A New Hybrid Schema-Sharing Technique for Multitenant Applications

Franclin S. Foping, Ioannis M. Dokas, John Feehan, Syed Imran
Cork Constraint Computation Centre
University College Cork
Ireland

E-mail: {f.foping,i.dokas,j.feehan,s.imran}@4c.ucc.ie

Abstract

This paper presents a new schema-sharing technique for multitenant applications. Our approach is built on top of the Extension table method and makes use of the native XML data support to store the additional data supplied by each tenant. Our proposed technique can be used to implement multitenancy on top of a standard relational database. This paper also describes an implementation of our approach using a real-world case study aiming at improving the communication channel of information related to drinking water quality parameters used by all stakeholders involved in the water treatment process in Ireland. These include the Environmental Protection Agency, the Health Service Executive, drinking water treatment plant staff, local authorities and everybody consuming water.

1. Introduction

1.1. The Software as a Service Business Model

The Software as a Service (SaaS) business model and its main advantages over on-premises software were presented in depth in [12, 15]. SaaS applications are deployed as hosted services and accessed over the Web. The idea of outsourcing software has been first explored in [6] through several experiences.

According to [28], the market share of SaaS applications is set to be more and more appealing as many companies are gradually considering investing on SaaS applications. SaaS big players include Microsoft, IBM, Google, Amazon as well as independent software vendors such as Oracle. SaaS applications offer various services at a reduced cost (sometimes free) compared to on-premises applications. SaaS applications are particularly suitable for distributed applications with users living in remote locations.

Unlike on-premises software which requires an up-front license fee, the pricing option of SaaS includes the license and hosting fees that are paid on a per-use basis. SaaS applications are entirely hosted and maintained on its provider's data center. Therefore, customers are not required to purchase additional hardware to run the application. SaaS applications run inside a Web browser and usually do not require the installation of further plugins.

As SaaS is getting more and more momentum in both the academia and the software industry, it has been used for many purposes. One of the first applications of the SaaS business model was noticed in customer relationship management with Salesforce.com [5]. It has also been used in the healthcare industry to develop an inter-organizational healthcare information systems [17]. [21] described an implementation of a digital campus system with the SaaS business model whereas [30] used SaaS to build an on-demand museum collection management.

1.2. Multitenancy

Multitenancy is a feature that allows a single instance of an application to handle several end-users at the same time. This result in the ability to consolidate several tenants within a single database and leverage the economy of scale. The main benefit of multitenancy is to reduce the operating costs of running software from the provider's perspective. This idea has been explored previously without any explicit connection with multitenancy [19].

Implementing multitenancy usually consists of mapping tenants' context into already existing patterns of relational databases. Several techniques have therefore been presented in the literature to implement multitenant databases: the shared machine, the shared process and the shared table processes [20]. The most interesting technique is the last one which aims at creating only once the application schema and mapping all tenants directly to this schema by making use of one of the available schema mapping techniques covered in Section 2.

Multitenancy brings about several issues on security, implementation challenges, customization and configurability, scalability, extensibility [2, 29]. A well-designed SaaS application should be optimized to support multitenancy, scalability and configurability [7]. These three issues were given a special attention in the same paper and were known as the *Three Headed Monster*. Scalability refers to the ability of an application to support an increasing number of users without noticing a significant performance overhead [18]. Customization is concerned with the support of specific features of users or meeting service level agreement by the means of configurations. Due to the distributed and shared nature of multitenant applications appropriate security policies should be devised to prevent unauthorized users from accessing other users' private data.

1.3. On Relational Databases

Relational databases were originally designed to address the high costs of deployment and maintenance for complex systems [9]. They recorded an outstanding success story in being adopted as the mainstream technology for both data centers and IT shops. Indeed, the market share associated to relational databases was worth billions of dollars in licensing revenue per year according to [26]. The universal dominance of relational databases can be explained by their simplicity, robustness, performance and scalability.

However, attempting to scale a huge number of accross server nodes makes the application overwhelming hence reducing the viability of relational databases for large distributed systems such as those found on the Web. Another drawback of relational databases is the lack of a "tenant" concept. As a consequence, they do not offer any support to handle the huge amount of metadata required by multitenant applications. Current relational databases implementations do not provide support for more sophisticated schema management.

1.4. Contributions of this paper

This paper aims at presenting a new hybrid schema-sharing technique for multitenant applications. Our approach is based on two existing schema-mapping techniques namely the extension tables and private tables approaches and make use of the native XML support provided by open source database management systems. Our approach addresses issues related to the customization of database. In this paper we follow the shared table pattern for implementing multitenant databases. The rationale of our approach is to separate the common contents that are used by all tenants to the specific content created by tenants in order to meet their various needs and waits. We show that this extension can be implemented in XML as recent projects in the field have paved the way for XML native support in relational databases. Our proposed method will be implemented in a real case study faced by water treatment plant practitioners in Ireland.

The rest of the paper continues with a background review of existing schema-mapping techniques in Section 2 followed by a full presentation of our schema-sharing technique in Section 4. Our case study will be presented in Section 3. The results of our implementation will be described in Section 5 and the paper will be concluded in Section 6.

2. Related Work

2.1. Chunk Folding

Chunk Folding (CF) is a schema-mapping technique introduced in [2]. The approach works by vertically partitioning logical tables into chunks that in turns are folded together into several physical multitenant tables and joined as needed. Our approach also focuses on devising a mechanism to handle data between the real physical tables and the tenant tables including options for tenant schema extension but can be implemented in open source relational database products such as PostgreSQL on the proviso that there is a native XML support.

2.2. Extension Tables

Extension tables were initially developed from the Decomposed Storage Model described in [11]. This technique consists of splitting up a table of n- columns into n 2-column tables that are joined through surrogate values. The main weakness of this technique lies in the way the actual partitioning is performed as it leaves them in naturally-occuring groups. Extension tables have been used to map object-oriented schemas to inheritance into the relational model [13].

2.3. Sparse Columns

As defined in [8], a sparse dataset consists of a large number of attributes with most objects having non-null values for a relatively small number of these attributes. Storing such a dataset can potentially decrease its performance due to the high number of NULL values it may contain. This approach has been implemented in most commercial on the shelf database systems such as Oracle, Microsoft SQL Server. The interpreted storage technique dominates in query efficiency and ease-of-use over the current horizontal storage and vertical schema approaches over a wide range of queries and sparse data sets [4] .

2.4. Pivot Tables

In this approach, the application actually owns the schema which in turn is mapped into generic structures in the database. In a pivot table, each field of each row in a logical table is supplied with its own row [1]. This additional row contains a flexible data type (usually VARCHAR) in which other data types can be easily casted to. The main advantage of this technique compared to sparse columns is the elimination of NULL values. However pivot tables require many columns of meta-data besides the actual data they may contain. Another drawback of this approach is the complexity of the reconstruction step, indeed building an n-column logical table requires $(n - 1)$ aligning joins. An implementation of pivot tables in HBase[1] was presented in [3].

2.5. Multitenant Shared Table

Described in [16], it aims at separating the common content from tenant specific information. This technique was built on top of the following principles: Enhance relational database management systems to support the concept of tenants at the database layer so that the database engine can actually bind a given tenant's request with her and thereby selecting the appropriate storage area. Secondly, only one schema instance is used per application.

2.6. XML Support for Relational Databases

The integration of XML with relational database systems has been explored in [27, 14]. This is achieved by either providing a native XML data type [23] or by inserting the XML document into the database as Large Object (Character or Binary) with fast insert and retrieval queries but at the expense of the overall performance. Another commonly used technique consists of shredding XML to relational tables. The XML data type is suitable to model each tenant specific information as it is loosely typed. The XML support has been implemented on several commercial database systems such IBM's pureXML [25].

3. Case Study

A recent guideline published by the Department of Environment in Ireland urged all local authorities within the country to publish water treatment data on their website in order to present to the general public all information regarding drinking water quality in their respective county.

However, the Environmental Protection Agency (EPA) which is the regulatory body in charge of drinking water issues has also stepped in and requested to local authorities to also submit these information to EPA so that the regulatory body may take appropriate measures such as getting in touch with the Health Service Executive (HSE) in case some parameters fall outside the acceptable range thereny causing a threat to people drinking that water. For instance, for a given water sample the amount of bacteria such as *E-Coli*, *cryptosporidium* and *coliforms* must be null in order to be considered safe for consumption. Otherwise, drinking water regulatory bodies (the EPA, local authority and the HSE) should work together to take defensive measures with the ultimate goal of averting an eventual outbreak.

As result, the EPA would like to provide a common Web portal in which all local authorities will instantly submit their data and the general user will be able to get detailed information on how well water treatment plants in their county are currently operating. At the same time, an HSE authority can also look at this data and request more information in order to carry out a further investigation. Water quality data include the current amount of bacteria such as cryptosporidium, water coagulants such as nitrate and aliminium sulfate and so forth.

Furthermore, during the publication process, a given water treatment plant manager can decide to provide additional information to the default parameters presented to him. For instance, a water treatment plant manager can decide to input data about the amount of arsenic. Hence the need to provide a support for extensibility in the database. The whole Web portal is used by many tenants at the same time hence its multitenancy property.

4. A New Hybrid Schema-Sharing Technique

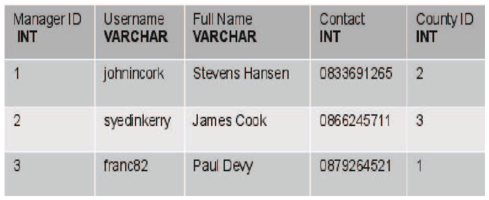
The main disavantage of relational databases in respect to our project was the lack of support of a tenant concept at the database layer. As a result, we had to store the tenant ID in the row of the common table. Our approach consists of two tables, one for the common content of tenants such as their ID, their full names as well as their contact details and another extension table specific to each tenant. The extension table consists of just two rows: one for the ID of the tenant and another one containing an XML document describing the additional information added by a given tenant. Using our approach each tenant is supplied with her own storage space to store her personal data. Our method merges idea from the extension tables introduced in Section 2 and stored additional information specific to each tenant in an XML document.

Figure 1 presents the schema of the common table shared by all tenants. Figure 3 on the other hand illustrates the extension table of each tenant in which managers with ID 2 and 3 respectively have added some information about their water treatment plants. Manager with ID 2 is in charge of a water treatment plant located in the county of Sligo and

[1]http://hadoop.apache.org/hbase

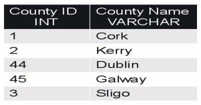
has amended the structure of the table by supplying information about the amount of *Arsenic* and *Aluminium* for that plant. Simultaneously, a user can actually see this extra information by selecting that particular water treatment plant on the public part of the website shown in Figure 4. Elsewhere, the manager with ID 1 has added information on the amount of *Turbidity* and *Chlorine* regarding the water treatment plant located in a town called *Inniscara*. In the common table, each tenant table contains a foreign key referring to the county where her currently lives. The county table itself is illustrated in Figure 2.

Furthermore, optimizing queries is now possible on a per-tenant basis as it is now possible to create indexes on the common tenant table. Also, extending the schema of the table becomes less daunting as the programmer will not be concerned of type safety. Querying customer extension data will not be hindered by the handling process of NULL values. Indeed, as each tenant is given her own extension table containing her specific data, there is no need to pad the table with NULL values.



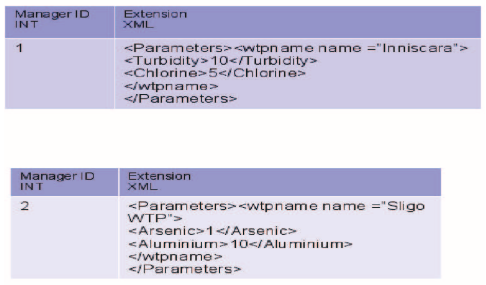
Manager ID INT	Username VARCHAR	Full Name VARCHAR	Contact INT	County ID INT
1	johnincork	Stevens Hansen	0833691265	2
2	syedinkerry	James Cook	0866245711	3
3	franc82	Paul Devy	0879264521	1

Figure 1. The common table



County ID INT	County Name VARCHAR
1	Cork
2	Kerry
44	Dublin
45	Galway
3	Sligo

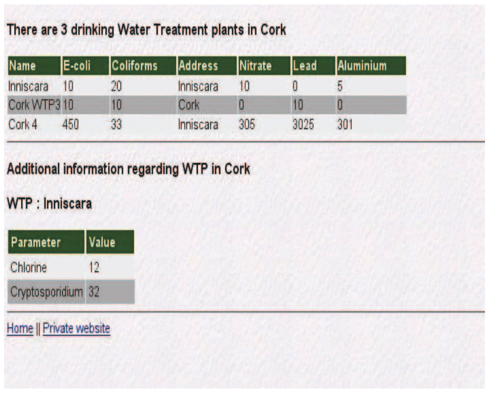
Figure 2. The county table



Manager ID INT	Extension XML
1	<Parameters><wtpname name ="Inniscara"> <Turbidity>10</Turbidity> <Chlorine>5</Chlorine> </wtpname> </Parameters>

Manager ID INT	Extension XML
2	<Parameters><wtpname name ="Sligo WTP"> <Arsenic>1</Arsenic> <Aluminium>10</Aluminium> </wtpname> </Parameters>

Figure 3. Extension tables



There are 3 drinking Water Treatment plants in Cork

Name	E-coli	Coliforms	Address	Nitrate	Lead	Aluminium
Inniscara	10	20	Inniscara	10	0	5
Cork WTP3	10	10	Cork	0	10	0
Cork 4	450	33	Inniscara	305	3025	301

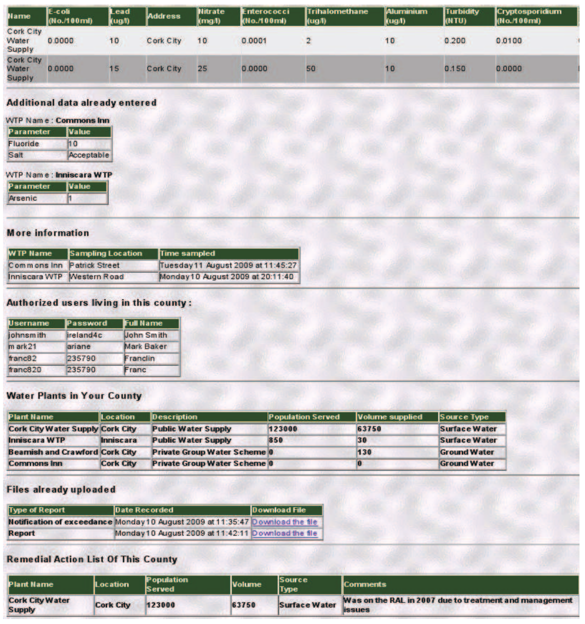
Additional information regarding WTP in Cork

WTP : Inniscara

Parameter	Value
Chlorine	12
Cryptosporidium	32

Home || Private website

Figure 4. A public view of the website



Name	E-coli (No./100ml)	Lead (ug/l)	Address	Nitrate (mg/l)	Enterococci (No./100ml)	Trihalomethane (ug/l)	Aluminium (ug/l)	Turbidity (NTU)	Cryptosporidium (No./100ml)
Cork City Water Supply	0.0000	10	Cork City	10	0.0001	2	10	0.200	0.0100
Cork City Water Supply	0.0000	15	Cork City	25	0.0000	50	10	0.150	0.0000

Additional data already entered

WTP Name : **Commons Inn**

Parameter	Value
Fluoride	10
Salt	Acceptable

WTP Name : **Inniscara WTP**

Parameter	Value
Arsenic	1

More information

WTP Name	Sampling Location	Time sampled
Commons Inn	Patrick Street	Tuesday 11 August 2009 at 11:45:27
Inniscara WTP	Western Road	Monday 10 August 2009 at 20:11:40

Authorized users living in this county :

Username	Password	Full Name
johnsmith	ireland4c	John Smith
mark21	ariane	Mark Baker
franc82	235790	Franclin
franc820	235790	Franc

Water Plants in Your County

Plant Name	Location	Description	Population Served	Volume supplied	Source Type
Cork City Water Supply	Cork City	Public Water Supply	123000	63750	Surface Water
Inniscara WTP	Inniscara	Public Water Supply	850	30	Surface Water
Beamish and Crawford	Cork City	Private Group Water Scheme	0	130	Ground Water
Commons Inn	Cork City	Private Group Water Scheme	0	0	Ground Water

Files already uploaded

Type of Report	Date Recorded	Download File
Notification of exceedance	Monday 10 August 2009 at 11:35:47	Download the file
Report	Monday 10 August 2009 at 11:42:11	Download the file

Remedial Action List Of This County

Plant Name	Location	Population Served	Volume	Source Type	Comments
Cork City Water Supply	Cork City	123000	63750	Surface Water	Was on the RAL in 2007 due to treatment and management issues

Figure 5. A manager's view of the system

5. Implementation details

Our approach was implemented with Open Source, free and platform-independent Web tools known as LAMP standing for Linux, Apache, MySQL, PHP and PostgreSQL. PHP 5.2 was used as the server scripting language while Apache 2.2 was our Web server. We choose PostgreSQL 8.3 for the database management system as it provides a native support for XML. Figures 4 and 5 show respectively a general user's view and a manager's.

The XML data was serialized before being inserted into the actual database and queried within the database server itself. Interestingly, PostgreSQL provides a built-in mechanism based on XPath 1.0 to query XML documents stored in a column of a table. Amongst other features, PostgreSQL

8.3 provides the ability to map XML documents to SQL databases, export the result of a query to an XML document. PostgreSQL 8.3 also provides specific methods to check the well-formedness of an XML document such as XMLPARSE and XMLSERIALIZE.

5.1. Security Patterns

Due to the distributed nature of our multi-tenant application, a key concern was to guarantee the data isolation of tenants so that private data of tenants will not be accessed by an unauthorized user at any time. Two main approaches have been reported in [29]: filter-based pattern in application level and the permission-based pattern. The latter will be explained in the next section.

The filter-based pattern works by adding the tenant ID to the WHERE clause of every SQL query. Recall in Section 4, we explain why we had to store the ID of a tenant in every table. This piece of data can be used here to filter the data belonging to a given tenant. Therefore, an SQL query to find out the drinking water parameters (the amount of E-coli, the name of the drinking water supply and so forth) added by a manager in Cork with a tenant ID equal to n will be stated as follows:

```
SELECT wtpname, ecoli, coliforms
FROM Plant
WHERE county='Cork' AND tenantID=n
```

5.2. Access control

Because a given tenant has many end-users with different roles and also the hierarchy within a tenant, our system has to be able to serve different users. For instance a local authority consists of several administrative tasks such as the county manager, the director of services, the department of water services and the department of environment. Each of these users has different tasks on the system. The individuals responsible of the implementation of the Minister's circular letter are part of the department of environment.

Our application was developed with the role-based access control model under which users are assigned to roles that are given permission to access resources on the system. This model has been presented in [24] and has been used to implement SaaS applications as reported in [17].

In our application, we have defined several roles for each tenant. For example, for a local authority, we had two roles: the senior engineer and the laboratory technician. The technician had a more restricted view of the system than the engineer.

6. Concluding Remarks

Throughout this paper, we have presented an innovative hybrid schema-sharing technique that can be used to implement multitenant applications on top of a standard relational database. Our approach works by splitting up the common content table shared by each tenant with the extension table containing additional information they may want to supply. These are stored in an XML document which in turn is stored in a PostgreSQL table and queried with XPath 1.0.

In this paper, we also present the previous techniques that have paved the way for our hybrid schema-sharing approach. Our technique made use of the Extension table method as well as the native XML support provided by any relational database management system. Our approach was implemented in a real-world case study aiming at improving the publication of drinking water parameters in Ireland. We have built a common and central platform where all stakeholders involved in the drinking water processing can use as a means of communication to share critical information on the status of water treatment plants.

Our approach suffers from some drawbacks that would be highlighted here. At first, the validity of the XML document stored in the PostgreSQL database cannot be checked against a schema. This is due to the fact this document is generated at runtime. In order to query the XML document, dedicated tools such as XQuery [10] need to be used. However, PostgreSQL 8.3 only supports XPath 1.0 which itself a serious limitation. Another problem is to accelerate these XPath queries in order to reduce the overhead associated with them as noticed in [22].

Another problem is related to the filter-based pattern that was used in our application. Although simple to implement, it presents serious security risks such as a SQL injection. Indeed, a malicious user might be tempted to alter the WHERE clause of a SQL query to insert the ID of another user. In order to handle this pitfall, the ID of tenant is never revealed to any user and is always used for internal purposes.

Finally, the role-based access control used in this implementation has some issues. Indeed, it is a static model and short of flexibility. It also lacks support for specifying control on individual users with some roles and on certain object instances. In collaborative environments such as ours, the role based access control model is not enough to have role permissions according to objects' types.

7. Acknowledgements

This paper is supported by the research project SCEWA (Grant No 2007-DRP-2-S5), funded by the Irish Environmental Protection Agency under the DERP grant scheme.

References

[1] R. Agrawal, A. Somani, and Y. Xu. Storage and querying of e-commerce data. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 149–158. Morgan Kaufmann Publishers Inc., 2001.

[2] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger. Multi-tenant databases for software as a service: schema-mapping techniques. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1195–1206, New York, NY, USA, 2008. ACM.

[3] S. Aulbach, D. Jacobs, A. Kemper, and M. Seibold. A Comparison of Flexible Schemas for Software as a Service. In *SIGMOD '09: Proceedings of the 2009 ACM SIGMOD international conference on Management of data*, pages 881–888. ACM, 2009.

[4] J. L. Beckmann, A. Halverson, R. Krishnamurthy, and J. F. Naughton. Extending RDBMS To Support Sparse Datasets Using an Interpreted Attribute Storage Format. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE06)*, page 58. IEEE Computer Society, 2006.

[5] J. R. Borck. CRM Anytime, Anywhere. *InfoWorld*, 2:30–39, 2005.

[6] O. P. Brereton and D. Budgen. Component-Based Systems: A Classification of Issues. *Computer*, pages 54–62, 2000.

[7] F. Chong and G. Carraro. Architecture Strategies for Catching the Long Tail. MSDN Library, Microsoft Corporation, 2006.

[8] E. Chu and J. Naughton. The Case for a Wide-Table Approach to Manage Sparse Relational Data Sets. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 821–832. ACM, 2007.

[9] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[10] W. W. W. Consortium. XQuery: The W3C query language for XML – W3C working draft. Available at http://www.w3.org/TR/xquery/, 2001.

[11] G. P. Copeland and S. N. Khoshafian. A decomposition storage model. In *Proceedings of the 1985 ACM SIGMOD international conference on Management of data*, pages 268–279. ACM, 1985.

[12] I. M. Dokas, R. J. Wallace, R. Marinescu, S. Imran, and F. S. Foping. Towards a Novel Early Warning Service for State Agencies: A Feasibility Study. In *Information Technologies in Environmental Engineering*, pages 162–175. Springer Berlin Heidelberg, 2009.

[13] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems, 5th Edition*. Addison-Wesley, 2007.

[14] D. Florescu and D. Kossmann. Storing and querying XML data using an RDMBS. *IEEE Data Eng. Bull*, 22(3):27–34, 1999.

[15] F. S. Foping, I. M. Dokas, J. Feehan, and S. Imran. On Using Software as a Service to Deploy an Early Warning Service. In *International Conference on Enterprise Information Systems and Web Technologies*, pages 161–168, Orlando, Florida, USA, 2009. ISRST.

[16] M. Grund, M. Schapranow, J. Krueger, J. Schaffner, and A. Bog. Shared table access pattern analysis for multi-tenant applications. pages 1–5, Sept. 2008.

[17] A. V. Hudli, B. Shivaradhya, and R. V. Hudli. Level-4 saas applications for healthcare industry. In *COMPUTE '09: Proceedings of the 2nd Bangalore Annual Compute Conference on 2nd Bangalore Annual Compute Conference*, pages 1–4, New York, NY, USA, 2009. ACM.

[18] M. Hui, D. Jiang, G. Li, and Y. Zhou. Supporting database applications as a service. In *ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 832–843, Washington, DC, USA, 2009. IEEE Computer Society.

[19] D. Jacobs. Enterprise software as service. *ACM Queue*, 6(3):36–42, 2005.

[20] D. Jacobs and S. Aulbach. Ruminations on multi-tenant databases. In *BTW*, pages 514–521, 2007.

[21] Z. Liang, X. Huang, L. Pan, and J. Li. A design and implementation of data access control in digital campus system using the rbac method. In *Information Technologies and Applications in Education, 2007. ISITAE '07. First IEEE International Symposium on*, pages 274–277, November 2007.

[22] P. Parys. Xpath evaluation in linear time with polynomial combined complexity. In *Principle Of Database Systems*, pages 55–64, 2009.

[23] N. Samokhvalov. XML Support in PostgreSQL. In *Proceedings of the Spring Young Researcher's Colloquium on Database and Information Systems*. CEUR, 2007.

[24] R. S. Sandhu, E. J. Coyne, and H. Feinstein. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, 1996.

[25] C. M. Saracca, D. Chamberlin, and R. Ahuja. Db2 9: pureXML - Overview and Fast Start. IBM, 2006.

[26] M. Seltzer. Beyond Relational Databases. *Communications of the ACM*, 51(7):52–58, July 2008.

[27] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, and D. J. D. J. F. Naughton. Relational databases for querying XML documents: Limitations and opportunities. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 302–314. Morgan Kaufmann Publishers Inc., 1999.

[28] E. Tenwolde. A Preliminary Look at Delivery Model Performance. Technical Report IDC No. 206240, IDC, 2007.

[29] Z. H. Wang, C. J. Guo, B. Gao, W. Sun, Z. Zhang, and W. H. An. A study and performance evaluation of the multi-tenant data tier design patterns for service oriented computing. In *e-Business Engineering, 2008. ICEBE '08. IEEE International Conference on*, pages 94–101, Oct. 2008.

[30] S. Wu and P. Chua. Museum collection management on-demand. In *ICEGOV '08: Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance*, pages 310–315, New York, NY, USA, 2008. ACM.