EDAF75
Database Technology

Lecture 8 – more normalization

Christian.Soderberg@cs.lth.se

Feb 13, 2025

From the previous lecture

- If a set of attributes, $\{A_1, \ldots, A_n\}$, uniquely determines another set of attributes, $\{B_1, \ldots, B_m\}$, we say that $\{A_1, \ldots, A_n\} \rightarrow \{B_1, \ldots, B_m\}$ is a *functional dependency* (FD)
- The left hand side of a FD is sometimes called its *determinant set*, and the right hand side its *dependent set*
- Given a set of FD's, we can expand a set of attributes into its *closure*, by applying the FD's until we can include no more attributes
- A *minimal* set of attributes $\{A_1, \ldots, A_n\}$ whose closure is all the attributes of a relation, is a *candidate key* for the relation – *by minimal we mean a set which contains no superfluous attributes*

Boyce Codd Normal Form (BCNF)

- A relation is in BCNF iff (*if-and-only-if*) all its functional dependencies have left hand sides which are superkeys
- We have a recipe to split a relation into several smaller relation, so that each relation will be in BCNF:
  1. Put all the attributes of a BCNF-breaking FD into a new relation (this will become a sort of 'look-up' table, with the left hand side of the FD as a key)
  2. Remove all the attributes on the right hand side of the BCNF-breaking FD from our original relation (we can always find the removed attributes by joining with our 'look-up'-table, using the left hand side of the FD as a join predicate)
  3. If any of our FDs still break BNCF for any of our relations, go to 1.

Food for thought

- How many rows in a table can we have with a given set of values for a superkey?
- How many rows in a table can we have with a given set of values for some attributes $\{A_1, \ldots, A_n\}$, if $\{A_1, \ldots, A_n\}$ is not a superkey?
- If we have a functional dependency $\{A_1, \ldots, A_n\} \rightarrow \{B_1, \ldots, B_m\}$, where $\{A_1, \ldots, A_n\}$ is not a superkey, in how many rows can we have the same values of $\{A_1, \ldots, A_n\}$, and hence $\{A_1, \ldots, A_n\}$ *and* $\{B_1, \ldots, B_m\}$?

Why we want to be in BCNF

- A relation is in BCNF when we have removed all redundancy based on functional dependencies
- Assume we have

  `movies(title, year, length, category, star)`

  and the FD's:

  FD_1: `title year star` → `length category`
  FD_2: `title year` → `length category`
- The attribute `star` must be included in the key, since we can't derive a star from just {`title`, `year`} (there can be more than one star in a movie) – so the key is {`title`, `year`, `star`}

Why we want to be in BCNF

`movies(title, year, length, category, star)`

FD_1: `title year star` → `length category`
FD_2: `title year` → `length category`
key: {`title`, `year`, `star`}

title	year	length	category	star
The Post	2017	116	drama	Meryl Streep
The Post	2017	116	drama	Tom Hanks
The Post	2017	116	drama	Sarah Paulson
The Post	2017	116	drama	Bob Odenkirk

- The left hand side of FD_2 is not a superkey, so it can be repeated many times in the table, *and each time, we'll get the same values of the attributes in it's right hand side* – that's why we get redundance
- By moving `length` and `category` into a separate table, with {`title`, `year`} as it's key, we don't have to keep them in this table, we can instead join them in using {`title`, `year`}
- In the other table, we'll only have one row for each combination of {`title`, `year`} (it's a key)

Getting into BCNF

`movies(title, year, length, category, star)`
FD_1: `title year star` → `length category`
FD_2: `title year` → `length category`
key: {`title`, `year`, `star`}

- Since FD_2 breaks BCNF, we break up the original relation `movies` into two relations, the first with all attributes of FD_2 (`movie_info`), the other is our original relation with all attributes on the right hand side of FD_2 removed (`movie_stars`):
  - `movie_info(title, year, length, category)`
  - `movie_stars(title, year, star)`

Getting into BCNF

Now we get:

`movie_info`:

title	year	length	category
The Post	2017	116	drama
Three Billboards …	2017	115	drama
Lady Bird	2017	94	comedy

`movie_stars`:

title	year	star
The Post	2017	Meryl Streep
The Post	2017	Tom Hanks
The Post	2017	Sarah Paulson
The Post	2017	Bob Odenkirk
…	…	…

Exercise

Exercise

Prove that all two-attribute relations, $R(A, B)$, are in BCNF.

Reconstruction

- We want to be able to reconstruct all information after we've deconstructed a table – such a deconstruction is called *lossless*
- To reconstruct relations deconstructed using the BCNF-deconstruction algorithm above, we only need to join using the attributes in the left hand side of the FD's we've decomposed on
- Even if all data can be reconstructed, some FD's may get lost when we decompose (more on than below)

Exercise

Exercise

We have a relation $R(A, B, C)$ with only one FD:

FD_1: $A \rightarrow B$

(a) Is R in BCNF?

(b) Decompose into BCNF

(c) What about the decomposition of R into $R_1(A, B)$ and $R_2(B, C)$? R_1 and R_2 are both in BCNF (all relations with two attributes are), but what do we get if we project and reconstruct

a	b	c
1	2	3
2	2	4

(d) Do the same projection and reconstruction using your own BCNF-decomposition

Third Normal Form

- Sometimes we lose important FD's when we decompose into BCNF
- Using Third Normal Form (3NF) instead might be an option:

Definition: Third Normal Form

A relation R is in *Third Normal Form* iff the left hand side of every non-trivial FD is a superkey for R, *or the right hand side is a member of some key*.

- 3NF is slightly weaker than BCNF (observe that we add the last phrase to an otherwise identical definition)

Exercise

We have:

`bookings(title, theater, city)`

FD_1: `theater` → `city`

FD_2: `title city` → `theater`

FD_1 means that the theaters will have unique names, and FD_2 means that two theaters in the same city never show the same movie (obviously, both FD's are unrealistic). The candidate keys are {`title`, `city`} and {`theater`, `title`}.

- Is `bookings` in BCNF?
- Decompose into BCNF, and enter the following data:

theater	city	title
Guild	Menlo Park	Phantom Thread
Park	Menlo Park	The Shape of Water

- Add a performance of "Phantom Thread" at "Park" (it breaks no constraints for any table), and then reconstruct.
- Is `bookings` in 3NF?

To Decompose, or Not To Decompose?

- Having our relations in BCNF makes it *much* easier to maintain our database
- But it comes with a cost: we often have to join tables to find data which could have been in one un-normalized table, and some functional dependencies may get lost
- So, sometimes it's perfectly reasonable to use tables which aren't normalized up to BCNF

Example

- Earlier we had the relation:

  `friends(name, phone, context)`

  and saw that it led to a lot of repetition:

name	phone	context
Liv	0708-111222	school
Liv	0708-111222	basketball
Liv	0708-111222	judo
Liv	0708-823123	school
Liv	0708-823123	basketball
Liv	0708-823123	judo
Adam	0708-222111	school
Adam	0708-222111	basketball

- Is `friends` in BCNF?

Multivalued Dependencies

- So, the relation:

  `friends(name, phone, context)`

  has the key (`name`, `phone`, `context`), and it is in BCNF, since there are no FD:s
- But there is another kind of dependency, for a given `name`, we have:
  - a given set of `phone` values (`name` ↠ `phone`), and
  - a given set of `context` values (`name` ↠ `context`)
- Technically, we say that `name` *multidetermines* `phone` (and `context`), but it's not a functional dependency, since a FD requires its right hand side to be a unique value
- Those two 'multideterminates' have the same left hand side, and since they are independent of each other, they generate lots of repetition when we combine them in one table
- Having two such independent 'multideterminates' with the same left hand side in a table is called having a *Multivalued Dependency* (MVD)

4th normal form

- Having only `name` ↠ `phone` (without `context`) would be fine, then we'd just be listing all different phone numbers of our friends – it's combining `name` ↠ `phone` and `name` ↠ `context` in one table which is problematic
- If we use the same machinery as in BCNF normalization, but replace FD:s with MVD:s, we could break out `name` ↠ `phone` into its own table, and then keep everything except the right hand side (`phone`) in the original table:

```
phones(name, phone)
friends(name, context)
```

- These tables are in *4th normal form*, and we can join together the initial table with:

```
SELECT   *
FROM     friends
         JOIN phones USING (name)
```

Normal Forms

- There are many normal forms (1NF, 2NF, 3NF, BCNF, 4NF, 5NF, 6NF)
- In the course, we focus on BCNF and 3NF
- The forms are progressively stricter:

$$4\text{NF} \subset \text{BCNF} \subset 3\text{NF} \subset \ldots$$

Property	3NF	BCNF	4NF
Eliminates redundancy due to FD's	most	yes	yes
Eliminates redundancy due to MVD's	no	no	yes
Preserves FD's	yes	some	some
Preserves MVD's	some	some	some

- We normally aim for BCNF for all relations, but sometimes accept 3NF if there are good reasons

Exercise

Exercise

Consider the student application example we've seen over and over again:

- Which are the functional dependencies
- Would one big table with all data be in BCNF?
- Normalize so we get into BCNF
- Compare with the tables we would get from an ER-model of the problem

Exercise

Exercise

The relation $R(A, B, C, D)$ has the following functional dependencies:

FD_1: $A\ B \to D$

FD_2: $B\ C \to A\ D$

FD_3: $D \to B$

(a) Which are the keys of the relation?

(b) Show that the relation isn't in BCNF. Is it in 3NF?

(c) Decompose the relation into relations which are in BCNF