CS203 DB Principals

IS206 Fundamentals of DB

Asst.Prof Asaad Alhijaj

Reference:

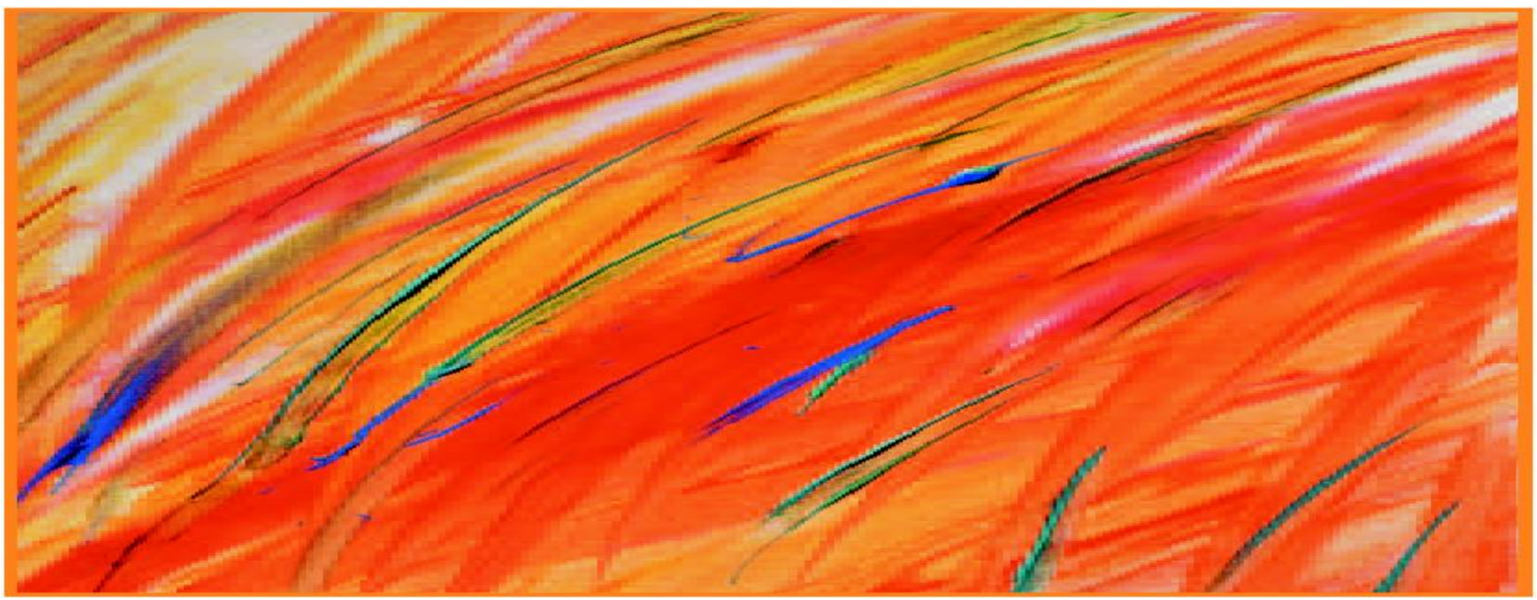
DAVID M. KROENKE'S DATABASE CONCEPTS, 2nd Edition

DAVID M. KROENKE'S

DATABASE CONCEPTS, 2nd Edition

Chapter Five

Database Design



Chapter Objectives

- **Learn how to transform E-R data models into relational designs**
- **Practice the normalization process from Chapter 2**
- **Understand the need for denormalization**
- **Learn how to represent weak entities with the relational model**
- **Know how to represent 1:1, 1:N, and N:M binary relationships**

Chapter Objectives (continued)

- Know how to represent 1:1, 1:N, and N:M recursive relationships
- Learn SQL statements for creating joins over binary and recursive relationships
- Understand the nature and background of normalization

Representing Entities with the Relational Model

- Create a relation for each entity
  - A relation has a descriptive name and a set of attributes that describe the entity
- The relation is then analyzed using the normalization rules
- As normalization issues arise, the initial relation design may need to change

Anomalies

- Relations that are not normalized will experience issues known as anomalies
  - – Insertion anomaly
    - Difficulties inserting data into a relation
  - – Modification anomaly
    - Difficulties modifying data into a relation
  - – Deletion anomaly
    - Difficulties deleting data from a relation

Solving Anomalies

- Most anomalies are solved by breaking an existing relation into two or more relations through a process known as normalization

Definition Review

- Functional dependency
  - The relationship (within the relation) that describes how the value of a one attribute may be used to find the value of another attribute
- Determinant
  - The attribute that can be used to find the value of another attribute in the relation
  - The right-hand side of a functional dependency

Definition Review

- Candidate key
  - The value of a candidate key can be used to find the value of every other attribute in the relation
  - A simple candidate key consists of only one attribute
  - A composite candidate key consists of more than one attribute

Normal Forms

- There are many defined normal forms:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)
  - Fifth Normal Form (5NF)
  - Domain/Key Normal Form (DK/NF)

Normalization

- For our purposes, a relation is considered normalized when:

Every determinant is a candidate key

[Technically, this is Boyce-Codd Normal Form (BCNF)]

Denormalization

- Normalizing relations (or breaking them apart into many component relations) may significantly increase the complexity of the data structure
- The question is one of balance
  - Trading complexity for anomalies
- There are situations where denormalized relations are preferred

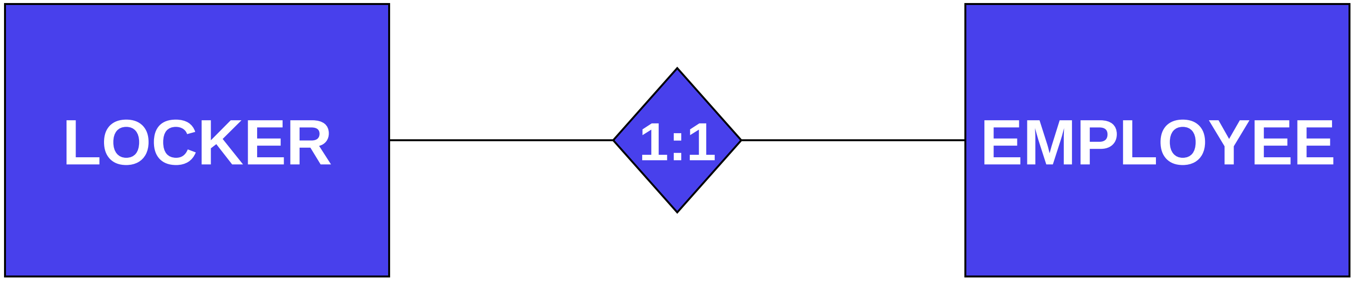
Weak Entities

- For an ID-dependent weak entity, the key of the parent becomes part of the key of the weak entity

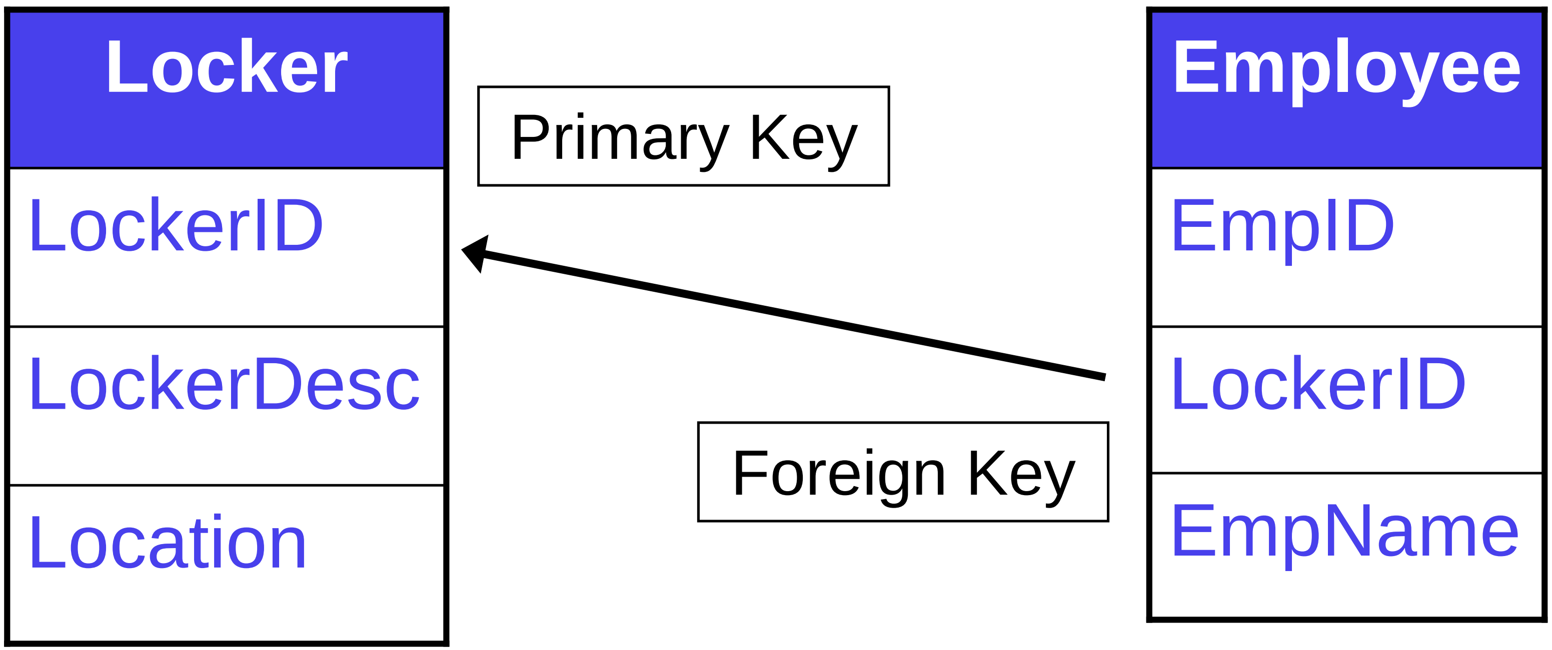
Representing Relationships

- The maximum cardinality determines how a relationship is represented
- 1:1 relationship
  - The key from one relation is placed in the other as a *foreign key*
  - It does not matter which table receives the foreign key

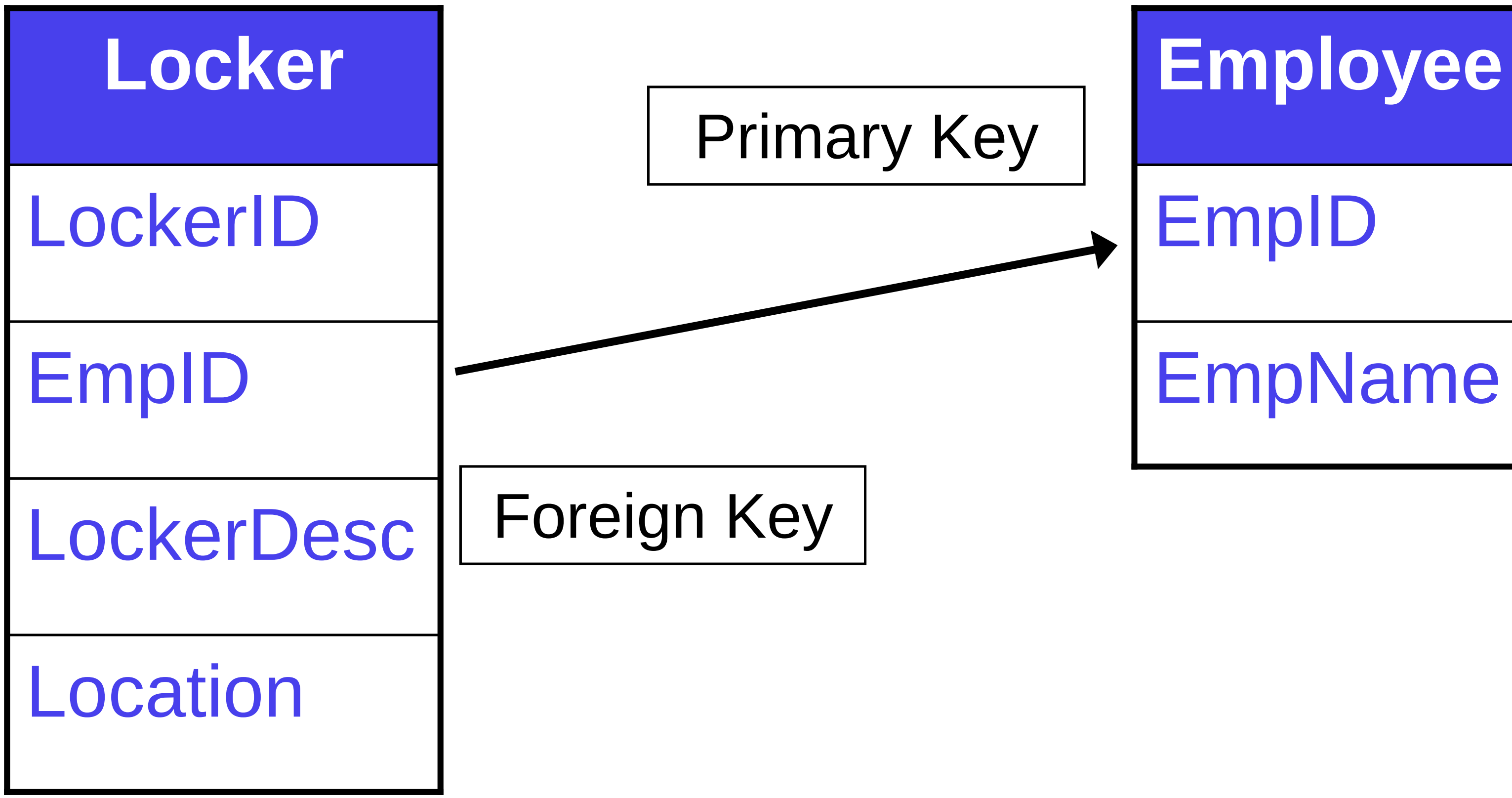
A One-to-One Relationship Example



One Representation of a One-to-One Relationship



Another Representation of a One-to-One Relationship



SQL For a 1:1 Join

```
SELECT      *
FROM        LOCKER, EMPLOYEE
WHERE       LOCKER.EmpID = EMPLOYEE.EmpID;

SELECT      *
FROM        LOCKER, EMPLOYEE
WHERE       LOCKER.LockerID = EMPLOYEE.LockerID;
```

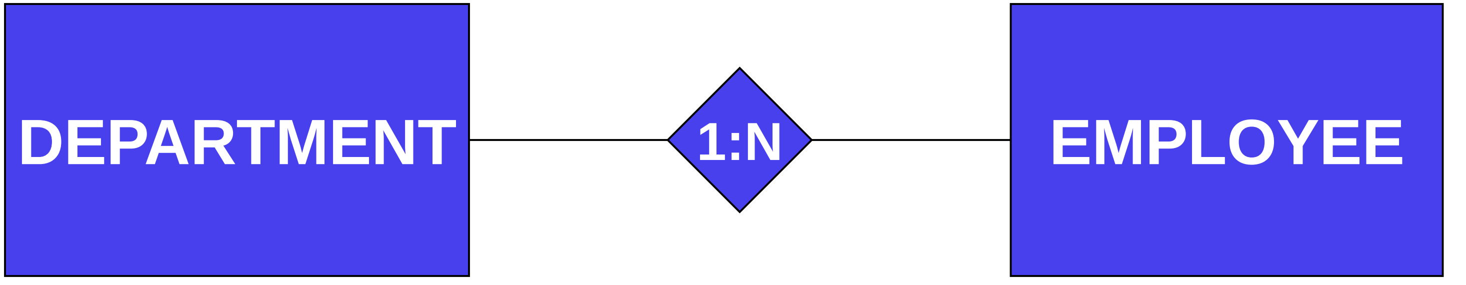
Mandatory One-to-One Relationships

- A mandatory 1:1 relationship can easily be collapsed back into one relation. While there are times when the added complexity is warranted…
  – Added security
  – Infrequently accessed data components
- …very often these relations are collapsed into one relation

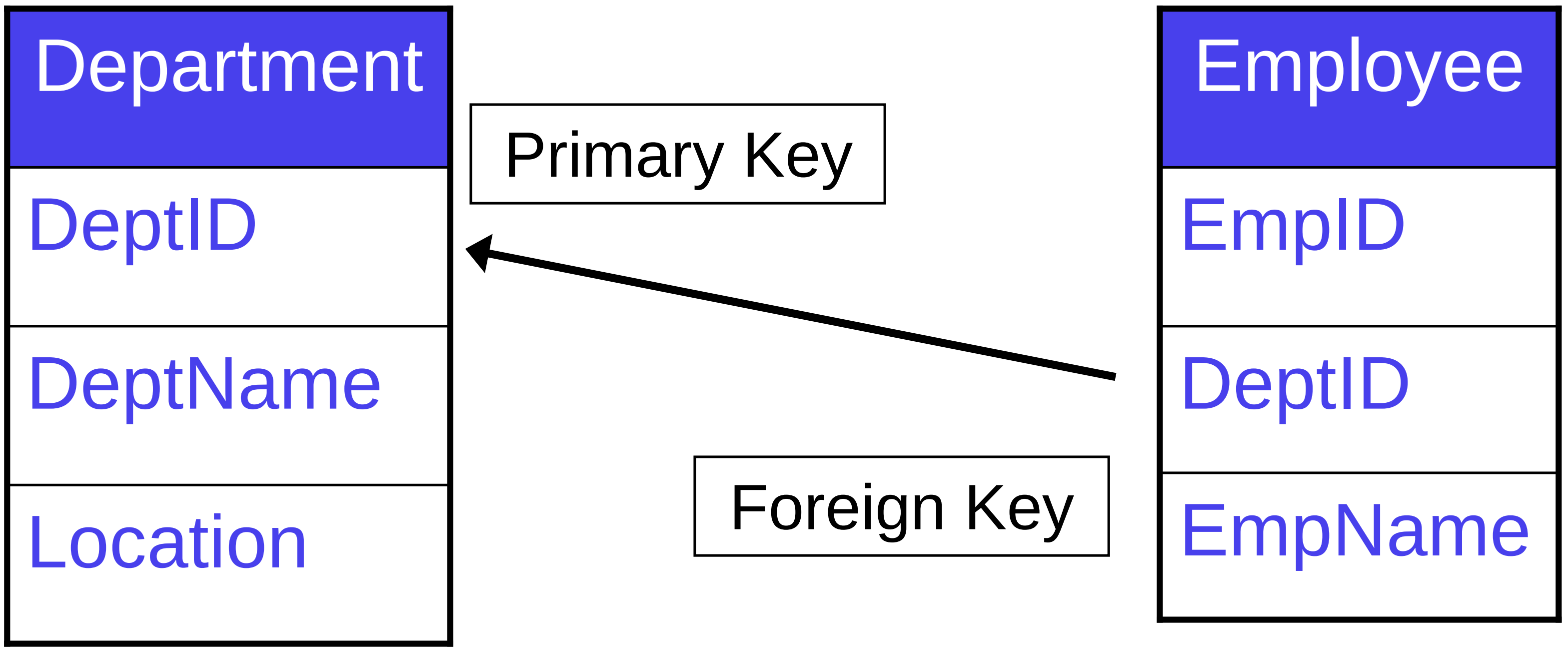
One-to-Many Relationships

- Like a 1:1 relationship, a 1:N relationship is saved by placing the key from one table into another as a foreign key
- However, in a 1:N the foreign key always goes into the many-side of the relationship

A One-to-Many Relationship Example



Representing a One-to-Many Relationship



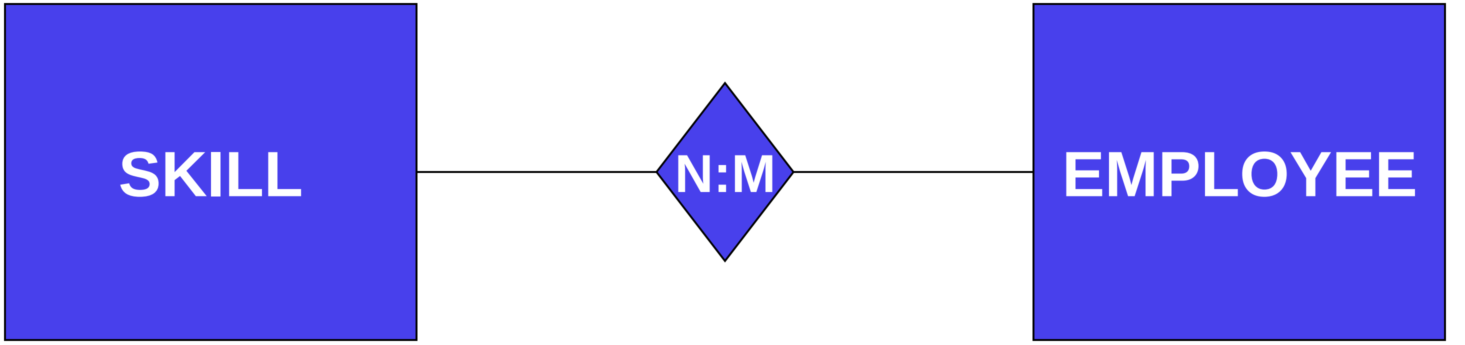
SQL For a 1:N Join

```
SELECT    *
FROM      DEPARTMENT, EMPLOYEE
WHERE     DEPARTMENT.DeptID = EMPLOYEE.DeptID;
```

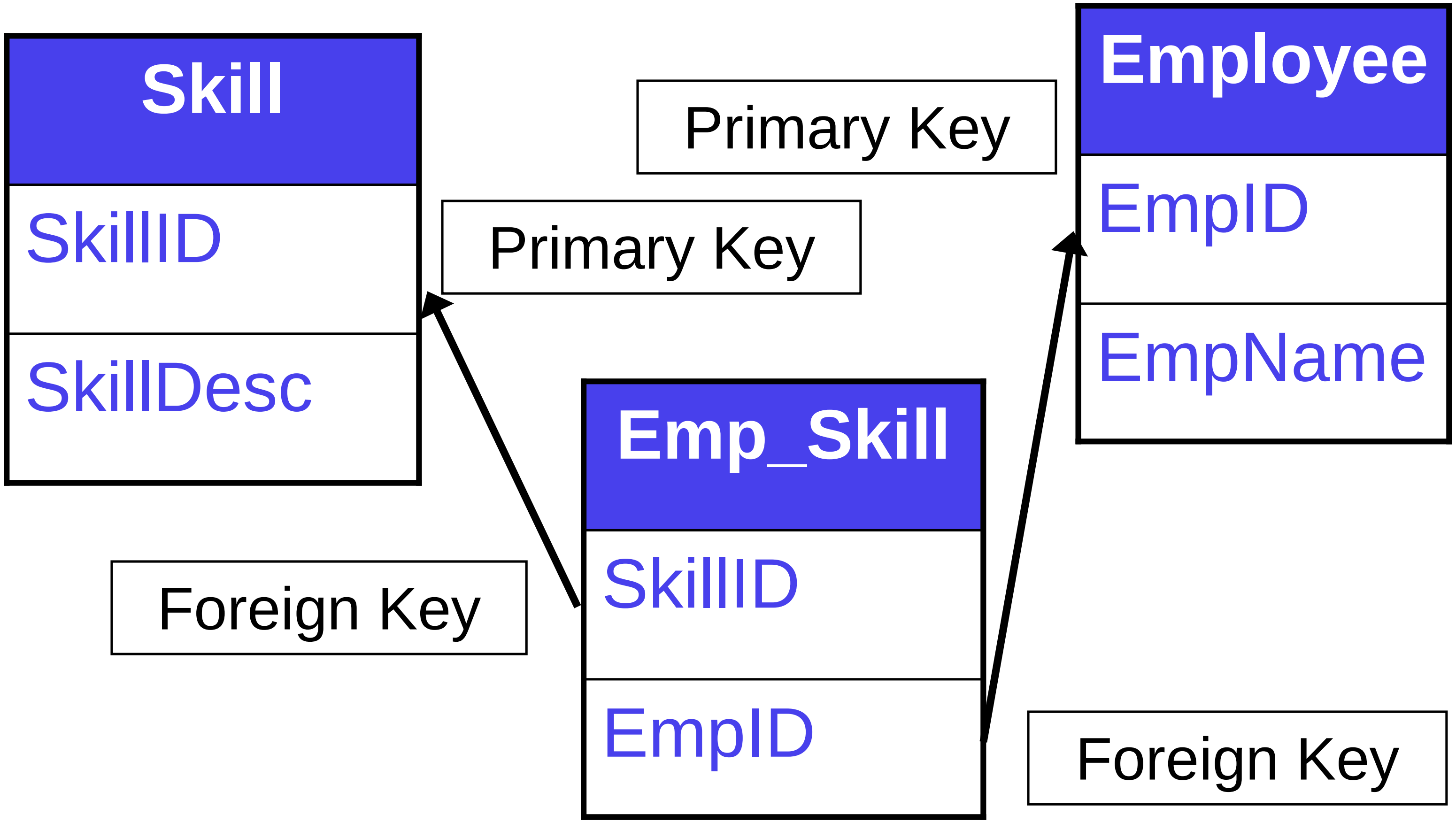
Representing Many-to-Many Relationships

- To save a M:N relationship, a new relation is created. This relation is called an *intersection relation*
- An intersection relation has a composite key consisting of the keys from each of the tables that formed it

A Many-to-Many Relationship Example



Representing a Many-to-Many Relationship



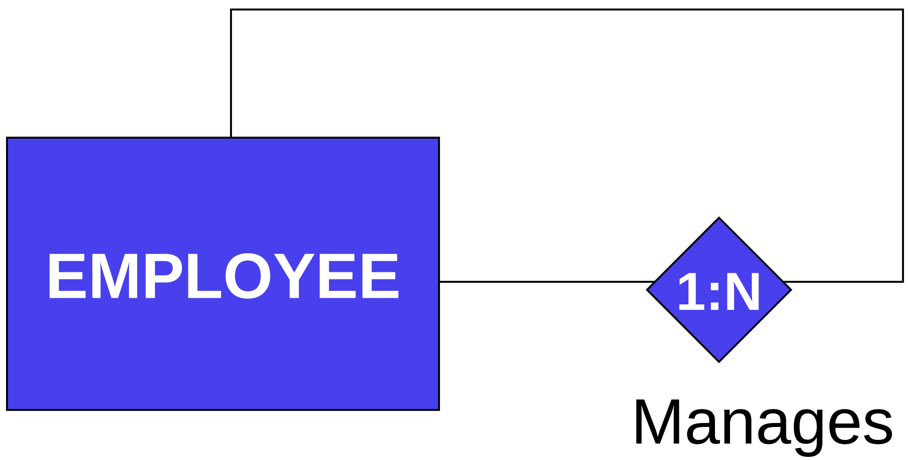
SQL For a N:M Join

```
SELECT        *
FROM          SKILL, EMP_SKILL,  EMPLOYEE
WHERE         SKILL.SkillID = EMP_SKILL.SkillID
        AND   EMP_SKILL.EmpID = EMPLOYEE.EmpID;
```

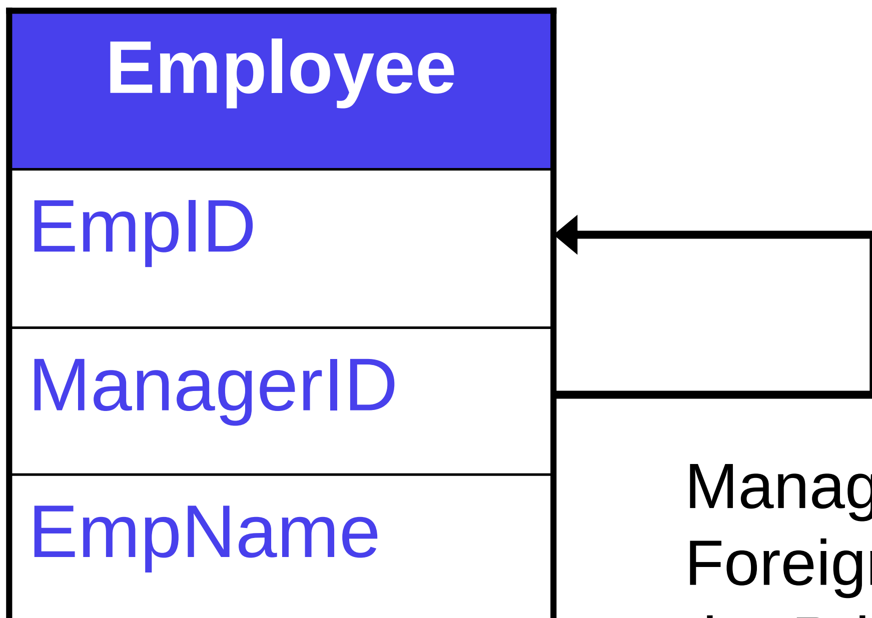
Representing Recursive Relationships

- A *recursive relationship* is a relationship that a relation has with itself.
- Recursive relationships adhere to the same rules as the binary relationships.
  - 1:1 and 1:M relationships are saved using foreign keys
  - M:N relationships are saved by creating an intersecting relation

A Recursive Relationship Example



Representing a Recursive Relationship



ManagerID is a Foreign Key referencing the Primary Key EmpID

SQL For a 1:1 Recursive Join

```
SELECT    A.EmpID, A.EmpName as 'Manager',
          B.EmpID, B.EmpName as 'Worker'
FROM      EMPLOYEE  A, EMPLOYEE  B
WHERE     A.EmpID = B.ManagerID;
```

Example results:

EmpID	Manager	EmpID	Worker
4	Bryant	1	Jones
4	Bryant	2	Adams
4	Bryant	3	Smith
5	Dean	4	Bryant

Cascading Behavior

- Cascading behavior describes what happens to child relations when a parent relation changes in value
- Cascading behaviors are defined by the type of operation
  - Cascade update
  - Cascade delete