7.1 WHAT CONSTITUTES A BAD DATABASE DESIGN?

Before telling how to design a good database scheme, let us see why some schemes might present problems. In particular let us suppose that we had chosen, in Example 2.14, to combine the relations SUPPLIERS and SUPPLIES of Figure 2.8 into one relation SUP_INFO, with scheme:

SUP_INFO(SNAME, SADDR, ITEM, PRICE)

that included all the information about suppliers. We can see several problems with this scheme.

1. *Redundancy.* The address of the supplier is repeated once for each item supplied.
2. *Potential inconsistency* (*update anomalies*). As a consequence of the redundancy, we could update the address for a supplier in one tuple, while leaving it fixed in another. Thus, we would not have a unique address for each supplier as we feel intuitively we should.
3. *Insertion anomalies.* We cannot record an address for a supplier if that supplier does not currently supply at least one item. We might put null values in the ITEM and PRICE components of a tuple for that supplier, but then, when we enter an item for that supplier, will we remember to delete the tuple with the nulls? Worse, ITEM and SNAME together form a key for the relation, and it might be impossible to look up tuples through a primary index, if there were null values in the key field ITEM.
4. *Deletion anomalies.* The inverse to problem (3) is that should we delete all of the items supplied by one supplier, we unintentionally lose track of the supplier's address.

The reader should appreciate that the problems of redundancy and potential inconsistency are ones we have seen before and dealt with in other models. In the network model, virtual fields were introduced for the purpose of eliminating redundancy and inconsistency. In the hierarchical model, we used virtual record types for the same purpose. The object model encourages references to objects to be made by pointers rather than by copying the object.

In the present example, all the above problems go away if we replace SUP_INFO by the two relation schemes

SUPPLIERS(SNAME, SADDR)
SUPPLIES(SNAME, ITEM, PRICE)

as in Figure 2.8. Here, SUPPLIERS, gives the address for each supplier exactly once; hence there is no redundancy. Moreover, we can enter an address for a supplier even if it currently supplies no items.

Yet some questions remain. For example, there is a disadvantage to the above decomposition; to find the addresses of suppliers of Brie, we must now take a join, which is expensive, while with the single relation SUP_INFO we could simply do a selection and projection. How do we determine that the above replacement is beneficial? Are there other problems of the same four kinds present in the two new relation schemes? How do we find a good replacement for a bad relation scheme?