Chapter 10, Functional Dependencies and Normalization for Relational Databases

- We need some formal measure of why the choice of attributes for a relation schema may be better than another.

- Functional dependencies among attributes within a relation is the main tool for formally measuring the appropriateness of attribute groupings into relation schemas.

10.1 Informal Design Guidelines for Relation Schemas

Four informal measures of quality for relation schema design.

- Semantics of the attributes.

- Reducing the redundant values in tuples.

- Reducing the null values in tuples.

- Disallowing the possibility of generating spurious tuples.

10.1.1 Semantics of the Relation Attributes

- The easier it is to explain the semantics of the relation, the better the relation schema design will be.

- 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, the meaning tends to be clear. Otherwise, the relation corresponds to a mixture of multiple entities and relationships and hence becomes semantically unclear.

- Example: A relation involves two entities – poor design.

EMP_DEPT

<table>
<thead>
<tr>
<th>ENAME</th>
<th>SSN</th>
<th>BDATE</th>
<th>ADDRESS</th>
<th>DNUMBER</th>
<th>DNAME</th>
<th>DMGRSSN</th>
</tr>
</thead>
</table>

1
10.1.2 Redundant Information in Tuples and Update Anomalies

- Grouping attributes into relation schemas has a significant effect on storage space. Compare two base relations EMPLOYEE and DEPARTMENT in Figure 10.2 (Fig 14.2 on e3) to an EMP.DEPT base relation in Figure 10.4 (Fig 14.4 on e3).

- Update anomalies for base relations EMP.DEPT and EMP.PROJ in Figure 10.4 (Fig 14.4 on e3).
  
  - Insertion anomalies: For EMP.DEPT relation in Figure 10.4 (Fig 14.4 on e3).
    * To insert a new employee tuple, we need to make sure that the values of attributes DNUMBER, DNAME, and DMGRSSN are consistent to other employees (tuples) in EMP.DEPT.
    * It is difficult to insert a new department that has no employees as yet in the EMP.DEPT relation.

  - Deletion anomalies: 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.

  - Modification anomalies: If we update the value of MGRSSN in a particular department, we must to update the tuples of all employees who work in that department; otherwise, the database will become inconsistent.

- GUIDELINE 2: Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations. If any anomalies are present, note them clearly and make sure the programs that update the database will operate correctly.

- It is advisable to use anomaly-free base relations and to specify views that include the JOINs for placing together the attributes frequently referenced to improve the performance.

10.1.3 Null Values in Tuples
• Having null values in tuples of a relation not only wastes storage space but also makes
the interpretation more difficult.

• 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 majority of tuples in the relation.

10.1.4 Generation of Spurious Tuples

• GUIDELINE 4: Design relation schemas so that they can be JOINed with equality
conditions on attributes that are either primary keys or foreign keys in a way that
guarantees that no spurious tuples are generated. Do not have the relations that
contains matching attributes other than foreign key - primary key combinations. If
such relations are unavoidable, do not join them on such attributes.

• For example, decomposing EMP.PROJ in Figure 10.4 (Fig 14.4 on e3) to EMP_LOCS
and EMP.PROJ1 in Figure 10.5 (Fig 14.5 on e3) is undesirable because spurious tuples
will be generated if NATURAL JOIN operation is performed (see Figure 10.6 (Fig 14.6
on e3)).

10.2 Functional Dependencies

Functional dependencies are the main tool for defining normal forms of relation schemas.

10.2.1 Definition of Functional Dependency

• A functional dependency (abbreviated as FD or f.d.), denoted by $X \rightarrow Y$, between two sets of attributes $X$ and $Y$ that are subsets of $R = \{A_1, A_2, \ldots, A_n\}$ specifies a constraint on the possible tuples that can form a relation state $r$ of $R$. The constraint is that, for any two tuples $t_1$ and $t_2$ in $r$ that have $t_1[X] = t_2[X]$, we must also have $t_1[Y] = t_2[Y]$.

• $X \rightarrow Y$ : $X$ functionally determines $Y$ or $Y$ is functionally dependent on $X$
• $X$ functionally determines $Y$ in a relation schema $R$ if and only if, whenever two tuples of $r(R)$ agree on their $X$-values, they must necessarily agree on their $Y$-values.

  - If $X$ is a candidate key of $R$, this implies that $X \rightarrow Y$ for any subset of attributes $Y$ of $R$.
  - If $X \rightarrow Y$ in $R$, this does not say whether or not $Y \rightarrow X$ in $R$.

• A functional dependency is a constraint that any relation extensions $r(R)$ must satisfy the functional dependency constraint at all times.

• Figure 10.3 (Fig 14.3 on e3) shows the diagrammatic notation for FDs.

### 10.2.2 Inference Rules for Functional Dependencies

• We denote by $F$ the set of functional dependencies that are specified on relation schema $R$. Typically, the schema designer specifies the FDs that are semantically obvious.

• It is practically impossible to specify all possible FDs that may hold in a relation schema. The set of all such FDs is called the closure of $F$ and is denoted by $F^+$.

• For example, let $F = \{SSN \rightarrow \{ENAME, BDATE, ADDRESS, DNUMBER\}, DNUMBER \rightarrow \{DNAME, DMGRSSN\}\}$. The following additional FDs can be inferred from $F$.

  $SSN \rightarrow \{DNAME, DMGRSSN\}$,
  $SSN \rightarrow SSN$,
  $DNUMBER \rightarrow DNAME$

• $\forall FD X \rightarrow Y \in F^+, X \rightarrow Y$ should hold in every relation state $r$ that is a legal extension of $R$.

• 6 well-known inference rules that can be used to infer new dependencies from a given set of dependencies $F$. ($F \models X \rightarrow Y$ denotes the FD $X \rightarrow Y$ is inferred from $F$.)

  - IR1 (reflexive rule): If $X \supseteq Y$, then $X \rightarrow Y$.
  - IR2 (augmentation rule): $\{X \rightarrow Y\} \models XZ \rightarrow YZ$. 

4
IR3 (transitive rule): \( \{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z \).

IR4 (decomposition, or projective rule): \( \{X \rightarrow YZ\} \models X \rightarrow Y \).

IR5 (union, or additive rule): \( \{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ \).

IR6 (pseudotransitive rule): \( \{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z \).

- A functional dependency \( X \rightarrow Y \) is **trivial** if \( X \supseteq Y \); otherwise, it is **nontrivial**.

- **Armstrong’s inference rules:** IR1, IR2, and IR3 are complete. That is, the set of dependencies \( F^+ \) can be determined from \( F \) by using only inference rules IR1 through IR3.

- The proofs for inference rules:

  - **Proof of IR1:** If \( X \supseteq Y \), then \( X \rightarrow Y \).
    Suppose that \( X \supseteq Y \) and that two tuples \( t_1 \) and \( t_2 \) exist in some relation instance \( r \) of \( R \) such that \( t_1[X] = t_2[X] \). Then \( t_1[Y] = t_2[Y] \) because \( X \supseteq Y \); hence, \( X \rightarrow Y \) must hold in \( r \).

  - **Proof of IR2:** \( \{X \rightarrow Y\} \models XZ \rightarrow YZ \).
    Assume that \( X \rightarrow Y \) holds in a relation instance \( r \) of \( R \) but that \( XZ \rightarrow YZ \) does not hold. Then there must exist two tuples \( t_1 \) and \( t_2 \) in \( r \) such that (1) \( t_1[X] = t_2[X] \), (2) \( t_1[Y] = t_2[Y] \), (3) \( t_1[XZ] = t_2[XZ] \), and (4) \( t_1[YZ] \neq t_2[YZ] \).
    This is not possible because from (1) and (3) we deduce (5) \( t_1[Z] = t_2[Z] \), and from (2) and (5) we deduce (6) \( t_1[YZ] = t_2[YZ] \), contradicting (4).

  - **Proof of IR3:** \( \{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z \).
    Assume that (1) \( X \rightarrow Y \) and (2) \( Y \rightarrow Z \) both hold in a relation \( r \). Then for any two tuples \( t_1 \) and \( t_2 \) in \( r \) such that \( t_1[X] = t_2[X] \), we must have (3) \( t_1[Y] = t_2[Y] \), from assumption (1); hence we must also have (4) \( t_1[Z] = t_2[Z] \), from (3) and assumption (2); hence \( X \rightarrow Z \) must hold in \( r \).

- All other inference rules (IR4 - IR6) can be proved by using IR1 through IR3.

  - **Proof of IR4:** \( \{X \rightarrow YZ\} \models X \rightarrow Y \).
    1. \( X \rightarrow YZ \) (given)
2. \( YZ \rightarrow Y \) (by IR1)
3. \( X \rightarrow Y \) (by IR3 and 1 and 2).

- **Proof of IR5:** \( \{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ \).
  1. \( X \rightarrow Y \) (given).
  2. \( X \rightarrow Z \) (given).
  3. \( XZ \rightarrow YZ \) (by IR2 and 1)
  4. \( X \rightarrow XZ \) (by IR2 and 2, note that \( XX = X \))
  5. \( X \rightarrow YZ \) (by IR3 and 3 and 4)

- **Proof of IR6:** \( \{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z \).
  1. \( X \rightarrow Y \) (given)
  2. \( WY \rightarrow Z \) (given)
  3. \( WX \rightarrow WY \) (by IR2 and 1)
  4. \( WX \rightarrow Z \) (by IR3 and 2 and 3)

- A systematic way to determine \( F^+ \): (1) determine each set of attributes \( X \) that appears as a left-hand side of some FDs in \( F \), (2) determine the set of all attributes that are dependent on \( X \).

- **Closure of \( X \) under \( F \) (denote \( X_F^+ \)):** The set of all attributes are functionally determined by \( X \) under \( F \).

  - **Algorithm 10.1** Determining \( X_F^+ \), the closure of \( X \) under \( F \)

    \[
    X_F^+ := X;
    \]
    
    repeat
    \[
    oldX_F^+ := X_F^+;
    \]
    
    for each functional dependency \( Y \rightarrow Z \) in \( F \) do
    \[
    \text{if } X_F^+ \supseteq Y \text{ then } X_F^+ := X_F^+ \cup Z;
    \]
    
    until \( (X_F^+ = oldX_F^+) \);

- Example: Let \( R(A, B, C, D, E, F) \) and
  \[
  F = \{A \rightarrow D, B \rightarrow EF, AB \rightarrow C\}.
  \]
  Then
What is the closure of \{A\} under \(F\)?
\[
X_F^+ = A_F^+ = A; \text{old}X_F^+ = A
\]
\[
X_F^+ = AD \quad \text{old}X_F^+ = AD
\]
\[
X_F^+ = AD
\]

What is the closure of \{B\} under \(F\)?
\[
X_F^+ = B_F^+ = B; \text{old}X_F^+ = B
\]
\[
X_F^+ = BEF \quad \text{old}X_F^+ = BEF
\]
\[
X_F^+ = BEF
\]

What is the closure of \{A, B\} under \(F\)?
\[
X_F^+ = (AB)_F^+ = AB; \text{old}X_F^+ = AB
\]
\[
X_F^+ = ABDEFC \quad \text{old}X_F^+ = ABDEFC
\]
\[
X_F^+ = ABDEFC
\]

What is \(F^+\)?
\[
\{A\}_F^+ = \{A, D\} \quad 3 \text{ functional dependencies}
\]
\[
\{B\}_F^+ = \{B, E, F\} \quad 7 \text{ functional dependencies}
\]
\[
\{A, B\}_F^+ = \{A, B, D, E, F, C\} \quad 31 \text{ functional dependencies}
\]
\[
F^+ = \{A\}_F^+ \cup \{B\}_F^+ \cup \{A, B\}_F^+ \quad 41 \text{ functional dependencies}
\]

- Another example, let \(R(A, B, C, D, E, I)\) and
\[
F = \{A \rightarrow D, \ AD \rightarrow E, \ BI \rightarrow E, \ CD \rightarrow I, \ E \rightarrow C\}.
\]

Then

What is the closure of \{A, E\} under \(F\)?
\[
X_F^+ = (AE)_F^+ = AE; \text{old}X_F^+ = AE
\]
\[
X_F^+ = AEDC \quad \text{old}X_F^+ = AEDC
\]
\[
X_F^+ = AEDCI \quad \text{old}X_F^+ = AEDCI
\]
\[ X_F^+ = \text{AEDCI} \]

- Testing membership with respect to \( F \), i.e., does \( F \vdash X \rightarrow Y \) ?

- **Algorithm 10.1A** MEMBER(\( X \rightarrow Y, \ F \) : BOOLEAN

  **Input:** A set \( F \) of functional dependencies and a functional dependency \( X \rightarrow Y \).
  
  **Output:** TRUE if \( F \vdash X \rightarrow Y \); FALSE otherwise.

  **begin**
  
  If \( X_F^+ \supseteq Y \) then return TRUE else return FALSE
  
  **End**

- Example: let \( F = \{ A \rightarrow D, \ AD \rightarrow E, \ BI \rightarrow E, \ CD \rightarrow I, \ E \rightarrow C \} \).
  
  Then does \( F \vdash \{ A, B \} \rightarrow \{ E \} \)?
  
  Find \( \{ A, B \}^+_F : (AB)^+_F = \text{ABDECI} \)
  
  Since \( \{ A, B, D, E, C, I \} \supseteq \{ E \} \Rightarrow F \vdash \{ A, B \} \rightarrow \{ E \} \)

10.2.3 Equivalence of Sets of Functional Dependencies

- A set of functional dependencies \( E \) is **covered by** a set of functional dependencies \( F \) if every FD in \( E \) is also in \( F^+ \).

- Two sets of functional dependencies \( E \) and \( F \) are **equivalent** if \( E^+ = F^+ \).

- Alternatively, \( E \) is equivalent to \( F \) if both \( E \) covers \( F \) and \( F \) covers \( E \) hold.

- Algorithm for determining whether \( F \) covers \( E \):

  - **Algorithm 10.1B** Determining whether \( F \) covers \( E \).
    
    for each functional dependency \( X \rightarrow Y \in E \)
    
    do if (not MEMBER(\( X \rightarrow Y, \ F \))
    
    then return false;
    
    return true;

- For example, let
  
  \( F_1 = \{ A \rightarrow BC, \ A \rightarrow D, \ CD \rightarrow E \} \)
\[ F_2 = \{ A \rightarrow BCE, \ A \rightarrow ABD, \ CD \rightarrow E \} \]
\[ F_3 = \{ A \rightarrow BCDE \} \]

Then

Is \( F_1 \) equivalent to \( F_2 \)?
Since \( F_1 \models F_2 \) and \( F_2 \models F_1 \Rightarrow F_1 \equiv F_2 \)

Is \( F_1 \) equivalent to \( F_3 \)?
\( F_1 \models F_3 \) but \( F_3 \not\models F_1 \Rightarrow F_1 \not\equiv F_3 \)

### 10.2.4 Minimal Sets of Functional Dependencies

- A set of functional dependencies \( F \) is **minimal** if it satisfies the following conditions:
  1. Single attribute on right-hand side for every FD in \( F \).
  2. \( F - \{ X \rightarrow A \} \cup \{ Y \rightarrow A \} \) is not equivalent to \( F \), for all \( Y \) that is a proper subset of \( X \) and for all \( X \rightarrow A \) in \( F \).
  3. \( F - \{ X \rightarrow A \} \) is not equivalent to \( F \), for all \( X \rightarrow A \) in \( F \).

- A **minimal cover** of a set of functional dependencies \( F \) is a minimal set of dependencies \( F_{\text{min}} \) that is equivalent to \( F \).

- Algorithm for finding a minimal cover \( G \) for \( F \).

  - **Algorithm 10.2** Finding a minimal cover \( G \) for \( F \)
    1. Set \( G := F \).
    2. Replace each functional dependency \( X \rightarrow \{ A_1, A_2, \ldots, A_n \} \) in \( G \) by the \( n \) functional dependencies \( X \rightarrow A_1, \ X \rightarrow A_2, \ldots, \ X \rightarrow A_n \).
    3. For each functional dependency \( X \rightarrow A \) in \( G \)
        for each attribute \( B \) that is an element of \( X \)
        if \( ((G - \{ X \rightarrow A \}) \cup \{(X - \{B\}) \rightarrow A\}) \) is equivalent to \( G \),
        then replace \( X \rightarrow A \) with \( (X - \{B\}) \rightarrow A \) in \( G \).
    4. For each remaining functional dependency \( X \rightarrow A \) in \( G \)
if \((G - \{X \to A\})\) is equivalent to \(G\),
then remove \(X \to A\) from \(G\).

- **Example:** find a minimal cover \(G\) of \(F = \{A \to BC, B \to C, AB \to D\}\).

  - **Step 1:**
    Let \(G = \{A \to BC, B \to C, AB \to D\}\).

  - **Step 2:**
    \(G = \{A \to B, A \to C, B \to C, AB \to D\}\).

  - **Step 3:**
    * Can \(A\) be eliminated from \(AB \to D\) from \(G\)?
      Let \(G_1 = \{A \to B, A \to C, B \to C, B \to D\}\).
      Is \(G_1 \equiv G\)?  \(\Rightarrow\) Is \(G_1\) cover \(G\) and Is \(G\) cover \(G_1\)?  \(\Rightarrow\) \(G_1 \models AB \to D?\ and \ G \models B \to D?\)
      Since \((AB)^+_G = \{A, B, C, D\} \supseteq \{D\} \Rightarrow G_1 \models AB \to D\)
      Since \((B)^+_G = \{B, C\} \not\supseteq \{D\} \Rightarrow G \not\models B \to D\)
      Therefore, \(G_1 \not\equiv G\) and \(A\) can not be eliminated from \(AB \to D\) in \(G\).

    * Can \(B\) be eliminated from \(AB \to D\) from \(G\)?
      Let \(G_1 = \{A \to B, A \to C, B \to C, A \to D\}\).
      Is \(G_1 \equiv G\)?  \(\Rightarrow\) Is \(G_1\) cover \(G\) and Is \(G\) cover \(G_1\)?  \(\Rightarrow\) \(G_1 \models AB \to D?\ and \ G \models A \to D?\)
      Since \((AB)^+_G = \{A, B, C, D\} \supseteq \{D\} \Rightarrow G_1 \models AB \to D\)
      Since \((A)^+_G = \{A, B, C, D\} \supseteq \{D\} \Rightarrow G \models B \to D\)
      Therefore, \(G_1 \equiv G\) and \(B\) can be eliminated from \(AB \to D\) in \(G\).

  - **Step 4:** \(G = \{A \to B, A \to C, B \to C, A \to D\}\).

    * Can \(A \to B\) be eliminated from \(G\)?
      Let \(G_1 = \{A \to C, B \to C, A \to D\}\).
      Is \(G_1 \equiv G\)?  \(\Rightarrow\) Is \(G_1\) cover \(G\) and Is \(G\) cover \(G_1\)\?  \(\Rightarrow\) \(G_1 \models A \to B?\)
      Since \((A)^+_G = \{A, C, D\} \not\supseteq \{B\} \Rightarrow G_1 \not\models A \to B\)
      Therefore, \(A \to B\) can not be eliminated from \(G\).
* Can $A \rightarrow C$ be eliminated from $G$?
* Can $B \rightarrow C$ be eliminated from $G$?
* Can $A \rightarrow D$ be eliminated from $G$?

- Another Example: Find a minimal cover $G$ for $F = \{A \rightarrow C, AC \rightarrow J, AB \rightarrow DE, AB \rightarrow CDI, C \rightarrow M, A \rightarrow M\}$.

10.2.5 Candidate Keys and Superkeys

Let $R = (U, F)$ be a relation schema, where $U$ is the set of attributes and $F$ is the set of functional dependencies.

- A subset $X$ of $U$ is a **superkey** for a relation schema $R$ if $(X \rightarrow U)$ is in $F^+$, i.e., $X_F^+ = U$.

- A subset $X$ of $U$ is a **candidate key** for $R$ if $X$ is a superkey and no proper subset $Y$ of $X$ is a superkey.

  That is, if $X$ is a **candidate key**, then $(X_F^+ = U)$ and ($\nexists Y$ such that $Y \subset X$ and $Y_F^+ = U$).

- Example: Is $(ABE)$ a superkey for $R = \{U, F\}$, where $U = \{A, B, C, D, E, I\}$ and $F = \{A \rightarrow D, AD \rightarrow E, BI \rightarrow E, CD \rightarrow I, E \rightarrow C\}$

  - Is $F \models X \rightarrow U$?
  
    $(ABE)_F^+ = ABEDCI = U$, therefore, $ABE \rightarrow U$ and $ABE$ is a superkey for $R$.

- Example: Is $(ABE)$ a candidate key for $R$? Where $R$ is as previous example.

  - We know $ABE$ is a superkey.
  
    - $A_F^+ = ADECI \neq U$
    
    - $B_F^+ = B \neq U$
    
    - $E_F^+ = EC \neq U$
    
    - $AB_F^+ = ABDECI = U \Rightarrow ABE$ is not a candidate key.
    
    - $AE_F^+ =$
\[ BE_F^+ = \]
\[ ABE_F^+ = \]

- How about finding all candidate keys for \( R = \{U, F\} \), where \( U = \{A, B, C, D, E\} \) and \( F = \{AB \rightarrow E, E \rightarrow AB, EC \rightarrow D\} \).
  - Find all candidate keys containing 1 attribute.
  - Find all candidate keys containing 2 attributes.
  - Find all candidate keys containing 3 attributes.
  - Find all candidate keys containing 4 attributes.
  - Find all candidate keys containing 5 attributes.

### 10.3 Normal Forms Based on Primary Keys

#### 10.3.1 Normalization of Relations

- The **normalization process** takes a relation schema through a series of tests to “certify” whether it satisfies a certain **normal form**. If the schema does not meet the normal form test, the relation is decomposed into smaller relation schemas that meet the tests.

- The purpose of **normalization** is to analyze the given relation schemas based on their FDs and primary keys to achieve the desirable properties of (1) minimizing redundancy and (2) minimizing the insertion, deletion, and update anomalies.

- 1NF, 2NF, 3NF, and BCNF are based on the functional dependencies among the attributes of a relation.

- 4NF is based on multivalued dependencies; 5NF is based on join dependencies.

- Database design as practiced in industry today pays particular attention to normalization only up to 3NF or BCNF (sometimes 4NF).
• An attribute of relation schema \( R \) is called a \textbf{prime attribute} of \( R \) if it is a member of some candidate key of \( R \). An attribute is called \textbf{nonprime} if it is not a prime attribute.

### 10.3.4 First Normal Form

• 1NF: the domain of an attribute must include only \textit{atomic values} and the value of any attribute in a tuple must be a \textit{single value} from the domain of that attribute.

• Example: Suppose we extend the DEPARTMENT relation in Figure 10.1 (Fig 14.1 on e3) by including the DLOCATIONS attributes as in Figure 10.8 (Fig 14.8 on e3).

  – Two cases:
    * Case1: 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 DNUMBER.
    * Case2: the domain of DLOCATIONS contains sets of values and hence is nonatomic. In this case, \( DNUMBER \rightarrow DLOCATIONS \).
    * Both cases violate the first normal form.

  – There are three decomposition (normalization) approaches for first normal form:
    * 1. Decompose \( \text{DEPARTMENT(DNAME, DNUMBER, DMGRSSN, DLOCATIONS)} \) into two relations \( \text{DEPARTMENT(DNAME, DNUMBER, DMGRSSN)} \) and \( \text{DEPT_LOCS(DNUMBER, DLOCATIONS)} \)
    * 2. Expand the key so that there will be a separate tuple in the original \( \text{DEPARTMENT} \) relation for each location of a department, as shown in Figure 10.8(c) (Fig 14.8(c) on e3).
      Introduce \textit{redundancy}.
    * 3. If it is known that at most three locations can exist for a department, then replace DLOCATIONS to three attributes: DLOCATION1, DLOCATION2, and DLOCATION3.
      Introduce \textit{null} values.
10.3.5 Second Normal Form

- 2NF is based on the concept of fully functional dependency.
- $X \rightarrow Y$ is a **full functional dependency** if for any attribute $A \in X$, $(X \setminus \{A\}) \not\rightarrow Y$.
- $X \rightarrow Y$ is a **partial functional dependency** if for some attribute $A \in X$, $(X \setminus \{A\}) \rightarrow Y$.
- 2NF: a relation schema $R$ is in 2NF if every nonprime attribute $A$ in $R$ is fully functionally dependent on the primary key of $R$.
- For example, the EMP_PROJ relation in Figure 10.10(a) (Fig 14.10(a) on e3) is in 1NF but not in 2NF.
  - Decomposition (Normalization) to 2NF:
    The FD1, FD2, and FD3 in Figure 10.10(a) (Fig 14.10(a) on e3) lead to the decomposition of EMP_PROJ into three relation schemas EP1, EP2, EP3.

10.3.6 Third Normal Form

- 3NF is based on the concept of transitive dependency.
- $X \rightarrow Y$ in a relation schema $R$ is a **transitive dependency** if there is a set of attributes $Z$ that is neither a candidate key nor a subset of any key of $R$, and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.
- 3NF: a relation schema $R$ is in 3NF if it satisfies 2NF and no nonprime attribute of $R$ is transitively dependent on the primary key.
- For example, the EMP_DEPT relation in Figure 10.10(b) (Fig 14.10(b) on e3) is in 2NF but not in 3NF.
  - Decomposition (Normalization) to 3NF:
    Decompose EMP_DEPT into ED1 and ED2 as shown in Figure 10.10(b) (Fig 14.10(b) on e3).
10.4 General Definitions of Second and Third Normal Forms

- In previous section, the definitions of 2NF and 3NF do not take other candidate keys into consideration. In this section, we give more general definitions of 2NF and 3NF that take all candidate keys of a relation into account.

10.4.1 General Definition of Second Normal Form

- 2NF: a relation schema $R$ is in 2NF if every nonprime attribute $A$ in $R$ is fully functionally dependent on every key of $R$.

- For example, the relation schema LOTS shown in Figure 10.11(a) (Fig 14.11(a) on e3), which has two candidate keys: PROPERTY_ID# and {COUNTY_NAME, LOTS#}. FD3 in LOTS violates 2NF. LOTS is decomposed into two relation schemas LOTS1 and LOTS2 as shown in Figure 10.11(b) (Fig 14.11(b) on e3).

10.4.2 General Definition of Third Normal Form

- 3NF: a relation schema $R$ is in 3NF if for any nontrivial functional dependency $X \rightarrow A$ holds in $R$, either (a) $X$ is a superkey of $R$, or (b) $A$ is a prime attribute of $R$.

- For example, the FD4 of LOTS1 in Figure 10.11(b) (Fig 14.11(b) on e3) violates the 3NF because AREA is not a superkey and PRICE is not a prime attribute in LOTS1. LOTS1 is decomposed into two relation schemas LOTS1A and LOTS1B as shown in Figure 10.11(c) (Fig 14.11(c) on e3).

10.5 Boyce-Codd Normal Form

- BCNF is a stronger normal form than 3NF. That is, every relation in BCNF is also in 3NF; however, a relation in 3NF is not necessarily in BCNF.

- BCNF: a relation schema $R$ is in BCNF if whenever a nontrivial functional dependency $X \rightarrow A$ holds in $R$, then $X$ is a superkey of $R$. 

15
• For example, consider the LOTS relation schema in Figure 10.11 (Fig 14.11 on e3). If there is another functional dependency FD5: AREA → COUNTY, NAME. LOTS1A is still in 3NF but not in BCNF. LOTS1A can be decomposed into two relation schemas LOTS1AX and LOTS1AY as shown in Figure 10.12 (Fig 14.12 on e3). Notice that, FD2 is lost in this decomposition.

• In practice, most relation schemas that are in 3NF are also in BCNF. Only if $X \rightarrow A$ holds in $R$ with $X$ not being a superkey and $A$ being a prime attribute will $R$ be in 3NF but not in BCNF. Figure 10.12(b) (Fig 14.12(b) on e3) shows the relation in the case.

• A relation not in BCNF should be decomposed so that to meet this property, while possibly forgoing the preservation of all functional dependencies and also a test is necessary to determine whether the decomposition is nonadditive (lossless), i.e., it will not generate spurious tuples after a join. See the example in Figure 10.13 (Fig 14.13 on e3).