Introduction and Database Technology

By EM Bakker

Databases and Data Mining

Databases (Chapters 1-7):
- The Evolution of Database Technology
- Data Preprocessing
- Data Warehouse (OLAP) & Data Cubes
- Data Cubes Computation
- Grand Challenges and State of the Art

Evolution of Database Technology

- 1960s:
  - (Electronic) Data collection, database creation, IMS (hierarchical database system by IBM) and network DBMS

- 1970s:
  - Relational data model, relational DBMS implementation

- 1980s:
  - RDBMS, advanced data models (extended-relational, OO, deductive, etc.)
  - Application-oriented DBMS (spatial, scientific, engineering, etc.)

Evolution of Database Technology

- 1990s:
  - Data mining, data warehousing, multimedia databases, and Web databases

- 2000 - 2014
  - Stream data management and mining
  - Data mining and its applications
  - Web technology
    - Data integration, XML
    - Social Networks (Facebook, etc.)
    - Cloud Computing
  - Global information systems
  - Emerging in-house solutions
  - In Memory Databases
  - Big Data

Back to The Future of the Past

  - Note: J.N. Gray received Turing Award 1998


Industry Profile (1994)

- The database industry $7 billion in revenue in 1994, growing at 35% per year.
- Leading corporations are US-based:
  - IBM, Oracle, Sybase, Informix, Computer Associates, and Microsoft
- Specialty vendors:
  - Tandem: fault-tolerant transaction processing systems; AT&T-Teradata: data mining systems
- Small companies for application-specific databases:
  - text retrieval, spatial and geographical data, scientific data, image data, etc.
- Emerging group of companies: object-oriented databases.
- Desktop databases: an important market focused on extreme ease-of-use, small size, and disconnected operation.

Worldwide Vendor Revenue Estimates from RDBMS Software, Based on Total Software Revenue, 2006 (Millions of Dollars)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>7,168.0</td>
<td>47.1</td>
<td>6,238.2</td>
<td>46.8</td>
<td>14.9</td>
</tr>
<tr>
<td>IBM</td>
<td>3,204.1</td>
<td>21.1</td>
<td>2,945.7</td>
<td>22.1</td>
<td>8.8</td>
</tr>
<tr>
<td>Microsoft</td>
<td>2,654.4</td>
<td>17.4</td>
<td>2,073.2</td>
<td>15.6</td>
<td>28.0</td>
</tr>
<tr>
<td>Teradata</td>
<td>494.2</td>
<td>3.2</td>
<td>467.6</td>
<td>3.5</td>
<td>5.7</td>
</tr>
<tr>
<td>Sybase</td>
<td>486.7</td>
<td>3.2</td>
<td>449.9</td>
<td>3.4</td>
<td>8.2</td>
</tr>
<tr>
<td>Other Vendors</td>
<td>1,206.3</td>
<td>7.9</td>
<td>1,149.0</td>
<td>8.6</td>
<td>5.0</td>
</tr>
<tr>
<td>Total</td>
<td>15,213.7</td>
<td>100.0</td>
<td>13,323.5</td>
<td>100.0</td>
<td>14.2</td>
</tr>
</tbody>
</table>

Source: Gartner Dataquest (June 2007)

Note: Big Data revenues in 2012 (WikiBon): ~11,500

Historical Perspective

36 years of Database Research

Period 1960 - 1996

Historical Perspective (1960-)

- Companies began automating their back-office bookkeeping in the 1960s
- COBOL and its record-oriented file model were the work-horses of this effort
- Typical work-cycle:
  1. a batch of transactions was applied to the old tape-master
  2. a new-tape-master produced
  3. printout for the next business day.
- COmmon Business-Oriented Language (COBOL 2002 standard)

COBOL

A quote by Prof. dr. E.W. Dijkstra (Turing Award 1972) 18 June 1975:

“The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence.”

But: In 2014 still vacancies:

**COBOL Code (just an example!)**

01 LOAN-WORK-AREA.
  02 LOAN-LOAN-ERROR-FLAG PIC X(2).
  02 LOAN-AMT PIC 9(06)V9(02) COMP.
  02 LOAN-INT-RATE PIC 9(02)V9(02) COMP.
  02 LOAN-NBR-PMTS PIC 9(03) COMP.
  02 LOAN-PMT PIC 9(06)V9(02) COMP.
  02 LOAN-INT-PMT PIC 9(01)V9(12) COMP.
  02 LOAN-TOTAL-PMTS PIC 9(06)V9(02) COMP.
  * 02 LOAN-TOTAL-INT PIC 9(06)V9(02) COMP.

040000 COMPUTE PAYMENT.
  MOVE 0 TO LOAN-LOAN-ERROR-FLAG.
  CMP (LOAN-LOAN-ERROR-FLAG) (LOAN-INT-RATE ZERO)
  OR (LOAN-LOAN-ERROR-FLAG) (LOAN-NBR-PMTS ZERO)
  OR (LOAN-LOAN-ERROR-FLAG)

COMPUTE LOAN-INT-PMT = LOAN-INT-RATE / 1200
  OR (LOAN-LOAN-ERROR-FLAG)
  MOVE 1 TO LOAN-LOAN-ERROR-FLAG
  GO TO 040000 EXT.

Note: Big Data revenues in 2012 (WikiBon): ~11,500
### Historical Perspective (1970's)

- **Online Databases:** Transition from handling transactions in daily batches to systems that managed an *on-line database* that captures transactions as they happened.
- At first these systems were *ad hoc*
- Late in the 60’s, “network” and “hierarchical” database products emerged.
- A network data model standard was defined by the database task group (DBTG), which formed the basis for most commercial systems during the 1970’s.
- In 1980 DBTG-based Cullinet was the leading software company.

### Network Model

- **Hierarchical model:** A tree of records, with each record having one parent record and many children.

### Historical Perspective

**IBM’s DBTG problems:**

- DBTG used a procedural language that was
  - low-level
  - record-at-a-time
- The programmer had to navigate through the database, following pointers from record to record
- If the database was redesigned, then all the old programs had to be rewritten

### The "relational" data model

**The "relational" data model success**

- Both industry and university research communities embraced the *relational data model* and extended it during the 1970s.
- It was shown that a high-level relational database query language could give performance comparable to the best record-oriented database systems. (!)
- This research produced a generation of systems and people that formed the basis for IBM’s DB2, Ingres, Sybase, Oracle, Informix and others.

### The "relational" data model success

**SQL**

- The SQL relational database language was standardized between 1982 and 1986.
- By 1990, virtually all database systems provided an SQL interface (including network, hierarchical and object-oriented database systems).
Ingres at UC Berkeley in 1972

Ingres at UC Berkeley in 1972 (Stonebraker, Rowe, Wong, and others) resulted in:

- A relational database system
- Query language (QUEL)
- Relational optimization techniques
- Storage strategies
- Work on distributed databases

Further work on:
- Database inference
- Active databases (automatic responding)
- Extensible databases

Ingres from Computer Associates and PostgreSQL

IBM: System R

Codd's relational model was very controversial:

- Too simplistic
- Could never give good performance.

- A 10-person IBM Research effort to prototype a relational system => a prototype, System R (evolved into the DB2 product)

Defined the fundamentals on:
- Query optimization
- Data independence (views)
- Transactions (logging and locking)
- Security (the grant-revoke model)

Note: SQL from System R became more or less the standard.

The System R group further research:
- Distributed databases (R*)
- Object-oriented extensible databases (Starburst).

The Future of 1997 (Gray)

Conclusions

- Database systems a key aspect of Computer Science & Engineering.
- Representing knowledge is one of the central challenges

- Representing and indexing data,
- Adding inference to data search: inductive reasoning
- Compiling queries more efficiently, parallel execution
- Integrating data from heterogeneous data sources,
- Analyzing performance
- Extending the transaction model to handle long transactions (transactions that involve humans).
- Very-large-scale (tertiary) storage
- Unifying object-oriented concepts with the relational model.
- New datatypes (image, document, drawing)
- Procedures to get active databases, data inference, and data encapsulation

The Future of 1996

Database Research: Achievements and Opportunities into the 21st Century.

Silberschatz, M. Stonebraker, J. Ullman Eds.
SIGMOD Record, Vol. 25, No. 1
pp. 52-63
March 1996

New Database Applications (1996)

- EOSDIS (Earth Observing System Data and Information System)
- Electronic Commerce
- Health-Care Information Systems
- Digital Publishing
- Collaborative Design

EOSDIS (Earth Observing System Data and Information System)

Challenges:
- On-line access to petabyte-sized databases and managing tertiary storage effectively.
- Supporting thousands of consumers with very heavy volume of information requests, including ad-hoc requests and standing orders for daily updates.
- Providing effective mechanisms for browsing and searching for the desired data,
**Electronic Commerce**

Heterogeneous information sources must be integrated. For example, something called a "connector" in one catalog may not be a "connector" in a different catalog.

- "Schema integration" is a well-known and extremely difficult problem.

Electronic commerce needs:
- Reliable
- Distributed
- Authentication
- Funds transfer.

**Health-Care Information Systems**

Transforming the health-care industry to take advantage of what is now possible will have a major impact on costs, and possibly on quality and ubiquity of care as well.

Problems to be solved:
- Integration of heterogeneous forms of legacy information.
- Access control to preserve the confidentiality of medical records.
- Interfaces to information that are appropriate for use by all health-care professionals.

**Digital Publishing**

- Management and delivery of extremely large bodies of data at very high rates. Typical data consists of very large objects in the megabyte to gigabyte range (1996).
- Delivery with real-time constraints.
- Protection of intellectual property, including cost-effective collection of small payments and inhibitions against reselling of information.
- Organization of and access to overwhelming amounts of information.

**The Information Superhighway**

Databases and database technology will play a critical role in this information explosion. Already Webmasters (administrators of World-Wide-Web sites) are realizing that they are database administrators...

**Support for Multimedia Objects (1996)**

- Tertiary Storage (for petabyte storage)
- New Data Types
- New Data Types
- Quality of Service
- Multi-resolution Queries
- User Interface Support

**New Research Directions (1996)**

- Problems associated with putting multimedia objects into DBMSs.
- Problems involving new paradigms for distribution of information.
- New uses of databases
- Data Mining
- Data Warehouses
- Repositories
- New transaction models
- Workflow Management
- Alternative Transaction Models
- Problems involving ease of use and management of databases.
Conclusions of the Forum (1996)

The database research community

- has a foundational role in creating the technological infrastructure from which database advancements evolve.
- New research mandate because of the explosions in hardware capability, hardware capacity, and communication (including the internet or "web" and mobile communication).
- Explosion of digitized information require the solution to significant new research problems:
  - support for multimedia objects and new data types
  - distribution of information
  - new database applications
  - workflow and transaction management
  - ease of database management and use

“One Size Fits All”: An Idea Whose Time Has Come and Gone.

M. Stonebraker, U. Cetintemel

Proceedings of The 2005 International Conference on Data Engineering

April 2005
http://www.cs.brown.edu/~ugur/fits_all.pdf

DBMS: “One size fits all.”

Single code line with all DBMS Services solves:

- Cost problem: maintenance costs of a single code line
- Compatibility problem: all applications will run against the single code line
- Sales problem: easier to sell a single code line solution to a customer
- Marketing problem: single code line has an easier market positioning than multiple code line products

DBMS: “One size fits all.”

To avoid these problems, all the major DBMS vendors have followed the adage “put all wood behind one arrowhead”.

In this paper it is argued that this strategy has failed already, and will fail more dramatically off into the future.
Early 1990's:
- gather together data from multiple operational databases into a data warehouse for business intelligence purposes.
- Typically 50 or so operational systems, each with an on-line user community who expect fast response time.
- System administrators were (and still are) reluctant to allow business-intelligence users onto the same systems, fearing that the complex ad-hoc queries from these users will degrade response time for the on-line community.
- In addition, business-intelligence users often want to see historical trends, as well as correlate data from multiple operational databases. These features are very different from those required by on-line users.

Data warehouses are very different from Online Transaction Processing (OLTP) systems:
- OLTP systems:
  - the main business activity is typically to sell a good or service
  - \( \Rightarrow \) optimized for updates
- Data warehouse:
  - ad-hoc queries, which are often quite complex.
  - periodic load of new data interspersed with ad-hoc query activity

The standard wisdom in data warehouse schemas is to create a fact table:
"who, what, when, where" about each operational transaction.

Data warehouse applications run much better using bit-map indexes
- OLTP (Online Transaction Processing) applications prefer B-tree indexes.
- materialized views are a useful optimization tactic in data warehousing, but not in OLTP worlds.

As a first approximation, most vendors have a
- warehouse DBMS (bit-map indexes, materialized views, star schemas and optimizer tactics for star schema queries) and
- OLTP DBMS (B-tree indexes and a standard cost-based optimizer), which are united by a common parser

Some other examples that show:
Why conventional DBDMs will not perform on the current emerging applications.
Emerging Sensor Based Applications

- Sensoring Army Battalion of 30,000 humans and 12,000 vehicles => $10^6$ sensors
- Monitoring Traffic (InfraWatch, 2010)
- Amusements Park Tags
- Health Care
- Library books
- Etc.

Conventional DBMSs will not perform well on this new class of monitoring applications.

For example: Linear Road, traditional solutions are nearly an order of magnitude slower than a special purpose stream processing engine.

Example: Financial-feed processing

Financial institutions subscribe to feeds that deliver real-time data on market activity, specifically:
- News
- Consummated trades
- Bids and asks
- Etc.

For example:
- Reuters
- Bloomberg
- Infodyne

Example: An existing application: financial-feed processing

Financial institutions have a variety of applications that process such feeds. These include systems that:
- Produce real-time business analytics,
- Perform electronic trading (2014: High Frequency Trading)
- Ensure legal compliance of all trades to the various company and SEC rules
- Compute real-time risk and market exposure to fluctuations in foreign exchange rates.

The technology used to implement this class of applications is invariably "roll your own", because no good off-the-shelf system software products exist.

Example: Detect Problems in Streaming stock ticks:

- Specifically, there are 4,500 securities, 500 of which are "fast moving".

Defined by rules:

- A stock tick on one of the fast securities is late if it occurs more than 5 seconds after the previous tick from the same security.

- The other 4,000 symbols are slow moving, and a tick is late if 60 seconds have elapsed since the previous tick.
Performance

- Implemented in the StreamBase stream processing engine (SPE) [5], a commercial, industrial-strength version of Aurora [8, 13].
- On a 2.8Ghz Pentium processor with 512 Mbytes of memory and a single SCSI disk, the workflow in the previous figure can be executed at 160,000 messages per second, before CPU saturation is observed.
- In contrast, StreamBase engineers could only get 900 messages per second using a popular commercial relational DBMS.

Why?: Outbound vs Inbound Processing

StreamBase (Inbound Processing)

Outbound vs Inbound Processing

- DBMSs are optimized for outbound processing
- Stream processing engines are optimized for inbound processing.
- Although it seems conceivable to construct an engine that is optimized for both inbound and outbound processing, such an engine design is clearly a research project.

Other Issues: Correct Primitives for Streams

- SQL systems contain a sophisticated aggregation system, for example a statistical computation over groupings of the records from a table in a database. When processing the last record in the table the aggregate calculation for each group of records is emitted.
- However, streams can continue forever, there is no notion of “end of table”. Consequently, stream processing engines extend SQL with the notion of time windows.
- In StreamBase, windows can be defined based on clock time, number of messages, or breakpoints in some other attribute.

Other Issues: Integration of DBMS Processing and Application Logic (1/2)

- Relational DBMSs were all designed to have client-server architectures.
- In this model, there are many client applications, which can be written by arbitrary people, and which are therefore typically untrusted.
- Hence, for security and reliability reasons, these client applications are run in a separate address space from the DBMS.
Other Issues: Integration of DBMS Processing and Application Logic (2/2)

- In an embedded processing model, it is reasonable to freely mix
  - application logic
  - control logic
  - DBMS logic

  This is what StreamBase does.

Other Issues: High Availability

- It is a requirement of many stream-based applications to have high availability (HA) and stay up 7x24.
- Standard DBMS logging and crash recovery mechanisms are ill-suited for the streaming world
- The obvious alternative to achieve high availability is to use techniques that rely on Tandem-style process pairs
- Unlike traditional data-processing applications that require precise recovery for correctness, many stream-processing applications can tolerate and benefit from weaker notions of recovery.

Other Issues: Synchronization

- Traditional DBMSs use ACID transactions between concurrent transactions submitted by multiple users for example to induce isolation. (heavy weight)
- In streaming systems, which are not multi-user, a concept like isolation can be simply achieved by: critical sections, which can be implemented through light-weight semaphores.

  ACID = Atomicity, Consistency, Isolation (transactions are executed in isolation), Durability

One Size Fits All?

Conclusions

- Data warehouses: store data by column rather than by row; read oriented
- Sensor networks: flexible light-way database abstractions, as TinyDB; data movement vs data storage
- Text Search: standard RDBMS too heavy weight and inflexible
- Scientific Databases: multi dimensional indexing, application specific aggregation techniques
- XML: how to store and manipulate XML data

The Fourth Paradigm

  eScience
  and the
  www.fourthparadigm.org (2009)
Four Science Paradigms (J.Gray, 2007)

- Thousand years ago: science was empirical describing natural phenomena
- Last few hundred years: theoretical branch using models, generalizations
- Last few decades: a computational branch simulating complex phenomena
- Today: data exploration (eScience) unify theory, experiment, and simulation

- Data captured by instruments
- Or generated by simulator
- Processed by software
- Information/Knowledge stored in computer
- Scientist analyzes database / files using data management and statistics

The eScience Challenge

Novel Tools needed for:
- Data Capturing
- Data Curation
- Data Analysis
- Data Communication and Publication Infrastructure

Gray’s Laws

Database-centric Computing in Science

How to approach data engineering challenges related to large scale scientific datasets:

- Scientific computing is becoming increasingly data intensive.
- The solution is in a "scale-out" architecture.
- Bring computations to the data, rather than data to the computations.
- Start the design with the "20 queries."
- Go from "working to working."


VLDB 2010


VLDB 2011

Keynotes
- T. O’Reilly, Towards a Global Brain.
- D. Campbell, Is it still “Big Data” if it fits in my pocket?,

Novel Subjects
- Social Networks,
- MapReduce (Hadoop) , Crowdsourcing, and Mining
- Information Integration and Information Retrieval
- Schema Mapping, Data Exchange, Disambiguation of Named Entities
- GPU Based Architecture and Column-store indexing
VLDB 2012 Keynotes

- Querying the Spatial Web
  - billions of queries/week
  - Web objects that are near the location where query was issued
  - New challenges on:
    - Spatial web data management
    - Relevance ranking based on text and location
    - Low latency

- Data Science for Smart Systems
  - Integration of information and control in complex systems
  - Smart bridges, transportation systems, health care systems, supply chains, etc.
  - Data characteristics: heterogeneous, volatile, uncertain
  - New data management techniques
  - New data analytics

VLDB2012 Subjects

- Spatial Queries
- Map Reduce
- Big Data
- Cloud Databases
- Crowdsourcing
- Social Networks and Mobility in the Cloud
- eHealth
- Web databases
- Mobility
- Data Semantics and Data Mining
- Parallel and Distributed Databases
- Graphs
- String and Sequence Processing
- Privacy
- Probabilistic Databases
- Data Flow; Hardware; Indexing; Query Optimization; Streams;

VLDB2012 Crowd Sourcing

[1] CDAS: A Crowdsourcing Data Analytics System
Xuan Liu et al.

Let people execute small tasks such as labeling and tagging images, lookup of phone numbers, compare items for joining and sorting

Example: Amazon's Mechanical Turk (MTurk)
Dataset processing with humans: MTurk programmatic interface for redesigning the workflow

VLDB 2012 10 Year Best Paper Award

Approximate Frequency Counts over Data Streams
By G. Singh Manku (Google Inc. USA), R. Motwani

- Data Stream Algorithms research started late 90s.
- Sensor networks
- Stock data
- Security monitoring
- Recently: personal data stream analysis

VLDB 2012 Spatial Queries

T. Lappas et al.

On the Spatiotemporal Burstiness of Terms.

- Burst identification
- Spatial, temporal
- Unusual high frequency of document streams observed in a short time frame queried from a spatially localized area
- Ranked lists of influential documents with spatiotemporal impact.

VLDB2012 Crowd Sourcing

[1] CDAS: A Crowdsourcing Data Analytics System
Xuan Liu et al.

Image taken from [1].
Google: estimated to have contributed 54 billion $ to the US economy in 2009
What is Big Data?
Is size the only thing that matters?
Challenges
  - Data acquisition: how to filter and compress without leaving out important stuff?
  - Automatic meta data generation, important for downstream analysis
  - Multiple and Changing Analysis Pipelines

Facebook

- Initially: Hive on Hadoop for data analysis
  - Designed for batch processing
  - SQL-like capabilities
  - Datavwarehousing
  - Relying on MapReduce
  - Could take hours on some queries
- Now: Presto
  - Query engine
  - Runs in memory
  - Much faster
  - Meecs to minutes on the 250PB data warehouse

VLDB 2013 Keynotes

- Web-scale data infrastructure (J. Parikh, Facebook):
  - 250 PB, 600TB/day
  - processing 10PB/day
  - technical and non-technical users
  - Scaling (Have a look at the video!)
- DataHub (S. Madden, MIT)
  - Hosting large scale data-analytics
  - data sharing, processing and visualization
- Tools:
  - Data loading and cleaning
  - Scalable, parallel SQL based analytics
  - Interactive visualization
- Privacy-Preserving Data-Analysis (S. Dwork, Microsoft Research)

VLDB 2014

epiC: an extensible and scalable system for processing Big Data, (D. Jiang et al. VLDB2014)

The Big Data challenge 3V features:
1. Volume: huge amounts of data
2. Velocity: the data ingestion rate is very high
3. Variety: data is mixed in structured, semi-structured and unstructured data

State of the art solutions are based on MapReduce (open source: Hadoop):
- The 3rd V is problematic: inconvenient and inefficient for structured data and graph data.
epiC

- An extensible system: epiC extensions for multi-structured datasets
- Specification of parallel computations: Actor-like concurrent programming model independent of data processing models
- Specialized data processing models for the different data types
- Automatic parallelization as in Hadoop
- Also run-time fault tolerance and inter machine communications as in Hadoop

Mapreduce

MapReduce is a programming model for processing and generating large data sets (Hadoop open source implementation).

User specify:
- A map function that processes a key/value pair to generate a set of intermediate key/value pairs.
- A reduce function that merges all intermediate values associated with the same intermediate key.

Mapreduce Programs are automatically parallelized and executed on a large cluster

The run-time system handles:
- The partitioning of the input data
- The program's execution scheduling
- Machine failures
- Inter-machine communication

=> No experience with parallel and distributed systems needed to utilize the resources of a large distributed system.

Optimizing Database Applications

DBMS implement ACID properties
- Atomicity
  - A database transaction is atomic.
- Consistency
  - A database is always in a consistent state.
- Isolation
  - Concurrent transactions are isolated.
- Durability
  - Committed data is stored resilient to system crashes etc.
- A DBMS translates SQL queries in an optimized execution plan (query optimization)

Optimizing Database Applications

Integration of DBS Query processing into an imperative programming language

- Impedance mismatch (See Cook et al. 2005)
  - Procedural types vs database types
  - Procedural program optimization vs query optimization
  - Concurrency vs transactions, etc.
- Tools focus on DB-API that allows mapping between value types and database types.

Optimizing compilers for query optimization
Optimizing Database Applications: previous work

LINQ (J. Dean et al., 2004)
- Extension of programming language for expression of declarative queries.
- Execution of queries hidden: can be on main memory, on a DBMS, etc.
- A query parse tree is executed at run-time, i.e., queries can be sent to DBMS at run time.
- Limited compile time optimization.

Also systems exist for C++ and others, all more focused on ease of programming, efficiency alerts, security (against injection), correctness, etc.

September 16, 2014  Databases and Data Mining  85

Optimizing Database Applications: previous work

- Previous examples: Database API integration in a programming language
- Challenge: impedance mismatch in optimization
- Database Programming Languages
  - Iteration over sets
  - DBPL compile time optimizations
  - Optimization of iterations over sets corresponding to joins
  - Parallelization of loops
- Tycoon
  - Functional language
  - Function calls have as last arguments another function call (Continuation)

September 16, 2014  Databases and Data Mining  86

Optimizing Database Applications: previous work

Holistic approaches
- Application code using databases (for example web applications)
- Transformations that optimize both at the same time
- Dbridge: JAVA code bases (M. Chavan et al. 2011)
  - Optimizing JAVA code
  - Query Optimizing
  - Integrated optimization
- StatusQuo (A. Cheung et al. 2013)
  - Applications that use JDBC
  - Partition database application into JAVA and SQL
  - Rewrite from JAVA to SQL and vica versa.
  - Imperative code to declarative code

September 16, 2014  Databases and Data Mining  87

Optimizing Database Applications: previous work

UltraLite (D.P. Yach, 2002)
- Program with embedded SQL
- SQL query is send to DB that parses and optimizes the query
- The optimized query execution plan is send back
- Generate C code that uses UltraLite runtime database functions from its runtime library
- For mobile devices no hard drive limited amount of memory.

September 16, 2014  Databases and Data Mining  88

Optimizing Database Applications: previous work

GignoMDA (2006)
- For new applications using a Model Driven Architecture
- During development (design process) hints/directives can be given like tables are read-only, etc.
- Optimizations can then be done across layers:
  - Presentations layer (Web-interface)
  - Business logic layer
  - Persistence layer (DBDMS)

September 16, 2014  Databases and Data Mining  89

Query Planning and Optimization

Traditionally:
- An execution plan for a given query
- Minimize disk I/O
- Minimize network traffic

Nowadays:
- Main memory Databases, for example MonetDB (P.A. Boncz et al., 2008)
- Main memory bandwidth optimization
- CPU cache re-use
- Vector processing, with sizes that fit in cache
- x100 query engine of MonetDB maps queries to primitives (C functions)
- Optimizing compilers with aggressive loop pipelining and vectorization optimizations
- 2 orders of magnitude performance improvements over existing DBMS’s

September 16, 2014  Databases and Data Mining  90
Query Planning and Optimization

Holistic query evaluation (K. Krikellas et al., 2010):
- An execution plan for a given query translated to imperative code
- Using code templates and shared libraries for optimized database server accesses
- Optimizing compiler transformations applied
- The Application is not yet taken into account

Data centric (T. Neumann, 2011)
- SQL statements translated to relational algebra
- LLVM assembly code generation
- Optimizing JIT compiler of LLVM used to optimize
- Optimization: Data is kept in CPU registers as long as possible
- Again: query optimization only.

Other approaches:
- DBToaster (2009) query compilation into C++ focused on maintaining aggregated views at high update rates.
- Etc.

Query Planning and Optimization

K.F.D. Rietveld (2014)
- Compiler optimizations for database applications
- Optimizing both the application and database queries in an integrated way.

Restructuring Compiler Optimization:
- Starts with the Application code.
- Replace DBMS API calls with forelem loop nests and result sets and other constructs.
- Apply dependency analysis and apply restructuring transformations:
  - Loop collapse,
  - Removing unused columns and rows by iteration space reduction transformations
  - Etc.

References