# Database Systems – Design Theory

Hasan M. Jamil

Department of Computer Science
University of Idaho
USA

## Design Theory of Relational Databases

Overview:

- Problems with "*bad*" designs.
- Motivation for splitting schemes
  - systematic splitting.
  - lossy join and lossless join decomposition.
- Integrity constraints as guidelines for doing "*good*" decompositions.
- Functional dependencies (FDs) – superkeys and keys.
- Theory of FDs
  - axiomatization.
  - closure.
- Normal forms.

Problems with "*bad*" designs

- storage redundancy.
- potential for inconsistency.
- insertion anomaly – certain meaningful insertions are not possible.
- deletion anomaly – "*unexpected*" loss of information with certain deletions.

Example 1:

empl_proj(name, id, bdate, addr, gender, sal, dno, pno, pname, plocation, hours)

Problems:

- employee information repeated once for each project (s)he works on.
- project information repeated for every employee working on it.

$\Rightarrow$ enormous redundancy.

$\Rightarrow$ concomitant potential for inconsistency.

For example: If an employee moves $\rightarrow$ changes and its consequences.

Example 2:

emp_info(name, id, bdate, gender, sal, dno, depen_name, dgender, dbdate, relationship}

– can't insert new employees unless they have dependents!

– solution! Use NULL values $\rightarrow$ complicates query processing.

Example 3:

Suppose materials are supplied by various suppliers to our company.

suppliers(sname, saddr, dno, item, qty, price)

– when some supplier ceases to supply anything to the company (momentarily), will loose his address – deletion anomaly.

*Solutions to problems*:

Example (1) Split empl_proj into

– emp(name, id, ..., dno).
– project(pno, pname, plocation, dno).
– e-p(id, pno, hours).

Example (2) Split emp-info into

– emp(name, id, ..., dno).
– dependents(id, ..., relationship).

Example (3) Split suppliers into

– supplierinfo(sname, saddr).
– supplyinfo(sname, dno, item, qty, price).

The problems go away upon splitting.

Moral: Split "*big*" relation schemes.
But how? Any splitting is good?

suppliers = 

| $sname$ | $saddr$ | $dno$ | $item$ | $qty$ | $price$ |
|---|---|---|---|---|---|
| $John$ | $a_1$ | $d_1$ | $i_1$ | 100 | $p_1$ |
| $John$ | $a_1$ | $d_2$ | $i_1$ | 200 | $p_1$ |
| $John$ | $a_1$ | $d_2$ | $i_2$ | 100 | $p_2$ |
| $Mary$ | $a_2$ | $d_1$ | $i_2$ | 150 | $p_3$ |
| $Mary$ | $a_2$ | $d_2$ | $i_3$ | 300 | $p_4$ |

Consider splitting into

$R_1$(sname, addr, dno), and
$R_2$(dno, item, qty, price).

The decomposed relations will not represent the same info as the original relation – information loss.

Compare $r$ with $\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$.

Always $r \subseteq \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$. Why?

However, $\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$ contains some extra tuples.

– Tuples which are not true wrt $r$.

For example, $< John, a_1, d_1, i_2, 150, p_3 >$ (and many others) is such an extra tuple in $\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$.

– lossy join decomposition.

However,

$R_3 =$ supplierinfo(sname, saddr), and
$R_4 =$ supplyinfo(sname, dno, item, qty, price).

is a lossless join decomposition.

Intuition: Associated with each supplier is a unique saddr.

In general, a decomposition of a relation scheme is:

$$R \rightarrow \begin{cases} R_1 \\ \vdots \\ R_m \end{cases}$$

$R_i \subseteq R$, and $\bigcup_{i=1}^{m} R_i = R$.

For a decomposition as above and for any instance $r(R)$,

always – $r \subseteq \Pi_{R_1}(r) \bowtie \ldots \bowtie \Pi_{R_m}(r)$.

When $r \not\subseteq$ "$RHS$" $\rightarrow$ decomposition is lossy join.

When $r =$ "$RHS$", $\forall$ instance $r$, decomposition is lossless join.

Note: If there are no constraints, all decompositions are lossy join!

For each "sname", $\exists$ a unique "saddr" – an example of a constraint.

For all instances satisfying this constraint, the decomposition

$$suppliers \rightarrow \begin{cases} R_3 & = & supplierinfo(sname,\ saddr) \\ R_4 & = & supplyinfo(sname,\ dno, \\ & & \qquad item,\ qty,\ price) \end{cases}$$

(on slide 7) is a lossless join.

$\Rightarrow$ Integrity constraints. – Functional Dependencies (FDs). $\rightarrow$ a generalization of the notion of keys.

<u>Example 4</u>:

$sname \rightarrow saddr$

Explanation (the connections between FDs and keys): In general, $R$ – relation scheme.

$X, Y \subseteq R$.
$X \rightarrow Y$.

Example 5:

In emp(name, id, bdate, addr, gender, sal, dno)

$id \to name.$
$id \to bdate.$
$\vdots$
$\{name, addr\} \to id.$
$\{name, addr\} \to bdate.$
$\vdots$

Let $X, Y \subseteq R$. $r(R)$ satisfies $X \to Y$ if
$\forall t_i, t_j \in r, \quad t_i[X] = t_j[X] \Rightarrow t_i[Y] = t_j[Y]$.

Example 6:

$$r = \left\{ \begin{array}{ccc} A & B & C \\ \hline a_1 & b_1 & c_1 \\ a_1 & b_1 & c_2 \\ a_2 & b_2 & c_3 \\ a_2 & b_3 & c_3 \end{array} \right.$$

$r$ satisfies:
$B \to A$ and
$C \to A$.

But not:
$A \to B$ and
$A \to C$

Why?

– Specifier of a database application supplies integrity constraints in informal language.

– Designer formalizes (some of) them as FDs.

– Constraints are meant to be enforced all the time.

– FDs can be used to determine candidate keys, and the process can be automated for large applications.

– Need to study logical interaction of FDs and to reason about them.

$\Rightarrow$ Armstrong's Axiom Systems for FDs.

## Logical implications of FDs

Example 7:

Let $r(ABC)$ be a relation, and
$F = \{A \rightarrow B,\ B \rightarrow C\}$.

Claim: Every $r$ satisfying $F$ has to satisfy
$A \rightarrow C$.

– informal argument (?).

In this case we write,
$F \models A \rightarrow C$. (i.e., $F$ logically implies $A \rightarrow C$).

– Given $F$, we need to know FDs implied by $F$ in order to determine keys (and for other purposes too).

– Need a mechanism for determining these logical implications.

## Armstrong's Axiom System for FDs

Let $U$ = set of all attributes.
Axioms : trivial FDs.
Inference rules : what are they?

- Axioms:
  - Reflexivity: If $X, Y \subseteq U$ and $Y \subseteq X$, then $X \rightarrow Y$ always holds.
- Inference Rules:
  - Augmentation: If $X \rightarrow Y$ and $Z \subseteq W \subseteq U$, then $XW \rightarrow YZ$.
  - Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.

This axiom system is sound (meaning ?) and complete (meaning ?).

Example 8:

Let $U = \{A_1, A_2, A_3, A_4, A_5\}$, and
$F = \{A_1 \rightarrow A_2,\ A_2 \rightarrow A_3,\ A_2A_3 \rightarrow A_4,$
$\qquad A_2A_3A_4 \rightarrow A_5\}$.

Prove: $F \models A_1 \rightarrow A_5$.

Proof:

| | | |
|---|---|---|
| (1) | $A_1 \rightarrow \underline{A_2}$ | given |
| (2) | $\underline{A_2} \rightarrow A_3$ | given |
| (3) | $A_1 \rightarrow A_3$ | transitivity: 1, 2 |
| (4) | $A_1 \rightarrow \underline{A_1A_2}$ | augmentation: 1 |
| (5) | $\underline{A_1A_2} \rightarrow A_2A_3$ | augmentation: 3 |
| (6) | $A_1 \rightarrow \underline{A_2A_3}$ | transitivity: 4, 5 |
| (7) | $A_2A_3 \rightarrow A_4$ | given |
| (8) | $\underline{A_2A_3} \rightarrow A_2A_3A_4$ | augmentation: 7 |
| (9) | $A_1 \rightarrow \underline{A_2A_3A_4}$ | transitivity: 6, 8 |
| (10) | $\underline{A_2A_3A_4} \rightarrow A_5$ | given |
| (11) | $A_1 \rightarrow A_5$ | transitivity: 9, 10 |

Example 9:

For $F$ and $U$ on slide 14,
consider the FD $A_2A_3A_4A_5 \to A_1$.

Claim: $F \not\models A_2A_3A_4A_5 \to A_1$.

Proof: By counterexample.

$$r = \left\{ \begin{array}{lllll} A_1 & A_2 & A_3 & A_4 & A_5 \\ \hline 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array} \right.$$

$r$ satisfies every FD in $F$, but it violates

$A_2A_3A_4A_5 \to A_1$.

## Additional Inference Rules for FDs

- Why need them?
  - convenience in making inferences.
- Additional inference rules
  - derivable from Armstrong inference system – implications.

Let $X, Y, Z \subseteq U$.

Union Rule: If $X \to Y$ and $X \to Z$ hold, then $X \to YZ$ holds. Recall $YZ = Y \cup Z$.

Decomposition Rule: If $X \to YZ$ holds, then $X \to Y$ and $X \to Z$ hold. (Dual of union rule)

Pseudo-transitivity Rule: If $X \to Y$ and $YZ \to W$ hold, then $XZ \to W$ holds.

– Explain and derive.

Example 10: Revisit of Example 8.

Let $U = \{A_1, A_2, A_3, A_4, A_5\}$, and
$F = \{A_1 \to A_2,\ A_2 \to A_3,\ A_2A_3 \to A_4,$
$A_2A_3A_4 \to A_5\}$.

Prove: $F \models A_1 \to A_5$.

Proof:

| | | |
|---|---|---|
| (1) | $A_1 \to \underline{A_2}$ | given |
| (2) | $\underline{A_2} \to A_3$ | given |
| (3) | $A_1 \to A_3$ | transitivity: 1, 2 |
| (4) | $A_1 \to \underline{A_2A_3}$ | union: 1, 3 |
| (5) | $\underline{A_2A_3} \to A_4$ | given |
| (6) | $A_1 \to A_4$ | transitivity: 4, 5 |
| (7) | $A_1 \to \underline{A_2A_3A_4}$ | union: 4, 6 |
| (8) | $\underline{A_2A_3A_4} \to A_5$ | given |
| (9) | $A_1 \to A_5$ | transitivity: 7, 8 |

– much simpler than using *only* Armstrong's rules.

– explanation.

## Closure

$F$ – a set of FDs.

$F^* = \{X \to Y \mid F \models X \to Y\}$.

– the logical (or consequence) closure of $F$
– explanation.

**More importantly**:

$F$ – a set of FDs.

$X \subseteq U$ – a set of attributes.

The *closure* of $X$ wrt $F$ is $X_F^+ = \{A \in U \mid F \models X \to A\}$

Recall: $A$ – denotes individual attributes

Example 11:

Let $U = \{A_1, A_2, A_3, A_4, A_5\}$, $X = \{A_1\}$, and
$F = \{A_1 \rightarrow A_2,\ A_2 \rightarrow A_3,\ A_2A_3 \rightarrow A_4,$
$\quad A_2A_3A_4 \rightarrow A_5\}$.

Then,

$$X_F^+ = (A_1)_F^+ = \{A_1, A_2, A_3, A_4, A_5\} = U$$

$\Rightarrow A_1$ is a superkey of $U$ (wrt $F$).
In fact, will see that $A_1$ is a key.

Note: *When $F$ is understood, denote closure by $X^+$.*

Notice that **closure** $\rightarrow$ **superkeys** $\rightarrow$ **keys**

- closure – related to keys.
- how to compute it?
  - inference system (e.g., Armstrong's) for FDs $\rightarrow$ sound and complete.
  - useful in computing closure.

**Algorithm Closure(X, F):**
Input: set of FDs $F$, and set of attributes $X$;
Output: $X^+$;
begin
  (1) $closure := X$;
  (2) while there is an FD: $W \rightarrow Z \in F$
      such that $W \subseteq closure$ and
      $Z \not\subseteq closure$ do
        $closure := closure \cup Z$
  (3) return $closure$;
end

Example 12:

$$F = \{A_1 \to A_2,\ A_2 \to A_3,\ A_2A_3 \to A_4,$$
$$A_2A_3A_4 \to A_5\}.$$
$X = \{A_1\}$.

Compute $X^+$: $A_1 \overset{1}{\Rightarrow} A_1A_2 \overset{2}{\Rightarrow} A_1A_2A_3$
$\overset{3}{\Rightarrow} A_1A_2A_3A_4 \overset{4}{\Rightarrow} A_1A_2A_3A_4A_5$

Remarks:
(1) $A \in X^+$ iff $F \models X \to A$.
(2) $X$ is a superkey of $U$ wrt $F$ iff $X^+ = U$.

– We have a (yet another) very quick proof that $F \models A_1 \to A_5$.

(3) Thus a key of $U$ wrt FDs $F$ is any minimal $X \subseteq U$ such that $X^+ = U$.

## Determining all keys for a relation scheme wrt FDs

Example 13:

Let $B$ = *Broker*, $O$ = *Office*, $I$ = *Investor*,
$S$ = *Stock*, $Q$ = *Quantity*, and
$D$ = *Dividend paid by a stock*.

FDs: $F = \{S \to D,\ I \to B,\ IS \to Q,\ B \to Q\}$.

Find all keys of $U = BOISQD$. Justify your answer.

Idea: Start with the biggest superkey of $U$ – which is $U$ itself.

– Throw away an attribute $A$ from current-set if (current-set $-$ $\{A\}$)$^+$ contains $A$.

– repeat until no change.

$$
\begin{array}{cl}
BOISQD & \\
\downarrow & \text{throw out } Q \text{ since } Q \in (BOISD)^+ \\
BOISD & \\
\downarrow & \text{throw out } D \text{ since } D \in (BOIS)^+ \\
BOIS & \\
\downarrow & \text{throw out } B \text{ since } B \in (OIS)^+ \\
OIS & \text{– can't discard any more attributes}
\end{array}
$$

What next:
(1) Show $OIS$ is a key.
(2) Show it is the only key.

Knowledge about keys –
essential for "normal forms"

(a) $OIS$ is a superkey:

$(OIS)^+$:
$OIS \overset{1}{\Rightarrow} OISD \overset{2}{\Rightarrow} OISDB \overset{3}{\Rightarrow} OISDBQ = U.$

(b) $OIS$ is a minimal superkey:

Sufficient to consider $X - A$ for each $A \in X$ – why?

$(OI)^+$: $OI \Rightarrow OIB \Rightarrow OIBQ \neq U$.
$(OS)^+$: $OS \Rightarrow OSD \neq U$.
$(IS)^+$: $IS \Rightarrow ISD \Rightarrow ISDB \Rightarrow ISDBQ \neq U$.

None of the above are superkeys

$\Rightarrow OIS$ is a minimal superkey, and hence a key.

## Covers of sets of FDs

- sets of FDs for an application come from a formalization of end user's informal specifications of the application.
- may contain some redundant information – need not be compact.
- FDs used in
  - decomposition of relation schemes $\rightarrow$ *normalization*.
  - determination of keys.
  - integrity maintenance.
- would be desirable to turn given FDs into equivalent compact form, so above activities can be done more efficiently.

Example 14:

$$F = \{A \to B,\ AB \to C,\ AB \to B,\ AC \to AD\}.$$

$F$ is logically equivalent to

$$G = \{A \to B, A \to C, A \to D\}$$

– informal reasoning.
– $G$ is more compact than $F$.

*Definition*: Let $F$, $G$ – sets of FDs. $F$ is logically equivalent to $G$, i.e., $F \equiv G$, if $F^* = G^*$. That is, for every FD $X \to Y \in G$, $F \models X \to Y$ and for every FD $W \to Z \in F$, $G \models W \to Z$.

– equivalence can be tested using (attribute set) closure.

– if $F \equiv G$, they are called *covers* of each other.

**Using closure to test if $F$ and $G$ are covers of each other**: Revisit previous example.

1. $F \models X \rightarrow Y$, for each $X \rightarrow Y \in G$.

   $(A)^+_F$ contains B.
   $(A)^+_F$ contains C.
   $(A)^+_F$ contains D.

2. $G \models W \rightarrow Z$, for each $W \rightarrow Z \in F$.

   $(A)^+_G$ contains B.
   $(AB)^+_G$ contains C.
   $(AB)^+_G$ contains B (trivially).
   $(AC)^+_G$ contains D.

$\Rightarrow F \equiv G$.

*What type of redundancies occur in FD sets (and can be eliminated)*:

- trivial FDs, like $AB \to A$ can be eliminated altogether.
- if the given set $F$ has $ABCD \to E$ and $F \models AB \to E$, (say) then we can replace $ABCD \to E$ by $AB \to E$. This is called *left-redundancy*.
- in general, it is advantageous to have 1 (one) attribute on the RHS of FDs.

  $\to$ an application of the decomposition rule will do the job.

## Review of Canonical Covers of FDs

Recall: A set of FDs $F$ is *canonical* provided

1. each FD in $F$ has a single attribute (only) on the RHS.
2. $F$ is non-redundant, i.e., $\forall$ FD $X \rightarrow A \in F$, $F - \{X \rightarrow A\} \not\models X \rightarrow A$, and
3. $F$ is *left-reduced*, i.e., $\forall$ FD $X \rightarrow A \in F$, $F \not\models Y \rightarrow A$, for any $Y \not\subseteq X$.
   – explanation.

Note: Eventhough a given set of FDs $F$ may not be canonical, it always has a cover.

How to compute it?

Method:

(1) Apply decomposition rule if necessary.
(2) Remove redundant FDs.
(3) Left-reduce FDs.

Example:

$F = \{AB \to C,\ C \to A,\ BC \to D,$
$ACD \to B,\ D \to GE,\ BE \to C,$
$CG \to B,\ CE \to AG\}.$

(1) $F \equiv F_1$, where $F_1 = \{AB \overset{1}{\to} C,\ C \overset{2}{\to} A,\ BC \overset{3}{\to} D,\ ACD \overset{4}{\to} B,\ D \overset{5}{\to} G,\ D \overset{6}{\to} E,\ BE \overset{7}{\to} C,\ CG \overset{8}{\to} B,\ CE \overset{9}{\to} A,\ CE \overset{10}{\to} G\}$

To see if an FD is redundant, compute the closure of its LHS WITHOUT using the FD.

(1) To see if $AB \to C$ is redundant, compute $(AB)^+_{F_1-\{1\}} = AB \Rightarrow 1$ is not redundant. So keep it.

(2) $(C)^+_{F_1-\{2\}} = C \Rightarrow 2$ is not redundant.

(3) $(BC)^+_{F_1-\{3\}} = BCA \Rightarrow 3$ is not redundant.

(4) $(ACD)^+_{F_1-\{4\}} = ACDGE\underline{B} \Rightarrow 4$ is redundant. So, remove it. $\to F_1 := F_1 - \{4\}$.

Similarly, can check that 5-8 are not redundant.

(9) $(CE)^+_{F_1-\{9\}} = CE\underline{A}G \ldots \Rightarrow 9$ is redundant. $\to F_1 := F_1 - \{9\}$.

(10) 10 is not redundant.

Resulting $F_1 = \{AB \to C,\ C \to A,\ BC \to D,$
$D \to G,\ D \to E,\ BE \to C, CG \to B,\ CE \to G\}$.

To test left-redundancy:
Take each FD $X \to A \in F_1$.
See $F_1 \models (X - B) \to A,\ \forall B \in X$.

(1) $F_1 \not\models A \to C$ since $(A)^+_{F_1} \not\ni C$. $F_1 \not\models B \to C$ since $(B)^+_{F_1} \not\ni C$.

In this example, none of the FDs in $F_1$ are left-redundant.

So, $F_1$ is a canonical cover of $F$.

<u>Example</u>:

$F = \{A \to B,\ B \to A,\ B \to C,\ AC \to D,$
$AD \to G,\ BG \to H, BD \to E\}$.

Can check that none of the FDs in $F$ are redundant.

How about left-redundancy?

(1) is not left-redundant, since $F \not\models \emptyset \to B$, i.e., $B \notin \emptyset^+$.

Similarly, (2) and (3) are not left-redundant.

(4) $AC \to D$.
$F \not\models C \to D$ since $D \notin (C)^+$ – not left-redundant.
$F \models A \to D$ since $D \in (A)^+ = ABCD\ldots$ – it is left-redundant.

Since $F \models A \to D$ and $F$ contains the FD $AC \to D$, this FD is left-redundant. So, replace it by $A \to D$.

(5) $AD \to G$.
$F \not\models D \to G$ since $G \notin (D)^+$ – not left-redundant.
$F \models A \to G$ since $G \in (A)^+ = ABCDG\ldots$ – it is left-redundant.

So, replace $AD \to G$ by $A \to G$.

Similarly, replace $BG \to H$ by $B \to H$, and $BD \to E$ by $B \to E$.

The resulting set of FDs is

$F' = \{A \to B,\ B \to A,\ B \to C,\ A \to D,$
$A \to G,\ B \to H, B \to E\}$.

Now can check no FD is left-redundant $\Rightarrow F'$ is a canonical cover of $F$.

Example:

$$F = \{A \overset{1}{\to} B,\ A \overset{2}{\to} C,\ AB \overset{3}{\to} C,\ ABC \overset{4}{\to} D\}.$$

Can check:
Only (3) is redundant. – Why?
(4) is left-redundant.

(4) may be initially replaced by say $AB \to D$. This is still left-redundant.

$\Rightarrow$ replaced by $A \to D$

$\Rightarrow$ A canonical cover is

$$\{A \to B,\ A \to C,\ A \to D\}$$

## Design Theory of Relational Databases
(Continued)

- A sufficient condition for a (binary) decomposition to be lossless-join.
  Give examples.
- A motivation for *normal forms*.
  - 1NF.
  - 3NF.
  - motivating examples.
- 3NF – definition and example.
- Algorithm for a lossless-join 3NF decomposition.
- Preservation of dependencies – an example and definition.
- BCNF – an example.
- Algorithm for BCNF decomposition.
- Discussion.

A sufficient condition for a (binary) decomposition to be lossless-join.

Binary decomposition of a scheme $R$ with a given set of FDs $F$.

$$R \rightarrow \begin{cases} S_1 \\ \\ S_2 \end{cases}$$

Suppose that $S_1 \cap S_2 \rightarrow S_1$ or $S_1 \cap S_2 \rightarrow S_2$ is logically implied by $F$. Then the decomposition above is lossless-join.

Why? – Explain.

Example 1:

Let $C$ = Course, $G$ = Grade, $S$ = Student, and $T$ = Teacher. Let $R = \{CGST\}$ and $F = \{CS \to G,\ C \to T\}$.

$$R = CSGT \to \begin{cases} S_1 = CSG \\ \\ S_2 = CST \end{cases}$$

$F \models CS \to G$, $S_1 \cap S_2 = CS$, and thus $F \models CS \to CSG$. Hence the above decomposition is lossless-join.

A Motivation for Normal Forms

Normal forms – formalization of design anomalies (in insertion, deletion and representation) and their solution.

How many normal forms are there?

First Normal Form (1NF)$\surd$
Second Normal Form (2NF)
Third Normal Form (3NF)$\surd$
Fourth Normal Form (4NF)
Fifth Normal Form (5NF)
Boyce-Codd Normal Form (BCNF)$\surd$
Project-Join Normal Form (PJNF)

$\surd$: dealt with in this course.

## First Normal Form (1NF)

A relation is in 1NF if each attribute in it has only atomic values (rather than set or tuple values).

Example 2:

Let $R = \{emp, child\}$ and $r(R)$ be

| $emp$ | $child$ |
|---|---|
| $Peter$ | $Jeromy$ |
| | $Joan$ |
| | $Mary$ |

Fig 1.

| $emp$ | $child$ |
|---|---|
| $Peter$ | $Jeromy$ |
| $Peter$ | $Joan$ |
| $Peter$ | $Mary$ |

Fig 2.

The structuring in Fig 1 is not allowed since $\underline{child}$ has a set value here.

In essence only "normal" tuples are allowed. For example, in 1NF the above information would be represented as in Fig 2.

Third Normal Form (3NF)

Example 3:

Let $R = \{name, id, addr, sal, dno, dname, mgrid\}$ and

$$F = \{\{name, addr\} \rightarrow id,$$
$$id \rightarrow \{name, addr, sal, dno, dname, mgrid\},$$
$$dno \rightarrow \{dname, mgrid\}\}$$

Candidate keys = {name, addr} and id (only keys)

In dno $\rightarrow$ dname, mgrid,

- dno is *not* a superkey.
- dname (mgrid too) is not an element of any candidate key.

What does it mean?

Well those FDs signify some design anomalies:

* department info (dname and mgrid) wastefully repeated for each emp working in the department. (We are assuming each emp works for one department.)

* Info about a new emp cannot be stored (without using null values) unless her department (info) is known.

*Definition*: $R$ – a relation scheme. $F$ – given FDs for $R$. (Can assume each FD is of the form $X \to A$.)

$R$ is in 3NF, if for each FD $X \to A$ in $F$

- either $A \in X$ (in this case, the FD is *trivial*, or
- $X$ is a superkey of $R$, or
- $A$ belongs to some candidate key of $R$.

Example 4:

Let $C =$ Course, $G =$ Grade, $S =$ Student, and $T =$ Teacher. Let $R = \{CGST\}$ and $F = \{C \rightarrow T,\ CS \rightarrow G\}$.
Assume only one section per course.

Candidate keys = CS (only key).

Consider the FDs:
$CS \rightarrow G$ – no problem wrt 3NF.
$C \rightarrow T$ – *violates* 3NF requirement.
Why?

Decompose $R = CSGT$ into $S_1 = CT$ and $S_2 = CSG$.

Note: This decomposition is different from the earlier decomposition for $R$.

- The decomposition is lossless-join (why?).
- $S_1$ and $S_2$ are both in 3NF (why?).

## Algorithm For Generating a Lossless-Join 3NF Decomposition

Input: a relation scheme $R$ and a set of FDs $F$ on $R$.

Output: A set $S$ of relation schemes $\{R_1, \ldots, R_m\}$ such that each $R_i$ is in 3NF and the decomposition is lossless-join.

begin

1. obtain a canonical cover $G$ for $F$;
2. Let $S =$;
3. for each FD $X \rightarrow A \in G$ do
   (a) if none of the relation schemes in $S$ contains $XA$ then create a relation scheme $XA$ and add it to the set $S$;
4. if no relation scheme added in $S$ in step (3) is a superkey, then add some cadidate key $K$ to $S$;
5. Return $S$;

end.

Example 5:

$F = \{AB \to C,\ C \to A,\ BC \to D, ACD \to B,$
$D \to GE,\ BE \to C, CG \to B,\ CE \to AG\}.$

You may check that $H$ below is a canonical cover of $F$.

$H = \{AB \to C,\ C \to A,\ BC \to D,\ D \to G,$
$D \to E,\ BE \to C, CG \to B,\ CE \to G\}.$

So, following the algorithm, $\underline{R} = \{ABC,\ BCD,$ $DG,\ DE,\ BEC,\ CGB,\ CEG\}$ is the output set of relation schemes.

Also, $(ABC)^+_H = ABC \stackrel{3}{\Rightarrow} ABCD \stackrel{4}{\Rightarrow} ABCDG \stackrel{5}{\Rightarrow} ABCDEG$.

$\Rightarrow$ ABC is a superkey.

$\Rightarrow$ $\underline{R}$ is a lossless-join 3NF decomposition of $R = ABCDEG$.

Example 5: (Continued)

Further refinements.

- $DG$ and $DE$ were created using FDs $D \to G$ and $D \to E$. Both have the *same LHS*. So, can combine these two into $DGE$.

Finally, the (output) required lossless-join 3NF decomposition is:

$\underline{R} = \{ABC,\ BCD,\ DGE,\ BEC,\ CGB,\ CEG\}$

## Preservation of Dependencies
– A Motivating Example

Let $R = CSZ$, where $C =$ City, $S =$ Street, and $Z =$ Zip code.

$F = \{CS \rightarrow Z,\ Z \rightarrow C\}$.

Suppose $R$ is decomposed into $CZ$ and $ZS$ (notice that this is lossless join). The info for $R$ is maintained in the 2 relations $CZ$ and $ZS$.

- FDs should be preserved against updates. – explain.
- Using $CZ$, we can always preserve $Z \rightarrow C$.
- Using $ZS$, can only "preserve" trivial FDs, like $ZS \rightarrow Z, \ldots$
- Only way to see if an insertion violates $CS \rightarrow Z$ is to join $CZ$ and $ZS$ and check. – Quite costly

In the previous example, FDs (wrt $F$) "captured" by $ZC$ are $\{Z \rightarrow C\}$, and FDs "captured" by $ZS$ are $\emptyset$.

Formally, for a decomposition

$$R \rightarrow \begin{cases} S_1 \\ \\ S_2 \end{cases}$$

and FDs $F$ for $R$,
$\Pi_{S_i}(F) = \{X \rightarrow A | A \notin X$ and $XA \subseteq S_i$ and $F \models X \rightarrow A$ (i.e., $A \in X^+$)$\}$, $i = 1, 2$.

$\Pi_{ZC}(F) = \{Z \rightarrow C\}$.
$\Pi_{ZS}(F) = \emptyset$.

**Note**: <u>Always</u> $F \models \Pi_{S_1}(F) \cup \Pi_{S_2}(F)$. Whenever $\Pi_{S_1}(F) \cup \Pi_{S_2}(F) \models F$, we say that the decomposition preserves FDs.

In the above example, $\Pi_{S_1}(F) \cup \Pi_{S_2}(F) \not\models CS \rightarrow Z$. So, this FD is lost.

**Note**: In the above, we could have

$$R \rightarrow \begin{cases} S_1 \\ S_2 \\ \vdots \\ S_n \end{cases} \qquad \text{instead of} \qquad R \rightarrow \begin{cases} S_1 \\ \\ S_2 \end{cases}$$

## Boyce-Codd Normal Form (BCNF)

Example

Let $R = CSZ$, and $F = \{CS \rightarrow Z,\ Z \rightarrow C\}$.

Only candidate keys $= CS$ and $ZS$.

In $Z \rightarrow C$, $Z$ is not a superkey. Yet, $C \in$ a candidate key, i.e., $C$ is a prime attribute.

$\Rightarrow CSZ$ is in 3NF.

However, City (name) is repeated once for each Street in the City.
– Redundancy.
– Can't store City info without Street info.

In $CSZ \rightarrow \begin{cases} CZ \\ \\ ZS \end{cases}$ redundancy is avoided.

$\Pi_{ZC}(F) = \{Z \rightarrow C\}$.
$\Pi_{ZS}(F) = \emptyset$.
$Z$ – a candidate key for $CZ$.

**Definition**: Let $R$ be a relation scheme, and $F$ be a set of FDs on $R$. (can assume each FD is of the form $X \rightarrow A$.) $R$ is in Boyce-Codd Normal Form (BCNF) if for each $X \rightarrow A$ in $F$,

- either $A \in X$, i.e., $X \rightarrow A$ is a trivial FD, or
- $X$ is a superkey of $R$.

<u>Example</u>

$CZ$ and $ZS$ are each in BCNF wrt their FDs, viz., $\{Z \rightarrow C\}$ and $\emptyset$, respectively.

**Note**: If $R$ is in BCNF wrt $F$ then $R$ is definitely in 3NF wrt $F$.

– Why?

## Algorithm for A Lossless-Join BCNF Decomposition of $R$

**Input**: $R$ and $F$

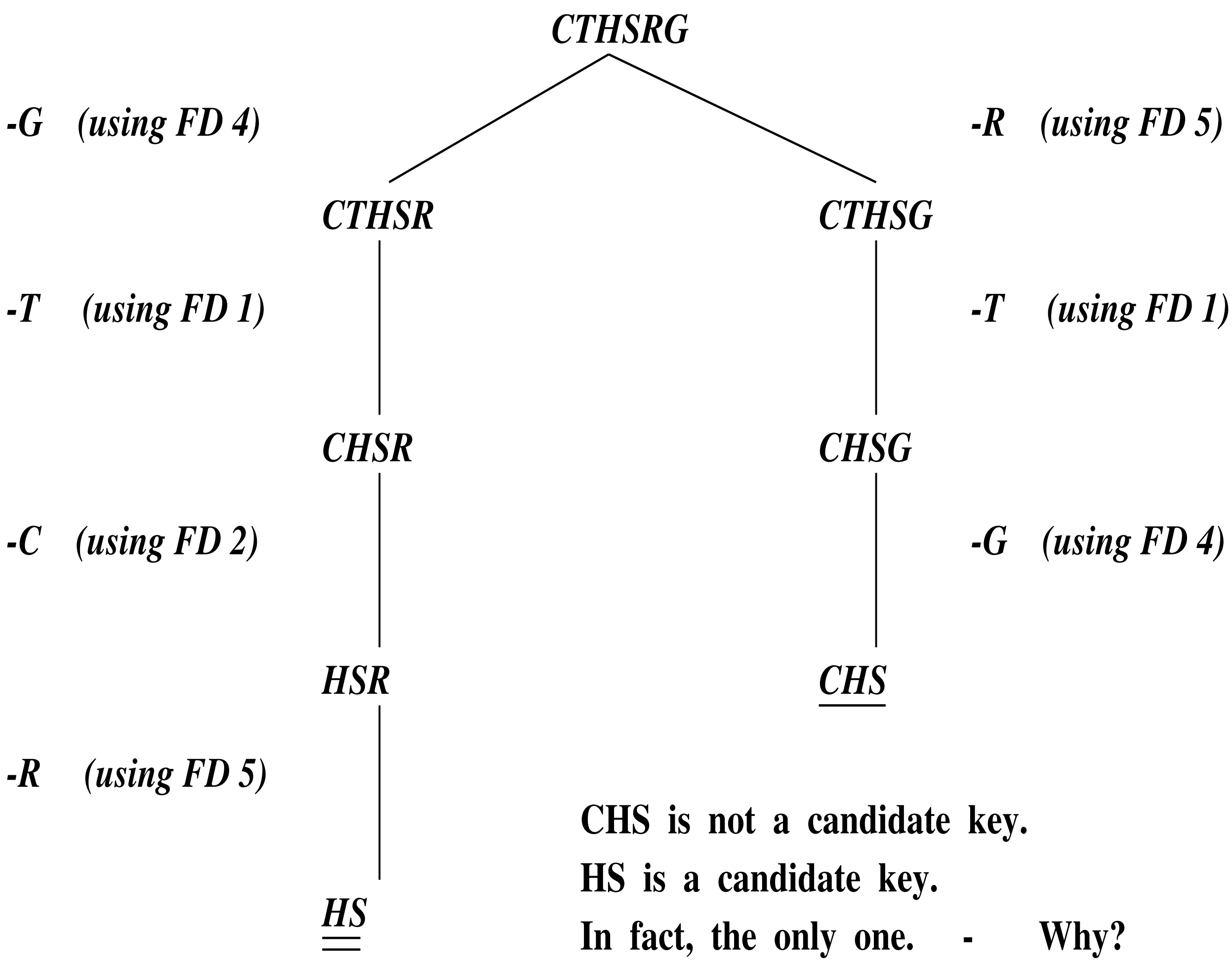
**Output**: $\underline{R}$ – lossless-join BCNF decomposition of $R$.

```
begin
  let R = {R};
  over := false;
  repeat
    if there is a relation scheme S in R that is
    not in BCNF wrt Π_S(F) then
      begin
        let X → A in Π_S(F) be the FD
          violating BCNF;
        replace S in R by S_1 = XA and
          S_2 = S − A;
      end
    else
      over := true;
  until over;
end;
```

## Example

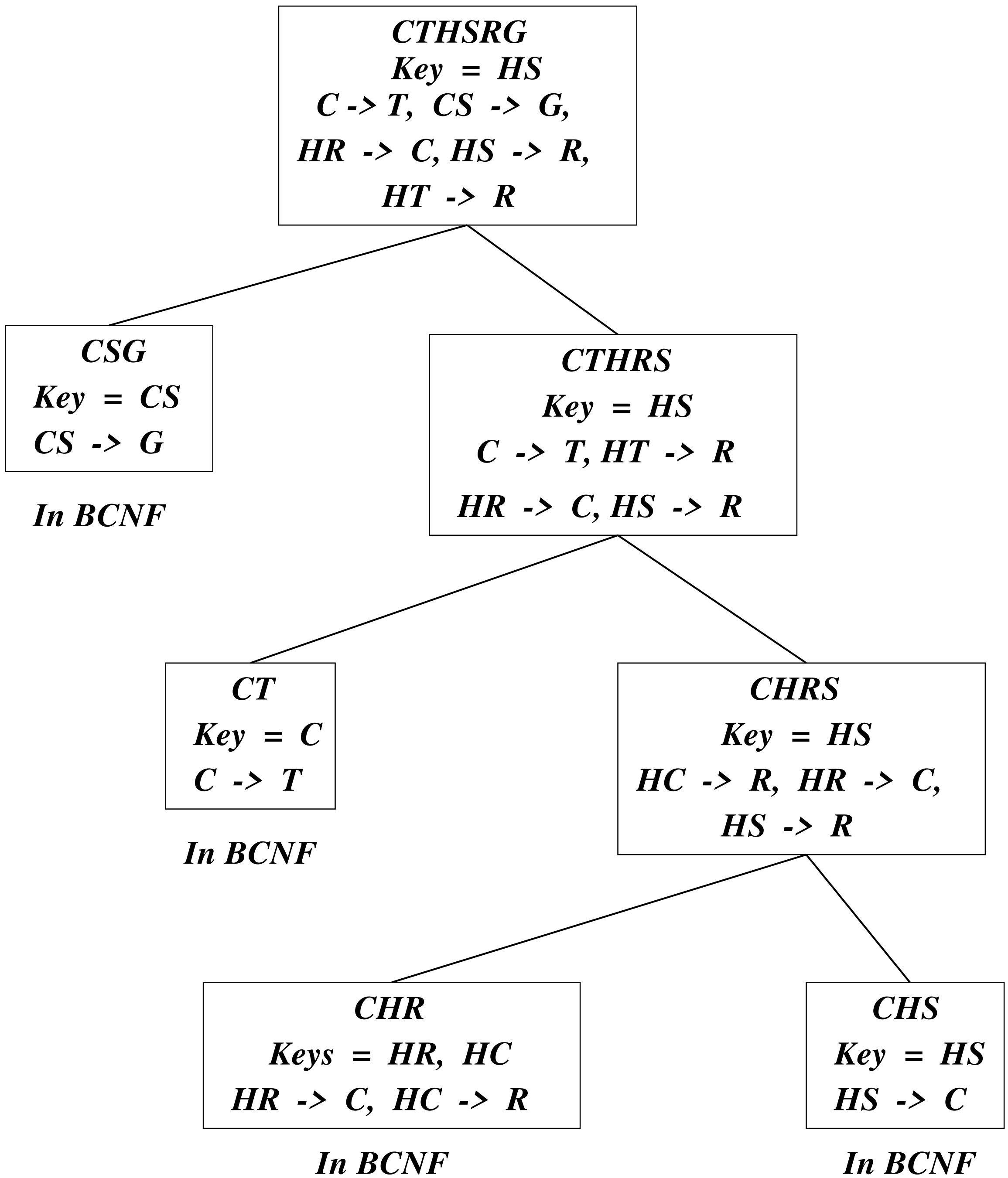
Let $R = CTHRSG$ where $C =$ Course, $T =$ Teacher, $H =$ Hour, $R =$ Room, $S =$ Student, and $G =$ Grade. Let $F = \{C \to T,\ HR \to C,\ HT \to R,\ CS \to G,\ HS \to R\}$.

Step 1: Find **all** candidate keys and prove that what you have are the only candidate keys.




**CHS is not a candidate key.**
**HS is a candidate key.**
**In fact, the only one. - Why?**

Step 2: Develop the decomposition tree.



**Output**:

$\underline{R} = \{CSG,\ CT,\ CHR,\ CHS\}$

$\Pi_{CSG}(F) = \{CS \rightarrow G\}$.

$\Pi_{CT}(F) = \{C \rightarrow T\}$.

$\Pi_{CHR}(F) = \{CH \rightarrow R,\ HR \rightarrow C\}$.

$\Pi_{CHS}(F) = \{HS \rightarrow C\}$.

$\Pi_{CSG}(F) \cup \Pi_{CT}(F) \cup \Pi_{CHR}(F) \cup \Pi_{CHS}(F) \not\models HT \rightarrow R$, where $HT \rightarrow R \in F$.

$\Rightarrow$ This FD is not preserved by decomposition. In other words, this is not a dependency preserving decomposition.