Quick, “Rule of Thumb” Guide to Drawing Top Down ERDs

Once analysis has been carried out you should be able to list everything that happens in an organisation in relation to the proposed system (or, if you are still learning, you may have a scenario you are modelling): we will call this the scenario.

1. Identify entities (DB table):

    a) Underline all the nouns in the scenario

    b) For each noun ask the following questions:

    - Is there more than one?
    - Can I think of a suitable key?
    - Can I think of data to hold?

    c) If the answer is “Yes!” to ALL of these you have probably found an entity (be careful this is only a rule of thumb, e.g. you may have found a form or report rather than an entity)

2. Draw the relationships (for more detail on cardinality and optionality see sheet “Determining Relationships for ERDs”):

    a) Think about the DIRECT relationships that your entities have. For example: a CUSTOMER will not have a DIRECT link to their hire CAR, it will be though a BOOKING entity; so the link will be CUSTOMER – BOOKING – CAR (NOT CUSTOMER – CAR).

    b) For working out the cardinality (one to many and so on) of each direct relationship between A and B you need to ask two questions:

    - For each A how many Bs are there? (this relationship will be noted next to B)
    - For each B how many As are there? (this relationship will be noted next to A)

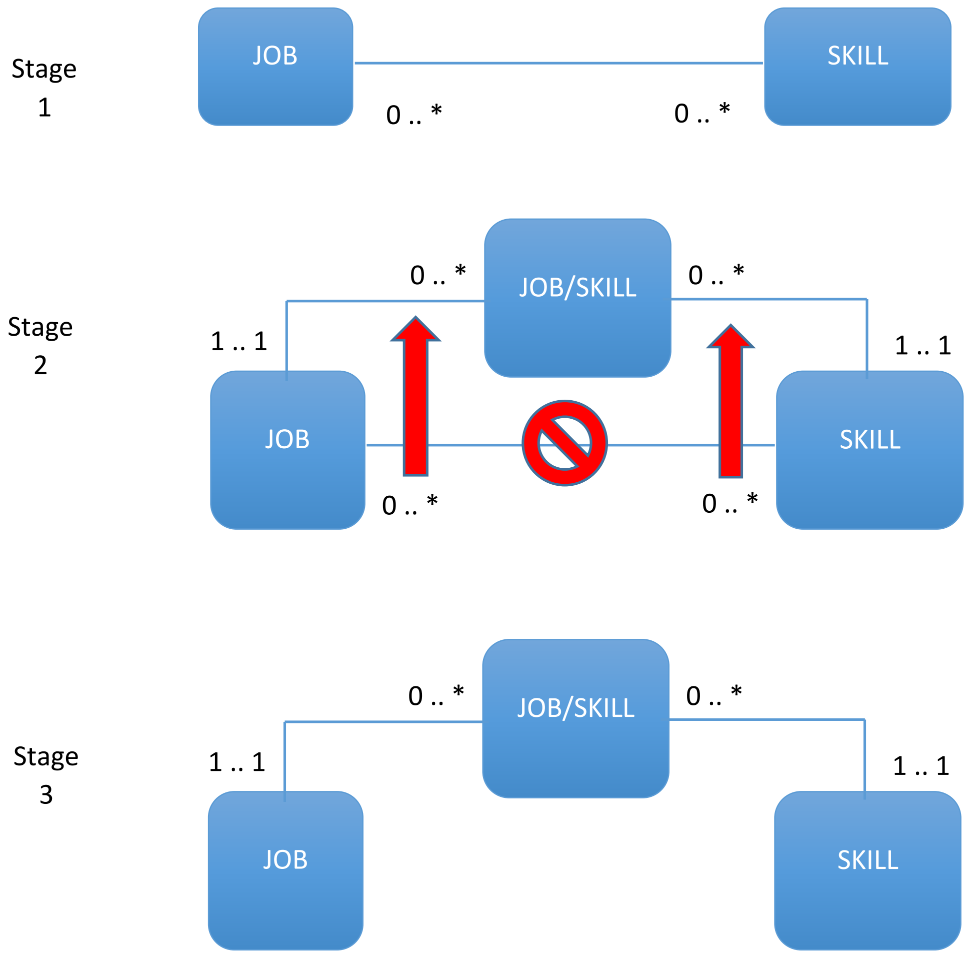
    c) For working out the optionality of each direct relationship between A and B you need to ask two questions:

    - If there is an A, need there be an B in the system?
    - If there is a B, need there be an A in the system?

3. Resolve many to many relationships:

    d) Once this is done you need to resolve (i.e. replace) any many to many relationships since these cannot be represented in relational databases.

    - A many to many between A and B is resolved by adding a link (associative) entity in between, flipping the “many ends” of the relationship so they sit next to the new entity, and adding “1 .. 1” next to the original entities. For example, in a system that allocates staff to jobs, a job that needs doing might require a number of skills, and a particular skill can be applied to many jobs. This is resolved as follows:



4. Add the primary keys. So, for example, in the above JOB and SKILL with have PRIMARY KEYS (unique identifiers) Job_ID and Skill_ID respectively. JOB/SKILL is a weak entity because it cannot be uniquely identified without referring to other entities (via foreign keys). Weak entities often have a primary key made up of two entities that they link, and that is true in this case: the primary key for JOB/SKILL will be a composite key of Job_ID AND Skill_ID.

5. Add the foreign keys. The relationships that you have need to be described in terms of foreign keys so that they can be coded into your database. This can be done by following a simple rule:

    - The many end of a relationship will extend from the primary key to grab all the composite keys and foreign keys that include that primary key

    So, wherever there is a many end grabbing a table you need to find the primary key (PK) this relates to and make that PK an FK in the table that has the many end grabbing it.

    For example, looking at the relationships above: JOB/SKILL has two many ends next to it, and so MUST have two foreign keys; Job_ID and Skill_ID. (In other words, the primary key for the JOB/SKILL table will be composed of two foreign keys.)