Lecture 3

Relational Database Design Theory

(part 1)

关系数据库设计理论

(第1部分)

Design Theory

- Dependencies 依赖
  - A well developed **theory** for relational databases
  - How to improve an initial relational schema to become a good design?
    - Problems: Anomalies 异常
    - Functional dependencies 函数依赖
      - Normalization: Normal Form 范式
      - Decomposition 分解
    - Multivalued dependencies 多值依赖
      - Eliminate redundancy

Outline

- Functional Dependencies

- Relational Database Schema Design

Outline

- Functional Dependencies
- Relational Database Normal Form

Functional Dependencies

- Functional Dependencies 函数依赖
  - State the constraints that apply to the relation
  - Generalizes the idea of a key for a relation
  - We shall see
    - How this theory gives us simple tools to improve our designs by the process of "decomposition"

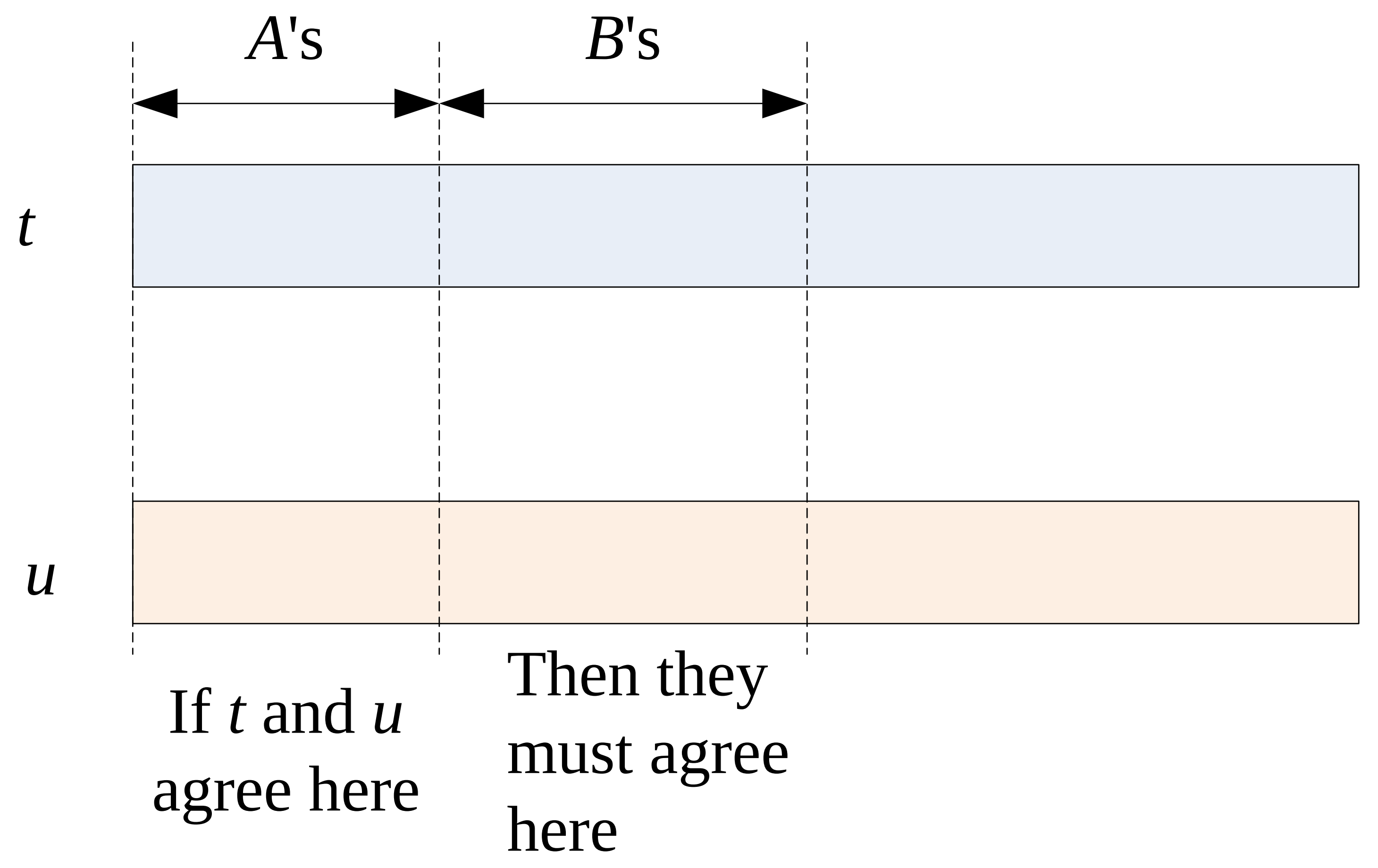
Functional Dependencies: Definition

- A functional dependency (FD)
  - on a relation R is a statement of the form
    - "If two tuples of R agree on all of the attributes $A_1, A_2, \ldots, A_n$,
    - then they must also agree on all of another list of attributes $B_1, B_2, \ldots, B_m$"

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

"$A_1, A_2, \ldots, A_n$ functionally determine $B_1, B_2, \ldots, B_m$"

Functional Dependencies: Definition



Functional Dependencies

- *R* satisfies the FD *F*   *R*满足函数依赖*F*
  - If we can be sure every instance of a relation *R*
    - will be one in which a given FD *F* is true
  - Asserting a constraint on *R*
    - Not just saying something about one particular instance of *R*

Functional Dependencies

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

Is equivalent to the set of FD's:

$$A_1A_2\ldots A_n \rightarrow B_1$$

$$A_1A_2\ldots A_n \rightarrow B_2$$

$$\ldots$$

$$A_1A_2\ldots A_n \rightarrow B_m$$

Functional Dependencies

- Example

  Movies1 (title, year, length, genre, studioName, starName)

  title year $\rightarrow$ length genre studioName ✓

  title year $\rightarrow$ starName ✗

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

Keys of Relations

- Key 键
  - A set of one or more attributes $\{A_1, A_2, \ldots, A_n\}$ is a key for a relation R if
    1. Those attributes functionally determine all other attributes of the relation.
       That is, it is impossible for two distinct tuples of R to agree on all of $A_1, A_2, \ldots, A_n$
    2. No proper subset of $\{A_1, A_2, \ldots, A_n\}$ functionally determines all other attributes of R;
       i.e., a key must be minimal

Keys of Relations

- Example

  Movies1 (title, year, length, genre, studioName, starName)

  - Key

  {title, year, starName}

  - Not a key

  {title, year}

  {year, starName}

  {title, starName}

Primary Key

- Primary Key 主键

- Sometime a relation has more than one key
  – Designate one of the keys as the **primary key**

Superkeys

- Superkey 超键
  – A set of attributes that contains a key
  – Short for "superset of a key"

  – Every superkey satisfies the first condition of a key
    - It functionally determines all other attributes of the relations
  – A superkey need not satisfy the second condition
    - Minimality

Superkeys

- Example

  Movies1 (title, year, length, genre, studioName, starName)

  – Superkeys

  {title, year, starName}

    - Any superset of this set of attributes

  {title, year, starName, length, studioName}

Reasoning About FD's

- Example 3.4

  $R(A, B, C)$ satisfies the FD's

  $A \rightarrow B$ and $B \rightarrow C$

  Does R also satisfy $A \rightarrow C$?

  – Proof
    - Let two tuples agreeing on A be (a, b_1, c_1) and (a, b_2, c_2)
    - Since R satisfies $A \rightarrow B$, then $b_1 = b_2$
    - The tuples are really (a, b, c_1) and (a, b, c_2)
    - Since R satisfies $B \rightarrow C$, then $c_1 = c_2$
    - Any two tuples of R that agree on A also agree on C

Several Definitions

- Equivalent 等价
  - 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*.
- Follows 推断
  - A set of FD's *S* follows from a set of FD's *T* if every 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 if and only if *S* follows from *T*, and *T* follows from *S*.

The Splitting/Combining Rule

- Splitting rule 分解规则
  - Replace an FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$
  - by a set of FD's $A_1A_2\ldots A_n \rightarrow B_i$ for $i=1,2,\ldots,m$
- Combing rule 组合规则
  - Replace a set of FD's $A_1A_2\ldots A_n \rightarrow B_i$ for $i=1,2,\ldots,m$
  - by the single FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$

The Splitting/Combining Rule (Cont'd)

- Example

  title year $\rightarrow$ length

  title year $\rightarrow$ genre

  title year $\rightarrow$ studioName

  – Is equivalent to the single FD

  title year $\rightarrow$ length genre studioName

The Splitting/Combining Rule (Cont'd)

- Example

  title year → length

  – If we try to split the left side into

  title → length

  year → length

  – Then we get two false FD's!

Trivial Functional Dependencies 平凡函数依赖

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

- Trivial FD's 平凡函数依赖

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

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

Trivial Functional Dependencies (Cont'd)

- Trivial-dependency rule 平凡依赖规则

  $A_1A_2...A_n \rightarrow B_1B_2...B_m$

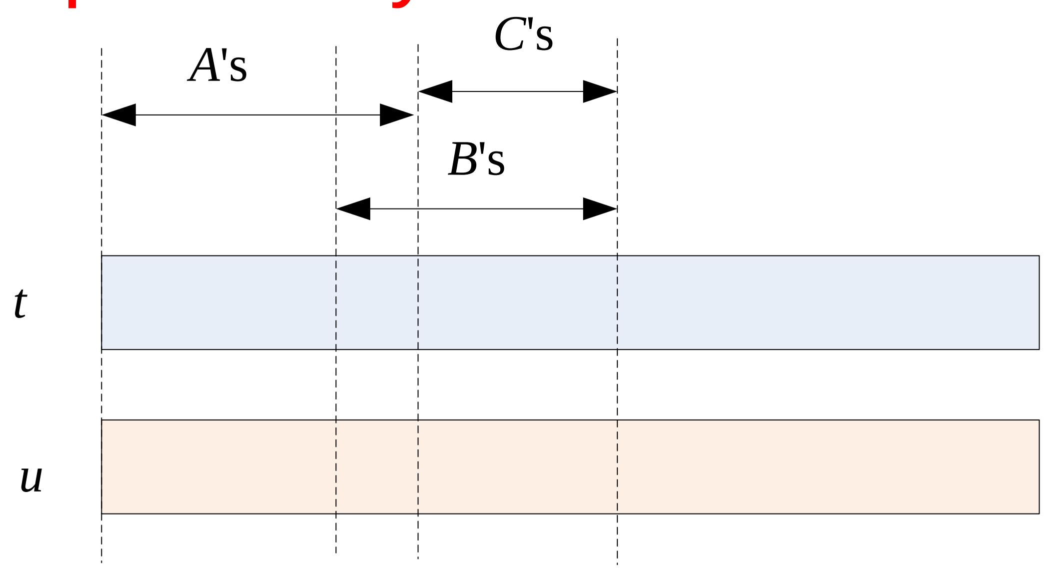
  is equivalent to

  $A_1A_2...A_n \rightarrow C_1C_2...C_k$

  Where the C's are all those B's that are not also A's

Trivial Functional Dependencies (Cont'd)

- Trivial-dependency rule 平凡依赖规则



If *t* and *u* agree on the *A*'s

Then they must agree on the *B*'s

So surely they agree on the *C*'s

Computing the Closure of Attributes 计算属性的闭包

- Closure of attributes 属性的闭包
  - Suppose $\{A_1, A_2, \ldots, A_n\}$ is a set of attributes
  - S is a set of FD's
  - The closure of $\{A_1, A_2, \ldots, A_n\}$ under the FD's in S
    - is the set of attributes B such that every relation that satisfies all the FD's in set S also satisfies

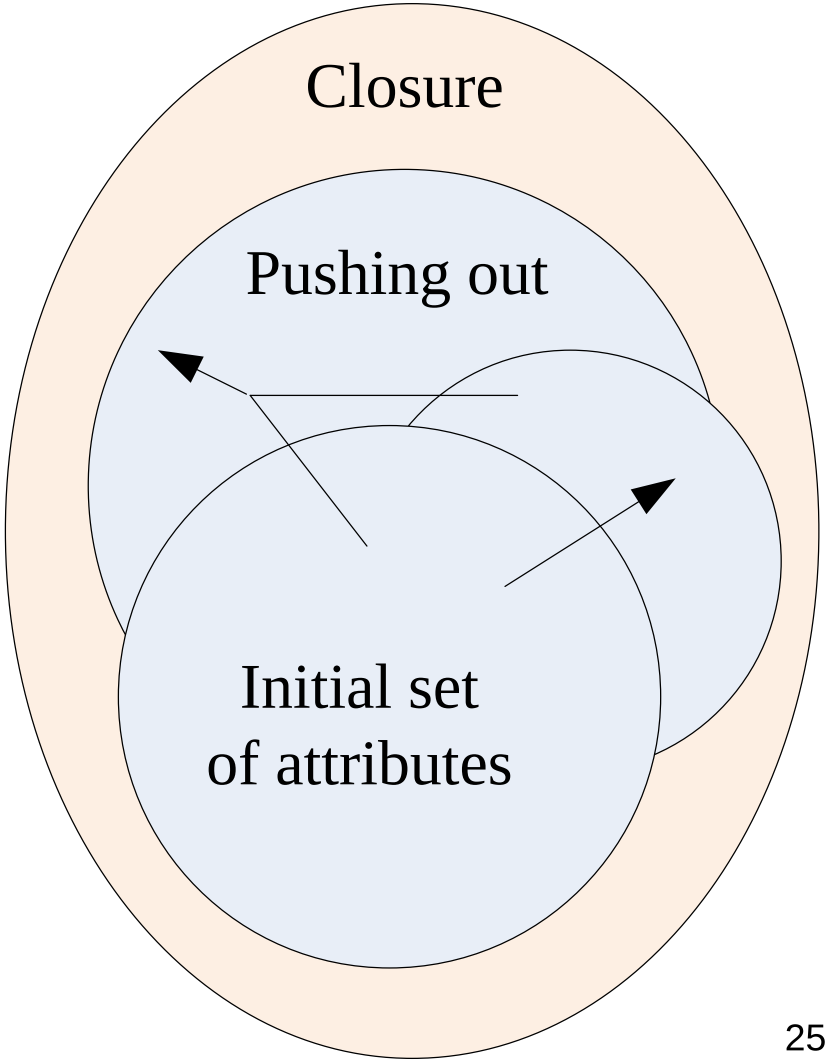
      $A_1A_2\ldots A_n \rightarrow B$

      - That is $A_1A_2\ldots A_n \rightarrow B$ follows from the FD's of S.
    - Denote the closure of a set of attributes $A_1, A_2, \ldots, A_n$ by $\{A_1, A_2, \ldots, A_n\}^+$

$A_1, A_2, \ldots, A_n$ are always in $\{A_1, A_2, \ldots, A_n\}^+$,
Because $A_1A_2\ldots A_n \rightarrow A_i$ is trivial when i is one of $1, 2, \ldots, n$

Computing the Closure of Attributes

- Process
  - Starting with the given set of attributes
  - Repeatedly expand the set by adding the right sides of FD's as soon as we have included their left sides
  - Eventually, we cannot expand the set any further, and the resulting set is the closure.



Algorithm 3.7: Closure of a Set of Attributes

INPUT: A set of attributes $\{A_1, A_2, \ldots, A_n\}$ and a set of FD's S

OUTPUT: The closure $\{A_1, A_2, \ldots, A_n\}^+$

1. If necessary, split the FD's of S, so each FD in S has a single attribute on the right.
2. Let X be a set of attributes that eventually will become the closure. Initialize X to be $\{A_1, A_2, \ldots, A_n\}$.
3. Repeatedly search for some FD

   $B_1 B_2 \ldots B_m \rightarrow C$

   such that all of $B_1, B_2, \ldots, B_m$ are in X, but C is not. Add C to the set X and repeat the search.
4. The set X, after no more attributes can be added to it, is the correct value of $\{A_1, A_2, \ldots, A_n\}^+$

Example

- Example 3.8

  $R(A, B, C, D, E, F)$

  FD's=$\{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B\}$

  $\{A, B\}^+ = ?$

  1. Split $BC \rightarrow AD$ into $BC \rightarrow A$ and $BC \rightarrow D$.

  2. $X = \{A, B\}$

  3. $AB \rightarrow C$, $X = \{A, B, C\}$

     $BC \rightarrow A$ and $BC \rightarrow D$, $X = \{A, B, C, D\}$

     $D \rightarrow E$, $X = \{A, B, C, D, E\}$

  4. $\{A, B\}^+ = \{A, B, C, D, E\}$

Theorem

$A_1A_2\ldots A_n \to B_1B_2\ldots B_m$ follows from set of FD's S if and only if all of B_1, B_2, …, B_m are in

$$\{A_1, A_2, \ldots, A_n\}^+$$

Example

- Example 3.9

  $R(A, B, C, D, E, F)$

  FD's={$AB \rightarrow C$, $BC \rightarrow AD$, $D \rightarrow E$, $CF \rightarrow B$}

  Test whether $AB \rightarrow D$ follows from these FD's?

  Solution:

  Compute $\{A, B\}^+ = \{A, B, C, D, E\}$

  Since D is a member of the closure, $AB \rightarrow D$ does follow.

  Question: Test whether $D \rightarrow A$ follows from these FD's?

The Transitive Rule

- Transitive Rule
  - If $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ and $B_1B_2\ldots B_m \rightarrow C_1C_2\ldots C_k$ hold in relation R, then $A_1A_2\ldots A_n \rightarrow C_1C_2\ldots C_k$ also holds in R.
    - If some of the C's are among the A's, we may eliminate them from the right side by the *trivial-dependencies rule*.

The Transitive Rule (Cont'd)

- Why the transitive rule holds
  - Compute the closure $\{A_1, A_2, \ldots, A_n\}^+$, with respect to the two given FD's
    - $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ tells us that all of $B_1, B_2, \ldots, B_m$ are in $\{A_1, A_2, \ldots, A_n\}^+$
    - Then, we can use the FD $B_1B_2\ldots B_m \rightarrow C_1C_2\ldots C_k$ to add $C_1, C_2, \ldots, C_k$ to $\{A_1, A_2, \ldots, A_n\}^+$
    - We conclude that $A_1A_2\ldots A_n \rightarrow C_1C_2\ldots C_k$ holds

The Transitive Rule (Cont'd)

- Example

title	*year*	*length*	*genre*	*studioName*	*studioAddr*
Star Wars	1977	124	sciFi	Fox	Hollywood
Eight Below	2005	120	drama	Disney	Buena Vista
Wayne's World	1992	95	comedy	Paramount	Hollywood

title year → studioName

studioName → studioAddr

- The transitive rule

title year → studioAddr

Closure and Keys

- Closure and Superkey
  – $\{A_1, A_2, \ldots, A_n\}^+$ is the set of all attributes of a relation if and only if $A_1, A_2, \ldots, A_n$ is a superkey for the relation.
- Closure and Key
  – Test if $A_1, A_2, \ldots, A_n$ is a key for a relation by checking first that $\{A_1, A_2, \ldots, A_n\}^+$ is all attributes,
  – and then checking that, for no set X formed by removing one attribute from $\{A_1, A_2, \ldots, A_n\}$, is X^+ the set of all attributes.

Minimal Basis of FD's

- Basis of FD's
  - If we are given a set of FD's *S*, then any set of FD's equivalent to *S* is said to be a basis for *S*
- Minimal Basis 最小基本集
  - A basis *B* that satisfies three conditions
    - All the FD's in *B* have singleton right sides
    - If any FD is removed from *B*, the result is no longer a basis
    - If for any FD in *B* we remove one or more attributes from the left side of it, the result is no longer a basis

Minimal Basis of FD's

- Example
  - $R(A, B, C)$
    - Each attribute functionally determines the other two
  - The full set of derived FD's
    - $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow A$, $B \rightarrow C$, $C \rightarrow A$, $C \rightarrow B$
    - $AB \rightarrow C$, $AC \rightarrow B$, $BC \rightarrow A$
    - $A \rightarrow BC$
    - $A \rightarrow A$
  - Minimal basis (only shows two of them)
    - $\{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B\}$
    - $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

Armstrong's Axioms

- Armstrong's Axioms   Armstrong公理
  - From which it is possible to derive any FD that follows from a given set

1. Reflexivity   自反律
   - If $\{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\}$, then $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ (trivial FD's)
2. Augmentation   增广律
   - If $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$, then $A_1A_2\ldots A_nC_1C_2\ldots C_k \rightarrow B_1B_2\ldots B_mC_1C_2\ldots C_k$

Armstrong’s Axioms

- Armstrong’s Axioms   Armstrong公理
  - From which it is possible to derive any FD that follows from a given set

  3. Transitivity   传递律
     - If $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ and $B_1B_2\ldots B_m \rightarrow C_1C_2\ldots C_k$ then $A_1A_2\ldots A_n \rightarrow C_1C_2\ldots C_k$

Projecting Functional Dependencies

- Question?
  - Suppose we have a relation R with set of FD's S
  - And we project R by computing $R_1 = \pi_L(R)$
  - What FD's hold in R_1?
- Answer 函数依赖集的投影
  - By computing the projection of functional dependencies S, which is all FD's that:
    - Follow from S, and
    - Involve only attributes of R_1

Projecting Functional Dependencies

Algorithm 3.12 Projecting a Set of Functional Dependencies

INPUT: A relation R and a second relation R_1 computed by the projection $R_1 = \pi_L(R)$. Also, a set of FD's S that hold in R

OUTPUT: The set of FD's that hold in R_1

METHOD:

1. Let T be the eventual output set of FD's. Initially, T is empty.
2. For each set of attributes X that is a subset of the attributes of R_1, compute X^+. This computation is performed with respect to the set of FD's S, and may involve attributes that are in the schema of R but not R_1. Add to T all nontrivial FD's $X \rightarrow A$ such that A is both in X^+ and an attribute of R_1

Projecting Functional Dependencies

Algorithm 3.12 Projecting a Set of Functional Dependencies

3. Now, T is a basis for the FD's that hold in R_1, but may not be a minimal basis. We may construct a minimal basis by modifying T as follows:
   (a) If there is an FD F in T that follows from the other FD's in T, remove F from T.
   (b) Let $Y \rightarrow B$ be an FD in T, with at least two attributes in Y, and let Z be Y with one of its attributes removed. If $Z \rightarrow B$ follows from the FD's in T (including $Y \rightarrow B$), then replace $Y \rightarrow B$ by $Z \rightarrow B$
   (c) Repeat the above steps in all possible ways until no more changes to T can be made

Projecting Functional Dependencies

- Example
  - Suppose $R(A, B, C, D)$ has FD's
  - $A \rightarrow B$, $B \rightarrow C$, and $C \rightarrow D$,
  - $R_1(A, C, D)$
  - Find the FD's for R_1?
    - $\{A\}^+ = \{A, B, C, D\}$
    - $A \rightarrow C$, $A \rightarrow D$ hold in R_1
    - $\{C\}^+ = \{C, D\}$
    - $C \rightarrow D$ hold in R_1
    - $\{D\}^+ = \{D\}$
    - Add nothing

Projecting Functional Dependencies

- Example
  - Suppose $R(A, B, C, D)$ has FD's
  - $A \rightarrow B$, $B \rightarrow C$, and $C \rightarrow D$,
  - $R_1(A, C, D)$
  - Find the FD's for R_1?

    - $\{C, D\}^+ = \{C, D\}$
    - Add nothing

    - FD's for $R_1 = \{A \rightarrow C, A \rightarrow D, C \rightarrow D\}$

Projecting Functional Dependencies

- Example
  - Suppose $R(A, B, C, D)$ has FD's
  - $A \rightarrow B$, $B \rightarrow C$, and $C \rightarrow D$,
  - $R_1(A, C, D)$
  - Find the FD's for R_1?

    - FD's for $R_1 = \{A \rightarrow C, A \rightarrow D, C \rightarrow D\}$
    - $A \rightarrow D$ follows from the other two (by transitivity)

    - Equivalent set of FD's for $R_1 = \{A \rightarrow C, C \rightarrow D\}$
    - A minimal basis for the FD's of R_1

Outline

- Functional Dependencies
- Relational Database Normal Form

Design of Relational Database Schemas

- Problem
  - Careless selection of a relational database schema
    - Can lead to redundancy and related anomalies.

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

Anomalies

- Anomalies 异常
  - Redundancy 冗余
    - Information may be repeated unnecessarily in several tuples
  - Update Anomalies 更新异常
    - We may change information in one tuple but leave the same information unchanged in another
  - Deletion Anomalies 删除异常
    - If a set of values becomes empty, we may lose other information as a side effect

Decomposing Relations

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

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

Decomposing Relations

- Example

Movies1

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

Movies2

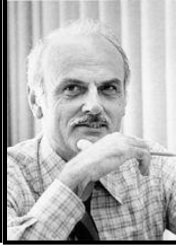
title	year	length	genre	studioName
Star Wars	1977	124	SciFi	Fox
Gone With the Wind	1939	231	drama	MGM
Wayne's World	1992	95	comedy	Paramount

Movies3

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

Boyce-Codd Normal Form

Boyce



- BCNF (Boyce-Codd Normal Form) (BC范式)

  - A relation R is in **BCNF** if and only if
  - whenever there is a nontrivial FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ for R,
  - it is the case that $\{A_1, A_2, \ldots, A_n\}$ is a **superkey** for R

  - **The left side of every nontrivial FD must contain a key**

Boyce-Codd Normal Form

- Example
  - Key {title, year, starName}
  - title year → length genre studioName

Movies1

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

- Example
  - Key {title, year, starName}
  - title year → length genre studioName

Movies1

Movies1 is not in BCNF

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

- Example
  - Key {title, year}
  - title year → length genre studioName

Movies2

title	year	length	genre	studioName
Star Wars	1977	124	SciFi	Fox
Gone With the Wind	1939	231	drama	MGM
Wayne’s World	1992	95	comedy	Paramount

Boyce-Codd Normal Form

- Example
  - Key {title, year}
  - title year → length genre studioName

Movies2 is in BCNF

Movies2

title	year	length	genre	studioName
Star Wars	1977	124	SciFi	Fox
Gone With the Wind	1939	231	drama	MGM
Wayne's World	1992	95	comedy	Paramount

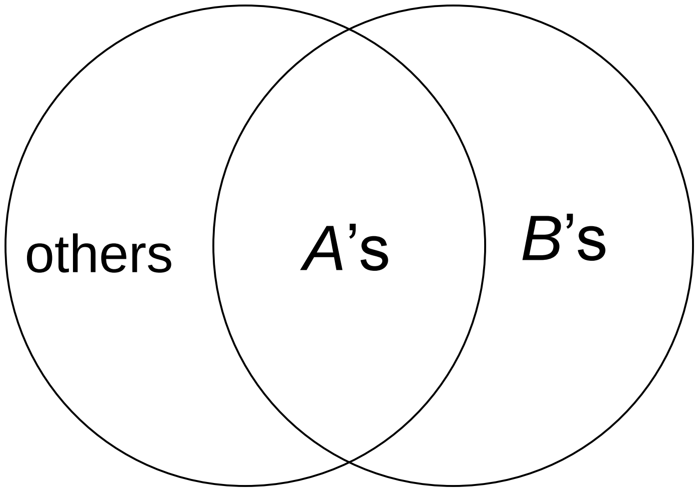
Boyce-Codd Normal Form

- Example
  - Any two-attribute relation is in BCNF
  - $R(A, B)$

1. $\{A, B\}$ is the only key. There are no nontrivial FD's
2. $\{A\}$ is the only key. $A \rightarrow B$
3. $\{B\}$ is the only key. $B \rightarrow A$
4. Both $\{A\}$ and $\{B\}$ are keys. $A \rightarrow B$ and $B \rightarrow A$

Decomposition into BCNF

- The strategy
  - Use the violating FD's to guide our decomposition
  - Look for a nontrivial FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ that violates BCNF

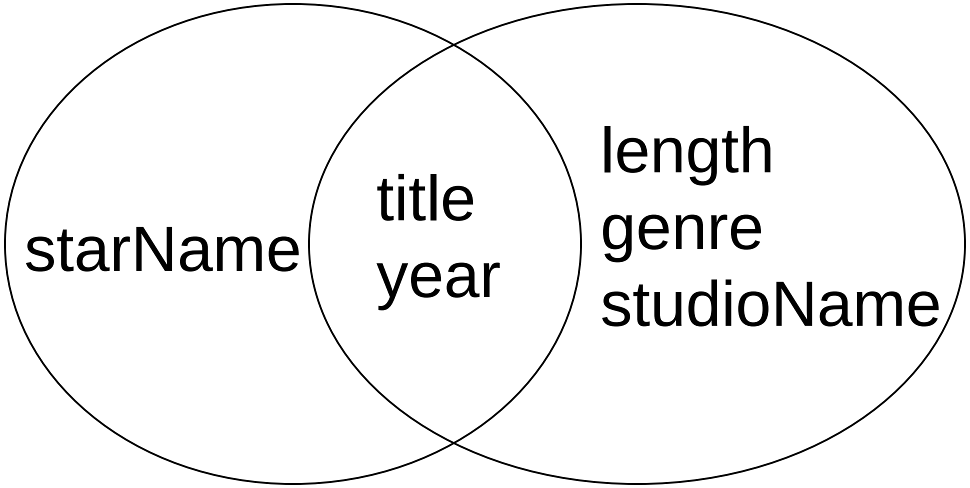


Decomposition into BCNF

- Example
  - title year → length genre studioName

  is a BCNF violation

  1. {title, year, length, genre, studioName}
  2. {title, year, starName}



Decomposition into BCNF

- Example
  - {title, year, studioName, president, presAddr}
    - title year → studioName
    - studioName → president
    - president → presAddr
    - {title, year} the only key
    - The last two FD's above violate BCNF

Decomposition into BCNF

- Example
  - {title, year, studioName, president, presAddr}
    - studioName → president presAddr
  - {title, year, studioName} key {title, year}
    - title year → studioName
  - {studioName, president, presAddr} key{studioName}
    - studioName → president
    - president → presAddr violate BCNF

Decompose again

Decomposition into BCNF

- Example
  - {title, year, studioName} key {title, year}
    - title year → studioName
  - {studioName, president} key {studioName}
    - studioName → president
  - {president, presAddr} key{president}
    - president → presAddr

All 3 relations in BCNF

Decomposition into BCNF

Algorithm 3.20: 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 of which are in BCNF.

METHOD: The following steps can be applied recursively to any relation R and set of FD's S.
Initially, apply them with R=R_0 and S=S_0

1. Check whether R is in BCNF. If so, nothing more needs to be done. Return $\{R\}$ as the answer.

Decomposition into BCNF

Algorithm 3.20: 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 of which are in BCNF.

METHOD: The following steps can be applied recursively to any relation R and set of FD's S.
Initially, apply them with $R=R_0$ and $S=S_0$.

2. If there are BCNF violations, let one be $X \rightarrow Y$. Use Algorithm 3.7 to compute X^+. Choose $R_1=X^+$ as one relation schema and let R_2 have attributes X and those attributes of R that are not in X^+

Decomposition into BCNF

Algorithm 3.20: 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 of which are in BCNF.

METHOD: The following steps can be applied recursively to any relation R and set of FD's S.
Initially, apply them with R=R_0 and S=S_0.

3. Use Algorithm 3.12 to compute the sets of FD's for R_1 and R_2; let these be S_1 and S_2, respectively.

Decomposition into BCNF

Algorithm 3.20: 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 of which are in BCNF.

METHOD: The following steps can be applied recursively to any relation R and set of FD's S.
Initially, apply then with $R=R_0$ and $S=S_0$.

4. Recursively decompose R_1 and R_2 using the algorithm. Return the union of the result of the decompositions.