Relationship Types, Sets, Roles, and Structural Constraints

Having presented a preliminary database schema for COMPANY, it is now convenient to clarify the concept of a relationship (which is the last of the three main concepts involved in the ER model).

**Relationship**: This is an association between two entities. As an example, one can imagine a STUDENT entity being associated to an ACADEMIC_COURSE entity via, say, an ENROLLED_IN relationship.

Whenever an attribute of one entity type refers to an entity (of the same or different entity type), we say that a relationship exists between the two entity types.

From our preliminary COMPANY schema, we identify the following relationship types (using descriptive names and ordering the participating entity types so that the resulting phrase will be in active voice rather than passive):

- EMPLOYEE MANAGES DEPARTMENT (arising from Manager attribute in DEPARTMENT)
- DEPARTMENT CONTROLS PROJECT (arising from ControllingDept attribute in PROJECT and the Projects attribute in DEPARTMENT)
- EMPLOYEE WORKS_FOR DEPARTMENT (arising from Dept attribute in EMPLOYEE and the Employees attribute in DEPARTMENT).
- EMPLOYEE SUPERVISES EMPLOYEE (arising from Supervisor attribute in EMPLOYEE)
- EMPLOYEE WORKS_ON PROJECT (arising from WorksOn attribute in EMPLOYEE and the Workers attribute in PROJECT)
- DEPENDENT DEPENDS_ON EMPLOYEE (arising from Employee attribute in DEPENDENT and the Dependents attribute in EMPLOYEE)

In ER diagrams, relationship types are drawn as diamond-shaped boxes connected by lines to the entity types involved. See Figure 3.2, page 62, Navathe & Elmasri.

**Note**: that attributes are depicted by ovals connected by lines to the entity types they describe (with multi-valued attributes in double ovals and composite attributes depicted by trees). The original attributes that gave rise to the relationship types are absent, having been replaced by the relationship types.

A **relationship set** is a set of instances of a relationship type. If, say, R is a relationship type that relates entity types A and B, then, at any moment in time, the relationship set of R will be a set of ordered pairs (x,y), where x is an instance of A and y is an instance of B.
Ordering of entity types in relationship types: Note that the order in which we list the entity types in describing a relationship is of little consequence, except that the relationship name (for purposes of clarity) ought to be consistent with it. For example, if we swap the two entity types in each of the first two relationships listed above, we should rename them IS_MANAGED_BY and IS_CONTROLLED_BY, respectively.

Degree of a relationship type: Also note that, in our COMPANY example, all relationship instances will be ordered pairs, as each relationship associates an instance from one entity type with an instance of another (or the same, in the case of SUPERVISES) relationship type. Such relationships are said to be binary, or to have degree two. Relationships with degree three (called ternary) or more are also possible, although not as common.

Roles in relationships: Each entity that participates in a relationship plays a particular role in that relationship, and it is often convenient to refer to that role using an appropriate name. For example, in each instance of a WORKS_FOR relationship set, the employee entity plays the role of worker or (surprise!) employee and each department plays the role of employer or (surprise!) department. Indeed, as this example suggests, often it is best to use the same name for the role as for the corresponding entity type.

An exception to this rule occurs when the same entity type plays two (or more) roles in the same relationship. (Such relationships are said to be recursive, which I find to be a misleading use of that term. A better term might be self-referential.) For example, in each instance of a SUPERVISES relationship set, one employee plays the role of supervisor and the other plays the role of supervisee.

Constraints on Relationship Types
Often, in order to make a relationship type be an accurate model of the miniworld concepts that it is intended to represent, we impose certain constraints that limit the possible corresponding relationship sets.

- Cardinality ratio:
  1:1 (one-to-one): Under this constraint, no instance of A may participate in more than one instance of R; similarly for instances of B. In other words, if (a1, b1) and (a2, b2) are (distinct) instances of R, then neither a1 = a2 nor b1 = b2.
  Example: Our informal description of COMPANY says that every department has one employee who manages it. If we also stipulate that an employee may not...
(simultaneously) play the role of manager for more than one department, it follows that MANAGES is 1:1.

1: N (one-to-many): Under this constraint, no instance of B may participate in more than one instance of R, but instances of A are under no such restriction. In other words, if (a1, b1) and (a2, b2) are (distinct) instances of R, then it cannot be the case that b1 = b2.

Example: CONTROLS is 1: N because no project may be controlled by more than one department. On the other hand, a department may control any number of projects, so there is no restriction on the number of relationship instances in which a particular department instance may participate. For similar reasons, SUPERVISES is also 1:N.

N: 1 (many-to-one): This is just the same as 1: N but with roles of the two entity types reversed. Example: WORKS_FOR and DEPENDS_ON are N: 1.

M: N (many-to-many): Under this constraint, there are no restrictions. (Hence, the term applies to the absence of a constraint!)

Example: WORKS_ON is M: N, because an employee may work on any number of projects and a project may have any number of employees who work on it.

Participation: specifies whether or not the existence of an entity depends upon its being related to another entity via the relationship.

1. Total participation (or existence dependency): To say that entity type A is constrained to participate totally in relationship R is to say that if (at some moment in time) R's instance set is \{ (a1, b1), (a2, b2), ... (am, bm) \}, then (at that same moment) A's instance set must be \{ a1, a2, ..., am \}. In other words, there can be no member of A's instance set that does not participate in at least one instance of R. According to our informal description of COMPANY, every employee must be assigned to some department. That is, every employee instance must participate in at least one instance of WORKS_FOR, which is to say that EMPLOYEE satisfies the total participation constraint with respect to the WORKS_FOR relationship.

2. Partial participation: the absence of the total participation constraint! (E.g., not every employee has to participate in MANAGES; hence we say that, with respect to MANAGES, EMPLOYEE participates partially. This is not to say that for all employees to be managers is not allowed; it only says that it need not be the case that all employees are managers.)
Attributes of Relationship Types
Relationship types, like entity types, can have attributes. A good example is WORKS_ON, each instance of which identifies an employee and a project on which (s) he works. In order to record (as the specifications indicate) how many hours are worked by each employee on each project, we include Hours as an attribute of WORKS_ON. In the case of an M: N relationship type (such as WORKS_ON), allowing attributes is vital. In the case of an N: 1, 1: N, or 1:1 relationship type, any attributes can be assigned to the entity type opposite from the 1 side. For example, the StartDate attribute of the MANAGES relationship type can be given to either the EMPLOYEE or the DEPARTMENT entity type.