τ-xSynopses - A System for Run-time Management of XML Synopses

Natasha Drukh
School of Computer Science
Tel Aviv University
kreimern@cs.tau.ac.il

Yossi Matias
School of Computer Science
Tel Aviv University
matias@cs.tau.ac.il

Leon Portman
School of Computer Science
Tel Aviv University
leonpo@cs.tau.ac.il

January 18, 2005

Data synopses are concise representations of data sets, that enable effective processing of approximate queries to the data sets. Approximate query processing provides important alternatives when exact query answers are not required. τ-Synopses [23] is a system designed to provide a run-time environment for remote execution of various synopses for relational databases. Our work introduces the τ-xSynopses, a system designed to support remote execution of synopses for XML databases. The τ-xSynopses system extends the design and the implementation of the τ-Synopses, enabling a single environment for managing synopses for both XML and relational databases. The system enables easy registration of new synopses from remote platforms, after which the system can manage these synopses, including triggering their construction, rebuild and update, and invoking them for approximate query processing. The system can also serve as a research platform for experimental evaluation and comparison of different synopses.

1 Motivation

In large data recording and warehousing environments, it is often advantageous to provide fast, approximate answers to queries, whenever possible. The goal is to provide a quick response in orders of magnitude less time than the time to compute an exact answer, by avoiding or minimizing the number of accesses to the base data.

Techniques for fast approximate answers can also be used in a more traditional role within a query optimizer to estimate plan costs, again with very fast response time.

Approximate query processing is supported by synopses that are compact representation of the original data, such as histograms, splines, sampling, wavelets or other methods.

Increased interest in approximate query processing resulted with proliferation of new synopses addressing new problems as well as proposed alternatives to previously suggested synopses. In particular, synopses are becoming more advanced, supporting updates to data, awareness to workload, and adaptive to changes in workload (e.g., [1, 5, 6, 7, 10, 11, 18]).

Several systems and projects address approximate query processing and data synopses. In the AQUA Project [12, 3, 2], synopses are precomputed and stored in a DBMS. It provides approximate answers by rewriting the queries to run on these synopses, and enables keeping synopses up-to-date as the database changes. Various aspects of approximate query processing were studied by Microsoft Research DB group (e.g., [5, 7]). The question of how to reconcile various synopses for large information sources with many tables was studied in [14, 13].

While the above research works deal mostly with relational database systems, we see increasing interest towards XML synopses. The extensible mark-up language (XML) is becoming ubiquitous as a data exchange and storage format. Almost all commercial RDBMSs include support for XML data; native XML databases such as Xyleme [22] are specially designed to store and query XML data on the web. Efficient query processing over XML data requires accurate estimation of the selectivities of the path expressions contained in the query. Thus, maintaining concise synopses structures is crucial for the XML databases as well as for the traditional relational databases. This problem has recently attracted the attention of the database research community, and several techniques [15, 19, 20, 9] have been proposed targeting different aspects of the problem.

Operational systems require the management and consolidation of many synopses. These include various synopses addressing different types of queries, each requiring its own type of synopsis; furthermore, even for the same type of query, many different instances of the same synopsis should be used to represent different data sets (e.g., different relations, or different columns of the same relation). Finally, in some cases it would be beneficial to hold more than one synopsis for the same type of queries, to benefit from the different properties of the various synopses (e.g., having an answer based on all supporting synopses, letting each synopsis support a particular subset of the queries etc).

For research purposes, it would also be beneficial to support many synopses. Indeed, for each query there may be different possible types of synopses that could be useful. Furthermore, for each particular synopsis, it would be useful to compare different implementations for evaluation and benchmarking purposes. Therefore, it is advantageous to have a system that can accommodate *multiple synopses*, and have an easy way to integrate new synopses and manage them.

The multiple synopses in use in either operational or research system could be placed in remote locations for various reasons: they may be implemented on different types of platforms, they may be summarizing remote data whose transfer is undesirable or impossible due to performance or security constraints, and it would be beneficial to share the load of operating a large number of synopses using different systems for load balancing and redundancy reasons. Therefore, it would be beneficial to have a system support *remote execution* of registered synopses.

Synopses are more effective when they are adaptive for predicted workload, changes in workload, changes in data, and changes in performance requirements (query time, accuracy, confidence, and available memory). This motivates the use of a system of *managed synopses*. This is a single system that registers all synopses, triggers their construction and maintenance (in response to new data, data changes, predicted changed workload, or by demand). Having a managed synopses system enables providing information required by all relevant synopses from a single repository (e.g., data, workload, changes in data and workload).

2 τ-Synopses framework overview

Motivated by the above, the τ-Synopses system [23] was designed to provide a run-time environment for remote execution of various synopses. It enables easy registration of new synopses from remote SOAP-enabled platforms, after which the system can manage these synopses, including triggering their construction, rebuild and update, and invoking them for approximate query processing. The system captures and analyzes query workloads, enabling its registered synopses to significantly boost their effectiveness (efficiency, accuracy, confidence), by exploiting workload information for synopses construction and update. The system can serve as a research platform for experimental evaluation and comparison of different synopses.

Initially, the τ-Synopses system provided a run-time environment for remote execution of synopses for relational databases only. With evolving of various XML synopses techniques, there was an increasing need of providing a similar system for managing synopses for XML data sources. In our work, we have proposed and implemented the τ-xSynopses system, a system designed to support remote execution of synopses for XML databases. The τ-xSynopses system was implemented by utilizing the τ-Synopses design and infrastructure, thus extending the τ-Synopses framework to support also XML data sources and manage XML synopses for approximate selectivity estimations. In what follows we describe the τ-Synopses essentials that were utilized by us and then focus on the τ-xSynopses specifics.

The τ-Synopses system supports two types of users: synopses providers who register their synopses within the system, and end-users who send queries to the system. The system administrator defines available data sources and provides general administration of the system.

When a new synopsis is registered, the relevant data set and the supported queries are defined. A query submitted to the system is executed using the appropriate synopsis, based on the registration and other information. The result is returned to the user or optionally processed by other modules in the system. The system transforms updated data from its original data source to be consistent with the format known to the synopses, so that synopses are not required to support any data transformation functionality or database connectivity logic. Any relational/xml database or even real-time data providers can be data sources in the system.

The τ-Synopses system has the following key features:

- *various data sources*: The system supports both relational and XML data sources.
- *multiple synopses*: The system can accommodate various types of synopses. New synopses can be added with their defined functionalities.
- *pluggable integration*: For integration purposes, a synopsis has to implement a simple interface, regardless of its internal implementation. By utilizing a light-weight host provided by the system, the synopsis can be executed on any SOAP-enabled platform.
- *remote execution*: Synopses can be transparently executed on remote machines, over TCP/IP or HTTP protocols, within local area networks or over the internet.
- *managed synopses*: The system allocates resources to synopses, triggers their construction and maintenance, selects appropriate synopses for execution, and provides all required data to the various synopses.
- *workload support*: Workload is captured, maintained and analyzed in a centralized location, and made available to the various synopses for construction and maintenance.
- *research platform*: The system provides a single, consistent source of data, training and test workload for experimental comparison and benchmarking, as well as performance measurements. It can therefore serve as an effective research platform for comparing different synopses without re-implementing them.

The rest of the paper is organized as follows. Section 3 supplies the necessary background for query optimization and approximate queries in both relational and XML databases. Section 4 reveals in details the functionality of the τ-xSynopses framework. Software engineering aspects are discussed in Sections 5 and 6. We present our experimental evaluation in Section 7 and draw conclusions in Section 8. Finally, Section 9 provides all the necessary guidelines to operate the system.

3 Background

This section supplies the necessary background for query optimization and approximate queries in both relational and XML databases.

3.1 Relational Databases

Approximate queries. So, what is fast approximate answers? In relational databases it used primarily for aggregate queries, calculation of which requires access to large part of the original data. The primary goal is to quickly report for example leading digits of answers in second instead of hours. For example, a calculation of the average salary over database of all people in US will take about 10 min, but approximate answers can be provided in 10 sec. The approximate answers can be achieved by answering the query based on some synopses of data that are orders of magnitude smaller than the original data.

Synopses. Approximate query processing is supported by synopses that are compact representation of the original data. To handle many base tables and many types of queries, a large number of the synopses may be needed. Moreover, for fast response times that avoid disk access altogether, synopses that are frequently used for query processing should remain in the main memory. Thus we evaluate effectiveness of a synopsis as a function of its size. The effectiveness of a synopsis can be measured by the *accuracy* of the answers it provides, and *response time*. Over the past few years there has been extensive research in data synopses, including precomputed samples, various histograms, and wavelet synopses. Some of the techniques boost accuracy of query answers by adapting synopsis structure to available workload information.

Precomputed samples are based on the use of random samples as synopses for large data sets (see [11, 1, 7] and references therein). Histograms are commonly used synopses used to capture the distribution of the data stored in a database relation and are used to guide selectivity estimation as well as approximate query processing. Many different histograms have been proposed in the literature and some have been deployed in commercial RDBMSs. Equi-depth histograms (e.g., [1]) are popular histograms that are easy to implement. New histograms types can be derived by combining effective aspects of different histogram methods, including compressed histograms, v-optimal histograms, and maxdiff histograms. Recently, *wavelet-based histograms*, built from a wavelet decomposition, were introduced as a mean for improved histogram accuracy [17, 21, 6]. Wavelets are a mathematical tool for hierarchical decomposition of functions, whose adaptive nature make them good candidate for a "lossy" data representation. The use of wavelet-based synopses in database has recently drawn increasing attention. One of the latest developments was the introduction of probabilistic wavelets synopses [10], a wavelet-based data reduction technique attempting to provide increased accuracy for individual approximate answers.

Aggregate queries. Queries in relational databases are typically formulated using the SQL query language. The below aggregate range query is supported by most synopses:

```
SELECT Sum(Data) FROM Relation
WHERE filter > l AND filter < h
```

The above query calculates the sum of data values in the data column in the specified range. There are number of different aggregation functions: such as count, average, maximum and minimum.

3.2 XML Databases

XML - extensible mark-up language, allows encapsulating semi-structured data in tree structured documents that might be stored either as a regular ASCII file or inside a native XML database. XML Schema provides a structure validating mechanism for XML documents.

XML [4] has rapidly evolved from a mark-up language to a de-facto standard for data exchange and integration over the web. A testament to this is the increasing volume of published XML data, together with the concurrent development of XML query processors that will allow users to tap into the vast amount of XML data available on the Internet. The successful deployment of such query processors depends crucially on the existence of high-level declarative query languages. There exist numerous proposals that cover a wide range of paradigms, but a common characteristic among all XML-language proposals is the use of *path expressions* as the basic method to access and retrieve specific elements from the XML database. A path expression essentially defines a complex navigational path, which can be predicated on the existence of sibling paths or on constraints on the values of visited elements. As a concrete example, in a bibliography database, the path expression `//author[book]/paper/sigmod/title` (which adheres to the syntax of the standard XPath language [8]) selects the set of all `title` data elements discovered by the label path `//author/paper/sigmod/title`, but only for `author` elements that have *at least one* `book` child (a condition specified by the `author[book]` branch). Selectivity of a path expression is the number of data elements retrieved by it (and not the retrieved data itself).

Similarly to relational optimization, optimizing XML queries with complex path expressions depends crucially on the existence of concise summaries that can provide effective compile-time estimates for the selectivity of these expressions over the underlying (large) graph-structured XML database. This problem has recently attracted the attention of the database research community, and several techniques [15, 19, 20, 9] have been proposed targeting different aspects of the problem. Among the techniques there are graph-structured solutions, markov based synopses and schema-based synopses. The synopses differ on the subset of supported XPath expressions, self-tuning capabilities, etc.

4 τ-xSynopses Specification

The main operational processes supported by the τ-xSynopses are: constructing of number of pluggable synopses, constructing query workload and approximate query processing. The system currently supports file XML data sources and querying an XML data source with XPath expressions.

The user interface provides an administrator user with a capability to manage data sources, synopses specifications, updates and pre-defined workloads. The framework provides also a user management mechanism. Figure 1 depicts the main adminstration module.

- *data sources*: XML data source is specified by an absolute path to a data file and optionally a path for a validating schema file. As an XML data source represents single XML document, there is no need for any further specifications. Relational data source is specified by the database connection information (host, user, password and database instance name). With relational databases, a user must also define a data relation - the subset data of the database. It is specified by determining the target table name plus filter and data column names.

- *synopses*: A synopsis is registered by providing the location of the execution host of the synopsis (server name plus the listening port) and the data relation for which the synopsis will be built. Invoking the construction of the synopsis requires setting the size limit and optionally the training workload. Both data relation and the workload are chosen among those registered in the system.

- *workloads*: The queries workload is produced by the system based on user specifications. The users set the number of queries in the workload, the data relation to be processed and the workload construction parameters. For XML workload the parameters include the minimum and maximum lengths of the path expression and of the branching predicates, as well as the maximum number of branching

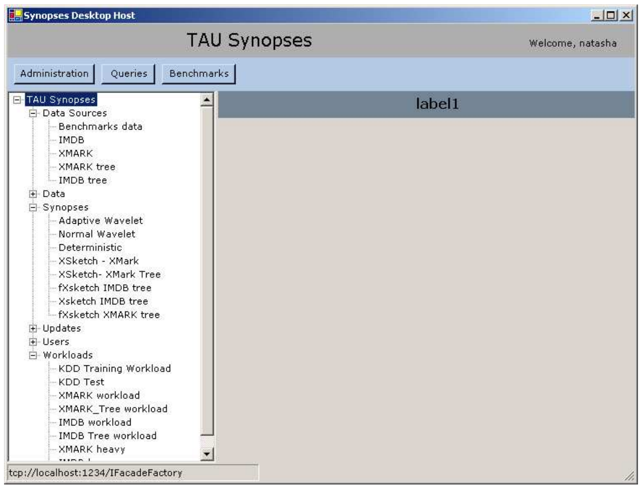
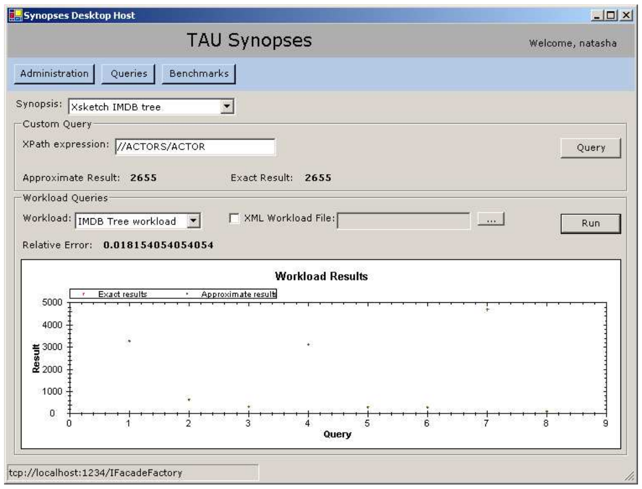


Figure 1: Administration Module



Synopsis: Xsketch IMDB tree

XPath expression: //ACTORS/ACTOR

Approximate Result: 2655 Exact Result: 2655

Workload: IMDB Tree workload

Relative Error: 0.018154054054054

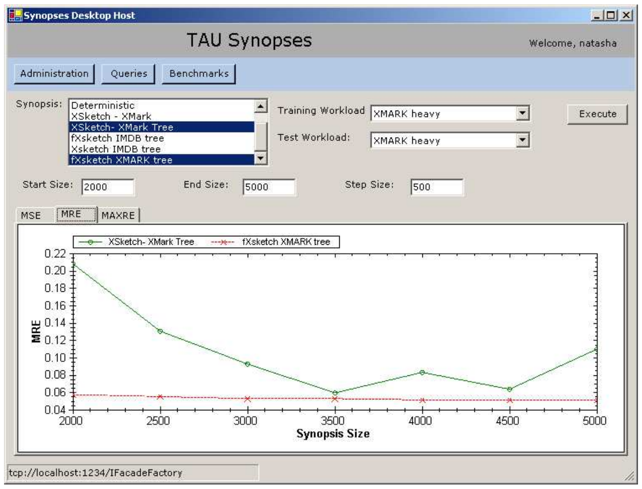
Figure 2: Query Mode

predicates and the probability of their appearance. For workloads on relational data the parameters are the zipfian skew, low and high values, focus and range values.

The end-user is supplied with the capability to test and compare different synopses that are registered in the system. The Query execution mode provides the user with the ability to evaluate single synopsis at a time, figure 2 depicts the Query mode UI. The Benchmark mode enables multiple synopses evaluation over pre-defined workloads and their comparison using visual display. The user interface for the benchmark mode is depicted in figure 3.

Query Mode: The user selects the synopsis to be evaluated. For XML synopsis, the query is an XPath expression. The system validates the user input expression for the XPath syntax validity and the tag labels are validated against existing labels of the underlying XML document. For querying the relational database, the user supplies the range query parameters.

After the query is executed (by the remote registered synopsis), the result is displayed. The system displays also the real result of evaluating the query over the data source itself; this functionality depicts the


Synopsis: XSketch- XMark Tree, fXsketch XMARK tree

Training Workload: XMARK heavy

Test Workload: XMARK heavy

Start Size: 2000 End Size: 5000 Step Size: 500

tcp://localhost:1234/IFacadeFactory

Figure 3: Benchmark Mode

accuracy of the tested synopsis and is relevant for research issues.

Query mode allows also evaluating multiple queries at a time by specifying a workload to be evaluated. The synopsis results are depicted together with the real results computed by the system.

Benchmark Mode: The user chooses the synopses to evaluate and the workload to be used for the evaluation. For the performance measurements, the minimum, maximum and step size for the synopses construction are set by the user. The system invokes construction of the different synopses, then all the synopses evaluated over the chosen workload. The system calculates the accuracy of the different synopses using several error metrics. MSE (mean square error), MRE (mean relative error) and MAXRE (maximum relative error) performance measurements are depicted in the results graph.

5 Architecture

We have adopted the τ-Synopses framework architecture for implementing the τ-xSynopses system.

5.1 Design Goals

In order to provide an effective operational and research platform τ-Synopses must commit to the following design goals.

- *research platform*- The system should provide a single, consistent source of data, training and test workload for experimental comparison and benchmarking, as well as performance measurements.
- *multiple synopses* - A system should be able to accommodate multiple synopses
- *easy integration* - To allow external user to easily integrate his synopsis, the system should provide an available, simple synopsis wrapper.
- *pluggable integration* - For integration purposes, a synopsis has to implement a simple interface, regardless of its internal implementation. By utilizing a light-weight host provided by the system, the synopsis can be executed on any SOAP-enabled platform.

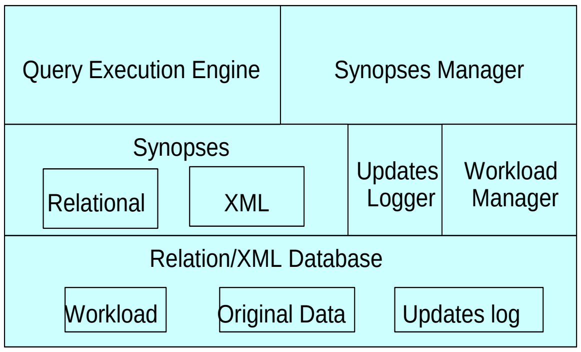


Figure 4: Synopses Framework Architecture

- *remote execution*- Synopses can be transparently executed on remote machines, over TCP/IP or HTTP protocols, within local area networks or over the internet.
- *distributed client-server environment* - The client application should reside locally on the end-station while the framework core should be centralized. In addition, the client and server deployment should be independent.
- *flexibility and scalability* - The framework design should allow easy extending of the framework for non-relational data sources.
- *privacy preservation* - The system should enable its users to protect their resources (synopses, workloads).
- *low bandwidth* - Low bandwidth support is important for efficient client-server communication and a communication with the remote synopses.

Case Study: The extended τ-Synopses framework actually hosts two independent applications - relational synopses framework and XML synopses framework. These applications are independent at first sight as they treat databases with totally different storage schema, different query languages and they even target different problems of approximate processing. Although the τ-Synopses framework initially supported synopses for relational databases only, the modularity of the framework design and implementation allowed easy adaptation of the already developed relational synopsis framework to an XML synopses framework.

5.2 High Level Design

The core of the τ-Synopses system architecture features the following components, and depicted in Figure 4: Query Execution Engine, Synopses Manager, Updates Logger, and Workload Manager. In addition, it includes a query application which is used by end-users, an administration application used by the administrator and by synopses-providers, and a pool of registered synopses.

The Synopses Manager is used for registration and maintenance of the synopses. A new synopsis is added to the system by registering its parameters (including list of supported queries and data sets) in the Synopses Manager Catalog.

The Query Execution Engine provides interface for receiving query request from end-users and invoking the appropriate synopsis (or synopses), as determined by the Synopses Manager in order to process such query.

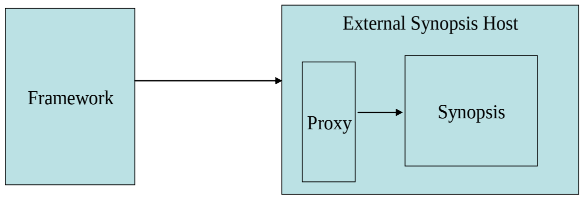


Figure 5: Synopses Integration

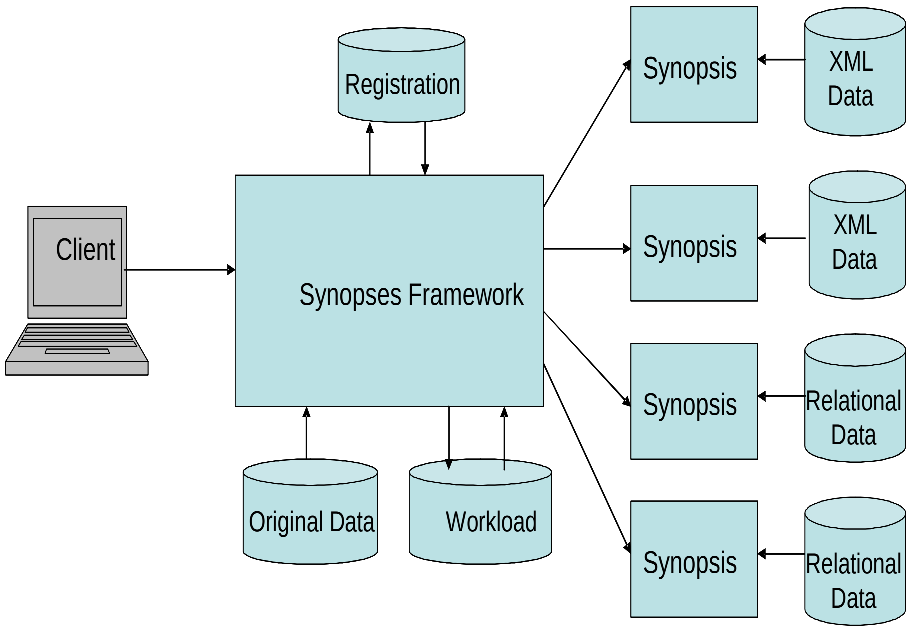


Figure 6: Distributed Environment

The Updates Logger provides all data updates to the registered synopses by intercepting data updates information in the data sources.

The Workload Manager captures, maintains and analyzes workload information for building, maintaining and testing synopses.

Figure 5 depicts integration process of a remote synopsis within the framework. The system provides a light-weight host process, inside which the custom synopsis will be running. The host is responsible for all communication with the system and is transparent to the synopsis. This design enables unconstraint deployment. A remote synopsis can be integrated into the system by deploying or adapting such host into the remote system, and connecting the synopsis module locally into the host.

Figure 6 illustrates an overall view of the system in a distributed environment, consisting of multiple remote synopses, each representing its local data source.

5.3 Physical Architecture

To answer the design goals stated above, the system was designed using the N-tier concept; Client -(remoting)- Facade - Business Logic - Data Access Layer - Database. See Figure 7.

The client layer implements the user interface of the system. All user actions are treated by the client, however the necessary processing is forwarded to the system server via remoting. Facade interface of the

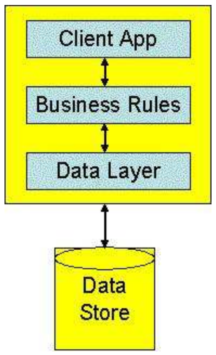


Figure 7: N-tier Architecture

server side provides the single point of access for the client application. The processing is further forwarded to an appropriate class inside the business logic layer. The data access layer is responsible for storing and retrieving information from the database.

The system stores all the internal data (administration information, synopses specifications, etc) in the SQL Server database. The data access layer enables the application to be easily converted to use another commercial database. Substituting the database provider remains transparent for the rest layers of the system.

5.4 The τ-Synopses Core Design

In this section we stand on the design of τ-Synopses server side, i.e. the system core. The core implements all the components revealed in the 5.2 section. Also, as stated in the 5.3 section, the server side is divided into several layers (Facade, Business logic and Data Access). Below we will reveal the major design principals.

The Facade layer is based on the the factory design pattern. The facade factory provides access to the following internal sub-systems:

- *DataSourcesSystem* - manages the data sources specification (database connection details for a relational data and data file location for an XML data)
- *RelationsSystem* - manages concrete row data specifications for relational data sources. A relation is specified for a pre-defined data source by determining the target table name, filter and data column names. This module is also responsible for evaluating the queries over the original data.
- *SynopsesSystem* - manages synopses specification as well as synopsis construction invocation.
- *UpdatesSystem* - manages update specifications
- *UsersSystem* - manages users in the system, thus enabling privacy and user privileges. ”
- *WorkloadsSystem* - manages workload specification, generation and storage.
- *QuerySystem* - responsible for invoking suitable synopsis for approximate estimation of a query.
- *BenchmarkSystem* - responsible for invoking the construction of various synopses and evaluating their accuracy based on the specified query workloads.

```
// relational row data
// or XML document file name
struct SourceData{
            int size;
            int *data;
            string XMLDataFile;
            string XMLSchemaFile;
}
// query is either:
// relational range query:low<x<high
// or XPath expression
struct QueryData
{
            int low;
            int high;
            string pathExpression;
}
// workload is a set of queries
struct WorkloadData
{
            int size;
            QueryData *queries;
}
```

```
 //build synopsis
SYNOPSIS_API int Build( int synopisSize,
                        SourceData sourceData,
                        WorkloadData workloadData);

 //update synopsis
SYNOPSIS_API int Update(UpdateData updateData,
                        WorkloadData workloadData);

//Query synopsis
SYNOPSIS_API QueryResult Query(
                        QueryData queryData);
```

Figure 8: Basic required interfaces

Each sub-system is divided into the above layers: facade - business logic - data access layer.

In order to integrate a new synopsis, it is sufficient to have it implemented on a SOAP-enabled platform. Figure 8 shows the interfaces required for the basic functionality. The synopsis should implement the Build, Update and Query interface methods. After incorporating these interfaces into a synopsis, it can already be used in the system.

5.5 Database Design

Figure 9 depicts the entities diagram (ERD) of the τ-Synopses. We maintain a separate table for each object managed by the system: data source, data relation, workload, synopsis, update and user. The dependency relations between the objects are captured by foreign key relations between the corresponding tables. For example we can see that update, workload and synopsis objects reference the relation object.

While designing the implementation of the τ-xSynopses framework, its requirements were mapped to existing objects of the relational synopses framework. The database schema was extended for supporting XML synopses by adding additional columns to existing tables, thus reflecting the adaptation of the objects to XML support.

6 System Implementation

Below are the major implementation characteristics of the τ-Synopses. The system modules were implemented in the .NET framework, with remote modules communicating through the .NET Remoting. Both Client and Framework applications were developed using the C# programming language. The Client is a windows-forms application while the framework is a windows console application. SQL Server was used as a database provider.

The .NET Framework supplies a rich variety of the ready-to-use components which makes the development much faster, the code much clearer and easier to maintain. The most significant components of the .NET Framework that were utilized in our system are ADO.NET, Remoting, Serialization, XML support and Data Binding. We will briefly review these components and how they were used.

- .NET *Remoting*: Remoting enables client applications to use objects in other processes on the same computer or on any other computer available on its network. In the τ-Synopses the client application is connecting to the server side using the .NET Remoting via a TCP/IP protocol. Synopses are transparently executed on remote machines, over TCP/IP or HTTP protocols, within local area networks or

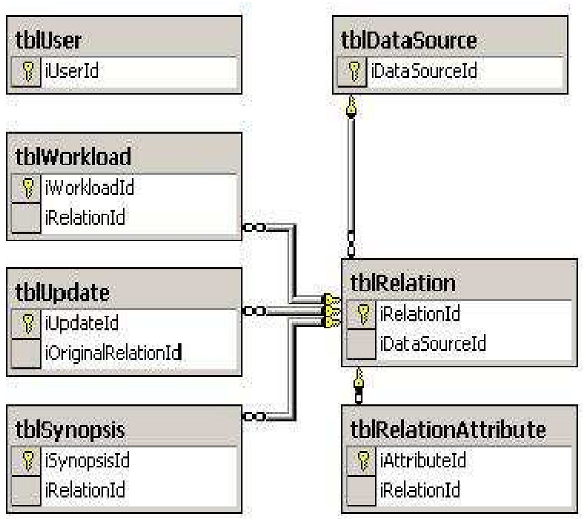


Figure 9: ERD

over the internet. The framework host and the synopsis host are console applications. Both of them are registered on a user specified port in order to be invocable by the remoting.

- *ADO.NET*: ADO.NET is a set of classes that expose data access services to the .NET programmer. In τ-Synopses the SQL database server is accessed using the ADO.NET , each data object represented by a typed dataset.
- *Data Binding*: All user input is managed by data binding of the input fields to the the corresponding column in a typed dataset.
- *XML support*: The .NET Framework provide a comprehensive and integrated set of classes, allowing you to work with XML documents and data. In τ-xSynopses XML classes were used while processing the XML data sources.
- *Serialization*: Serialization is the process of converting the state of an object into a form that can be persisted or transported. Binary serialization provided by the .NET framework Binary Formatter class was used for workload storage implementation.

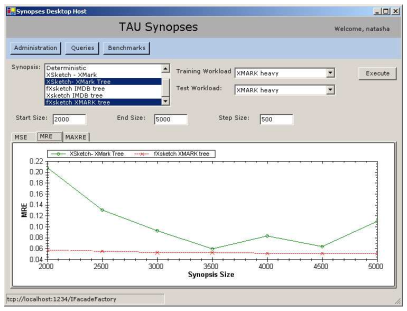
Network disconnected operational mode. The data access layer enables the application to be easily converted to use another commercial database. Substituting the database provider remains transparent for the rest layers of the system. The storage of the system internal data can be even implemented inside XML data files. This feature was indeed implemented in our synopsis framework in order to make it more portable. Just by substituting the data access library we can create a light weight version of our system suitable for any disconnected laptop running Microsoft Windows operational system. However in a disconnected mode the system is limited to the XML data sources unless there are any relational databases locally installed. The version without database storage was implemented by serializing the typed datasets to XML documents and storing them in files.

7 Experience with the System

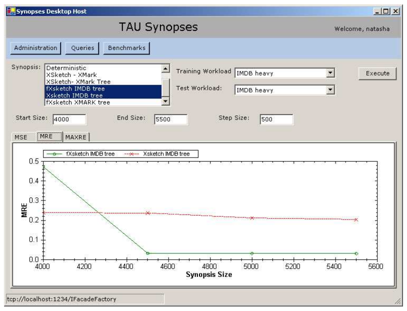
The τ-Synopses framework was successfully tested by deploying remote synopses for both relational and XML data sources.

Several synopses for approximate queries over relational databases were tested and evaluated using the framework run-time environment. The platform allowed comparison of different techniques - histogram based, sampling and wavelet based solutions were evaluated and compared [16].

The XML support of the system was tested with two already implemented synopses techniques. The XSketches [19] and the fXSKETCHES [9] structured synopses for XML databases were integrated into the framework and successfully evaluated. Figure 10 depicts the performance comparison of XSketch and fXSketch synopses constructed over two different data sets.


Synopsis: Deterministic, XSketch - XMark, XSketch- XMark Tree, fXsketch IMDB tree, Xsketch IMDB tree, fXsketch XMARK tree
Training Workload: XMARK heavy
Test Workload: XMARK heavy
Start Size: 2000
End Size: 5000
Step Size: 500
tcp://localhost:1234/IFacadeFactory

(a) XMARK Data Set


Synopsis: Deterministic, XSketch - XMark, XSketch- XMark Tree, fXsketch IMDB tree, Xsketch IMDB tree, fXsketch XMARK tree
Training Workload: IMDB heavy
Test Workload: IMDB heavy
Start Size: 4000
End Size: 5500
Step Size: 500
tcp://localhost:1234/IFacadeFactory

(b) IMDB Data Set

Figure 10: Performance comparison of two XML synopses - XSketch and fXSketch

The τ-Synopses framework was proved to be an efficient research platform.

8 Conclusions

The τ-xSynopses system provide a run-time environment for remote execution and management of pluggable synopses for XML databases. The τ-Synopses Framework extended with the XML support was proved to be an efficient research platform for benchmarking and comparison of approximate queries techniques for both relational and XML data sources.

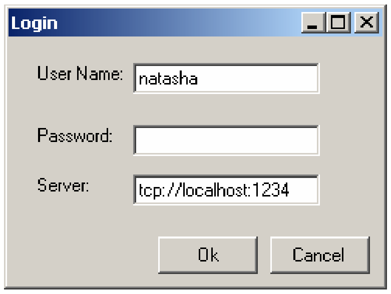
We now encourage other research groups connect their synopses to the τ-Synopses system. This would allow access to a wide variety of different datasources and workloads, and a fair comparison with other synopses with little effort.

9 User Scenario

The aim of this section is to guide a user through the different capabilities of the τ-xSynopses system. Performing all the described actions in the order they appear below will provide the user with full introduction to the system.

We will demonstrate: (i) how to define a data source; (ii) how to register and run remote synopses; (iii) how to construct query workloads; (iv) how users can execute approximate queries on synopses (of unavailable data source); (v) how to use the benchmark mode for performance comparison between two XML synopses techniques.

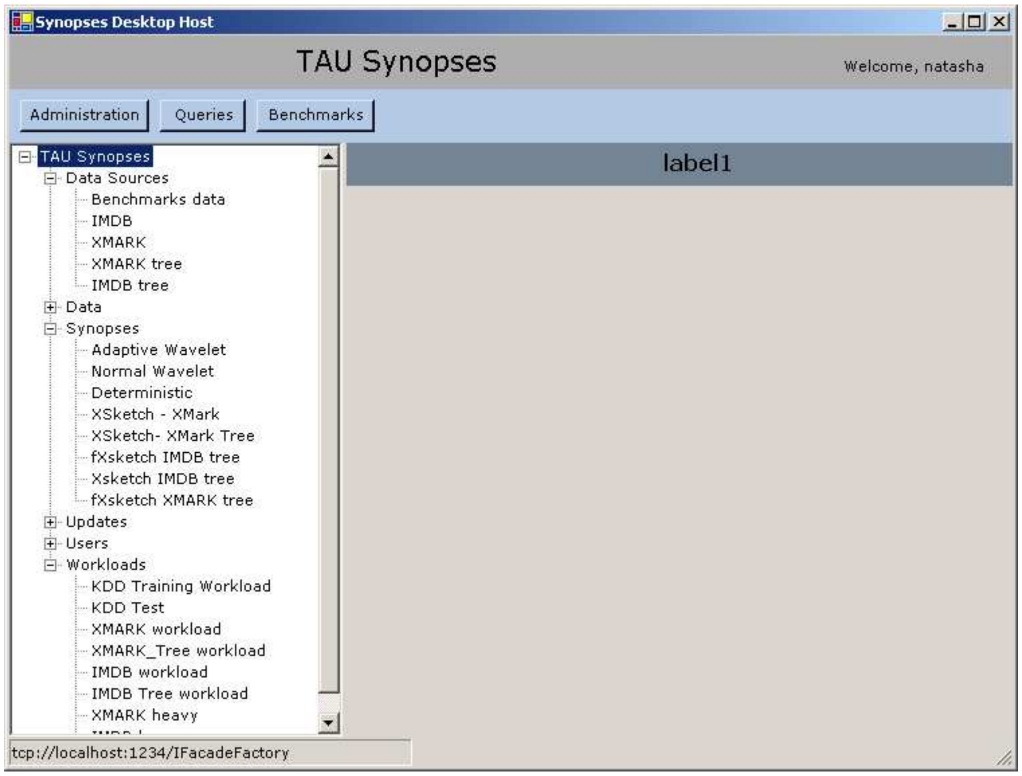
System Login: Execute the provided framework client (Synopsis.Client.exe). Login to the system (please contact us for the login registration and for the Server URL), see Figure 11



User Name: natasha
Password:
Server: tcp://localhost:1234

Figure 11: Login screen

The system loads the administration module where navigation tree is available, see Figure 12. There are two additional operational modes - Query and Benchmark. Switching between these modes is performed through the three mode buttons that always appear at the top of the screen.



- TAU Synopses
  - Data Sources
    - Benchmarks data
    - IMDB
    - XMARK
    - XMARK tree
    - IMDB tree
  - Data
  - Synopses
    - Adaptive Wavelet
    - Normal Wavelet
    - Deterministic
    - XSketch - XMark
    - XSketch- XMark Tree
    - fXsketch IMDB tree
    - Xsketch IMDB tree
    - fXsketch XMARK tree
  - Updates
  - Users
  - Workloads
    - KDD Training Workload
    - KDD Test
    - XMARK workload
    - XMARK_Tree workload
    - IMDB workload
    - IMDB Tree workload
    - XMARK heavy

tcp://localhost:1234/IFacadeFactory

Figure 12: Administration Module

9.1 Administration Execution Mode

Data Sources:

1. To define new data source, select ’Data Sources’ node, right click with the mouse and select the Add menu option. New node is added to the ’Data Sources’ administration node.
2. Select this node and edit the data source - Figure 13.
   - For XML data source, set the ’Type’ input field to ’XML’.
   - Type the full path for the XML document file in the ’XML File’ input field or browse to select the file.
3. Press ’Save’ button to save the input data.

Synopses:

1. To define new synopsis, select ’Synopses’ node, right click with the mouse and select the Add menu option. New node is added to the ’Synopses’ administration node.
2. Select this node and configure the synopsis - Figure 14.
   - For XML data source, set the ’Kind’ input field to ’XML’.
   - Choose the desired XML data source in the ’Data’ input field.
   - Fill in the location information of the remote synopsis - the server address and the listener port.
3. Press ’Save’ button to save the configuration.

Workloads:

1. To construct a workload, select ’Workloads’ node, right click with the mouse and select the Add menu option. New node is added to the ’Workloads’ administration node.
2. Select this node and configure the workload parameters - Figure 15.
   - For XML workload, set the ’Type’ input field to ’XML XPath Query’.
   - Choose the desired XML data source in the ’Data’ input field.
   - Fill in the workload construction parameters.
3. Press ’Save’ button to save the configuration.
4. Press ’Build’ button to invoke workload construction for the specified data source. The workload is constructed by the system and saved in the database for future use.
5. The XPath expressions of the constructed workload can be viewed by saving the workload to a file - press ’save to File’ button and choose file location.

9.2 Synopses Evaluation

The system supplies the capability to test and compare different synopses that are registered in the system. The Query execution mode provides the user with the ability to evaluate single synopsis at a time, figure 16 depicts the Query mode UI. The Benchmark mode enables multiple synopses evaluation over pre-defined workloads and their comparison using visual display. The user interface for the benchmark mode is depicted in figure 17.

9.2.1 Query Execution Mode

Enter the Query execution mode by pressing the ”Queries” button at the top of the screen.
Building the Synopsis:

1. When evaluating single synopsis, you have first to invoke its construction. In the administration mode, choose the node of the synopsis to test.

2. Set the desired size of the synopsis and the training workload (if required).

3. Press ’Save’ to save the settings and then press ’Build’, see Figure 14.

4. At this point the remote synopsis agent must be available at the location that was configured for this synopsis. See further instructions how to integrate your synopses with the remote synopses host. After the connection to the remote synopsis is established and the synopsis is built, the system is ready for processing user queries or predefined workloads.

Querying the Synopsis: In the Query mode, choose the synopsis to evaluate among the list of registered synopses. There are two possible ways to query the synopsis - the custom query (the upper part of the Query mode screen) and the workload queries (the beneath part), see Figure 16.

Custom Query. When the chosen synopsis is of XML type, the input field for query XPath expression is available. Insert a valid XPath expression and press ’Query’ button. The system will send the query to the remote synopsis and display the returned estimation of path selectivity. The system will also process the query over the XML document that underly the synopsis and will display real selectivity.

Workload Queries. The selected synopsis can be also queried with a predefined workload. A workload can be either chosen among the predefined workloads or it can be loaded from external file, such file can be created by saving a workload to a file (see workload instructions). Choose a workload for the same XML data source as the synopsis data source. Press the ’Run’ button to evaluate the workload. The system will send the workload queries to the remote synopsis and display the returned results in the ’Workload Results’ graph where each point reflects the returned value for a query. The system will also process the workload queries over the XML document that underly the synopsis and will display the real selectivities. The average relative error of synopsis evaluation will be displayed above the results graph.

Note that the synopsis must be pre-constructed before being queried - see ’Building the Synopsis’ paragraph.

9.2.2 Benchmark Execution Mode

The Benchmark mode enables multiple synopses evaluation over pre-defined workloads and their comparison using visual display. The user interface for the benchmark mode is depicted in figure 17.

1. Enter the Benchmark execution mode by pressing the ”Benchmark” button at the top of the screen.

2. Select one or more synopses to test, each of them should be defined for the same XML data source.

3. Choose a training workload (if needed for construction of these synopses) and a test workload to be evaluated. The workloads should be for the same data source as the synopses.

4. Set the sizes of synopses to be constructed and evaluated against the workload. Set the minimum size, the maximum size and the step size, all the sizes are in bytes. The system will invoke construction of $(max - min)/step$ synopses of each type. Note, in the benchmark mode there is no mean to the size specified in the synopsis configuration (administration mode).

5. When all the benchmark data is configured press the ’Execute’ button.

6. After all the synopses are constructed and evaluated (it may take some time) the system calculates the accuracy of the different synopses using several error metrics. MSE (mean square error), MRE (mean relative error) and MAXRE (maximum relative error) performance measurements are depicted in the results graph, switch between the graph tabs to see the errors.

This mode provides an effective way to compare the performance of different synopses.

9.3 Integrating Remote Synopsis

For integration purposes, a synopsis has to implement a simple interface, regardless of its internal implementation. Figure 18 shows the interfaces required for the basic functionality. The synopsis should implement the Build, Update and Query interface methods.

Implement your synopsis in any language of your choice. Integrate with the remote synopses host in one of the following ways, depending on the platform on which the synopsis runs:

a. Win32 platform: write a wrapper class named ExternalSynopsis that will support the remote synopsis interface presented at internal implementation. Compile it as a DLL.

b. Others: Supply a Web-Service interface that implements a remote synopsis interface.

Next instructions are for the Win32 platform.

1. A light-weight host is provided by the system - RemoteHost.exe.

2. Copy your compiled synopsis to RemoteHost.exe directory.

3. Execute RemoteHost.exe and select a desired port, see Figure 19.

4. Configure the synopsis location in the administration module of τ-xSynopses system. Your synopsis is ready for use through the system.

References

[1] A. Aboulnaga and S. Chaudhuri. Self-tuning histograms: building histograms without looking at data. In *Proceedings of the 1999 ACM SIGMOD*, pages 181–192, 1999.

[2] S. Acharya, P. Gibbons, and V. Poosala. Aqua: A fast decision support system using approximate query answers. In *Proceedings of the 1999 VLDB*, 1999.

[3] S. Acharya, P. Gibbons, V. Poosala, and S. Ramaswamy. The aqua approximate query answering system. In *Proceedings of the 1999 ACM SIGMOD*, 1999.

[4] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. “Extensible Markup Language (XML) 1.0 (Second Edition)”. W3C Recommendation (available from `http://www.w3.org/TR/REC-xml/`), Oct. 2000.

[5] N. Bruno, S. Chaudhuri, and L. Gravano. STHoles: a multidimensional workload-aware histogram. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 30(2):211–222, 2001.

[6] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. *VLDB Journal: Very Large Data Bases*, 10(2–3):199–223, 2001.

[7] S. Chaudhuri, G. Das, and V. Narasayya. A robust, optimization-based approach for approximate answering of aggregate queries. In *Proceedings of the 2001 ACM SIGMOD*, 2001.

[8] J. Clark and S. DeRose. "XML Path Language (XPath), Version 1.0". W3C Recommendation (available from `http://www.w3.org/TR/xpath/`), Nov. 1999.

[9] N. Drukh, N. Polyzotis, M. Garofalakis, and Y.Matias. "Fractional XSKETCH Synopses for XML Databases". In *"Proceedings of XSym'2004"*, 2004.

[10] M. Garofalakis and P. B. Gibbons. Wavelet synopses with error guarantees. In *Proceedings of the 2002 ACM SIGMOD*, pages 476–487, 2002.

[11] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *Proceedings of the 1998 ACM SIGMOD*, pages 331–342, 1998.

[12] P. B. Gibbons, Y. Matias, and V. Poosala. AQUA project white paper. Technical report, Bell Labs, Murray Hill, New Jersey, U.S.A., 1997.

[13] A. C. Konig and G. Weikum. Automatic tuning of data synopses. *Information Systems*, 28(1-2). Special Issue of papers from EDBT 2002.

[14] A. C. Konig and G. Weikum. A framework for the physical design problem for data synopses. March 2002.

[15] L. Lim, M. Wang, S. Padmanabhan, J. Vitter, and R. Parr. XPathLearner: An On-Line Self-Tuning Markov Histogram for XML Path Selectivity Estimation. In *Proceedings of the 28th Intl. Conf. on Very Large Data Bases*, 2002.

[16] Y. Matia, Y. Matias, and L. Portman. Synopses - taxonomy and experimental evaluation. Technical Report, in preparation, 2004.

[17] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In *Proceedings of the 1998 ACM SIGMOD*, pages 448–459, 1998.

[18] Y. Matias, J. S. Vitter, and M. Wang. Dynamic maintenance of wavelet-based histograms. In *The VLDB Journal*, pages 101–110, 2000.

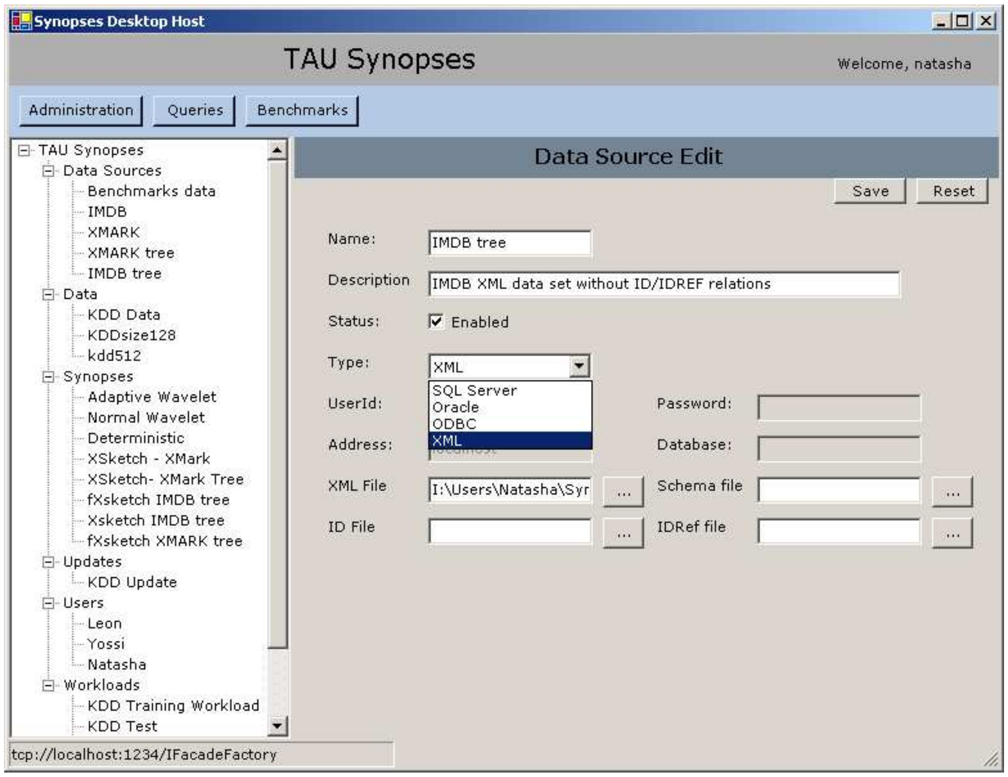
[19] N. Polyzotis and M. Garofalakis. "Statistical Synopses for Graph Structured XML Databases". In *Proceedings of the 2002 ACM SIGMOD Intl. Conf. on Management of Data*, June 2002.

[20] N. Polyzotis and M. Garofalakis. "Structure and Value Synopses for XML Data Graphs". In *Proceedings of the 28th Intl. Conf. on Very Large Data Bases*, 2002.

[21] J. S. Vitter and M. Wang. Approximate computation of multidimensional aggregates of sparse data using wavelets. In *Proceedings of the 1999 ACM SIGMOD*, pages 193–204, 1999.

[22] Xyleme home page.

[23] Y.Matias and L.Portman. "τ-synopses: a system for run-time management of remote synopses". In *"Software Demo, ICDE'04, EDBT'04"*, 2004.



Synopses Desktop Host

TAU Synopses

Welcome, natasha

- TAU Synopses
  - Data Sources
    - Benchmarks data
    - IMDB
    - XMARK
    - XMARK tree
    - IMDB tree
  - Data
    - KDD Data
    - KDDsize128
    - kdd512
  - Synopses
    - Adaptive Wavelet
    - Normal Wavelet
    - Deterministic
    - XSketch - XMark
    - XSketch- XMark Tree
    - fXsketch IMDB tree
    - Xsketch IMDB tree
    - fXsketch XMARK tree
  - Updates
    - KDD Update
  - Users
    - Leon
    - Yossi
    - Natasha
  - Workloads
    - KDD Training Workload
    - KDD Test

Data Source Edit

Name: IMDB tree

Description IMDB XML data set without ID/IDREF relations

Status: [✓ Enabled]

Type: XML

SQL Server
Oracle
ODBC
[✓ XML]

UserId:

Password:

Address: localhost

Database:

XML File I:\Users\Natasha\Syr

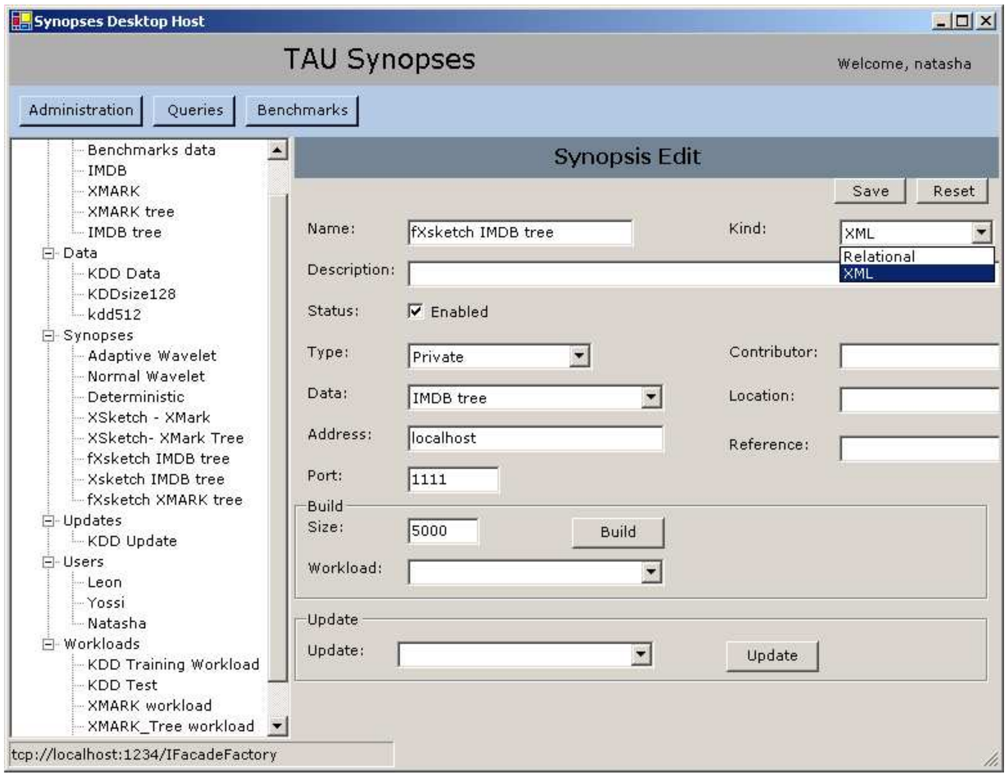
Schema file

ID File

IDRef file

tcp://localhost:1234/IFacadeFactory

Figure 13: Data Source Edit Mode



Synopses Desktop Host

TAU Synopses

Welcome, natasha

```
    Benchmarks data
    IMDB
    XMARK
    XMARK tree
    IMDB tree
⊟ Data
    KDD Data
    KDDsize128
    kdd512
⊟ Synopses
    Adaptive Wavelet
    Normal Wavelet
    Deterministic
    XSketch - XMark
    XSketch- XMark Tree
    fXsketch IMDB tree
    Xsketch IMDB tree
    fXsketch XMARK tree
⊟ Updates
    KDD Update
⊟ Users
    Leon
    Yossi
    Natasha
⊟ Workloads
    KDD Training Workload
    KDD Test
    XMARK workload
    XMARK_Tree workload
```

Synopsis Edit

Name:	fXsketch IMDB tree	Kind:	XML (Relational / XML)
Description:			
Status:	☑ Enabled		
Type:	Private	Contributor:	
Data:	IMDB tree	Location:	
Address:	localhost	Reference:	
Port:	1111		

Build

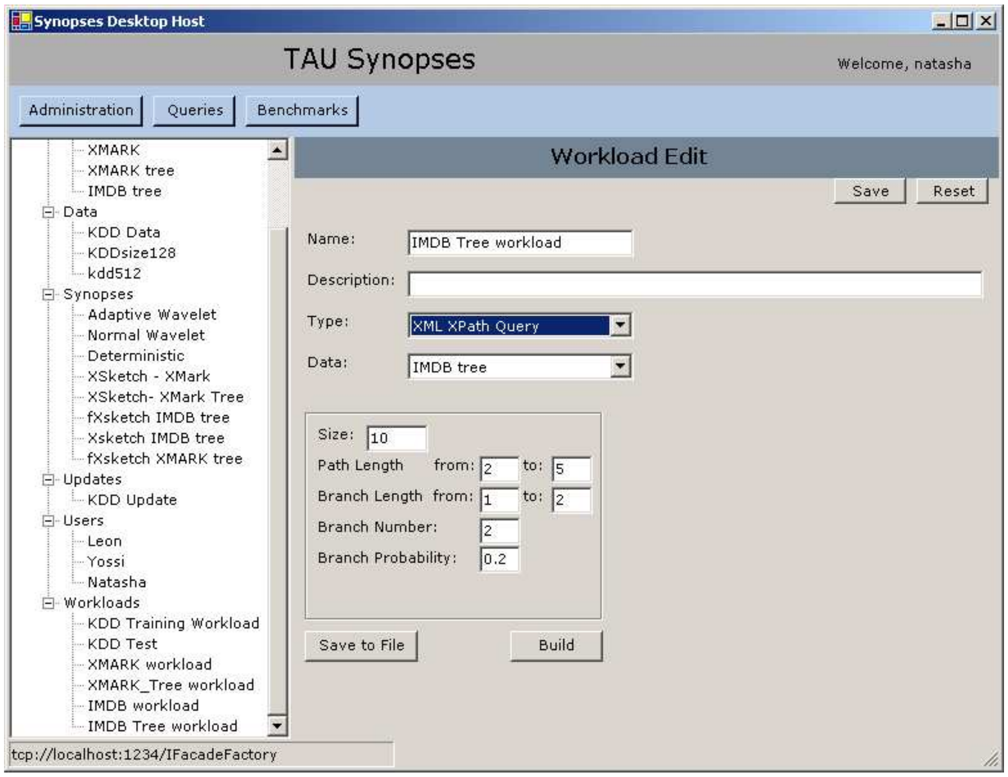
Size:	5000
Workload:	

Update

Update:	

tcp://localhost:1234/IFacadeFactory

Figure 14: Synopsis Configuration Mode



Name: IMDB Tree workload

Description:

Type: XML XPath Query

Data: IMDB tree

Size: 10

Path Length from: 2 to: 5

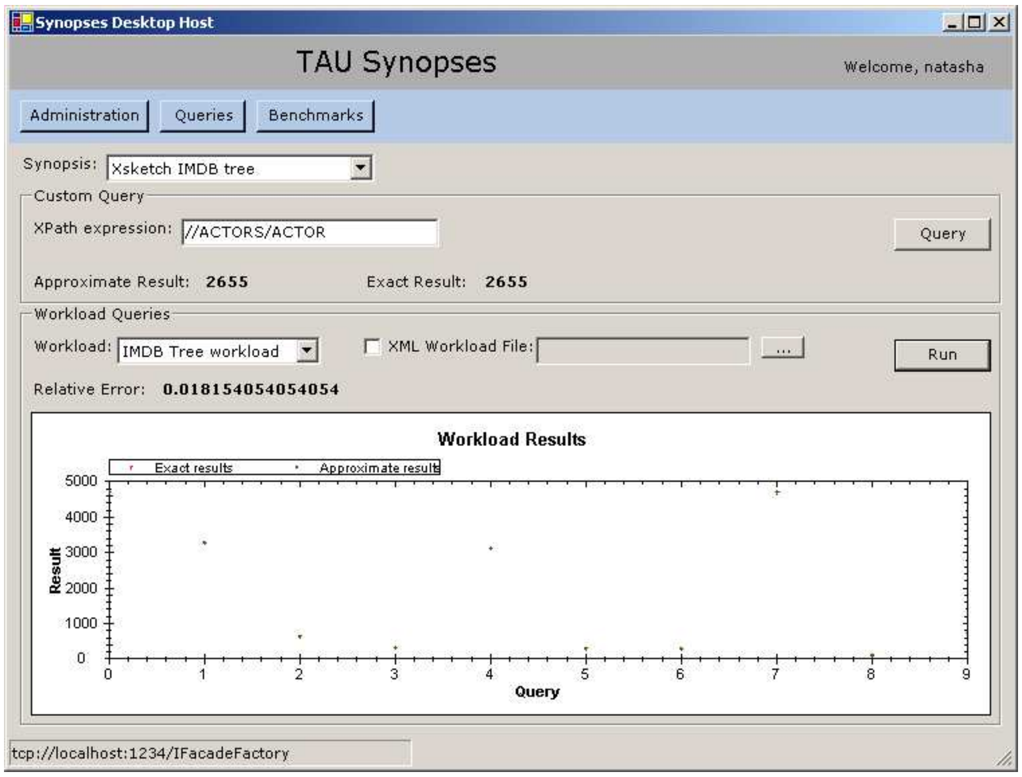
Branch Length from: 1 to: 2

Branch Number: 2

Branch Probability: 0.2

tcp://localhost:1234/IFacadeFactory

Figure 15: Workload Configuration Mode



Synopsis: Xsketch IMDB tree

Custom Query

XPath expression: //ACTORS/ACTOR

Approximate Result: **2655** Exact Result: **2655**

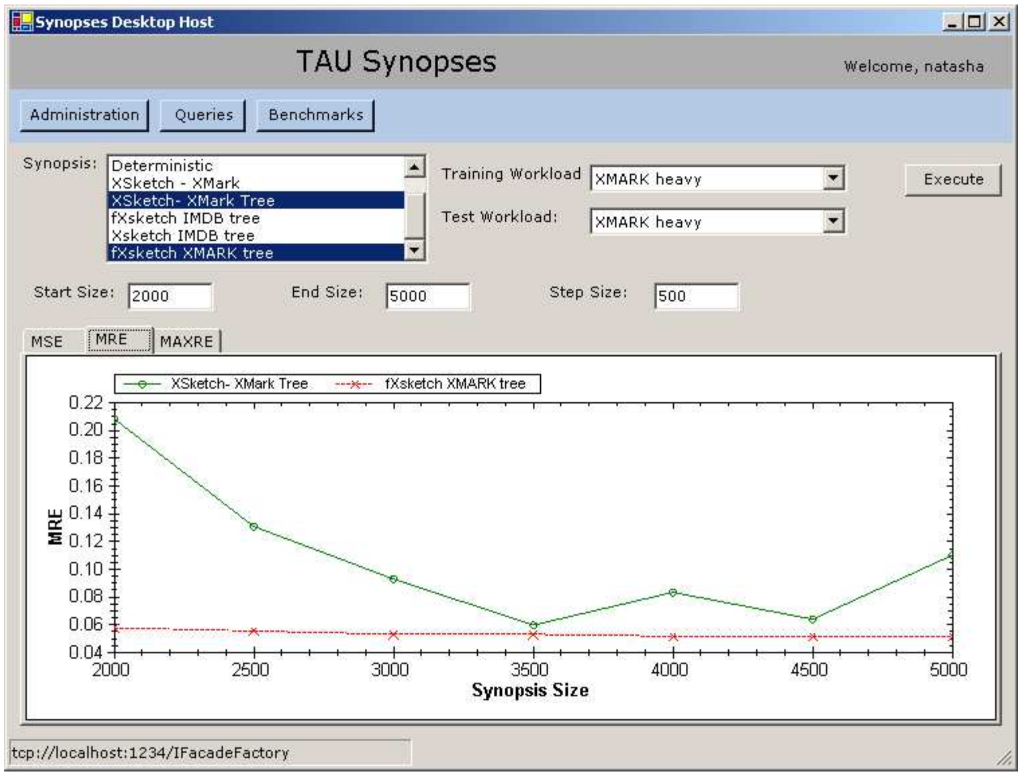
Workload Queries

Workload: IMDB Tree workload ☐ XML Workload File:

Relative Error: **0.018154054054054**

tcp://localhost:1234/IFacadeFactory

Figure 16: Query Mode



Synopsis: Deterministic, XSketch - XMark, XSketch- XMark Tree, fXsketch IMDB tree, Xsketch IMDB tree, fXsketch XMARK tree

Training Workload: XMARK heavy

Test Workload: XMARK heavy

Start Size: 2000 End Size: 5000 Step Size: 500

Figure 17: Benchmark Mode

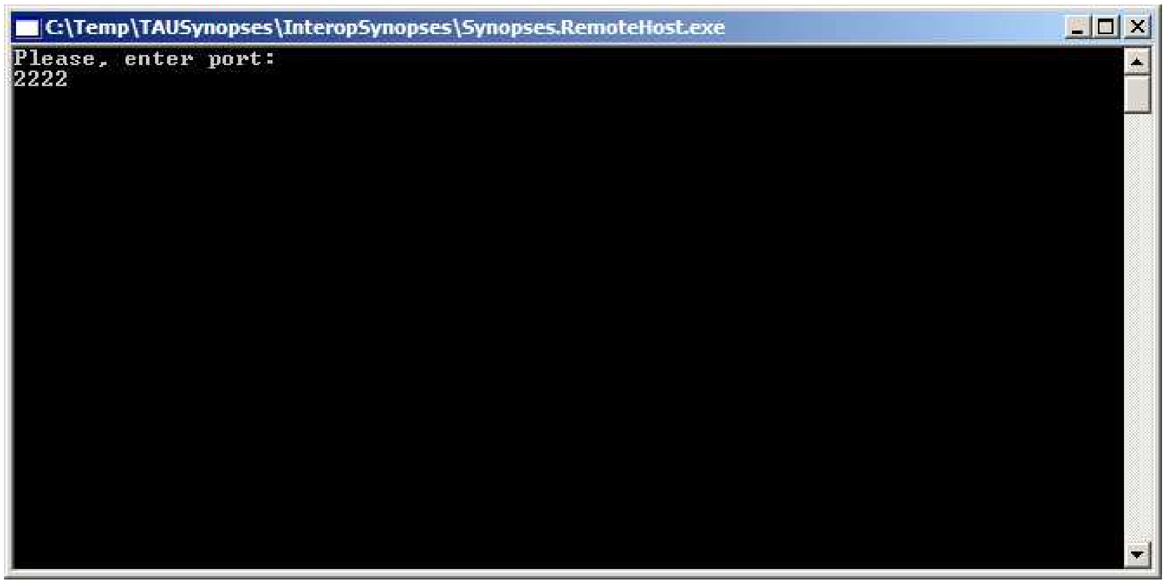
```
// relational row data
// or XML document file name
struct SourceData{
        int size;
        int *data;
        string XMLDataFile;
        string XMLSchemaFile;
}
// query is either:
// relational range query:low<x<high
// or XPath expression
struct QueryData
{
        int low;
        int high;
        string pathExpression;
}
// workload is a set of queries
struct WorkloadData
{
        int size;
        QueryData *queries;
}
```

```
 //build synopsis
SYNOPSIS_API int Build( int synopisSize,
                SourceData sourceData,
                WorkloadData workloadData);

 //update synopsis
SYNOPSIS_API int Update(UpdateData updateData,
                WorkloadData workloadData);

//Query synopsis
SYNOPSIS_API QueryResult Query(
                QueryData queryData);
```

Figure 18: Remote Synopsis Interface



```
Please, enter port:
2222
```

Figure 19: Remote Host