STORED PROCEDURE LANGUAGE EXTENSIONS FOR COUPLING WEB SERVICES MANAGERS AND DBMS

Michael Goul
Arizona State University

Tim Chenoweth
Arizona State University

Andy Philippakis
Arizona State University

Stored Procedure Language Extensions for Coupling Web Services Managers and DBMS

Michael Goul, Tim Chenoweth, and Andy Philippakis
School of Accountancy and Information Management
College of Business
Arizona State University
Michael.Goul@asu.edu Tim.Chenoweth@asu.edu
Andy.Philippakis@asu.edu

Abstract

With the promises of a web services evolution implying dramatic changes in enterprise deployment and management of applications, strategies for coupling web services managers with database management systems will likely abound in the near future. The tracking and reporting of externally acquired web services utilization performance data, status monitoring data, bandwidth utilization data and memory utilization data are similar to traditional database monitoring and management approaches. This similarity suggests potential advantages to the tight coupling of web service managers with database management systems. However, several new extensions to a database management system's stored procedure language features are necessary. These extensions are necessary due to the dynamic nature of web services utilization over the Internet, the nature of web service invocation as a long-lived transaction, the need for an inclusion of new mechanisms to recover from web service failure, and other idiosyncrasies associated with NetCentric computing.

Introduction

Web service publication and associated UDDI registration is well underway as new integrated development environments such as Microsoft .NET simplify associated steps and procedures, and emerging brokerages exemplified by http://www.salcentral.com make it possible to search for and link to published web services ready for incorporation into an organization's applications. Gartner has coined the term, "composite application," to refer to any organizational application that embeds the utilization of web services into its operational logic [2001]. The suite of an organization's applications and the associated set of web service dependencies have been referred to as an organization's "web services-based information systems supply chain" [Goul and Hershauer, 2002]. To manage the supply chain, researchers and practitioners are beginning to use the phrase "Web Services Manager" to denote the software and hardware necessary to provide a centralized monitor of an organization's web services-based information systems supply chain. It is compelling to consider the commonalities between a web services manager and that of similar monitors and associated reporting facilities designed for the database management system (DBMS) context. For example, DBMS monitors track and facilitate reporting on query performance and real-time DBMS status including elements of bandwidth utilization and memory utilization. Similar tracking and reporting requirements are now viewed as an important pretext for the wide-scale adoption of web services. Oracle, for example, has recognized this aspect as indicated in the following:

> "[Oracle's] Enterprise Manager provides facilities for managing and monitoring web services. All the tools it provides for monitoring application status, performance, CPU usage, memory utilization, and throughput can be used to monitor web services deployed on Oracle9iAS. And monitoring web services will become vital as those services become a key part of any company's business. As web services become an integral part of a business's operations, it's critical that you be able to manage, monitor, and analyze web services usage and performance" [Edwards, 2002].

Issues in Coupling Web Services and DBMS Managers

There are important differences between managing, monitoring and analyzing enterprise DBMS applications and web services-based applications. Several of these differences are as follows:

1. In the DBMS context, one doesn't necessarily track processes outside the organization's boundaries. For example, most DBMS are tuned to perform well within an organization, and associated tuning strategies and tactics assume the existence of parameters under direct control of that organization. This is in contrast to the NetCentric [from March, et al., 2000] environment where web services are invoked outside of the organization, the CPU upon which a web service executes may not be under the organization's control, and the infrastructure to transport the inputs to and outputs from a web service is outside direct organizational control. This implies a need for handling web service failures due to uncontrollable network congestion, high external CPU loads, etc. that may result in a controllable series of repeated invocation attempts as well as code segment checkpoints for managing failure loop-back.

2. There are not typically agreed upon service levels (e.g., QoS) in contracts within a DBMS, but ongoing contract performance with respect to external providers must be managed and monitored within the web services context. Therefore, contract specified delay parameters should be embedded in the DBMS supporting the composite applications, along with the associated logic to handle contract exceptions.

3. The alerts associated with DBMS triggers are typically domain range controls placed on attributes such that when an update is out of a pre-specified range, an alert is raised, and an intervention is required to approve or disapprove of the update. In contrast, the trigger of an alert in a web services manager may require the discovery of a new web service from a different provider and/or the procurement of the web service along with a negotiated service level contract. These alerts may also require different levels of human intervention. For example, an organization may choose to have a purchasing agent engaged in web service procurement, or if there are designated secondary web service providers, an organization may choose to automate the handling of an alert by switching in real-time to a secondary provider.

Requirements for Extensions to DBMS Stored Procedure Languages

DBMS typically include a capability to build and maintain stored procedures to handle basic extensions to the relational model such as transitive closure and triggers/alerts. The DBMS environment manages the stored procedures and typically prescribes a set of reserved words for functions relating to update and delete operations, shared procedure ownership, etc. Stored procedure languages like Oracle's PL/SQL combine SQL commands with procedural commands for manipulating variables, evaluating if-then logic structures, and looping controls such as repeat-until, while and for constructs. Stored procedure executions are tightly coupled with the DBMS environment, and therefore tracking, monitoring and performance reporting to support managing the DBMS (and its associated stored procedures) requires no additional functionality. However, given the differences between DBMS applications and the web services environment, there is a need for new constructs.

The target for extensions to the DBMS for handling the web services environment is the stored procedure. Since stored procedures are intended to handle all DBMS issues outside the mathematics of the relational algebra/calculus, they represent an appropriate target for additional functionality designed to handle the key environmental differences discussed in the previous section. The first of these differences involves timeout parameters. Within a stored procedure, it is necessary to be able to specify how long a composite application should wait before raising an alert regarding the failure of a web service. Programmatically, this structure would include a specific statement preceding a web services invocation, e.g., TIMEOUT 5000; <web-services-call>; <…> .The timeout specifies the maximum number of seconds of delay expected before an alert is raised. Within this timeout limit, it is desirable to control the number of retries that should be attempted before an exception is raised. For example, after 10 retries to invoke a web service, and even within the timeout limit, an exception should be raised. In addition, there is a need for a code entry point to designate the beginning of the code segment on any retry failure. Thus, for a retry, an extended stored procedure language needs a command to specify the number of retries and the exact code re-entry point for initiating a retry on a failure. For example, the following code segment includes the necessary additional functionality.

```
TIMEOUT 5000;
RETRY 5 TIMES THEN RETURN(exception);
ON FAILURE RETRY <code entry point>
<…>
<code entry point>
<web-services-call>
<…>
```

Contract information relevant to a web services invocation can be maintained in DBMS schema designed to provide an element of data independence to the stored procedure language if we extend the language to include variable assignments from database queries. As a simple example, the above TIMEOUT parameter could be a query result, e.g., TIMEOUT (SELECT seconds FROM contracts WHERE ContractId=53372); . This type of extension would also facilitate the tracking of web service performance data such as the actual time it took to invoke and execute a web service since that data could be stored back into a DBMS relation.

Alerts requiring human intervention and/or automated exception handling procedures that fall outside the stored procedure are an additional need. The RETURN(exception) addition to a stored procedure requires that 'exception' be a coded value, used by the DBMS engine to dictate the appropriate resolution strategy. For example, an organization could have a code designation for all stored procedures invoking web services that distinguishes between an exception requiring human intervention or automated processing such as shifting to a back-up stored procedure containing a web service call to a secondary service provider.

Summary

This snapshot of research-in-progress demonstrates several of the necessary preconditions for extending the functionality of DBMS stored procedure languages to handle the management and monitoring of composite applications and their associated web services invocations. The approach described has drawn from DBMS literature and widely available DBMS platforms like Oracle, distributed systems language constructs, supply chain management research and nascent research surrounding the impending web services arena. The language constructs identified are anticipated to be a part of a denotational semantic leading to a novel DBMS stored procedure language designed for this context.

References

Edwards, J., "Web Services are Real," *Oracle Magazine*, March/April, 2002, 66-75.

Gartner Research, "How Web Services Mean Business," *Research Note*, Andrews, W., D. Plummer and D. Smith, May 9, 2001.

Goul, M. and J.C. Hershauer, "A Middleware Model of Web Services-Based Information systems as Specialized Supply Chains," *Working Paper*, September, 2001.

March, S., A. Hevner and S. Ram, "Research Commentary: An Agenda for Information Technology Research in Hetereogeneous and Distributed Environments," *Information Systems Research*, (11) 4, December 2000, 327-341.