Ian Kenny

November 27, 2016

Databases

Lecture 9

In this lecture

- More on normalisation.
- Properties of decompositions.

Properties of decomposition

Normalisation usually involves the decomposition of relations, i.e. breaking them into smaller relations. This could result in a loss of information.

Generally we don't want to lose information in the process of decomposition.

Information about tables includes the attributes, the tuples and the functional dependencies.

We need to check that our decompositions don't result in a loss of information.

Properties of decomposition

We will now consider

- Preservation of attributes.
- Lossless/lossy joins.
- Preservation of dependencies.
- Determining if a decomposition is in BCNF.

Note that we will consider decompositions that are independent of each other in terms of the above properties. For example, when we consider the question of whether or not a decomposition has the lossless join property, that decomposition may or may not be in BCNF and may or may not be dependency preserving.

Preservation of attributes

We require that a decomposition does not result in the loss of attributes.

If we have a relation

$R(A_1, A_2, ..., A_n)$

And we decompose it into two relations

$R_1(B_1, B_2, ..., B_m)$
$R_2(C_1, C_2, ..., C_k)$

Then we require that $\bar{A} = \bar{B} \cup \bar{C}$

Where

$\bar{A}$ is the set of attributes $\{A_1, A_2, ..., A_n\}$
$\bar{B}$ is the set of attributes $\{B_1, B_2, ..., B_m\}$
$\bar{C}$ is the set of attributes $\{C_1, C_2, ..., C_k\}$

Preservation of attributes

Consider relation R

$R(A, B, C, D)$

Which has been decomposed into

$R_1(A, B)$
$R_2(A, C, D)$

We require that $\{A, B\} \cup \{A, C, D\} = \{A, B, C, D\}$

Which it does, in this case.

Lossless join

We also require that the natural join of two relations (and by extension, more than two) that have resulted from a decomposition is equal to the original non-decomposed relation.

For example, if relation R has been decomposed into R_1 and R_2 then

$R_1 \bowtie R_2 = R$

This is called a *lossless join*.

If we don't maintain this property when decomposing relations, then we are losing information about the original relation.

Lossless join

When decomposing a relation into further relations, the decomposition is effectively distributing the relation's attributes across the new relations through the *projection* operator.

If relation R with schema $R(A, B, C, D)$ has been decomposed into R_1 with schema $R_1(A, B, C)$ and R_2 with schema $R_2(A, D)$ then

$R_1 = \Pi_{A,B,C}(R)$
$R_2 = \Pi_{A,D}(R)$

Lossless join

If relation R with schema $R(A, B, C, D)$ has been decomposed into R_1 with schema $R_1(A, B, C)$ and R_2 with schema $R_2(A, D)$ then the lossless join of R_1 and R_2 is defined as

$\Pi_{A,B,C}(R) \bowtie \Pi_{A,D}(R) = R$

Lossless join

Consider a relation R with schema (A,B,C,D)

A	B	C	D
1	a	x	10
2	b	x	15
1	c	y	15
2	a	x	10

R

And a decomposition of R into R_1 and R_2

$R_1(A, B)$
$R_2(A, C, D)$

Lossless join

$R_1(A, B)$
$R_2(A, C, D)$

We can see that we have achieved the preservation of the attributes since

$\{A, B\} \cup \{A, C, D\} = \{A, B, C, D\}$

But what about a lossless join?

Is it the case that $R_1 \bowtie R_2 = R$?

Lossless join

A	B
1	a
2	b
1	c
2	a

R_1

A	C	D
1	x	10
2	x	15
1	y	15

R_2

A	B	C	D
1	a	x	10
1	c	x	10
1	a	y	15
1	c	y	15
2	b	x	15
2	a	x	15
2	b	x	10
2	a	x	10

$R_1 \bowtie R_2$

A	B	C	D
1	a	x	10
2	b	x	15
1	c	y	15
2	a	x	10

R

Lossless join

We can see clearly that $R_1 \bowtie R_2 \neq R$.

So, what has gone wrong in this case?

The problem is that the decomposition was incorrect. You cannot decompose a relation arbitrarily. The decomposition must be guided by the functional dependencies.

Lossless join

For the decomposition of relation R into R_1 and R_2 to be lossless, the attributes common to R_1 and R_2 must contain a key for either R_1 or R_2.

If you look again, A is not a key hence cannot be used as the basis of a lossless decomposition.

The join of R_1 and R_2 is called a *lossy* join even though apparently more information has been added. More tuples have been added but they have added erroneous information so we know less about the relation. A lossy join may also result in fewer rows than the original relation.

Lossless join

To determine if a join is lossless, we don't need to create the tables with sample data. There is a simple test to determine if a decomposition has the lossless join property.

Before we start, we note that the set of functional dependencies also has a closure. The closure of a set of FDs is the set of all FDs implied by the set. For a set of FDs F the closure is denoted F^+.

We have looked at how this set is derived in a previous session.

Lossless join

Consider

- A relation R.
- A set of FDs F on R and F^+ the closure of F.
- A decomposition of R: R_1 with attributes $\bar{X}$ and R_2 with attributes $\bar{Y}$.

The decomposition of R is lossless if at least one of these FDs is in F^+

$\bar{X} \cap \bar{Y} \rightarrow \bar{X}$, or
$\bar{X} \cap \bar{Y} \rightarrow \bar{Y}$.

In other words, in the closure of F, an FD must exist with a determinant that is in the set of attributes in both R_1 and R_2 that also determines all of the attributes of R_1 **or** R_2.

The reason this is true is that this property ensures that the attributes involved in the natural join of R_1 and R_2 are a candidate key in at least one of the relations hence we don't get spurious tuples because the value will be unique in at least one of the relations.

Lossless join

Consider a relation $R(A, B, C, D, E)$ with FDs
$F = \{C \to D, D \to E\}$.

Consider a decomposition of R

$R_1(A, C, E), R_2(A, B, D)$

Does this decomposition have the lossless join property?

Firstly, find the intersection of the sets of attributes

$\{A, C, E\} \cap \{A, B, D\} = \{A\}$.

A is not a determinant for all of the attributes in either R_1 or R_2.
The following FDs are not in F^+.

$A \twoheadrightarrow ACE$
$A \twoheadrightarrow ABD$

To see this we need to compute the closure of F.

Therefore, this decomposition does not have the lossless join property.

Lossless join

Consider the same relation $R(A, B, C, D, E)$ with FDs $\{C \rightarrow D, D \rightarrow E\}$.

Consider another decomposition of R

$R_1(C, D, E), R_2(A, B, C)$

Does this decomposition have the lossless join property?

Find the intersection of the sets of attributes

$\{C, D, E\} \cap \{A, B, C\} = \{C\}$.

Checking F^+ (this is a partial list of FDs in F^+)

$C \rightarrow D$
$D \rightarrow E$
$C \rightarrow E$
$C \rightarrow DE$

Therefore, this decomposition does have the lossless join property since C is the determinant of an FD that determines all of the attributes of R_1.

Dependency preservation

Decompositions should preserve data and not add spurious data, as discussed above.

However, relations also have functional dependencies and it would be better if those could also be preserved in the process of decomposition.

This matters because functional dependencies are constraints on relations.

We can determine whether a decomposition is dependency preserving or not.

Dependency preservation

Consider a relation $R(A, B, C, D)$ with FDs $\{A \rightarrow B, B \rightarrow C\}$.

Consider a decomposition of R

$R_1(A, B), R_2(A, C), R_2(A, D)$

Does this decomposition have the dependency preservation property?

Dependency preservation

$R(A, B, C, D)$ with FDs $\{A \rightarrow B, B \rightarrow C\}$.

A decomposition of R

$R_1(A, B), R_2(A, C), R_3(A, D)$

We can see that the FD $B \rightarrow C$ is not preserved in any relation.

If we compute the closure of the FD in each of the decomposed relations, there is no way we can generate $B \rightarrow C$. The only relation that has an FD from the original set is R_1 which has $A \rightarrow B$. If we add that to the closure and follow the rules for deriving FDs we cannot derive $B \rightarrow C$ (indeed we cannot derive anything *other than* $A \rightarrow B$).

Dependency preservation

Consider the same relation $R(A, B, C, D)$ with FDs $\{A \rightarrow B, B \rightarrow C\}$.

Consider another decomposition of R

$R_1(A, B), R_2(B, C), R_2(A, D)$

Does this decomposition have the dependency preservation property?

Dependency preservation

Relation $R(A, B, C, D)$ with FDs $\{A \rightarrow B, B \rightarrow C\}$.

Decomposition of R

$R_1(A, B), R_2(B, C), R_2(A, D)$

We can see that all FDs are preserved directly in the relations. No derivation is necessary.

Dependency preservation

We want to know if we have lost any functional dependencies in a decomposition. We want to try and avoid doing that.

If F is the set of FDs over a relation R, and $\{R_1, R_2, ..., R_n\}$ is a decomposition of R, and F_i is the set of FDs over R_i, then we require

$$\{F_1 \cup F_2 \cup ... \cup F_n\}^+ = F^+$$

That is, the closure of the sets of FDs for all relations in the decomposition should be equal to the closure of the FDs for the original relation. So, the FD must exist or be derivable. It is better if each FD is preserved in a relation and not dependent on a join to recreate it, to avoid the use of a join to check dependency constraints.

Dependency preservation

The set of dependencies F_1, for example, is the set of dependencies from F that have the attributes in R_1 as determinants. The set of dependencies F_2 is the set of dependencies from F that have the attributes in R_2 as determinants, and so on.

A decomposed relation may have **none** of the FDs from F in it. That is OK. We are looking to preserve the original set of FDs (and FDs implied by the original set).

When we take all of the FDs in F_1, F_2, etc. together, their closure (i.e. all of the FDs implied by these FDs), should be equal to the original set of FDs. If that is the case, then we have not lost any FDs in the decomposition.

It is preferable if each FD is preserved in a single relation. However, for the dependency preservation property, it is simply enough that all original FDs are derivable.

Dependency preservation

Consider relation $R(A, B, C)$ with FDs $\{AB \rightarrow C, C \rightarrow B\}$.

And a decomposition of R

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

This is a lossless decomposition, but is it dependency preserving?

No. The FD $AB \rightarrow C$ is lost and cannot be derived.

Dependency preservation

Consider relation $R(A, B, C, D, E, F)$ with FDs $\{ABC \rightarrow DEF, BC \rightarrow E, C \rightarrow B\}$.

And a decomposition of R

$R_1(A, B, C, D, F), R_2(B, C, E)$

Is this decomposition dependency preserving?

We can see immediately that $BC \rightarrow E$ and $C \rightarrow B$ are preserved in R_2, but what about $ABC \rightarrow DEF$? That is not immediately apparent.

Dependency preservation

Relation $R(A, B, C, D, E, F)$ with FDs
$\{ABC \rightarrow DEF, BC \rightarrow E, C \rightarrow B\}$.

Decomposition: $R_1(A, B, C, D, F), R_2(B, C, E)$

We have

$ABC \rightarrow D$ (in R_1)
$ABC \rightarrow F$ (in R_1)
$BC \rightarrow E$ (in R_2)
$ABC \rightarrow AE$(using the augmentation rule)
$ABC \rightarrow E$ (using the splitting rule)
$ABC \rightarrow DEF$ (using the combining rule)

Thus the dependencies are preserved by this decomposition.

Dependency preservation

Consider relation $R(A, B, C, D, E, F)$ with *different* FDs $\{ABC \rightarrow DEF, BC \rightarrow F, EF \rightarrow B\}$.

And another decomposition of R

$R_1(A, B, C, D, F), R_2(B, E, F)$

Is this decomposition dependency preserving?

We can see immediately that $EF \rightarrow B$ and $BC \rightarrow F$ are preserved in R_2, but what about $ABC \rightarrow DEF$? That is not immediately apparent.

Dependency preservation

Relation $R(A, B, C, D, E, F)$ with FDs
$\{ABC \rightarrow DEF, BC \rightarrow F, EF \rightarrow B\}$.

Decomposition: $R_1(A, B, C, D, F), R_2(B, E, F)$

Is this decomposition dependency preserving?

In this case we cannot find $ABC \rightarrow E$. It is not represented.

Determining if relations are in BCNF

Starting with relation $R(A, B, C, D, E, F)$ with FDs $\{ABC \rightarrow DEF, CDE \rightarrow A\}$, we can check if different decompositions are in BCNF.

Firstly, we should check that R is not already in BCNF?

We require that the determinant of all FDs be a key.

For $ABC \rightarrow DEF$ the determinant is clearly a key since $\{A, B, C\}^+ = \{A, B, C, D, E, F\}$.

But for $CDE \rightarrow A$ the determinant is not a key.
Hence R is not in BCNF.

Determining if relations are in BCNF

Consider another decomposition of R.

Decomposition: $R_1(A, B, C), R_2(D, E, F)$.

Neither of these relations includes an FD. They do, however, both have minimal candidate keys equal to the trivial superkey.

In this case both relations are in BCNF since, in this trivial case, all of the attributes in both relations are part of a key.

Determining if relations are in BCNF

Consider another case.

Decomposition: $R_1(B, C, D, E, F), R_2(C, D, E, A)$.

Again, R_1 has no dependencies, hence its key is the entire schema and it is trivially in BCNF.

R_2 has the FD $CDE \rightarrow A$ hence $\{C, D, E\}$ is in BCNF.