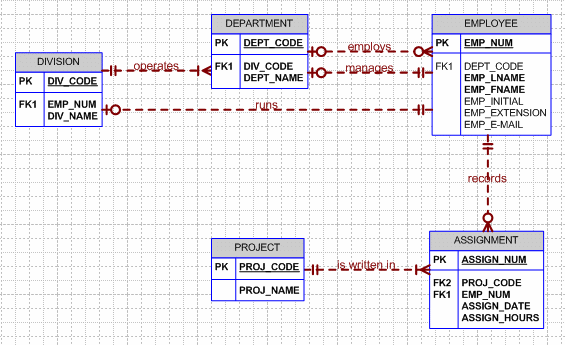
* **Use the following business rules to create a ERD. Write all appropriate connectivities and cardinalities in the ERD.**
  + 1. **A department employs many employees, but each employee is employed by one department.**
    2. **Some employees, known as “rovers,” are not assigned to any department.**
    3. **A division operates many departments, but each department is operated by one division.**
    4. **An employee may be assigned many projects, and a project may have many employees assigned to it.**
    5. **A project must have at least one employee assigned to it.**
    6. **One of the employees manages each department, and each department is managed by only one employee.**
    7. **One of the employees runs each division, and each division is run by only one employee.**

**ERD**



1. A department employs many employees, but each employee is employed by one department.

One-to-many (1:M) relationship between departments and employees.

1. Some employees, known as “rovers,” are not assigned to any department.

Many-to-zero (M:0) relationship between employees and department.

1. A division operates many departments, but each department is operated by one division.

One-to-many (1:M) relationship between divisions and departments.

1. An employee may be assigned many projects, and a project may have many employees assigned to it.

Many-to-many (M:N) relationship between employees and projects.

1. A project must have at least one employee assigned to it.

One-to-many (1:M) relationship between projects and employees.

1. One of the employees manages each department, and each department is managed by only one employee.

One-to-one (1:1) relationship between employees and departments for management.

1. One of the employees runs each division, and each division is run by only one employee.

One-to-one (1:1) relationship between employees and divisions for management.

Table: Department

| **department\_id (PK)** | **department\_name** | **division\_id (FK)** |
| --- | --- | --- |
| 1 | HR | 1 |
| 2 | Finance | 1 |
| ... | ... | ... |

CREATE TABLE Department (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255),

division\_id INT,

FOREIGN KEY (division\_id) REFERENCES Division(division\_id)

);

Table: Employee

| **employee\_id (PK)** | **employee\_name** | **department\_id (FK)** | **is\_rover** |
| --- | --- | --- | --- |
| 1 | John | 1 | false |
| 2 | Alice | 2 | false |
| 3 | Bob | NULL | true |
| ... | ... | ... | ... |

CREATE TABLE Employee (

employee\_id INT PRIMARY KEY,

employee\_name VARCHAR(255),

department\_id INT,

is\_rover BIT, -- Using BIT data type for boolean values

FOREIGN KEY (department\_id) REFERENCES Department(department\_id)

);

Table: Division

| **division\_id (PK)** | **division\_name** | **division\_head\_id (FK)** |
| --- | --- | --- |
| 1 | Operations | 1 |
| 2 | Sales | 2 |
| ... | ... | ... |

CREATE TABLE Division (

division\_id INT PRIMARY KEY,

division\_name VARCHAR(255),

division\_head\_id INT,

FOREIGN KEY (division\_head\_id) REFERENCES Employee(employee\_id)

);

Table: Project

| **project\_id (PK)** | **project\_name** |
| --- | --- |
| 1 | Project A |
| 2 | Project B |
| ... | ... |

CREATE TABLE Project (

project\_id INT PRIMARY KEY,

project\_name VARCHAR(255)

);

Table: Employee\_Project

| **employee\_id (FK)** | **project\_id (FK)** |
| --- | --- |
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |
| 3 | 2 |
| ... | ... |

CREATE TABLE Employee\_Project (

employee\_id INT,

project\_id INT,

PRIMARY KEY (employee\_id, project\_id),

FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id),

FOREIGN KEY (project\_id) REFERENCES Project(project\_id)

);

The create table queries are failing due to foreign key constraints, it's likely because the tables are being created in an incorrect order, where referenced tables are being created after the tables referencing them. To resolve this issue, you can follow these steps:

1. Create the tables without foreign key constraints.
2. Add the foreign key constraints after all tables have been created.

Here's how you can do it:

-- Step 1: Create tables without foreign key constraints

-- Department table

CREATE TABLE Department (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255),

division\_id INT

);

-- Employee table

CREATE TABLE Employee (

employee\_id INT PRIMARY KEY,

employee\_name VARCHAR(255),

department\_id INT,

is\_rover BIT

);

-- Division table

CREATE TABLE Division (

division\_id INT PRIMARY KEY,

division\_name VARCHAR(255),

division\_head\_id INT

);

-- Project table

CREATE TABLE Project (

project\_id INT PRIMARY KEY,

project\_name VARCHAR(255)

);

-- Employee\_Project table

CREATE TABLE Employee\_Project (

employee\_id INT,

project\_id INT,

PRIMARY KEY (employee\_id, project\_id)

);

-- Step 2: Add foreign key constraints

-- Add foreign key constraint for department\_id in Employee table

ALTER TABLE Employee

ADD CONSTRAINT FK\_Employee\_Department FOREIGN KEY (department\_id) REFERENCES Department(department\_id);

-- Add foreign key constraint for division\_id in Department table

ALTER TABLE Department

ADD CONSTRAINT FK\_Department\_Division FOREIGN KEY (division\_id) REFERENCES Division(division\_id);

-- Add foreign key constraint for division\_head\_id in Division table

ALTER TABLE Division

ADD CONSTRAINT FK\_Division\_Employee FOREIGN KEY (division\_head\_id) REFERENCES Employee(employee\_id);

-- Add foreign key constraints for employee\_id and project\_id in Employee\_Project table

ALTER TABLE Employee\_Project

ADD CONSTRAINT FK\_Employee\_Project\_Employee FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id);

ALTER TABLE Employee\_Project

ADD CONSTRAINT FK\_Employee\_Project\_Project FOREIGN KEY (project\_id) REFERENCES Project(project\_id);

This way, you first create all the tables without any foreign key constraints, and then you add the foreign key constraints afterward. This ensures that all referenced tables exist before the foreign key constraints are applied.

Next, try to add data to our tables:

-- Inserting data into Department table

INSERT INTO Department (department\_id, department\_name, division\_id)

VALUES

(1, 'HR', 1),

(2, 'Finance', 1),

(3, 'Marketing', 2),

(4, 'IT', 2),

(5, 'Operations', 1);

-- Inserting data into Employee table

INSERT INTO Employee (employee\_id, employee\_name, department\_id, is\_rover)

VALUES

(1, 'John', 1, 0),

(2, 'Alice', 2, 0),

(3, 'Bob', NULL, 1),

(4, 'Emma', 3, 0),

(5, 'Mike', 4, 0);

-- Inserting data into Division table

INSERT INTO Division (division\_id, division\_name, division\_head\_id)

VALUES

(1, 'Management', 1),

(2, 'Sales', 2),

(3, 'Marketing', 4),

(4, 'IT', 5),

(5, 'Operations', 1);

-- Inserting data into Project table

INSERT INTO Project (project\_id, project\_name)

VALUES

(1, 'Project A'),

(2, 'Project B'),

(3, 'Project C'),

(4, 'Project D'),

(5, 'Project E');

-- Inserting data into Employee\_Project table

INSERT INTO Employee\_Project (employee\_id, project\_id)

VALUES

(1, 1),

(1, 2),

(2, 1),

(3, 2),

(4, 3);

In this step, If the insert commands are failing due to foreign key constraints, it's likely because the data being inserted violates the referential integrity enforced by those constraints. Here are a few potential solutions:

1. Insert Data in Correct Order: Ensure that you insert data into tables in the correct order, starting with the parent tables (tables without foreign key references) and then moving to the child tables (tables with foreign key references). For example, insert data into the Division table first, followed by the Department table, and so on.
2. Check Referential Integrity: Make sure that the values being inserted into the foreign key columns in child tables exist in the corresponding parent tables. For example, before inserting data into the Employee table, ensure that the department\_id values being inserted exist in the Department table.
3. Use NULL Values Where Appropriate: If a foreign key column allows NULL values, you can insert NULL values to bypass the foreign key constraint temporarily. For example, if an Employee's department\_id is NULL because they are a "rover" not assigned to any department, you can insert such records with department\_id set to NULL.
4. Disable Foreign Key Constraints Temporarily: Temporarily disable foreign key constraints, insert the data, and then re-enable the constraints. However, this approach should be used with caution as it can lead to data integrity issues if not handled properly.

Here's an example of how you can temporarily disable and enable foreign key constraints in SQL Server:

First: disable foreign key constraints

ALTER TABLE Employee NOCHECK CONSTRAINT FK\_Employee\_Department;

Now, insert data into Employee table and then: enable foreign key constraints again

ALTER TABLE Employee CHECK CONSTRAINT FK\_Employee\_Department;

Alternatively, you can temporarily disable and enable foreign key constraints for all tables in SQL Server:

-- Disable all foreign key constraints

EXEC sp\_MSforeachtable 'ALTER TABLE ? NOCHECK CONSTRAINT ALL';

-- Insert data into tables

-- Enable all foreign key constraints

EXEC sp\_MSforeachtable 'ALTER TABLE ? WITH CHECK CHECK CONSTRAINT ALL';

These commands will disable all foreign key constraints, insert data into the tables, and then enable all foreign key constraints again. This approach should be used with caution, as it temporarily suspends referential integrity checks, which could potentially lead to data integrity issues if not handled carefully. Make sure to thoroughly test the integrity of your data after performing such operations.