Module 4 notes

Informal Design Guidelines for Relation Schemas

- Four *informal guidelines* that may be used as *measures to determine the quality* of relation schema design:

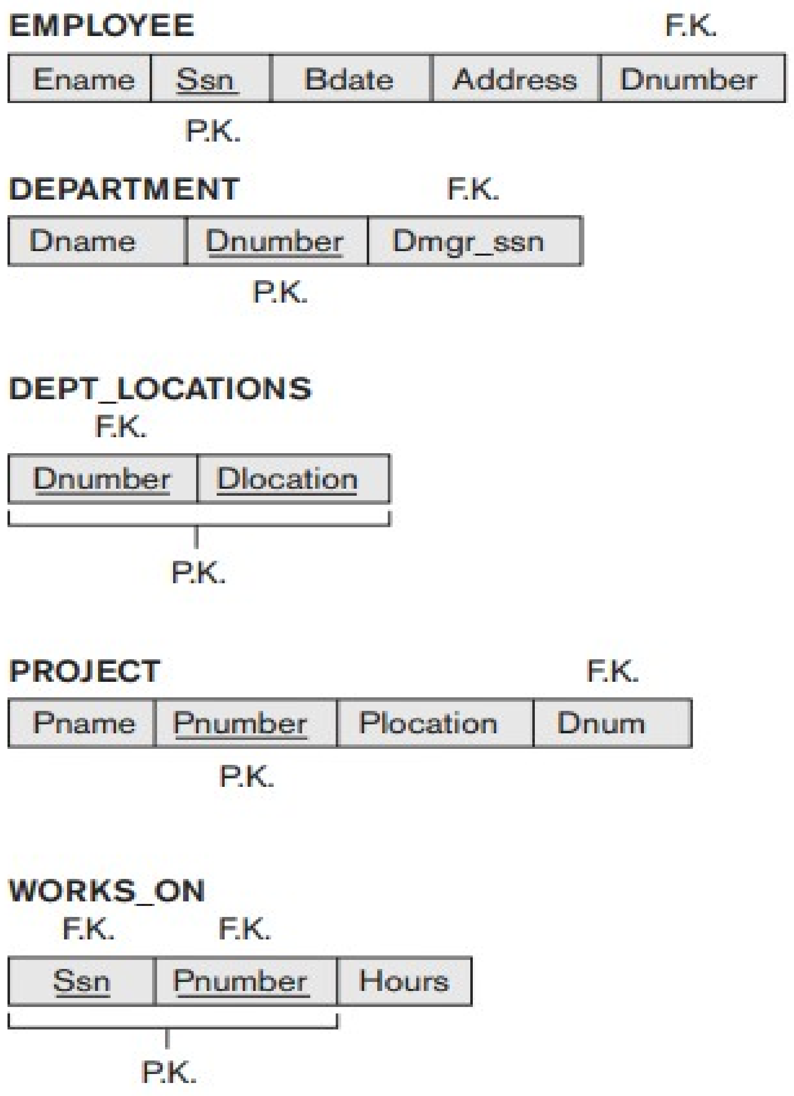
■ Making sure that the semantics of the attributes is clear in the schema
■ Reducing the redundant information in tuples
■ Reducing the NULL values in tuples
■ Disallowing the possibility of generating spurious tuples

Imparting Clear Semantics to Attributes in Relations

- Whenever we group attributes to form a relation schema, we assume that attributes belonging to one relation have certain real-world meaning and a proper interpretation associated with them.
- In general, the easier it is to explain the semantics of the relation the better the relation schema design will be.
- To illustrate this, consider Figure 14.1, a simplified version of the COMPANY relational database The meaning of the EMPLOYEE relation schema is simple: Each tuple represents an employee, with values for the employee's name (Ename), Social Security number (Ssn), birth date (Bdate), and address (Address), and the number of the department that the employee works for (Dnumber). The Dnumber attribute is a foreign key that represents an *implicit relationship* between EMPLOYEE and DEPARTMENT.
- Each DEPARTMENT tuple represents a department entity, and each PROJECT tuple represents a project entity. Similarly other relational schema are also easy to explain

- Hence, all the relation schemas in Figure 14.1 may be considered as easy to explain and therefore good it is having clear semantics. We can thus formulate the following informal design guideline.

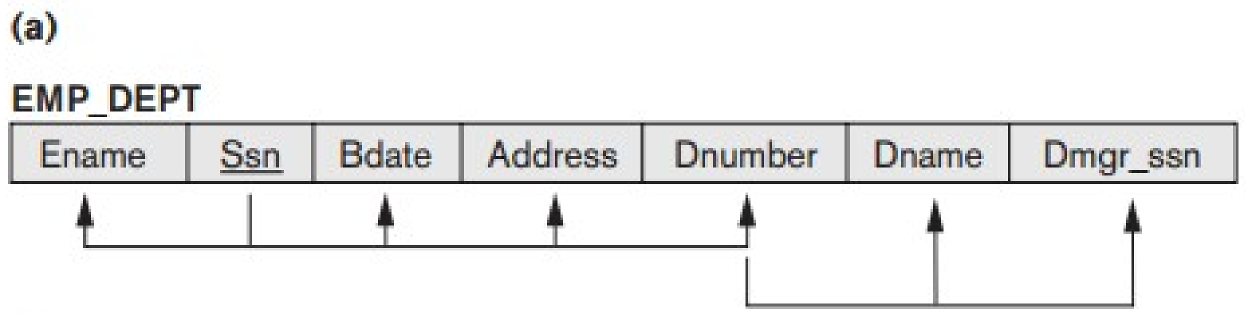
Figure 14.1
A simplified COMPANY relational database schema.

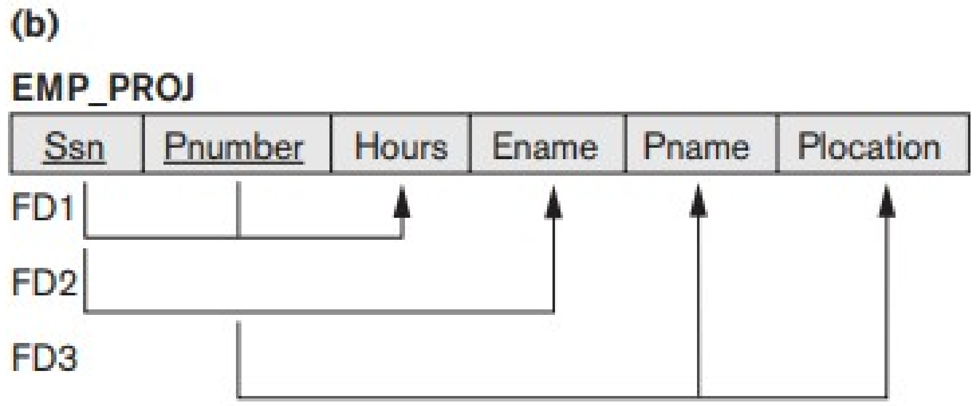


Guideline 1.

- **Guideline 1.** Design a relation schema so that it is easy to explain its meaning.
- Do not combine attributes from multiple entity types and relationship types into a single relation.
- Intuitively, if a relation schema corresponds to one entity type or one relationship type, it is straightforward to explain its meaning.
- Otherwise, if the relation corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and the relation cannot be easily explained.
- ***Examples of Violating Guideline 1.***
- The relation schemas in Figures 14.3(a) and 14.3(b) also have clear semantics.
- Although there is nothing wrong logically with these two relations, they violate Guideline 1 by mixing attributes from distinct real-world entities: EMP_DEPT mixes attributes of employees and departments, and EMP_PROJ mixes attributes of employees and projects and the WORKS_ON relationship. Hence, they fare poorly against the above measure of design quality. ***They may be used as views, but they cause problems when used as base relations.***

Figure 14.3
Two relation schemas suffering from update anomalies.
(a) EMP_DEPT and
(b) EMP_PROJ.





Impact of violation of guidelines 1

1. **Redundant Information in Tuples**
2. **Insertion Anomalies.**
3. **Deletion Anomalies.**
4. **Modification Anomalies.**

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

EMPLOYEE

Ename	Ssn	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

Insertion Anomalies

2. Insertion Anomalies. It is explained by the following examples based on the EMP_DEPT relation:

a. It is difficult to insert a new department that has no employees as yet in the EMP_DEPT relation. The only way to do this is to place NULL values in the attributes for employee. This violates the entity integrity for EMP_DEPT because its primary key Ssn cannot be null. This problem does not occur in the design of Figure 14.2 because a department is entered in the DEPARTMENT relation whether or not any employees work for it, and whenever an employee is assigned to that department, a corresponding tuple is inserted in EMPLOYEE.

Deletion Anomalies.

Deletion Anamalies

- If we delete from EMP_DEPT an employee tuple that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database.
- This problem does not occur in the database of Figure 14.2 because DEPARTMENT tuples are stored separately.

Modification/updation Anomalies

- In EMP_DEPT, if we change the value of one of the attributes of a particular department—say, the manager of department 5—we must update the tuples of *all* employees who work in that department; otherwise, the database will become inconsistent.
- If we fail to update some tuples, the same department will be shown to have two different values for manager in different employee tuples, which would be wrong.

Guideline 2.

- Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations.
- In general, it is advisable to use anomaly-free base relations and to specify views that include the joins for placing together the attributes frequently referenced in important queries.

Guideline 3

- **NULL Values in Tuples**
  If many of the attributes do not apply to all tuples in the relation, we end up with many NULLs in those tuples.
  - This can waste space at the storage level
  - and may also lead to problems with understanding the meaning of the attributes
  - Another problem with NULLs is how to account for them when aggregate operations such as COUNT or SUM are applied.
  - SELECT and JOIN operations involve comparisons; if NULL values are present, the results may become unpredictable.
  - Moreover, NULLs can have multiple interpretations, such as the following:
  - The attribute *does not apply* to this tuple. For example, Visa_status may not apply to U.S. students.
  - The attribute value for this tuple is *unknown*. For example, the Date_of_birth may be unknown for an employee.
  - The value is *known but absent*; that is, it has not been recorded yet. For example, the Home_Phone_Number for an employee may exist, but may not be available and recorded yet.

Guideline 3.

- As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL. If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation.

Generation of Spurious Tuples

- Consider the two relation schemas EMP_LOCS and EMP_PROJ1 in Figure 14.5(a),

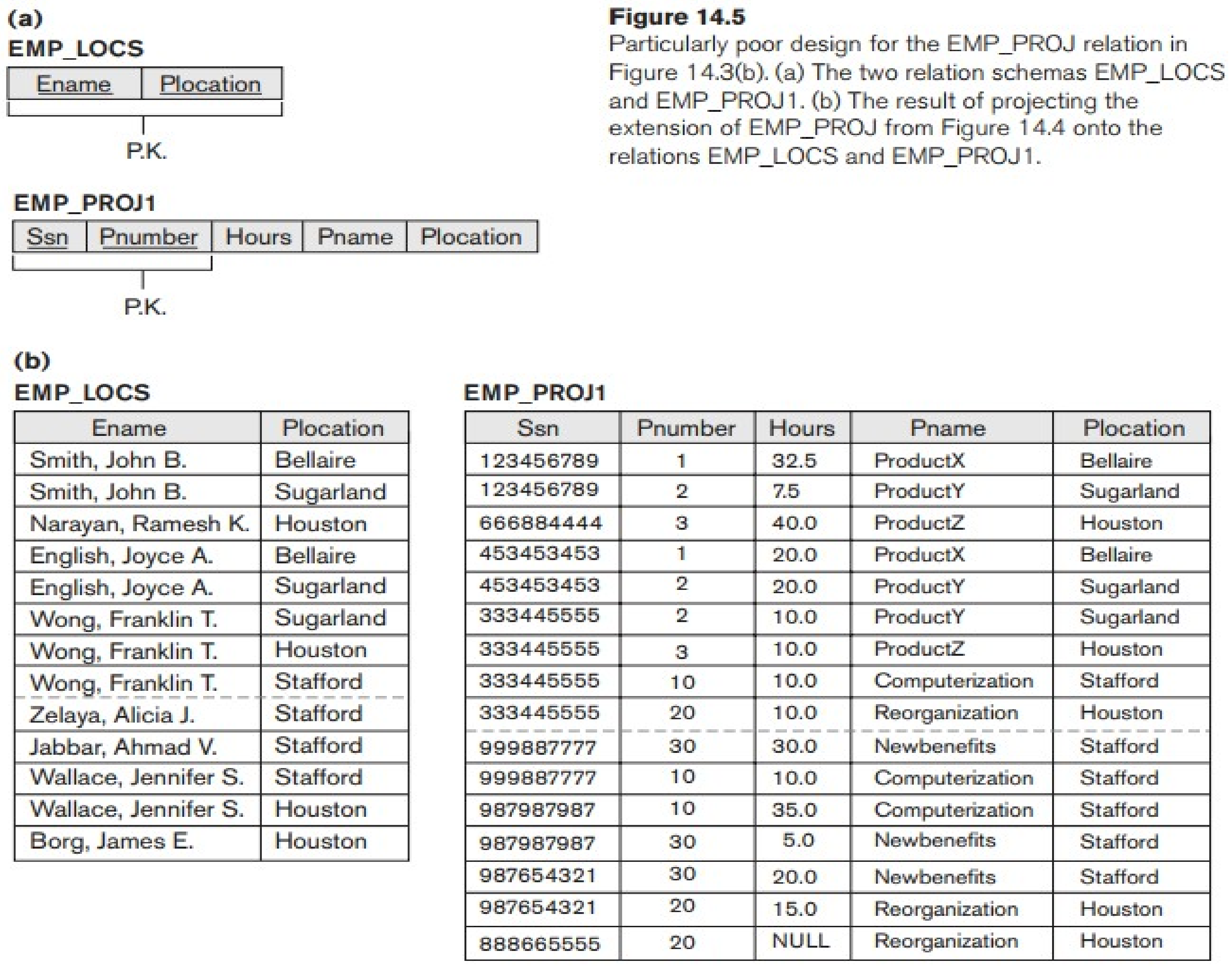


Figure 14.5
Particularly poor design for the EMP_PROJ relation in Figure 14.3(b). (a) The two relation schemas EMP_LOCS and EMP_PROJ1. (b) The result of projecting the extension of EMP_PROJ from Figure 14.4 onto the relations EMP_LOCS and EMP_PROJ1.

(a)

EMP_LOCS

<u>Ename</u>	<u>Plocation</u>

P.K.

EMP_PROJ1

<u>Ssn</u>	<u>Pnumber</u>	Hours	Pname	Plocation

P.K.

(b)

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston

- A tuple in EMP_LOCS means that the employee whose name is Ename works on *at least one* project located at Plocation. A tuple in EMP_PROJ1 refers to the fact that the employee whose Social Security number is Ssn works the given Hours per week on the project whose name, number, and location are Pname, Pnumber, and Plocation
- If we attempt a NATURAL JOIN operation on EMP_PROJ1 and EMP_LOCS, the result produces many more tuples than the original set of tuples in EMP_PROJ. In Figure 14.6, the result of applying the join to only the tuples for employee with Ssn = “123456789” is shown (to reduce the size of the resulting relation).
- Additional tuples that were not in EMP_PROJ are called **spurious tuples** because they represent spurious information that is not valid. The spurious tuples are marked by asterisks (*) in Figure 14.6.

	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.
	453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith, John B.
	453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith, John B.
*	333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
	333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.

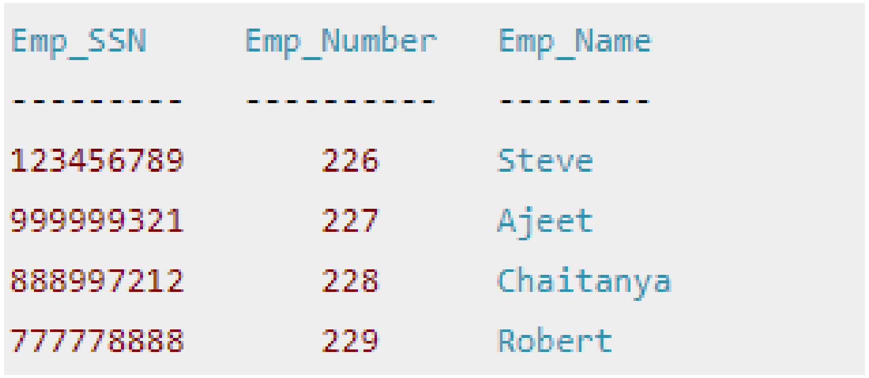
Guideline 4

- **Guideline 4.** Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated.
- Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples

Normalization of Relations

- **Normalization of data** can be considered a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of
  (1) minimizing redundancy
- (2) minimizing the insertion, deletion, and update
  anomalies
- *Normalization rules divides larger tables into smaller tables and links them using relations.*
- normalization process was first proposed by Codd

- Let’s take an example to understand this:
  Table: Employee



```
Emp_SSN       Emp_Number    Emp_Name
---------     ----------    --------
123456789        226        Steve
999999321        227        Ajeet
888997212        228        Chaitanya
777778888        229        Robert
```

- **Super keys**: The above table has following super keys. All of the following sets of super key are able to uniquely identify a row of the employee table.
- {Emp_SSN}
- {Emp_Number}
- {Emp_SSN, Emp_Number}
- {Emp_SSN, Emp_Name}
- {Emp_SSN, Emp_Number, Emp_Name}
- {Emp_Number, Emp_Name}

- **Candidate Keys**: a candidate key is a minimal super key with no redundant attributes. The following two set of super keys are chosen from the above sets as there are no redundant attributes in these sets.
- {Emp_SSN}
- {Emp_Number}
- Only these two sets are candidate keys as all other sets are having redundant attributes that are not necessary for unique identification.

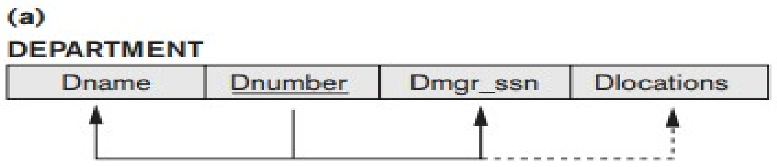
- **Formal Defination of super key.**
- A **superkey** of a relation schema $R = \{A1, A2, \ldots, An\}$ is a set of attributes $S \subseteq R$ with the property that no two tuples $t1$ and $t2$ in any legal relation state r of R will have $t1[S] = t2[S]$.

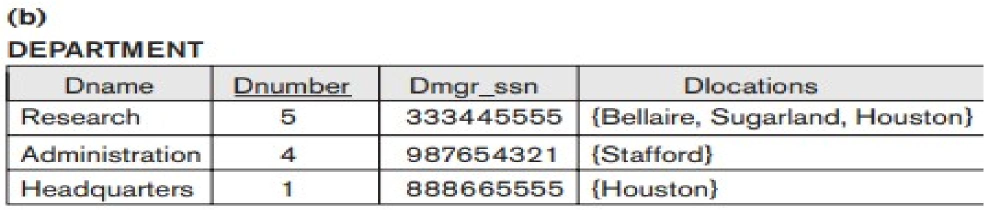
- **Defination of prime and non-prime attributes.**
- **Definition.** An attribute of relation schema *R* is called a **prime attribute** of *R* if it is a member of *some candidate key* of *R*.

- An attribute is called **nonprime** if it is not a prime attribute—that is, if it is not a member of any candidate key

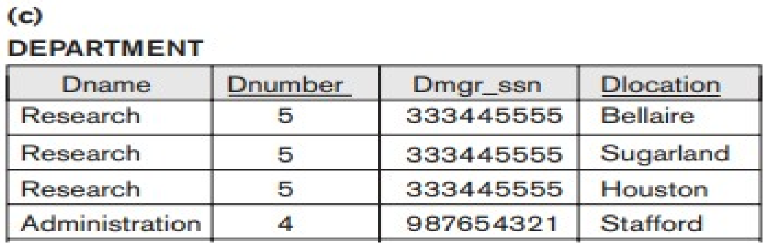
First Normal Form

- 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.*
- Consider the DEPARTMENT relation schema whose primary key is Dnumber, and suppose that we extend it by including the Dlocations attribute. Below figure is 14.9




(b)
DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}


(c)
DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford

- There are two ways we can look at the Dlocations attribute:

1. 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.
2. The domain of Dlocations contains sets of values and hence is nonatomic.

As we can see, this is not in 1NF because Dlocations is not an atomic attribute, There are three main techniques to achieve first normal form for such a relation:

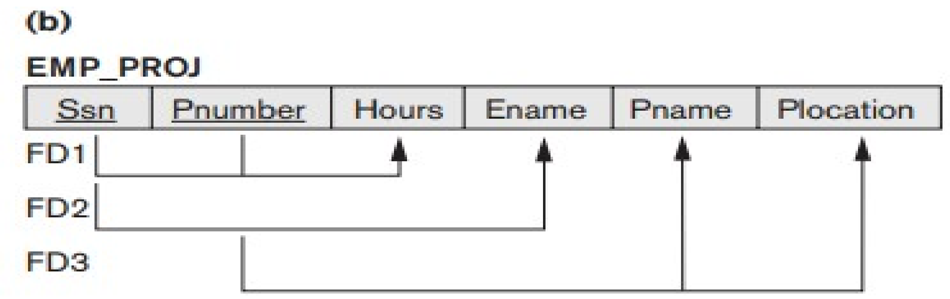
3. Remove the attribute Dlocations that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key Dnumber of DEPARTMENT. The primary key of this newly formed relation is the combination {Dnumber, Dlocation}, A distinct tuple in DEPT_LOCATIONS exists for *each location* of a department. This decomposes the non-1NF relation into two 1NF relations.
4. 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 14.9(c). This solution has the disadvantage of introducing *redundancy* in the relation and hence is rarely adopted.
5. 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 spurious semantics about the ordering among the location values;

- Of the three solutions above, the first is generally considered best because it does not suffer from redundancy and it is completely general; it places no maximum limit on the number of values.

Functional Dependencies

- A functional dependency is a constraint between two sets of attributes from the database.
- A **functional dependency**, denoted by *X* → *Y*, between two sets of attributes *X* and *Y* that are subsets of *R* specifies a *constraint* on the possible tuples. The constraint is that, for any two tuples *t*1 and *t*2 in *r* that have *t*1[*X*] = *t*2[*X*], they must also have *t*1[*Y*] = *t*2[*Y*].
- This means that the values of the *Y* component of a tuple in *r* depend on, or are *determined by*, the values of the *X* component; alternatively, the values of the *X* component of a tuple uniquely (or **functionally**) *determine* the values of the *Y* component.
- We also say that there is a functional dependency from *X* to *Y*, or that *Y* is **functionally dependent** on *X*.
- The abbreviation for functional dependency is **FD** or **f.d.**

- Consider the relation schema EMP_PROJ in Figure 14.3(b); from the semantics of the attributes and the relation, we know that the following functional dependencies should hold:



- a. Ssn → Ename
  b. Pnumber → {Pname, Plocation}
  c. {Ssn, Pnumber} → Hours
  These functional dependencies specify that
- (a) the value of an employee's Social Security number (Ssn) uniquely determines the employee name (Ename),
- (b) the value of a project's number (Pnumber) uniquely determines the project name (Pname) and location (Plocation), and
- (c) a combination of Ssn and Pnumber values uniquely determines the number of hours the employee currently works on the project per week (Hours). Alternatively, we say that Ename is functionally determined by (or functionally dependent on) Ssn, or *given a value of Ssn*, *we know the value of Ename*, and so on.

Second Normal Form

Second normal form (2NF) is based on the concept of *full functional dependency.* A functional dependency *X → Y is a* ***full functional dependency if removal of any*** attribute *A from X means that the dependency does not hold anymore;*

Figure 14.3(b), {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.

- **To be in second normal form, a relation must satisfy 2 Conditions**

1. The relation must be in first normal form
2. Relation must not contain any partial dependency.

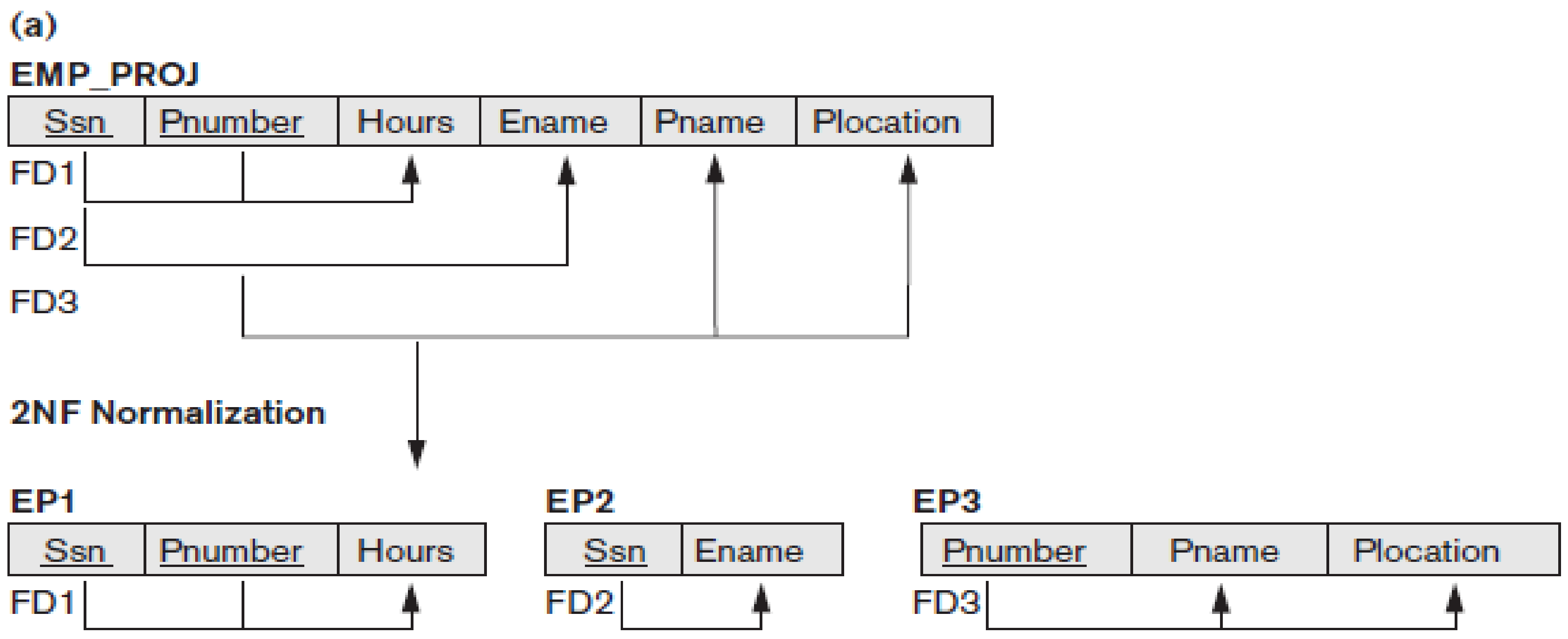
2 NF Definition. A relation schema *R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R*.

- Second Normal Form applies to relations with composite keys, that is, relations with a primary key composed of two or more attributes.
- A relation with a single-attribute primary key is automatically in at least 2NF.
- A relation that is not in 2NF may suffer from the update anomalies.

The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key.

Example

- The EMP_PROJ relation in Figure 14.3(b) is in 1NF but is not in 2NF. The nonprime attribute Ename violates 2NF because of FD2,
- The nonprime attributes Pname and Plocation violates 2NF because of FD3.
- Each of the functional dependencies FD2 FD3 violates 2NF because Ename can be functionally determined by only Ssn, and both Pname and Plocation can be functionally determined by only Pnumber. Attributes Ssn and Pnumber are a part of the primary key {Ssn, Pnumber} of EMP_PROJ, thus violating the 2NF test.

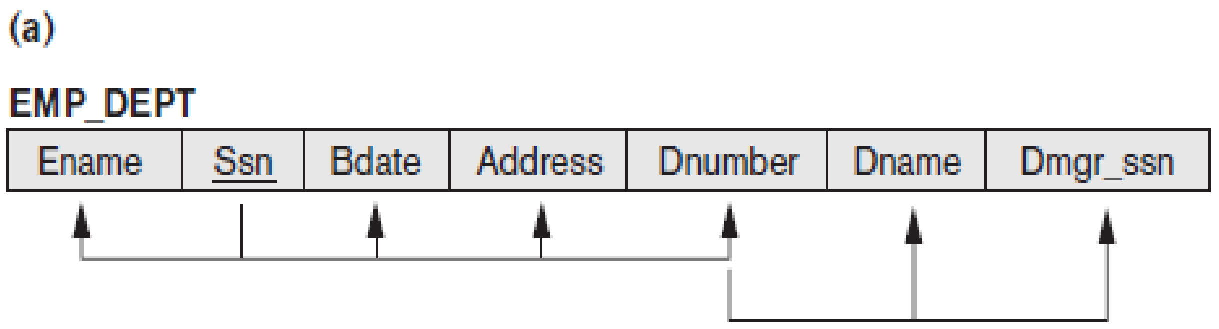

(a)
EMP_PROJ
Ssn
Pnumber
Hours
Ename
Pname
Plocation
FD1
FD2
FD3
2NF Normalization
EP1
Ssn
Pnumber
Hours
FD1
EP2
Ssn
Ename
FD2
EP3
Pnumber
Pname
Plocation
FD3

Third Normal Form

- **Third normal form (3NF) is based on the concept of *transitive dependency.***
- ***A functional*** dependency *X → Y in a relation schema R is a* **transitive dependency *if there*** exists a set of attributes *Z in R that is neither a candidate key nor a subset of any key of R, and both X → Z and Z → Y hold.*

- **Definition. According to Codd's original definition, a relation schema *R is in* 3NF if it satisfies 2NF *and no nonprime attribute of R is transitively dependent*** on the primary key.

Figure 14.3
Two relation schemas suffering from update anomalies.
(a) EMP_DEPT and
(b) EMP_PROJ.



Example:-

- *The dependency Ssn → Dmgr_ssn is transitive* through Dnumber in EMP_DEPT in Figure 14.3(a), because both the dependencies Ssn → Dnumber and Dnumber → Dmgr_ssn hold *and Dnumber is neither a key itself nor a* subset of the key of EMP_DEPT.

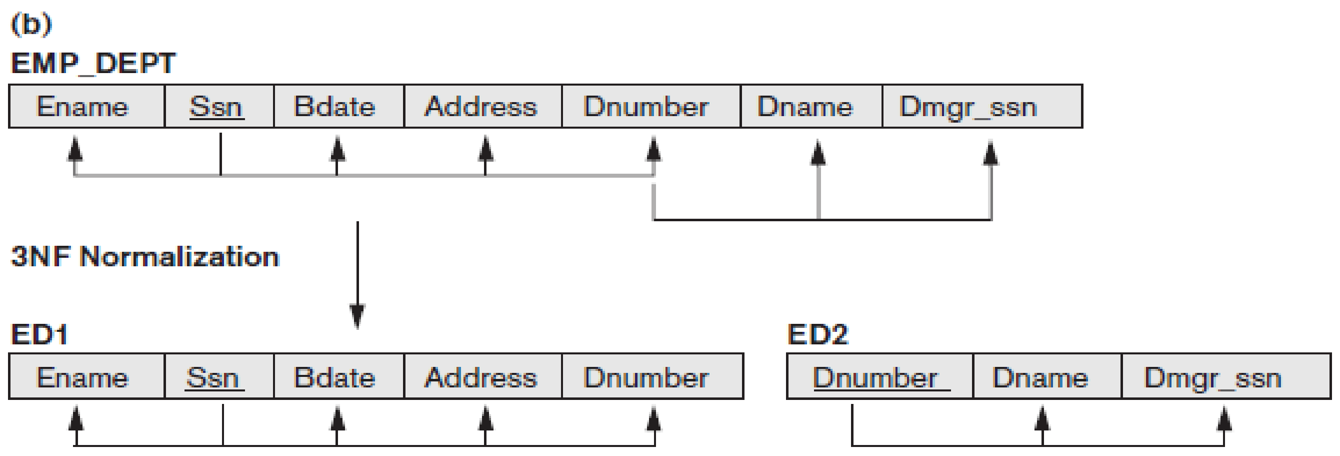


Figure 14.11
Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

We can normalize EMP_DEPT by decomposing it into the two 3NF relation schemas ED1 and ED2 shown in Figure 14.11(b). Intuitively, we see that ED1 and ED2 represent independent

facts about employees and departments, both of which are entities in their own right. A NATURAL JOIN operation on ED1 and ED2 will recover the original relation EMP_DEPT without generating spurious tuples.

Table 14.1 Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

Boyce Codd normal form (BCNF)

- BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every Functional Dependancy X->Y, X should be the candidate key or super key of the table.
- **Example**: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:
-

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp

- **Functional dependencies in the table above**:
  emp_id -> emp_nationality
  emp_dept -> {dept_type, dept_no_of_emp}
- **Candidate key**: {emp_id, emp_dept}
- The table is not in BCNF as neither emp_id nor emp_dept alone are keys.
- To make the table comply with BCNF we can break the table in three tables like this:
  emp_nationality table:
- emp_id   emp_nationality
- **emp_dept table:**
- emp_dept dept_type dept_no_of_emp
- **emp_dept_mapping table:**
- emp_id emp_dept

- **Functional dependencies**:
  emp_id -> emp_nationality
  emp_dept -> {dept_type, dept_no_of_emp}
- **Candidate keys**:
  For first table: emp_id
  For second table: emp_dept
  For third table: {emp_id, emp_dept}
- This is now in BCNF as in both the functional dependencies left side part is a key.

Multivalued Dependency

- In the relation state shown in Figure 14.15(a), the employee with Ename Smith works on two projects ‘X’ and ‘Y’ and has two dependents ‘John’ and ‘Anna’, and therefore there are four tuples to represent these facts together.

- **A multivalued dependency represents a dependancy between attributes A,B, c in relation R such that for each value of A there is a set of values of B and a set of values of C exists.**

- **Minimum 3 columns exists.**

- To address this situation, the concept of *multivalued dependency (MVD) was proposed* and, based on this dependency, the *fourth normal form was defined*.

Fourth Normal Form (4NF)

- **Definition. A relation schema *R is in 4NF if***

1. ***It is in BCNF and***
2. ***It should not have multi-valued dependancy***

sid	course	hobby
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	PHP	Hockey

SID	Course

SID	HOBBY