Benchmarking XML data warehouses

Hadj Mahboubi and Jérôme Darmont

ERIC laboratory, University of Lyon 2
5 avenue Pierre Mendès–France, 69676 Bron Cedex, France
{hadj.mahboubi | jerome.darmont}@eric.univ-lyon2.fr

Abstract. With the emergence of XML as a new standard for representing business data, new decision-support applications (namely, XML data warehouses) are being developed. To ensure their feasibility, the issue of performance must be addressed. Performance in general, and the efficiency of performance optimization techniques in particular, is usually assessed with the help of benchmarks. However, there are, to the best of our knowledge, no XML decision-support benchmark.

In this paper, we present the XML Warehouse Benchmark (XWB), which aims at filling this gap. XWB is based on an original reference model for XML data warehouses, and proposes a test XML data warehouse and its associated XQuery decision-support workload that are derived from the well-known, relational decision-support benchmark TPC-H. Though at an early stage of development, XWB has been successfully used to test the efficiency of indexing and view materialization techniques in XML data warehouses.

Keywords: XML data warehouses, XQuery decision-support queries, XML benchmarking, performance evaluation.

1 Introduction

The development of XML-native Database Management Systems (DBMSs) is quite recent, and a tremendous amount of research is currently in progress to help them in becoming a credible alternative to XML-compatible, relational DBMSs. Several performance evaluation tools (namely, benchmarks) have been proposed to support this effort.

XML is also gaining importance as the standard for representing business data (Beyer et al., 2005): several proposals aim at designing and building XML data warehouses, and the XQuery language is being extended to support analytic queries such as OLAP aggregation queries. Furthermore, decision-support applications nowadays exploit heterogeneous data from various sources, and XML is particularly adapted to describe and store such complex data (Darmont et al., 2005b).

The existing XML benchmarks are ill-suited to evaluate the performances of decision-oriented applications. Whever they are document-centric, data-centric or both, their databases are not aimed at decision-support but at transactional applications, and their workloads do not feature typical analytic queries.

Hence, we propose in this paper the first (to the best of our knowledge) XML decision-support benchmark. It is named the XML Warehouse Benchmark (XWB; pronounced X-Web in reference to the DWEB relational data warehouse benchmark (Darmont et al., 2005a) it is related to). Our objective with XWB is to design a test XML data warehouse and its associated XQuery decision-support workload, for performance evaluation purposes. XWB is based on an original reference model for XML data warehouses, and its warehouse and workload are derived from the well-known, relational decision-support benchmark TPC-H.

The remainder of this paper is organized as follows. In Section 2, we present the state of the art regarding relational decision-support benchmarks, XML benchmarks and XML data warehouses, respectively. In Section 3, we detail the specifications of XWB (XML warehouse reference model, database and workload). We finally conclude this paper and provide future research directions in Section 4.

2 Related work

2.1 Relational decision-support benchmarks

The Transaction Processing Performance Council (TPC)[1] plays a central role in the standardization of relational benchmarks. It has issued several decision-support benchmarks. TPC-D (TPC, 1998) has appeared in the mid-nineties, and forms the base of TPC-R (TPC, 2003) and TPC-H (TPC, 2005b), which have replaced it. TPC-R and TPC-H are actually identical, only their usage varies. TPC-R is aimed at reporting (queries are known in advance and adequate optimizations are possible), while TPC-H aims at ad-hoc querying (queries are not known in advance and optimizations are not allowed). TPC-H is the only decision-support benchmark that is currently supported by the TPC. TPC-R and TPC-H exploit the same database schema than TPC-D: a classical *product-order-supplier* model; as well as TPC-D's workload, enriched with five new queries. More precisely, this workload is constituted of twenty-two SQL-92, parameterized, decision-support queries and two refreshing functions that insert tuples into and delete tuples from the database. Query parameters are randomly instantiated following a uniform law. Three primary metrics are used in these benchmarks. They describe performance in terms of power, throughput, and a combination of these two criteria. Power and throughput are the geometric and arithmetic mean values of database size divided by the workload execution time, respectively.

Data warehouses nowadays constitute a key decision-support technology. However, TPC-H's database schema is not truly dimensional, i.e., it is not a star-like schema that is typical in data warehouses. Furthermore, its workload does not include any On-Line Analytical Processing (OLAP) query. TPC-DS, which is currently under development (TPC, 2005a), fills in this gap. Its schema represents the decision-support functions of a retailer under the form of a constellation schema with several fact tables and shared dimensions. TPC-DS' workload is constituted of four classes of queries: reporting queries, ad-hoc decision-support queries, interactive OLAP queries, and extraction queries. SQL-99 query templates help in randomly

[1]http://www.tpc.org

generating a set of about five hundred queries, following non-uniform distributions. The warehouse maintenance process includes a full ETL (Extract, Transform, Load) phase, and handles dimensions according to their nature (non-static dimensions scale up while static dimensions are updated). One primary throughput metric is proposed in TPC-DS. It takes both query execution and the maintenance phase into account.

As in all the other TPC benchmarks, scaling in TPC-H and TPC-DS is achieved through a scale factor that helps defining the database's size (from 1 GB to 100 TB). Both the database schema and the workload are fixed. The number of generated queries in TPC-DS directly depends on the scale factor, for instance.

There are few decision-support benchmarks out of the TPC, and their specifications are rarely integrally published. Some are nonetheless of great interest. APB-1 is presumably the most famous. It has been published in 1998 by the OLAP council, a now inactive organization founded by four OLAP solution vendors. APB-1 has been intensively used in the late nineties. Its warehouse dimensional schema is structured around four dimensions: *Customer, Product, Channel,* and *Time*. Its workload of ten queries is aimed at sale forecasting. APB-1 is quite simple and proved limited to evaluate the specificities of various activities and functions (Thomsen, 1998). It is now difficult to find.

Eventually, while the TPC standard decision-support benchmarks are invaluable to users for comparing the performances of different systems, they are less useful to system engineers for testing the effect of various design choices. They are indeed not tunable enough and fail to model different data warehouse schemas. By contrast, the Data Warehouse Engineering Benchmark (DWEB) helps in generating various ad-hoc synthetic data warehouses (modeled as star, snowflake, or constellation schemas) and workloads that include typical OLAP queries (Darmont et al., 2005a). DWEB is fully parameterized to fulfill data warehouse design needs. Thus, it may be viewed more like a benchmark generator than an actual, single benchmark. It is indeed very important to achieve the different kinds of schemas that are used in data warehouses, and to allow designers to select the precise architecture they need to perform tests on.

2.2 XML benchmarks

Several studies address the issue of XML benchmarking. X-Mach1 (Böhme and Rahm, 2001, 2002), XMark (Schmidt et al., 2003), XOO7 (an extension of the object-oriented benchmark OO7 (Carey et al., 1993)) and XBench (Yao et al., 2003, 2004) are so-called application benchmarks. Their objective is to evaluate the global performances of an XML-native or compatible DBMS, and more particularly of its query processor. Each of them implements a mixed XML database that is both data-oriented (structured data) and document-oriented (in general, random texts built from a dictionary). However, except for XBench that proposes a true mixed database, their orientation is more particularly focused on data (XMark, XOO7) or documents (X-Mach1). These benchmarks also differ in:

- the fixed or flexible nature of the XML schema (one or several Document Type Definitions or XML schemas);
- the number of XML documents used to model the database at the physical level (one or several);

- the inclusion or not of update operations in the workload.

We can also underline that only XBench helps in evaluating all the functionalities offered by the XQuery language.

Micro-benchmarks have also been proposed to evaluate the individual performances of basic operations such as projections, selections, joins, and aggregations, rather than more complex queries. The Michigan Benchmark (so-named in reference to the relational Wisconsin Benchmark developed in the eighties) (Runapongsa et al., 2006) and MemBeR (Afanasiev et al., 2005) are made for XML documents storage solution designers, who can isolate critical issues to optimize, rather than for users seeking to compare different systems. Furthermore, MemBeR proposes a methodology for building micro-databases, to help users in adding datasets and specific queries to a given performance evaluation task.

2.3 XML data warehouses

The studies that address the issue of designing and building XML data warehouses use XML documents to manage or represent the facts and/or dimensions of the warehouse. This allows the native storage of documents and their easy interrogation with XML query languages.

Some of these approaches are user-driven. They are applied when an organization has fixed warehouse requirements. Nassis *et al.* (Nassis et al., 2004) and Rajugan *et al.* (Rajugan et al., 2005) propose methods to conceptually design and build an XML repository, based on object oriented concepts and a view-driven approach, respectively. This repository represents the warehouse analysis context. Vrdoljak *et al.* propose a design approach for web warehouses that is based on XML schemas describing data sources (Vrdoljak et al., 2003). All these approaches assume that the warehouse is composed of XML documents representing facts.

Other approaches are explicitly based on the classical warehouse logical models. For instance, Pokorný models a star schema in XML by defining dimension hierarchies as a sets of logically connected collections of XML data, and facts as XML data elements (Pokorný, 2002). Park *et al.* propose an XML multidimensional model in which each fact is described by a single XML document and dimension data are grouped into a repository of XML documents (Park et al., 2005). Eventually, Hümmer *et al.* propose a family of templates, called XCube, to describe a multidimentional structure (dimension and fact data) for integrating several data warehouses into a virtual or federated data warehouse (Hümmer et al., 2003). All these approaches assume that the warehouse is composed of XML documents that represent both facts and dimensions. They are used when dimensions are dynamic and allow the support of end-user analytical tools.

3 XWB specification

Typically, a benchmark is constituted of two main elements: a database model (conceptual schema and extension) and a workload model (set of read and write operations) to apply on this database, following a predefined protocol. As shown in Figure 1, XWB consists of two parts: an XML data warehouse and an XQuery decision-support workload. Performance evaluations are performed by applying this workload onto the XML data warehouse.

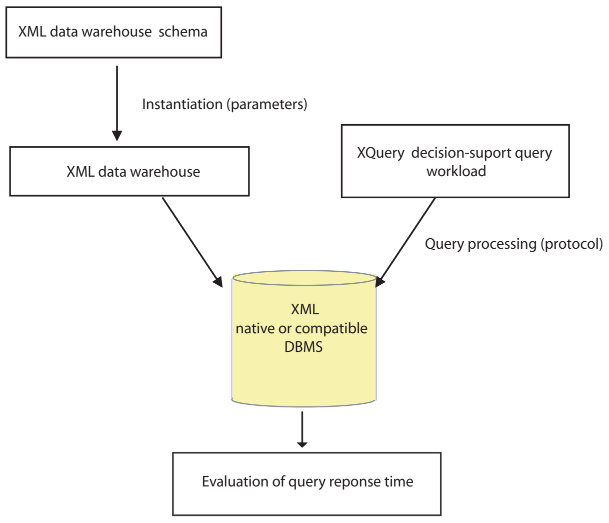
XML data warehouse schema
Instantiation (parameters)
XML data warehouse
XQuery decision-suport query workload
Query processing (protocol)
XML
native or compatible
DBMS
Evaluation of query reponse time

FIG. 1 – *XWB components*

3.1 XML warehouse reference model

XML data warehouse studies (section 2.3) converge toward a unified XML warehouse model. They mostly differ in the way dimensions are handled and the number of XML documents that are used to store facts and dimensions.

Some authors use a collection of XML documents to represent both facts and dimensions (Boussaid et al., 2006; Park et al., 2005). In opposition, we store the facts only one document, like Hümmer et al. (2003). This helps in decreasing the scan cost of facts and allows to model constellation schemas without duplicating dimension information.

We also define each dimension and its hierarchical levels in one XML document. Hence, query and update operations are more easily and efficiently performed than if dimensions were either embedded with the facts (Boussaid et al., 2006; Park et al., 2005) or all stored in one document (Hümmer et al., 2003).

We also define another XML document to represent the warehouse schema. Hence, our reference data warehouse is composed of the following XML documents: *facts.xml* specifies the facts, i.e., dimension identifiers and measure descriptions and values; $dimension_d.xml$ defines dimension d, characterized by its attributes and their values; and *dw-model.xml*.

dw-model.xml defines the multidimensional structure of the warehouse. Its root node, *dw-model*, is composed of two types of nodes, *dimension* and *FactDoc*. The *dimension* node defines one dimension, its hierarchical levels and attribute types. The *FactDoc* element defines facts, i.e., measure values and their corresponding dimensions. Figure 2 shows how the *dw-model.xml* document is structured.

$dimension_d.xml$ allows the instantiation of dimension XML documents. A dimension document stores one dimension and its hierarchical levels. Its structure is described in Figure 3(b). The document root node, *dimension*, is composed of *Level* nodes. Each one defines a

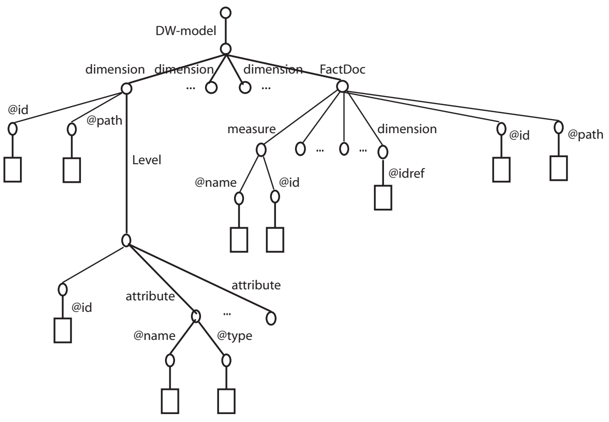

FIG. 2 – *dw-model.xml structure*

dimension level, composed of *instance* nodes representing instances of the level. An *instance* defines the attributes of a level and their values.

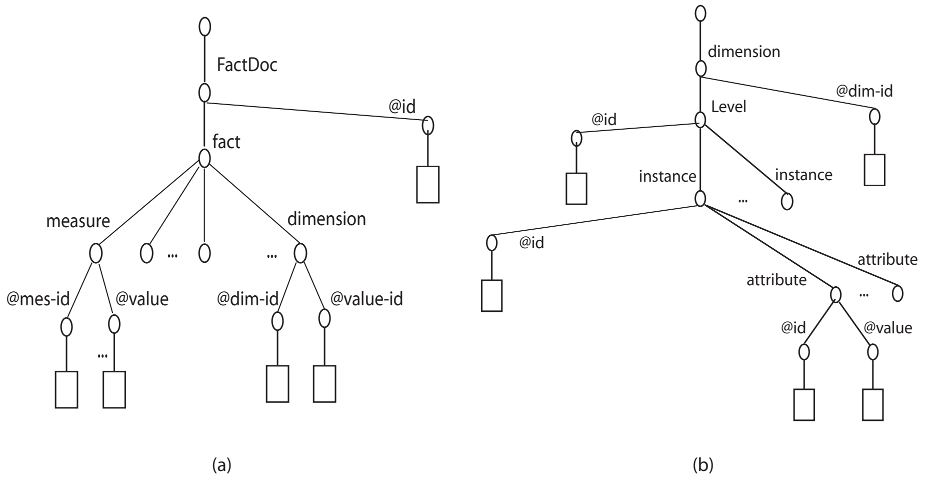

FIG. 3 – *facts.xml (a) and* $dimension_d.xml$ *(b) structure*

Finally, *facts.xml* stores the facts and is composed of *fact* nodes defining measures and dimension references (Figure 3(a)).

3.2 XWB database

3.2.1 Schema.

Our benchmark database model is inspired from that proposed in the TPC-H relational benchmark. Since TPC-DS' specifications are not yet finalized, we indeed preferred to rely on the well-known and simpler TPC-H. However, we extended the schema from TPC-H to obtain a snowflake schema. Figure 4 shows a UML class diagram representing the warehouse conceptual model we obtain. It represents a classical sales facts case study. These facts are described by the *products, customers, supplies* and *date* dimensions. Most of these dimension contain hierarchies at the logical and physical levels. This schema is represented in XML and stored in the *dw-model.xml* document (Figure 5 (a)).

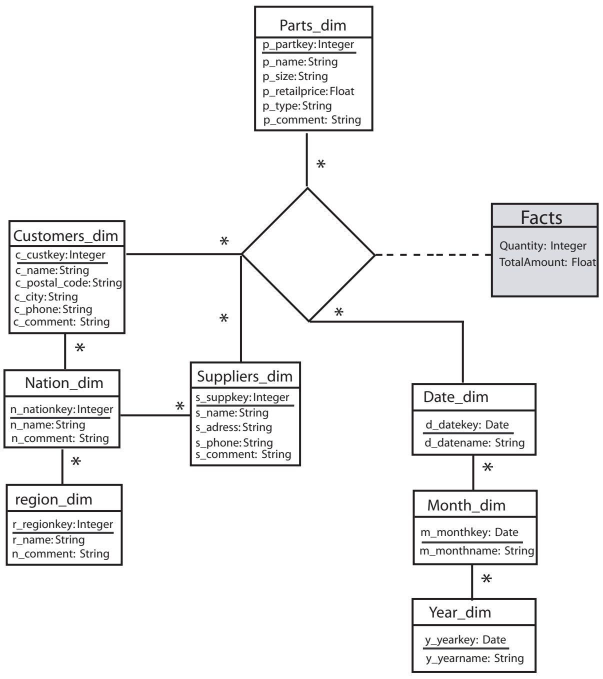

FIG. 4 – *XWB warehouse schema*

3.2.2 Schema instantiation.

We also selected TPC-H to benefit from its data generator, `dbgen`, a feature that does not exist in TPC-DS yet. The schema instantiation process is achieved in two steps: first, building the dimension XML documents, and then, building the facts document.

Dimension data are obtained from `dbgen` as flat files. They are then stored in the dimension XML documents (Figure 5 (c)). Dimension specifications, hierarchical levels and attribute names are obtained from the *dw-model.xml* document.

Facts are generated randomly (Algorithm 6). This process uses the notion of density introduced in DWEB, which helps in controlling the size of the fact document. A density of one indicates that all the combinations of dimension identifiers are present in the facts. Figure 5 (b) shows two examples of XML facts.

```
<dw-model>
        <FactDoc>
                <measure name="quantity" id=""/>
                <measure name="totalamount" id=""/>
                <dimension name="customers" path="custoers.dat">
                <dimension name="suppliers" path="suppliers.dat">
                <dimension name="date" path="date.dat">
                <dimension name="parts" path="parts.dat">
        </FactDoc>

        <dimensions id="customers" path="customers.dat">
                <Level id="customers">
                        <attribute name="c_custkey" type="Integer"/>
                        <attribute name="c_name" type="String"/>
                        <attribute name="c_adress" type="String"/>
                        ....
                </Level>

                <Level id="nation">
                        <attribute name="n_nationkey" type="Integer"/>
                        ...
                </Level>
                <Level id="region">
                ....
                </Level>
        </dimensions>
        <dimensions>
        ......
        </dimensions>
...
</dw-model>
```

(a) dw-model.xml sample

```
<FactDoc>
        <fact>
                <measure mes-id="quantity" value="250"/>
                <measure mes-is="amount" value="68"/>
                <dimension dim-id="customers" value="15">
                <dimension dim-id="suppliers" value="1">
                <dimension dim-id="date" value="25">
                <dimension dim-id="parts" value="30">
        </fact>
        ...
        <fact>
                <measure mes-id="quantity" value="98"/>
                <measure mes-is="amount" value="25.23"/>
                <dimension dim-id="customers" value="25">
                <dimension dim-id="suppliers" value="2">
                <dimension dim-id="date" value="55">
                <dimension dim-id="parts" value="87">
        </fact>
        ....
</FactDoc>
```

(b) facts.xml sample

```
<dimensions dim-id="customers">
                <Level id="customers">
                        <instance> ...
                        </instance>
                </Level>
                <Level id="nation">
                        <instance>
                                <attribute name="n_nationkey" value="1"/>
                                <attribute name="n_nationname" value="France"/>
                                ...
                        </instance>
                ...
                </Level>
        </dimensions>
        <dimensions>
</dimensions>
```

(c) sample dimension$_{customers}$.xml

FIG. 5 – *XWB XML documents*

3.2.3 Parameterization.

XWB's database parameters basically help users in controlling the warehouse size. Size (S) is actually controlled by the scale factor parameter (SF) inherited from TPC-H. It can be estimated as follows: $S = S_{dimensions} + S_{facts}$, where $S_{dimensions}$ is the size of dimensions, which does not change when SF is fixed, and S_{facts} is the size of facts, which depends on density.

The size of the dimension and fact XML documents may be estimated as follows:

$$S_{dimension} = \sum_{d \in D} |d|_{SF} \times nodesize(d)$$
$$\text{and } S_{facts} = \prod_{d \in D} |d|_{SF} \times density \times cellsize$$

where D is the set of dimensions, $|d|_{SF}$ the size of dimension d, $nodesize(d)$ the average node size in $dimension_d.xml$, and $cellsize$ the average fact node size.

Table 1 shows the size of the $facts.xml$ document for $SF = 1$ and $density = 1$. We consider in this example that the node size is 220 bytes.

```
For each c ∈ Customers_dim do
    For each p ∈ Parts_dim do
       For each s ∈ Suppliers_dim do
          For each d ∈ Date_dim do
             If Random(0, 1) ≤ Density then
                Quantity = Random(1, 10000)
                TotalAmount = Quantity × p.p_retailprice
                Create_Fact(p, s, d, Quantity, TotalAmount)
             End if
          End for
       End for
    End for
End for
```

FIG. 6 – *XWB fact generation algorithm*

|Customers_dim|	|Suppliers_dim|	|Parts_dim|	|Dates_dim|	S_{facts}(GB)
400	400	500	126	2065

TAB. 1 – *facts.xml size for $SF = 1$*

3.3 XWB workload

The XQuery language (Boag et al., 2004) allows the formulation of decision-support queries, unlike simpler languages such as XPath. Complex queries, including aggregation operations and join queries over multiple documents, may indeed be expressed with the *FLWOR* syntax. However, analytic queries are difficult to express and execute efficiently with XQuery, which does not include an explicit grouping construct comparable to the `group by` clause in SQL (Beyer et al., 2005), for instance. And though grouping queries are possible in XQuery, there are many problems with the results (Beyer et al., 2005). Hence, as many authors, we chose to implement an explicit `group by` clause to extend XQuery. Though this is not standard XQuery, most XML DBMSs feature Application Programming Interfaces (APIs) that make this modification easy (our Java `group by` function is also available to any users). Furthermore, this extension is so widely acknowledged as necessary that it should definitely make its way into the XQuery language. Figure 7 provides an example of analytic query that exploits a multiple `group by` clause.

We again inspired from TPC-H to design XWB's workload, which is currently composed of fifteen decision-support queries labeled $Q1$ to $Q15$. These queries exploit the warehouse schema through join, selection and grouping operations. Their specifications are provided in Table 2. They are presented in natural language for space constraints, but their XQuery formulation is available on demand.

Finally, we recommend to exploit this workload by applying TPC-H's execution protocol:

1. a load test (storing the warehouse in the XML DBMS);

2. a performance test that is executed twice (cold run and warm run) and that is subdivided in a power test and throughput test (Section 2).

```
Q1   for $a in //dimensionData/classification/Level
     [@node='customers']/node,
     $x in //CubeFacts/cube/Cell
     let $q := $b/attribute[@name='c_name']/@value
     let $q := $b/attribute[@name='c_postal_code']/@value
     where $a/attribute/@name='c_city'
     and $a/attribute/@value='Lyon'
     and $x/dimension /@node=$a/@id
     and $x/dimension/@id='customers'
     group by(p_name,@m_name, @d_name)
     return name='c_name', sum(quantity)
```

FIG. 7 – *Example of XML decision-support query*

Finally, note that updates are diversely taken into account in XML DBMSs, and XQuery's syntax does not feature them yet. Hence, we did not include any refreshing operation in XWB yet.

4 Conclusion and perspectives

XWB is one first proposal of XML decision-support benchmark. Since the context of XML data warehouses is not stable yet, we chose to root our work upon a well-known and widely used relational decision-support benchmark (TPC-H), and to favor simplicity in the design and development of our own benchmark. Thus, we can easily modify our tool if research in XML warehousing evolves or a standard emerges. XWB is currently relatively simple, but it is still under development. However, we successfully used it to experimentally validate an indexing strategy (Mahboubi et al., 2006b) and a view materialization strategy (Mahboubi et al., 2006a) for XML warehouses.

Furthermore, many enhancements are scheduled. First, we are probably going to expand the warehouse schema with more dimensions with hierarchies. We also envisage to propose, as an option, a constellation architecture with several fact documents. Of course, the workload will have to evolve as well, to take these changes into account.

Though the XQuery language is currently limited for formulating analytical and update queries, it is about to become the standard for querying XML data. However, we anticipate the development of already-identified extensions, namely regarding grouping queries and OLAP operators (Beyer et al., 2005; Borkar and Carey, 2004; Deutsch et al., 2004; Paparizos et al., 2002).

It is also important to include the ETL process in our workload (at least refreshing fuctions), and to design an execution protocol for XWB that is specific to XML systems.

Query	**Specification**
$Q1$	Number of sales for customers from Lyon grouped by part, month and day
$Q2$	Number of sales grouped by part, month and day
$Q3$	Number of sales grouped by part and supplier
$Q4$	Number of sales for parts grouped by region and city
$Q5$	Total quantity grouped by part and city
$Q6$	Sum of total amounts grouped by city and part
$Q7$	Number of sales grouped by city
$Q8$	Average quantity grouped by customer and city
$Q9$	Total quantity grouped by customer and city
$Q10$	Sum of total amounts grouped by year and part
$Q12$	Number of sales grouped by customer and year
$Q13$	Sum of $Q11$ results
$Q14$	Sum of $Q12$ results
$Q15$	Sum of total amounts grouped by supplier and month

TAB. 2 – *XWB workload specification*

References

Afanasiev, L., I. Manolescu, and P. Michiels (2005). MemBeR: A Micro-benchmark Repository for XQuery. In *3rd International XML Database Symposium on Database and XML Technologies (XSym 05), Trondheim, Norway*, Volume 3671 of *LNCS*, pp. 144–161.

Beyer, K. S., D. D. Chamberlin, L. S. Colby, F. Özcan, H. Pirahesh, and Y. Xu (2005). Extending XQuery for Analytics. In *2005 ACM SIGMOD International Conference on Management of Data (SIGMOD 05), Baltimore, USA*, pp. 503–514.

Boag, S., D. Chamberlin, M. Fernández, D. Florescu, J. Robie, and J. Siméon (2004). XQuery 1.0: An XML Query Language. W3C Working Draft, http://www.w3.org/TR/xquery/.

Böhme, T. and E. Rahm (2001). XMach-1: A Benchmark for XML Data Management. In *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW 01), Oldenburg, Germany*, pp. 264–273.

Böhme, T. and E. Rahm (2002). Multi-user Evaluation of XML Data Management Systems with XMach-1. In *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, VLDB 2002 Workshop EEXTT and CAiSE 2002 Workshop DTWeb. Revised Papers,*, Volume 2590 of *LNCS*, pp. 148–158.

Borkar, V. and M. Carey (2004). Extending XQuery for Grouping, Duplicate Elimination, and Outer Joins. In *XML 2004, Washington DC, USA*.

Boussaid, O., R. B. Messaoud, R. Choquet, and S. Anthoard (2006). X-Warehousing: An XML-Based Approach for Warehousing Complex Data. In *10th East-European Conference on Advances in Databases and Information Systems (ADBIS 06), Thessaloniki, Greece*, Volume 4152 of *LNCS*, pp. 39–54.

Carey, M. J., D. J. DeWitt, and J. F. Naughton (1993). The OO7 Benchmark. In *1993 ACM*

SIGMOD International Conference on Management of Data (SIGMOD 93), Washington, USA, pp. 12–21.

Darmont, J., O. Boussaid, and F. Bentayeb (2005a). DWEB: A Data Warehouse Engineering Benchmark. In *7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 05), Copenhagen, Denmark*, Volume 3589 of *LNCS*, pp. 85–94.

Darmont, J., O. Boussaid, J.-C. Ralaivao, and K. Aouiche (2005b). An Architecture Framework for Complex Data Warehouses. In *7th International Conference on Enterprise Information Systems (ICEIS 05), Miami, USA*, pp. 370–373.

Deutsch, A., Y. Papakonstantinou, and Y. Xu (2004). The NEXT Logical Framework for XQuery. In *30th International Conference on Very Large Data Bases (VLDB 04), Toronto, Canada*, pp. 168–179.

Hümmer, W., A. Bauer, and G. Harde (2003). XCube: XML for data warehouses. In *6th International Workshop on Data Warehousing and OLAP (DOLAP 03), New Orleans, USA*, pp. 33–40.

Mahboubi, H., K. Aouiche, and J. Darmont (2006a). Materialized View Selection by Query Clustering in XML Data Warehouses. In *4th International Multiconference on Computer Science and Information Technology (CSIT 06), Amman, Jordan*, pp. 68–77.

Mahboubi, H., K. Aouiche, and J. Darmont (2006b). Un index de jointure pour les entrepôts des données XML. In *6ème journées Extraction et Gestion des Connaissances (EGC 06), Lille, France*, Volume E-6 of *RNTI*, pp. 89–94.

Nassis, V., R. Rajugan, T. S. Dillon, and J. W. Rahayu (2004). Conceptual Design of XML Document Warehouses. In *6th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 04), Zaragoza, Spain*, pp. 1–14.

Paparizos, S., S. Al-Khalifa, H. Jagadish, L. Lakshmanan, A. Nierman, D. Srivastava, and Y. Wu (2002). Grouping in XML. In *XML-Based Data Management and Multimedia Engineering (EDBT/XMLDM 02), Prague, Czech Republic*, Volume 2490 of *LNCS*, pp. 128–147.

Park, B. K., H. Han, and I. Y. Song (2005). XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses. In *7th International Conference on Data Warehousing and Knowledge Discovery, (DaWaK 05), Copenhagen, Denmark*, Volume 3589 of *LNCS*, pp. 32–42.

Pokorný, J. (2002). XML Data Warehouse: Modelling and Querying. In *5th International Baltic Conference (BalticDB&IS 02), Tallin, Estonia*, pp. 267–280.

Rajugan, R., E. Chang, and T. S. Dillon (2005). Conceptual Design of an XML FACT Repository for Dispersed XML Document Warehouses and XML Marts. In *20th International Conference on Computer and Information Technology (CIT 05), Shanghai, China*, pp. 141–149.

Runapongsa, K., J. M. Patel, H. V. Jagadish, Y. Chen, and S. Al-Khalifa (2006). The Michigan benchmark: towards XML query performance diagnostics. *Information Systems 31*(2), 73–97.

Schmidt, A., F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu, and R. Busse (2003). Assessing XML Data Management with XMark. In *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, VLDB 2002 Workshop EEXTT and CAiSE*

2002 Workshop DTWeb, revised parers, Volume 2590 of *LNCS*, pp. 144–145.

Thomsen, E. (1998). Comparing different approaches to OLAP calculations as revealed in benchmarks. Intelligence Enterprise's Database Programming & Design. http://www.dbpd.com/vault/9805desc.htm.

TPC (1998). TPC Benchmark D Standard Specification version 2.1. Transaction Processing Performance Council. http://www.tpc.org.

TPC (2003). TPC Benchmark R Standard Specification version 2.2.0. Transaction Processing Performance Council. http://www.tpc.org.

TPC (2005a). TPC Benchmark DS (Decision Support) Draft Specification revision 32. Transaction Processing Performance Council. http://www.tpc.org.

TPC (2005b). TPC Benchmark H Standard Specification revision 2.3.0. Transaction Processing Performance Council. http://www.tpc.org.

Vrdoljak, B., M. Banek, and S. Rizzi (2003). Designing Web Warehouses from XML Schemas. In *5th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 03), Prague, Czech Republic*, Volume 2737 of *LNCS*, pp. 89–98.

Yao, B. B., M. T. Özsu, and J. Keenleyside (2003). XBench - A Family of Benchmarks for XML DBMSs. In *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web, VLDB 2002 Workshop EEXTT and CAiSE 2002 Workshop DTWeb, revised papers*, Volume 2590 of *LNCS*, pp. 162–164.

Yao, B. B., M. T. Özsu, and N. Khandelwal (2004). XBench Benchmark and Performance Testing of XML DBMSs. In *20th International Conference on Data Engineering (ICDE 04), Boston, USA*, pp. 621–633.

Résumé

Avec l'émergence d'XML comme le nouveau standard pour représenter les données, de nouvelles applications décisionnelles (entrepôts de données XML, principalement) sont développées. Pour assurer leur faisabilité, le problème de la performance doit être traité. La performance en général, et l'efficacité des techniques d'évaluation de performance en particulier, est habituellement évaluée à laide de bancs d'essais. Cependant, il n'existe pas à notre connaissance de banc d'essais XML décisionnel. Dans cette article , nous présentons XWB (*the XML Warehouse Benchmark*) qui comble ce manque. XWB est bâti sur un modèle de référence original pour entrepôts de données XML et propose un entrepôt XML de test et sa charge décisionnelle XQuery associée, qui sont dérivés du banc d'essais décisionnel reconnu TPC-H. Bien qu'à un stade de développement précoce, XWB nous a permis de tester l'efficacité de techniques d'indexation et de matérialisation de vues dans les entrepôts de données XML.

Mots clés: Entrepôts de données XML, Requêtes décisionnelles XQuery, Bancs d'essais XML, Evaluation de performance.