Lossless Join Decomposition (I)

- Each decomposition of a relation schema R into n relation schemas $R_1, ..., R_n$ must fulfil the attribute preservation condition, i.e.,

$$\bigcup_{i=1}^{n} R_i = R$$

- Each attribute in relation schema R should appear in at least one relation schema R_i of the decomposition; no attribute should be lost
- An arbitrary decomposition of a relation schema R into n relation schemas $R_1, ..., R_n$ does not make sense
- Example: Split of the schema *LectureProf*(<u>id</u>, title, pers-id, room) into the schemas R_1(<u>id</u>, title) and R_2(pers-id, room) is incorrect since we cannot reconstruct *LectureProf* from R_1 and R_2
- A decomposition of a relation schema R into the relation schemas $R_1, ..., R_n$ has the lossless (nonadditive) join property, or is lossless, with respect to a set F of FDs on R if, for *every* relation r of R that satisfies F, the following holds: $\pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_n}(r) = r$

Lossless Join Decomposition (II)

- Example
  - This example demonstrates
    - the importance of the lossless join property
    - that normal forms alone are not sufficient to guarantee a good database design
  - We again consider the schema *CarIndex*(<u>model-id</u>, <u>manufacturer</u>, manufacturer-id) with the FDs
    - FD1: {model-id, manufacturer} $\rightarrow$ {manufacturer-id}
    - FD2: {manufacturer-id} $\rightarrow$ {manufacturer}
  - We know that this relation schema is in the 3NF
  - A decomposition of this schema into two BCNF schemas is not straightforward
  - We abstract from and rewrite this example and consider the schema $R(\underline{A}, \underline{B}, C)$ with the FDs $AB \rightarrow C$ (FD1) and $C \rightarrow B$ (FD2)

Lossless Join Decomposition (III)

- Example (*continued*)
  - The three possible decompositions of R into two relation schemas are:
    - $R_1(\underline{A}, \underline{B})$ and $R_2(\underline{A}, \underline{C})$
    - $R_1(B, \underline{C})$ and $R_2(\underline{B}, \underline{A})$
    - $R_1(\underline{C}, \underline{A})$ and $R_2(\underline{C}, B)$
  - FD1 is not preserved by any of these decompositions since it involves three attributes but all relation schemas have two attributes only
  - We translate the three decompositions back into the original context
    - $R_1(\underline{\text{model-id}}, \underline{\text{manufacturer}})$ and $R_2(\underline{\text{model-id}}, \underline{\text{manufacturer-id}})$
    - $R_1(\text{manufacturer}, \underline{\text{manufacturer-id}})$ and $R_2(\underline{\text{manufacturer}}, \underline{\text{model-id}})$
    - $R_1(\underline{\text{manufacturer-id}}, \underline{\text{model-id}})$ and $R_2(\underline{\text{manufacturer-id}}, \text{manufacturer})$
  - All three decompositions lead to two relational schemas R_1 and R_2 that are both in the BCNF
  - Question: Which of the three BCNF decompositions should be selected?

Lossless Join Decomposition (IV)

- Example (*continued*)
  - We check R_1(model-id, manufacturer) and R_2(model-id, manufacturer-id)
    - Can we reconstruct R from R_1 and R_2, i.e., does for *every* relation r of R that satisfies F hold that $\pi_{R_1}(r) \bowtie \pi_{R_2}(r) = r$?
    - The answer is *no* since, e.g., the same value for the common attribute *model-id* could be used in tuples in R_1 and R_2 but the *manufacturer* value in the R_1 tuple does not fit to the *manufacturer-id* value in the R_2 tuple; in other words, two different car brands have different cars with the same *model-id* value ($\rightarrow$ inconsistency)
    - We call this a lossy (join) decomposition
    - The problem is that neither {model-id} $\rightarrow$ {manufacturer} nor {model-id} $\rightarrow$ {manufacturer-id} holds

Lossless Join Decomposition (V)

- Example (*continued*)
  - We check R_1(manufacturer, <u>manufacturer-id</u>) and R_2(<u>manufacturer</u>, <u>model-id</u>)
    - The answer is *no* since, e.g., with respect to the common attribute *manufacturer* the same *manufacturer* value can appear with different *manufacturer-id* values in R_1
    - All these R_1 tuples will be joined to all R_2 tuples with the same *manufacturer* value ($\rightarrow$ spurious tuples, lossy decomposition)
    - The problem here is that {manufacturer} $\rightarrow$ {manufacturer-id} does not hold in R_1
  - We check R_1(<u>manufacturer-id</u>, <u>model-id</u>) and R_2(<u>manufacturer-id</u>, manufacturer)
    - The answer is *yes* since with respect to the common attribute *manufacturer-id* each *manufacturer-id* value of an R_1 tuple *uniquely* finds one R_2 tuple with that *manufacturer-id* value (primary key)
    - The reason is that {manufacturer-id} $\rightarrow$ {manufacturer} holds

Lossless Join Decomposition (VI)

- Generalization of our observation: Nonadditive Join Test for Binary Decompositions (NJB) (only), which is independent of normal forms
- A decomposition of a relation schema R into *two* relation schemas R_1 and R_2 has the lossless (nonadditive) join property (or: is lossless) with respect to a set F of FDs on R if, and only if, either
  - $(R_1 \cap R_2) \to (R_1 - R_2) \in F^+$ [or: $(R_1 \cap R_2) \to R_1 \in F^+$], or
  - $(R_1 \cap R_2) \to (R_2 - R_1) \in F^+$ [or: $(R_1 \cap R_2) \to R_2 \in F^+$]
- This means: If $R_1 \cap R_2$ forms a superkey of either R_1 or R_2, the decomposition of R into R_1 and R_2 is lossless
- Alternative algorithmic formulation
  - Let $R = A \cup B \cup C$, $R_1 = A \cup B$, and $R_2 = A \cup C$ with pairwise disjoint attribute sets A, B, $C \subseteq R$; obviously $R_1 \cap R_2 = A$ holds
  - Then the two conditions above can be checked by
    - $B \subseteq \mathit{CalculateAttributeClosure}(F, A)$, or
    - $C \subseteq \mathit{CalculateAttributeClosure}(F, A)$

Lossless Join Decomposition (VII)

- Example of a lossy join decomposition
  - Let us consider the decomposition of a relation $r(R)$ into the relations $r_1(R_1)$ and $r_2(R_2)$

$r =$

A	B	C
1	2	3
4	2	5

$r_1 = \pi_{A,B}(r) =$

A	B
1	2
4	2

$r_2 = \pi_{B,C}(r) =$

B	C
2	3
2	5

$R = R_1 \cup R_2, R_1 \cap R_2 = \{B\}$

$r_1 \bowtie r_2 =$

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

$\neq r$

The tuples (1, 2, 5) and (4, 2, 3) are spurious. The reason is that according to the NJB test neither $B \rightarrow A$ nor $B \rightarrow C$ holds.

Lossless Join Decomposition (VIII)

- The Chase test provides a general method for testing whether any decomposition of a relation schema R into n relation schemas $R_1, \ldots, R_n$ is lossless with respect to a given set F of FDs on R, i.e., it allows to check whether for *every* relation r of R that satisfies F holds:
  $\pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_n}(r) = r$
- Observations to motivate the idea of the Chase test
  - The result of the natural join is the set of tuples t such that for all $1 \leq i \leq n$ tuple t projected onto the set R_i of attributes is a tuple in $\pi_{R_i}(r)$, i.e.,
    $\forall\ 1 \leq i \leq n : \pi_{R_i}(\{t\}) \subseteq \pi_{R_i}(r)$
  - Any tuple t in r is surely contained in the result of the natural join, i.e., $\forall\ t \in r : t \in \pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_n}(r)$ [or: $r \subseteq \pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_n}(r)$] since the projection of t onto R_i is surely in $\pi_{R_i}(r)$ for each i, and hence, by the previous point, t is contained in the result of the natural join
  - Consequently, the result of the natural join is equal to r if, and only if, every tuple in the natural join is also in r, i.e., $\pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_n}(r) \subseteq r$ with the set F of FDs $\Leftrightarrow \forall\ t \in \pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_n}(r) : t \in r$

Lossless Join Decomposition (IX)

- The last point indicates that the membership test right of the '$\Leftrightarrow$' symbol is all we need to verify that a decomposition has a lossless join
- The Chase test performs this membership test and checks by using the FDs in F whether any tuple in $\pi_{R_1}(r) \bowtie \ldots \bowtie \pi_{R_n}(r)$ can be proved also to be a tuple in r
- If a tuple t is contained in the natural join, then there must be tuples $t_1, \ldots, t_n$ in r such that t is the result of the natural join of the projections of each t_i onto R_i for $1 \leq i \leq n$, i.e., $\pi_{R_1}(\{t_1\}) \bowtie \ldots \bowtie \pi_{R_n}(\{t_n\}) = \{t\}$
- Therefore, we know that each tuple t_i agrees with tuple t on the values of the attributes in R_i but each t_i has unknown values for the attributes not in R_i
- We use this insight and construct a *matrix* or *tableau* according to the following rules:
  - If R has the attributes $A, B, \ldots$, we use $a, b, \ldots$ for the components of t
  - For the t_i we use the same letter as t in the components that are in R_i
  - We subscript the letter with i if the component is not in R_i

Lossless Join Decomposition (X)

- Example
  - Assume a relation schema $R(A, B, C, D)$ is decomposed into the relation schemas $R_1(A, D)$, $R_2(A, C)$, and $R_3(B, C, D)$
  - The matrix for this decomposition is

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

The ith row corresponds to tuple $t_i \in r$. The components for the attributes A and D of R_1 are represented by the unscripted letters a and d in t_1. For the other attributes b and c we add the subscript 1 to show that they are unknown values.

This makes sense since the tuple $t_1 = (a, b_1, c_1, d)$ represents a tuple of r that contributes to tuple $t = (a, b, c, d)$ by being projected onto $\{A, D\}$ and then joined with other tuples. Since the B- and C- components of t_1 are projected out, we know nothing about the values t_1 has for those attributes. The second and third row can be explained similarly. Since row i uses number i as a subscript, the only symbols that can appear more than once are the unsubscripted letters.