Normalization of Databases

The objective of this lecture is to learn about rules of good design. We have already referred to these rules in our previous lectures, but here you can see them listed in one place in an organized fashion.

The purpose of design is to:

- Increase efficiency. Information should be organized in such a manner as to foster efficiency:
    1. Reduce redundancy. Information that gets repeated in each record should be separated and put into a different table so that we reduce repetition. For example in a table of encounters, patient ID replaces patient's name, contact information and demographic so that we do not need to repeat this information in each record of a visit.
    2. Reduce missing data entries. Information that are logically impossible are transferred to another table so that we are not forced to leave it blank. For example, since a pregnant male is not possible, we would like to have pregnancy information kept in a different table than gender information so that we do not need to enter banks for pregnancy field for males.
- Allow users access to data without knowing location of data. It should be possible for a user to search for a piece of information in one field and not have to guess what other fields may contain the same information.

Design Principles

To achieve these objectives, mathematicians have developed a set of rules that if they are followed result in more efficient and easier to use databases. These rules are not a matter of design preference but a requirement for effective and efficient database.

Rule 1: Each table should correspond to a single entity

The first rule we would like you to remember is that each table correspond to a single entity or a single relationship. You cannot have a table containing information about both the patients and the providers. Doing so unnecessary adds confusion about where data might be found. It also forces us to add provider information when adding patient information, increasing the redundancy of the data. Always ask yourself what is the table about. If you can tell it is about one entity and one entity alone, then you are on a good start to design of databases.

Rule 2: Rows in the table should correspond to individual occurrences of the entity

The second rule makes sure that each row in the table is an occurrence of the entity and nothing else. . A row in a table often is referred to as a record. Every record should be an occurrence of the entity. So for example, in the patient table, each record should be one patient. In the provider table, each record should be a provider. This seems very simple rule but I have seen it violated. Consider a table of visits. Each record should be about one visit. I have seen occasions where multiple records are given to the same visit or some records in the visit table are about the provider's characteristics. These are not reasonable ways to construct a table. Always check the rows of the table to see each is an example of the entity.

Rule 3: Primary key should uniquely identify the individual occurrences of the entity or relationship

The third rule makes sure that the primary key uniquely identifies the occurrences of the entity. If we know the primary key, we should know which row in the table is involved. A primary key should not be used for two rows in the same table. For example, first name and last name are often insufficient for uniquely identify a patient as many patients have same names. But a combination of first, middle, last name and date of birth might uniquely identify the patient. A social security number uniquely identifies the patient.

Rule 4: Non-key fields should be facts about the occurrence identified by the primary key

Rule 4 is to make sure that all fields in the table are facts about the primary key. If the fact is about something else, it does not belong to the table. For example, a name belongs to a table about patients. It is a fact about the primary key that reflects individual patients. Likewise, date of birth is another fact about the patient and belongs to the table. But is diagnosis a fact about the patient? At some level yes, but in reality it is not just about the patient. A patient may have many different diagnoses at different times, so in reality diagnosis is a feature of the patient's visit and not really the patient. It is not a persistent fact about the patient. Therefore, diagnosis belongs to the table of visits and not to the table of patients.

Rule 5: Each fact should be presented only once in the database

Rule 5 is to make sure that we do not present a fact more than once anywhere in the database. Many students think that data should be presented in the sequence found or reported so that it is convenient to do use the data. But Standard Query Language makes all data in the database available at all times. There is no need to duplicate the information. For example, it is not reasonable to list the name of the patient's provider in the visit table, when the name is available in the provider table. We collect and often report who took care of which patient, but the information gets stored in different places. It is comfortable to have everything we want in a table -- it feels that the data is accessible. But in reality we are talking of a machine and all data from all tables are accessible. Therefore, there is no need to duplicate the data. .

First Normal Form

Normalization is the process of applying principles of design to data structures so that they conform to our expectations. In this lecture we discuss three additional rules that are used as part of putting a database in Normal form.

Household			
Address	Zip	First person	Second person
1319 Ozkan	22101	Jim	Jill
14 Yates	22112	George	Janet

Table 1: Violation of first Normal form

Take a look at the table in Table 1 titled households. It has four fields, street address, zip code, first person in the household, and second person in the house hold. Is this a reasonable table structure? Think about it what could go wrong. I see two problems in this table structure. First, user needs to know how occupants are listed. How would anyone know if they are the first person or the second person to be listed? Second, the listing is not extensible, what if the household had more than two people? Where would we put their name?

We can avoid the error we just talked about by putting tables in the first Normal form. A table is said to be in the first Normal form if and only if all fields contain only atomic values and there are no repeating fields in a row. By atomic we mean that it is not composite of two facts, e.g. total charges is a composite of hospital and outpatient charges and cannot be a field in a table in first Normal form. By non-repeating fields I mean that the same information should not be stored in two different fields. In the previous table household residents were listed in two fields. The first Normal form does not allow this. For another example, if the patient has 5 diagnoses during a hospital visit, it is not reasonable to list them as five fields in the same table. In these circumstances, the recommendation is to list a code that points to a different table that lists the diagnosis as 5 separate records in the table. The point is that the first Normal form does not allow any repeating fields.

Household			
Number	Street	Zip	Name
1319	Ozkan	22101	Jim
1319	Ozkan	22102	Jill

14	Yates	22112	George
14	Yates	22112	Janet

Table 2: Solution for violation of first Normal form

Table 2 provides the revision of the household table. We have put this table in first Normal form. The composite field address has been separated into two more atomic fields: street number and street name. The repeating fields have been listed as one field, requiring the need to add an additional record for each resident of the household. You can, of course, avoid the row duplication by separating the Household table into two tables, one containing the resident names and the other containing the address.

Second Normal Form

The discussion of the second Normal form requires us to introduce the concept of dependency. An Attribute Y is Functionally Dependent on an Attribute X, if a Value for X Determines a Unique Value for Y. X may be a Set of Attributes or a single attribute. We show this as an arrow going from X to Y and read it as X determines Y.

For example, we could say that an employee number functionally determines the employee name. This is the same as to say if we know the employee number we will know his name. The reverse is not always true. Knowing an employee name is not sufficient for guessing the employee number. So “Employee name” is functionally dependent on employee number but not vice versa.

We need one more definition before we can state the second Normal form. We say that an attribute is fully functionally dependent on another set of attributes, if and only if it is not dependent on any smaller subset of attributes. An employee number may be fully functionally dependent on employee name and date of birth as it is not dependent on either the name or date of birth by themselves.

Here is another example: employee name is not fully functionally dependent on the attributes employee number and department as it is functionally dependent on just employee number. Just knowing the employee number by itself is sufficient to know employee name. We do not need the additional information about the department. In essence, a fully functional dependency is the smallest subset need to establish functional dependency.

Now we are ready to state the second Normal form. A Table is in Second Normal Form if and only if facts in the table are fully functional dependent on the primary key. This is a very important principle. It says that the primary key is the minimum subset necessary to determine each fact in the table.

Household				
Number	Street	Zip	Name	Employed
1319	Ozkan	22101	Jim	Yes
1319	Ozkan	22102	Jill	No
14	Yates	22112	George	Yes
14	Yates	22112	Janet	No

Table 3: Violation of second Normal form

Consider the table of Households in Table 3. Is it in violation of the second Normal form? In the table there are five fields, street number, street name, zip code, name and employment. This is a typical table that might be constructed from a survey of residents of a household regarding their employment.

Household			
Number	Street	Zip	Name
1319	Ozkan	22101	Jim

Employment	
Name	Employed
Jim	Yes

1319	Ozkan	22102	Jill
14	Yates	22112	George
14	Yates	22112	Janet

Jill	No
George	Yes
Janet	No

Table 4: Solution to violation of second Normal form

Note the problem in Table 3, employment is not fully functionally dependent on the household address. Employment is a fact about the residents of the house. Therefore it should be separated into a different table. Each fact in the table should be about the primary key and fully functionally dependent on the primary key (see Table 4). The example we have used often by now, whether diagnosis belongs to a table of patients can now be addressed more formally. No. Diagnosis cannot be in the patient table because it is fully functionally dependent on the patient as well as date of visit. The primary key in the patient table does not have the additional date of visit and therefore diagnosis cannot be put in this table.

Third Normal Form

A Table is in Third Normal form if and only if there are no combination of strictly informational fields (not primary key fields) that determines the value of another field.

Clinic Visits				
Name	Employer contributions	Charge	Co-pay	Net deductible
Jim	0	54.65	5	Not yet
Jill	240.23	250.23	10	Yes
George	107.5	112.5	5	Yes
Janet	40	45	5	Yes

Table 5: Violation of third Normal form

Consider the table about clinic visit charges in Table 5. Is the table in violation of third Normal form? For simplicity consider that what is not paid by employer is paid by employee. Think it through. Is there a combination of facts in the table that functionally determines another fact? Yes, three of the four fields (charges, co-pay, whether patient has met deductible and employee contributions) will tell us the fourth. So, one piece of information is not needed. Depending on the use we want to put the table to, we could drop the last two fields and keep only employers contribution and charges. We can deduce that the rest is paid by employee. If a piece of information can be computed from other facts in the table, it should not be part of the table.