Bridging the Gap between OLAP and SQL

Jens-Peter Dittrich\textsuperscript{1,*} Donald Kossmann\textsuperscript{1,2} Alexander Kreutz\textsuperscript{2}

\textsuperscript{1}ETH Zurich, Switzerland

\textsuperscript{*}Affiliation 2003-2004: SAP BW OLAP Technology

\textsuperscript{2}i-TV-T AG, Germany
# OLAP: The CEO’s View

## Flexible Reporting

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tangible</td>
<td>Yes</td>
</tr>
<tr>
<td>Canada</td>
<td>No</td>
</tr>
<tr>
<td>Fiscal Year</td>
<td>2004</td>
</tr>
<tr>
<td>Fiscal Month</td>
<td>1</td>
</tr>
<tr>
<td>Last Fiscal Year</td>
<td>2003</td>
</tr>
<tr>
<td>Last Fiscal Month</td>
<td>12</td>
</tr>
</tbody>
</table>

## Report

<table>
<thead>
<tr>
<th>Team</th>
<th>Projects</th>
<th>Target</th>
<th>142-SURF FG9</th>
<th>146-SURF LGD</th>
<th>157-W/BR LGD</th>
<th>199-LID ALL</th>
<th>TOTAL</th>
<th>023-CEO APPE</th>
<th>162-BN OGDLE</th>
<th>167-ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advertising/Media</td>
<td>2</td>
<td>43,883.62</td>
<td>1,130.71</td>
<td>522.62</td>
<td>2,140.72</td>
<td>5,880.38</td>
<td>9,274.33</td>
<td>750.21</td>
<td>1,266.17</td>
<td></td>
</tr>
<tr>
<td>Body</td>
<td>2</td>
<td>64,417.81</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCD/Sales</td>
<td>1</td>
<td>15,036.66</td>
<td>203.03</td>
<td>112.72</td>
<td>390.88</td>
<td>964.82</td>
<td>1,661.48</td>
<td>1,176.81</td>
<td>402.27</td>
<td></td>
</tr>
<tr>
<td>CKI</td>
<td>2</td>
<td>40,986.92</td>
<td>12.24</td>
<td>7.34</td>
<td>22.03</td>
<td>53.85</td>
<td>95.47</td>
<td>1,336.30</td>
<td>660.54</td>
<td></td>
</tr>
<tr>
<td>Cleaning</td>
<td>1</td>
<td>43,350.21</td>
<td>116.03</td>
<td>95.20</td>
<td>226.64</td>
<td>554.35</td>
<td>964.31</td>
<td>930.62</td>
<td>335.25</td>
<td></td>
</tr>
<tr>
<td>Conditioners</td>
<td>1</td>
<td>92,044.68</td>
<td>88.71</td>
<td>49.41</td>
<td>170.07</td>
<td>417.58</td>
<td>726.24</td>
<td>2,597.96</td>
<td>904.75</td>
<td></td>
</tr>
<tr>
<td>Demo</td>
<td>4</td>
<td>40,123.69</td>
<td></td>
<td>12,050.54</td>
<td>7,165.28</td>
<td>24,015.84</td>
<td>80,025.12</td>
<td>105,033.54</td>
<td>24,130.72</td>
<td>9,230.64</td>
</tr>
<tr>
<td>Face</td>
<td>1</td>
<td>100.00</td>
<td></td>
<td>-125.93</td>
<td>-77.65</td>
<td>-289.14</td>
<td>-657.95</td>
<td>-1,144.27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finance</td>
<td>0</td>
<td>100.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONG</td>
<td>1</td>
<td>12,708.91</td>
<td>1,060.72</td>
<td>1,031.44</td>
<td>3,593.79</td>
<td>8,743.77</td>
<td>15,219.72</td>
<td>2,390.41</td>
<td>816.15</td>
<td></td>
</tr>
<tr>
<td>HR</td>
<td>1</td>
<td>19,675.23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistics</td>
<td>2</td>
<td>39,699.09</td>
<td>185.34</td>
<td>103.07</td>
<td>356.52</td>
<td>866.13</td>
<td>1,512.05</td>
<td>13,933.73</td>
<td>4,600.04</td>
<td></td>
</tr>
<tr>
<td>MET</td>
<td>2</td>
<td>59,771.21</td>
<td>99.56</td>
<td>54.56</td>
<td>199.37</td>
<td>458.97</td>
<td>802.48</td>
<td>3,066.59</td>
<td>1,299.09</td>
<td></td>
</tr>
<tr>
<td>NPI</td>
<td>1</td>
<td>62,341.55</td>
<td>908.57</td>
<td>544.30</td>
<td>1,366.75</td>
<td>4,661.32</td>
<td>8,013.63</td>
<td>3,074.34</td>
<td>1,323.70</td>
<td></td>
</tr>
<tr>
<td>PV/Bars</td>
<td>1</td>
<td>117,112.18</td>
<td>1,500.26</td>
<td>927.50</td>
<td>2,443.01</td>
<td>7,144.22</td>
<td>12,316.77</td>
<td>52,999.52</td>
<td>21,501.37</td>
<td></td>
</tr>
</tbody>
</table>

Download EXCEL file  Create Diagram  Create bookmark
OLAP: The Ph.D.’s View

1. OLAP is just another application on top of existing DBMS.  
   - Not true!
2. DBMS performance for OLAP is great.  
   - Not true!
3. The relational model is well suited for OLAP.  
   - Not true!
4. SQL is a great language for doing OLAP.  
   - Not true!
What do OLAP Engines do?...

1. Result Formatting **Good!**

2. Query Processing
   1. Joins
   2. Aggregations
   3. Pivot and Cube Computation
   4. Caching

3. Misc
   1. Currency Conversions
   2. Summarizability Checks
   3. Authorization
   4. ...

...OLAP engines bridge the gap between OLAP and SQL!
The Gap

- Relational Model
- Pivot Tables

Bridging the Gap

Closing the Gap
Input

<table>
<thead>
<tr>
<th>State</th>
<th>Customer</th>
<th>Product</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C1</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>P2</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Relational Model

```
SELECT State, Customer, Product, sum(Profit) 
FROM Profits 
GROUP BY ROLLUP (State, Customer, Product) 
ORDER BY State, Customer, Product;
```
Relational Model: NULL-values

<table>
<thead>
<tr>
<th>State</th>
<th>Customer</th>
<th>Product</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C1</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S1</td>
<td>NULL</td>
<td>P2</td>
<td>2.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>NULL</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S1</td>
<td>NULL</td>
<td>NULL</td>
<td>4.0</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>8.0</td>
</tr>
</tbody>
</table>

Take care

NULL has two different meanings in SQL:

1. Aggregate (from Rollup operation)
2. Value does not exist (e.g. from outer joins)

The semantics of a NULL-value can be obtained calling GROUPING().

Result of SQL Rollup

1. Differentiate between different semantics of NULL
Relational Model: order on rows

<table>
<thead>
<tr>
<th>State</th>
<th>Customer</th>
<th>Product</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C1</td>
<td>NULL</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C1</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>NULL</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S1</td>
<td>NULL</td>
<td>NULL</td>
<td>4.0</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>8.0</td>
</tr>
</tbody>
</table>

The result table is ordered lexicographically.

Result of SQL Rollup

1. differentiate between different semantics of NULL
2. assume order on result table
Relational Model: multi columns

Result of SQL Rollup

<table>
<thead>
<tr>
<th>State</th>
<th>Customer</th>
<th>Product</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C1</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C1</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S1</td>
<td>NULL</td>
<td>NULL</td>
<td>4.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>NULL</td>
<td>2.0</td>
</tr>
<tr>
<td>S1</td>
<td>NULL</td>
<td>NULL</td>
<td>4.0</td>
</tr>
</tbody>
</table>

Interpreting NULL-values as multi columns

1. differentiate between different semantics of NULL
2. assume order on result table
3. merge columns containing NULLs to form multi column cells
Relational Model: multi rows

1. differentiate between different semantics of NULL
2. assume order on result table
3. merge columns containing NULLs to form multi column cells
4. merge adjacent rows containing similar values to form multi row cells
Relational Model: order on columns

1. differentiate between different semantics of NULL
2. assume order on result table
3. merge columns containing NULLs to form multi column cells
4. merge adjacent rows containing similar values to form multi row cells
5. interpret order on columns as hierarchy
Relational Model: Pivot Tables

- Pivot operation moves at least one of the attributes to the columns.
- Some of the sums of the pivot are not part of the rollup, e.g. \((\Sigma \Sigma , P1)\), \((\Sigma \Sigma , P2)\).
- We have to use CUBE() here.
- Should we still call this a “table”?

### Pivot Table Example

<table>
<thead>
<tr>
<th>State</th>
<th>Customer</th>
<th>Product</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(\Sigma)</td>
<td>2.0</td>
</tr>
<tr>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\Sigma)</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\Sigma\Sigma)</td>
<td>4.0</td>
<td></td>
</tr>
</tbody>
</table>

Pivot

<table>
<thead>
<tr>
<th>State</th>
<th>Customer</th>
<th>P1</th>
<th>P2</th>
<th>(\Sigma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>C1</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>S1</td>
<td>(\Sigma)</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>S2</td>
<td>(\Sigma)</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
</tr>
<tr>
<td>(\Sigma\Sigma)</td>
<td>4.0</td>
<td>4.0</td>
<td>8.0</td>
<td></td>
</tr>
</tbody>
</table>

Attribute values become attribute names.
Relational Model: Summary

<table>
<thead>
<tr>
<th>State</th>
<th>Customer</th>
<th>Product</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C1</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
<td>P2</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>P1</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
<td>P2</td>
<td>1.0</td>
</tr>
</tbody>
</table>

SQL to Pivot Recipe

1. differentiate between different semantics of NULL
2. assume order on result table
3. merge columns containing NULLs to form multi column cells
4. merge adjacent rows containing similar values to form multi row cells
5. interpret order on columns as hierarchy
6. let attribute values become attribute names
The Gap

Bridging the Gap

Closing the Gap
Example: BTell’s Operator Model

- BTell has two different types of operators:
  1S: returns one output stream
  [Graefe: Volcano]
  3S: returns three output streams:
  1. one for the x-axis
  2. one for the y-axis
  3. one for the xy-axis
More Examples

1. Caching (Special Caching Operator)
2. Pivot Computation (based on 3S operator model)
3. Check for Computability of Aggregates

see paper
The Gap

Bridging the Gap

Closing the Gap
How to reach OLAP Heaven?

- 3 possible paths to follow:
  1. Add even more OLAP stuff to SQL
     Open questions:
     How to handle non-relational data? (nested relations?)
How to reach OLAP Heaven?

- 3 possible paths to follow:
  1. Add even more OLAP stuff to SQL
     Open questions:
     How to handle non-relational data? (nested relations?)
How to reach OLAP Heaven?

- 3 possible paths to follow:
  1. Add even more OLAP stuff to SQL
     Open questions:
     How to handle non-relational data? (nested relations?)
     - Improbable
  2. Make a new query language
     Hard to agree upon
     Works only for part of the market (see e.g. MDX)
How to reach OLAP Heaven?

- 3 possible paths to follow:
  1. Add even more OLAP stuff to SQL
     Open questions:
     How to handle non-relational data? (nested relations?)
  2. Make a new query language
     Hard to agree upon
     Works only for part of the market (see e.g. MDX)
How to reach OLAP Heaven?

- 3 possible paths to follow:
  1. Add even more OLAP stuff to SQL
     Open questions:
     How to handle non-relational data? (nested relations?)
  2. Make a new query language
     Hard to agree upon
     Works only for part of the market (see e.g. MDX)
  3. Ride the XML wave
     Recent proposal for ‘group by’ in XQuery from IBM [SIGMOD 2005]
How to reach OLAP Heaven?

- 3 possible paths to follow:
  1. Add even more OLAP stuff to SQL
     - Open questions:
       - How to handle non-relational data? (nested relations?)
  2. Make a new query language
     - Hard to agree upon
     - Works only for part of the market (see e.g. MDX)
  3. Ride the XML wave
     - Recent proposal for ‘group by’ in XQuery from IBM [SIGMOD 2005]
Why XML/XQuery?

1. Great data model
2. Powerful query language
3. Highly extensible
Example: Pivot Result

<table>
<thead>
<tr>
<th>Profits</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Customer</td>
</tr>
<tr>
<td>S1</td>
<td>C1</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
</tr>
<tr>
<td>S1</td>
<td>Σ</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
</tr>
<tr>
<td>S2</td>
<td>Σ</td>
</tr>
<tr>
<td>ΣΣ</td>
<td></td>
</tr>
</tbody>
</table>

<pre>
<profits>
 <rows>
  <S1>
   <C1> <1/> <2/> <3/> </C1>
   <C2> <4/> <5/> <6/> </C2>
   <sum> <7/> <8/> <9/> </sum>
  </S1>
  <S2>
   <C1> <10/> <11/> <12/> </C1>
   <C2> <13/> <14/> <15/> </C2>
   <sum> <16/> <17/> <18/> </sum>
  </S2>
  <sum> <19/> <20/> <21/> </sum>
 </rows>
 <columns>
  <P1> <1/> <4/> <7/> <10/> <13/> <16/> <19/> </P1>
  <P2> <2/> <5/> <8/> <11/> <14/> <17/> <20/> </P2>
  <sum> <3/> <6/> <9/> <12/> <15/> <18/> <21/> </sum>
 </columns>
 <data>
  <1> 1.0 </1>
  <2> 1.0 </2>
  <3> 2.0 </3>
  <4> 1.0 </4>
  ...
  <21> 8.0 </21>
 </data>
</profits>
</pre>
XQuery+OLAP Extensions

- **Goal**
  Get 95% of the OLAP query processing pushed into the DBMS (today: 10-20%)

- **Consequences**
  - Only minimal data transfer between DBMS and client
  - Overhead introduced by XML is negligible
  - Coding an OLAP engine becomes much easier
XQuery/OLAP: ON ROWS, ON COLUMNS

- Specify **where** to place dimensions:

```xquery
for $f in //profits
group by $f/state, $f/customer ON ROWS,
$f/product ON COLUMNS
return ... 
```
XQuery/OLAP: ROLLUP

- Provide Statements for Cube, Rollup, Pivot:

```xml
for $f$ in //profits
  group by ROLLUP ($f/state, $f/customer) on rows,
  ROLLUP ($f/product) on columns
return ...`

XQuery/OLAP: MDVIEW (1/2)

- Provide Statements for **multidimensional results**: 

  ```
  for $f$ in //profits
  group by rollup ($f/state, $f/customer) on rows,
  rollup($f/product) on columns
  return AS MDVIEW
  ```
for $f$ in //profits

group by rollup ($f$/state, $f$/customer) on rows,
rollup($f$/product) on columns
return AS MDVIEW

<table>
<thead>
<tr>
<th>Profits</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Customer</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>S1</td>
<td>C1</td>
</tr>
<tr>
<td>S1</td>
<td>C2</td>
</tr>
<tr>
<td>S1</td>
<td>Σ</td>
</tr>
<tr>
<td>S2</td>
<td>C1</td>
</tr>
<tr>
<td>S2</td>
<td>C2</td>
</tr>
<tr>
<td>S2</td>
<td>Σ</td>
</tr>
<tr>
<td>ΣΣ</td>
<td></td>
</tr>
</tbody>
</table>
XQuery/OLAP: SESSIONS, DEFINE&REDEFINE

- Enable declaration of XQuery session:

```xml
DEFINE SESSION $s AS
  for $f in //profits
  group by $f/state on rows,
  $f/product on columns
  return as mdview

$ret = EVAL($s)

REDEFINE SESSION $s
  INSERT $f/customer$ AFTER $f/state on rows

$ret = eval($s)
```
XQuery/OLAP: SESSIONS, NOTIFY (1/2)

- **Subscribe** to changes (Observer-pattern):

```xml
define session $s as
    for $f in //profits
    group by $f/state on rows,
    $f/product on columns
    return as mdview

define function notify(
    $res as $s/result,
    $metadata as $s/metadata
)
ON $s CHANGED
{
    (: code to handle query result $res :)
}
```
XQuery/OLAP: SESSIONS, NOTIFY (2/2)

- Could be used to implement **push**-based OLAP:

```plaintext
define session $s$ as
    for $f$ in //profits
    group by $f/state$ on rows,
    $f/product$ on columns
    return as mdview

define function notify(
    $res$ as $s/result$,
    $metadata$ as $s/metadata$
)
ON $s$ CHANGED
{
    call redraw_result_screen($res, $metadata)
}
```
Conclusions

- OLAP engines replicate DBMS functionality
- Reason: SQL/relational model is not powerful enough for OLAP

Observations
- XML/XQuery works
- But: XQuery needs some extensions

Future Work

- Create XQuery/OLAP language proposal
- Build prototype query engine that implements our proposal
Thanks for your attention!

Questions?

Contact Author
jens.dittrich @ inf.ethz.ch
www.inf.ethz.ch/~jensdi

More about my research on Thursday, Software Demo Group 6
Backup Slides
The Gap

- Relational Model
- Pivot Tables

Bridging the Gap

Closing the Gap
Navigation (1/2)

- Typical OLAP session is as follows:
  1. User selects initial query
  2. User navigates through the data by doing either a
     - roll-up
     - drill-down
     - slice
     - etc.

- However, every time the user navigates, i.e. the query gets altered, the DBMS receives the entire query definition.
- DBMS has no notion of navigation.
Navigation (2/2)

- Navigation is best explained by transition between states
- How come query languages do not support this?
- Could easily be exploited to optimize caching on all tiers!
Caching (1/2)

- From a 10,000 feet perspective all tiers do the same
  1. Receive and store some input data
  2. Perform algebraic query processing and optimization on the data
  3. Store some output data, send some of it to the next tier
Caching (2/2)

- All caches outside the DBMS have to be kept in sync manually.
- All caching outside DBMS has to be hand-coded.
- This is cumbersome and error-prone.

Too bad: