ICS 321 Data Storage & Retrieval Normal Forms (ii)

Asst. Prof. Lipyeow Lim
Information & Computer Science Department
University of Hawaii at Manoa

Redundancies & Decompositions

Hourly_Emps

	<u>SSN</u>	Name	Lot	Rating	Hourly_wages	Hours_worked
۱	123-22-2366	Attishoo	48	8	10	40
	231-31-5368	Smiley	22	8	10	30
	131-24-3650	Smethurst	35	5	7	30
	434-26-3751	Guldu	35	5	7	32
	612-67-4134	Madayan	35	8	10	40

Hourly_Emps

<u>SSN</u>	Name	Lot	Rating	Hours_ worked
123-22-2366	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

RatingWages

Rating	Hourly_ wages
5	7
8	10

Lossless-join Decomposition

 Decomposition of R into X and Y is <u>lossless-join</u> w.r.t. a set of FDs F if, for every instance r that satisfies F:

$$\pi_{X}(r)$$
 join $\pi_{Y}(r) = r$

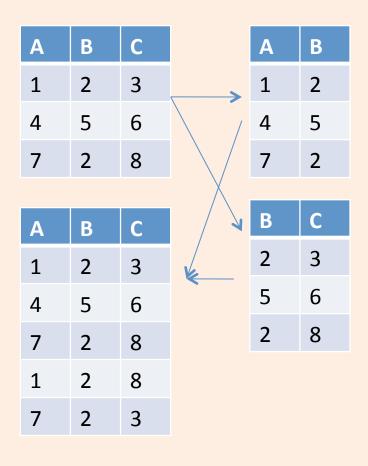
- In general one direction $\pi_{\chi}(r)$ join $\pi_{\gamma}(r) \subseteq r$ is always true, but the other may not hold.
- Definition extended to decomposition into 3 or more relations in a straightforward way.
- It is essential that all decompositions used to deal with redundancy be lossless! (Avoids Problem (2).)

Conditions for Lossless Join

 The decomposition of R into X and Y is losslessjoin wrt F if and only if the closure of F contains:

$$- X \cap Y \rightarrow X, \text{ or}$$
$$- X \cap Y \rightarrow Y$$

 In particular, the decomposition of R into UV and R - V is losslessjoin if U → V holds over R.

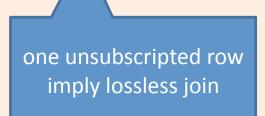


Chase Test for Lossless Join

- R(A,B,C,D) is decomposed into S1={A,D}, S2={A,C}, S3={B,C,D}
- Construct a Tableau
 - One row for each decomposed relation
 - For each row i, subscript an attribute with i if it does not occur in Si.
- FDs: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$
- Rules for "equating two rows" using FDs:
 - If one is unsubscripted, make the other the same
 - If both are subscripted, make the subscripts the same
- Goal: one unsubscripted row

Α	В	С	D	
а	b_1	c ₁	d	S1
a	b ₂	С	d_2	S2
a_3	b	С	d	S3

Α	В	C	D
a	b_1	K C	d
a	X ₂ b ₁	С	d_2
$\frac{3}{3}$ a	b	С	d



Dependency-preserving Decomposition

<u>Student</u>	<u>Course</u>	Instructor		<u>Student</u>	<u>Instructor</u>		Course	<u>Instructor</u>
Smith	OS	Mark	\rightarrow	Smith	Mark	_	OS	Mark
$F = \{ SC \rightarrow I, I \rightarrow C \}$				Checking	$SC \rightarrow Ire$	q۱	uires a jo	oin!

- Dependency preserving decomposition (Intuitive):
 - If R is decomposed into X, Y and Z, and we enforce the FDs that hold on X, on Y and on Z, then all FDs that were given to hold on R must also hold. (Avoids Problem (3).)
- Projection of set of FDs F: If R is decomposed into X, ... projection of F onto X (denoted F_X) is the set of FDs U → V in F⁺ (closure of F) such that U, V are in X.

Dependency-preserving Decomp. (Cont)

Decomposition of R into X and Y is

<u>dependency preserving</u> if $(F_x \text{ union } F_y)^+ = F^+$

Important : F⁺, not F

- If we consider only dependencies in the closure F⁺ that can be checked in X without considering Y, and in Y without considering X, these imply all dependencies in F⁺.
- Example: ABC decomposed into AB and BC.
 - $F={A \rightarrow B, B \rightarrow C, C \rightarrow A}.$
 - Is this dependency preserving?
- Dependency preserving does not imply lossless join:
 - Eg. ABC, A → B, decomposed into AB and BC.
 - And vice-versa! (Example?)

Decomposition into BCNF

- Consider relation R with FDs F. How do we decompose R into a set of small relations that are in BCNF?
- Algorithm:
 - If X → Y violates BCNF,
 decompose R into R-Y and XY
 - Repeat until all relations are in BCNF.
- Example: ABCD, $B \rightarrow C$, $C \rightarrow D$, $C \rightarrow A$.
- Order in which we deal with the violating FD can lead to different relations!

BCNF Decomposition Algorithm (3.20)

- Input: R_0 , set of FDs S_0
- Output: A decomposition of R₀ into a collection of relations, all of which are in BCNF
- Initially $R = R_0$, $S = S_0$
- 1. If R is in BCNF, return {R}
- 2. Let $X \rightarrow Y$ be a violation.
 - a. Compute X+.
 - b. Choose $R_1=X+$
 - c. Let $R_2 = X$ union (R-X+)
- 3. Compute FD projections S₁ and S₂ for R₁ and R₂
- 4. Recursively decompose R₁ and R₂ and return the union of the results

BCNF & Dependency Preservation

- BCNF decomposition using Algo 3.20 is lossless join
- BUT in general there may not be a dependency preserving decomposition into BCNF
 - Example: Bookings(<u>Title</u>, <u>City</u>, <u>Theater</u>), with FD's :
 Th→C, <u>TiC</u>→Th
 - Not in BCNF.
 - Can't decompose while preserving 2nd FD;
- This is the motivation for 3NF.

Decomposition into 3NF

- Obviously, the algorithm for lossless join decomp into BCNF can be used to obtain a lossless join decomp into 3NF (typically, can stop earlier).
- How can we ensure dependency preservation ?
 - If X→Y is not preserved, add relation XY.
 - Problem is that XY may violate 3NF!
 - Example: $JP \rightarrow C$ is not preserved, so add JPC. What if FDs also include $J \rightarrow C$?
- Refinement: Instead of the given set of FDs F, use a minimal cover for F.

Minimum Cover for a Set of FDs

- Minimal cover or basis G for a set of FDs F:
 - Closure of F = closure of G.
 - Right hand side of each FD in G is a single attribute.
 - If we modify G by deleting an FD or by deleting attributes from an FD in G, the closure changes.
- Intuitively, every FD in G is needed, and ``as small
 as possible'' in order to get the same closure as F.
- e.g., A →B, ABCD →E, EF→GH, ACDF →EG has the following minimal cover:
 - A→B, ACD→E, EF→G and EF→H

Computing the Minimal Cover

- Algorithm
 - Put the FDs into standard form X → A. RHS is a single attribute.
 - 2. Minimize the LHS of each FD. For each FD, check if we can delete an attribute from LHS while preserving F⁺.
 - 3. Delete redundant FDs.
- Minimal covers are not unique. Different order of computation can give different covers.
- e.g., A →B, ABCD →E, EF→GH, ACDF →EG has the following minimal cover:
 - $-A \rightarrow B$, ACD $\rightarrow E$, EF $\rightarrow G$ and EF $\rightarrow H$

3NF Decomposition Algorithm (3.26)

- Input: R, set of FDs F
- Output: A decomposition of R into a collection of relations, all of which are in BCNF
- 1. Find a minimal basis/cover for F, say G
- For each FD X → A in G, use XA as one of the decomposed relations.
- 3. If none of the relations from Step 2 is a superkey for R, add another relation whose schema is a key for R.

Summary of Schema Refinement

- If a relation is in BCNF, it is free of redundancies that can be detected using FDs. Thus, trying to ensure that all relations are in BCNF is a good heuristic
- If a relation is not in BCNF, we can try to decompose it into a collection of BCNF relations.
 - Must consider whether all FDs are preserved.
 - If a lossless-join, dependency preserving decomposition into BCNF is not possible (or unsuitable, given typical queries), should consider decomposition into 3NF.
 - Decompositions should be carried out and/or re-examined while keeping performance requirements in mind.