Team3\_Creation.sql

The script creates Database and Schema “CompanyDB” and “Staff” if not exist

Creates a sorge procedure “CreateTables” that Create 3 tables if not exist

Table1: **Departments**

Schema: Staff

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| DepartmentID | INT | (Primary Key) Unique identifier for each department. |
| DepartmentName | NVARCHAR(100), NOT NULL | Name of the department. |
| Location | VARCHAR(100) | Location of the department. |

Table2: **Employees**

Schema: Staff

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| EmployeeID | INT | (Primary Key) Unique identifier for each employee. |
| FirstName | NVARCHAR(50), NOT NULL | First name of the employee. |
| LastName | NVARCHAR(50), NOT NULL | Last name of the employee. |
| DepartmentID | INT | (Foreign Key) Identifier for the department to which the employee belongs. |
| HireDate | DATE, NOT NULL | Date when the employee was hired. |
| Position | NVARCHAR(80) | Job title or position of the employee. |
| Salary | NUMERIC(12, 3) | Salary of the employee. |

Table3: **Projects**

Schema: Staff

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| ProjectID | INT | (Primary Key) Unique identifier for each Project. |
| ProjectName | NVARCHAR(100), NOT NULL | Name of the Project. |
| StartDate | DATE | Starting date of the project. |
| EndDate | DATE | Ending date of the project. |
| Budget | MONEY | Total cost available for the project |

Table4: **Assignments**

Schema: Staff

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| AssignmentID | INT | (Primary Key) Unique identifier for each Assignment. |
| EmployeeID | INT | (Foreign Key) Identifier for the employee to which the assignments belongs. |
| ProjectID | INT | (Foreign Key) Identifier for the project to which the assignments belongs. |
| Role | NVARCHAR(50) | Ending date of the project. |
| StartDate | DATE | Starting date of the assignments. |
| EndDate | DATE | Ending date of the assignments. |

Team3\_Constraints.sql

The script modifies the Employees and Assignments tables within the Staff schema.

A new foreign key constraint named employees\_fk is added to the DepartmentID column to establish a relationship with the DepartmentID column in the Departments table.

**Additional Constraints**:

* employees\_chks: ensures that Salary is greater than 0.
* employees\_chkd: ensures that HireDate is a past or current date (not a future date).

A new foreign key constraint named assignments\_fk1 is added to the EmployeeID column to establish a relationship with the EmployeeID column in the Employees table.

A new foreign key constraint named assignments\_fk2 is added to the ProjectID column to establish a relationship with the ProjectID column in the Projects table.

Team3\_Insertion.sql

The Team3\_Insertion.sql file creates a stored procedure that facilitates the migration of data from the SourceDB database to the Staff schema within the CompanyDB database. It ensures that the Employees, Departments, Projects, and Assignments tables are populated with the corresponding data from the source database.

This procedure is essential for synchronizing data between different databases, making sure that the CompanyDB has up-to-date and accurate information from SourceDB.

**File Content Breakdown**

1. **Database Context:**

USE CompanyDB;

GO

This command sets the context to the CompanyDB database, ensuring that subsequent commands are executed within this database.

1. **Stored Procedure: Staff.InsertDataFromAnotherDatabase**
   * **Procedure Definition:**

CREATE PROCEDURE Staff.InsertDataFromAnotherDatabase

AS

BEGIN

-- Assuming SourceDB is already linked and accessible

INSERT INTO Staff.Employees (EmployeeID, FirstName, LastName, DepartmentID)

SELECT EmployeeID, FirstName, LastName, DepartmentID

FROM SourceDB.dbo.Employees;

INSERT INTO Staff.Departments (DepartmentID, DepartmentName)

SELECT DepartmentID, DepartmentName

FROM SourceDB.dbo.Departments;

INSERT INTO Staff.Projects (ProjectID, ProjectName)

SELECT ProjectID, ProjectName

FROM SourceDB.dbo.Projects;

INSERT INTO Staff.Assignments (AssignmentID, EmployeeID, ProjectID, "Role",StartDate, EndDate)

SELECT AssignmentID, EmployeeID, ProjectID, "Role", StartDate, EndDate

FROM SourceDB.dbo.Assignments;

END;

GO

**Purpose:** This procedure inserts data from the SourceDB database into the Staff schema in the CompanyDB database. It handles data migration for the following tables:

* + - Employees
    - Departments
    - Projects
    - Assignments
  + **Assumptions:**
    - The SourceDB database is already linked and accessible from the CompanyDB database.
  + **Actions:**
    - **Employees Table:**

INSERT INTO Staff.Employees (EmployeeID, FirstName, LastName, DepartmentID)

SELECT EmployeeID, FirstName, LastName, DepartmentID

FROM SourceDB.dbo.Employees;

This inserts employee data from SourceDB.dbo.Employees into Staff.Employees.

* + - **Departments Table:**

INSERT INTO Staff.Departments (DepartmentID, DepartmentName)

SELECT DepartmentID, DepartmentName

FROM SourceDB.dbo.Departments;

This inserts department data from SourceDB.dbo.Departments into Staff.Departments.

* + - **Projects Table:**

INSERT INTO Staff.Projects (ProjectID, ProjectName)

SELECT ProjectID, ProjectName

FROM SourceDB.dbo.Projects;

This inserts project data from SourceDB.dbo.Projects into Staff.Projects.

* + - **Assignments Table:**

INSERT INTO Staff.Assignments (AssignmentID, EmployeeID, ProjectID, "Role", StartDate, EndDate)

SELECT AssignmentID, EmployeeID, ProjectID, "Role", StartDate, EndDate

FROM SourceDB.dbo.Assignments;

This inserts assignment data from SourceDB.dbo.Assignments into Staff.Assignments.

1. **Procedure Execution:**

EXEC Staff.InsertDataFromAnotherDatabase;

**CREATE QUERIES.SQL**

The script defines a stored procedure named PerformQueryTasks that encapsulates several SQL queries. This stored procedure is designed to execute a series of data retrieval tasks to gather comprehensive information about the company's staff and their assignments.

**Queries within the Stored Procedure**

1. **Retrieving Employees Table**

SELECT \* FROM staff.Employees;

This query retrieves all records from the Employees table.

1. **Retrieving Projects Table**

SELECT \* FROM staff.Projects;

This query retrieves all records from the Projects table.

1. **Retrieving Department Info**

SELECT \* FROM staff.Departments;

This query retrieves all records from the Departments table.

1. **Summary of Employees' Names and Positions**

SELECT

FirstName + ' ' + LastName AS full\_name,

DepartmentName

FROM staff.Employees e

JOIN staff.Departments d

ON e.DepartmentID = d.DepartmentID;

This query generates a summary of employees' names and their respective departments.

1. **Employees and Their Assigned Assignments**

SELECT

FirstName + ' ' + LastName AS full\_name,

StartDate,

EndDate

FROM staff.Employees e

JOIN staff.Assignments a

ON e.EmployeeID = a.EmployeeID

WHERE EndDate IS NOT NULL

ORDER BY StartDate DESC;

This query retrieves information about employees and their assignments, specifically focusing on completed assignments.

1. **Employees with Multiple Assignments**

SELECT

e.FirstName,

e.LastName,

COUNT(DISTINCT a.ProjectID) AS NumProjectsAssigned

FROM staff.Employees e

JOIN staff.Assignments a

ON e.EmployeeID = a.EmployeeID

GROUP BY e.EmployeeID, e.FirstName, e.LastName

HAVING COUNT(DISTINCT a.ProjectID) > 1;

This query identifies employees who have been assigned to multiple projects.

1. **Project Names and Count in Each Department**

SELECT

d.DepartmentName,

COUNT(DISTINCT p.ProjectID) AS NumProjects

FROM staff.Projects p

JOIN staff.Assignments a

ON p.ProjectID = a.ProjectID

JOIN staff.Employees e

ON a.EmployeeID = e.EmployeeID

JOIN staff.Departments d

ON e.DepartmentID = d.DepartmentID

GROUP BY d.DepartmentName;

This query provides the names of the projects in each department and the count of projects per department.

**Execution of the Stored Procedure**

The script executes the stored procedure with the following command:

EXEC PerformQueryTasks;

**CreateViews.SQL**

The script defines a stored procedure named CreateView. This procedure is responsible for creating two specific views: EmployeeCurrentProject and EmployeeDepartmentDetails.

**Views Created by the Stored Procedure**

1. **EmployeeCurrentProject**

CREATE VIEW EmployeeCurrentProject AS

SELECT

e.EmployeeID,

e.FirstName,

e.LastName,

p.ProjectName,

a.Role

FROM

staff.Employees e

JOIN staff.Assignments a

ON e.EmployeeID = a.EmployeeID

JOIN staff.Projects p

ON a.ProjectID = p.ProjectID

WHERE

a.EndDate IS NULL; -- Assuming NULL EndDate means currently assigned

This view provides a list of currently assigned employees along with their project details.

1. **EmployeeDepartmentDetails**

CREATE VIEW EmployeeDepartmentDetails AS

SELECT

e.EmployeeID,

e.FirstName,

e.LastName,

d.DepartmentName

FROM

staff.Employees e

JOIN

staff.Departments d

ON e.DepartmentID = d.DepartmentID;

This view provides details about employees and their respective departments.

**Execution of the Stored Procedure**

The script executes the stored procedure with the following command:

EXEC CreateView;

To retrieve data from the created views, the following queries are executed:

1. Retrieve data from EmployeeCurrentProject:

SELECT \* FROM EmployeeCurrentProject;

1. Retrieve data from EmployeeDepartmentDetails:

SELECT \* FROM EmployeeDepartmentDetails;

**RunAllSetupProcedures.SQL**

It’s designed to run all the procedure we have made in one procedure when we run it

**1. Checking for Procedure Existence**

IF OBJECT\_ID(N'Staff.RunAllSetupProcedures', N'P') IS NULL

This line checks if a stored procedure named Staff.RunAllSetupProcedures exists in the database. The OBJECT\_ID function is used for this purpose, where:

* N'Staff.RunAllSetupProcedures' specifies the name of the object.
* N'P' specifies the type of object, in this case, a stored procedure.

If the OBJECT\_ID function returns NULL, it indicates that the procedure does not exist.

**2. Creating the Stored Procedure**

BEGIN

EXEC('CREATE PROCEDURE Staff.RunAllSetupProcedures

AS

BEGIN

EXEC Staff.CreateTables;

EXEC Staff.CreateConstraintsAndRelationships;

EXEC Staff.InsertDataFromAnotherDatabase;

EXEC Staff.PerformQueryTasks;

EXEC Staff.CreateViews;

EXEC Staff.CreateSpecificViews;

END;');

END

If the procedure does not exist, the script proceeds to create it. The EXEC statement is used to execute a dynamically constructed SQL string, which defines the stored procedure Staff.RunAllSetupProcedures.