WolfPubDb

For the WolfCity publishing house

CSC 540 Database Management Systems Project Report 2

Team Members

Suraj Sunil Pawar (spawar2)
Rishal Kamlesh Shah (rshah27)
Ankit Arvind Tiwari (atiwari4)
Amit Prafullkumar Ghetiya (aghetiy)

March 23rd, 2020

Assumption:

1. Publication Id does not depend on anything.
2. Every book or article can have any number of authors except 0.
3. Cost of a book is defined in the database.
4. Key = Primary key (used in defining relations).
5. Publication will have the title attribute which is the name/title of an issue or a book.
6. Book entity will deal with different editions of the book.
7. Issue entity will have different issues of the same publication.
8. Articles can belong to more than one Issue.
9. Contributor entity store temporary / guest or permanent employees information.
10. Order can be made only of a particular book edition or an issue. If a distributor wants more books or an issue, separate order must be made by them.

1. Global Relational Database Schema:

Publication(publication_id, title, typical_topics, type, periodicity):
publication_id-> title, typical_topics, type, periodicity
This relation holds true because each publication_id represents a unique publication. We can get all the information like title, typical_topics, type and periodicity related to that particular publication if we have the publication_id. No other combination of the other attributes would hold as it would limit the possibilities of the data in our database. Since, the publication_id is a key, therefore the functional dependency (FD) is in BCNF (and therefore also in 3NF).

Book(ISBN, date_of_creation, publication_date, edition_no, publication_id):
ISBN-> date_of_creation, publication_date, edition_no, publication_id
publication_id, edition_no -> ISBN, date_of_creation, publication_date
ISBN represents a unique book which will then give us all the details related to the corresponding book. Since, ISBN is a key, thus this functional dependency satisfies BCNF and 3NF criteria.
The second relation also satisfies BCNF and 3NF criteria as there are two attributes on the left hand side of the functional dependency, which combined is also the superkey of the entity.
Thus, this table is in 3NF.

Chapter(chapter_id, chapter_no, chapter_name, content, isbn)
chapter_id -> chapter_no, chapter_name, content, isbn
chapter_no, isbn -> chapter_id, chapter_name, content
The chapter_id identifies a unique chapter in a book and provides all the details of that particular book. Since chapter_id is a key, hence the FD satisfies BCNF (and also 3NF) criteria.
The second FD shows that any combination of chapter_no and isbn is always unique and hence will give the details of the book. Since these two attributes combined on the left side of FD are a superkey, this FD also satisfies BCNF (and 3NF) criteria.
Thus, this table is in 3NF.

Issue(issue_id, date_of_issue, publication_id):
issue_id -> date_of_issue, publication_id
Each issue_id refers to a distinct issue and hence gives us all the information related to an issue like date_of_issue and its publication_id which has all the corresponding publication details. The issue_id is a key, and hence the functional dependency satisfies BCNF and 3NF criteria.
Thus, this table is in 3NF.

Article(<u>article_id</u>, title, date_of_creation, content):
<u>article_id</u> -> title, date_of_creation, content
article_id represents the article and its details. Using the article_id we can get all the details of an article. Since the article_id is a key, hence the functional dependency satisfies BCNF and 3NF criteria.
Thus, this table is in 3NF.

bookAuthor(<u>isbn</u>, <u>author_id</u>):
<u>Isbn, author_id</u> -> isbn,author_id
Since, there are only 2 attributes in this relation, the table is in 3NF.

articleAuthor(<u>article_id, author_id</u>):
<u>article_id</u>, <u>author_id</u> -> article_id, author_id
Since, there are only 2 attributes in this relation, the table is in 3NF.

consistOf(<u>issue_id</u>, <u>article_id</u>):
<u>issue_id</u>, <u>article_id</u> -> issue_id, article_id
Since, there are only 2 attributes in this relation, the table is in 3NF.

Contributor(<u>contributor_id</u>, name, designation, job_type, salary, payment_frequency):
<u>contributor_id</u> -> name, designation, job_type, salary, payment_frequency
The contributor_id provides the details of a contributor like name, designation, etc. None of the other attributes can be used to get all the unique details of a contributor. Also, the contributor_id is a key, and hence, the functional dependency satisfies BCNF and 3NF criteria.
Thus, this table is in 3NF.

WorksFor(<u>contributor_id</u>, <u>publication_id</u>):
<u>contributor_id</u>, <u>publication_id</u> -> contributor_id, publication_id
Since, there are only 2 attributes in this relation, the table is in 3NF.

Pays(<u>contributor_id</u>, <u>publication_id</u>, amount, payment_date):
<u>contributor_id</u>, <u>publication_id</u> -> amount, payemnt_date
The combination of contributor_id and publication_id lets us know the amount that the publisher is paying to the contributor and on which date the payment was made. Since, the left hand side of FD are 2 attributes whose combination is the key of the table, hence, the functional dependency satisfies BCNF and 3NF criteria.
Thus, this table is in 3NF.

Distributor(distributor_id, name, dist_type, address, city, contact_person, balance, phone_number):

distributor_id-> name, dist_type, address, city, contact_person, balance, phone_number

The above FD provides the details of the distributors using the distributor_id. All the details including name, dist_type, address, etc. can be picked up from the distributor_id. None of the other attributes can be used to extract unique information about the distributor. For eg: there can be multiple distributors in the same city or with the same name. Hence, they cannot be used to uniquely identify a distributor. Since the distributor_id is a key, hence the functional dependency satisfies BCNF and 3NF criteria.

Thus, this table is in 3NF.

Order(order_id, shipping_cost, price, order_date, number_of_copies, book_id, issue_id, payment_status, distributor_id):

order_id -> shipping_cost, price, order_date, number_of_copies, book_id, issue_id, payment_status, distributor_id

The order_id provides all the details of an order made by the distributor. The orders contain information on what the order contains, its quantity, price, etc. and all these details are referenced by an order_id. The other attributes do not uniquely identify an order as multiple orders can be made on the same book or issue or by the same distributor. Hence, there is only one FD that is viable. Since the order_id is a key, hence the functional dependency satisfies BCNF and 3NF criteria.

Thus, this table is in 3NF.

2. Design for Global Schema:

2) (120 points) Describe any design decisions for the global schema. Identify and explain all integrity constraints of the following types: NOT NULL, key, and referential integrity. Describe which attributes are allowed to be NULL, why, and what a NULL value means for each attribute on which it is allowed. Submit all your descriptions and explanations.

- If not mentioned, then the assumption for any attribute is "Not a key, NOT NULL, No referential entity". If only partial description is given, it means that all the other constraints which are not mentioned are having default values (eg: if NULL is not mentioned but some other constraints are mentioned, then assume NOT NULL (default) for that attribute).
- The Key is the primary key.

Publication (
publication_id key,
title,
typical_topics NULL,
type,
periodicity
)

Explanation:
The "title" attribute can be the same for many publications. Hence we will use "publication_id" as the key which will be used to identify different publications.
The attribute typical_topics can be null as not every publisher tends to brand their publication into a label. Also, typical topics do not make sense for a novel, as it would have more of a "genre", than a typical topic.

Book(
ISBN key,
date_of_creation,
publication_date,
combination of publication_id and edition_no must be unique,
publication_id reference to id in a publication
)

Explanation:
The publication will have the title of the book but books tend to have more than one edition over time and hence each edition has its own isbn (the key). Also, this entity has a reference to the publication entity (publication_id). When the distributors have to order books, then they would not reference the publication but the isbn of the book. This will specify which edition they want.

Chapter (
chapter_id key,
chapter_no,
chapter_name,
content,
isbn reference to id in Book,
combination of chapter_no and isbn must be unique
)

Explanation:
Books have Chapters. Here each chapter is associated with an ISBN. Sometimes, different editions have different content within the same chapter. This entity will allow such scenarios. The chapter_id will be the key. We could have used chapter_no, chapter_name & "ISBN" together as the key but chapter_id alone is more efficient. The last attribute isbn references the isbn in the book entity.

Issue(
issue_id key,
date_of_issue,
publication_id reference to publication_id in a publication
)

Explanation:
An issue can be recognized by its id. It will be tied to a publication and that's why we have a reference to the publication entity. When a distributor has to order a publication's magazine or journal, it would want a specific issue of it. Hence, it would order with issue's id instead of publication's id.

[* Only one among book_id or issue_id can be null]

Order (

order_id key,
shipping_cost,
price,
order_date,
no_of_copies,
book_id*,
issue_id*,
payment_status,
distributorID reference to distributor_id in Distributor
book_id reference to isbn in Book
issue_id reference to issue_id in Issue

)

Explanation:

Each order will have an ID (order_id). Each order will have a reference to the distributor which has placed the order.

Now, we have given two attributes, book_id (a reference to isbn in Book) and issue_id (a reference to issue_id in Issue), to recognize what package has been ordered. If books are ordered, book_id is filled and the issue_id is null and vice-versa when an issue is ordered. But both can't be null or both can't be filled; this is ensured by a trigger that is defined.

Another trigger calculates total price as shipping cost + no of copies * price and then adds it to the distributor's balance when inserted. If the payment is done, another trigger updates the distributor's balance.

consistOf (

issueID and articleID combined are the key,
issueID is the reference to the key of Issue entity,
articleID is the reference to the key of Article Entity

)

Explanation:

This entity provides many to many relationship support between Issue and Article. Since articles are generally re-used in many issues by different publications, we decided to make it a many to many relation. issueID and articleID together become the key. Both are referenced to their respective table.

Contributor (
    id key,
    name,
    designation,
    job_type permanent or temp,
    salary NULL,
    payment_frequency,
)

Explanation:
Contributors can be a guest employee or a permanent one, as job_type specifies. Here the salary is allowed to be null as temporary employees or guest employees may not have a salary defined. Their transaction can be captured by the Pays entity.

worksFor(
    contributorID and publicationID combined are the key,
    contributorID is the reference to id in Contributor,
    publicationID is the reference to id in Publication
)

Explanation:
Different Contributors work for many publications and hence there is a many to many relationships. This entity captures that. Both the columns together are keys and have reference to their parent table.

Pays (
    contributor_id is the reference to contributor_id in Contributor,
    publication_id is the reference to publication_id in Publication,
    contributor_id, publication_id combined are the key,
    amount,
    payment_date
)

Explanation:
This entity captures how much pay each contributor gets and references the publication for which they are getting paid. This assumes that each permanent employee works for at least one publication in order to get paid. IDs of contributor and publication together form a key and both are referenced to their parent table.

Distributor(
id key,
name,
type NULL,
address,
city,
contact_person,
balance,
phone_number
)

Explanation:
Each distributor is identified by a unique key.
contact_person and phone_number are not null because we are assuming that distributors who order must be providing a contact person and number for communication.
Distributor type can be NULL as publication houses may decide to accept an order from individuals and they may not fall under any type provided in the narrative.
The balance attribute is changed automatically when an order is made or updated by a trigger.

Article (
article_id key,
title,
date_of_creation,
content
)

Explanation:
Each article is given an unique id and the details of the article are present (not null) in this table.

bookAuthor (
isbn is the reference to isbn in Book,
author_id is the reference to contributor_id in Contributor,
isbn and author_id combined are the keys
)

Explanation: ISBN is referenced from the Book entity and author_id is referenced from the contributor_id from the Contributor entity. The combination of both the attributes is the key for this entity.

articleAuthor(

article_id is the reference to article_id in Article,

author_id is the reference to contributor_id in Contributor,

article_id and author_id combined are the key

)

Explanation:

The article_id is referenced from the Article entity and author_id is referenced from the contributor_id from Contributor entity. The combination of both the attributes is the key for this entity.

3. Base Relations:

```
CREATE TABLE Publication (
        publication_id      INT,
        title               VARCHAR(128)    NOT NULL,
        typical_topics      VARCHAR(128),
        type                VARCHAR(128)    NOT NULL,
        periodicity         VARCHAR(128)    NOT NULL,
        PRIMARY KEY(publication_id)
);

CREATE TABLE Book (
        isbn                VARCHAR(128),
        publication_id      INT             NOT NULL,
        date_of_creation    DATE            NOT NULL,
        publication_date    DATE            NOT NULL,
        edition_no          VARCHAR(128)    NOT NULL,
        PRIMARY KEY(isbn),
        UNIQUE(publication_id, edition_no),
        FOREIGN KEY(publication_id) REFERENCES Publication(publication_id)
                ON UPDATE RESTRICT
);

CREATE TABLE Chapter (
        chapter_id          INT             NOT NULL,
        chapter_no          INT             NOT NULL,
        chapter_name        VARCHAR(128)    NOT NULL,
        content             VARCHAR(128)    NOT NULL,
        isbn                VARCHAR(128)    NOT NULL,
        UNIQUE (chapter_no, isbn),
        PRIMARY KEY(chapter_id),
        FOREIGN KEY(isbn) REFERENCES Book(isbn) ON UPDATE RESTRICT
);
```

```
CREATE TABLE Issue (
        issue_id            INT,
        date_of_issue       DATE        NOT NULL,
        publication_id      INT,
        PRIMARY KEY(issue_id),
        FOREIGN KEY(publication_id) REFERENCES Publication(publication_id)
                ON UPDATE RESTRICT
);

CREATE TABLE `Order` (
        order_id            INT             NOT NULL,
        shipping_cost       INT             NOT NULL,
        price               INT             NOT NULL,
        order_date          DATE            NOT NULL,
        no_of_copies        INT             NOT NULL,
        book_id             VARCHAR(128),
        issue_id            INT,
        payment_status      VARCHAR(128)    NOT NULL,
        distributor_id      INT             NOT NULL,
        PRIMARY KEY(order_id),
        FOREIGN KEY(distributor_id) REFERENCES Distributor(distributor_id)
                ON UPDATE RESTRICT,
        FOREIGN KEY(book_id) REFERENCES Book(isbn)
                ON UPDATE RESTRICT,
        FOREIGN KEY(issue_id) REFERENCES Issue(issue_id)
                ON UPDATE RESTRICT,
        CONSTRAINT bookOrIssuePresent CHECK (book_id is NULL XOR issue_id is
NULL)
);


CREATE TABLE Article (
        article_id          INT,
        title               VARCHAR(128)    NOT NULL,
        date_of_creation    DATE            NOT NULL,
        content             VARCHAR(128)    NOT NULL,
        PRIMARY KEY(article_id)
);
```

```
CREATE TABLE consistOf (
        issue_id                INT,
        article_id              INT,
        PRIMARY KEY(issue_id, article_id),

        FOREIGN KEY(issue_id) REFERENCES Issue(issue_id)
                        ON UPDATE RESTRICT,
        FOREIGN KEY(article_id) REFERENCES Article(article_id)
                        ON UPDATE RESTRICT
);

CREATE TABLE bookAuthor(
        isbn            VARCHAR(128),
        author_id       INT,
        PRIMARY KEY(isbn, author_id),

        FOREIGN KEY(isbn) REFERENCES Book(isbn)
                        ON UPDATE RESTRICT,
        FOREIGN KEY(author_id) REFERENCES Contributor(contributor_id)
                        ON UPDATE RESTRICT
);

CREATE TABLE articleAuthor(
        article_id      INT,
        author_id       INT,
        PRIMARY KEY(article_id, author_id),

        FOREIGN KEY(article_id) REFERENCES Article(article_id)
                        ON UPDATE RESTRICT,
        FOREIGN KEY(author_id) REFERENCES Contributor(contributor_id)
                        ON UPDATE RESTRICT
);
```

```
CREATE TABLE  worksFor(
        contributor_id          INT,
        publication_id          INT,
        PRIMARY KEY(contributor_id, publication_id),

        FOREIGN KEY(contributor_id) REFERENCES Contributor(contributor_id)
                        ON UPDATE RESTRICT,
        FOREIGN KEY(publication_id) REFERENCES Publication(publication_id)
                        ON UPDATE RESTRICT
);

CREATE TABLE Contributor (
        contributor_id          INT,
        name                    VARCHAR(128)    NOT NULL,
        designation             VARCHAR(128)    NOT NULL,
        job_type                VARCHAR(128)    NOT NULL,
        salary                  INT             NULL,
        payment_frequency       VARCHAR (128)   NOT NULL,
        PRIMARY KEY(contributor_id)
);

CREATE TABLE Distributor (
        distributor_id          INT,
        name                    VARCHAR(128)    NOT NULL,
        dist_type               VARCHAR(128),
        address                 VARCHAR(128)    NOT NULL,
        city                    VARCHAR(128)    NOT NULL,
        contact_person          VARCHAR(128)    NOT NULL,
        balance                 INT             NOT NULL,
        phone_number            VARCHAR(128)    NOT NULL,
        PRIMARY KEY(distributor_id)
);
```

```
CREATE TABLE Pays(
        contributor_id          INT,
        publication_id          INT,
        amount                  INT     NOT NULL,
        payment_date            DATE NOT NULL,
        PRIMARY KEY(contributor_id, publication_id),
        FOREIGN KEY(contributor_id) REFERENCES Contributor(contributor_id)
                        ON UPDATE RESTRICT,
        FOREIGN KEY(publication_id) REFERENCES Publication(publication_id)
                        ON UPDATE RESTRICT
);
```

Triggers:

DELIMITER $$

1) This trigger ensures that only one of the book_id or issue_id is NULL. Also, it checks if the status of the payment is pending and automatically adds to the balance of the distributor.
This trigger runs before an insertion in the Order table.

```
Create Trigger `checkIfBookOrIssuePresent` BEFORE INSERT ON `Order`
for each row
begin
        DECLARE totalPrice INT;
        SET totalPrice= NEW.shipping_cost + NEW.price*NEW.no_of_copies;
        IF NEW.book_id is NULL THEN
        IF NEW.issue_id is NULL THEN
        signal sqlstate '45000' set message_text = 'Only one of book_id or issue_id must be null';
        END IF;
        END IF;
        IF NEW.book_id is NOT NULL THEN
        IF NEW.issue_id is NOT NULL THEN
        signal sqlstate '45000' set message_text = 'Only one of book_id or issue_id must be null';
        END IF;
        END IF;
        IF NEW.payment_status = "PENDING" THEN
        update `Distributor` set balance = balance + totalPrice where distributor_id =
NEW.distributor_id;
        END IF;
END $$
DELIMITER ;
```

2) This trigger ensures that only one of the book_id or issue_id is NULL. Also, it checks if the status of the payment is paid and automatically deducts from the balance of the distributor.
This trigger runs before an update on the Order table.

```
DELIMITER $$
Create Trigger `checkIfBookOrIssuePresentUpdate` BEFORE UPDATE ON `Order`
for each row
begin
	DECLARE totalPrice INT;
	SET totalPrice= NEW.shipping_cost + NEW.price*NEW.no_of_copies;
	IF NEW.book_id is NULL THEN
	IF NEW.issue_id is NULL THEN
	signal sqlstate '45000' set message_text = 'Only one of book_id or issue_id must be null';
	END IF;
	END IF;
	IF NEW.book_id is NOT NULL THEN
	IF NEW.issue_id is NOT NULL THEN
	signal sqlstate '45000' set message_text = 'Only one of book_id or issue_id must be null';
	END IF;
	END IF;

	IF NEW.payment_status = "PAID" THEN
	update `Distributor` set balance = balance - totalPrice where distributor_id = NEW.distributor_id;
	END IF;
END $$
DELIMITER ;
```

Select * statements:

MariaDB [atiwari4]> show tables;

```
+--------------------+
| Tables_in_atiwari4 |
+--------------------+
| Article            |
| Book               |
| Chapter            |
| Contributor        |
| Distributor        |
| Issue              |
| Order              |
| Pays               |
| Publication        |
| articleAuthor      |
| bookAuthor         |
| consistOf          |
| worksFor           |
+--------------------+
13 rows in set (0.00 sec)
```

MariaDB [atiwari4]> select * from Article;

article_id	date_of_creation	content
1	2019-05-02	We are not prepared for a pandemic
2	2019-05-11	This might be a good time to start preparing for a pandemic
3	2019-05-08	Ironman saves the day by sacrificing himself
4	2019-11-02	Who is Spiderman ?
5	2019-01-28	What goes with golden? The answer may not shock you. Its nothing. STRAP witht the golden
6	2019-05-02	0
7	2019-06-11	Is math related to science
8	2019-08-16	The check shirts are making a comeback

8 rows in set (0.00 sec)

MariaDB [atiwari4]> select * from Book;

isbn	publication_id	date_of_creation	publication_date	edition_no
123456789	1	2003-06-21	2003-07-21	1
456123789	2	1813-01-28	1813-01-31	1
741258963	2	1813-07-05	1813-05-31	2
987654321	1	2007-07-05	2007-08-05	2

4 rows in set (0.00 sec)

```
MariaDB [atiwari4]> select * from Chapter;
+------------+------------+---------------+-----------------------------------------------------------------+-----------+
| chapter_id | chapter_no | chapter_name  | content                                                         | isbn      |
+------------+------------+---------------+-----------------------------------------------------------------+-----------+
|          1 |          1 | Dudley Demented | Harry saves Dudley                                            | 123456789 |
|          2 |          1 | Dudley Demented | The best Marvel movie of 2020                                 | 987654321 |
|          3 |          2 | A pack of Dals  | Ministry of Magic arrests Harry                               | 123456789 |
|          4 |          2 | A pack of Dals  | Ministry of Magic tries to arrest Harry but Dumbledore saves him | 987654321 |
|          5 |          1 | Chapter 1       | Mrs. Bennet tries to get her daughters married                | 456123789 |
|          6 |          1 | Chapter 1       | Mrs. Bennet tries to get her daughters married                | 741258963 |
|          7 |          2 | Chapter 2       | Mr. Bennet is interested but acts as if he is not             | 741258963 |
|          8 |          2 | Chapter 2       | Mr. Bennet is not interested                                  | 456123789 |
|          9 |          9 | Black Widow     | The best movie of 2020                                        | 987654321 |
+------------+------------+---------------+-----------------------------------------------------------------+-----------+
9 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> select * from Contributor;
+----------------+-------------+-------------+-----------+--------+--------------------+
| contributor_id | name        | designation | job_type  | salary | payment_frequency  |
+----------------+-------------+-------------+-----------+--------+--------------------+
|              1 | Jane Austen | Editor      | Permanent |  12000 | Monthly            |
|              2 | Tom         | Freelance   | Temporary |   NULL | Once               |
|              3 | Dick        | Freelance   | Temporary |   NULL | Once               |
|              4 | Harry       | Freelance   | Temporary |   NULL | Once               |
|              5 | JK Rowling  | Author      | Permanent | 250000 | Annual             |
+----------------+-------------+-------------+-----------+--------+--------------------+
5 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> select * from Distributor;
+----------------+-------------+-------------------+-----------------------+-----------+-----------------+---------+--------------+
| distributor_id | name        | dist_type         | address               | city      | contact_person  | balance | phone_number |
+----------------+-------------+-------------------+-----------------------+-----------+-----------------+---------+--------------+
|              1 | Llama Drama | wholesale distributor | 1354-CC-NC-27605    | Raleigh   | Daniel Bothelel |     269 | 987258788    |
|              2 | NCSU Library | library          | Cent-Raleigh-NC-27606 | Raleigh   | Matt Jones      |    7490 | 785584256    |
|              3 | Billys      | bookstore         | 87-AA-NC-27605        | Durham    | Billy Boi       |    1569 | 898432154    |
|              4 | Swedish Nerd | bookstore        | 4448-LL-NC-27606      | Apex      | Felix Kjelberg  |    7289 | 647850969    |
+----------------+-------------+-------------------+-----------------------+-----------+-----------------+---------+--------------+
4 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> select * from Issue;
+----------+---------------+----------------+
| issue_id | date_of_issue | publication_id |
+----------+---------------+----------------+
|        1 | 2020-02-04    |              5 |
|        2 | 2020-02-11    |              5 |
|        3 | 2020-01-01    |              4 |
|        4 | 2020-02-01    |              4 |
|        5 | 2020-01-01    |              3 |
|        6 | 2020-01-02    |              3 |
+----------+---------------+----------------+
6 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> select * from `Order`;
+----------+---------------+-------+------------+---------------+-----------+----------+----------------+----------------+
| order_id | shipping_cost | price | order_date | no_of_copies  | book_id   | issue_id | payment_status | distributor_id |
+----------+---------------+-------+------------+---------------+-----------+----------+----------------+----------------+
|        1 |            20 |    20 | 2020-01-02 |            59 | 987654321 |     NULL | Pending        |              1 |
|        2 |            20 |    10 | 2020-02-10 |            78 | 741258963 |     NULL | Pending        |              2 |
|        3 |             0 |    10 | 2020-05-15 |           150 | NULL      |        1 | Pending        |              3 |
|        4 |             0 |   100 | 2019-10-10 |            72 | NULL      |        4 | Pending        |              4 |
|        5 |            30 |    10 | 2019-08-15 |            69 | 741258963 |     NULL | Pending        |              2 |
|        6 |            15 |    15 | 2020-05-15 |            47 | NULL      |        2 | Pending        |              2 |
+----------+---------------+-------+------------+---------------+-----------+----------+----------------+----------------+
6 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> select * from Pays;
+---------------+----------------+--------+--------------+
| contributor_id | publication_id | amount | payment_date |
+---------------+----------------+--------+--------------+
|              1 |              2 |  66000 | 2019-12-30   |
|              1 |              5 |   6969 | 0000-00-00   |
|              2 |              3 |   1200 | 2020-01-05   |
|              3 |              2 |   6969 | 0000-00-00   |
|              3 |              4 |   2900 | 2019-06-09   |
|              4 |              5 |     22 | 2019-09-06   |
|              5 |              1 |  32000 | 2020-03-20   |
+----------------+----------------+--------+--------------+
7 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> select * from Publication;
+---------------+-----------------------------------+----------------+---------+-------------+
| publication_id | title                             | typical_topics | type    | periodicity |
+---------------+-----------------------------------+----------------+---------+-------------+
|              1 | Harry Potter and the order of phoenix | Fiction        | Book    | Once        |
|              2 | Pride and Prejudice               | Fiction        | Book    | Once        |
|              3 | Everyday Science                  | Science        | Journal | Weekly      |
|              4 | Avengers                          | Science Fiction | Magazine | Monthly     |
|              5 | Vogue                             | Fashion        | Magazine | Weekly      |
|              6 | ZigWheels                         | Cars           | Magazine | Monthly     |
+---------------+-----------------------------------+----------------+---------+-------------+
6 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> select * from articleAuthor;
+------------+-----------+
| article_id | author_id |
+------------+-----------+
|          1 |         2 |
|          2 |         2 |
|          3 |         3 |
|          4 |         3 |
|          5 |         2 |
|          6 |         2 |
|          7 |         4 |
|          8 |         4 |
+------------+-----------+
8 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> select * from bookAuthor;
+-----------+-----------+
| isbn      | author_id |
+-----------+-----------+
| 123456789 |         5 |
| 456123789 |         1 |
| 741258963 |         1 |
| 987654321 |         5 |
+-----------+-----------+
4 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> select * from consistOf;
+----------+------------+
| issue_id | article_id |
+----------+------------+
|        1 |          5 |
|        1 |          6 |
|        2 |          8 |
|        3 |          4 |
|        3 |          7 |
|        4 |          3 |
|        4 |          7 |
|        5 |          1 |
|        5 |          2 |
|        6 |          7 |
+----------+------------+
10 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> select * from worksFor;
+----------------+----------------+
| contributor_id | publication_id |
+----------------+----------------+
|              1 |              2 |
|              1 |              5 |
|              2 |              3 |
|              2 |              4 |
|              2 |              5 |
|              3 |              3 |
|              3 |              4 |
|              3 |              5 |
|              4 |              3 |
|              4 |              4 |
|              4 |              5 |
|              5 |              1 |
+----------------+----------------+
12 rows in set (0.00 sec)
```

4. SQL Queries

Editing and Publishing:

Enter publication info:
MariaDB [atiwari4]> INSERT INTO Publication VALUES (6, 'ZigWheels', 'Cars', 'Magazine', 'Monthly');
Query OK, 1 row affected (0.00 sec)

Update Publication Info:
MariaDB [atiwari4]> UPDATE Publication SET periodicity = 'Weekly' where publication_id = 3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Assign Editor(s) with id 1 & 4 to publication with id 5:
MariaDB [atiwari4]> INSERT INTO worksFor VALUES (2,5), (1, 5);
Query OK, 1 row affected (0.00 sec)

Let each editor = 1 view the information on the publications he/she is responsible for.

```
MariaDB [atiwari4]> SELECT * FROM Publication
    -> WHERE publication_id
    -> IN
    -> (SELECT publication_id FROM worksFor
    -> WHERE contributor_id = 1);
+----------------+---------------------+----------------+----------+-------------+
| publication_id | title               | typical_topics | type     | periodicity |
+----------------+---------------------+----------------+----------+-------------+
|              2 | Pride and Prejudice | Fiction        | Book     | Once        |
|              5 | Vogue               | Fashion        | Magazine | Weekly      |
+----------------+---------------------+----------------+----------+-------------+
2 rows in set (0.00 sec)
```

Add article number 7 into periodic publication issue number 3:
MariaDB [atiwari4]> INSERT INTO consistOf VALUES (3, 7);
Query OK, 1 row affected (0.00 sec)

Production of a book edition or of an issue of a publication:

Enter new book / issue of a publication

MariaDB [atiwari4]> INSERT INTO Book VALUES (123456788, 1, '2003-06-21',' 2003-11-21', 3);
Query OK, 1 row affected (0.00 sec)

Update book edition / issue

MariaDB [atiwari4]> UPDATE Book SET edition_no = 1 where isbn = 123456789;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1 Changed: 0 Warnings: 0

Delete book edition / issue

MariaDB [atiwari4]> DELETE FROM Book WHERE isbn = 123456788;
Query OK, 1 row affected (0.01 sec)

Enter article 7 into issue 4

MariaDB [atiwari4]> INSERT INTO consistOf VALUES (4, 7);
Query OK, 1 row affected (0.00 sec)

Update article 6's title and date

MariaDB [atiwari4]> UPDATE Article SET content='Black Widow is back with a new s*i*' , date_of_creation = '2020-02-29' where article_id=6;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Enter chapter 9 to Book (987654321)

MariaDB [atiwari4]> INSERT INTO Chapter VALUES (9, 9, 'Black Widow', 'The best movie of 2020', 987654321);
Query OK, 1 row affected (0.01 sec)

Update chapter 2 of book 987654321

MariaDB [atiwari4]> UPDATE Chapter SET content='The best Marvel movie of 2020' where chapter_id = 2;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1 Changed: 0 Warnings: 0

Update text of an article 4

MariaDB [atiwari4]> UPDATE Article SET content = 'Who is Spiderman ?' where article_id = 4;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Find books and articles by topic, date, author's name

```
MariaDB [atiwari4]> SELECT B.isbn AS 'Book/Article', publication_date AS Date
    -> FROM Book B
    -> WHERE publication_date = '2003-07-21'
    -> UNION
    -> SELECT A.article_id AS 'Book/Article', date_of_creation AS Date
    -> FROM Article A
    -> WHERE date_of_creation = '2003-07-21';
+--------------+------------+
| Book/Article | Date       |
+--------------+------------+
| 123456789    | 2003-07-21 |
+--------------+------------+
1 row in set (0.00 sec)
```

```
MariaDB [atiwari4]> SELECT X.isbn AS 'Book/Article', Y.typical_topics AS Topic
    -> FROM Book X
    -> JOIN Publication Y ON X.publication_id = Y.publication_id
    -> WHERE typical_topics = 'Fiction'
    -> UNION
    -> SELECT A.article_id AS 'Book/Article' , C.typical_topics AS Topic
    -> FROM consistOf A
    -> JOIN Issue B ON A.issue_id = B.issue_id
    -> JOIN Publication C ON B.publication_id = C.publication_id
    -> WHERE  typical_topics = 'Fiction';
+--------------+---------+
| Book/Article | Topic   |
+--------------+---------+
| 123456789    | Fiction |
| 987654321    | Fiction |
| 456123789    | Fiction |
| 741258963    | Fiction |
+--------------+---------+
4 rows in set (0.00 sec)
```

```
MariaDB [atiwari4]> SELECT B.isbn AS 'Book/Article', name
    -> FROM Book B JOIN bookAuthor BA
    -> ON B.isbn = BA.isbn
    -> JOIN Contributor C
    -> ON C.contributor_id = BA.author_id
    -> WHERE name = 'JK Rowling'
    -> UNION
    -> SELECT D.article_id AS 'Book/Article', name
    -> FROM Article D JOIN articleAuthor E
    -> ON D.article_id = E.author_id
    -> JOIN Contributor F
    -> ON F.contributor_id = E.author_id
    -> WHERE name = 'JK Rowling';
+--------------+------------+
| Book/Article | name       |
+--------------+------------+
| 123456789    | JK Rowling |
| 987654321    | JK Rowling |
+--------------+------------+
2 rows in set (0.00 sec)
```

Enter payment for author or editor

MariaDB [atiwari4]> INSERT INTO Pays VALUES (1, 5, 6969, 2020-03-20);
Query OK, 1 row affected, 1 warning (0.00 sec)

Keep track of when each payment was claimed by its addressee.

```
MariaDB [atiwari4]> SELECT * FROM Pays WHERE contributor_id = 2;
+----------------+----------------+--------+--------------+
| contributor_id | publication_id | amount | payment_date |
+----------------+----------------+--------+--------------+
|              2 |              3 |   1200 | 2020-01-05   |
+----------------+----------------+--------+--------------+
1 row in set (0.00 sec)
```

Distribution:

Enter new distributor

MariaDB [atiwari4]> INSERT INTO Distributor VALUES(5, 'XYZ', 'bookstore', '2364-CC-NC-27606', 'California', 'John Doe', 1200, 987258964);
Query OK, 1 row affected (0.01 sec)

Update distributor information

MariaDB [atiwari4]> UPDATE Distributor SET phone_number = 987258968 WHERE distributor_id = 1;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1 Changed: 0 Warnings: 0

Delete a distributor

MariaDB [atiwari4]> DELETE FROM Distributor WHERE distributor_id = 5;
Query OK, 1 row affected (0.00 sec)

Input orders from distributors, for a book edition or an issue of a publication per distributor, for a certain date.

MariaDB [atiwari4]> INSERT INTO `Order` VALUES(6,30, 40, '2020-03-04', 69, 741258963, NULL, 'Pending', 2);
Query OK, 1 row affected (0.00 sec)

Bill distributor for an order

[Assumption: Cost of a book is not defined in the database]

```
MariaDB [atiwari4]> SELECT SUM(price* no_of_copies) AS Total_Cost FROM `Order` WHERE order_id = 2;
+------------+
| Total_Cost |
+------------+
|        780 |
+------------+
1 row in set (0.00 sec)
```

Change the outstanding balance of a distributor on receipt of payment.

NOTE: For this query, a trigger has been used to change payment status in Order table and referenced in Portion 3 of the report.

MariaDB [atiwari4]> UPDATE Distributor SET balance = 200 WHERE distributor_id = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Reports:

Generate monthly reports: The number and total price of copies of each publication bought per distributor per month

```
MariaDB [atiwari4]> SELECT
    -> SUM(price*no_of_copies), SUM(no_of_copies), distributor_id,
    -> DATE_FORMAT(order_date,'%M %Y'), Book.publication_id from `Order`, `Book`
    -> WHERE
    -> `Order`.book_id = `Book`.isbn
    -> GROUP BY
    -> distributor_id, publication_id, DATE_FORMAT(order_date,'%M %Y')
    -> UNION ALL
    -> SELECT
    -> SUM(price*no_of_copies), SUM(no_of_copies), distributor_id,
    -> DATE_FORMAT(order_date,'%M %Y'), Issue.publication_id
    -> FROM
    -> `Order`, `Issue`
    -> WHERE
    -> `Order`.issue_id = `Issue`.issue_id
    -> GROUP BY
    -> distributor_id, publication_id, DATE_FORMAT(order_date,'%M %Y');
+-----------------------+-------------------+---------------+-----------------------------------+----------------+
| SUM(price*no_of_copies) | SUM(no_of_copies) | distributor_id | DATE_FORMAT(order_date,'%M %Y') | publication_id |
+-----------------------+-------------------+---------------+-----------------------------------+----------------+
|                  1180 |                59 |             1 | January 2020                      |              1 |
|                   690 |                69 |             2 | August 2019                       |              2 |
|                   700 |                70 |             2 | February 2020                     |              2 |
|                  1175 |                47 |             2 | March 2020                        |              5 |
|                  1500 |               150 |             3 | March 2020                        |              5 |
|                  7200 |                72 |             4 | October 2019                      |              4 |
+-----------------------+-------------------+---------------+-----------------------------------+----------------+
6 rows in set (0.00 sec)
```

Total revenue of the publishing house

MariaDB [atiwari4]> SELECT SUM(no_of_copies*price) AS total FROM `Order`;

```
MariaDB [atiwari4]> SELECT SUM(no_of_copies*price) AS total FROM `Order`;
+-------+
| total |
+-------+
| 12445 |
+-------+
1 row in set (0.00 sec)
```

Total expenses (i.e., shipping costs and salaries)

```
MariaDB [atiwari4]> SELECT SUM(price*no_of_copies) AS TOTAL FROM `Order` WHERE shipping_cost=0;
+-------+
| TOTAL |
+-------+
|  6700 |
+-------+
1 row in set (0.00 sec)
```

```
MariaDB [atiwari4]> SELECT SUM(shipping_cost) FROM `Order`;
+---------------------+
| SUM(shipping_cost) |
+---------------------+
|                  85 |
+---------------------+
1 row in set (0.01 sec)

MariaDB [atiwari4]> SELECT SUM(amount) FROM `Pays`;
+-------------+
| SUM(amount) |
+-------------+
|      116068 |
+-------------+
1 row in set (0.00 sec)
```

Calculate the total current number of distributors.

```
MariaDB [atiwari4]> SELECT COUNT(*) AS Number_of_Distributors  FROM Distributor;
+-------------------------+
| Number_of_Distributors |
+-------------------------+
|                       4 |
+-------------------------+
1 row in set (0.00 sec)
```

Calculate total revenue (since inception) per city, per distributor, and per location.

```
MariaDB [atiwari4]> SELECT SUM(price*no_of_copies)
    -> FROM `Order`
    -> JOIN `Distributor` ON Order.distributor_id=Distributor.distributor_id
    -> GROUP BY city;
+-----------------------+
| SUM(price*no_of_copies) |
+-----------------------+
|                  7200 |
|                  1500 |
|                  3745 |
+-----------------------+
3 rows in set (0.00 sec)

MariaDB [atiwari4]>
MariaDB [atiwari4]> SELECT distributor_id, SUM(price*no_of_copies) AS Revenue  FROM `Order` GROUP BY distributor_id;
+----------------+---------+
| distributor_id | Revenue |
+----------------+---------+
|              1 |    1150 |
|              2 |    2550 |
|              3 |    1500 |
|              4 |    7200 |
+----------------+---------+
4 rows in set (0.00 sec)
```

Calculate total payments to the editors and authors, per time period and per work type (book authorship, article authorship, or editorial work)

```
MariaDB [atiwari4]> SELECT SUM(amount) AS Total_Amount FROM Pays WHERE payment_date BETWEEN '2019-12-30' and '2020-03-04';
+--------------+
| Total_Amount |
+--------------+
|        67200 |
+--------------+
1 row in set (0.00 sec)
```

```
MariaDB [atiwari4]> SELECT SUM(amount), Publication.type, Contributor.designation
    -> FROM `Pays`
    -> JOIN `Publication` ON Pays.publication_id = Publication.publication_id
    -> JOIN `Contributor` ON Pays.contributor_id = Contributor.contributor_id
    -> where Publication.type IN (SELECT DISTINCT(type) FROM Publication)
    ->  AND Contributor.designation IN (SELECT DISTINCT(designation) FROM Distributor)
    -> GROUP BY Publication.type, Contributor.designation;
+-------------+----------+-------------+
| SUM(amount) | type     | designation |
+-------------+----------+-------------+
|       32000 | Book     | Author      |
|       66000 | Book     | Editor      |
|        6969 | Book     | Freelance   |
|        1200 | Journal  | Freelance   |
|        6969 | Magazine | Editor      |
|        2922 | Magazine | Freelance   |
+-------------+----------+-------------+
6 rows in set (0.01 sec)
```

4.2

1) Keep track of when each payment was claimed by its addressee.

```
MariaDB [atiwari4]> select * from Pays where contributor_id = 2;
+----------------+----------------+--------+--------------+
| contributor_id | publication_id | amount | payment_date |
+----------------+----------------+--------+--------------+
|              2 |              3 |   1200 | 2020-01-05   |
+----------------+----------------+--------+--------------+
1 row in set (0.00 sec)
```

```
MariaDB [atiwari4]> explain select * from Pays where contributor_id = 2;
+------+-------------+-------+------+---------------+---------+---------+-------+------+-------+
| id   | select_type | table | type | possible_keys | key     | key_len | ref   | rows | Extra |
+------+-------------+-------+------+---------------+---------+---------+-------+------+-------+
|    1 | SIMPLE      | Pays  | ref  | PRIMARY       | PRIMARY | 4       | const |    1 |       |
+------+-------------+-------+------+---------------+---------+---------+-------+------+-------+
1 row in set (0.00 sec)
```

```
MariaDB [atiwari4]> create index `contributorIndex` on Pays(contributor_id);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [atiwari4]> explain select * from Pays where contributor_id = 2;
+------+-------------+-------+------+-------------------------+---------+---------+-------+------+-------+
| id   | select_type | table | type | possible_keys           | key     | key_len | ref   | rows | Extra |
+------+-------------+-------+------+-------------------------+---------+---------+-------+------+-------+
|    1 | SIMPLE      | Pays  | ref  | PRIMARY,contributorIndex | PRIMARY | 4       | const |    1 |       |
+------+-------------+-------+------+-------------------------+---------+---------+-------+------+-------+
1 row in set (0.00 sec)
```

2) SELECT SUM(price*no_of_copies) AS TOTAL FROM `Order` WHERE shipping_cost=0;

```
MariaDB [atiwari4]> SELECT SUM(price*no_of_copies) AS TOTAL FROM `Order` WHERE shipping_cost=0;
+-------+
| TOTAL |
+-------+
|  8700 |
+-------+
1 row in set (0.00 sec)
```

```
MariaDB [atiwari4]> EXPLAIN SELECT SUM(price*no_of_copies) AS TOTAL FROM `Order` WHERE shipping_cost=0;
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | Order | ALL  | NULL          | NULL | NULL    | NULL |    7 | Using where |
+------+-------------+-------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

```
MariaDB [atiwari4]> create index `shippingCostIndex` on `Order`(shipping_cost);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
MariaDB [atiwari4]> EXPLAIN SELECT SUM(price*no_of_copies) AS TOTAL FROM `Order` WHERE shipping_cost=0;
+------+-------------+-------+------+-------------------+-------------------+---------+-------+------+-------+
| id   | select_type | table | type | possible_keys     | key               | key_len | ref   | rows | Extra |
+------+-------------+-------+------+-------------------+-------------------+---------+-------+------+-------+
|    1 | SIMPLE      | Order | ref  | shippingCostIndex | shippingCostIndex | 4       | const |    2 |       |
+------+-------------+-------+------+-------------------+-------------------+---------+-------+------+-------+
1 row in set (0.00 sec)
```

4.3

1) Calculate total revenue (since inception) per city:

```
SELECT SUM(price * no_of_copies)
FROM `Order`
JOIN `Distributor`
ON Order.distributor_id = Distributor.distribution_id
GROUP BY city;
```

$$\pi_{SUM(price * no\_of\_copies)}(\gamma_{city,\ SUM(price*no\_of\_copies)}\ (\text{Order} \bowtie_{Order.distributor\_id = Distributor.distributor\_id} \text{Distributor}))$$

Suppose o is any tuple in the Order relation and d is any tuple in the Distributor relation, then on applying o.distributor_id = d.distributor_id we get all the information of the orders made by the distributors. Then on grouping the distributors by city we calculate the total revenue generated by the publication house from each city by multiplying the price and no_of_copies which then gives us the required information on running the query.

2) Find books and articles by author name:

```
SELECT B.isbn AS 'Book/Article', name
FROM Book B JOIN bookAuthor BA
ON B.isbn = BA.isbn
JOIN Contributor C
ON C.contributor_id = BA.author_id
WHERE name = 'JK Rowling'
UNION
SELECT D.article_id AS 'Book/Article', name
FROM Article D JOIN articleAuthor E
ON D.article_id = E.author_id
JOIN Contributor F
ON F.contributor_id = E.author_id
WHERE name = 'JK Rowling';
```

$$\pi_{Book.isbn\ as\ 'Book/Article',\ name}\ (\sigma_{name='JK\ Rowling'}\ ((\text{Book} \bowtie_{Book.isbn\ =\ bookAuthor.isbn} \text{bookAuthor}) \bowtie_{bookAuthor.author\_id\ =\ Contributor.contributor\_id} \text{Contributor})) \cup$$
$$\pi_{Article.article\_id\ as\ 'Book/Article',\ name}\ (\sigma_{name='JK\ Rowling'}\ ((\text{Article} \bowtie_{Article.article\_id\ =\ articleAuthor.article\_id} \text{articleAuthor}) \bowtie_{articleAuthor.author\_id\ =\ Contributor.contributor\_id} \text{Contributor}))$$

This query consists of two parts - one between Book, bookAuthor and Contributor and other between Article, articleAuthor and Contributor and then taking Union of their outputs.

For the first part, suppose b is a tuple for Book, ba is a tuple for bookAuthor and c is a tuple for contributor then on applying the join b.isbn = ba.isbn and c.contributor_id = ba.author_id we get the tuples which give us books written by authors. Then on applying the where clause, we can get the answer for a particle author.

Similarly, we can design the solution for articles written by authors where d is a tuple for Article, e is a tuple for articleAuthor and f is a tuple for contributor then on applying the join d.article_id = e.author_id and f.contributor_id = e.author_id we will get the tuples of articles written by authors. Then again on applying the where clause, we can get the answer for a particle author.

Finally on doing the union operation we can get the combination of books and articles written by any author. So, this is exactly what the query was supposed to retrieve.

THE END