Design Theory for Relational Databases

Spring 2011

Instructor: Hassan Khosravi

Chapter 3: Design Theory for Relational Database

Section 3.1

FUNCTIONAL DEPENDENCIES

Definition of Functional Dependency

Keys of Relations

Superkeys

Definition of Functional Dependency

- Given a relation R, attribute Y of R is functionally dependent on attribute X of R if each X - value in R has associated with precisely one Y - value in R at any time.
- No X-values are mapped to two or more Y-values
- We denote it as: X → Y

Definition of Functional Dependency

- X and Y can be a set of attributes.
- So, if X = {A,B,C} and Y = {D,E} then,
  ABC → DE
- The above functional dependency is equivalent to:
  ABC → D and
  ABC → E

Definition of Functional Dependency

Example 3.1

Consider the following relation:

Title	*Year*	*Length*	*Genre*	*StudioName*	*StarName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Title Year → Length Genre StudioName

But the FD Title Year → StarName doesn't hold.

When we say R satisfies a FD, we are asserting a constraint on all possible Rs not just an instance of R.

Definition of Functional Dependency

- An attribute Y is said to be Fully Functionally dependent (not Partially Dependent) on X if Y functionally depends on X but not on any proper subset of X.
- From now on, if we mean full FD, then we denote it by FFD.
- A functional dependency is a special form of integrity constraint.
- In other words, every legal extension (tabulation) of that relation must satisfies that constraint.

Example

Consider a relation R(A,B,C,D,E) with functional dependencies:

$A,B \rightarrow C$

$C,D \rightarrow E.$

Suppose there are at most 3 different values for each of A, B, and D. What is the maximum number of different values for E?

- ○ 27
- ○ 9
- ○ 3
- ○ 81

Skip

Submit

Example

Consider a relation R(A,B,C,D,E) with functional dependencies:

A,B → C
C,D → E.

Suppose there are at most 3 different values for each of A, B, and D. What is the maximum number of different values for E?

- (•) **27**
- ( ) **9**
- ( ) **3**
- ( ) **81**

Show Explanation

Continue

Correct

Functional Dependencies

- Data Storage – Compression
- Reasoning about queries – Optimization
- Good exam questions ☺

- Student (Ssn, Sname, address, hscode, hsname, hscity, gpa, priority)
- Apply(sid, cname, state, date major)

- Priority is determined by GPA
  - Gpa > 3.8 → priority=1
  - 3.3 < Gpa < 3.8 → priority=2
  - Gpa < 3.3 → priority=3

  - Two tuples with the same gpa have the same priority.
    - t.gpa = u.gpa → t.priority = u.priority
    - gpa → priority

- Student (Ssn, Sname, address, hscode, hsname, hscity, gpa, priority)
  - ssn → sname
  - ssn → address
  - hscode → hsname, hscity
  - hsname, hscity → hscode
  - Ssn → gpa
  - gpa → priority
  - ssn → priority

- Apply(sid, cname, state, date major)
  - Colleges receive applications only on a specific date
    - cname → date
  - Students can only apply to only one major in each university
    - ssn,cname → major
  - Students can only apply to colleges in one state
    - Ssn → state

Example

For the relation Apply(SSN,cName,state,date,major), what real-world constraint is captured by SSN,cName → date?

- ○ A student can only apply to one college.
- ○ A student can apply to each college only once.
- ○ A student must apply to all colleges on the same date.
- ○ Every application from a student to a specific college must be on the same date.

Skip

Submit

Example

For the relation Apply(SSN,cName,state,date,major), what real-world constraint is captured by SSN,cName → date?

- ○ A student can only apply to one college.
- ○ A student can apply to each college only once.
- ○ A student must apply to all colleges on the same date.
- ◉ Every application from a student to a specific college must be on the same date.

Correct

Show Explanation

Continue

3.1.2 Keys of Relations

- A set of attributes $\{A_1, A_2, \ldots, A_n\}$ is a Key of R if:
  1. The set functionally determines R.
  2. No proper subset of it functionally determines all other attributes of R. (Minimal)
- If a relation has more than one key, then we designate one of them as Primary Key.

3.1.2 Keys of Relations (cont'd)

Consider the following relation:

Title	*Year*	*Length*	*Genre*	*StudioName*	*StarName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

{Title, Year} is a key for the above relation. Why?

{Title, Year, StarName} is a key for the above relation. Why?

{Title, Year, StarName, genre} is a key for the above relation. Why?

3.1.3 Superkeys

- A set of attributes that contain a key is called a Superkey.
- Superkey: “Superset of a Key”.

{Title, Year} is a superkey for the above relation. Why?

{Title, Year, StarName} is a superkey for the above relation. Why?

{Title, Year, StarName, genre} is a superkey for the above relation. Why?

Section 3.2

RULES ABOUT FUNCTIONAL DEPENDENCIES

3.2 Rules About Functional Dependencies

Reasoning About Functional Dependencies

The Splitting/Combining Rule

Trivial Functional Dependencies

Computing the Closure of Attributes

The Transitive Rule

Projecting Functional Dependencies

3.2.1 Reasoning About FD's

Example 3.4 (transitive rule)

If relation R(A,B,C) has the following FD's:

A → B

B → C

Then we can deduce that R also has:

A → C FD as well.

(a,b_1,c_1) (a,b_2,c_2) ⇒ (a,b,c_1) (a,b,c_2) ⇒ (a,b,c) (a,b,c)

3.2.1 Reasoning About FD's (cont'd)

- Definition: Equivalency of FD's set
  Two sets of FD's S and T are equivalent if the set of relation instances satisfying S is exactly the same as the set of relation instances satisfying T.
- Definition: S follows T
  A set of FD's S follows from a set of FD's T if any relation instance that satisfies all the FD's in T also satisfies all the FD's in S.

- Two sets of FD's S and T are equivalent iff S follows from T and T follows from S.

3.2.2 The Splitting / Combining Rule

- Splitting Rule:

  $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$

  Is equivalent to:

  $A_1A_2\ldots A_n \rightarrow B_1$

  $A_1A_2\ldots A_n \rightarrow B_2$

  …

  $A_1A_2\ldots A_n \rightarrow B_m$

The Splitting / Combining Rule (cont'd)

- Combining Rule:

Consider the following FD's:

$A_1A_2\ldots A_n \rightarrow B_1$

$A_1A_2\ldots A_n \rightarrow B_2$

…

$A_1A_2\ldots A_n \rightarrow B_m$

We can combine them in one FD as:

$A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$

The Splitting / Combining Rule

Example 3.5

Consider the following FD's:

title year → length

title year → genre

title year → studioName

is equivalent to:

title year → length genre studioName

FD: title year → length

is NOT equivalent to:

title → length

year → length

Trivial FD's

Definition: Trivial Constraint

A constraint on a relation is said to be trivial if it holds for every instance of the relation, regardless of what other constraints are assumed.

For example, the following FD's are trivial:

title year → title

title → title

- In general, the FD:

  $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is trivial if the following condition satisfies:

  $\{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\}$

Trivial FD's

- Trivial FD
  - $A \rightarrow B \quad B \subseteq A$
- Non Trivial FD
  - $A \rightarrow B \quad B \not\subseteq A$
    - There may be some attributes in A that are repeated in B but not all of them
    - title year → title, length
- Completely nontrivial FD
  - $A \rightarrow B \quad A \cap B = \emptyset$
- If there are some attributes in the right side that has been repeated in the left side, just remove them.
- For example, in the following FD:

  $A_1A_2 \ldots A_n \rightarrow A_1B_1B_2 \ldots B_m$
  A_1 can be removed from the right side.

Computing the Closure of Attributes

- Definition: Closure
  Suppose A = $\{A_1,A_2,\ldots,A_n\}$ is a set of attributes of R and S is a set of FD's.

  The closure of A under the set S, denoted by A^+, is the set of attributes B such that any relation that satisfies all the FD's in S also satisfies $A_1A_2\ldots,A_n \rightarrow A^+$
- In other words, $A_1,A_2,\ldots,A_n \rightarrow A^+$ follows from the FD's of S.
- $A_1,A_2,\ldots,A_n$ are always in the $\{A_1,A_2,\ldots,A_n\}^+$

- Suppose R(A,B,C,D,E,F) and the
  - FD's AB→C, BC→AD, D→E, and CF→B satisfy.
  - Compute $\{A,B\}^+ =\{A,B,C,D,E\}$.

Computing the Closure of Attributes

Algorithm 3.7: Constructing Closure

1. Split the FD's so that each FD has a single attribute on the right side.
2. Initialize the closure set X by the set of given attributes.
3. Repeatedly search for the FD $B_1,B_2,\ldots,B_m \rightarrow C$ such that all of B_i are in the set X. Then add C to the X if it is not already there.
4. Continue until no more attribute can be added to the X.
5. X would be the closure of the A

Computing the Closure of Attributes

- Suppose R(A,B,C,D,E,F) and the
  - FD's AB→C, BC→AD, D→E, and CF→B satisfy.

Compute $\{A,B\}^+$.

First split BC→AD into BC→A and BC→D.

Start with X={A,B} and consider AB→C; A and B are in X, so, we add C to X. Now X={A,B,C}.

From BC→D, add D. Now X={A,B,C,D}

From D→E, add E. Now X={A,B,C,D,E}

Nothing new can be added. So, X={A,B,C,D,E}.

Computing the Closure of Attributes

- Suppose R(A,B,C,D,E,F) and the
  - FD's AB→C, BC→AD, D→E, and CF→B satisfy.

We wish to test whether AB→D follows from the set of FD's?

We compute $\{A,B\}^+$ which is {A,B,C,D,E}.

Since D is a member of the closure, we conclude that it follows.

Computing the Closure of Attributes

- Suppose R(A,B,C,D,E,F) and the
  - FD's AB→C, BC→AD, D→E, and CF→B satisfy.
  - We wish to test whether D→A follows from the set of FD's?

We compute $\{D\}^+$ first.

We start from X={D}.

From D→E, add E to the set. Now X= {D,E}.

We are stuck and no other FD's you can find that the left side is in X.

Since A is not in the list, so, D→A doesn't follow.

Example

- Student (Ssn, Sname, address, hscode, hsname, hscity, gpa, priority)
  - ssn → sname, address, gpa
  - hscode → hsname, hscity
  - Gpa → priority
- $\{ssn, hscode\}^{+} =$ {ssn, hscode}
  - = {ssn, hscode, sname, address, gpa}
  - = {ssn, hscode, sname, address, gpa, hsname, hscity }
  - = {ssn, hscode, sname, address, gpa, hsname, hscity, priority }
  - This forms a key for the relation

»

The Transitive Rule

Definition:

If $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ and

$B_1B_2\ldots B_m \rightarrow C_1C_2\ldots C_k$ holds, then

$A_1A_2\ldots A_n \rightarrow C_1C_2\ldots C_k$ also holds.

If there are some A's in the C's, you can eliminate them based on trivial-dependencies rule.

- Using the closure algorithm in two steps.
- Due to $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$, $\{A_1A_2\ldots A_n\}^+$ contains $B_1B_2\ldots B_m$
- since all Bs are in closure of A, due to $B_1B_2\ldots B_m \rightarrow C_1C_2\ldots C_k$, $\{C_1C_2\ldots C_k\}$ are also in closure of $\{A_1A_2\ldots A_n\}^+$ so $A_1A_2\ldots A_n \rightarrow C_1C_2\ldots C_k$ holds.

The Transitive Rule (cont'd)

Title	Year	Length	Genre	StudioName	studioAddr
Star Wars	1977	124	SciFi	Fox	Hollywood
Eight Below	2005	120	Drama	Disney	Buena Vista
Star Wars	1992	95	Comedy	Paramount	Hollywood

Two FD's that hold:

Title year → studioName

studioName → studioAddr

The transitive rule holds, so, we get:

Title year → studioAddr

if $\{A_1, A_2, \ldots, A_n\}^+$ contains the whole attributes of the relation, then $\{A_1,A_2,\ldots,A_n\}$ is a superkey of the relation. Because this is the only situation that the set of A's functionally determine all other attributes.

The Transitive Rule (cont'd)

- One way of testing if a set of attributes, let's say A, is a key, is:

1. Find it's closure A^+.
2. Make sure that it contains all attributes of R.
3. Make sure that you cannot create a smaller set, let's say A', by removing one or more attributes from A, that has the property 2.

Example

Consider the relation R(A,B,C,D,E) and suppose we have the functional dependencies:

$AB \rightarrow C$

$AE \rightarrow D$

$D \rightarrow B$

Which of the following attribute pairs is a key for R?

- ○ AB
- ○ AC
- ○ AD
- ○ AE

Skip

Submit

Example

Consider the relation R(A,B,C,D,E) and suppose we have the functional dependencies:

$AB \rightarrow C$

$AE \rightarrow D$

$D \rightarrow B$

Which of the following attribute pairs is a key for R?

- ○ AB
- ○ AC
- ○ AD
- ◉ AE

Correct

Show Explanation

Continue

Example

Consider the relation R(A,B,C,D,E) and the set of functional dependencies S1 = {AB → C, AE → D, D → B}.

Which of the following sets S2 of FDs does NOT follow from S1?

- ○ **S2 = {AD → C}**
- ○ **S2 = {AD → C, AE → B}**
- ○ **S2 = {ABC → D, D → B}**
- ○ **S2 = {ADE → BC}**

Skip

Submit

Example

Consider the relation R(A,B,C,D,E) and the set of functional dependencies S1 = {AB → C, AE → D, D → B}.

Which of the following sets S2 of FDs does NOT follow from S1?

- ○ S2 = {AD → C}
- ○ S2 = {AD → C, AE → B}
- ◉ S2 = {ABC → D, D → B}
- ○ S2 = {ADE → BC}

Correct

Show Explanation

Continue

Closing Sets of Functional Dependencies

Definition: Basis

The set of FD's that represent the full set of FD's of a relation is called a basis.

Minimal basis satisfies 3 conditions:

1. Singleton right side
2. If we remove any FD's from the set, the result is no longer a basis.
3. If we remove any attribute from the left side of any FD's, the result is not a basis.

Closing Sets of Functional Dependencies

Consider R(A, B, C)

Each attribute functionally determines the other two attributes.

The full set of derived FD's are six:

A → B, A → C, B → A, B → C, C → A, C → B.

But we don't need all of these to represent the FD's.

What is the minimal basis?

- A → B, B → C, C → A.
- A → B, B → A, B → C, C → B.

Projecting FD's

- Given a relation R and a set of FD's S
- What FD's hold if we project R by:
  $R_1 = \prod_L (R)$?
- We should compute the projection of functional dependencies S.
- This new set S' should:
  1. Follows from S
  2. Involves only attributes of R_1

- S={A→B, B→C, C→D}, and R1(A,C,D) is a projection of R. Find FD's for R1.
  - S'={A→C, C→D}

- The algorithm for calculating S' is exponential in $|R_1|$

Projecting FD's (cont'd)

Algorithm 3.12: Projecting a set of functional dependencies

Inputs:

R: the original relation

R1: the projection of R

S: the set of FD's that hold in R

Outputs:

T: the set of FD's that hold in R1

Projecting FD's

Algorithm 3.12: Projecting a set of functional dependencies

Method:

1. Initialize T={}.
2. Construct a set of all subsets of attributes of R1 called X.
3. Compute X_i^+ for all members of X under S. X_i^+ may consists of attributes that are not in R1.
4. Add to T all nontrivial FD's X→A such that A is both in X_i^+ and an attributes of R1.
5. Now, T is a basis for the FD's that hold in R1 but may not be a minimal basis. Modify T as follows:
   (a) If there is an FD F in T that follows from the other FD's in T, remove F.
   (b) Let Y→B be an FD in T, with at least two attributes in Y. Remove one attribute from Y and call it Z. If Z→B follows from the FD's in T, then replace Z→B with Y→B.
   (c) Repeat (b) until no more changes can be made to T.

Projecting FD's

Suppose R(A,B,C,D), S={A→B, B→C, C→D},

R1(A,C,D) is a projection of R. Find FD's for R1.

We should find all subsets of {A,C,D} which has 8 members but all of them are not needed.

To prune some members, note that:

- {} and {A,C,D} will give us trivial FD's.
- If the closure of some set X has all attributes , then we cannot find any new FD's by closing supersets of X.

- First $\{A\}^+$={A,B,C,D}. Thus, A→A, A→B, A→C, and A→D hold in R. but A→A is trivial, A→B contains B that is not in R1. So, we pick A→C, and A→D that would hold on R1.
- Second $\{C\}^+$={C,D}. Thus, C→C, and C→D hold in R. Again C→C is trivial, So, we pick C→D that would hold on R1.
- Third $\{D\}^+$={D}. Thus, D→D holds in R which is trivial.

Projecting FD's

$\{A\}^+=\{A,B,C,D\}$ that consists of all attributes of R, thus, we cannot find any new FD's by closing supersets of A. So, we don't need to compute $\{A,C\}^+$, $\{A,D\}^+$.

Forth, $\{C,D\}^+=\{C,D\}$. Thus, CD→C, and CD→D hold for R which both are trivial.

So, T={A→C, A→D, C→D} holds for R1.

A→D follows from the other two by transitive rule. Thus, T={A→C, C→D} is the minimal basis of R1.

DESIGN OF RELATIONAL DATABASE SCHEMAS

Anomalies

Decomposing Relations

Boyce - Codd Normal Form

Decomposition into BCNF

Anomalies

- The problem that is caused by the presence of certain dependencies is called anomaly. The principal kinds of anomalies are:
  - Redundancy: Unnecessarily repeated info in several tuples
    - Star Wars, 1977, 124, SciFi, and Fox is repeated.
  - Update Anomaly: Changing information in one tuple but leaving the same info unchanged in another
    - If you find out that Star Wars is 125 minute and you don't update all of them, you will lose the integrity.
  - Deletion Anomaly: Deleting some info and losing other info as a side effect
    - If you delete the record containing Vivien Leigh, then you'll lose the info for the movie "Gone with the wind"

Title	Year	Length	Genre	StudioName	StarName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Decomposing Relations

- The accepted way to eliminate the anomalies is to decompose the relation into smaller relations.
- It means we can split the attributes to make two new relations.
- The new relations won't have the anomalies.
- But how can we decompose?

3.3.2 Decomposing Relations (cont'd)

- Given a relation $R(A_1,A_2,\ldots,A_n)$, we may decompose R into two relations:
  $S(B_1,B_2,\ldots,B_m)$ and $T(C_1,C_2,\ldots,C_k)$ such that:

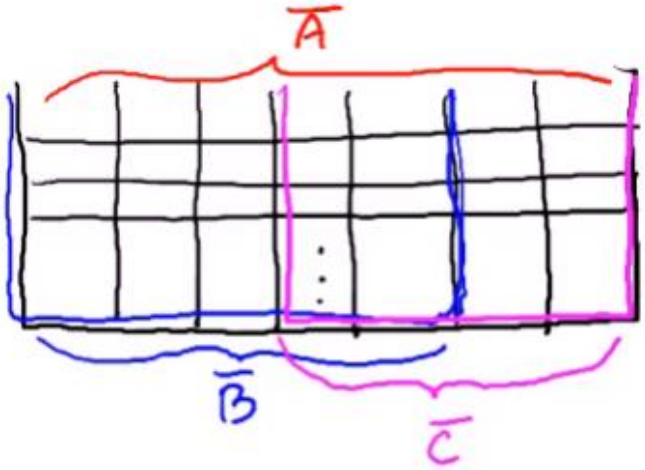
1. $\{A_1,A_2,\ldots,A_n\} = \{B_1,B_2,\ldots,B_m\} \cup \{C_1,C_2,\ldots,C_k\}$
2. $S = \pi_{B_1, B_2, \ldots, B_m}(R)$
3. $T = \pi_{C_1, C_2, \ldots, C_k}(R)$

S Natural Join T =R

$R(A_1, \ldots, A_n)$ $\bar{A}$

↳ $R_1(B_1, \ldots, B_k)$ $\bar{B}$ $\qquad \bar{B} \cup \bar{C} = \bar{A}$ ✓

$R_2(C_1, \ldots, C_m)$ $\bar{C}$ $\qquad R_1 \bowtie R_2 = R$ ✓



- Student (ssn, sname, address, hscode, hsname, hscity, gpa, priority)
  - S1(ssn, sname, address, hscode, gpa, priority)
  - S2(hscode, hsname, hscity)
    - S1 UNION S2 = Student
    - S1 Natural Join S2 = Student
  - S3(ssn, sname, address, hscode, hscity gpa, priority)
  - S4(sname, hsname,gpa, priority)
    - S3 UNION S4 = Student
    - S3 Natural Join S4 <> Student

Decomposing Relations

We can decompose the previous relation into Movie2 and Movie3 as follows:

Title	Year	Len	Genre	StudioName
Star Wars	1977	124	SciFi	Fox
Gone with the wind	1939	231	Drama	MGM
Wayne's World	1992	95	Comedy	Paramount

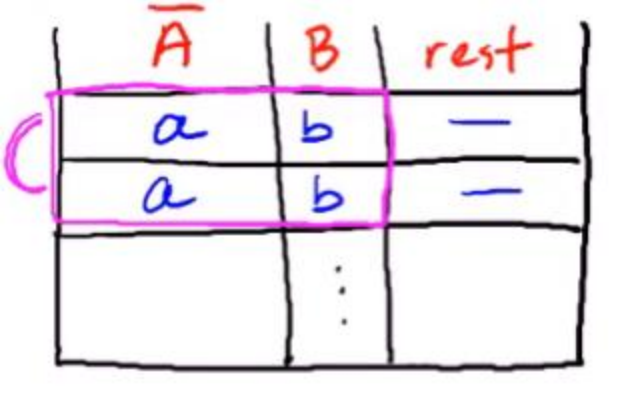
Title	Year	StarName
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone with the wind	1939	Vivien Leigh
Wayne's World	1992	Dana Carvey
Wayne's World	1992	Mike Meyers

- Do you think that the anomalies are gone?
  - Redundancy
  - Update
  - Delete

Title	Year	Length	Genre	StudioName	StarName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Boyce-Codd Normal Form

- Boyce-Codd Normal Form (BCNF) guarantees that the previous mentioned anomalies won't happen.
- A relation is in BCNF
  Iff whenever a nontrivial FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ holds,
  then $\{A_1,A_2,\ldots,A_n\}$ is a superkey of R.
- In other words, the left side of any FD must be a superkey.
- Note that we don't say minimal superkey.
- Superkey contains a key.



3.3.3 Boyce-Codd Normal Form (cont'd)

Consider the following relation:

Title	Year	Length	Genre	StudioName	StarName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

This relation is NOT in BCNF because FD

Title Year → length holds but {Title, Year} is NOT a superkey.

Note that the key of this relation is {Title, Year, StarName}

3.3.3 Boyce-Codd Normal Form (cont'd)

Example 3.16

Consider the following relation

Title	Year	Len	Genre	StudioName
Star Wars	1977	124	SciFi	Fox
Gone with the wind	1939	231	Drama	MGM
Wayne's World	1992	95	Comedy	Paramount

This relation is in BCNF because the key of this relation is {Title, Year} and all other FD's in this relation contain this key.

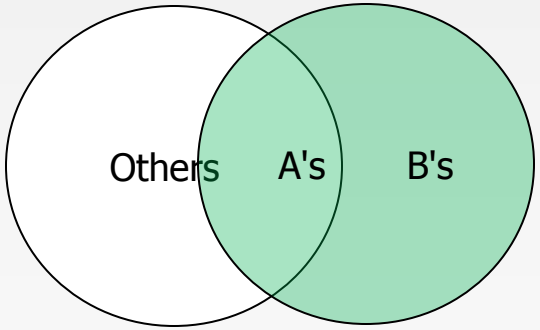
Decomposition into BCNF

- If we can find a suitable decomposition algorithm, then by repeatedly applying it, we can break any relation schema into a collection of subset of its attributes with the following properties:

1. These subsets are in BCNF
2. We can reconstruct the original relation from the decomposed relations.

One strategy we can follow is to find a nontrivial FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ that violates BCNF, i.e., $\{A_1,A_2,\ldots,A_n\}$ is not a superkey.

Then we break the attributes of the relation into two sets, one consists all A's and B's and the other contains A's and the remaining attributes.



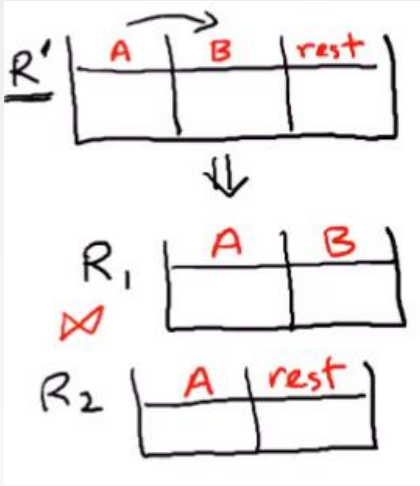
Decomposition into BCNF (cont'd)

BCNF Decomposition

Input: A relation R_0 with a set of FD's S_0.

Output: A decomposition of R_0 into a collection of relations, all in BCNF

1. Suppose that X → Y is a BCNF violation.
2. Compute X^+ and put $R_1 = X^+$
3. R_2 contain all X attributes and those that are not in X^+
4. Project FD's for R_1 and R_2
5. Recursively decompose R_1 and R_2



Decomposition into BCNF (cont'd)

Consider the relation Movie1

The following FD is a BCNF violation: title year → length genre studioName

Title	Year	Length	Genre	StudioName	StarName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

We can decompose it into:

{title, year, length, genre, studioName} and

{title, year, starName}

Title	Year	Len	Genre	StudioName
Star Wars	1977	124	SciFi	Fox
Gone with the wind	1939	231	Drama	MGM
Wayne's World	1992	95	Comedy	Paramount

Title	Year	StarName
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone with the wind	1939	Vivien Leigh
Wayne's World	1992	Dana Carvey
Wayne's World	1992	Mike Meyers

Example

- Student (Ssn, Sname, address, hscode, hsname, hscity, gpa, priority)
  - ssn → sname, address, gpa
  - hscode → hsname, hscity
  - gpa → priority
- The key for the relation is {ssn, hscode}
  - This is not in BCNF

Example

- Student (Ssn, Sname, address, hscode, hsname, hscity, gpa, priority)
  - ssn → sname, address, gpa
  - hscode → hsname, hscity
  - Gpa → priority
- Pick a violation and decompose (hscode → hsname, hscity)
  - S1(hscode, hsname, hscity)
  - S2(Ssn, Sname, address, hscode, gpa, priority)
- Pick a violation and decompose (gpa → priority)
  - S1(hscode, hsname, hscity)
  - S3 (gpa, priority)
  - S4(ssn, sname, address, hscode, gpa)
- Pick a violation and decompose (ssn → sname, address, gpa)
  - S1(hscode, hsname, hscity)
  - S3 (gpa, priority)
  - S5(ssn, sname, address, gpa)
  - S6(ssn, hscode)

Boyce-Codd Normal Form

- Prove that any two-attribute relation is in BCNF.
  - Let's assume that the attributes are called A, B.
  - The only way the BCNF condition violates is when there is a nontrivial FD which is not a superkey. Let's check all possible cases:
  - 1. There is no nontrivial FD
    - BCNF condition must hold because only a nontrivial FD can violate.
  - 2. A → B holds but B → A doesn't hold
    - The only key of this relation is A and all nontrivial FD, which in this case is just A → B, contain A. So, there shouldn't be any violation.
  - 3. B → A holds but A → B doesn't hold
    - Proof is the same as case # 2
  - 4. Both A → B and B → A hold
    - Then both A and B are keys and any FD's contain one of these.
      - some key be contained in the left side of any nontrivial FD

Consider relation Apply(SSN,cName,state,date,major) with FDs cName → state and SSN,cName → date,major. What schema would be produced by the BCNF decomposition algorithm?

- ○ Apply(SSN,cName,state,date,major)
- ○ A1(cName,state), A2(SSN,cName,date,major)
- ○ A1(cName,state), A2(SSN,date,major)
- ○ A1(cName,state), A2(SSN,cName,date), A3(SSN,cName,major)

Skip

Submit

Consider relation Apply(SSN,cName,state,date,major) with FDs cName → state and SSN,cName → date,major. What schema would be produced by the BCNF decomposition algorithm?

- ○ Apply(SSN,cName,state,date,major)
- ◉ A1(cName,state), A2(SSN,cName,date,major)
- ○ A1(cName,state), A2(SSN,date,major)
- ○ A1(cName,state), A2(SSN,cName,date), A3(SSN,cName,major)

Correct

Show Explanation

Continue

DECOMPOSITION: THE GOOD, BAD, AND UGLY

Recovering Information from a Decomposition

The Chase Test for Lossless Join

Why the Chase Works

Dependency Preservation

Decomposition: The Good, Bad, and Ugly

- When we decompose a relation using the algorithm 3.20, the resulting relations don't have anomalies. This is the Good.

- Our expectations after decomposing are:
  1. Elimination of Anomalies
  2. Recoverability of Information
     - Can we recover the original relation from the tuples in its decompositions?
  3. Preservation of Dependencies
     - Can we be sure that after reconstructing the original relation from the decompositions, the original FD's satisfy?

Decomposition: The Good, Bad, and Ugly (cont'd)

- The BCNF decomposition of algorithm 3.20 gives us the expectations number 1 and 2 but it doesn't guarantee about the 3.

- Proof of Recovering Information from a Decomposition
  - If we decompose a relation according to Algorithm 3.20, then the original relation can be recovered exactly by the natural join. (lossless join)

Proof of Recovering Information from a Decomposition

- Suppose we have the relation R(A, B, C) and B→ C holds. A sample for R can be shown as the following relation:

A	B	C
a	b	c

Then we decompose R into R1 and R2 as follows:

A	B
a	b

B	C
b	c

Joining the two would get the R back.

Proof of Recovering Information from a Decomposition

- However, getting the tuples we started back is not enough to assume that the original relation R is truly represented by the decomposition.

A	B	C
a	b	c
d	b	e

Then we decompose R into R1 and R2 as follows:

A	B
a	b
d	b

B	C
b	c
b	e

Because of B→ C, we can conclude that c=e are the same so really there is only one tuple in R2

Proof of Recovering Information from a Decomposition

- Note that the FD should exists, otherwise the join wouldn't reconstruct the original relation as the next example shows
- Suppose we have the relation R(A, B, C) but neither B → A nor B→ C holds. A sample for R can be shown as the following relation:

A	B	C
a	b	c
d	b	e

Then we decompose R into R1 and R2 as follows:

A	B
a	b
d	b

B	C
b	c
b	e

Proof of Recovering Information from a Decomposition

Since both R1 and R2 share the same attribute B, if we natural join them, we'll get:

A	B	C
a	b	c
a	b	e
d	b	c
d	b	e

We got two bogus tuples, (a, b, e) and (d, b, e).

3.4.2 The Chase Test for Lossless Join

- Let's consider more general situation
- The algorithm decides whether the decomposition is lossless or not.
  - Input
    - A relation R
    - A decomposition of R
    - A set of Functional Dependencies
  - Output
    - Whether the decomposition is lossless or not
    - $\prod_{S1}(R) \bowtie \prod_{S2}(R) \bowtie \ldots\ldots \bowtie \prod_{Sk}(R) = R$?
- Three things
  - Natural join is associative and commutative. It doesn't matter what order we join
  - Any tuple t in R is surely in $\prod_{S1}(R) \bowtie \prod_{S2}(R) \bowtie \ldots\ldots \bowtie \prod_{Sk}(R)$. Projection of t to S_1 is surely in $\prod_{S1}(R)$
  - We have to check to see any tuple in the $\prod_{S1}(R) \bowtie \prod_{S2}(R) \bowtie \ldots\ldots \bowtie \prod_{Sk}(R)$ is in relation R or not

Tableau

- Suppose we have relation R(A,B,C,D), we have decomposed into
  - S1{A,D}, S2{A,C}, S3{B,C,D}
  - FD: A→B, B→C, CD→A

A	B	C	D
a	b1	c1	d
a	b2	c	d2
a3	b	c	d

- A→B,

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

B→C,

A	B	C	D
a	b1	c	d
a	b1	c	d2
a3	b	c	d

CD→A

A	B	C	D
a	b1	c	d
a	b1	c	d2
a	b	c	d

Example 3.23

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a	b	c	d

- S1{A,D},

A	D
a	d
a	d2

S2{A,C},

A	C
a	c1
a	c

S3{B,C,D}

B	C	D
b1	c1	d
b2	c	d2
b	c	d

- A→B,

A	C	D
a	c1	d
a	c	d2
a	c	D
a	c1	d2

B→C,

CD→A

A	B	C	D
a	b1	c1	d
a	b2	c	d2
a	b	c	d

Example 3.24

- Suppose we have relation R(A,B,C,D), we have decomposed into
  - S1{A,B}, S2{B,C}, S3{C,D}
  - FD B→AD

A	B	C	D
a	b	c1	d1
a2	b	c	d2
a3	b3	c	d

  - B→AD

A	B	C	D
a	b	c1	d1
a	b	c	d1
a3	b3	c	d

Example 3.24

A	B	C	D
a	b	c1	d1
a	b	c	d1
a3	b3	c	d

- S1{A,B},

A	B
a	b
a3	b3

S2{B,C},

B	C
b	c1
b	c
b3	c

S3{C,D}

C	D
c1	d1
c	d1
c	d

B→AD

A	B	C
a	b	c1
a	b	c
a3	b3	c

A	B	C	D
a	b	c1	d1
a	b	c	d1
a	b	c	d
a3	b3	c	d1
a3	b3	c	d

3.4.4 Dependency Preservation

- Example Bookings
  - Title name of movie
  - Theater, name of theaters showing the movie
  - City
    - Theater→ city
    - Title,city → theater (not booking a movie into two theaters in a city)

Theatre	City	title
guild	M P	Antz

- Keys? Check for closure
  - {title, city}
  - {theatre, title}
- Theater→ city violates BCNF

Theatre	City	title
guild	M P	Antz

- Lets decomposed the table based on that violation

Theatre	city
guild	Menlo Park

Theatre	title
guild	Antz

- {theater, city} and {theater, title}

- This decomposition cannot handle Title,city → theater

Theatre	city
guild	Menlo Park
Park	Menlo Park

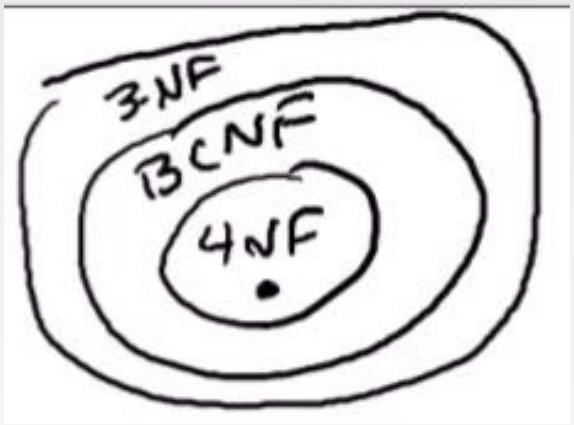
Theatre	title
guild	Antz
Park	Antz

Theatre	City	title
guild	M P	Antz
Park	M P	Antz

THIRD NORMAL FORM

Definition of Third Normal Form

- An attribute that is a member of some key is called a prime.
- Definition: 3rd Normal Form (3NF)
  A relation R is in 3rd normal form if:
  - For each nontrivial FD, either the left side is a superkey (BCNF), or the right side consists of prime attributes only.



- Our expectations after decomposing are:
  1. Elimination of Anomalies
  2. Recoverability of Information
     - Can we recover the original relation from the tuples in its decompositions?
  3. Preservation of Dependencies
     - Can we be sure that after reconstructing the original relation from the decompositions, the original FD's satisfy?

- 3rd Normal form can give us 2 and 3, but not 1

3.5.2 The Synthesis Algorithm for 3NF Schemas

Algorithm 3.26: Synthesis of 3NF Relations with a lossless join and dependency preservation

Input: A relation R and a set of FD's called F

Output: A decomposition of R into a collection of relations in 3NF

1. Find a minimal basis for F, say G.
2. For each FD like X → A, use XA as the schema of one of the relations in the decomposition.
3. If none of the relations is a superkey, add another relation whose schema is a key for R.

The Synthesis Algorithm for 3NF Schemas

- Consider R(A,B,C,D,E) with
  - AB→C, C→B, and A→D.
  - First, check if the FD's are minimal.
  - To verify, we should show that we cannot eliminate any of FD's. That is, we show using Algorithm 3.7, that no two of the FD's imply the third.
    - We find $\{A,B\}^+$ using the other two FD's C→B, and A→D.
      - $\{A,B\}^+=\{A,B,D\}$ It contains D and not C. Thus, this FD does not follow the other two.
    - We find $\{C\}^+$ using the other two FD's AB→C, and A→D.
      - $\{C\}^+ =\{C\}$ which doesn't have B
    - We find $\{A\}^+$ using the other two FD's AB→C, and C→B.
      - $\{A\}^+ =\{A\}$ which doesn't have D

3.5.2 The Synthesis Algorithm for 3NF Schemas (cont'd)

- Similarly, you can prove that S is minimal. (we cannot eliminate any attributes from left side of any of the FDs)
  - Check both A→C , or B→C for not being implied by others

- Make a new relation using the FD's, therefore, we would have S1(A,B,C), S2(C,B), and S3(A,D)
  - When we have S1(A,B,C), then we drop S2(C,B).
  - Verify that {A,B,E} and {A,C,E} are keys of R.

Neither of these keys is a subset of the schemas chosen so far. Thus, we must add one of them, say S4(A,B,E).

- The final decompositions would be: S1(A,B,C), S3(A,D), S4(A,B,E)

3.5.3 Why the 3NF Synthesis Algorithm Works

- Two things to prove
  - Lossless join: we can use the chase algorithm. Start with the table with attributes k that includes a super key.
    - Since k contains key then k+ contains all the attributes which means there is a row in tableau that contains no subscriptions.
      - S1(A,B,C), S2(A,D), s3{A,B,E}
      - FD AB→C, C→B, and A→D

A	B	C	D	E
a	b	c	d1	e1
a	b2	c2	d	e2
a	b	c3	d3	e

A	B	C	D	E
a	b	c	d1	e1
a	b2	c2	d	e2
a	b	c	d3	e

A	B	C	D	E
a	b	c	d	e1
a	b2	c2	d	e2
a	b	c	d	e

  - Dependency Preservation: each FD of the minimal basis has all its attributes in some relation.

The Closure algorithm (extended)

- We can check whether an FD X →Y follows from a given set of FD's F using the chase algorithm.
  - We have relation R(A,B,C,D,E,F)
  - FD's AB→C, BC→AD, D→E, CF→B
  - Check whether AB→D holds or not

A	B	C	D	E	F
a	b	c1	d1	e1	f1
a	b	c2	d2	e2	f2

- AB→C , BC→AD

A	B	C	D	E	F
a	b	c1	d1	e1	f1
a	b	c1	d1	e2	f2

- Since the two tuples now agree on D we conclude that AB→D Follows

Section 3.6

MULTI-VALUED DEPENDENCIES

3.6 Multi-valued Dependencies

Attribute Independence and Its Consequent Redundancy

Definition of Multi-valued Dependencies

Reasoning About Multi-valued Dependencies

Fourth Normal Form

Decomposition into Fourth Normal Form

Relationships Among Normal Forms

Attribute Independence and Its Consequent Redundancy

- BCNF eliminates redundancy in each tuple but may leave redundancy among tuples in a relationship
- This typically happens if two many-many relationships (or in general: a combination of two types of facts) are represented in one relation

name	*street*	*city*	*title*	*year*
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
C. Fisher	123 Maple St.	Hollywood	Empire Strikes Back	1980
C. Fisher	123 Maple St.	Hollywood	Return of the Jedi	1983
C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977
C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980
C. Fisher	5 Locust Ln.	Malibu	Return of the Jedi	1983

- Every street address is given 3 times and every title is repeated twice
- What is the key?
  - All of the attributes
- This table does not violate BCNF but has redundancy among tuples.

Definition of Multi-valued Dependencies

- A MVD is a statement about some relation R that when you fix the values for one set of attributes, then the values in certain other attributes are independent of the values of all the other attributes in the relation

$R \quad \bar{A} \twoheadrightarrow \bar{B} \quad A_1, \ldots, A_n \quad B_1, \ldots, B_n$

$\forall t, u \in R : t[\bar{A}] = u[\bar{A}]$ then

$\exists v \in R : v[\bar{A}] = t[\bar{A}]$ and

$v[\bar{B}] = t[\bar{B}]$ and

$v[\text{rest}] = u[\text{rest}]$

	$\bar{A}$	$\bar{B}$	rest
t	$\bar{a}$	$\bar{b}_1$	$\bar{r}_1$
u	$\bar{a}$	$\bar{b}_2$	$\bar{r}_2$
v	$\bar{a}$	$\bar{b}_1$	$\bar{r}_2$
w	$\bar{a}$	$\bar{b}_2$	$\bar{r}_1$

Example

- Name →→ street, city

	name	*street*	*city*	*title*	*year*
t	C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
v	C. Fisher	123 Maple St.	Hollywood	Empire Strikes Back	1980
	C. Fisher	123 Maple St.	Hollywood	Return of the Jedi	1983
w	C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977
u	C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980
	C. Fisher	5 Locust Ln.	Malibu	Return of the Jedi	1983

N * M

name	*street*	*city*	*title*	*year*
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
C. Fisher	123 Maple St.	Hollywood	Empire Strikes Back	1980
C. Fisher	123 Maple St.	Hollywood	Return of the Jedi	1983
C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977
C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980
C. Fisher	5 Locust Ln.	Malibu	Return of the Jedi	1983

N + M

name	*street*	*city*
C. Fisher	123 Maple St.	Hollywood
C. Fisher	5 Locust Ln.	Malibu

name	*title*	*year*
C. Fisher	Star Wars	1977
C. Fisher	Empire Strikes Back	1980
C. Fisher	Return of the Jedi	1983

Example

Apply(SSN, cName, hobby)

SSN →→ cName

	SSN	cName	hobby
t	123	Stanford	trumpet
u	123	Berkeley	tennis
	⋮	⋮	⋮

Apply(SSN, cName, hobby)

SSN →→ cName

	SSN	cName	hobby
t	123	Stanford	trumpet
u	123	Berkeley	tennis
v	123	Stanford	tennis
w	123	Berkeley	trumpet
	⋮	⋮	⋮

Consider a relation R(A,B,C) with multivalued dependency $A \twoheadrightarrow B$. Suppose there at least 3 different values for A, and each value of A is associated with at least 4 different B values and at least 5 different C values.

What is the minimum number of tuples in R?

- ○ 60
- ○ 15
- ○ 12
- ○ 27

Skip

Submit

Consider a relation R(A,B,C) with multivalued dependency A $\twoheadrightarrow$ B. Suppose there at least 3 different values for A, and each value of A is associated with at least 4 different B values and at least 5 different C values.

What is the minimum number of tuples in R?

- (•) 60
- ( ) 15
- ( ) 12
- ( ) 27

Show Explanation

Continue

Correct

For the relation Apply(SSN,cName,date,major) with functional dependency SSN,cName → date, what real-world constraint is captured by SSN ↠ cName,date?

- ○ A student can only apply to one major at each college.
- ○ A student can apply to different majors at each college, but each major must be applied for on a different date.
- ○ A student must apply to the same set of majors at all colleges.
- ○ A student must apply to a different major at each college.

Skip

Submit

For the relation Apply(SSN,cName,date,major) with functional dependency SSN,cName → date, what real-world constraint is captured by SSN ↠ cName,date?

- ○ A student can only apply to one major at each college.
- ○ A student can apply to different majors at each college, but each major must be applied for on a different date.
- ◉ A student must apply to the same set of majors at all colleges.
- ○ A student must apply to a different major at each college.

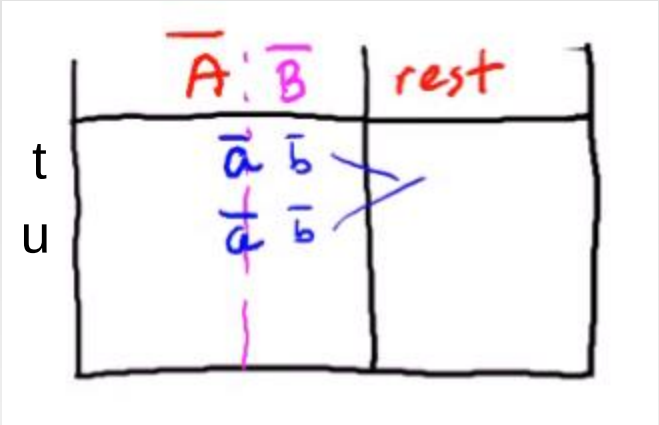
Show Explanation

Continue

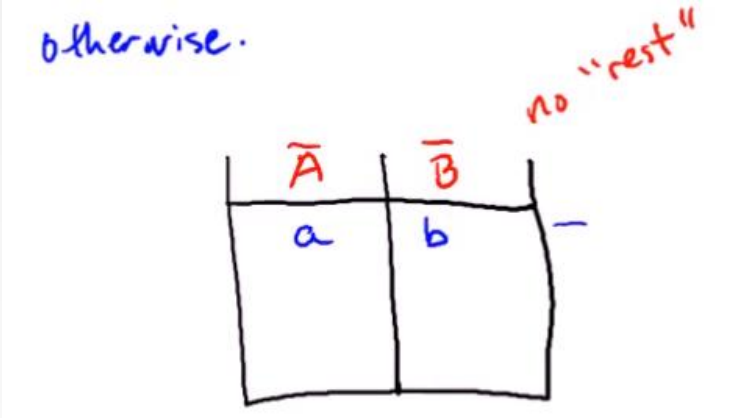
Correct

Reasoning About Multi-valued Dependencies

- Trivial MVD
  - $A_1,A_2,\ldots A_n \rightarrow\rightarrow B_1, B_2,\ldots.B_m$ holds if $B_1, B_2,\ldots.B_m \subset A_1,A_2,\ldots A_n$

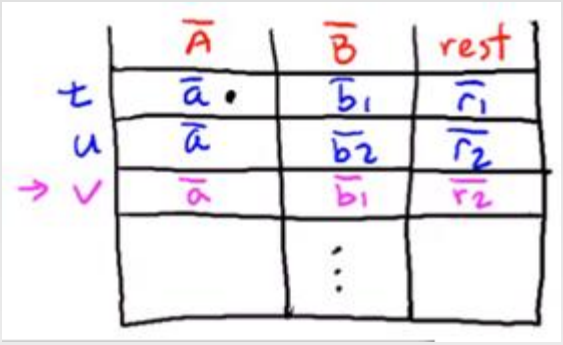


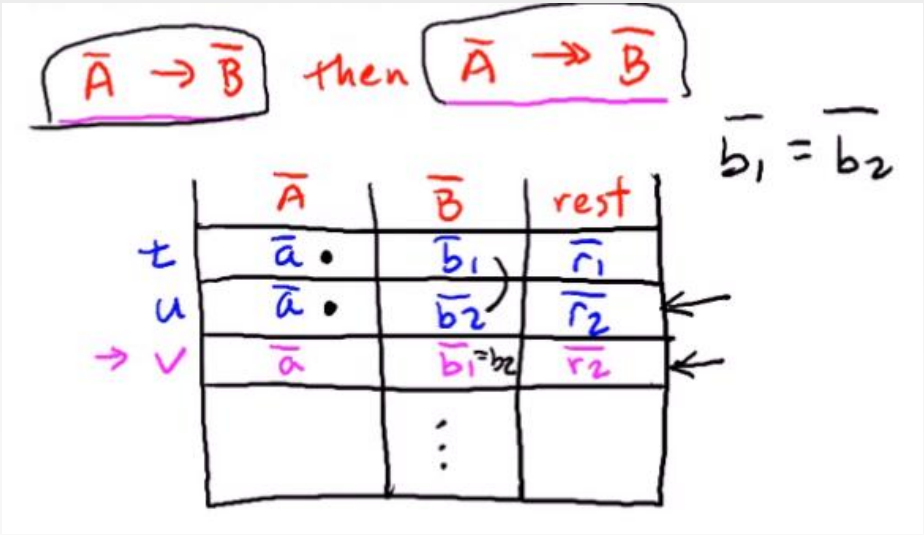
  - $A_1,A_2,\ldots A_n \rightarrow\rightarrow B_1, B_2,\ldots.B_m$ holds if $A_1,A_2,\ldots A_n \cup B_1, B_2,\ldots.B_m$



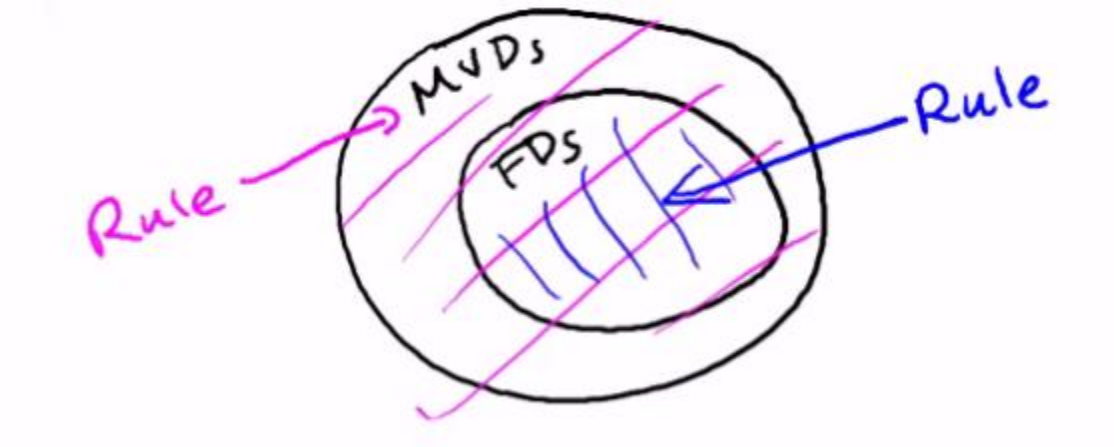
Reasoning About Multi-valued Dependencies

- If $A_1, A_2, \ldots A_n \rightarrow B_1, B_2, \ldots B_m$ then $A_1, A_2, \ldots A_n \rightarrow\rightarrow B_1, B_2, \ldots B_m$





Relation between MVDs and FDs



Reasoning About Multi-valued Dependencies

- Complementation Rule
  - If $A_1,A_2,\ldots A_n \rightarrow B_1, B_2,\ldots B_m$ then $A_1,A_2,\ldots A_n \rightarrow\rightarrow C_1, C_2,\ldots C_k$ holds if $C_1, C_2,\ldots C_k$ are all attributes of the relation not among the A's and B's

name	*street*	*city*	*title*	*year*
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
C. Fisher	123 Maple St.	Hollywood	Empire Strikes Back	1980
C. Fisher	123 Maple St.	Hollywood	Return of the Jedi	1983
C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977
C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980
C. Fisher	5 Locust Ln.	Malibu	Return of the Jedi	1983

  - Name →→ street, city
  - Name →→ title, year

Reasoning About Multi-valued Dependencies

- Splitting rule DOES NOT apply to MVDs
  - Name →→ street ,city is not the same as
    - Name →→ street
    - Name →→ city

Fourth Normal Form

- **Informal def:** a relation is in 4th normal form if it cannot be meaningfully decomposed into two relations. More precisely

- A relation R is in **4th normal form** (4NF) if whenever
  - $A_1,A_2,\ldots A_n \rightarrow\rightarrow B_1, B_2,\ldots.B_m$ is a nontrivial MVD, $A_1,A_2,\ldots A_n$ is a superkey.

name	*street*	*city*	*title*	*year*
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
C. Fisher	123 Maple St.	Hollywood	Empire Strikes Back	1980
C. Fisher	123 Maple St.	Hollywood	Return of the Jedi	1983
C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977
C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980
C. Fisher	5 Locust Ln.	Malibu	Return of the Jedi	1983

- Name →→ street, city
- Name →→ title, year

name	*street*	*city*
C. Fisher	123 Maple St.	Hollywood
C. Fisher	5 Locust Ln.	Malibu

name	*title*	*year*
C. Fisher	Star Wars	1977
C. Fisher	Empire Strikes Back	1980
C. Fisher	Return of the Jedi	1983

Consider relation StudentInfo(sID, name, dorm, major) with functional dependency sID → name and multivalued dependency sID ↠ dorm. What schema would be produced by the 4NF decomposition algorithm?

- StudentInfo(sID, name, dorm, major)
- S1(sID,name), S2(sID,dorm,major)
- S1(sID,name,dorm), S2(sID,major)
- S1(sID,name), S2(sID,dorm), S3(sID,major)

Consider relation StudentInfo(sID, name, dorm, major) with functional dependency sID → name and multivalued dependency sID ↠ dorm. What schema would be produced by the 4NF decomposition algorithm?

- ○ StudentInfo(sID, name, dorm, major)
- ○ S1(sID,name), S2(sID,dorm,major)
- ○ S1(sID,name,dorm), S2(sID,major)
- ◉ S1(sID,name), S2(sID,dorm), S3(sID,major)

Show Explanation

Continue

Correct

Decomposition into Fourth Normal Form

Suppose we have a relation R which is not in 4NF. Then there is a nontrivial MVD

$$A_1 A_2 \ldots A_n \twoheadrightarrow B_1 B_2 \ldots B_m$$

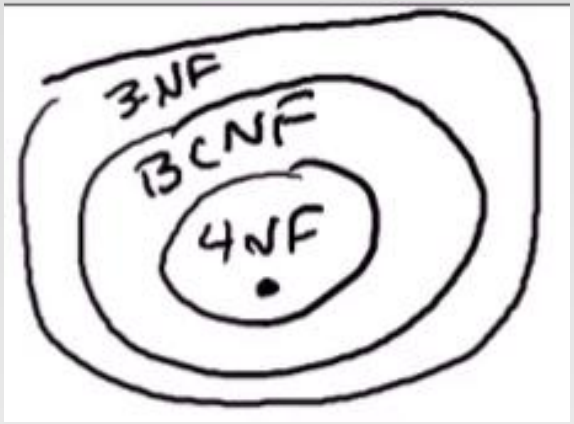
which is not unavoidable.

To eliminate the MVD we split R into two relations:

- One with all attributes of R except $B_1, B_2, \ldots, B_m$.
- One with attributes $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$.

If any of the resulting relations is not in 4NF, the process is repeated.

Relationships Among Normal Forms



Property	3NF	BCNF	4NF
Eliminate redundancies due to FD's	No	Yes	Yes
Eliminate redundancies due to MVD's	No	No	Yes
Preserves FD's	Yes	No	No
Preserve MVD's	No	No	No

The Closure algorithm for MVDs

- We can check whether an FD X →Y follows from a given set of FD's F using the chase algorithm.
  - We have relation R(A,B,C,D,E,F)
  - FD's AB→C, BC→AD, D→E, CF→B
  - Check whether AB→D holds or not

A	B	C	D	E	F
a	b	c1	d1	e1	f1
a	b	c2	d2	e2	f2

- AB→C , BC→AD

A	B	C	D	E	F
a	b	c1	d1	e1	f1
a	b	c1	d1	e2	f2

- Since the two tuples now agree on D we conclude that AB→D Follows

Extending to MVDs

- The Method can be applied to infer MVD's

- Example: Relation(A,B,C,D)
  - A→B, B→→C
  - Check whether A→→C holds or not

A	B	C	D
a	b1	c	d1
a	b	c2	d

We start with this, if the row (a,b,c,d) is produced we can conclude that A→→C holds

A→B

A	B	C	D
a	b	c	d1
a	b	c2	d

Extending to MVDs

B→→C

A	B	C	D
a	b	c	d1
a	b	c2	d

We can use this rule because the two rows have the same B value. We produce two new rows when using MVD rules producing the v and w row

	A	B	C	D
t	a	b	c	d1
u	a	b	c2	D
v	a	b	c	d
w	a	b	c2	d1

Row a,b,c,d is produced so A→→C follows