"Assignment 1: Analyse 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.

Entities:

Student Entity:

Attributes:

- StudentID **(Primary Key):** A unique identifier for each student.
- Name: The full name of the student.
- Email: The email address of the student.
- Phone: The contact number of the student.

Note: The combination of these attributes ensures that each student record is unique and contains all necessary information for communication and identification.

Instructor Entity:

Attributes:

- InstructorID **(Primary Key):** A unique identifier for each instructor.
- Name: The full name of the instructor.
- Email: The email address of the instructor.
- Phone: The contact number of the instructor.

Note: Similar to the Student entity, these attributes ensure unique identification and communication capabilities.

Course Entity:

Attributes:

- CourseID **(Primary Key):** A unique identifier for each course.
- Title: The name of the course.
- Credits: The number of credits awarded for the course.

Note: This entity provides a catalog of courses with essential academic details.

Section Entity:

Attributes:

- SectionID **(Primary Key):**: A unique identifier for each section.
- CourseID **(Foreign key)**: Foreign key linking to the Course entity.
- InstructorID **(Foreign key)**:: Foreign key linking to the Instructor entity.
- Schedule: The timetable for the section.
- Capacity: The maximum number of students allowed in the section.

Note: This entity links courses with instructors and schedules, organizing the delivery of education.

Description of Relationships

Registration Relationship:

A student can register for multiple sections of courses. A section can have multiple students registered.

Attributes:

- StudentID: Foreign key linking to the Student entity.
- SectionID: Foreign key linking to the Section entity.
- Cardinality: Many-to-Many

Note: This relationship allows students to enroll in multiple sections and ensures that each section can accommodate multiple students.

Teaching Relationship:

An instructor can teach multiple sections of courses. A section is taught by one instructor.

Attributes:

- InstructorID: Foreign key linking to the Instructor entity.
- SectionID: Foreign key linking to the Section entity.
- Cardinality: One-to-Many

Note: This relationship ensures that each section is taught by one instructor, but an instructor can teach multiple sections.

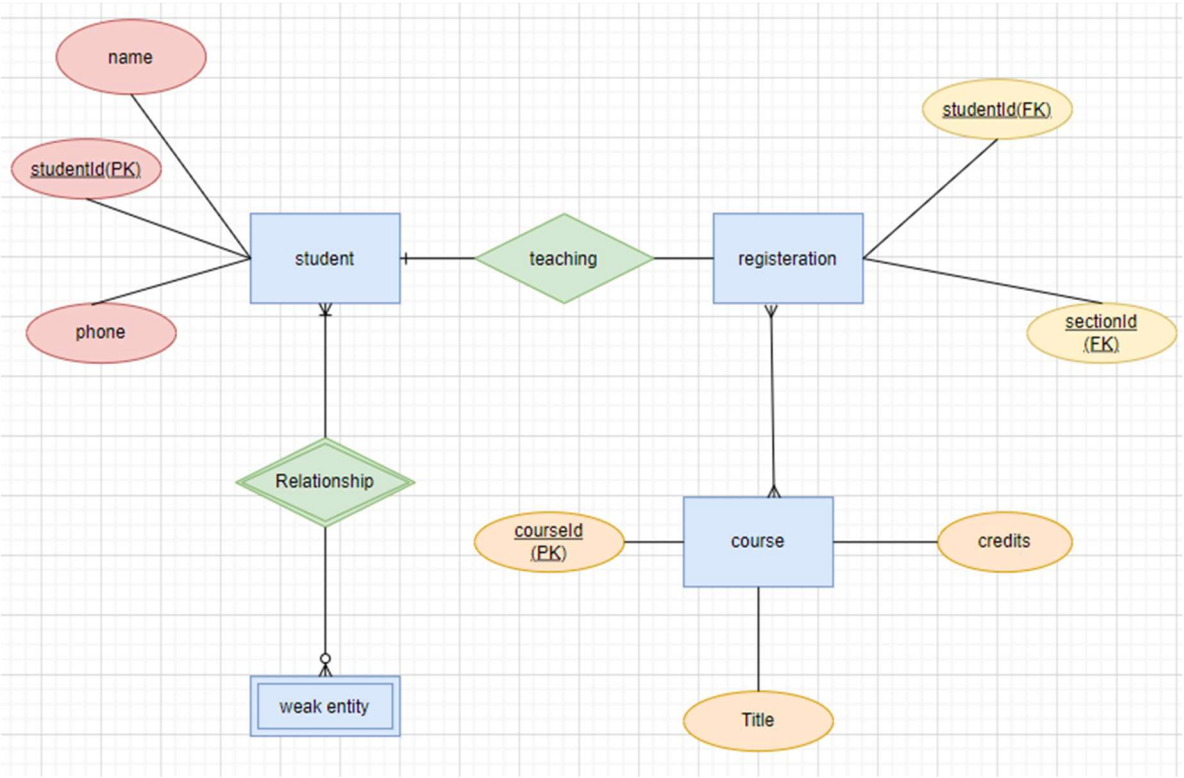
Student - Registration - Section

- One student can register for multiple sections.
- One section can have multiple students.
- Many-to-Many relationship.

Instructor - Teaching - Section

- One instructor can teach multiple sections.
- One section is taught by one instructor.
- One-to-Many relationship.

ER Diagram



First Normal Form (1NF):

Ensures that the table structure is flat, with no repeating groups or arrays.

Each field contains only atomic (indivisible) values.

Example: In the Student entity, the Phone attribute contains only a single phone number, not a list of numbers.

Second Normal Form (2NF):

Ensures that all non-key attributes are fully functionally dependent on the primary key.

Removes partial dependencies.

Example: In the Section entity, attributes like Schedule and Capacity depend entirely on SectionID, not partially on CourseID or InstructorID.

Third Normal Form (3NF):

Ensures there are no transitive dependencies.

Each non-key attribute is dependent only on the primary key.

Example: In the Course entity, Credits depend only on CourseID, not on Title.

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.

```
CREATE TABLE Authors (
 AuthorID INT PRIMARY KEY,
 Name VARCHAR(100) NOT NULL,
 Nationality VARCHAR(50)
);

CREATE TABLE Books (
 BookID INT PRIMARY KEY,
 Title VARCHAR(255) NOT NULL,
 AuthorID INT,
 ISBN VARCHAR(20) UNIQUE,
 Genre VARCHAR(50),
 PublishedYear INT,
 FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
);

CREATE TABLE Borrowers (
 BorrowerID INT PRIMARY KEY,
 Name VARCHAR(100) NOT NULL,
 Email VARCHAR(100) NOT NULL,
 Phone VARCHAR(15),
 UNIQUE (Email)
);

CREATE TABLE Transactions (
 TransactionID INT PRIMARY KEY,
 BookID INT,
 BorrowerID INT,
 TransactionDate DATE NOT NULL,
 DueDate DATE,
 ReturnDate DATE,
 FOREIGN KEY (BookID) REFERENCES Books(BookID),
 FOREIGN KEY (BorrowerID) REFERENCES Borrowers(BorrowerID),
```

```
 CHECK (ReturnDate >= TransactionDate)
);
```

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.

ACID Properties of a Transaction:

Atomicity:

Atomicity ensures that a transaction is treated as a single unit of work. It means that either all the operations within the transaction are successfully completed, or none of them are.

Consistency:

Consistency ensures that the database remains in a consistent state before and after the transaction. It means that the data must meet all integrity constraints, including referential integrity and domain constraints.

Isolation:

Isolation ensures that concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially. It means that transactions are executed independently of each other, and the intermediate state of one transaction is not visible to other transactions until it is committed.

Durability:

Durability ensures that once a transaction is committed, its changes are permanent and survive system failures. It means that the changes made by the transaction are persistent and will not be lost even in the event of a system crash or restart.

SQL Statements for Transaction with Locking and Isolation Levels:

```
BEGIN TRANSACTION;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT * FROM Accounts WHERE AccountID IN (123, 456) FOR UPDATE;
UPDATE Accounts SET Balance = Balance - 100 WHERE AccountID = 123;
UPDATE Accounts SET Balance = Balance + 100 WHERE AccountID = 456;
COMMIT;
```

ACID Properties:

Nested Transactions: Atomicity also applies to nested transactions. A nested transaction is a transaction that takes place within the context of another transaction. If an inner transaction fails, the outer transaction can still succeed if it can compensate for the failure.
Recovery Manager: Durability is often ensured by a recovery manager. The recovery manager can use techniques such as write-ahead logging and shadow paging to ensure that committed changes survive system failures.
Concurrency Control Mechanisms: Isolation is typically achieved through concurrency control mechanisms such as locks and timestamps. These mechanisms control the order in which transactions are executed to prevent inconsistencies.
SQL Transactions:
Locking: The FOR UPDATE clause in your SQL statement is a form of locking. It locks the selected rows to prevent other transactions from modifying them until your transaction commits.
Savepoints: You can use savepoints in a transaction to define points to which you can roll back without aborting the entire transaction. This can be useful in large transactions where you want to be able to recover from errors without starting over.
Rollback: If something goes wrong during a transaction, you can use the ROLLBACK statement to undo all the changes made in the transaction.
Deadlocks: Deadlocks are a condition where two or more transactions indefinitely wait for each other to release resources. SQL Server has a deadlock detector that periodically checks for lock conditions and terminates one of the transactions to break the deadlock.

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 a New Database and Tables:

```
CREATE DATABASE LibraryDB;

USE LibraryDB;

CREATE TABLE Authors (
 AuthorID INT PRIMARY KEY,
 Name VARCHAR(100) NOT NULL,
 Nationality VARCHAR(50));
```

```
CREATE TABLE Books (
 BookID INT PRIMARY KEY,
 Title VARCHAR(255) NOT NULL,
 AuthorID INT,
 ISBN VARCHAR(20) UNIQUE,
 Genre VARCHAR(50),
 PublishedYear INT,
 FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID));

CREATE TABLE Borrowers (
 BorrowerID INT PRIMARY KEY,
 Name VARCHAR(100) NOT NULL,
 Email VARCHAR(100) NOT NULL,
 Phone VARCHAR(15),
 UNIQUE (Email));

CREATE TABLE Transactions (
 TransactionID INT PRIMARY KEY,
 BookID INT,
 BorrowerID INT,
 TransactionDate DATE NOT NULL,
 DueDate DATE,
 ReturnDate DATE,
 FOREIGN KEY (BookID) REFERENCES Books(BookID),
 FOREIGN KEY (BorrowerID) REFERENCES Borrowers(BorrowerID),
 CHECK (ReturnDate >= TransactionDate)
);
```

Modifying Table Structures with ALTER Statements:

```
ALTER TABLE Books ADD Description TEXT;
```

Dropping a Redundant Table:

```
DROP TABLE Borrowers;
```

Explanation:

1. CREATE DATABASE LibraryDB; - This statement creates a new database named LibraryDB.
2. USE LibraryDB; - This statement selects LibraryDB as the database for subsequent SQL statements.
3. The CREATE TABLE statements define four tables (Authors, Books, Borrowers, Transactions) with various columns. Each table has a primary key, and the Books and Transactions tables have foreign keys that reference the primary keys of other tables.
4. ALTER TABLE Books ADD Description TEXT; - This statement adds a new column named Description of type TEXT to the Books table.
5. DROP TABLE Borrowers; - This statement removes the Borrowers table from the database.

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 analyse the impact on query execution.

Creating an Index:

Suppose we want to create an index on the Title column of the Books table:

CREATE INDEX idx_title ON Books (Title);

How it Improves Query Performance:

Creating an index on the Title column will speed up queries that involve searching, sorting, or filtering based on the book title. When a query is executed that involves the Title column, the database engine can use the index to quickly locate the relevant rows without having to scan the entire table sequentially. This improves query performance by reducing the time required to retrieve data, especially for large datasets.

Dropping the Index:

Drop the index on the Title column of the Books table:

DROP INDEX idx_title ON Books;

Impact on Query Execution:

Without the index, queries that involve searching, sorting, or filtering based on the book title may experience slower performance compared to when the index was present. The database engine will need to perform a full table scan to locate the relevant rows, which can

be time-consuming, especially for large tables. Queries that heavily rely on the Title column may suffer from degraded performance after dropping the index, as the database engine lacks the optimized access path provided by the index.

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.

Create a new database user

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

Grant specific privileges to the user on a specific database

GRANT SELECT, INSERT, UPDATE ON librarydb.* TO 'new_user'@'localhost';

Revoke certain privileges from the user

REVOKE INSERT ON librarydb.* FROM 'new_user'@'localhost';

Drop 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."

Inserting new records

INSERT INTO Authors (AuthorID, Name, Nationality)

VALUES (1, 'Sreenath', 'India');

INSERT INTO Books (BookID, Title, AuthorID, ISBN, Genre, PublishedYear)

VALUES (1, 'Harry Potter and the Philosopher"s Stone', 1, '978-3-16-148410-0', 'Fantasy', 1997);

INSERT INTO Borrowers (BorrowerID, Name, Email, Phone)

VALUES (1, 'Sreenath', 'sreenath1999@gmail.com', '9874561230');

INSERT INTO Transactions (TransactionID, BookID, BorrowerID, TransactionDate, DueDate, ReturnDate)

VALUES (1, 1, 1, '2024-05-21', '2024-06-21', NULL);

Updating existing records

```
UPDATE Books
SET Title = 'Harry Potter and the Sorcerer"s Stone'
WHERE BookID = 1;
```

UPDATE Borrowers

```
SET Email = 'sreenath@gamil.com'
WHERE BorrowerID = 1;
```

Deleting records based on specific criteria:

```
DELETE FROM Books
WHERE BookID = 1;
DELETE FROM Transactions
WHERE TransactionDate < '2024-01-01';
```

Bulk insert operations

```
LOAD DATA INFILE 'path/to/your/file.csv'
INTO TABLE Books
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

'path/to/your/file.csv' with the actual path to your CSV file. The FIELDS TERMINATED BY ',' and LINES TERMINATED BY '\n' specify that the fields in your CSV file are separated by commas and each line is a new record. The IGNORE 1 ROWS is used to skip the header row of the CSV file.