Bridging the Fjord between OLAP and SQL

Jens-Peter Dittrich[1,*] Donald Kossmann[1,2] Alexander Kreutz[2]

[1]ETH Zurich, Switzerland

*Affiliation 2003-2004: SAP BW OLAP Technology

[2]i-TV-T AG, Germany

OLAP: The CEO's View

BTell

Flexible Reporting

tangible: ✓
canada: ✗
first year: **2004**
first month: **1**
last year: **2004**
last month: **12**

Page: 1 2

navigator | update

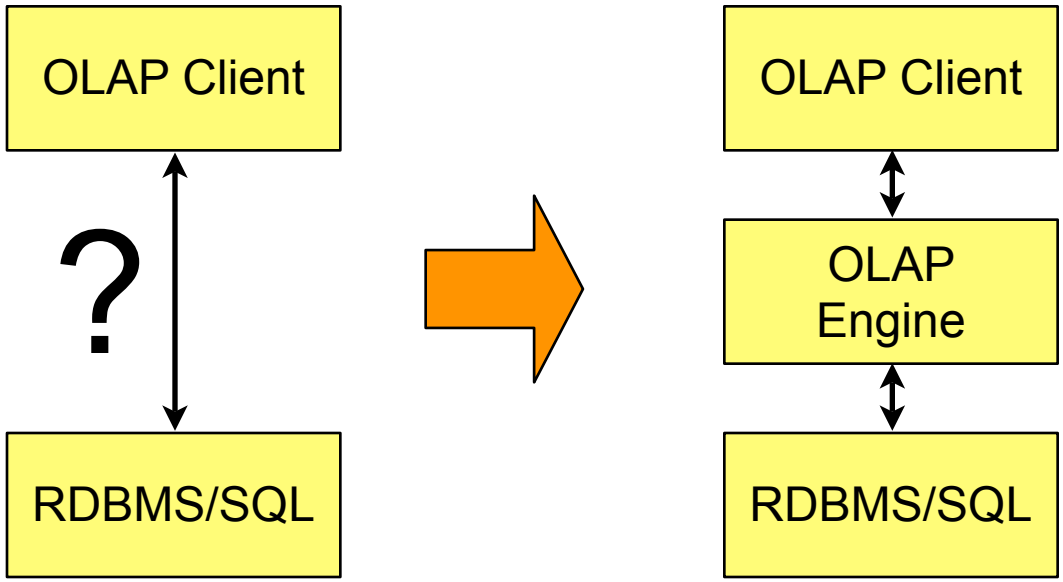
Report

			subcategory	Fabric Cleaning				Fabric Cleaning	Fabric Conditioning		
			brand	142-SURF PWD	148-SURF LIQ	187-WISK LIQ	189-LIQ ALL	TOTAL	023-DEG AP/DE	162-SNUGGLE	167-ALL
team	#projects	target		full sav.	full sav.	full sav.	full sav.	full sav.	full sav.	full sav.	
Advertising/Media	2	43,883.52		1,130.71	622.52	2,140.72	5,380.38	9,274.33			
Body	2	64,417.81							750.21	256.67	
CCD/Sales	1	15,039.66		203.03	112.72	390.89	954.82	1,661.46	1,176.81	402.27	
CMI	2	40,088.92		12.24	7.34	22.03	53.85	95.47	1,936.30	660.94	
Cleaning	1	43,358.21		118.03	65.29	226.64	554.35	964.31	980.63	335.25	
Conditioners	1	32,844.86		88.71	49.41	170.87	417.35	726.34	2,587.96	884.75	
Deo	4	40,123.69		12,950.54	7,165.26	24,892.64	60,825.10	105,833.54	24,130.72	8,238.64	
Face	1	100.00		-139.53	-77.65	-269.14	-657.95	-1,144.27			
Finance	0	100.00									
GIO	1	50,705.91		1,860.72	1,031.44	3,583.79	8,743.77	15,219.72	2,388.41	816.15	
Hair	1	15,675.23									
Logistics	2	39,497.00		185.34	103.07	355.52	868.13	1,512.05	13,693.73	4,680.04	
MET	2	59,777.18		99.50	54.56	189.37	458.97	802.40	3,806.59	1,299.89	
NPI	1	62,341.65		980.57	544.39	1,886.75	4,601.92	8,013.63	3,874.34	1,323.70	
PW Bars	1	117,113.19		1,500.25	827.59	2,843.61	7,144.32	12,315.77	62,899.52	21,501.37	

Download EXCEL file Create Diagramm Create bookmark

OLAP: The Ph.D.'s View

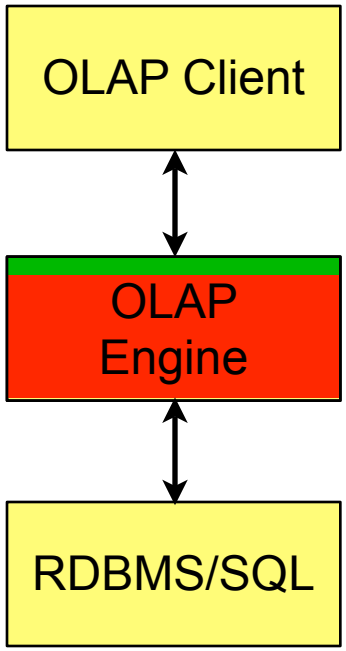
1. OLAP is just another application on top of existing DBMS. Not true!
2. DBMS performance for OLAP is great. Not true!
3. The relational model is well suited for OLAP. Not true!
4. SQL is a great language for doing OLAP. Not true!



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. ...

Bad!



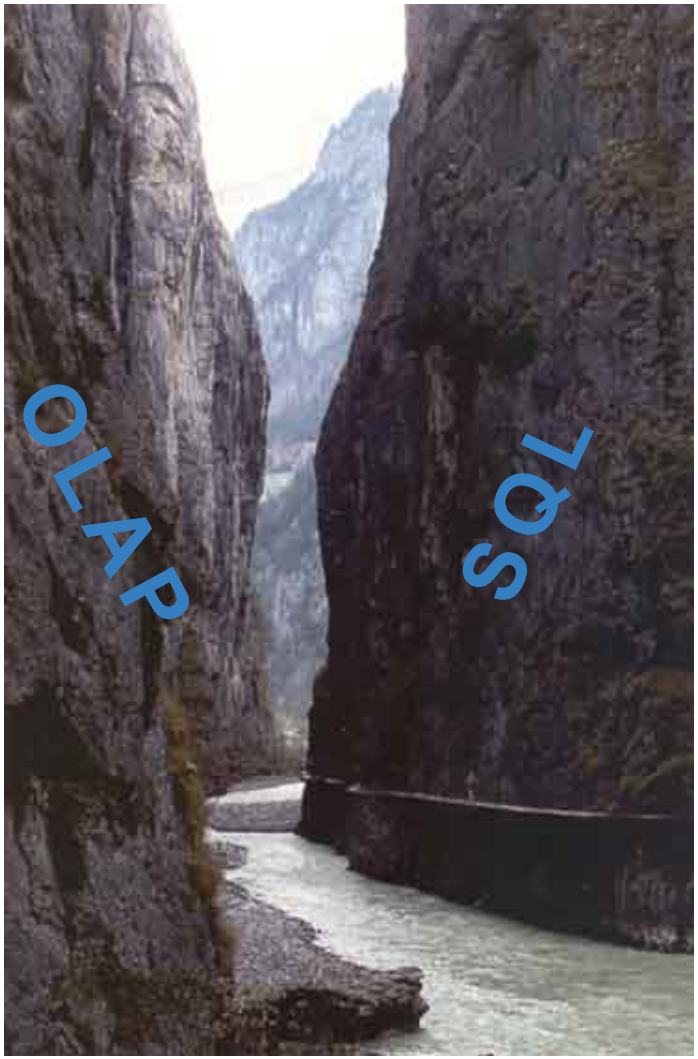
...OLAP engines bridge the gap between OLAP and SQL!

The Gap

- Relational Model
- Pivot Tables

Bridging the Gap

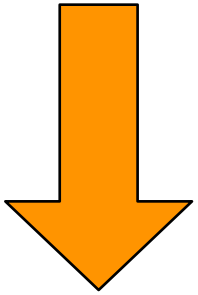
Closing the Gap



Relational Model

Input

State	Customer	Product	Profit
S1	C1	P1	1.0
S1	C1	P2	1.0
S1	C2	P1	1.0
S1	C2	P2	1.0
S2	C1	P1	1.0
S2	C1	P2	1.0
S2	C2	P1	1.0
S2	C2	P2	1.0



```
SELECT State, Customer, Product, sum(Profit)
FROM Profits
GROUP BY ROLLUP (State, Customer, Product)
ORDER BY State, Customer, Product;
```

Result of SQL Rollup

Profits

State	Customer	Product	Profit
S1	C1	P1	1.0
S1	C1	P2	1.0
S1	C1	NULL	2.0
S1	C2	P1	1.0
S1	C2	P2	1.0
S1	C2	NULL	2.0
S1	NULL	NULL	4.0
S2	C1	P1	1.0
S2	C1	P2	1.0
S2	C1	NULL	2.0
S2	C2	P1	1.0
S2	C2	P2	1.0
S2	C2	NULL	2.0
S1	NULL	NULL	4.0
NULL	NULL	NULL	8.0

Relational Model: NULL-values

Profits			
State	Customer	Product	Profit
S1	C1	P1	1.0
S1	C1	P2	1.0
S1	C1	NULL	2.0
S1	C2	P1	1.0
S1	C2	P2	1.0
S1	C2	NULL	2.0
S1	NULL	NULL	4.0
S2	C1	P1	1.0
S2	C1	P2	1.0
S2	C1	NULL	2.0
S2	C2	P1	1.0
S2	C2	P2	1.0
S2	C2	NULL	2.0
S1	NULL	NULL	4.0
NULL	NULL	NULL	8.0

Result of
SQL Rollup

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().

1. differentiate between different semantics of NULL

Relational Model: order on rows

Profits			
State	Customer	Product	Profit
S1	C1	P1	1.0
S1	C1	P2	1.0
S1	C1	NULL	2.0
S1	C2	P1	1.0
S1	C2	P2	1.0
S1	C2	NULL	2.0
S1	NULL	NULL	4.0
S2	C1	P1	1.0
S2	C1	P2	1.0
S2	C1	NULL	2.0
S2	C2	P1	1.0
S2	C2	P2	1.0
S2	C2	NULL	2.0
S1	NULL	NULL	4.0
NULL	NULL	NULL	8.0

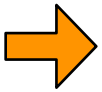
Result of
SQL Rollup

The result table is ordered lexicographically.

1. differentiate between different semantics of NULL
2. assume order on result table

Relational Model: multi columns

Profits			
State	Customer	Product	Profit
S1	C1	P1	1.0
S1	C1	P2	1.0
S1	C1	NULL	2.0
S1	C2	P1	1.0
S1	C2	P2	1.0
S1	C2	NULL	2.0
S1	NULL	NULL	4.0
S2	C1	P1	1.0
S2	C1	P2	1.0
S2	C1	NULL	2.0
S2	C2	P1	1.0
S2	C2	P2	1.0
S2	C2	NULL	2.0
S1	NULL	NULL	4.0
NULL	NULL	NULL	8.0



Profits			
State	Customer	Product	Profit
S1	C1	P1	1.0
S1	C1	P2	1.0
S1	C1	Σ	2.0
S1	C2	P1	1.0
S1	C2	P2	1.0
S1	C2	Σ	2.0
S1	ΣΣ		4.0
S2	C1	P1	1.0
S2	C1	P2	1.0
S2	C1	Σ	2.0
S2	C2	P1	1.0
S2	C2	P2	1.0
S2	C2	Σ	2.0
S2	ΣΣ		4.0
ΣΣΣ			8.0

Result of SQL Rollup

Interpreting NULL-values as multi columns

1. differentiate between different semantics of NULL
2. assume order on result table
3. merge columns containg NULLs to form multi column cells

Relational Model: multi rows

Profits

State	Customer	Product	Profit
S1	C1	P1	1.0
S1	C1	P2	1.0
S1	C1	NULL	2.0
S1	C2	P1	1.0
S1	C2	P2	1.0
S1	C2	NULL	2.0
S1	NULL	NULL	4.0
S2	C1	P1	1.0
S2	C1	P2	1.0
S2	C1	NULL	2.0
S2	C2	P1	1.0
S2	C2	P2	1.0
S2	C2	NULL	2.0
S1	NULL	NULL	4.0
NULL	NULL	NULL	8.0

Result of SQL Rollup

Profits

State	Customer	Product	Profit
S1	C1	P1	1.0
S1	C1	P2	1.0
S1	C1	Σ	2.0
S1	C2	P1	1.0
S1	C2	P2	1.0
S1	C2	Σ	2.0
S1	ΣΣ		4.0
S2	C1	P1	1.0
S2	C1	P2	1.0
S2	C1	Σ	2.0
S2	C2	P1	1.0
S2	C2	P2	1.0
S2	C2	Σ	2.0
S2	ΣΣ		4.0
ΣΣΣ			8.0

Interpreting NULL-values as multi columns

Profits

State	Customer	Product	Profit
S1	C1	P1	1.0
		P2	1.0
		Σ	2.0
	C2	P1	1.0
		P2	1.0
		Σ	2.0
	ΣΣ		4.0
S2	C1	P1	1.0
		P2	1.0
		Σ	2.0
	C2	P1	1.0
		P2	1.0
		Σ	2.0
	ΣΣ		4.0
ΣΣΣ			8.0

Interpreting adjacent similar values as multi rows

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

Relational Model: order on columns

These cells look different if we chose a different order on the dimensions, e.g. Customer, State, Product

Profits

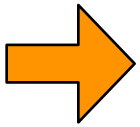
State	Customer	Product	Profit
S1	C1	P1	1.0
		P2	1.0
		Σ	2.0
	C2	P1	1.0
		P2	1.0
		Σ	2.0
	ΣΣ		4.0
S2	C1	P1	1.0
		P2	1.0
		Σ	2.0
	C2	P1	1.0
		P2	1.0
		Σ	2.0
	ΣΣ		4.0
ΣΣΣ			8.0

1. differentiate between different semantics of NULL
2. assume order on result table
3. merge columns containg 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

Profits			
State	Customer	Product	Profit
S1	C1	P1	1.0
		P2	1.0
		Σ	2.0
	C2	P1	1.0
		P2	1.0
		Σ	2.0
	ΣΣ		4.0
S2	C1	P1	1.0
		P2	1.0
		Σ	2.0
	C2	P1	1.0
		P2	1.0
		Σ	2.0
	ΣΣ		4.0
ΣΣΣ			8.0

Pivot



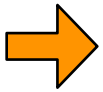
Profits		Product		
State	Customer	P1	P2	Σ
S1	C1	1.0	1.0	2.0
S1	C2	1.0	1.0	2.0
S1	Σ	2.0	2.0	4.0
S2	C1	1.0	1.0	2.0
S2	C2	1.0	1.0	2.0
S2	Σ	2.0	2.0	4.0
ΣΣ		4.0	4.0	8.0

attribute values become attribute names

- 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. (Σ Σ,P1), (Σ Σ,P2).
- We have to use CUBE() here.
- Should we still call this a “table”?

Relational Model: Summary

State	Customer	Product	Profit
S1	C1	P1	1.0
S1	C1	P2	1.0
S1	C2	P1	1.0
S1	C2	P2	1.0
S2	C1	P1	1.0
S2	C1	P2	1.0
S2	C2	P1	1.0
S2	C2	P2	1.0



Profits		Product		
State	Customer	P1	P2	Σ
S1	C1	1.0	1.0	2.0
S1	C2	1.0	1.0	2.0
S1	Σ	2.0	2.0	4.0
S2	C1	1.0	1.0	2.0
S2	C2	1.0	1.0	2.0
S2	Σ	2.0	2.0	4.0
$\Sigma\Sigma$		4.0	4.0	8.0

SQL to Pivot Recipe

1. differentiate between different semantics of NULL
2. assume order on result table
3. merge columns containg 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

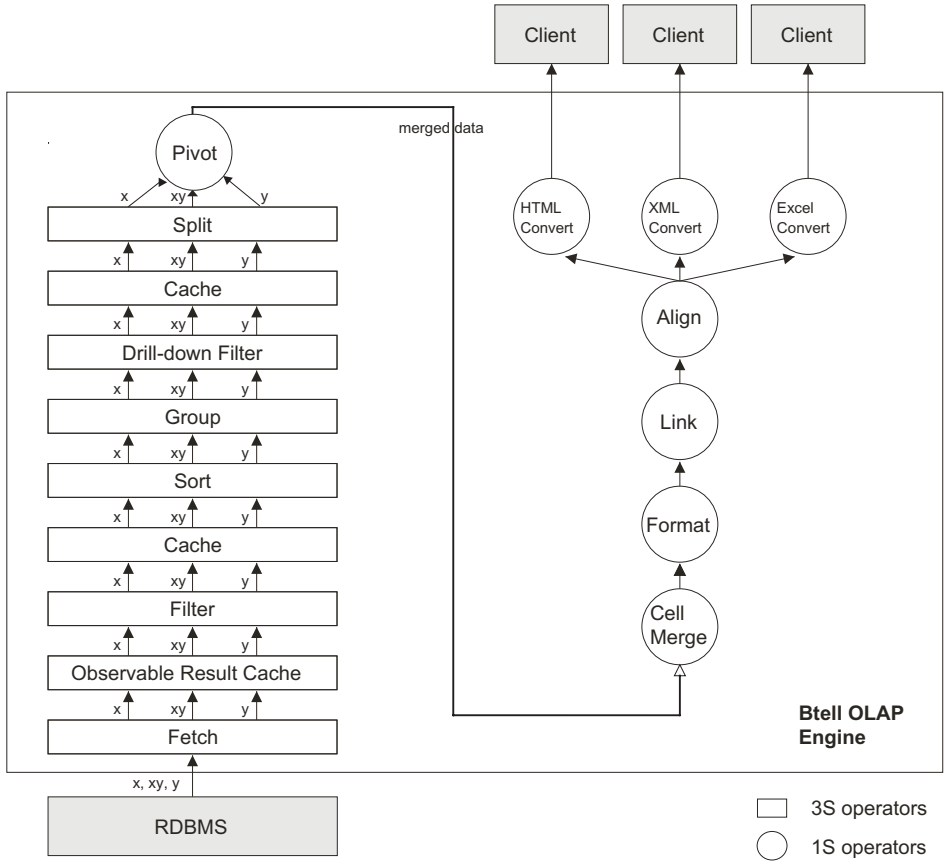
The Gap

Bridging the Gap

Closing the Gap

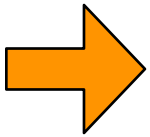
Example: BTell's Operator Model

- 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

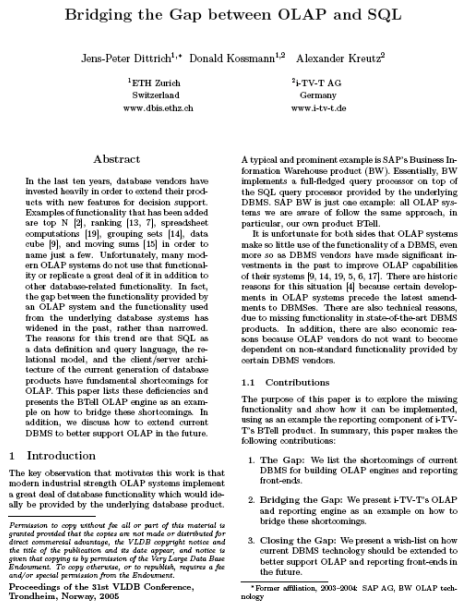


More Examples

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



see paper



Bridging the Gap between OLAP and SQL

Jens-Peter Dittrich[1,*] Donald Kossmann[1,2] Alexander Kreutz[2]

[1]ETH Zurich
Switzerland
www.dbis.ethz.ch

[2]i-TV-T AG
Germany
www.i-tv-t.de

Abstract

In the last ten years, database vendors have invested heavily in order to extend their products with new features for decision support. Examples of functionality that has been added are top N [2], ranking [13, 7], spreadsheet computations [19], grouping sets [14], data cube [9], and moving sums [15] in order to name just a few. Unfortunately, many modern OLAP systems do not use that functionality or replicate a great deal of it in addition to other database-related functionality. In fact, the gap between the functionality provided by an OLAP system and the functionality used from the underlying database systems has widened in the past, rather than narrowed. The reasons for this trend are that SQL as a data definition and query language, the relational model, and the client/server architecture of the current generation of database products have fundamental shortcomings for OLAP. This paper lists these deficiencies and presents the BTell OLAP engine as an example on how to bridge these shortcomings. In addition, we discuss how to extend current DBMS to better support OLAP in the future.

1 Introduction

The key observation that motivates this work is that modern industrial strength OLAP systems implement a great deal of database functionality which would ideally be provided by the underlying database product. A typical and prominent example is SAP's Business Information Warehouse product (BW). Essentially, BW implements a full-fledged query processor on top of the SQL query processor provided by the underlying DBMS. SAP BW is just one example: all OLAP systems we are aware of follow the same approach, in particular, our own product BTell.

It is unfortunate for both sides that OLAP systems make so little use of the functionality of a DBMS, even more so as DBMS vendors have made significant investments in the past to improve OLAP capabilities of their systems [9, 14, 19, 3, 6, 17]. There are historic reasons for this situation [4] because certain developments in OLAP systems precede the latest amendments to DBMSes. There are also technical reasons, due to missing functionality in state-of-the-art DBMS products. In addition, there are also economic reasons because OLAP vendors do not want to become dependent on non-standard functionality provided by certain DBMS vendors.

1.1 Contributions

The purpose of this paper is to explore the missing functionality and show how it can be implemented, using as an example the reporting component of i-TV-T's BTell product. In summary, this paper makes the following contributions:

1. The Gap: We list the shortcomings of current DBMS for building OLAP engines and reporting front-ends.
2. Bridging the Gap: We present i-TV-T's OLAP and reporting engine as an example on how to bridge these shortcomings.
3. Closing the Gap: We present a wish-list on how current DBMS technology should be extended to better support OLAP and reporting front-ends in the future.

Proceedings of the 31st VLDB Conference,
Trondheim, Norway, 2005

* Former affiliation, 2003–2004: SAP AG, BW OLAP technology

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?)

Improbable

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?)
     Improbable
  2. Make a new query language
     Hard to agree upon
     Works only for part of the market (see e.g. MDX)
     Improbable

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)
     Improbable
  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?)
     Improbable
  2. Make a new query language
     Hard to agree upon
     Works only for part of the market (see e.g. MDX)
     Improbable
  3. Ride the XML wave
     Recent proposal for 'group by' in XQuery from IBM [SIGMOD 2005]
     Probable!

Why XML/XQuery?

1. Great data model
2. Powerful query language
3. Highly extensible

Example: Pivot Result

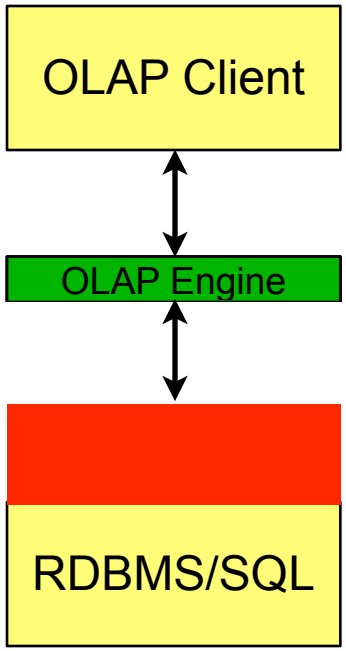
Profits		Product		
State	Customer	P1	P2	Σ
S1	C1	1.0	1.0	2.0
S1	C2	1.0	1.0	2.0
S1	Σ	2.0	2.0	4.0
S2	C1	1.0	1.0	2.0
S2	C2	1.0	1.0	2.0
S2	Σ	2.0	2.0	4.0
ΣΣ		4.0	4.0	8.0

```
<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>
```

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:

```
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**:

```
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
```

XQuery/OLAP: MDVIEW (2/2)

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

Profits		Product		
State	Customer	P1	P2	Σ
S1	C1	1.0	1.0	2.0
S1	C2	1.0	1.0	2.0
S1	Σ	2.0	2.0	4.0
S2	C1	1.0	1.0	2.0
S2	C2	1.0	1.0	2.0
S2	Σ	2.0	2.0	4.0
ΣΣ		4.0	4.0	8.0

```
<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>
```

XQuery/OLAP: SESSIONS, DEFINE&REDEFINE

- Enable declaration of XQuery **sessions**:

```
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):

```
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)

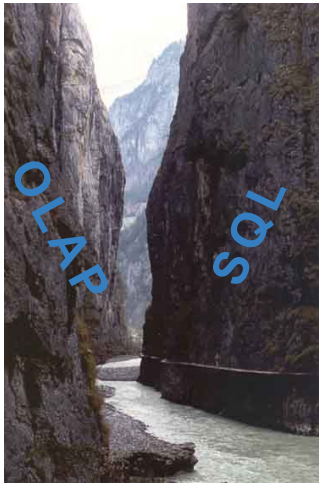
- Could be used to implement **push**-based OLAP:

```
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

iMeMex.org

Personal Information Management System

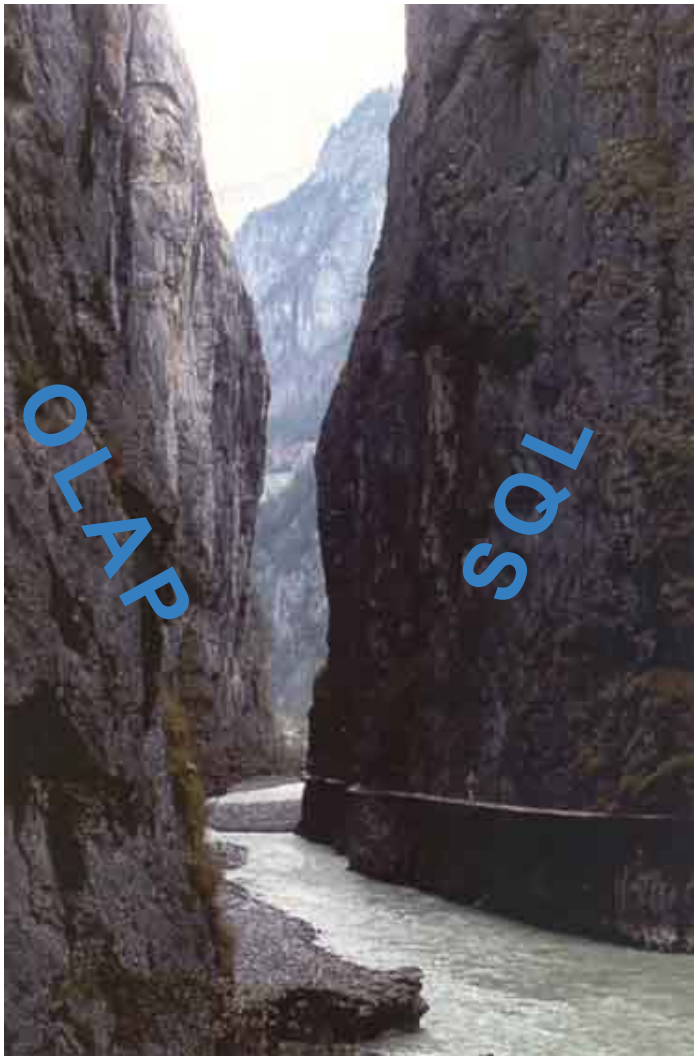
Backup Slides

The Gap

- Relational Model
- Pivot Tables

Bridging the Gap

Closing the Gap



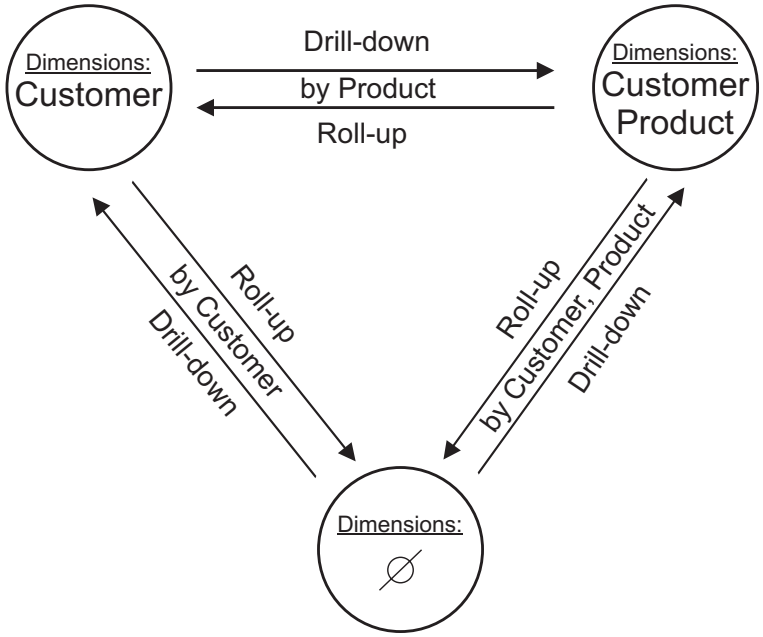
Navigation (1/2)

- 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.
3. Goto 2.

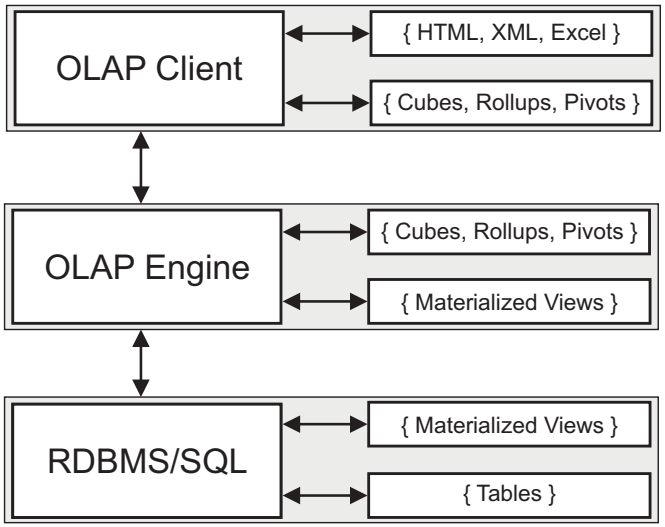
- However, every time the user navigates, i.e. the query gets altered, the DBMS receives the **entire** query definition.
- DBMS has no notion of navigation.

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!

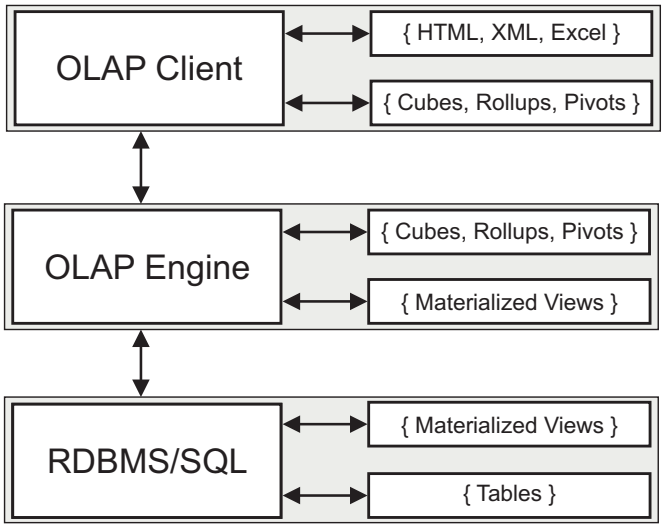
Caching (1/2)



10,000 feet perspective

- 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)



10,000 feet perspective

Too bad:

- 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.