Database Technology

Erwin M. Bakker & ***Stefan Manegold***

https://homepages.cwi.nl/~manegold/DBDM/
http://liacs.leidenuniv.nl/~bakkerem2/dbdm/

s.manegold@liacs.leidenuniv.nl
e.m.bakker@liacs.leidenuniv.nl

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 -
  - 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

1960’s

- 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.

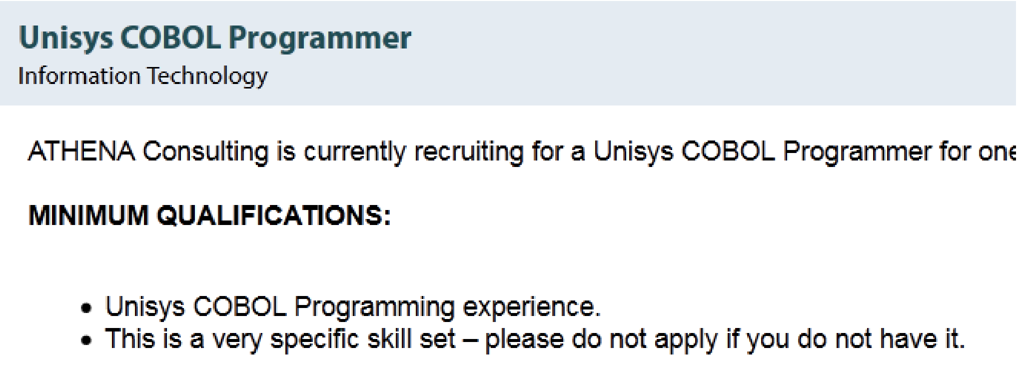
- **CO**mmon **B**usiness-**O**riented **L**anguage (COBOL 2002 standard)

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.”

September 2015:



Unisys COBOL Programmer
Information Technology

ATHENA Consulting is currently recruiting for a Unisys COBOL Programmer for one

MINIMUM QUALIFICATIONS:

- Unisys COBOL Programming experience.
- This is a very specific skill set – please do not apply if you have do not have it.

COBOL Code (just an example!)

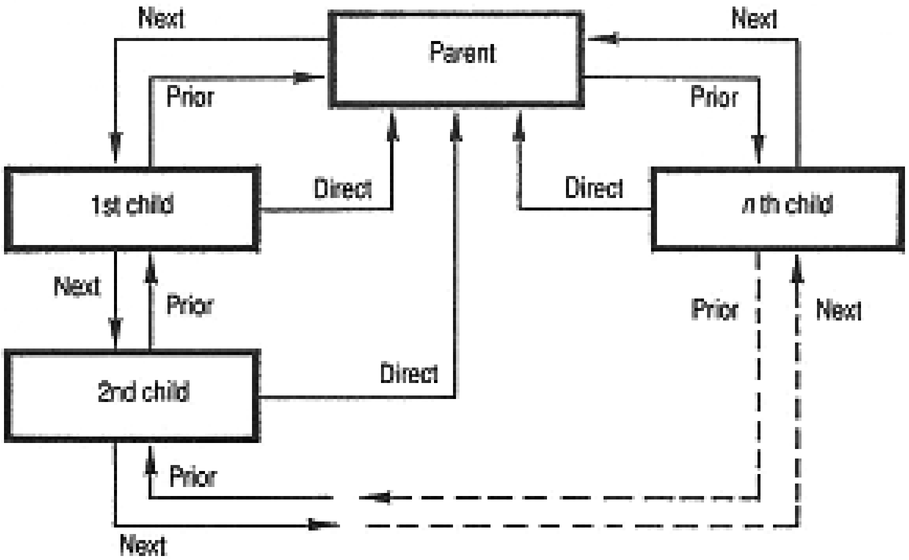
```
01 LOAN-WORK-AREA.
     03 LW-LOAN-ERROR-FLAG     PIC 9(01)       COMP.
     03 LW-LOAN-AMT            PIC 9(06)V9(02) COMP.
     03 LW-INT-RATE            PIC 9(02)V9(02) COMP.
     03 LW-NBR-PMTS            PIC 9(03)       COMP.
     03 LW-PMT-AMT             PIC 9(06)V9(02) COMP.
     03 LW-INT-PMT             PIC 9(01)V9(12) COMP.
     03 LW-TOTAL-PMTS          PIC 9(06)V9(02) COMP.
     03 LW-TOTAL-INT           PIC 9(06)V9(02) COMP.
*
   004000-COMPUTE-PAYMENT.
*
     MOVE 0 TO LW-LOAN-ERROR-FLAG.

     IF (LW-LOAN-AMT ZERO)
       OR
      (LW-INT-RATE ZERO)
       OR
      (LW-NBR-PMTS ZERO)
       MOVE 1 TO LW-LOAN-ERROR-FLAG
       GO TO 004000-EXIT.

     COMPUTE LW-INT-PMT = LW-INT-RATE / 1200
       ON SIZE ERROR
         MOVE 1 TO LW-LOAN-ERROR-FLAG
         GO TO 004000-EXIT.
```

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



A closed chain of records in a navigational database model (e.g. CODASYL), with *next pointers*, *prior pointers* and *direct pointers* provided by keys in the various records.

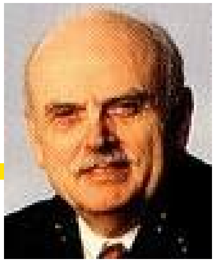
- network model: each record can have multiple parent and child records, i.e., a lattice of records

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, by Ted Codd (Turing Award 1981) in his landmark 1970 article *“A Relational Model of Data for Large Shared Data Banks",* was a major advance over DBTG.

- The relational model unified data and metadata => only one form of data representation.
- A non-procedural data access language based on algebra or logic.
- The data model is easier to visualize and understand than the pointers-and-records-based DBTG model.
- Programs written in terms of the "abstract model" of the data, rather than the actual database design =>

  programs insensitive to changes in the database design.

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

Stonebraker (Turing award 2014), Rowe, Wong, and others:

- a relational database system, query language (QUEL)
- relational optimization techniques
- storage strategies
- work on distributed databases

Further work on:

- database inference
- active databases (automatic responsing)
- 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).

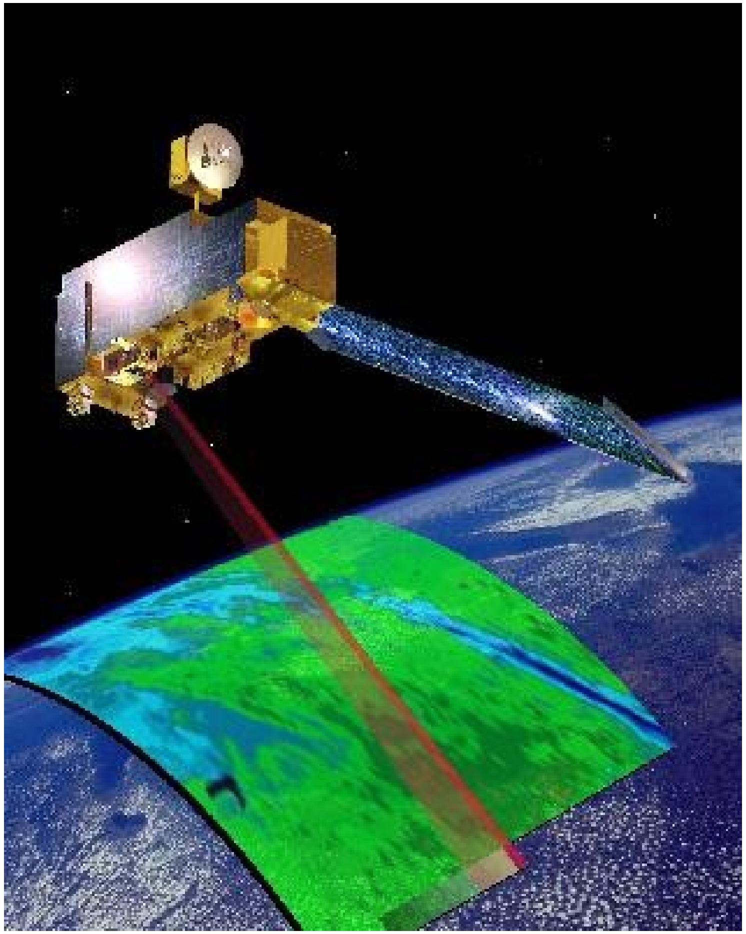
The Big New Database Applications of 1990's

- 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

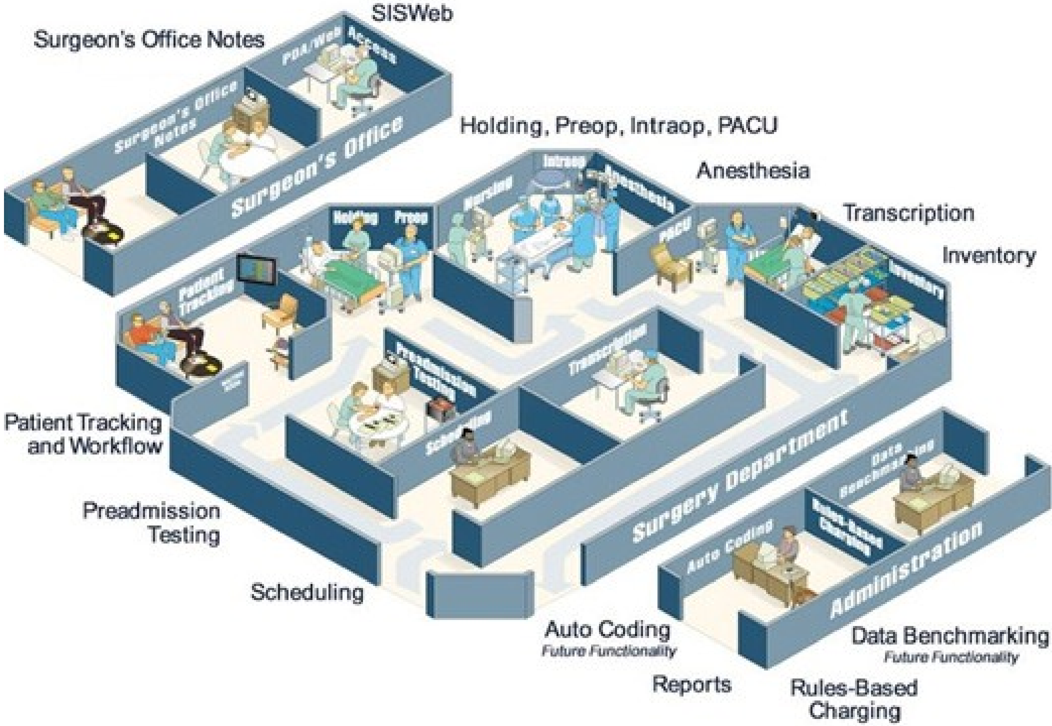
- "schema integration“ is a well-known and extremely difficult problem.

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:

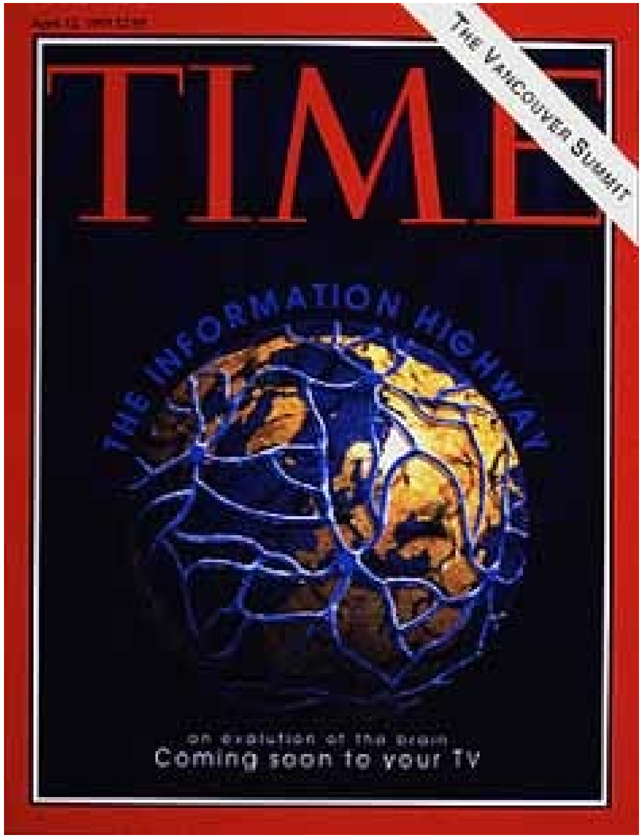
- Integration of heterogeneous forms of legacy information.
- Access control to preserve the confidentiality of medical records.
- Interfaces to information that are appropriate for use by all health-care professionals.

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 (1990's)
- 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 (1990's)

- 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
  - timely and realistic presentation of the data?
  - gracefully degradation service? Can we interpolate or extrapolate some of the data? Can we reject new service requests or cancel old ones?
- Content-Based Retrieval
- User Interface Support

Conclusions of/for DB Community

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

New Research Directions (1990's)

- Problems associated with putting multimedia objects into DBMSs: new data types
- Problems involving new paradigms for distribution and processing of information.
- **New uses of databases**
  - Data Mining
  - Data Warehouses
  - Repositories
- New transaction models
  - Workflow Management
  - Alternative Transaction Models (long transactions)
- Problems involving **ease of use** and management of databases.

“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://ww.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

Data Warehousing

- 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 Warehousing

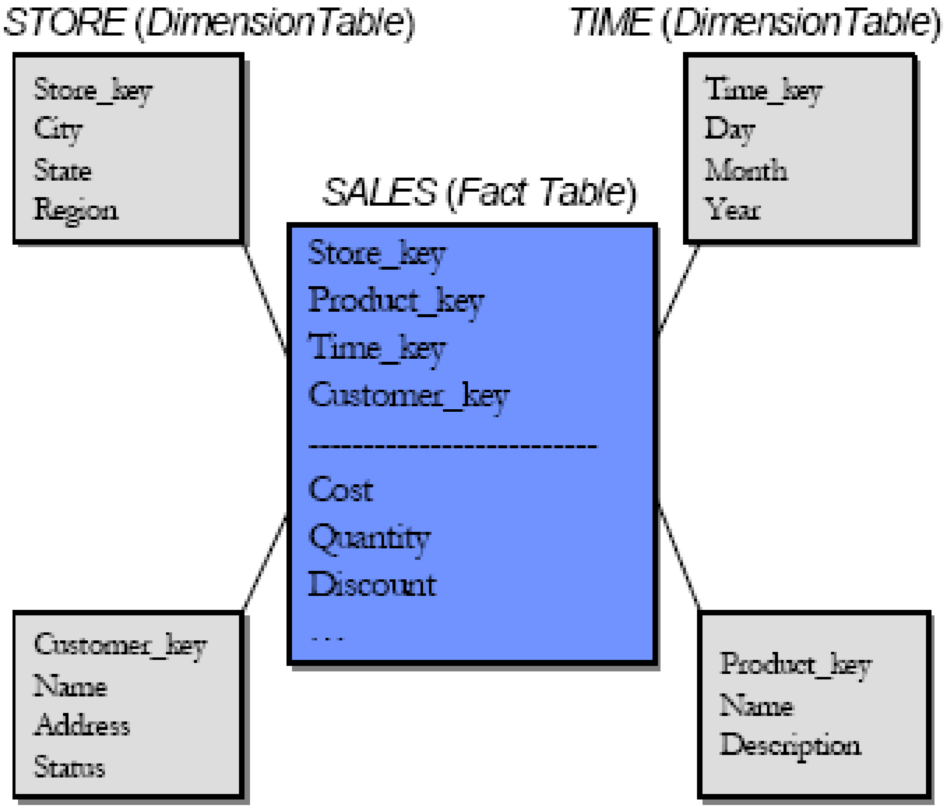
Data warehouses are very different from Online Transaction Processing (OLTP) systems:

- OLTP systems:
  - the main business activity is typically to sell a good or service
  - => optimized for updates
- Data warehouse:
  - ad-hoc queries, which are often quite complex.
  - periodic load of new data interspersed with ad-hoc query activity

Data Warehousing

The standard wisdom in data warehouse schemas is to create a fact table:

“who, what, when, where” about each operational transaction.



STORE (DimensionTable)

Store_key
City
State
Region

TIME (DimensionTable)

Time_key
Day
Month
Year

SALES (Fact Table)

Store_key
Product_key
Time_key
Customer_key

Cost
Quantity
Discount
…

Customer_key
Name
Address
Status

Product_key
Name
Description

Data Warehousing

- 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.

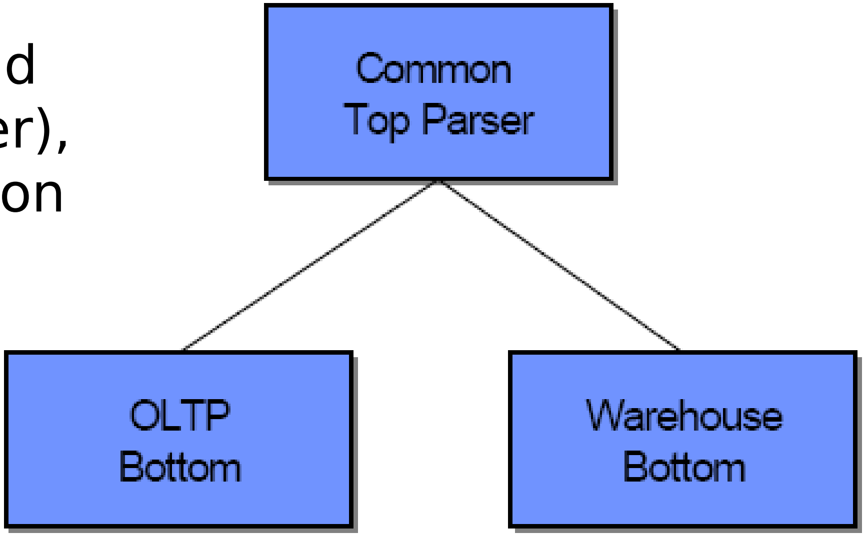
Data Warehousing

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

		Bitmaps	
Index	Gender	F	M
1	Female	1	0
2	Female	1	0
3	Unspecified	0	0
4	Male	0	1
5	Male	0	1
6	Female	1	0



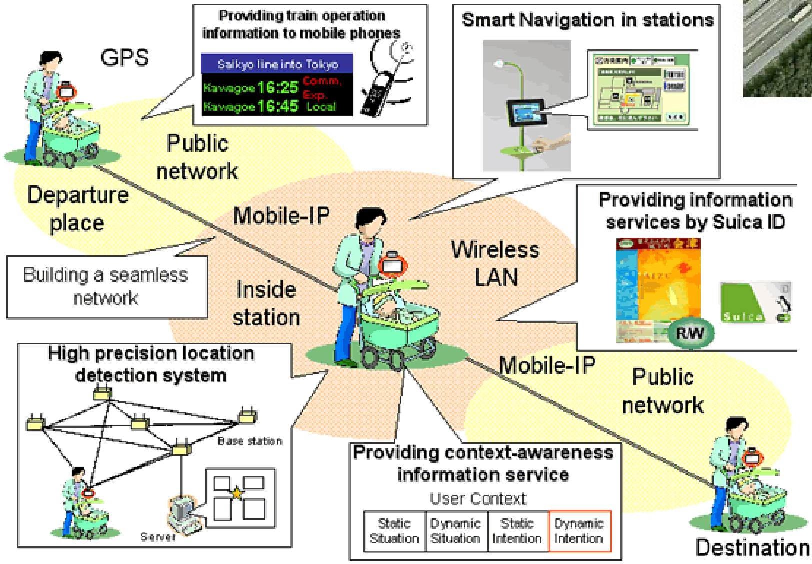
Emerging Applications

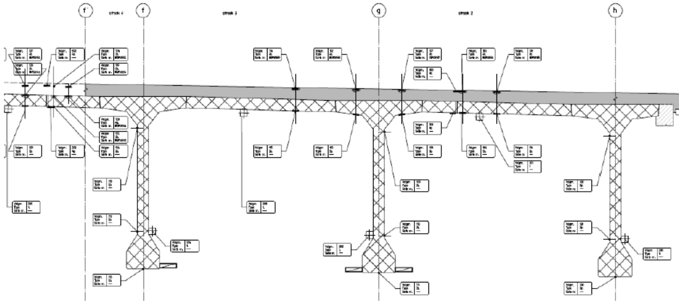
Some other examples that show:

Why conventional DBDMs will
not perform on the current emerging applications.

Emerging Sensor Based Applications

- Sensoring Army Battalion of 30000 humans and 12000 vehicles => x.10^6 sensors
- Monitoring Traffic (InfraWatch, 2010)
- Amusements Park Tags
- Health Care
- Library books
- Etc.

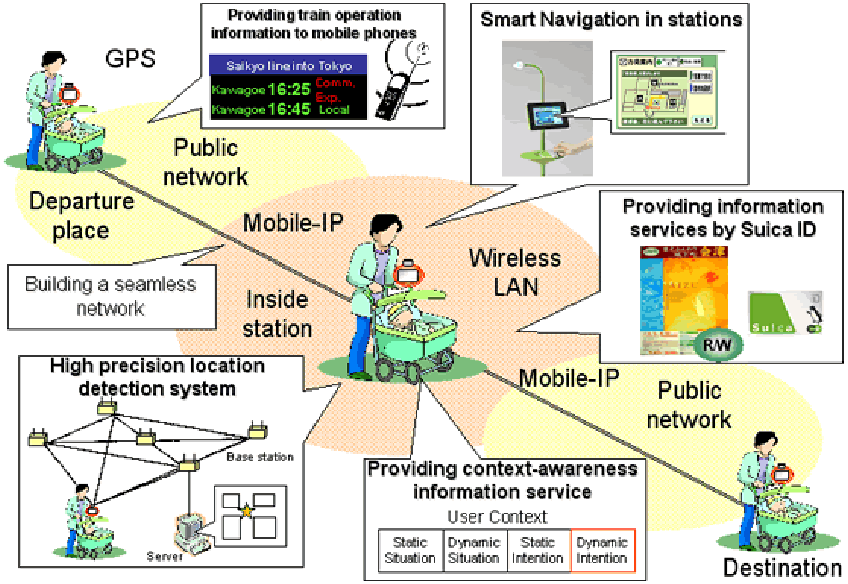




Emerging Sensor Based Applications

- 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. (2005)

Example: An existing application: financial-feed processing

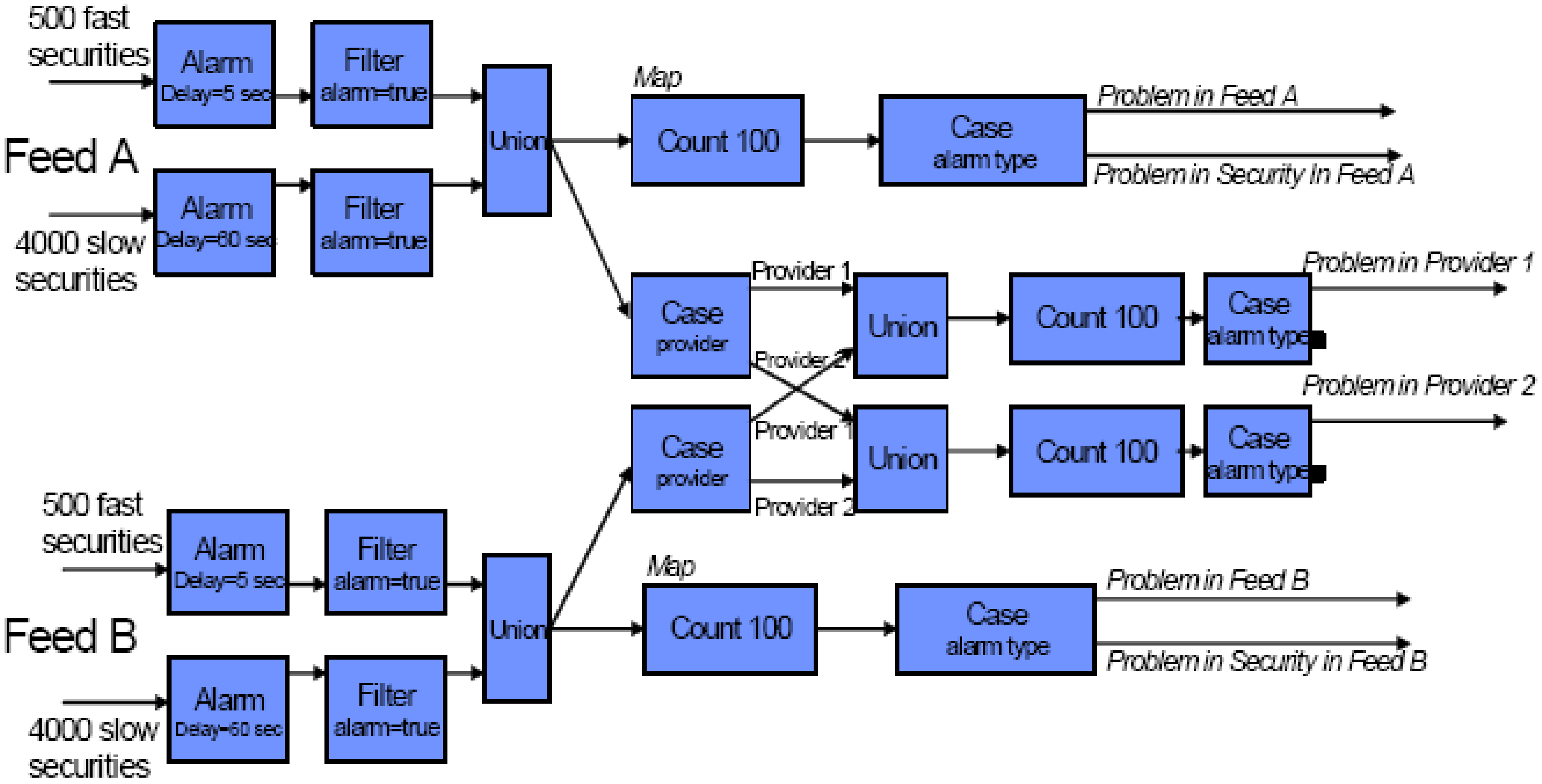
Detect Problems in Streaming stock ticks:

- Specifically, there are 4500 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 4000 symbols are slow moving, and a tick is late if 60 seconds have elapsed since the previous tick.

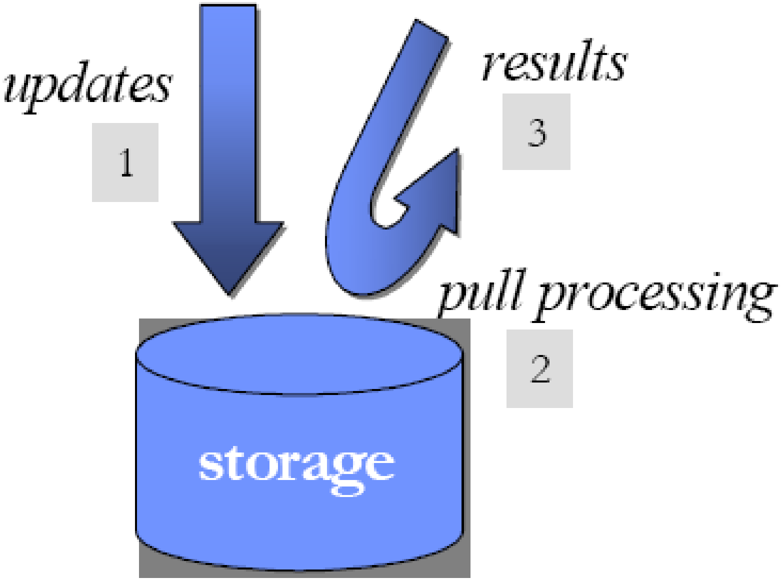
Stream Processing



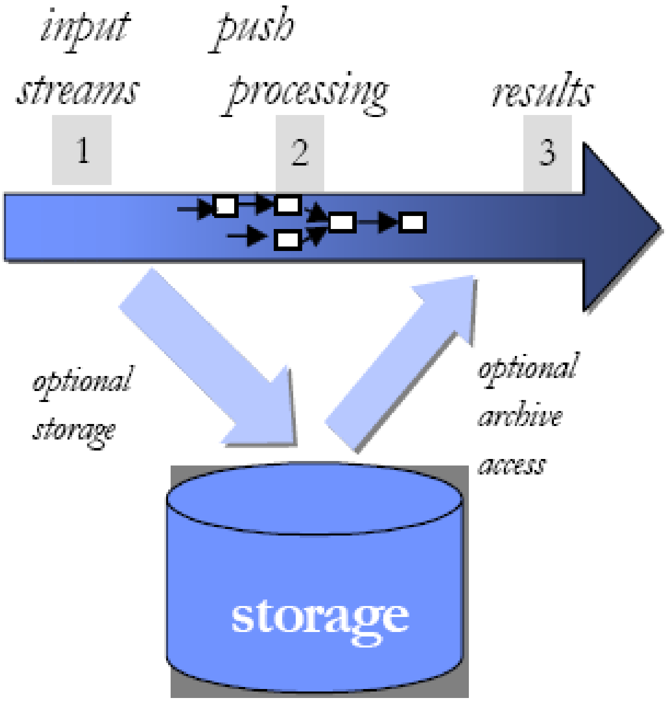
Performance

- 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

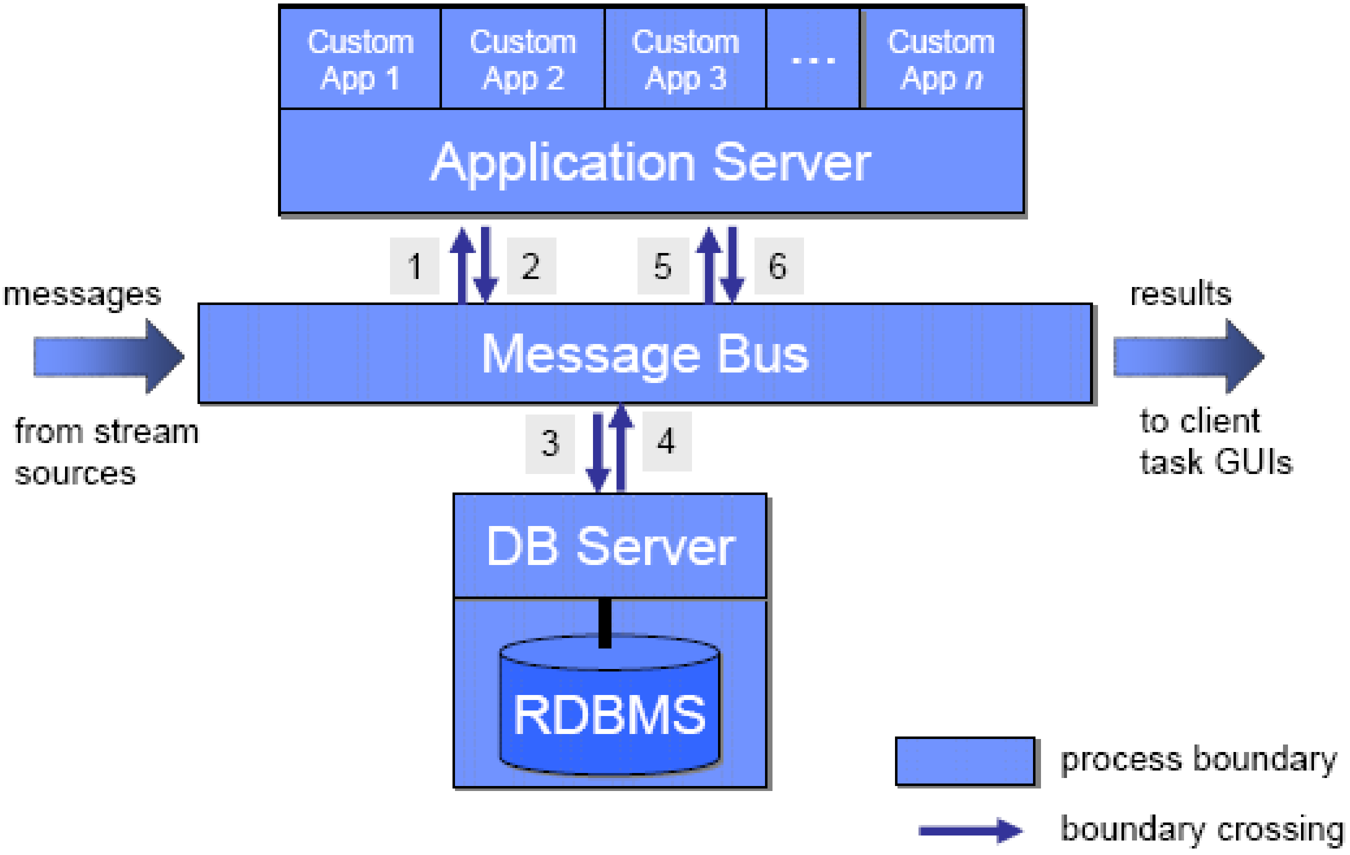


RDBMS
(Outbound Processing)



StreamBase
(Inbound Processing)

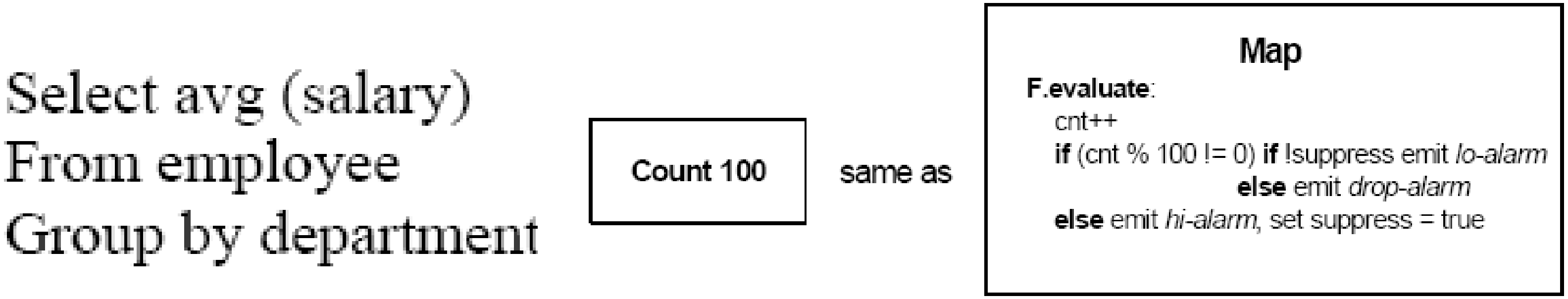
Inbound Processing



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



Select avg (salary)
From employee
Group by department

Count 100

same as

Map

```
F.evaluate:
    cnt++
    if (cnt % 100 != 0) if !suppress emit lo-alarm
                        else emit drop-alarm
    else emit hi-alarm, set suppress = true
```

- 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 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 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 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?

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