Database Design

Nguyễn Văn Diêu

HO CHI MINH CITY UNIVERSITY OF TRANSPORT

2020

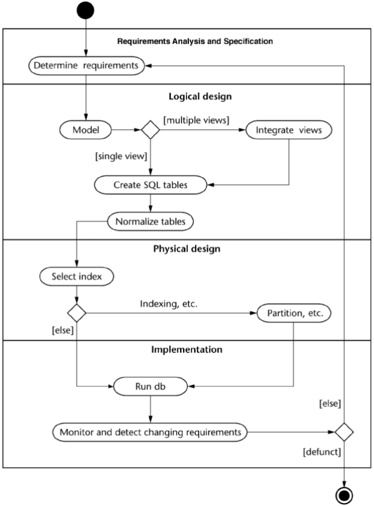
Kiến thức - Kỹ năng - Sáng tạo - Hội nhập

Sứ mệnh - Tầm nhìn

Triết lý Giáo dục - Giá trị cốt lõi

Table of contents

1.1. Database Design Life Cycle



1.2. Requirements Analysis

1. Interviewing:
   - producers
   - users

   Producing a formal requirements specification.

2. That specification includes:
   - Data required for processing
   - The natural data relationships
   - the software platform for the database implementation

1.3. Logical Database Design

1. Developing a conceptual model of the database
2. Refines that model into normalized SQL tables
3. Using techniques such as:
   - Entity-relationship (ER)
   - Unified Modeling Language (UML)

1.4. Physical Database Design

1. The goal of physical design is to maximize the performance of the database
2. It involves the selection:
   - Indexes
   - Partitioning
   - Clustering
   - Selective materialization of data
3. Begin after the SQL tables have been defined and normalized
4. It focuses on the methods:
   - Storing tables on disk
   - Accessing tables on disk
   - Enable the database to operate with high efficiency

1.5. Implementation, monitoring, and modification

1. Once the logical and physical design is completed, the database can be created through implementation of the formal schema using the data definition language (DDL) of a DBMS
2. Then the data manipulation language (DML) can be used to query and update the database, as well as to set up indexes and establish constraints such as referential integrity
3. The language SQL contains both DDL and DML constructs

2.1. Functional Dependencies Definition

Let:

- $R(A_1, \cdots, A_n)$ be a relational scheme,
- X and Y be subsets of $\{A_1, \cdots, A_n\}$

We say:

- $X \rightarrow Y$, read "*X functionally determines Y*" or "*Y functionally depends on X*" if:

  $\forall (t_1, t_2) \in \forall r(R):$

  Not possible: $t_1(X) = t_2(X)$ *but* $t_1(Y) \neq t_2(Y)$

- In the FD $X \rightarrow Y$, X is called the *Left side* and Y called the *Right side*.

2.2. Notational Conventions

- $A, B, C, \cdots$ stand for single attributes
- $U, V, \cdots, Z$ stand for sets of attributes
- R is used to denote a *relation scheme*
- $r(R)$ *relation*, the current instance of scheme R
- $A_1 \cdots A_n$ set of attributes $\{A_1, \cdots, A_n\}$
- XY stand for $X \cup Y$
- XA or AX stand for $X \cup \{A\}$

e.g.

StudentGrade(StudentID, StudentName,SubjectID, SubjectName, Grades)

One current **StudentGrade**:

StudentGrade	(StudentID	StudentName	SubjectID	SubjectName	Grades)
	T01	Nam	S01	Database	7
	T01	Nam	S02	Math	6
	T02	Tuan	S01	Database	9
	T02	Tuan	S02	Math	5
	T03	Cuong	S01	Database	8
	T04	Quoc	S02	Math	8

StudentID → StudentName

SubjectID → SubjectName

StudentID, SubjectID → Grades

StudentID, SubjectID, Graves -> Grades

StudentName, SubjectID -> SubjectName

StudentID, SubjectName -> Grades

2.3. Satisfaction of Dependencies

- r *satisfies* FD $X \to Y$

  if $\forall (t_1, t_2) \in r(R) : t_1(X) = t_2(X)$ *then* $t_1(Y) = t_2(Y)$

- "if ... then" statement, it can be satisfied either by:
  1. $t_1(X) \neq t_2(X)$ or
  2. $t_1(Y) = t_2(Y)$
- If r does not satisfy $X \to Y$, the r violate $X \to Y$.
- If we declared $X \to Y$ hold for R, then all instance r of scheme R will satisfy $X \to Y$.
- FD $X \to \emptyset$ is *trivially satisfied* by any relation.
- FD $\emptyset \to Y$ is satisfied by those relations in which every tuple has the same $Y - value$.

2.4. FDs is an Integrity Constraints

Let: $R(ABC)$; FD: $A \to B$

There is an *integrity constraint* of scheme R.

- Context: R
- Condition:
  $\forall (t_1, t_2) \in \forall r(R) : t_1.A = t_2.A$

  $t_1.B = t_2.B$

  end.
- Influence table:

	Insert	Delete	Update
R	+	-	+(A/B)

2.5. Inference FD

Let R; A, B, C : attributes.
FD: $A \rightarrow B$ and $B \rightarrow C$ are known to hold in R.
We claim that $A \rightarrow C$ must also hold in R

Proof by Contradiction

Suppose r satisfies $A \rightarrow B$ and $B \rightarrow C$,

but $\exists(t_1, t_2) \in r : t_1.A = t_2.A \ and \ t_1.C \neq t_2.C$

- if $t_1.B \neq t_2.B$, then $r \ violate \ A \rightarrow B$
- if $t_1.B = t_2.B$ but $t_1.C \neq t_2.C$ then $r \ violate \ B \rightarrow C$

Hence r must satisfy $A \rightarrow C$ ■

Direct Proof

$\forall(t_1, t_2) \in \forall r(R) \ : \ t_1.A = t_2.A \implies t_1.B = t_2.B \quad (A \rightarrow B)$
$\implies t_1.C = t_2.C \quad (B \rightarrow C)$

from FD definition $\implies A \rightarrow C$ ■

2.6. Logically Inference

- Let F set of functional dependencies hold on R
- $X \rightarrow Y$ be a functional dependency.
  - F *logically inference* $X \rightarrow Y$ written $F \vDash X \rightarrow Y$
    if $\forall r(R)$ *satisfies* F *also satisfies* $X \rightarrow Y$

e.g. $F = \{ A \rightarrow B,\ B \rightarrow C \}$
then $A \rightarrow C$ is logically inference by F
that is, $\{ A \rightarrow B,\ B \rightarrow C \} \vDash A \rightarrow C$

3.1. Amstrong's Inference Axioms

Giving U be a *universal set* of attributes. F be a FDs set involving only attributes in U. The inference rule are:

1. *Reflexivity*. If $Y \subseteq X \subseteq U$, then $X \rightarrow Y$ is logically implied by F. This rules gives the *trivial dependencies* . It hold in every relation.
2. *Augmentation*. $\{\ X \rightarrow Y,\ Z \subseteq U\ \} \models XZ \rightarrow YZ$.
3. *Transitivity*. $\{\ X \rightarrow Y,\ Y \rightarrow Z\ \} \models X \rightarrow Z$.

e.g.

Let $R(ABCD)$, $F = \{A \to C, B \to D\}$.

$F \models AB \to ABCD$?

1. $A \to C$ (given)
2. $AB \to ABC$ [augmentation of (1) by AB]
3. $B \to D$ (given)
4. $ABC \to ABCD$ [augmentation of (3) by ABC]
5. $AB \to ABCD$ [transitivity applied to (2) and (4)] ■

3.2. Lemma

Amstrong's Axioms are sound. That is,

if $X \to Y$ is deduced from F using the axioms, then $X \to Y$ is true in any relation in which the dependencies of F are true.

Proof:

Reflexivity. Clearly sound.

Augmentation. Suppose relation r satisfies $X \to Y$, but

$\exists (t_1, t_2) : t_1(XZ) = t_2(XZ)$ *and* $t_1(YZ) \neq t_2(YZ)$

$\implies t_1.Y \neq t_2.Y$

So $t_1.X = t_2.X$ *and* $t_1.Y \neq t_2.Y$, violating $X \to Y$ holds for r. ■

Transitivity. Similarly, as an exercise.

3.3. Lemma

There are several other inference *rules* that follow from Amstrong's Axioms

Lemma:

1. *Union*. $\{ X \rightarrow Y, \; X \rightarrow Z \} \vDash X \rightarrow YZ$
2. *Pseudotransitivity*. $\{ X \rightarrow Y, \; WY \rightarrow Z \} \vDash WX \rightarrow Z$
3. *Decomposition*. $\{ X \rightarrow Y, \; Z \subseteq Y \} \vDash X \rightarrow Z$

Proof

Proof:

Union. Given $X \to Y \vDash X \to XY$ (augment)

Given $X \to Z \vDash XY \to YZ$ (augment)

By transitivity $\{\, X \to XY,\; XY \to YZ \,\} \vDash X \to YZ$

Pseudotransitivity. Given $X \to Y \vDash WX \to WY$ (augment)

Since given $WY \to Z \vDash WX \to Z$ (transitivity)

Decomposition. Given $Z \subseteq Y \vDash Y \to Z$ (reflexivity)

Given $X \to Y$, by transitivity, $\vDash X \to Z$ ■

e.g.

$F = \{\ AB \to CD,\ C \to B,\ A \to C,\ BD \to G,\ H \to K\ \}$

Show $F \vDash AH \to GK$

$AB \to CD \vDash AB \to D$ [decomposition] (1)

$\{\ A \to C,\ C \to B\ \} \vDash A \to B$ [transitivity] (2)

(2) $A \to B \vDash A \to AB$ [augment] (3)

$\{\ (3)\ A \to AB, (1)\ AB \to D\ \} \vDash A \to D$ [transitivity] (4)

$\{\ (2)\ A \to B, (4)\ A \to D\ \} \vDash A \to BD$ [union] (5)

$\{\ (5)\ A \to BD,\ BD \to G\ \} \vDash A \to G$ [transitivity] (6)

$H \to K \vDash GH \to GK$ [augment] (7)

$\{\ (6)\ A \to G,\ GH \to GK\ \} \vDash AH \to GK$ [pseudotransitivity] ■

4.1. Closure of FDs Set

- We define F^+, the *closure* of F, to be the set of functional dependencies that are logically implied by F; i.e.,
- $F^+ = \{ X \to Y \mid F \vDash X \to Y \}$

e.g. $F = \{ AB \to C,\ C \to B \}$ hold on $R(ABC)$, then:

$$F^+ = \{ A \to A,\ AB \to A,\ AC \to A,\ ABC \to A,\ B \to B,$$
$$AB \to B,\ BC \to B,\ ABC \to B,\ C \to C,\ AC \to C,$$
$$BC \to C,\ ABC \to C,\ AB \to AB,\ ABC \to AB,\ AC \to AC,$$
$$ABC \to AC,\ BC \to BC,\ ABC \to BC,\ ABC \to ABC,$$
$$AB \to C,\ AB \to AC,\ AB \to BC,\ AB \to ABC,\ C \to B,$$
$$C \to BC,\ AC \to B,\ AC \to AB \}$$

e.g.

Let $R(ABC)$ and $F = \{ A \to B,\ B \to C \}$. Then F^+ consists of all those dependencies $X \to Y$ such that either

1. X contains A e.g., $ABC \to AB$, $AB \to BC$, or $A \to C$,
2. X contains B but not A, and Y does not contain A e.g., $BC \to B$, $B \to C$, or $B \to \varnothing$, and
3. $X \to Y$ is one of the three dependencies $C \to C$, $C \to \varnothing$, or $\varnothing \to \varnothing$.

4.2. Closure of Attribute Set

Let:

- R be an attributes set.
- F be a functional dependencies set hold on R.
- X be a subset of R.

We define X_F^+ is the *closure* of X (with *respect* of F) is the set of attributes A such that $X \rightarrow A$ can be logically implied from F (by FD definition).

$$X_F^+ = \{ A \mid F \vDash X \rightarrow A \}$$

e.g.

$R(ABCD)$, $F = \{ B \rightarrow D,\ A \rightarrow B,\ C \rightarrow B \}$

$A_F^+ = ABD$

4.3. Algorithm

Closure of a set of attributes with respect to a set of functional dependencies.

Input: R, F, $X \subseteq R$

Output: X_F^+

Method: Computing a sequence attributes sets $X_F^{(0)}, X_F^{(1)}, \ldots$ by the rule:

1. $X_F^{(0)} \leftarrow X$
2. Repeat $Y \rightarrow Z \in F : Y \subseteq X_F^{(i)}$

   $X_F^{(i+1)} \leftarrow X_F^{(i)} \cup Z$

   Until $X_F^{(i+1)}$ is not change

Since $X = X_F^{(0)} \subseteq \cdots \subseteq X_F^{(i)} \subseteq \cdots \subseteq R$, and R is finite, we must eventually reach i such that $X_F^{(i)} = X_F^{(i+1)}$.

e.g.

$$F = \{\ AB \to C,\quad D \to EG,\quad C \to A,\quad BE \to C$$
$$BC \to D,\quad CG \to BD,\quad ACD \to B,\quad CE \to AG\}$$

$X = BD$, compute X_F^+:

$X_F^{(0)} = BD$

$X_F^{(1)} = BDEG$

$X_F^{(2)} = BCDEG$

$X_F^{(3)} = ABCDEG$

$X_F^{(3)} = X_F^{(4)} = \cdots$

Thus $(DB)_F^+ = ABCDEG$

4.4. Lemma

Dependencies Logically Implied by Closure Set.

Let R , F , $X \subseteq R$, X_F^+, then:

- $X \rightarrow X_F^+$
- $Z \subseteq X_F^+$, then $X \rightarrow Z$

Proof: $X \rightarrow Z$:

Suppose $\exists (t_1, t_2) : t_1.X = t_2.X$, but $t_1.Z \neq t_2.Z$

So $\exists$ $A \in Z : t_1.A \neq t_2.A$. That is contradiction with closure set definition, because of $A \in Z \in X_F^+$ ■

Similarly with $X \rightarrow X_F^+$ ■

4.5. Testing Membership in F^+

Let $X \rightarrow Y$ and F
Determine $F \vDash X \rightarrow Y$? We have three methods:

1. Using FD definition
2. Using Amstrong's axioms
3. Test if $X \rightarrow Y \in F^+$

In the 3^{th} method, F^+ can be very lager

How to show $X \rightarrow Y \in F^+$ without generating all of F^+

Algorithm:

1. X_F^+
2. *if* $Y \subseteq X_F^+$ *then* $F \vDash X \rightarrow Y$
   if $Y \nsubseteq X_F^+$ *then* $F \nvDash X \rightarrow Y$

e.g.

$$F = \{\ A \to D,\ AB \to E,\ BI \to E,\ CD \to I,\ E \to C\ \}$$

Proofing items below by using Amstrong's axioms as an exercise.

1. $AE \to AI \in F^+$?

   $AE_F^+ = ACDEI$ contain the right side of the FD, so

   $AE \longrightarrow AI \in F^+$ or $F \vDash AE \to AI$ ∎

2. $DB \to C \in F^+$?

   $DB_F^+ = DB$ not contain the right side of the FD, so

   $DB \to C \notin F^+$ or $F \nvDash DB \to C$ ∎

4.6. Covers and Equivalence of Dependencies Sets

Definition:

- Two FDs F and G over R are *equivalent* if $F^+ = G^+$
- Denote $F \equiv G$
- If $F \equiv G$, then F is a *cover* for G

Test whether F and G are equivalent:

- *Step 1:* $F \vDash G$
  - Test every $g : X \to Y \in G$ is $g \in F^+$
  - $g : X \to Y \in G$, compute X_F^+
  - If $Y \subseteq X_F^+$, then $g \in F^+$, if not: $g \notin F^+$
  - If $\exists\ g \in G, \notin F^+$, then surely $F^+ \neq G^+$
- *Step 2:* $G \vDash F$
  Analogous manner *Step 1:*, test every $f \in F$ is $f \in G^+$

e.g.

$$F = \{\ A \to BC,\ A \to D,\ CD \to E\ \}$$
$$G = \{\ A \to BCE,\ A \to ABD,\ CD \to E\ \}$$
$$F \equiv G\ ?$$

Step 1: $F \models G$.

$A \to BCE$: $A_F^+ = ABCDE \Rightarrow F \models A \to BCE$

$A \to ABD$: $A_F^+ = ABCDE \Rightarrow F \models A \to ABD$

$CD \to E$: $CD_F^+ = CDE \Rightarrow F \models CD \to E$

Step 2: $G \models F$.

$A \to BC$: $A_G^+ = ABCED \Rightarrow G \models A \to BC$

$A \to D$: $A_G^+ = ABCED \Rightarrow G \models A \to D$

$CD \to E$: $CD_G^+ = CDE \Rightarrow G \models CD \to E$

$F \equiv G\ \blacksquare$

4.7. Minimum Covers

Definition

- F have many cover sets, its equivalent to F
- We define a set F_c is a *minimal cover* of F if:

1. Every *right side* of FD in F_c is a single attribute
2. $\nexists\ X \rightarrow A \in F_c\ :\ \left\{ F_c - \{ X \rightarrow A \} \right\} \equiv F_c$
3. $\nexists\ X \rightarrow A \in F_c$ *and* $Z \subseteq X$:
   $\left\{ F_c - \{X \rightarrow A\} \right\} \cup \{Z \rightarrow A\} \equiv F_c$

One dependencies G maybe there are many minimum covers.

4.8. Algorithm

Find one minimum cover of dependencies set.
Input: Dependencies F.
Output: One minimum cover.
Method: Compute a sequence steps:

1. Decompose right side FDs in F to one attribute.
2. For each $X \to A$ in F:
   if $F - \{X \to A\} \equiv F$ then
   $F = F - \{X \to A\}$
3. For each $X \to A$ in F:
   if $Z \subset X$ and $\left\{ F - \{X \to A\} \right\} \cup \{Z \to A\} \equiv F$ then
   $F = \left\{ F - \{X \to A\} \right\} \cup \{Z \to A\}$.

Easy proofing three steps, we see as an exercise ■

e.g.

$$F = \{\ AB \to C,\quad D \to EG,\quad C \to A,\quad BE \to C,\quad BC \to D$$
$$CG \to BD,\quad ACD \to B,\quad CE \to AG\ \}$$

Step 1, split right side FDs:

$AB \to C,\ \ D \to E,\ \ D \to G,\ \ C \to A,\ \ BE \to C,\ \ BC \to D$

$CG \to B,\ \ CG \to D,\ \ ACD \to B,\ \ CE \to A,\ \ CE \to G$

Step 2, remove $CE \to A$, $CG \to B$

Step 3, $ACD \to B$ replaced by $CD \to B$

One minimum cover

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

4.9. Projected Functional Dependencies

R, F ; R_1: Attributes Set.

F_1 = *projected FDs of* F *on* R_1,

denote $F_1 = \pi_{R_1}(F)$:

- F_1 follow from F and
- Involve only attributes of R_1.

4.10. Algorithm

Input: R, F ; R_1 : Attributes Set

Output: $F_1 = \pi_{R_1}(F)$

Method: Compute a sequence steps

1. $F_1 \leftarrow \varnothing$
2. *For each* $X \subset \mathcal{R}_1$
   - Compute X_F^+
   - If $X_F^+ \cap R_1 \neq \varnothing$:

     $$F_1 = F_1 + \left\{\ X \rightarrow X_F^+ \cap R_1\ \right\}$$
3. $F_1 \leftarrow$ a minimal cover of F_1

e.g.

$R(ABCD)$,$F = \{\ A \to B,\ B \to C,\ C \to D\ \}$

Find $F_1 = \pi_{R_1(ACD)}(F)$

- *First*, $A_F^+ = ABCD \Rightarrow F_1 = \{\ A \to C,\ A \to D\ \}$
- *Next*, $C_F^+ = CD \Rightarrow F_1 = \{\ A \to C,\ A \to D,\ C \to D\ \}$
- *Since* $A_F^+ = R_1$, no any superset of A can find new FDs.

  Thus, the only CD can be compute, but $CD^+ = CD$, add nothing.
- *Done* $F_1 = \{A \to C,\ A \to D,\ C \to D\}$
- Find a minimum cover of F_1,

  Thus, $F_1 = \{\ A \to C,\ C \to D\ \}$ ■

5.1. Keys by Functional Dependencies

$R(A_1, \cdots, A_n)$ and F; subset $K \subseteq A_1A_2\cdots A_n$

K is a *key* of R if:

1. $K \rightarrow A_1A_2\cdots A_n$ is in F^+, and
2. $\nexists\ Y \subsetneq K : Y \rightarrow A_1A_2\cdots A_n$ in F^+

- May be more than one key, one call *primary key*
- The keys not chose primary key is called *candidate key*
- Set consists key is called *super key*
- One key show in relation scheme by *an underline*

e.g. $R(ABC)$, $F = \{ A \rightarrow B,\ B \rightarrow C \}$.
Only one key A, since $A \rightarrow ABC$ is in F^+ ($F \models A \rightarrow ABC$),
and *no set* X dose not contain A, *is also* $X \rightarrow ABC$.

$$\mathcal{R}(\underline{A}BC)$$

e.g.

– $R_1(ABCD)$, $F_1 = \{\ AB \rightarrow CD,\ C \rightarrow A,\ D \rightarrow BC\ \}$
R_1 have two keys: AB and D

$R_1(\underline{AB}\ C\ \underline{D})$

– $R_2(ABC)$, $F_2 = \varnothing$
R_2 have one key, it all attributes.

$R_2(\underline{ABC})$

– $R_3(ABC)$, $F_3 = \{\ A \rightarrow B,\ B \rightarrow C,\ C \rightarrow A\ \}$
R_3 have three keys: A, B and C

$R_3(\underline{A}\ \underline{B}\ \underline{C})$

5.2. Algorithm

Input: R , F.
Output: All keys of R.
Method: Computing a sequence steps:

1. Generate all subsets of R, not $\varnothing$ set. *Sorting it by ascending*.
2. Computing closure those sets.
3. Only keep the sets, such that *its closure equal* R.
4. Remove the sets, it *contains any other sets* in list.
5. All sets remain are the all keys of relation scheme R.

e.g.

$R(ABC)$, $F = \{\ A \to B,\ B \to C,\ B \to A\ \}$
Find all keys.

Step 1, Step 2:

$A^+ = ABC, \quad B^+ = ABC, \quad C^+ = C, \quad AB^+ = ABC,$
$AC^+ = ABC, \quad BC^+ = ABC, \quad ABC^+ = ABC$

Step 3:

$A^+ = ABC, \quad B^+ = ABC, \quad AB^+ = ABC$
$AC^+ = ABC, \quad BC^+ = ABC, \quad ABC^+ = ABC$

Step 4:

A

B

Step 5:

$\mathcal{R}(\underline{A}\ \underline{B}\ C)$

5.3. Remarkable

We note some thing about keys algorithm above:

1. Attributes do not in FDs, must in every keys.
2. Attributes do only in left side FDs, then it must in every keys.
3. Attributes do only in right side FDs, then it must do not in any key.
4. Attributes do both in left side and right side FDs, then it either in or not in any key.

5.4. Algorithm

Input: Q , F.
Output: All keys on Q.
Method: Computing a sequence steps:

1. L : attributes in the *left side* of FDs
   R : attributes in the *right side* of FDs
   $NF = Q - (L + R)$: attributes *do not in* FDs
   $OL = L - R$: attributes *only in left side* of FDs
   $LR = L \cap R$: att. *both in the left side and right side* of FDs
2. Generate all subsets of *LR including* $\varnothing$.
   Sorting it by ascending.
3. Add *NF* and *OL* into subsets above.
4. Computing closure for all subsets.
   Decide the keys like general algorithm.

e.g.

$Q(ABCDEG)$, $F = \{\ B \to CD,\ D \to C,\ C \to EG\ \}$.
Find all keys.

Step 1: $L = BCD$, $R = CDEG$
$NF = A$, $OL = B$, $LR = CD$
Step 2: $\emptyset$, C, D, CD
Step 3: AB, CAB, DAB, $CDAB$
Step 4: $AB^+ = ABCDEG$
$\cancel{CAB}$
$\cancel{DAB}$
$\cancel{CDAB}$

One key result: AB

$Q(\underline{AB}CDEG)$

R1(ABCDEG), F1(A->BC,C->EA, B->C)
B1: L = ACB, R = BCEA
NF = DG, OL = 0, LR = ABC
B2: 0, A, B, C, AB, AC, BC, ABC
B3:
DG+ = DG
ADG+ = ADGBCE
BDG+ = BDGCEA
CDG+ = CDGEAB

e.g. $Q(ABCD)$, $F = \{\ B \to CD,\ A \to C\ \}$, find all keys.

e.g.

$Q(ABCD)$, $F = \{\ AB \to CD,\ CD \to AB,\ B \to C\ \}$.
Find all keys.

Step 1: $L = ABCD$, $R = ABCD$, $NF = \varnothing$, $OL = \varnothing$, $LR = ABCD$

Step 2: $\varnothing$, D, BC, ABD, A, AB, BD, ACD
B, AC, CD, BCD, C, AD, ABC, $ABCD$

Step 3: Nothing change

Step 4: $\varnothing$, $AB^+ = ABCD$, $CD^+ = CDAB$, $\cancel{ACD}$,
$A^+ = A$, $AC^+ = AC$, $\cancel{ABC}$, $\cancel{BCD}$,
$B^+ = BC$, $AD^+ = AD$, $\cancel{ABD}$, $\cancel{ABCD}$,
$C^+ = C$, $BC^+ = BC$,
$D^+ = D$, $BD^+ = BDCA$

Three keys result: $(AB), (BD), (CD)$

5.5. FDs logically implied by Keys

R , F,
$K_1, K_2, \cdots K_n$: keys of R, then we have FDs:

$K_1 \to R$, or $K_1 \to (R - K_1)$
...
$K_n \to R$, or $K_n \to (R - K_n)$

e.g.
$R(ABCDE)$, $F = \{ AB \to CD,\ D \to E,\ BC \to A \}$

Two keys: $K_1 = AB$, $K_2 = BC$
Then, there are FDs logically implied by keys:

$AB \to CDE$
$BC \to ADE$

6.1. Normal Form

- Base on Keys checking system of DBMS, we have a concept to check all FDs on relation.
- This concept is call *Normal Form*.

A good Normal form has mainly two purposes:

- Removing duplicated data, redundancy (repetition) data.
- Ensuring data is logically stored.

6.2. First Normal Form (1NF)

- Relation scheme R is in *first normal form (1NF)* if every attribute A in R, $\text{Dom}(A)$ are atomic.
- That is, the values in the domain are not lists or sets of values or composite values.
- Database scheme D is in 1NF if every relation scheme in D is in 1NF.

6.3. e.g.

Relation **Gender**, shown below, is not in 1NF because it contains values that are sets of atomic values.

Gender(Name, Sex)

Gender	(Name	Sex)
	{Tuan, Nam, Cuong}	male
	{Nhung, Hang}	female

e.g.

To be in INF, gender should be stored like this:

Gender(Name, Sex)

Gender	**(Name**	**Sex)**
	Tuan	male
	Nam	male
	Cuong	male
	Nhung	female
	Hang	female

Specially *DateTime* data type is a type with atomic domain.

6.4. e.g.

Assign(Flight, Day, Pilot, Gate)

with $F = \left\{ \; Flight, Day \rightarrow Pilot; \;\; Flight \rightarrow Gate \; \right\}$

Key: $\{Flight, Day\}$

Assign	**(Flight**	**Day**	**Pilot**	**Gate)**
	VN123	20/10/2018	Cuong	7
	VN123	27/10/2018	Nhan	7
	VN456	28/10/2018	Tuan	8

We would like to update the first tuple:

ssign(VN123, 20/10/2018, Cuong; Gate $=$ 9)

e.g.

Assign	(Flight	Day	Pilot	Gate)
	VN123	20/10/2018	Cuong	9
	VN123	27/10/2018	Nhan	7

Assign will violate the FD *Flight* → *Gate*.

To avoid violating the FD, every time an update is made, we have to scan the relation and update the gate number every place the flight number appears.

The flight number and gate number information is duplicated, thus making the *data redundant*.

e.g.

We can represent the same information as a database of two relations, **PilotAssign** and **GateAssign**:

PilotAssign

(Flight	Day	Pilot)
VN123	20/10/2018	Cuong
VN123	27/10/2018	Nhan
VN456	28/10/2018	Tuan

GateAssign

(Flight	Gate)
VN123	7
VN456	8

e.g.

To reconstruct the original relation Assign, taking

$$PilotAssign \bowtie GateAssign$$

To update anomaly in Assign, altering to change in GateAssign.

We have also removed data redundancy of Gate attribute.

6.5. Fully dependent

- Given R , F;
  $X, \ Y \subseteq R$
- Y is called *fully dependent* upon X if:
  1. $X \to Y \in F^+$
  2. $\nexists \, X' \subsetneq X : X' \to Y \in F^+$

e.g. $F = \{ \ Flight, Day \to Pilot, Gate; \ Flight \to Gate \ \}$

$Gate$ is *not fully dependent* upon $\{Flight, Day\}$.

$Pilot$ is *fully dependent* upon $\{Flight, Day\}$.

6.6. Prime and nonprime attributes

- Given R , F
- A is an attribute in R
- A is *prime* in R with respect to F if A is *contained in some key* of R.
- Otherwise A is *nonprime* in R.

e.g. **Assign(Flight, Day, Pilot, Gate)**

with $\mathcal{F} = \{Flight, Day \rightarrow Pilot; Flight \rightarrow Gate\}$

Key: $\{Flight, Day\}$

Prime: $Flight$ and Day.
Nonprime: $Pilot$ and $Gate$.

6.7. Second normal form (2NF)

- R is in *second normal form (2NF)* respect to FDs F
- if it is in 1NF and every nonprime attribute is fully dependent on every key of R.
- A database scheme D is in second normal form if every relation scheme in 2NF.

e.g.

Assign(Flight, Day, Pilot, Gate)

with $F = \{\ Flight, Day \to Pilot;\ \ Flight \to Gate\ \}$

Key: $\{Flight, Day\}$

Assign is not in 2NF, because $Gate$ is not fully dependent on key $\{Flight, Day\}$.

If we let $D = \{\ R_1(Flight, Day, Pilot);\ \ R_2(Flight, Gate)\ \}$,
then both R_1 and R_2 are in 2NF,

so D is in 2NF.

6.8. Transitively Dependent

- R , F
- $X \subseteq R$, $A \in R$
- A : *transitively dependent* upon X in R if there is a $Y \subseteq R$ with:
  - $X \rightarrow Y \in F^+$
  - $Y \nrightarrow X \in F^+$
  - $Y \rightarrow A \in F^+$
  - $A \notin XY$

e.g.

Conference(Room, Day, Reporter, Name)

$F = \{\ Room, Day \to Reporter, Name;\ \ Reporter \to Name\ \}$

- $Room, Day \to Reporter$
- $Reporter \nrightarrow Room, Day$
- $Reporter \to Name$
- $Name \notin \{Room, Day\} \cup \{Reporter\}$

Name is *transitively dependent*

6.9. Third Normal Form (3NF)

Classic Definition

R is in *third normal form (3NF)* if:

- It is in 1NF, and
- *no nonprime attribute* in R is *transitively dependent* upon a key of R.

Database scheme D is in *third normal form* if every relation scheme R in D is in third normal form.

e.g.

Conference(Room, Day, Reporter, Name)

$F = \{\ Room, Day \rightarrow Reporter, Name;\ \ Reporter \rightarrow Name\ \}$

It is not in 3NF because of the transitive dependency of $Name$ upon $\{Room, Day\}$.

If $D = \{\ R_1(Room, Day, Reporter);\ \ R_2(Reporter, Name)\ \}$, then

D is in 3NF because both R_1 and R_2 are in 3NF.

6.10. Third Normal Form (3NF)

Modern Definition

R is in *third normal form (3NF)* if:

- It is in 1NF, and
- Whenever $X \to Y$ is a nontrivial FD in F^+, either
  - X is a superkey, or
  - Each $B \in Y$ is a member of some key of R

Database scheme D is in *third normal form* if every relation scheme R in D is in third normal form.

6.11. Lemma 3NF $\Rightarrow$ 2NF

Any relation scheme R in 3NF is in 2NF.

Proof Using method:

- $(\mathcal{X} \Rightarrow \mathcal{Y}) \Leftrightarrow (\neg\mathcal{Y} \Rightarrow \neg\mathcal{X})$
- $(3NF \Rightarrow 2NF) \Leftrightarrow (\neg 2NF \Rightarrow \neg 3NF)$
- Partial dependency $\Rightarrow$ Transitive dependency.

R is not in 2NF $\Rightarrow$ exists non-prime attribute A is not fully dependent upon key $K \subseteq R$.
So that $\exists\ K' \subsetneq K$:

- $K' \rightarrow A \in F^+$ and because of
- $K' \nrightarrow K \in F^+$
- $A \notin K = K' \cup K$

Therefore A is transitively dependent upon key K, so that R is not in 3NF. ■

6.12. Boyce-Codd Normal Form (BCNF)

Classic Definition

R : *Boyce-Codd normal form (BCNF)* if:

- It is in 1NF, and
- *No attribute* in R is *transitively dependent* upon a key of R.

Database scheme D is in *Boyce-Codd normal form* if every relation scheme R in D is in Boyce-Codd normal form.

- Because of *(attribute in R)* $\supseteq$ *(nonprime attribute in R)*
- So *if R is in BCNF, then in 3NF*.

6.13. Boyce-Codd Normal Form (BCNF)

Modern Definition

R : *Boyce-Codd normal form (BCNF)* if:

- It is in 1NF, and
- for every $Y \subseteq R$ and for every attribute $A \in (R - Y)$
- if $Y \rightarrow A$, then $Y \rightarrow R$
- That is, if Y non-trivially determines any attribute of R, then Y is a superkey for R.

Database scheme D is in *Boyce-Codd normal form* if every relation scheme R in D is in Boyce-Codd normal form.

e.g.

Student(Id, Name, Addr, Hobby)

with $F = \{\ Id \rightarrow Name, Addr\ \}$

Key: $\{Id, Hobby\}$

Student is in 1NF, not in 2NF, 3NF or BCNF.

e.g. **Assign(Manager, Project, Branch)**, with F:

$Manager \rightarrow Branch$: Each manager works in a particular branch.

$Project, Branch \rightarrow Manager$: Each project has several managers, and runs on several branches; however, a project has a unique manager for each branch.

Key: $\{Manager, Project\}, \{Project, Branch\}$

Assign is in 3NF, not in BCNF.

7.1. Preserve Information from a Decomposition ?

SuppInfo(Name, Addr, Item, Price), with

$F = \{ \ Name, Item \rightarrow Price; \ \ Name \rightarrow Addr \ \}$

SuppInfo is in 1NF. We can replace it into:

$\rho = \{ \ Supp(Name, Addr) \ in \ BCNF,$

$\quad SuppDetail(Name, Item, Price) \ in \ BCNF \ \}$

- r is a relation of SuppInfo.
- Instead of store it in SuppInfo, we store it by two relation:
  1. $r(Supp) = \pi_{Name,Addr}(r)$, and
  2. $r(SuppDetail) = \pi_{Name,Item,Price}(r)$

Question: $r(Supp)$ and $r(SuppDetail)$ contain the same data as r.

Only way to recover r is by taking $r(Supp) \bowtie r(SuppDetail)$.

It is right if: $r = r(Supp) \bowtie r(SuppDetail)$

7.2. Preserve Information Decomposition

Decomposition of $\{R,\ F\}$ *is its replacement:*

- $\rho = \{\ R_1, R_2, \cdots, R_k\ \}$
- $R = R_1 \cup R_2 \cup \cdots \cup R_k$
- R_i is no requirement to be disjoin.

This decomposition is *Preserve information* if for every $r(R)$ satisfying F:

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_k}(r)$$

In general:

$$r \subseteq \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_k}(r)$$

7.3. Tableau Test for Preserve Information

Tableau, *Chase* Algorithm.

Input: $R = \{ A^1, A^2, \cdots, A^n \}$, F,
decomposition $\rho = \{ R_1, R_2 \cdots, R_k \}$

Ouput: Whether ρ is *Recovering information*.

Method:

1. Construct a *tableau* T: n columns, k rows.
   - Column A^j corresponds to attribute A^j
   - Row i corresponds to R_i
   - For all (i, j) cell:

     If $A^j \in R_i$:
       $(i, j) \leftarrow 0$
     else
       $(i, j) \leftarrow i$ (the same Row no.)

Tableau Test (cont.)

2. Repeatedly until no more change tableau T:

   For each $X \to Y \in F$

   if $\exists\ (t_1, t_2) \in T\ :\ t_1.X = t_2.X \wedge t_1.Y \neq t_2.Y$

   if $t_1.Y = 0$ make $t_2.Y \leftarrow 0$

   if $t_2.Y = 0$ make $t_1.Y \leftarrow 0$

   if $t_1.Y = t_1,\ t_2.Y = t_2$ make them both t_1 or t_2

3. If exits a row has become all 0 then

   ρ is *Preserve information*.

   If not, ρ is *Not Preserve information*. ■

7.4. e.g.

$R(ABCD)$, $F = \{ A \to B,\ B \to C,\ CD \to A \}$
Decomposed into $\{ R_1(AD),\ R_2(AC),\ R_3(BCD) \}$
Preserve information testing.

The initial tableau:

	A	B	C	D
$R_1(AD)$	0	1	1	0
$R_2(AC)$	0	2	0	2
$R_3(BCD)$	3	0	0	0

Apply $A \to B$ to equate 1, 2. Chosing 1 as the representative symbol (Figure below).

7.4. e.g. (cont.)

	A	B	C	D
$R_1(AD)$	0	1	1	0
$R_2(AC)$	0	1	0	2
$R_3(BCD)$	3	0	0	0

$F = \{\ A \to B,\ B \to C,\ CD \to A\ \}$

Apply $B \to C$ to equate 1, 0. Chosing 0 as the representative symbol (Figure below).

7.4. e.g. (cont.)

	A	B	C	D
$R_1(AD)$	0	1	0	0
$R_2(AC)$	0	1	0	2
$R_3(BCD)$	3	0	0	0

$F = \{\ A \to B,\ \ B \to C,\ \ CD \to A\ \}$

Apply $CD \to A$ for row 1 and row 3, to equate 0, 3. Chosing 0 as the representative symbol (Figure below).

	A	B	C	D
$R_1(AD)$	0	1	0	0
$R_2(AC)$	0	1	0	2
$R_3(BCD)$	0	0	0	0

The third row has all 0. Preserve information decomposition ■

7.5. e.g.

SuppInfo(Name, Addr, Item, Price),

$F = \{ \ Name, Item \rightarrow Price; \ \ Name \rightarrow Addr \ \}$

Decompose:

R_1 = **Supp(Name, Addr)**,
R_2 = **SuppDetail(Name, Item, Price)**

	Name	**Addr**	**Item**	**Price**
R_1	0	0	1	1
R_2	0	2	0	0

7.5. e.g. (cont.)

Since $Name \to Addr$ and two rows agree on $Name$,
we may equate their symbols for $Addr$, making 2 become 0.

The resulting table is:

	Name	Addr	Item	Price
R_1	0	0	1	1
R_2	0	0	0	0

The second row has all 0. Preserve information decomposition ■

7.6. e.g.

$R(ABCDE)$, $F = \{ A \to C,\ B \to C,\ C \to D,\ DE \to C,\ CE \to A \}$,

$R_1(AD),\ R_2(AB),\ R_3(BE),\ R_4(CDE),\ R_5(AE)$

The initial tableau:

	A	**B**	**C**	**D**	**E**
$R_1(AD)$	0	1	1	0	1
$R_2(AB)$	0	0	2	2	2
$R_3(BE)$	3	0	3	3	0
$R_4(CDE)$	4	4	0	0	0
$R_5(AE)$	0	5	5	5	0

Apply $A \to C$ for rows: 1, 2, 5 to equate 1, 2, 5. Choosing 1 as the representative symbol (Figure below).

7.6. e.g. (cont.)

	A	B	C	D	E
$R_1(AD)$	0	1	1	0	1
$R_2(AB)$	0	0	1	2	2
$R_3(BE)$	3	0	3	3	0
$R_4(CDE)$	4	4	0	0	0
$R_5(AE)$	0	5	1	5	0

$F = \{\ A \to C,\ \ B \to C,\ \ C \to D,\ \ DE \to C,\ \ CE \to A\ \}$

Apply $B \to C$ for rows: 2, 3 to equate 1, 3. Choosing 1 as the representative symbol. (Figure below)

7.6. e.g. (cont.)

	A	B	C	D	E
$R_1(AD)$	0	1	1	0	1
$R_2(AB)$	0	0	1	2	2
$R_3(BE)$	3	0	1	3	0
$R_4(CDE)$	4	4	0	0	0
$R_5(AE)$	0	5	1	5	0

$$F = \{\ A \to C,\ \ B \to C,\ \ C \to D,\ \ DE \to C,\ \ CE \to A\ \}$$

Apply $C \to D$ for roes: 1, 2, 3, 5 to equate 0, 2, 3, 5. The result symbol must be 0. (Figure below)

7.6. e.g. (cont.)

	A	B	C	D	E
$R_1(AD)$	0	1	1	0	1
$R_2(AB)$	0	0	1	0	2
$R_3(BE)$	3	0	1	0	0
$R_4(CDE)$	4	4	0	0	0
$R_5(AE)$	0	5	1	0	0

$$F = \{ A \to C,\ B \to C,\ C \to D,\ DE \to C,\ CE \to A \}$$

Apply $DE \to C$ for rows: 3, 4, 5 to equate 1 with 0.
(Figure below)

7.6. e.g. (cont.)

	A	B	C	D	E
$R_1(AD)$	0	1	1	0	1
$R_2(AB)$	0	0	1	0	2
$R_3(BE)$	3	0	0	0	0
$R_4(CDE)$	4	4	0	0	0
$R_5(AE)$	0	5	0	0	0

$F = \{\ A \rightarrow C,\ B \rightarrow C,\ C \rightarrow D,\ DE \rightarrow C,\ CE \rightarrow A\ \}$

Apply $CE \rightarrow A$ for rows: 3, 4, 5 to equate 3, 4, 0 must be with 0.
(Figure below)

7.6. e.g. (cont.)

	A	B	C	D	E
$R_1(AD)$	0	1	1	0	1
$R_2(AB)$	0	0	1	0	2
$R_3(BE)$	0	0	0	0	0
$R_4(CDE)$	0	4	0	0	0
$R_5(AE)$	0	5	0	0	0

7.6. e.g. (cont.)

	A	B	C	D	E
$R_1(AD)$	0	1	1	0	1
$R_2(AB)$	0	0	1	0	2
$R_3(BE)$	0	0	0	0	0
$R_4(CDE)$	0	4	0	0	0
$R_5(AE)$	0	5	0	0	0

The third row has all 0. Preserve information decomposition ∎

7.7. Preserve Information Decomposition Theorem

Lossless Join Decomposition.

R , F;

$\rho = \{ R_1, \ R_2 \}$: Decomposition.

ρ has a *recovered information decomposition* if and only if:

- $(R_1 \cap R_2) \rightarrow (R_1 - R_2) \in F^+$, or
- $(R_1 \cap R_2) \rightarrow (R_2 - R_1) \in F^+$

Note: Two FDs need not be in F; it can be in F^+

Proof: The initial table is shown in below:

	$R_1 \cap R_2$	$R_1 - R_2$	$R_2 - R_1$
row for R_1	$0 \cdots 0$	$0 \cdots 0$	$1 \cdots 1$
row for R_2	$0 \cdots 0$	$2 \cdots 2$	$0 \cdots 0$

7.7. Theorem (cont.)

If $(R_1 \cap R_2) \rightarrow (R_1 - R_2) \in F^+$.
Result has the second row to be all 0.

	$R_1 \cap R_2$	$R_1 - R_2$	$R_2 - R_1$
row for R_1	$0 \cdots 0$	$0 \cdots 0$	$1 \cdots 1$
row for R_2	$0 \cdots 0$	$0 \cdots 0$	$0 \cdots 0$

If $(R_1 \cap R_2) \rightarrow (R_2 - R_1) \in F^+$.
Result has the first row to be all 0.

	$R_1 \cap R_2$	$R_1 - R_2$	$R_2 - R_1$
row for R_1	$0 \cdots 0$	$0 \cdots 0$	$0 \cdots 0$
row for R_2	$0 \cdots 0$	$2 \cdots 2$	$0 \cdots 0$

We has ρ Preserve information decomposition ■

7.8. e.g.

$R(ABC)$, $F = \{ A \rightarrow B \}$

1. R into $R_1(AB)$ and $R_2(AC)$: Preserve information.

Since $R_1 \cap R_2 = A$; $R_1 - R_2 = B$ and $A \rightarrow B$

2. R into $R_1(AB)$ and $R_2(BC)$: Not Preserve information.

Since $R_1 \cap R_2 = B$, and:

- $R_1 - R_2 = A$ and $F \not\models B \rightarrow A$
- $R_2 - R_1 = C$ and $F \not\models B \rightarrow C$

R into $R_1(AB)$ and $R_2(BC)$: Not Preserve information.

Considering $r(R) = \{(1,1,1),(2,1,2)\}$

A	**B**	**C**
1	1	1
2	1	2

$\pi_{AB}(r)$

A	**B**
1	1
2	1

$\pi_{BC}(r)$

B	**C**
1	1
1	2

7.8. e.g. (cont.)

$$\pi_{AB}(r) \bowtie \pi_{BC}(r) = \{\ (1,1,1),\ (1,1,2),\ (2,1,1),\ (2,1,2)\ \}$$

A	B	C
1	1	1
1	1	2
2	1	1
2	1	2

This natural join is proper superset of r.

Preserve Dependencies

R , F.

R decomposed into $\rho = \{ R_1, R_2, \cdots, R_n \}$.

Decomposition ρ preserve F if the union of all the dependencies in $\pi_{R_i}(F)$, for $i = \overline{1..n}$ logically implies all the dependencies in F:

$$\{ \pi_{R_1}(F) \cup \pi_{R_2}(F) \cup \cdots \cup \pi_{R_n}(F) \} \vDash F, \text{ aka}$$

$$\bigcup_{i=1}^{n} \pi_{R_i}(F) \vDash F$$

Note: The converse is always true; F always implies all its projections, and therefore implies their union.

$$F \vDash \bigcup_{i=1}^{n} \pi_{R_i}(F)$$

7.10. e.g.

Store(Category, Warehouse, Product), with

$F = \{\ Product \rightarrow Category;\ \ Category, Warehouse \rightarrow Product\ \}$

Key: $K_1 = \{Category, Warehouse\}$,

$K_2 = \{Warehouse, Product\}$

Decopose **Store** into:

- $R_1(Category, Product)$,

  $F_1 = \{Product \rightarrow Category\}$, $K = \{Product\}$

- $\mathcal{R}_2(Product, Warehouse)$,

  $F_2 = \varnothing$, $K = \{Product, Warehouse\}$

7.10. e.g. (cont.)

This is a reserve information, since: $F \vDash (R_1 \cap R_2) \to (R_1 - R_2)$
But this decomposition does not preserve dependencies, since

$$F_1 \cup F_2 \not\equiv F$$

R_1

(Category	Product)
1	1
1	2

R_2

(Product	Warehouse)
1	1
2	1

However, this join violates FD:
$Category, Warehouse \to Product$

Store

(Category	Warehouse	Product)
1	1	1
1	1	2

7.11. Testing Preserve Dependencies

R , F , $\rho = \{R_1, \cdots, R_n\}$.

Easy to test ρ preserves F by:

1. Computing $F_i = \pi_{R_i}(F^+)$.
2. Test whether: $\bigcup\limits_{i=1}^{n} F_i \models F$?

7.12. e.g.

$R(ABCD)$, $F = \{ A \to B,\ B \to C,\ C \to D,\ D \to A \}$. Decomposition

$\rho = \{ R_1(AB),\ R_2(BC),\ R_3(CD) \}$

Project F^+ onto R_i, we get result:

$F_1 = \{ A \to B,\ B \to A \}$,

$F_2 = \{ B \to C,\ C \to B \}$,

$F_3 = \{ C \to D,\ D \to C \}$

So $\bigcup_{i=1}^{3} F_i \vDash F$

Decomposition Algorithm

Input: R_0 , F_0.
Output: A decomposition of R_0 into a collection of relations, all of which are in BCNF.
Method: $R = R_0$ and $F = F_0$.
Apply recursively $< R, \ F >$.

1. If R is in BCNF, return R. Done.
2. If R violated BCNF by $X \rightarrow Y$ (X is not a super key), decomposition R into $R_1(X_F^+)$ and $R_2(X \cup (R - X_F^+))$.
3. Compute $F_1 = \pi_{R_1}(F^+)$, $F_2 = \pi_{R_2}(F^+)$.
4. Recursively decompose $< \mathcal{R}_1, \ F_1 >$ and $< R_2, \ F_2 >$ using this algorithm.
5. Return the union of the results of these decompositions.

7.14. e.g.

Instock(Product, Category, Supply, Date, Quantity), with

$$F = \{\ Product \to Category;\ \ Category \to Supply;\\ \quad Product, Date \to Quantity\ \}$$

Key $= \{Product, Date\}$ and **Instock** is in 1NF.

1. $R =$ **Instock**
   R violated BCNF by $Product \to Category$
   $Product^+ = \{Product, Category, Supply\}$
   $R_1(Product, Category, Supply)$,
   $F_1 = \{Product \to Category; Category \to Supply\}$
   Key $= \{Product\}$, violated BCNF by $Category \to Supply$

   $R_2(Product \cup (R - Product^+))$, $R_2(Product, Date, Quantity)$
   $F_2 = \{Product, Date \to Quantity\}$
   Key $= \{Product, Date\}$, it is in BCNF.

7.14. e.g. (cont.)

2. $R_1(Product, Category, Supply)$
   $F_1 = \{Product \rightarrow Category;\ Category \rightarrow Supply\}$
   Key $= \{Product\}$, violated BCNF by $Category \rightarrow Supply$
   $Category^+ = \{Category, Supply\}$
   $R_{11}(Category, Supply)$
   $F_{11} = \{Category \rightarrow Supply\}$
   Key $= \{Category\}$, it is in BCNF.

   $R_{12}(Product, Category)$
   $F_{12} = \{Product \rightarrow Category\}$
   Key $= \{Product\}$, it is in BCNF.

7.14. e.g. (cont.)

Renumber relation in BCNF result:

$R_1(Category, Supply)$, in BCNF
$F_1 = \{Category \rightarrow Supply\}$
Key $= \{Category\}$

$R_2(Product, Category)$, in BCNF
$F_2 = \{Product \rightarrow Category\}$
Key $= \{Product\}$

$R_3(Product, Date, Quantity)$, in BCNF
$F_3 = \{Product, Date \rightarrow Quantity\}$
Key $= \{Product, Date\}$

This result can be show by tree below.
With Product = P; Category = C; Supply = S; Date = D; Quantity = Q.

7.14. e.g. (cont.)

- $R(PCSDQ)$; Key $= PD$; 1NF
  $F = \{P \to C;\ C \to S;\ PD \to Q\}$
  $P \to C$
  - $R_1(PCS)$; Key $= P$; 2NF
    $F_1 = \{P \to C;\ C \to S\}$
    $C \to S$
    - $R_{11}(CS)$; Key $= C$; BCNF
      $F_{11} = \{C \to S\}$
    - $R_{12}(PC)$; Key $= P$; BCNF
      $F_{12} = \{P \to C\}$
  - $R_2(PDQ)$; Key $= PD$; BCNF
    $F_2 = \{PD \to Q\}$

Notice: This algorithm decomposition into BCNF, its preserve information but not sure about preserve FDs.

7.15. Preserve Information Decomposition into 3NF

Synthesis Algorithm

It decompose into 3NF Relations With preserve information and dependency.

Input: R , F

Output: Decomposition ρ, each of which is in 3NF. It has preserve information and dependency.

Method: Perform the following steps:

1. Find a minimum cover of F, say F_c.
2. For each $X \rightarrow A \in F_c$, use XA as the scheme of one of the relations in the decomposition.
3. If none of the relation schemes from Step 2 is a superkey of R, add another relation whose scheme is a key of R.

7.16. e.g.

$R(ABCDE)$, $F = \{\ AB \to C,\ C \to B,\ A \to D\ \}$

F is also a minimum cover. So we have:

$R_1(ABC)$, $R_2(BC)$, $R_3(AD)$.

Not to use a relation whose is a proper subset of another relation, so we can drop R_2.

R has two keys: $K_1 = ABE$ and $K_2 = ACE$.
Neither of these keys is a subset of the schemes chosen so far. Thus, we must add one of them, say $R_4(ABE)$.

The final decomposition of R is thus:

$R_1(ABC)$, $R_3(AD)$ and $R_4(ABE)$.

8.1. e.g.

Supply(Name, Item, Delivery)

Supply	**(Name**	**Item**	**Delivery)**
	Amazon	Laptop	Moto
	Amazon	Washer	Moto
	Amazon	Laptop	Pick-Up
	Amazon	Washer	Pick-Up
	McD	Hamb	Drone
	McD	Hamb	Scooter

⋆ (n, i, d) ∈ **Supply**: supply name n supplied item i and can use delivery type t for that item.

⋆ There is no FDs $Name \rightarrow Item$ or $Name \rightarrow Delivery$ on **Supply**, yet **Supply** decomposes losslessly onto:

(Name, Item) and **(Name, Delivery)**.

8.1. e.g. (cont.)

SupItem

(Name	Item)
Amazon	Laptop
Amazon	Washer
McD	Hamb

SupDeli

(Name	Delivery)
Amazon	Moto
Amazon	Pick-Up
McD	Drone
McD	Scooter

SupItem $\bowtie$ SupDeli

(Name	Item	Delivery)
Amazon	Laptop	Moto
Amazon	Laptop	Pick-Up
Amazon	Washer	Moto
Amazon	Washer	Pick-Up
McD	Hamb	Drone
McD	Hamb	Scooter

8.1. e.g. (cont.)

Another instance of the relation **Supply**:

Supply	(Name	Item	Delivery)
	Amazon	Laptop	Moto
	Amazon	Washer	Moto
	Amazon	Washer	Pick-Up
	McD	Hamb	Drone
	McD	Hamb	Scooter

Like above instance, if we decompose this instance onto:
(Name, Item) and **(Name, Delivery)**.

8.1. e.g. (cont.)

And join the two projections, *we do not get back original instance.*

SupItem

(Name	**Item)**
Amazon	Laptop
Amazon	Washer
McD	Hamb

SupDeli

(Name	**Delivery)**
Amazon	Moto
Amazon	Pick-Up
McD	Drone
McD	Scooter

SupItem $\bowtie$ **SupDeli**

(Name	**Item**	**Delivery)**
Amazon	Laptop	Moto
Amazon	Laptop	Pick-Up
Amazon	Washer	Moto
Amazon	Washer	Pick-Up
McD	Hamb	Drone
McD	Hamb	Scooter

8.2. Multivalued Dependencies

Difference of two instance above is in the first instance:
If t_1(n, i, d), t_2(n, i', d'), then t_3(n, i', d).
Definition $R(XYZ)$; X, Y be disjoint subsets of R, and

$$Z = \mathcal{R} - (XY).$$

$r(R)$ satisfies the multivalued dependency (MVD) $X \twoheadrightarrow Y$

if $\forall (t_1, t_2) \in r(R)$: $t_1.X = t_2.X$

$$\begin{aligned} \exists\ t_3 \in r(R):\ & t_3.X = t_1.X = t_2.X, \\ & t_3.Y = t_1.Y, \text{ and} \\ & t_3.Z = t_2.Z. \end{aligned}$$

The symmetry of t_1 and t_2 in this definition implies:

$$\begin{aligned} \exists\ t_4 \in r(R):\ & t_4.X = t_1.X = t_2.X, \\ & t_4.Y = t_2.Y, \text{ and} \\ & t_4.Z = t_1.Z. \end{aligned}$$

Notice: Dislike FD, this definition concern one $r(\mathcal{R})$ not $\forall r(\mathcal{R})$.

Suppose r on $\mathcal{R}(XYZ)$ satisfies the MVD $X \twoheadrightarrow Y$

Representation of the definition below for easy remember:

r ($\mathcal{R}$:	**X**	**Y**	**Z**)
t_1:	1	1	1
t_2:	1	2	2
t_3:	1	1	2
t_4:	1	2	1

r ($\mathcal{R}$:	**X**	**Y**	**Z**)
t_1:	x	y	z
t_2:	x	y'	z'
t_3:	x	y	z'
t_4:	x	y'	z

8.2. Definition (cont.)

Note 1: The $X \twoheadrightarrow Y$ says that the relationship between X and Y is independent of the relationship between X and $(R - Y)$.

Note 2: $X \twoheadrightarrow Y$ holds in R if whenever we have:

- $(t_1, t_2) \in r(R)$: $t_1(X) = t_2(X)$, then
- We can swap two values $t_1(Y)$ *and* $t_2(Y)$ and get new (t_3, t_4) that are also $\in r(R)$.

8.3. Lemma

Relation $r(R)$ satisfies the $X \twoheadrightarrow Y$:

1. $Z = R - XY$, then $r(R)$ satisfies $X \twoheadrightarrow Z$.
2. If X, Y are not disjoin: $Y' = Y - X$, then $X \twoheadrightarrow Y'$
3. $X' \subseteq X$, then $X \twoheadrightarrow X'Y$

Proof

1. Easy to see when we change role of Y and Z
2. $Z = R - (XY) = R - (XY')$
   Since $X \twoheadrightarrow Y$
   $\forall\ (t_1, t_2) \in r(R)$: $t_1.X = t_2.X$
   $\exists t_3$: $t_3.X = t_1.X = t_2.X$,
   $t_3.Y = t_1.Y$ and $t_3.Z = t_2.Z$

   Since $Y' \subseteq Y$, so that $t_3.Y = t_1.Y \Rightarrow t_3.Y' = t_1.Y'$

   Definition implies $r(R)$ satisfies $X \twoheadrightarrow Y'$ ■

8.3. Lemma (cont.)

Relation $r(R)$ satisfies the $X \twoheadrightarrow Y$:

1. $Z = R - XY$, then $r(R)$ satisfies $X \twoheadrightarrow Z$.
2. If X, Y are not disjoin: $Y' = Y - X$, then $X \twoheadrightarrow Y'$
3. $X' \subseteq X$, then $X \twoheadrightarrow X'Y$

Proof

3. $Z = R - (XX'Y) = R - (XY)$
   Since $X \twoheadrightarrow Y$
   $\forall\ (t_1, t_2) \in r(R)$: $t_1.X = t_2.X$
   $\exists t_3$: $t_3.X = t_1.X = t_2.X$,
   $t_3.Y = t_1.Y$ and $t_3.Z = t_2.Z$

   Since $X' \subseteq X$, so that $t_3.X' = t_1.X'$, then $t_3.X'Y = t_1.X'Y$

   Definition implies $r(R)$ satisfies $X \twoheadrightarrow X'Y$ ■

8.4. e.g.

$r(R : ABCD)$ shown below satisfies the $AB \twoheadrightarrow BC$, hence it satisfies the $AB \twoheadrightarrow D$, $AB \twoheadrightarrow C$ and $AB \twoheadrightarrow ABC$

r ($\mathcal{R}$:	**A**	**B**	**C**	**D**)
t_1:	1	1	1	1
t_2:	1	1	2	2
t_3:	1	1	1	2
t_4:	1	1	2	1
t_5:	1	2	2	1
t_6:	2	1	1	2

8.5. Trivial MVDs

Definition

- R and $X, Y \subseteq R$
- MVD $X \twoheadrightarrow Y$ is *trivial* if any relation $r(R)$ satisfies $X \twoheadrightarrow Y$.
- So that
  - $Y \subseteq X$ or
  - $XY = R$

8.6. Theorem

$R, \quad X \subseteq R, \quad Y \subseteq R, \quad Z = R - (XY)$

$r(R)$ satisfies the $X \twoheadrightarrow Y$

$\Leftrightarrow$ $r(R)$ reserve information decomposition into $\begin{cases} r_1 : R_1(XY) \\ r_2 : R_2(XZ) \end{cases}$ **Proof**

$$r(XYZ), \quad r_1 = \pi_{XY}(r), \quad r_2 = \pi_{XZ}(r)$$

$$r: X \twoheadrightarrow Y \Leftrightarrow r = r_1 \bowtie r_2$$

1 "$\Rightarrow$": $r : X \twoheadrightarrow Y \Rightarrow r = r_1 \bowtie r_2$

i.e. $r : X \twoheadrightarrow Y \Rightarrow \begin{cases} r \subseteq r_1 \bowtie r_2, \\ r_1 \bowtie r_2 \subseteq r \end{cases}$

8.6. Theorem (cont.)

1.a $r \subseteq r_1 \bowtie r_2$

i.e. $\forall t \in r \Rightarrow t \in r_1 \bowtie r_2$. This is evident without to MVD:

$\forall t \in r$

$$\left.\begin{array}{l} t_1 = t.XY \Rightarrow t_1 \in \pi_{XY}(r) \\ t_2 = t.XZ \Rightarrow t_2 \in \pi_{XZ}(r) \end{array}\right\} \Rightarrow t_1 \bowtie t_2 \in \pi_{XY}(r) \bowtie \pi_{XZ}(r)$$

Since $t = t_1 \bowtie t_2 \Rightarrow t \in r_1 \bowtie r_2$.

1.b $r_1 \bowtie r_2 \subseteq r$

i.e $\forall t \in r_1 \bowtie r_2 \Rightarrow t \in r$.

$$\forall t \in r_1 \bowtie r_2 \Rightarrow \left\{\begin{array}{l} \exists t_1 \in r_1, \\ \exists t_2 \in r_2 \end{array}\right. : t = t_1 \bowtie t_2$$

$$\text{i.e. } \left\{\begin{array}{l} t.XY = t_1, \\ t.XZ = t_2, \\ t_1.X = t_2.X \end{array}\right.$$

8.6. Theorem (cont.)

So we have:

$$\begin{cases} t.X = t_1.X = t_2.X, & (1) \\ t.Y = t_1.Y, & (2) \\ t.Z = t_2.Z & (3) \end{cases}$$

$$t_1 \in \pi_{XY}(r) \Rightarrow \exists t_1' \in r : t_1'.XY = t_1 \qquad (4)$$

$$t_2 \in \pi_{XZ}(r) \Rightarrow \exists t_2' \in r : t_2'.XZ = t_2 \qquad (5)$$

$$(4,1,5) \Rightarrow t_1'.X = t_1.X = t_2.X = t_2'.X$$

$$\Rightarrow t_1'.X = t_2'.X$$

$$\left.\begin{array}{r} (4) \Rightarrow t_1'.Y = t_1.Y \\ (2) \end{array}\right\} \Rightarrow t.Y = t_1'.Y$$

8.6. Theorem (cont.)

$$\left.\begin{array}{r} (5) \Rightarrow t'_2.Z = t_2.Z \\ (3) \end{array}\right\} \Rightarrow t.Z = t'_2.Z$$

We have:

$$\begin{cases} \exists (t'_1, t'_2) \in r,\ \ t'_1.X = t'_2.X \\ \exists t \in r,\ \ t.X = t'_1.X = t'_2.X \\ \qquad\qquad t.Y = t'_1.Y \\ \qquad\qquad t.Z = t'_2.Z \end{cases}$$

This is the definition of MVD $X \twoheadrightarrow Y$ holds on $r : R(XYZ)$.
t satisfied all requirements above so that $t \in r$.

8.6. Theorem (cont.)

2 "$\Leftarrow$": $r = r_1 \bowtie r_2 \Rightarrow r : X \twoheadrightarrow Y$

$\forall (t_1, t_2) \in r : t_1.X = t_2.X$

$\exists t_1' \in (r_1 = \pi_{XY}(r)) : t_1' = t_1.XY \qquad (1)$

$\exists t_2' \in (r_2 = \pi_{XZ}(r)) : t_2' = t_2.XZ \qquad (2)$

Since $r = r_1 \bowtie r_2$, so:

$\exists t \in r : t = t_1' \bowtie t_2'$
So that,

since (1): $t.XY = t_1' = t_1.XY \qquad (3)$

since (2): $t.XZ = t_2' = t_2.XZ \qquad (4)$

$(3) \Rightarrow t.X = t_1.X, t.Y = t_1.Y \qquad (5)$

$(4) \Rightarrow t.X = t_2.X, t.Z = t_2.Z \qquad (6)$

8.6. Theorem (cont.)

All above we have:

$$\begin{cases} \forall (t_1, t_2) \in r : t_1.X = t_2.X \\ \exists t : t.X = t_1.X = t_2.X \\ (5) \Rightarrow t.Y = t_1.Y \\ (6) \Rightarrow t.Z = t_2.Z \end{cases}$$

Follow definition r satisfied $X \twoheadrightarrow Y$ on $r : R(XYZ)$ ■

8.7. Test MVDs

$r : R(XYZ)$:

1. Theorem 8.6 gives us a method to test if a relation $r(R)$ satisfies $X \twoheadrightarrow Y$:

1. $r_1 = \pi_{XY}(r), \quad r_2 = \pi_{XZ}(r)$
2. Test if $r = r_1 \bowtie r_2$?

2. Another method: Use *only some sorting and counting.*

1. Given X-value x:
2. n_1: tuples in r_1 with X-value x
3. n_2: tuples in r_2 with X-value x
4. n: tuples in r with X-value x
5. If $n = n_1 \times n_2$ for any X-value x, then $r = r_1 \bowtie r_2$

8.7. Test MVDs (cont.)

Define the function counts the number of different W-values associated with a given X-value in r.

$$\mathcal{C}_W[X = x](r) = |\pi_W(\sigma_{X=x}(r))|$$

The condition for $X \twoheadrightarrow Y$ can be stated as:
For any X-value x in r:

$$\mathcal{C}_{\mathcal{R}}[X = x](r) \quad = \quad \mathcal{C}_{XY}[X = x](r) \quad \times \quad \mathcal{C}_{XZ}[X = x](r)$$

Since $\mathcal{C}_{WX}[X = x](r) \ = \ \mathcal{C}_W[X = x](r)$, we can simplify to:

$$\mathcal{C}_{\mathcal{R}}[X = x](r) \quad = \quad \mathcal{C}_{Y}[X = x](r) \quad \times \quad \mathcal{C}_{Z}[X = x](r)$$

8.8. e.g.

$R(ABCD)$, Test $AB \twoheadrightarrow C$

r ($\mathcal{R}$:	**A**	**B**	**C**	**D**)
t_1:	1	1	1	1
t_2:	1	1	2	2
t_3:	1	1	1	2
t_4:	1	1	2	1
t_5:	1	2	2	1
t_6:	2	1	1	2

$\mathcal{C}_{ABCD}[AB = 11](r) = 4$; $\mathcal{C}_C[AB = 11](r) = 2$; $\mathcal{C}_D[AB = 11](r) = 2$
$\mathcal{C}_{ABCD}[AB = 12](r) = 1$; $\mathcal{C}_C[AB = 12](r) = 1$; $\mathcal{C}_D[AB = 12](r) = 1$
$\mathcal{C}_{ABCD}[AB = 21](r) = 1$; $\mathcal{C}_C[AB = 21](r) = 1$; $\mathcal{C}_D[AB = 21](r) = 1$

9.1. Inference Axioms MVDs Alone

$r(\mathcal{R})$; W, X, Y, Z be subsets of R.

M1. Reflexivity

$$Y \subseteq X \models X \twoheadrightarrow Y$$

M2. Augmentation

$$\left.\begin{array}{r} X \twoheadrightarrow Y \\ W \subseteq Z \end{array}\right\} \models XZ \twoheadrightarrow YW$$

M3. Transitivity

$$\left.\begin{array}{r} X \twoheadrightarrow Y \\ Y \twoheadrightarrow Z \end{array}\right\} \models X \twoheadrightarrow Z - Y$$

M4. Additivity

$$\left.\begin{array}{r} X \twoheadrightarrow Y \\ X \twoheadrightarrow Z \end{array}\right\} \models X \twoheadrightarrow YZ$$

9.1. Inference Axioms MVDs Alone (cont.)

M5. Decomposition

$$\left.\begin{array}{r} X \twoheadrightarrow Y \\ X \twoheadrightarrow Z \end{array}\right\} \vDash \left\{\begin{array}{l} X \twoheadrightarrow Y \cap Z \\ X \twoheadrightarrow Y - Z \\ X \twoheadrightarrow Z - Y \end{array}\right.$$

M6. Pseudo Transitivity

$$\left.\begin{array}{r} X \twoheadrightarrow Y \\ WY \twoheadrightarrow Z \end{array}\right\} \vDash XW \twoheadrightarrow Z - (YW)$$

M7. Complementation

$$\left.\begin{array}{r} X \twoheadrightarrow Y \\ Z = \mathcal{R} - (XY) \end{array}\right\} \vDash X \twoheadrightarrow Z$$

9.2. Inference Axioms FDs and MVDs

$r(\mathcal{R})$; W, X, Y, Z be subsets of R.

C1. Replication

$$X \rightarrow Y \models X \twoheadrightarrow Y$$

C2. Coalescence

$$\left.\begin{array}{r} X \twoheadrightarrow Y \\ \exists W,\ W \cap Y = \varnothing \\ W \rightarrow Z \\ Z \subseteq Y \end{array}\right\} \models X \rightarrow Z$$

9.3. e.g.

$R(ABCDE)$, $D = \{ A \to BC,\ DE \twoheadrightarrow C \}$

$D \models A \twoheadrightarrow BDE$?

$A \to BC \models A \twoheadrightarrow BC$ (C1)

$A \twoheadrightarrow BC \models A \twoheadrightarrow DE$ (M7)

$$\left.\begin{array}{l} A \twoheadrightarrow DE \\ DE \twoheadrightarrow C \end{array}\right\} \models A \twoheadrightarrow C \text{ (M3)}$$

$A \twoheadrightarrow C \models A \twoheadrightarrow BDE$ (M7) ■

9.4. Finding X_F^+ by Chase

R , F , $X \subseteq R$;
Compute X_F^+ all attributes that functionally depend on X.

1. Start with a tableau having two rows that agree only on X.
2. Chase the tableau using the FDs of F.
3. The columns in final tableau agrees two rows is X_F^+.

e.g.
$R(ABCDEH)$, $F = \{\ AB \to C,\ BC \to AD,\ D \to E,\ CH \to B\ \}$

$F \models AB \to D$?
Start with the tableau:

A	B	C	D	E	H
0	0	1	1	1	1
0	0	2	2	2	2

9.4. Finding X_F^+ by Chase (cont.)

Apply $AB \rightarrow C$ to infer $1 = 2$; say both become 1. The resulting tableau is:

A	B	C	D	E	H
0	0	1	1	1	1
0	0	1	2	2	2

Next, apply $BC \rightarrow AD$ to infer that $1 = 2$, and apply $D \rightarrow E$ to infer $1 = 2$. At this point, the tableau is:

A	B	C	D	E	H
0	0	1	1	1	1
0	0	1	1	1	2

Since the two tuples agree in the D column implies $F \vDash AB \rightarrow D$.

Other hand: $AB_F^+ = (ABCDE)$.

In this tableau, only one AB_F^+ not other attributes.

9.5. Tableau Chase Test for MVDs

$X \twoheadrightarrow Y$ holds in $r : \mathcal{R}(XYZ)$ if only if:

$$r(XYZ) = r_1(XY) \bowtie r_2(XZ)$$

So we can use Tableau to check $r_1(XY)$ and $r_2(XZ)$ reserve information with $r(XYZ)$ whether implies $X \twoheadrightarrow Y$.

e.g.

$R(ABCD)$, $\mathfrak{D} = \{ \ A \rightarrow B, \ B \twoheadrightarrow C \ \}$

$\mathfrak{D} \models A \twoheadrightarrow C$ holds in $r(R)$?

Start with tableau:

R	**A**	**B**	**C**	**D**
$r_1(AC)$	0	1	0	1
$r_2(ABD)$	0	0	2	0

9.5. Tableau Chase Test for MVDs (cont.)

Apply $A \rightarrow B$ to infer that $1 = 0$.
The tableau becomes:

R	**A**	**B**	**C**	**D**
$r_1(AC)$	0	0	0	1
$r_2(ABD)$	0	0	2	0

Apply the $B \twoheadrightarrow C$, since two rows agree in the B column. We swap C columns to get two more rows which we add to the tableau, which becomes:

R	**A**	**B**	**C**	**D**
$r_1(AC)$	0	0	0	1
$r_2(ABD)$	0	0	2	0
	0	0	2	1
	0	0	0	0

A row with all 0, so that $F \models A \twoheadrightarrow C$ ∎

9.6. e.g.

$$R(ABCGHI)\ ,\ \mathfrak{D} = \{\ A \twoheadrightarrow B,\ \ B \twoheadrightarrow HI,\ \ CG \twoheadrightarrow H\ \}$$
$$\mathfrak{D} \models AB \twoheadrightarrow BH$$

Using Tableau Chase Test:

R	**A**	**B**	**C**	**G**	**H**	**I**	
$r_1(AGBH)$	0	0	1	0	0	1	t_1
$r_2(AGCI)$	0	2	0	0	2	0	t_2
$A \twoheadrightarrow B$	0	0	0	0	2	0	t_3
using (t_1, t_2)	0	2	0	0	2	0	t_4
$B \twoheadrightarrow HI$	0	0	1	0	2	0	t_5
using (t_1, t_3)	0	0	0	0	0	1	t_6
$CG \twoheadrightarrow H$	0	0	0	0	0	0	t_7
using (t_3, t_6)	0	0	0	0	2	1	t_8

There is a row with all 0, so that $\mathfrak{D} \models AB \twoheadrightarrow BH$ ■

9.7. Projecting MVDs

$R(ABCDE)$, $\mathfrak{D} = \{\ A \twoheadrightarrow CD\ \}$;
Decomposition $S(ABC)$. Finding MVDs imply in $S(ABC)$?

MVDs maybe have left side are A or B or C, since orther trivial MVDs.
Furthermore, we claim $A \twoheadrightarrow C$ holds in $\mathcal{S}$, as does $A \twoheadrightarrow B$ (by the complementation rule).

Start with the tableau:

R	**A**	**B**	**C**	**D**	**E**
$r_1(AC)$	0	1	0	1	1
$r_2(ABDE)$	0	0	2	0	0

9.7. Projecting MVDs (cont.)

Use $A \twoheadrightarrow CD$ to swap the C and D components of these two rows to get two new rows:

R	**A**	**B**	**C**	**D**	**E**
$r_1(AC)$	0	1	0	1	1
$r_2(ABDE)$	0	0	2	0	0
	0	1	2	0	1
	0	0	0	1	0

Notice: The last row has 0 in all the attributes of S, that is, A, B, and C. That is enough to conclude that $A \twoheadrightarrow C$ holds in S ■

9.8. Minimum Dependency Basis

Algorithm 1 David Maier (The Theory of Relational Databases)

Input: R, $\mathfrak{D} = \left\{\ FDs,\ \ MVDs\ \right\}$, $X \subseteq R$

Output: Minimum dependency basis for X base on $\mathfrak{D}$

Denote: $X^{++}_{\mathfrak{D}}$

Method:

1. – Change all FDs in $\mathfrak{D}$ to MVDs

   – $X^{++}_{\mathfrak{D}} \leftarrow \left\{A_1,\ A_2, \cdots\ A_k\right\}$, $A_i \in X$

   – $\forall X' \twoheadrightarrow Y' \in \mathfrak{D}\ :\ X' \subseteq X$

   $$X^{++}_{\mathfrak{D}} \leftarrow X^{++}_{\mathfrak{D}}\ +\ \{Y'\}\ +\ \{R - X'Y'\}$$

2. $\forall\{Y_1, Y_2\} \in X^{++}_{\mathfrak{D}} : Y_1 \cap Y_2 \neq \varnothing :$

   $$X^{++}_{\mathfrak{D}} \leftarrow X^{++}_{\mathfrak{D}}\ -\ \{Y_1, Y_2\}\ +\ \{Y_1 \cap Y_2\}\ +\ \{Y_1 - Y_2\}\ +\ \{Y_2 - Y_1\}$$

9.8. Minimum Dependency Basis (cont.)

3. $\forall W \twoheadrightarrow Z \in \mathfrak{D} \; : \; W \subseteq Y = Y_1 Y_2 \cdots Y_k \; , \; Y_i \in X_{\mathfrak{D}}^{++}$
   - $\mathfrak{D} \models Y \twoheadrightarrow Z$ (augmentation)
   - $\mathfrak{D} \models X \twoheadrightarrow \left\{Z - Y\right\}$ (transitivity)
   - If $\left\{Z - Y\right\}$ is NOT the union of some sets in $X_{\mathfrak{D}}^{++}$:

     $$X_{\mathfrak{D}}^{++} \leftarrow X_{\mathfrak{D}}^{++} + \left\{Z - Y\right\}$$
4. If no MVD in $\mathfrak{D}$ can be used to change $X_{\mathfrak{D}}^{++}$, stop.
   If not, turn to step 2

Like X_F^+ for FDs, with $X_{\mathfrak{D}}^{++}$ we can implies all MVDs with which left side is X and right side is subset of $X_{\mathfrak{D}}^{++}$

9.9. e.g.

$R(ABCDE)$, $\mathfrak{D} = \{A \to BC,\ DE \twoheadrightarrow C\}$

Find $A^{++}_{\mathfrak{D}}$.

1. – Change to MVDs: $\mathfrak{D} = \{A \twoheadrightarrow BC,\ DE \twoheadrightarrow C\}$
   – Initial $A^{++}_{\mathfrak{D}} = \{A\}$
   – $A \twoheadrightarrow BC$:
   $$A^{++}_{\mathfrak{D}} = \{A,\ BC,\ DE\}$$
2. None
3. $DE \twoheadrightarrow C$ with $Y = DE$:
   $$A^{++}_{\mathfrak{D}} = \{A,\ BC,\ DE,\ C\}$$
2. $Y_1 = BC,\ Y_2 = C$:
   $$A^{++}_{\mathfrak{D}} = \{A,\ B,\ C,\ DE\}$$
3. None 2. None 3. None

Result: $A^{++}_{\mathfrak{D}} = \{A,\ B,\ C,\ DE\}$

9.10. e.g.

$R(ABCDE)$, $\mathfrak{D} = \{A \to BC,\ DE \twoheadrightarrow C\}$

Find $AD^{++}_{\mathfrak{D}}$.

1. – Change to MVDs: $\mathfrak{D} = \{A \twoheadrightarrow BC,\ DE \twoheadrightarrow C\}$
   – Initial $AD^{++}_{\mathfrak{D}} = \{A,\ D\}$
   – $A \twoheadrightarrow BC$:
   $$AD^{++}_{\mathfrak{D}} = \{A,\ D,\ BC,\ DE\}$$
2. $Y_1 = D,\ \ Y_2 = DE$:
   $$AD^{++}_{\mathfrak{D}} = \{A,\ D,\ E,\ BC\}$$
3. $DE \twoheadrightarrow C$ with $Y = DE$:
   $$AD^{++}_{\mathfrak{D}} = \{A,\ D,\ E, BC,\ C\}$$
2. $Y_1 = BC,\ \ Y_2 = C$:
   $$AD^{++}_{\mathfrak{D}} = \{A,\ D,\ E,\ B,\ C\}$$

3. None
2. None
3. None

Result: $AD_{\mathfrak{D}}^{++} = \{A, \ B, \ C, \ D, \ E\}$

So that: $\mathfrak{D} \vDash AD \twoheadrightarrow BE$

We can test this MVD by use the *Axioms or Tableau and Chase.*

9.11. Another Algorithm

Minimum Dependency Basis, Beeri [1980]

Input: R, $\mathfrak{D} = \{\ FDs,\ \ MVDs\ \}$, $X \subseteq R$
Output: Minimum dependency basis for X base on $\mathfrak{D}$
Denote: $X^{++}_{\mathfrak{D}}$
Method:

1. Change all FDs in $\mathfrak{D}$ to MVDs

   $X^{++}_{\mathfrak{D}} \leftarrow \{\ R - X\ \}$
2. While (No more change $X^{++}_{\mathfrak{D}}$) Do

   For each $V \twoheadrightarrow W \in \mathfrak{D}$

   If ($\exists Y \in X^{++}_{\mathfrak{D}}$: $Y \cap W \neq \varnothing$, $Y \cap V = \varnothing$) then

   $X^{++}_{\mathfrak{D}} \leftarrow X^{++}_{\mathfrak{D}}\ -\ \{Y\}\ +\ \{Y \cap W\}\ +\ \{Y - W\}$
3. $X^{++}_{\mathfrak{D}}$ is dependency basis of X

10.1. Fourth Normal Form

Definition

R , $\mathfrak{D} = \{FDs,\ MVDs\}$

R is in fourth normal form (4NF) if for every $X \twoheadrightarrow Y$:

- $X \twoheadrightarrow Y$ is a *trivial* MVD or
- X is a *superkey* for R

e.g.

$R(ABCDE)$, $\mathfrak{D} = \{A \rightarrow BC,\ C \twoheadrightarrow DE\}$.

R is not in 4NF because of the $C \twoheadrightarrow DE$.

R consisting of the two $R_1(ABC)$ and $R_2(CDE)$ is in 4NF with respect to $\mathfrak{D}$,

even though $A \twoheadrightarrow B$ is implied by $\mathfrak{D}$ and applies to R_1.

$A \twoheadrightarrow B$ is not trivial, but A is a key for R_1.

10.2. Lemma

R , $\mathfrak{D} = \{FDs,\ MVD\}$. If R is in 4NF, then R is in BCNF.

Proof (use $\mathscr{C}_1 \Rightarrow \mathscr{C}_2 \Leftrightarrow \neg\mathscr{C}_2 \Rightarrow \neg\mathscr{C}_1$)
Suppose R is not in BCNF. Then we must have subsets K, Y, and A of R such that K is a key for $\mathcal{R}$ and:

1. $K \to Y$,
2. $Y \nrightarrow K$,
3. $Y \to A$,
4. $A \notin (KY)$

We try to prove $Y \twoheadrightarrow A$ is not trivial and Y is not a superkey.

– $Y \to A \vDash Y \twoheadrightarrow A$
  Since $A \notin Y$ and $YA \neq \mathcal{R}$, so $Y \twoheadrightarrow A$ is not trivial.
– $Y \nrightarrow K$, so Y is not a key $\Rightarrow$ Y not a superkey

Therefore, R is not in 4NF ■

10.3. Decomposition into 4NF

Quite analogous to the BCNF decomposition algorithm.

Algorithm Decomposition into 4NF.

Input: R_0 with $\mathfrak{D}_0 = \{FDs,\ MVDs\}$
Output: $\rho = \{R_1, R_2, \ldots, R_k\}$ in 4NF, conserve information
Method: Do the following steps, with $R = R_0$ and $\mathfrak{D} = \mathfrak{D}_0$

1. Find a 4NF violation in R; $X \twoheadrightarrow Y$ in R, X is not a superkey.
   If none, return True; R by itself is a suitable decomposition.
2. If have a 4NF violation, break R into two:
   - $R_1(XY)$
   - $R_2(X(R - XY))$

10.3. Decomposition into 4NF (cont.)

3. Find the FD's and MVD's: $\mathfrak{D}_1 = \pi_{R_1}(\mathfrak{D})$ and $\mathfrak{D}_2 = \pi_{R_2}(\mathfrak{D})$.
4. Recursively decompose $\{R_1, \mathfrak{D}_1\}$ and $\{R_2, \mathfrak{D}_2\}$

e.g.

$R(ABCDEI)$, $\mathfrak{D} = \{A \twoheadrightarrow BCD,\ B \rightarrow AC,\ C \rightarrow D\}$.

- Key $=$ (BEI). R is not in 4NF because of $A \twoheadrightarrow BCD$ is a nontrival MVD and A is not a key for R
- Decompose R into:
  $R_1(ABCD)$, $\mathfrak{D}_1 = \{B \rightarrow AC,\ C \rightarrow D\}$, key $=$ B, 2NF
  $R_2(AEI)$, $\mathfrak{D}_2 = \varnothing$, 4NF
- Use $C \twoheadrightarrow D$ for $R_1(ABCD)$:
  $R_{11}(CD)$, $\mathfrak{D}_{11} = \{C \rightarrow D\}$, key $=$ C, 4NF
  $R_{12}(ABC)$, $\mathfrak{D}_{12} = \{B \rightarrow AC\}$, key $=$ B, 4NF

Result: $\rho = \{\mathcal{R}_{11}, \mathcal{R}_{12}, \mathcal{R}_2\}$ is thus in 4NF rescpect to $\mathfrak{D}$.

11.1. e.g.

For example, we have $r(ABC)$ decomposes conserve information onto AB, AC and BC.

r	(A	B	C)
	1	1	1
	1	2	2
	3	3	3
	4	3	4
	5	5	5
	6	6	5

$$r = \pi_{AB}(r) \bowtie \pi_{AC}(r) \bowtie \pi_{BC}(r)$$

11.1. e.g. (cont.)

$\pi_{AB}(r) =$

(A	B)
1	1
1	2
3	3
4	3
5	5
6	6

$\pi_{AC}(r) =$

(A	C)
1	1
1	2
3	3
4	4
5	5
6	5

$\pi_{BC}(r) =$

(B	C)
1	1
2	2
3	3
3	4
5	5
6	5

However, r satisfies no nontrivial MVDs, so it has no conserve information decomposition onto any pair of R_1 and R_2 such that $R_1 \neq ABC$ and $R_1 \neq ABC$.

11.2. Join Dependency

$R = \{R_1, R_2, ..., R_p\}$ be a set of relation schemes over $U = R_1 R_2 ... R_p$. A relation $r(U)$ satisfies

Join Dependency: $(JD)\ \ * [R_1, R_2, ..., R_p]$

if r decomposes conserve information onto $R_1, R_2, \cdots, R_p$. That is,

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_p}(r)$$

We also write $*[R_1, R_2, \cdots, R_p]$ as $*[R]$

e.g.

In example 11.1, r satisfies the JD $*[AB,\ AC,\ BC]$

11.2. Join Dependency (cont.)

We know:

1. $r(R)$ satisfies $X \twoheadrightarrow Y \Leftrightarrow r$ decomposes conserve information onto XY and XZ, where $Z = R - (XY)$.
2. This condition is just JD $*[XY, XZ]$.
3. JD $*[R_1, R_2]$ is the same as the $R_1 \cap R_2 \twoheadrightarrow R_1$.
4. We can also define JDs in a manner similar to the definition of MVDs.
5. Let r satisfy $*[R_1, R_2, \cdots, R_p]$. If r contains tuples $t_1, t_2, \cdots, t_p$ such that
$$t_i(R_i \cap R_j) = t_j(R_i \cap R_j)$$
for all i, j,
r must contain t such that $t(R_i) = t_i(R_i)$, $1 \leqslant i \leqslant p$.

11.3. e.g.

Suppose relation $r(ABCDE)$ satisfies the JD $*[ABC,\ BD,\ CDE]$ and contains the three tuples shown below.

r	**(A**	**B**	**C**	**D**	**E)**
t_1	1	1	1	1	1
t_2	2	1	2	2	3
t_3	3	2	1	2	2

r must also contain tuple $t = (1,\ 1,\ 1,\ 2,\ 2)$

JD $*[R_1, R_2, \cdots, R_p]$ over R is *trivial* if $R = R_1$ for some i.

JD $*[R_1, R_2, \cdots, R_p]$ *applies* to R if $R = R_1 R_2 \cdots R_p$.

11.4. Project Join Normal Form

R, $F = \{FDs,\ JDs\}$ over R.
R is in *project join normal form* (PJNF) with respect to F
if for every JD $*[R_1, R_2, \cdots, R_p]$ implied by F, that implies to R, and:

- JD is trivial or,
- Every R_i is a superkey for R.

e.g. Let $R(ABCDEI)$,
$F = \{*[ABCD, CDE, BDI], *[AB, BCD, AD], A \rightarrow BCDE, BC \rightarrow AI\}$.

Key $= (A)$, (BC)
R is not PJNF because of JD $*[ABCD, CDE, BDI]$.
If decompose $\rho = \{R_1(ABCD), R_2(CDE), R_3(BDI)\}$

11.4. Project Join Normal Form (cont.)

$\rho = \{R_1(ABCD), R_2(CDE), R_3(BDI)\}$

$R_1(ABCD)$, $F_1 = \{*[AB, BCD, AD],\ A \to BCD,\ BC \to A\}$

Key $=$ (A), BC. So R_1 is in PJNF.

$R_2(CDE)$, $\mathcal{F}_2 = \varnothing$, PJNF.

$R_3(BDI)$, $F_3 = \varnothing$, PJNF.

12.1. Embedded Functional Dependencies

Given R, $X \rightarrow Y$ holds on R.
If any $S \supseteq R$. Whether $X \rightarrow Y$ holds on S ?

This is evident for FDs because the definition of FDs did not concert to other XY set.

r (R: **X**	**Y**)
1	2
1	2
3	4
3	4

r (S: **X**	**Y**	**Z**	**W**)
1	2	☺	☹
1	2	🚹	🚺
3	4		
3	4	☯	☹

12.2. Embedded Multivalued Dependencies

Consider the relation $r(ABCD)$. The projection $\pi_{ABC}(r)$ satisfies $A \twoheadrightarrow B$, but r itself does not.

r	**(A**	**B**	**C)**
	1	1	1
	1	2	1
	1	1	2
	1	2	2
	2	2	2

r	**(A**	**B**	**C**	**D)**
	1	1	1	1
	1	2	1	1
	1	1	2	2
	1	2	2	1
	2	2	2	2

Definition: Relation $r(R)$ satisfies the embedded multivalued dependency $X \twoheadrightarrow Y|Z$ if the $X \twoheadrightarrow Y$ is satisfied by the $\pi_{X\cup Y\cup Z}(r)$.

Read: 'X multivalued determines Y in the context of Z'.

12.3. Embedded Join Dependencies

Multivalued dependencies is spectial of Join dependencies.
$r(R)$, $X \twoheadrightarrow Y$.
Another description is JD $*[XY, XZ]$ with $Z = R - XY$.

Definition

- Relation $r(R)$ satisfies the embedded join dependency (EJD) $*[R_1, R_2, ..., R_p]$ if: $\pi_S(r)$ satisfies $*[R_1, R_2, ..., R_p]$ as a regular JD, where $S = R_1 R_2 ... R_p$.
- Allow $R = S$. That is, every JD is an EJD.
- Also write the embedded multivalued dependency (EMVD) $*[XY, XZ]$ as $X \twoheadrightarrow Y|Z$.

References

- Maier, D. (1983). *The Theory of Relational Database.*. Rockville, USA: Computer Science Press.
- Ulman, J. D. (1988). *Principles of Database and Knowledge - Base Systems.* (Vol. I). Rockville, USA: Computer Science Press.