Database Technology for SaaS
(Software as a Service)

Multi-Tenant Database Enhancements
Quality of Service Enabled Databases

Alfons Kemper
Fakultät für Informatik
Technische Universität München
Co-Authors and Papers

- Stefan Aulbach, Torsten Grust, Dean Jacobs (SAP), Alfons Kemper, Jan Rittinger
  Multi-Tenant Databases for Software as a Service
  presented at ACM SIGMOD 2008 (SIGMOD 2008),
  June 9 - 12, 2008, Vancouver, BC, Canada

- Stefan Aulbach, Dean Jacobs, Alfons Kemper, Michael Seibold
  A Comparison of Flexible Schemas for Software as a Service
  ACM SIGMOD 2009 (SIGMOD 2009),
  June 29 - July 2, 2009, Providence, RI, USA

- Daniel Gmach, Stefan Krompass, Andreas Scholz, Martin Wimmer, Alfons Kemper
  Adaptive Quality of Service Management for Enterprise Services
  ACM Transactions on the Web (TWEB), Vol. 2, No. 1, Article 8, February 2008

- Stefan Krompass, Daniel Gmach, Andreas Scholz, Stefan Seltzsam, Alfons Kemper
  Quality of Service Enabled Database Applications
  Service Oriented Computing - ICSOC 2006: Fourth International Conference on
  Service Oriented Computing
  December 4 - 7, 2006, Chicago, Illinois, USA
  Lecture Notes in Computer Science (LNCS), Vol. 4294, pages 215-226
Outline

• **Overview of SaaS Applications & Market**
  - Cloud versus SaaS

• **Multi-Tenant Database Enhancements**
  - Schema Design
  - Performance Characteristics

• **Quality of Service Enabled Databases**
  - SLA Basics
  - Dynamic Prioritization of Requests
  - Mixed Workloads
    - Real Time Business Intelligence
Abgrenzung
SaaS ↔ Cloud Computing
Utility Computing Classes

- **EC2**
  - Virtual machines

- **Azure**
  - .Net libraries

- **Google AppEngine**
  - Application framework
  - Too constrained?
Obstacles/Opportunities for Cloud Computing

1. Availability
2. Data Lock-in
3. Data Confidentiality
4. Data Transfer Bottleneck
5. Performance Unpredictability
6. Scalable Storage
7. Bugs in Large Scale Distributed Systems
8. Scaling Quickly
9. Reputation Fate Sharing
10. Software Licensing

Multi-Tenant Databases for Software as a Service

Dealing with Highly Varied Data
Multi-Tenant Databases (MTD)

- Consolidating multiple businesses onto same operational system
  - Consolidation factor dependent on size of the application and the host machine

- Support for schema extensibility
  - Essential for ERP applications

- Support atop of the database layer
  - Non-intrusive implementation
  - Query transformation engine maps logical tenant-specific tables to physical tables
  - Various problems, for example:
    - Various table utilization ("hot spots")
    - Metadata management when handling lots of tables
Classic Web Application

- Pack multiple tenants into the same tables by adding a tenant id column
- Great consolidation but no extensibility

| TenId | AcctId | Name | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1</td>
<td>Acme</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>Gump</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>Ball</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>1</td>
<td>Big</td>
<td></td>
</tr>
</tbody>
</table>
Private Table

• Each tenant gets his/her own private schema
  – No sharing
  – SQL transformation: Renaming only

• High meta-data/data ratio
  – Low buffer utilization
Handling Lots of Tables

• Simplifying assumption: No extensibility

• Testbed setup:
  − CRM schema with 10 tables
  − 10,000 tenants are packed onto one (classic) DBMS
  − Data set size remains constant

• Parameter: Schema Variability
  − Number of tenants per schema instance

• Metrics:
  − Baseline Compliance: 95% percentile of “classic” Web Application configuration (SV 0.0)
  − Throughput [1/min]
Handling Lots of Tables – Results

Query Percentage

100
90
80
70
60
50
0
0.5 0.65 0.8 1

Transactions/Minute

8000
6000
4000
2000
0
0.5 0.65 0.8 1

Buffer Hit Ratio (%)

100
95
90
85
80
0
0.5 0.65 0.8 1

10 fully shared Tables
100.000 private Tables

Schema Variability
Data
Index
Extension Table

- Split off the extensions into separate tables
  - Additional join at runtime
  - Row column for reconstructing the row
- Better consolidation than Private Table layout
  - Number of tables still grows in proportion to number of tenants
Universal Table

• Generic structure with VARCHAR value columns
  – \( n \)-th column of a logical table is mapped to \( ColN \) in an universal table
  – Extensibility

• Disadvantages
  – Very wide rows \( \rightarrow \) Many \( NULL \) values
  – Not type-safe \( \rightarrow \) Casting necessary
  – No index support

<table>
<thead>
<tr>
<th>Tenant</th>
<th>Table</th>
<th>Col1</th>
<th>Col2</th>
<th>Col3</th>
<th>Col4</th>
<th>Col5</th>
<th>Col6</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>0</td>
<td>1</td>
<td>Acme</td>
<td>St. Mary</td>
<td>135</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>2</td>
<td>Gump</td>
<td>State</td>
<td>1042</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>1</td>
<td>Ball</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>42</td>
<td>2</td>
<td>1</td>
<td>Big</td>
<td>65</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Pivot Table

• Generic type-safe structure
  – Each field of a row in logical table is given its own row.
  – Multiple pivot tables for each type (int, string, e.g.)
  – Eliminates handling many NULL values

• Performance
  – Depends on the column selectivity of the query (number of reconstructing joins)
Row Fragmentation

• Possible solution for addressing table utilization issues
  – Various storage techniques for individual fragments
  – Hunt for densely populated tables
• Idea: Split rows according to their “popularity”
Chunk Table

- **Generic structure**
  - Suitable if dataset can be partitioned into dense subsets
  - Derived from Pivot table

- **Performance**
  - Fewer joins for reconstruction if densely populated subsets can be extracted
  - Indexable
  - Reduced meta-data/data ratio dependant on chunk size
Row Fragmentation

• Combine different schema mappings for getting a best fit
  – Mixes Extension and Chunk Tables
  – Each fragment can be stored in an optimal schema layout

• Optimal row fragmentation depends on, e.g.
  – Workload
  – Data distribution
  – Data popularity

<table>
<thead>
<tr>
<th>AccountRow</th>
<th>Tenant</th>
<th>Row</th>
<th>Aid</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>0</td>
<td>1</td>
<td>Acme</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>2</td>
<td>Gump</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>0</td>
<td>1</td>
<td>Ball</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>0</td>
<td>1</td>
<td>Big</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ChunkRow</th>
<th>Tenant</th>
<th>Table</th>
<th>Chunk Row</th>
<th>Int1</th>
<th>Str1</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>135</td>
<td>St. Mary</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1042</td>
<td>State</td>
</tr>
<tr>
<td>42</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>65</td>
<td>–</td>
</tr>
</tbody>
</table>
Querying Chunk Tables

• Query Transformation
  – Row reconstruction needs many self- and equi-joins
  – Can be automatically translated

• Compilation Scheme:
  1. Collect all table names and their corresponding columns from the logical source query
  2. For each table, obtain the Chunk Tables and the meta-data identifiers representing the used columns
  3. For each table, generate a query that filters the correct columns (based on the meta-data identifiers) and aligns the different chunk relations on their ROW column.
  4. Each table reference in the logical source query is extended by its generated table definition query
Join Overhead Costs

Join Overhead

No aligning joins
Quality of Service Enabled Database Applications

Stefan Krompass, Daniel Gmach, Andreas Scholz, Stefan Seltzsam, Alfons Kemper
Introduction
Service Level Agreements (SLAs)

- Contracts between service provider and client for Web Service directly invoked by the client
  - Challenge: provide end-to-end quality of service control
Static Prioritization

<table>
<thead>
<tr>
<th>Priority</th>
<th>Customer A</th>
<th>Customer B</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>medium</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SLA Conformance</th>
<th>Customer A</th>
<th>Customer B</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>90</td>
<td>95</td>
</tr>
</tbody>
</table>
Limitations of the Static Prioritization

- SLA (taken from TPC-C)
  - 90% of all transactions have to be processed in less than 5 seconds

- Static prioritization no longer sufficient
  - High priority customers overachieve their SLAs
Adaptive Penalties

Customer B with higher priority than A

Priority

<table>
<thead>
<tr>
<th></th>
<th>Customer A</th>
<th>Customer B</th>
</tr>
</thead>
<tbody>
<tr>
<td>medium</td>
<td>high</td>
<td>high</td>
</tr>
</tbody>
</table>

SLA Conformance

<table>
<thead>
<tr>
<th></th>
<th>Customer A</th>
<th>Customer B</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
SLA Penalty

Process 90% of all requests in less than 5 seconds
Penalty $900 for each 10% of underfulfillment
Maximum penalty: $1800
Evaluation period: one month
Quality of Service Model – SLA

Two-steps

• Compute penalty for an individual request
  – Opportunity costs
  – Marginal gain

• Compute deadline constraints for individual request
Opportunity Costs

Model the danger of falling to a lower service level
Marginal Gain

Models the chance of re-achieving a higher service level
Adaptive Penalty

Maximum of opportunity costs and marginal gain
Penalty for Individual Requests

![Graph showing penalties for individual requests based on service level conformance. The graph includes different service levels and the associated penalties, with a focus on the current SLA conformance.]
Time Constraints for Individual Requests

Deadline constraint for query $q_2$
Deadline constraint for query $q_1$

Deadline constraint for the transaction

Average running time monitored from previous transaction invocations
Deadline for $q_1$
Architecture

- SLA Component
  - Processing time
  - Penalty function
  - Penalty-carrying request

Client
Dual Queue Scheduling

Requests of active transactions

Queue A

Requests of new transactions

Queue B
SLA Conformance – Static Prioritization

![Graph showing SLA conformance across terminals with different priority levels. The highest priority is represented in yellow, medium in grey, and low in brown. The graph indicates a trend of decreasing conformance with increasing terminal number.](image-url)
Mixed workloads

- Business analysis
- Order entry
- Maintenance
- Customer relations
- Database
- Administrator
- Sales
Workload management

![Diagram of workload management system]

- Workload Objectives
- Workload Queries
- Policy controller
  - Admission Controller
  - Scheduler
  - Execution Controller
- DBMS
  - Database Executor
  - Performance monitor

Control loops:
- Business control loop
- Query control loop
- Policy control loop
Ongoing Work

• Adaptive virtualized infrastructure
  – Including database and application servers
  – DFG Kooperationsprojekt mit Wirtschaftsinformatik-Kollege Martin Bichler

• Build multi-tenancy support into the database management system
  – Schema mapping
  – QoS-handling
  – Schema evolution
  – Data sharing

• Build Main Memory Database System
  – OLTP
  – OLAP