Normalisation to BCNF

Database Systems Lecture 12
Natasha Alechina

In This Lecture

- More normalisation
- Brief review of relational algebra
- Lossless decomposition
- Boyce-Codd normal form (BCNF)
- Higher normal forms
- Denormalisation
- For more information
  - Connolly and Begg chapter 14

Normalisation so Far

- First normal form
  - All data values are atomic
- Second normal form
  - In 1NF plus no non-key attribute is partially dependent on a candidate key
- Third normal form
  - In 2NF plus no non-key attribute depends transitively on a candidate key

Lossless decomposition

- To normalise a relation, we used projections
  - If R(A,B,C) satisfies A → B
    then we can project it on A,B and A,C without losing information
- Lossless decomposition: R = π_{AB}(R) ⋈ π_{AC}(R)
  where π_{AB}(R) is projection of R on AB and ⋈ is natural join.

Reminder of projection:

\[
\begin{array}{cccc}
R & \pi_{AB}(R) & \pi_{AC}(R) \\
A & B & C & A & B \\
1 & x & c & c & c
\end{array}
\]

Relational algebra reminder: selection

\[
\begin{array}{cccc}
R & \sigma_{C=a}(R) \\
A & B & C & D \\
1 & x & c & c \\
2 & y & a & a \\
3 & z & b & y \\
4 & u & c & u \\
5 & w & d & w
\end{array}
\]

Relational algebra reminder: product

\[
\begin{array}{cccc}
R_1 & R_2 & R_1 \times R_2 \\
A & B \\
1 & x \\
2 & y \\
C \\
1 & w \\
2 & v \\
3 & u \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & A & C \\
1 & x & 1 & w \\
2 & y & 2 & v \\
3 & z & 3 & u \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & A & C \\
1 & x & 1 & w \\
2 & y & 2 & v \\
3 & z & 3 & u \\
\end{array}
\]
While I am on the subject...

```
SELECT A,B
FROM R1, R2, R3
WHERE (some property \( \alpha \) holds)
```

translates into relational algebra

```
\pi_{A,B} \sigma_{\alpha}(R1 \times R2 \times R3)
```

Relational algebra: natural join

\[ R1 \bowtie R2 = \pi_{R1.A,R2.B,R3.C} \sigma_{R1.A = R2.A}(R1 \times R2) \]

When is decomposition lossless:

Module \rightarrow Lecturer

<table>
<thead>
<tr>
<th>R</th>
<th>\pi_{\text{Module,Lecturer}}R</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBS</td>
<td>nza</td>
</tr>
<tr>
<td>DBS</td>
<td>nza</td>
</tr>
<tr>
<td>RDB</td>
<td>nza</td>
</tr>
<tr>
<td>APS</td>
<td>nza</td>
</tr>
</tbody>
</table>

\[ \pi_{\text{Module,Lecturer}}R = \pi_{\text{Module,Text}}R \]

<table>
<thead>
<tr>
<th>S</th>
<th>\pi_{\text{First,Last}}S</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Smith</td>
</tr>
<tr>
<td>John</td>
<td>Brown</td>
</tr>
<tr>
<td>John</td>
<td>Brown</td>
</tr>
<tr>
<td>Mary</td>
<td>Smith</td>
</tr>
</tbody>
</table>

\[ \pi_{\text{First,Last}}S \bowtie \pi_{\text{First,Last}}S \]

<table>
<thead>
<tr>
<th>S</th>
<th>\pi_{\text{First,Age}}S</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Smith</td>
</tr>
<tr>
<td>John</td>
<td>Brown</td>
</tr>
<tr>
<td>John</td>
<td>Brown</td>
</tr>
<tr>
<td>Tom</td>
<td>Brown</td>
</tr>
</tbody>
</table>

When is decomposition is not lossless: no fd

\[ \pi_{\text{First,Last}}S = \pi_{\text{First,Last}}S \]

<table>
<thead>
<tr>
<th>S</th>
<th>\pi_{\text{First,Age}}S</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Smith</td>
</tr>
<tr>
<td>John</td>
<td>Brown</td>
</tr>
<tr>
<td>John</td>
<td>Brown</td>
</tr>
<tr>
<td>Tom</td>
<td>Brown</td>
</tr>
</tbody>
</table>

Normalisation Example

- We have a table representing orders in an online store
- Each entry in the table represents an item on a particular order

<table>
<thead>
<tr>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
</tr>
<tr>
<td>Product</td>
</tr>
<tr>
<td>Customer</td>
</tr>
<tr>
<td>Address</td>
</tr>
<tr>
<td>Quantity</td>
</tr>
<tr>
<td>UnitPrice</td>
</tr>
</tbody>
</table>

Primary key is \{Order, Product\}
Functional Dependencies

- Each order is for a single customer
- Each customer has a single address
- Each product has a single price
- From FDs 1 and 2 and transitivity
  \[ \text{Order} \rightarrow \text{Customer} \]
  \[ \text{Customer} \rightarrow \text{Address} \]
  \[ \text{Product} \rightarrow \text{UnitPrice} \]
  \[ \text{Order} \rightarrow \text{Address} \]

Normalisation to 2NF

- Second normal form means no partial dependencies on candidate keys
  \[ \text{Order} \rightarrow \text{Customer, Address} \]
  \[ \text{Product} \rightarrow \text{UnitPrice} \]
- To remove the first FD we project over
  \[ \text{Order, Customer, Address} \] (R1)
  \[ \text{Order, Product, Quantity, UnitPrice} \] (R2)

Normalisation to 3NF

- R1 is now in 2NF, but there is still a partial FD in R2
  \[ \text{Product} \rightarrow \text{UnitPrice} \]
- To remove this we project over
  \[ \text{Product, UnitPrice} \] (R3)
  \[ \text{Order, Product, Quantity} \] (R4)
- R3 and R4 are in 3NF
- R1 has a transitive FD on its key
- To remove
  \[ \text{Order} \rightarrow \text{Customer} \]
  \[ \text{Customer} \rightarrow \text{Address} \]
- we project R1 over
  \[ \text{Order, Customer} \]
  \[ \text{Customer, Address} \]

Normalisation

- 1NF:
  \[ \text{Order, Product, Customer, Address, Quantity, UnitPrice} \]
- 2NF:
  \[ \text{Order, Customer, Address}, \text{Product, UnitPrice}, \text{Order, Product, Quantity} \]
- 3NF:
  \[ \text{Product, UnitPrice}, \text{Order, Product, Quantity}, \text{Order, Customer}, \text{Customer, Address} \]

The Stream Relation

- Consider a relation, Stream, which stores information about times for various streams of courses
- For example: labs for first years
- Each course has several streams
- Only one stream (of any course at all) takes place at any given time
- Each student taking a course is assigned to a single stream for it
The Stream Relation

<table>
<thead>
<tr>
<th>Student</th>
<th>Course</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Databases</td>
<td>12:00</td>
</tr>
<tr>
<td>Mary</td>
<td>Databases</td>
<td>12:00</td>
</tr>
<tr>
<td>Richard</td>
<td>Databases</td>
<td>15:00</td>
</tr>
<tr>
<td>Richard</td>
<td>Programming</td>
<td>10:00</td>
</tr>
<tr>
<td>Mary</td>
<td>Programming</td>
<td>10:00</td>
</tr>
<tr>
<td>Rebecca</td>
<td>Programming</td>
<td>13:00</td>
</tr>
</tbody>
</table>

Candidate keys: (Student, Course) and (Student, Time)

FDs in the Stream Relation

- Stream has the following non-trivial FDs
- \((\text{Student, Course}) \rightarrow (\text{Time})\)
- \((\text{Time}) \rightarrow (\text{Course})\)
- Since all attributes are key attributes, Stream is in 3NF

Anomalies in Stream

- INSERT anomalies
  - You can't add an empty stream
- UPDATE anomalies
  - Moving the 12:00 class to 9:00 means changing two rows
- DELETE anomalies
  - Deleting Rebecca removes a stream

Boyce-Codd Normal Form

- A relation is in Boyce-Codd normal form (BCNF) if for every FD \(A \rightarrow B\) either
  - \(B\) is contained in \(A\) (the FD is trivial), or
  - \(A\) contains a candidate key of the relation,
- In other words: every determinant in a non-trivial dependency is a (super) key.
- The same as 3NF except in 3NF we only worry about non-key Bs
- If there is only one candidate key then 3NF and BCNF are the same

Stream and BCNF

- Stream is not in BCNF as the FD \(\{\text{Time}\} \rightarrow \{\text{Course}\}\) is non-trivial and \(\{\text{Time}\}\) does not contain a candidate key

Conversion to BCNF

Stream has been put into BCNF but we have lost the FD \((\text{Student, Course}) \rightarrow (\text{Time})\)
Decomposition Properties

- Lossless: Data should not be lost or created when splitting relations up.
- Normalisation to 3NF is always lossless and dependency preserving.
- Normalisation to BCNF is lossless, but may not preserve all dependencies.

Higher Normal Forms

- BCNF is as far as we can go with FDs.
- Normalisation to BCNF is lossless, but may not preserve all dependencies.

Normalisation in Exams

(i) Explain why this relation is in second normal form, but not in third normal form.

(ii) Show how this relation can be converted to third normal form. You should show what functional dependencies are being removed, explain why they need to be removed, and give the relation(s) that result.

(iii) Give an example of a relation that is in third normal form, but that is not in Boyce-Codd normal form, and explain why it is in third, but not Boyce-Codd, normal form.
Next Lecture

- Physical DB Issues
  - RAID arrays for recovery and speed
  - Indexes and query efficiency
- Query optimisation
  - Query trees
- For more information
  - Connolly and Begg chapter 21 and appendix C.5