RDBMS Fundamentals Assignment

Assignment 1: Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form.

Business Scenario :

A university wants to create a database to manage student enrollment, courses, and faculty. Each student has a student ID, name, address, phone number, and email. Students can enroll in multiple courses, and each course has a course ID, title, description, credits, and a faculty member assigned to teach it. Each faculty member has a faculty ID, name, department, and office location.

ER Diagram :

STUDENT {

int StudentID PK

varchar Name

varchar Address

varchar Phone

varchar Email

}

COURSE {

int CourseID PK

varchar Title

varchar Description

```
    int Credits
    int FacultyID FK
}
FACULTY {
    int FacultyID PK
    varchar Name
    varchar Department
    varchar OfficeLocation
}
ENROLLMENT {
    int EnrollmentID PK
    int StudentID FK
    int CourseID FK
    date EnrollmentDate
}
STUDENT ||--o{ ENROLLMENT : enrolls_in
COURSE ||--o{ ENROLLMENT : has_enrollment
FACULTY }o--|| COURSE : teaches
```

- The ER diagram consists of four entities: STUDENT, COURSE, FACULTY, and ENROLLMENT.
- The STUDENT entity has attributes for StudentID (primary key), Name, Address, Phone, and Email.
- The COURSE entity has attributes for CourseID (primary key), Title, Description, Credits, and FacultyID (foreign key referencing the FACULTY entity).

- The FACULTY entity has attributes for FacultyID (primary key), Name, Department, and OfficeLocation.
- The ENROLLMENT entity serves as a junction table between STUDENT and COURSE, representing the many-to-many relationship between students and courses. It has attributes for EnrollmentID (primary key), StudentID (foreign key referencing the STUDENT entity), CourseID (foreign key referencing the COURSE entity), and EnrollmentDate.
- The relationships are:
    1. many-to-many relationship
    2. one-to-many relationship
- The diagram follows proper normalization up to the third normal form (3NF).

Assignment 2: Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and CHECK. Include primary and foreign keys to establish relationships between tables.

Here is a database schema design for a library system, including tables, fields, constraints, primary keys, and foreign keys:

1. Books table:

Fields:

- book_id (Primary key)
- title
- author
- genre
- publication_year

Constraints:

- title (NOT NULL)

- author (NOT NULL)
- publication_year (CHECK publication_year >= 0)

2. Members table:

Fields:

- member_id (Primary key)
- name
- email
- address

Constraints:

- name (NOT NULL)
- email (UNIQUE, NOT NULL)

3. Borrowings table:

Fields:

- borrowing_id (Primary key)
- book_id (Foreign key referencing Books)
- member_id (Foreign key referencing Members)
- borrow_date
- return_date

Constraints:

- borrow_date (NOT NULL)
- return_date (CHECK return_date >= borrow_date)

The Borrowings table establishes a many-to-many relationship between Books and Members, allowing multiple books to be borrowed by multiple members, and vice versa.

Assignment 3: Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.

- Atomicity: This property ensures that either all the operations within a transaction are completed successfully, or none of them are.
- Consistency: This property ensures that the database remains in a valid state before and after the transaction.
- Isolation: Isolation ensures that the execution of transactions concurrently does not result in interference between them.
- Durability: Durability guarantees that once a transaction is committed, its changes are permanent and will not be lost, even in the event of a system failure. The changes made by a committed transaction are stored permanently in the database.

Simulating a Transaction with Locking and Isolation Levels:

Let's simulate a simple transaction where a member borrows a book from the library. We'll use two tables: Books and Members.

#Begin transaction

START TRANSACTION;

#Selecting a book to borrow (Assume book_id = 1)

SELECT * FROM Books WHERE book_id = 1 FOR UPDATE;

#Updating the book to mark it as borrowed

UPDATE Books SET available = 0 WHERE book_id = 1;

#Inserting a new borrowing record

INSERT INTO Borrowings (book_id, member_id, borrow_date) VALUES (1, 1, NOW());

Commit the transaction

COMMIT;

The different isolation levels:

1.Read Uncommitted

SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

2.Read committed

SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;

3.Repeatable Read

SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;

4.Serializable

SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;

Assignment 4: Write SQL statements to CREATE a new database and tables that reflect the library schema you designed earlier. Use ALTER statements to modify the table structures and DROP statements to remove a redundant table.

```
# Creating Database
CREATE DATABASE LibraryDatabase;
USE LibraryDatabase;
# Creating Tables
CREATE TABLE authors (
    author_id INT PRIMARY KEY,
    author_name VARCHAR(50)
);
```

```
CREATE TABLE books (
    book_id INT PRIMARY KEY,
    title VARCHAR(100),
    author_id INT,
    FOREIGN KEY (author_id) REFERENCES authors(author_id)
);
CREATE TABLE members (
    member_id INT PRIMARY KEY,
    member_name VARCHAR(50),
    member_type VARCHAR(20)
);
CREATE TABLE loans (
    loan_id INT PRIMARY KEY,
    book_id INT,
    member_id INT,
    loan_date DATE,
    return_date DATE,
    FOREIGN KEY (book_id) REFERENCES books(book_id),
    FOREIGN KEY (member_id) REFERENCES members(member_id)
);
# Modifying Table
ALTER TABLE books
ADD COLUMN publication_year INT;
# Droping Table
```

DROP TABLE old_table;

Assignment 5: Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyze the impact on query execution.

Assume we have a large number of books in the table and frequently search for books by their title.

- Creating an Index :

  To create an index on the title column of the books table, we use this statement

  CREATE INDEX idx_books_title

  ON books (title);

- Analyzing Query Performance with an Index

  EXPLAIN SELECT * FROM books WHERE title = 'The Great Gatsby';

- Removing the Index

  To remove the index, you can use the DROP INDEX statement

  DROP INDEX idx_books_title ON books;

Assignment 6: Create a new database user with specific privileges using the CREATE USER and GRANT commands. Then, write a script to REVOKE certain privileges and DROP the user.

Creating a user

CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'password';

```
# Granting privileges
GRANT SELECT, INSERT, UPDATE ON database_name.* TO 'new_user'@'localhost';
# Revoking specific privileges
REVOKE UPDATE ON database_name.* FROM 'new_user'@'localhost';
# Dropping the user
DROP USER 'new_user'@'localhost';
```

Assignment 7: Prepare a series of SQL statements to INSERT new records into the library tables, UPDATE existing records with new information, and DELETE records based on specific criteria. Include BULK INSERT operations to load data from an external source.

1. INSERT New Records:

```
INSERT INTO authors (author_id, author_name)
VALUES (4, 'J.K. Rowling');

INSERT INTO books (book_id, title, author_id, publication_year)
VALUES (10, 'Harry Potter and the Philosopher\'s Stone', 4, 1997);

INSERT INTO members (member_id, member_name, member_type)
VALUES (5, 'John Doe', 'Student');
```

```
INSERT INTO loans (loan_id, book_id, member_id, loan_date, return_date)
VALUES (5, 10, 5, '2022-01-01', '2022-01-31');
```

2. UPDATE Existing Records:

```
UPDATE books
SET publication_year = 1998
WHERE book_id = 10;

UPDATE members
SET member_type = 'Faculty'
WHERE member_id = 5;
```

3. DELETE Records:

```
DELETE FROM books
WHERE book_id = 10;

DELETE FROM members
WHERE member_id = 5;
```

4.BULK INSERT Operations:

```
BULK INSERT books
FROM 'C:\Path\To\books.csv'
WITH (
```

```
    FORMATFILE = 'C:\Path\To\books.xml',
    FIRSTROW = 2,
    IGNOREBLANKROWS = 1
);
```