## Theory

## One-to-one (1:1)

A relationship is one-to-one **if and only if one record from table A is related to a maximum of one record in table B.**

To establish a one-to-one relationship, the primary key of table B (with no orphan record) must be the secondary key of table A (with orphan records).

**For example:**

CREATE TABLE User (

    UserID INT PRIMARY KEY,

    UserName VARCHAR(50),

    Email VARCHAR(50)

);

CREATE TABLE UserProfile (

    ProfileID INT PRIMARY KEY,

    UserID INT UNIQUE,

    Bio VARCHAR(255),

    FOREIGN KEY (UserID) REFERENCES User(UserID)

);

*-- Insert data*

INSERT INTO User (UserID, UserName, Email)

VALUES (1, 'Alice', 'alice@example.com'),

       (2, 'Bob', 'bob@example.com'),

       (3, 'Charlie', 'charlie@example.com');

INSERT INTO UserProfile (ProfileID, UserID, Bio)

VALUES (101, 1, 'Loves programming'),

       (102, 2, 'Data scientist'),

       (103, 3, 'AI enthusiast');

## One-to-many (1:M)

A relationship is one-to-many **if and only if one record from table A is related to one or more records in table B**. However, **one record in table B cannot be related to more than one record in table A**.

To establish a one-to-many relationship, the primary key of table A (the "one" table) must be the secondary key of table B (the "many" table).

**For example:**

CREATE TABLE Vendor(

    VendorNumber number(4) PRIMARY KEY,

    Name varchar2(20),

    Address varchar2(20),

    City varchar2(15),

    Street varchar2(2),

    ZipCode varchar2(10),

    Contact varchar2(16),

    PhoneNumber varchar2(12),

    Status varchar2(8),

    StampDate date

);

CREATE TABLE Inventory(

    Item varchar2(6) PRIMARY KEY,

    Description varchar2(30),

    CurrentQuantity number(4) NOT NULL,

    VendorNumber number(2) REFERENCES Vendor(VendorNumber),

    ReorderQuantity number(3) NOT NULL

);

## Many-to-many (M:M)

A relationship is many-to-many if and only if one record from table A is related to one or more records in table B and vice-versa.

To establish a many-to-many relationship, create a third table called "ClassStudentRelation" which will have the primary keys of both table A and table B.

**For example:**

CREATE TABLE Class(

    ClassID varchar2(10) PRIMARY KEY,

    Title varchar2(30),

    Instructor varchar2(30),

    Day varchar2(15),

    Time varchar2(10)

);

CREATE TABLE Student(

    StudentID varchar2(15) PRIMARY KEY,

    Name varchar2(35),

    Major varchar2(35),

    ClassYear varchar2(10),

    Status varchar2(10)

);

CREATE TABLE ClassStudentRelation(

    StudentID varchar2(15) NOT NULL,

    ClassID varchar2(14) NOT NULL,

    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),

    FOREIGN KEY (ClassID) REFERENCES Class(ClassID),

    UNIQUE (StudentID, ClassID)

);

1. **Practice**

### ****One-to-One Relationship Exercises****

#### ****Exercise 1: Create and Populate Tables****

* Create two tables: User and UserProfile where:
  1. Each user has exactly one profile.
  2. UserProfile must reference the User table.
* Populate the tables with at least 3 users and their profiles.

#### ****Exercise 2: Retrieve Data****

* Write a query to:
  1. Retrieve all users along with their profile details.
  2. Find users who don’t have a profile (orphan records in the User table).

#### ****Exercise 3: Update and Delete****

* Update the profile information of a user.
* Delete a user and ensure their profile is also deleted automatically (use ON DELETE CASCADE).

### ****One-to-Many Relationship Exercises****

#### ****Exercise 4: Create a One-to-Many Relationship****

* Create two tables: Author and Book where:
  1. Each author can write multiple books.
  2. Each book is written by only one author.
* Populate the tables with at least 3 authors and 5 books.

#### ****Exercise 5: Retrieve Data****

* Write a query to:
  1. List all authors and the titles of their books.
  2. Find authors who haven’t written any books.
  3. Count the number of books written by each author.

#### ****Exercise 6: Update and Delete****

* Update the author of a specific book.
* Delete an author and ensure their books are not deleted (enforce SET NULL or NO ACTION).

### ****Many-to-Many Relationship Exercises****

#### ****Exercise 7: Create a Many-to-Many Relationship****

* Create three tables: Student, Course, and Enrollment where:
  1. A student can enroll in multiple courses.
  2. A course can have multiple students enrolled.
  3. Use a junction table (Enrollment) to represent the many-to-many relationship.

#### ****Exercise 8: Populate Data****

* Populate the tables with:
  + 4 students.
  + 3 courses.
  + At least 6 enrollments (e.g., multiple students enrolled in the same course).

#### ****Exercise 9: Retrieve Data****

* Write queries to:
  1. Find all students enrolled in a specific course.
  2. List all courses a specific student is enrolled in.
  3. Identify courses with no students enrolled.

#### ****Exercise 10: Advanced Query****

* Find the total number of students enrolled in each course.
* List students enrolled in multiple courses.

### ****Bonus (Combined Exercises)****

#### ****Exercise 11: Real-Life Scenario****

* Design tables for a system where:
  + Each person has one passport (**one-to-one**).
  + Each passport can have multiple travel visas (**one-to-many**).
  + A person can travel to multiple countries and a country can host multiple visitors (**many-to-many**).

#### ****Exercise 12: Data Integrity****

* Modify the Enrollment table from the many-to-many relationship to:
  + Include additional columns like EnrollmentDate and Grade.
  + Prevent duplicate enrollments (i.e., the same student enrolling in the same course twice).

1. **Answers**

### ****One-to-One Relationship Solutions****

#### ****Exercise 1: Create and Populate Tables****

CREATE TABLE User (

    UserID INT PRIMARY KEY,

    UserName VARCHAR(50),

    Email VARCHAR(50)

);

CREATE TABLE UserProfile (

    ProfileID INT PRIMARY KEY,

    UserID INT UNIQUE,

    Bio VARCHAR(255),

    FOREIGN KEY (UserID) REFERENCES User(UserID)

);

*-- Insert data*

INSERT INTO User (UserID, UserName, Email)

VALUES (1, 'Alice', 'alice@example.com'),

       (2, 'Bob', 'bob@example.com'),

       (3, 'Charlie', 'charlie@example.com');

INSERT INTO UserProfile (ProfileID, UserID, Bio)

VALUES (101, 1, 'Loves programming'),

       (102, 2, 'Data scientist'),

       (103, 3, 'AI enthusiast');

#### ****Exercise 2: Retrieve Data****

1. **All users and their profiles**:

SELECT u.UserName, u.Email, p.Bio

FROM User u

LEFT JOIN UserProfile p ON u.UserID = p.UserID;

1. **Users without profiles**:

SELECT UserName, Email

FROM User

WHERE UserID NOT IN (SELECT UserID FROM UserProfile);

#### ****Exercise 3: Update and Delete****

1. **Update a profile**:

UPDATE UserProfile

SET Bio = 'Loves nature photography'

WHERE UserID = 1;

1. **Delete a user with** ON DELETE CASCADE: Add the ON DELETE CASCADE clause to the foreign key:

ALTER TABLE UserProfile

ADD CONSTRAINT fk\_UserProfile\_User FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE;

Then delete a user:

DELETE FROM User WHERE UserID = 1;