Introduction to Database Systems

Schema Refinement: Normal Forms

Lee Mong Li

National University of Singapore

Recap

- **Functional dependencies**
  - Constraints on values of attributes
  - If salary of an employee is determined by position, then employees with the same position must have the same salary
- **Minimal cover of a set of FDs**
  - Remove redundant attributes in FDs and redundant FDs

Normalization

- **Process of decomposing relational tables based on functional dependencies to remove anomalies**

company

eNumber	firstName	lastName	address	dept	position	salary
1XU3	Dewi	Srijaya	12a Jln Lempeng	Toys	Clerk	2000
4W3E	Izabel	Leong	10 Outram Park	Sports	Trainee	1200
3XXE	John	Smith	107 Clementi Rd	Toys	Clerk	2000
5SD2	Axel	Bayer	55 Cuscaden Rd	Sports	Trainee	1200
6RG5	Winnie	Lee	10 West Coast Rd	Sports	Manager	2500
755Y	Sylvia	Tok	22 East Coast Ln	Toys	Manager	2600
2SD3	Eric	Wei	100 Jurong drive	Toys	Assistant manager	2200
?	?	?	?	?	Security guard	1500

Redundant storage

Update anomaly

Potential deletion anomaly

Insertion anomaly

FD: position → salary

Schema Decomposition

- **Decomposition of a schema R is a set of schemas $\{R_1, R_2, \ldots, R_n\}$ such that $R_i \subseteq R$ and $R = R_1 \cup R_2 \cup \ldots \cup R_n$**

- **If $\{R_1, R_2, \ldots, R_n\}$ is a decomposition of R, then for any relation *r* of R, we have $r \subseteq \pi_{R1}(r) \otimes \pi_{R2}(r) \otimes \ldots \otimes \pi_{Rn}(r)$**

Decomposition: Example

employee

eNumber	firstName	lastName	address	depart-ment	position
1XU3	Dewi	Srijaya	12a Jln Lempeng	Toys	Clerk
4W3E	Izabel	Leong	10 Outram Park	Sports	Trainee
3XXE	John	Smith	107 Clementi Rd	Toys	Clerk
5SD2	Axel	Bayer	55 Cuscaden Rd	Sports	Trainee
6RG5	Winnie	Lee	10 West Coast Rd	Sports	Manager
755Y	Sylvia	Tok	22 East Coast Ln	Toys	Manager
2SD3	Eric	Wei	100 Jurong drive	Toys	Assistant manager

salary

position	salary
Clerk	2000
Trainee	1200
Manager	2500
Assistant manager	2200
Security guard	1500

Properties of Schema Decomposition

- **Decomposition must preserve information**
  - Data in original relation ≡ Data in decomposed relations
  - Crucial for correctness
- **Decomposition should preserve FDs**
  - FDs in original schema ≡ FDs in decomposed schemas
  - Facilitates checking of FD violations

Lossless-Join Decomposition

- **It is important that a decomposition preserves information; we can reconstruct *r* from joining its projections $\{r_1, r_2, \ldots, r_n\}$**

- **A decomposition of R (with FDs F) into $\{R_1, R_2, \ldots, R_n\}$ is a lossless-join decomposition with respect to F if for every relation *r* of R that satisfies F,**

  $$\pi_{R1}(r) \otimes \pi_{R2}(r) \otimes \ldots \otimes \pi_{Rn}(r) = r$$

- **A decomposition that is not lossless-join is a lossy-join (or lossy) decomposition**

Lossy vs. Lossless-Join Decomposition

Consider R(A, B, C) with FDs F = {A → B}

Example 1: Decomposition of R into { R_1(A, B), R_2(B, C) }

r

A	B	C
a1	b1	c1
a2	b1	c2

→

r1

A	B
a1	b1
a2	b1

r2

B	C
b1	c1
b1	c2

→

r1 ⊗ r2

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

Since $r \subset r1 \otimes r2$, this is not a lossless-join decomposition

Example 2: Decomposition of R into { R_1(A, B), R_3(A, C) }

r

A	B	C
a1	b1	c1
a2	b1	c2

→

r1

A	B
a1	b1
a2	b1

r3

A	C
a1	c1
a2	c2

→

r1 ⊗ r3

A	B	C
a1	b1	c1
a2	b1	c2

Since $r = r1 \otimes r3$, this is a lossless-join decomposition

Lossless-Join Decomposition

- How to determine if $\{R_1, R_2\}$ is a lossless-join decomposition of R?
- Theorem: The decomposition of R (with FDs F) $\{R_1, R_2\}$ is lossless with respect to F if and only if F^+ contains the FD

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

- Attributes common to R_1 and R_2 must contain a key for either R_1 and R_2

Lossless-Join Decomposition

- **How to decompose R into $\{R_1, R_2\}$ such that it is a lossless-join decomposition?**
- **Corollary: If $\alpha \rightarrow \beta$ holds on R and $\alpha \cap \beta = \phi$, then the decomposition of R into $\{ R - \beta , \alpha\beta \}$ is a lossless-join decomposition**

Lossy vs. Lossless-Join Decomposition

Consider R(A, B, C) with FDs F = {A → B}

Example 1: Decomposition of R into { R_1(A, B), R_2(B, C) }

r

A	B	C
a1	b1	c1
a2	b1	c2

→

r1

A	B
a1	b1
a2	b1

r2

B	C
b1	c1
b1	c2

→

r1 ⊗ r2

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

$AB \cap BC = B$ *and neither* $B \rightarrow R_1$ *nor* $B \rightarrow R_2$ *holds on R*

Example 2: Decomposition of R into { R_1(A, B), R_3(A, C) }

r

A	B	C
a1	b1	c1
a2	b1	c2

→

r1

A	B
a1	b1
a2	b1

r3

A	C
a1	c1
a2	c2

→

r1 ⊗ r3

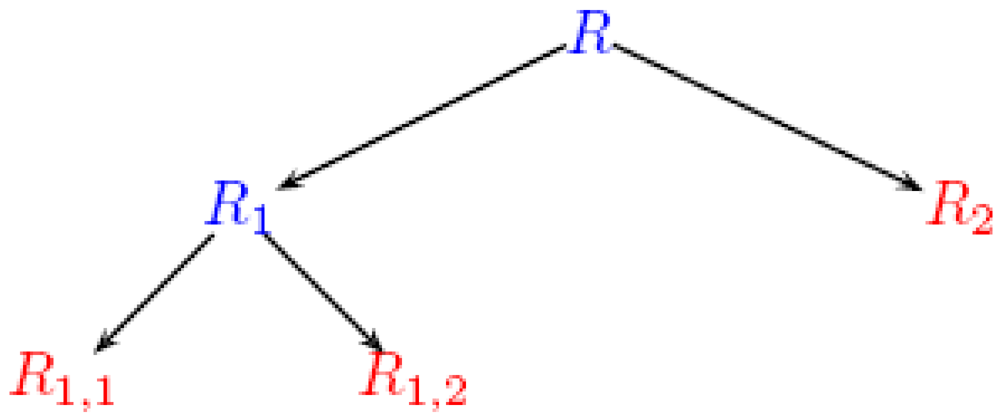
A	B	C
a1	b1	c1
a2	b1	c2

$AB \cap AC = A$ *and* $A \rightarrow R_1$ *holds on R*

Lossless-Join Decomposition

Theorem:

If $\{R_1, R_2\}$ is a lossless join decomposition of R, and $\{R_{11}, R_{12}\}$ is a lossless join decomposition of R_1, then $\{R_{11}, R_{12}, R_2\}$ is a lossless join decomposition of R



Example

Consider MovieList (title, director, address, phone, time) with FDs
F = { {title} → {director}, {address} → {phone}, {address, time} → title} }

MovieList

title → director

Movie (title, director)

CS (address, phone, time, title)

address → phone

Cinema (address, phone)

Screens (address, time, title)

- **{Movie, CS} is a lossless-join decomposition of MovieList**
- **{Cinema, Screens} is a lossless-join decomposition of CS**
- **{Movie, Cinema, Screens} is a lossless-join decomposition of Movie**

Projection of FDs

The projection of F on X (denoted by F_X) is the set of FDs in F+ that involves only attributes in X.

Example:

MovieList (title, director, address, phone, time)

decompose to **Movie (title, director)**

Cinema (address, phone)

Screens (address, time, title)

$F_{Movie} = \{ title \rightarrow director \}$

$F_{cinema} = \{ address \rightarrow phone \}$

$F_{screens} = \{ address, time \rightarrow title \}$

Algorithm: Computing FD Projections

Input: Set of attributes $X \subseteq R$

Set of FDs F on R

Output: F_X

1. Initialize result = ϕ
2. For each $Y \subseteq X$ do
3. $T = Y^+$ (w.r.t. F)
4. result = result $\cup \{Y \rightarrow T \cap X\}$
5. Return result

Example

Consider R(A, B, C) with FDs F = {A $\rightarrow$ B, B $\rightarrow$ C, C $\rightarrow$ A}. Compute F_{AB}

- A^+ (w.r.t F) = ABC, so we have A $\rightarrow$ ABC $\cap$ AB
- B^+ (w.r.t F) = BCA, so we have B $\rightarrow$ BCA $\cap$ AB
- AB^+ (w.r.t F) = ABC, we have AB $\rightarrow$ ABC $\cap$ AB
- F_{AB} = {A $\rightarrow$ AB, B $\rightarrow$ AB, AB $\rightarrow$ AB}

Dependency Preserving Decomposition

- A decomposition $\{R_1, R_2, \ldots, R_n\}$ of R is dependency preserving if

$$F^+ = (F_{R1} \cup F_{R2} \cup \ldots \cup F_{Rn})^+$$

- Dependency preserving decomposition is important because any update to a relation R_i only requires us to enforce F_{Ri} in relation R_i

Example

Consider R(A, B, C) with FDs F = { B → C, AC → B }.
Decomposition {R_1(A, B), R_2(B,C)} is not dependency preserving

- Non-trivial FDs in $F_{R1} = \phi$
- Non-trivial FDs in $F_{R2} = \{B \rightarrow C\}$
- $AC \rightarrow B$ is not preserved because it is not in $(F_{R1} \cup F_{R2})^+$

r

A	B	C
a1	b1	c1
a2	b1	c2

→

r1

A	B
a1	b1
a2	b1

r2

B	C
b1	c1
b2	c2

- Inserting a new tuple (a1, b2, c1) into r will violate $AC \rightarrow B$
- But inserting (a1, b2) into r1 and (b2, c1) into r2 does not violate any FDs in F_{R1} and F_{R2} respectively
- Need to compute $r1 \otimes r2$ to detect violation of $AC \rightarrow B$

Checking for Preservation of Dependencies

- Is $\{R_1, R_2, \ldots, R_n\}$ a dependency-preserving decomposition of R (with FDs F) ?
- If there exists some FD $f \in F$ such that $(F_{R1} \cup F_{R2} \cup \ldots \cup F_{Rn})^+$ does not imply f, then the answer is no, else the answer is yes.

Normal Forms

- A **normal form** restricts the set of data dependencies that are allowed to hold on a schema to avoid certain undesirable redundancy and update problems in the database.
- There are several normal forms, each providing guidance on good schema designs
- We focus on two normal forms that are based on FDs:
  - Boyce-Codd Normal Form (BCNF)
  - Third Normal Form (3NF)
- Definitions of BCNF and 3NF assume that each FD is of the form $X \rightarrow A$ where A is a single attribute.

Boyce-Codd Normal Form (BCNF)

- **Consider a relation schema R with FDs F**
- **R is in Boyce-Codd Normal Form (BCNF) if for every non-trivial FD $X \rightarrow A$ in F, X is a superkey of R.**
- **A non-trivial FD $X \rightarrow A$ that holds on R is said to violate BCNF if X is not a superkey of R**

Example

- **Consider MovieList schema with FDs F = { {title} → {director}, {address} → {phone}, {address, time} → {title} }**
- **Recall that the only key is {address, time}**
- **FDs in F that violate BCNF are**
  - {title} → {director}
  - {address} → {phone}
- **Thus, MovieList is not in BCNF**

Decomposition into BCNF

- **Given schema R with FDs F**
- **Let $X \rightarrow A$ be an FD in F that violate BCNF of R**
- **Decompose R into $R_1 = XA$ and $R_2 = R - A$**
- **If R_1 or R_2 is not in BCNF, then decompose them further as described.**

Decomposition into BCNF

Let $X \rightarrow A$ be an FD in F that causes violation of BCNF
Decompose R into

$$R_1 = XA$$
$$R_2 = R - A$$

If R_1 or R_2 is not in BCNF, then decompose them further as described.

MovieList (title, director, address, phone, time)

title $\rightarrow$ director

Movie (title, director)

CS (title, address, phone, time)

address $\rightarrow$ phone

Cinema (address, phone)

Screens (title, address, time)

Decomposition into BCNF

- **Decomposition $\{R_1, R_2, \ldots, R_n\}$ is in BCNF if each R_i is in BCNF (w.r.t. F_{Ri})**
- **BCNF decompositions are lossless join decomposition**
- **However, not all schema has a dependency-preserving BCNF decomposition**

Example

Consider R (course, prof, time) with FDs

$$F = \{\{course\} \rightarrow \{prof\}, \{prof, time\} \rightarrow \{course\}\}$$

- Keys are {course, time} and {prof, time}
- R is not in BCNF because *course* is not a superkey of R
- The decomposition $\{R_1(course, prof), R_2(course, time)\}$ does not preserve $\{prof, time\} \rightarrow \{course\}$

Third Normal Form (3NF)

- **3NF is a less restrictive normal form that always guarantees a lossless join decomposition that preserves dependencies.**
- **A relation schema R (with FDs F) is in Third Normal Form (3NF) if for every non-trivial FD X $\rightarrow$ A in F, X is a superkey or A is a prime attribute.**
- **A non-trivial FD X $\rightarrow$ A that holds on R is said to violate 3NF if X is not a superkey of R and A is a nonprime attribute**
- **R in BCNF $\Rightarrow$ R in 3NF**

Example

Consider again R (course, prof, time) with FDs {{course} $\rightarrow$ {prof}, {prof, time} $\rightarrow$ {course}}

- **Keys are {course, time} and {prof, time}**
- **R is in 3NF because both prof and course are prime attributes**

Instance of R

prof	time	course
Codd	Tue 3pm	DB101
Codd	Thur 9am	DB101
Gray	Tue 4pm	CS323
Gray	Fri 10am	IT201

Decomposition into 3NF

Input: Schema R with FDs F which is a minimal cover

Output: A dependency preserving, lossless join 3NF decomposition of R

1. **Initialize D = ϕ**
2. **Apply union rule to combine FDs with same LHS into a single FD.**
3. **Let F = $\{f_1, f_2, \ldots, f_n\}$ be the resultant set of FDs**
4. **For each f_i of the form $X_i \rightarrow A_i$ do**

    Create a relation schema R_i (X_i, A_i) for FD f_i

    Insert the schema R_i into D
5. **Choose a key K of R and insert a relation schema R_{n+1}(K) into D**
6. **Remove redundant schema from D**

    Delete R_i from D if $R_i \subseteq R_j$ where $R_j \in D$
7. **Return D**

Synthesis Approach

Example

- **Consider R(A, B, C, D, E) with FDs F = {ABCD $\rightarrow$ E, E $\rightarrow$ D, A $\rightarrow$ B, AC $\rightarrow$ D}**
- **A minimal cover of F is {AC $\rightarrow$ E, E $\rightarrow$ D, A $\rightarrow$ B}**
- **Only key is AC**
- **R is not in 3NF because A $\rightarrow$ B violates 3NF**
- **3NF decomposition of R**
  - Create a schema for each FD:

    R_1 (A, C, E), R_2 (E, D), R_3 (A, B)
  - Create a schema for a key of R: R_4 (A, C)
  - Remove redundant schema R_4 because $R_4 \subseteq R_1$
  - 3NF decomposition is R_1 (A, C, E), R_2 (E, D), R_3 (A, B)

Remarks on 3NF Decomposition

- **A decomposition $\{R_1, R_2, \ldots, R_n\}$ is in 3NF if each R_i is in 3NF (w.r.t. F_{Ri})**
- **The 3NF decomposition produced by synthesis approach may not be unique:**
  - Choice of minimal cover
  - Choice of redundant relation schema being removed

BCNF vs. 3NF

- **BCNF is lossless join (may not be dependency preserving**
- **3NF is lossless join and dependency preserving**
- **Recall R(course, prof, time) with FDs**

  $$\{\{course\} \rightarrow \{prof\}, \{prof, time\} \rightarrow \{course\}\}$$

  - Keys are {course, time} and {prof, time}
  - R is in 3NF but not in BCNF
  - BCNF decomposition { R_1(course, prof), R_2(course, time) } is lossless but not dependency preserving

Another Example

Consider schema R (contractid, supplierid, projectid, deptid, partid, qty, value), CSJDPQV for short

- **Contract C is an agreement that supplier S will supply Q items of part P to project J associated with department D; value of this contract is V**
- **Contract id C is a key: C $\rightarrow$ CSJDPQV**
- **A project purchase a part using a single contract: JP $\rightarrow$ C**
- **A department purchase at most one part from a supplier: SD $\rightarrow$ P**
- **Each project deals with a single supplier: J $\rightarrow$ S**

Example – BCNF Decomposition

- **FDs F = { C → CSJDPQV, JP → C, SD → P, J → S }**
- **From JP → C, C → CSJDPQV and transitivity, we have JP → CSJDPQV**
- **SD → P violates BCNF since SD is not a key**
- **Decompose CSJDPQV into CSJDQV and SDP**
- **From J → S, decompose CSJDQV into JS and CJDQV**
- **Decomposition is lossless**
- **Decomposition does not preserve FD JP → C**
  - Need to join relations to check the FD is not violated.
  - Can add a relation CJP to decomposition if CJP is in BCNF

Example – 3NF Synthesis

- **FDs F = { C $\rightarrow$ CSJDPQV, JP $\rightarrow$ C, SD $\rightarrow$ P, J $\rightarrow$ S }**
- **F is not a minimal cover**
  - Replace C $\rightarrow$ CSJDPQV with

    { C $\rightarrow$ S, C $\rightarrow$ J, C $\rightarrow$ D, C $\rightarrow$ P, C $\rightarrow$ Q, C $\rightarrow$ V }
  - Remove C $\rightarrow$ P from F since it is implied by {C $\rightarrow$ S, C $\rightarrow$ D, SD $\rightarrow$ P}
  - Remove C $\rightarrow$ S from F since it is implied by {C $\rightarrow$ J, J $\rightarrow$ S}
- **Extended minimal cover**

  G = {C$\rightarrow$ JDQV, JP $\rightarrow$ C, SD $\rightarrow$ P, J $\rightarrow$ S }
- **Create relations CJDQV, CJP, SDP, JS**
- **Can combine relations with C as key, e.g., CJDQV and CJP to CJDQVP**

Remarks on Decomposition

- **Too much decomposition can be harmful**
- **Example: R(prof, dept, phone, office) with FD {prof} $\rightarrow$ {dept, phone, office}**
- **Possible to further decompose R into {R1(prof, dept), R2(prof, phone), R3(prof, office)}**
  - Some queries now become expensive to evaluate
  - Example: Find the phone number and office of all professors in CS department
- **Physical Database Design: might consider de-normalization for performance reasons**

Summary

- Normal forms provide a guide to good schema designs
- A schema design is refined by decomposing it into some normal form
- Schema decompositions must be lossless-join and should be dependency preserving
- Both BCNF and 3NF decompositions are lossless-join
- BCNF decomposition are not necessarily dependency preserving