

ARCo Multi Cluster Support

EXECUTIVE SUMMARY

In the past, users have voiced the request for having the ability to write the reporting data of more than one SGE cluster into a single ARCo database.

Arco multi cluster supports means that each DBWriter from an SGE cluster will write the reporting data into the same database. The reporting GUI and command line will support the selection of the specific reporting data from all managed SGE clusters. Since each cluster is considered as a separate entity where the data objects like job id's, project names, user names are not necessarily synchronized, there will be no cross-cluster reporting generation supported in the first release of ARCo multi cluster support.

The multi cluster solution can be achieved by keeping the existing table structure and installing identical tables for each cluster using multiple schemas (for MySQL multiple databases). The details of this implementation will be discussed further below.

SEPARATE SCHEMA APPROACH

The advantage of this approach is that it requires minimal changes to the current database definition and the DBWriter side. Small drawback is that Postgres user will be required to create additional schemas, one per each cluster. If users have already multiple clusters defined on separate database servers, they will be advised on the upgrade procedure. Most likely they will have to create the new users and schemas on a new dedicated database server and then using the Postgres dump file migrate the old data before running the upgrade. This implementation also presents a small culprit and that is the different notion of schema in each database (discussed below).

Oracle

In Oracle there is automatically one schema created per user. Since there is exactly 1 to 1 relationship between a user and a schema, these two terms are often used interchangeably.

In order for one user to access objects from a different schema, the user needs to be granted select on those objects and access them with the fully qualified name, i.e. <schema_name>.<table_name>, or there need to be synonyms created for those objects. User does not need to use the fully qualified name if he is accessing objects in its own schema.

Postgres

Unlike databases, schemas are not rigidly separated: a user may access objects in any of the schemas in the database he is connected to, if he has privileges to do so. Schemas are analogous to directories at the operating system level, except that schemas cannot be nested.

In Postgres to create schema this command needs to be issued:

```
CREATE SCHEMA myschema;
```

Creating the schema in Postgres should follow the same pattern as in Oracle `schema_name = user_name`. The owner of the schemas must be the `clusterid_arco_write` user.

To create schema owned by someone else use:

```
CREATE SCHEMA schemaname AUTHORIZATION username;
```

By default, users cannot access any objects in schemas they do not own. To allow that, the owner of the schema needs to grant the `USAGE` privilege on the schema. To allow users to make use of the objects in the schema, additional privileges may need to be granted, as appropriate for the object.

```
GRANT USAGE ON SCHEMA schemaname TO username;
```

Usage has to be granted to the `clusterid_arco_read` user.

In Postgres synonyms don't exist. Qualified names are tedious to write, and it's often best not to wire a particular schema name into applications anyway. Therefore, tables are often referred to by unqualified names, which consist of just the table name. The system determines which table is meant by following a search path, which is a list of schemas to look in. The first matching table in the search path is taken to be the one wanted. If there is no match in the search path, an error is reported, even if matching table names exist in other schemas in the database.

In default setup the search path is: `"$user",public`

To put schema in the path use:

```
SET search_path TO schemaname,public;
```

MySQL

In MySQL schema is equivalent to database. Command `CREATE SCHEMA` is the same as `CREATE DATABASE`.

Changes Required

Database Structure

In `dbdefiniton.xml` for Postgres:

- change table definition so it uses the following syntax:
`CREATE TABLE myschema.mytable (..);`
- `GRANT USAGE ON clusterid_arco_write` to `clusterid_arco_read`
- change views definitions to refer to the new schema tables

No changes for Oracle and MySQL

DBWriter Java code

No changes

Install and Update Scripts

In inst_dbwriter / updatedb.sh for Postgres only:

- changes to use the supplied schema name to create object in it.

In inst_reporting:

- changes to prompt user to enter all the arco_read users and pwd for all clusters
- for Postgres and MySQL also prompt for the name of the schemas, databases respectively. Certain naming patterns should be followed, but in Postgres there could still be the public schema.

Reporting UI

The reporting UI will introduce a new component - a drop down menu which will be populated with the supplied cluster names. New page with a form to supply additional schema, arco_read and passwords will be provided. It is possible that this will be the only way to define additional clusters and not changes in inst_reporting will be required. Cluster names and user names can be retrieved from schema name if users follow our proposed naming conventions.

Changes:

- creating Bean, View and Model classes or changing some existing one possibly IndexViewBean to incorporate the new component
- writing the handling functionality, life-cycle processing
- com.sun.grid.arco.model needs to be changed to support the new configuration
- jsp pages changes
- updating UI help
- Changing classes dealing with Connection so they can include multiple connection pools, one for each cluster/schema
- Changing code that connects to database to choose connection based on the cluster menu selection

Arcorun

- add additional switch option to specify the schema/database on which to run the query
- validation to check if the schema/db exists
- maybe display a menu for user to select a correct schema when the query is run from command line.

Documentation

- Comprehensive documentation changes

Miscellaneous

It might also be beneficial to allow users to specify a different tablespace for each cluster database schema. This could further improve performance, as tablespaces are related to the physical data storage. The tablespaces need to be created by the DBA. Tablespaces are supported in Oracle and Postgres.

In MySQL, unlike Oracle and Postgres, you can't allocate certain named tablespaces and assign tables into them. You either have a shared tablespace (which can be composed of multiple data *files* spread across multiple disks...) or you have multiple tablespaces, one per

each table.

CONCLUSION

Separating data into different schemas allows to organize database objects into logical groups to make them more manageable. As specific challenges arise from trying to combine data from multiple clusters to form a logical result, it is preferable to initially only provide a single viewing access point to multiple cluster data (single Reporting installation), and only after extensive user requirements evaluation, provide enhancements.