

Data Warehousing (Advanced Query Processing)

Jens Dittrich
Donald Kossmann
<http://www.dbis.ethz.ch>

Selected References

- General
 - Chaudhuri, Dayal: An Overview of Data Warehousing and OLAP Technology. SIGMOD Record 1997
 - Lehner: Datenbanktechnologie für Data Warehouse Systeme. Dpunkt Verlag 2003
 - (...)
- New Operators and Algorithms
 - Agrawal, Srikant: Fast Algorithms for Association Rule Mining. VLDB 1994
 - Barateiro, Galhardas: A Survey of Data Quality Tools. Datenbank Spektrum 2005
 - Börszonyi, Kossmann, Stocker: Skyline Operator. ICDE 2001
 - Carey, Kossmann: On Saying Enough Already in SQL. SIGMOD 1997
 - Dalvi, Suciu: Efficient Query Evaluation on Probabilistic Databases. VLDB 2004
 - Gray et al.: Data Cube... ICDE 1996
 - Helmer: Evaluating different approaches for indexing fuzzy sets. Fuzzy Sets and Systems 2003
 - Olken: Database Sampling - A Survey. Technical Report LBL.
 - (...)

References (ctd.)

- Projects & Systems
 - Aurora, Borealis Systems (Brown, Brandeis, MIT)
 - Dittrich, Kossmann, Kreutz: Bridging the Gap between OLAP and SQL. VLDB 2005 (Btell Demo)
 - Garlic (IBM) - Haas et al. VLDB 1997
 - STREAM (Stanford)
 - Telegraph (Berkeley)
 - Trio (Stanford)
- SQL Extensions
 - Jensen et al. The Consensus Glossary of Temporal Database Concepts. Dagstuhl 1997 (also see TSQL)
 - Kimball, Strehlo: Why decision support fails and how to fix it. SIGMOD Record 1995
 - Witkowski et al.: Spreadsheets in RDBMS. SIGMOD 2003

History of Databases

- Age of Transactions (70s - 00s)
 - Goal: reliability - make sure no data is lost
 - 60s: IMS (hierarchical data model)
 - 80s: Oracle (relational data model)
- Age of Business Intelligence (95 -)
 - Goal: analyze the data -> make business decisions
 - Aggregate data for boss. Tolerate imprecision!
 - SAP BW, Microstrategy, Cognos, ... (rel. model)
- Age of „Data for the Masses“
 - Goal: everybody has access to everything, M2M
 - Google (text), GoogleBase (XML?)

Purpose of this Class

- Age of Transactions (70s - 00s)
 - Goal: reliability - make sure no data is lost
 - 60s: IMS (hierarchical data model)
 - 80s: Oracle (relational data model)
- **Age of Business Intelligence (95 -)**
 - **Goal: analyze the data -> make business decisions**
 - **Aggregate data for boss. Tolerate imprecision!**
 - **SAP BW, Microstrategy, Cognos, ... (rel. model)**
- Age of „Data for the Masses“
 - Goal: everybody has access to everything, M2M
 - Google (text), GoogleBase (XML?)

Overview

- Motivation and Architecture
- SQL extensions for Data Warehousing (DSS)
- Algorithms and Query Processing Techniques
- Data Mining
- Virtual Databases (Data Integration)
- Data Cleaning
- Time Series (SQL / T)
- Continuous Queries and Streams
- Probabilistic Databases

OLTP vs. OLAP

- OLTP – Online Transaction Processing
 - Many small transactions (point queries: UPDATE or INSERT)
 - Avoid redundancy, normalize schemas
 - Access to consistent, up-to-date database
- OLTP Examples:
 - Flight reservation (see IS-G)
 - Order Management, Procurement, ERP
- Goal: 6000 Transactions per second (Oracle 1995)

OLTP vs. OLAP

- OLAP – Online Analytical Processing
 - Big queries (all the data, joins); no Updates
 - Redundancy a necessity (Materialized Views, special-purpose indexes, de-normalized schemas)
 - Periodic refresh of data (daily or weekly)
- OLAP Examples
 - Management Information (sales per employee)
 - Statistisches Bundesamt (Volkzählung)
 - Scientific databases, Bio-Informatics
- Goal: Response Time of seconds / few minutes

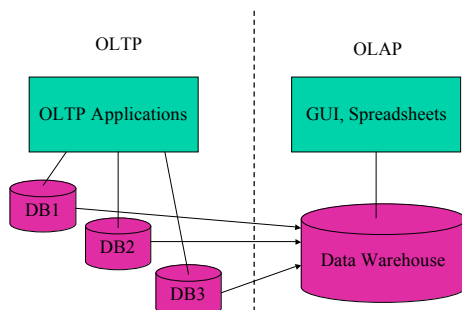
OLTP vs. OLAP (Water and Oil)

- Lock Conflicts: OLAP blocks OLTP
- Database design:
 - OLTP normalized, OLAP de-normalized
- Tuning, Optimization
 - OLTP: inter-query parallelism, heuristic optimization
 - OLAP: intra-query parallelism, full-fledged optimization
- Freshness of Data:
 - OLTP: serializability
 - OLAP: reproducibility
- Precision:
 - OLTP: ACID
 - OLAP: Sampling, Confidence Intervals

Solution: Data Warehouse

- Special Sandbox for OLAP
- Data input using OLTP systems
- Data Warehouse aggregates and replicates data (special schema)
- New Data is **periodically** uploaded to Warehouse
- Old Data is deleted from Warehouse
 - Archiving done by OLTP system for legal reasons

Architecture



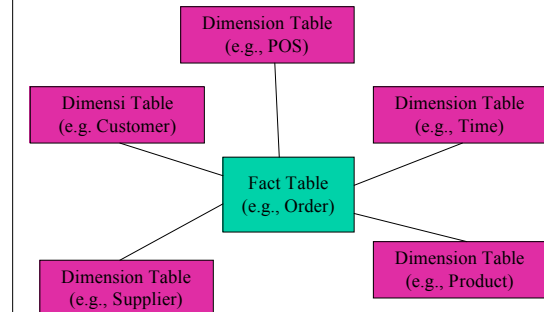
Data Warehouses in the real World

- First industrial project in 1995
- At beginning, 80% failure rate of projects
- Consultants like Accenture dominate market
- Why difficult: Data integration + cleaning, poor modeling of business processes in warehouse
- Data warehouses are expensive (typically as expensive as OLTP system)
- Success Story: WalMart - 20% cost reduction because of Data Warehouse (just in time...)

Products and Tools

- Oracle 10g, IBM DB2, Microsoft SQL Server, ...
 - All data base vendors
- SAP Business Information Warehouse
 - ERP vendors
- MicroStrategy, Cognos
 - Specialized vendors
 - „Web-based EXCEL“
- Niche Players (e.g., Btell)
 - Vertical application domain

Star Schema (relational)



Fact Table (Order)

No.	Cust.	Date	...	POS	Price	Vol.	TAX
001	Heinz	13.5.	...	Mainz	500	5	7.0
002	Ute	17.6.	...	Köln	500	1	14.0
003	Heinz	21.6.	...	Köln	700	1	7.0
004	Heinz	4.10.	...	Mainz	400	7	7.0
005	Karin	4.10.	...	Mainz	800	3	0.0
006	Thea	7.10.	...	Köln	300	2	14.0
007	Nobbi	13.11.	...	Köln	100	5	7.0
008	Sarah	20.12	...	Köln	200	4	7.0

Fact Table

- Structure:
 - key (e.g., Order Number)
 - Foreign key to all dimension tables
 - measures (e.g., Price, Volume, TAX, ...)
- Store *moving data* (*Bewegungsdaten*)
- Very large and normalized

Dimension Table (PoS)

Name	Manager	City	Region	Country	Tel.
Mainz	Helga	Mainz	South	D	1422
Köln	Vera	Hürth	South	D	3311

- De-normalized: City -> Region -> Country
 - Avoid joins
- fairly small and constant size
- Dimension tables store *master data* (*Stammdaten*)
- Attributes are called *Merkmale* in German

Snowflake Schema

- If dimension tables get too large
 - Partition the dimension table
- Trade-Off
 - Less redundancy (smaller tables)
 - Additional joins needed
- Exercise: Do the math!

Typical Queries

```
SELECT d1.x, d2.y, d3.z, sum(f.z1), avg(f.z2)
FROM Fact f, Dim1 d1, Dim2 d2, Dim3 d3
WHERE a < d1.feld < b AND d2.feld = c AND
      Join predicates
GROUP BY d1.x, d2.y, d3.z;
```

- Select by Attributes of Dimensions
 - E.g., region = „south“
- Group by Attributes of Dimensions
 - E.g., region, month, quarter
- Aggregate on measures
 - E.g., sum(price * volumen)

Example

```
SELECT f.region, z.month, sum(a.price * a.volume)
FROM Order a, Time z, PoS f
WHERE a.pos = f.name AND a.date = z.date
GROUP BY f.region, z.month
```

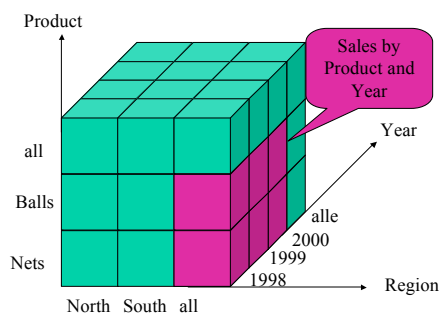
South	May	2500
North	June	1200
South	October	5200
North	October	600

Drill-Down und Roll-Up

- Add attribute to GROUP BY clause
 - More detailed results (e.g., more fine-grained results)
- Remove attribute from GROUP BY clause
 - More coarse-grained results (e.g., big picture)
- GUIs allow „Navigation“ through Results
 - Drill-Down: more detailed results
 - Roll-Up: less detailed results
- Typical operation, drill-down along hierarchy:
 - E.g., use „city“ instead of „region“

Btell Demo (Part 1)

Data Cube



Moving Sums, ROLLUP

- Example:
 - GROUP BY ROLLUP(country, region, city)
 - Give totals for all countries and regions
- This can be done by using the ROLLUP Operator
- Attention: The order of dimensions in the GROUP BY clause matters!!!
- Again: Spreadsheets (EXCEL) are good at this
- The result is a table! (Completeness of rel. model!)

ROLLUP alla IBM UDB

```
SELECT Country, Region, City, sum(price*vol)
FROM Orders a, PoS f
WHERE a.pos = f.name
GROUP BY ROLLUP(Country, Region, City)
ORDER BY Country, Region, City;
```

Also works for other aggregate functions; e.g., avg().

Result of ROLLUP Operator

D	North	Köln	1000
D	North	(null)	1000
D	South	Mainz	3000
D	South	München	200
D	South	(null)	3200
D	(null)	(null)	4200

Summarizability (Unit)

- Legal Query

```
SELECT product, customer, unit, sum(volume)
FROM Order
GROUP BY product, customer, unit;
```
- Legal Query (product -> unit)

```
SELECT product, customer, sum(volume)
FROM Order
GROUP BY product, customer;
```
- Illegal Query (add „kg“ to „m“!!!)

```
SELECT customer, sum(volume)
FROM Order
GROUP BY customer;
```

Summarizability (de-normalized data)

Region	Customer	Product	Volume	Populat.
South	Heinz	Balls	1000	3 Mio.
South	Heinz	Nets	500	3 Mio.
South	Mary	Balls	800	3 Mio.
South	Mary	Nets	700	3 Mio.
North	Heinz	Balls	1000	20 Mio.
North	Heinz	Nets	500	20 Mio.
North	Mary	Balls	800	20 Mio.
North	Mary	Nets	700	20 Mio.

Customer, Product -> Revenue
 Region -> Population

Summarizability (de-normalized data)

- What is the result of the following query?

```
SELECT region, customer, product, sum(volume)
FROM Order
GROUP BY ROLLUP(region, customer, product);
```
- All off-the-shelf databases get this wrong!
- Problem: Total Revenue is 3000 (not 6000!)
- BI Tools get it right: keep track of functional dependencies
- Problem arises if reports involves several unrelated measures.

Cube Operator

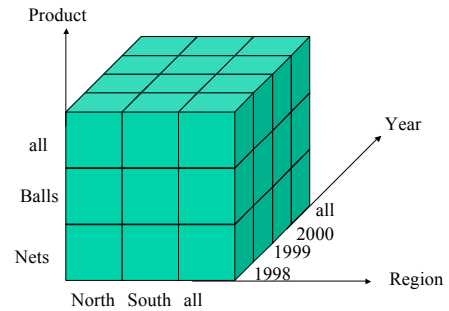
- Operator that computes all „combinations“
- Result contains „(null)“ Values to encode „all“

```
SELECT product, year, region, sum(price * vol)
FROM Orders
GROUP BY CUBE(product, year, region);
```

Result of Cube Operator

Product	Region	Year	Revenue
Netze	Nord	1998	...
Bälle	Nord	1998	...
(null)	Nord	1998	...
Netze	Süd	1998	...
Bälle	Süd	1998	...
(null)	Süd	1998	...
Netze	(null)	1998	...
Bälle	(null)	1998	...
(null)	(null)	1998	...

Visualization as Cube



Pivot Tables

- Define „columns“ by group by predicates
- Not a SQL standard! But common in products
- Reference: Cunningham, Graefe, Galindo-Legaria: PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS. VLDB 2004

UNPIVOT (material, factory)

matcode	matName	factory	vol. total	price level
APP	Apples	Berlin	32,000,000.00	0.44
		Dallas	36,800,000.00	0.28
		Duisburg	28,800,000.00	0.42
B 125	Bottle 125 ml	Hanoi	24,000,000.00	0.20
		Madison	40,000,000.00	0.28
		Hanoi	92,000.00	70.00
B 250	Bottle 250 ml	Hanoi	48,000.00	109.20
BAN	Bananas	Hanoi	36,000.00	87.60
BEE	Ground Beef	Hanoi	72,000,000.00	0.10
		Berlin	2,400,000.00	2.25
CAL	Calcium	Dallas	6,000,000.00	1.00
		Duisburg	800,000.00	2.25
BLA	Blackberries, dried	Madison	6,000.00	224.00
CAL	Calcium	Berlin	6,000.00	120.00
		Dallas	6,000.00	112.00

PIVOT (material, factory)

matcode	matName	factory	Berlin	Dallas	Duisburg	Hanoi	Madison
APP	Apples		32,000,000.00	36,800,000.00	28,800,000.00	24,000,000.00	40,000,000.00
B 125	Bottle 125 ml				92,000.00	70.00	
B 250	Bottle 250 ml				48,000.00	109.20	
BAN	Bananas				72,000,000.00	0.10	
BEE	Ground Beef		2,400,000.00	2.25	6,000,000.00	1.00	800,000.00
BLA	Blackberries, dried		6,000.00	120.00	6,000.00	120.00	6,000.00
CAL	Calcium		15,600,000.00	0.45	13,200,000.00	0.27	14,400,000.00
CAU	Cauliflower				60,000.00	10.00	
CRB 500	Cardboard Box 500g		60,000.00	10.00	1,200,000.00	4.00	
CHR	Ground Chicken				8,000.00	156.00	
CHO	Chocunone						6,000.00
CON	Cinnamon						6,000.00
COR	Corn		18,400,000.00	0.12	30,000,000.00	0.06	

Implementation

- ROLAP – Extend RDBMS
 - Special Star-Join Techniques
 - Bitmap Indexes
 - Partition Data by Time (Bulk Delete)
 - Materialized Views
- MOLAP – special multi-dimensional systems
 - Implement cube as (multi-dim.) array
 - Pro: potentially fast (random access in array)
 - Problem: array is very sparse
- Religious war (ROLAP wins in industry)

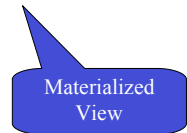
Materialized Views

- Compute the result of a query using the result of another query
- Principle: Subsumption
 - The set of all German researchers is a subset of the set of all researchers
 - If query asks for German researchers, use set of all researchers, rather than all people
- Subsumption works well for GROUP BY

Materialized View

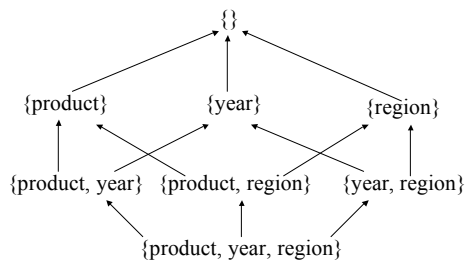
```
SELECT product, year, region, sum(price * vol)
FROM Order
GROUP BY product, year, region;
```

GROUP BY
product, year



```
SELECT product, year, sum(price * vol)
FROM Order
GROUP BY product, year;
```

Computation Graph of Cube



Online Aggregation

- Get approximate result very quickly
- Result (conf. intervals) get better over time
- Based on random sampling (difficult!)
- No product supports this yet

```
SELECT cust, avg(price)
FROM Order
GROUP BY cust
```

Cust	Avg	+/-	Conf	
Heinz	1375	5%	90%	<input checked="" type="checkbox"/>
Ute	2000	5%	90%	<input type="checkbox"/>
Karin	-	-	-	<input checked="" type="checkbox"/>

Ranking, Top N

- Top 10 bestselling products in 2000
- (Syntax: Carey, Kossmann 1997)

```
SELECT z.product, sum(z.price * z.vol) as rev
FROM Order a, Time z
WHERE a.date = z.date AND z.year = 2000
ORDER BY rev DESC
STOP AFTER 10;
```

Top N: Example 2

- How rich are the top 10.000 on an average?

```
SELECT AVG(r.score)
FROM (SELECT p.wealth as score
FROM People p
ORDER BY score DESC
STOP AFTER 10000) r;
```

Top N and Updates

- Reduce salary of worst 5% batters by 20%.

```
UPDATE Player SET salary = 0.8 * salary
WHERE id IN (SELECT id
             FROM Player
             ORDER BY batting_avg
             STOP AFTER 5% WITH TIES);
```

Top N in Database Products

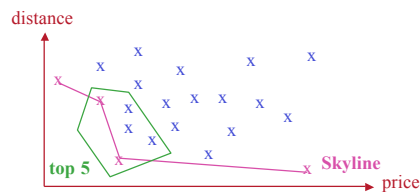
- Microsoft SQL Server
 - Separate **set rowcount N** Clause
- Oracle
 - **rownum < N** Predicate in WHERE Clause
- IBM UDB
 - **fetch first N rows only**
- Questions to ask:
 - Is it possible to output the rank?
 - How to deal with ties?

Skyline Queries

- What happens if the user is interested in hotels which are cheap and near the lake
- that is, there is more than one target
- 1. Try: top N query with **scoring function**
 - e.g., **ORDER BY price * x + distance * y**
 - how do you set *x, y*?
 - result contains many similar hotels
- 2. Try: Skyline query

Skyline Queries

- Return all **incomparable** hotels



Skyline

- Search for cheap hotels near the beach
- (Syntax: Börszöny, Kossmann, Stocker 2001)

```
SELECT *
FROM Hotels h
WHERE h.island = „Bahamas“
SKYLINE OF price MIN, distance MIN;
```

Skyline of New York

Find buildings that are high and close to the river.
Consider each street differently.

```
SELECT *
FROM Buildings
WHERE city = „New York“
SKYLINE OF heigth MAX, distance MIN, street DIFF;
```