Lesson 29

25.03.2024

```
public class Test1 {
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 0, j = 0; i < 6 & j < 5; ++i, j = i + 1) {
            sum = +i;
            System.out.println(sum);
        }
    }
}
```

```
public class Test2 {
    public static void main(String[] args) {
        int arr[] = {11, 22, 33};
        for (int i = 0; i < arr.length; i++)
            System.out.println(arr[i] + " ");

        int arrr[] = new int[3];
        arrr[] = {11, 22, 33};
        for (int i = 0; i < arrr.length; i++)
            System.out.println(arrr[i] + " ");
    }
}
```

```
public class Test3 {
    public static void main(String[] args) {
        int[][] arr1 = new int[2][3];
        int[][] arr2 = new int[2][];
        int[][] arr3 = new int[][];
        int[][] arr4 = new int[][3];
    }
}
```

```
public class Test4 {
    public static void main(String[] args) {
        for (int i = 0; i < 1; System.out.println("Java")) {
            System.out.println("Scala");
        }
    }
}
```

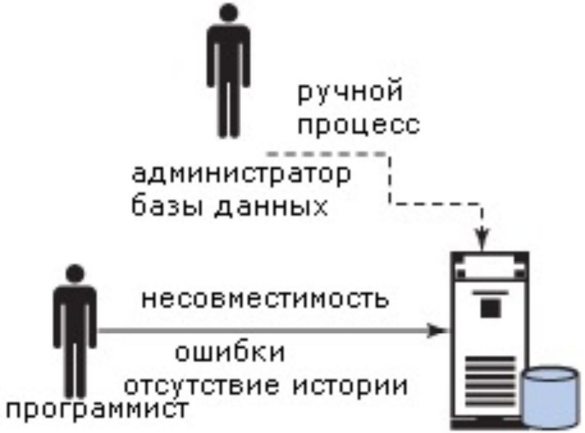

5 java questions

1. Відмінності String/StringBuilder/StringBuffer
2. Interface vs Abstract Class
3. override vs overload
4. Регіони пам'яті в JVM
5. java.util.collection.

Міграція баз даних

Розробка програмного забезпечення призводить до таких проблем:

- ручне внесення змін в БД;
- різні версії БД у різних учасників команди розробників;
- непослідовні підходи до внесення змін (в базу даних або дані);
- неефективні механізми ручного управління змінами при переходах між версіями баз даних.



Liquibase

Flyway

Управління змінами в базі даних за допомогою LiquiBase (www.liquibase.rg)

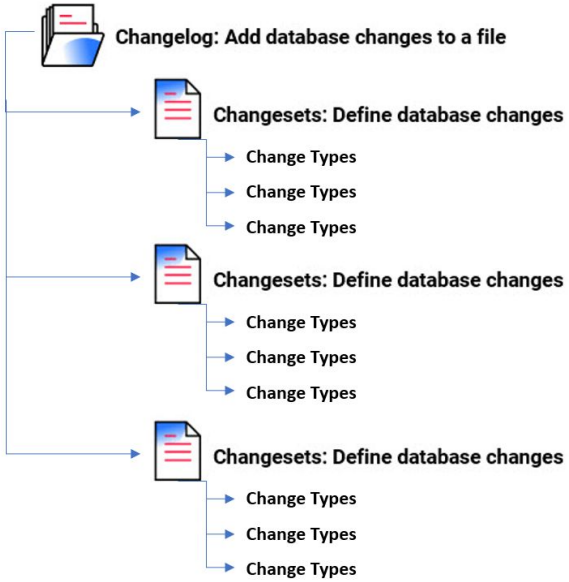
Liquibase — це незалежна від бази даних бібліотека для відстеження, керування та застосування змін схеми бази даних. Для того, щоб внести зміни в БД, створюється файл міграції (*changeset*), який включається в головний файл (*changeLg*), який контролює версії та керує всіма змінами. У якості опису структури та змін бази даних використовуються формати XML, YAML, JSN і SQL.

Основні функціональні можливості:

- Оновлення бази даних до поточної версії
- Відкат останніх змін у базі даних / на визначену дату / час / тега
- SQL для оновлення бази даних і відкатів можна зберегти для ручного перегляду
- «Контексти» для включення / виключення набору змін для виконання
- Отчет о различиях баз данных
- Генерація змін в базі даних
- Можливість створення журналу змін для створення існуючої бази даних
- Створення документації по змінам бази даних
- Перевірка СУБД, перевірка користувача та попередні умови перевірки SQL
- Можливість розбивати журнал змін на кілька файлів для посилення управління
- Виконується через командну строку, Apache Ant, Apache Maven, контейнер сервлетів або Spring Framewrk

Щоб розпочати роботу з LiquiBase, потрібно виконати чотири кроки:

- Створити файл із журналом змін у базі даних (**change log**).
- визначити набір змін (**change set**) у цьому файлі.
- Призначити набір змін до бази даних через командну строку або сценарій збірки.
- Перевірити зміни в базі даних.



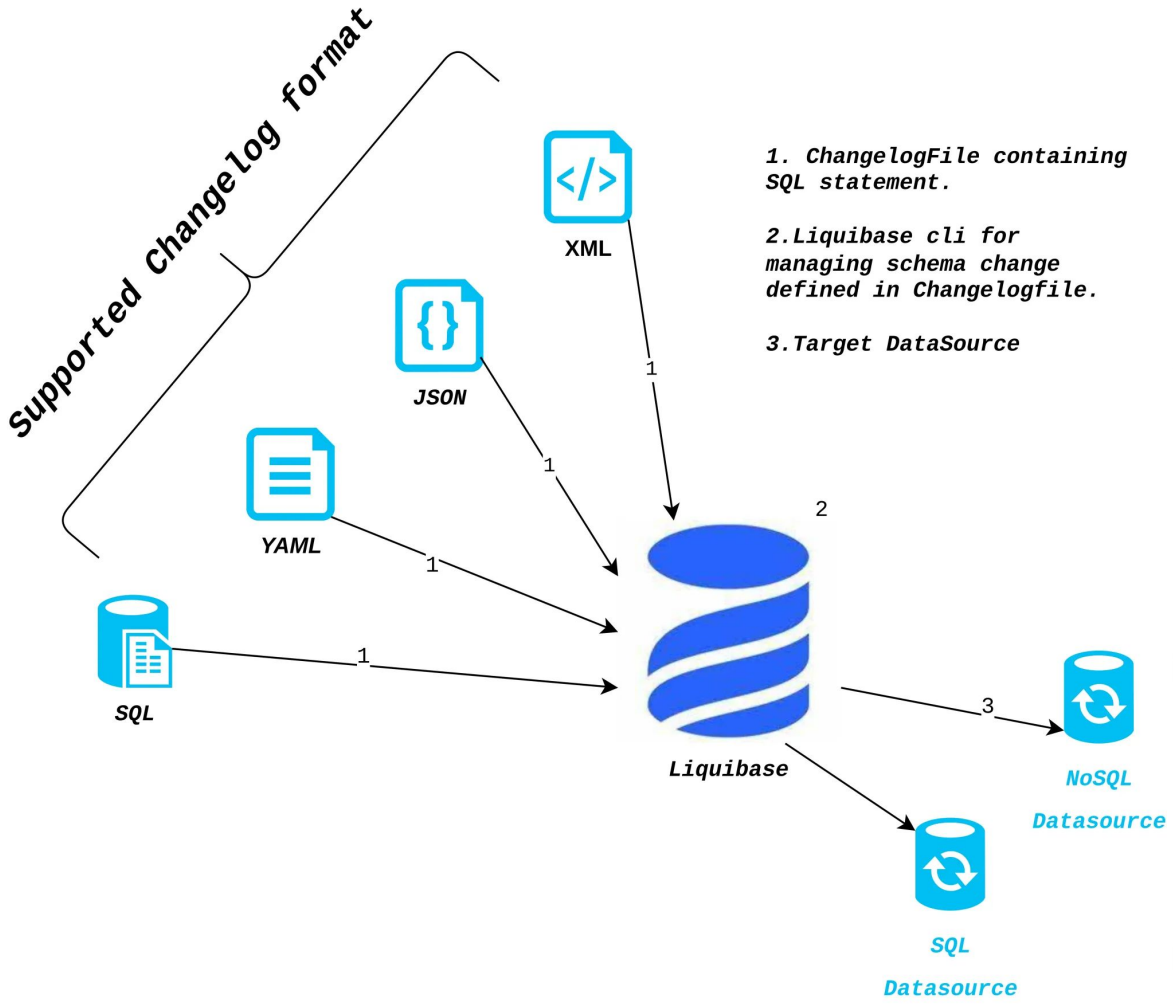
Changesets: Define database changes

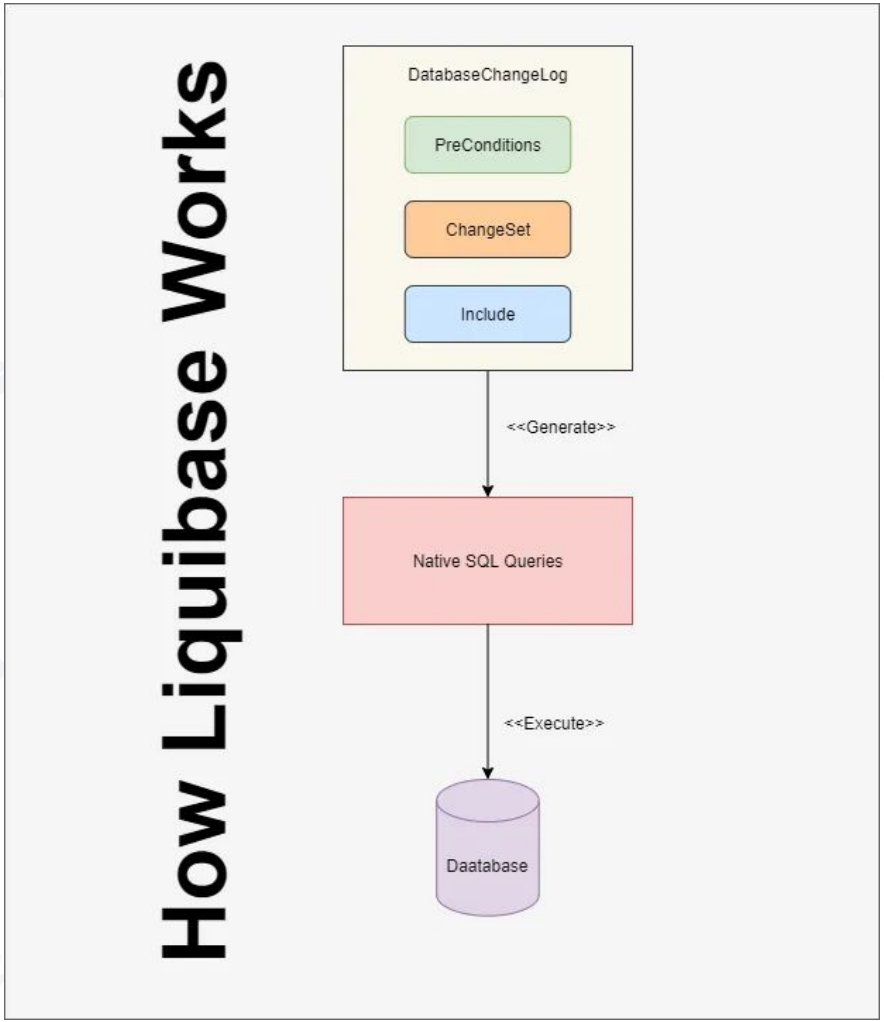
Changelog: Add database changes to a file

Update: Run the command to deploy database changes

DATABASECHANGELOG and DATABASECHANGELOGLOCK:
Track and version database changes

Manage your database changes with other Liquibase commands


Supported Changelog format
XML
JSON
YAML
SQL
1
1
1
1
2
3
Liquibase
NoSQL
Datasource
SQL
Datasource
1. ChangelogFile containing SQL statement.
2.Liquibase cli for managing schema change defined in Changelogfile.
3.Target DataSource


How Liquibase Works
DatabaseChangeLog
PreConditions
ChangeSet
Include
<<Generate>>
Native SQL Queries
<<Execute>>
Daatabase