Ian Kenny

November 18, 2016

Databases

Lecture 8

- Test 3 date and time.
- Test 1 feedback.

In this lecture

- Database normalisation.

Database normalisation

Database normalisation is a process for removing unwanted dependencies from database tables to prevent the anomalies discussed in an earlier session, i.e. update, insertion and deletion anomalies.

Generally, normalisation involves taking larger tables that cause such anomalies and breaking them down into smaller tables that do not cause the anomalies. The process can be automated.

Normal forms

Relations can be described as being in a particular *normal form*. There are a series of levels of normal forms and each level subsumes the previous one. Thus if a relation is in a particular normal form then it is automatically in all lower forms.

We will look at the normal forms up to *Boyce-Codd Normal Form* (BCNF). However, we won't consider the lower forms in detail since they are largely obsolete.

Normalisation usually refers to the process of converting relations to what is called *Third Normal Form*, not lower forms.

First Normal Form (1NF)

A relation in 1NF has *only atomic values*. This seems a very basic requirement for a database table. The relation below is not in 1NF

empid	name	phoneNumber
1	Jones	0121 333 4444, 07876453432
2	Smith	0141 222 7777, 07874545454
3	Adams	0161 111 3333, 07754545445

Converting to 1NF

empid	name	homeNumber	mobileNumber
1	Jones	0121 333 4444	07876453432
2	Smith	0141 222 7777	07874545454
3	Adams	0161 111 3333	07754545445

Second Normal Form (2NF)

2NF depends on the notion of keys. A *key attribute* is an attribute that is part of a candidate key. A *non-key attribute*, therefore, is an attribute that is not part of a candidate key.

A relation in 2NF if it is in 1NF **and** *no non-key attribute is partially dependent on a candidate key*.

In other words, every non-key attribute is dependent on the *whole* of every candidate key. This means that a relation is automatically in 2NF if every candidate key has a single attribute (is not a composite key).

Recall that a *partial dependency* is a functional dependency in which an attribute can be removed from the determinant but the dependency still exists.

Second Normal Form (2NF)

The Employee table is not currently in 2NF. Consider the candidate key

$\{empid, job, project\}$

For example, *manager* is dependent on *project*. Thus a non-key attribute depends on only part of the key. This violates 2NF.

$project \rightarrow manager$

empid	name	job	salary	project	manager	prlength	prcost
1	Jones	engineer	35000	Gherkin	Davis	5	500m
2	Wilson	sales	28000	Tunnel	Fulton	6	250m
1	Jones	engineer	35000	Tunnel	Fulton	6	250m
3	Peters	tech	24000	Gherkin	Davis	5	500m
3	Peters	tech	24000	Dome	Mandelson	2	2000m
4	Price	runner	17500	Dome	Mandelson	2	2000m
5	Dollis	designer	45000	Airport	Craig	5	500m

Employee

Converting to 2NF

We will not explicitly consider conversion to 2NF.

Third Normal Form (3NF)

A relation is in 3NF if it is in 2NF and *no non-key attribute is* **transitively dependent** *on a candidate key*.

Third Normal Form (3NF)

Consider a Project table, i.e., the project data taken from the Employee table and placed in a separate table.

If the key of this table is $\{project\}$ then the table is automatically in 2NF since the key is not a composite key. It is not in 3NF, however, since a transitive dependency exists

$project \rightarrow prlength$
$prlength \rightarrow prcost$, therefore
$project \rightarrow prcost$

project	manager	prlength	prcost
Gherkin	Davis	5	500m
Tunnel	Fulton	6	250m
Dome	Mandelson	2	2000m
Airport	Craig	5	1000m

Project

Converting to 3NF

We will consider conversion to 3NF incidentally in the conversion to a higher form.

Boyce-Codd Normal Form (BCNF)

BCNF is similar to 3NF.

A relation is in BCNF if it is 3NF and *the determinant of every FD is a key*.

Hence, we do not allow (non-trivial) dependencies that don't depend on a key.

3NF permits functional dependencies that do not have a key as the determinant. BCNF does not permit such functional dependencies.

Conversion to BCNF

We will now convert the Employee table to BCNF.

empid	name	job	salary	project	manager	prlength	prcost
1	Jones	engineer	35000	Gherkin	Davis	5	500m
2	Wilson	sales	28000	Tunnel	Fulton	6	250m
1	Jones	engineer	35000	Tunnel	Fulton	6	250m
3	Peters	tech	24000	Gherkin	Davis	5	500m
3	Peters	tech	24000	Dome	Mandelson	2	2000m
4	Price	runner	17500	Dome	Mandelson	2	2000m
5	Dollis	designer	45000	Airport	Craig	5	500m

Employee

Conversion to BCNF

For this process we need the schema of the relation, the candidate keys and the functional dependencies.

The schema is the attributes of the relation

$Employee(empid, name, job, salary, project, manager, prlength, prcost)$.

The candidate key is $\{empid, job, project\}$.

We will select the following FDs

$project \rightarrow manager$
$job \rightarrow salary$
$empid \rightarrow name$
$project \rightarrow prlength, prcost$

Conversion to BCNF

Firstly, we note that the relation is not already in BCNF (it is not even in 2NF). None of the determinants of the FDs is a candidate key.

$project \rightarrow manager$
$job \rightarrow salary$
$empid \rightarrow name, job$
$project \rightarrow prlength, prcost$

Conversion to BCNF

We select an FD to start the process. Let's select

$job \rightarrow salary$

We firstly place those attributes in a new relation. Let's call it E_1.

$E_1(job, salary)$

We select a key for this new relation: $\{job\}$.

We then create a new relation which contains all of the attributes of *Employee* but with the attributes of E_1 removed **except** for the key of E_1. Call it E_2

$E_2(empid, name, job, project, manager, prlength, prcost)$.

Decomposition

This has resulted in a decomposition of *Employee* into two relations E_1 and E_2. To check that this decomposition has not lost any information we check:

If the union of the attributes of E_1 and E_2 is the set of attributes in *Employee*.

If the natural join of E_1 and E_2 results in *Employee*.

We will consider this topic shortly.

Decomposition

We consider the relations E_1 and E_2. Are they in BCNF? It is fairly clear that E_1 must be. There is only one non-prime attribute and it depends on the key.

It is also clear that E_2 cannot be since that is still not even in 2NF.

Conversion to BCNF

We have dealt with the dependency

$job \rightarrow salary$

So now we need to choose another one that exists in table E_2. Technically, we should recompute all of the FDs for the new tables but we will proceed for now with the original set since they still apply.

We could choose

$empid \rightarrow name, job$

Conversion to BCNF

$empid \rightarrow name, job$

We remove those attributes from E_2 except for the key which is $\{empid\}$ to create new tables E_3 and E_4.

E_3 contains just the attributes $\{empid, name, job\}$, whilst E_4 contains all of the attributes of E_2 with $\{name, job\}$ removed.

$E_3(empid, name, job)$
$E_4(empid, project, manager, prlength, prcost)$

Conversion to BCNF

$E_3(empid, name, job)$
$E_4(empid, project, manager, prlength, prcost)$

E_3 and E_4 are a decomposition of E_2. Thus we require that the union of their attributes be equal to the attributes in E_2 and that the natural join of E_3 and $E_4 = E_2$. Again, we will return to this point.

Conversion to BCNF

Which relations do we have at this point?

We have E_1, E_3 and E_4. *Employee* was decomposed into E_1 and E_2 and then E_2 was decomposed into E_3 and E_4. Thus *Employee* and E_2 have gone.

We have also now dealt with the FD $empid \rightarrow name, job$.

Conversion to BCNF

Thus we have relations

$E_1(job, salary)$
$E_3(empid, name, job)$
$E_4(empid, project, manager, prlength, prcost)$

We also have the FDs

$project \rightarrow manager$
$project \rightarrow prlength, prcost$

We can see that the combining rule allows us to state the FD

$project \rightarrow manager, prlength, prcost$

Conversion to BCNF

E_4 is still not even in 2NF.

We need to decompose E_4 based on the remaining FD.

$project \rightarrow manager, prlength, prcost$

Thus, we need to decompose E_4 into two further relations. We place all of the attributes of the FD into E_5. In E_6 we remove all of the attributes of E_5 except the key, which is $\{project\}$.

$E_5(project, manager, prlength, prcost)$
$E_6(empid, project)$

Thus, E_4 is now deleted.

Conversion to BCNF

Unfortunately, whilst E_5 is in 2NF it is not yet in 3NF (hence not in BCNF). Here is E_5

$E_5(project, manager, prlength, prcost)$

Although we combined the FDs

$project \rightarrow manager$
$project \rightarrow prlength, prcost$

To create

$project \rightarrow manager, prlength, prcost$

This did not remove the transitive dependencies.

Conversion to BCNF

We still have the dependency

$prlength \rightarrow prcost$

We need to carry on with the decomposition.

Conversion to BCNF

We have

$E_5(project, manager, prlength, prcost)$

$prlength \rightarrow prcost$

We follow the same process as previously to obtain

$E_7(prlength, prcost)$
$E_8(project, manager, prlength)$

Conversion to BCNF

We are left with

$E_1(job, salary)$
$E_6(empid, project)$
$E_3(empid, name, job)$
$E_7(prlength, prcost)$
$E_8(project, manager, prlength)$

Conversion to BCNF

Is it the case that the union of the attributes of these relations is equal to the set of attributes in *Employee*?

We have

$$\{project, manager, prlength\} \cup \{prlength, prcost\} \cup \{empid, name, job\} \cup \{job, salary\} \cup \{empid, project\}$$

$$=$$

$$\{empid, name, job, salary, project, manager, prlength, prcost\}$$

So the answer is yes, we have the original attributes.

Conversion to BCNF

Finally, we need to know whether we can recover all of the original data using natural joins.

That is, does

$$Employee = E_1 \bowtie E_6 \bowtie E_3 \bowtie E_8 \bowtie E_7$$

Conversion to BCNF

Let's rename the relations

$E_1 = job$
$E_6 = worksOn$
$E_3 = employeeDetails$
$E_8 = projectDetails$
$E_7 = project$

So, does

$Employee = job \bowtie worksOn \bowtie employeeDetails \bowtie projectDetails \bowtie project$

Conversion to BCNF

Here's the Employee table reconstructed from the new tables. It contains the same data.

prlength	project	job	empid	salary	name	manager	prcost
5	gherkin	engineer	1	35000	Jones	Davis	500
6	tunnel	engineer	1	35000	Jones	Fulton	250
6	tunnel	sales	2	28000	Wilson	Fulton	250
5	gherkin	tech	3	24000	Peters	Davis	500
2	dome	tech	3	24000	Peters	Mandelson	2000
2	dome	runner	4	17500	Price	Mandelson	2000
5	airport	designer	5	45000	Dollis	Craig	500

(7 rows)

Lossless decomposition

In the next session we will consider these properties of decompositions and the desirability of a *lossless* decomposition. There are other desirable 'side effects' of decomposition too, which we will also consider.