Pivoted Table Index for Querying Product-Property-Value Information

Hyunja Lee and Junho Shim
Dept of Computer Science
Sookmyung Women's University
52 Hyochangwon-gil, Yongsan-gu
Seoul, 140-742, Korea
Email: {hyunjalee, jshim}@sookmyung.ac.kr

Abstract—**The query for triple information on product–attribute (property)–value is one of the most frequent queries in e-commerce. In storing the triple (product–attribute–value) information, a vertical schema is effective for avoiding sparse data and schema evolution, while a conventional horizontal schema often shows better query performance, since the properties are queried as groups clustered by each product. Therefore, we propose two storage schemas: a vertical schema as a primary table structure for the triple information in RDBMS and a pivoted table index created from the basic vertical table as an additional index structure for accelerating query processing. The pivoted table index is beneficial to improving the performance of the frequent pattern query on the group properties associated with each product class.**

Index Terms—**Ontology, index, RDBMS, e-commerce, pivoted table, vertical schema**

I. Introduction

There are many studies on storage schema that manage data effectively and process queries efficiently [1, 2, 3, 5, 6, 8, 14, 20]. In recent work, a vertical schema (also known as a column-oriented schema and a narrow schema) has been preferred as a storage structure for web ontology data such as OWL and RDF/S. In particular, subject-property-object information (RDF/S) has been stored vertically in RDBMS tables because a vertical schema is in general advantageous for supporting multi-valued attributes and avoiding sparse data and schema revolution [1, 4, 16, 17, 18, 19]. For that reason, vertical schemas are more useful for many applications in web-based domains, including e-commerce, than are the conventional horizontal schemas (also known as wide-type schemas and row-oriented schemas), which have properties such as field name and instances of the product in a row.

Fig. 1 shows an example where subject–property (attribute)–object information of RDF/S is stored vertically in an RDBMS table. RDBMS has been suggested for effective and efficient management and storage of Product Ontology, which is a conceptualization of specifying product information in terms of classes, properties, relationships, and constraints [9, 11, 13].

The product–attribute–value information of Product Ontology is frequently queried and is generally a very large amount of data. In addition, a database of product ontology tends to expand continuously while adding more information for new products. In order to provide more efficient processing on a product–attribute–value type of query, the database may be clustered by the attributes associated with each product.

The primary author of this paper is Hyunja Lee, and the corresponding author is Junho Shim.

© 2010 ACADEMY PUBLISHER
doi:10.4304/jsw.5.2.160-167

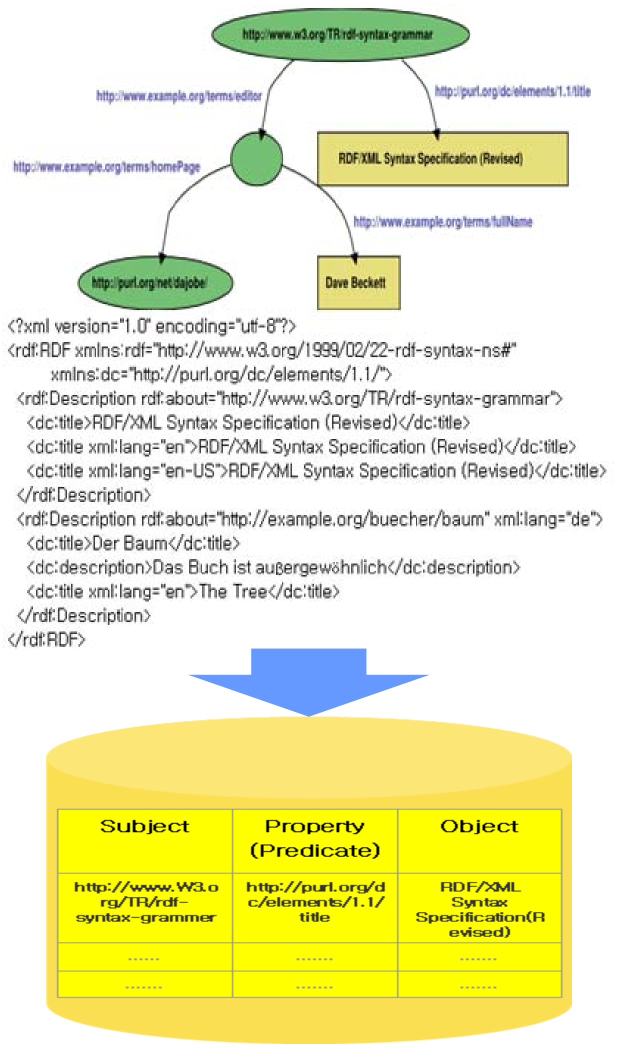


Figure 1. An example of storing triples of RDF(S)

These characteristics of product–attribute–value information can lead to problems if the information is stored in a conventional wide-view schema on RDBMS. First, since Product Ontology has many products and each product has many distinct attributes, if the triples are stored in a horizontal table, a considerable number of fields are required to represent all of the attributes. In the meantime, the schema may require its schema evolution and produce a number of null values for such attributes that are not associated with certain products.

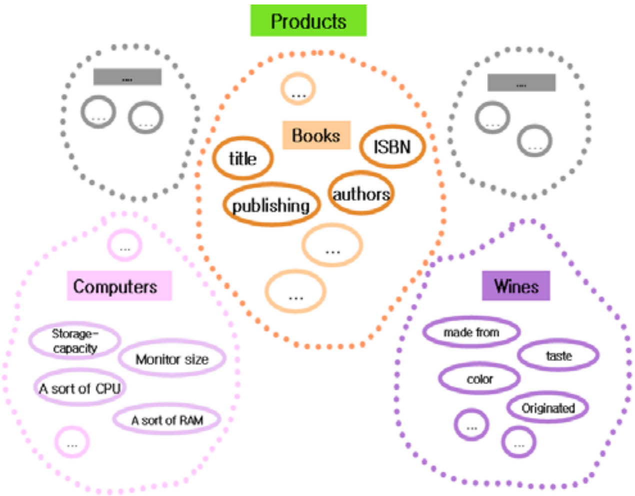


Figure 2. An example of attributes clustered by each product of Product Ontology

In this paper, henceforth, we suggest employing a vertical schema for querying product–attribute–value information as a basic storage structure on RDBMS. However, as shown in Fig. 2, properties may be clustered according to each product and the query on the property is often required as a property group. For example, in Fig. 2, books may be queried on a property group containing title, ISBN, and authors, while wines on a property group taste, color, and origin. Then, the wide-view table is more efficient than the vertical table in response to queries, since each product may have many instances. It may be beneficial to improve the performance for this frequent query of product–attribute–value and to be able to present the query results of property group queries without rewriting the query.

In the long run, two schemas are needed to meet the needs of all cases, so we suggest two storage schemas for product–attribute–value information of Product Ontology: a vertical schema as a basic structure for storing the triples of Product Ontology, and the index table that is created by pivoting the vertical table that stores the triples of Product Ontology. The index, once built, forms a horizontal schema (wide-view schema). Since the index table is created for each product, each table has a manageable number of columns in RDBMS. The pivoting algorithm that is given in SQL can be executed to create the index table.

The rest of this paper is organized as follows: Section 2 discusses related work. Section 3 illustrates the schema for storing product–attribute–value information and shows the pivoted table index for accelerating query performance. Section 4 evaluates the performance of our index, and Section 5 provides the conclusion.

© 2010 ACADEMY PUBLISHER

II. Related Work

In this section, we survey the previous works that discuss vertical schemas and schema conversion using a pivot function.

A. *Vertical Schema*

Agrawal et al. presented the vertical three-column scheme (object id–property–value) for representing e-commerce data that is rapidly evolving and sparsely populated [2]. The authors provided transformation algebra and techniques for implementing the scheme non-intrusively on top of a SQL database system based on Schema SQL, which is an extension to SQL that enables multi-database interoperability [15]. Their work is one of the pioneer works relating the vertical schema to a storage scheme for managing the e-commerce data.

Abadi et al. proposed the vertical portioning schema, which is created by rewriting the three-way vertical schema and has as many column tables as the number of unique properties in the data [1]. In [20], Willinson suggested an alternative storage scheme in the form of a property table comprised of one column containing a subject statement plus one or more columns containing property values for that subject in Jena. Jena uses a vertical schema for triples. Property tables augment but do not replace the triple storage, which is used for statements containing a predicate that has no property table. All the object values for a given property are stored in either a property table or triple storage, but never both.

Liu et al. proposed an indexing mechanism called the XML Table Index, which is more efficient than the path and value index approach for property groups of queries. The key idea behind building the XML Table Index is to pivot a group of property data into multiple columns in a relational table instead of storing each of the properties in a separate row, as is typically done in schema-agnostic solutions [14].

In the biomedical domain, heterogeneous data is managed with vertical schemas in RDBMS by schema transformation with pivoting [6, 18].

B. *Schema Conversion using Pivoting*

Cunningham et al. presented the pivoted table built by implementing the operation inside the RDBMS with pivot and unpivot operations included explicitly in the query language [7], rather than by post-processing the operation outside of query processing. The idea of employing the query language to build a pivoted table is adapted in our work.

There are some references in the algorithm for transforming vertical to horizontal or horizontal to vertical [2, 3, 15, 18]. Among these, Valentin et al. described three alternative algorithms for performing a pivoting table: using full outer joins, using left outer joins, and using hash tables and memory to perform the equivalent of multiple joins [18].

Broekstra et al. addressed two approaches, the left outer join and the pivot function, for vertical-to-horizontal translation and presented a comparison of the respective table sizes of each schema over the number of possible attributes [3].

III. Logiclal Schema For Product Information

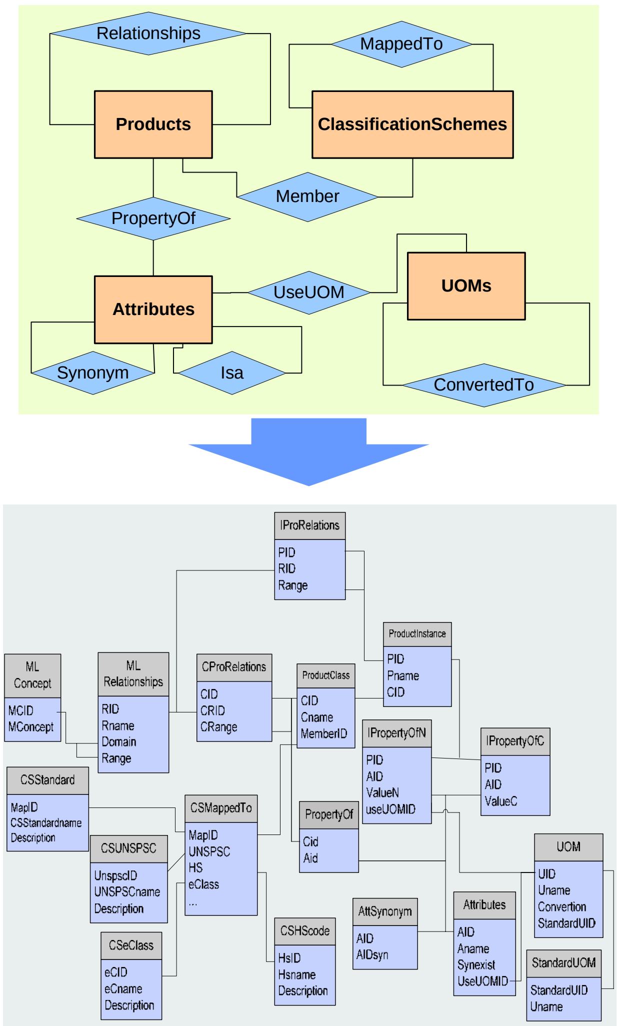


Figure 3. An example of a logical schema in RDBMS from the Product Ontology model

In this section, we briefly describe a Product Ontology model and its logical schema in RDBMS for storing Product Ontology, including product–attribute–value information. Fig. 3 shows an example that illustrates how to store a Product Ontology in a relational database schema.

Product Ontology's key concepts are products, classification schemes, attributes, and UOMs, and it includes various relationships among those concepts [9, 10, 11, 12]. The products, the most important concept, are the goods or services. The classification schemes and the attributes are used for the classifications and descriptions of products, respectively. The UOM is short for the unit of measurement, and it may be associated with the attributes.

Product Ontology includes the relationship between product class and a set of properties associated with each product on the conceptual level of the Product Ontology (i.e., class level), as well as relationships among the instances of product class and each of the sets of properties on the instance level of the Product Ontology.

Product–attribute–value is key information of Product Ontology and is very frequently queried, that is, the table containing the triple information is frequently accessed. Therefore, in order to query and manage the queries, it is important to design a proper schema for storing triples. We suggest two storage schemas for product–attribute–value:

- a vertical schema as a basic table structure for triples
- a pivoted table index for frequent pattern queries

A. Vertical Schema for Product-Property-Value

The idea of having a vertical schema for the product–property–value information is to having separate vertical tables for storing each piece of information, herein product, property of the product, and value of the property, rather than having a merged horizontal table for storing the triples. In a vertical schema, as in any relational schema, the tables are associated through the keys and foreign keys.

Fig. 4 shows a part of an exemplary RDB schema for the triples. The ProductClass table and the ProductInstance table contain information on the product class and all the instances of all the product classes, respectively. The Attributes table includes all the attributes lists. The PropertyOf table stores the relationships between the product classes and their associated properties, while the IPropertyOf table stores the relationships among all the instances of all the product classes and each of the associated properties of the instances. We adopt the two-column schema and the three-column schema for the information of product class–attribute and the information of instance (of product)–attribute–value in the Product Ontology, respectively.

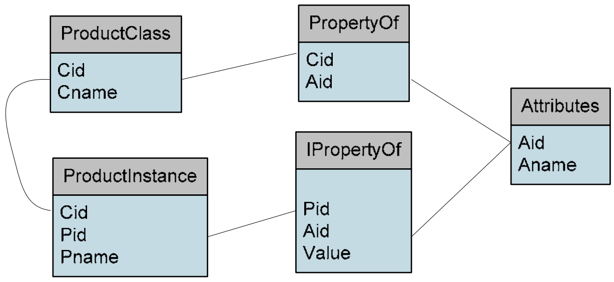


Figure 4. An example of a logical schema for product–attribute–value of product information

The detailed logical schema for the triples is as follows:

ProductClass (Cid, Cname)

ProductINstance (Cid, Pid, Pname)

PropertyOf (Cid, Rid, Range)

IPropertyOf (Pid, Aid,Value)

Attributes (Aid, Aname)

This vertical schema may have several benefits, including:

- In most cases, good query performance
- Freedom from schema evolution
- Non-null values
- Freedom from limitations in the number of columns manageable by RDBMS
- Better performance over value-centered schema by required access to only one table.

B. Creating the Pivoted Table Index

In this subsection, we show how a pivoted table index can be created from the basic vertical schema containing product–attribute–value information.

© 2010 ACADEMY PUBLISHER

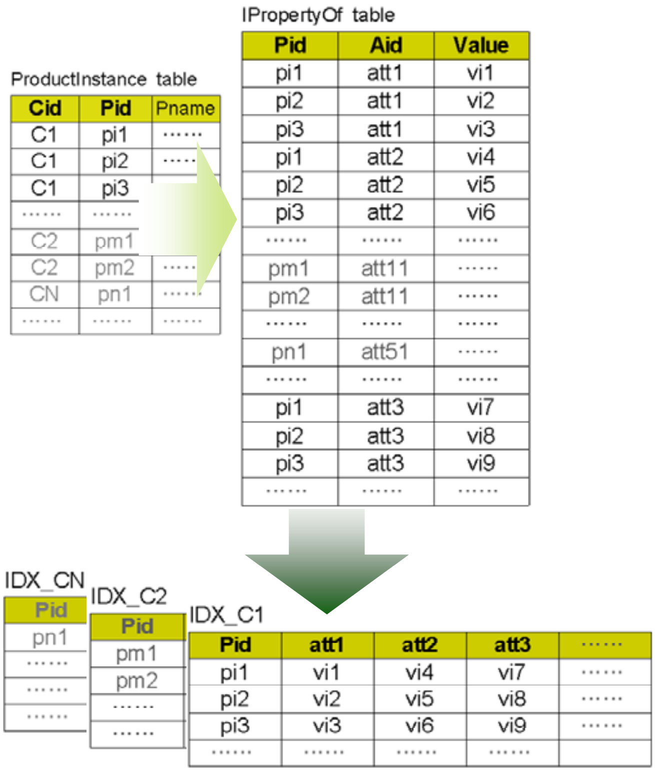


Figure 5. Transformation from a vertical schema to a pivoted property table index

Although the vertical schema is efficient for querying the triples in most case, the horizontal schema may be more advantageous than the vertical schema in terms of supporting the queries on property groups. In the e-commerce area, a query on a certain property group rather than a single property may be frequently found in general. For example, as in Fig. 2, when users want to find a book, they often give search values to a group of properties: *author*, *book title*, and *publisher*. Similarly, users often give their favorite *colors*, *tastes*, and *origination* values when they search for wines.

This is why we need the *pivoted table index* in addition to the vertical schema. The index tables are created from the vertical schema (i.e., the IPropertyOf table) of every product class. Therefore, the number of index tables is the same as the number of product classes.

Fig. 5 illustrates how the pivoted property index tables can be created from the vertical schema of product–property–value triples. For example, in Fig. 5, product class *C1* has three individual products, i.e., product instances, *pi1*, *pi2*, and *pi3*. This information is stored in the ProductInstance table. In the IPropertyOf table, you can find that the product *pi1* contains the properties *att1*, *att2*, *att3* of which the values are *vi1*, *vi4*, and *vi7*, respectively. Other products *pi2* and *pi3* have *vi2*, *vi5*, *vi8* and *vi3*, *vi6*, *vi9* for *att1*, *att2*, and *att3* properties. Then, a pivoted property table *IDX_C1* is created for the *C1* product class to contain all the product–property–value information of any product belonging to the *C1* products.

This pivoted property index has several benefits, including:

- better performance for frequent pattern queries on property groups of specific product classes
- ease of reporting in the query result
- ease of query writing

In Fig. 6, we present the SQL code for transforming from a vertical schema to a wide-type schema by using the pivot function. The pivoted table index can be created by SQL in the query processor (i.e., in the MS-SQL2003 version later, the pivot function enables in SQL code).

```
SELECT *
FROM
(SELECT * into IDX_"Cname" FROM IPropertyOf  ipr
 WHERE exists (SELECT ipr.pid
                  FROM ProductInstance ip, Products pd
                WHERE    ipr.pid=ip.pid    and    and
                         ip.cid=pd.pid   and   pd.cname=
                         'Cname')) as so
 pivot ( max(so.value) for  so.Aid  in ([Attribute List ]) ) as
p
```

Figure 6. SQL code for transforming to a horizontal schema

We assume that the value of each instance's attribute has a single value. In Fig. 6, the *'Cname'* is the product class name and *IDX_"Cname"* becomes the name of the pivoted index table. [*Attribute List*] becomes the column list of the pivoted index table (i.e., when implementation of the attribute list is obtained from the PropertyOf table). The pivot function is used with an aggregation function such as max or min (e.g., in Fig. 8, the *max* is used, and we assume that the null value is the smallest of all the values). Each aggregation function produces the value for each matching attribute, and null otherwise.

© 2010 ACADEMY PUBLISHER

IV. Performance Evaluation

We performed the experimental evaluation to see the performance of our suggested index and the pivoted table index.

Fig. 7 illustrates the comparison of query performance between a horizontal schema and a vertical schema in RDBMS. The experiment is conducted to show the different performance levels from using a vertical schema and a horizontal schema in querying product–attribute–value information. We measured the CPU elapsed time to see the query performance.

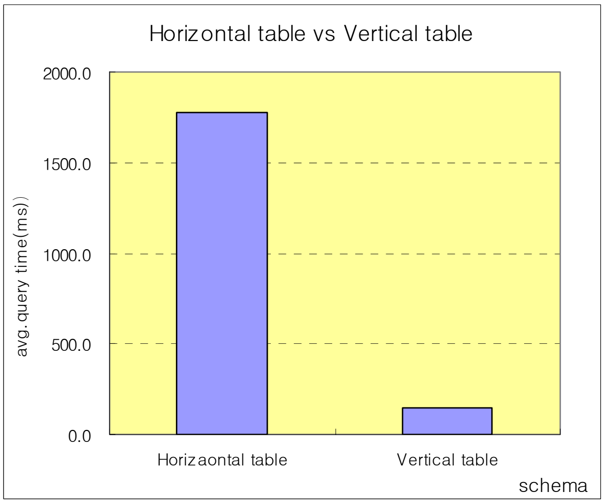


Figure 7. Comparison query performance between an H-table and a V-table

Fig. 8 shows that the triples are stored in a vertical table and a horizontal table, respectively. As shown in the figure, there are a lot of null values and sparse data in a conventional wide-view table since each product has distinct attributes.

In order to perform this experiment, we used MS-SQL2005 as RDBMS and Java 1.4v as a programming language. Our experimental platform was Windows XP running on a 2.4 GHz Intel Pentium 4 machine with 2 GB ram.

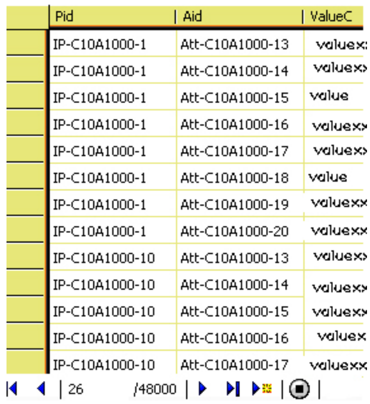
The following query patterns are used in the experiment.

- Retrieve all properties of a certain product.
- Retrieve properties of a certain product instance.

For the experiment, we use our synthetic data set, which has the following characteristics:

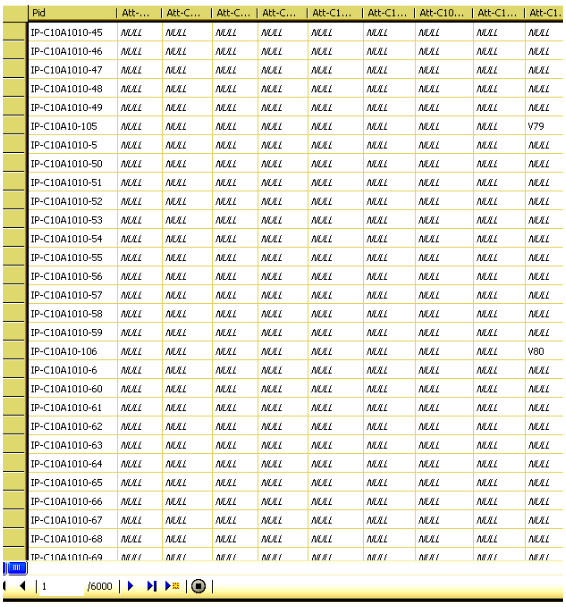
- Number of classes: 100
- Number of product instances: 60 per class
- Number of attributes: 8 per class

The attributes of each product class are different—there are no shared attributes among the products; in the vertical table, there are 48,000 rows and 3 columns, while the horizontal table has 6000 rows and 800 columns. Generally, the number of attributes (properties) of all the products is more than 5000 and the number of attributes of each product varies from around 8~25.



Pid	Aid	ValueC
IP-C10A1000-1	Att-C10A1000-13	valuex:
IP-C10A1000-1	Att-C10A1000-14	valuex›
IP-C10A1000-1	Att-C10A1000-15	value
IP-C10A1000-1	Att-C10A1000-16	valuex›
IP-C10A1000-1	Att-C10A1000-17	valuex›
IP-C10A1000-1	Att-C10A1000-18	value
IP-C10A1000-1	Att-C10A1000-19	valuexx
IP-C10A1000-1	Att-C10A1000-20	valuexx
IP-C10A1000-10	Att-C10A1000-13	valuex›
IP-C10A1000-10	Att-C10A1000-14	valuex›
IP-C10A1000-10	Att-C10A1000-15	valuex›
IP-C10A1000-10	Att-C10A1000-16	valuex
IP-C10A1000-10	Att-C10A1000-17	valuexx

26 /48000



Pid	Att-...	Att-C...	Att-C...	Att-C...	Att-C1...	Att-C1...	Att-C10...	Att-C1...	Att-C1.
IP-C10A1010-45	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-46	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-47	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-48	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-49	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A10-105	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	V79
IP-C10A1010-5	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-50	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-51	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-52	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-53	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-54	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-55	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-56	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-57	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-58	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-59	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A10-106	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	V80
IP-C10A1010-6	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-60	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-61	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-62	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-63	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-64	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-65	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-66	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-67	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-68	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IP-C10A1010-69	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

1 /6000

Figure 8. Snapshots of a vertical schema and a horizontal schema on storing triples, respectively

Current commercial RDBMS, such as Oracle9i and MS-SQL2005, can allow up to 1024 columns, so not all instances of all products can be stored in one horizontal table in RDBMS. Therefore, we limit our synthetic data set to 100 product classes and 8 attributes per product class. In

© 2010 ACADEMY PUBLISHER

total, then, the number of columns is 800, which does not exceed the maximum number of columns (1024 columns) allowed by RDBMS such as Oracle9i and MS-SQL.

As shown in Fig. 7, the query performance in the vertical schema is better than that in the horizontal schema.

The second experiment illustrates the query performance of our pivoted table index. This experimental platform was Windows XP running on a 2.4 GHz Intel Pentium 4 machine with 2 GB ram. We used MS-SQL2005 as RDBMS and java 1.4v as a programming language.

Our synthetic data set used on the implementation had the following characteristics:

- Number of product classes: avg. 500
- Number of attributes per product class: avg. 12 (min 8, max 20)
- The number of product instances per product class: avg. 150 (min 90, max 180)

The vertical table (i.e., IPropertyOf table) is just one table, while there are as many of our pivoted table indexes as there are product classes. The number of tuples of the vertical table affects the performance. Table 1 and Fig. 9 show that our pivoted property table index for querying triples outperforms the vertical table. In Table 1, when implemented with 100,000 tuples of the IPropertyOf table stored information of product–attribute–value, our pivoted table index outperforms the vertical table by a factor of 5, while with 2,000,000 tuples, it outperforms the vertical table by a factor of 48.

TABLE 1.

AVERAGE RESPONSE TIME TO QUERYING PRODUCT–ATTRIBUTE–VALUE INFORMATION

No. of tuples (of IPropertyOf table)	Average response time (ms)	
	Without index (IPropertyOf table access)	With index (pivoted table index table access)
100,000	326.33	60.50
500,000	918.00	60.50
1,000,000	1396.50	65.05
2,000,000	4381.50	90.00

From the result of the performance test, our index shows better performance for querying group properties associated with a product class.

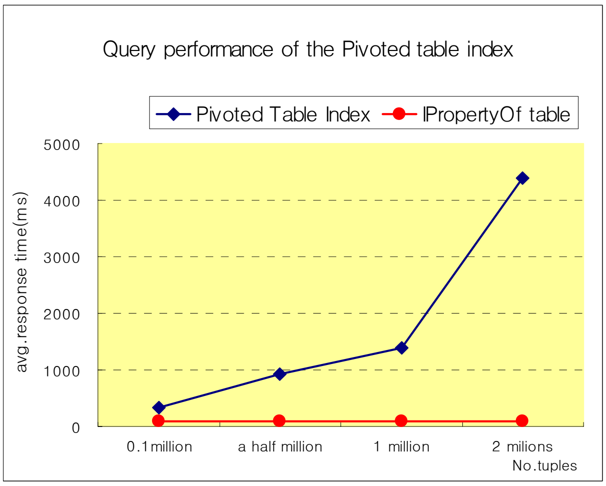


Figure 9. Performance comparison of a vertical schema (IPropertyOf table) and of a pivoted property table index for a group property query

V. CONCLUSION

We suggest a vertical schema for storing the triple information of product–attribute–value since the schema is beneficial to sparcity, schema evolution, performance, multi-value support and so on. In order to improve the performance for pattern queries for group properties of specific product classes, we present an auxiliary pivoted property table index created from the basic vertical table. We performed the experimental performance evaluation to see the performance of our proposed schema and indexing scheme. The experiment was run in a conventional database computing environment using the two leading RDBMS in industry. The results show that our index is efficient for queries on group properties associated with a product class.

ACKNOWLEDGMENTS

This Research was supported by the Sookmyung Women's University Research Grants 2008.

REFERENCES

[1] J. D. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach, Scalable Semantic Web Data Management Using Vertical Partitioning, In Proc. of 33rd International Conference on Very Large Data Bases (VLDB 2007), ACM, 2007.

[2] R. Agrawal, A. Somani, and Y. Xu, Storage and Querying of E-Commerce Data, In Proc. of 27th International Conference

© 2010 ACADEMY PUBLISHER

on Very Large Data Bases (VLDB 2001), Morgan Kaufmann, 2001.

[3] J. Beckmann, A. Halverson, R. Krishnamurthy, J. Naughton, Extending RDBMSs To Support Sparse Datasets Using An Interpreted Attribute Storage Format, In Proc. of 22nd International Conference on Data Engineering (ICDE 2006), IEEE Computer Society, 2006.

[4] J. Broekstra, A. Kampman and F. v. Harmelen, Sesame:A generic Architecture for Storing and Querying RDF and RDF Schema, In Proc. of International Semantic Web Conference (ISWS 2002), Springer, 2002.

[5] G. P. Copeland and S. N. Khoshafian. A decomposition storage model. In Proc. of the 1985 ACM SIGMOD International Conference on Management of Data, ACM, 1985.

[6] J. Corwin, A. Silberschatz, P. L. Miller, and L. Marenco. Dynamic tables: An architecture for managing evolving, heterogeneous biomedical data in relational database management systems. Journal of the American Medical Informatics Association, Vol. 14(1), 2007.

[7] C. Cunningham, C. A. Galindo-Legaria, and G. Graefe, PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS, In Proc. of 30th International Conference on Very Large Data Bases (VLDB 2004), Morgan Kaufmann, 2004.

[8] S. Khoshafian, G. P. Copeland, T. Jagodis, H. Boral, and P. Valduriez. A query processing strategy for the decomposed storage model. In Proc. of the Third International Conference on Data Engineering (ICDE 1987), IEEE Computer Society, 1987.

[9] H. Lee and J. Shim, Conceptual and Formal Ontology Model of e-Catalogs, In Proc. of the 6th International Conference on Electronic Commerce and Web Technologies (EC-Web 2005), Springer-Verlag, 2005.

[10] H. Lee and J. Shim, Product Ontology and OWL Correspondence, In Proc. of the IEEE Pacific Rim International Workshop on Electronic Commerce (IEEE-PRIWEC 2006), IEEE Computer Society, 2006.

[11] I. Lee, S. Lee, T. Lee, S.-g. Lee, D. Kim, J. Chun, H. Lee, and J. Shim, Practical Issues for Building a Product Ontology System, In Proc. of the International Workshop on Data Engineering Issues in E-Commerce (DEEC2005), IEEE Society, 2005.

[12] J. Lee and R. Goodwin, Ontology Management for Large-Scale E-Commerce Applications, Electronic Commerce Research and Applications, Vol. 5(1), Elsevier, 2006.

[13] T. Lee, I. Lee, S. Lee, S. Lee, D. Kim, J. Chun, H. Lee, and J. Shim, Building an Operational Product Ontology System, Electronic Commerce Research and Applications, Vol. 5(1), 2006.

[14] Z. H. Liu, M. Krishnaprasad, H. J. Chang, and V. Arora, XMLTable Index – An Efficient Way of Indexing and Querying XML Property Data, In Proc. of the 23rd International Conference on Data Engineering (ICDE2007), IEEE Computer Society, 2007.

[15] L. V. S. Lakshmanan, F. Sadri, and S. N. Subramanian. On Efficiently Implementing SchemaSQL on a SQL Database, In Proc. of 25th International Conference on Very Large Data Bases (VLDB 1999), Morgan Kaufmann, 1999.

[16] B. McBirds, Jena:A semantic web toolkit, IEEE Internet Computing, Vol. 6(6), IEEE Computer Society, 2002.

[17] Z. Pan and J. Heflin, DLDB: Extending Relational Databases to Support Semantic Web Queries, In Proc. of International Workshop on Practical and Scalable Semantic Web Systems, 2003.

[18] D. Valentin, P. Nadkarni, and C. Brandt, Pivoting approaches for bulk extraction of Entity-Attribute-Value data, Comp Meth Programs Biomed, Vol. 82(1), 2006.

[19] R. Volz, D. Oberle, S. Staab, and B. Motik, KAON SERVER - A Semantic Web Management System, In Proc. of the 12th International World Wide Web Conference (WWW 2003), ACM, 2003.

[20] K. Wilkinson. Jena property table implementation. In Proc. of the 2nd International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006), 2006.

[21] K. Wilkinson, C. Sayers, H. A. Kuno, and D. Reynolds, Efficient RDF Storage and Retrieval in Jena2, In Proc. of the International Workshop on Semantic Web and Databases (SWDB 2003), 2003.

Hyunja Lee received her B.S., M.S., and Ph.D. degrees in Computer Science from Sookmyung Women's University at Seoul, Korea, in 1996, 2004, and 2009 respectively. This work was performed when she was a Ph.D. student at Sookmyung Women's University. Her research interests include database, e-commerce, semantic web, ontology and any other web technology.

She is currently a POST-DOC at Université de Bourgogne, Dijon, France.

Junho Shim received his B.S. and M.S. degrees from Seoul National University at Seoul, Korea, in 1994 and the Ph.D. degree in Computer Science from Northwestern University at Evanston, Illinois, USA, in 1998. His research interests include database systems, data warehousing, product information, and e-commerce.

He is currently a PROFESSOR at Department of Computer Science in Sookmyung Women's University, Seoul, Korea. Previously he worked at Computer Associates International, New York, USA, and held assistant professor position with Drexel University, Pennsylvania, USA.

© 2010 ACADEMY PUBLISHER