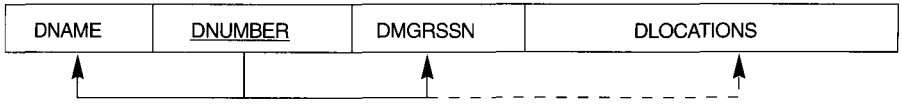
10.3.4 First Normal Form

First normal form (1NF) is now considered to be part of the formal definition of a relation in the basic (flat) relational model;[12] historically, it was defined to disallow multivalued attributes, composite attributes, and their combinations. It states that the domain of an attribute must include only *atomic* (simple, indivisible) *values* and that the value of any attribute in a tuple must be a *single value* from the domain of that attribute. Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a *single tuple*. In other words, 1NF disallows "relations within relations" or "relations as attribute values within tuples." The only attribute values permitted by 1NF are single **atomic (or indivisible) values.**

Consider the DEPARTMENT relation schema shown in Figure 10.1, whose primary key is DNUMBER, and suppose that we extend it by including the DLOCATIONS attribute as shown in Figure 10.8a. We assume that each department can have *a number of* locations. The DEPARTMENT schema and an example relation state are shown in Figure 10.8. As we can see,

(a)



(b) DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c) DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

FIGURE 10.8 Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of same relation with redundancy.

12. This condition is removed in the *nested relational model* and in *object-relational systems* (ORDBMSs), both of which allow *unnormalized relations* (see Chapter 22).

this is not in 1NF because DLOCATIONS is not an atomic attribute, as illustrated by the first tuple in Figure 10.8b. There are two ways we can look at the DLOCATIONS attribute:

- The domain of DLOCATIONS contains atomic values, but some tuples can have a set of these values. In this case, DLOCATIONS *is not* functionally dependent on the primary key DNUMBER.
- The domain of DLOCATIONS contains sets of values and hence is nonatomic. In this case, DNUMBER → DLOCATIONS, because each set is considered a single member of the attribute domain.[13]

In either case, the DEPARTMENT relation of Figure 10.8 is not in 1NF; in fact, it does not even qualify as a relation according to our definition of relation in Section 5.1. There are three main techniques to achieve first normal form for such a relation:

1. Remove the attribute DLOCATIONS that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key DNUMBER of DEPARTMENT. The primary key of this relation is the combination {DNUMBER, DLOCATION}, as shown in Figure 10.2. A distinct tuple in DEPT_LOCATIONS exists for *each location* of a department. This decomposes the non-1NF relation into two 1NF relations.
2. Expand the key so that there will be a separate tuple in the original DEPARTMENT relation for each location of a DEPARTMENT, as shown in Figure 10.8c. In this case, the primary key becomes the combination {DNUMBER, DLOCATION}. This solution has the disadvantage of introducing *redundancy* in the relation.
3. If a *maximum number of values* is known for the attribute—for example, if it is known that *at most three locations* can exist for a department—replace the DLOCATIONS attribute by three atomic attributes: DLOCATION1, DLOCATION2, and DLOCATION3. This solution has the disadvantage of introducing *null values* if most departments have fewer than three locations. It further introduces a spurious semantics about the ordering among the location values that is not originally intended. Querying on this attribute becomes more difficult; for example, consider how you would write the query: "List the departments that have "Bellaire" as one of their locations" in this design.

Of the three solutions above, the first is generally considered best because it does not suffer from redundancy and it is completely general, having no limit placed on a maximum number of values. In fact, if we choose the second solution, it will be decomposed further during subsequent normalization steps into the first solution.

First normal form also disallows multivalued attributes that are themselves composite. These are called **nested relations** because each tuple can have a relation *within it*. Figure 10.9 shows how the EMP_PROJ relation could appear if nesting is allowed. Each tuple represents an employee entity, and a relation PROJS(PNUMBER, HOURS) *within each*

13. In this case we can consider the domain of DLOCATIONS to be the **power set** of the set of single locations; that is, the domain is made up of all possible subsets of the set of single locations.

(a) **EMP_PROJ**

SSN	ENAME	PROJS	
		PNUMBER	HOURS

(b) **EMP_PROJ**

SSN	ENAME	PNUMBER	HOURS
123456789	Smith,John B.	1	32.5
		2	7.5
666884444	Narayan,Ramesh K.	3	40.0
453453453	English,Joyce A.	1	20.0
		2	20.0
333445555	Wong,Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya,Alicia J.	30	30.0
		10	10.0
987987987	Jabbar,Ahmad V.	10	35.0
		30	5.0
987654321	Wallace,Jennifer S.	30	20.0
		20	15.0
888665555	Borg,James E.	20	null

(c) **EMP_PROJ1**

SSN	ENAME

EMP_PROJ2

SSN	PNUMBER	HOURS

FIGURE 10.9 Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a "nested relation" attribute PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

tuple represents the employee's projects and the hours per week that employee works on each project. The schema of this EMP_PROJ relation can be represented as follows:

```
EMP_PROJ(SSN, ENAME, {PROJS(PNUMBER, HOURS)})
```

The set braces { } identify the attribute PROJS as multivalued, and we list the component attributes that form PROJS between parentheses ( ). Interestingly, recent trends for supporting complex objects (see Chapter 20) and XML data (see Chapter 26) using the relational model attempt to allow and formalize nested relations within relational database systems, which were disallowed early on by 1NF.

Notice that SSN is the primary key of the EMP_PROJ relation in Figures 10.9a and b, while PNUMBER is the **partial** key of the nested relation; that is, within each tuple, the nested relation must have unique values of PNUMBER. To normalize this into 1NF, we remove the nested relation attributes into a new relation and *propagate the primary key* into it; the primary key of the new relation will combine the partial key with the primary key of the original relation. Decomposition and primary key propagation yield the schemas EMP_PROJ1 and EMP_PROJ2 shown in Figure 10.9c.

This procedure can be applied recursively to a relation with multiple-level nesting to **unnest** the relation into a set of 1NF relations. This is useful in converting an unnormalized relation schema with many levels of nesting into 1NF relations. The existence of more than one multivalued attribute in one relation must be handled carefully. As an example, consider the following non-1NF relation:

PERSON (SS#, {CAR_LIC#}, {PHONE#})

This relation represents the fact that a person has multiple cars and multiple phones. If a strategy like the second option above is followed, it results in an all-key relation:

PERSON_IN_1NF (SS#, CAR_LIC#, PHONE#)

To avoid introducing any extraneous relationship between CAR_LIC# and PHONE#, all possible combinations of values are represented for every SS#, giving rise to redundancy. This leads to the problems handled by multivalued dependencies and 4NF, which we discuss in Chapter 11. The right way to deal with the two multivalued attributes in PERSON above is to decompose it into two separate relations, using strategy 1 discussed above: P1(SS#, CAR_LIC#) and P2(SS#, PHONE#).

10.3.5 Second Normal Form

Second normal form (2NF) is based on the concept of *full functional dependency*. A functional dependency $X \rightarrow Y$ is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold any more; that is, for any attribute $A \in X$, $(X - \{A\})$ does *not* functionally determine Y. A functional dependency $X \rightarrow Y$ is a **partial dependency** if some attribute $A \in X$ can be removed from X and the dependency still holds; that is, for some $A \in X$, $(X - \{A\}) \rightarrow Y$. In Figure 10.3b, {SSN, PNUMBER} → HOURS is a full dependency (neither SSN → HOURS nor PNUMBER → HOURS holds). However, the dependency {SSN, PNUMBER} → ENAME is partial because SSN → ENAME holds.

Definition. A relation schema R is in **2NF** if every nonprime attribute A in R is *fully functionally dependent* on the primary key of R.

The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key. If the primary key contains a single attribute, the test need not be applied at all. The EMP_PROJ relation in Figure 10.3b is in 1NF but is not in 2NF. The nonprime attribute ENAME violates 2NF because of FD2, as do the nonprime attributes PNAME and PLOCATION because of FD3. The functional dependencies FD2 and FD3 make ENAME, PNAME, and PLOCATION partially dependent on the primary key {SSN, PNUMBER} of EMP_PROJ, thus violating the 2NF test.

If a relation schema is not in 2NF, it can be "second normalized" or "2NF normalized" into a number of 2NF relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent. The functional dependencies FD1, FD2, and FD3 in Figure 10.3b hence lead to the decomposition of EMP_PROJ into the three relation schemas EP1, EP2, and EP3 shown in Figure 10.10a, each of which is in 2NF.

10.3.6 Third Normal Form

Third normal form (3NF) is based on the concept of *transitive dependency*. A functional dependency $X \rightarrow Y$ in a relation schema R is a **transitive dependency** if there is a set of

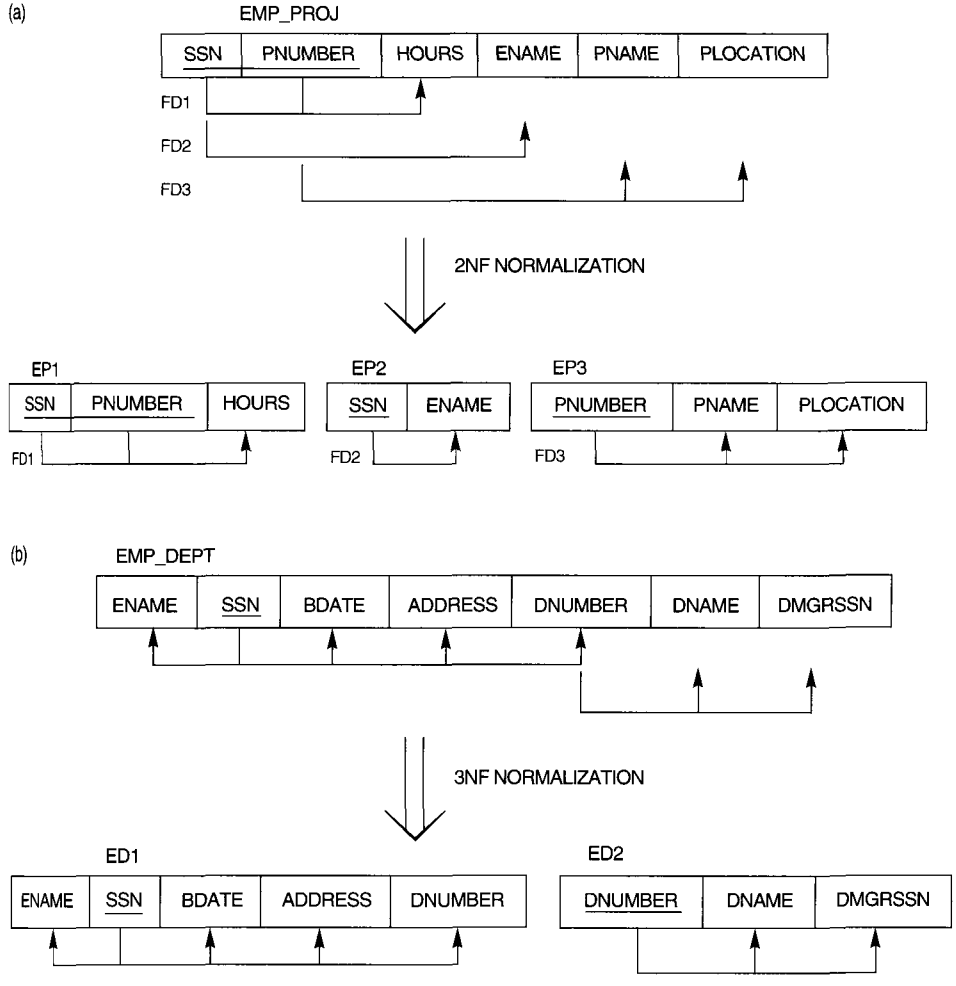


FIGURE 10.10 Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.