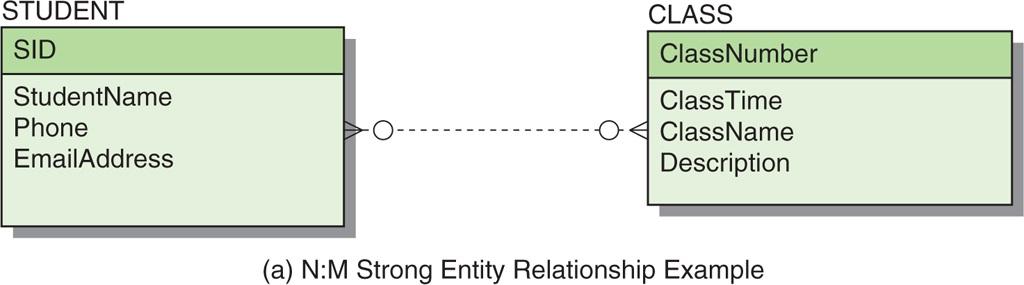
#### Representing N:M Strong Entity Relationships

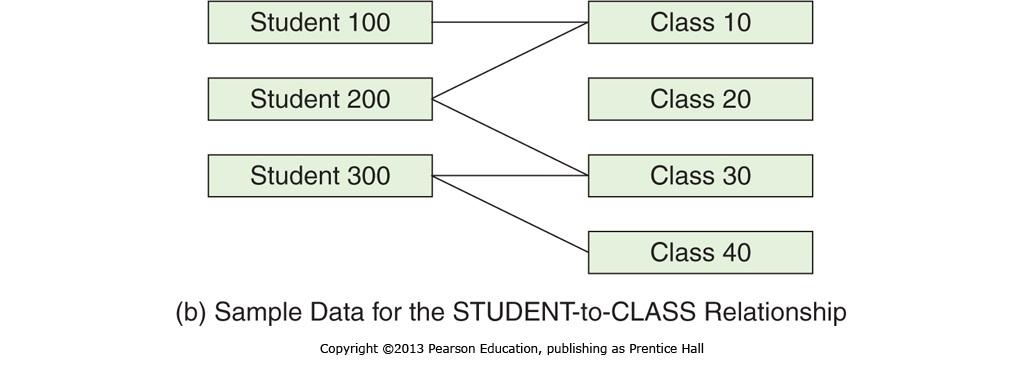
* The third type of binary relationship is N:M.
* Many entities of each type correspond to many entities of the other type.
* Figure 5-13(a) shows an E-R diagram of the N:M relationship between students and classes.

**Figure 5-13(a): N:M Strong Entity Relationship**



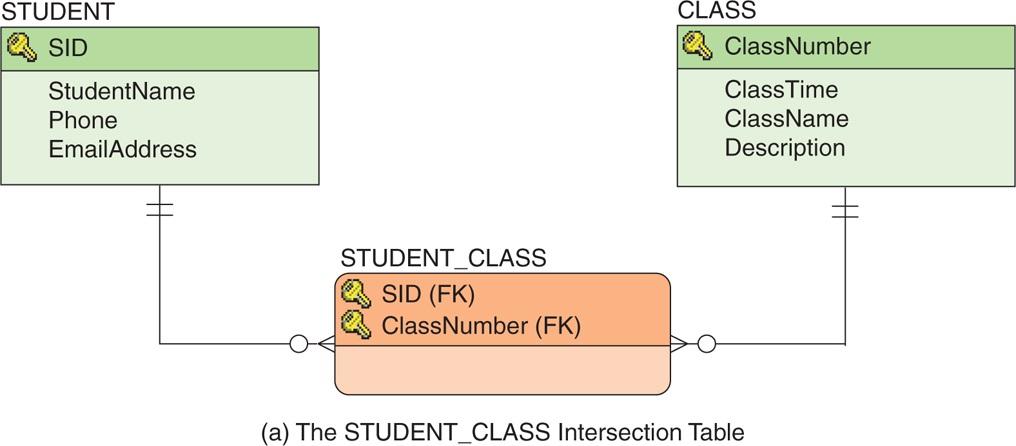
* A STUDENT entity can correspond to many CLASS entities.
* A CLASS entity can correspond to many STUDENT entities.
* Both participants in the relationship are optional:
  + A student does not need to be enrolled in a class.
  + A class is not required to have any students.
* Figure 5- 13(b) gives sample data.

**Figure 5-13(b): Sample Data for the STUDENT-to-CLASS Relationship**



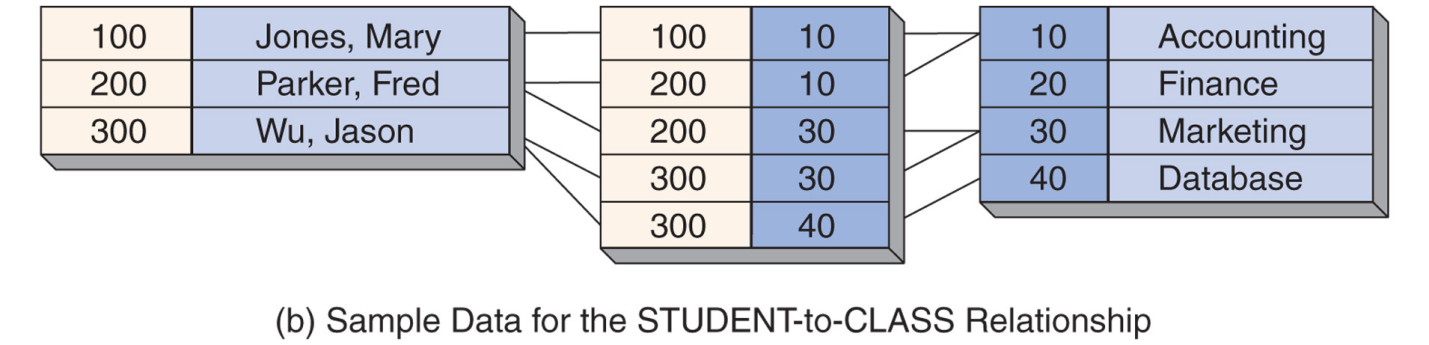
* N:M relationships cannot be represented directly by relations in the same way that 1:1 and 1:N relationships are represented.
* The solution to this problem is to create a third table, called an **intersection table**.
* The intersection table represents the relationship itself.
* Define a table named STUDENT\_CLASS, as shown in Figure 5-15(a):
  + STUDENT (SID, StudentName, Phone, EmailAddress)
  + CLASS (ClassNumber, ClassTime, ClassName, Description)
  + STUDENT\_CLASS (*SID*, *ClassNumber*)
* The referential integrity constraints are:
  + SID in STUDENT\_CLASS must exist in SID in STUDENT.
  + ClassNumber in STUDENT\_CLASS must exist in ClassNumber in CLASS.

**Figure 5-15(a): Representing an N:M Strong Entity Relationship**



* Some instances of this relation are shown in Figure 5-15(b).

**Figure 5-15(b): Sample Data for the STUDENT-to-CLASS relationship**



* Such relations are called intersection tables because each row documents the intersection of a particular student with a particular class.
* In Figure 5-15(b), the intersection relation has one row for each line between STUDENT and CLASS, as in Figure 5-13(b).
* In Figure 5-15(a), the relationship from STUDENT to STUDENT\_CLASS is 1:N, and the relationship from CLASS to STUDENT\_CLASS is also 1:N.
* We have decomposed the M:N relationship into two 1:N relationships.
* The key of STUDENT\_CLASS is (SID, ClassNumber), which is the combination of the primary keys of both of its parents.
* The key for an intersection table is always the combination of parent keys.
* The parent relations are both required because a parent must now exist for each key value in the intersection relation.

To actually implement an N:M relationship in a database:

* Create a new intersection table.
* Add foreign key columns linking to the two tables in the N:M relationship.
* These foreign key columns will be the corresponding primary keys of the two tables.
* The foreign keys together will form a composite primary key in the intersection table.
* Obtain data about students and classes by using the following SQL statement:
  + /\* \*\*\* SQL-Query-CH05-04 \*\*\* \*/  
    SELECT \*  
    FROM STUDENT, CLASS, STUDENT\_CLASS  
    WHERE STUDENT.SID = STUDENT\_CLASS.SID  
     AND STUDENT\_CLASS.ClassNumber = CLASS.ClassNumber;
* The result of this SQL statement is a table with all columns for a student and the classes the student takes.
* The student data will be repeated in the table for as many classes as the student takes.
* The class data will be repeated in the relation for as many students as are taking the class.

**Ch5-Prac-N**

**In-Class:** Chapter 5 Practice N:M Relationship - In-Class \_\_\_ \_\_\_ \_\_