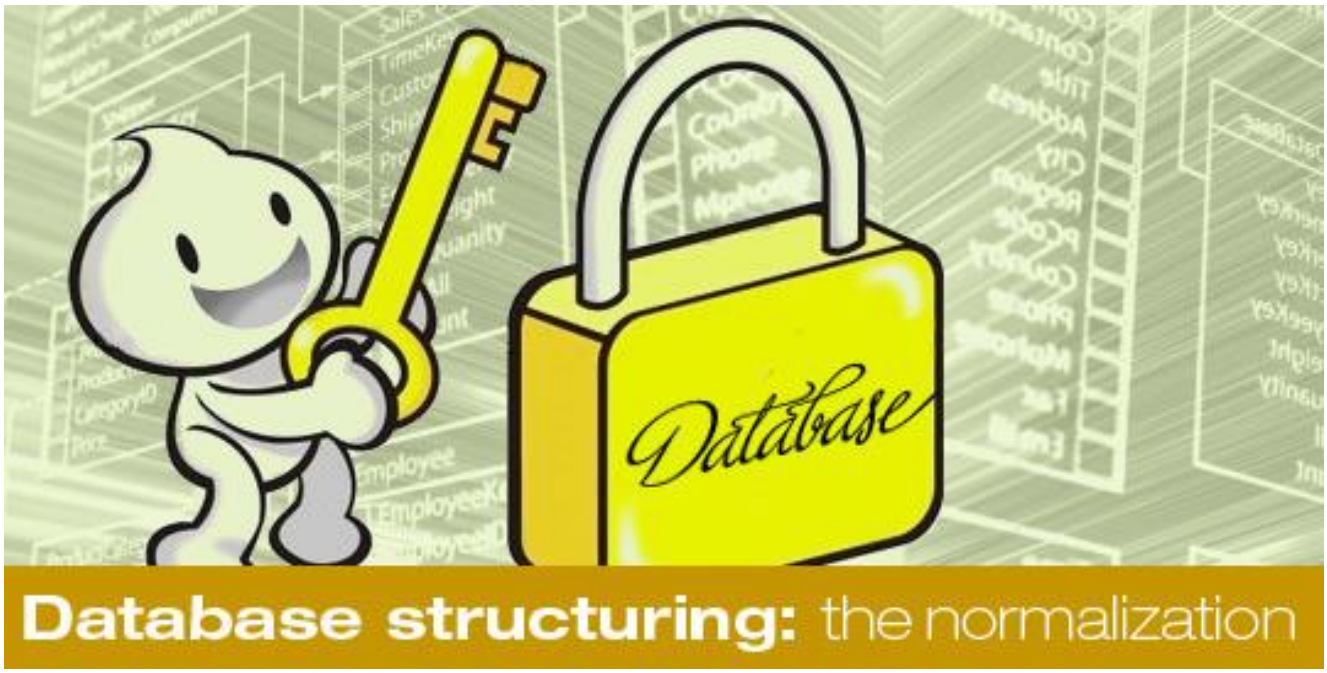
Lecture note

Table Design and Normalisation

Mads Brix and Finn Nordbjerg
English version by Kis B. Hansen,

UCN 2010

Revised by KNOL 2019



Informal and formal quality objectives of table design

The following describes the requirements for tables in the relational model. The requirements are described both informal and formal based on the normalization theory. It also contains an example of a bad table design that illustrates the problems that can arise.
The text is written based on Chapter 14 of the applied database book [Elmasri].

Prerequisites

Assuming that you are familiar with the concepts of the relational model, i.e. tables, attributes, primary and foreign keys etc...

Informal rules for table design

By following a few simple rules, also achieved by relying on ER models or object models (class diagram), usually a good table design are obtained. These rules are:

- The meaning of a table should be easy to understand
    - There should not a mix of information on different things in the same table.
- Avoid redundancy
- Minimize the number of null values

Redundancy

Redundancy occurs when the same fact is recorded several times.
Example: It is redundant to register the city name every time the postal code is recorded.
Redundancy is problematic for two reasons, first, it takes up more space than is required, and there are problems with update anomalies.

Update anomalies

A bad table design causes problems collectively known as update anomalies and is divided into insertion, deletion and modification anomalies. To explain these anomalies, we use an example.

An example

As an example, the following table which is (part of) a database to record information in a company. The table Employee_Project represents information about the employees, which projects are in progress and who have worked in the project and for how many hours. The primary key is composed of (employee_no, project_no).

Employee_Project

employee_no	project_no	projectname	hours	firstname	lastname	dateofbirth	education	monthly salary	department	department name
19	23	Tower block Living joy	103	Peter	Jensen	240165	Bricklayer	20000	1	New construction
19	45	Kindergarten Play Happy	10	Peter	Jensen	240165	Bricklayer	20000	1	New construction
2	23	Tower block Living joy	43	Anne	Andersen	230245	Secretary	19000	1	New construction
23	23	Tower block Living joy	250	Svend	Eriksen	111253	Carpenter	20000	2	Renovation
89	45	Kindergarten Play Happy	81	Arne	Jensen	071271	Carpenter	20000	1	New construction
145	88	Danmarksgade 23	93	Ingrid	Larsen	300369	Bricklayer	20000	2	Renovation

Anomalies

- Modification anomalies have trouble remembering to update all copies of redundant data. In the example, in order to change the name of department 1 you have to change four instances in the table.
- Insertion anomalies are problems associated with the deployment of new data. In the example, it would be impossible to create a new project without having at least one employee to work on it since employee_no is part of the primary key.
- Deleting anomalies are problems related to the deletion of data. In the example, deletion of a project which is the only project a given employee is assigned to, leads to information about the employee also are deleted. E.g. the deletion of project number 88 causes also the employee number 145 to be deleted from the database.

Formal rules of table design - normalization

Normalization is a process in which the tables being worked on are examined to check if they meet some rules formulated in the individual normal forms. The aim is to ensure the quality of table design, including by removing redundancy.
Formal there are 6 normal forms, 1st, 2nd, 3rd, BC, 4th and 5th normal form. The weird naming is rooted in historical reasons, BC normal form (Boyce-Codd Normal Form) was formulated after the introduction of 4th and 5th normal form. Normal Form is often shortened to NF, e.g. Boyce-Codd Normal Form is called BCNF.
A statement of 4th and 5th normal form is outside the scope of this text[1], a detailed description of 4th and 5th NF is found in chapter 21 in *[Date]*

[1] Claim: a table design based on a thoroughly prepared EER model will never experience problems with 4th and 5th normal form.

Functional dependencies

Both the 2nd, 3rd and Boyce-Codd NF is based on the concept of functional dependencies (FD). A functional dependency describes a relationship between two sets of attributes, where the value of one set of attributes (determinant) determines the value of the second set of attributes. A functional dependency is usually written X -> Y, wherein X and Y are one or more attributes. The functional dependence applies to Zip -> City. It is functional dependencies, which cause redundancy, that establish normal form rules for the functional dependencies that must be in a table. It is important to realize that functional dependencies reflect conditions from reality, so in the end we must investigate reality in order to find out whether a given rule (and thereby a functional dependency) applies .
Note in a table there will always be functional dependencies between the primary key (and possibly other candidate keys) and each of the other attributes.

Boyce-Codd Normal Form (BCNF)

BCNF says:

- **For all functional dependencies, it must be the case that the determinant is a candidate key.**

If a table is in BCNF is also in 1st, 2nd, and 3rd NF and can therefore in principle be based on this. For the sake of understanding (and general education), it is necessary to look at 1st, 2nd and 3rd NF. Note also that BCNF is related to how a good table should look like, while the 1st, 2nd and 3rd NF relate to what may not be a good table.
A more informal formulation of BCNF is that the data you store in a table must depend **on the key, the whole key and nothing but the key ('So help me Codd').**[2]

The Relational Model

Normal forms are understood in the context of the relational model, which was stated as a prerequisite for for reading this document. To use normalization techniques on relational databases, it must be ensured that the tables are spceified such that they conform to the relational model, which is based on **mathematical sets**. Since the relational model spcifies a *relation* as a *set* of *tuples*, **rows in tables must be unique**, as sets do not allow duplicate values. This uniqueness is guaranteed by identifying a **primary key**. Also, it must be understood that **no guarantees about the sequence of rows** are provided, as *sets* are *unordered*. These restrictions on tables thus implicitly form the basis of the concept of normal forms, however, they are not mentioned explicitly in either of them.

1st Normal Form (1NF)

For a table to be in 1NF it may only contain simple attributes, ie no multi-valued or complex attributes. 1NF is a relic from the old days, since a table in the relational model always is in 1NF. 1NF has become a part of the definition of the relational model. It is mostly included for historical reasons.[3]

[2] It may be noted that Codd is the father behind the relational data model, and readers with a penchant towards cheap American courtroom series will then maybe find this remark funny.
[3] It should be noted that it is discussed changes to the definition of the relational model, as to allow multi-valued and possibly compound attributes. This will mean that the tables will no longer need to be in 1NF, but provides some other advantages, which it is beyond the scope of this text to explain.

2nd Normal Form (2NF)

2NF is related to partial functional dependencies. A partial functional dependency is a functional dependency, where only a part of the determinant is of importance. If the funtional dependency isn't partial, it is full. In a full functional dependency there can not be remove any attributes from the determinant without the functional dependency is repealed.
A partial dependency exists in the example where it applies to *projectname* that does not depend on the entire key *(employee_no, project_no)*, but only of *project_no*. The functional dependency *(employee_no, project_no) -> projectname* is a partial dependency. Note - this functional dependency of course violates BCNF. Informally, you could say that *projectname* is only a property of *project_no*, and not of the entire key.
A table is in 2NF when

- it is in 1NF
- all non-key attributes are fully functionally dependent on a master key

3rd Normal Form (3NF)

3NF is related to transitive functional dependencies. A transitive functional dependency is a functional dependency that goes via a different set of attributes.
In the example there would be a transitive functional dependency between *(employee_no, project_no)* and *departmentname* going via *department*.
A table is in 3NF when

- it is in 2NF
- no non-key attribute is transitively functionally dependent on a candidate key.

Differences between 3NF and BCNF

There are in fact tables which are in 3NF without being in BCNF. This is primarily due to the historical formulation "all non-key attributes ..." which says that if an attribute is part of a key, it shouldn't meet any requirements. If in the example it is assumed that more projects can not have the same name, there will also be a candidate key consisting of *(employee_no, project_no)*. Thus, the functional dependency *project_no -> projectname* no longer is a problem since the *projectname* is a key attribute. However, there are clear redundancy problems, such as the fact that *project_no* 23 called "Tower block Living joy " is registered three times.
Note that for a table to be in 3NF without being at BCNF, there must be overlapping candidate keys.

How to normalize?

First, identify the functional dependencies that violate BCNF.
For each of these functional dependencies (X -> Y) split up the table in order to create a new table consisting of all attributes (both right and left side, X and Y) of the functional dependency that causes the problem. In the old table the determinant (X) is left, which now becomes a foreign key, pointing to the new table.
This process must be repeated for all problematic functional dependencies, also for tables created as a result of the normalization of the other tables.

Controlled redundancy

It should be mentioned that in certain situations there can be controlled redundancy to optimize the speed of queries. This can be done in situations where the functional dependency reflects a relationship that typically can not be changed (eg. Zip -> City) and where you choose to use the extra space to save some time. It should be emphasized that controlled redundancy should only be introduced in exceptional cases and only after normalization has removed all redundancy.

Literature

[Elmasri]
Elmasri/Navathe
Fundamentals of Database Systems, 2nd ed.
The Benjamin/Cummings Publishing Company
ISBN 0-8053-1753-8

[Date]
C.J. Date
An introduction to Database Systems, Volume 1, 5th ed.
Addison-Wesley Publishing Company
ISBN 0-201-52878-9