

## Chapter 11

### Further Normalization I :

#### 1NF , 2NF , 3NF , BCNF

**11.1** Heath's theorem states that if  $R\{A,B,C\}$  satisfies the FD  $A \rightarrow B$  (where  $A$ ,  $B$ , and  $C$  are sets of attributes), then  $R$  is equal to the join of its projections  $R1$  on  $\{A,B\}$  and  $R2$  on  $\{A,C\}$ . In the following proof of this theorem, we adopt our usual informal shorthand for tuples, writing, e.g.,  $(a,b,c)$  for  $\{A:a,B:b,C:c\}$ .

First we show that no tuple of  $R$  is lost by taking the projections and then joining those projections back together again. Let  $(a,b,c) \in R$ . Then  $(a,b) \in R1$  and  $(a,c) \in R2$ , and so  $(a,b,c) \in R1 \text{ JOIN } R2$ . |

Next we show that every tuple of the join is indeed a tuple of  $R$  (i.e., the join does not generate any "spurious" tuples). Let  $(a,b,c) \in R1 \text{ JOIN } R2$ . In order to generate such a tuple in the join, we must have  $(a,b) \in R1$  and  $(a,c) \in R2$ . Hence there must exist a tuple  $(a,b',c) \in R$  for some  $b'$ , in order to generate the tuple  $(a,c) \in R2$ . We therefore must have  $(a,b') \in R1$ . Now we have  $(a,b) \in R1$  and  $(a,b') \in R1$ ; hence we must have  $b = b'$ , because  $A \rightarrow B$ . Hence  $(a,b,c) \in R$ . |

The converse of Heath's theorem would state that if  $R\{A,B,C\}$  is equal to the join of its projections on  $\{A,B\}$  and on  $\{A,C\}$ , then  $R$  satisfies the FD  $A \rightarrow B$ . This statement is false. For example, Fig. 12.2 in the next chapter shows a relation that is certainly equal to the join of two of its projections and yet does not satisfy any (nontrivial) FDs at all.

**11.2** The claim is almost but not quite valid. The following pathological counterexample is taken from reference [5.5]. Consider the relvar USA  $\{\text{COUNTRY}, \text{STATE}\}$ , interpreted as "STATE is a member of COUNTRY," where COUNTRY is the United States of America in every tuple. Then the FD

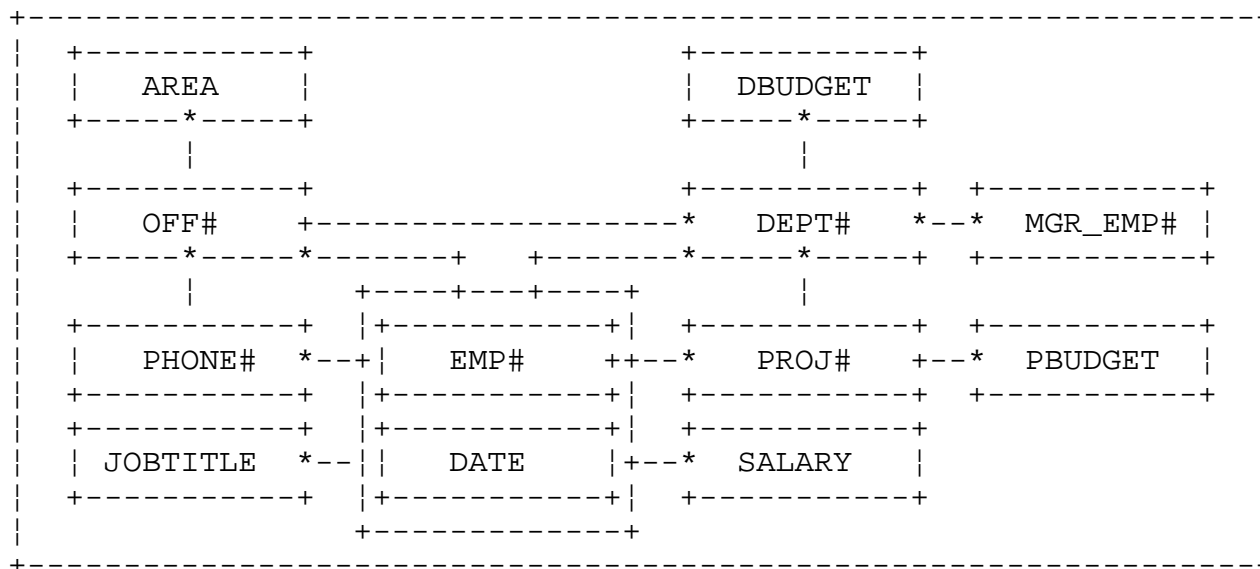
$\{ \} \rightarrow \text{COUNTRY}$

holds in this relvar, and yet the empty set  $\{ \}$  is not a candidate key. So USA is not in BCNF (it can be nonloss-decomposed into its

two unary projections -- though whether it actually should be further normalized in this way could be the subject of debate).

Note, incidentally, that it is quite possible in general to have a candidate key that *is* the empty set! See the answer to Exercise 8.7 in Chapter 8 for further discussion.

**11.3** The figure below shows the most important functional dependencies, both those implied by the wording of the exercise and those corresponding to reasonable semantic assumptions (stated explicitly below). The attribute names are intended to be self-explanatory.



*Semantic assumptions:*

- No employee is the manager of more than one department at a time.
- No employee works in more than one department at a time.
- No employee works on more than one project at a time.
- No employee has more than one office at a time.
- No employee has more than one phone at a time.
- No employee has more than one job at a time.
- No project is assigned to more than one department at a time.

- No office is assigned to more than one department at a time.
- Department numbers, employee numbers, project numbers, office numbers, and phone numbers are all "globally" unique.

*Step 0: Establish initial relvar structure*

Observe first that the original hierarchic structure can be regarded as a 1NF relvar DEPT0 with *relation-valued attributes*:

```
DEPT0 { DEPT#, DBUDGET, MGR_EMP#, XEMP0, XPROJ0, XOFFICE0 }
      KEY { DEPT# }
      KEY { MGR_EMP# }
```

Attributes DEPT#, DBUDGET, and MGR\_EMP# are self-explanatory, but attributes XEMP0, XPROJ0, and XOFFICE0 are relation-valued and do require a little more explanation:

- The XPROJ0 value within a given DEPT0 tuple is a relation with attributes PROJ# and PBUDGET.
- Likewise, the XOFFICE0 value within a given DEPT0 tuple is a relation with attributes OFF#, AREA, and (say) XPHONE0, where XPHONE0 is relation-valued in turn. XPHONE0 relations have just one attribute, PHONE#.
- Finally, the XEMP0 value within a given DEPT0 tuple is a relation with attributes EMP#, PROJ#, OFF#, PHONE#, and (say) XJOB0, where XJOB0 is relation-valued in turn. XJOB0 relations have attributes JOBTITLE and (say) XSALHIST0, where XSALHIST0 is once again relation-valued (XSALHIST0 relations have attributes DATE and SALARY).

Thus the complete hierarchy can be represented by the following nested structure:

```
DEPT0 { DEPT#, DBUDGET, MGR_EMP#,
      XEMP0 { EMP#, PROJ#, OFF#, PHONE#,
            XJOB0 { JOBTITLE,
                  XSALHIST0 { DATE, SALARY } } },
      XPROJ0 { PROJ#, PBUDGET },
      XOFFICE0 { OFF#, AREA, XPHONE0 { PHONE# } } }
```

*Note:* Instead of attempting to show candidate keys, we have used *italics* here to indicate attributes that are at least "unique within parent" (in fact, DEPT#, EMP#, PROJ#, OFF#, and PHONE# are, according to our stated assumptions, all *globally* unique).

*Step 1: Eliminate relation-valued attributes*

Now let us assume for simplicity that we wish every relvar to have a *primary* key specifically -- i.e., we will always designate one candidate key as primary for some reason (the reason is not important here). In the case of DEPT0 in particular, let us choose {DEPT#} as the primary key (and so {MGR\_EMP#} becomes an alternate key).

We now proceed to get rid of all of the relation-valued attributes in DEPT0, since as noted in Section 11.6 such attributes are usually undesirable:\*

- For each relation-valued attribute in DEPT0 -- i.e., attributes XEMP0, XPROJ0, and XOFFICE0 -- form a new relvar with attributes consisting of the attributes of the applicable relations together with the primary key of DEPT0. The primary key of each such relvar is the combination of the attribute that previously gave "uniqueness within parent," together with the primary key of DEPT0. (Note, however, that many of those "primary keys" will include attributes that are redundant for unique identification purposes and will be eliminated later in the overall reduction procedure.) Remove attributes XEMP0, XPROJ0, and XOFFICE0 from DEPT0.
- If any relvar *R* still includes any relation-valued attributes, perform an analogous sequence of operations on *R*.

We obtain the following collection of relvars, with (as indicated) all relation-valued attributes eliminated. Note, however, that while the resulting relvars are in 1NF (of course), they are not necessarily in any higher normal form.

```

DEPT1 { DEPT#, DBUDGET, MGR_EMP# }
      PRIMARY KEY { DEPT# }
      ALTERNATE KEY { MGR_EMP# }

EMP1 { DEPT#, EMP#, PROJ#, OFF#, PHONE# }
     PRIMARY KEY { DEPT#, EMP# }

JOB1 { DEPT#, EMP#, JOBTITLE }
     PRIMARY KEY { DEPT#, EMP#, JOBTITLE }

SALHIST1 { DEPT#, EMP#, JOBTITLE, DATE, SALARY }
         PRIMARY KEY { DEPT#, EMP#, JOBTITLE, DATE }

PROJ1 { DEPT#, PROJ#, PBUDGET }
      PRIMARY KEY { DEPT#, PROJ# }

```

OFFICE1 { DEPT#, OFF#, AREA }  
PRIMARY KEY { DEPT#, OFF# }

PHONE1 { DEPT#, OFF#, PHONE# }  
PRIMARY KEY { DEPT#, OFF#, PHONE# }

*Step 2: Reduce to 2NF*

We now reduce the relvars produced in Step 1 to an equivalent collection of relvars in 2NF by eliminating any FDs that are not irreducible. We consider the relvars one by one.

DEPT1: This relvar is already in 2NF.

EMP1: First observe that DEPT# is actually redundant as a component of the primary key for this relvar. We can take {EMP#} alone as the primary key, in which case the relvar is in 2NF as it stands.

JOB1: Again, DEPT# is not required as a component of the primary key. Since DEPT# is functionally dependent on EMP#, we have a nonkey attribute (DEPT#) that is not irreducibly dependent on the primary key (the combination {EMP#,JOBTITLE}), and hence JOB1 is not in 2NF. We can replace it by

JOB2A { EMP#, JOBTITLE }  
PRIMARY KEY { EMP#, JOBTITLE }

and

JOB2B { EMP#, DEPT# }  
PRIMARY KEY { EMP# }

However, JOB2A is a projection of SALHIST2 (see below), and JOB2B is a projection of EMP1 (renamed as EMP2 below), so both of these relvars can be discarded.

SALHIST1: As with JOB1, we can project away DEPT# entirely. Moreover, JOBTITLE is not required as a component of the primary key; we can take the combination {EMP#,DATE} as the primary key, to obtain the 2NF relvar

SALHIST2 { EMP#, DATE, JOBTITLE, SALARY }  
PRIMARY KEY { EMP#, DATE }

PROJ1: As with EMP1, we can consider DEPT# as a nonkey attribute; the relvar is then in 2NF as it stands.

OFFICE1: Similar remarks apply.

PHONE1: We can project away DEPT# entirely, since the relvar (DEPT#,OFF#) is a projection of OFFICE1 (renamed as OFFICE2 below). Also, OFF# is functionally dependent on PHONE#, so we can take {PHONE#} alone as the primary key, to obtain the 2NF relvar

```
PHONE2 { PHONE#, OFF# }
        PRIMARY KEY { PHONE# }
```

Note that this relvar is not necessarily a projection of EMP2 (phones or offices might exist without being assigned to employees), so we cannot discard this relvar.

Hence our collection of 2NF relvars is

```
DEPT2 { DEPT#, DBUDGET, MGR_EMP# }
        PRIMARY KEY { DEPT# }
        ALTERNATE KEY { MGR_EMP# }

EMP2 { EMP#, DEPT#, PROJ#, OFF#, PHONE# }
        PRIMARY KEY { EMP# }

SALHIST2 { EMP#, DATE, JOBTITLE, SALARY }
          PRIMARY KEY { EMP#, DATE }

PROJ2 { PROJ#, DEPT#, PBUDGET }
        PRIMARY KEY { PROJ# }

OFFICE2 { OFF#, DEPT#, AREA }
          PRIMARY KEY { OFF# }

PHONE2 { PHONE#, OFF# }
          PRIMARY KEY { PHONE# }
```

*Step 3: Reduce to 3NF*

Now we reduce the 2NF relvars to an equivalent 3NF set by eliminating transitive dependencies. The only 2NF relvar not already in 3NF is the relvar EMP2, in which OFF# and DEPT# are both transitively dependent on the primary key {EMP#} -- OFF# via PHONE#, and DEPT# via PROJ# and also via OFF# (and hence via PHONE#). The 3NF relvars (projections) corresponding to EMP2 are

```
EMP3 { EMP#, PROJ#, PHONE# }
        PRIMARY KEY { EMP# }
```

```

X { PHONE#, OFF# }
  PRIMARY KEY { PHONE# }

Y { PROJ#, DEPT# }
  PRIMARY KEY { PROJ# }

Z { OFF#, DEPT# }
  PRIMARY KEY { OFF# }

```

However, X is PHONE2, Y is a projection of PROJ2, and Z is a projection of OFFICE2. Hence our collection of 3NF relvars is simply

```

DEPT3 { DEPT#, DBUDGET, MGR_EMP# }
  PRIMARY KEY { DEPT# }
  ALTERNATE KEY { MGR_EMP# }

EMP3 { EMP#, PROJ#, PHONE# }
  PRIMARY KEY { EMP# }

SALHIST3 { EMP#, DATE, JOBTITLE, SALARY }
  PRIMARY KEY { EMP#, DATE }

PROJ3 { PROJ#, DEPT#, PBUDGET }
  PRIMARY KEY { PROJ# }

OFFICE3 { OFF#, DEPT#, AREA }
  PRIMARY KEY { OFF# }

PHONE3 { PHONE#, OFF# }
  PRIMARY KEY { PHONE# }

```

Finally, it is easy to see that each of these 3NF relvars is in fact in BCNF. |

Note that, given certain (reasonable) additional semantic constraints, this collection of BCNF relvars is **strongly redundant** [5.1], in that the projection of relvar PROJ3 over {PROJ#,DEPT#} is at all times equal to a projection of the join of EMP3 and PHONE3 and OFFICE3.

Observe finally that it is possible to "spot" the BCNF relvars from the FD diagram (how?). Answer: Loosely, there will be one such relvar for each box that has an arrow emerging from it; that relvar will include the attributes from that original box as a candidate key, together with an attribute for every box pointed to from the original box (and no other attributes). Of course, some refinement is needed to this loose statement in order to take care of relvars like DEPT3 that have two or more candidate keys.

Note: We do not claim that it is always possible to "spot" a BCNF decomposition -- only that it is often possible to do so in practical cases. A more precise statement is the following. Given a relvar  $R$  satisfying a set of FDs  $S$ , the algorithm below (Steps 0 to 8) is guaranteed to produce a decomposition  $D$  of  $R$  into 3NF (not BCNF) relvars that is both nonloss and dependency-preserving:

0. Initialize  $D$  to the empty set.
1. Let  $I$  be an irreducible cover for  $S$ .
2. Let  $X$  be a set of attributes appearing on the left-hand side of some FD  $X \rightarrow Y$  in  $I$ .
3. Let the complete set of FDs in  $I$  with left-hand side  $X$  be  $X \rightarrow Y_1, X \rightarrow Y_2, \dots, X \rightarrow Y_n$ .
4. Let the union of  $Y_1, Y_2, \dots, Y_n$  be  $Z$ .
5. Replace  $D$  by the union of  $D$  and the projection of  $R$  over  $X$  and  $Z$ .
6. Repeat Steps 3 through 5 for each distinct  $X$ .
7. Let  $A_1, A_2, \dots, A_n$  be those attributes of  $R$  (if any) still unaccounted for (i.e., still not included in any relvar in  $D$ ). Replace  $D$  by the union of  $D$  and the projection of  $R$  over  $A_1, A_2, \dots, A_n$ .
8. If no relvar in  $D$  includes a candidate key of  $R$ , replace  $D$  by the union of  $D$  and the projection of  $R$  over some candidate key of  $R$ .

And the following algorithm (Steps 0 to 3) is guaranteed to produce a decomposition  $D$  of  $R$  into BCNF relvars that is nonloss but not necessarily dependency-preserving:

0. Initialize  $D$  to contain just  $R$ .
1. For each nonBCNF relvar  $T$  in  $D$ , execute Steps 2 and 3.
2. Let  $X \rightarrow Y$  be an FD for  $T$  that violates the requirements for BCNF.
3. Replace  $T$  in  $D$  by two of its projections, viz. that over  $X$  and  $Y$  and that over all attributes except those in  $Y$ .

To revert to the company database example: As a subsidiary exercise -- not much to do with normalization as such, but very relevant to database design in general -- try extending the foregoing design to incorporate the necessary *foreign key* specifications as well. Answer:

```
DEPT3 { DEPT#, DBUDGET, MGR_EMP# }
      PRIMARY KEY { DEPT# }
      ALTERNATE KEY { MGR_EMP# }
      FOREIGN KEY { RENAME MGR_EMP# AS EMP# } REFERENCES EMP3

EMP3 { EMP#, PROJ#, PHONE# }
     PRIMARY KEY { EMP# }
     FOREIGN KEY { PROJ# } REFERENCES PROJ3
     FOREIGN KEY { PHONE# } REFERENCES PHONE3

SALHIST3 { EMP#, DATE, JOBTITLE, SALARY }
         PRIMARY KEY { EMP#, DATE }
         FOREIGN KEY { EMP# } REFERENCES EMP3

PROJ3 { PROJ#, DEPT#, PBUDGET }
      PRIMARY KEY { PROJ# }
      FOREIGN KEY { DEPT# } REFERENCES DEPT3

OFFICE3 { OFF#, DEPT#, AREA }
        PRIMARY KEY { OFF# }
        FOREIGN KEY { DEPT# } REFERENCES DEPT3

PHONE3 { PHONE#, OFF# }
       PRIMARY KEY { PHONE# }
       FOREIGN KEY { OFF# } REFERENCES OFFICE3
```



ORDLINE { ORD#, LINE#, ITEM#, QTYORD, QTYOUT }  
KEY { ORD#, LINE# }

ITEM { ITEM#, DESCN }  
KEY { ITEM# }

IP { ITEM#, PLANT#, QTYOH, DANGER }  
KEY { ITEM#, PLANT# }