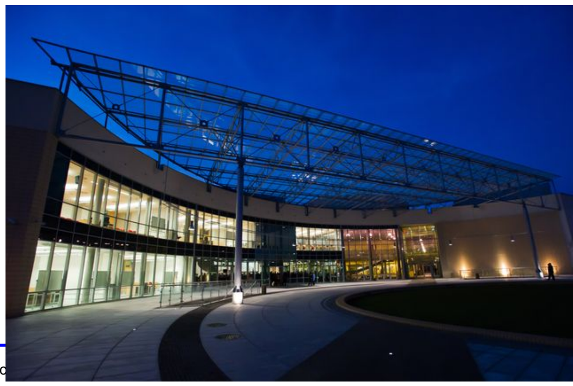
# Research at the Poznań University of Technology

Robert Wrembel
Poznań University of Technology
Institute of Computing Science
Poznań, Poland
Robert.Wrembel@cs.put.poznan.pl

# Outline

- A (short?) presentation of the Poznań University of Technology and the Faculty of Computing Science
- Research works conducted at FCS
- Focus on research on data processing



# Poznań University of Technology

- 10 faculties
- approx. 20 000 students



# Faculty of Computing

- Institute of Computing Science
  - 9 full professors
  - 11 post doctoral degrees (PhD DSc)
  - 54 PhDs
- Dept. of Control and Systems Engineering
- Dept. of Computer Engineering

# Education

- Bachelor in engineering (7 sem)
- Master of science (3 sem)
- PhD
- Fields of studies
  - Informatics
    - 140-180 Bac students
    - 150-210 MSc students
  - Bioinformatics (since 2010, with AMU)
  - Automatics and Robotics
- Full-time and part-time studies



# Informatics MSc

- Electronic economy
- Intelligent Decision Support Systems
- Management Information Systems
- Computer Network and Distributed Systems
- Embedded and Mobile Systems
- Data Processing Technologies
- Software Development Technologies
- Software Engineering
- Computer Engineering

# Faculty of Computing

- Evaluation by state committees
  - First category (research)
  - Distinction - top category (teaching)



# Students

- Holders of scholarships from the Ministry of Higher Education
- International finals of IEEE Computer Society International Design Competition (CSIDC) (2000-2006)
- International finals of Microsoft Imagine Cup (2006-2010)

Imagine Cup (France, 2008)


Imagine Cup (Yokohama, 2005)

# Institute of Computing Science

- Lab (Chair) of **Operational Research and Artificial Intelligence**
- Lab (Chair) of **Intelligent Decision Support Systems**
- Lab (Chair) of **Information Systems**
- Lab (Chair) of **Algorithm Design and Programming Systems**



# EU Projects

- CALIBRE - programming environments (6 FP, 2005-2006)
- COMPUVAC - designing vaccines (7 FP, 2005-2009)
- BIOPTRAIN - optimization of algorithms for bioinformatics (7 FP, 2005-2009)
- METAFUNCTIONS - genomics (7 FP, 2005-2008)

# EU Projects

- **INDECT** - urban security (7 FP, 2009-2013)
- **ICT WIELKOPOLSKA** – regional cluster of companies from IT (7 FP, 2009-2010)
- **SOA** - new technologies for e-economy and e-government (Structural Funds, 2009-2012)
- **PROTEUS** - integrated mobile system for crisis management (Structural Funds, 2009-2013)
- **ERA INŻYNIERA** - increasing capacity of teaching engineers (EFS, 2008-2012)
- **TECH-INFO** - increasing capacity of teaching computer science (Human Capital, 2010-2014)



# Cooperation with Business

- Poznań Public Transport
- Volkswagen Poznań
- Allegro
- Hospitals in Poland
- Comarch
- IBM Poland
- Roche

## Cooperation with Business

- 2006: Microsoft Innovation Center
- 2007: Eclipse Support Center
- 1993: Poznań Supercomputer and Network Center
- Two spin-off companies



## Research at the Faculty of Computing Science

- LAB of Algorithm Design and Programming Systems
  - algorithms design and the complexity analysis of combinatorial problems
  - scheduling theory, especially in multiprocessor systems
  - design of parallel algorithms
  - compilers design
  - computer aided design of electronic systems
  - combinatorial aspects of molecular biology
  - sequencing by bybridization (GPU computation)
  - protein data analysis
  - software engineering
    - software design methods (the XPrince method, based on Prince2)
    - estimating costs of software development (cooperation with Roche)
    - software testing methods (cooperation with Roche)

# Research at the Faculty of Computing Science

## ➲ LAB of Operational Research and Artificial Intelligence

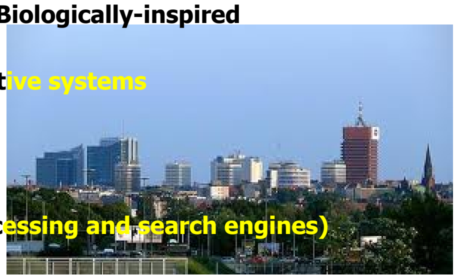
- algorithms for discrete-continuous scheduling problems
- expert system for multicriteria project scheduling
- intelligent building systems
- algorithms for traffic control
- signal processing
- speech recognition
- multimedia
- multiagent expert systems
- reasoning under uncertainty with knowledge updating
- mobile computing
- computer control systems for environmental protection
- metaheuristics



# Research at the Faculty of Computing Science

## ➲ LAB of Intelligent Decision Support Systems

- methodology and techniques of decision support (multicriteria decision analysis)
- rough set theory, fuzzy set theory and neural networks
- machine learning, data mining, knowledge discovery, intelligent data analysis
- decision support from visual information
- evolutionary computation and Biologically-inspired computing
- artificial life and complex adaptive systems
- mathematical programming
- medical informatics
- software engineering
- information retrieval (text processing and search engines)



# Research at the Faculty of Computing Science

## ➲ LAB of Computing Systems

- distributed algorithms design
  - specification, verification, complexity analysis, self-stabilizing algorithms, failure resilient algorithms
- computer networks
  - design and monitoring, protocol modeling and performance evaluation
- distributed operating systems
  - distributed shared memory, replication management, consistency models and protocols, distributed resource allocation, distributed deadlock and termination detection, distributed recovery, fault tolerant and dependable systems, failure detectors
- distributed programming environments

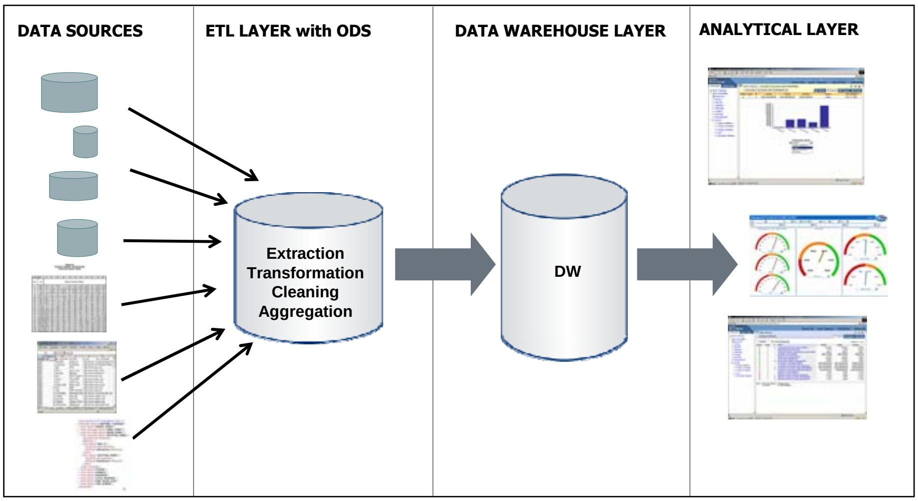


# Research at the Faculty of Computing Science

## ➲ LAB of Computing Systems

- data warehousing
  - evolution, indexing and query processing efficiency, data compression
- sequential OLAP
  - models of processing, query language, efficiency, data structures
- data mining
  - algorithms, data structures
- social networks
  - mining, data structures
- processing XML
  - transactions for WEB Services
- GPU processing

# Standard DW Architecture

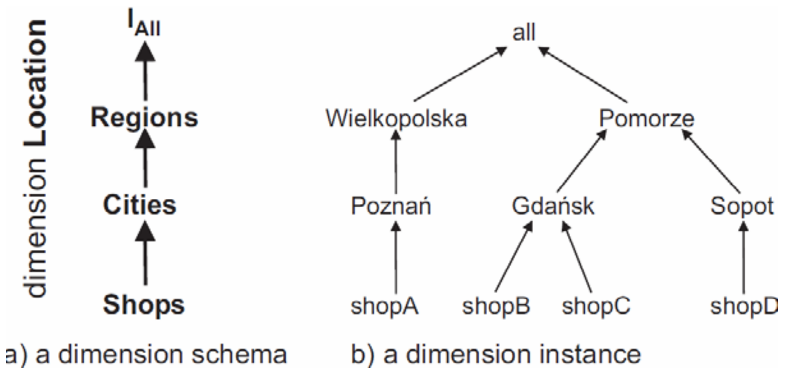





# DW Data Model

- **Facts**
  - **data being analyzed**
    - **sales, telephone calls, insurance**
  - **their quantity is characterized by means of measures**
    - **the number of products sold, tel. call duration, insurance fee**
- **Dimensions**
  - **define the context of an analysis**
    - **sales of chocolate (product) by Walmart (shop) in consecutive months of a given year (time)**
  - **composed of hierarchically organized levels**

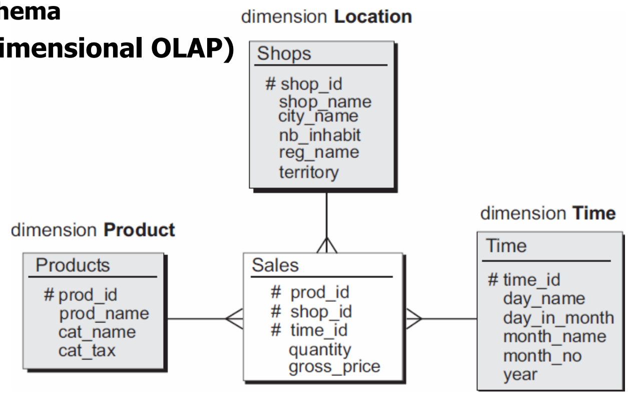
# DW Data Model






# DW Data Model

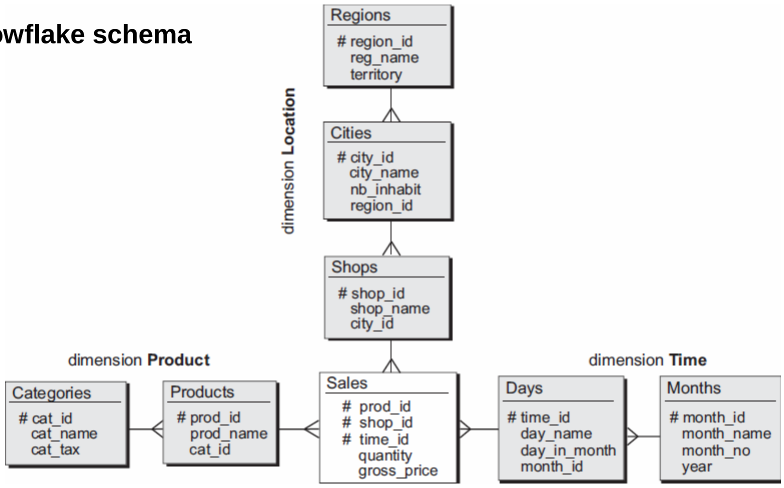
- **Implementation**
  - **ROLAP (Relational OLAP)**
    - **star schema**
    - **snowflake schema**
  - **MOLAP (Multidimensional OLAP)**




**star schema**

# DW Data Model

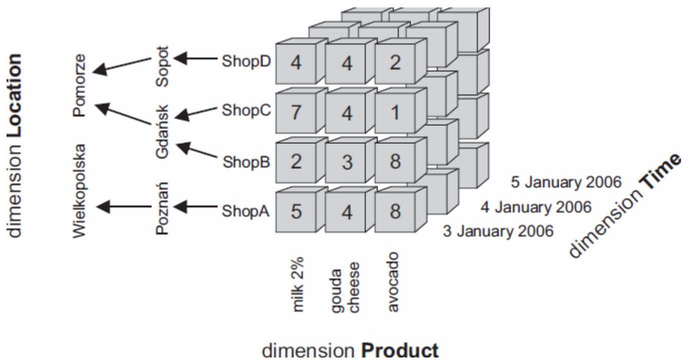
**snowflake schema**






# DW Implementation

**data cube**



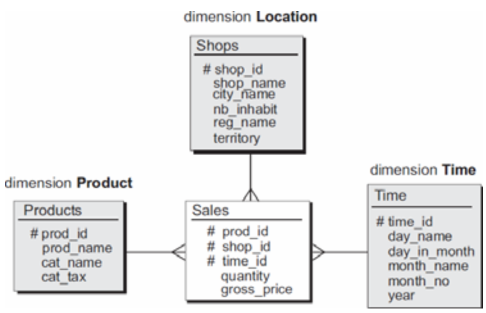
## Star Queries

➲ Star queries
- join fact table with dimension tables
- select ranges of values or sets of values
- compute aggregates along a dimension hierarchy ⇨ roll-up
- most of star queries include the TIME dimension ⇨ necessity to join fact table with the TIME dimension



## Star Queries

```
select shop_name, prod_name, year, sum(gross_price)
from sales sa, products p, shops sh, time t
where sa.prod_id=p.prod_id
and sa.shop_id=sh.shop_id
and sa.time_id=t.time_id
group by shop_name, prod_name, year;
```



# Star Queries

- ➲ Challenge ⇨ star queries' performance
- ➲ Solutions
  - materialized views and query rewriting
  - parallel processing
  - partitioning
  - indexing



# Indexing DW Data

- ➲ **Typically applied indexes**
  - **Bitmap index**
  - **B-tree**
  - **Join index**
  - **Bitmap join index**
- ➲ **Hierarchical indexes**
  - **Multi-resolution bitmap index**
  - **Hierarchical bitmap index**

# Bitmap Index

- Composed of bitmaps
- A bitmap is a vector of bits
  - Every value from a domain has its own bitmap
  - The number of bits = the number of records
- Basic characteristics
  - Efficient in answering equality and range queries
  - BI size depends on the cardinality of an indexed attribute

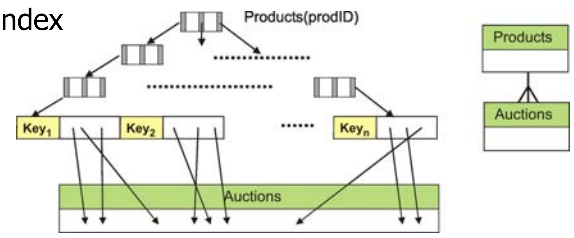
| Auct_ID | Prod_ID | ... |
|---|---|---|
| 1 | IBM T43 | |
| 2 | Dell Vostro 3700 | |
| 3 | IBM T43 | |
| 4 | Toshiba Tecra | |
| 5 | Toshiba Tecra | |
| 6 | Toshiba Tecra | |
| 7 | Dell Vostro 3700 | |
| 8 | Dell Vostro 3700 | |
| 9 | IBM T43 | |
| 10 | Toshiba Tecra | |

| Toshiba Tecra | IBM T43 | Dell Vostro 3700 |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |

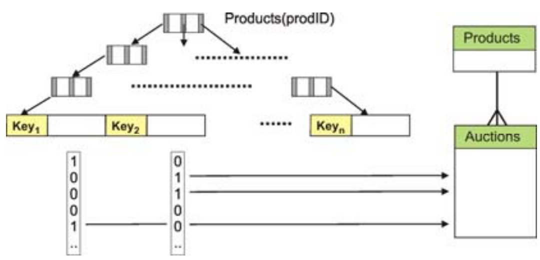


# Join Index

- Join index



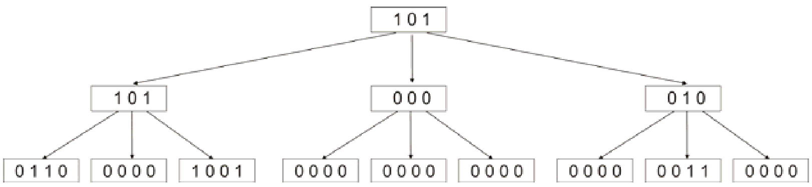

- Bitmap join index




- Do not reflect the hierarchy of a dimension
- Do not support roll-up queries

# Hierarchical indexes

- Multi-resolution bitmap index
  - indexing scientific data
  - lower level: standard bitmap indexes
  - upper level: binned bitmap indexes (defined on data ranges)
- Hierarchical bitmap index
  - Indexing set-valued attributes, optimizing subset, superset, and similarity queries




- Do not support star queries



# Bitmap compressions

- **Byte-aligned Bitmap Compression (BBC)**
  - **[Antoshenkov, Ziauddin, VLDB Journ.96]**
- **Word-Aligned Hybrid**
  - **[Stockinger et al., DOLAP2002, Wu et al., VLDB2004]**
- **Position List WAH (PLWAH)**
  - **[Deliege, Pedersen, PhD 2009]**
- **Run-Length Huffman (RLH)**
  - **[Stabno, Wrembel, Inf. Systems 2009]**

# Bitmap compressions

- Based on the run-length encoding
  - homogeneous vectors of bits are replaced with a bit value (0 or 1) and the vector length
  - `0000000 1111111111 000` ⇨ 07 110 03
- Bitmap is divided into words
  - BBC uses 8-bit words
  - WAH uses 31-bit words
  - PLWAH uses 31-bit words
  - RLH n-bit words



# RLH (1)

- Modified run-length encoding
  - measures and encoded distances between bits of value 1

```
  1   0  0    3     0     3     0 0   1   0 0 0
1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1
```

Clients

| ID | sex |
|---|---|
| 1 | male |
| 2 | female |
| 3 | female |
| 4 | female |
| 5 | male |
| 6 | male |
| 7 | male |
| 8 | female |
| 9 | female |
| 10 | male |
| 11 | male |
| 12 | male |
| 13 | female |
| 14 | female |
| 15 | female |
| 16 | male |
| 17 | female |
| 18 | female |
| 19 | female |

bitmap index

| female | male |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 1 | 0 |
| 1 | 0 |
| 0 | 1 |
| 0 | 1 |
| 0 | 1 |
| 1 | 0 |
| 1 | 0 |
| 0 | 1 |
| 0 | 1 |
| 0 | 1 |
| 1 | 0 |
| 1 | 0 |
| 1 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 0 |
| 1 | 0 |

```
female: 100303001000

male:   030020033
```

# RLH (2)

- **Huffman encoding**
  - **step1: computing frequencies of symbols (distances) in encoded bitmaps**

```
female: 100303001000
male:   030020033
```

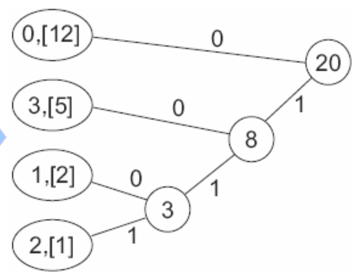
| distance | frequency |
|---|---|
| 0 | 12 |
| 3 | 5 |
| 1 | 2 |
| 2 | 1 |



# RLH (3)

- **Huffman encoding**
  - **step2: building a Huffman tree**

| distance | frequency |
|---|---|
| 0 | 12 |
| 3 | 5 |
| 1 | 2 |
| 2 | 1 |



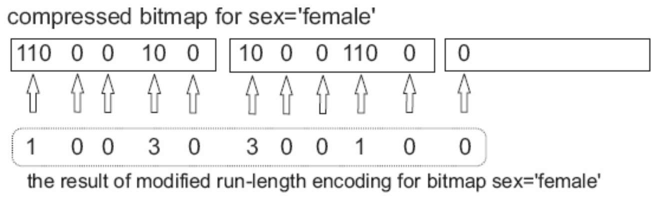

| distance | code |
|---|---|
| 0 | 0 |
| 3 | 10 |
| 1 | 110 |
| 2 | 111 |

  - **an encoded symbol is represented by a path from the root to a leaf**

# RLH (4)

- **Huffman encoding**
  - **step3: replacing distances with their Huffman codes**

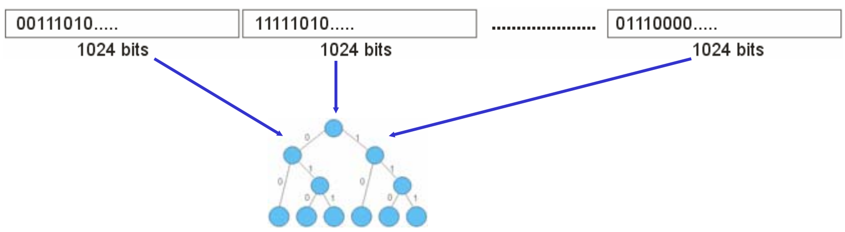
| distance | code |
|---|---|
| 0 | 0 |
| 3 | 10 |
| 1 | 110 |
| 2 | 111 |






# RLH-1024 Compression (1)

- **Dividing a bitmap into 1024-bit sections**
  - **constructing one Huffman tree based on frequencies of distances from all 1024-bit sections**



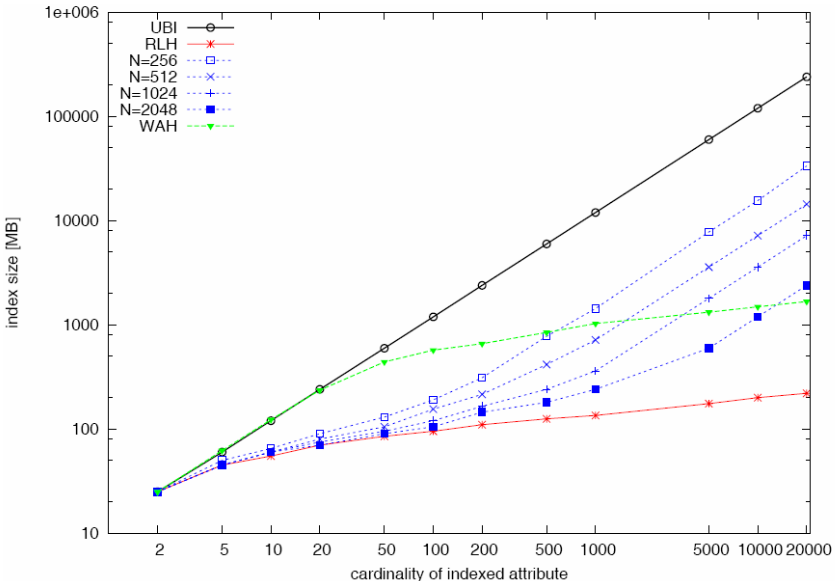

- **Including in the HT all possible distances that may appear in a 1024-bit section**
  - **non-existing distances have assigned the frequency of 1**

# Experimental Evaluation

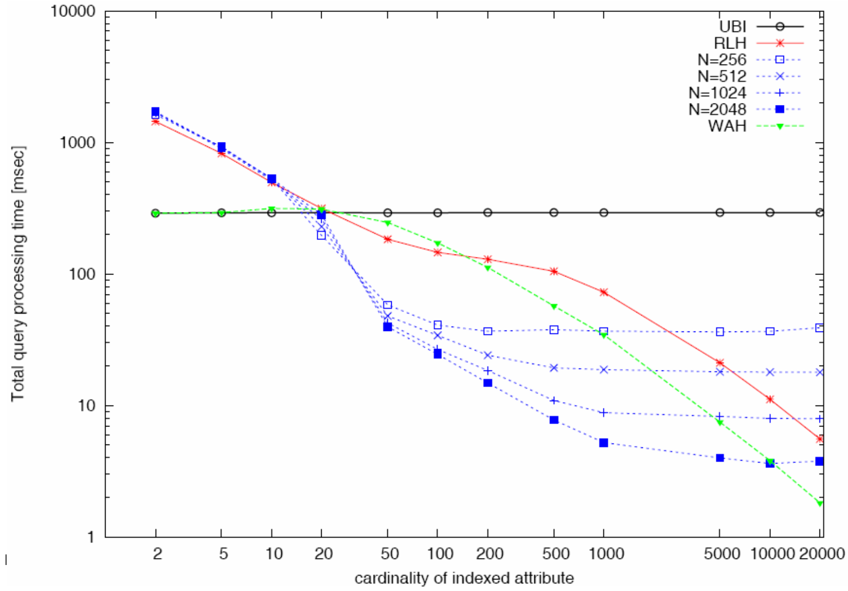
- Comparing RLH, WAH, and uncompressed bitmaps with respect to
  - bitmap sizes
  - query response times
- Implementation in Java
  - data and bitmap indexes stored on disk in OS files
- Experiments run on
  - PC, AMD Athlon XP 2500+; 768 MB RAM; Windows XP
- Data
  - 100 000 000 indexed rows
  - indexed attribute of type integer
    - cardinality from 2 to 1000
    - randomly distributed values



# Size



# Query processing



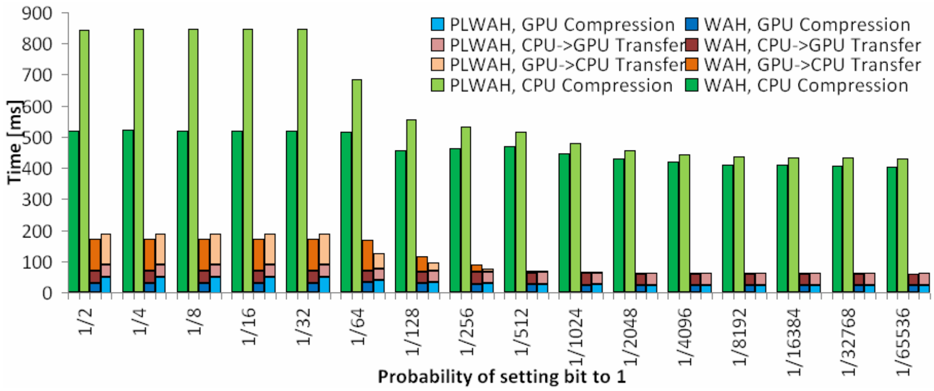



# GPU Processing

- GPU are much faster than CPU
- nGPU in one graphic card ⇨ parallel processing
- GPU processed WAH and PLWAH

- Experiments
  - input bitmaps of size 200MB
  - CPU: Core i7 2.8 GHz
  - GPU: NVIDIA Geforce 285 GTX

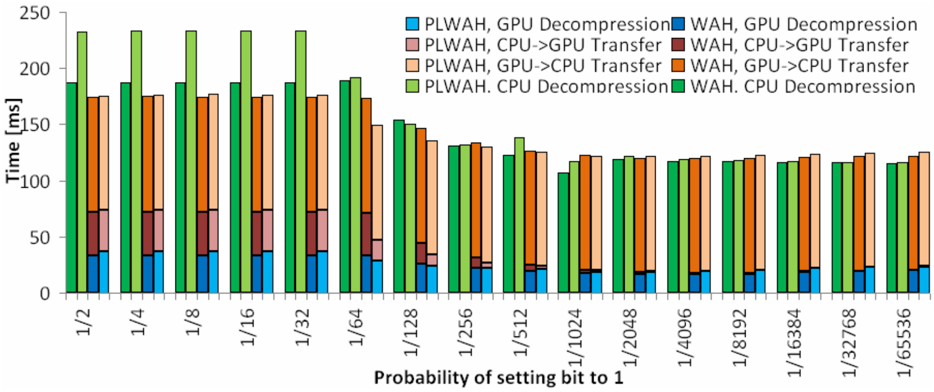
# WAH and PLWAH




**Compression and data transfer**



# WAH and PLWAH




**Decompression and data transfer**

## DW Evolution

- ➲ **Dynamic nature of EDSs**
  - **data dynamics**
    - **user data processing in EDSs ➔ DW refreshing**
  - **schema dynamics**
    - **new user requirements**
    - **dynamic nature of a real world**
    - **what-if analysis**

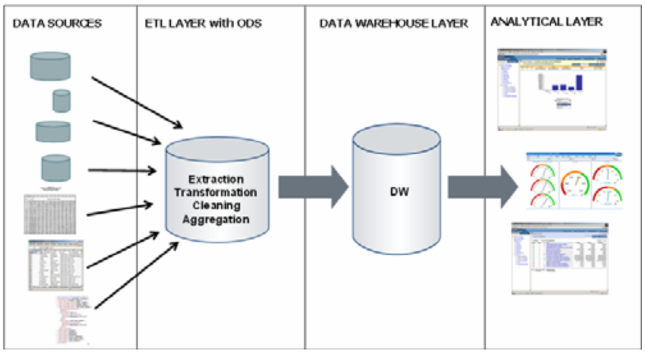


## DW Evolution

- ➲ **Structural changes in data sources**
  - **Wikipedia schema changed every 9-10 days on the average during the last 4 years**
  - **Telecommunication data sources changed their schemas every 7-13 days, on the average**
  - **Banking data sources changed their schemas every 2-4 weeks, on the average**
  - **The most frequent changes concerned increasing the length of a column, changing a data type of a column, and adding a new column**

# ETL Evolution

➲ Structural changes in EDS have impact on
- ETL layer
- DW layer
- Data mart layer
- OLAP layer






# Related Approaches

➲ ETL modeled by materialized views
- rules for view evolution + view definition language
- [Rundensteiner et al., SIGMOD99] ⇨ EVE

➲ Hecateus
- ETL modeled as graph
- rules of ETL evolution
- not compatible with commercial ETL modeling environments
- [Papastefanos et al., ICEIS2008, MEDWa2009]

# ETL Evolution Problems

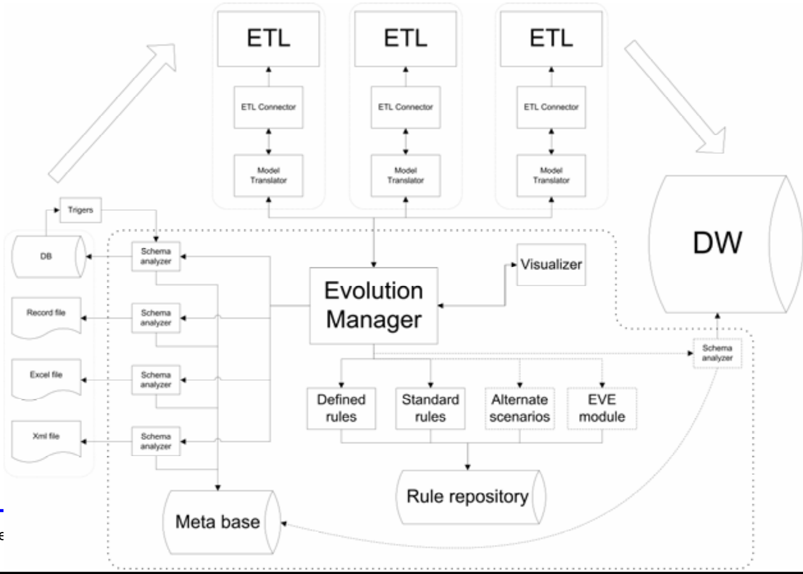
- Workflow representation
  - graph
    - complex
    - easier for expressing evolution rules
  - workflow (ODI, IBM WebSphere Data Stage, MS SQL Server Integration Services)
    - less complex
    - internal implementation may be not known
    - more difficult for expressing evolution rules
    - compatible with commercial tools
- Checking correctness of an evolving ETL



# ETL Evolution

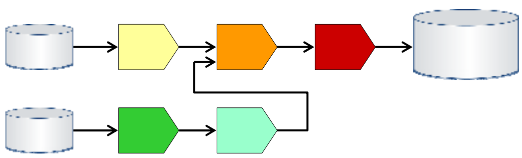
- Detecting changes in EDS
  - triggers
  - metadata snapshot comparison
- Evolution rules attached to ETL steps
- Integrated with Microsoft SQL Server Integration Services
- Limitation ⇨ ETL steps expressed by SQL commands

# ETL Evolution



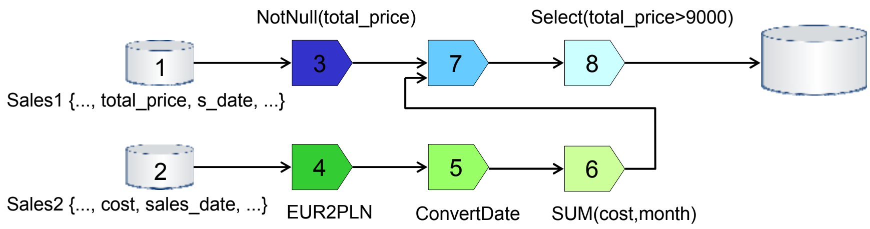



# ETL unsolved problems



- ➲ **ETL optimization**
- ➲ **Workflow transformation**
  - **reordering tasks**
  - **parallelizing tasks**
  - **merging splitting tasks**
- ➲ **Figuring out the set of correct transformations**
- ➲ **Defining cost model of executions**

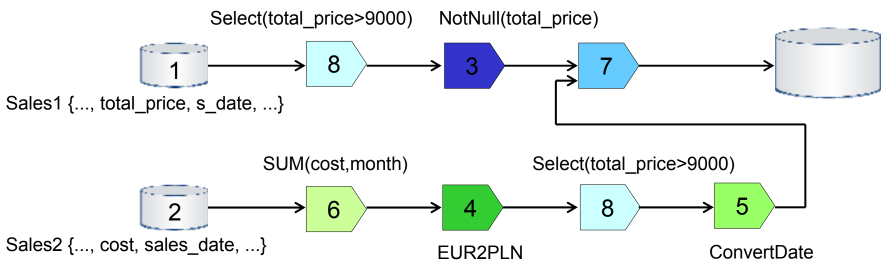
# Example




- **Sales1**
  - **total_price [PLN]**
  - **s_date [yyyy-mm-dd]**
  - **monthly sales**
- **Sales2**
  - **cost [EUR]**
  - **sales_date [dd/mm/yy]**
  - **daily sales**



# Example




- **Minimize the amount of processed data**

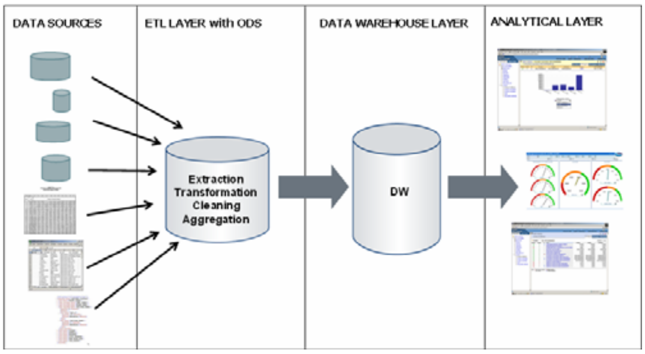
# ETL unsolved problems

- Tasks are often expressed as programs in procedural languages
  - constructing cost model
  - programs may have input parameters and conditional constructs
  - how to interpret and optimize code?
- Commercial systems
  - ???



# ETL Evolution

- Structural changes in EDS have impact on
  - ETL layer
  - DW layer
  - Data mart layer
  - OLAP layer



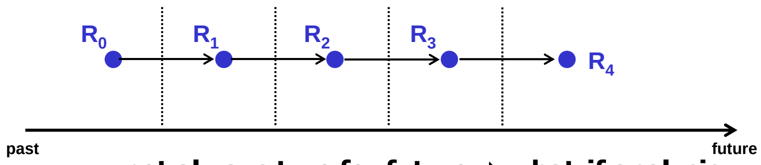
# Related Approaches

- Schema and data evolution
  - [Koeller et al., DOLAP98], [Blaschka et al., DaWaK99], [Hurtado et al., ICDE99, DOLAP99], [Pedersen et al., ICEIS2004], [Fan, Poulovassilis CAiSE2004], [Bentayeb et al., ICAE2008]
- Simulation
  - [Balmin et al., VLDB2000, ICDE2000], [Bellahsene DEXA98]
- Temporal extensions
  - [Chamoni et al., DaWaK99], [Mendelzon et al., VLDB00], [Eder et al., DaWaK01, CAISE02], [Bruckner, Tjoa, JIIS2002], [Malinowski Zimanyi, Springer 2008]
- Versioning
  - [Body et al. DOLAP02, ICDE2003], [Vaisman, Mendelzon, DBPL2001], [Golfarelli et al., ERWorkshops2004, ICDE2007], [Ravat et al., DAWAK2006]
  - MVDW

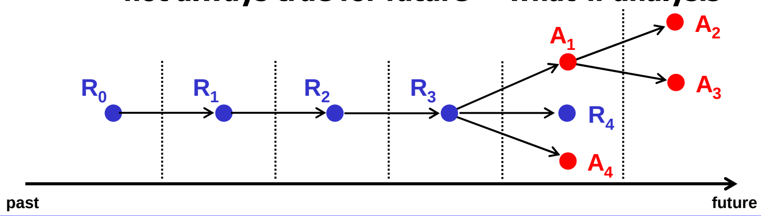


# Limitations

- **The approaches assume that time is linear (DW states are ordered by time)**
  - **true for past**

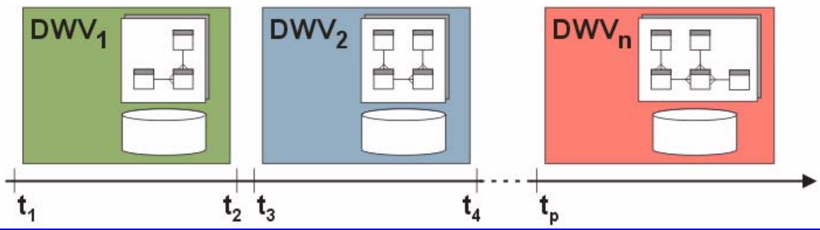



  - **not always true for future ⇨ what-if analysis**



# MVDW Approach

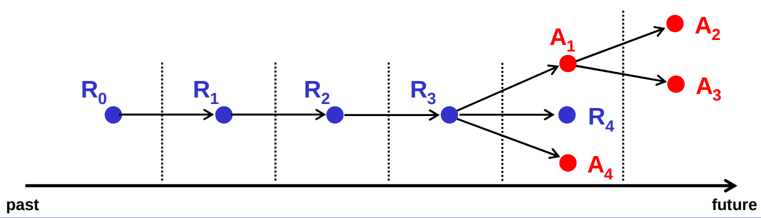
- **Multiversion Data Warehouse**
  - **MVDW is composed of a sequence of its versions**
  - **changes in a DW structure and data reflected in a new explicitly derived version of a DW**
- **DW Version**
  - **a schema version (facts, dimensions, levels, level instances)**
  - **an instance version (stores the set of data consistent with its schema version; measures/cell values)**



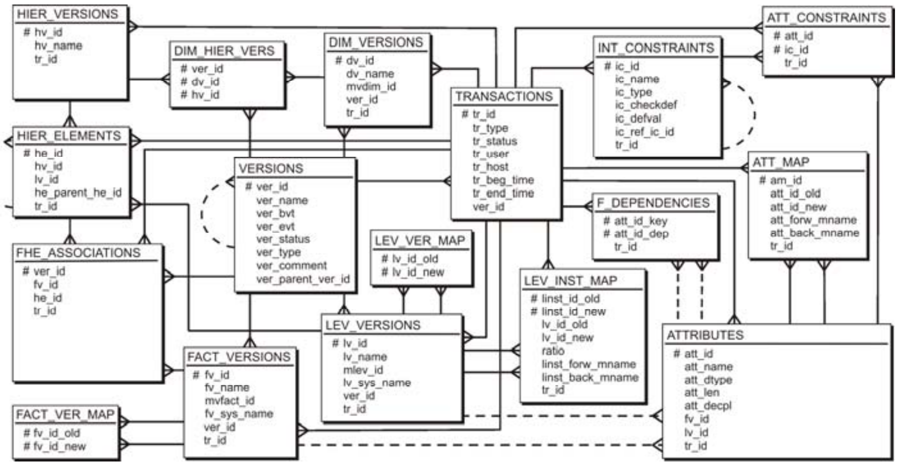



# MVDW Approach

- **Types of DW versions**
  - **real**
    - **reflects changes in real world**
    - **linearly ordered by time they are valid within**
    - **derived from another real version**
  - **alternative**
    - **created for simulation purposes (what-if analysis)**
    - **form DAG**
    - **derived from another real or alternative version**



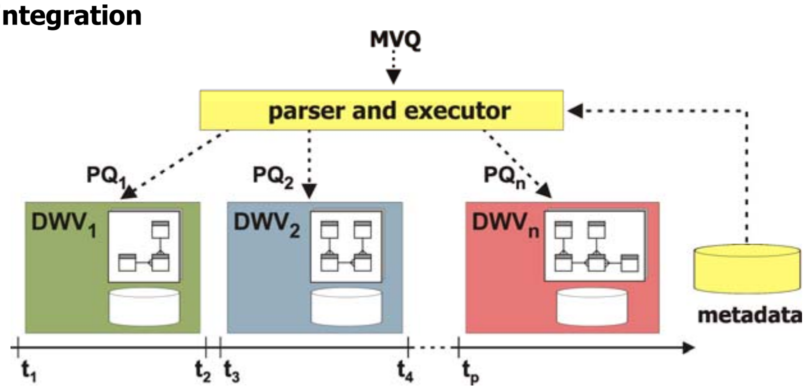
# MVDW Metaschema






# Querying MVDW

- Query decomposition
- PQ execution
- PQ retrieval and presentation
- PQ integration



## Modes of Querying

- Querying the current DW version
  - by default a user addresses the latest real DW version
- Querying the set of real DW versions
  - by specifying time period of interest, real versions are valid within
  - begin validity time - end validity time

```
select ...
from ...
where ...
group by ...
version from date 'begin date' to date 'end date'
```

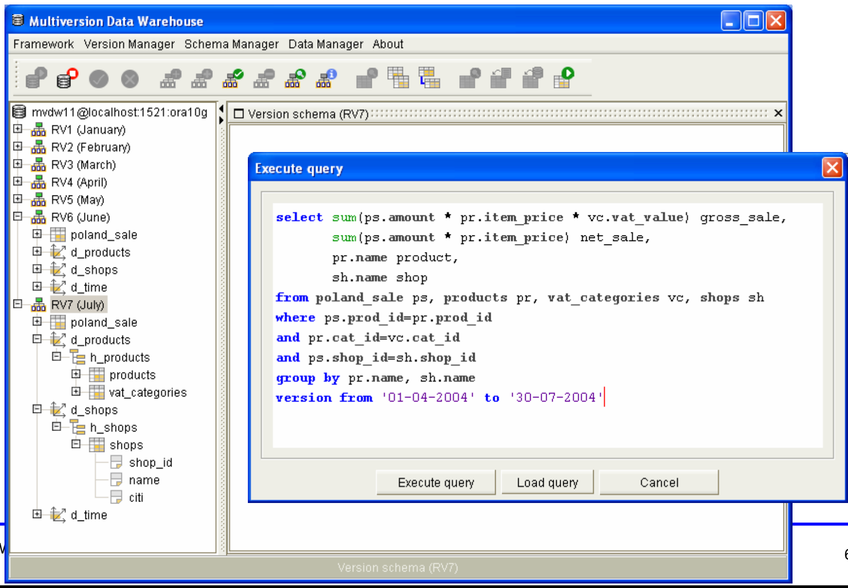


## Modes of Querying

- Querying the set of alternative DW versions
  - a user has to explicitly provide a set of alternative versions of interest

```
select ...
from ...
where ...
group by ...
alternative version in (ver_id | ver_name,..., )
```

# User Interface






# Data Query Limitations

- **All predicates of the SELECT command apply to all DW versions ➔ it is not possible to express a predicate on a single DW version**
- **The query parser is unable to infer appropriate versions of interest from the WHERE clause**
- **The query parser is able to compute an integrated result set of a multiversion query using basic aggregate functions: SUM, MIN, MAX, AVG**

# Metadata Queries

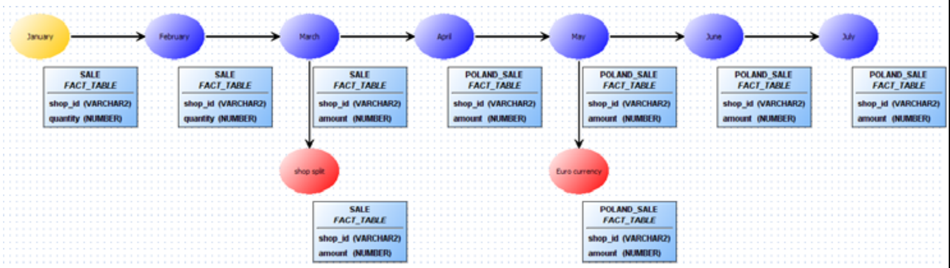
- Querying metadata for the purpose of analyzing the MVDW change history
- Query types
  - **version query** ⇨ a query searching for DW versions that include an indicated schema object or a dimension instance
  - **object evolution query** ⇨ a query retrieving the evolution history of an indicated schema object or a dimension instance



# Version Query

- Example
  - show all DW versions whose schema includes fact table Sale and the structure of Sale in DW version from February includes two attributes: shop_id and quantity

```
show versions having fact table Sale
of structure ( shop_id, quantity )
in version 'February';
```



# Object Evolution Query

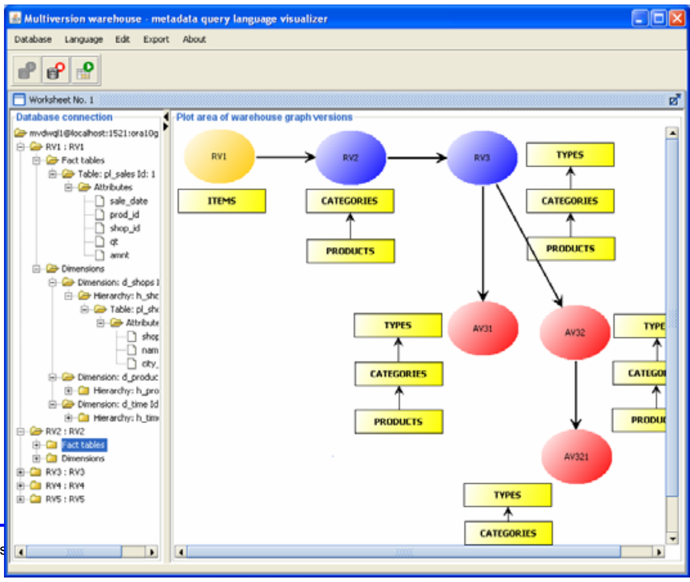
➲ **Example**

- **show the evolution of hierarchy H_Product in dimension Product that originally exists in base version from March**

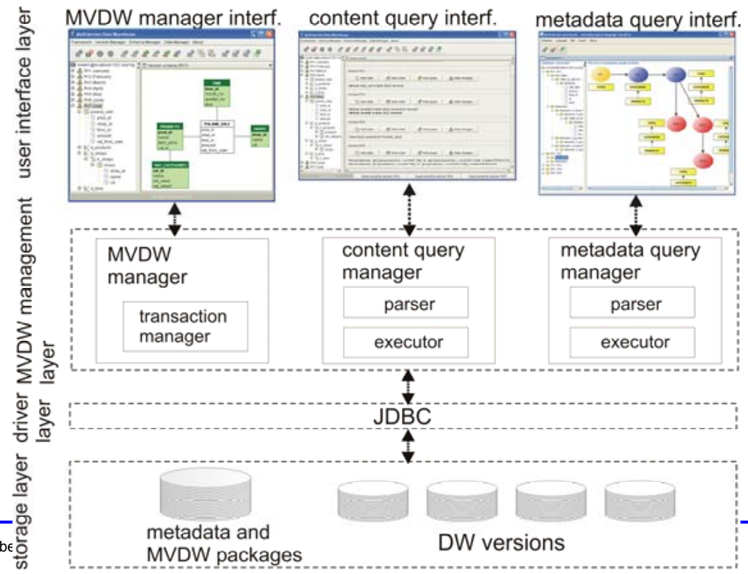
```
show evolution of dimension Product
hierarchy H_Product
in version 'R_March';
```



# Object Evolution Query



# MVDW Prototype






# Sharing Multiversion Data

- Copying vs. sharing
- Data redundancy and data anomalies vs. data access efficiency

# Data Sharing

- **BitmapSharing: information on versions a given record belongs to is represented by bitmaps stored with data**
  - **bitmap - vector of 0 and 1**
  - **one bitmap is allocated for one shared version of DW table**
  - **the number of bits = the number of records in shared table**
  - **bit position corresponds to the position of a record in a table**
  - **1 - record is shared; 0 - record is not shared**

Products($R_1$)

| prod_id | name | price | category |
|---|---|---|---|
| p_1 | baguette | 0,80 | breadstuff |
| p_2 | croissant | 1,10 | breadstuff |
| p_3 | milk 3% | 4,50 | dairy |

bitmap

| $R_2$ |
|---|
| 1 |
| 1 |
| 1 |

bitmap

| $R_3$ |
|---|
| 1 |
| 1 |
| 0 |

$R_1 \rightarrow R_2 \rightarrow R_3$



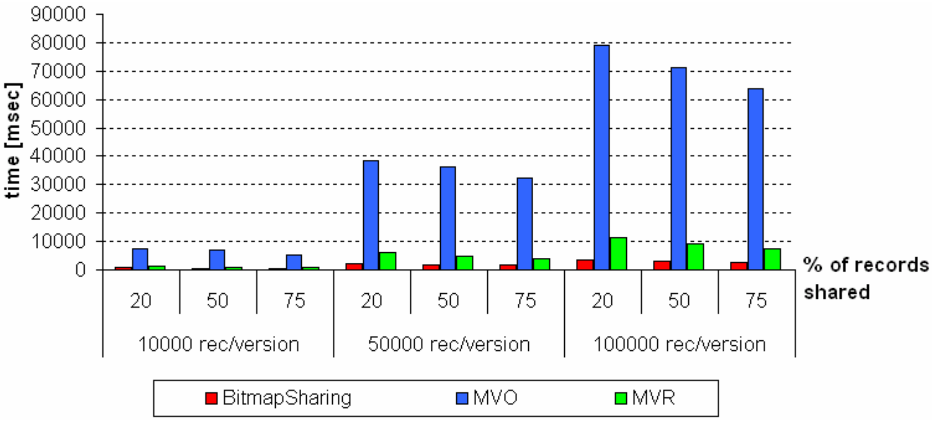
# Data Sharing

- **Alternative data sharing techniques**
  - **few approaches**
  - **two the most advanced include:** [Cellary, Jomier, VLDB1990], [Salzberg et al., EDBT2004]

| | | | |
|---|---|---|---|
| **MVObject** | **OID** | **value** | **$V_1, V_2, V_3, ..., V_n$** |
| **MVRecord** | **PK** | **value** | **$V_{begin}, V_{end}$** |

# Experimental evaluation

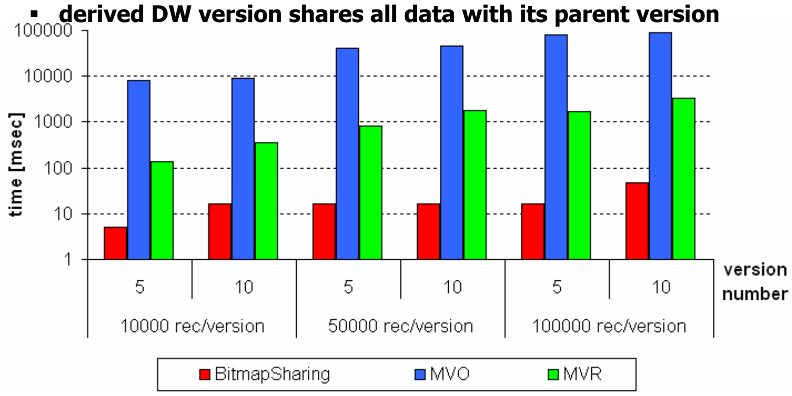
- **Deriving DW versions sharing data**
  - **variable number of shared records**

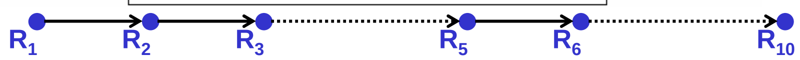





# Experimental evaluation

- **Finding a set of records belonging to a given DW version**
  - **test DW composed of 10 DW versions**
  - **derived DW version shares all data with its parent version**



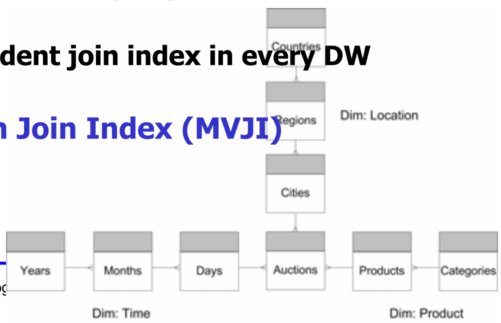
# Still Open Problems

- **ETL**
  - **handling its evolution**
  - **optimizing executions**
- **DWS**
  - **more efficient techniques for**
    - **sharing MV data**
    - **indexing MV data**
  - **more advanced query languages for MVDW**
  - **MVDW design environments**
  - **transaction concepts for MVDW**
  - **integrity constraints for standard DW and MVDW**
  - **applying GPUs for data processing (querying, data compression)**
  - **index structures**



# Query Optimization

- **Star query**
  - **joins a fact table with its dimension tables**
  - **traditional DW: optimized by means of a join index (materialized join)**
- **MVDW ⇨ star queries addressing multiple DW versions (MV star queries)**
  - **optimization more difficult ⇨ star query executed in multiple DW versions**
  - **naive approach ⇨ independent join index in every DW version**
- **Our approach: Multiversion Join Index (MVJI)**

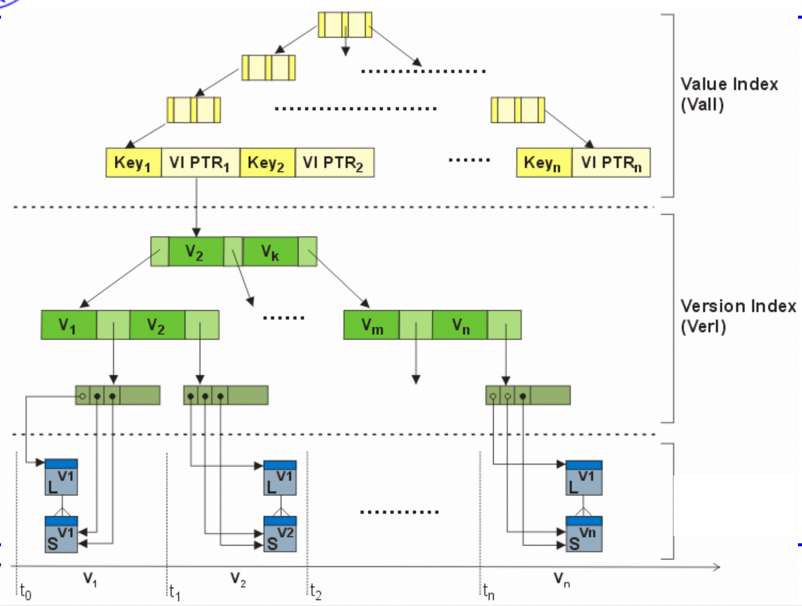


# Related Approaches

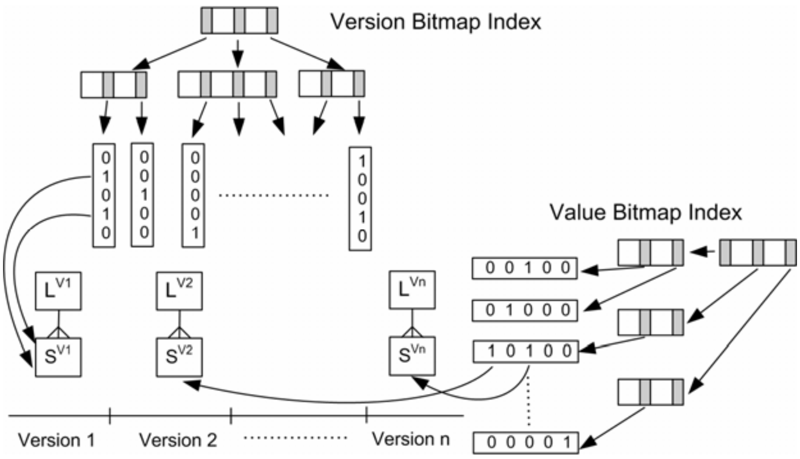
- B-tree based for managing temporal data
  - [Elmasri et al., ICDE91], [Lanka, Mays SIGMOD91], [Becker et al., VLDBJour.96], [Nascimiento, Dunham TKDE99], [Jiang et al., VLDB2000], ...
- Indexing 2-dimensional space (transaction time ⇔ valid time)
  - [Nascimiento, Dunham, SAC96, IDEAS97]
- Indexing 2-dimensional space (transaction time ⇔ value)
  - [Manolopoulos, Kapetanakis JCIT90], [Tzouramanis et al., DKE99]
- Summary
  - **support for storing and searching versions of data that originate from the same table**
  - **not aiming at optimizing queries that join multiple tables**



# MVJI



# Bitmap-based MVJI

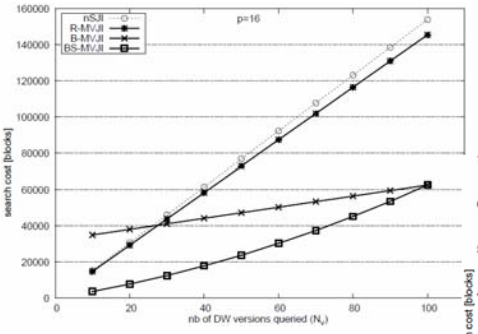


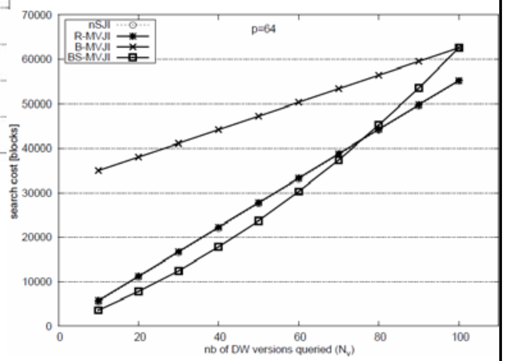



# Experimental evaluation

- Implementation: C++
- Hardware: 8 core Xeon, 16GB RAM
- Software: Linux
- MVDW and data parameters
  - nb DW versions = 100
  - nb rec. in every DW version = 100 000
  - avg data rec. size = 64B
  - pointer size = 32B
  - index data key size = 32B
  - data block size = 4096B
  - data block flling = 75%
  - nb of DW versions accessed: 4 to 100

# Experimental evaluation

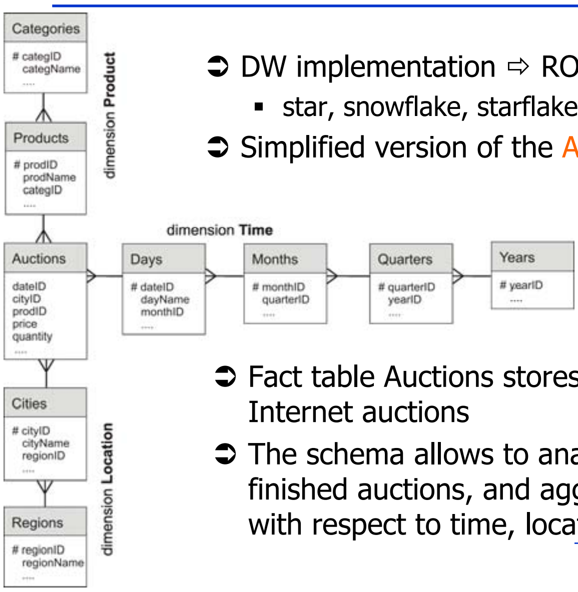




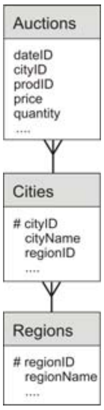

# Indexing Dimensions

- ➲ DW implementation ⇨ ROLAP
  - star, snowflake, starflake schema
- ➲ Simplified version of the Allegro (Allegro.pl) DW




- ➲ Fact table Auctions stores data about finished Internet auctions
- ➲ The schema allows to analyze the number of finished auctions, and aggregate purchase costs with respect to time, locations, and sold products

# Example query




- Compute the sum of sales prices of products, per cities where customers live

```
SELECT Cities.cityName, SUM(Auctions.price)
FROM Auctions, Cities
WHERE Auctions.cityID = Cities.cityID
GROUP BY Cities.cityName;
```

- By rolling up the result of the above query along the hierarchy of dimension **Location**, one can compute the sum of product sales prices per regions

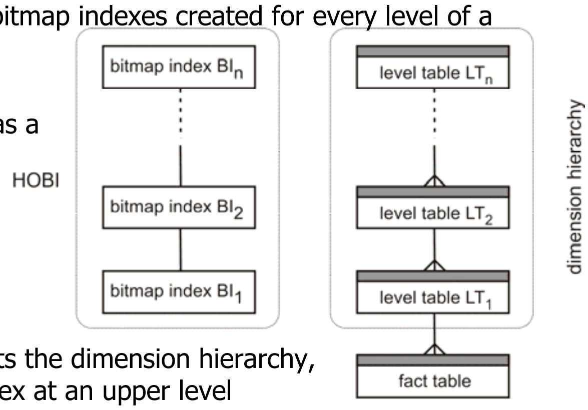
```
SELECT Regions.regionName, SUM(Auctions.price)
FROM Auctions, Cities, Regions
WHERE Auctions.cityID = Cities.cityID
AND Cities.regionID = Regions.regionID
group by Regions.regionName;
```

- Execution: level tables **Cities** and **Regions** need to be joined with fact table **Auctions**
- Optimization: additional data structure at level Regions, pointing to appropriate regional auctions



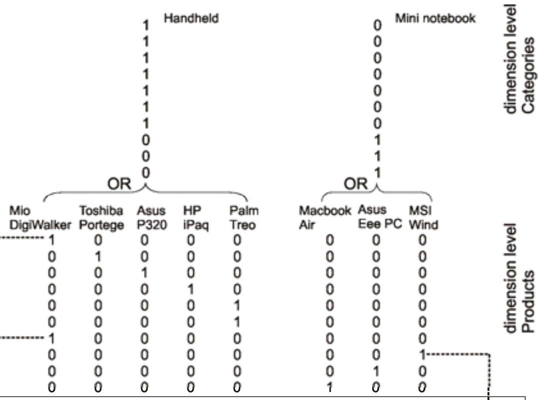
# HOBI: Hierarchically Organized Bitmap Index

- Simple idea (J. Chmiel, T. Morzy, R. Wrembel. HOBI: Hierarchically Organized Bitmap Index for Indexing Dimensional Data (DaWaK, 2009))
  - HOBI is composed of bitmap indexes created for every level of a dimension hierarchy
  - BI are also organized as a hierarchy




  - the BI hierarchy reflects the dimension hierarchy, such that a bitmap index at an upper level aggregates bitmap indexes from a lower level

# HOBI: Example



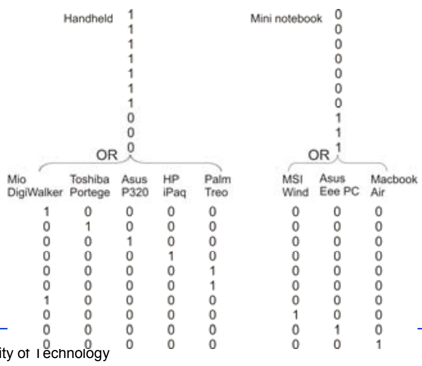

fact table Auctions

| prodID | cityID | dateID | price | quantity | ... |
|---|---|---|---|---|---|
| Mio-DW | POZ | 25-FEB-2009 | 100 | 3 | ... |
| T-Port | WAW | 02-MAR-2009 | 140 | 1 | ... |
| A-P320 | WAW | 02-MAR-2009 | 90 | 2 | ... |
| H-iPaq | WR | 13-MAR-2009 | 110 | 1 | ... |
| P-Treo | WR | 14-APR-2009 | 120 | 4 | ... |
| P-Treo | KRA | 15-APR-2009 | 105 | 2 | ... |
| Mio-DW | KRA | 25-MAY-2009 | 100 | 1 | ... |
| M-Wind | POZ | 12-MAY-2009 | 300 | 1 | ... |
| A-Eee | POZ | 14-JUN-2009 | 320 | 2 | ... |
| M-Air | WR | 17-JUL-2009 | 400 | 1 | ... |



# HOBI for Roll-up Queries

- Query: compute the sum of sales concerning products in category 'Mini notebook'
- Optimization:
  - use bitmap 'Mini notebook' (defined for level **Categories**) for finding appropriate auctions



# Experimental Evaluation

- HOBI vs. Oracle bitmap join index
- HOBI - implemented on top of Oracle10g
- Tested queries
  - range scan
  - roll-up
- Real dataset (from the Allegro Group)
  - 100 000 000 fact rows on finished Internet auctions from April 2007 until March 2008
  - stored in Oracle10g
- Hardware
  - Intel Dual Core 2GHz, 2GB RAM
- 10 runs of one experiment

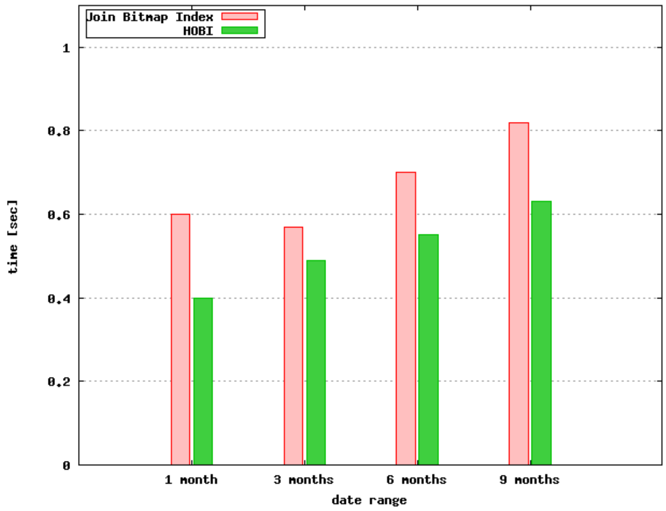


# Range Scan Query

- The query computed the number of auctions in a given time period defined on the Days level
- The bitmap join index was created on attribute Days.d_date
- HOBI was created for the Time dimension
- Time period parameterized: 1, 3, 6, and 9 months

```
SELECT COUNT(1)
FROM auctions, days
WHERE auctions.start_day = days.d_date
AND days.d_date >= date-A AND days.d_date < date-B;
```

## Range Scan Query




$t_{BJI}/t_{HOBI}$=1.50 (1m)
$t_{BJI}/t_{HOBI}$=1.16 (3m)
$t_{BJI}/t_{HOBI}$=1.27 (6m)
$t_{BJI}/t_{HOBI}$=1.48 (9m)

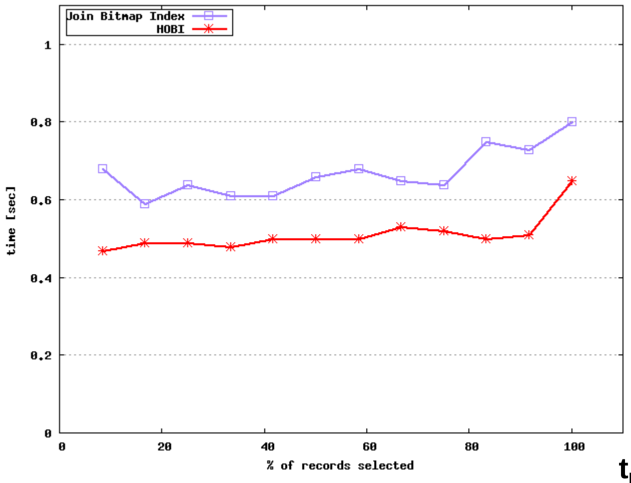


## Roll-up Query

- **The query computed the number of auctions in a given time period defined on the upper level Months**
- **The bitmap join index was created on Days.d_date**
- **HOBI was created for the Time dimension**
- **Time period parameterized: from 10% to 100% of days stored in the database**

```
SELECT COUNT(1)
FROM auctions, days, months
WHERE auctions.start_day = days.d_date
AND days.month = months.id
AND months.month >= date-A AND months.month < date-B;
```

# Roll-up Query




$t_{BJI}/t_{HOBI}$=1.22 (10-20%)
$t_{BJI}/t_{HOBI}$=1.50 (75-85%)



# HOBI: Summary

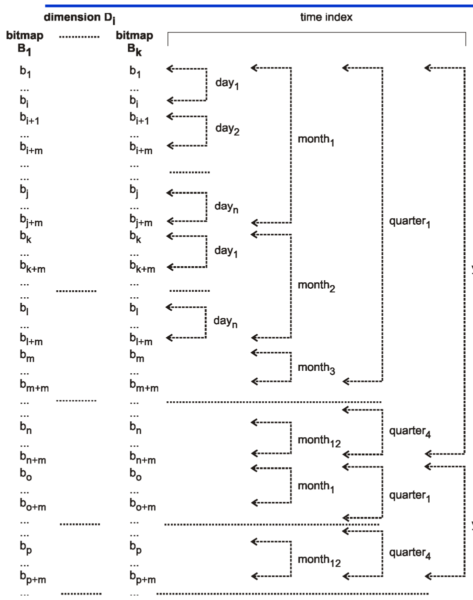
- A simple idea of using bitmap indexes organized hierarchically along a dimension
- Experiments run on a real data set
- Reducing query processing time
  - range queries: max 30%
  - roll-up queries: max 30%
- HOBI: built-in on top of Oracle10g ⇨ additional processing overhead

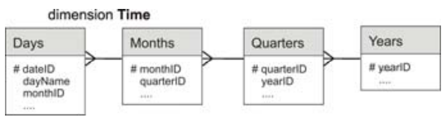
# Motivation for Time-HOBI

- Time dimension is used in most of star queries
- In order to eliminate the frequent join operation of a fact table and the Time dimension we propose to implicitly encode the Time dimension in other dimensions ⇨ **Time-HOBI**
- Assumption: data stored in a fact table are sorted by time
  - G. Moerkotte. Small materialized aggregates: A light weight index structure for data warehousing (VLDB, 1998)
  - G. Graefe. Fast loads and fast queries (DaWaK, 2009)
  - DW is loaded incrementally in time intervals
  - data can be easily sorted by time in ETL layer
- **Time−HOBI** combines HOBI with time index
- **Time index** (TI) is created on bitmaps
  - it stores ranges of bit numbers belonging to a given time interval
  - time intervals compose the same hierarchy as defined in the Time dimension



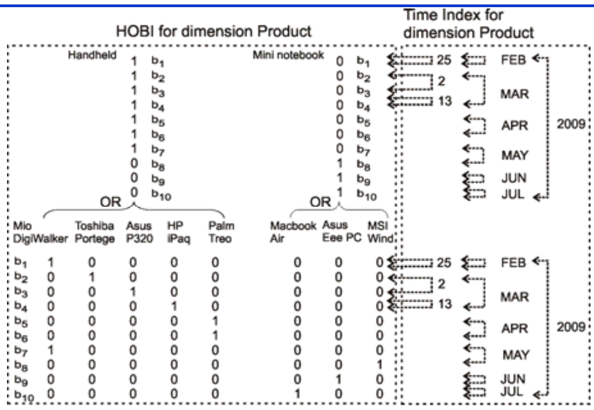
# Time-HOBI: concept




- all HOBIs created for all dimensions in a given DW schema point to the same number of rows in a fact table ⇨ only one time index is required to index all dimensions

# Time-HOBI: example




| prodID | cityID | dateID | price | quantity | ... |
|---|---|---|---|---|---|
| Mio-DW | POZ | 25-FEB-2009 | 100 | 3 | ... |
| T-Port | WAW | 02-MAR-2009 | 140 | 1 | ... |
| A-P320 | WAW | 02-MAR-2009 | 90 | 2 | ... |
| H-iPaq | WR | 13-MAR-2009 | 110 | 1 | ... |
| P-Treo | WR | 14-APR-2009 | 120 | 4 | ... |
| P-Treo | KRA | 15-APR-2009 | 105 | 2 | ... |
| Mio-DW | KRA | 25-MAY-2009 | 100 | 1 | ... |
| M-Wind | POZ | 12-MAY-2009 | 300 | 1 | ... |
| A-Eee | POZ | 14-JUN-2009 | 320 | 2 | ... |
| M-Air | WR | 17-JUL-2009 | 400 | 1 | ... |

fact table Auctions

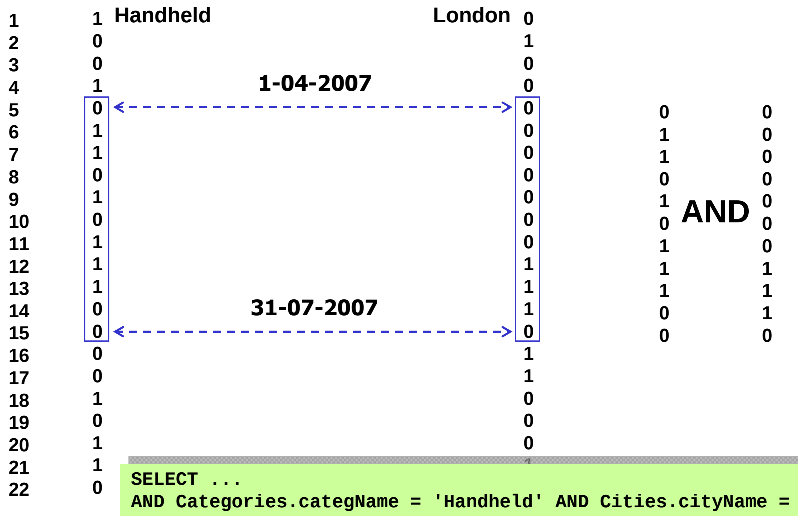


# Time-HOBI for Queries

- Star queries with selection predicates on time ⇨ no need to join fact table with Time dimension
- Count the number of auctions that started between April 1st and July 31st, 2007, of products from category 'Handheld' and customers from London

```
SELECT COUNT(1)
FROM Auctions, Products, Categories, Days
WHERE Auctions.prodID = Products.prodID
AND Products.categID = Categories.categID
AND Auctions.dateID = Days.dateID
AND Categories.categName = 'Handheld'
AND Cities.cityName = 'London'
AND Days.dateID
     BETWEEN to_date('1-04-2007','dd-mm-yyyy')
     AND to_date('31-07-2007','dd-mm-yyyy');
```

# Time-HOBI for Queries




```
SELECT ...
AND Categories.categName = 'Handheld' AND Cities.cityName = 'London'
AND Days.dateID
    BETWEEN to_date('1-04-2007','dd-mm-yyyy')
    AND to_date('31-07-2007','dd-mm-yyyy');
```



# Implementation

- C and Python
- Data and indexes stored in flat files
- Focus on the performance of filtering queries with range predicates defined on time
- Real dataset acquired from the **Allegro** Group (Allegro.pl)
  - 10 000 000 of fact rows describing finished auctions in a simplified DW schema
- PC (Intel Dual Core 2GHz, 2GB RAM) under Ubuntu Linux 8.04
- Tested performance characteristics of indexes only ⇨ computing the final bitmaps ⇨ COUNT
  - traditional bitmap index
  - HOBI
  - Time-HOBI
- Index block size: 2048B, 4096B, 8192B
- The same query was executed 10 times

# Experiments

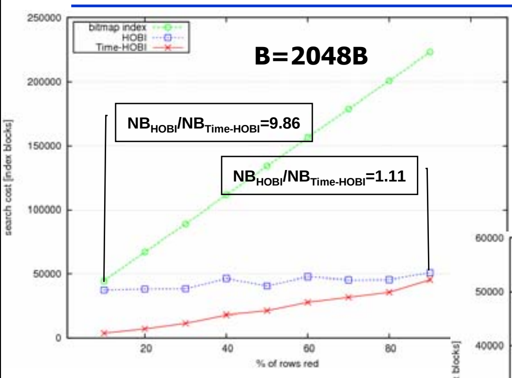
- Query: compute the number of auctions in a given time period defined on the Days level

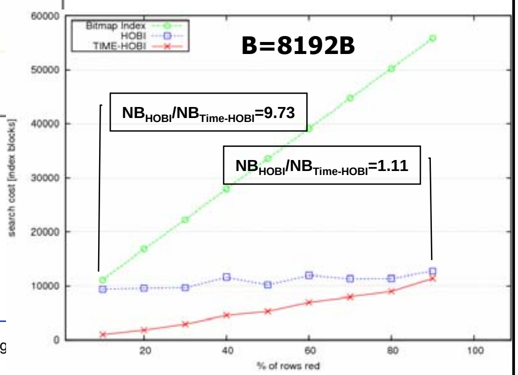
```
SELECT COUNT(1)
FROM Auctions, Products, Days
WHERE Auctions.prodID = Products.prodID
AND Auctions.dateID = Days.dateID
AND Days.dateID >= 'date-begin'
AND Days.dateID <= 'date-end';
```

- Time−HOBI created for the Product dimension
- HOBI created for the Product and the Time dimensions
- Bitmap indexes created for all the foreign keys in fact table Auctions
- Parameterized time period ⇨ the number of records fulfilling the selection criteria ranged from 10% to 90%



# Results



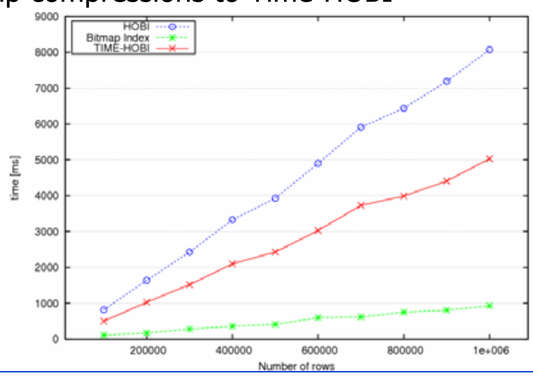
# Time-HOBI: Summary

- The TIME dimension is used in most of star queries ⇨ eliminating the TIME dimension from star queries reduces the number of joins
- A simple idea ⇨ Time-HOBI:
  - using bitmap indexes organized hierarchically along a dimension
  - implicitly including time hierarchy into the indexes
- Reducing query processing time (on a real data set)
  - from 1.1 to 9.86 times as compared to HOBI
- Time-HOBI applicable to dimension other than TIME provided that the dimension is used for ordering fact rows and is frequently used in star queries



# Future and Ongoing Work

- Implementation in PostgreSQL (ongoing)
- Index update algorithm (ongoing)
- Running experiments on a synthetic data set (TPC-H)
- Applying bitmap compressions to Time-HOBI



# Sequential OLAP

- Origin: analyzing passengers traffic
- An itinerary results in a sequence of events
  - get into a bus [TS1] ⇨ pay ⇨ get off [TS2]
  - get another bus [TS3] ⇨ pay ⇨ get off [TS4]
  - get a subway [TS5] ⇨ pay ⇨ get off [TS6]
- Queries
  - what is an average duration time/length of an itinerary?
  - what is the most popular line?
  - what is a rush hour?
  - ...



# Sequential OLAP

```
SELECT              COUNT(*)
FROM                Event
WHERE               time >= 2007-10-01T00:00 AND
                    time < 2007-12-31T24:00
CLUSTER BY          card-id AT individual,
                    time AT day
SEQUENCE BY         time ASCENDING
SEQUENCE GROUP BY   card-id AT fare-group,
                    time AT day
CUBOID BY           SUBSTRING (X, Y, Y, X) WITH
                    X AS location AT station,
                    Y AS location AT station
                    LEFT-MAXIMALITY (x1, y1, y2, x2) WITH
                    x1.action = "in" AND
                    y1.action = "out" AND
                    y2.action = "in" AND
                    x2.action = "out"
```

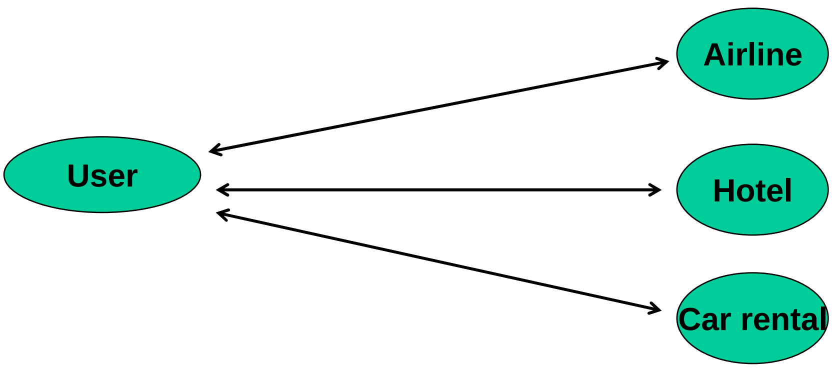
- How to aggregate sequences?
- How to join sequences?

# Sequential OLAP

- **Problems**
  - SOLAP data model for storing sequences
  - Query language for analyzing data
  - Data structures (indexes, materialized views, ...)



# WEB Services




- **Lack of transaction concept, standards**
- **Managing distributed transactions**

# GPU Processing

- Query processing on GPUs
- Processing compressed data structures without a need to decompress them
- Porting compression algorithms to the GPU platform