

[R] Evolution of Database Technology

By EM Bakker

Evolution of Database Technology

- 1960s:
 - Data collection, database creation, IMS 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.)
- 1990s:
 - Data mining, data warehousing, multimedia databases, and Web databases
- 2000s:
 - Stream data management and mining
 - Data mining and its applications
 - Web technology (XML, data integration) and global information systems

**"Database Systems:
A Textbook Case of Research Paying Off",**

**J.N. Gray, Microsoft
1997**

<http://www.cs.washington.edu/homes/lazowska/cra/database.html>

Industry Profile (1994)

- The database industry \$7 billion in revenue in 1994, growing at 35% per year.
- Second only to operating system software.
- All of the leading corporations are US-based: **IBM, Oracle, Sybase, Informix, Computer Associates, and Microsoft**
- Specialty vendors: **Tandem** (\$1 billion/year) of fault-tolerant transaction processing systems, and **AT&T-Teradata**, (\$500 million/year, 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.

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:
 - a batch of transactions was applied to the old-tape-master
 - a new-tape-master produced
 - printout for the next business day.



Historical Perspective (1970's)

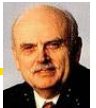
- During this era, there was considerable experimentation with systems to manage an on-line database that could capture transactions as they happened, rather than in daily batches.
- At first these systems were ad hoc
- Late in the 60's, "network" and "hierarchical" database products emerged.
- A network data model standard (DBTG) was defined, which formed the basis for most commercial systems during the 1970's.
- In 1980 DBTG-based Cullinet was the leading software company.

Historical Perspective

DBTG problems:

- DBTG used a low-level, record-at-a-time procedural language
- 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, enunciated by Ted Codd in a landmark 1970 article, was a major advance over DBTG.

- The relational model unified data and metadata so that there was only one form of data representation.
- A non-procedural data access language based on algebra or logic.
- It was easier for end-users to visualize and understand than the pointers-and-records-based DBTG model.
- Programs could be written in terms of the "abstract model" of the data, rather than the actual database design => programs were insensitive to changes in the database design.

The "relational" data model

- The research community (both industry and university) embraced the **relational data model** and **extended** it during the **1970s**.
- Most significantly, researchers showed 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 **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, in addition to relational systems).

1980's

The database research agenda:

- geographically **distributed databases**
- **parallel** data access.
- Theoretical work on **distributed databases** led to **prototypes** which in turn led to products. *Note: Today, all the major database systems offer the ability to distribute and replicate data among nodes of a computer network.*
- **Execution** of each of the relational data operators in **parallel** => hundred-fold and thousand-fold speedups. *Note: The results of this research are now beginning to appear in the products of several major database companies.*

Some Case Studies

The government has **funded** a number of **database research** efforts from 1970 to the present from which **two case studies** are selected:

- Projects at **UCLA** gave rise to Teradata and produced many excellent students.
- Projects at **CCA** (SDD-1, Daplex, Multibase, and HiPAC) pioneered distributed database technology and object-oriented database technology.
- Projects at **Stanford** created deductive database technology, data integration technology, and query optimization technology.
- Work at **CMU** gave rise to general transaction models and ultimately to the Transarc Corporation.
- There have been many other successes from AT&T, the University of Texas at Austin, Brown, Harvard, Maryland, Michigan, MIT, Princeton, and Toronto, among others.

Ingres

The project Ingres started at UC Berkeley in 1972. Inspired by Codd's work on the relational database model, (Stonebraker, Rowe, Wong, and others) a project that resulted in:

- design and build a relational database system.
- a query language (QUEL) was invented,
- relational optimization techniques,
- a language binding technique
- storage strategies.
- pioneering work on distributed databases.

The Ingres academic system formed the basis for the Ingres product now owned by Computer Associates.

The Ingres project went on to investigate:

- distributed databases
- database inference
- active databases (automatic responding)
- extensible databases.

Renamed Postgres, (also the basis for a new object-relational system)

System R

- Codd's ideas were inspired by seeing the problems IBM and its customers were having with the DBTG network data model and with IBM's product based on this model (IMS).
- Codd's relational model was at first very **controversial**: people thought that the model was too simplistic and that it could never give good performance. IBM Research management took a gamble and chartered a **10-person effort to prototype a relational system** => a prototype, **System R**, that eventually grew into the **DB2** product series.
- **Studies on: query optimization, data independence (views), transactions (logging and locking), and security (the grant-revoke model).** In addition, the SQL query language from System R was the basis for the standard that emerged.)
- The System R group went on to investigate distributed databases (**project R***) and object-oriented extensible databases (project Starburst).

The Future of 1997

- Database systems continue to be a key aspect of Computer Science & Engineering today. **Representing knowledge** within a computer is one of the central challenges of the field. Database research has focused primarily on this fundamental issue.
- There continues to be active and valuable research on:
 - representing and **indexing** data,
 - adding **inference** to data search,
 - compiling queries more **efficiently**,
 - executing queries in **parallel**,
 - **integrating** data from heterogeneous data sources,
 - analyzing **performance**, and
 - **extending the transaction model** to handle long transactions and workflow (transactions that involve human as well as computer steps).
 - The availability of very-large-scale (tertiary) storage devices has prompted the study of models for **queries on very slow devices**.

The Future of 1997

- **Unifying object-oriented** concepts with the relational model.
- **New datatypes** (image, document, drawing) are best viewed as the methods that implement them rather than the bytes that represent them.
- By adding procedures to the database system, one gets **active databases, data inference, and data encapsulation**. The object-oriented approach is an area of active research.

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

Conclusions of the Forum

- The database research community plays a foundational role in creating the technological infrastructure from which database advancements evolve.
- Next-generation database applications enabled by the explosion of digitized information over the last five years will require the solution to significant new research problems:
 - support for multimedia objects,
 - distribution of information,
 - new database applications,
 - workflow and transaction management, and
 - ease of database management and use.
- A new research mandate for the database community is provided by the technology developments of the recent past -- the explosions in hardware capability, hardware capacity, and communication (including the internet or "web" and mobile communication).

New Research Achievements

- Object-Oriented and Object-Relational Database Systems
 - Commercial systems available (in contrast to 1990); new breeds (deductive object oriented databases, ...)
- Support for new Data Types
 - GIS, Temporal Data (TSQL2), multimedia
- Transaction Processing

Transaction Processing

- Requirements and definitions during 1970's and early 1980's
- Alternative's to atomic transactions already studied from 1970 onward.
- Some commercial products support replicated data, allowing consistent views of the same information to exist at various nodes of a network.
- Some object-oriented systems now support "long transactions," where data is "checked out" and not restored for hours or days.
- Some commercial database systems also support "versions and configurations," the ability to construct a history of related objects ("versions" of that object) and to combine specific versions of different objects into "configurations."

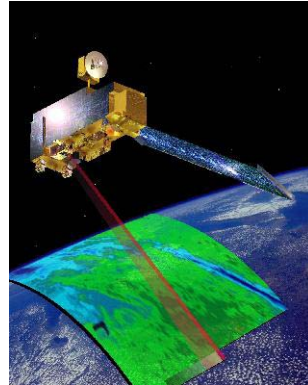
New Database Applications

- 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:

- Providing on-line access to petabyte-sized databases and managing tertiary storage effectively.
- Supporting thousands of information 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
- 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,

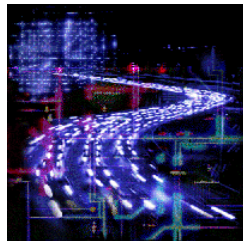


New Research Directions (1996)

- 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.

Distribution of Information (1996)

- Degree of Autonomy
- Accounting and Billing
- Security and Privacy
- Replication and Reconciliation
- Data Integration and Conversion
- Information Retrieval and Discovery
- Data Quality



Support for Multimedia Objects (1996)

- Tertiary Storage (for petabyte storage)
 - Tape silos
 - Disk juke-boxes
- New Data Types
 - The operations available for each type of multimedia data, and the resulting implementation tradeoffs.
 - The integration of data involving several of these new types.
- Quality of Service
 - How best do we assure timely and realistic presentation of the data in its desired form?
 - When the system is unable to meet all expectations for service, how do we gracefully degrade service? Can we interpolate or extrapolate some of the data? Can we reject new service requests or cancel old ones?
- Multi-resolution Queries
 - Content Based Retrieval
- User Interface Support



Ease of Use (1996)

- Intelligent tools for system managers
- **Electronic Wizards**
- Intelligent support for **performance enhancement**
- Electronic **physical database design tool** for index selection and database schema design

"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.

Data Warehousing

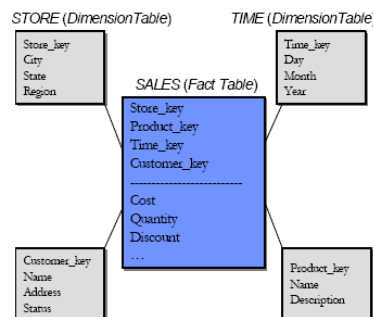
- In the early 1990's, a new trend appeared: Enterprises wanted to gather together data from **multiple operational databases** into a **data warehouse** for business intelligence purposes.
- A typical large enterprise has 50 or so **operational systems**, each with an on-line user community who **expect 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 Warehousing

- Data warehouses are **very different from OLTP systems**. OLTP systems have been optimized for updates, as the main business activity is typically to sell a good or service.
- In contrast, the main activity in data warehouses is **ad-hoc queries**, which are often **quite complex**.
- Hence, **periodic load of new data** interspersed with **ad-hoc query activity** is what a typical warehouse experiences.

Data Warehousing

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



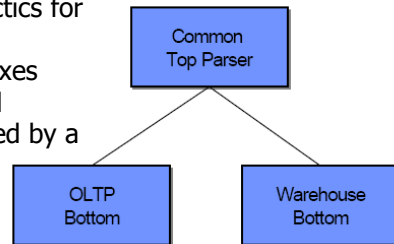
Data Warehousing

- It is a well known homily that warehouse applications run much better **using bit-map indexes** while OLTP users prefer **B-tree indexes**.
- In addition, **materialized views** are a useful optimization tactic in warehouse worlds, but never in OLTP worlds. In contrast, **normal ("virtual") views** find acceptance in OLTP environments.

Data Warehousing

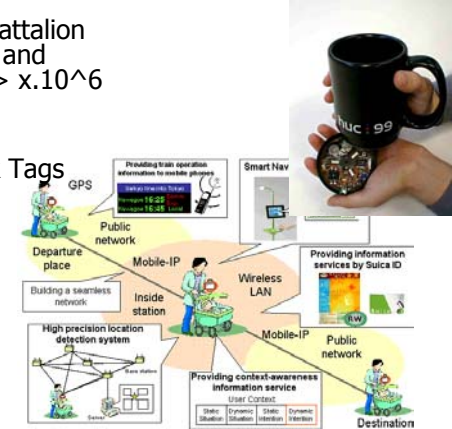
To a first approximation, most vendors have a

- **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



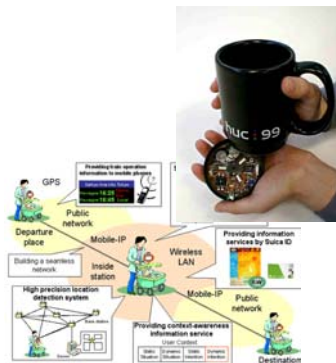
Emerging Sensor Based Applications

- Sensing Army Battalion of 30000 humans and 12000 vehicles => $\times 10^6$ sensors
- Monitoring Traffic
- Amusements Park Tags
- Health Care
- Library books
- Etc.



Emerging Sensor Based Applications

- There is widespread speculation that **conventional DBMSs will not perform well** on this new class of monitoring applications.
- In fact, on Linear Road, traditional solutions are nearly an **order of magnitude slower** than a special purpose stream processing engine



Example: An existing application: financial-feed processing

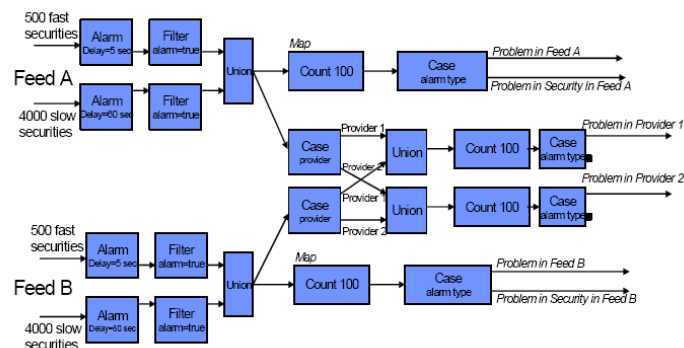
- Most large financial institutions subscribe to feeds that deliver real-time data on market activity, specifically news, consummated trades, bids and asks, etc. for example: **Reuters, Bloomberg and Infodyne**
- Financial institutions have a variety of applications that process such feeds. These include systems that
 - produce real-time business analytics,
 - perform electronic 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 application experts have not had good luck with off-the-shelf system software products.

Example: An existing application: financial-feed processing

Example Streaming stock ticks:

- Specifically, there are 4500 securities, 500 of which are "fast moving".
- A stock tick on one of these securities is **late** if it **occurs more than five seconds after the previous tick** from the same security.
- The other 4000 symbols are slow moving, and a tick is **late** if **60 seconds have elapsed since the previous tick**.

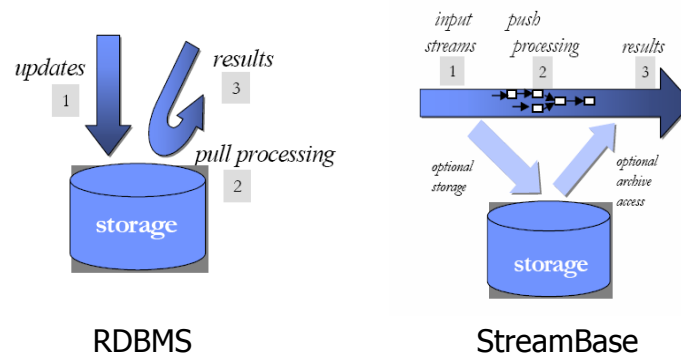
Stream Processing



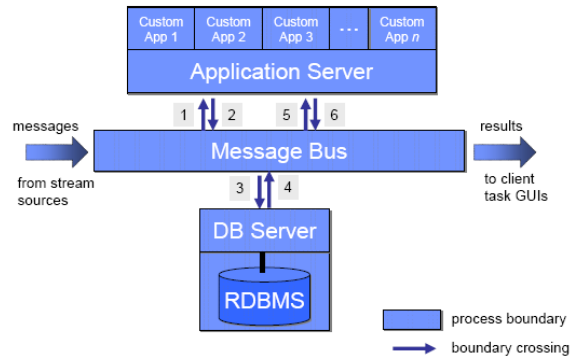
Performance

- The example application was implemented in the **StreamBase** stream processing engine (SPE) [5], which is basically 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 coax **900 messages per second** from an implementation of the same application using a popular commercial relational DBMS.

Why?: Inbound vs Outbound Processing



Inbound vs Outbound Processing

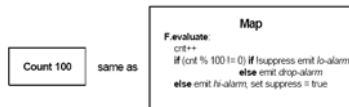


Inbound vs Outbound Processing

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

Other Issues: Correct Primitives

Select avg (salary)
From employee
Group by department



- SQL systems contain a **sophisticated aggregation system**, whereby a user can run a statistical computation over groupings of the records from a table in a database.
- When the execution engine processes the last record in the table, it can emit the aggregate calculation for each group of records.
- However, **streams** can continue forever and 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

- 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

- In an embedded processing model, it is reasonable to freely mix
 - application logic
 - control logic and
 - 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** (e.g., [22]) are ill-suited for the **streaming world** as they introduce several key problems.
- 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 to provide isolation (among others things) between concurrent transactions submitted by multiple users.
- In streaming systems, which are not multi-user, such isolation can be effectively achieved through simple critical sections, which can be implemented through light-weight semaphores.

One Size Fits All?



One Size Fits All?

- 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