Introduction to Databases, Fall 2004
IT University of Copenhagen

Lecture 5: Normalization II; Database design case studies

October 1, 2004

Lecturer: Rasmus Pagh

Today's lecture

- What you should remember from previously.

Normalization II:

- Multivalued dependencies.
- 4th normal form.
- Some observations on normalization.

Case studies in database design:

- Internet bookstore.
- TV series database.

— What you should remember from previously —

In this lecture I will assume that you remember:

- Basic concepts of normalization:
  - Decomposition
  - Functional dependency
  - Boyce-Codd normal form and 3rd normal form

Next: Multivalued dependencies.

Redundancy in BCNF relations

Boyce-Codd normal form eliminates redundancy in each tuple, but may leave redundancy among tuples in a relation.

This happens, for example, if two many-many relationships are represented in a relation.

`[Figure 3.29 shown on slide]`

Example: In the relation `StarsIn(name, street, city, title, year)` we could represent two many-many relationships: between actors and addresses, and between actors and movies.

Curing it with NULL values?

Then what about something like one of these:

name	*street*	*city*	*title*	*year*
C. Fisher	123 Maple St.	Hollywood	NULL	NULL
C. Fisher	5 Locust Ln.	Malibu	NULL	NULL
C. Fisher	NULL	NULL	Star Wars	1977
C. Fisher	NULL	NULL	Empire Strikes Back	1980
C. Fisher	NULL	NULL	Return of the Jedi	1983

name	*street*	*city*	*title*	*year*
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980
C. Fisher	NULL	NULL	Return of the Jedi	1983

Decomposition

A better idea is to eliminate redundancy by decomposing `StarsIn` as follows:

name	*street*	*city*
C. Fisher	123 Maple St.	Hollywood
C. Fisher	5 Locust Ln.	Malibu

name	*title*	*year*
C. Fisher	Star Wars	1977
C. Fisher	Empire Strikes Back	1980
C. Fisher	Return of the Jedi	1983

When can we decompose?

When can we decompose a relation `R`? Suppose we decompose into two relations (for simplicity we assume that there is just one common attribute):

```
R1(A, B1, B2,..., Bm)
R2(A, C1, C2,..., Ck)
```

Now consider a specific value a for attribute A, occurring in the set of tuples T_1 from `R1` and in the set of tuples T_2 from `R2`.

When we join `R1` and `R2`, *every pair of tuples* from T_1 and T_2 are combined.

When can we decompose (2)?

Example:

a	b
1	N
1	S
2	U
2	D

a	c
1	E
1	W
1	NE
1	NW
1	SE
1	SW
2	45
2	90

Multivalued dependencies

When we can decompose R into relations

`R1(A1, A2,...An, B1, B2,..., Bm)`

`R2(A1, A2,...An, C1, C2,..., Ck)`

(with no Bs among the Cs) then we say that there is a **multivalued dependency** (MVD) from the As to the Bs, written

$$\texttt{A1 A2...An} \twoheadrightarrow \texttt{B1 B2...Bm}$$

Example: Since `StarsIn` can be decomposed into

`StarsIn1(name, street, city)` and `StarsIn2(name, title, year)`

it has the MVD `name` $\twoheadrightarrow$ `street city`.

Multi-valued dependencies, book's definition

`A1 A2...An` $\twoheadrightarrow$ `B1 B2...Bm`

holds exactly if:

> For every pair of tuples t and u from R that agree on all As, we can find some tuple v in R that agrees:
>
> - With both t and u on the As
> - With t on the Bs
> - With u on the Cs

`[Figure 3.30 shown on slide]`

Problem session (5 minutes):
Try to convince yourselves that this definition, used in the coursebook, is equivalent to the one given previously in this lecture.

Unavoidable and trivial MVDs

If $\{$A1,A2,...,An$\}$ form a superkey, then for any B1, B2,..., Bm we unavoidably have:

A1 A2...An $\twoheadrightarrow$ B1 B2...Bm

An MVD is said to be **trivial** if either

- One of the Bs is among the As, or
- All the attributes of R are among the As and Bs.

Next: 4th normal form.

4th normal form

Roughly speaking, a relation is in 4th normal form if it cannot be meaningfully decomposed into two relations. More precisely:

> A relation is in **fourth normal form** (4NF) if any multivalued dependency among its attributes is either unavoidable or trivial.

Example: `StarsIn` has the MVD `name` $\twoheadrightarrow$ `street city` which is nontrivial. Since `name` is not a superkey the relation is not in 4NF.

Decomposing a relation into 4NF

Suppose we have a relation R which is not in 4NF. Then there is a nontrivial MVD

$$A_1A_2\ldots A_n \twoheadrightarrow B_1B_2\ldots B_m$$

which is not unavoidable.

To eliminate the MVD we split R into two relations:

- One with all attributes of R except $B_1, B_2, \ldots, B_m$.
- One with attributes $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$.

If any of the resulting relations is not in 4NF, the process is repeated.

4NF decomposition example

Recall the relation `StarsIn` with schema

`StarsIn(name, street, city, title, year)`

It has the following nontrivial MVD, which is not unavoidable:

$$\texttt{name} \twoheadrightarrow \texttt{street city}$$

Thus the decomposition yields the following relations (both in 4NF):

`StarsIn1(name, street, city)`

`StarsIn2(name, title, year)`

Problem session (5 minutes)

What would happen if we tried to do the decomposition:

- According to an unavoidable MVD?
- According to an MVD including all attributes of R?
- According to an MVD with a common attribute on the left and right hand side?

Next: Some observations on normalization

Relationship among normal forms

Inclusion among normal forms:

Any relation in 4NF is also in BCNF.

Any relation in BCNF is also in 3NF.

`[Figure 3.31 shown on slide]`

Properties of normal forms:

A "higher" normal form has less redundancy, but may not preserve functional and multivalued dependencies.

`[Figure 3.32 shown on slide]`

How should normal forms be used?

The various normal forms may be seen as *guidelines* for designing a good relation schema. Some complexities that arise are:

- Should we split keys, introducing dependencies between relations (in 3NF we do not)?
- What is the effect of decomposition on performance?
- How does decomposition affect query programming?

Most important points in this lecture

After this week you should:

- Be able to determine whether a relation is in 4th normal form.
- Be able to split a relation in several relations to achieve 4th normal form.