Hibernate ID's and versioning

How to

To upgrade, get a version of grouper newer than 1.4, and build 7/30/8. Then start grouper or do an: ant schemaexport. This will generate a script. You should review it before running it. It will create hibernate_version_number cols, and init them in all tables. Also it will create old_id and old_uuid cols, and copy data from id and uuid there. Then it will set the id to col to be the value of the uuid. Then it will drop the uuid col. When you are satisfied that everything is ok, you should set the grouper.properties ddlutils.dropBackupUuidCols to true, and the next time you schemaexport it will drop the old_id and old_uuid cols. Here is a sample oracle upgrade script (might be out of date)

Troubleshooting

You need to pay attention to each error, and resume the script after fixing problems. You should not have grouper or ant run this script for you. Here is sample oracle troubleshooting:

```
ORA-01408: such column list already indexed
Details: CREATE UNIQUE INDEX type_name_idx ON GROUPER_TYPES (NAME)
>>> resolution, drop unique constraint on name, drop existing index, then run again

Details: ALTER TABLE GROUPER_MEMBERSHIPS DROP COLUMN MEMBERSHIP_UUID
ORA-12991: column is referenced in a multi-column constraint
>>> resolution, drop multi-column constraint, then run again

Details: ALTER TABLE GROUPER_MEMBERSHIPS ADD CONSTRAINT fk_memberships_list_name_type FOREIGN KEY
  (LIST_NAME, LIST_TYPE) REFERENCES GROUPER_FIELDS (NAME, TYPE)
ORA-02270: no matching unique or primary key for this column-list
ORA-02429: cannot drop index used for enforcement of unique/primary key
>>> resolution, add a unique constraint on name and type in grouper fields
>>> ALTER TABLE AUTHZADM.GROUPER_FIELDS ADD  CONSTRAINT fields_name_type_unq
>>> UNIQUE (NAME, TYPE) ENABLE VALIDATE

Details: DROP INDEX SUBJECTATTRIBUTE_ID_NAME_IDX
ORA-02429: cannot drop index used for enforcement of unique/primary key
>>> this worked when i tried again

Details: ALTER TABLE SUBJECTATTRIBUTE ADD CONSTRAINT
fk_subjectattr_subjectid FOREIGN KEY (SUBJECTID) REFERENCES SUBJECT (SUBJECTID)
ORA-02298: cannot validate (AUTHZADM.FK_SUBJECTATTR_SUBJECTID)
- parent keys not found
>>> this worked when I tried again
```

Introduction

In the grouper dev call on 3/5/8 we discussed removing the non-uuid id columns of some tables. e.g. grouper_groups has a column named "id", and a column named "uuid". Both are identifiers for the row. The "id" col is not used in any foreign keys or in any api's and is just used for hibernate as a key, and the primary key of the table. The "uuid" is used for unenforced foreign keys. The following tables are candidates for removing the id cols: grouper_composites, grouper_fields, grouper_groups, grouper_members, grouper_sessions, grouper_types, grouper_stems. It seems like grouper_memberships is not a candidate since there can be multiple rows with the same membership_uuid...

Why remove?

It is confusing to have two id cols for a table. Also, it takes space to keep the index for this other row. I think the current design is not the original design, and we can clean it up.

Challenges

We cannot simply remove the id col, and keep the uiud column. The hibernate id col is used by hibernate to know if a sql change is an update or insert. If there is a key there, it is an update, if not, it is an insert. And the key gets created right when the record gets inserted. However, Grouper's logic design leverages the unenforced foreign keys based on uuid'd which are created before the row is inserted to setup a batch of sql's (complete with unenforced foreign key references) before a single insert occurs.

Solution

We should use the uuid col for the primary key, and for hibernate's identity. We should ditch the id col. To get around the fact that hibernate needs to know if something is an insert or update, we should add a new col which is a number which will be the hibernate version of the row. It starts at 0 and increments with every update.

http://www.hibernate.org/hib_docs/reference/en/html/mapping.html#mapping-declaration-version

However, this feature would throw exceptions if two grouper processes stepped on each other's toes by throwing "StaleState" exceptions. e.g. if the version that a process updates is not the same as when it was selected, that means some other process updated it, and a StaleState exception is thrown, which indicates that the process should refresh the data and make the edits again. This will not work with Grouper for two reasons: we cache data, and we make more updates than are absolutely necessary. e.g. we update the Stem object when a child group is created. So if two grouper processes created a group under the same stem around the same period of time, it would throw exception. Eventually we can think about using real hibernate versioning, but not right now. To use it, we should clean up our updates to only update what has changed (dirty checking), and have more intelligent cache refreshes. Until then we can use versioning that does not throw exceptions if stale. So the version column will not be exactly accurate, but it will be a good approximation. Also, we should remove support for hibernate2 since we are requiring hib3, and these changes make hib2 not compile...

I have made these changes locally, and they are not all that complex or risky. All unit tests pass.

Yes we still have the same number of columns, but now the column is not an id, it is just an int. It also gives good information (how many times updated), does not have an index on it, and greases the skids if we ever go to strict hibernate versioning.

The implementation is done with a normal hibernate column mapping, an interface on the DAOs which are versioned (Hib3HibernateVersioned), and a hibernate interceptor which sets / increments this val for us automatically. Also, before where the hibernate id was passed back from the DAO to the DTO since hibernate originated the data, now the version will. Not a lot of code changes.

Migration path

It is not that hard to migrate the DB of existing grouper deployments. It is a bit easier if the db col is still the "id" col, but we just move the uuid data in there. Then remove the uuid col. We can either see which databases need this and make a simple script for it. Or we could try to use DdlUtils to make an ant script that does this for any db. Before making the changes, a backup should be made, and maybe an export.