ER and Database diagram hints & checklists

Juhani Välimäki 2002-2013. Compiled version Delta 2013-09-18 16:36

Understand the difference between ER and Database diagrams

Understand the difference by understanding the difference between conceptual modeling (real life, real business domain) and logical modeling (the chosen relational database structure based on the domain ER).

Never mix these models! Write clearly in a note or header which one you are creating. Do not put even primary key definitions to ER (they belong to database!).

Do not duplicate the attributes for future foreign keys to the referencing table in ER. So in ER diagram Employee does NOT have departmentId as an attribute. Just the association line between Employee and Department is enough in ER.

ER diagram checklist:

- ER models the "real world", so names do not have to be technical names yet. E.g. "First name" or "Street Address" COULD be used. Although then you will have to change the technical name later to "firstName", "first_name", "streetAddres", "street_address" or similar.
- Identifiers (Ids) that do not exist in the real world are USUALLY left out. E.g. student number exists in the real world and thus could be in ER diagram, but e.g. feedbackEntryId could be just a surrogate key, which means it is an artificial key not found in the real world. So a FeedbackEntry entity might be without feedbackEntryId in ER diagram.
- Entity candidates from the given description might end up being: 1) Entities 2) Roles of the entity in some associations 3) Associations between entities 4) Attributes of the entity

From ER diagram => Database diagram:

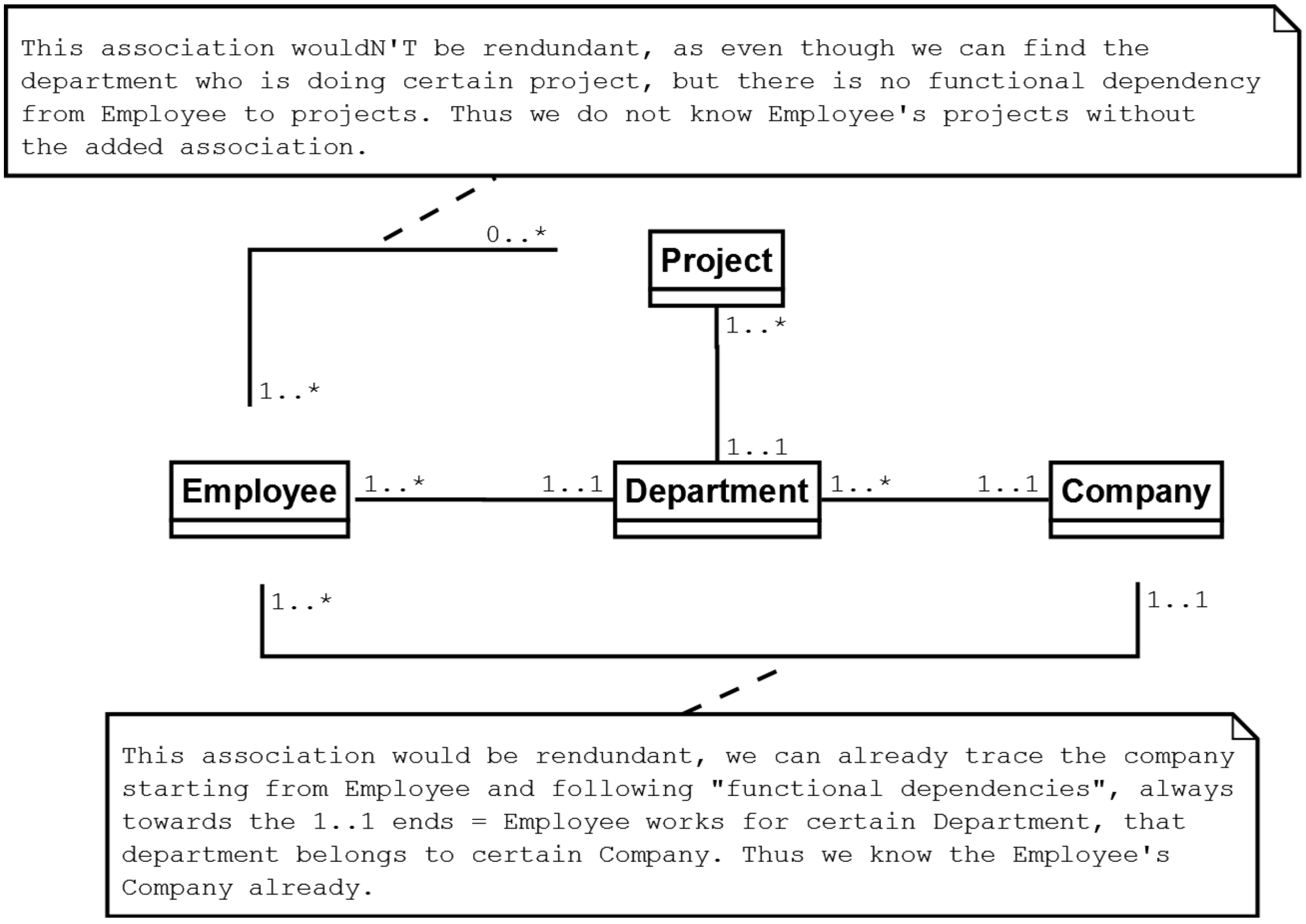
- Add the needed surrogate keys (artificial keys) like orderId. They could be auto-implemented using technologies called auto-increment, (number) sequence, auto-key, identity column, etc. etc.
- Define primary keys for each table. It might consist of many columns, sometimes even consisting of all the columns of the table
- Decompose any M-to-N relationship into 1-to-N and M-to-1 relationships by adding a new linkage table (join table) in between. The cardinalities M and N change their order as the meaning read from original entity A or original entity B must remain the same.
- Define the Foreign keys. Every single line in the diagram drawing must be replaced with a Foreign key that points to a Primary key in target table.
- Foreign keys always go to the many-end (0..*, 1..*, 2..*, sometimes even 0..1) and they refer to the one-end (1..1, 0..1) (one primary key row). If the one-end is (0..1) then the foreign key must allow NULL.
- If the referenced table has a multipart primary key, then the referencing table must have a multipart foreign key as well.

ER and Database diagram common checklist

- If there is any loop in the diagram it must represent two different logical meanings. E.g. 1) student enrolled for course implementation, 2) student has a grade from course implementation. In these cases the loop is allowed and the looped connections are not signs of redundancy.

- But in some other cases there is a risk of redundant loops. Following this logic you can avoid some of them: Even if employees work for a company, the direct association between employee and company is not needed, if you can trace back from employee to company using 1-ends (Each employee works for 1..1 department, each department belongs to 1..1 Company).

- Many-ends are not functional dependencies so they will not be enough:



This association wouldN'T be rendundant, as even though we can find the department who is doing certain project, but there is no functional dependency from Employee to projects. Thus we do not know Employee's projects without the added association.

This association would be rendundant, we can already trace the company starting from Employee and following "functional dependencies", always towards the 1..1 ends = Employee works for certain Department, that department belongs to certain Company. Thus we know the Employee's Company already.

While reading and interpreting the ER diagram/Database diagram/Class diagram

* Always read the association in both directions

* Start with the word "Each"

* Always forget the closest cardinality numbers _totally_

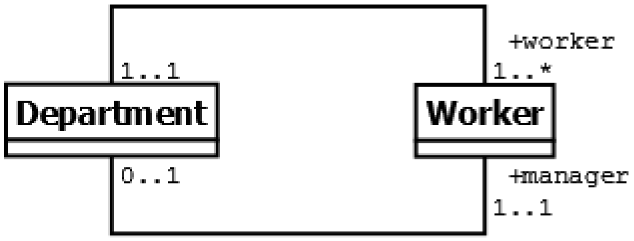
* Roles you can read both for the starting end and the final end

* Try to see what is not yet in the model. Business process knowledge and old forms and systems might help.

* (ER diagram is not the whole Conceptual data model. ER diagram + Entity descriptions + Entity attribute descriptions + possible additional business rules is the Conceptual data model)

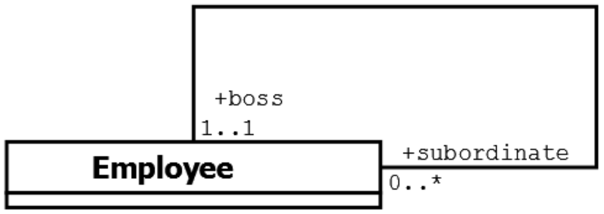
* (Neither is the Database diagram the whole Logical data model. Table and column descriptions and constraints/triggers for enforcing business rules/roles, and privileges/ and additional information is needed before we have the whole database defined)

Example 1: (Two entities and two associations between them. Employee entity has roles for both associations)



a1) Each department has 1..* employees working for it in worker role
a2) Each employee (as worker) works for 1..1 department
b1) Each department has 1..1 employee as manager
b2) Each employee might be a manager for one department (or is not manager for any)

Example 2: (Only one entity Employee, one unary association from Employee to another Employee, two roles, one at each end of that only association/relationship)



a1) Each employee has 1..1 boss (Each employee in subordinate role has 1..1 employee as a boss) a2) Each employee has 0..* subordinates (Each employee in boss role has 0..* employees as subordinates. 0 = means is not in boss role at all)

Do methodological and careful work

Always check twice! The most important thing here is to do the work carefully and methodologically. Mistakes might be costly if they are made in ER and Database diagrams and found out later while creating the application.

Juhani Välimäki
M.Sc., Senior Lecturer
Business Information Technology
HAAGA-HELIA University of Applied Sciences
Helsinki, Finland