BCNF

Relational Database Design _Part 4a

Outline

Features of Good Relational Design

Atomic Domains and First Normal Form

Functional Dependencies

Normal Forms

Functional Dependency Theory

Decomposition Using Functional Dependencies

Algorithms for Decomposition using Functional Dependencies

Decomposition Using Multivalued Dependencies

More Normal Form

Lossless-join Decomposition

For the case of $R = (R_1, R_2)$, we require that for all possible relations r on schema R

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

A decomposition of R into R_1 and R_2 is lossless join if at least one of the following dependencies is in F^+:

- $R_1 \cap R_2 \rightarrow R_1$
- $R_1 \cap R_2 \rightarrow R_2$

The above functional dependencies are a sufficient condition for lossless join decomposition; the dependencies are a necessary condition only if all constraints are functional dependencies

Example

$R = (A, B, C)$
$F = \{A \rightarrow B, B \rightarrow C)$

- Can be decomposed in two different ways

$R_1 = (A, B), \quad R_2 = (B, C)$

- Lossless-join decomposition:

$$R_1 \cap R_2 = \{B\} \text{ and } B \rightarrow BC$$

- Dependency preserving

$R_1 = (A, B), \quad R_2 = (A, C)$

- Lossless-join decomposition:

$$R_1 \cap R_2 = \{A\} \text{ and } A \rightarrow AB$$

- Not dependency preserving
  (cannot check $B \rightarrow C$ without computing $R_1 \bowtie R_2$)

Dependency Preservation

Let F_i be the set of dependencies F^+ that include only attributes in R_i.

- A decomposition is **dependency preserving**, if

  $(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$

- If it is not, then checking updates for violation of functional dependencies may require computing joins, which is expensive.

Testing for Dependency Preservation

To check if a dependency $\alpha \rightarrow \beta$ is preserved in a decomposition of R into $R_1, R_2, \ldots, R_n$ we apply the following test (with attribute closure done with respect to F)

- *result* = α
  while (changes to *result*) do
  for each R_i in the decomposition
  $t = (result \cap R_i)^+ \cap R_i$
  $result = result \cup t$
- If *result* contains all attributes in β, then the functional dependency
  $\alpha \rightarrow \beta$ is preserved.

We apply the test on all dependencies in F to check if a decomposition is dependency preserving

This procedure takes polynomial time, instead of the exponential time required to compute F^+ and $(F_1 \cup F_2 \cup \ldots \cup F_n)^+$

Example

$R = (A, B, C)$
$F = \{A \rightarrow B$
$B \rightarrow C\}$
Key = $\{A\}$

R is not in BCNF

Decomposition $R_1 = (A, B)$, $R_2 = (B, C)$

- R_1 and R_2 in BCNF
- Lossless-join decomposition
- Dependency preserving

Testing for BCNF

To check if a non-trivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF

1. compute α^+ (the attribute closure of α), and
2. verify that it includes all attributes of *R*, that is, it is a super key of *R*.

Simplified test: To check if a relation schema *R* is in BCNF, it suffices to check only the dependencies in the given set *F* for violation of BCNF, rather than checking all dependencies in F^+.

- If none of the dependencies in *F* causes a violation of BCNF, then none of the dependencies in F^+ will cause a violation of BCNF either.

However, **simplified test using only *F* is incorrect when testing a relation in a decomposition of R**

- Consider $R = (A, B, C, D, E)$, with $F = \{ A \rightarrow B, BC \rightarrow D\}$
  - Decompose *R* into $R_1 = (A,B)$ and $R_2 = (A,C,D, E)$
  - Neither of the dependencies in *F* contain only attributes from (A,C,D,E) so we might be mislead into thinking R_2 satisfies BCNF.
  - In fact, dependency $AC \rightarrow D$ in F^+ shows R_2 is not in BCNF.

BCNF Decomposition Algorithm

result := $\{R\}$;
done := false;
compute F^+;
while (not *done*) **do**
if (there is a schema R_i in *result* that is not in BCNF)
then begin
let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+,
and $\alpha \cap \beta = \emptyset$;
$result := (result - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$;
end
else *done* := **true;**

Note: each R_i is in BCNF, and decomposition is lossless-join.

Testing Decomposition for BCNF

To check if a relation R_i in a decomposition of R is in BCNF,

- Either test R_i for BCNF with respect to the **restriction** of F to R_i (that is, all FDs in F^+ that contain only attributes from R_i)
- or use the original set of dependencies F that hold on R, but with the following test:
  - for every set of attributes $\alpha \subseteq R_i$, check that α^+ (the attribute closure of α) either includes no attribute of $R_i - \alpha$, or includes all attributes of R_i.
  - If the condition is violated by some $\alpha \rightarrow \beta$ in F, the dependency

    $\alpha \rightarrow (\alpha^+ - \alpha) \cap R_i$

    can be shown to hold on R_i, and R_i violates BCNF.
  - We use above dependency to decompose R_i

Example of BCNF Decomposition

$R = (A, B, C)$
$F = \{A \rightarrow B$
$\quad B \rightarrow C\}$
Key = $\{A\}$

R is not in BCNF ($B \rightarrow C$ but B is not superkey)

Decomposition

- $R_1 = (B, C)$
- $R_2 = (A,B)$

Example of BCNF Decomposition

class (*course_id*, *title*, *dept_name*, *credits*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)

Functional dependencies:

- *course_id*→ *title*, *dept_name*, *credits*
- *building*, *room_number*→*capacity*
- *course_id*, *sec_id*, *semester*, *year*→*building*, *room_number*, *time_slot_id*

A candidate key {*course_id*, *sec_id*, *semester*, *year*}.

BCNF Decomposition:

- *course_id*→ *title*, *dept_name*, *credits* holds
  - but *course_id* is not a superkey.
- We replace *class* by:
  - *course*(*course_id*, *title*, *dept_name*, *credits*)
  - *class-1* (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)

BCNF Decomposition (Cont.)

course is in BCNF

- How do we know this?

building, room_number→*capacity* holds on *class-1*

- but {*building, room_number*} is not a superkey for *class-1*.
- We replace *class-1* by:
  - *classroom* (*building, room_number, capacity*)
  - *section* (*course_id, sec_id, semester, year, building, room_number, time_slot_id*)

classroom and *section* are in BCNF.

BCNF and Dependency Preservation

It is not always possible to get a BCNF decomposition that is dependency preserving

$R = (J, K, L)$
$F = \{JK \rightarrow L$
$\quad L \rightarrow K\}$
Two candidate keys = JK and JL

R is not in BCNF

Any decomposition of R will fail to preserve

$$JK \rightarrow L$$

This implies that testing for $JK \rightarrow L$ requires a join