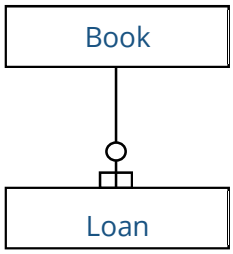
8 DENORMALIZATION

In this illustration, three examples of denormalization are given for a situation in which a 1:(N) relationship exists between two entity types. The situations 1:N, (1):N, (1):(N), 1:(1), and (1):(1) speak for themselves. (See section 4.21.3 for information about the notation method.)

8.1 1:(N) with independent existence

Problem description



The normalized data model of a library application shows that there is a 1:(N) relationship between the entity types *Book* and *Loan*. A book does not have to be loaned out and, so, optionality is a factor in the relationship. If a loan is made, it always relates to one *Book* (no optionality).

The library's business rule is that a *Book* can be deleted only if a *Loan* is no longer linked to it.

Does this case involve one or two logical files?

Discussion

Book and *Loan* have a 1:(N) relationship. According to the table in section 4.21.5, the number of logical files is determined on the basis of entity dependence. Because the library may delete a *Book* only when a *Loan* is no longer linked to it, we can conclude that *Loan* also has a separate significance to the application apart from *Book* and, therefore, is entity independent with respect to *Book*. (See situation 2 in the discussion about (in)dependence in a 1:(N) relationship in section 4.21.4.) There are, then, two logical files.

Solution

Count two internal logical files.

References to the standard

4.21 and 5.2.a

8.2 1:(N) with dependent existence

Problem description

The normalized data model of a library application shows that there is a 1:(N) relationship between the entity types *Book* and *Loan*. A book does not have to be loaned out and, so, optionality is a factor in the relationship. If a *Loan* is made, it always relates to one *Book* (no optionality).

The business rule of this library, however, is that if *Book* is taken from the collection (is deleted), the library is no longer interested in *Loan* and, therefore, it may be deleted automatically when *Book* is deleted.

How many logical files must be identified in this case?

Discussion

A 1:(N) relationship exists between *Book* and *Loan*. According to the table in section 4.21.5, the number of logical files is determined on the basis of the entity dependence. Because a *Book* can always be deleted, and because any *Loan* linked to a *Book* may be deleted automatically with that *Book*, we can conclude that *Loan* is not significant to the application when separated from *Book*. Therefore, *Loan* is entity dependent with respect to *Book*. (See situation 1 in the discussion about (in)dependence in a 1:(N) relationship in section 4.21.4.) This means that there is only one logical file.

Solution

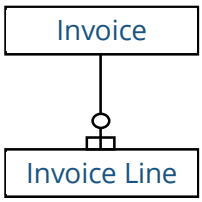
Count one logical file with two record types.

References to the standard

4.21 and 5.2.a

8.3 1:(N) with dependent existence

Problem description



The normalized data model of an invoicing system indicates that a 1:(N) relationship exists between *Invoice Header* and *Invoice Line*. The application allows users to create an *Invoice Header* first to which lines can be added later on; hence, the optionality. If users decide at a given moment to delete the *Invoice Header*, the *Invoice Lines* are also automatically deleted.

How many logical files should be distinguished here?

Discussion

Invoice Header and *Invoice Line* have a 1:(N) relationship. According to the table in section 4.21.5, the number of logical files is determined based on entity dependence. Because of the business rule that any *Invoice Lines* linked to the *Invoice Header* are deleted automatically when the *Invoice Header* is deleted, we can conclude that *Invoice Line* is entity dependent with respect to *Invoice Header*. (See situation 1 in the discussion about (in)dependence in a 1:(N) relationship in section 4.21.5) There is, then, one logical unit called *Invoice* that contains the entity types *Invoice Header* and *Invoice Line*.

Solution

Count one logical file with two record types.

References to the standard

4.21 and 5.2.a