Advanced DB

CHAPTER 7
RELATIONAL DB DESIGN

Chapter 7: Relational Database Design

- First Normal Form
- Pitfalls in Relational Database Design
- Functional Dependencies
- Decomposition
- Boyce-Codd Normal Form
- Third Normal Form
- Multivalued Dependencies and Fourth Normal Form
- Overall Database Design Process

First Normal Form

- Domain is atomic if its elements are considered to be indivisible units
  - Examples of non-atomic domains:
    - set of names, composite attributes
    - identification numbers like CS101 that can be broken up into parts
- A relational schema is in *first normal form (1NF)* if the domains of all attributes are atomic
- Atomicity is actually a property of how the elements of the domain are used
  - Student ID numbers: *CS0012, EE1127, …*
  - If the first two characters are extracted to find the department => the domain is not atomic
- Non-atomic attributes
  - leads to encoding of information in application program rather than in the database
  - complicate storage and query processing
- We assume all relations are in first normal form

Example

Lending-schema = (*branch-name, branch-city, assets,*
customer-name, loan-number, amount)

branch-name	*branch-city*	*assets*	*customer-name*	*loan-number*	*amount*
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	9000000	Jackson	L-14	1500

- Redundancy:
  - Data for *branch-name, branch-city,* assets are repeated for each loan that a branch makes
  - Wastes space
  - Complicates updating, introducing possibility of inconsistency of *assets* value
- Null values
  - Can use null values, but they are difficult to handle.

Redundancy creates problems

- Anomalies (by Codd)
  - *Insertion anomaly*: cannot store information about a branch if no loans exist
  - *Deletion anomaly*: lose branch info when that last account for the branch is deleted
  - *Update anomaly*: what happens when you modify asset for a branch in only a single record?
- The problems are caused by redundancy!
- Solution => decompose schema so that each information content is represented only once (later)
  - information content: relationship between attributes

Relational Theory

- Goal: Devise a theory for the following
- Decide whether a particular relation R is in "good" form.
- In the case that a relation R is not in "good" form, decompose it into a set of relations $\{R_1, R_2, ..., R_n\}$ such that
  - each relation is in good form
  - the decomposition is a lossless-join decomposition
- Our theory is based on:
  - functional dependencies
  - multivalued dependencies (not covered in this semester)

Functional Dependencies

- Constraints on the set of legal relations.
- Require that the value for a certain set of attributes determines uniquely the value for another set of attributes.
- A functional dependency is a generalization of the notion of a *key*.

Functional Dependencies (Cont.)

- Let R be a relation schema

  $$\alpha \subseteq R \quad \textit{and} \quad \beta \subseteq R$$

- The *functional dependency* $\alpha \rightarrow \beta$ holds on R if and only if
  - for any *legal* relations $r(R)$,
  - whenever any two tuples t_1 and t_2 of r agree on the attributes α,
  - they also agree on the attributes β.
  - That is,

    $$t_1[\alpha] = t_2[\alpha] \quad \Rightarrow \quad t_1[\beta] = t_2[\beta]$$

- Example
  - Consider $r(A,B)$ with the following instance of r

1	4
1	5
3	7

  - On this instance, $A \rightarrow B$ does **NOT** hold, but $B \rightarrow A$ does hold

Trivial FD

- *A* functional dependency is trivial if it is satisfied by all instances of a relation
  - *E.g.*
    - *customer-name, loan-number* → *customer-name*
    - *customer-name* → *customer-name*

- Lemma: $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$

Closure of a Set of FDs

- Given a set F of FDs, there are certain other FDs that are logically implied by F
  - E.g. If $A \rightarrow B$ and $B \rightarrow C$, then we can infer that $A \rightarrow C$
- The set of all functional dependencies logically implied by F is the *closure* of F.
- We denote the *closure* of F by F^+
- We can find all of F^+ by applying *Armstrong's Axioms*:
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ **(reflexivity)**
  - if $\alpha \rightarrow \beta$, then $\gamma\ \alpha \rightarrow \gamma\ \beta$ **(augmentation)**
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ **(transitivity)**
- These rules are
  - sound (generate only functional dependencies that actually hold) and
  - complete (generate all functional dependencies that hold).

Example

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H \}$
- some members of F^+
  - $A \rightarrow H$
    - by transitivity from $A \rightarrow B$ *and* $B \rightarrow H$
  - $AG \rightarrow I$
    - by augmenting $A \rightarrow C$ with G, to get $AG \rightarrow CG$
      and then transitivity with $CG \rightarrow I$
  - $CG \rightarrow HI$
    - from $CG \rightarrow H$ *and* $CG \rightarrow I$: "union rule" can be inferred from
      - definition of functional dependencies, or
      - Augmentation of $CG \rightarrow I$ to infer $CG \rightarrow CGI$, augmentation of $CG \rightarrow H$ to infer $CGI \rightarrow HI$, and then transitivity

Decomposition

- Redundancy causes problems: Solution => decompose schema so that each information content is represented only once
- Definition: Let R be a relation scheme

  $\{R_1, ..., R_n\}$ is a decomposition of R if $R = R_1 \cup ... \cup R_n$

  (i.e., all of R's attributes are represented)
- We will deal mostly with binary decomposition:
  - R into $\{R_1, R_2\}$ where $R = R_1 \cup R_2$

student(s_id, name, dept, dept_head, dept_phone, grade)

=> student'(s_id, name, dept, grade)

학과(dept, dept_head, grade)

Lending = (b_name, asset, b_city, loan#, c_name, amount)

=> Branch = (b_name, asset, b_city)

Loan = (loan#, c_name, amount)

Lossy Decomposition

- Careless decomposition leads to loss of information: Lossy decomposition

Lending = (*b_name, asset, b_city, loan#, c_name, amount*)

: anomalies due to repetition of information

=> *Branch* = (*b_name, asset, b_city*)

Loan = (*loan#, c_name, amount*)

- problem: relationship between *loan* and *branch* is lost
- loss of information

=> *Branch* = (*b_name, asset, b_city*)

Loan = (*loan#, c_name, amount, b_city*)

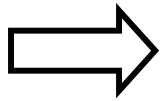
- more tuples in natural join
- but we have lost the relationship
- loss of information

Lossy Decomposition (cont.)

- Decomposition of $R = (A, B)$ into

  $R_1 = (A)$ and $R_2 = (B)$

A	B
α	1
α	2
β	1



A
α
β

$\Pi_A(r)$

B
1
2

$\Pi_B(r)$

- Can we *recover* the *original information content*?

$$\Pi_A(r) \bowtie \Pi_B(r)$$

A	B
α	1
α	2
β	1
β	2

Lossy!

Lossless-join Decomposition

- For $r(R)$ and decomposition $\{R_1, R_2\}$, it is always the case that

$$r \subseteq \Pi_{R1}(r) \bowtie \Pi_{R2}(r)$$

- *Definition:* Decomposition $\{R_1, R_2\}$ is a *lossless-join decomposition* of R if

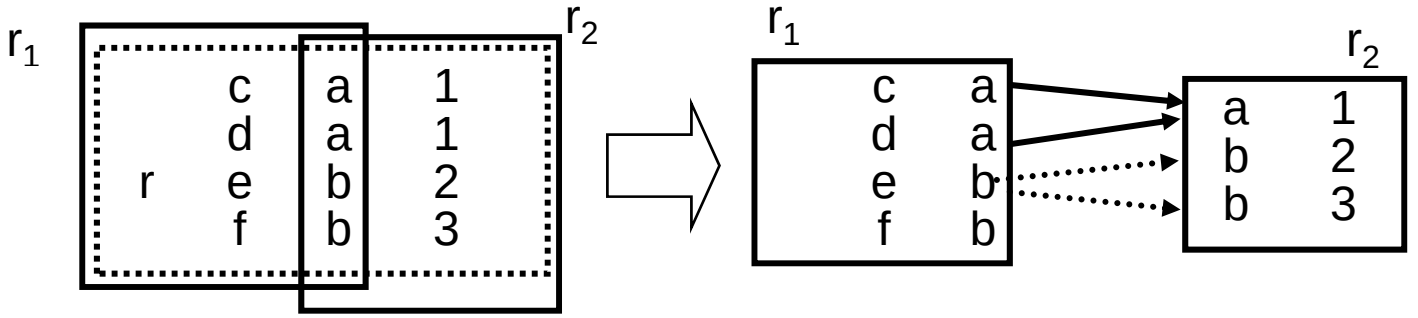
$$r = \Pi_{R1}(r) \bowtie \Pi_{R2}(r)$$

  - The information content of the original relation r is always the basis

- *Lemma:* $\{R_1,...,R_n\}$ is a lossless decomposition if

$$R_1 \cap R_2 \rightarrow R_1, \quad \text{or} \quad R_1 \cap R_2 \rightarrow R_2$$

  - i.e., if one of the two subschemas hold the key of the other subschema



Goal for decomposition

- When we decompose a relation schema R with a set of functional dependencies F into R_1, R_2, …, R_n we want
  1. Lossless decomposition
  2. No redundancy
  3. Dependency preservation

Boyce-Codd Normal Form

- We want a way to decide whether a particular relation R is in "good" form.
- Definition:

  A relation schema R is in BCNF (with respect to a set F of FDs) if for each FD $\alpha \rightarrow \beta$ in F^+ ($\alpha \subseteq R$ and $\beta \subseteq R$), at least one of the following holds:
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R

- Example

  $R = (A, B, C), \quad F = \{A \rightarrow B ; \ B \rightarrow C\}, \quad \text{Key} = \{A\}$
  - R is not in BCNF
  - Decompose into $R_1 = (A, B), \ R_2 = (B, C)$
    - R_1 and R_2 in BCNF
    - Lossless-join decomposition
    - Dependency preserving

Third Normal Form

- Third Normal Form
  - Allows some redundancy (with resultant problems)
  - But FDs can be checked on individual relations without computing a join
  - There is always a lossless-join, dependency-preserving decomposition into 3NF
- A relation schema R is in *third normal form* (*3NF*) if

  for all $\alpha \rightarrow \beta$ in F^+ at least one of the following holds:
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \in \alpha$)
  - α is a superkey for R
  - Each attribute A in $\beta - \alpha$ is contained in a candidate key for R.

    (*NOTE:* each attribute may be in a different candidate key)
- If a relation is in BCNF it is in 3NF (since in BCNF one of the first two conditions above must hold).

Example

$R = (J, K, L)$
$F = \{JK \rightarrow L, L \rightarrow K\}$

- Two candidate keys: JK and JL
- R is in 3NF

  $JK \rightarrow L$ JK is a superkey
  $L \rightarrow K$ K is contained in a candidate key

- There is some redundancy in this schema
- BCNF decomposition has (JL) and (LK)

  => Testing for $JK \rightarrow L$ requires a join

Comparison of BCNF and 3NF

- It is always possible to decompose a relation into relations in 3NF and
  - the decomposition is lossless
  - the dependencies are preserved
- It is always possible to decompose a relation into relations in BCNF and
  - the decomposition is lossless
  - it may *not* be possible to preserve dependencies.

Comparison of BCNF and 3NF (cont.)

R(street, city, zip)

street city → zip

zip → city

St	*Zp*	*C*
s_1	z_1	c_1
s_2	z_1	c_1
s_3	z_2	c_1
null	z_3	c_2

/* 3NF but not in BCNF (nontrivial & zip is not key) */

- repetition of information (e.g., the relationship z_1, c_1)
- need to use null values (e.g., to represent the relationship z_3, c_2 where there is no corresponding value for *St*)

R1(street, zip)

R2(city, zip)

/* violate 하는 FD의 attr로 하나의 relation을 */

Now R1, R2 are in BCNF but not dependency-preserving.

St	*Zp*
s_1	z_1
s_2	z_1
s_3	z_2

Zp	*C*
z_1	c_1
z_2	c_1
z_3	c_2

MultiValued Dependencies

- Basic Concept
  - $R=XYZ$: relation scheme. An MVD $X \rightarrow\rightarrow Y$ holds iff each X-value in R is associated with a set of Y-values in a way that does not depend on Z-values.
  - Formally, for any pair of tuples $t1$, $t2$ of $r(R)$ such that $t1[X]=t2[X]$
    There exists $t3$, $t4$ in r such that
    $t1[X]=t2[X]= t3[X]=t4[X]$
    $t3[Y]=t1[Y]$
    $t3[Z]=t2[Z]$
    $t4[Y]=t2[Y]$
    $t4[Z]=t1[Z]$

Name ->> Child
Name -> Salary

Name	***Child***	***Salary***
A	*B*	1000
A	*C*	1000
A	*D*	1000
F	*G*	1200
F	*H*	1200

MVD

- Note
  - If $X \rightarrow\rightarrow Y$ then $X \rightarrow\rightarrow Z$ where $R = XYZ$
  - $X \rightarrow\rightarrow Y$ is a trivial MVD if

$$Y \subseteq X \quad \text{or} \quad Y \cup X = R$$

  - We can always make *r* satisfy a given MVD by adding more tuples

Name	Child	Phone
A	B	1234
A	C	1235
A	B	1235
A	C	1234

MVD (Cont.)

- Tabular representation of $\alpha \rightarrow\rightarrow \beta$

	α	β	$R - \alpha - \beta$
t_1	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
t_2	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$b_{j+1} \ldots b_n$
t_3	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_4	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$a_{j+1} \ldots a_n$

Example

- Let R be a relation schema with a set of attributes that are partitioned into 3 nonempty subsets.

  $$Y, Z, W$$

- We say that $Y \rightarrow\rightarrow Z$ *(*Y multidetermines Z*)* if and only if for all possible relations *r(R)*

  $$< y_1, z_1, w_1 > \in r \text{ and } < y_2, z_2, w_2 > \in r$$

  then

  $$< y_1, z_1, w_2 > \in r \text{ and } < y_1, z_2, w_1 > \in r$$

- Note that since the behavior of Z and W are identical it follows that

  $Y \rightarrow\rightarrow Z$ if $Y \rightarrow\rightarrow W$

Examples & Exercises

1. List all nontrivial MVDs

A	*B*	*C*
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c3

2 . Add tuples to the following table so that it will satisfy $X \rightarrow\rightarrow Y$

X	*Y*	*Z*	*W*
x1	y1	z1	w1
x1	y1	z1	w2
x1	y2	z2	w1

3. Prove that $\alpha \rightarrow\rightarrow \beta$ implies $\alpha \rightarrow\rightarrow \beta$.

MVDs and Redundancies

- Relations with nontrivial MVDs introduce redundancies
  - no nontrivial FDs => BCNF
  - MVD *C_name* $\rightarrow\!\!\rightarrow$ *C_street*, *C_city* holds
    - street and city info need to be repeated for each occurrence of *c_name*

Loan#	C_name	C_street	C_city
L-23	Smith	North	Rye
L-23	Smith	Main	Manchester
L-93	Curry	Lake	Horseneck

Use of Multivalued Dependencies

- We use multivalued dependencies in two ways:
  1. To test relations to determine whether they are legal under a given set of functional and multivalued dependencies
  2. To specify constraints on the set of legal relations. We shall thus concern ourselves *only* with relations that satisfy a given set of functional and multivalued dependencies.
- If a relation r fails to satisfy a given multivalued dependency, we can construct a relations r' that does satisfy the multivalued dependency by adding tuples to r.

Fourth Normal Form (4NF)

- Relation schema R is in 4NF
  - w.r.t. a set of dependencies D (FDs & MVDs)
  - if for all MVD $\alpha \rightarrow\rightarrow \beta$ in D^+
    - $\alpha \rightarrow\rightarrow \beta$ is a trivial MVD, OR
    - α is a superkey for R
  - *Effect: if there is any nontrivial MVD => it must be FD*

- *Lemma: If R is in 4NF then it is in BCNF*

Decomposition

- No FD
- MVD: $Name \rightarrow\rightarrow Child$

  $Name \rightarrow\rightarrow Phone$

Name	Child	Phone
A	B	1234
A	C	1235
A	B	1235
A	C	1234

- Decompose
  - MVDs become trivial

Name	Child
A	B
A	C

Name	Phone
A	1234
A	1235

Lossless Join Decomposition

- R: relation scheme; D: set of FDs & MVDs
  $\{R_1, R_2\}$ is a lossless decomposition of R iff

  $R_1 \cap R_2 \rightarrow\rightarrow R_1$, or
  $R_1 \cap R_2 \rightarrow\rightarrow R_2$

  - a more general statement for lossless-join
  - not always possible to obtain a dependency-preserving lossless join decomposition into 4NF

Example

- $R = (A, B, C, G, H, I)$

  $F = \{ A \rightarrow\rightarrow B$

  $B \rightarrow\rightarrow HI$

  $CG \rightarrow\rightarrow H \}$

- R is not in 4NF since $A \rightarrow\rightarrow B$ and A is not a superkey for R
- Decomposition

  a) $R_1 = (A, B)$ (R_1 is in 4NF)

  b) $R_2 = (A, C, G, H, I)$ (R_2 is not in 4NF)

  c) $R_3 = (C, G, H)$ (R_3 is in 4NF)

  d) $R_4 = (A, C, G, I)$ (R_4 is not in 4NF)

- Since $A \rightarrow\rightarrow B$ and $B \rightarrow\rightarrow HI$, $A \rightarrow\rightarrow HI$, $A \rightarrow\rightarrow I$

  e) $R_5 = (A, I)$ (R_5 is in 4NF)

  f) $R_6 = (A, C, G)$ (R_6 is in 4NF)

Example

- Example: library information system
- Each book has
  - title,
  - a set of authors,
  - Publisher, and
  - a set of keywords
- Non-1NF relation *books*

MVDs

$title \rightarrow\rightarrow author$

$title \rightarrow\rightarrow keyword$

$title \rightarrow day, month, year$

title	*author-set*	*publisher*	*keyword-set*
		(*name*, *branch*)	
Compilers	{Smith, Jones}	(McGraw-Hill, New York)	{parsing, analysis}
Networks	{Jones, Frick}	(Oxford, London)	{Internet, Web}

1NF Version

- 1NF version of *books*

title	*author*	*pub-name*	*pub-branch*	*keyword*
Compilers	Smith	McGraw-Hill	New York	parsing
Compilers	Jones	McGraw-Hill	New York	parsing
Compilers	Smith	McGraw-Hill	New York	analysis
Compilers	Jones	McGraw-Hill	New York	analysis
Networks	Jones	Oxford	London	Internet
Networks	Frick	Oxford	London	Internet
Networks	Jones	Oxford	London	Web
Networks	Frick	Oxford	London	Web

flat-books

4NF Decomposition

- Remove awkwardness of *flat-books* by assuming that the following multivalued dependencies hold:
  - *title* $\rightarrow\rightarrow$ *author*
  - *title* $\rightarrow\rightarrow$ *keyword*
  - *title* $\rightarrow\rightarrow$ *pub-name, pub-branch*
- Decompose *flat-doc* into 4NF using the schemas:
  - *(title, author)*
  - *(title, keyword)*
  - *(title, pub-name, pub-branch)*

4NF Decomposition

authors

title	*author*
Compilers	Smith
Compilers	Jones
Networks	Jones
Networks	Frick

keywords

title	*keyword*
Compilers	parsing
Compilers	analysis
Networks	Internet
Networks	Web

books4

title	*pub-name*	*pub-branch*
Compilers	McGraw-Hill	New York
Networks	Oxford	London

END OF CHAPTER 7