CSE 705 - SEMINAR REPORT
MULTI-TENANT DATABASES FOR SOFTWARE AS A SERVICE

Ranajn Bangalore Seetharama
rbs9@buffalo.edu

OVERVIEW

Multi Tenant Architecture is basically nothing more than "I am going to run multiple customers on one instance." So for example, Salesforce.com puts all of its customers on the same database, with just a simple partition in the database to make sure they can't look at each other's data. It gives them all relatively unsophisticated front-end functionality so they can tweak it much like you might tweak a My Yahoo interface. In the implementation of hosted business services, multiple tenants are often consolidated into the same database to reduce total cost of ownership.
Common practice is to map multiple single-tenant logical schemas in the application to one multi-tenant physical schema in the database. Such mappings are challenging to create because enterprise applications allow tenants to extend the base schema, e.g., for vertical industries or geographic regions.

The Internet has enabled an alternative model – Software as a Service (SaaS) – where ownership and management of applications are outsourced to a service provider. Businesses subscribe to a hosted service and access it remotely using a Web browser and/or Web Service clients. The fundamental limitation on scalability for this approach is the number of tables the database can handle. To get good consolidation, certain tables must be shared among tenants. Choice of appropriate schema is very essential to ensure optimal performance.

This paper describes a new schema-mapping technique for multi-tenancy called Chunk Folding, where the logical tables are vertically partitioned into chunks that are folded together into different physical multi-tenant tables and joined as needed. The database's "meta-data budget" is divided between application-specific conventional tables and a large fixed set of generic structures called Chunk Tables. Good performance is obtained by mapping the most heavily-utilized parts of the logical schemas into the conventional tables and the remaining parts into Chunk Tables that match their structure as closely as possible. Steps to obtain logical schemas from these chunk tables are as below:

1. All table names and their corresponding columns in the logical source query are collected.
2. For each table name, the Chunk Tables and the Meta data identifiers that represent the used columns are obtained.
3. For each table, a query is generated that filters the correct columns (based on the meta-data identifiers from the previous step) and aligns the different chunk relations on their Row columns. The resulting queries are all flat and consist of conjunctive predicates only.
4. Each table reference in the logical source query is extended by its generated table definition query.

Results of several experiments designed to measure the efficacy of Chunk Folding are described in detail with graphs of time and space complexities. Experiments are also conducted to study the performance of standard relational databases on OLTP queries formulated over Chunk Tables.

DETAILED COMMENTS

- Strengths: Since Chunk Folding has a generic component, it does not place limitations on consolidation or extensibility and it allows logical schema changes to occur while the database is on-line. At the same time, and in contrast to generic structures that use only a small, fixed number of tables, Chunk Folding attempts to exploit the database's entire meta-data budget in as effective a way as possible. Good performance is obtained by mapping the most heavily-utilized parts of the logical schemas into the conventional tables and the remaining parts into Chunk Tables that match their structure as closely as possible. Chunk Folding exhibits a response time improvement of more than 50 percent over conventional vertical partitioning.

- Weakness: Since the Chunk column is part of the primary index in the Chunk Folding case, there is overhead for fetching this column into the index buffer pools. A number of joins have to be performed to obtain the logical tables from chunk tables. This overhead produces up to 25% more physical data reads and a response time degradation of 10%.

- Various technical aspects such as schema mapping techniques with their consolidation (row wise) and extensibility (column wise) factors are described in detail. The formulation of query transformation techniques coupled with the time and space complexity analysis of experimental results makes the paper technically sound.

- This paper provides insight into the working of multi-tenant databases by analyzing the existing schema mapping techniques, describing their weaknesses and proposing a new schema mapping technique which would overcome the shortcomings of the existing conventional techniques. Schema mapping techniques used by most hosted services today provide only limited support for extensibility and/or achieve only limited amounts of consolidation. The 'Chunk-folding' schema-mapping technique proposed here allows the creation of an arbitrary number of tables with arbitrary shapes and thus do not place limitations on consolidation (row wise) or extensibility (column wise).

- Initial discussions during presentation were about the new aspects introduced by 'Chunk folding', its advantages and overhead introduced by this schema mapping technique. Main concerns were about the overhead introduced by several indexed joins to construct the logical tables from chunk tables. A slight overhead is acceptable in the view of the extensibility needed. There was also discussion regarding the query transformation and the efficiency of various query engines in optimizing nested joins. Overall observation revealed that the response time degradation could be optimized to stay within bounds to provide desired extensibility.