Columnar databases

OLAP Query Performance in Column-Oriented Databases

Overview

1. Introduction
   - What is OLAP and data warehousing?
   - Multidimensional data model
2. Column-oriented model
   - Column-oriented storage model
   - Column-oriented processing model
3. Basic optimization techniques
   - Compression
   - Block Iteration
   - Late materialization
4. Advanced optimization techniques
   - Invisible join
   - DDTA-join
   - Parallelization
5. Experiments
   - CDDTA-join
   - Parallelization and SADAS database system

Overview

1. Introduction
   - What is OLAP and data warehousing?
   - Multidimensional data model
2. Column-oriented model
   - Column-oriented storage model
   - Column-oriented processing model
3. Basic optimization techniques
   - Compression
   - Block Iteration
   - Late materialization
4. Advanced optimization techniques
   - Invisible join
   - DDTA-join
   - Parallelization
5. Experiments
   - CDDTA-join
   - Parallelization and SADAS database system

What is **OLAP**?

- An abbreviation for OnLine Analytical Processing.
  - A category of database processing
  - Used in **data warehouses**, which are decision support systems
- Term points out the differences between *operational systems* and *decision support systems*
  - *Data warehouses* use *OLAP* (OnLine Analytical Processing)
  - *Operational systems* (e.g. invoicing system) use *OLTP* (OnLine Transaction Processing)

1. Introduction (2/3)

- Some typical features of OLAP systems

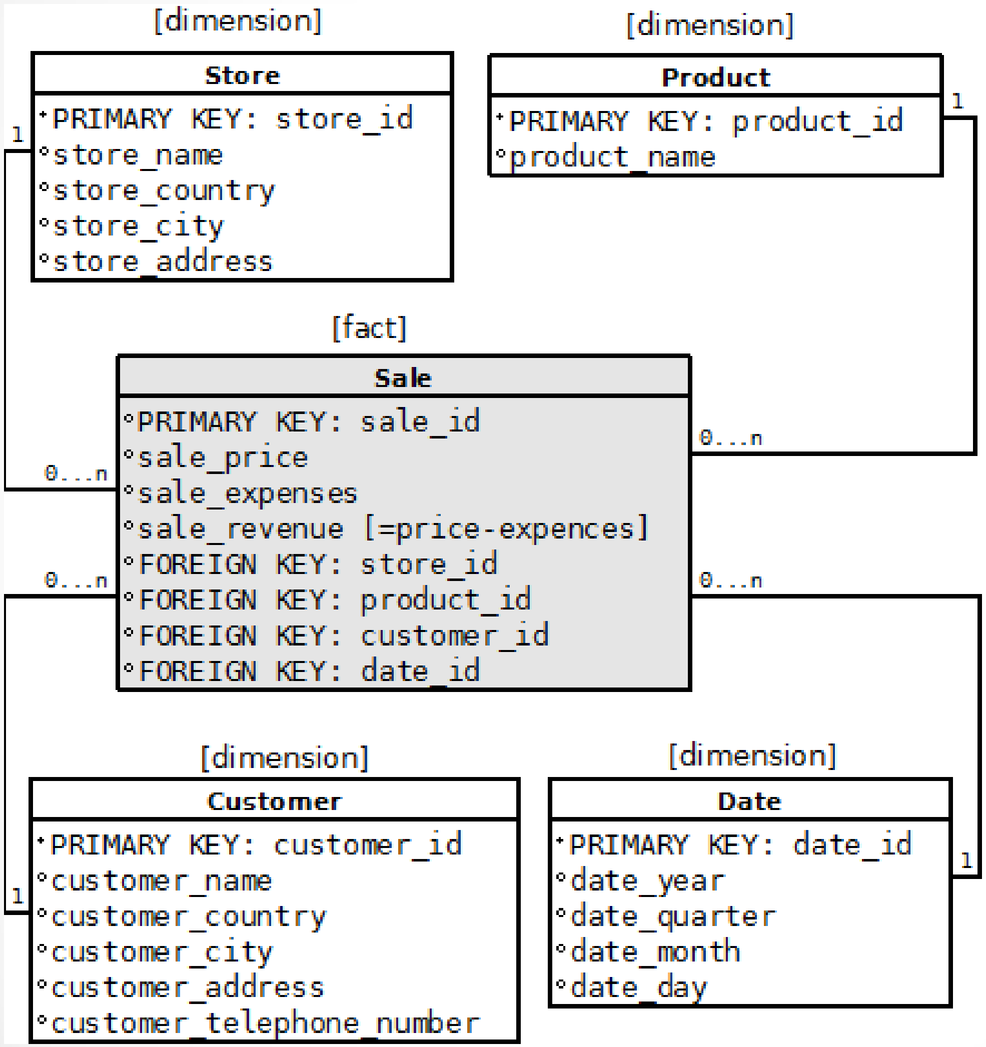
  1. **Multidimensional data model**
  2. **Complex ad hoc, read intensive queries**
  3. **Query response time is important**

- Some typical features of OLTP systems

  1. Normalized data model
  2. Predictable read, **update and delete** operations
  3. Transaction throughput time is important

1. Introduction (3/3)

Multidimensional data model



- One fact table and several dimension tables
- **Large fact table** holds millions of records
  - Quickly enlarging table
- **Smaller dimension tables** hold few thousand records
  - Static data
- Fact table measures are viewed through dimensions
  - For example all sales in year 1993
- Also called **Star Schema**

Overview

1. Introduction
   - What is OLAP and data warehousing?
   - Multidimensional data model
2. Column-oriented model
   - Column-oriented storage model
   - Column-oriented processing model
3. Basic optimization techniques
   - Compression
   - Block Iteration
   - Late materialization
4. Advanced optimization techniques
   - Invisible join
   - DDTA-join
   - Parallelization
5. Experiments
   - CDDTA-join
   - Parallelization and SADAS database system

2. Column-oriented model (1/1)

Column-oriented database

- An alternative to traditional row-oriented database
- Relations are stored and/or processed as columns
- Aims at better I/O and cache efficiency
- Good results with query intensive OLAP systems
- **May be implemented at processing or at storage level:**

	Column-oriented **storage** model	Row-oriented **storage** model
Column-oriented **processing** model	**Native** query processing engines	**Enabled** query processing engines
Row-oriented **processing** model	**Enabled** query processing engines	**Native** query processing engines

Overview

1. Introduction
   - What is OLAP and data warehousing?
   - Multidimensional data model
2. Column-oriented model
   - Column-oriented storage model
   - Column-oriented processing model
3. Basic optimization techniques
   - Compression
   - Block Iteration
   - Late materialization
4. Advanced optimization techniques
   - Invisible join
   - DDTA-join
   - Parallelization
5. Experiments
   - CDDTA-join
   - Parallelization and SADAS database system

3. Basic optimization techniques (1/1)

- Compression
  - Diminishes the disk and memory space needed for storing the data
  - Improves I/O and cache efficiency
  - **Effective especially in column-oriented databases**
- Block iteration
  - Data is fetched as blocks, not one row / function call
  - **Diminishes the number of expensive function calls**
- Late materialization
  - Materialization refers to the constructing of columns into final tuples of the resultset
  - **Late materialization postpones the constructing of resultset tuples (rows) to the end part of the query execution**

Overview

1. Introduction
   - What is OLAP and data warehousing?
   - Multidimensional data model
2. Column-oriented model
   - Column-oriented storage model
   - Column-oriented processing model
3. Basic optimization techniques
   - Compression
   - Block Iteration
   - Late materialization
4. Advanced optimization techniques
   - Invisible join
   - DDTA-join
   - Parallelization
5. Experiments
   - CDDTA-join
   - Parallelization and SADAS database system

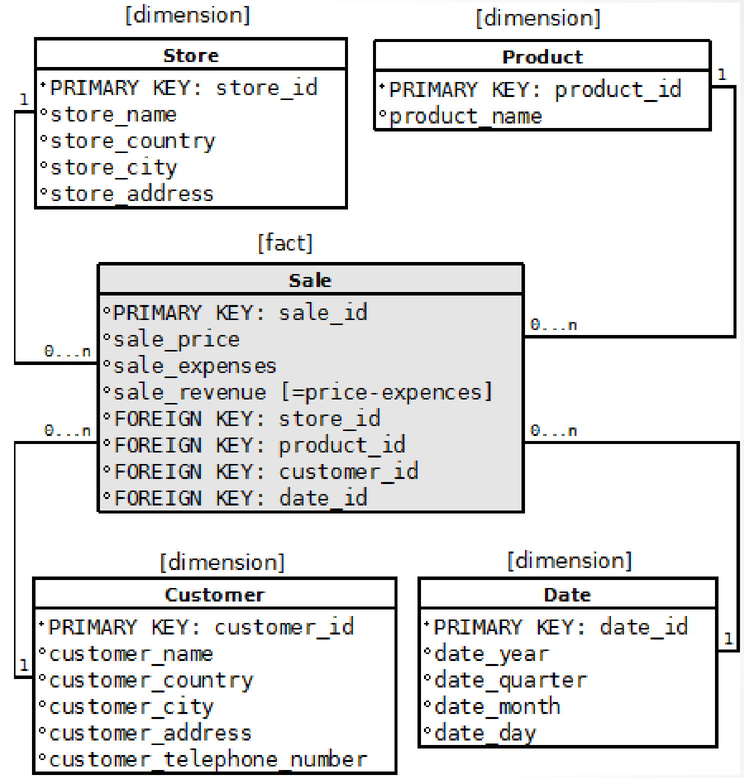
4. Advanced optimization techniques (1/9)

- Invisible join
  - Late materialization technique
  - Designed especially for multidimensional star schema
- DDTA-join
  - Directly Dimensional Tuple Accessing
    - Dimension table column attributes may be accessed directly, with memory address mapped against foreign key in fact table
  - Row-oriented processing for fact table and column-oriented processing for dimension tables
- Parallelism
  - Two categories
    1. Shared address space (shared memory system)
    2. Distributed address space (shared nothing system)
  - May be implemented with the help of special software and library APIs

3. Basic optimization techniques (2/9)

Example query:

```
SELECT
        c.customer_country, st.store_country,
        d.date_year,
        SUM(s.sale_price) AS sum_price
FROM
        customer c, sale s, store st, date d
WHERE
        s.customer_id = c.customer_id
        AND s.store_id = st.store_id
        AND s.date_id = d.date_id
        AND c.customer_country = ‘Russia’
        AND st.store_country = ‘Finland’
        AND d.date_year >= 1992
        AND d.date_year <= 1997
GROUP BY
        c.customer_country,
        st.store_country,
        d.date_year
ORDER BY
        d.date_year ASC,
        s.sale_price DESC;
```



Result: Total sum of yearly prices that Russian customers paid in 1992-1997 for products they bought from stores located in Finland

Customer_country	Store_country	Date_year	Sum_price
Russia	Finland	1993	200

4. Advanced optimization techniques (3/9)

Invisible join (1/3): creating filters

Apply ”country = 'Russia'” On Customer Table

[DIMENSION]

CUSTOMER	
customer_id	customer_country
1	Finland
2	Russia
3	Russia
4	Finland
5	Sweden

0
1
1
0
0

Hash Table (or bit-map)
Containing Keys 2 and 3

Apply ”country = 'Finland'” On Store Table

[DIMENSION]

STORE	
store_id	store_country
1	Finland
2	Russia
3	Sweden

1
0
0

Hash Table (or bit-map)
Containing Key 1

Apply "year in [1992,1997]" On Date Table

[DIMENSION]

DATE	
date_id	date_year
02031990	1990
04091993	1993
11121993	1993
13101996	1996
13091997	1997
25051998	1998

0
1
1
1
1
0

Hash Table Containing Keys
04091993, 11121993,
13101996 and 13091997

Create a hash filter for dimension columns

- customer.customer_country
- store.store_country
- date.date_year

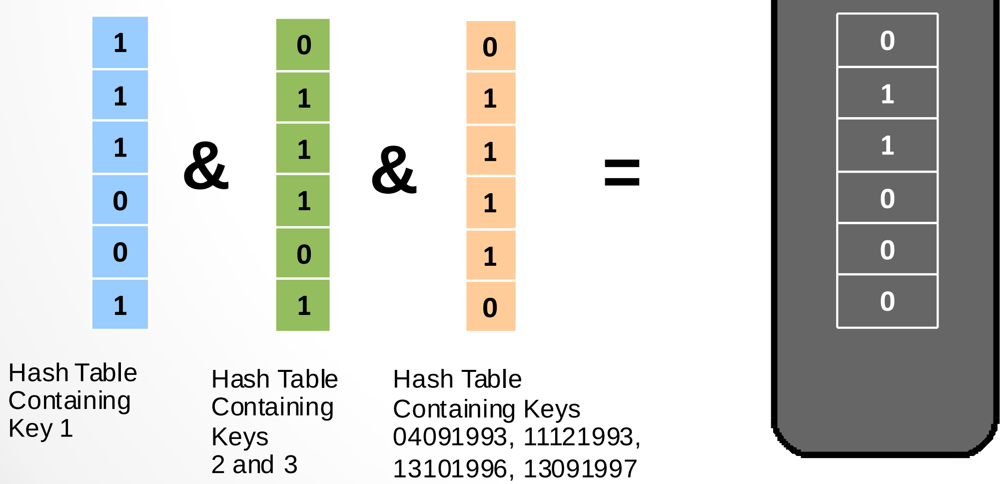
based on predicate selection of the query

4. Advanced optimization techniques (4/9)

Invisible join (2/3): Generating Global result vector

[FACT]

SALE				
sale_id	store_id	customer_id	date_id	sale_price
1	1	1	02031990	100
2	1	2	04091993	50
3	1	3	11121993	150
4	3	3	13101996	300
5	2	4	13091997	50
6	1	2	25051998	200



Generate a global bitmap vector using the hash filters and the corresponding foreign keys

- sale.customer_id
- sale.store_id
- sale.date_id

in fact table.

The global bitmap vector indicates the position of all records in 'sale'-table, which satisfy the predicate selection of dimension columns.

4. Advanced optimization techniques (5/9)

Invisible join (3/3): Output join results

customer_id
1
2
3
3
4
2

+

0
1
1
0
0
0

→

customer_id
2
3

+

customer_country
Finland
Russia
Russia
Finland
Sweden

=

Customer_country
Russia
Russia

store_id
1
1
1
3
2
1

+

0
1
1
0
0
0

→

store_id
1
1

+

store_country
Finland
Russia
Sweden

=

store_country
Finland
Finland

date_id
02031990
04091993
11121993
13101996
13091997
25051998

+

0
1
1
0
0
0

→

date_id
04091993
11121993

join

date_id	date_year
02031990	1990
04091993	1993
11121993	1993
13091997	1997
25051998	1998

=

date_year
1993
1993

Extract the wanted attributes from dimension columns, using the global bitmap vector on foreign keys in 'sale' table and then the foreign keys on dimension columns.

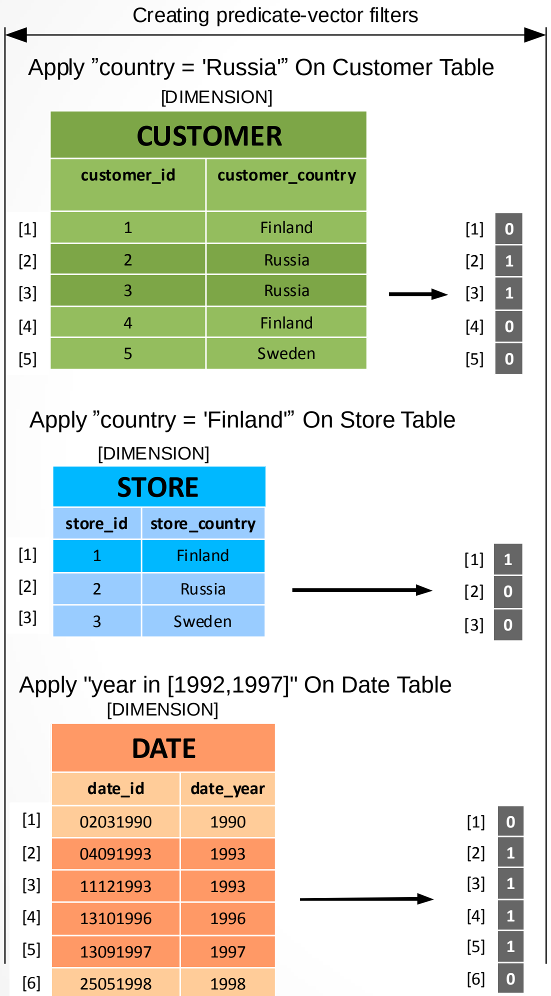
4. Advanced optimization techniques (6/9)

- Invisible join disadvantages
  - The foreign key columns in 'sale' table are scanned twice
  - Join result bitmaps in phase 2 are produced on 'sale' table, that has a huge size.

- DDTA-join tries to improve these weaknesses
  - Row-oriented processing for fact table and column-oriented processing for dimension tables
  - Large fact table is scanned only once
  - No need to create bitmap for fact table
  - Foreign keys in the fact table are mapped directly to the memory address of (memory resident) dimension column values

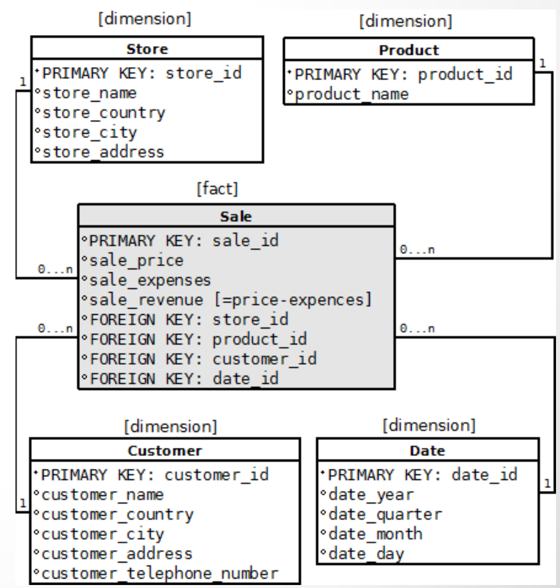
4. Advanced optimization techniques (7/9)

DDTA-join (1/3): Creating predicate-vector filters



Create predicate-vector bitmaps for dimension tables

- customer
- Store
- date



4. Advanced optimization techniques (8/9)

DDTA-join (2/3): Perform full table scan on fact table

Creating predicate-vector filters

Apply ”country = 'Russia'” On Customer Table

[DIMENSION]

CUSTOMER

	customer_id	customer_country
[1]	1	Finland
[2]	2	Russia
[3]	3	Russia
[4]	4	Finland
[5]	5	Sweden

[1]	0
[2]	1
[3]	1
[4]	0
[5]	0

Apply ”country = 'Finland'” On Store Table

[DIMENSION]

STORE

	store_id	store_country
[1]	1	Finland
[2]	2	Russia
[3]	3	Sweden

[1]	1
[2]	0
[3]	0

Apply "year in [1992,1997]" On Date Table

[DIMENSION]

DATE

	date_id	date_year
[1]	02031990	1990
[2]	04091993	1993
[3]	11121993	1993
[4]	13101996	1996
[5]	13091997	1997
[6]	25051998	1998

[1]	0
[2]	1
[3]	1
[4]	1
[5]	1
[6]	0

DDTA-JOIN

[FACT]

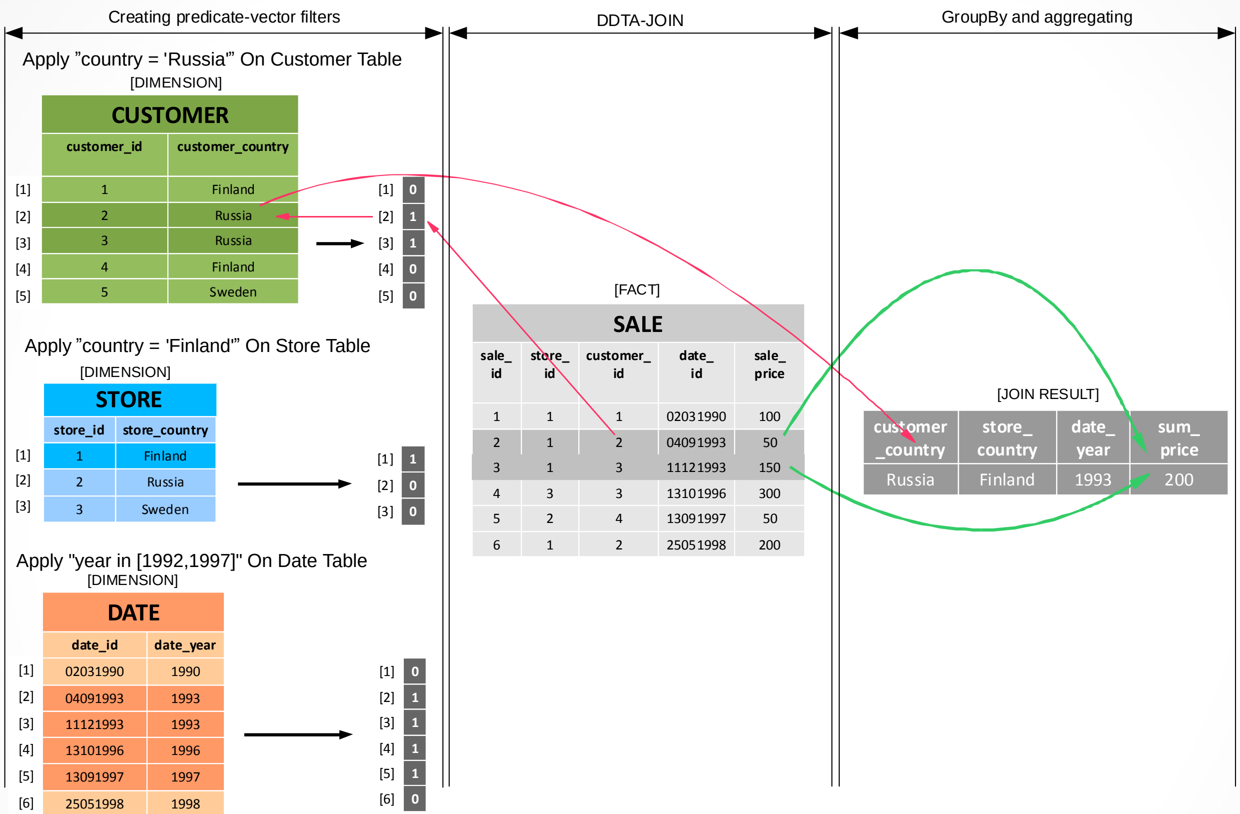
SALE

sale_ id	store_ id	customer_ id	date_ id	sale_ price
1	1	1	02031990	100
2	1	2	04091993	50
3	1	3	11121993	150
4	3	3	13101996	300
5	2	4	13091997	50
6	1	2	25051998	200

- Perform a full table scan on ‘sale’ table.
- For each tuple in ‘sale’ table find the corresponding value from dimension columns, using predicate-vector bitmaps . This is a fast array lookup, because foreign keys in ‘sale’ table can be mapped to the memory address of dimension columns.

4. Advanced optimization techniques (9/9)

DDTA-join (3/3): GroupBy and aggregating



- Create a join between 'sale'-tuple and dimension column if it satisfies the query expression.
- Perform aggregation, grouping and ordering operations on tuples.
- Return the resultset.

Overview

1. Introduction
   - What is OLAP and data warehousing?
   - Multidimensional data model
2. Column-oriented model
   - Column-oriented storage model
   - Column-oriented processing model
3. Basic optimization techniques
   - Compression
   - Block Iteration
   - Late materialization
4. Advanced optimization techniques
   - Invisible join
   - DDTA-join
   - Parallelization
5. Experiments
   - CDDTA-join
   - Parallelization and SADAS database system

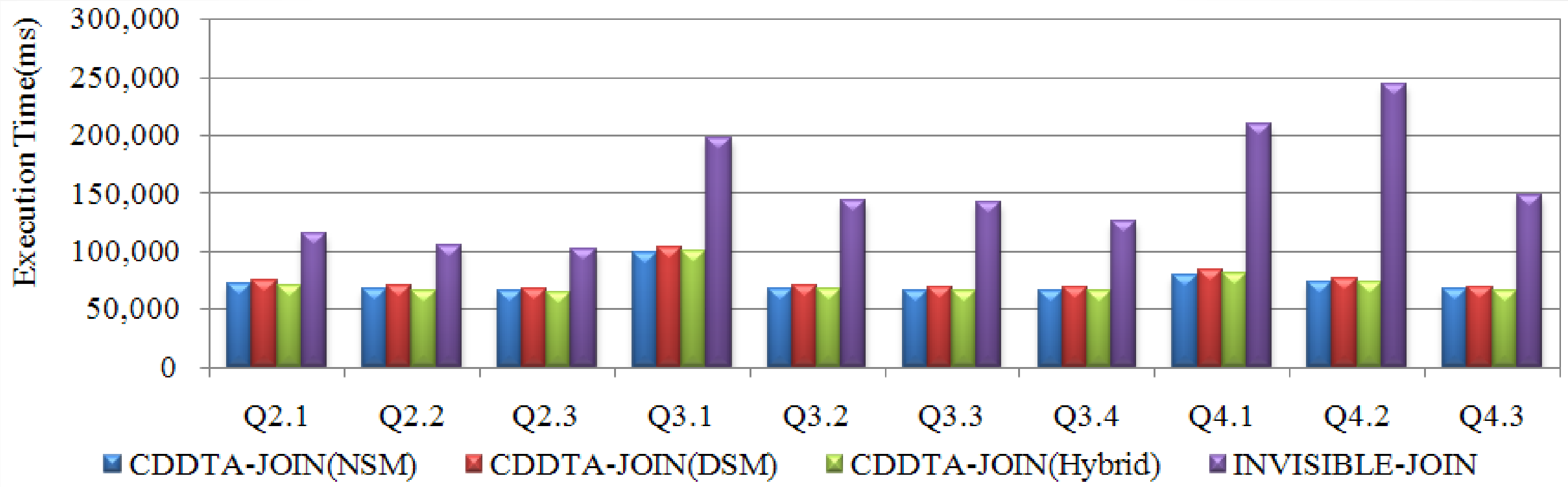
5. Experiments (1/4)

Experiment with CDDTA-join

- CDDTA-join = DDTA-join with column-oriented model
- 3 alternative storage models for the fact table
  - Row-oriented storage model
  - Column-oriented storage model
    - DDTA-join uses row-wise processing model for the fact table, so the fact table attributes must to be converted on-the-fly into rows
  - Hybrid storage model.
    - Only the foreign keys of the fact table are organized as row table and measure attributes were left as column arrays

5. Experiments (2/4)

CDDTA-join experiment results



- Q2.1 – Q4.3 refer to different queries in star schema benchmark
- CDDTA-join performed remarkably well, sometimes even halving the response time compared to invisible join.

5. Experiments (3/4)

- Parallel porting and SADAS database
  - SADAS is a commercial, column-oriented database for data warehousing
  - Experimental work of changing the SADAS kernel to support shared memory and distributed memory parallelism

- Experiment details
  - OpenMP software for shared address space model
  - MPI technology for distributed address space model

5. Experiments (4/4)

SADAS parallelization experiment results:

Version	Number of nodes	Duration
Sequential (Original code)	1	9,15
Shared Memory (OpenMP)	2	4,71
Distributed Memory (MPI)	2	4,65
OpenMP + MPI	2	2,43

- Best result was obtained by using both technologies, OpenMP and MPI, together
- With 4 nodes (excluded from the table) the execution time dropped as low as to 1,25 with OpenMP/MPI hybrid solution

Conclusion

- OLAP technology is used in data warehouses and Decision Support Systems
- Columnar approach in databases may be implemented at
  - Storage level
  - Processing level
- Basic optimization techniques in Column-oriented databases include
  - Compression
  - Block iteration
  - Late materialization
- Advanced optimization techniques include
  - Invisible join
  - CDDTA-join
  - Parallel porting
- DDTA-join and parallelization experiments with column-oriented databases show good results

References

1) D. J. Abadi, S. R. Madden and N. Hachem, "Column-stores vs. row-stores: how different are they really?," *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, Vancouver, Canada, 2008.
2) R. Aversa, B. Di Martino, S. Venticinque and L. De Rosa , "Parallel porting and performance evaluation of a column based OLAP system," in *Proceedings of the IADIS International Conference Applied Computing 2009, 19-21 November, Rome, Italy, 2 Volumes*, 2009, pp. 67–75.
3) N. Bruno, "Teaching an old elephant new tricks," *In fourth biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, California, USA*, 2009.
4) S. Chaudhuri , U. Dayal, "An overview of data warehousing and OLAP technology," *ACM SIGMOD Record*, vol. 26, 1997, pp. 65-74.
5) E.F. Codd, S.B. Codd and C.T. Salley, "Providing OLAP to user-analysts: an IT mandate," Technical report, E.F. Codd and Associates, 1993.
6) M. Jiao, Y. Zhang, S. Wang and X. Zhou, "CDDTA-JOIN: one-pass OLAP algorithm for column-oriented databases," in *Web Technologies and Applications: 14th Asia-Pacific Web Conference, APWeb 2012, Kunming, China, April 11-13, Proceedings*, vol. 7235, 2012, pp. 448-459.
7) P. O'Neil, B. O'Neil, X. Chen, "Star schema benchmark," *2009*, Available: http://www.cs.umb.edu/~poneil/StarSchemaB.PDF
8) Y. Zhang, W. Hu and S. Wang ,MOSS-DB: A hardware-aware OLAP Database, in *Web-Age Information Management, 11th International Conference, WAIM 2010, Jiuzhaigou, China, July 15-17, 2010. Proceedings*, vol. 6184, 2010, pp. 582–594.
9) M. Zukowski , N. Nes , P. Boncz, DSM vs. NSM: CPU performance tradeoffs in block-oriented query processing, in *Proceedings of the 4th international workshop on Data management on new hardware*, Vancouver, Canada, 2008

Thanks for listening

Columnar databases

OLAP Query Performance in Column-Oriented Databases