Copyright 2000

David Tanneh

(ORMS)

Functional Dependencies and Normal DB Design Theory for Relational DB
Deletion anomaly: We may lose the City for a Zip

Update anomaly: Possible to update inconsistency

Redundancy: “New York” is stored many times

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sid</td>
<td>Boston</td>
<td>02155</td>
</tr>
<tr>
<td>Jim</td>
<td>Boston</td>
<td>02111</td>
</tr>
<tr>
<td>Mac</td>
<td>New York</td>
<td>10024</td>
</tr>
<tr>
<td>Joe</td>
<td>New York</td>
<td>10024</td>
</tr>
</tbody>
</table>

Consider the table \( R(\text{Person, Zip, City}) \)

A RELATIONAL DESIGN WITH FLAWS
A compact notation is

\[ (P(Z) \cdot Z)^y = (\mathbb{C}, Z^y) \]

\[ (P(Z')^y = (Z, P)^y) \]

A decomposition of \( R(Person, Zip, City) \):

\[ (P)^y = \forall^2 R \]

\[ V = \forall^u \cdot V \cup V^1 \] and \( V^1 \subseteq V^1 \)

\[ \forall \text{ Each } \forall^1 \subseteq V^1 \]

where \( \forall^u \text{ relations } R^1(\forall^1) \cdot \forall^u \text{, } \forall \text{ decomposition of } R(\forall) \)

To remedy, we look for a good decomposition of \( R(\forall) \)

Suppose a relation \( R(\forall) \) over the attributes \( \forall \) sufferers

**Solution:** DECOMPOSE THE RELATION
WHAT ARE THE PROS AND CONS OF:
The foundation is the theory of functional dependencies:

- With 3NF, we can get (c) and (e)
- With BCNF, we can get (a) and (b)

"normal forms"

2. Algorithms for decomposing relations into various

- Preservation of constraints (c)
- Expressiveness (b)
- Irredundancy (a)

I. Define important properties for a decomposition:

DESIGN THEORY

GOALS AND RESULTS OF RELATIONAL
\[ \text{Zip City} \]

\[ \text{Person Zip} \]

Some dependencies for \( R(\text{Person, Zip, City}) \):

\[ \lambda \in X \]

- If two tuples of \( R \) are equal on \( X \), then they are equal on \( \lambda \).

\( \text{Dependency} X \leftarrow \lambda \) asserts an integrity constraint:

\[ \forall \lambda \subseteq \lambda \in X \text{ and } A \subseteq A \]

Let \( R \) be a relation with attributes \( A \), and suppose many-one relationships functional dependencies generalize the concept of functional dependencies (FDs):
then r.CITY = s.CITY
if r.PERSON = s.PERSON and r.ZIP = s.ZIP

- For all rows r, s ∈ T:
  - ⊤ satisfies (PERSON, ZIP → CITY) means:
    - if r.ZIP = s.ZIP then r.CITY = s.CITY

- For all rows r, s ∈ T:
  - ⊤ satisfies (ZIP → CITY) means:
    - ⊤ satisfies (ZIP, CITY) means:
      - 0 < ZIP ≤ CITY

Consider a table ⊤(PERSON, ZIP, CITY)

Constraints on Tables
\[ \text{EID} \leftarrow \text{Name}, \text{Salary} \]

In other words, valid tables must satisfy:

If \( r.\text{Name} = s.\text{Name} \) and \( r.\text{Salary} = s.\text{Salary} \)

For rows \( r, s \in \text{Employee} \):

Consider \( \text{Employee}(\text{EID}, \text{Name}, \text{Salary}) \)

Keys are Dependency Constraints
Which dependencies should be satisfied by valid tables?

\[
P \rightarrow C \rightarrow Z
\]

\[
Z \rightarrow Z \rightarrow C
\]

\[
Z \rightarrow C \rightarrow Z
\]

\[
Z \rightarrow P \rightarrow \cdot
\]

---

**QUIZ:** WHICH PDS ARE SATISFIED?
Describes employees who use tools on projects and work
having a certain skill and working in a certain location
various locations and have various skills. All employees
meet in a specified room once a week.

\[
\begin{align*}
R_o &= \text{Room for meetings} \\
S_k &= \text{Skill} \\
L_o &= \text{Location} \\
P_r &= \text{Project} \\
H_o &= \text{Hours per week} \\
E_m &= \text{Employee} \\
T_o &= \text{Tool}
\end{align*}
\]

Attributes:

THE EMPLOYEES-SKILLS LOCATION RELATION


RO: Each room is in one location only

SKLO: All employees working in a location having a certain skill always meet in the same room

EmTO: Ho: An employee uses each tool for the same number of hours each week

To: Each tool can be used on a single project

TO: Each employee uses a single tool and works on a single project

The relation satisfies the following FDs:

**THE FDs**
\[(\forall \leftarrow B) \text{ does not imply } \{(B \leftarrow \forall)\} \]

\[(C \leftarrow \forall) \text{ implies } \{(C \leftarrow B) \cdot (B \leftarrow \forall)\} \]

Hence, \(b\) is an objective consequence of \(\forall\).

\(\forall\) also satisfies \(\forall\) \(\forall\) \(\forall\) \(\forall\) \(\forall\) \(\forall\) \(\forall\)

Every relation that satisfies all the dependencies implies \(\forall\) \(\forall\) \(\forall\) \(\forall\) \(\forall\) \(\forall\) \(\forall\)

\[(CZd \leftarrow d) \text{ implies } \{(C \leftarrow Z) \cdot (Z \leftarrow d)\}\]

Claim: IMPLIED DEPENDENCIES
Lesson: Many dependencies are implied but not stated.

\[
A \leftarrow B -
\]

- \(A \leftarrow C\) -

Then which dependencies must \(T\) necessarily satisfy?

- \(B \leftarrow C\) -

- \(B \leftarrow A\) -

Suppose we know that \(T(A, B, C)\) satisfies:

Implied Dependencies – Quiz 1
Relation, Given:

Find some implied dependencies for EMPLOYEE

IMPLIED DEPENDENCIES — QUIZ 2
\[ \forall x \in \text{all attributes determined by A} \]

\[ Z \forall x = +(Z \forall x) - \]

\[ Z \forall x = +(Z \forall x) \quad \forall x = +(\forall x) \quad \forall x \forall x = +(\forall x) - \]

\[ Z = +Z \quad \forall x = +\forall x \quad \forall x \forall x = +x - \quad \text{Given} \]

\[ (Z \leftarrow \lambda), (\lambda \leftarrow x) \quad \text{Hence,} 
\]

\[ A \subseteq A \quad \text{and} \quad A \leftarrow B \quad \text{is the largest set such that} \]

**The Closure A+ of a Set of Attributes**
\[ S' = \text{Res} \]
\[ \text{end loop} \]
\[ \text{else break} \]
\[ \{x\} \cap \text{Res} = \text{Res} \]
\[ \text{then} \]
\[ \text{we have } \forall s \in \text{Res} \text{ and } X \not\in s \]
\[ X \leftarrow \forall \text{ dependency } \]
\[ \text{loop} \]
\[ \text{Res} = S' \]

**Computing the Closure**
Example of Computing a Closure

Given the rules:

1. \( X \leftarrow Y \) by rule (1)

2. \( X Y \leftarrow \) by rule (2)

3. \( X Y Z \leftarrow \) by rule (3)

Answer = \( XYZ \) because no more can be added

Steps:

Compute: \( X + \)

\( Z \leftarrow X Y \)

Compute: \( X \)
Every superkey has a subset which is a key

\[ \text{EID, Name, Salary} \]

\[ \text{EID, Name, Salary} \]

\[ \text{EID, Name, Salary} \]

Employee(EID, Name, Salary)

That is a superkey means: \( S + = \text{A} \)

Let A be all attributes, and \( S \subseteq \text{A} \)

**Super-keys**
\[(Z \leftarrow X)'(X \leftarrow \lambda)'(\lambda \leftarrow X)\]

3. \[(Z \leftarrow \lambda X)'(\lambda \leftarrow X)\]

2. \[(Z \leftarrow \lambda)'(\lambda \leftarrow X)\]

1. \[(Z \leftarrow \lambda)'(\lambda \leftarrow X)\]

What are the keys for:

- \{EID\} is a key

- \{EID\, Salary\} is a superkey, but not a key

- For \{EID\, Name, Salary\}:

  - A key is a minimal superkey

**KEYS — EXACT DEFINITION**
How can we reduce the search?

Answer: $2^r$

How many subsets $S$ are there?

$\forall$ every $S$ whether $S+ = A$

To find superkeys, we must in principle check for

This means finding all minimal superkeys

**Goal:** Find keys in relation
{Em, Sk} ⊆ key ⊆ {Em, Sk, To, Ro} • 
Hence, {Em, Sk} ⊆ key ⊆ {Em, Sk, To, Ro} • 

Every key must be disjoint from R • 
Every key must include L and N • 

Let N = attributes not appearing at all • 
Let R = attributes appearing only on right sides • 

Let L = attributes appearing only on left sides • 

(Sk → Ro) \ (Ro → L) • 
(Em → Top) \ (Top → Pr) • (Em → Ho) • 

NARROWING THE SEARCH FOR KEYS
Hence, keys are $\{E_m, SK, T_o\}$ and $\{E_m, SK, R_o\}$

$E_m, SK, R_o = + \{E_m, SK, T_o, Pr, Ho, Ro, To\}$

$E_m, SK, T_o = + \{E_m, SK, T_o, Pr, Ho, Ro, To\}$

$E_m, SK = + \{E_m, SK, T_o, Pr, Ho\}$

$(SK, T_o \leftarrow R_o, (Ro \leftarrow T_o, (Pr \leftarrow E_m) \leftarrow Ho, E_m))$

$(E_m, SK, T_o, Pr, Ho, Ro) \leftarrow \{\text{key} \subseteq \{E_m, SK, T_o, Ro\}$

Calculating the Keys
\[ R_1(\text{Person}) \times R_2(\text{Zip}) \]

Can \( R(\text{Person, Zip}) \) be represented as:

\[ R(\text{Person, Zip, City}) \]

Can \( R(\text{Person, Zip, City}) \) be represented as:

\[ R \]

\[ u^n \times \cdots \times R^n \]

\[ R = R_1 \times \cdots \times R^n \]

By natural join \( \times \)

Because \( R_1, \ldots, R^n \) share attributes, we connect them

We should be able to reconstruct \( R \) from \( R_1, \ldots, R^n \)

Suppose \( R \) is replaced by \( R_1, \ldots, R^n \)

**Reconstructing a Decomposed Relation**
representability

Not hard to show that lossless-join is equivalent to

\[(R)^u \forall \mu \land \cdots \land (R)^i \forall \mu = R - \]

conforming to \( R \) is the join of its projections:

\( R \) is lossless-join if every relation \( R \) has a \( \forall \land \cdots \land \forall \) such that every subset of \( A \)

\( \forall \land \cdots \land \forall \) be subsets of \( A \)

Let \( A \) be attributes with functional dependencies \( R \)

representable in the decomposition

Lossless-join property ensures that every relation is

Valid ("Lossless-Join") Decompositions
Whenver \( X \rightarrow \lambda \) then \( X \) must be a superkey.

\[\text{Definition of BCNF:}\]

Sam 10024 New York
Joe 10024 New York

\( \text{Example: (Person} \rightarrow \text{Zip, Zip} \rightarrow \text{City)} \]

may occur in the tables

\( X \rightarrow \lambda \) but \( X \) is not a superkey, then redundancy

\[\text{THE BCNF IRREDUNDANCY CONDITION}\]
condition

I.e., does any rule violate the BCNF irredundancy

\{ Keys are \{ Ze, Sk, Ro \} and \{ Em, Sk, Ro \} \}

( Sk → Ro, Ro → To, ( To → Ho, ( Em → Ho, ( Em → To, ( To → Pr, ( Pr → Ho, )

Quiz: Is EMPROPHOSKOLORO in BCNF?
A flaw with BCNF decomposition
\[ + X \subseteq S \quad \text{and} \quad X \in \mathcal{A} \quad \text{and} \quad \forall A, B \in \mathcal{H}, A \subseteq B \implies B \in \mathcal{F} \]

Computation of the projected dependencies:

- If \( X \) is implied iff \( X \leftarrow S \)

These are called the projections of \( H \) onto the \( A \):

\[ ^u A \quad \cdots \quad \forall A \]

Relations for \( A^1 \) for what are the implied constraints on the

\( \forall A \quad \cdots \quad \forall A \)

Consider integrity constraints \( H \) on relations

Suppose we decompose \( A \) into \( A^1 \) with attributes \( A \)

Consider the dependencies

\text{PROJECTING THE DEPENDENCIES}
needed for each insert or update

Without it, a join followed by check against 

constraints $H_1, \ldots, H_r$ on the separate tables

desirable, because it means $H$ can be enforced by the

then decomposition preserves dependencies

If all the dependencies $H_1 \cap \cdots \cap H_r$ together imply $H$

$^2$ Let $H$ be the projection of $H$ onto $\mathcal{A}^2$

decomposition $\mathcal{A}^1, \ldots, \mathcal{A}^r$

given attributes $\mathcal{A}$ with dependencies $H$, and

Preservation of Dependencies
3NF: A Modification of BCNF

- (Addr,City → Zip), (Zip → City),
- Keys = \{Addr,City\} and \{Addr,Zip\}
- (Zip → City) violates BCNF:
  - But decomposing via (Zip → City) went too far, and broke a dependency
- Solution: Define a weaker condition, 3NF, for which (Zip → City) is not a violation
- Definition of 3NF:
  - For all nontrivial (X → Y), X is a superkey or Y is a subkey
\{\text{Keys} \mid \text{Em, Sk, Lo} \} = \text{Em, Sk, Lo}

(\text{Sk} \rightarrow \text{Ro}, \text{Ro} \rightarrow \text{Lo})

(\text{Em} \rightarrow \text{Ho}, \text{Ho} \rightarrow \text{Em})

\text{Em, To, Pr, Ho, Sk, Lo}

\text{Person, Em, Ho, Sk, Lo}

(\text{Person} \rightarrow \text{Age}, \text{Age} \rightarrow \text{Weight})

(\text{Person} \rightarrow \text{City}, \text{City} \rightarrow \text{Zip})

(\text{City} \rightarrow \text{Zip}, \text{Zip} \rightarrow \text{Addr})

\text{City, Addr, Zip}

\text{Which schemas are in 3NF?}
What algorithm can find and delete the redundant constraint — Removing (1) and (4) doesn’t change the overall set is not minimal, because:

\( (C \leftarrow Z \leftarrow P) \)

\( (C \leftarrow Z) \)

\( (Z \leftarrow P) \)

\( (P \leftarrow C) \)

Consider the dependencies:

Redundant Functional Dependencies
The final value of $D$ has no redundancies.

1. By assigning $D = D', \ y \leftarrow X +', \ x$, remove

4. If $\ y \subseteq X +$, then $(X \leftarrow y)$ is redundant, so remove.

3. Let $X^+$ be the closure of $X$ under $D$.

2. Let $D'$ be the other rules in $D$.

1. Choose a rule $(X \leftarrow y)$ in $D$.

Repeat until none can be removed.

Let $D$ be a set of dependences.

REMOVING REDUNDANT DEPENDENCIES
– Can C be reached from B, without A?

    When considering whether A can be removed, the

    Hence, the attribute B is redundant in (AB – C)

    D and D' are equivalent

    \( (A \leftarrow C) \leftarrow (A \leftarrow B) \leftarrow (A \leftarrow B') \)

    \( \text{Let } D' = (A \leftarrow B) \leftarrow (A \leftarrow B') \leftarrow (A \leftarrow C) \)
∀ the rules of D are used for this closure

∀ belongs to X+

Then we can replace with (X \rightarrow Y) iff:

Suppose (\forall Y \rightarrow X) \land Y is in D, and A is a single attribute

Let D be a set of dependencies

THE LEFT SIDES

TESTING FOR REDUNDANT ATTRIBUTES ON
3NF DECOMPOSITION ALGORITHM

1. Find keys, and check if the relation is already in 3NF
2. Decompose the right hand sides of the dependencies
3. Remove redundant attributes on the left hand sides
4. Remove redundant functional dependencies
5. Combine dependencies with same left hand sides
6. Create a relation for each functional dependency
7. Remove relations contained in other relations
8. If no relation contains a key of the original relation, add a relation whose attributes form such a key
Hence, \( \{ E \} \subseteq \text{sk, E, s, k, T, o, l, o} \).

- Every key must be disjoint from \( R \).
- Every key must include \( L \) and \( N \).

\( \emptyset = \) all attributes not appearing at all,
\( R = \) attributes appearing only on right sides,
\( L = \) attributes appearing only on left sides,
\( E \subseteq \text{sk, e, s, k, t, o, l, o} \).

\((R \leftarrow R, O \leftarrow L)\)
\((S \leftarrow E, T \leftarrow O, P \leftarrow T)\)
\((E \leftarrow H, O \leftarrow H)\)

**Finding the Keys**
Hence, keys are $\{E_m', SK_{To}, Ho, Ro\} \Rightarrow$ $E_m', SK_{To} = + \{E_m', SK_{To}, Ho, Ro\} \Rightarrow$
$E_m', SK_{To} = + \{E_m', SK_{To}, Ho, Ro\} \Rightarrow$

$E_m', SK_{To} = + \{E_m', SK_{To}, Ho, Ro\} \Rightarrow$

$E_m', SK_{To} = + \{E_m', SK_{To}, Ho, Ro\} \Rightarrow$

$E_m', SK_{To} = + \{E_m', SK_{To}, Ho, Ro\} \Rightarrow$

$E_m', SK_{To} = + \{E_m', SK_{To}, Ho, Ro\} \Rightarrow$

$\{E_m', SK_{To}, Ho\} \subseteq \{E_m', SK_{To}, Ho\} \Rightarrow$

$\{E_m', SK_{To}, Ho\} \subseteq \{E_m', SK_{To}, Ho\} \Rightarrow$

Finding The Keys (Cont'd)
$R_0 \leftarrow I_0$

$S_k T_0 \leftarrow R_0$

$E_m T_0 \leftarrow H_0$

$T_0 \leftarrow P_r$

$E_m \leftarrow P_r$

$E_m \leftarrow T_0$

This gives us:

$\leftarrow X \leftarrow Y \leftarrow X \,” YZ becomes X \leftarrow Z \leftarrow X$ and $Y \leftarrow X$. Decompose the right hand sides of the functional dependencies, e.g., $X \leftarrow YZ becomes X \leftarrow Z \leftarrow X$. Decomposing Right Hand Sides
cannot be replaced

- Both $SK$ and $L_0$ are needed to get $Ho$, so the rule

  \[
  \text{Then check } (SKL_0 \leftarrow Ho) \text{ for redundancy on left:}
  \]

  \[
  (EM \leftarrow Ho) \quad \text{Hence, we can replace with } (EM \leftarrow Ho)
  \]

  \[
  \text{Since } EM^+ = EM \cup Ho, EM \text{ suffices for Ho}
  \]

  \[
  \text{Since } EM^+ = EM + Ho, EM \text{ is needed to obtain Ho}
  \]

- First check $(EML_0 \leftarrow Ho)$ for redundancy on left:

  \[
  \text{Remove redundant attributes from the left sides}
  \]

**REMOVE REDUNDANT ATTRIBUTES**
The Simplified Rules
- Redundant, since Pr is in $E^+m$.

$$E^+m = EM_{topHo}$$

- $E^+m = EM_{topHo}$

$(E^+m \leftarrow Pr) \cdot$ Compute $E^+m$ using I, 3, 4, 5, 6.

- Since $To$ not in $EM_{topHo}$, this rule is not redundant.

- $E^+m = EM_{topHo}$

2, 3, 4, 5, 6:

Is $(E^+m \leftarrow To)$ redundant? Compute $E^+m$ using rules

$(SkTo \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro, (To \leftarrow Ro$,

Remove the Redundant Rules
5. \( R_0 \leftarrow I_0 \)
4. \( S_1 \leftarrow R_0 \)
3. \( E_m \leftarrow H_0 \)
2. \( T_0 \leftarrow P_1 \)
1. \( E_m \leftarrow T_0 \)

Resulting rules, after renumbering:

- Similarly, rules 4, 5 and 6 are not redundant
- Hence, not redundant

\(-\)
\(-\)
\( T_0 = T_0 \)

(\( T_0 \leftarrow P_1 \). Compute \( T_0 \) using 1, 4, 5, 6 (but not 2))

- Hence, remove rule 2
COMBINING FUNCTIONAL DEPENDENCIES

• Combine all dependencies with the same left side

• This gives:
  1. Em → ToHo
  2. To → Pr
  3. SkLo → Ro
  4. Ro → Lo
Creating Relations

This gives:

Create a relation for each dependency
3. Skills

2. Top

1. EmToHo

This gives:

• Remove relations contained in other relations

REMOVE REDUNDANT RELATIONS
– EmToHo, Top, StLoc, EmToHo, EmToHo, Top, StLoc, EmToHo

- Two choices for the final decomposition into 3NF:

  - Hence, one of these has to be added

None of the relations contains EmToHo or EmToHo

then add a relation whose attributes form such a Key

If no relation contains a Key of the original relation,

ASSURING STORAGE OF A GLOBAL KEY.
But $AX, BY, AB$ is in 3NF

Hence, the decomposition $AX, BY$ is not in 3NF

The global key is $AB$

Consider $R(A, B, X, Y)$ with $(A \leftarrow X)$ and $(B \leftarrow Y)$
UNIT

TOPICS COVERED IN THIS (AND PREVIOUS)

• 3NF Decomposition Algorithm
• Normal Forms: BCNF and 3NF
• Dependencies
  – Non-redundancy, Lossless-Join, Preservation of Desirable Properties of a Decomposition
  – Closures, Superkeys, Keys, Implied Dependencies
• Functional Dependencies

49