Basic Database Design

Gerhard Beck

Note: This paper ignores the distinction between logical and physical data models.

Design Methodology

Database design is an iterative process which begins once the project scope is known (or earlier) and ends when the users accept the design *in the form of a working application*. Users really don't care about that the database design is; they just want their data (in an accessible application). Asking the users to sign off on a logical database is similar to asking someone to choose a wife based on a personality description. She might look good on paper, but you still want to meet her first!

The basic iterative design methodology is to query the user, update the design and then get the user's feedback. In general, getting the user's feedback means letting the user tell you what you did wrong. Its far easier for users to see what you did wrong with a design than what you did right. At various points, you may get the user to sign off on various aspects of the design. That does not mean that the user can not change their minds! It just means that the users have not found any problems with your design to date. In the end, the only test the users care about is will the software get the job done (or at least make the users' life easier).

A user oriented methodology is used. Each user designs the data which that user needs. The data is normalized across all users, but each user reviews the data design for that user's applications. This is done because typically customers do not want to spend the time needed to review a full integrated data model - because what the customer usually wants is applications not paper. It is always preferable to get a customer representative to review the complete data model, but don't count on it (unless you are in a Business Process Reengineering [BPR] project).

The basic process is to do the following steps for each type of user:

1. Interview the user
2. Develop a draft data design for that user
3. Integrate the user's data design with the overall data design
4. Review the revised data design with the user
5. Repeat steps 2-4 until you and the user are satisfied

In the user interview, you will need to:

1. Find out what the user does. During this stage just take notes and get a general idea of what the user does.
2. Find out what the system must do for this user to be satisfactory. These are the prime test questions. You should eventually review these with the customer's management for scope. Scope actually has three determinations: must have, nice to have and don't do. Of course there is a continuum from must have to don't do; the really key thing is to know must have and don't do.

For some reason, the authorities can not agree on a single set of terms. Here is a translation table:

Flat File Database	Relational Database	ERD
File	Table	Entity
Fields	Columns	Attribute
Records	Rows	n/a

3. Decide which tasks the user does are in the scope of the project. If there is any doubt, assume the task is in the project. (Promise 80% of scope, plan for 100%, design for 120%)
4. Go through each task in detail getting all input and outputs for the user. Get copies of all forms the user gets data from and forms the user produces. Computer screens are forms for this purpose. The process information will be needed in developing the application (including requirements and the test plan) as well as for database design.
5. Go back to your office and write up a description of what the user does, what the inputs and outputs are and try to come up with an ERD.
6. Continue the interview by having the user review the description of the user's job.
7. Review the ERD with the user.[1]
    a) Review the entities and broad relationships. Show the user how the entities relate to the forms and other information the user deals with. You could use a list of the entities with their description for this.
    b) Review the links between the entities with the user. Use an ERD with no attributes for this purpose.
    c) Review the keys with the user. Use an entity listing with table name, description, key name and description. Try to identify real world keys (including real world assigned keys). The idea is to find out if there is some way to uniquely identify a row in the real world. Usually there is no fail-safe way to do this, but we want to know how close we can come for purpose of creating look-ups and possible business rules to enforce. If we can hide internal one-up keys from the user, we will.
    d) Review the attributes assignments with the user. Attributes must be assigned to entities. Use tables with sample data for this purpose.
    e) Review detailed attribute definitions with the user. Use a data element dictionary report for this purpose organized by entity. The attribute should include a description of how the attribute is used.

At the end of this process you should have:[2]

[1] I've never actually done all of these steps on a project. Do as many as necessary to convince yourself you understand the data enough to code the application.

[2] It' important to have all this stuff. However, for fast small project most of this will not be formalized and delivered to the user before the system is complete. The idea is not to do a paper drill but to make sure you know enough to build an application. The less you know and the more unreasonable the client, the

1. A descriptions of the tasks each user does which are included in the system. The description will include the processing done by the user which is relevant to the system.
2. A listing of inputs and outputs for the user.
3. An ERD of the user's view of the data.
4. A data element dictionary for each user view.
5. Something relating the ERD and data element dictionary to the user's inputs, outputs and processes.
6. Some sample data.
7. Notes on how each entity and attributes relates to the project scope (in, out, nice to have).
8. An ERD for the entire system.
9. A data element dictionary for the entire system.

Normalization

To get a full understanding of ERDs and normalization, read *Data Modeling Essentials* by Graeme Simsion (International Thomson Computer Press, 1994). It is good reading and actually makes sense!

A simple definition of normalization:

> Each attribute is placed in an entity where it is dependent on the key, the whole key, and nothing but the key . . . so help me, Codd![3]

First, a relatively formal definition of the rules for normalization from Fleming & Holle[4]:

First Normal Form: Reduce entities to *first normal form* (1NF) by removing repeating or multivalued attriburtes to another, child entity.

Second Normal Form: Reduce first normal form entities to *second normal form* (2NF) by removing attributes that are not dependent on the *whole* primary key.

Third Normal Form: Reduce second normal form entities to *third normal form* (3NF) by removing attributes that depend on other, nonkey attributes (other than alternate keys).

Boyce-Codd Normal Form: Reduce third normal form entities to *Boyce/Codd normal form* (BCNF) by ensuring that they are in third normal form for any feasible choice of candidate key as primary key.

Fourth Normal Form: Reduce Boyce/Codd normal form entities to *fourth normal form* (4NF) by removing any independently multivalued components of the primary key to two new parent entities. Retain the original (now child) entity only if it contains other, nonkey attributes.

---

more paper you produce to protect yourself. BUT never start coding without a formal (at least one paragraph) scope document and an ERD.

[3] *Information Engineering*, Clive Finkelstein (Addison-Wesley Publishing Company, 1990).

[4] *Handbook of Relational Database Design*, Candace C. Fleming and Barbara von Halle (Addison-Wesley Publishing Company, 1989).

Fifth Normal Form: Reduce fourth normal form entities to *fifth normal form* (5NF) by removing pairwise *cyclic dependencies* (appearing within composite primary keys with three or more component attributes) to three or more new parent entities.

Next, Finkelstein's rules for "Business Normalization":[5]

First Business Normal Form: Identify and remove repeating group attributes to another entity. The primary key of this other entity is made up of a compound key, comprising the primary key of the entity in which the repeating group originally resided, together with the repeating group key itself, or instead another unique key based on the business needs. The name of the new entity may initially be based on a combination of the name of the repeating group and the name of the entity in which the repeating group originally resided. It may be later renamed according to its final attribute content after business normalization is completed.

Second Business Normal Form: Indentify and remove attributes into another entity which are only **partially** dependent on the primary key and are also dependent on one or more other key attributes, or which are dependent on only **part** of the compound primary key and possibly one or more other key attributes.

Third Business Normal Form: Identify and remove into another entity those attributes which are dependent on a key other than the primary (or compound) key.

Completing the list from Simsion:[6]

Boyce Codd Normal Form (BCNF): Every determinant must be a candidate key.

Fifth Business Normal Form: Keep splitting tables until:

1. the only splits left are trivial, or

2. any further splitting would lead to tables that could not be joined to produce the original table

Does all this sound like too much trouble for the real world? It is. I have never seen anyone apply the rules of normalization in a text-book fashion. Normally you just start with an ERD, add the entities and then put the attributes where they belong. Rule of normalization are useful for demonstrating that you know appropriate buzz words and if you are losing an argument about where to put something on the ERD.

Although you typically do not approach a data design by formally applying the normalization rules, ***ALL DATABASES*** MUST ***BE IN THIRD NORMAL FORM***. Usually, all database are done to fifth normal form (although usually no one knows how to prove it). And an occasional processing column slips into the recipe to make the system actually work (which is the entire goal anyway).[7]

---

[5] *Information Engineering*, Clive Finkelstein (Addison-Wesley Publishing Company, 1990).

[6] *Data Modeling Essentials*, Graeme Simsion (International Thomson Computer Press, 1994)

[7] For an interesting way to approach data modeling, see Al Foster's "A New ERA for Data Modeling" in Appendix A of the *Microsoft SQL Server Training Volume 2* (Microsoft Press, 1996, Version 6.5).