BC)^+$= {BCDEAG}=R; $(BD)^+$= {BDEACG}=R;

  $(AE)^+$= {AEG} ≠ R, so AE isn't a super key of R.

  Hence, relation R with set F3 isn't in BCNF.

B.

Let L represents the set that containing the attributes appearing on the left side in a functional dependency; let R represent the set that containing the attributes appearing on the right side in a functional dependency; Similarly, let LR represent the set containing both left and right sides attributes and let N represent the set of none attributes appearing in both left and right side in FD.

- For set of functional dependency F1= {ABC → D, BC → EA, BCE → G },
  L={B,C}; R={D,G}; LR={A,E}; N={Ø}
  Then $(BC)^+$= {BCEAGD} = R
  Therefore, the candidate keys of R for F1 is BC.

- For set of functional dependency F2={A→B, C→AD, AE→CG, BC → C},
  L={E}; R={D,G}; LR={A,B,C}; N={Ø}
  Then $(E)^+$= {E} ≠ R; $(EA)^+$= {EACGDB} = R; $(EB)^+$= {EB} ≠ R; $(EC)^+$= {ECADBG} = R
  Thus, the candidate keys of R for F2 is EA, EC.

- For set of functional dependency F3 = {AC → B, BC → D, BD → E, AE → G, ED → A, DA → C},
  L={Ø}; R={G}; LR={A,B,C,D,E}; N={Ø}
  Then $(A)^+$= {A} ≠ R; $(B)^+$= {B} ≠ R; $(C)^+$= {C} ≠ R; $(D)^+$= {D} ≠ R; $(AB)^+$= {AB} ≠ R; $(AC)^+$= {ACBDEG} = R; $(AD)^+$= {ADCBEG} = R;

$(AE)^+$= {AEG} ≠ R; $(BC)^+$= {BCDEAG} = R; $(BD)^+$= {BDEAGC} = R;

$(CD)^+$= {CD} ≠ R; $(CE)^+$= {CE} ≠ R; $(DE)^+$= {DEACBG} = R;

Therefore, the candidate keys of R for F3 is AC, AD, BC, BD, DE.

C.

Given R is partitioning into 3 sub relations R1{A,B,C} , R2{D,E,G} , R3{B,C,D}.

R1∩R2={Ø} R1∩R3={B,C} R2∩R3={D}

R1∩R2 = {Ø} which means there is no common attribute between R1 and R2, it doesn't provide information about whether the decomposition is lossless. We can say that R1 and R2 are lossy.

- For set of functional dependency F1= {ABC → D, BC → EA, BCE → G },

  R1∩R3={B,C}, $(BC)^+$={BCEAG}, since R1-R3={A} ,so BC→A which can show R1 and R3 are lossless.

  R13=R1 ∪ R3={A,B,C,D}, R13∩R2={D}, $(D)^+$={D} ≠ R13-R2 or R2-R13, so R13 and R2 are lossy.

  Then see R2∩R3={D}, $(D)^+$={D} ≠ R2-R3 or R3 -R2, so R2 and R3 are lossy.

  Therefore, we can conclude that for FD F1, this decomposition is not lossless.

- For set of functional dependency F2={A→B, C→AD, AE→CG, BC → C},

R1∩R3={B,C}, $(BC)^+$={BCAD}, since R1-R3={A} ,so BC→A which can show R1 and R3 are lossless.

R13=R1 ∪ R3={A,B,C,D}, R13∩R2={D}, $(D)^+$={D} ≠ R13-R2 or R2-R13, so R13 and R2 are lossy.

Then see R2∩R3={D}, $(D)^+$={D} ≠ R2-R3 or R3 -R2, so R2 and R3 are lossy.

Therefore, we can conclude that for FD F2, this decomposition is not lossless.

- For set of functional dependency F3 = {AC → B, BC → D, BD → E, AE → G, ED → A, DA → C},

  R1∩R3={B,C}, $(BC)^+$= {BCDEAG}, since R3-R1={D}, so BC → D which shows R1 and R3 are lossless.

  R13 = R1 ∪ R3 = {A,B,C,D}, R13∩R2={D}, since $(D)^+$={D} ≠ R13-R2 or R2-R13, so R2 and R13 are lossy.

  Now see R2∩R3={D}, $(D)^+$={D} ≠ R3-R2 or R2-R3, so R2 and R3 are lossy.

  Therefore, we can conclude that for FD F3, this decomposition is not lossless.

Part 2:

A.(Relations)

- person(<u>person_id</u>, name, dob, gender)
  - All non-key attributes are not NULL.
- employee(<u>person_id</u>, schedule, employee_type, salary_per_hour)
  - All non-key attributes are not NULL.
  - employee(person_id) ⊆ person(person_id)

- desk_employee(person_id, schedule, employee_type, salary_per_hour)
    - All non-key attributes are not NULL.
    - desk_employee(person_id) ⊆ employee(person_id)
- trainer(person_id, schedule, employee_type, salary_per_hour, credentials)
    - All non-key attributes are not NULL.
    - trainer(person_id) ⊆ employee(person_id)
- member(person_id, membership_id)
    - All non-key attributes are not NULL.
    - member(person_id) ⊆ person(person_id)
- family(person_id, membership_id, credit_card)
    - All non-key attributes are not NULL.
    - family(person_id) ⊆ member(person_id)
- university_affiliate(person_id, membership_id, department)
    - All non-key attributes are not NULL.
    - university_affiliate(person_id) ⊆ member(person_id)
- student(person_id, membership_id, department, student_type)
    - All non-key attributes are not NULL.
    - student(person_id) ⊆ university_affiliate(person_id)
- non_student(person_id, membership_id, member_type, department, credit_card)
    - All non-key attributes are not NULL.
    - non-student(person_id) ⊆ university_affiliate(person_id)
- related(f.person_id, u.person_id)
    - All non-key attributes are not NULL.
    - related(f.person_id) ⊆ family(person_id)
    - related(u.person_id) ⊆ university_affiliate(person_id)
    - family(person_id) ⊆ related(f.person_id)
- entry_log(person_id, timestamp)

    - All non-key attributes are not NULL.
    - entry_log(person_id) ⊆ person(person_id)
- employee_exit_log(person_id, timestamp)
    - All non-key attributes are not NULL.
    - employee_exit_log(person_id) ⊆ employee(person_id)
- space(space_id, description, max_capacity)
    - All non-key attributes are not NULL.
- events(event_id, description, start_time, end_time, capacity)
    - All non-key attributes are not NULL.
- equipment(equipment_id, equipment_type, is_available)
    - All non-key attributes are not NULL.
- hosted_in(event_id, space_id)
    - All non-key attributes are not NULL.
    - hosted_in(event_id) ⊆ events(event_id)
    - hosted_in(space_id) ⊆ space(space_id)
    - events(event_id) ⊆ hosted_in(event_id)
- contains(equipment_id, space_id)
    - All non-key attributes are not NULL.
    - contains(equipment_id) ⊆ equipment(equipment_id)
    - contains(space_id) ⊆ space(space_id)
    - equipment(equipment_id) ⊆ contains(event_id)
- location_sensor(sensor_id, coverage)
    - All non-key attributes are not NULL.
- equipment_sensor(sensor_id, coverage)
    - All non-key attributes are not NULL.
- location_reading(person_id, space_id, sensor_id, timestamp)
    - location reading(person_id)) ⊆ person(person_id)
    - location reading(space_id)) ⊆ space(space_id)
    - location reading(sensor_id)) ⊆ location sensor(sensor_id)
- usage_reading(person_id, equipment_id, sensor_id, timestamp)

- usage reading(person_id) ⊆ member(person_id)
- usage reading(equipment_id) ⊆ equipment(equipment_id)
- usage reading(sensor_id) ⊆ equipment sensor(sensor_id)

B.(SQL DDL Statements)

- ```
  CREATE TABLE person(
        person_id INT NOT NULL,
        name VARCHAR(40) NOT NULL,
        dob DATE NOT NULL,
        gender VARCHAR(40) NOT NULL,
        PRIMARY KEY(person_id)
  );
  ```

- ```
  CREATE TABLE employee(
        person_id INT NOT NULL,
        schedule VARCHAR(255) NOT NULL,
        employee_type VARCHAR(40) NOT NULL,
        salary_per_hour DECIMAL(10,2) NOT NULL,
        PRIMARY KEY(person_id),
        FOREIGN KEY(person_id) REFERENCES person(person_id)
  );
  ```

- ```
  CREATE TABLE desk_employee(
        person_id INT NOT NULL,
        schedule VARCHAR(255) NOT NULL,
        employee_type VARCHAR(40) NOT NULL,
        salary_per_hour DECIMAL(10,2) NOT NULL,
        PRIMARY KEY(person_id),
        FOREIGN KEY(person_id) REFERENCES employee(person_id)
  );
  ```

- ```
  CREATE TABLE trainer(
        person_id INT NOT NULL,
        schedule VARCHAR(255) NOT NULL,
        employee_type VARCHAR(40) NOT NULL,
        salary_per_hour DECIMAL(10,2) NOT NULL,
  ```

```
    credentials VARCHAR(255) NOT NULL,
    PRIMARY KEY(person_id),
    FOREIGN KEY(person_id) REFERENCES employee(person_id)
);
```

- ```
  CREATE TABLE member(
      person_id INT NOT NULL,
      membership_id INT NOT NULL,
      PRIMARY KEY(person_id),
      FOREIGN KEY(person_id) REFERENCES person(person_id)
  );
  ```

- ```
  CREATE TABLE family(
      person_id INT NOT NULL,
      membership_id INT NOT NULL,
      credit_card VARCHAR(255) NOT NULL,
      PRIMARY KEY(person_id),
      FOREIGN KEY(person_id) REFERENCES member(person_id)
  );
  ```

- ```
  CREATE TABLE university_affiliate(
      person_id INT NOT NULL,
      membership_id INT NOT NULL,
      department VARCHAR(40) NOT NULL,
      PRIMARY KEY(person_id),
      FOREIGN KEY(person_id) REFERENCES member(person_id)
  );
  ```

- ```
  CREATE TABLE student(
      person_id INT NOT NULL,
      membership_id INT NOT NULL,
      department VARCHAR(40) NOT NULL,
      student_type VARCHAR(20) NOT NULL,
      PRIMARY KEY(person_id),
      FOREIGN KEY(person_id) REFERENCES
      university_affiliate(person_id)
  );
  ```

- ```
  CREATE TABLE non_student(
        person_id INT NOT NULL,
        membership_id INT NOT NULL,
        member_type VARCHAR(40) NOT NULL,
        department VARCHAR(40) NOT NULL,
        credit_card VARCHAR(255) NOT NULL,
        PRIMARY KEY(person_id),
        FOREIGN KEY(person_id) REFERENCES
        university_affiliate(person_id)
  );
  ```

- ```
  CREATE TABLE related(
        fperson_id INT NOT NULL,
        uperson_id INT NOT NULL,
        PRIMARY KEY(fperson_id),
        FOREIGN KEY(fperson_id) REFERENCES family(person_id),
        FOREIGN KEY(uperson_id) REFERENCES
  university_affiliate(person_id)
  );
  ```

- ```
  CREATE TABLE entry_log(
        person_id INT NOT NULL,
        timestamp TIMESTAMP NOT NULL,
        PRIMARY KEY(person_id, timestamp),
        FOREIGN KEY(person_id) REFERENCES person(person_id)
  );
  ```

- ```
  CREATE TABLE employee_exit_log(
        person_id INT NOT NULL,
        timestamp TIMESTAMP NOT NULL,
        PRIMARY KEY(person_id, timestamp),
        FOREIGN KEY(person_id) REFERENCES employee(person_id)
  );
  ```

- ```
  CREATE TABLE space(
        space_id INT NOT NULL,
        description TEXT NOT NULL,
        max_capacity INT NOT NULL,
        PRIMARY KEY(space_id)
  ```
  ```

```
);
```

- ```
  CREATE TABLE events(
        event_id INT NOT NULL,
        description TEXT NOT NULL,
        start_time DATETIME NOT NULL,
        end_time DATETIME NOT NULL,
        PRIMARY KEY(event_id)
  );
  ```

- ```
  CREATE TABLE equipment(
        equipment_id INT NOT NULL,
        equipment_type VARCHAR(40) NOT NULL,
        is_available BOOLEAN NOT NULL,
        PRIMARY KEY(equipment_id)
  );
  ```

- ```
  CREATE TABLE hosted_in(
        event_id INT NOT NULL,
        space_id INT NOT NULL,
        PRIMARY KEY(event_id),
        FOREIGN KEY(event_id) REFERENCES events(event_id),
        FOREIGN KEY(space_id) REFERENCES space(space_id)
  );
  ```

- ```
  CREATE TABLE contains(
        equipment_id INT NOT NULL,
        space_id INT NOT NULL,
        PRIMARY KEY(equipment_id),
        FOREIGN KEY(equipment_id) REFERENCES
        equipment(equipment_id),
        FOREIGN KEY(space_id) REFERENCES space(space_id)
  );
  ```

- ```
  CREATE TABLE location_sensor(
        sensor_id INT NOT NULL,
        coverage VARCHAR(255) NOT NULL,
        PRIMARY KEY(sensor_id)
  ```
  ```

```
);
```

- ```
  CREATE TABLE equipment_sensor(
        sensor_id INT NOT NULL,
        coverage VARCHAR(255) NOT NULL,
        PRIMARY KEY(sensor_id)
  );
  ```

- ```
  CREATE TABLE location_reading(
        person_id INT NOT NULL,
        space_id INT NOT NULL,
        sensor_id INT NOT NULL,
        timestamp TIMESTAMP NOT NULL,
        PRIMARY KEY(person_id, space_id, sensor_id, timestamp),
        FOREIGN KEY(person_id) REFERENCES person(person_id),
        FOREIGN KEY(space_id) REFERENCES space(space_id),
        FOREIGN KEY(sensor_id) REFERENCES
  location_sensor(sensor_id)
  );
  ```

- ```
  CREATE TABLE usage_reading(
        person_id INT NOT NULL,
        equipment_id INT NOT NULL,
        sensor_id INT NOT NULL,
        timestamp TIMESTAMP NOT NULL,
        PRIMARY KEY(person_id, equipment_id, sensor_id,
  timestamp),
        FOREIGN KEY(person_id) REFERENCES person(person_id),
        FOREIGN KEY(equipment_id) REFERENCES
  equipment(equipment_id),
        FOREIGN KEY(sensor_id) REFERENCES
  equipment_sensor(sensor_id)
  );
  ```
  ```