Chapter 10
Functional Dependencies

We will discuss a fundamental concept in database design, the functional dependence, (FD). Basically, a FD is a many-to-one relationship from one set of attributes to another within a given relvar.

For example, in the shipments relvar $SP$, there is such a dependency from the $\{S\#, P\#\}$ to $\{QTY\}$. This dependence means that 1) for any given value for the pair $\{S\#, P\#\}$, there is just one corresponding value of $QTY$, and 2) many distinct values of the pair of $S\#$ and $P\#$ can have the same corresponding $QTY$ value.
Basic definitions

Given the following value of a relvar SCP:

<table>
<thead>
<tr>
<th>S#</th>
<th>CITY</th>
<th>P#</th>
<th>QTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>London</td>
<td>P1</td>
<td>100</td>
</tr>
<tr>
<td>S1</td>
<td>London</td>
<td>P2</td>
<td>100</td>
</tr>
<tr>
<td>S2</td>
<td>Paris</td>
<td>P1</td>
<td>200</td>
</tr>
<tr>
<td>S2</td>
<td>Paris</td>
<td>P2</td>
<td>200</td>
</tr>
<tr>
<td>S3</td>
<td>Paris</td>
<td>P2</td>
<td>300</td>
</tr>
<tr>
<td>S4</td>
<td>London</td>
<td>P2</td>
<td>400</td>
</tr>
<tr>
<td>S4</td>
<td>London</td>
<td>P4</td>
<td>400</td>
</tr>
<tr>
<td>S4</td>
<td>London</td>
<td>P5</td>
<td>400</td>
</tr>
</tbody>
</table>

We firstly consider the definition of FD, for the value of a specific relvar at a particular time.

Let $r$ be a relation, and let $X$ and $Y$ be arbitrary subsets of the set of attributes of $r$. Then we say that $Y$ is functionally dependent on $X$, $X \rightarrow Y$ iff each $X$ value in $r$ has associated with it precisely one $Y$ value in $r$. 
For example, \{S\#\} \rightarrow \{CITY\}. Besides this FD, indeed, SCP also satisfies several other FDs:

1. \{S\#,P\#\} \rightarrow \{QTY\}
2. \{S\#,P\#\} \rightarrow \{CITY\}
3. \{S\#,P\#\} \rightarrow \{CITY, QTY\}
4. \{S\#,P\#\} \rightarrow \{S\#\}
5. \{S\#,P\#\} \rightarrow \{S\#,P\#,CITY,QTY\}
6. \{S\#\} \rightarrow \{QTY\}
7. \{QTY\} \rightarrow \{S\#\}

The left- and right-hand sides of an FD are sometimes called the determinant and the dependent, respectively. When such a set consists of just one attribute, we often drop the set braces. E.g., S\# \rightarrow CITY.
We are more interested in another case, i.e., the FDs not only hold for the value of a relvar for the moment, but for all the possible values that relvar could have.

For example, $S\# \rightarrow \text{CITY}$ holds for all possible values of $\text{SCP}$, since at any given time, a given supplier has precisely one corresponding city, thus, any two tuples appearing in $\text{SCP}$ at the same time with the same supplier number must necessarily have the same city as well.
The general definition

Let $R$ be a relation variable, and let $X$ and $Y$ be arbitrary subsets of the attributes of $R$. Then we say that $Y$ is functionally dependent on $X$, $X \rightarrow Y$, iff, in every possible legal value of $R$, each $X$ value has associated with it precisely one $Y$ value.

From now on, by FD, we mean this more demanding, time-independent sense. Below are some examples:  

\[
\{S\# , P\# \} \rightarrow \{QTY\} \\
\{S\# \} \rightarrow \{CITY\} 
\]

We also have the following, implied, FDs:  

\[
\{S\# , P\# \} \rightarrow \{S\# \} \\
\{S\# , P\# \} \rightarrow \{CITY\} \\
\{S\# , P\# \} \rightarrow \{CITY, QTY\} 
\]

We notice that $\{S\# \} \rightarrow \{QTY\}$ does not hold in general for SCP (?)
A couple of points

1. FDs are derived from user’s specification. For example, when asking users about the relationship among various attributes in SCP, they might tell us that a supplier is based in only one place, which leads to \( \{S\# \} \rightarrow \{\text{CITY}\} \).

Some of them also come out of common sense. For example, \( S\# \) alone does not uniquely identify \( \text{QTY} \), we have to add in \( P\# \) as well. Thus, the FD: \( \{S\#, \ P\# \} \rightarrow \{\text{QTY}\} \).

2. FDs are closely related to candidate keys, since they uniquely identify things. More specifically, if \( X \) is a candidate key of \( R \), then all attributes \( Y \) of \( R \) must be functionally dependent on \( X \).
3. Any FD can be enforced by applying an integrity constraint. For example, the FD: \{S#\} → \{CITY\} can be enforced with the following integrity constraint on relvar SCP:

```
CONSTRAINT S#_CITY__FD
  COUNT(SCP{S#})=COUNT(SCP{S#,CITY});
```

As a relvar usually comes with a large number of FDs, we would like to find a way to cut the number of FDs to its minimum, in the sense that it will tell the same story, but in the fewest words.

More specifically, for a given FD set \( S \), we want to find some other FD set \( T \), such that \(|T| \ll |S|\), and every FD in \( S \) is implied by some FD in \( T \). If such a set \( T \) can be found, then the DBMS only needs to enforce (a much smaller) \( T \), while every FD in \( S \) is automatically enforced.

There exists an algorithm to do just that.
4. FDs are used to derive well-designed database tables, particularly in cutting down redundancy.

In fact, if $R$ satisfies an FD $A \rightarrow B$, but $A$ is not a candidate key, then $R$ will involve some redundancy. For example, in the above value of SCP, \{S#\} $\rightarrow$ \{CITY\}, but S# is not a candidate key, thus, the CITY information will appear many times. This is bad.

We can use FDs to guide us through a process to eliminate such redundancy, which will be fully discussed in the next Chapter.

**Homework:** Exercise 11.1.