Liquibase

Liquibase is a database-independent library for tracking, managing and applying database changes (in particular, schema changes).

Appian features using Liquibase: [[RDBMS Integration Design]], [[Tempo Engine Design]]

Versions used in Appian

- Appian 21.1 -- Liquibase 3.8.9
- Appian 23.1 – Liquibase 4.17.1

When you need to find information

- Overview and getting started:
  - Overview
  - Quickstart
  - Best Practices
  - Liquibase Recipes - Appian Best Practices
- Docs:
  - Docs Home
- Bug tracker:
  - Bug Tracker
- Forums:
  - Liquibase Forum
- Blog:
  - Liquibase Blog
- Comparison with other database migration tools:
  - Stackoverflow discussion
  - Flyway
  - DbMaintain

Databases supported by Liquibase

Supported Databases

Common questions

Logging

https://forum.liquibase.org/t/how-do-i-use-loglevel/3729

Extending Liquibase

Liquibase Extensions

Issues

Waiting for changelog lock

Liquibase uses a row in a table as a lock for the database migration. Before running the migration, Liquibase updates the one row in DATABASECHANGELOGLOCK to have the LOCKED column set to true. Once the migration is complete, Liquibase updates the value back to false. This handles the case when multiple JVMs run Liquibase simultaneously, which can happen in a multiple app servers environment when several servers are booted at the same time.

If you kill the JVM while Liquibase is in the middle of running the migration, the LOCKED row will remain set to true, and all subsequent attempts to run Liquibase will be stuck indefinitely "Waiting for changelog lock". To fix this, login to the database and manually update the row to have the LOCKED column set to false.

Of course, if you're just doing development, you can also just drop the entire database or schema.

Auto-increment vs sequences
Forum Discussion

Appending the MySQL engine when creating tables
Modifying SQL Documentation

columnDataType attribute on renameColumn tag doesn't support java.sql.Types
CORE-942

Changelog file path stored in DATABASECHANGELOG
Liquibase stores the file path of each changelog file and uses it as part of the composite primary key identifying each changeset. The path is therefore also used as part of the MD5 hash of each changeset. This means that when a changeset file is moved or renamed, the primary key of the changesets inside it will change and Liquibase will think that it needs to apply them again. To work around this problem, Liquibase provides a "logicalFilePath" attribute on <databaseChangeLog> tag: Liquibase changelog

References:
- Liquibase Migration Fix Ticket
- Moving or Renaming a Changelog File
- Forum Discussion

Dropping the schema is very slow on Oracle
- Forum discussion about dropAll
- indexExists Slowness Fix

Creating custom changes with Java
- Liquibase Custom Change

Performance
As of 6.5.1, these are the rough timings:
- Running migration on an empty database.
    - Hibernate SessionFactory init: 3 sec
    - Liquibase: 2 sec
    - Hibernate validation: 200 ms
- Running migration on a fully valid existing schema (Liquibase still has to be executed to perform the check for whether all migration changesets have been applied to the schema).
    - Hibernate SessionFactory init: 3 sec
    - Liquibase: 1 sec
    - Hibernate validation: 200 ms

Testing

Migration testing should be performed for all liquibase changes to discover potential issues when migrating to supported database platforms. AN-69760 is an example of a problem that could have been discovered by migration testing.