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
  - Ullman and Widom chapter 3.6

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 $A \rightarrow B$ then we can project it on $A,B$ and $A,C$ without losing information
- Lossless decomposition: $R = \pi_{AB}(R) \bowtie \pi_{AC}(R)$
  - $\pi_{AB}(R)$ is projection of $R$ on $AB$ and $\bowtie$ is natural join.

Relational algebra reminder: selection

<table>
<thead>
<tr>
<th>R</th>
<th>$\sigma_{C=\alpha}(R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
</tr>
<tr>
<td>3</td>
<td>z</td>
</tr>
<tr>
<td>4</td>
<td>u</td>
</tr>
<tr>
<td>5</td>
<td>w</td>
</tr>
</tbody>
</table>

Relational algebra reminder: product

<table>
<thead>
<tr>
<th>R1</th>
<th>R2</th>
<th>R1\times R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>u</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>z</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>u</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>w</td>
<td>2</td>
</tr>
</tbody>
</table>

Connection to SQL

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

$R1 \Join R2 = \pi_{R1.A,B,C} \sigma_{R1.A = R2.A}(R1 \times R2)$

When is decomposition lossless: Module $\rightarrow$ Lecturer

<table>
<thead>
<tr>
<th>Module</th>
<th>Lecture</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBS</td>
<td>nza</td>
<td>CB</td>
</tr>
<tr>
<td>DBS</td>
<td>nza</td>
<td>UW</td>
</tr>
<tr>
<td>RDB</td>
<td>nza</td>
<td>UW</td>
</tr>
<tr>
<td>APS</td>
<td>nzb</td>
<td>B</td>
</tr>
</tbody>
</table>

$\pi_{\text{Module}, \text{Lecturer}} R$

$\pi_{\text{Module}, \text{Text}} R$

When is decomposition is not lossless: no fd

<table>
<thead>
<tr>
<th>First Age</th>
<th>First Last</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith 20</td>
<td>John Smith 10</td>
<td>20</td>
</tr>
<tr>
<td>John Brown 30</td>
<td>John Brown 20</td>
<td>30</td>
</tr>
<tr>
<td>Tom Brown 10</td>
<td>Tom Brown 10</td>
<td>10</td>
</tr>
</tbody>
</table>

$\pi_{\text{First,Last}} S$

$\pi_{\text{First,Age}} S$

Heath’s theorem

- A relation $R(A,B,C)$ that satisfies a functional dependency $A \rightarrow B$ can always be non-loss decomposed into its projections $R1 = \pi_{A,B}(R)$ and $R2 = \pi_{C}(R)$.

Proof:
- First we show that $R \subseteq \pi_{A,B}(R) \bowtie \pi_{C}(R)$. This actually holds for any relation, does not have to satisfy $A \rightarrow B$.
- Assume $r \in R$. We need to show $r \in \pi_{A,B}(R) \bowtie \pi_{C}(R)$. Since $r \in R$, $r(A,B) \in \pi_{A,B}(R)$ and $r(A,C) \in \pi_{C}(R)$. Since $r(A,B)$ and $r(A,C)$ have the same value for $A$, their join $r(A,B,C) = r$ is in $\pi_{A,B}(R) \bowtie \pi_{C}(R)$.
Heath’s theorem

- Now we show that $\pi_B(R) \Rightarrow \pi_C(R)$. This only holds if $R$ satisfies $A \rightarrow B$.
- Assume $r \in \pi_B(R) \Rightarrow \pi_C(R)$.
- So, $r(A,B) \in \pi_B(R)$ and $r(A,C) \in \pi_C(R)$.
- By the definition of projection, if $r(A,B) \in \pi_B(R)$, then there is a tuple $s_1 \in R$ such that $s_1(A,B) = r(A,B)$.
- Similarly, since $r(A,C) \in \pi_C(R)$, there is $s_2 \in R$ such that $s_2(A,C) = r(A,C)$.
- Since $s_1(A,B) = r(A,B)$ and $s_2(A,C) = r(A,C)$, $s_1(A) = s_2(A)$. So because of $A \rightarrow B$, $s_1(B) = s_2(B)$. This means that $s_1(A,B,C) = s_2(A,B,C) = r$ and $r \in R$.

Normalisation Example

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

- Columns
  - Order
  - Product
  - Customer
  - Address
  - Quantity
  - UnitPrice
- Primary key is
  - $\{\text{Order, Product}\}$

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
  - $\{\text{Order}\} \rightarrow \{\text{Customer, Address}\}$
  - $\{\text{Product}\} \rightarrow \{\text{UnitPrice}\}$
  - $\{\text{Order}\} \rightarrow \{\text{Customer, Address}\}$
  - $\{\text{Product}\} \rightarrow \{\text{UnitPrice}\}$
- To remove the first FD we project over
  - $\{\text{Order, Customer, Address}\}$ (R1)
  - $\{\text{Order, Product, Quantity, UnitPrice}\}$ (R2)

Normalisation to 3NF

- R has now been split into 3 relations - R1, R2, and R3
- $R_1$ has a transitive FD on its key
- To remove
  - $\{\text{Order}\} \rightarrow \{\text{Customer}\}$
  - $\{\text{Order, Customer}\}$
- we project $R_1$ over
  - $\{\text{Order, Customer}\}$
  - $\{\text{Customer, Address}\}$
Normalisation

- **1NF:**
  - \(\text{(Order, Product, Customer, Address, Quantity, UnitPrice)}\)
- **2NF:**
  - \(\text{(Order, Customer, Address), (Product, UnitPrice), and (Order, Product, Quantity)}\)
- **3NF:**
  - \(\text{(Product, UnitPrice), (Order, Product, Quantity), (Order, Customer), and (Customer, Address)}\)

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
  - \(\text{(Student, Course) \rightarrow (Time)}\)
  - \(\text{(Time) \rightarrow (Course)}\)
- 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 B’s
- 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.

<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>

Conversion to BCNF

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

Decomposition Properties

- Lossless: Data should not be lost or created when splitting relations up.
- Dependency preservation: It is desirable that FDs are preserved 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.
  - Higher normal forms are based on other sorts of dependency.
  - Fourth normal form removes multi-valued dependencies.
  - Fifth normal form removes join dependencies.

Denormalisation

- Normalisation
  - Removes data redundancy
  - Solves INSERT, UPDATE, and DELETE anomalies
  - This makes it easier to maintain the information in the database in a consistent state
- However
  - It leads to more tables in the database.
  - Often these need to be joined back together, which is expensive to do.
  - So sometimes (not often) it is worth ‘denormalising’.

Denormalisation

- You might want to denormalise if:
  - Database speeds are unacceptable (not just a bit slow).
  - There are going to be very few INSERTs, UPDATEs, or DELETEs.
  - There are going to be lots of SELECTs that involve the joining of tables.

Address

<table>
<thead>
<tr>
<th>Number</th>
<th>Street</th>
<th>City</th>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address 1</td>
<td>Number</td>
<td>Street</td>
<td>Postcode</td>
</tr>
<tr>
<td>Address 2</td>
<td>Postcode</td>
<td>City</td>
<td></td>
</tr>
</tbody>
</table>
Normalisation in exams (and the last coursework)

- Consider a relation Book with attributes Author, Title, Publisher, City, Country, Year, ISBN. There are two candidate keys: ISBN and (Author, Title, Publisher, Year). City is the place where the book is published, and there are functional dependencies Publisher → City and City → Country. Is this relation in 2NF? Explain your answer. (4 marks)

- Is this relation in 3NF? Explain your answer. (5 marks)

- Is the relation above in BCNF? If not, decompose it to BCNF and explain why the resulting tables are in BCNF. (5 marks)

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, Ullman and Widom 5.2.8