Department of Computer Science
University of Cyprus

EPL342 –Databases

Lab 3

ER Modeling (Relationships) in DDS Lite & Conceptual Modeling in SQL Server 2008 (2/2)

Panayiotis Andreou

http://www.cs.ucy.ac.cy/courses/EPL342

Before We Begin

- Start the DDS Lite
  - Start → All Programs → Chilli Source → DDS-Lite
- Start the SQL Server Management Studio
  - Start → All Programs → Microsoft SQL Server →SQL Server Management Studio

  Server: APOLLO.IN.CS.UCY.AC.CY

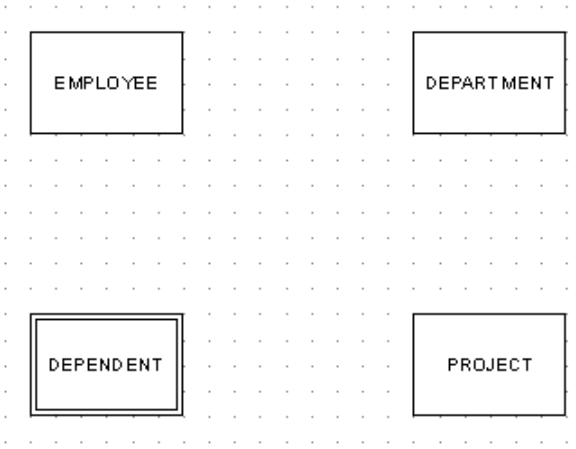
  Authentication: SQL Server Authentication

  Username: <check your email>

  Password: <check your email>

COMPANY Database

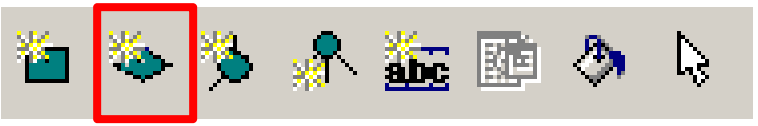
- During your lecture 4, you have identified 4 entities consisting of the COMPANY db:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT (weak)



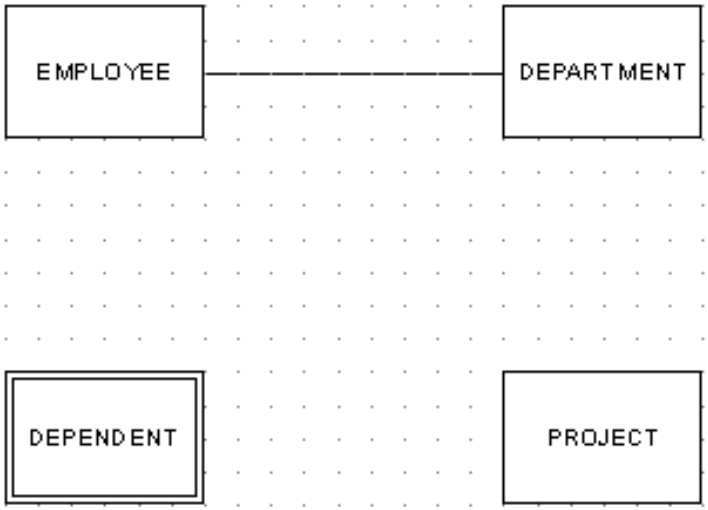
- Our second job is to design the entity/table relationships and adjust the table designs accordingly

DDS Lite – Create new Relationship

- To create a new relationship go to Insert → Relationship or



- Next, click on the main panel of DDS
- Drag the pencil cursor from one entity to another (e.g., DEPARTMENT to EMPLOYEE
- A new Relationship (Rel 1) will be created



DDS Lite – Relationship Properties

- To change the properties of a relationship right click on the relationship and select properties

Rel1	☐	DEPARTMENT	1 : M	EMPLOYEE	1	N		
		EMPLOYEE		DEPARTMENT	0	N	SET NULL	CASCADE

- Rename Rel1 to WORKS_FOR

DDS Lite – Relationship Properties (Min,Max)

- Participation properties

  Relationship: WORKS_FOR

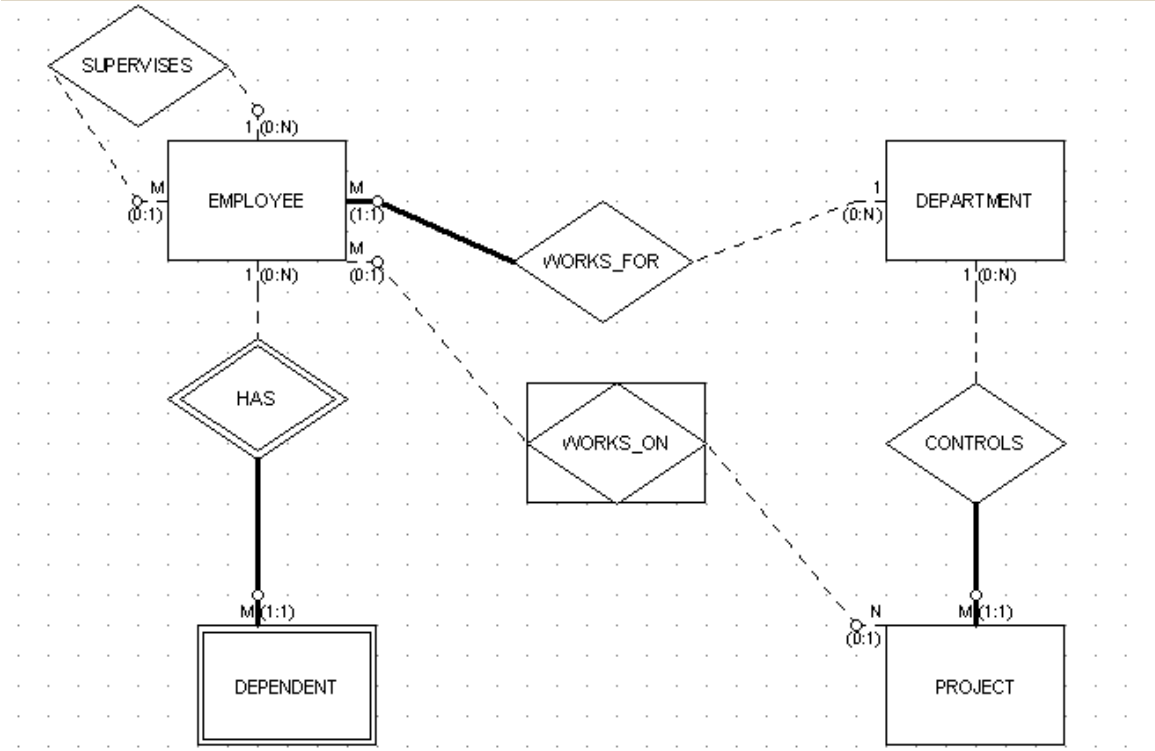
  - A Department may have 0 or more employees (min=0, max=N)
  - An employee is always assigned to exactly one department (min=1, max=1)

WORKS_FOR	☐	DEPARTMENT	1 : M	EMPLOYEE	0	N		
		EMPLOYEE		DEPARTMENT	1	1	RESTRICT	CASCADE

Practice: Create Relationships

- HAS: Employee has dependents
- WORKS_ON: Employees work on various projects
- SUPERVISES: Employee supervises other employees
- CONTROLS: Departments control projects

Practice solution



Relationship	Weak	From Entity	Connectivity	To Entity	Min	Max
CONTROLS	☐	DEPARTMENT	1 : M	PROJECT	0	N
		PROJECT		DEPARTMENT	1	1
HAS	☑	EMPLOYEE	1 : M	DEPENDENT	0	N
		DEPENDENT		EMPLOYEE	1	1
SUPERVISES	☐	EMPLOYEE	1 : M	EMPLOYEE	0	N
		EMPLOYEE		EMPLOYEE	0	1
WORKS_FOR	☐	DEPARTMENT	1 : M	EMPLOYEE	0	N
		EMPLOYEE		DEPARTMENT	1	1
WORKS_ON	☐	EMPLOYEE	M : N	PROJECT	0	1
		PROJECT		EMPLOYEE	0	1

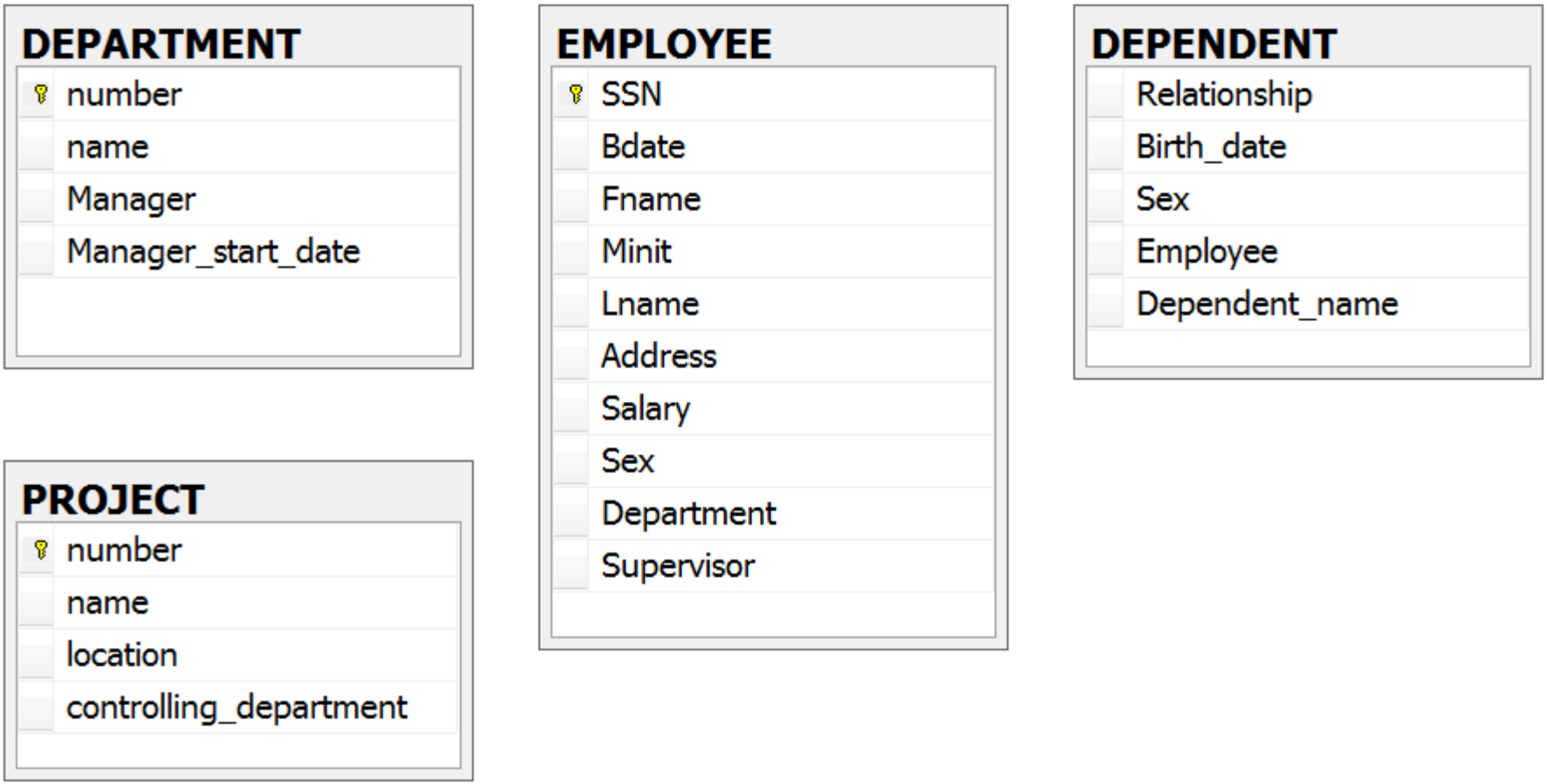
Discussion:

- Department may control various Projects (0,1)
- Project is always controlled by exactly one Department (1,1)
- Employee may have various Dependents (0,1)
- A Dependent belongs to exactly one Employee

Lab 3

Conceptual Modeling
in
SQL Server 2008
(Relationships)

COMPANY database diagram



Best Practices (1/?)

- When no explicit requirement exists, Primary Keys (PK) are usually named:

  <table_name>_id

When a PK is used as a Foreign Key (FK) then you can immediately pinpoint the table that the PK came from.

Relationships

There are three types of relationships between tables. The type of relationship that is created depends on how the related columns are defined.

- One-to-Many Relationship
- Many-to-Many Relationships
- One-to-One Relationships

One-to-Many Relationship

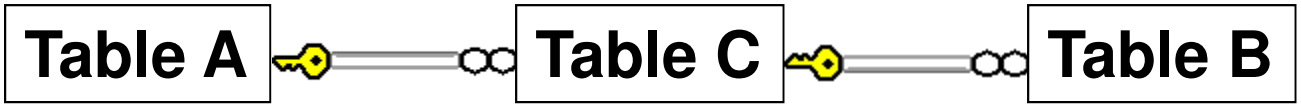
In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in

- Make a one-to-many relationship if only one of the related columns is a primary key or has a unique constraint.
- The primary key side of a one-to-many relationship is denoted by a key symbol. The foreign key side of a relationship is denoted by an infinity symbol.

Many-to-Many Relationship

In a many-to-many relationship, a row in table A can have many matching rows in table B, and vice versa.

You create such a relationship by defining a third table, called a junction table, whose primary key consists of the foreign keys from both table A and table B.



One-to-One Relationship

An a one-to-one relationship, a row in table A can have no more than one matching row in table B, and vice versa.

This type of relationship is not common because most information related in this way would be all in one table. You might use a one-to-one relationship to:

- Divide a table with many columns.
- Isolate part of a table for security reasons.
- Store data that is short-lived and could be easily deleted
- Store information that applies only to a subset of table.

The primary and foreign key side of a one-to-one relationship is denoted by the key symbol.

Create Relationship (Table Designer)

To create a foreign key relationship in Table Designer

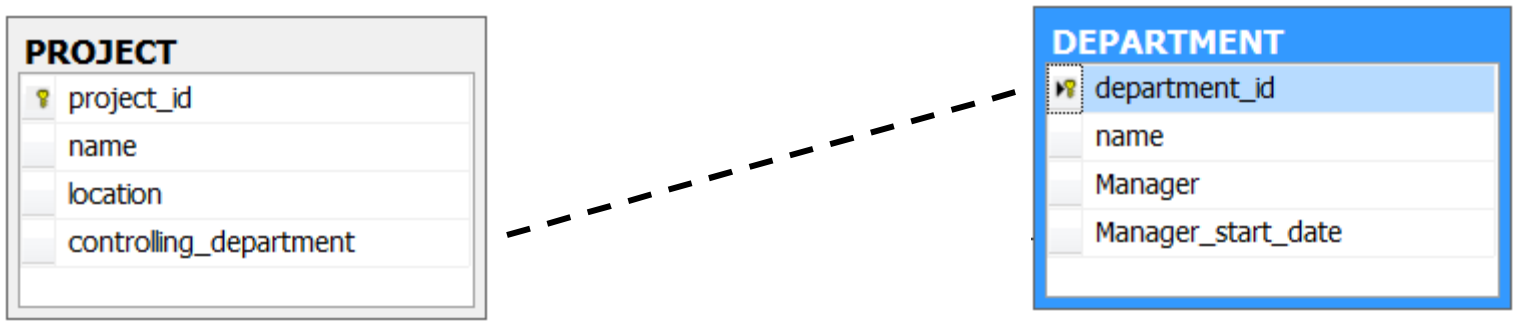
1. In Object Explorer, right-click the table that will be on the foreign-key side of the relationship and click Design.
2. From the Table Designer menu, click Relationships.
3. In the Foreign-key Relationships dialog box, click Add.
4. The relationship appears in the Selected Relationship list with a system-provided name in the format FK_<*tablename*>_<*tablename*>
5. Click the relationship in the Selected Relationship list.

Create Relationship (Table Designer)

6. Click Tables and Columns Specification in the grid to the right and click the ellipses (…) to the right of the property.
7. In the Tables and Columns dialog box, in the Primary Key drop-down list, choose the table that will be on the primary-key side of the relationship.
8. In the grid beneath, choose the columns contributing to the table's primary key. In the adjacent grid cell to the left of each column, choose the corresponding foreign-key column of the foreign-key table.
9. Table Designer suggests a name for the relationship. To change this name, edit the contents of the Relationship Name text box.
10. Choose OK to create the relationship.

Create Relationship with Database Diagrams

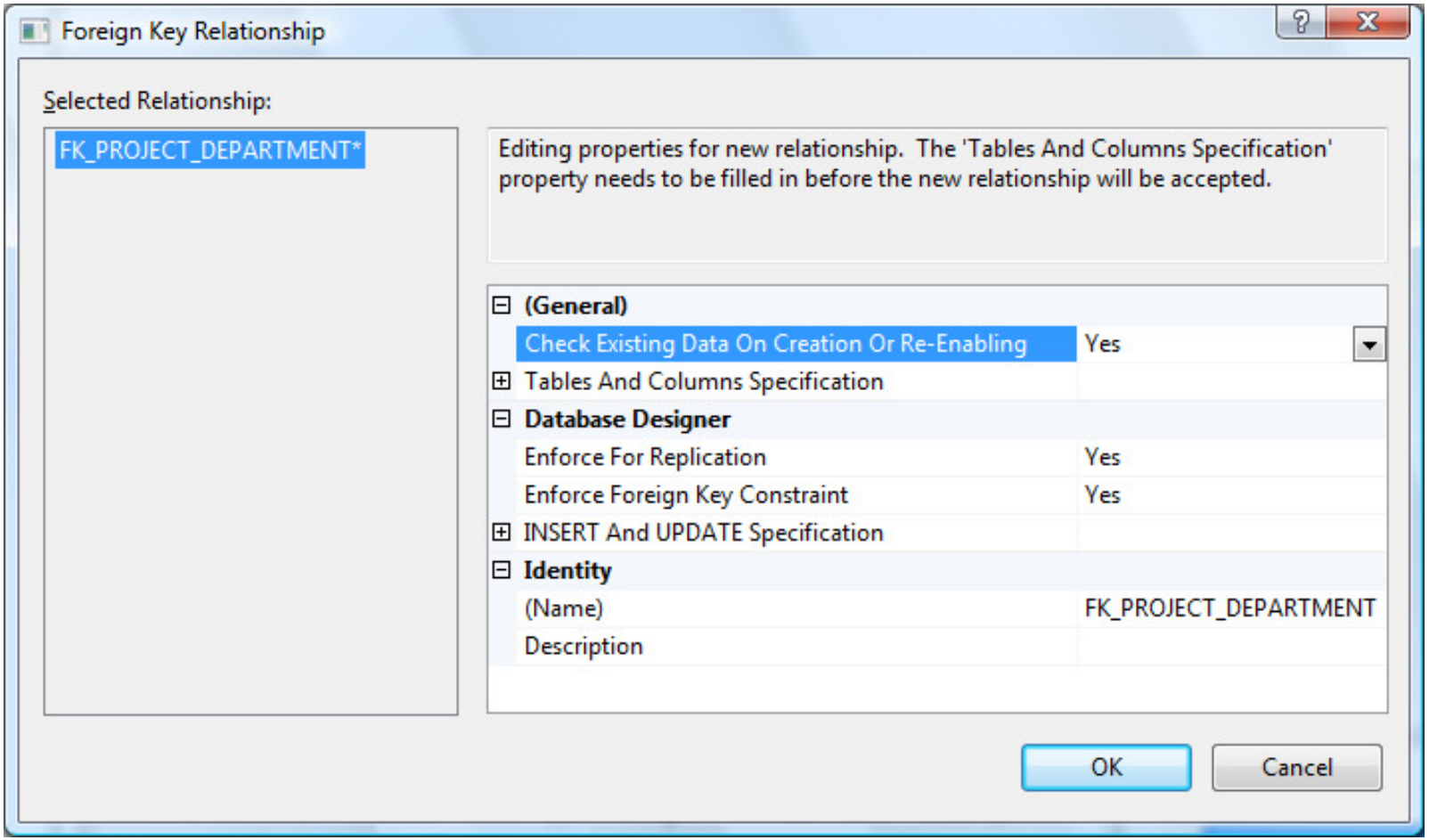
- Simply drag the primary key of the Table A to the foreign key of Table B



- You can only create a relationship if the columns have identical data types (Data type, Length, Precision)

Relationship Options

Department Has Many (One-To-Many) Projects



Relationship Options

- **Check Existing Data on Creation or Re-Enabling**
  Verify all existing data in the table before the constraint was created or re-enabled, against the constraint.
- **Enforce Foreign Key Constraint**
  Specify whether changes are allowed to the data of the columns in the relationship if those changes would invalidate the integrity of the foreign key relationship. Choose **Yes** if you do not want to allow such changes, and choose **No** if you do want to allow them.
- **Enforce For Replication**
  Indicates whether to enforce the constraint when a replication agent performs an insert, update, or delete on this table.

Relationship Options

INSERT and UPDATE Specification Category
Expand to show information for the **Delete Rule** and the **Update Rule** for the relationship.

Delete Rule Specify what happens if a user tries to delete a row with data that is involved in a foreign key relationship:

- **No Action**: An error message tells the user that the deletion is not allowed and the DELETE is rolled back.
- **Cascade:** Deletes all rows containing data involved in the foreign key relationship.
- **Set Null:** Sets the value to null if all foreign key columns for the table can accept null values.
- **Set Default:** Sets the value to the default value defined for the column if all foreign key columns for the table have defaults defined for them.

Relationship Options

Update Rule Specify what occurs if a user tries to update a row with data that is involved in a foreign key relationship:

- **No Action:** An error message tells the user that the update is not allowed and the UPDATE is rolled back.
- **Cascade:** Updates all rows that contain data involved in the foreign key relationship.
- **Set Null:** Sets the value to null if all foreign key columns for the table can accept null values.
- **Set Default:** Sets the value to the default value that is defined for the column if all foreign key columns for the table have defaults defined for them.

COMPANY relationships

Create the following relationships

1. **FK_DEPARTMENT_PROJECT**

   DEPARTMENT.departmentid→
   PROJECT. controlling_department

2. **FK_DEPARTMENT_EMPLOYEE**

   DEPARTMENT.departmentid→
   EMPLOYEE.department

3. **FK_EMPLOYEE_DEPENDENT**

   EMPLOYEE.ssn→
   DEPENDENT.employee

When necessary, adjust the data types of the foreign keys accordingly