1. All functional dependencies in D^+ that include only attributes of R_i.
2. All multivalued dependencies of the form:

$$\alpha \twoheadrightarrow \beta \cap R_i$$

where $\alpha \subseteq R_i$ and $\alpha \twoheadrightarrow \beta$ is in D^+.

7.6.3 4NF Decomposition

The analogy between 4NF and BCNF applies to the algorithm for decomposing a schema into 4NF. Figure 7.16 shows the 4NF decomposition algorithm. It is identical to the BCNF decomposition algorithm of Figure 7.11, except that it uses multivalued dependencies and uses the restriction of D^+ to R_i.

If we apply the algorithm of Figure 7.16 to (*ID*, *dept_name*, *street*, *city*), we find that *ID*$\twoheadrightarrow$ *dept_name* is a nontrivial multivalued dependency, and *ID* is not a superkey for the schema. Following the algorithm, we replace it with two schemas:

(*ID*, *dept_name*)
(*ID*, *street*, *city*)

This pair of schemas, which is in 4NF, eliminates the redundancy we encountered earlier.

As was the case when we were dealing solely with functional dependencies, we are interested in decompositions that are lossless and that preserve dependencies. The following fact about multivalued dependencies and losslessness shows that the algorithm of Figure 7.16 generates only lossless decompositions:

---

result := {*R*};
done := false;
compute D^+; Given schema R_i, let D_i denote the restriction of D^+ to R_i
while (**not** *done*) **do**
  if (there is a schema R_i in *result* that is not in 4NF w.r.t. D_i)
    then begin
      let $\alpha \twoheadrightarrow \beta$ be a nontrivial multivalued dependency that holds
      on R_i such that $\alpha \rightarrow R_i$ is not in D_i, and $\alpha \cap \beta = \emptyset$;
      result := (*result* − R_i) ∪ (R_i − β) ∪ (α, β);
    end
    else *done* := true;

Figure 7.16 4NF decomposition algorithm.