Database Management Homework 2

Po-Yen Chu

1 Question 1

(a) According to the question, the answer is provided.

- (i) True. We can infer that a dean must be a teacher from the relationship `COLLEGE - DEAN - INSTRUCTOR`, which $(1,1)$ between `COLLEGE` and `DEAN` also infers that one college has exactly one dean.
- (ii) False. A instructors does not need to teach at least one class, according to the $(0,N)$ from `INSTRUCTOR` to `TEACHES` on figure 1.
- (iii) False. A student does not need to belong to a dept, he/she can belong to zero or one dept, according to the $(0,1)$ from `STUDENT` to `HAS` on figure 1.
- (iv) True. The $(0,N)$ and $(5,N)$ in `STUDENT - TAKES - SECTION` relationship can infer that the statement is true.
- (v) True. The `CONAME` is underlined, showing that it is a key attribute.

(b) The relational schema diagram is as Figure 1.

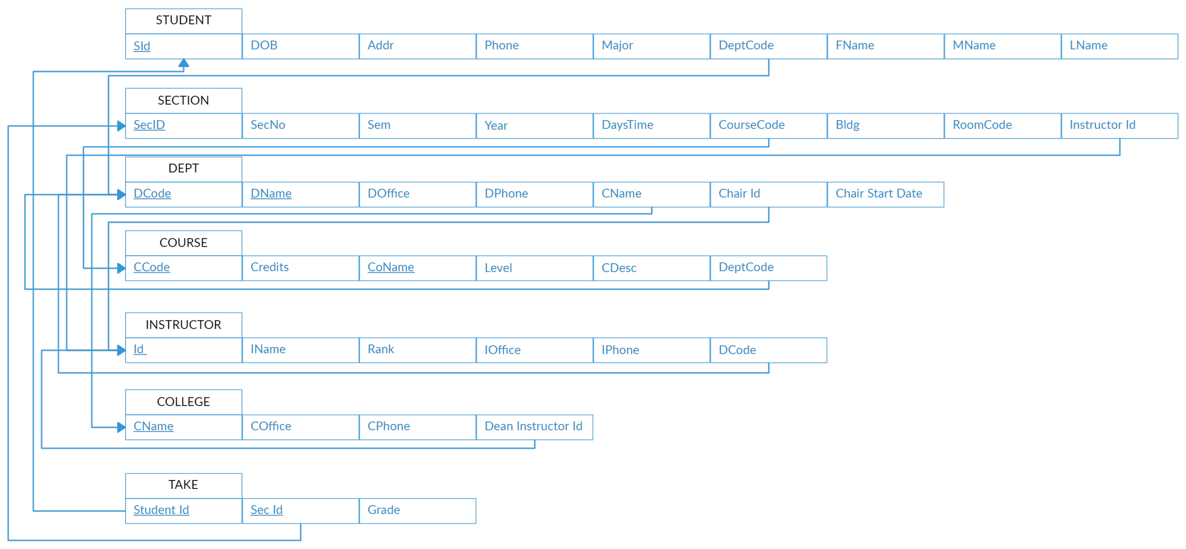

Figure 1: Relational Schema Diagram of a University Affairs System

(c) The database is composed of 7 tables: `Student`, `Section`, `Department`, `Course`, `Instructor`, `College`, `Take Sections`. The relationship between them includes:

(i) The one and only chair of a department must be an instructor, where the start date is also included in the table of department.

(ii) An instructor must belongs to one department, while a department can employ zero or multiple instructors.

(iii) A course can have no, one or many sections, but one section Id can only belongs to one course.

(iv) A course is also belongs to only one department, while a department can offer zero or multiple courses.

(v) A department must belongs to only one college, while a college can have zero or multiple departments

(vi) The relation between table `STUDENT` and `SECTION` is recorded in table `TAKE`, which includes columns of the primary keys of the two table and the corresponding grade.

2 Question 2

(a) The ER Diagram is as Figure 2.

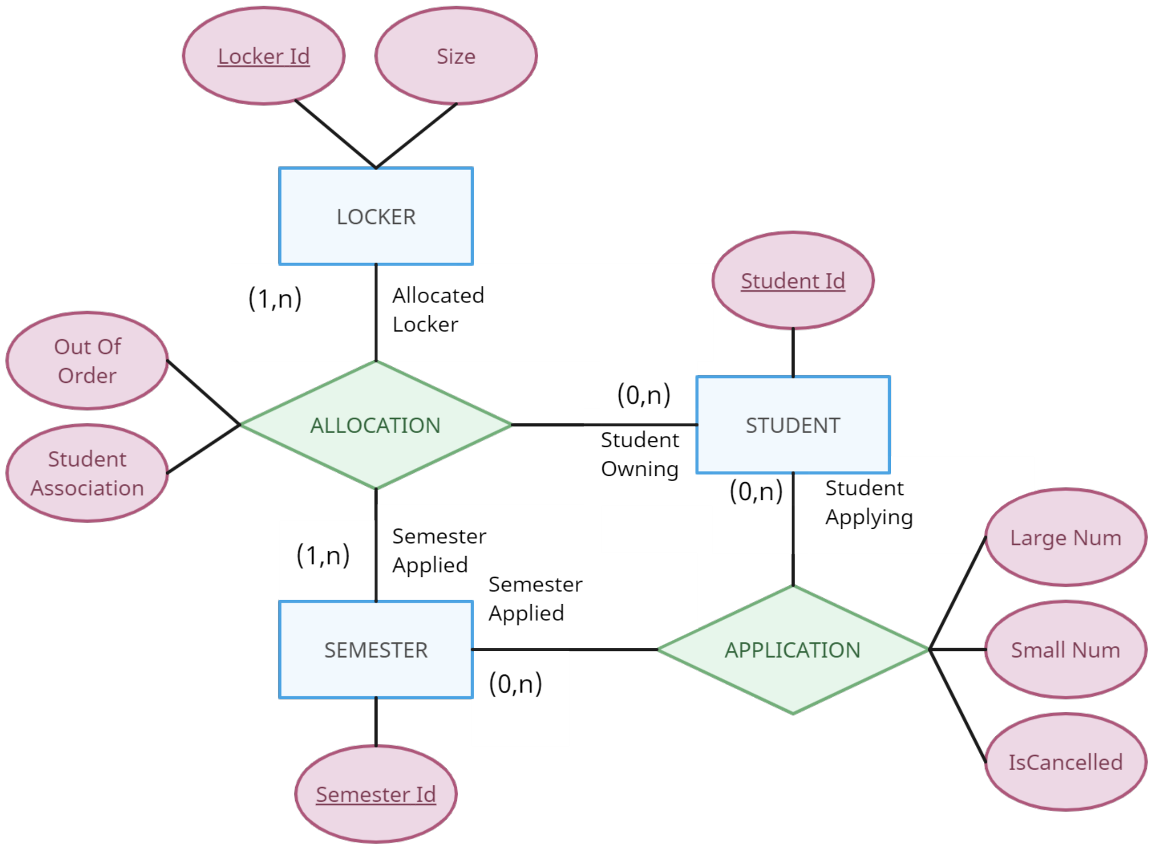

Figure 2: ER Diagram of IM Locker System

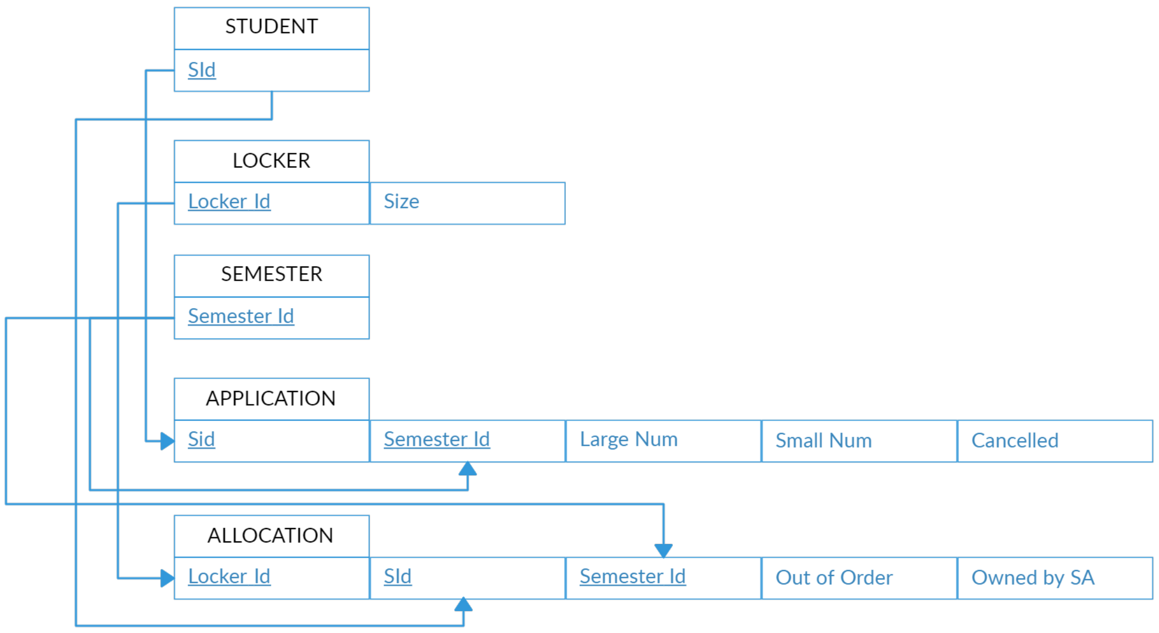

Figure 3: Relational Schema Diagram of IM Locker System

(b) According to Figure 2, the relational schema diagram is as Figure 3.

(c) The SQL commands are as follows:

```
CREATE TABLE public."ALLOCATION" (
    "Locker Id" text NOT NULL,
    "SId" text,
    "Semester Id" text NOT NULL,
    "Out of Order" boolean DEFAULT false NOT NULL,
    "Owned by SA" boolean DEFAULT false NOT NULL
);

ALTER TABLE public."ALLOCATION" OWNER TO postgres;

CREATE TABLE public."APPLICATION" (
    "SId" text NOT NULL,
    "Semester Id" text NOT NULL,
    "Large Num" integer DEFAULT 0 NOT NULL,
    "Small Num" integer DEFAULT 0 NOT NULL,
    "Cancelled" boolean DEFAULT false NOT NULL
);

ALTER TABLE public."APPLICATION" OWNER TO postgres;

CREATE TABLE public."LOCKER" (
    "Locker Id" text NOT NULL,
    "Size" text NOT NULL,
    CONSTRAINT "LOCKER_Size_check" CHECK (("Size" = ANY (
       ARRAY['large'::text, 'small'::text])))
);

ALTER TABLE public."LOCKER" OWNER TO postgres;

CREATE TABLE public."SEMESTER" (
    "Semester Id" text NOT NULL,
    CONSTRAINT "SEMESTER_Semester Id_check" CHECK (("
       Semester Id" ~ '^[0-9]+-(1|2)$'::text))
);

ALTER TABLE public."SEMESTER" OWNER TO postgres;

CREATE TABLE public."STUDENT" (
    "SId" text NOT NULL,
    CONSTRAINT "STUDENT_SId_check" CHECK (("SId" ~ '^[a-z
       ][0-9]{8}$'::text))
);

ALTER TABLE public."STUDENT" OWNER TO postgres;

ALTER TABLE ONLY public."LOCKER"
    ADD CONSTRAINT "LOCKER_pkey" PRIMARY KEY ("Locker Id")
       ;

ALTER TABLE ONLY public."SEMESTER"
    ADD CONSTRAINT "SEMESTER_pkey" PRIMARY KEY ("Semester
       Id");

ALTER TABLE ONLY public."STUDENT"
    ADD CONSTRAINT "STUDENT_pkey" PRIMARY KEY ("SId");

ALTER TABLE ONLY public."ALLOCATION"
    ADD CONSTRAINT "ALLOCATION_Locker Id_fkey" FOREIGN KEY
        ("Locker Id") REFERENCES public."LOCKER"("Locker
       Id");

ALTER TABLE ONLY public."ALLOCATION"
    ADD CONSTRAINT "ALLOCATION_SId_fkey" FOREIGN KEY ("SId
       ") REFERENCES public."STUDENT"("SId");

ALTER TABLE ONLY public."ALLOCATION"
    ADD CONSTRAINT "ALLOCATION_Semester Id_fkey" FOREIGN
       KEY ("Semester Id") REFERENCES public."SEMESTER"("
       Semester Id");

ALTER TABLE ONLY public."APPLICATION"
    ADD CONSTRAINT "APPLICATION_SId_fkey" FOREIGN KEY ("
       SId") REFERENCES public."STUDENT"("SId") ON UPDATE
       CASCADE;
```

3 Question 3

(Note: In Question 3, the ”specified semester” is replaced by 112-2 in the queries, one who utilized these queries should replace with the right specified semester id. Besides, all descriptions are commented in the commands.)

(a) The corresponding SQL commands are as follows:

```
SELECT
    "APPLICATION"."SId" AS Student_ID,  -- Student Id
    ("APPLICATION"."Large Num" + "APPLICATION"."Small Num"
       ) AS Apply_Num,  -- Total_Apply_Num
    COALESCE("ALLOCATION".Allocate_Num, 0) AS Allocate_Num
         -- Total_Allocate_Num (show 0 if no allocation)
FROM "APPLICATION"
LEFT JOIN (
    SELECT
        "ALLOCATION"."SId",
        COUNT("ALLOCATION"."Locker Id") AS Allocate_Num
    FROM "ALLOCATION"
    WHERE "ALLOCATION"."Semester Id" = '112-2'  -- 112-2
       can be replaced with the specified semester
    GROUP BY "ALLOCATION"."SId"
) AS "ALLOCATION" ON "APPLICATION"."SId" = "ALLOCATION"."
   SId"
WHERE "APPLICATION"."Semester Id" = '112-2'  -- 112-2 can
   be replaced with the specified semester
  AND "APPLICATION"."Cancelled" = FALSE
  AND ("ALLOCATION".Allocate_Num = 2 OR  -- Move the
     calculation into the WHERE clause
       "ALLOCATION".Allocate_Num IS NULL)
ORDER BY COALESCE("ALLOCATION".Allocate_Num, 0) DESC
```

(b) To assure the rules, here’s the first part of SQL commands.

```
WITH Last_Semester AS (
    SELECT MAX("Semester Id") AS last_semester
```

```sql
    FROM "SEMESTER"
    WHERE "Semester Id" < '112-2' -- 112-2 can be replaced
        with the specified semester
),

-- CTE to find students who applied but were not allocated
    in the previous semester
Previous_Unallocated AS (
    SELECT
        A."SId"
    FROM
        "APPLICATION" A
    LEFT JOIN
        "ALLOCATION" AL ON A."SId" = AL."SId"
        AND AL."Semester Id" = (SELECT last_semester FROM
           Last_Semester)
    WHERE
        A."Semester Id" = (SELECT last_semester FROM
           Last_Semester)
        AND A."Cancelled" = FALSE
        AND AL."Locker Id" IS NULL  -- Not allocated
),

Current_Unallocated AS (
    SELECT
        A."SId"
    FROM
        "APPLICATION" A
    LEFT JOIN
        "ALLOCATION" AL ON A."SId" = AL."SId"
        AND AL."Semester Id" = '112-2'  -- Replace with
           actual current semester ID
    WHERE
        A."Semester Id" = '112-2'  -- Replace with actual
           current semester ID
        AND A."Cancelled" = FALSE
        AND AL."Locker Id" IS NULL  -- Not allocated
),

-- CTE for students who were allocated at least one locker
    in the previous semester
Previous_Allocated AS (
    SELECT
        AL."SId"
    FROM
        "ALLOCATION" AL
    WHERE
        AL."Semester Id" = (SELECT last_semester FROM
           Last_Semester)
    GROUP BY
        AL."SId"
    HAVING
        COUNT(AL."Locker Id") >= 1  -- At least one locker
            allocated
),

Current_Allocated AS (
    SELECT
        AL."SId"
    FROM
        "ALLOCATION" AL
    WHERE
        AL."Semester Id" = '112-2'  -- Replace with actual
            current semester ID
    GROUP BY
        AL."SId"
)

-- Group 1: Students who applied but were not allocated in
    both semesters
SELECT
    'Students who applied but were not allocated in both
       semesters' AS Group_Description,
    PUA."SId"  -- Students from previous semester not
       allocated
FROM
    Previous_Unallocated PUA
JOIN
    Current_Unallocated CUA ON PUA."SId" = CUA."SId"  --
       Ensure both conditions are fulfilled

UNION ALL

-- Group 2: Students who were allocated at least one
   locker in previous semester and allocated in current
   semester
SELECT
    'Students who were allocated at least one locker in
       previous semester and allocated in current semester
       ' AS Group_Description,
    PA."SId"  -- Students from previous semester allocated
FROM
```

```
    Previous_Allocated PA
JOIN
    Current_Allocated CA ON PA."SId" = CA."SId";  --
       Ensure both conditions are fulfilled
```

The second part is as follows:

```
WITH Last_Semester AS (
    SELECT MAX("Semester Id") AS last_semester
    FROM "SEMESTER"
    WHERE "Semester Id" < '112-2'  -- Replace with the
       specified semester
),

-- CTE for students who were allocated exactly one locker
   in the previous semester
Previous_Allocated_One AS (
    SELECT
        AL."SId"
    FROM
        "ALLOCATION" AL
    WHERE
        AL."Semester Id" = (SELECT last_semester FROM
           Last_Semester)
    GROUP BY
        AL."SId"
    HAVING
        COUNT(AL."Locker Id") = 1  -- Exactly one locker
           allocated
),

-- CTE for students who were allocated exactly two lockers
    in the previous semester
Previous_Allocated_Two AS (
    SELECT
        AL."SId"
    FROM
        "ALLOCATION" AL
    WHERE
        AL."Semester Id" = (SELECT last_semester FROM
           Last_Semester)
    GROUP BY
        AL."SId"
    HAVING
        COUNT(AL."Locker Id") = 2  -- Exactly two lockers
           allocated
),

-- CTE for students who applied but were not allocated in
   the current semester
Current_Unallocated AS (
    SELECT
        A."SId"
    FROM
        "APPLICATION" A
    LEFT JOIN
        "ALLOCATION" AL ON A."SId" = AL."SId"
        AND AL."Semester Id" = '112-2'  -- Replace with
           actual current semester ID
    WHERE
        A."Semester Id" = '112-2'  -- Replace with actual
           current semester ID
        AND A."Cancelled" = FALSE
        AND AL."Locker Id" IS NULL  -- Not allocated
),

-- CTE for students who have been allocated in the current
    semester
Current_Allocated AS (
    SELECT
        AL."SId"
    FROM
        "ALLOCATION" AL
    WHERE
        AL."Semester Id" = '112-2'  -- Replace with actual
            current semester ID
    GROUP BY
        AL."SId"
)

-- Group 1: Students who were allocated exactly one locker
    in previous semester and unallocated in current
   semester
SELECT
    'Students who were allocated exactly one locker in
       previous semester and unallocated in current
       semester' AS Group_Description,
    PAO."SId"  -- Students from previous semester
       allocated exactly one
FROM
    Previous_Allocated_One PAO
JOIN
```

```
    Current_Unallocated CUA ON PAO."SId" = CUA."SId"  --
       Ensure both conditions are fulfilled

UNION ALL

-- Group 2: Students who were allocated exactly two
   lockers in previous semester and allocated in current
   semester
SELECT
    'Students who were allocated exactly two lockers in
       previous semester and allocated in current semester
       ' AS Group_Description,
    PAT."SId"  -- Students from previous semester
       allocated exactly two
FROM
    Previous_Allocated_Two PAT
JOIN
    Current_Allocated CA ON PAT."SId" = CA."SId";  --
       Ensure both conditions are fulfilled
```

(c) To update, the SQL commands are as follows:

```
UPDATE "APPLICATION"
SET
    "Large Num" = 2,  -- Replace with the new large count
    "Small Num" = 0   -- Replace with the new small count
WHERE
    "SId" = 'b10704031'  -- Replace with the student's ID
    AND "Semester Id" = (SELECT MAX("Semester Id") FROM "
       SEMESTER")  -- Get the latest semester
    AND "Cancelled" = FALSE;  -- Ensure the application
       has not been canceled
```

To delete, the SQL commands are as follows:

```
UPDATE "APPLICATION"
SET
    "Cancelled" = TRUE  -- Mark the application as
       canceled
WHERE
    "SId" = 'b10704031'  -- Replace with the student's ID
    AND "Semester Id" = (SELECT MAX("Semester Id") FROM "
       SEMESTER")  -- Get the latest semester
    AND "Cancelled" = FALSE;  -- Ensure the application
       has not been canceled
```