SQL Repository Implementation

SQL repository implements both repository and audit API. It facilitates Hibernate 5 as persistence framework, and for connection pooling HikariCP (previou sly c3p0).

Audit records are stored synchronously and implementation currently shares connection pool with main repository. There are few planned improvements for audit, see MID-4745.

Configuration

Repository component configuration is placed in main `config.xml` file in `midpoint.home` folder. Basic configuration is quite simple. It will use embedded H2 database which will be placed in midpoint.home folder.

```
<configuration>
  <midpoint>
  <repository>
     <repositoryServiceFactoryClass>com.evolveum.midpoint.repo.sql.SqlRepositoryFactory<
/repositoryServiceFactoryClass>
     <baseDir>${midpoint.home}</baseDir>
  </repository>
  </midpoint>
</configuration>
```

For more complex configurations using different DB vendors see Repository Configuration.

DB tables, structures

Our data objects defined in common-3.xsd schema which are managed by SQL repository are quite complex. They are stored in `m_object` table in `fullO bject` column as blobs. Parts of these objects are stored in related tables for better searching.

General

PrismObject represented as XML objects contain one natural primary key - oid attribute. Some objects also contain PrismContainers which use id as container identifier (assignments, inducements, triggers, etc.). Identifiers for objects and containers are generated in midPoint. Object references introduced in XSD schema are transformed to entities with weak reference which means reference doesn't use foreign key to achieve full consistency across midPoint repository. This will simplify work with objects during import or delete as it's not necessary to correctly sort objects before making any changes. To further improve hibernate performance we've introduced `EntityState` interface. This interface is used within most entities. Developer knows in most cases whether entity is transient or detached, therefore hibernate doesn't need to do any checks (we'll get rid of many unnecessary select queries).

Storing xsd:any attributes

xsd:any container can be found in:

- ObjectType - extension (and subclasses)
- ShadowType - attributes
- AssignmentType - extension

Schema for extension containers is static for first and third case. ShadowType attributes use dynamic schema which depends on defined resources. Extension container can contain elements of different types. SQL repository supports `String, Long, Date, Clob, ObjectReferenceType, PolyStringType`. All elements are aggregated to this types (tables).

- Long, Integer, Short -> **Long**
- Date (Gregorian calendar) -> **Date**
- String, Double, Float -> **String**
- ObjectReferenceType and PolyString types are handled via custom tables
- Everything else goes to **Clob**

Long, Date and String values are indexed, clob is not indexed. Real types for aggregated types are saved as QName values. Aggregation translation for xsd:any values is provided by RAnyConverter. Repository uses dictionary implementation to store/load different properties, see ExtItemDictionary. New items are added to dictionary in transaction other than current add/modify operation is in. In such case parent transaction is restarted.

Audit structures

Audit tables uses `IDENTITY` columns (autoincrement).

Transactions, locking failover

SQL repository uses simple transactions which last during single operation (e.g. add/modify/delete/search). When transaction fails because of locking (table lock timeout, record timeout), SQL repository tries to repeat operation for number SqlBaseService **LOCKING_MAX_ATTEMPTS** times. As this implementation supports databases from multiple vendors, transaction and locking configuration is different for each vendor. Default configurations are defined in `SqlRepositoryConfiguration`.

Querying

SQL repository contains `com.evolveum.midpoint.repo.sql.query2.QueryInterpreter2` which can interpret queries defined by query-3 schema. `QueryInterpreter` uses `QueryRegistry` to translate queried value name to real hibernate entity property name. `QueryRegistry` loads entity and attribute definitions during initialisation. Currently supported query filters are:

- equal
- substring (with anchor start/end)
- and/or/not
- inOid

Tests

Most tests are located in module `repo-sql-impl-test`.

Overall code description

todo

- SqlRepositoryBeanConfig - spring wiring and initialization
- SqlRepositoryConfiguration - sql repository configuration, handles config.xml and such
- DataSourceFactory - connection pool initialisation and configuration