LOGICAL DATABASE DESIGN Part #2/2

Relational Database Design

• Preliminary Remarks:

There are always many alternative approaches to database design; some viable, some not.

Example of a bad database design:

supplier (S_NAME, ADDR, PART#, PRICE)
- Problem#1: Repetition of Information - supplier address is unnecessarily repeated for every supplied part.
- Problem#2: Update Anomaly - an address change should be reflected in all tuples involving the supplier; but will it?
- Problem#3: Insertion Anomaly - impossible to store the name and address of a supplier, unless it supplies at least one part.
- Problem#4: Deletion Anomaly - opposite of insertion anomaly, i.e. if all supplied parts are deleted, we will unwillingly delete the supplier (i.e., the name and address).
Example of a good database design: Decomposition

supplier (S_NAME, ADDR)
supplies (S_NAME, PART#, PRICE)

- No repetition of information - each supplier is represented by only one tuple (Note, the repetition of S_NAME in supplies relation is necessary!)
- No update anomaly - address change involved only one tuple.
- No insertion anomaly - one can insert the name and address of a new supplier, even though the supplier supplies no known part.
- No deletion anomaly - all parts supplied by a supplier can be deleted, while retaining the name and the address of the supplier in the DB.

Some problems exist in this design, but not in the previous one; e.g. expensive join operation.

Questions to be answered:
1) When is the decomposition advantageous?
2) Does the new design suffer from the same problems as the original one?
3) How can optimal decomposition be achieved?

• Schema Decomposition

Definition: Decomposition of a schema R=A1,A2,...,An is a set of schemas \{R_1, R_2, ..., R_k\}, s.t. \( \forall R_i \subseteq R \) and \( R = R_1 \cup R_2 \cup ... \cup R_k \).

Motive: to eliminate information repetition and anomalies.

Two problems can occur when performing schema decomposition:
- Loss of information and Loss of FDs.
Problem #1 of 2: Loss of Information

Definition: Let \( r(R) \) be a relation, and let \( d=\{R_1, R_2, ..., R_k\} \) be a decomposition of \( r(R) \). Let \( F \) be a set of FDs of \( r(R) \). The decomposition \( d \) causes loss of information (w.r.t. \( F \)) if

\[
r \neq \prod_{i=1}^{k} R_i(r) \end{linenomath}

Example:

directory (NAME, PHONE, CITY)

\begin{array}{ccc}
\text{n1} & \text{p1} & \text{c1} \\
\text{n1} & \text{p2} & \text{c2} \\
\text{n2} & \text{p3} & \text{c1}
\end{array}

\text{NAME} --> \text{PHONE} \notin F+ \text{ and } \text{NAME} --> \text{CITY} \notin F+

Let \( r_1=\Pi_{\text{NAME, PHONE}} \) (directory) and \( r_2=\Pi_{\text{NAME, CITY}} \) (directory):

\[
r_1 (\text{NAME, PHONE}), \quad r_2 (\text{NAME, CITY})
\end{linenomath}

\begin{array}{ccc}
\text{n1} & \text{p1} & \text{n1} & \text{c1} \\
\text{n1} & \text{p2} & \text{n1} & \text{c2} \\
\text{n2} & \text{p3} & \text{n2} & \text{c1}
\end{array}

Let \( r' = r_1 \bowtie r_2 \) (i.e., natural join)

\[
r' (\text{NAME, PHONE, CITY})
\end{linenomath}

\begin{array}{ccc}
\text{n1} & \text{p1} & \text{c1} \\
\text{n1} & \text{p1} & \text{c2} \\
\text{n1} & \text{p2} & \text{c1} \\
\text{n1} & \text{p2} & \text{c2} \\
\text{n2} & \text{p3} & \text{c1}
\end{array}
r' has more tuples but less information, since we cannot tell in which city person n1 has phone p1 and in which city he has phone p2. However, if the FDs NAME→PHONE or NAME→CITY hold on the original schema, this decomposition will be good.

Testing Algorithm

See Appendix.

- Problem #2 of 2: Loss of FDs

Projection of F on a set of attributes Z, denoted by Π_Z(F), is a set of FDs X→Y∈F+ s.t. X∪Y ⊆Z

A decomposition d={R1,R2,...,Rk} preserves FDs in F if all FDs in F can be logically derived from the set of FDs in G=Π_{R1}(F)∪Π_{R2}(F)∪...∪Π_{Rk}(F), i.e. if F⊆G+

Recall: FDs present constraints on a relation r(R). Thus, a loss of FDs implies a potential loss of some constraints on the DB.
Algorithm: Test "loss of FDs" problem (FD preservation): Use this for BCNF decomposition.

Input: F and d
Output: "Yes" if no loss of FDs; "No" otherwise

Method:

\[
\text{begin} \\
\quad \text{for every FD } X \rightarrow Y \in F \text{ do} \\
\quad \quad \text{begin} \\
\quad \quad \quad Z = X; \\
\quad \quad \quad \text{while}(Z \neq Z') \text{ do} \\
\quad \quad \quad \quad \text{begin} \\
\quad \quad \quad \quad \quad Z' = Z \\
\quad \quad \quad \quad \quad \text{for } i = 1 \text{ to } k \text{ do} \\
\quad \quad \quad \quad \quad \quad \text{begin} \\
\quad \quad \quad \quad \quad \quad \quad Z = Z \cup (\{Z \cap R_i\}^+ \cap R_i); \\
\quad \quad \quad \quad \quad \quad \quad //X^+ \text{ computation is in LectureNote#5} \\
\quad \quad \quad \quad \quad \quad \text{end} \\
\quad \quad \quad \quad \text{end} \\
\quad \quad \quad \quad \text{if } (Y \not\subseteq Z \land Y \neq Z) \text{ then return } \text{"No"} \\
\quad \text{end} \\
\quad \text{return } \text{"Yes"} \\
\text{end}
\]

Example:

\[
R = ABCD, \ d = \{AB, BC, CD\}, \ F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}
\]

Obviously, A→B, B→C, and C→D are preserved. How about D→A?

\[
\text{initially } Z = Z' = \{D\}; \\
(i=1) \ Z = \{D\} \cup ((\{D\} \cap \{A,B\})^+ \cap \{A,B\}) = \{D\} \cup \emptyset = \{D\} \\
(i=2) \ Z = \{D\} \cup ((\{D\} \cap \{B,C\})^+ \cap \{B,C\}) = \{D\}
\]
(i=3) \( Z = \{ D \} \cup ((\{ D \} \cap \{ C, D \}) \cap \{ C, D \}) = \{ D \} \cup (\{ A, B, C, D \} \cap \{ C, D \}) = \{ C, D \} \)

\( Z = \{ C, D \} \neq \{ D \} = Z' \)

(i=1) \( Z = \{ C, D \} \cup ((\{ C, D \} \cap \{ A, B \}) \cap \{ A, B \}) = \{ C, D \} \)

(1=2) \( Z = \{ C, D \} \cup ((\{ C, D \} \cap \{ B, C \}) \cap \{ B, C \}) = \{ C, D \} \cup (\{ A, B, C, D \} \cap \{ B, C \}) = \{ B, C, D \} \)

(i=3) \( Z = \{ B, C, D \} \cup ((\{ B, C, D \} \cap \{ C, D \}) \cap \{ C, D \}) = \{ B, C, D \} \)

\( Z = \{ B, C, D \} \neq \{ C, D \} = Z' \)

(i=1) \( Z = \{ B, C, D \} \cup ((\{ B, C, D \} \cap \{ A, B \}) \cap \{ A, B \}) = \{ A, B, C, D \} \)

(i=2) \( Z = \{ A, B, C, D \} \cup ((\{ A, B, C, D \} \cap \{ B, C \}) \cap \{ B, C \}) = \{ A, B, C, D \} \)

(i=3) \( Z = \{ A, B, C, D \} \cup ((\{ A, B, C, D \} \cap \{ C, D \}) \cap \{ C, D \}) = \{ A, B, C, D \} \)

\( Z = \{ A, B, C, D \} \neq \{ B, C, D \} = Z' \)

(i=1) \( Z = \{ A, B, C, D \} \cup ((\{ A, B, C, D \} \cap \{ A, B \}) \cap \{ A, B \}) = \{ A, B, C, D \} \)

(i=2) \( Z = \{ A, B, C, D \} \cup ((\{ A, B, C, D \} \cap \{ B, C \}) \cap \{ B, C \}) = \{ A, B, C, D \} \)

(i=3) \( Z = \{ A, B, C, D \} \cup ((\{ A, B, C, D \} \cap \{ C, D \}) \cap \{ C, D \}) = \{ A, B, C, D \} \)

\( Z = \{ A, B, C, D \} = \{ A, B, C, D \} = Z' \) => return "Yes"
• **Normal Forms (Normalization)**
Represent restrictions on relational DBs aimed at preventing anomalies

Classification: 1NF, 2NF, 3NF, BCNF, 4NF ...

• **First Normal Form (1NF)**
Definition: A relation r(R) is in 1NF if all tuples contain single-valued fields (i.e., a field cannot contain a set of values from its domain). A relational DB is in 1NF if all relations are in 1NF.

Advantages:

(1) Without the 1NF restriction, it would be impossible to express FDs.

(2) Without the 1NF, the update would be complicated

Example:

graduates ( STUDENT# F_NAME YEAR )
{001, 005} {John, Peter} 1997
{003, 010, 101} {Ann, Ivan, Sam} 2001

Problem(1): STUDENT# --> F_NAME?
Problem(2): postpone Ivan's graduation to 2002.

With the following 1NF, those problems do not occur:

graduates ( STUDENT# F_NAME YEAR )
001 Peter 1997
005 John 1997
. . .
Remaining Problems: Recall, the supplier relation (Lecture Note #6, P. 1) is in 1NF and there are four problems (i.e., repetition of info. and anomalies). The Partial FD causes these problems.

A relation has at least one partial FD if it is in 1NF but not in 2NF

Partial FDs:

Definition: Let r(R) and X, Y ⊂ R. A non-trivial FD X→Y is a full FD if ∀X' ⊂ X, s.t. X'→Y. Otherwise, if ∃X' ⊂ X, s.t. X'→Y, then X→Y is a partial FD.

Example: supplier (S_NAME, ADDR, PART#, PRICE),

F{S_NAME→ADDR, S_NAME P#→P}

S_NAME P#→ADDR is a partial FD!

ADDR is not fully dependent on the primary key! Therefore, an ADDR value must be repeated for all tuples whose S_NAME values are the same.

By eliminating every partial FD, one can eliminate some anomalies!
• **Second Normal Form (2NF)**

Introduced to eliminate partial FDs.

Definition: Let attributes of candidate keys be called primary attributes, and all others secondary attributes. A relation \( r(R) \) is in 2NF iff it is in 1NF and every secondary attribute is fully dependent on the candidate keys of the relation \( r(R) \).

A relational DB is in 2NF if every relation is in 2NF.

Example:

supplier \((S\_NAME,\ ADDR,\ PART#,\ PRICE)\)

\( F=\{S\_NAME--> ADDR,\ S\_NAME\ PART# -->\ PRICE\}\)

\( K=\{S\_NAME,\ PART#\} \) // the only candidate key of this attribute

This relation is not in 2NF, since:

1. \( \{S\_NAME,\ PART#\} \) is a key and
2. this key partially determines ADDR (i.e., \( S\_NAME\ PART# --> ADDR \) is a partial FD).

supplier \((S\_NAME,\ ADDR)\)

\( F1=\{S\_NAME --> ADDR\}\)

\( K1=\{S\_NAME\} \) // the only candidate key of this relation

supplies \((S\_NAME,\ PART#,\ PRICE)\)

\( F2=\{S\_NAME\ PART# --> PRICE\}\)

\( K2=\{S\_NAME,\ PART#\} \) // the only candidate key of this relation

These relations are in 2NF, since:

1. In "supplier", the secondary attribute is fully dependent on \( K1 \)
2. In "supplies", the secondary attribute is fully dependent on \( K2 \)
If r(R) has no secondary attributes, then it is automatically in 2NF.

If, in r(R), all candidate keys have a single attribute, then it is automatically in 2NF.

Note that, by the definition of 2NF, primary attributes need not be fully dependent on candidate keys. For example,

\[ r(ABCDEG) \] is in 1NF.

\[ F = \{ ABC \rightarrow DE, DE \rightarrow ABC, DE \rightarrow G, AB \rightarrow D, E \rightarrow C \} \]

\[ K = \{ D,E \}, \{ A,B,C \}, \{ A,B \} \]

The only secondary attribute G is fully dependent on DE, ABC, and ABE. Thus, the relation is in 2NF.

However, DE \rightarrow C is a partial FD since E \rightarrow C. Also, ABC \rightarrow D is a partial FD since AB \rightarrow D.

Algorithm: 2NF Decomposition Algorithm (**NO PRACTICAL USE**).

Input: r(R) not in 2NF but in 1NF and F
Output: d = \{ R1, R2, ..., Rk \}, where Ri \subset R, for all i = 1,...,k.
Method:

\[
\begin{align*}
\text{begin} \\
(1) & \text{ R1 = R; } i=1; \ T = \{ \} \ //\text{note, } T \text{ is a set of sets} \\
(2) & \text{ Find an FD } X' \rightarrow Y \text{ such that } X' \not\in T \text{ is a subset of a candidate key and } \\
& \text{ Y } \subseteq R \text{ is the set of all the secondary attributes that are determine by } \\
& \text{ (dependent on) } X'; \text{ Add } X' \text{ to } T; \\
(3) & \text{ if } (i == 1 \text{ or } (X'Y \not\subset Rj \text{ and } X'Y \not= Rj, \text{ for all } j=2,...,i)) \text{ then} \\
& i=i+1; \\
& R1=R1-Y; \ Ri=X'Y; \\
& (\text{note, } F1=\prod_{R1}(F), \ Fi=\prod_{Ri}(F)); 
\end{align*}
\]
(4) Repeat Steps (2)&(3) until there is no FD $X' \rightarrow Y$ such that $X' \notin T$ is a subset of a candidate key and $Y \subseteq R$ is the set of all the secondary attributes that are determined by (dependent on) $X'$.

(5) for $j=1$ to $i$ { if $R_j$ is not in 2NF, call this algorithm recursively; }
end

Example 1 *(Revised 4/16/2002)*:

supplier $(S\_NAME\ ADDR\ PART\#\ PRICE)$

$F=\{S\_NAME--\rightarrow ADDR, S\_NAME\ PART\# \rightarrow PRICE\}$

$K=\{S\_NAME, PART\#\}$ //the only candidate key

$R_1=\{S\_NAME, ADDR, PART\#, PRICE\}$

$S\_NAME \rightarrow ADDR$

Update $R_1=\{S\_NAME, PART\#, PRICE\}$

Create $R_2=\{S\_NAME, ADDR\}$

output $\{S\_NAME, PART\#, PRICE\}, \{S\_NAME, ADDR\}$

Both s1 and s2 are in 2NF.

We can show that this decomposition is lossless and FD preserving using the algorithms in this lecture note. Try it.

Example 2 *(Added 4/16/2002)*:

$R=\{A,B,C,D,E\}, F=\{A \rightarrow C, B \rightarrow C, C \rightarrow D, AB \rightarrow E\}$

CKs: $\{AB\}$ //the only candidate key

$R_1=\{A,B,C,D,E\}$

$A \rightarrow CD$:

Update $R_1=\{A,B,C,D,E\}-\{C,D\}=\{A,B,E\}$

Create $R_2=\{A,C,D\}$

$B \rightarrow CD$:

Update $R_1=\{A,B,E\}-\{C,D\}=\{A,B,E\}$

Create $R_3=\{B,C,D\}$

output: $\{A,B,E\}, \{A,C,D\}, \{B,C,D\}$ (all the schemas are in 2NF)
**Remaining Problems**: Restriction imposed by 2NF is not rigorous enough to prevent all anomalies!

Example: Consider the following relation in 2NF

department (DEPT D_OFFICE SECRETARY)

\[ F=\{\text{SECRETARY} \rightarrow \text{DEPT}, \text{DEPT} \rightarrow \text{D_OFFICE}\} \]

Note, DEPT \( \not\rightarrow \) SECRETARY, D_OFFICE \( \not\rightarrow \) SECRETARY

That is, a department can have more than one secretary working in the department office. Each secretary must work in one department. Each department has only one office.

(1) Repetition of Information: D_OFFICE must be unnecessary repeated for every secretary working in the same DEPT.

(2) Update Anomaly: If the department office changes its address, many tuples must be updated.

(3) Insertion Anomaly: New department cannot be inserted before at least one secretary is assigned.

(4) Deletion Anomaly: If all secretary of a department retire, the department will be deleted.

**Transitive Dependencies**

Definitions: Let \( r(R) \) and \( X,Y,Z \subset R \). Then, \( X \rightarrow Z \) is a transitive dependency iff \( X \rightarrow Y \), \( Y \rightarrow Z \), and \( Y \not\rightarrow X \) hold and \( Y \rightarrow Z \) is a non-trivial FD (therefore, \( Z \not\rightarrow X \), but \( Z \rightarrow Y \) is not prohibited)

\[
\begin{array}{c}
X \\
\uparrow \\
Y \\
\downarrow \\
Z
\end{array}
\]
Example: department (DEPT D_OFFICE SECRETARY)

SECRETARY --> DEPT
DEPT --> D_OFFICE
DEPT -/-> SECRETARY

Note: If Z is a set of secondary attributes of R, and Y-->Z (where, Y is not a candidate key), then there is a transitive dependency X-->Z for every candidate key X on R (see the above schema where SECRETARY is the only key)

By eliminating every transitive FD, we can eliminate some anomalies!
• **Third Normal Form (3NF)**

Definition: \( r(R) \) is in 3NF iff it is in 1NF and no secondary attribute of \( r \) is transitively dependent on a candidate key of \( r \). A relational DB is in 3NF if every relation is in 3NF.

Lemma 1: If \( r(R) \) is in 3NF, then it is in 2NF.

Proof: Let \( F \) be the set of FDs on \( R \). Prove that \( r(R) \) has no transitive FD of secondary attributes (\( Z \)) on a candidate key (\( X \)), then it has no partial FD violating 2NF.

Suppose \( r(R) \) is in 3NF, but there is a partial FD \( X \rightarrow Z \), s.t. \( \exists X' \rightarrow Z \ (X' \subset X) \), \( X \) is a candidate key on \( R \), and \( Z \) consists of secondary attributes. Then \( X \rightarrow X' \), \( X' \rightarrow Z \), \( X' \rightarrow X \), and \( X' \rightarrow Z \) is non-trivial. Therefore, there is a transitive FD. Contradiction!

Lemma 2: If \( r(R) \) is in 2NF, then it need not be in 3NF.

(see the previous "department" example)

3NF Normalization: If \( r(R) \) is not in 3NF, then \( \exists \) lossless and FD preserving decomposition \( d \) of \( r(R) \) into a set of 3NF relations.

Algorithm: A lossless and FD preserving 3NF Normalization

(Revised 4/16/2002)

Input: \( r(R) \) not in 3NF and \( F \)

Output: \( d=(R_1,R_2,\ldots,R_k) \), where each \( R_i \subset R \), for all \( i=1,\ldots,k \)

Method:

begin

(1) \( R_1=R \); \( i=1 \); \( T={} \); note, \( T \) is a set of sets
(2) Find X (a candidate key), Y \not\in T, and Z (a set of secondary attributes) that satisfy \(X \rightarrow Y, Y \rightarrow Z\) (must be non-trivial), and \(Y \rightarrow \rightarrow X\). Note, Z represents the set of all the secondary attributes that are determined by Y; Add Y to T;

(3) if (i==1 or (YZ \not\subset R_j and YZ \neq R_j, for all j=2,\ldots,i)) then
   
   i = i + 1;
   
   \(R_1 = R_1 - Z\); \(R_i = YZ\);
   
   (note, \(F_1 = \prod_{R_1}(F), F_i = \prod_{R_i}(F)\));

(4) Repeat Steps (2) & (3) until there is no more transitive FD.

(5) for \(j = 1\) to \(i\) {if \(R_j\) is not in 3NF, call this algorithm recursively;}

end

Example 1 (Revised 4/16/2002):

department (DEPT D_OFFICE SECRETARY)

\(F = \{\text{SECRETARY} \rightarrow \text{DEPT}, \text{DEPT} \rightarrow \text{D_OFFICE}\}\)

DEPT \rightarrow \rightarrow \text{SECRETARY}

(1) \(R_1 = \{\text{DEPT}, \text{D_OFFICE}, \text{SECRETARY}\}\)

(2) \(X = \{\text{SECRETARY}\}, Y = \{\text{DEPT}\}, Z = \{\text{D_OFFICE}\}\)

(3) \(R_1 = \{\text{DEPT}, \text{SECRETARY}\}, R_2 = \{\text{DEPT}, \text{D_OFFICE}\}\)

(4) No more transitive FD

We can show that this decomposition is lossless and FD preserving using the algorithms in this lecture note. Try it.

Example 2 (Added 4/16/2002):

\(r(A,B,C,D,E,F)\)

\(F = \{AB \rightarrow CD, CD \rightarrow E, E \rightarrow F\}\)

Candidate Key: \(\{A,B\}\) // \(r\) has only one candidate key

(1) \(R_1 = \{A,B,C,D,E,F\}\)

(2) \(X = \{A,B\}, Y = \{C,D\}, Z = \{E,F\}\)

(3) \(R_1 = \{A,B,C,D,E,F\} - \{E,F\} = \{A,B,C,D\}, \)

\( R_2 = \{C,D,E,F\} \)
(2) \( X = \{A, B\}, Y = \{E\}, Z = \{F\} \)
(3) \( \{E, F\} \subseteq R2 \)
(4) No more transitive FD
(5) \( R1 = \{A, B, C, D\} \) is in 3NF but \( R2^* = \{C, D, E, F\} \) is not in 3NF
   (1) \( R1 = \{C, D, E, F\} \)
   (2) \( X = \{C, D\}, Y = \{E\}, Z = \{F\} \)
   (3) \( R1 = \{C, D, E, F\} - \{F\} = \{C, D, E\} \)
       \( R2 = \{E, F\} \)
   (4) No more transitive FD
   (5) No unnecessary schema
   (6) Both \( R1 = \{C, D, E\} \) and \( R2 = \{E, F\} \) are in 3NF
       output: \( \{C, D, E\}, \{E, F\} \)
R2* is replaced by the above output.
final output: \( \{A, B, C, D\}, \{C, D, E\}, \{E, F\} \)

Remaining Problems: Does not eliminate transitive FDs of primary attributes on candidate keys, which can also cause anomalies.

Example: address (STREET CITY ZIP),
\( F = \{\text{STREET CITY \rightarrow ZIP}, \text{ZIP \rightarrow CITY}\} \)
Candidate Keys: \( \{\text{STREET CITY}\}, \{\text{ZIP STREET}\} \)
Since "CITY" is not a secondary attribute, the relation is in 3NF. However, there are anomalies:
(1) Repetition of Information: CITY is unnecessary repeated for every STREET with the same ZIP.
(2) Update Anomaly: If CITY changes its name, then many tuples must be changed.
(3) Insertion Anomaly: (CITY ZIP) cannot be represented without a STREET.
(4) Deletion Anomaly: If we delete every STREET of a ZIP, then the <CITY ZIP> pair will be deleted.
• **Boyce-Codd Normal Form (BCNF)**

Definition: Let \( r(R), X, Y \subseteq R \), and \( X \cap Y = \emptyset \). The relation \( r(R) \) is in BCNF iff it is in 1NF and \( \forall F(D) X \rightarrow Y \), \( X \) is a candidate key of \( r(R) \).

Relational DB is in BCNF if all relations are in BCNF

**Lemma 1:** If \( r(R) \) is in BCNF, then it is in 3NF.

**Proof:** Let \( r(R) \) be in BCNF, and let \( \exists F(D) X \rightarrow Y, Y \rightarrow Z, Y \nrightarrow X \), and \( X \) is a candidate key, and \( Y \rightarrow Z \) is non-trivial. Since \( Y \rightarrow Z \) and \( r(R) \) is in BCNF, \( Y \) is a candidate key of \( r(R) \). Subsequently, \( Y \rightarrow X \). Contradiction (i.e., no transitive dependency)! Therefore, \( r(R) \) is in 3NF.

**Lemma 2:** If \( r(R) \) is in 3NF, then it need not be in BCNF.

**Example:** addresses (C S Z)

\[ F = \{ CS \rightarrow Z, Z \rightarrow C \} \]

The relation is in 3NF, but not in BCNF since \( Z \rightarrow C \) and \( Z \) is not a candidate key!

**BCNF Normalization:** If \( r(R) \) is NOT in BCNF, then \( \exists \) decomposition \( d \) of \( r(R) \) into a set of subrelations which are in BCNF (the decomposition is lossless but may induce loss of FDs) ➔ Use the Algorithm in P. 5 ➔ If decomposing a relation \( r(R) \), which is in 3NF, into a set of relations in BCNF causes loss of FDs, do not decompose the relation \( r(R) \).
Algorithm: Lossless BCNF Normalization Algorithm
(Revised 4/16/2002)

Input: r(R) not in BCNF and a set of FDs F on R.
Output: d={R1, R2, ..., Rk}, where each Ri⊂R, for all i=1,..,k.
Method:
begin
(1) R1=R; i=1; T={}; //note, T is a set of sets
(2) Find X→Y such that X∉T is not a candidate key and Y = X+−X≠∅;
   Add X to T;
(3) if (i == 1 or (XY ⊄ Rj and XY≠Rj, for all j=2,..,i)) then
   i=i+1;
   R1=R1-Y; Ri=XY;
   (note, F1=ΠR1(F), Fi=ΠRi(F));
(4) Repeat Steps (2)&(3) until there is no more X→Y such that X∉T is
    not a candidate key and Y = X+−X≠∅;
(5) for j=1 to i { if Rj is not in BCNF, call this algorithm recursively; }
end

Example (Revised 4/15/2002):
r (A,B,C,D), F={AB→CD, BD→AC, A→D, D→A}
Candidate Keys: {A,B}, {B,D}
(1) R1={A,B,C,D}
(2) A→D
(3) R1={A,B,C,D}-{D}={A,B,C}
   R2={A,D}
(2) D→A
(3){D,A} is equal to R2
(4) No more such FD
(5) Both R1&R2 are in BCNF
output: {A,B,C}, {A,D}
Remaining Problems

Example: storage (ITEM, COLOR, SEND_TO)
Key={ITEM, COLOR, SEND_TO}, F={} 

Since there is no non-trivial FD, the relation is in BCNF.

Anomalies:
(1) insertion anomaly - ITEM COLOR cannot be represented without an SEND_TO;
(2) deletion anomaly - ITEM COLOR deleted if all SEND_TO are deleted.

The 3NF decomposition is lossless and FD preserving.

Normalization Step: Partition your tables using the 3NF decomposition algorithms until all the subschemas are in 3NF. Then partition each of the subschemas using the BCNF algorithm. If decomposition into BCNF of a relation r(R), which is in 3NF, causes loss of FDs, do not decompose the relation r(R), or keep them in BCNF and say “at least we tried hard”.

Query Design: Write database queries in a certain query language (e.g., SQL)

Optimization Step: Build appropriate indices.
APPENDIX

Algorithm: Loss of Information Testing

Input: R, F, d //Note, not F+ but F is used
Output "Yes" if no loss of information; "No" otherwise

Method:

(1) construct a matrix M with k rows and n columns (row i corresponds to the rel. schema R_i; column j corresponds to attribute A_j).

(2) set M[i, j] to j if A_j ∈ R_i or (i+1)*j if A_j ∉ R_i

(3) Consider all FDs X --> Y in F until no more changes in M occur after all FDs in F have been examined.

begin

(3.1) For each FD X --> Y, find rows in M that are same for all attributes in X; If found, equate all fields of these rows for attributes in Y (i.e., for each attribute A_j in Y, if there is a found row whose j^{th} value is j, then set the j^{th} value of every found row to j. Otherwise, set the j^{th} value of every found row to that of the first found row)

(3.2) If there is a row <1, 2, 3, ..., n> in M, return "Yes" (i.e., no loss of information)

end

return "No" (i.e., loss of information)
Example#1 of 2
Input: R=SAIP, F={S-->A, SI-->P}, d={SA, SIP}

After Steps (1) and (2)
\[
M = \begin{array}{cccc}
1 & 2 & 6 & 8 \\
1 & 6 & 3 & 4 \\
\end{array}
\]

After Step (3.1) with S --> A:
\[
M = \begin{array}{cccc}
1 & 2 & 6 & 8 \\
1 & 2 & 3 & 4 \\
\end{array} -----> no loss!
\]

Example#2 of 2
Input: R=ABCDE,
\[
F=\{A-->C, B-->C, C-->D, DE-->C, CE-->A\},
d=\{AD, AB, BE, CDE, AE\}
\]

After Steps (1) and (2):
\[
M = \begin{array}{cccccc}
1 & 4 & 6 & 4 & 10 \\
1 & 2 & 9 & 12 & 15 \\
4 & 2 & 12 & 16 & 5 \\
5 & 10 & 3 & 4 & 5 \\
1 & 12 & 18 & 24 & 5 \\
\end{array}
\]
After Step (3.1) with A-->C:

\[
M = \begin{bmatrix}
1 & 4 & 6 & 4 & 10 \\
1 & 2 & 6 & 12 & 15 \\
4 & 2 & 12 & 16 & 5 \\
5 & 10 & 3 & 4 & 5 \\
1 & 12 & 6 & 24 & 5
\end{bmatrix}
\]

After Step (3.1) with B-->C:

\[
M = \begin{bmatrix}
1 & 4 & 6 & 4 & 10 \\
1 & 2 & 6 & 12 & 15 \\
4 & 2 & 6 & 16 & 5 \\
5 & 10 & 3 & 4 & 5 \\
1 & 12 & 6 & 24 & 5
\end{bmatrix}
\]

After Step (3.1) with C-->D:

\[
M = \begin{bmatrix}
1 & 4 & 6 & 4 & 10 \\
1 & 2 & 6 & 4 & 15 \\
4 & 2 & 6 & 4 & 5 \\
5 & 10 & 3 & 4 & 5 \\
1 & 12 & 6 & 4 & 5
\end{bmatrix}
\]

After Step (3.1) with DE-->C:

\[
M = \begin{bmatrix}
1 & 4 & 6 & 4 & 10 \\
1 & 2 & 6 & 4 & 15 \\
4 & 2 & 3 & 4 & 5 \\
5 & 10 & 3 & 4 & 5 \\
1 & 12 & 3 & 4 & 5
\end{bmatrix}
\]
After Step (3.1) with CE--->A:

\[
\begin{array}{ccccc}
1 & 4 & 6 & 4 & 10 \\
1 & 2 & 6 & 4 & 15 \\
1 & 2 & 3 & 4 & 5 \\
1 & 10 & 3 & 4 & 5 \\
1 & 12 & 3 & 4 & 5 \\
\end{array}
\]

In Step (3.2):

\[
\begin{array}{ccccc}
1 & 4 & 6 & 4 & 10 \\
1 & 2 & 6 & 4 & 15 \\
1 & 2 & 3 & 4 & 5 \\
1 & 10 & 3 & 4 & 5 \\
1 & 12 & 3 & 4 & 5 \\
\end{array}
\]

--> no loss of Info.!