:years: 2015-2022

Persistence

TBD

Database support

Fineract supports multiple databases:

- MySQL compatible databases (e.g. MariaDB)
- PostgreSQL

The platform differentiates between these database types in certain cases when there's a need to use some database specific tooling. To do so, the platform examines the JDBC driver used for running the platform and tries to determine which database is being used.

The currently supported JDBC driver and corresponding mappings can be found below.

JDBC driver class name	Resolved database type
`org.mariadb.jdbc.Driver`	MySQL
`com.mysql.jdbc.Driver`	MySQL
`org.postgresql.Driver`	PostgreSQL

The actual code can be found in the `DatabaseTypeResolver` class.

Data-access layer

The data-access layer of Fineract is implemented by using JPA (Java Persistence API) with the EclipseLink provider. Despite the fact that JPA is used quite extensively in the system, there are cases where the performance is a key element for an operation therefore you can easily find native SQLs as well.

The data-access layer of Fineract is compatible with different databases. Since a lot of the native queries are using specific database functions, a wrapper class - `DatabaseSpecificSQLGenerator` - has been introduced to handle these database specifics. Whenever there's a need to rely on new database level functions, make sure to extend this class and implement the specific functions provided by the database.

Fineract has been developed for 10+ years by the community and unfortunately there are places where entity relationships are configured with `EAGER` fetching strategy. This must not confuse anybody. The long-term goal is to use the `LAZY` fetching strategy for every single relationship. If you're about to introduce a new one, make sure to use `LAZY` as a fetching strategy, otherwise your PR will be rejected.

Database schema migration

As for every system, the database structure will and need to evolve over time. Fineract is no different. Originally for Fineract, Flyway was used until Fineract 1.6.x.

After 1.6.x, PostgreSQL support was added to the platform hence there was a need to make the data-access layer and the schema migration as database independent as possible. Becuase of that, from Fineract 1.7.0, Flyway is not used anymore but Liquibase is.

Some of the changesets in the Liquibase changelogs have database specifics into it but they only run for the relevant databases. This is controller by Liquibase contexts.

The currently available Liquibase contexts are:

- `mysql` - only set when the database is a MySQL compatible database (e.g. MariaDB)
- `postgresql` - only set when the database is a PostgreSQL database
- configured Spring active profiles
- `tenant_store_db` - only set when the database migration runs the Tenant Store upgrade
- `tenant_db` - only set when the database migration runs the Tenant upgrade
- `initial_switch` - this is a technical context and should **NOT** be used

The switch from Flyway (1.6.x) to Liquibase (1.7.x) was planned to be as smooth as possible so there's no need for manual work hence the behavior is described as following:

- If the database is empty, Liquibase will create the database schema from scratch
- If the database contains the latest Fineract 1.6.x database structure which was previously migrated with Flyway. Liquibase will seamlessly upgrade it to the latest version. Note: the Flyway related 2 database tables are left as they are and are not deleted.
- If the database contains an earlier version of the database structure than Fineract 1.6.x. Liquibase will **NOT** do anything and **will fail the application during startup**. The proper approach in this case is to first upgrade your application version to the latest Fineract 1.6.x so that the latest Flyway changes are executed and then upgrade to the newer Fineract version where Liquibase will seamlessly take over the database upgrades.

Troubleshooting

1. During upgrade from Fineract 1.5.0 to 1.6.0, Liquibase fails

After dropping the flyway migrations table (schema_version), Liquibase runs it's own migrations which fails (in recreating tables which already exisit) because we are aiming to re-use DB with existing data from Fineract 1.5.0.

Solution: The latest release version (1.6.0) doesn't have Liquibase at all, it still runs Flyway migrations. Only the develop branch (later to be 1.7.0) got sqitched to Liquibase. Do not pull the develop before upgrading your instance.

Make sure first you upgrade your instance (aka database schema with Fineract 1.6.0). Then upgrade with the current develop branch. Check if some migration scripts did not run which led to some operations failing due to slight differences in schema. Try with running the missing migrations manually.

Note: develop is considered unstable until released.

1. Upgrading database from MySQL 5.7 as advised to Maria DB 10.6, fails. If we use data from version 18.03.01 it fails to migrate the data. If we use databases running on 1.5.0 release it completes the startup but the system login fails.

Solution: A database upgrade is separate thing to take care of.

1. We are getting `ScehmaUpgradeNeededException: Make sure to upgrade to Fineract 1.6 first and then to a newer version` error while upgrading to `tag 1.6`.

1.6 version shouldn't include Liquibase. It will only be released after 1.6. Make sure Liquibase is dropping `schema_version` table, as there is no Flyway it is not required. Drop Flyway and use Liquibase for both migrations and database independence. In case, if you still get errors, you can use git SHA `746c589a6e809b33d68c0596930fcaa7338d5270` and Flyway migration will be done to the latest.

```
TENANT_LATEST_FLYWAY_VERSION = 392;
TENANT_LATEST_FLYWAY_SCRIPT_NAME =
"V392__interest_recovery_conf_for_rescedule.sql";
TENANT_LATEST_FLYWAY_SCRIPT_CHECKSUM = 1102395052;
```