Lossless Decomposition

by

Dr. Pratik Roy

Introduction

- Let R be a relation schema and let R_1 and R_2 form a decomposition of R.
- Decomposition is a **lossless decomposition** if there is no loss of information by replacing R with two relation schemas R_1 and R_2.
- Loss of information occurs if it is possible to have an instance of a relation r(R) that includes information that cannot be represented if instead of the instance of r(R) we must use instances of $r_1(R_1)$ and $r_2(R_2)$.

Introduction

- Decomposition is lossless if, for all legal database instances, relation r contains the same set of tuples as the result of the following SQL query:

  select *

  from (select R_1 from r)

  natural join

  (select R_2 from r)

  $$\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r$$

  If we project r onto R_1 and R_2, and compute the natural join of the projection results, we get back exactly r.

- Decomposition is **lossy** if when we compute the natural join of the projection results, we get a proper superset of the original relation.

$$r \subset \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$$

Loss of information via a bad decomposition

ID	name	street	city	salary
⋮				
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

employee

ID	name
⋮	
57766	Kim
98776	Kim
⋮	

name	street	city	salary
⋮			
Kim	Main	Perryridge	75000
Kim	North	Hampton	67000
⋮			

natural join

ID	name	street	city	salary
⋮				
57766	Kim	Main	Perryridge	75000
57766	Kim	North	Hampton	67000
98776	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

- Decomposition is lossy.
- More tuples but less information

Nonadditive (Lossless) Join Property of Decomposition

- No spurious tuples are generated when a NATURAL JOIN operation is applied to the relations resulting from the decomposition.
- Lossless join property is always defined with respect to a specific set F of dependencies.
- **Definition:** A decomposition $D = \{R_1, R_2, ..., R_m\}$ of R has the **lossless (nonadditive) join property** with respect to the set of dependencies F on R if, for every relation state r of R that satisfies F, the following holds, where * is the NATURAL JOIN of all the relations in D:

$$*(\pi_{R_1}(r), ..., \pi_{R_m}(r)) = r.$$

- Loss in lossless refers to loss of information, not to loss of tuples.
- If a decomposition does not have the lossless join property, we may get additional spurious tuples after the PROJECT (π) and NATURAL JOIN (*) operations are applied.
- These additional tuples represent erroneous or invalid information.

Testing for Nonadditive Join Property

Algorithm

- **Input:** A universal relation R, a decomposition D = $\{R_1, R_2, ..., R_m\}$ of R, and a set F of functional dependencies.

1. Create an initial matrix S with one row i for each relation R_i in D, and one column j for each attribute A_j in R.
2. Set S(i, j):= b_{ij} for all matrix entries.
3. For each row i representing relation schema R_i

```
{
    for each column j representing attribute A_j
    {
        if (relation R_i includes attribute A_j) then
            set S(i, j):= a_j
    }
}
```

Testing for Nonadditive Join Property

4. Repeat the following loop until a complete loop execution results in no changes to S:

   for each functional dependency $X \rightarrow Y$ in F

   {

   for all rows in S that have the same symbols in the columns corresponding to attributes in X

   {

   make the symbols in each column that correspond to an attribute in Y be the same in all these rows as follows:

   If any of the rows has an ***a*** symbol for the column, set the other rows to that same ***a*** symbol in the column.

   If no ***a*** symbol exists for the attribute in any of the rows, choose one of the ***b*** symbols that appears in one of the rows for the attribute and set the other rows to that same ***b*** symbol in the column.

   }

   }

Testing for Nonadditive Join Property

5. If a row is made up entirely of ***a*** symbols, then the decomposition has the nonadditive join property; otherwise, it does not.

Nonadditive join test for n-ary decompositions

$R = \{Ssn, Ename, Pnumber, Pname, Plocation, Hours\}$ **EMP_PROJ**

$D = \{R_1, R_2\}$

$R_1 = EMP\_LOCS = \{Ename, Plocation\}$
$R_2 = EMP\_PROJ1 = \{Ssn, Pnumber, Hours, Pname, Plocation\}$

$F = \{Ssn \rightarrow Ename; Pnumber \rightarrow \{Pname, Plocation\}; \{Ssn, Pnumber\} \rightarrow Hours\}$

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

No changes to matrix after applying functional dependencies

Decomposition of EMP_PROJ into EMP_PROJ1 and EMP_LOCS fails test.

Nonadditive join test for n-ary decompositions

EMP

Ssn	Ename

PROJECT

Pnumber	Pname	Plocation

WORKS_ON

Ssn	Pnumber	Hours

$R = \{\text{Ssn, Ename, Pnumber, Pname, Plocation, Hours}\}$
$R_1 = \text{EMP} = \{\text{Ssn, Ename}\}$
$R_2 = \text{PROJ} = \{\text{Pnumber, Pname, Plocation}\}$
$R_3 = \text{WORKS\_ON} = \{\text{Ssn, Pnumber, Hours}\}$

$D = \{R_1, R_2, R_3\}$

$F = \{\text{Ssn} \rightarrow \text{Ename};\ \text{Pnumber} \rightarrow \{\text{Pname, Plocation}\};\ \{\text{Ssn, Pnumber}\} \rightarrow \text{Hours}\}$

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(Original matrix S at start of algorithm)

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	$\cancel{b_{32}}\ a_2$	a_3	$\cancel{b_{34}}\ a_4$	$\cancel{b_{35}}\ a_5$	a_6

(Matrix S after applying the first two functional dependencies; last row is all "a" symbols so we stop)

Decomposition of EMP_PROJ into EMP, PROJECT, and WORKS_ON satisfies test.
Decomposition of EMP_PROJ that has the lossless join property.

Nonadditive Join Test for Binary Decompositions

- There is a special case of a decomposition called a **binary decomposition.**
  - Decomposition of a relation R into two relations.
- A decomposition D = $\{R_1, R_2\}$ of R has the lossless (nonadditive) join property with respect to a set of functional dependencies F on R if at least one of the following functional dependencies is in F^+:
  - $R1 \cap R2 \rightarrow R1$
  - $R1 \cap R2 \rightarrow R2$
- If $R_1 \cap R_2$ forms a superkey for either R_1 or R_2, the decomposition of R is a lossless decomposition.

Nonadditive Join Test for Binary Decompositions

- Consider the schema

  in_dep (ID, name, salary, dept_name, building, budget)

- We decompose into instructor and department schemas:

  instructor (ID, name, dept_name, salary)

  department (dept_name, building, budget)

- Consider the intersection of these two schemas, which is dept_name.
- Dept_name→ dept name, building, budget
- Decomposition is lossless.

Example

- Let R (A, B, C, D) be a relational schema with the following functional dependencies:

$$A \rightarrow B,\ B \rightarrow C,\ C \rightarrow D \text{ and } D \rightarrow B$$

Check whether the decomposition of R into (A, B), (B, C), (B, D) is lossless or lossy.

Lossless Decomposition

- Lossless join decomposition is a decomposition of a relation R into relations R_1, R_2 such that if we perform natural join of relation R_1 and R_2, it will return the original relation R.
- This is effective in removing redundancy from databases while preserving the original data.
- By lossless decomposition it becomes feasible to reconstruct the relation R from decomposed tables R_1 and R_2 by using Joins.

Closure of Attribute Sets

- Let α be a set of attributes. We call the set of all attributes functionally determined by α under a set F of functional dependencies the closure of α under F. We denote it by α^+.
- **Algorithm to compute α^+**

```
result := α;
repeat
    for each functional dependency β → γ in F do
    begin
        if β ⊆ result then result := result ∪ γ;
    end
until (result does not change)
```

Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, M} and the set of functional dependencies

{{E, F} -> {G},

{F} -> {I, J},

{E, H} -> {K, L},

K -> {M},

L -> {N}}

on R.

What is the key for R?

Consider a schema R(A,B,C,D) and functional dependencies A->B and C->D. Decomposition of R into R_1(AB) and R_2(CD) is not lossless.

R = (A, B, C, D, E). We decompose it into R_1 = (A, B, C), R_2 = (A, D, E). The set of functional dependencies is: A → BC, CD → E, B → D, E → A. Show that this decomposition is a lossless-join decomposition.

Same R and F. R_1 = (A, B, C), R_2 = (C, D, E). Show that this decomposition is not a lossless-join decomposition.

- R = (B, O, I, S, Q, D).
- I → B, IS → Q, B → O, S → D

- Find the candidate key for R.
- Give a lossless-join decomposition of R into BCNF.