Advanced Data Management Technologies
Project Module 1 – Data Warehouse

Julian Aron Premner

February 5, 2014
# Contents

1. **Domain, Requirements and Modelling** 3
   1.1 Business Process Modelling 3
   1.2 Dimensions and Bus Matrix 4
   1.3 Queries 6
   1.4 Relevant Measures, Granularity and Additivity 7

2. **Conceptual Design** 9
   2.1 The Extraction Fact 9
   2.2 The Sale Fact 10
   2.3 The Inventory Fact 12

3. **Logical Design** 13
   3.1 Mapping Multiple Arcs 13
   3.2 Snowflake against will 13
   3.3 The time dimension anomaly 13
   3.4 The Extraction Fact 14
   3.5 The Sale Fact 15
   3.6 The Inventory Fact 16
   3.7 Simple Queries 16
      3.7.1 Extraction Fact 16
      3.7.2 Sale Fact 19

4. **Physical Design** 22
   4.1 Simple ROLAP Queries 22

5. **Advanced Querying** 26
   5.1 Ranking Query 26
   5.2 Windowsing Query 26
   5.3 Period-to-Period Query 27
   5.4 Materialized Views 29
1 Domain, Requirements and Modelling

The *Covelano Murble Spa.*, a company in Silandro - South Tyrol, has specialized in the manufacturing, extraction and commercialization of marble and is considering the creation of a data warehouse to aid them in business decisions, mainly where to dig in the quarry and where to extend business efforts. Murble from this quarry has been used to built castles in South Tyrol, churches in Rome, mosques in the Near east and lately in the luxury flats of the One57 skyscraper in New York.

Murble blocks are excavated from the quarry at 2,200m over the sea surface (it is the heighest marble quarry in Europe). These raw blocks are then pre-machined and cut into smaller blocks as to make them safely and easily transportable. At a later time the blocks are brought down by road into the valley to the storehouse next to the factory in Silandro.

Once arrived at the factory, the blocks are roughly sorted by type and vein pattern. While some blocks are left as they are and sold as a whole, most of them are cut into slabs by big machines. Before a great deal of the slabs is cut into tiles however, they are further processed; their surface is treated, depending on their later use. Most slabs that end up as tiles are being sanded and polished. Slabs too, are, if the customer wishes, sold as a whole or cut to a specific size. After the surface has been treated they are examined for cracks, flaws and other imperfections.

Recently, a system has been installed that automatically photographes all the slabs on the conveyor belt immediately after they leave the grinding and polishing machine using a line camera. In the future, the system should also be able to detect, identify and locate flaws and defects of any kind. The data collected in this manner will then be, amongst other data, loaded into the data warehouse in the context of the ETL process.

As one of the last steps, flawed slabs and blocks are processed to gravel and crushed rock and solts intended for processing to tiles are cut into tiles of varying size. Eventually a part of the tiles are treated with resin.

All products are then sorted and moved into the warehouse.

1.1 Business Process Modelling

I proposed the following business processes to be modeled:

**Extraction** Each block extracted from the quarry is recorded, along with important measures such as its dimensions, weight and type. The date of the extraction, the gallery, the height and depth at which the block has been extracted, the date of transportation to the storehouse is stored, as well as the machinery used and the workers involved; the latter two are arguably mock and artificial and have been added mostly to make this example more convoluted and bump the number of dimensions over the required minimum.
Sale Each sale is registered, along with the date of purchase and a detailed customer profile.

Inventory Each product in the warehouse is recorded, together with details such as its dimensions, pattern etc. Additionally, the date of storage, quantity and possible reservations by customers are noted.

1.2 Dimensions and Bus Matrix

Before we can set up a Bus Matrix we need to consider necessary dimensions:

Date Dimension An identical conformed dimension that is shared across all business process fact tables, as can be easily be spotted on the Bus Matrix below. Granularity is discussed at a later point.

Product Dimension The product dimension is by far the biggest of the seven dimensions. It contains all products, that is blocks, slabs and tiles with all their myriad different properties. The product dimension is shared and among different fact tables and mostly identical. It contains the conformed surface dimension.

Surface Dimension This dimension contains information about the quality of the stone’s surface. That is its type (e.g. Black and White or Vena d’Oro) and details about the vein patterns such as color, intensity, angle, thickness and cloudyess.

Customer Dimension This dimension represents a customer profile, that is name, addresses, and other details. It is shared and identical.

Worker Dimension Lists all workers and operators of the company and is used to record who was involved in specific processes. This dimension is only used in the EXTRACTION fact.

Machine Dimension Similarly, this dimension is a list of all the machines the company owns. It allows to find out which machines were used for a given business process. This dimension is only used in the EXTRACTION fact.

Gallery Dimension A simple and straightforward dimension. A quarry usually (and in this case too) has multiple galleries. It is generally interesting to know the originating gallery for a given product. This dimension is only used in the EXTRACTION fact.

Position Dimension A simple dimensions that is mostly comprised of positional tuples (depth, height): each gallery is divided into floors (called ”bancata” by the specialist). Further, a block is excavated at a specific depth (called ” avanzamento” by the specialists). We will simply call them depth and height. This dimension is only used in the EXTRACTION fact.

Dimensions are discussed with more detail in section 2.
Considerations about the Gallery and Position Dimensions  The gallery and position dimensions could be unified to a single new position dimension, possibly with the following functional dependencies defined upon it: position → gallery, position → depth, position → height. But the company is mostly interested in restricting queries to galleries. Thus, in order to simplify queries they were separated. Of course, it is always nice too, to get a dimension for free when there is a minimum number of dimensions required.

Table 1 shows a possible bus matrix for this data warehouse.

<table>
<thead>
<tr>
<th>Date</th>
<th>Product</th>
<th>Surface</th>
<th>Customer</th>
<th>Worker</th>
<th>Machine</th>
<th>Gallery</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extraction</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sale</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inventory</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Simple and straightforward, the Bus Matrix for the planned data warehouse.
1.3 Queries

Table 2 lists possible questions the company would like to have answered.

<table>
<thead>
<tr>
<th>EXTRACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. What is the average amount of stone (in tons) extracted per month per gallery ?</td>
</tr>
<tr>
<td>2. What is the average volume of stone (in m$^3$) extracted per month per gallery ?</td>
</tr>
<tr>
<td>3. What is the average volume of stone (in tons) extracted per month per gallery per worker ?</td>
</tr>
<tr>
<td>4. What is the average volume of stone (in tons) extracted per month per gallery per type ?</td>
</tr>
<tr>
<td>5. What is the gallery from which the most amount of Vena’d Oro (marble of a specific type) has been extracted ?</td>
</tr>
<tr>
<td>6. What is the gallery from which the most amount (volume) of flawed stone has been extracted ?</td>
</tr>
<tr>
<td>7. What is the longest time a block has been exposed directly to the effects of weathering per month (i.e. the time between the extraction and transportation from the quarry to the storehouse) ?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SALE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. What are our best customers ?</td>
</tr>
<tr>
<td>2. Where are our best customers ?</td>
</tr>
<tr>
<td>3. What is the best selling stone type (e.g. Venosta) by continent ?</td>
</tr>
<tr>
<td>4. What are the trends in time in terms of stone type that customers want ?</td>
</tr>
<tr>
<td>5. To which country is most of the stone exported by mass or volume ?</td>
</tr>
<tr>
<td>6. In what country is selling stone most profitable, possibly broken down by year ?</td>
</tr>
</tbody>
</table>
INVENTORY

1. How much stone is stored in our warehouses?
2. How much stone is added on average to the warehouses per month or year, or even day?
3. How much stone is removed on average (hopefully sold) from the warehouses per month or year?
4. Break down the inventory by product type and express as percentages, to give e.g. answer to the question: How much percent of the stone in the stock are blocks, tiles etc.?
5. Refine the previous query and add stone type or vein patterns. That is, to what products, in percentages, has the stone of a certain type been processed?
6. Is there a recognizable trend that we will be out of stock for a specific product in the long future.

Table 2: A few questions that the CTO of the Covelano Murble Spa.
would like to have answered.

It is obvious that the queries for the SALE fact can perfectly be matched with the queries for the EXTRACTION. If the most successful stone type is Silvergold, concentrate on the gallery that yielded most stone of that type as it is likely that there is more there. If the company gets a large order for a specific stone type, or even vein pattern that cannot be satisfied from the stock they have at least a clue where to dig for it.

1.4 Relevant Measures, Granularity and Additivity

In the following, the appropriate fact granularity and measures are discussed to at least be able to answer the queries above. Additional measures and attributes will be added in the conceptual design phase.

As far as the EXTRACTION fact is concerned,

- The hierarchy built on the surface dimension must include at least a type attribute.
- The date hierarchy must include at least the day, month and year attributes.
- The amount is calculated by aggregating the weight measure along any dimension. It is fully additive. It is measured in kilograms.
• The *volume* measure is added when aggregating every dimension and is calculated indirectly from the *height*, *width* and *diameter* measures.

• The *flawed volume* measure is indirectly calculated by summing up the products of the *flawed* measure, which records how much percent of an extracted block had flaws of any kind, and the *volume* measure. It is additive and measured in m³, while the *flawed* measure itself is non-additive.

• The *height*, *width* and *diameter* measures are by themselves non-additive along any dimension (except maybe for things like ”all the blocks we extracted stacked on top of each other extend to the moon and back” etc.). They can, however, be aggregated using $AVG()$, $MIN()$ and $MAX()$ along any dimension.

As far as the *sale* fact is concerned,

• The hierarchy built on the *customer* dimension must include at least an address attribute, country and continent attributes.

• The hierarchy built on the *product* dimension must include at least a *surface type* attribute.

• The date hierarchy must include at least the day, month and year attributes.

• The *quantity* measure is added when aggregating every dimension, i.e. is additive.

• The *profit* measure is added when aggregating every dimension, i.e. is additive.

As far as the *inventory* fact is concerned,

• The *Inventory Periodic Snapshot* model is used.

• The *quantity* measure is added when aggregating along the product and customer dimension. It is non-additive for the *date* dimension.
2 Conceptual Design

The structure of the facts and dimensions has already been briefly alluded to and partly anticipated in the previous chapter. Here, we will succinctly but in detail discuss specific aspects of the modelling, mostly with reference to the *dimensional fact model*, hereafter referred simply as DFM.

The notation used for the DFM models on the following pages mostly follows the notation outlined in [Riz08] and [GR09], although slightly different notations are used there.

It holds true for dimensions, and facts too, that only an arbitrarily selected subset of attributes and measures will be depicted, and that conversely a number of them have been omitted, as the author is thoroughly convinced that there lies no didactic value within complicating this example and the corresponding diagrams.

2.1 The Extraction Fact

The extraction fact (Figure 1) models the fact already mentioned in the analysis section with its measures and dimensions. The date dimension is shared for the roles transport (date of transportation) and excavate (date of excavation from the quarry), which can be, but not necessarily are, the same date.

Since possibly multiple workers and machines are involved in the extraction process, a many-to-many association exists between those dimensions and the fact, denoted by the double line. These "multiple arcs" generally mean the loss of the summarizability property, which can, however, be restored by weighting [LS97]. Since we assume that each worker contributes the same effort a weight is not necessary, at least as far as workers are concerned. Generally, these kind of situations are to be avoided, for they are highly problematic when it comes to mapping the conceptual schema to a logical or physical design. This is especially true for ROLAP. Fair enough, these kind of situations seem to appear not all that rare and I think there should be better ways to deal with them; maybe array or set types, that some of the more modern relational DBMS support (e.g. Postgres, which supports array types) could remedy this problem. In our case, multiple arcs were added to make this otherwise rather dull example a little more exciting.

I slightly deviate from the guideline in [Riz08]. Instead of marking non-additivity with a dashed line, the possible aggregates are listed in parenthesis behind the measures so as not to clutter and make unreadable the diagram.

The opening (i.e. when they first started digging in that gallery) descriptive attribute for the gallery and the last revision (e.g. security checks) attribute for the machine dimension were mostly added to show off notation and to meet the appropriate requirements and are arguable.
Additionally attributes could be added to any of the dimensions, specifically to the worker and machine dimensions, but they would be circumstantial, needlessly complicate everything and consume time spent better elsewhere.

Figure 1: The DFM for the EXTRACTION fact

2.2 The Sale Fact

The customer dimension is incomplete (or ragged) for the city branch, as different nations have different ways of dividing the country into administrative units. Large countries, such as the U.S. have a very thorough subdivision, while e.g. a microstate in constrast is very flat in this regard. There is convergence between the sales district on one and the city, country, country attributes on the other side, for they both determine the continent attribute. Only the most important attributes for the customer dimensions are depicted, in real world additional attributes might be added.
The surface dimension has two multiple arcs for vein and type. This is due to the fact that a product (mostly blocks, but occasionally also slabs) can have patterns of two, three or even more types. Likewise, it might have veins of different colors, intensity etc.

The type attribute does only weakly functionally determine the vein attribute.

Figure 2: The DFM for the SALE fact
2.3 The Inventory Fact

There isn’t much left to say about the INVENTORY fact, for it strongly resembles the sale fact (which is a bit worrying in terms of example chosen).

The fact has two shared date dimensions, that represent the time a product has been stored and reserved, respectively.

The quantity measure is non-additive with respect to the date dimension as far as the store role is concerned.

The date dimension in the reserve role is optional (i.e. an optional arc) since not all products are reserved.

Figure 3: The DFM for the INVENTORY fact
3 Logical Design

We "resist normalization urges" [KR02] and limit ourselves to star schemas (and so escape the pains of drawing snowflake schemas). In the following, the star schemas for our simple examples are depicted. The abbreviations (FK) and (PK) after the measures and attributes stand for foreign key and primary key, respectively.

3.1 Mapping Multiple Arcs

Except for the dimensions with multiple arcs (machine, worker, product, surface, logical integration is straightforward. As for those others, I decided to use bridge tables. Initially I did consider "inlining" by using arrays or hstore (Postgres) in the later physical design step. However, it is very hard to maintain summarizability using e.g hstore because support for range queries on hstore values for specific keys is fairly limited. And so it happened that bridge tables were used for all multiple arcs relations. To keep summarizability, the bridge tables for the surface type and veins need a weight field. The machine and worker bridge tables do not; for it is assumed that weights are evenly distributed amongst workers and machines.

3.2 Snowflake against will

Although a star schema was chosen, the schemas depicted below look suspiciously like a snowflake schema. This is due to the bride tables, that were used to map multiple arcs. This is, of course, not without consequence. Consider a query that is only interested in the vein color attribute, it cannot be carried out without a join, possibly inhibiting performance. In theory, multiple arcs could be "unfold". This would considerably increase the size of the dimension tables, which would however, considering that most of the data is kept in the fact tables anyway, be negligible. Unfortunately, there is, as far as the author is aware, no support for this in any DBMS.

3.3 The time dimension anomaly

It seems to be usual to model the time dimension linearly in the conceptual design; that means we have for instance a functional dependency \( \text{day} \rightarrow \text{month} \), which is of course utter nonsense unless one would e.g. store the days as Julian Day Numbers. Then still, however, the indirect functional dependency \( \text{day} \rightarrow \text{year} \) would be invalid. This could be solved by storing days since epoch (e.g. days since January 1970), but the day field should be easily readable. Now, after having it done the wrong way, I would break up the usual dependencies and make the bottom \( \text{date} \) attribute fully determine all the other attributes in the dimension. Currently, however, the \( \text{day} \) attribute has been logically modelled to use Julian Day Numbers.
3.4 The Extraction Fact

Figure 4: The logical schema for the extraction fact along with its dimensions.
3.5 The Sale Fact

Figure 5: The logical schema for the simpler SALES fact.
3.6 The Inventory Fact

![Diagram of the logical schema for the inventory fact]

Figure 6: The logical schema for the inventory fact, all too similar to the sale fact.

3.7 Simple Queries

We give queries to answer two simple business questions:

3.7.1 Extraction Fact

What is the total amount of stone extracted (volume and mass) in the year 2008 for each gallery?

```sql
SELECT gallery.name, date.year, 
SUM(height * width * diameter) AS volume,
```
\[
\text{SUM(weight) AS mass}
\]

FROM extraction, date, gallery
WHERE extraction.excavation_date_id = date.id
AND extraction.gallery_id = gallery.id
AND date.year = 2008
GROUP BY date.year, gallery.name

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>year</th>
<th>semester</th>
<th>quartal</th>
<th>month</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2007-11-26</td>
<td>2007</td>
<td>2</td>
<td>4</td>
<td>NOV</td>
<td>330</td>
</tr>
<tr>
<td>1</td>
<td>2008-03-08</td>
<td>2008</td>
<td>1</td>
<td>2</td>
<td>MAR</td>
<td>068</td>
</tr>
<tr>
<td>2</td>
<td>2008-04-26</td>
<td>2008</td>
<td>1</td>
<td>2</td>
<td>APR</td>
<td>117</td>
</tr>
<tr>
<td>3</td>
<td>2008-05-07</td>
<td>2008</td>
<td>1</td>
<td>2</td>
<td>MAY</td>
<td>128</td>
</tr>
<tr>
<td>4</td>
<td>2009-01-25</td>
<td>2009</td>
<td>1</td>
<td>1</td>
<td>JAN</td>
<td>025</td>
</tr>
<tr>
<td>5</td>
<td>2009-10-01</td>
<td>2009</td>
<td>2</td>
<td>4</td>
<td>OCT</td>
<td>274</td>
</tr>
<tr>
<td>6</td>
<td>2010-01-04</td>
<td>2010</td>
<td>1</td>
<td>1</td>
<td>JAN</td>
<td>004</td>
</tr>
<tr>
<td>7</td>
<td>2010-05-03</td>
<td>2010</td>
<td>1</td>
<td>2</td>
<td>MAY</td>
<td>123</td>
</tr>
<tr>
<td>8</td>
<td>2010-08-10</td>
<td>2010</td>
<td>2</td>
<td>3</td>
<td>AUG</td>
<td>222</td>
</tr>
<tr>
<td>9</td>
<td>2010-09-24</td>
<td>2010</td>
<td>2</td>
<td>4</td>
<td>SEP</td>
<td>267</td>
</tr>
<tr>
<td>10</td>
<td>2011-11-01</td>
<td>2011</td>
<td>2</td>
<td>4</td>
<td>NOV</td>
<td>305</td>
</tr>
</tbody>
</table>

Table 3: A sample date dimension table – all data is fictional
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>opening</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>G1</td>
<td>2007-07-22</td>
</tr>
<tr>
<td>1</td>
<td>G2</td>
<td>2008-03-31</td>
</tr>
<tr>
<td>2</td>
<td>G3</td>
<td>2008-04-14</td>
</tr>
<tr>
<td>3</td>
<td>G4</td>
<td>2008-07-08</td>
</tr>
<tr>
<td>4</td>
<td>G5</td>
<td>2009-03-04</td>
</tr>
<tr>
<td>5</td>
<td>G6</td>
<td>2009-04-06</td>
</tr>
<tr>
<td>6</td>
<td>G7</td>
<td>2011-08-02</td>
</tr>
<tr>
<td>7</td>
<td>G8</td>
<td>2011-06-02</td>
</tr>
<tr>
<td>8</td>
<td>G9</td>
<td>2011-07-03</td>
</tr>
<tr>
<td>9</td>
<td>G10</td>
<td>2012-04-01</td>
</tr>
<tr>
<td>10</td>
<td>G11</td>
<td>2012-05-10</td>
</tr>
</tbody>
</table>

Table 4: A sample *gallery* dimension table – opening dates are fictional
<table>
<thead>
<tr>
<th>excavation_date_id</th>
<th>transportation_date_id</th>
<th>gallery_id</th>
<th>position_id</th>
<th>machine_group_id</th>
<th>worker_group_id</th>
<th>surface_id</th>
<th>width</th>
<th>height</th>
<th>diameter</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>16</td>
<td>0</td>
<td>2</td>
<td>14</td>
<td>2.89</td>
<td>1.25</td>
<td>1.40</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>7</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>24</td>
<td>2.68</td>
<td>1.26</td>
<td>1.17</td>
<td>11.00</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>9</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>9</td>
<td>2.93</td>
<td>1.25</td>
<td>1.22</td>
<td>12.41</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>9</td>
<td>14</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2.66</td>
<td>1.49</td>
<td>1.39</td>
<td>15.39</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>28</td>
<td>2.79</td>
<td>1.54</td>
<td>1.34</td>
<td>15.94</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>16</td>
<td>2</td>
<td>4</td>
<td>15</td>
<td>2.66</td>
<td>1.49</td>
<td>1.42</td>
<td>15.63</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>21</td>
<td>2.71</td>
<td>1.34</td>
<td>1.16</td>
<td>11.70</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>2.68</td>
<td>1.27</td>
<td>1.32</td>
<td>12.47</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2.75</td>
<td>1.45</td>
<td>1.20</td>
<td>13.27</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>7</td>
<td>14</td>
<td>3</td>
<td>1</td>
<td>13</td>
<td>2.85</td>
<td>1.40</td>
<td>1.34</td>
<td>14.91</td>
</tr>
</tbody>
</table>

Table 5: A sample extraction fact table – all data is fictional, although realistic. Width, height and diameter is given in meters, weight in tons.

<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
<th>Volume</th>
<th>Mass</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>G10</td>
<td>5.598</td>
<td>15.40</td>
</tr>
<tr>
<td></td>
<td>G6</td>
<td>5.798</td>
<td>15.94</td>
</tr>
</tbody>
</table>

Table 6: The result set for the query above.

### 3.7.2 Sale Fact

What is the profit made on the north-american market for the last three years?
<table>
<thead>
<tr>
<th>id</th>
<th>first_name</th>
<th>last_name</th>
<th>address</th>
<th>sales_district</th>
<th>city</th>
<th>county</th>
<th>state</th>
<th>country</th>
<th>continent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Laila</td>
<td>Fend</td>
<td>57 St Georges Hill</td>
<td>ANGLO</td>
<td>Crawcrook and Greenside Ward</td>
<td>Tyne and Wear</td>
<td>NULL</td>
<td>UK</td>
<td>EUROPE</td>
</tr>
<tr>
<td>1</td>
<td>Corrinne</td>
<td>Jaret</td>
<td>2150 Morley St</td>
<td>ANGLO</td>
<td>Dee Ward</td>
<td>Dumfries and Galloway</td>
<td>NULL</td>
<td>UK</td>
<td>EUROPE</td>
</tr>
<tr>
<td>2</td>
<td>Margurite</td>
<td>Loperfido</td>
<td>218 Greenbank Drive</td>
<td>ANGLO</td>
<td>Devizes</td>
<td>Wiltshire</td>
<td>NULL</td>
<td>UK</td>
<td>EUROPE</td>
</tr>
<tr>
<td>3</td>
<td>Pok</td>
<td>Molaison</td>
<td>211 Hobart St</td>
<td>ANGLO</td>
<td>Newquay</td>
<td>Cornwall</td>
<td>NULL</td>
<td>UK</td>
<td>EUROPE</td>
</tr>
<tr>
<td>4</td>
<td>Ahmad</td>
<td>Alsaqri</td>
<td>21 Pickwick St</td>
<td>ANGLO</td>
<td>Sutton cum Dukemanton</td>
<td>Derbyshire</td>
<td>NULL</td>
<td>UK</td>
<td>EUROPE</td>
</tr>
<tr>
<td>5</td>
<td>Murray</td>
<td>Fode</td>
<td>59 W Century Rd</td>
<td>ANGLO</td>
<td>Pointe-Claire</td>
<td>NULL</td>
<td>QC</td>
<td>CA</td>
<td>NORTH AMERICA</td>
</tr>
<tr>
<td>6</td>
<td>Lavelle</td>
<td>Lillywhite</td>
<td>5 S Taylor Ave</td>
<td>ANGLO</td>
<td>La Malbaie</td>
<td>NULL</td>
<td>QC</td>
<td>CA</td>
<td>NORTH AMERICA</td>
</tr>
<tr>
<td>7</td>
<td>Truman</td>
<td>Mondale</td>
<td>1657 N Green St</td>
<td>ANGLO</td>
<td>Peterborough</td>
<td>NULL</td>
<td>ON</td>
<td>CA</td>
<td>NORTH AMERICA</td>
</tr>
<tr>
<td>8</td>
<td>Lea</td>
<td>Steinhaus</td>
<td>80 Maplewood Dr 34</td>
<td>ANGLO</td>
<td>Bradford</td>
<td>NULL</td>
<td>ON</td>
<td>CA</td>
<td>NORTH AMERICA</td>
</tr>
<tr>
<td>9</td>
<td>Olga</td>
<td>Adessa</td>
<td>8507 Upland St</td>
<td>ANGLO</td>
<td>Burlington</td>
<td>NULL</td>
<td>ON</td>
<td>CA</td>
<td>NORTH AMERICA</td>
</tr>
<tr>
<td>10</td>
<td>Stephaine</td>
<td>Barfield</td>
<td>47154 Whipple Ave Nw</td>
<td>ANGLO</td>
<td>Gardena</td>
<td>Los Angeles</td>
<td>CA</td>
<td>US</td>
<td>NORTH AMERICA</td>
</tr>
<tr>
<td>11</td>
<td>Fannie</td>
<td>Lungren</td>
<td>17 Us Highway 111</td>
<td>ANGLO</td>
<td>Round Rock</td>
<td>Williamson</td>
<td>TX</td>
<td>US</td>
<td>NORTH AMERICA</td>
</tr>
<tr>
<td>12</td>
<td>Kasandra</td>
<td>Semidey</td>
<td>369 Latham St 500</td>
<td>ANGLO</td>
<td>Saint Louis</td>
<td>Saint Louis City</td>
<td>MO</td>
<td>US</td>
<td>NORTH AMERICA</td>
</tr>
<tr>
<td>13</td>
<td>Carey</td>
<td>Dopico</td>
<td>87393 E Highland Rd</td>
<td>ANGLO</td>
<td>Indianapolis</td>
<td>Marion</td>
<td>IN</td>
<td>US</td>
<td>NORTH AMERICA</td>
</tr>
<tr>
<td>14</td>
<td>Yuki</td>
<td>Whobrey</td>
<td>1 State Route 27</td>
<td>ANGLO</td>
<td>Taylor</td>
<td>Wayne</td>
<td>MI</td>
<td>US</td>
<td>NORTH AMERICA</td>
</tr>
<tr>
<td>15</td>
<td>Iseal</td>
<td>Calizo</td>
<td>2 Landmeier Rd</td>
<td>ANGLO</td>
<td>Wombeyan Caves</td>
<td>NULL</td>
<td>NS</td>
<td>AU</td>
<td>AUSTRALIA</td>
</tr>
<tr>
<td>16</td>
<td>Coletta</td>
<td>Thro</td>
<td>64865 Main St</td>
<td>ANGLO</td>
<td>North Fremantle</td>
<td>NULL</td>
<td>WA</td>
<td>AU</td>
<td>AUSTRALIA</td>
</tr>
<tr>
<td>17</td>
<td>Carman</td>
<td>Robasciotti</td>
<td>4 Spinning Wheel Ln</td>
<td>ANGLO</td>
<td>Grunya</td>
<td>NULL</td>
<td>VI</td>
<td>AU</td>
<td>AUSTRALIA</td>
</tr>
<tr>
<td>18</td>
<td>Terina</td>
<td>Wildeboer</td>
<td>462 Morris Ave</td>
<td>ANGLO</td>
<td>Seddon</td>
<td>NULL</td>
<td>VI</td>
<td>AU</td>
<td>AUSTRALIA</td>
</tr>
<tr>
<td>19</td>
<td>Leatha</td>
<td>Block</td>
<td>6926 Orange Ave</td>
<td>ANGLO</td>
<td>Two Rocks</td>
<td>NULL</td>
<td>WA</td>
<td>AU</td>
<td>AUSTRALIA</td>
</tr>
</tbody>
</table>

Table 7: A sample customer dimension table with english-speaking customers – all data is fictional
Table 8: A sample sale fact table – all data is fictional.

<table>
<thead>
<tr>
<th>date_id</th>
<th>product_id</th>
<th>customer_id</th>
<th>quantity</th>
<th>profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>13</td>
<td>5</td>
<td>83227</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>371</td>
<td>79598</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>5</td>
<td>350</td>
<td>71346</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>6</td>
<td>264</td>
<td>66579</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>7</td>
<td>15</td>
<td>20067</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>3</td>
<td>110</td>
<td>60668</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>3</td>
<td>251</td>
<td>83514</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>4</td>
<td>99</td>
<td>83287</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>104433</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>5</td>
<td>68</td>
<td>56410</td>
</tr>
</tbody>
</table>

\[
\text{SELECT customer.continent, date.year, } \text{SUM(profit)} \text{ FROM sale, date, customer} \\
\text{WHERE sale.customer_id = customer.id} \\
\text{AND sale.date_id = date.id} \\
\text{AND date.year IN (2009, 2010)} \\
\text{AND customer.continent = 'NORTH_AMERICA'} \\
\text{GROUP BY customer.continent, date.year}
\]

<table>
<thead>
<tr>
<th>Continent</th>
<th>Year</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>North America</td>
<td>2009</td>
<td>66579</td>
</tr>
<tr>
<td></td>
<td>2010</td>
<td>252256</td>
</tr>
</tbody>
</table>

Table 9: The result set for the query above.
4 Physical Design

4.1 Simple ROLAP Queries

```sql
SELECT date.year, date.semester, date.quartal,
    SUM(height * width * diameter) AS volume,
    SUM(weight) AS mass
FROM extraction, date
WHERE extraction.excavation_date_id = date.id
GROUP BY ROLLUP (date.year, date.semester, date.quartal)
```

A note about Postgres    Unfortunately, *Postgres* has at this point in time no support for ROLAP queries; neither ROLLUP nor CUBE groupings are supported, which is surprising, considering that even MySQL has support (although very basic) for these kind of queries. What remains is the possibility to "paraphrase" ROLLUP queries by creating the union of multiple subqueries. The above query would look then as follows. Note that UNION ALL was used, as the UNION keyword defaults to UNION DISTINCT, which would induce unnecessary processing (e.g. sorting). Obviously, this does not perform as well as a corresponding ROLLUP would. Also, controlling the order in which rows should appear seems to be tricky.

```sql
SELECT date.year as year,
    date.semester as semester,
    date.quartal as quartal,
    SUM(height * width * diameter) AS volume,
    SUM(weight) AS mass
FROM extraction, date
WHERE extraction.excavation_date_id = date.id
GROUP BY date.year, date.semester, date.quartal
UNION ALL
SELECT date.year,
    date.semester,
    NULL as "quartal",
    SUM(height * width * diameter) AS volume,
    SUM(weight) AS mass
FROM extraction, date
WHERE extraction.excavation_date_id = date.id
GROUP BY date.year, date.semester
UNION ALL
SELECT date.year,
    date.semester,
    NULL AS "quartal",
    SUM(height * width * diameter) AS volume,
    SUM(weight) AS mass
FROM extraction, date
WHERE extraction.excavation_date_id = date.id
GROUP BY date.year
UNION ALL
SELECT NULL AS year,
NULL AS "semester",
NULL AS "quartal",
SUM(height * width * diameter) AS volume,
SUM(weight) AS mass
FROM extraction, date
WHERE extraction.excavation_date_id = date.id
ORDER BY year, semester, quartal;
<table>
<thead>
<tr>
<th>year</th>
<th>semester</th>
<th>quartal</th>
<th>volume</th>
<th>mass</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>1</td>
<td>1</td>
<td>96.188185836</td>
<td>264.505</td>
</tr>
<tr>
<td>2007</td>
<td>1</td>
<td>2</td>
<td>225.807796467</td>
<td>620.974</td>
</tr>
<tr>
<td>2007</td>
<td>2</td>
<td>3</td>
<td>383.097466294</td>
<td>1053.507</td>
</tr>
<tr>
<td>2007</td>
<td>2</td>
<td>4</td>
<td>230.967760251</td>
<td>635.163</td>
</tr>
<tr>
<td>2007</td>
<td>1</td>
<td></td>
<td>321.995982303</td>
<td>885.479</td>
</tr>
<tr>
<td>2007</td>
<td>2</td>
<td></td>
<td>614.065222545</td>
<td>1688.670</td>
</tr>
<tr>
<td>2007</td>
<td></td>
<td></td>
<td>936.061204848</td>
<td>2574.149</td>
</tr>
<tr>
<td>2008</td>
<td>1</td>
<td>1</td>
<td>333.987827350</td>
<td>918.463</td>
</tr>
<tr>
<td>2008</td>
<td>1</td>
<td>2</td>
<td>275.250892176</td>
<td>756.923</td>
</tr>
<tr>
<td>2008</td>
<td>2</td>
<td>3</td>
<td>183.719783027</td>
<td>505.209</td>
</tr>
<tr>
<td>2008</td>
<td>2</td>
<td>4</td>
<td>237.817976417</td>
<td>653.965</td>
</tr>
<tr>
<td>2008</td>
<td>1</td>
<td></td>
<td>609.238719526</td>
<td>1675.386</td>
</tr>
<tr>
<td>2008</td>
<td>2</td>
<td></td>
<td>421.537759444</td>
<td>1159.174</td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td></td>
<td>1030.776478970</td>
<td>2834.560</td>
</tr>
<tr>
<td>2009</td>
<td>1</td>
<td>1</td>
<td>217.306085913</td>
<td>597.578</td>
</tr>
<tr>
<td>2009</td>
<td>1</td>
<td>2</td>
<td>311.630122536</td>
<td>857.038</td>
</tr>
<tr>
<td>2009</td>
<td>2</td>
<td>3</td>
<td>242.391642842</td>
<td>666.621</td>
</tr>
<tr>
<td>2009</td>
<td>2</td>
<td>4</td>
<td>259.711394078</td>
<td>714.202</td>
</tr>
<tr>
<td>2009</td>
<td>1</td>
<td></td>
<td>528.936208449</td>
<td>1454.616</td>
</tr>
<tr>
<td>2009</td>
<td>2</td>
<td></td>
<td>502.103036920</td>
<td>1380.823</td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td></td>
<td>1031.039245369</td>
<td>2835.439</td>
</tr>
<tr>
<td>2010</td>
<td>1</td>
<td>1</td>
<td>235.904361442</td>
<td>648.704</td>
</tr>
<tr>
<td>2010</td>
<td>1</td>
<td>2</td>
<td>239.787009053</td>
<td>659.440</td>
</tr>
<tr>
<td>2010</td>
<td>2</td>
<td>3</td>
<td>292.159687454</td>
<td>803.368</td>
</tr>
<tr>
<td>2010</td>
<td>2</td>
<td>4</td>
<td>163.936997149</td>
<td>450.822</td>
</tr>
<tr>
<td>2010</td>
<td>1</td>
<td></td>
<td>475.691370495</td>
<td>1308.144</td>
</tr>
<tr>
<td>2010</td>
<td>2</td>
<td></td>
<td>456.095784603</td>
<td>1254.190</td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td></td>
<td>931.787155098</td>
<td>2562.334</td>
</tr>
<tr>
<td>2011</td>
<td>1</td>
<td>1</td>
<td>105.546086406</td>
<td>290.233</td>
</tr>
<tr>
<td>2011</td>
<td>1</td>
<td>2</td>
<td>241.993527613</td>
<td>665.492</td>
</tr>
<tr>
<td>2011</td>
<td>2</td>
<td>3</td>
<td>215.725887074</td>
<td>593.263</td>
</tr>
<tr>
<td>2011</td>
<td>2</td>
<td>4</td>
<td>280.632694755</td>
<td>771.739</td>
</tr>
<tr>
<td>2011</td>
<td>1</td>
<td></td>
<td>347.539614019</td>
<td>955.725</td>
</tr>
<tr>
<td>2011</td>
<td>2</td>
<td></td>
<td>496.358581829</td>
<td>1365.002</td>
</tr>
<tr>
<td>2011</td>
<td></td>
<td></td>
<td>843.889158384</td>
<td>2320.727</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4773.562280133</td>
<td>13127.209</td>
</tr>
</tbody>
</table>

Table 10: The result set for the query given above. Empty cells denote *NULL* values.
Again, this query cannot be run on Postgres. The use of the GROUPING function makes this query even harder to translate into something Postgres can understand.

```
SELECT CASE
    WHEN GROUPING(date.year)=1 THEN 'all years'
    ELSE date.year
END,
CASE
    WHEN GROUPING(date.semester)=1 THEN 'all semesters'
    ELSE date.semester
END,
CASE
    WHEN GROUPING(date.quartal)=1 THEN 'all quartals'
    ELSE date.quartal
END,
SUM(height * width * diameter) AS volume,
SUM(weight) AS mass
FROM extraction, date
WHERE extraction.excavation_date_id = date.id
GROUP BY ROLLUP(date.year, date.semester, date.quartal)
```

<table>
<thead>
<tr>
<th>year</th>
<th>semester</th>
<th>quartal</th>
<th>volume</th>
<th>mass</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>1</td>
<td>1</td>
<td>96.188185836</td>
<td>264.505</td>
</tr>
<tr>
<td>2007</td>
<td>1</td>
<td>2</td>
<td>225.807796467</td>
<td>620.974</td>
</tr>
<tr>
<td>2007</td>
<td>2</td>
<td>3</td>
<td>383.897466294</td>
<td>1053.507</td>
</tr>
<tr>
<td>2007</td>
<td>2</td>
<td>4</td>
<td>230.967756251</td>
<td>635.163</td>
</tr>
<tr>
<td>2007</td>
<td>1</td>
<td>all quartals</td>
<td>321.995982303</td>
<td>885.479</td>
</tr>
<tr>
<td>2007</td>
<td>2</td>
<td>all quartals</td>
<td>614.065222545</td>
<td>1688.670</td>
</tr>
<tr>
<td>2007</td>
<td>all semesters</td>
<td>all quartals</td>
<td>936.061204848</td>
<td>2574.149</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>all years</td>
<td>all semesters</td>
<td>all quartals</td>
<td>4773.562280133</td>
<td>13127.209</td>
</tr>
</tbody>
</table>

Table 11: The result set for the query given above. The NULL values have been replaced with a descriptive value.
5 Advanced Querying

5.1 Ranking Query

In each sales district, financial resources (e.g. for advertising) are distributed evenly across its countries; however, if sales in a country were good it gets an additional bonus. In order not to disadvantage any sales district, the bonuses depend only on how well the other countries in the same district did. The countries in a district are sorted by total profit, then the countries in the top 30% get a "high" bonus, the following top 30% a "low" bonus and the remaining lower 30% get no bonus at all. The following query helps the management to assign bonus to the countries in the sales district for the English-speaking countries. The best and worst profit sale is removed to expunge outliers and increase fairness.

Listing 1: For each country in the 'ANGLO' sales district, list the minimum and maximum profit, and its "bonus category".

```sql
SELECT customer.country,
       SUM(sale.profit),
       MIN(sale.profit),
       MAX(sale.profit),
       NTILE(3) OVER (ORDER BY (SUM(sale.profit) - MIN(sale.profit) - MAX(sale.profit)))
FROM sale, customer
WHERE sale.customer_id = customer.id
AND customer.sales_district = 'ANGLO'
GROUP BY customer.country;
```

<table>
<thead>
<tr>
<th>country</th>
<th>total profit</th>
<th>minimum profit</th>
<th>maximum profit</th>
<th>bonus category</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU</td>
<td>13578425.000</td>
<td>10700.000</td>
<td>108674.000</td>
<td>1</td>
</tr>
<tr>
<td>CA</td>
<td>14576231.000</td>
<td>10001.000</td>
<td>109854.000</td>
<td>1</td>
</tr>
<tr>
<td>US</td>
<td>15718433.000</td>
<td>10348.000</td>
<td>109118.000</td>
<td>2</td>
</tr>
<tr>
<td>UK</td>
<td>15900180.000</td>
<td>10120.000</td>
<td>109702.000</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 12: The result set for the query given above. The United Kingdom will get a "high" bonus, the U.S. a "low" bonus.

5.2 Windowing Query

Consider the query above. To distribute bonuses in a fair fashion, we removed the best and worst profit sale. The management decides that that's not enough and want's me
to remove the best and worst 15% of the sales.

Listing 2: Same as above, but outliers are removed more reliably.

```sql
CREATE TEMPORARY VIEW sale_centile AS
SELECT customer.country,
    sale.profit,
    NTILE(100) OVER {
        PARTITION BY customer.country ORDER BY sale.profit
    } AS centile
FROM sale, customer
WHERE sale.customer_id = customer.id
AND customer.sales_district = 'ANGLO';

SELECT customer.country,
    SUM(sale.profit),
    MIN(sale.profit),
    MAX(sale.profit),
    NTILE(3) OVER (ORDER BY {
        SUM(sale.profit) -
        (SELECT SUM(profit) FROM sale_centile WHERE centile < 15 OR centile > 85)
    })
FROM sale, customer
WHERE sale.customer_id = customer.id
AND customer.sales_district = 'ANGLO'
GROUP BY customer.country;
```

<table>
<thead>
<tr>
<th>country</th>
<th>total profit</th>
<th>minimum profit</th>
<th>maximum profit</th>
<th>bonus category</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU</td>
<td>13578425.000</td>
<td>10700.000</td>
<td>108674.000</td>
<td>1</td>
</tr>
<tr>
<td>CA</td>
<td>14576231.000</td>
<td>10001.000</td>
<td>109854.000</td>
<td>1</td>
</tr>
<tr>
<td>US</td>
<td>15718433.000</td>
<td>10348.000</td>
<td>109118.000</td>
<td>2</td>
</tr>
<tr>
<td>UK</td>
<td>15900180.000</td>
<td>10120.000</td>
<td>109702.000</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 13: The result set for the query given above. Nothing changed, because there seem to have been no outliers.

### 5.3 Period-to-Period Query

The MARMOMACC is the biggest international fair for operators in the marble sector. Ever since 2009 the Covelano Marmi Srl. is represented with a stand. It is by far the most important advertising event of the year. The management would like to know the impact of this event on the number of sales. Are sales notably increasing? Is it worth
putting much money and effort into this event? We need to write a query that com-
pares the number of sales made within 100 days before the event with the sales made
100 events after it.

Since each year MARMOMACC takes place at slightly different days we first create
a temporary table of the begin and end dates of the fair for the last couple of years.

<table>
<thead>
<tr>
<th>Year</th>
<th>Begin</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009</td>
<td>30-09-2009</td>
<td>03-10-2009</td>
</tr>
<tr>
<td>2010</td>
<td>29-09-2010</td>
<td>02-10-2010</td>
</tr>
<tr>
<td>2011</td>
<td>21-09-2011</td>
<td>24-09-2011</td>
</tr>
<tr>
<td>2012</td>
<td>26-09-2012</td>
<td>29-09-2012</td>
</tr>
<tr>
<td>2013</td>
<td>25-09-2013</td>
<td>28-09-2013</td>
</tr>
</tbody>
</table>

Listing 3: For each

```
CREATE TEMPORARY VIEW before AS
SELECT date.year as year, COUNT(*) AS count
FROM date, sale
WHERE sale.date_id = date.id
AND date.date BETWEEN {
    SELECT begin_date
    FROM marmomacc_dates
    WHERE year = date.year
} - 100
AND
{
    SELECT begin_date
    FROM marmomacc_dates
    WHERE year = date.year
}
GROUP BY year;
```

```
CREATE TEMPORARY VIEW after AS
SELECT date.year as year, COUNT(*) AS count
FROM date, sale
WHERE sale.date_id = date.id
AND date.date BETWEEN {
    SELECT begin_date
    FROM marmomacc_dates
    WHERE year = date.year
}
```
SELECT DISTINCT date.year, before.count, after.count
FROM sale, date, before, after
WHERE sale.date_id = date.id
AND after.year = date.year
AND before.year = date.year;

<table>
<thead>
<tr>
<th>Year</th>
<th>Sales before</th>
<th>Sales afterwards</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009</td>
<td>53</td>
<td>56</td>
</tr>
<tr>
<td>2010</td>
<td>53</td>
<td>36</td>
</tr>
<tr>
<td>2011</td>
<td>43</td>
<td>55</td>
</tr>
<tr>
<td>2012</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td>2013</td>
<td>50</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 14: The result set for the query given above. Our sample data does not reflect reality.

5.4 Materialized Views

We choose three queries that might be executed frequently. For the sake of simplicity, we reduce one-to-many to one-to-one relations, more specifically we assume that the extraction fact has a one-to-one sort dimension and an additional vein angle degenerate dimension that is stored in the fact table as a measure. Also, additional dimensions have been removed or simplified, see Figure 7.
1. What is the amount of stone excavated per gallery and year, semester and quartal. The query is similar to a previous one. This query takes ca. 14ms to execute with 10'000 tuples in the fact table. The grouping is (year).

```sql
SELECT date.year,
       SUM(weight) AS amount
FROM extraction, date
WHERE extraction.excavation_date_id = date.id
GROUP BY date.year
ORDER BY date.year;
```

2. What is the amount of stone excavated per sort and quartal (of a specific year)? Executing this query is surprisingly fast; about 5 milliseconds for 10'000 tuples. The only grouping is (quartal, sortname).

```sql
SELECT date.quartal,
       sort.name AS sort,
       SUM(extraction.weight) AS amount
FROM extraction, date, sort
WHERE extraction.excavation_date_id = date.id
AND extraction.sort_id = sort.id
AND date.year = 2009
GROUP BY date.quartal, sort.name
ORDER BY date.quartal, sort.name;
```

3. It important for the company to know the angles of the vein patterns because two different blocks can only be sold if angles more or less match, so when tiles are cut from the blocks there is a uniform pattern. Thus, the following query might be of interest: What is the amount of stone excavated per semester (of a specific
year) grouped by vein angle? The vein angle is rounded to the nearest multiple of 10. Query execution time was around 5 milliseconds. The only grouping is \((\text{semester}, \text{veinangle})\).

```sql
SELECT date.semester,
       ceil(vein_angle / 10) * 10 AS angle,
       SUM(extraction.weight) AS amount
FROM extraction, date
WHERE extraction.excavation_date_id = date.id
AND date.year = 2009
GROUP BY date.semester, angle
ORDER BY date.semester, angle;
```

![Lattice diagram](image)

Figure 8: The lattice representing dependencies between sets of attributes that are suitable for materialization to improve the execution time of the queries above. Note that we have \(\text{quartal} > \text{semester}\).

**Getting Physical**  Fortunately, Postgres does support materialized views since version 9.3. The following piece of SQL creates the view and executing it takes over 200 milliseconds. The following view can be used by all three queries.

```sql
CREATE MATERIALIZED VIEW mv1 AS
SELECT date.year,
       date.quartal,
       sort.name,
       extraction.vein_angle,
       SUM(extraction.weight) AS amount
FROM extraction, date, sort
WHERE extraction.excavation_date_id = date.id
AND extraction.sort_id = sort.id
GROUP BY date.year, date.quartal, sort.name, extraction.vein_angle
ORDER BY date.year, date.quartal, sort.name;
```

We can rewrite the previous queries to make use of the materialized view. Unfortunately, Postgres does not support automatic query rewriting, so we do it by hand.

1. ```sql
   SELECT year,
          SUM(amount)
   FROM mv1
   GROUP BY year
   ORDER BY year;
```
2. SELECT quartal, 
   name as sort, 
   SUM(amount) 
FROM mv1 
WHERE year = 2009 
GROUP BY quartal, sort 
ORDER BY quartal, sort;

3. SELECT ceil(quartal / 2.0) AS semester, 
   ceil(vein_angle / 10) * 10 AS angle, 
   SUM(amount) 
FROM mv1 
WHERE year = 2009 
GROUP BY semester, angle 
ORDER BY semester, angle;

As Figure 8 shows, we might be able to reduce the query execution time further by creating additional materialized views. We will now create all materialized views and later discuss whether it was worth doing so. The following SQL code creates all different views.

CREATE MATERIALIZED VIEW mv2 AS 
SELECT date.year, 
   SUM(extraction.weight) AS amount 
FROM extraction, date 
WHERE extraction.excavation_date_id = date.id 
GROUP BY date.year 
ORDER BY date.year;

CREATE MATERIALIZED VIEW mv3 AS 
SELECT date.year, 
   date.quartal, 
   sort.name, 
   SUM(extraction.weight) AS amount 
FROM extraction, date, sort 
WHERE extraction.excavation_date_id = date.id 
AND extraction.sort_id = sort.id 
GROUP BY date.year, date.quartal, sort.name 
ORDER BY date.year, date.quartal, sort.name;

CREATE MATERIALIZED VIEW mv4 AS 
SELECT date.year, 
   date.semester, 
   extraction.vein_angle, 
   SUM(extraction.weight) AS amount 
FROM extraction, date, sort 
WHERE extraction.excavation_date_id = date.id 
GROUP BY date.year, date.semester, extraction.vein_angle 
ORDER BY date.year, date.semester;
The following table shows the execution time for the various query depending on whether they did make use of the materialized views or not or which view they used.

<table>
<thead>
<tr>
<th>Query</th>
<th>No mat. view</th>
<th>Generic mat. view</th>
<th>Specific mat. view</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>13</td>
<td>5</td>
<td>0.4</td>
</tr>
<tr>
<td>Query 2</td>
<td>5</td>
<td>2</td>
<td>0.7</td>
</tr>
<tr>
<td>Query 3</td>
<td>5</td>
<td>4</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Table 15: Comparison of the execution times (in ms.) for the example queries on different materialized views. The generic materialized view is the view that all queries can take advantage of, the specific materialized view is the view that only this specific query can make use of.

### Mat. View | Size
---|---
mv1 (Generic) | 7621
mv2 (for Query 1) | 7
mv3 (for Query 2) | 196
mv4 (for Query 3) | 2438

Table 16: Comparison of the materialized view sizes.

**Comparing result with theoretic expectations** Given the table sizes in Table 16 we can analyze the experimental data. It’s clear that the greedy algorithms presented in the lectures would choose mv2 first, then mv3, finally mv4. In practice, however, it’s worth considering creating mv4 instead of mv3, given that the improvement in execution time for the 3rd query is almost as good as for the 2nd query; this would depend on how often the respective queries are executed. Generally, execution times are fairly short, given the relatively small amount of data. In this specific scenario it might be perfectly fine to not create any materialized views at all.
References


