Normalization

Normalization is a process for determining what attributes go into what tables, in order to reduce the redundant copies (i.e., unnecessary duplicates).

In order to understand this process, we need to know certain definitions:

• **Functional Dependency:**

  Attribute B is functionally dependent on Attribute A, (or group of attributes A) if for each value of A there is only one possible value of B. We say A --> B (A decides B) or (B is functionally dependent on A). Notice this dependent relation is different from the one in real world, such as parent-children dependent relation, why?

Consider the following tables – and list the functional dependencies

<table>
<thead>
<tr>
<th>Table</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>StNum, SocSecNum, StName, age</td>
</tr>
<tr>
<td>Emp</td>
<td>EmpNum, SocSecNum, Name, age, start-date, CurrSalary</td>
</tr>
<tr>
<td>Registration</td>
<td>StNum, CNum, grade</td>
</tr>
</tbody>
</table>

Dependency defines whether the data of the dependent can be retrieved via this relation.

**Rule 1:**

Dependency is defined on the behalf of information search and storage in database, not on importance or any other dependent relation in real life. Therefore, in this class, we have children ➔ parent, not parent ➔ children.

• **primary key:**

  1. A single attribute (or minimal group of attributes) that functionally determine all other attributes.
  2. If more than one attribute meets this condition, they are considered as candidate keys, and one is chosen as the primary key

**Rule 2:**

A key contains a unique value for each row of data.

• **candidate keys**

  for the Student table are ___________________________

  for the Emp table ___________________________

  for the registration table ___________________________
For the tables with more than one candidate key, what should be chosen for the primary key?

**Determining functional dependencies**
Consider the following table $\text{Emp(Enum,Ename,age,acctNum,AcctName)}$, what are the functional dependencies? Can we ensure?

In order to determine the functional dependencies we need to know about the “real world model” the database is based on. For example: does each employee work on each single account or can there be multiple accounts assigned? Can an account have more than one employee assigned to it? …

Suppose we know:  
1) each Account is assigned to one employee,
2) an employee can work on more than one Account

Now we can determine the functional dependencies. They are:

Suppose we have this information for a new housing development $\text{Houses(LotNum, Address, builder, model, upGradeNum, upGradeName, squareFeet)}$

What are the functional dependencies?

We can’t determine them, unless we know more about the “real world situation.”

Suppose we know:
1) Each builder has a list of models they offer, but each model is unique to a specific builder. For example:
   Builder A may offer 3 models: the Ashley, the Brigham, and the Cambridge
   Builder B may offer 2 models: the Axton and the Braxton
2) Each model always has the same amount of square feet
3) Each specific house comes with a list of upgrades, for example,
   house with LotNum 101 might have
   - upNum 652 granite countertops
   - upNum 639 hardwood floors
   - upNum 622 finished basement

   house with LotNum 122 might have
   - upNum 639 hardwood floors
   - upNum 622 finished basement

Now what are the functional dependencies?

**1st Normal form (1NF)**

According to Date's definition of 1NF, a table is in 1NF if and only if it is "isomorphic to some relation", which means, specifically, that it satisfies the following four conditions for a valid table:

1. There's no top-to-bottom ordering to the rows.
2. There's no left-to-right ordering to the columns.
3. There are no duplicate rows.
4. All columns are regular [i.e. rows have no hidden components such as row IDs, object IDs, or hidden timestamps].

Plus

5. Every row-and-column intersection contains exactly one value from the applicable domain (and nothing else).


Violation of any of these conditions would mean that the table is not strictly relational, and therefore that it is not in 1NF.

Simply, a table is in first normal form if there is no any multi-value record.

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>First Name</th>
<th>Surname</th>
<th>Telephone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Robert</td>
<td>Ingram</td>
<td>555-861-2025</td>
</tr>
<tr>
<td>456</td>
<td>Jane</td>
<td>Wright</td>
<td>555-403-1659</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>555-776-4100</td>
</tr>
<tr>
<td>789</td>
<td>Maria</td>
<td>Fernandez</td>
<td>555-808-9633</td>
</tr>
</tbody>
</table>

Repeating groups across rows - This will violate the policy of keys!
Repeating groups across columns - The designer might attempt to get around this restriction by defining multiple Telephone Number columns:

- Difficulty in querying the table. Answering such questions as "Which customers have telephone number X?" and "Which pairs of customers share a telephone number?" is awkward.
- The storage redundancy is also the issue. See the wasted space of No. 2 and 3 for Robert and Maria in the below table.

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>First Name</th>
<th>Surname</th>
<th>Tel. No. 1</th>
<th>Tel. No. 2</th>
<th>Tel. No. 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Robert</td>
<td>Ingram</td>
<td>555-861-2025</td>
<td></td>
<td></td>
</tr>
<tr>
<td>456</td>
<td>Jane</td>
<td>Wright</td>
<td>555-403-1659</td>
<td>555-776-4100</td>
<td>555-403-1659</td>
</tr>
<tr>
<td>789</td>
<td>Maria</td>
<td>Fernandez</td>
<td>555-808-9633</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Restriction of the number of telephone numbers per customer to three. If a customer with four telephone numbers comes along, we are constrained to record only three and leave the fourth unrecorded. This means that the database design is imposing constraints on the business process, rather than (as should ideally be the case) vice-versa.
- Inability to enforce uniqueness of Customer-to-Telephone Number links through the RDBMS. For instance, Customer 789 might mistakenly be given a Tel. No. 2 value that is exactly the same as her Tel. No. 1 value.
- The designer might, alternatively, retain the single Telephone Number column, by making it a string of sufficient length, to accommodate multiple telephone numbers...
numbers. Then, Telephone Number heading becomes semantically woolly, as it can now represent either any telephone number, or a list of telephone numbers. A query such as "Which pairs of customers share a telephone number?" is more difficult to formulate.

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>First Name</th>
<th>Surname</th>
<th>Telephone Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Robert</td>
<td>Ingram</td>
<td>555-861-2025</td>
</tr>
<tr>
<td>456</td>
<td>Jane</td>
<td>Wright</td>
<td>555-403-1659, 555-776-4100</td>
</tr>
<tr>
<td>789</td>
<td>Maria</td>
<td>Fernandez</td>
<td>555-808-9633</td>
</tr>
</tbody>
</table>

Converts your tables to 1st Normal forms.

a) Determine the primary key, and the corresponding key dependencies.
b) Remove each multi-value column to a new table.
c) Copy the relevant key, which decides that repetition column, from the original table to the new table.
d) Determine the new key in that new table. Usually, it implies a switch of the key role (i.e., make the multi-value column the key in the new table).

- The repetition has two different meanings, one is multi-record and the other is multi-value. When the repetition of data appears in key column, its unique appearance can be identified as the whole key value, with the help from other key (column) attributes. Such repetition is called multi-record, appearing in different rows in the table. When the repetition of data appears in non-key column, it is called multi-value. Such a repetition cannot be distributed to other rows due to the restriction of the key definition. Therefore, the above problems need to be solved.
- For the above phase b), refer to materials available at http://www.1keydata.com/database-normalization/first-normal-form-1nf.php
- For the above phase c), fine a single (or a group of) key attribute(s) that decides the repetition attribute in the original table. Copy this key attribute with each removing in the phase b). Now in the new table, you must obtain this key dependency as shown in http://www.1keydata.com/database-normalization/first-normal-form-1nf.php
- The new table needs a new key! This is done in phase d). For a simple case, this can be done simply, by switching the key roles in that key dependency. Otherwise, the key could be the entire list of attributes. See the class discussion on this part of materials.

Assume the original table as shown in the above is:

```
Customer(ID, FN, LN, TN)
```

a) What is the primary key?
b) The Customer table would be broken up into two tables. They would be

```
Customer_Name (  
Customer_Phone (  
```

c) What is the primary key in each table?

d) What is the relationship between the two tables? Which tables have foreign keys pointing to the primary key of another table (a foreign key is an attribute in one table which matches the primary key in another table.)

Normalization Exercise 1:

For each of the following tables
- Convert to a group of tables in first normal form
- Show the primary key of each table
- Show the foreign key of each table (and what table it points to)

```
Student(StNum, StName, SocSecNum, age, clubs, awards)  
Faculty(FacNum, SocSecNum, rank, committees, papers-written)  
```

2nd Normal Form (2NF)

2NF was originally defined by E.F. Codd in 1971. A table that is in first normal form (1NF) must meet additional criteria if it is to qualify for second normal form. Specifically: a 1NF table is in 2NF if and only if, given any candidate key K and any attribute A that is not a constituent of a candidate key, A depends upon the whole of K rather than just a part of it.

- A 2NF must be a 1NF first!
- A 1NF is in 2NF if and only if all its non-prime attributes are functionally dependent on the whole of every candidate key.
- A 1NF without any composite candidate keys (candidate keys consisting of more than one attribute) is automatically in 2NF.
Consider the following table - Where each project has many employees and each employee works on many projects. Sample data is shown.

<table>
<thead>
<tr>
<th>ProjectInfo(PrjNum, PrjName, budget, EmpNum, EmpName, HrsWorked)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P22   Cyclone   50000  E1001   Joe     12</td>
</tr>
<tr>
<td>P22   Cyclone   50000  E2002   Pat     50</td>
</tr>
<tr>
<td>P21   IMB       20000  E3003   Ed      40</td>
</tr>
<tr>
<td>P21   IMB       20000  E2002   Pat     30</td>
</tr>
<tr>
<td>P21   IMB       20000  E1001   Joe     70</td>
</tr>
</tbody>
</table>

a) What problems do you see keeping all this data in one table?

b) What are the functional dependencies?

c) What is the key? (minimum group of attributes that determine all other attributes)

d) What are the partial dependencies? Is it in 2\textsuperscript{nd} N.form?

Converts your 1NF to 2\textsuperscript{nd} Normal forms.

\textbf{a) Remove each column partially dependent on the original key to a new table.}
\textbf{b) Copy the key to the new table (the same in phase-c of 1NF).}

- For the above phase b), copy the key attribute in that partial dependency to the new table as we do it in phase-c in 1NF. But unlike 1NF, the key role is kept as the original without any change.

In the above example, we would create 2 new tables, from the partial dependencies

\textbf{Project(PrjNum, \_)} \quad \textbf{Emp(EmpNum)}

The original table would include the key and all attributes dependent on the whole key

\textbf{Assignments(\_)}

Normalization Exercise 2:

Consider the following table- With some sample data shown (assume each order can contain various numbers of different products, but is placed by one customer on one date), and

- determine the functional dependencies
- determine the primary key
- determine if there are any partial dependencies
Normalization Exercise 3:

Consider the following table involving course info and the grades given in those courses, and

- determine the functional dependencies
- determine the primary key
- determine if there are any partial dependencies
- convert to 2nd N.F.  Show all keys (primary and foreign)

Courses (Cnum, Cname, credits, stuNum, stName, stAge, semester, grade)

Normalization Exercise 4:

Given the follow table $T( A,B,C,D,E,F,G)$

Suppose we have the following dependencies:
$A+B\rightarrow C; A+B\rightarrow F; A\rightarrow D; A\rightarrow E; B\rightarrow G$

a) What is the primary key? (the minimum attributes that determine all the other attributes)

b) What are the partial dependencies?

c) Convert to set of tables in 2nd N.F.  Indicate primary and foreign keys.

3rd Normal Form

The third normal form (3NF) is a normal form used in database normalization. 3NF was originally defined by E.F. Codd in 1971.[1] Codd's definition states that a table is in
3NF if and only if both of the following conditions hold:

- The relation R (table) is in **second normal form (2NF)**
- Every non-prime attribute of R is non-transitively dependent (i.e. directly dependent) on every **candidate key** of R.

A **non-prime attribute** of R is an attribute that does not belong to any candidate key of R.[2] A transitive dependency ([http://en.wikipedia.org/wiki/Transitive_dependency](http://en.wikipedia.org/wiki/Transitive_dependency)) is a **functional dependency** in which \( X \rightarrow Z \) (\( X \) determines \( Z \)) indirectly, by virtue of \( X \rightarrow Y \) and \( Y \rightarrow Z \) (where it is not the case that \( Y \rightarrow X \)).[3]

Simply saying, a table is in **3rd N.F. if**

- it is in **2nd N.F and**
- every dependency will involve the keys.

Consider the following table: \textbf{Student(SocSecNum, StNum, name, age)}
What are the dependencies?

Consider the following table (where each employee works in one department):

\textbf{Emp(empNum, SocSecNum, age, deptNum, deptName)}

What is the primary key?

What are the dependencies?

Is it in **2nd N. Form? Explain!**

Is it in **3rd N. Form? Explain!**

What problems do you see using the employee table above?

**Converts your 2NF to 3rd Normal forms.**

\textbf{Keep the 2NF, copy any non-key dependency to a new table, and reset the key from this original dependency in that new table.}

- For the transitive relation \( Y \rightarrow Z \) in the 2NF table \((X, Y, Z)\) where key \( X \) implies the existence of relations \( X \rightarrow Y \) and \( X \rightarrow Z \), its existence is invisible because we never expect any relation between non-key columns. Here, we use 3NF to copy this relation out in a new table, highlighting its existence in the entire system.
- Do not worry about the duplicate that still exists in the original table after such a copy process. That redundancy can be handled easily after the use of inheritance. In many existing 3NF methods, some relation information may be lost or destroyed because they do not use sufficient-and-necessary conversion.

Here is an example of 2NF \( T (A, B, C, D, E, F, G) \) with dependencies
A→B,C,D,E,F,G; D→E,F; C→G

Create two new tables from the dependencies D→E,F, C→G
T1(D,E,F) T2(C,G)

Keep E, F, and G in Table T (since they depend on the primary key)
T(A,B,C,D,E,F,G)

We now have 3 tables—with the following primary and foreign keys
T(A,B,C,D,E,F,G) T1(D,E,F) T2(C,G)

In the above example: Emp(empNum,SocSecNum,age,deptNum,deptName)

the candidate keys are _______________________

and the dependency ______________________ doesn’t involve these keys

We create a table from the dependency that doesn’t involve the candidate keys

Dept(______________________)

What are the keys of the Dept and Emp tables above (primary and foreign)?

Normalization Exercise 5:

Consider the following table (with each order going to a single customer and shipped from one warehouse)

Order(ordNum,date, warehouseNum, warehouseLoc, custNum, custName)

<table>
<thead>
<tr>
<th>ordNum</th>
<th>date</th>
<th>warehouseNum</th>
<th>warehouseLoc</th>
<th>custNum</th>
<th>custName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Or55</td>
<td>1/1/07</td>
<td>w5</td>
<td>NY</td>
<td>C55</td>
<td>Acme</td>
</tr>
<tr>
<td>Or66</td>
<td>1/1/07</td>
<td>w3</td>
<td>WC</td>
<td>C66</td>
<td>IBM</td>
</tr>
<tr>
<td>Or77</td>
<td>4/4/07</td>
<td>w5</td>
<td>NY</td>
<td>C77</td>
<td>Intel</td>
</tr>
<tr>
<td>Or88</td>
<td>4/12/07</td>
<td>w3</td>
<td>WC</td>
<td>C55</td>
<td>Acme</td>
</tr>
</tbody>
</table>

a) What are the functional dependencies of the order table?

b) What is the primary key?

c) Why must it be in 2nd Normal Form?

d) Is it in 3rd Normal Form? Explain (any dependencies not involving candidate keys?)

e) Convert to 3rd N.Form. Indicate primary and foreign keys.
Normalization Exercise 6:

Consider the following table with the given dependencies.
\[ T(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}, \mathbf{G}) \quad \mathbf{E} \rightarrow \mathbf{G} \quad \mathbf{B} \rightarrow \mathbf{C}, \mathbf{A} \quad \mathbf{D} \rightarrow \mathbf{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{E}, \mathbf{F}, \mathbf{G}} \]

a) What is the primary key? _____________________

b) Why must it be in 2NF? (Why can’t there be any partial dependencies)

c) Is it in 3NF (any dependencies involving attributes that are not candidate keys)?

d) Convert to a set of tables in 3NF. Indicate primary and foreign keys.

Normalization Exercise 7:

Consider the following table and dependencies
\[ T(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}, \mathbf{G}, \mathbf{H}) \quad \mathbf{D+E} \rightarrow \mathbf{A}, \mathbf{B}, \mathbf{C} \quad \mathbf{D} \rightarrow \mathbf{F}, \mathbf{E} \rightarrow \mathbf{G}, \mathbf{H} \quad \mathbf{H} \rightarrow \mathbf{G} \]

a) What is the key?_______________

b) Are there any partial dependencies? List them!

c) Is \( T \) is 2\(^{nd}\) Normal form?

d) Convert to a group of tables in 2\(^{nd}\) N.F

e) Are the tables in part C in 3\(^{rd}\) N.Form? Explain

f) Convert to 3\(^{rd}\) N. Form.

Normalization Exercise 8:

Consider the table involving a chain of bookstores, books and publishers
- each branch can sell a book at what ever Selling-price they want
- each book has a (preset, fixed) list price
- each book is published by a single publisher
(BranchNum, BranchAddr, BkNum, Tittle, PubNum, PubName, List-Price, InStock, list-Price, Selling-Price)

a) What are the functional dependencies?

b) If we keep everything in this one table what is the primary key?

____________________

c) Is the table in 2\textsuperscript{nd} N.F.? _______

d) List any dependencies which involve part of the primary key (partial dependencies) ?

e) Convert to a set of tables in 2\textsuperscript{nd} N.F. (show the primary key of each table)


f) Is the table in 3\textsuperscript{rd} N.F? _______

g) List any dependencies that don’t involve the candidate keys

h) Convert to a set of tables in 3\textsuperscript{rd} N.F. Show all primary and foreign keys