Issues in combined static and dynamic data management


Daniela Nicklas

Department für Informatik
Database and Internet Technologies
Carl von Ossietzky Universität Oldenburg
Germany

dnicklas@acm.org
Overview

- Introduction
- Some existing approaches
  - for dynamic data management
  - for static data management
- Federated Architecture
- Issues
- Outlook

**Issues in combined static and dynamic data management**

Application Areas (... of my current interest)

- Traffic monitoring
- Safe Off-shore Operations
- Vessel Traffic Management
- Logistics
- Driver assistant systems
- Smart Factory
- ... and many more!

- Crew
- Congestion
- Snow
- Pedestrians
- Goods
- Trucks
- Parcels
- Obstacles
- Lanes
- Cars
- Roads
- Wind
- Cargo
- Processes
- Sea
- Ships
- Ground
- Products
- Workers
- Machines

dnicklas@acm.org
Example Application Domain: Safe Offshore Operations

- Goal: Improve the safety of offshore operation for people, machines, and processes  (4 year project, more @ http://soop.offis.de)
Static and Dynamic Data Management …

Data Base Management Systems
- "store, then process"
- analytical queries
- one-time queries
- well-known technology
- user/application is active, data management passive

Data Stream Management Systems
- "on-the-fly" data processing
- real-time reaction on changes
- store only what needs to be kept
- register long-running queries
- user/application is passive, data management pushes data
- Complex event processing for temporal patterns
Typical queries in database management systems

- **Database user or application**
  - one-time query
  - results

- **Database administrator, user, or application**
  - insert
  - update
  - delete

- **Database**
  - Database management system
Typical queries in data stream processing systems

- Continuous updates to data stream
- Data stream administrator, user, or application
- Continuous query (register)
- Data stream processing system
- Non-stream data
- Database / DBMS
- (partial) archive
- Results
- Data stream user, or application
Example: NexusDS

NexusDS Applications

NexusDS Middleware

core services
- query service
- operator execution service

domain specific services
- visualization service
- history service

constraints

Physical Layer
- static data
- dynamic data

logical processing graph

source
sink
operator
node with execution service

output

© Nazario Cipriani

see Friday 27, Session 4, 11:30 - 13:00 Trust, Privacy and Security.
NexusDSS: A System for Security Compliant Processing of Data Streams
DSM – Challenges

- **Data stream**
  - Continuously arriving data
  - Potential infinite

- **Challenges**
  - Blocking behaviour of query operators not adequate for stream processing
    - E.g., how to compute the average?
      → unblocking by windows
  - Temporal relationship of streaming data
Window definitions

- Monotone window operator to split stream into segments
- Window size can be based on:
  - number of elements (e.g., last 100 elements)
  - time (e.g., last 5 seconds)
  - predicates of elements (e.g., value between two thresholds)
- Window stride: how it moves
  - jumping or tumbling (no overlaps, data processed once)
  - sliding (continuously, overlaps, data processed more than once)
  - sampling (no overlaps but gaps, some data is not processed)
- Window realization (next slide)
  - interval approach
  - positive/negative approach

dnicklas@acm.org
Features of Data Stream Management Systems

- Programming Abstraction
  - declarative: query
  - functional: flow graph
    → enables optimizations
    → better maintainance of systems
    → using a DSMS on data streams is like using a DBMS instead of files

- Data flow vs. event bus (as in many CEP engines)
  - execution of data flow processes only data items that are needed by a query

- Parallel execution of operators in graph
  → no shared memory

- Data streams can be unbounded:
  - issues with sorting, joins, aggregation
    → approximate answers
    → window semantics

SELECT ego.pos FROM ego
RANGE 10 seconds,
radar RANGE 15 seconds
WHERE ego.speed > 30 AND
radar.speed > 30
AND s2.pos - ego.pos < 15
However …

Data Base Management Systems
- ACID properties, persistent storage
- well-established semantics (rel. algebra)
- well-established optimizers
- often deeply integrated into enterprise IT infrastructure
- can also cope with dynamic management (triggers, repeated optimized queries)

Data Stream Management Systems
- main memory processing
- often non-deterministic results (depend on timing of streams)
- no standard query semantics (often relational algebra with extras, or many user-defined operators)
- new system in IT infrastructure
- has to be combined with static data management

dnicklas@acm.org
Can’t we do that with a DBMS?

• Active Data Management in DBMS (Triggers, CQN)
• Magic Updates: The Bonn Approach
Dynamic Data in DBMS

- **Database triggers**
  - Are designed to intercept database change requests and transaction requests
  - Are part of a transaction – synchronous
  - Allow to check validity of changes
  - Widely used in applications
  - But: since triggers can trigger triggers, complex logic gets hard to maintain

- **Database Log Miner (e.g., Oracle)**
  - The log miner is designed to select specific journal entries in batch or real time mode
  - Works asynchronously
  - Shows ‘dirty’ data
  - Works closely with dissemination services such as Oracle Streams and Oracle GoldenGate
  - Used by many infrastructure services such as replication, logical standby, real time warehousing, security auditing, real time analytics through TimesTen (In Memory Database), CEP
Continuous Query Notification - CQN

- CQN is designed to notify users/applications if the result set of a query has changed
  - Requests are defined as registered queries – normal query plus a notification destination
  - Works asynchronously
  - Shows only ‘visible’ data

- The following events generate notifications:
  - Committed DML Transactions
  - Committed DDL Statements
  - Deregistration
  - Global Events
Continuous queries 'made' in Bonn:

• We’re experts for update propagation techniques.

• A considerable amount of stream applications can be realized using conventional DBS.

• Continuous queries are implemented using materialized views.

• Incremental techniques are used for their maintenance.

• In order to process static domain knowledge with dynamic stream data, we employ Magic Updates!
Data Stream research today:

• It is widely believed that conventional DBS are not suited for processing continuous queries.

• Instead, main memory-based DSMS are employed:
  - no transactions (no recovery, no rollback,…)
  - Limited multi-user access
  - no history recorded
  - etc.

  ⇒ Basically, only SQL survived

• DSMS can process fast stream data but static domain knowledge only in a limited way.
Case study: View-based analysis of aircraft movements using the Airspace Monitoring System (AIMS)

- Which aircrafts approach each other critically?
- Which aircrafts are currently landing?
- Which planes are late?
- Which flight are entering areas with bad weather conditions?
- Which planes are changing their course unexpectedly?
- What is the average number of flights for an arbitrarily chosen area?
Architecture of AIMS

Clients

- Browser
- Java GUI
- Other Systems

Web Infrastructure

- Load Balancing, Caching, Authentication, Encryption

Application Server

- Web Server
- API
  - RESTful Web Service
  - JSON
  - KML
  - XML
- Feeder
  - periodic data push
  - 2000 tuples every 4 seconds

Database Server

- materialized views
- triggers
- DBMS
  - event log
  - domain knowledge

© Andreas Behrend, Universität Bonn
**dynamic track data**

**Flights**
- Call sign of a flight
- Aircraft type
- Registration
- Transponder code
- Position by latitude, longitude, altitude
- Vertical speed
- Horizontal speed
- Heading of the aircraft in degrees
- Time stamp
- …

**static domain data**

**Airports**
- Name of airport, city, country
- IATA / ICAO-Code
- Position by latitude, longitude, altitude
- Region code like “E” for Europe
- …

**Flight plan**
- Waypoints: beacon-code, position, speed, time
- Plan ID, source, country code
- Callsign, aircraft type, destination airport
- …
An aircraft is considered to be landing if

- its vertical velocity is negative
- its flight level is below zero
- it is currently approaching an airport less than 20 miles away.

```sql
CREATE VIEW landeAnflug AS
SELECT f.*
FROM flight AS f
WHERE f.Incline < 0 AND
  f.altitude < 'FL0' AND
  EXISTS(SELECT * FROM airport AS a
          WHERE getDist(f.Lon,f.Lat,a.Lon,a.Lat) < 20000)
```
Aircraft positions frequently change:

<table>
<thead>
<tr>
<th>CallSign</th>
<th>Type</th>
<th>Longitude</th>
<th>Latitude</th>
<th>…</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACA876</td>
<td>B763</td>
<td>-4.27333</td>
<td>52.50833</td>
<td>…</td>
<td>23-03-2007 14:17:03</td>
</tr>
<tr>
<td>ACA876</td>
<td>B763</td>
<td>-4.82512</td>
<td>51.33</td>
<td>…</td>
<td>23-03-2007 14:17:07</td>
</tr>
<tr>
<td>ACA876</td>
<td>B763</td>
<td>-5.92113</td>
<td>51.1211</td>
<td>…</td>
<td>23-03-2007 14:17:11</td>
</tr>
</tbody>
</table>

Time stamped data stream

**CQ:** Which aircrafts are currently landing?

Landing state **remains relatively stable!**

Motivation for

\[ \Rightarrow \text{incremental evaluation} \]
CREATE MATERIALIZED VIEW landeAnflug AS
SELECT f.CallSign, f.Time
FROM flight AS f
WHERE f.Incline < 0 AND
...

INSERT INTO landeAnflug
SELECT nf.CallSign, nf.Time
FROM new_flight_data AS nf
WHERE nf.Incline < 0 AND
...

<table>
<thead>
<tr>
<th>CS</th>
<th>Longitude</th>
<th>Latitude</th>
<th>...</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ACA873</td>
<td>-4.556667</td>
<td>52.578333</td>
<td>...</td>
<td>14:17:03</td>
</tr>
<tr>
<td>ACA876</td>
<td>-4.273333</td>
<td>52.508333</td>
<td>...</td>
<td>14:17:05</td>
</tr>
<tr>
<td>AFR1341</td>
<td>-4.803333</td>
<td>52.183333</td>
<td>...</td>
<td>14:17:08</td>
</tr>
</tbody>
</table>

Flight data
Performance achieved so far with AIMS:

- AIMS typically processes 2000 tuples (20 attributes) every 3-4 seconds
- Determination of landing/starting planes <1 second
- Determination of critical approximations <1 second
- Counting of flights within a user-defined region <2 seconds
Update propagation is suited for analyzing a wide spectrum of stream applications.

- Air Traffic Control
- Stochastic Analysis of Radar Data
- Optimizing ETL jobs
- Analyzing Stock Market Data
DBMS approaches assessment

- Consider updates and the change of queries
- Are good up to a couple thousand of tuples/second

- However, there are applications that need much higher throughput
  - real-time multi-sensor fusion
  - dynamic context models
Sensor Fusion Example (project SaLsA)

- Context-Model Generation for Safe Autonomous Transport Vehicles
  - Autonomic vehicles perform mobility operations with walking speed to ensure safety
  - Faster mobility requires specific knowledge of the environment
  - Realization via a dynamic context model based on sensor data with uncertainties
  - Using Data Stream Management Technology for Sensor Fusion
Sensor Fusion Example (project SaLsA)

Operators:
- **Union Operator**: Union and transmission of updated context models according to their timestamps
- **Store Operator**: Storing of new occupancy grid into the Context Store
- **Merge Operator**: Update of existent occupancy grid with new sensor data using Bayesian updater on each beam
- **Spread Operator**: Pessimistic prediction
- **Context Store**: Storage of multiple occupancy grids with their time of validity

- 2 Laserscanners with up to 1080 beams (@0.25°), resulting in 54,000 points per second per sensor
DBMS approaches assessment

- Consider updates and the change of queries
- Are good up to a couple thousand of tuples/second

However, there are applications that need much higher throughput
- real-time multi-sensor fusion (54000 points/second/sensor)
- dynamic context models
- distributed data processing
- specialized operators
- not all data needs to be stored

→ a federated approach might be a good idea!
To boldly go for a Federation
Federation: the Big Picture

Issues
- semantics?
- data flow?
- optimization?
- adaptation?
- provenance?
- persistance?
- interaction?
- distribution?

Application

Federation Service

DSMS
- cont. query, CEP

DBMS
- cont. query/trigger

"STATIC" DATA

DYNAMIC DATA FROM ACTIVE SOURCES

dnicklas@acm.org
Query Semantics

- How do we formulate federated queries?

  **DBMS:**
  - SQL (but no standardized window semantics)
  - ECA rules, stored procedures, CQN

  **DSMS:**
  - CQL (Arasu, Babu, and Widom 2006)
  - StreamSQL (StreamBase Systems 2012)
  - Sase (Gyllstrom et al. 2007)
  - Aurora’s boxes and arrows (Abadi et al. 2003)
  - IBM Infosphere Stream’s Stream Programming Language SPL (Biem et al. 2010)

→ for federated queries, we need a common query language and semantics!
Optimized Data Flow

- How do we decide about the data flow?
- Data from active resources is often produced decentralized
- In classical federation approaches, data is sent to the federation software
  - might not be the best idea here
- Decide:
  - where to process which operator?
  - send data from DBMS to application or from DSMS?

... data flow must be part of the processing optimization.
Optimized Cont. Query Distribution

- How do we decide about the optimal distribution of the continuous queries?

- Depends on
  - data location
  - update rate of data
  - required report rate of query result
  - required persistence of data
    - if system crashes and recovers, is the application interested in old (recovered) results?

  Options range from CQN/triggers to pure DSMS queries to mixed executions

Issues

- semantics?
- data flow?
- optimization?
- adaptation?
- provenance?
- persistance?
- interaction?
- distribution?
Query execution options

Options
pre-process in DSMS, register CQN in DBMS

Application
Federation Service
DBMS
DSMS

cont. query, CEP
DYNAMIC DATA FROM ACTIVE SOURCES

Options

1

cont. query/trigger/magic updates

pre-process in DSMS, register CQN in DBMS
Query execution options

1. Process in DSMS, register CQN in DBMS
2. Process in DSMS, get data from DBMS

Options

- pre-process in DSMS, register CQN in DBMS
- process in DSMS, get data from DBMS

Application

DSMS

Federation Service

DBMS

DYNAMIC DATA FROM ACTIVE SOURCES

cont. query, CEP

dnicklas@acm.org
Adaptation and Migration

- If the parameters used for the query distribution change, how do we adapt the continuous query to the new situation?
  - monitoring of
    • data location
    • update rate of data
    • required report rate of query result
    • required persistence of data

- If we decide to adapt – how do we migrate from old state to new state?
  - have to take care of stateful operators

Issues

- semantics?
- data flow?
- optimization?
- adaptation?
- provenance?
- persistance?
- interaction?
- distribution?
Adaptation and Migration

- Some works has been done for migrating pure DSMS queries:
  - F: Fresh start: start a new continuous query
  - I: Immediate stop: the query stops processing immediately
  - D: Drain stop: the query will be gradually stopped, finishing partially results

- Resulting stateflow diagram for continuous queries:

Kyumars Sheykh Esmaili, Tahmineh Sanamrad, Peter M. Fischer, Nesime Tatbul: Changing flights in mid-air: a model for safely modifying continuous queries. SIGMOD 2011:613-624

dnicklas@acm.org
Data and Query Provenance

- How can we know why a notification was produced (or not?)

- Many applications would code complex decision trees or situation recognition in federated DSMS/DBMS query plans

- Data provenance:
  - for a given federated result, which data tuples contributed to it?

- Query provenance:
  - when was / is which query plan active?
    - can be used to re-construct data provenance
Data and Query Provenance

- Use Cases
  - Ad-hoc human inspection
  - Continuous query debugging
  - Indicator-based assurance
  - Event warehousing

- Challenges
  - Infinity, Performance, and Aggregation
  - Non-determinism (of some DSMS)
  - Order (of DSMS processing)

- Some work for DSMS:

  Boris Glavic, Kyumars Sheykh Esmaili, Peter M. Fischer, Nesime Tatbul: The Case for Fine-Grained Stream Provenance. BTW Workshops 2011:58-61
Persistance and Intelligent Archiving

How can we realize persistance in a world of high volume data streams?
- not everything has to be persistent
- but how to decide?

1. manually
   - by registering archiving queries
2. randomly
   - by sampling parts of the stream
3. automatically intelligent = model-based & adaptive
   - by learning patterns / models in the stream, e.g., by stream data mining
     • archive only new patterns in high resolution
     • count the other (= store aggregated information)
Query execution options

1. Process in DSMS, get data from DBMS
2. Register CQN in DBMS
3. Persist or store results (and queries) in DBMS

Options

- Pre-process in DSMS, register CQN in DBMS
- Process in DSMS, get data from DBMS
- If persistence or provenance is needed, store results (and queries) in DBMS

Contact: dnicklas@acm.org
DSMS and DBMS interaction

- How do the DSMS and the DBMS interact?
  - who controls whom?
  - how tightly are they coupled?

1. The federation controls both system
   - low coupling between the systems
   - only running queries are affected if a system changes (interface, schema, …)

2. The DSMS controls the DBMS
   - it issues queries to the DBMS to access static data
   - this function is integrated in most DSMS

3. The DBMS controls the DSMS
   - would require concepts to include DSMS queries in, e.g., trigger definitions
Distributed Continuous Query Processing

- How to optimize the distribution of the query processing?
- Many active data sources are physically distributed or even mobile (sensor networks, mobile sensors in smart phones or trucks)
- Many sensing devices have (some) processing power
- Should be considered in query optimization
- e.g.: project „Resource constrained Distributed Stream Processing“ @Universität Erlangen, Germany

Issues

- semantics?
- data flow?
- optimization?
- adaptation?
- provenance?
- interaction?
- distribution?

Michael Daum, Frank Lauterwald, Philipp Baumgärtel, Niko Pollner, Klaus Meyer-Wegener: Efficient and cost-aware operator placement in heterogeneous stream-processing environments. DEBS 2011:393-394
Conclusion

- To combine static and dynamic data management, many issues have to be addressed:
  - DBMS has to be extended for temporal semantics
  - DSMS have to mature in processing semantics, determinism and persistence concepts
  - Together, a data management biotop has to have a

- In the long run, DSMS functionalities will be part of a DBMS system

- Outlook: consideration of NoSQL-concepts
Thank’s for the fish!

… also to Andreas Behrend (Uni Bonn), Nazario Cipriani (Uni Stuttgart), Dieter Gawlick (Oracle), Marco Grawunder (Uni Oldenburg) and Christian Kuka (OFFIS) for donating slides and ideas