Strategies For Data Integration

Sakai Enterprise Discussion Group

April 28, 2005 (draft)

Duffy Gillman <duffy@email.arizona.edu>
University of Arizona

The majority of enterprise integration discussions around Sakai center on the user-oriented services. These are services that handle information about the people using Sakai, their groupings and the things they are authorized to do within Sakai. These services have been identified as Agent/Group, Authorization, Permissions and SuperStructure. Initial development for these services has been directed toward a tight-coupling in order to achieve efficiency through shared database tables. Mark Norton points out that Authorization will scale much better if it can reliable join tables used by the other services.

Data Synchronization

Offline/Batch

In this scenario a periodic process queries the university's information system for data about courses, enrollment and HR concerns. Data from these queries are used to populate the Sakai database.

Wholesale Data Rebuild

This method would rebuild the entire Sakai database with every periodic synchronization attempt. The burden of data consistency is placed upon the synchronization service as it will require that ancillary data within Sakai's database be properly mapped to the course and user records after the records are rebuilt.

Partial Update

In this scheme only data that have changed since the last synchronization are considered. Updates, inserts and deletes[1] are performed on Sakai records based to match the changes from the external system. Data consistency is still an issue: the synchronization service needs to ensure that the proper ancillary data are created

1Experience at the University of Arizona cautions against deletion of records in an LMS based upon changes in the registrar database. Use cases here need to be carefully considered. Students may be unenrolled from a course by mistake, or only temporarily. When enrollment is reinstated it is desirable to be able to recover the student's work. Marking students as deleted/unenrolled may be more productive than actual removal of records.

for inserted records, and are deleted for deleted records.

Staging Database Tables

This method has been advocated by Charles Severance and other members of the core development team. Here a set of tables would exist that complement those used by the core Sakai service APIs. These tables would distill user, group and course records down to the essential elements required from an external datasource (eg. for a user: name, email, unique id). A Sakai service would periodically poll the staging tables for new data, and would create records for the new data in the core Sakai database. In this way all ancillary data related to a user or course (eg. a site id, a set of permissions) could be properly initialized. Direct injection into the Sakai database would possibly create objects within the Sakai installation that lack critical data.

Plug-Ins

A framework for service plug-ins could be developed that would periodically poll data from an external source. These plugins could be scheduled to retrieve data on a periodic basis in the same manner in which imports from a Staging Database might be scheduled.

Just-In-Time Plug-Ins

Under investigation by SEPP members from UCLA is a mechanism for more efficient polling of an external data source. This notion involves the synchronization of data from an external plug-in on a just-in-time (JIT) basis. Data might for a user or course worksite may not exist in Sakai until the first time a user attempts to login and view materials for a particular course. Thereafter access of the same data would require the decision whether or not the data has been polled from the external database within a certain allowable time frame. If the data has been recently polled it is used directly from the Sakai database. If the data is old it is synchronized again with the external data source.

API Replacement

Sakai services are developed to be replaceable by services that implement the appropriate API. Using this scheme the Sakai database is not consulted for operations on the replaced API. Instead an external service is consulted directly. API Replacement carries a heavy cost. First, the external service must be responsive enough to handle the demands Sakai has for data without degrading appreciably. Second, the very services under consideration have been coupled tightly in the Sakai 2.0 design. On a deep level this allows cross-table SQL joins to be performed between tables 'owned' by these various services. Unfortunately this may mean that services can not easily be replaced in a piecemeal fashion. Finally, it is unclear how the 'attendant data' for users and courses read from an external datasource will be managed.