ASSET Queries: A Set-Oriented and Column-Wise Approach to Modern OLAP

Damianos Chatziantoniou, Yannis Sotiropoulos
Department of Management Science & Technology
Athens University of Economics and Business (AUEB)
damianos@aueb.gr, yannis@aueb.gr

Athens University of
Economics and Business

Outline

- *Introduction*
- Motivation Part I: Large Data Analytics (SocNets)
- Motivation Part II: Data Streams (Finance)
- Expressing ASSET Queries: SQL+ & DataMingler
- Architecture and Implementation – Databases
- Architecture and Implementation – Data Streams
- Related Work and Future Work

Athens University of
Economics and Business

SQL: Querying Databases (1)

- SQL: everyone knows about it – CS, IT or otherwise.
- However, what is the usage pattern of SQL in industry?
  - People use SQL to synthesize information into "views" (selections, projections, joins).
  - People *do not* use SQL to analyze/process information – they use cursors, programming languages, DB connectivity interfaces & platforms (finance, medical informatics, direct marketing, etc.)

Athens University of
Economics and Business

SQL: Querying Databases (2)

- Why?
  - Specialized data processing
  - Performance (optimization failures, scalability)
  - Procedural "inclination" (declarative ➔ maturity)
- In the last few years: MapReduce, Hive, Pig, Hadoop, etc. (programming platforms)
- Punchline 1: SQL is great and its here to stay. But complex analytics over different data sources require novel querying approaches.

Athens University of
Economics and Business

Group By

- Approach 1 (traditional): model it as a relational operator, have `group by` clause, etc.
- Approach 2 (procedural):

```
for each value v in V {
      define subset S_v of R (R.V = v);
      aggregate subset S_v;
}
```

- Formed "groups" are not mutually disjoint any longer (e.g. R.V <> v)

Athens University of
Economics and Business

Grouping Variables

- Grouping Variables [10]: for each value of the grouping attributes, define a sequence of subsets of the groups (grouping variables) and aggregate over these (subsets may be correlated).
- Grouping Variables++ [11,12,13]: same concept, but grouping variables are not limited within the group, i.e. can be subsets of any relation.

Athens University of
Economics and Business

MapReduce

- Consists of two phases:
  - Map phase: create (key, value) pairs
  - Reduce phase: group pairs on the same key and apply some aggregate function on the values.
- Idea: Form subsets represented by a value (key) and reduce them.
- Punchline 2: A subset-formation language seems useful in complex data analysis (expression + performance)

Athens University of
Economics and Business

Outline

- Introduction
- *Motivation Part I: Large Data Analytics (SocNets)*
- Motivation Part II: Data Streams (Finance)
- Expressing ASSET Queries: SQL+ & DataMingler
- Architecture and Implementation – Databases
- Architecture and Implementation – Data Streams
- Related Work and Future Work

Athens University of
Economics and Business

A Social Net with a Streaming Service

```
VideoPageViews (userid, sessionid, videoid, timespent,...)
    Users (userid, type, age, country, financial, ...)
Videos (videoid, categ, videotype, ownerid, duration, ...)
```

- Interesting Queries:
  - **Q1:** For each video compare income of viewers watching less vs. more than 50% of the video's duration
  - **Q2:** For each video compute income demographics of the viewers and compare to those of video's category
  - **Q3:** For each user over 25, find the most frequent category of videos the user watches

Athens University of
Economics and Business

Common Pattern: Subset Formation

- These queries share a pattern: for each tuple of a table, they compute a "define-subset, reduce-subset" sequence, where each "define-subset" phase of a step uses previously defined aggregates and/or the tuple's attributes.
- Examples: Pivoting, Hierarchical Comparisons, Trends, Correlated Aggregation, etc.

Athens University of
Economics and Business

Example Q3 – SQL Syntax

- **Q3:** For each user over 25, find the most frequent category of videos the user watches.

```
create view UC (userid, categ, cnt) as
select u.userid, s.categ, count(*)
from VideoPageViews v, Users u, Videos s
where v.userid=u.userid and
      s.videoid=v.videoid and u.age>25
group by u.userid, s.categ;
```

```
select UC.userid, UC.categ
from (select userid,
             max(cnt) as max_cnt
      from UC
      group by userid) as G,UC
where UC.userid=G.userid and
UC.cnt=G.max_cnt;
```

- While SQL is the main option, one could formulate it in a stepwise, set-oriented fashion.

Athens University of
Economics and Business

Example Q3 – Stepwise, Set-Oriented

- **Q3:** For each user over 25, find the most frequent category of videos the user watches.

Defines a subset V of VideoPageViews

Defines an aggregate F (set-valued) over V

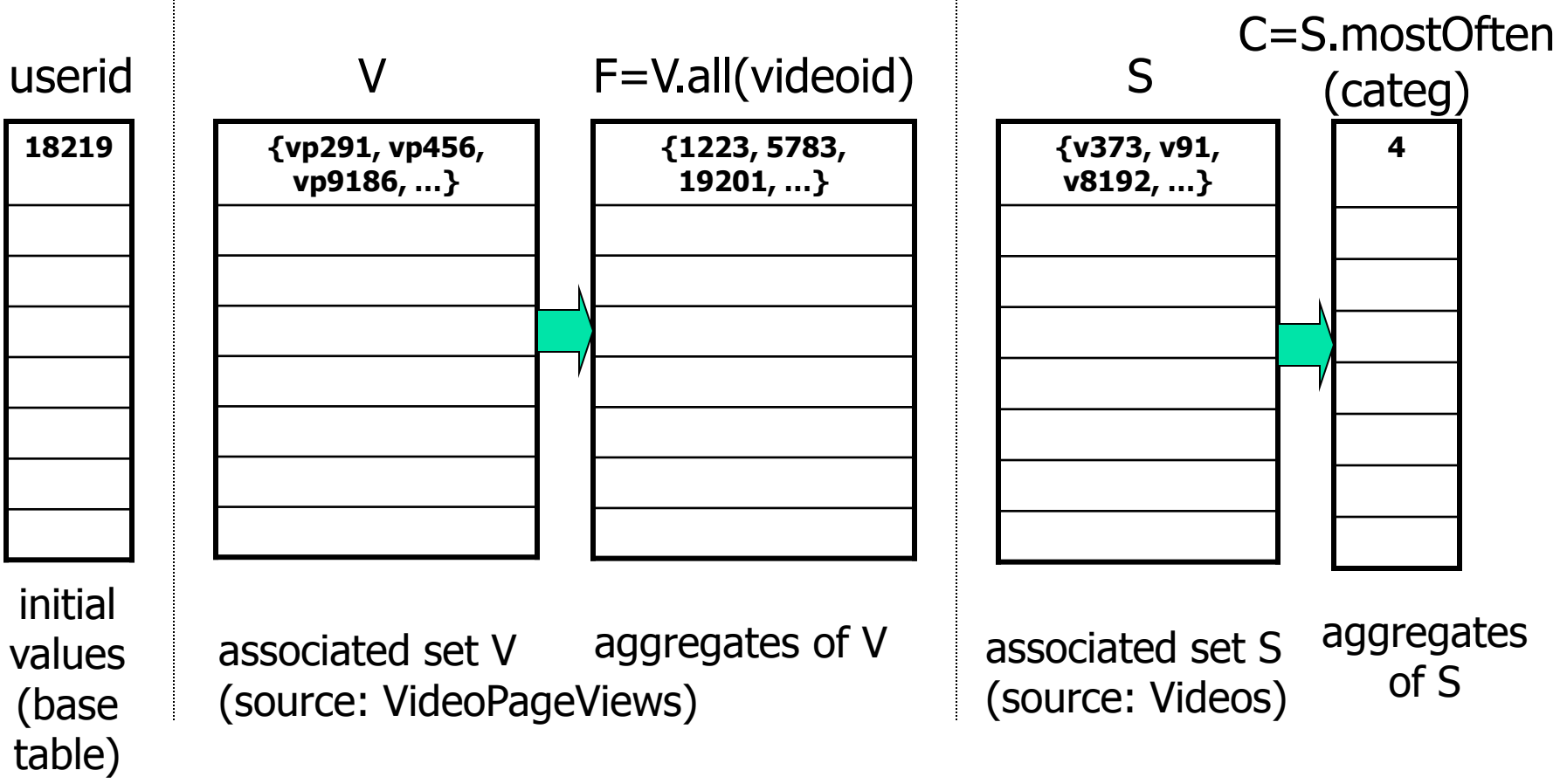
```
1:    for each user u {
2:      V={find the videopageviews of u};
3:      F={videoids of V's members};
4:      S={find the videos with videoid in F};
5:      C= mostOften(S.categ);
6:      print (u, C);
7:    }
```

Defines a subset S of Videos

Defines an aggregate C over S

Athens University of
Economics and Business

Example Q3 – Arrange in Columns



Athens University of
Economics and Business

Remarks (1)

- Initial values: usually a relation (the base table)
- An associated set has:
  - a data source (not necessarily a relation)
  - a defining (populating) condition (C++)
  - aggregates (C++)
- There may be dependencies between associated sets
  - Associated set S uses V's aggregates in its defining condition
- Such a query resembles a spreadsheet document (columns may use formulas involving previous columns)
- A practical and useful class of data analysis queries

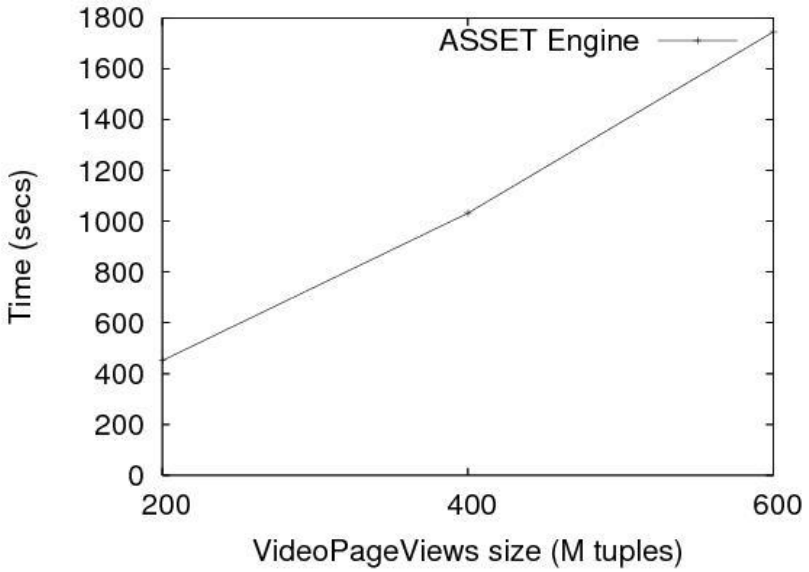
Athens University of
Economics and Business

Remarks (2)

- Representation:
  - description of the query in a spreadsheet-like format (a tabular assignment of data structures)
  - optimizations:
    - choice of data structures, e.g. inverted lists for F
    - indexing, e.g. hash index for userids
  - distributed processing:
    - send initial values to nodes, get partial results, "roll-up"
    - invariant: recursive definition of base table

Example Q3 – Performance

- Varying VideoPageViews size from 100M to 600M records (15GB to 90GB)–one partition
- We assumed 10M users and 10K videos. Using standard SQL on PostgreSQL did not return any results for 200M records for at least 2 hours.
- Linux Dell machine with a Quad Core Intel Xeon CPU@ 3.00GHz having 12 disks, 300GB each at 15K rpm, RAID5 configuration and 32GB of main memory.



Athens University of
Economics and Business

Outline

- Introduction
- Motivation Part I: Large Data Analytics (SocNets)
- *Motivation Part II: Data Streams (Finance)*
- Expressing ASSET Queries: SQL+ & DataMingler
- Architecture and Implementation – Databases
- Architecture and Implementation – Data Streams
- Related Work and Future Work

Athens University of
Economics and Business

Real-Time Stock Analysis

Relation:

Stocks(stockID, categoryID, description)

Data streams:

Prices(stockID, timestamp, price)

Volumes(stockID, volume, timestamp)

- Interesting Queries:
  - **F1:** Monitor for each stock the running min, max and average price
  - **F2:** We want for each stock to continuously know when its average price of the last 10 reported prices is greater than its running avg price
  - **F3:** Monitor the running total volume of each stock, but summation should take place only when the average price of the last 10 reported prices is greater than the running average price of the stock. Then, we want to contrast this with the (regular) running total volume

Athens University of
Economics and Business

Example F2

- **F2:** We want for each stock to continuously know when its average price of the last 10 reported prices is greater than its running average price

Defines set X containing stock prices

Defines set Y containing the last 10 reported prices

```
1:    for each stockID s in Stocks {
2:      X={v in Prices: v.stockID==s};
3:      Y={v in Prices: v.stockID==s and Y.size(10)};
4:      compute(Y.avg(price)>X.avg(price));
```

Defines aggregates over X and Y

Athens University of
Economics and Business

Outline

- Introduction
- Motivation Part I: Large Data Analytics (SocNets)
- Motivation Part II: Data Streams (Finance)
- *Expressing ASSET Queries: SQL+ & DataMingler*
- Architecture and Implementation – Databases
- Architecture and Implementation – Data Streams
- Related Work and Future Work

Athens University of
Economics and Business

What is an Associated Set (ASSET)?

- Just a collection of data structures, each one associated with a value of a domain.
- Definition: Given a relation B and a data source S with relational schema *S*, then any set A = {S_b: b∈B, S_b a multiset of tuples having schema *S*} is called an associated set with schema (B, *S*) . B is called the base relation of A and S the data source of A.

Athens University of
Economics and Business

What is an ASSET Query?

- Incrementally build ASSET queries (recursive definition):

1: B_0 a relation (initial base table);
2: $B = B_0$;
3: Define $A_1, A_2, \ldots, A_n$ associated sets having base table B;
4: $B' = B$ extended by the aggregates of $A_1, A_2, \ldots, A_n$;
5: $B = B'$;
6: Goto step 1 if not done;

Athens University of
Economics and Business

SQL Extensions

- We extend SQL with a new clause, **`extended by`** and **`such that`**, following the formalism of grouping variables

```
select L from R where θ group by A
extended by A1(S1), A2(S2), … , An(Sn)
such that θ1, θ2, ..., θn
```

A_i : associated set name

S_i : data source of associated set Ai

θ_i : defining condition of associated set Ai

L list may include associated sets' aggregates

Athens University of
Economics and Business

Extended SQL Syntax – Q3

```
select userid, mostOften(Y.categ)
from Users
where age>25
extended by X(VideoPageViews), Y(Videos)
such that X.userid=userid,
          Y.videoid in X.all(videoid)
```

Athens University of
Economics and Business

DataMingler – A Spreadsheet-Like GUI

- While an SQL extension is a mandate to our SQL-centric universe, spreadsheet-like query tools have been praised for their simplicity and flexibility [25,26,27].
- DataMingler, manages data sources, user-defined aggregate functions and ASSET queries.
- It has been implemented in C++ code using the Qt4 C++ library from Trolltech (platform independent).

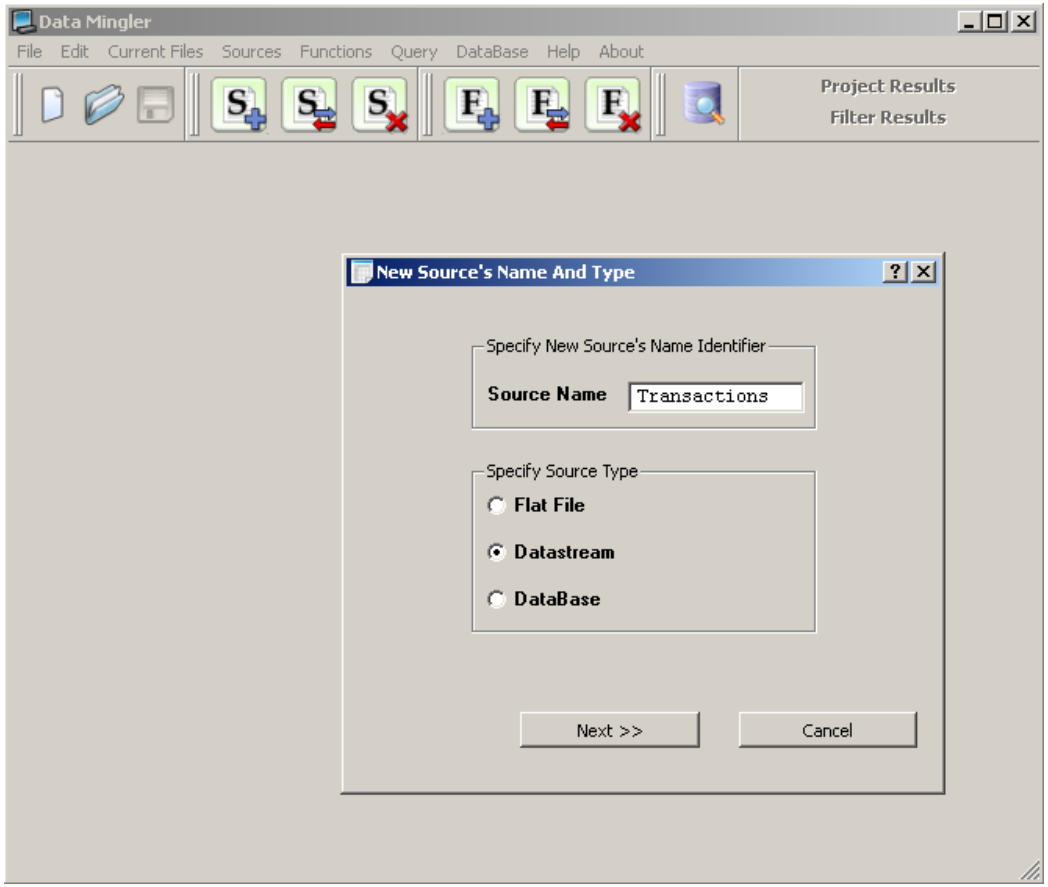
Athens University of
Economics and Business

DataMingler – Data Sources (1)

- Data sources can be added/modifed/deleted using DataMingler. Each description consists of the source's schema and a number of attributes specific to the type of the source (e.g. delimiter and location for flat files; IP, port, username and password for databases, etc.)
- All data sources' descriptions are stored in an XML-based specification file.
- Currently, DataMingler support databases (PostgreSQL, MySQL), flat files and socket-based streams. All data sources may consist of multiple partitions, not necessarily of the same schema – only common attributes appear in query formulation.

Athens University of
Economics and Business

DataMingler – Data Sources (2)



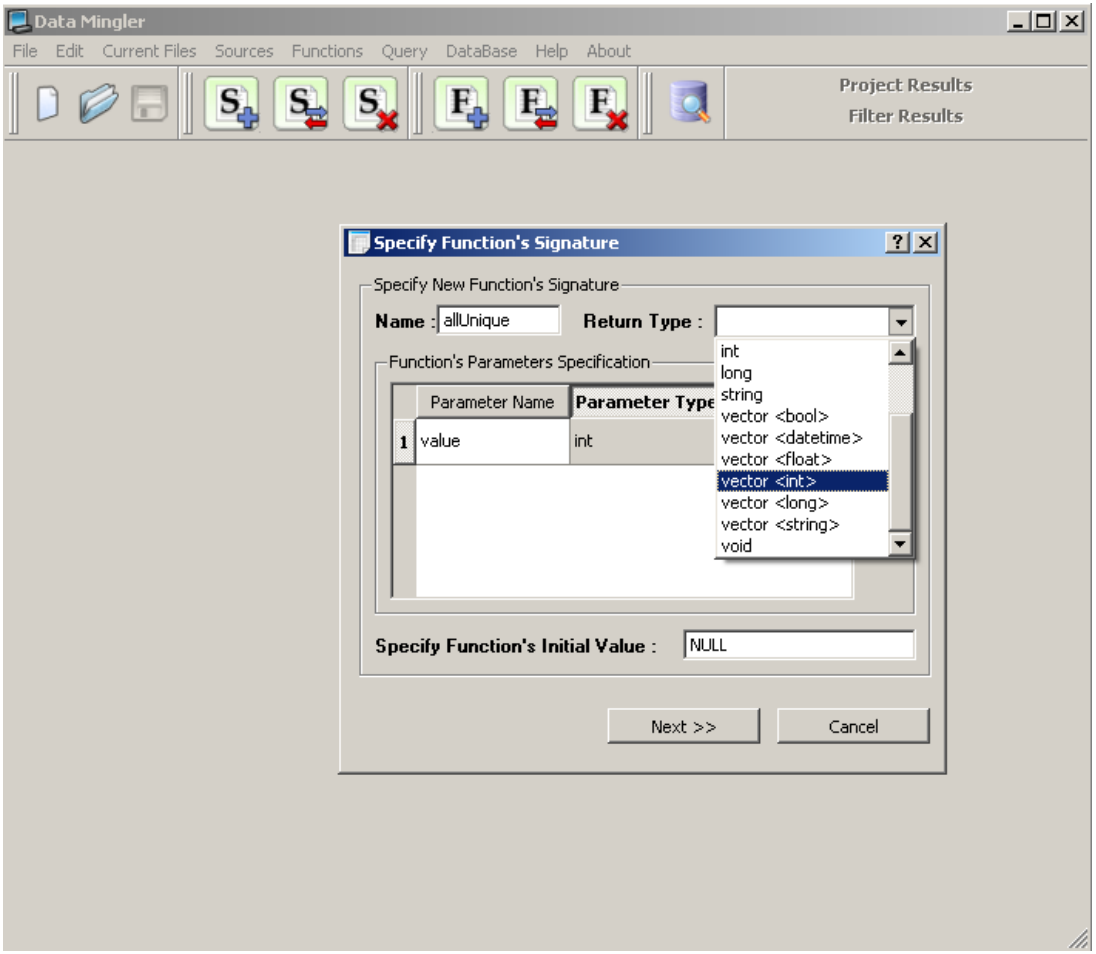
Athens University of
Economics and Business

DataMingler – Aggregate Functions (1)

- The goal is to describe the signature of a C++ function, so some type-checking and user-guidance can take place. The user specifies the input parameters and their types and the type of the return value. S/he also specifies a "roll-up" function, in the case of distributed computation of an associated set.
- Aggregate functions can be either distributive or algebraic. Holistic computations can be achieved through aggregate functions returning the entire or part of the associated set and the use of null-source associated sets.

Athens University of
Economics and Business

DataMingler – Aggregate Functions (2)

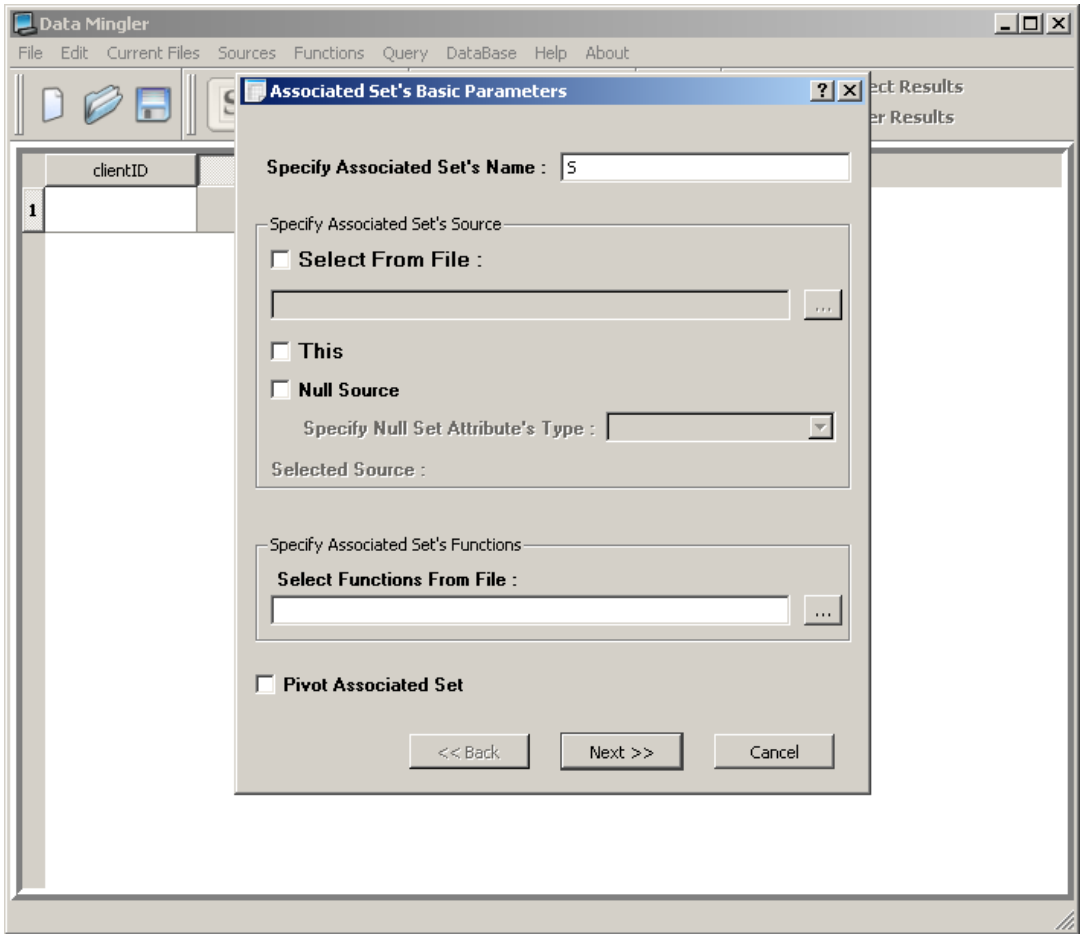


Athens University of
Economics and Business

DataMingler – ASSET Queries (1)

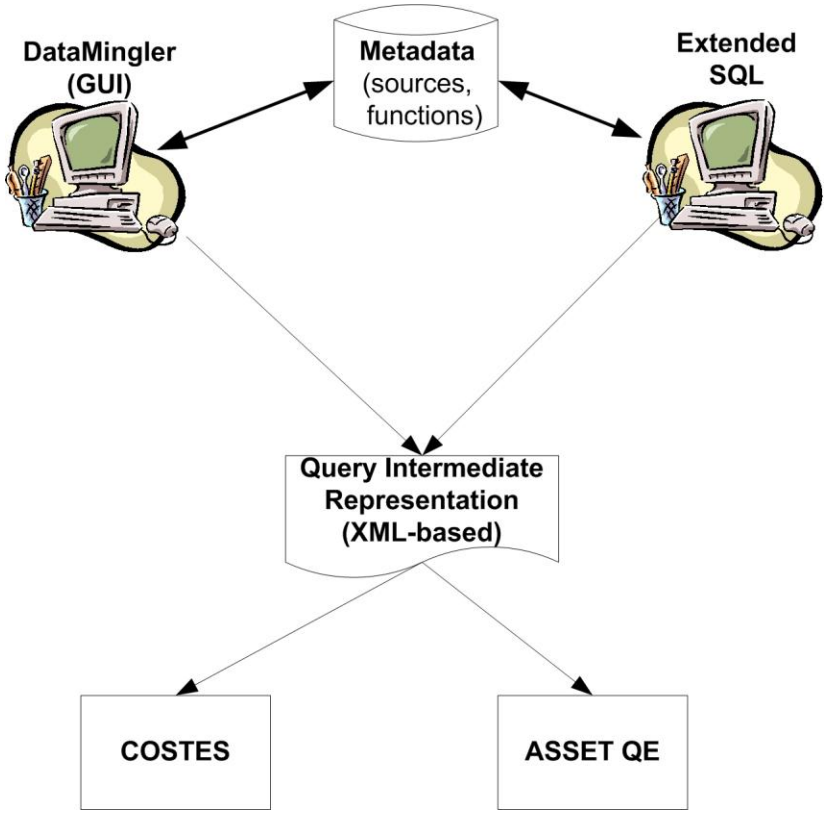
- Users specify ASSET Queries through DataMingler in a spreadsheet-like manner, column by column:
  - The user initially specifies a base-values table that can be an SQL query over one of the database sources, the contents of a flat file source or manually inserted schema and values.
  - The spreadsheet is extended with columns representing associated sets, one at a time. The user specifies the name, source, defining condition and aggregate functions.
  - The data source can be (a) one of the existing data sources described earlier through DataMingler, (b) of type "this", in which case the so-far defined spreadsheet table serves as the data source to the associated set, and (c) of type "null", in which case the user just specifies an expression involving aggregates of previously defined columns.

DataMingler – ASSET Queries (2)



Athens University of
Economics and Business

Query Formulation



Athens University of
Economics and Business

Outline

- Introduction
- Motivation Part I: Large Data Analytics (SocNets)
- Motivation Part II: Data Streams (Finance)
- Expressing ASSET Queries: SQL+ & DataMingler
- *Architecture and Implementation – Databases*
- Architecture and Implementation – Data Streams
- Related Work and Future Work

Athens University of
Economics and Business

ASSET Query Engine (ASSET QE)

- Mention:
  - ASSET QE generates C++ code
  - An ASSET query is represented in a tabular format (ASSET structure)
  - Main idea: scan and fill the ASSET structure
  - Main processing:
    - Assign associated sets to computational rounds (based on dependencies)
    - For each round, assign associated sets to sources (actually partitions)

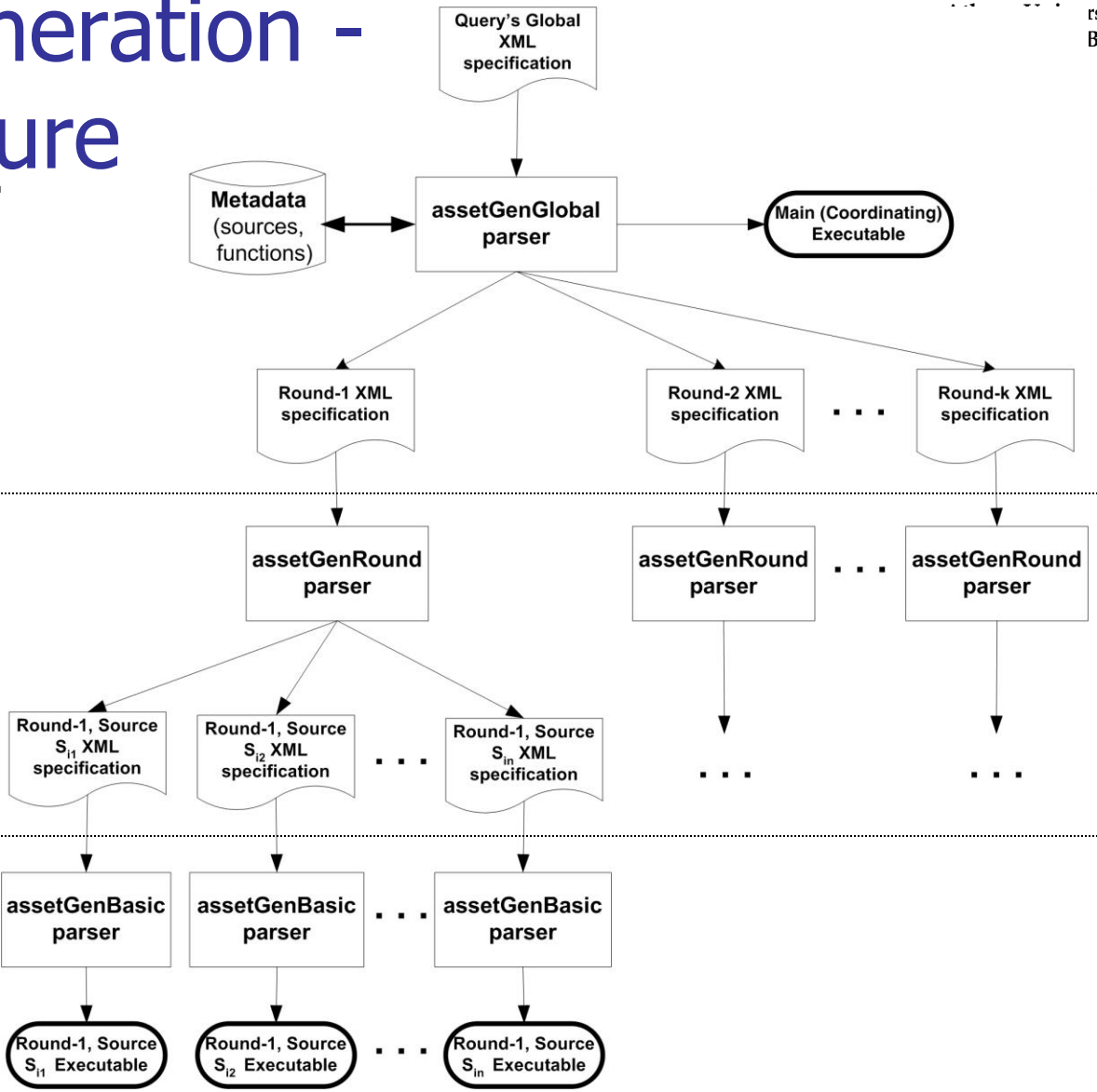
rsity of Business

Code Generation - Architecture

Top Level

Round Level

Source (Partition) Level



Athens University of
Economics and Business

assetGenGlobal Parser

- Top-level parser of the ASSET QE. It gets the XML-based specification of an ASSET query and generates:
  - the round-related XML specifications of the query, and
  - the main (coordinating) C++ program for the query.
- Each round-related XML specification contains the data sources' description of the round and the associated sets that will be computed.
- The query's main C++ program, instantiates and populates all the necessary data structures, creates all the local indexes and decorrelation lists over the ASSET structure and coordinates all the basic computational threads executing locally or remotely.

Athens University of
Economics and Business

`assetGenRound` Parser

- Round-level parser. It assigns the associated sets of the round by source and generates an XML-based specification file for each source. *A source at this point means partition of a data source.*
- It determines whether the computation over the source will execute locally or remotely, deduces the indexes and decorrelation lists over the base table and resolves the minimal base table that has to be sent to the remote node (in case of remote computation.)
- Currently supported indexes are hash maps, binary trees and inverted lists, deduced by the defining condition of the associated sets.

Athens University of
Economics and Business

assetGenBasic Parser

- Source-level parser (partition). It gets a source-specific XML-based specification file and generates an efficient C++ program (the "basic computational thread") to scan the data source and compute the associated sets related to that source.
- This thread communicates with the main program to receive the round-specific base table (only the required columns), builds indexes and decorrelation lists over the base table, computes the associated sets and serializes the result back to the coordinating program (if executing remotely).

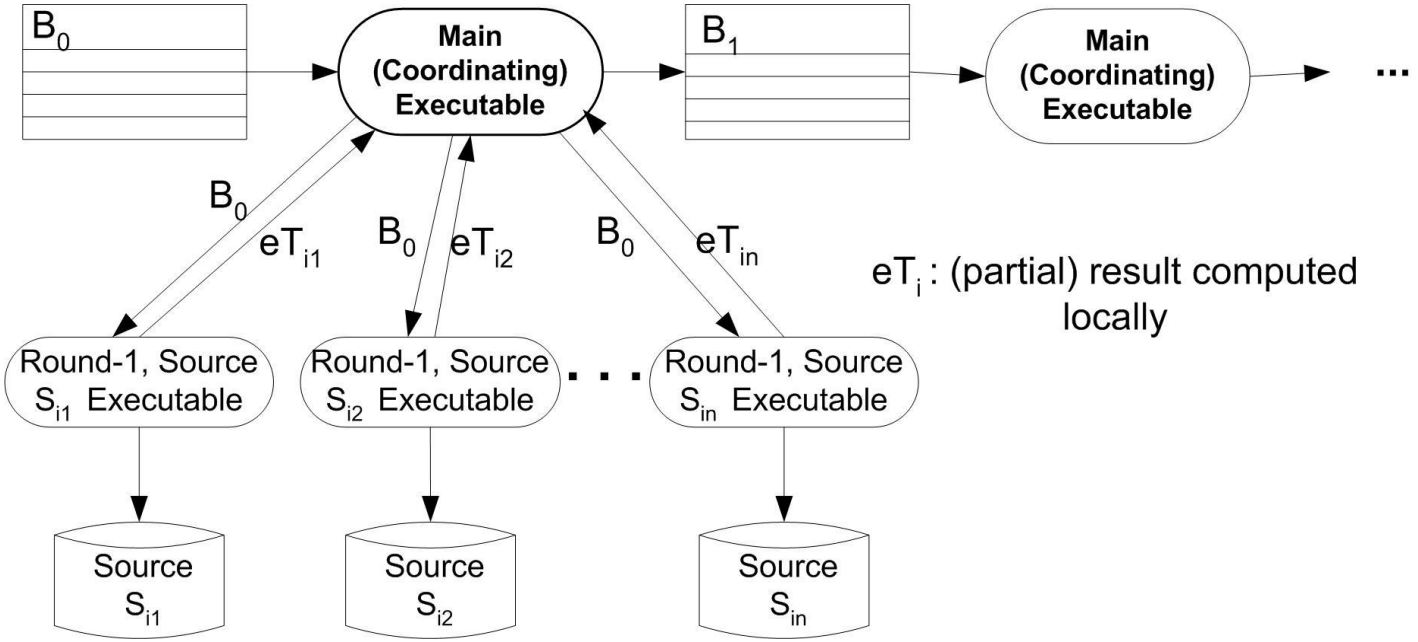
Athens University of
Economics and Business

Execution Plan (1)

- Once all the basic computational threads have been generated, then the whole process is driven by the query's main C++ program.
- The assumption is that the entire ASSET structure (the output of the ASSET query) fits in main memory – which is not unrealistic for a large class of ASSET queries and today's memory sizes.
- However, since the entire code generation assumes boundary limits of the ASSET structure, one can easily specify the computation of an ASSET query in horizontal chunks - currently has to be done manually, by altering the query's main C++ program.

Execution Plan (2)

- Simple execution plan



Athens University of
Economics and Business

Execution Plan (3)

- This plain execution strategy can be generalized to a more elaborate one, where base tables are horizontally partitioned according to their estimated sizes and the computation of the ASSET query "flows" from left to right, possibly in parallel.
- In general, one can think of an evaluation strategy represented by a graph with fork and join points at the end of computational rounds.

Athens University of
Economics and Business

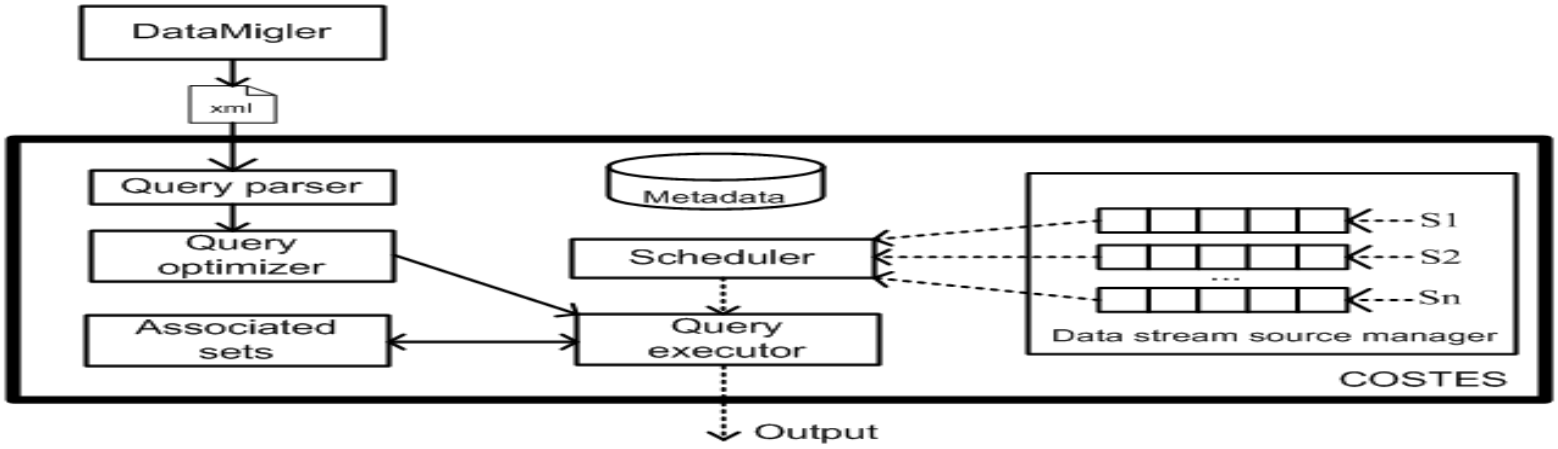
Outline

- Introduction
- Motivation Part I: Large Data Analytics (SocNets)
- Motivation Part II: Data Streams (Finance)
- Expressing ASSET Queries: SQL+ & DataMingler
- Architecture and Implementation – Databases
- *Architecture and Implementation – Data Streams*
- Related Work and Future Work

Athens University of
Economics and Business

COSTES: Continuous Spreadsheet-like Computations

- **Features:**
  - Ad-hoc continuous query declaration
  - Real time results
  - Results in a spreadsheet-like report
  - Create/Modify/Save/Reload query
  - Logical and time window support
  - Support multiple data sources
  - Multiplatform support (wxWidgets C++ implementation platform)



Athens University of
Economics and Business

COSTES Modules (1)

- **Query Input:** Users can use a textual interface to write the query or formulate it using the DataMigler
- **Query Parser:** validates the query and store information in the metadata catalog (e.g. base table schema, data source schema)
- **Data Stream Source Manager (DSSM):** is a multithreaded module allowing concurrent data retrieval from many data sources. Currently our system supports as finite data sources:
  - Flat files
  - Databases (ODBC)
  - Web sources (HTTP protocol)

Athens University of
Economics and Business

COSTES Modules (2)

- **Query Optimizer:** analyzes the query and identify the possibility to create an index. For example:
  - Determine index creation according to *such that* predicate (e.g. in equality conditions to quickly locate rows of the base table and avoid a full scan)
  - Process query to keep in corresponding associated sets only the attributes needed for the evaluation of aggregation functions and avoid holding all the attributes of stream tuples
- **Scheduler:** retrieves stream tuples from DSSM queues in a round robin fashion and forwards them to Query Executor
- **Associated Sets Manager:** build and handle suitable data structures for associated sets. Currently our system supports logical and time windows

Athens University of
Economics and Business

COSTES Modules (3)

- **Query Executor:**
  - Evaluates the query taking into consideration optimizations that have been determined by the Query Optimizer module
  - Allocates the initial base table
  - Builds index structures according to optimization parameters
  - Interacts with the Scheduler to receive stream tuples and with the Associated Sets Manager to access the associated set structures
  - Maintains results in a memory resident data structure (i.e. ASSET query result)

Athens University of
Economics and Business

Outline

- Introduction
- Motivation Part I: Large Data Analytics (SocNets)
- Motivation Part II: Data Streams (Finance)
- Expressing ASSET Queries: SQL+ & DataMingler
- Architecture and Implementation – Databases
- Architecture and Implementation – Data Streams
- *Related Work and Future Work*

Athens University of
Economics and Business

Related Work (1)

- Grouping Variables [10,11,12,13,14,15]
  - Chatziantoniou, D., Ross, K.A.: Querying Multiple Features of Groups in Relational Databases. In: 22nd VLDB, pp. 295--306. Morgan Kaufmann, (1996)
  - Chatziantoniou, D.: Using grouping variables to express complex decision support queries. DKE. 61, 114--136 (2007)
  - Chatziantoniou, D.: Evaluation of Ad Hoc OLAP: In-Place Computation. In: 11th SSDBM, pp.34—43. IEEE Computer Society, (1999)
  - Chatziantoniou D.: The PanQ Tool and EMF SQL for Complex Data Management. In: 5th ACM SIGKDD, pp.420—424. ACM (1999)
  - Chatziantoniou, D., Akinde, M.O., Johnson, T., Kim, S.: The MD-join: An Operator for Complex OLAP. In: 17th ICDE, pp. 524—533. IEEE Computer Society, (2001)
  - Akinde, M.O., Böhlen, M.H., Johnson, T., Lakshmanan, L.V.S., Srivastava, D.: Efficient OLAP Query Processing in Distributed Data Warehouses. In: EDBT 2002. LNCS, vol. 2287 , pp. 336--353. Springer, (2002)

Athens University of
Economics and Business

Related Work (2)

- MapReduce [18,19,20,21,22]
  - Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: 6th Symposium on Operating System Design and Implementation, pp. 137—150. USENIX Association, (2004)
  - DeWitt, D.J, Stonebraker, M.: MapReduce: A major step backwards. The Database Column, http://www. databasecolumn.com/2008/01/mapreduce-a-major-step-back.html.
  - Pavlo, A., et al.: A Comparison of Approaches to Large-Scale Data Analysis. In ACM SIGMOD Conference, pp. 165--178. ACM (2009)
  - Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A: Pig latin: a not-so-foreign Language for Data Processing. In: SIGMOD Conference, 2008, pp. 1099-1110. ACM (2008)
  - Abouzeid, A., et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Worksloads. In VLDB Conference, 2009 (to appear)

Athens University of
Economics and Business

Related Work (3)

- Spreadsheets [25,26,27]
  - Winslett, M.: Interview with Jim Gray. SIGMOD Record 32, 53--61 (2003)
  - Witkowski, A., Bellamkonda, S., Bozkaya, T., Dorman, G., Folkert, N., Gupta, A., Sheng, L., Subramanian, S.: Spreadsheets in RDBMS for OLAP. In: ACM SIGMOD International Conference on Management of Data, pp. 52--63. ACM (2003)
  - Liu, B., H.V. Jagadish, H.V.: A Spreadsheet Algebra for a Direct Data Manipulation Query Interface. In: 25th International Conference on Data Engineering, pp. 417--428. IEEE (2009)
- Query Optimization, OLAP [all the rest]

Athens University of
Economics and Business

Future Work

- More work/testing is needed on the distributed version of ASSET QE.
- Much more work on data streams: it seems an excellent interface to specify stream queries.
- We believe that ASSET queries show promise both linguistically and computationally. Little has been done in terms of theoretical work though, since the focus was rapid prototyping.

Athens University of
Economics and Business

Conclusions

- Move from traditional SQL/DB to programmable platforms. ASSET queries can be thought as another proposal in that approach.
- ASSET – one structure consisting of data structures – scan data sources and update it – the bottleneck is CPU!
- We consider it an interesting and useful class of queries.

Athens University of
Economics and Business

Acknowledgments

- European Union ICT Grant ST-5-034957-STP.
- Greek Secretariat of Research and Technology PAVE Grant 05/184.
- Part of the work (ASSET QE) has been performed while the first author was visiting AsterData Systems Inc., Redwood Shores, CA.