7.4.3: Cardinality

Cardinality is a constraint on a relationship specifying the number of entity instances that a specific entity may be related to via the relationship. Suppose we have the following rules for departments and employees:

- A department can have several employees that work in the department
- An employee is assigned to work in one department.

From these rules we know the cardinalities for the *works in* relationship and we express them with the cardinality symbols *1* and *n* below.

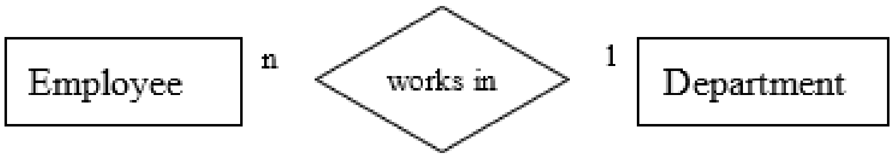


Figure 7.19: One-to-many relationships are most common

The *n* represents an *arbitrary number of instances*, and the *1* represents *at most one instance*. For the above works in relationship we have

- a specific employee works in at most only one department, and
- a specific department may have many (zero or more) employees who work there.

n, *m*, *N*, and *M* are common symbols used in ER diagrams for representing an arbitrary number of occurrences; however, any alphabetic character will suffice.

Based on cardinality there are three types of binary relationships: *one-to-one*, *one-to- many*, and *many-to-many*.

One-to-One

One-to-one relationships have *1* specified for both cardinalities. Suppose we have two entity types Driver and Vehicle. Assume that we are only concerned with the current driver of a vehicle, and that we are only concerned with the current vehicle that a driver is operating. Our two rules associate an instance of one entity type with at most one instance of the other entity type:

- a driver operates at most one vehicle, and
- a vehicle is operated by at most one driver.

And so, the relationship is one-to-one.

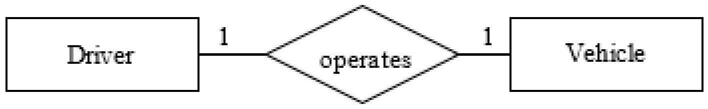


Figure 7.20: One-to-one relationship

One-to-Many

One-to-many relationships are the most common ones in database designs. Suppose we have customer entities and invoice entities and:

- an invoice is for exactly one customer, and
- a customer could have any number (zero or more) of invoices at any point in time.

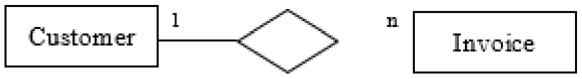


Figure 7.21: One-to-many relationship

Because one instance of an Invoice can only be associated with a single instance of Customer, and because one instance of Customer can be associated with any number of Invoice instances, this is a one-to-many relationship:

Many-to-Many

Suppose we are interested in courses and students and the fact that students register for courses. Our two rule statements are:

- any student may enroll in several courses,
- a course may be taken by several students.

This situation is represented as a many-to-many relationship between Course and Student:

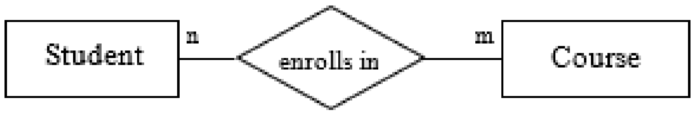


Figure 7.22: Many-to-many relationship

As will be discussed again later, a many-to-many relationship is implemented in a relational database in a separate relation. In a relational database for the above, there would be three relations: one for Student, one for Course, and one for the many-to-many. (Sometimes this 3rd relation is called an intersection table, a composite table, a bridge table.)

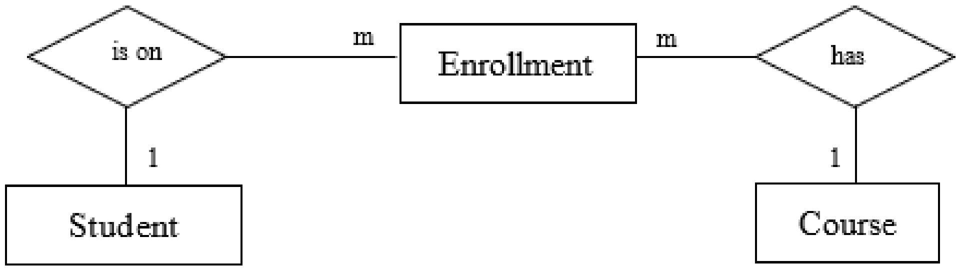


Figure 7.23: Many-to-many becomes two one-to-many relationships

Partly because of the need for a separate structure when the database is implemented, many modellers will ‘resolve’ a many-to-many relationship into two one-to-many relationships as they are modelling. We can restructure the above many-to-many as two one-to-many relationships where we have ‘invented’ a new entity type called Enrollment:

> A student can have many enrollments, and each course may have many enrollments.
> An enrollment entity is related to one student entity and to one course entity.

7.4.4: Recursive Relationships

A relationship is *recursive* if the same entity type appears more than once. A typical business example is a rule such as “an employee *supervises* other employees”. The *supervises* relationship is recursive; each instance of *supervises* will specify two employees, one of which is considered a *supervisor* and the other the *supervised*. In the following diagram the relationship symbol joins to the Employee entity type twice by two separate lines. Note the relationship is one-to-many: an employee may supervise many employees, and an employee may be supervised by at most one other employee.

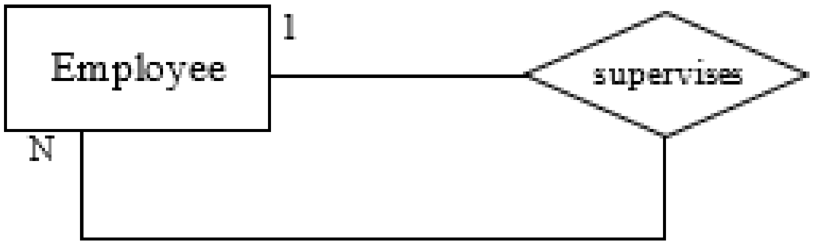


Figure 7.24: Recursive relationship involving Employee twice

With recursive relationships it is appropriate to name the roles each entity type plays. Suppose we have an instance of the relationship:

John *supervises* Terry

Then with respect to this instance, John is the *supervisor* employee and Terry is the *supervised* employee. We can show these two roles that entity types play in a relationship by placing labels on the relationship line:

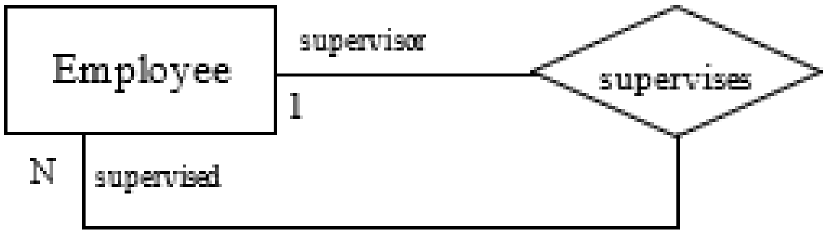


Figure 7.25: Recursive relationship with role names

This one-to-many *supervises* relationship can be visualized as a hierarchy. In the following we show five instances of the relationship: John supervises Lee, John supervises Peter, Peter supervises Don, Peter supervises Mary, and John supervises Noel.

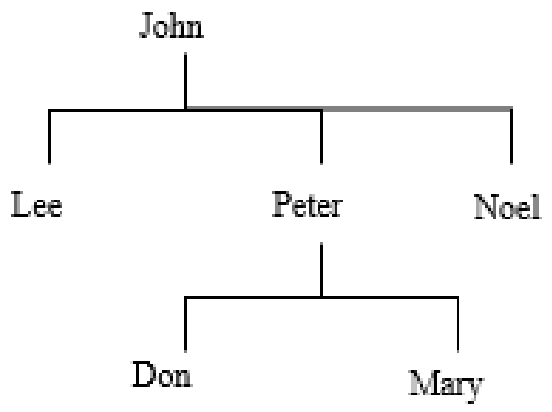


Figure 7.26: The supervising hierarchy

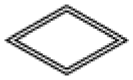
In the above example note the participation constraint at both ends of *supervises* is optional. This must be the case because some employee will not be supervised, and, for some employees there are no employees they supervise.

Generally recursive relationships are difficult to master. Some other situations where recursive relationships can be used:

- A person marries another person
- A person is the parent of a person
- A team plays against another team
- An organizational unit reports to another organizational unit
- A part is composed of other parts.

7.4.5: Identifying Relationships

When entity types were first introduced, we discussed an example where a department offers courses and that a course must exist in the context of a department. In that case the Course entity type is considered a weak entity type as it is existence-dependent on Department. It is typical in such situations that the key of the strong entity type is used in the identification scheme for the weak entity type. For example, courses could be identified as MATH-123 or PHYS-329, or as Mathematics-123 or Physics-329. In order to convey the composite identification scheme for a weak entity type we specify the relationship as an *identifying* relationship which is visualized using a double-lined diamond symbol:



Additionally, in situations where we have an identifying relationship we usually have:

- a weak entity type with a partial key
- a weak entity type that must participate in the relationship (total participation) and so the ERD for our hypothetical educational institution could be:

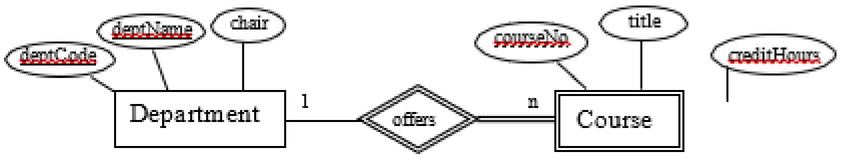


Figure 7.27: An identifying relationship

Note the keys for the strong entity type appear only at the strong entity type. The identifying relationship tells one that a department key will be needed to complete the identification of a course.