Detailed Data Modelling: Attribute Collection and Normalisation of Data (2) Worked Example
Normalisation

An orderly three-step process:

- **1NF** - Identify PKs, remove repeating groups
- **2NF** - Remove partial dependencies
- **3NF** - Remove transitive dependencies
1NF 1\textsuperscript{st} Normal Form

Identify PKs, remove repeating groups

- Determine which attributes in a relation are necessary to uniquely ID an instance of that relation
- Determine which groups of attributes repeat, and where in the relation they repeat
- Remove repeating groups
- An entirely precise mathematical process!
1NF 1\textsuperscript{st} Normal Form

An precise mathematical process!

- For any relation $R$
  $$R \ (a, b, (c, d, (e, f)))$$

  Remove repeating groups

  $$R_a \ (a, b)$$
  $$R_{a,c} \ (a, c, d)$$
  $$R_{a,c,e} \ (a, c, e, f)$$

- Just separate each group and combine the keys of each group in which it nests
2NF 2\textsuperscript{nd} Normal Form

Remove partial dependencies

- Test each part of the key against each other part of the key
- Test each part of the key against each non-key attribute
- Remove any partial dependencies
- Replace the PK of the dependency back in the group from which it was removed
  - Becomes the FK link to the dependent group
3NF 3rd Normal Form

Remove transitive dependencies

- Test each non-key attribute against each other non-key attribute

- Remove any transitive dependencies

- Replace the PK of the dependency back in the group from which it was removed
  - Becomes the FK link to the dependent group
3NF 3\textsuperscript{rd} Normal Form

Remove transitive dependencies

- Collect all final 3NF relations

- Merge any 3NF relations with the same PK

- Check the merged relation for any reintroduced dependencies and remove them

- Assemble final list of 3NF relations for conversion to DSD
Normalisation – Worked Example

<table>
<thead>
<tr>
<th>Cust No</th>
<th>Cust Name</th>
<th>Cust Address</th>
<th>Rental Order No</th>
<th>Rental Order Date</th>
<th>Rental Duration</th>
<th>Video No</th>
<th>Video Name</th>
<th>Video Copy No</th>
<th>Daily Rental Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Manoora College</td>
<td>Bundoora</td>
<td>200</td>
<td>1/10/01</td>
<td>2 days</td>
<td>111</td>
<td>History</td>
<td>1</td>
<td>$5.00</td>
</tr>
<tr>
<td>100</td>
<td>Manoora College</td>
<td>Bundoora</td>
<td>200</td>
<td>1/10/01</td>
<td>3 days</td>
<td>111</td>
<td>History</td>
<td>2</td>
<td>$5.00</td>
</tr>
<tr>
<td>100</td>
<td>Manoora College</td>
<td>Bundoora</td>
<td>230</td>
<td>2/10/01</td>
<td>5 days</td>
<td>222</td>
<td>Science</td>
<td>5</td>
<td>$8.00</td>
</tr>
<tr>
<td>120</td>
<td>Manoora College</td>
<td>Banrock</td>
<td>201</td>
<td>1/10/01</td>
<td>2 days</td>
<td>111</td>
<td>History</td>
<td>4</td>
<td>$5.00</td>
</tr>
<tr>
<td>140</td>
<td>Carrum High</td>
<td>Carrum</td>
<td>229</td>
<td>4/10/01</td>
<td>4 days</td>
<td>222</td>
<td>Science</td>
<td>3</td>
<td>$8.00</td>
</tr>
<tr>
<td>140</td>
<td>Carrum High</td>
<td>Carrum</td>
<td>229</td>
<td>4/10/01</td>
<td>7 days</td>
<td>444</td>
<td>Maths</td>
<td>2</td>
<td>$10.00</td>
</tr>
<tr>
<td>170</td>
<td>Berwick College</td>
<td>Berwick</td>
<td>245</td>
<td>12/10/01</td>
<td>2 days</td>
<td>111</td>
<td>History</td>
<td>1</td>
<td>$5.00</td>
</tr>
</tbody>
</table>

Firstly,

- Establish rules apparent in dataset
- Establish rules apparent in provided specifications
- Understand nature of system
Normalisation – Worked Example

Apparent Rules

- Cust No uniquely IDs customer but names are not unique
- Customers can have many orders
- Orders can have many videos
- Videos have many copies
- Different copies of same video can have different loan durations
- Each video has a fixed rental rate
- More than one copy of the same video title can be rented on the same rental
- An order is for the video copy not the video name
Normalisation – Worked Example

Our dataset can be expressed as the following unsorted relation:

CUSTOMER (Cust No, Cust Name, Cust Address, Rental Order No, Rental Order Date, Rental duration, Video No, Video Name, Video Copy No, Daily Rental Rate)

This needs rearranging to make sense because

- The daily rental rate is a feature of the particular video title being rented
- The rental duration is a feature of the particular video copy being rented
Normalisation – Worked Example

Rearranging we end up with:

CUSTOMER (Cust No, Cust Name, Cust Address, Rental Order No, Rental Order Date, Video No, Video Name, Daily Rental Rate, Video Copy No, Rental Duration)

ID PKs and repeating groups:

CUSTOMER (Cust No, Cust Name, Cust Address, (Rental Order No, Rental Order Date, (Video No, Video Name, Daily Rental Rate, (Video Copy No, Rental Duration)))))

- This relation reflects the fact that the copy has a duration, and the video name has a rate
- A 4-part PK is necessary to uniquely ID a particular rented video copy
Normalisation – Worked Example

Remove repeating groups

CUSTOMER (Cust No, Cust Name, Cust Address, (Rental Order No, Rental Order Date, (Video No, Video Name, Daily Rental Rate, (Video Copy No, Rental Duration))))

CUSTOMER (Cust No, Cust Name, Cust Address)
CUSTOMER_RENTAL_ORDER (Cust No, Rental Order No, Rental Order Date)
CUSTOMER_RENTAL_ORDER_VIDEO (Cust No, Rental Order No, Video No, Video Name, Daily Rental Rate)
CUSTOMER_RENTAL_ORDER_VIDEO_COPY (Cust No, Rental Order No, Video No, Video Copy No, Rental Duration)

These are our 1NF relations
Normalisation – Worked Example

- Now we must remove partial dependencies to get to 2NF
  - Remember it is impossible to have partial dependencies if a relation only has a single-part key

So… the 1NF relation CUSTOMER CUSTOMER (Cust No, Cust Name, Cust Address) is already in 2NF.
Normalisation – Worked Example

CUSTOMER_RENTAL_ORDER
(Cust No, Rental Order No, Rental Order Date)

Testing this relation we find
• for each value of Rental Order No, there is only ever one value of Customer No
• for each value of Rental Order No, there is only ever one value of Rental Order Date

• Because both attributes, Customer No and Rental Order Date comprise the whole relation and are both dependent on the same part of the key, Rental Order No, we do not ‘remove’ the dependency by separating the relation – we make Rental Order No the sole PK, and make Customer No a foreign key.

This leaves us with the 2NF relation

CUSTOMER_RENTAL_ORDER
(Cust No, Rental Order No, Rental Order Date)
Normalisation – Worked Example

CUSTOMER_RENTAL_ORDER_VIDEO

(Cust No, Rental Order No, Video No, Video Name, Daily Rental Rate)

Testing this relation we find

- for each value of Rental Order No, there is only ever one value of Customer No
- for each value of Video No, there is only ever one value of Video Name and Daily Rental Rate

Removing these dependencies leaves us with the 2NF relations

RENTAL_ORDER (Rental Order No, Cust No)
VIDEO (Video No, Video Name, Daily Rental Rate)
RENTAL_ORDER_VIDEO (Rental Order No, Video No)
Normalisation – Worked Example

CUSTOMER_RENTAL_ORDER_VIDEO_COPY

\[ \text{(Cust No, Rental Order No, Video No, Video Copy No, Rental Duration)} \]

Testing this relation we find
- for each value of Rental Order No, there is only ever one value of Customer No

Removing this dependency leaves us with the 2NF relations

CUSTOMER_RENTAL_ORDER \text{(Rental Order No, Cust No)}

RENTAL_ORDER_VIDEO_COPY \text{(Rental Order No, Video No, Video Copy No, Rental Duration)}
Normalisation – Worked Example

So we are left with the final list of 2NF relations:

CUSTOMER (Cust No, Cust Name, Cust Address)
CUSTOMER_RENTAL_ORDER (Cust No, Rental Order No, Rental Order Date)
RENTAL_ORDER (Rental Order No, Cust No)
VIDEO (Video No, Video Name, Daily Rental Rate)
RENTAL_ORDER_VIDEO (Rental Order No, Video No)
CUSTOMER_RENTAL_ORDER (Rental Order No, Cust No)
RENTAL_ORDER_VIDEO_COPY (Rental Order No, Video No, Video Copy No, Rental Duration)

Three of these relations share the same PK, Rental Order No– we can merge these and end up with five 2NF relations

CUSTOMER (Cust No, Cust Name, Cust Address)
CUSTOMER_RENTAL_ORDER (Cust No, Rental Order No, Rental Order Date)
VIDEO (Video No, Video Name, Daily Rental Rate)
RENTAL_ORDER_VIDEO (Rental Order No, Video No)
RENTAL_ORDER_VIDEO_COPY (Rental Order No, Video No, Video Copy No, Rental Duration)
Normalisation – Worked Example

• Now we must remove transitive dependencies

CUSTOMER (Cust No, Cust Name, Cust Address)
CUSTOMER_RENTAL_ORDER (Cust No, Rental Order No, Rental Order Date)
VIDEO (Video No, Video Name, Daily Rental Rate)
RENTAL_ORDER_VIDEO (Rental Order No, Video No)
RENTAL_ORDER_VIDEO_COPY (Rental Order No, Video No, Video Copy No, Rental Duration)

• Remember, only relations that have two or more non-key attributes can have transitive dependencies

So…

  RENTAL_ORDER_VIDEO (Rental Order No, Video No)
  RENTAL_ORDER_VIDEO_COPY (Rental Order No, Video No, Video Copy No, Rental Duration)

are already in 3NF
Normalisation – Worked Example

We need to check

CUSTOMER (Cust No, Cust Name, Cust Address)
CUSTOMER_RENTAL_ORDER (Cust No, Rental Order No, Rental Order Date)
VIDEO (Video No, Video Name, Daily Rental Rate)

CUSTOMER and CUSTOMER_RENTAL_ORDER have no transitive dependencies so they are already in 3NF

VIDEO appears it may have a dependency where for each value of Video Name there is only one value of Daily Rental Rate. However, Video Name, while definitely a candidate key, is not as stable as Video No, and may in fact have duplicates.
Therefore no dependency exists and VIDEO is also already in 3NF

This leaves us with the final 3NF relations:
CUSTOMER (Cust No, Cust Name, Cust Address)
CUSTOMER_RENTAL_ORDER (Cust No, Rental Order No, Rental Order Date)
VIDEO (Video No, Video Name, Daily Rental Rate)
RENTAL_ORDER_VIDEO (Rental Order No, Video No)
RENTAL_ORDER_VIDEO_COPY (Rental Order No, Video No, Video Copy No, Rental Duration)
Normalisation

- Remember to test all attributes against each other when looking for dependencies.
- If you find that a key attribute is dependent on a non-key attribute you may not have correctly identified the PKs in UNF-> 1NF – repeat the process.
- Use a table to ensure that you have performed all necessary dependency checks, and that these checks are correct.

<table>
<thead>
<tr>
<th>For each value of</th>
<th>Is there only ever 1 value of</th>
<th>Y/N</th>
<th>Why</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Why Y/N
Is there only ever 1 value of