COMPSCI 351 The University of Auckland + Southwest University

Fundamentals of Database Systems - Assignment 1

2024

Note: Collaboration on assignments is encouraged, but you must write up your work individually and in your own words.

1. **(The Relational Model of Data)**

   Consider the relation schema `CUSTOMER` which stores customer information such as a *customer_id*, the *name* of each customer as well as their *address* and *email.*

   (a) Which **keys** does the relation over *CUSTOMER* below satisfy? [2 marks]

   CUSTOMER

customer_id	name	address	email
001	Alan Turing	Byte Avenue 101	a.turing@np-computations.com
002	Peter Chen	Entity Street 5	p.chen@erm.com
003	Peter Chen	Relationship Road 1	p.chen@conceptual-modeling.com
004	Edgar Codd	Byte Avenue 101	e.codd@np-computations.com
005	Edgar Codd	Normal Road 3	e.codd@bcnf-computations.com

   (b) Which tuples from the relation can be removed so that the same keys as noted in a) are still satisfied and the relation has as few tuples as possible? (Note that by removing no additional keys should be satisfied) [2 marks]

   (c) Give an example of a relation over the schema `CUSTOMER` that:

   - satisfies the keys {*customer_id*} and {*name, email*}

   - does not satisfy any super key aside from those super keys that contain {*customer_id*} or {*name, email*} as subset [2 marks]

   (d) In addition to the relation schema `CUSTOMER` consider the relation schema `ORDERS` with attributes *order_id, customer_id, address, order_date* and the relations as shown underneath.

   CUSTOMER

customer_id	name	address	email
001	Peter Chen	Entity Street 5	p.chen@erm.com
002	Peter Chen	Relationship Road 1	p.chen@conceptual-modeling.com
003	Edgar Codd	Normal Road 3	e.codd@bcnf-computations.com
004	Edgar Codd	Normal Road 3	e.codd@three-nf-computations.com

   ORDERS

order_id	customer_id	address	order_date
1	003	Normal Road 3	2024-01-01

   Give an example of a tuple that added to the relation over the schema `ORDERS` will:

   - satisfy the key {order_id}

   - satisfy the foreign key [customer_id] ⊆ `CUSTOMER`[customer_id] over `ORDERS`

   - violate the foreign key [customer_id, address] ⊆ `CUSTOMER`[customer_id, address] over `ORDERS`

   - satisfy the inclusion dependency `ORDERS`[address] ⊆ `CUSTOMER`[address], i.e. for all values of *address* in `ORDERS` there exists a matching values of *address* in `CUSTOMER` [2 marks]

(Note: An inclusion dependency is a generalisation of foreign key where the referenced attribute(s) might not constitute a key in the parent table)

SOLUTION:

(a) The relation satisfies the keys {*customer_id*}, {*email*} and {*name*, *address*}. Every other set would be a super key that is containing one of these keys as subsets.

(b) Through removal of tuples none of the keys in a) can end up being violated. We cannot remove the tuples relating to *customer_id* 001 or 004 as otherwise *address* would become a key. Similarly we need two tuples carrying the same *name* as otherwise *name* will be a key. We could remove tuples relating to *customer_id* 002 and 003 as we then have two tuples matching on *name* and two tuples mathching on *address.* No further tuples can be removed.

(c) We have to make sure that {*customer_id*} and {*name, email*} are keys. However, neither {*name*}, nor {*email*} can be key as otherwise {*name, email*} becomes super key, but not key. This means that the values for {*customer_id*} need to be unique, the values for the combination of {*name, email*} need to be unique and we need two tuples matching on {*name*} and two tuples matching on {*email*}.

In addition, all super keys aside from those that contain {*customer_id*} or {*name, email*} as subset should be violated. These are {*address, email*} and {*address, name*}. So we need two tuples matching on {*address, email*} and two matching on {*address, name*}.

The following relation is such an example.

CUSTOMER

customer_id	name	address	email
001	Peter Chen	Database Avenue 1	p.chen@erm.com
002	Peter Chen	Database Avenue 1	modeling@db.com
003	Edgar Codd	Database Avenue 1	modeling@db.com

(d) We need a tuple with a 'new' order_id, a customer_id that exists in the parent table, an address that exists in the parent table, but the combination of customer_id and address does not exist in the parent table. The following is an example:

ORDERS

order_id	customer_id	address	order_date
1	003	Normal Road 3	2024-01-01
2	003	Entity Street 5	2024-01-01

2. **(SQL: DDL and DML)**

(a) Consider the relation schema `CUSTOMER`={*customer_id, name, address, email*} and the following incomplete SQL table definition:

```
CREATE TABLE CUSTOMER (
customer_id INTEGER NOT NULL,
......);
```

How would you have to complete the table definition so that:
- *customer_id* is key of the table
- the values for *name* have to exist
- the combination of *name* and *email* has to be unique
- a value for *address* or *email* is not necessarily required

Make appropriate assumptions about the domain for each attribute. [2 marks]

(b) Consider the following SQL schemata

```
CREATE TABLE CUSTOMER (
customer_id INTEGER NOT NULL,
name VARCHAR(25) NOT NULL,
address VARCHAR(40) NOT NULL,
email VARCHAR(40),

PRIMARY KEY (customer_id),
UNIQUE (email));


CREATE TABLE ORDERS (
order_id INTEGER NOT NULL,
customer_id INTEGER NOT NULL,
address VARCHAR(40) NOT NULL,
order_date DATE NOT NULL,

PRIMARY KEY (order_id),
FOREIGN KEY(customer_id) REFERENCES CUSTOMER(customer_id));
```

together with the instances shown below:

CUSTOMER

customer_id	name	address	email
001	Alan Turing	Byte Avenue 101	a.turing@np-computations.com
002	Peter Chen	Entity Street 5	p.chen@erm.com
003	Edgar Codd	Normal Road 3	e.codd@bcnf-computations.com

ORDERS

order_id	customer_id	address	order_date
1	003	Normal Road 3	2024-01-01
2	002	Entity Street 5	2024-01-01
3	002	Entity Street 5	2024-01-02

Which of the following operations can be performed? If an operation cannot be performed please state why not.

- DELETE FROM CUSTOMER WHERE customer_id = 002;
- UPDATE ORDERS SET customer_id = 000 WHERE address = ’Normal Road 3’;

- INSERT INTO CUSTOMER
  VALUES (004, 'Peter Chen', 'Relationship Road 1', null);
- DELETE FROM ORDERS WHERE customer_id = 003;

[4 marks]

SOLUTION:

(a) CREATE TABLE CUSTOMER (
customer_id INTEGER NOT NULL,
name VARCHAR(25) NOT NULL,
address VARCHAR(40),
email VARCHAR(40),

PRIMARY KEY (customer_id),
UNIQUE (name, email));

(b)
- DELETE FROM CUSTOMER WHERE customer_id = 002: Cannot be performed as it violates foreign key.
- UPDATE ORDERS SET customer_id = 000 WHERE address = 'Normal Road 3': Cannot be performed as it violates foreign key.
- INSERT INTO CUSTOMER
  VALUES (004, 'Peter Chen', 'Relationship Road 1', null): Can be performed (does not violate UNIQUE(email) on CUSTOMER)
- DELETE FROM ORDERS WHERE customer_id = 003: Can be performed (does not violate foreign key)

3. **(SQL as a query language)**

Consider the relational database schema {`CUSTOMER`, `ORDER`, `DELIVERY`} with the relation schemata:

- `CUSTOMER`={*customer_id, name, address, email*} with key {customer_id}

- `ORDERS`={*order_id, customer_id, name, address, order_date*} with key {order_id} and with foreign keys

  [customer_id] $\subseteq$ `CUSTOMER`[customer_id]

  [address, order_date] $\subseteq$ `DELIVERY`[address, order_date]

- `DELIVERY`={*address, order_date, method, order_status*} with key {address, order_date}

(a) Write the following (English language) query **in SQL**:

'What are the dates where no order with method 'express' had been placed?'

Ensure that the query only returns distinct dates, i.e. does not return duplicates. [2 marks]

(b) Write the following (English language) query **in SQL**:

'List the names of customers that have more than one address (different accounts with different addresses) together with the amount of addresses.'

[2 marks]

(c) Translate the following SQL query into and **English language description**:

```
SELECT C.name, COUNT(C.name) AS amount
FROM CUSTOMER C, ORDERS O, DELIVERY D
WHERE C.customer_id = O.customer_id
AND O.address = D.address
AND O.order_date = D.order_date
AND D.method = 'standard'
GROUP BY C.name ORDER BY amount DESC
```

[1 marks]

(d) Consider the English language query

'Return all pairs of customers that share the same address.'

and the proposed SQL query

```
SELECT C1.name, C2.name
FROM CUSTOMER C1, CUSTOMER C2
WHERE C1.address = C2.address;
```

What is not great about this SQL query and how could we fix this? [1 marks]

SOLUTION:

(a) There exist many equivalent queries. One possible answer is

```
SELECT DISTINCT O.order_date
FROM ORDERS O
WHERE O.order_date NOT IN (
SELECT D.order_date
FROM DELIVERY D
WHERE D.method = 'express')
```

(b)
```
SELECT C.name, COUNT(DISTINCT C.address)
FROM CUSTOMER C
GROUP BY C.name
HAVING COUNT(*) > 1
```

(c) Who are customers (what are the customer names) together with the amount of orders they placed that are of delivery method 'standard' ordered by the amount of orders in descending order?

(d) The query returns tuples where C1 and C2 refer to the same customer, i.e. tuples of the form ('John', 'John') and returns tuples where C1 and C2 refer to different customers twice, each time in a different order, i.e. ('John', 'Joe') and ('Joe', 'John'). This can be fixed for example including the condition C1.name < C2.name.

Possible Marks: 20