SQL Programming and Normalization

Colton Blackwell
CMPT 354
301451278

DBMS environment used: mySQL Workbench

Connection Details

Name:	local host
Host:	127.0.0.1
Port:	3306
Login User:	root
Current User:	root@localhost
SSL cipher:	TLS_AES_256_GCM_SHA384

Server

Product:	MySQL Community Server - GPL
Version:	8.0.35

Connector

Version:	C++ 8.1.0

Part 1: SQL Programming

Task 1: Create Table statements (employee, types, sales)

employee

Table: employee

Columns:

eid	int PK
name	varchar(20)
salary	int
dept	varchar(20)

Table:

employee

Create Table:

```
CREATE TABLE `employee` (
  `eid` int NOT NULL,
  `name` varchar(20) DEFAULT NULL,
  `salary` int DEFAULT NULL,
  `dept` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`eid`),
  CONSTRAINT `employee_chk_1` CHECK (((`salary` >= 5000) and (`salary` <= 20000)))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

*Used VARCHAR(20) instead of CHAR(20)

types

Table: types

Columns:

item varchar(20) PK
color varchar(20) PK

Table:	types
Create Table:	CREATE TABLE \`types\` (\`item\` varchar(20) NOT NULL, \`color\` varchar(20) NOT NULL, PRIMARY KEY (\`item\`,\`color\`)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

---

sales

Table: sales

Columns:

dept varchar(20) PK
item varchar(20) PK

Table:	sales
Create Table:	CREATE TABLE \`sales\` (\`dept\` varchar(20) NOT NULL, \`item\` varchar(20) NOT NULL, PRIMARY KEY (\`dept\`,\`item\`)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

Task 2: Insert records into the tables

employee

```
17:01:06  INSERT INTO employee (eid, name, salary, dept) VALUES (666,"Hoffman", 16000, "Cosmetics")
17:01:06  INSERT INTO employee (eid, name, salary, dept) VALUES (555,"Nelson", 6000, "Toy")
17:01:06  INSERT INTO employee (eid, name, salary, dept) VALUES (444,"Lewis", 12000, "Stationary")
17:01:06  INSERT INTO employee (eid, name, salary, dept) VALUES (333,"Morgan", 10000, "Cosmetics")
17:01:06  INSERT INTO employee (eid, name, salary, dept) VALUES (222,"Anderson", 8000, "Toy")
17:01:06  INSERT INTO employee (eid, name, salary, dept) VALUES (111,"Jane", 8000, "Household")
```

eid	name	salary	dept
111	Jane	8000	Household
222	Anderson	8000	Toy
333	Morgan	10000	Cosmetics
444	Lewis	12000	Stationary
555	Nelson	6000	Toy
666	Hoffman	16000	Cosmetics

*** Row with name = ‘Peter’ was added in Question 7**

types

```
17:08:19  INSERT INTO types VALUES ("ink","blue")
17:08:19  INSERT INTO types VALUES ("ink","red")
17:08:19  INSERT INTO types VALUES ("puzzle","black")
17:08:19  INSERT INTO types VALUES ("pen","black")
17:08:19  INSERT INTO types VALUES ("lipstick","red")
17:08:19  INSERT INTO types VALUES ("pen","red")
```

item	color
ink	blue
ink	red
lipstick	red
pen	black
pen	red
puzzle	black
NULL	NULL

Sales

17:05:09	INSERT INTO sales VALUES ("Toy", "lipstick")
17:05:09	INSERT INTO sales VALUES ("Sports", "skates")
17:05:09	INSERT INTO sales VALUES ("Household", "disk")
17:05:09	INSERT INTO sales VALUES ("Stationary", "ink")
17:05:09	INSERT INTO sales VALUES ("Toy", "puzzle")
17:05:09	INSERT INTO sales VALUES ("Cosmetics", "lipstick")
17:05:09	INSERT INTO sales VALUES ("Stationary", "pen")

dept	item
Cosmetics	lipstick
Household	disk
Sports	skates
Stationary	ink
Stationary	pen
Toy	lipstick
Toy	puzzle
NULL	NULL

Task 3: Query Data for Specific instances

1. Compute the maximum salary for each department that sells at least two distinct items.

Limit to 1000 rows

```sql
SELECT dept, MAX(salary) AS max_salary
FROM employee
WHERE dept IN (
    SELECT dept
    FROM sales
    GROUP BY dept
    HAVING COUNT(DISTINCT item) >= 2
)
GROUP BY dept;
```

Result Grid Filter Rows: Export: Wrap Cell Content:

dept	max_salary
Toy	8000
Stationary	12000

---

2. Compute the names of the employees who work in a department that sells some item in black colour.

```sql
SELECT DISTINCT e.name
FROM employee e
WHERE e.dept IN (
    SELECT DISTINCT s.dept
    FROM sales s
    WHERE s.item IN (
        SELECT DISTINCT t.item
        FROM types t
        WHERE t.color = 'black'
    )
);
```

Result Grid Filter Rows: Export: Wrap Cell Content:

name
Anderson
Lewis
Nelson

---

3. For each department that has a larger average salary than that of “Stationary” department, find its average salary.

```sql
SELECT e.dept, AVG(e.salary) AS avg_salary
FROM employee e
WHERE e.dept <> "Stationary"
GROUP BY e.dept
HAVING AVG(e.salary) > (SELECT AVG(salary) FROM employee WHERE dept = "Stationary");
```

dept	avg_salary
Cosmetics	13000.0000

---

4. Find the number of the departments that have a smaller average salary than that of “Stationary” department.

```sql
SELECT e.dept, COUNT(e.dept) AS num_departments
FROM employee e
GROUP BY e.dept
HAVING AVG(e.salary) < (SELECT AVG(salary) FROM employee WHERE dept = 'Stationary');
```

dept	num_departments
Household	1
Toy	2

---

5. Which department pays every of its employees at least 7000?

```sql
SELECT dept
FROM employee
GROUP BY dept
HAVING MIN(salary) >= 7000;
```

Result Grid | Filter R

dept
Household
Cosmetics
Stationary

---

6. Which department sells all items sold by ‘Cosmetics’ department?

```
SELECT DISTINCT s1.dept
FROM sales s1
WHERE NOT EXISTS (
    SELECT item
    FROM sales s2
    WHERE s2.dept = 'Cosmetics'
    AND NOT EXISTS (
        SELECT *
        FROM sales s3
        WHERE s3.dept = s1.dept AND s3.item = s2.item
    )
);
```

dept
Cosmetics
Toy

Part 2: FD and Normalization

Question 1: Anomalies in Relation R with FD’s F

Relation below is used for example purposes:

S	A	C	D	T
Johnny	324 Redwood St	Math	Math	Prof. Hughes
Johnny	324 Redwood St	History	History	Prof. Petterson
Max	929 Cedar St	Science	Science	Prof. Kuzmenko
Ted	223 Oak St	Math	Math	Prof. Hughes

Data Redundancy:
The address is repeated for each course a student takes, leading to redundancy

- Ex. Student Johnny has the same address for Math and History courses.

Update anomaly:
If a student updates their address, we need to update multiple rows to maintain consistency.

- Ex. If Student Johnny moves from 324 Redwood St to a different address, we need to update that for both occurrences of his Math and History courses.

Insertion Anomaly:
To insert a new course, we have to provide information about a teacher and department due to functional dependencies. If a department does not have any teachers at the moment, we cant insert info about a new course for that department until a teacher is assigned.

Deletion Anomaly:
If a student drops all their courses, we lose info about their address. Additionally, if a teacher stops teaching all courses, we lose information about their department. Deleting a department also deletes information about all the teachers in that department.

Question 2:

Q2 (1): Find all keys of R with respect to F.

If a key must uniquely determine all oher attributes in the relation R, and given…

R = (S, A, C, D, T)

With functional dependencies…

C -> T
S -> A
T -> D

{S, C} does not occur in RHS of any FD which means that they must be a part of every key

Key Computation:
SC^+ = {SACDT}

∴ SC is the only key!

Q2 (2): Test if R in BCNF with respect to F, why?

Example of redundancy:

Student	Address	Course	Department	Teacher
Johnny	324 Redwood St	MACM 101	Math	Prof. Hughes
Johnny	324 Redwood St	MACM 101	Math	Prof. Hughes
Max	929 Cedar St	MACM 101	Math	Prof. Hughes
Max	929 Cedar St	MACM 101	Math	Prof. Hughes

Student -> Address   Course -> Teacher   Teacher -> Department

Based on results form Question 2 (1), we know that the only key is {Student, Course}.
In each FD, the left side is not a superkey.

Ex. In Student -> Address, Student is not a superkey therefore there are multiple instances of redundant data for Johnny -> 324 Redwood St

---

Q2 (3): Produce a BCNF decomposition through a series of binary decomposition.

We start with reviewing the data that we are given

R(Student, Address, Course, Department, Teacher)

Course -> Teacher
Student -> Address
Teacher -> Department

Key = {Student, Course}

We then pick any BCNF violation that we observe in F. In this case, they are all violations on BCNF. We choose Course because it is not the candidate key.

{Course}+ = {Course, Teacher, Department}

Decompose R into…

1. **R1 = X + (all attributes determined by X)**
2. **R2 = (R – X +)X (all attributes not determined by X, plus X)**

Relation1(Course, Teacher, Department)
Relation2(Course, Address, Student)

For relation1 FD's are implied that course->Teacher and Teacher-> department. Course is the superkey in relation1 so relation1 is in BCNF.

For relation2 the only FD is Student -> Address so we repeat decomposition on R

Student+ = {Student, Address}

(new)Relation2(Student, Address)
Relation3(Student, Course)

Relation3 holds for BCNF as Student is superkey. The final decomposition is composed of sub-relations that all their functional dependencies hold BCNF.

New and revised Relations that hold for BCNF

Relation1		
Course	**Teacher**	**Department**
MACM 101	Prof. Hughes	Math
CMPT 125	Prof. Petterson	Computer Science
CMPT 321	Prof. Tanev	Computer Science

Relation2	
Student	**Address**
Elias	123 Cedar Road
Quinn	221 Pine ave.
Alex	252 Thorne blvd

Relation3	
Student	**Course**
Elias	MACM 101
Quinn	CMPT 321
Alex	MACM 101

Q2 (4): Why are the new decomposed tables in a better representation?

The new decomposed tables are in a better representation as they reduce anomalies as shown in Part 2 Question 1. These relations are more specialized in fulfilling their designated roles, effectively minimizing data redundancy. The new relations address deletion anomlies, ensuring that removing a student's record does not accidently delet other attributes, such as courses from the table.
Relation1 provides insights into teachers, their associated departments, and the course they instruct.
Relation2 details information about students and their respective addresses.
Relation3 specifically outlines the students and the courses which they are currently enrolled in.

Q2 (5): Is the final decomposition in 3 dependency-preserving ?

Yes the final decomposition is dependency preserving as all the FD's remained in their respective separate tables.

Relation2: Student -> Address
Relation1: Course -> Teacher
Teacher -> Department

Q2 (6): Is the original schema R in 3NF with respect to F?

No the original schema is not in 3NF because there is a transitive dependency. Additionally, SC is the only key which means C, S, T are not superkeys and T, A, D are not prime

C -> T and T -> D

The non-prime D depends on non-prime T

Question 2 (7): Produce a 3NF decomposition that is lossless and dependency-preserving.

Consider R = {SACDT} and F = {C -> T, S -> A, T -> D}

All FD’s are already in minimal cover and are not redundant.

For each FD X -> A in the minimal cover, add a relation Ri = XA.

R1 = CT
R2 = SA
R3 = TD

No key is contained in any Ri, so the addition of the key in decomposition is mandatory.

R4 = SC

3NF Decomposition:
{CT, SA, TD, SC}

Question 2 (8): Is the decomposition produced in 7 in BCNF?

The outcome is in BCNF as the left-hand side of each relation has a superkey within their respective relation.