Lecture 3

Relational Database Design Theory (part 2)

关系数据库设计理论

(第2部分)

Outline

- Relation Decomposition
- Third Normal Form
- Multivalued Dependencies

Outline

- Relation Decomposition
- Third Normal Form
- Multivalued Dependencies

Decomposition: The Good, Bad and Ugly

- “good” decomposition ?
  - Before decomposition: Anomalies
  - After decomposition: No anomalies
- Bad or ugly decomposition ?
- Three properties
  - Elimination of Anomalies
  - Recoverability of Information
  - Preservation of Dependencies

Recovering Information from a Decomposition

- Lossless Join 无损连接
  - Why not decompose relation *R* into a set of relations whose schemas is a pair of *R*'s attribute?
  - Answer
    - The data in the decomposed relations might not allow us to join the relations of the decomposition and get the instance of *R* back.
    - If we do get *R* back, then we say the decomposition has a lossless join.

Lossless Join

- The conclusion
  - If we decompose a relation according to Algorithm 3.20, then original relation can be recovered exactly by the natural join.
    - For any sets of attributes X, Y and Z
    - If $Y \rightarrow Z$ holds in R, whose attributes are $X \cup Y \cup Z$
    - Then $R = \pi_{X \cup Y}(R) \bowtie \pi_{Y \cup Z}(R)$

Lossless Join

- Example
  - $R(A, B, C)$
  - $B \rightarrow C$ is a BCNF violation
  - $R_1(A, B) = \pi_{A,B}(R)$ and $R_2(B, C) = \pi_{B,C}(R)$
  - Let t and v be two tuples of R
    - $t=(a, b, c)$ and $v=(d, b, e)$
    - $u=(a, b)$, project t onto $R_1(A, B)$
    - $w=(b, e)$, project v onto $R_2(B, C)$
    - Join u and w, get $x=(a, b, e)$

Is it possible that x is a bogus tuple?
That is, could (a, b, e) not be a tuple of R?

Lossless Join

Is it possible that x is a bogus tuple?
That is, could (a, b, e) not be a tuple of R?

- Example
  - The answer is "no"
    - Since we assume $B \to C$ for relation R
    - $t=(a, b, c)$ and $v=(d, b, e)$
    - t and v agree in their B components, they also agree on their C components
    - That means $c=e$
    - Thus, (a, b, e) is really (a, b, c); that is, $x=t$

Lossless Join

- Another example
  - $R(A, B, C)$
  - Neither $B \rightarrow A$ nor $B \rightarrow C$ holds

A	B	C
1	2	3
4	2	5

$$R_1 = \pi_{A,B}(R) \quad R_2 = \pi_{B,C}(R)$$

A	B
1	2
4	2

B	C
2	3
2	5

$$R_3 = R_1 \bowtie R_2$$

A	B	C
1	2	3
1	2	5
4	2	3
4	2	5

The Chase Test for Lossless Join

- A more general situation
  - Relation R
  - Relations with sets of attributes $S_1, S_2, \ldots, S_k$
  - A given set of FD's F that hold in R

  Question:
    - Is it true that if we project R onto the relations of the decomposition, then we can recover R by taking the natural join of all these relations?

$$\pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \cdots \bowtie \pi_{S_k}(R) = R \quad ?$$

The Chase Test for Lossless Join

- Three important things
  1. The natural joins is associative and commutative
     - The result is the set of tuples t such that
     - for all $i = 1, 2, \ldots, k$
     - t projected onto the set of S_i is a tuple in $\pi_{S_i}(R)$

The Chase Test for Lossless Join

- Three important things

  2. Any tuple t in R is surely in

$$\pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \cdots \bowtie \pi_{S_k}(R)$$

  - The reason is that the projection of t onto S_i is surely in $\pi_{S_i}(R)$ for each i,
  - and therefore by our first point above, t is in the result of the join.

The Chase Test for Lossless Join

- Three important things

3. As a consequence,

$$\pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \cdots \bowtie \pi_{S_k}(R) = R$$

when the FD's in F hold for R if and only if

- every tuple in the join is also in R.

The membership test is all we need to verify that the decomposition has a lossless join.

The Chase Test for Lossless Join

- The chase test
  - An organized way to see whether a tuple t in

$$\pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \cdots \bowtie \pi_{S_k}(R)$$

can be proved also to be a tuple in R

- using the FD's in F

The Chase Test for Lossless Join

- The chase test
  - If t is in the join
  - Then there must be tuples in R, say t_1, t_2, …, t_k,
  - Such that t is the join of the projections of each t_i onto the set of attributes S_i, for i=1, 2, …, k.

  - t_i agrees with t on the attributes of S_i
  - t_i has unknown values in its components not in S_i

The Chase Test for Lossless Join

- Example
  - Relation $R(A, B, C, D)$
  - Decomposed into $S_1=\{A,D\}$, $S_2=\{A,C\}$, $S_3=\{B,C,D\}$
  - Tableau (图例) for this decomposition

A	B	C	D
a	b_1	c_1	d
a	b_2	c	d_2
a_3	b	c	d

The Chase Test for Lossless Join

- Our goal
  - Is to use the given set of FD's *F* to prove that *t* is really in *R*.
  - We "chase" the tableau by applying the FD's in *F* to equate symbols in the tableau whenever we can.

  - If we discover that one of the rows is actually the same as *t*
  - Then we have proved that any tuple *t* in the join of the projections was actually a tuple of *R*.

The Chase Test for Lossless Join

- Example
  - FD's = $\{A \rightarrow B,\ B \rightarrow C,\ CD \rightarrow A\}$

A	B	C	D
$\boldsymbol{a}$	b_1	c_1	d
$\boldsymbol{a}$	b_2	c	d_2
a_3	b	c	d

$A \rightarrow B$

A	B	C	D
$\boldsymbol{a}$	$\boldsymbol{b_1}$	c_1	d
$\boldsymbol{a}$	$\boldsymbol{b_1}$	c	d_2
a_3	b	c	d

The Chase Test for Lossless Join

- Example
  - FD's = $\{A \rightarrow B, B \rightarrow C, CD \rightarrow A\}$

A	B	C	D
a	$\mathbf{b_1}$	c_1	d
a	$\mathbf{b_1}$	c	d_2
a_3	b	c	d

$B \rightarrow C$ ⇒

A	B	C	D
a	$\mathbf{b_1}$	$\mathbf{c}$	d
a	$\mathbf{b_1}$	$\mathbf{c}$	d_2
a_3	b	c	d

The Chase Test for Lossless Join

- Example
  - FD's = $\{A \to B, B \to C, CD \to A\}$

A	B	C	D
a	b_1	$\mathbf{c}$	$\mathbf{d}$
a	b_1	c	d_2
a_3	b	$\mathbf{c}$	$\mathbf{d}$

$CD \to A$

A	B	C	D
$\mathbf{a}$	b_1	$\mathbf{c}$	$\mathbf{d}$
a	b_1	c	d_2
$\mathbf{a}$	b	$\mathbf{c}$	$\mathbf{d}$

$t=(a, b, c, d)$

Why the Chase Works

- Example
  - Relation $R(A, B, C, D)$
  - FD's = $\{B \rightarrow AD\}$
  - Decomposed into $S_1=\{A,B\}$, $S_2=\{B,C\}$, $S_3=\{C,D\}$

A	B	C	D
a	b	c_1	d_1
a_2	b	c	d_2
a_3	b_3	c	d

$B \rightarrow AD$

Why the Chase Works

- Example
  - Relation $R(A, B, C, D)$
  - FD's = $\{B \rightarrow AD\}$
  - Decomposed into $S_1=\{A,B\}$, $S_2=\{B,C\}$, $S_3=\{C,D\}$

A	B	C	D
a	$\mathbf{b}$	c_1	d_1
a_2	$\mathbf{b}$	c	d_2
a_3	b_3	c	d

$B \rightarrow AD$

A	B	C	D
$\mathbf{a}$	$\mathbf{b}$	c_1	$\mathbf{d_1}$
$\mathbf{a}$	$\mathbf{b}$	c	$\mathbf{d_1}$
a_3	b_3	c	d

Why the Chase Works

No more changes can be made.
There is no row that is fully unsubscripted.

- Example
  - Relation $R(A, B, C, D)$
  - FD's = $\{B \rightarrow AD\}$
  - Decomposed into $S_1=\{A,B\}$, $S_2=\{B,C\}$, $S_3=\{C,D\}$

Thus, this decomposition does not have a lossless join.

A	B	C	D
a	$\mathbf{b}$	c_1	d_1
a_2	$\mathbf{b}$	c	d_2
a_3	b_3	c	d

$B \rightarrow AD$

A	B	C	D
$\mathbf{a}$	$\mathbf{b}$	c_1	$\mathbf{d_1}$
$\mathbf{a}$	$\mathbf{b}$	c	$\mathbf{d_1}$
a_3	b_3	c	d

Why the Chase Works

- Example
  - Relation $R(A, B, C, D)$
  - FD's = $\{B \rightarrow AD\}$
  - Decomposed into $S_1=\{A,B\}$, $S_2=\{B,C\}$, $S_3=\{C,D\}$

No more changes can be made.
There is no row that is fully unsubscripted.

Thus, this decomposition does not have a lossless join.

A	B	C	D
a	b	c_1	d_1
a	b	c	d_1
a_3	b_3	c	d

⇒

A	B
a	b
a_3	b_3

$\bowtie$

B	C
b	c_1
b	c
b_3	c

$\bowtie$

C	D
c_1	d_1
c	d_1
c	d

Why the Chase Works

No more changes can be made.
There is no row that is fully unsubscripted.

- Example

A	B	C	D
a	b	c_1	d_1
a	b	c	d_1
a_3	b_3	c	d

Thus, this decomposition does not have a lossless join.

A	B
a	b
a_3	b_3

$\bowtie$

B	C
b	c_1
b	c
b_3	c

$\bowtie$

C	D
c_1	d_1
c	d_1
c	d

$=$

A	B	C	D
a	b	c_1	d_1
a	b	c	d_1
$\mathbf{a}$	$\mathbf{b}$	$\mathbf{c}$	$\mathbf{d}$
a_3	b_3	c	d_1
a_3	b_3	c	d

Dependency Preservation
保持依赖

- In some cases
  - It is not possible to decompose a relation into BCNF relations
  - that have both the lossless-join and dependency-preservation

Dependency Preservation

- Example
  - We need to make a tradeoff between preserving dependencies and BNCF
  - Relation Bookings(title, theater, city)
    - title, the name of a movie
    - theater, the name of a theater where the movie is being shown
    - city, the city where the theater is located

Dependency Preservation

- Example
  - We need to make a tradeoff between preserving dependencies and BNCF
  - Relation Bookings(title, theater, city)

    theater → city

    title city → theater

  Keys?

Dependency Preservation

- Example
  - We need to make a tradeoff between preserving dependencies and BNCF
  - Relation Bookings(title, theater, city)

    theater → city

    title city → theater

  Keys:

    {title, city}

    {theater, title} BCNF violation?

Dependency Preservation

- Example
  - We need to make a tradeoff between preserving dependencies and BNCF
  - Relation Bookings(title, theater, city)

    theater $\rightarrow$ city BCNF violation!

    title city $\rightarrow$ theater

  Keys:

    {title, city}

    {theater, title}

Dependency Preservation

- Example
  - We need to make a tradeoff between preserving dependencies and BNCF
  - Relation Bookings(title, theater, city)

    theater → city   BCNF violation!

    title city → theater

  Decomposition:

    {theater, city}

    {theater, title}

Dependency Preservation

- Example
  - We need to make a tradeoff between preserving dependencies and BNCF
  - Relation Bookings(title, theater, city)

    theater → city

    title city → theater

  Decomposition:

    {theater, city}

    {theater, title}

theater	*city*
Guild	Menlo Park
Park	Menlo Park

theater	*title*
Guild	Antz
Park	Antz

Dependency Preservation

- Example
  - Relation Bookings(title, theater, city)

    theater → city

    title city → theater

    Decomposition:

    {theater, city}

    {theater, title}

theater	*city*
Guild	Menlo Park
Park	Menlo Park

theater → city satisfy

theater	*title*
Guild	Antz
Park	Antz

title city → theater Violate

theater	*city*	*title*
Guild	Menlo Park	Antz
Park	Menlo Park	Antz

Outline

- Relation Decomposition
- Third Normal Form
- Multivalued Dependencies

Third Normal Form 第三范式

- Definition
  - A relation R is in third normal form (3NF) if:
    - Whenever $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is a nontrivial FD,
    - either $\{A_1, A_2, \ldots, A_n\}$ is a superkey,
    - or those of $B_1, B_2, \ldots, B_m$ that are not among the A's,
    - are each a member of some key
      (not necessarily the same key)

Third Normal Form

- Prime　主属性
  - An attribute that is a member of some key is said to be prime.
- 3NF
  - For each nontrivial FD,
    - either the left side is a superkey,
    - or the right side consists of prime attributes only.

First Normal Form 第一范式

- 1NF
  - All attribute values in a relation *R* are atomic (cannot be decomposed into smaller pieces)
  - A relation *R* must have a key, then *R* is in 1NF

"原子、有键"

姓名	课号列表
张三	数据库、操作系统、编译原理
李四	数据库、操作系统

课号列表 不是原子属性
不符合第一范式

姓名	课号
张三	数据库
张三	操作系统
张三	编译原理
李四	数据库
李四	操作系统

Second Normal Form 第二范式

- 2NF
  - A 1NF relation *R* is in 2NF if and only if
  - all its non-prime attributes are functionally dependent on the whole of every keys (rather than just a part of them)

  "无部分函数依赖"
  - Note:
    - When a 1NF relation has no composite keys, the relation is automatically in 2NF.

Second Normal Form 第二范式

- Example
  - The redundancy makes the table vulnerable to update anomalies.

Employees' Skills

Employee	Skill	Current Work Location
Jones	Typing	114 Main Street
Jones	Shorthand	114 Main Street
Jones	Whittling	114 Main Street
Bravo	Light Cleaning	73 Industrial Way
Ellis	Alchemy	73 Industrial Way
Ellis	Flying	73 Industrial Way
Harrison	Light Cleaning	73 Industrial Way

Second Normal Form 第二范式

- Example
  - Neither of these tables suffers from update anomalies.

Employees

Employee	Current Work Location
Jones	114 Main Street
Bravo	73 Industrial Way
Ellis	73 Industrial Way
Harrison	73 Industrial Way

Employees' Skills

Employee	Skill
Jones	Typing
Jones	Shorthand
Jones	Whittling
Bravo	Light Cleaning
Ellis	Alchemy
Ellis	Flying
Harrison	Light Cleaning

Second Normal Form 第二范式

- Example
  - 2NF and the primary key

Electric Toothbrush Models

Manufacturer	**Model**	**Model Full Name**	**Manufacturer Country**
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZBrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch	Toothmaster	Hoch Toothmaster	Germany
Hoch	X-Prime	Hoch X-Prime	Germany

Second Normal Form 第二范式

- Example
  - – 2NF and the primary key

Electric Toothbrush Manufacturers

Manufacturer	Manufacturer Country
Forte	Italy
Dent-o-Fresh	USA
Kobayashi	Japan
Hoch	Germany

Electric Toothbrush Models

Manufacturer	Model	Model Full Name
Forte	X-Prime	Forte X-Prime
Forte	Ultraclean	Forte Ultraclean
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZBrush
Kobayashi	ST-60	Kobayashi ST-60
Hoch	Toothmaster	Hoch Toothmaster
Hoch	X-Prime	Hoch X-Prime

Second Normal Form 第二范式

- Example
  - Not all 2NF tables are free from update anomalies.

Tournament Winners

Tournament	Year	Winner	Winner Date of Birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

  - Why?

[E.F. Codd, Further Normalization of the Data Base Relational Model. 1971]

Third Normal Form 第三范式

- 3NF (Codd's definition)
  - Relation *R* is in 2NF
  - Every non-prime attribute of *R* is non-transitively dependent on every key of *R*.

  "无传递依赖"

Tournament Winners

Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Player Dates of Birth

Player	Date of Birth
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

[C. Zaniolo. A New Normal Form for the Design of Relational Database Schemata. ACM Transactions on Database Systems 7(3), 1982.]

Third Normal Form 第三范式

- 3NF (Zaniolo's definition)
  - A relation is in 3NF if and only if,
  - for each of its FD's $X \rightarrow A$, at least one of the following conditions holds:
    - X contains A (i.e., $X \rightarrow A$ is a trivial FD), or
    - X is a superkey, or
    - $A - X$, the set difference between A and X is a prime attribute (i.e., $A - X$ is contained within a key)

Third Normal Form 第三范式

- Derivation of Zaniolo's conditions
  - Let $X \to A$ be a nontrivial FD
  - and let A be a non-key attribute.
  - Also let Y be a key of R. Then $Y \to X$.
  - Therefore A is not transitively dependent on Y if and only if $X \to Y$,
    that is, if and only if X is a superkey.

Witness

- I promise to tell

  the truth,

  the whole truth and

  nothing but the truth,

  so help me God.

Nothing but the key

- 1NF: the key
  - Requiring existence of “the key”
- 2NF: the whole key
  - Requiring the non-prime attributes be dependent on “the whole key”
- 3NF: nothing but the key
  - Requiring the non-prime attributes be dependent on “nothing but the key”

A Memorable Summary

- Non-prime attributes depend on

  the key,

  the whole key, and

  nothing but the key,

  so help me Codd.

The Synthesis Algorithm for 3NF

- How we decompose a relation *R* into a set of relations such that:
  - a) The relations of the decomposition are all in 3NF.
  - b) The decomposition has a lossless join.
  - c) The decomposition has the dependency-preservation property.

The Synthesis Algorithm for 3NF

Algorithm 3.26

Input: A relation R and a set F of FD's that hold for R.

Output: A decomposition of R into a collection of relations, each of which is in 3NF. The decomposition has the lossless-join and dependency-preservation properties.

Method: Perform the following steps:

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

The Synthesis Algorithm for 3NF

- Example
  - $R(A, B, C, D, E)$
  - FD's = { $AB \rightarrow C$, $C \rightarrow B$, $A \rightarrow D$ }
- Answer
  - The given FD's are their own minimal basis (p103-104)
  - $S_1(A, B, C)$, $S_2(B, C)$, $S_3(A, D)$ (drop S_2)
  - $S_1(A, B, C)$, $S_3(A, D)$

The Synthesis Algorithm for 3NF

- Example
  - $R(A, B, C, D, E)$
  - FD's = { $AB \rightarrow C$, $C \rightarrow B$, $A \rightarrow D$ }
- Answer
  - The given FD's are their own minimal basis
  - $S_1(A, B, C)$, $S_3(A, D)$
  - Keys: $\{A, B, E\}$ and $\{A, C, E\}$
  - The final decomposition of R:
    $S_1(A, B, C)$, $S_3(A, D)$, $S_4(A, B, E)$

Why the 3NF Synthesis Algorithm Works

- We need to show three things
  1. Lossless Join
     - Start with a relation of the decomposition whose set of attributes K is a superkey.
     - Consider the sequence of FD's that are used in Algorithm 3.7 to expand K to become K^+. Since K is a superkey, we know K^+ is all the attributes.
     - The same sequence of FD applications on the tableau cause the subscripted symbols in the row corresponding to K to be equated to unsubscripted symbols in the same order as the attribute were added to the closure.
     - Thus, the chase test concludes that the decomposition is lossless.

Why the 3NF Synthesis Algorithm Works

- We need to show three things

  2. Dependency Preservation

     - Each FD of the minimal basis has all its attributes in some relation of the decomposition.
     - Thus, each dependency can be checked in the decomposed relations.

Why the 3NF Synthesis Algorithm Works

- We need to show three things

  3. Third Normal Form

     - If we have to add a relation whose schema is a key, then this relation is surely in 3NF.
       (All attributes of this relation are prime.)
     - For the relations whose schemas are derived from the FD's of a minimal basis, the proof that they are in 3NF is beyond the scope of our course.
       (The argument involves showing that a 3NF violation implies that the basis is not minimal.)

Outline

- Relation Decomposition
- Third Normal Form
- Multivalued Dependencies

Multivalued Dependencies

- Multivalued Dependency 多值依赖
  - An assertion that two attributes or sets of attributes are independent of one another
  - A generalization of a functional dependency

Attribute Independence

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

- The only way to express the fact
  - that addresses and movies are **independent**
  - have each address appear with each movie
- Obvious redundancy
  - When we repeat address and movie facts in all combinations

Definition of Multivalued Dependencies

- Definition
  - A **multivalued dependency** (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

Definition of Multivalued Dependencies

- Definition
  - More precisely, we say the MVD

$$A_1A_2\ldots A_n \to\to B_1B_2\ldots B_m$$

  - holds for a relation R if when we restrict ourselves to the tuples of R that have particular values for each of the attributes among the A's,
  - then the set of values we find among the B's is independent of the set of values we find among the attributes of R that are not among the A's or B's.

Definition of Multivalued Dependencies

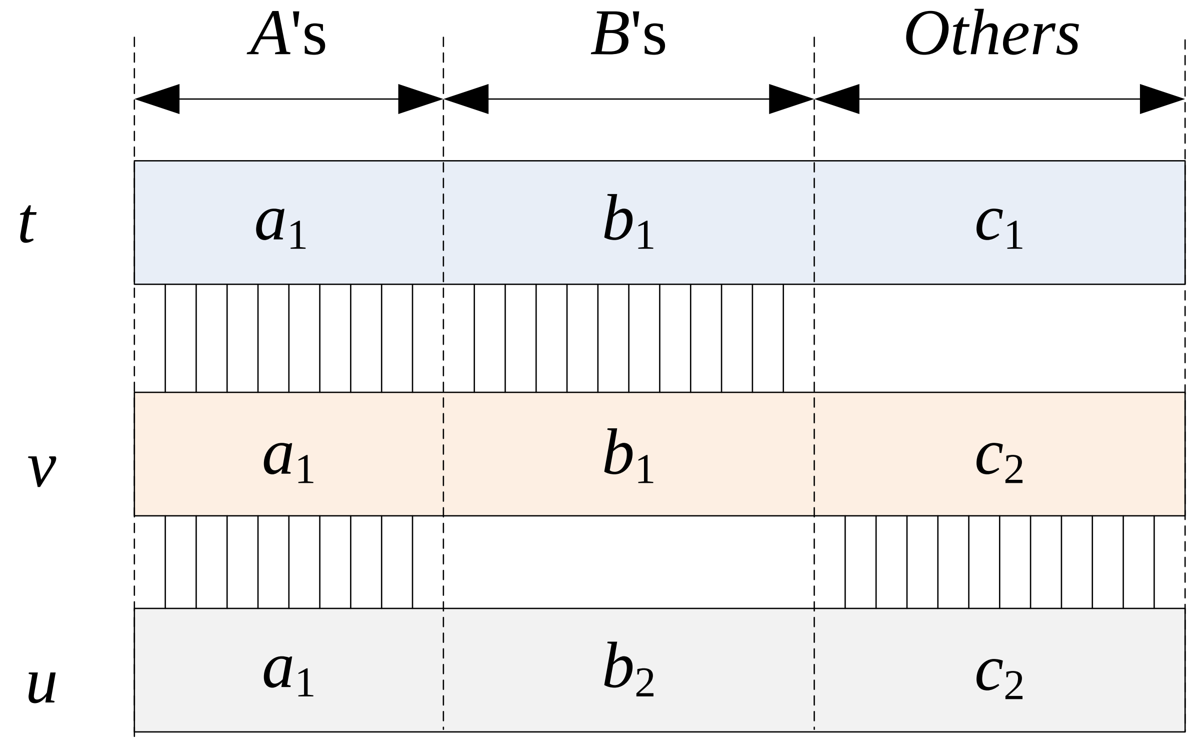
- Definition
  - Still more precisely, we say this MVD holds if
  - For each pair of tuples *t* and *u* of relation *R* that agree on all the *A*'s, we can find in *R* some tuple *v* that agrees:
    1. With both *t* and *u* on the *A*'s,
    2. With *t* on the *B*'s, and
    3. With *u* on all attributes of *R* that are not among the *A*'s or *B*'s.

Definition of Multivalued Dependencies

- Definition
  - As a consequence,
    - For any fixed values of the *A*'s,
    - the associated values of the *B*'s and
    - the other attributes appear in
    - all possible combinations in different tuples.

Definition of Multivalued Dependencies

- How *v* relates to *t* and *u* when an MVD holds.



Definition of Multivalued Dependencies

- Example
  - An MVD

$$\text{name} \rightarrow\rightarrow \text{street city}$$

	name	*street*	*city*	*title*	*year*
t	C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
u	C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980

Definition of Multivalued Dependencies

- Example
  - An MVD

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
u	C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980

Definition of Multivalued Dependencies

- Example
  - An MVD

name $\rightarrow\rightarrow$ street city

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

Definition of Multivalued Dependencies

- Formal definition
  [Silberschatz. Database System Concepts (5th ed.). p. 295]
  - Let R be a relation schema and $\alpha \subseteq R$ and $\beta \subseteq R$
  - The multivalued dependency $\alpha \rightarrow\rightarrow \beta$
    holds on R if, in any legal relation $r(R)$, for all pairs of tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r such that

Definition of Multivalued Dependencies

- Formal definition
  - The multivalued dependency $\alpha \rightarrow\rightarrow \beta$ holds on R if, in any legal relation $r(R)$, for all pairs of tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r such that

$$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_3[\alpha]$$

$$t_3[\beta] = t_1[\beta]$$

$$t_3[R - \beta] = t_2[R - \beta]$$

$$t_4[\beta] = t_2[\beta]$$

$$t_4[R - \beta] = t_1[R - \beta]$$

Definition of Multivalued Dependencies

- In more simple words
  - The above condition can be expressed as follows:
    - If we denote by (x, y, z) the tuple having values for
    - α,β, R –α–β collectively equal to x, y, z, correspondingly, then whenever the tuples

      (a, b, c) and (a, d, e) exist in r, the tuples

      (a, b, e) and (a, d, c) should also exist in r.

Reasoning About MVD's

- Trivial MVD's
  - The MVD $A_1A_2\ldots A_n \rightarrow\rightarrow B_1B_2\ldots B_m$ holds in any relation if $\{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\}$

Reasoning About MVD's

- Transitive Rule
  - If $A_1A_2\ldots A_n \rightarrow\rightarrow B_1B_2\ldots B_m$ and $B_1B_2\ldots B_m \rightarrow\rightarrow C_1C_2\ldots C_k$
  - then so does $A_1A_2\ldots A_n \rightarrow\rightarrow C_1C_2\ldots C_k$
  - MVD's do not obey the splitting part of the splitting/combining rule (p. 109)

Reasoning About MVD's

- FD Promotion
  - Every FD is a MVD. That is, if $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ then $A_1A_2\ldots A_n \rightarrow\rightarrow B_1B_2\ldots B_m$
  - Why? (p. 109)

Reasoning About MVD's

- Complementation Rule
  - If $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is an MVD for relation R, then R also satisfies

    $A_1A_2\ldots A_n \rightarrow\rightarrow C_1C_2\ldots C_k$

    where the C's are all attributes of R not among the A's and B's.

Reasoning About MVD's

- More Trivial MVD's
  - If all the attributes of relation R are

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

    then

    $A_1A_2\ldots A_n \twoheadrightarrow B_1B_2\ldots B_m$ holds in R

  - Why? (p. 110)

Fourth Normal Form

- Fourth Normal Form 4NF 第四范式
  - A relation R is in fourth normal form (4NF) if whenever

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

    is a nontrivial MVD, $\{A_1, A_2, \ldots, A_n\}$ is a superkey.

Fourth Normal Form

- Example

  name $\rightarrow\rightarrow$ street city

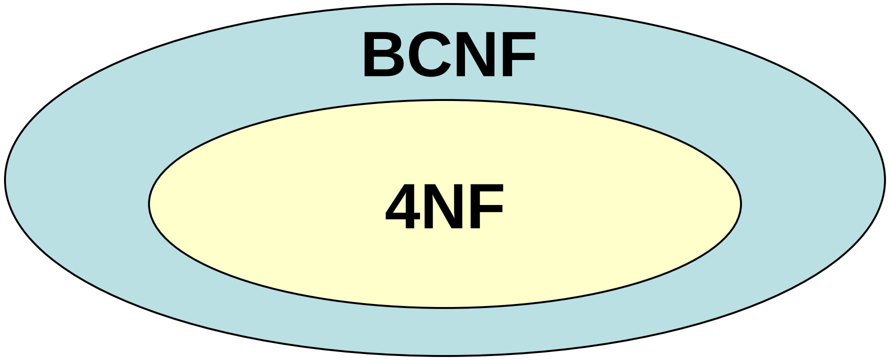
  is a nontrivial MVD,

  name by itself is not a superkey.

  The only key is all the attributes.

Fourth Normal Form

- Generalization
  - 4NF is a generalization of BCNF
  - Every FD is also a MVD
  - Every BCNF violation is also a 4NF violation
  - Every relation is in 4NF is therefore in BCNF



Decomposition into 4NF

Algorithm 3.33: Decomposition into 4NF

(quite analogous to the BCNF decomposition algorithm)

INPUT: A relation R_0 with a set of functional and multivalued dependencies S_0.

OUTPUT: A decomposition of R_0 into relations all of which are in 4NF. The decomposition has the lossless-join property.

METHOD: Do the following steps, with R=R_0 and S=S_0:

1. Find a 4NF violation in R, say $A_1A_2\ldots A_n \to\to B_1B_2\ldots B_m$, where $\{A_1, A_2, \ldots, A_n\}$ is not a superkey.
Not this MVD could be a true MVD in S, or it could be derived from the corresponding FD $A_1A_2\ldots A_n \to B_1B_2\ldots B_m$ in S, since every FD is an MVD

If there is none, return; R by itself is a suitable decomposition.

Decomposition into 4NF

Algorithm 3.33: Decomposition into 4NF

INPUT: A relation R_0 with a set of functional and multivalued dependencies S_0.

OUTPUT: A decomposition of R_0 into relations all of which are in 4NF. The decomposition has the lossless-join property.

METHOD: Do the following steps, with R=R_0 and S=S_0:

2. If there is such a 4NF violation, break the schema for the relation R that has the 4NF violation into two schemas:

(a) R_1, whose schema is A's and the B's.

(b) R_2, whose schema is the A's and all attributes of R that are not among the A's or B's.

Decomposition into 4NF

Algorithm 3.33: Decomposition into 4NF

INPUT: A relation R_0 with a set of functional and multivalued dependencies S_0.

OUTPUT: A decomposition of R_0 into relations all of which are in 4NF. The decomposition has the lossless-join property.

METHOD: Do the following steps, with R=R_0 and S=S_0:

3. Find the FD's and MVD's that hold in R_1 and R_2. (Section 3.7 explains how to do this task in general.) Recursively decompose R_1 and R_2 with respect to their projected dependencies.

Decomposition into 4NF

- Example

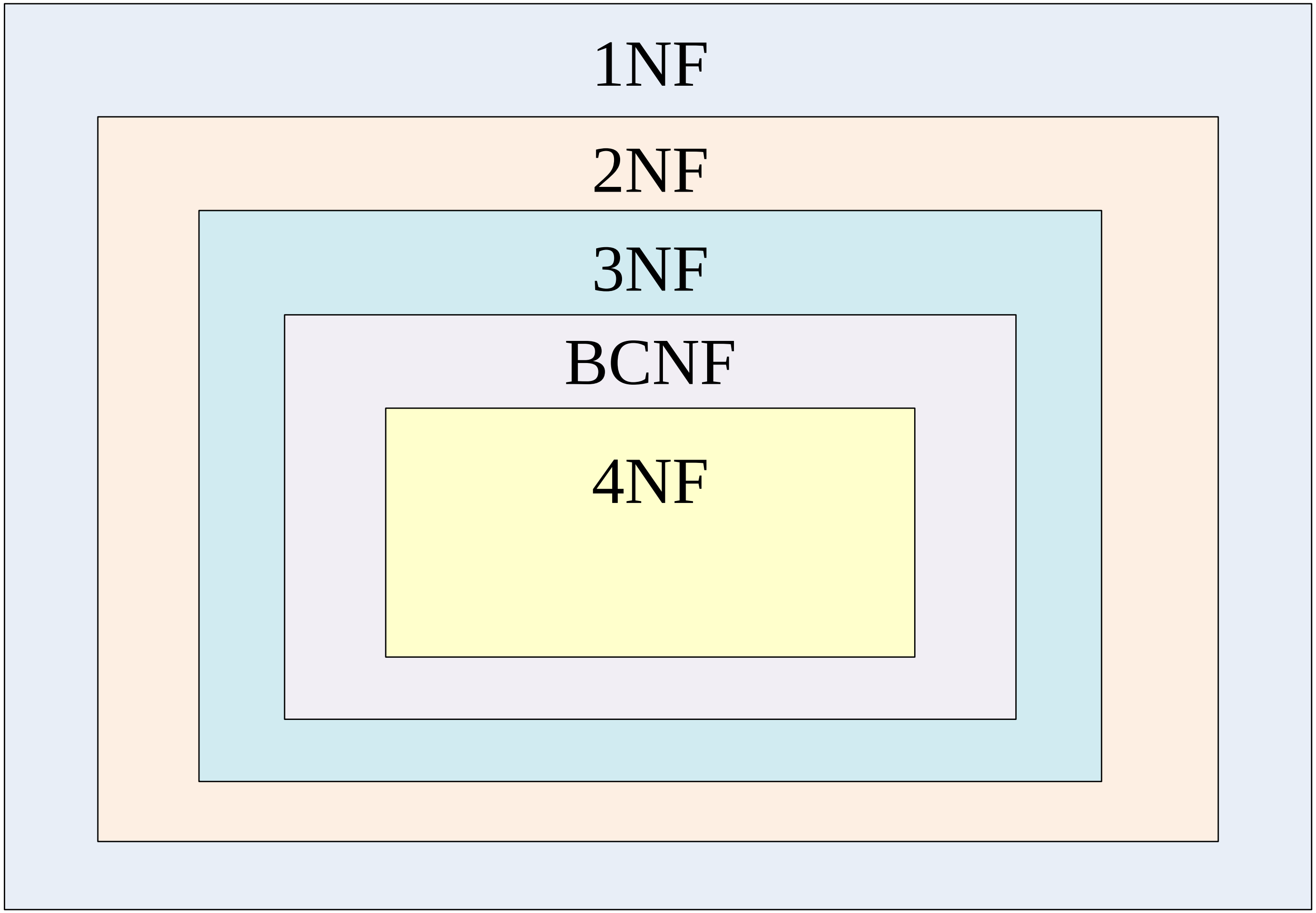
  name →→ street city   4NF violation

  {name, street, city}   in 4NF

  {name, title, year}   in 4NF

  - In each schema there are no nontrivial multivalued (or functional) dependencies,
  - so they are in 4NF.

Relationships Among Normal Forms



Relationships Among Normal Forms

- Properties of normal forms and their decompositions

Property	**3NF**	**BCNF**	**4NF**
Eliminates redundancy due to FD's	No	Yes	Yes
Eliminates redundancy due to MVD's	No	No	Yes
Preserves FD's	Yes	No	No
Preserves MVD's	No	No	No

…The End of This Lecture…