University of
St Andrews

Database Management with SQLite

School of Computer Science

IS5102 - Coursework 2

By
220033001

Task - 1:

The relational schema for the given E-R model for a Bookstore scenario is as follows:

Customer	(customer_id, first_name, last_name, email)
Customer_Phone	(phonenumber, customer_id, phone_type)
Customer_Address	(customer_id, street, city, post_code, country)
Book	(book_id, title, author_name, publisher)
Genre	(book_id, genre_name)
Review	(customer_id, book_id, rating)
Edition	(book_id, book_edition, edition_type, order_id, price, quantity_in_stock)
Book_order	(order_id, customer_id, street, city, post_code, country, date_ordered, date_delivered)
Order_Contains	(order_id, book_id, book_edition, edition_type, amount)
Supplier	(supplier_id, first_name, last_name, account_number)
Supplier_Phone	(phonenumber, supplier_id)
Supplier_Supplies	(supplier_id, book_id, book_edition, edition_type, supply_price)

The following are the design choices made for the relational schema above:

A given Customer can have multiple phone numbers and addresses which was the reason to have different tables for customer address and phone number.

Customer:

primary-key: customer_id (single valued attribute)

attributes:

- name (composite attribute, varchar) - first_name, last_name
- email (single valued attribute, varchar)

Customer_Phone:

primary-key: phonenumber (single valued attribute, varchar)

foreign key: customer_id

attributes: phone_type (single valued attribute, varchar)

Customer_Address:

primary-key is a foreign key which is customer_id

attributes:

- street (single valued attribute, varchar)
- City (single valued attribute, varchar)
- post_code (single valued, varchar)
- Country (single valued, varchar)

Book has a multi valued attribute genre, so genre will have a separate table.

Book:

primary-key: book_id (single valued, varchar)

attributes:

- Title (single valued, varchar)
- Author (single valued, varchar) - author_name
- Publisher (single valued, varchar)

Genre is the multi valued attribute of book entity.

Genre:

primary-key is a foreign key i.e., book_id + genre_name (varchar).

Review is a relationship between customer and book.

Review:

primary-key: combination of customer_id + book_id, both of which are foreign keys.

rating(single valued, numeric) is the only attribute as the assumption here is that a customer can have only one review for a given book. Customer can change the rating given to a book but only one rating exists at a time.

Edition is in an identifying relationship with book and also in a relation with Book_order and Supplier.

<u>Edition:</u>

primary-key: combination of book_id + book_edition(single valued, numeric) + edition_type(single valued, varchar)

foreign key: order_id (single valued, varchar)

attributes:

- Price (single valued, numeric)
- quatity_in_stock (single valued, varchar)

<u>Book_order:</u>

primary-key: order_id (single valued, varchar)

foreign key: customer_id

attributes:

- Street (single valued, varchar)
- City (single valued, varchar)
- post_code (single valued, varchar)
- Country (single valued, varchar)
- date_ordered (single valued, text)
- date_delivered (single valued, text)

Order_Contains is a relationship between Book_order and Edition entities.

Order_contains:

primary-key: order_id (which is also a foreign key)

foreign-key: book_id

attributes:

- book_edition (single valued, numeric)
- editon_type (single valued, varchar)
- Amount (single valued, numeric)

Supplier also has a multi valued attribute phone which is why it is made a different table for itself.

Supplier:

primary-key: supplier_id (single valued, varchar)

attributes:

- Name (composite attribute, varchar) - first_name, last_name
- account_number (single valued, varchar)

Supplier_Phone:

primary-key: phonenumber (single valued, varchar)

foreign key: supplier_id

Supplier_Supplies is a relationship between edition and supplier.

Supplier_Supplies:

primary-key: combination of supplier_id + book_id + book_edition + edition_type

supply_price is the single valued attribute, numeric type.

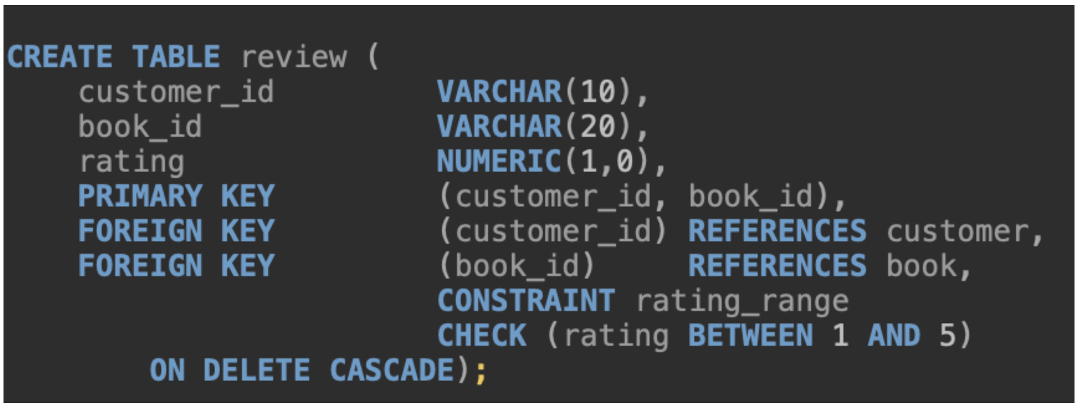
Task-2:

Included cascading actions on all the tables which have the foreign keys to maintain the integrity of the database. customer_phone, customer_address, genre, review, book_order, edition, supplier_phone entities or tables perform cascading only on delete as nothing will change if the parent entities are updated.

order_contains and supplier_supplies both perform cascading on update and delete as they have the attributes such as book_edition, edition_type which on update have to be change in the subsequent child entities as well.

No attribute in any table can have a null value as all the entities are tightly coupled with each other.

There is a constraint for review entity's attribute called rating whose range should be between 1 and 5 as per the requirement.



```
CREATE TABLE review (
    customer_id         VARCHAR(10),
    book_id             VARCHAR(20),
    rating              NUMERIC(1,0),
    PRIMARY KEY         (customer_id, book_id),
    FOREIGN KEY         (customer_id) REFERENCES customer,
    FOREIGN KEY         (book_id)     REFERENCES book,
                        CONSTRAINT rating_range
                        CHECK (rating BETWEEN 1 AND 5)
        ON DELETE CASCADE);
```

Integrity constraints are also enforced with the 'PRAGMA foreign_keys = TRUE;' statement.

Task-3:

Queries:

1. List all books published by “Ultimate Books” which are in the “Science Fiction” genre.

```
SELECT title
      FROM book
      WHERE publisher = 'Ultimate Books'
      AND book.book_id IN (
            SELECT book_id
            FROM genre
            WHERE genre_name = 'Science Fiction');
```

Output:

```
title            |
—————————+
Earth Metals     |
Neptune          |
Zero             |
```

2. List titles and ratings of all books in the “Science and Technology” genre, ordered first by rating (top rated first), and then by the title.

```
SELECT
      review.rating,
      book.title
      FROM review
      NATURAL JOIN book
      WHERE book_id IN (
            SELECT book_id
            FROM genre
            WHERE genre_name = 'Science and Technology')
      ORDER BY rating DESC, title;
```

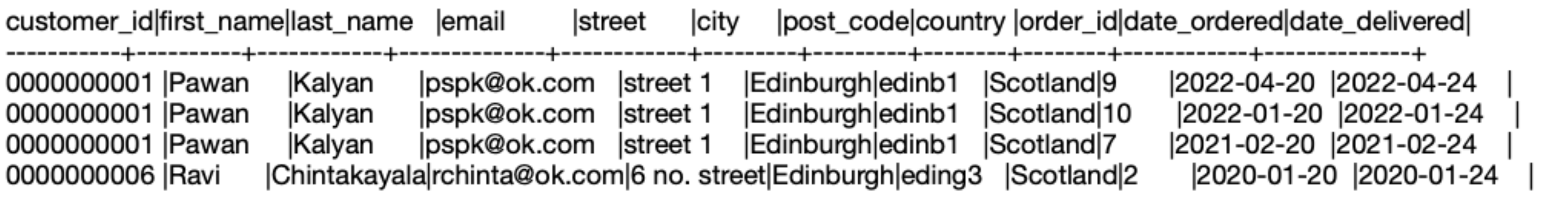
Output:

```
rating|title|
------+-----+
     5|Chips|
     4|Chips|
     4|Nanos|
     3|Nanos|
     3|Super|
     1|Chips|
```

3. List all orders placed by customers with customer address in the city of Edinburgh, since 2020, in chronological order, latest first.

```
SELECT *
    FROM customer
    NATURAL JOIN customer_address
    NATURAL JOIN book_order
    WHERE book_order.date_ordered > '2019-12-31'
    AND customer_address.city = 'Edinburgh'
    ORDER BY date_ordered DESC;
```

Output:



```
customer_id|first_name|last_name  |email         |street     |city     |post_code|country |order_id|date_ordered|date_delivered|
-----------+----------+-----------+--------------+-----------+---------+---------+--------+--------+------------+--------------+
0000000001 |Pawan     |Kalyan     |pspk@ok.com   |street 1   |Edinburgh|edinb1   |Scotland|9       |2022-04-20  |2022-04-24    |
0000000001 |Pawan     |Kalyan     |pspk@ok.com   |street 1   |Edinburgh|edinb1   |Scotland|10      |2022-01-20  |2022-01-24    |
0000000001 |Pawan     |Kalyan     |pspk@ok.com   |street 1   |Edinburgh|edinb1   |Scotland|7       |2021-02-20  |2021-02-24    |
0000000006 |Ravi      |Chintakayala|rchinta@ok.com|6 no. street|Edinburgh|eding3  |Scotland|2       |2020-01-20  |2020-01-24    |
```

4. List all book editions which have less than 5 items in stock, together with the name, account number and supply price of the minimum priced supplier for that edition.

```
SELECT *
      FROM supplier
      NATURAL JOIN edition
      NATURAL JOIN supplier_supplies
      WHERE supplier_supplies.supply_price IN (
            SELECT MIN(supply_price)
                  FROM supplier_supplies
                  GROUP BY supplier_id)
            AND edition.quantity_in_stock < 5;
```

Output:

```
supplier_id|first_name|last_name|account_number|book_id|book_edition|edition_type|order_id|price|quantity_in_stock|supply_price|
-----------+----------+---------+--------------+-------+------------+------------+--------+-----+-----------------+------------+
1a         |Rehman    |Shekar   |ab12          |1      |           1|Hard Cover  |1       |39.99|3                |       19.99|
1c         |Ranga     |Rao      |ab14          |3      |          10|Hard Cover  |3       |   10|2                |           5|
1i         |David     |Russell  |ab20          |9      |           1|Hard Cover  |9       |    8|1                |           4|
```

5. Calculate the total value of all audiobook sales since 2020 for each publisher.

```
SELECT book.publisher, SUM(order_contains.amount)
      FROM book
      NATURAL JOIN order_contains
      NATURAL JOIN book_order
      WHERE order_contains.edition_type = 'Audio Book’
      AND book_order.date_ordered > ‘2019-12-31'
      GROUP BY book.publisher;
```

Output:

```
publisher      |SUM(order_contains.amount)|
———————————+————————————————————+
Authentic Books|                     79.98|
```

6. Calculate the total number of books ordered in 'Science Fiction' genre.

```
SELECT COUNT(*), genre.genre_name
    FROM genre
    NATURAL JOIN order_contains
    WHERE order_contains.book_id = genre.book_id AND
genre.genre_name = 'Science Fiction';
```

Output:

```
COUNT(*)|genre_name    |
————+————————+
       4|Science Fiction|
```

7. List all the phone numbers and customer names that are living in 'Hyderabad' city.

```
SELECT customer_phone.phonenumber,
    customer.first_name, customer.last_name
    FROM customer
    NATURAL JOIN customer_address
    NATURAL JOIN customer_phone
    WHERE customer_address.city = 'Hyderabad';
```

Output:

```
phonenumber|first_name|last_name|
———————-+—————+—————+
1234567893 |Rama      |Rao      |
```

8. List all the orders delivered in less than 4 days.

```
SELECT order_id, JULIANDAY(date_delivered) -
JULIANDAY(date_ordered) AS d
    FROM book_order
    WHERE d < 4;
```

Output:

```
order_id| d  |
————+——+
4         |0.0|
11        |3.0|
```

9. Give the name of the author and book title along with the number of books sold with book_id '5'.

```
SELECT book.author_name,
       book.title,
       COUNT(order_contains.book_id)
       FROM book
       NATURAL JOIN order_contains
       WHERE order_contains.book_id = '5';
```

Output:

```
author_name                    |title     |COUNT(order_contains.book_id)|
——————————————————+———+——————————————————+
Charles Sobhraj, Inkodu Evaro|Nanos |                              2|
```

Views:

1. View for all the details related to Supplier and the books they supply.

```
CREATE VIEW supplier_supplying_books AS
       SELECT *
       FROM supplier
       NATURAL JOIN supplier_phone
       NATURAL JOIN supplier_supplies
       NATURAL JOIN book;
```

Query example for the above view is:
List all the details of the supplier and books they supply with supplier_id '1g'.

SELECT * **FROM** supplier_supplying_books
    WHERE supplier_id = '1g';

<u>Output:</u>

```
supplier_id|first_name   |last_name|account_number|phonenumber|book_id|book_edition|edition_type|supply_price|title       |author_name|publisher    |
-----------+-------------+---------+--------------+-----------+-------+------------+------------+------------+------------+-----------+--------------+
1g         |Chennakeshava|Reddy    |ab18          |123456786  |1      |           3|Hard Cover  |        12.9|Earth Metals|Frank Leo  |Ultimate Books|
```

2. View for all the details of the customers with their email and phone numbers who have rated atleast one book along with the genres.

CREATE VIEW customer_reviewed **AS**
    SELECT *
    FROM customer
    NATURAL JOIN customer_phone
    NATURAL JOIN review
    NATURAL JOIN genre;

Query for the above view is:
List all the customers with their phonenumbers and email who have reviewed for genre 'Science Fiction'.

SELECT *
    FROM customer_reviewed
    WHERE genre_name = 'Science Fiction';

<u>Output:</u>

```
customer_id|first_name|last_name   |email        |phonenumber|phone_type|book_id|rating|genre_name     |
-----------+----------+------------+-------------+-----------+----------+-------+------+---------------+
0000000001 |Pawan     |Kalyan      |pspk@ok.com  |1234567890 |Mobile    |1      |     4|Science Fiction|
0000000002 |Trivikram |Srinivas    |guruji@ok.com |1234567891 |Home      |1      |     4|Science Fiction|
0000000006 |Ravi      |Chintakayala|rchinta@ok.com|1234567895 |Personal  |8      |     2|Science Fiction|
0000000005 |Nageshwar |Rao         |anr@ok.com   |1234567894 |Mobile    |1      |     4|Science
Fiction|
```

3. View for the time taken to deliver all the orders.

```
CREATE VIEW delivery_time AS
      SELECT order_id, JULIANDAY(date_delivered) -
JULIANDAY(date_ordered) AS d
      FROM book_order;
```

Query sample for the above stated view is:
List the delivery time taken for all the orders.

```
SELECT *
 FROM delivery_time;
```

Output:

```
order_id|d   |
--------+----+
1       | 4.0|
2       | 4.0|
3       |34.0|
4       | 0.0|
5       | 4.0|
6       | 4.0|
7       | 4.0|
8       | 4.0|
10      | 4.0|
11      | 3.0|
9       | 4.0|
```

Task-4:

Understanding the scenario wasn't difficult at all and considering all the cardinality and multiplicity constraints and coming up with the relational schema was intuitive. The combination of lectures and the examples provided were useful and at times when the task 2 and 3 were challenging, these were the things that helped with get going. I faced a bit problems with regard to DATE type but going through the documentation fixed the problems. Also in the task 3 i.e., the data manipulation, even though my logic was correct for the query, I was struggling with the format or structure of my query which presented errors but I got the relational schema that I designed, held it as a look up table and structured the queries by looking up the entities and the relationships established. This helped me a lot and made it easier for me to structure my queries.