Spring Batch
Analysis for Alternatives for DB Persistence

Table of Contents

Introduction

Spring Batch is a lightweight batch processing framework. It provides functionality for processing large datasets. The software also provides functionality such as processing statistics, job restart, skip and resource management.

The default architecture requires Spring Batch to maintain it's state in a Database. By default, it supports multiple databases such as Oracle, Postgres, Sybase and H2 (in-memory). It even provides a non-DB, in-memory solution using MapJobRepositoryFactoryBean.

The persistence layer is used every time Job is started, restarted, before execution of each Step, commit interval of a Chunk etc. Further, the data in these tables can be used for monitoring and metrics of the batch jobs.

In order to support the persistence requirements of Spring Batch, we are currently using Oracle DB. As new components are added, we are running into environments where a DB is not available. An analysis was conducted to look at alternatives for current approach.

Analysis

Based on analysis of the Spring Batch API the following are the key findings:

1. The Spring Batch API uses RDBMS to persist Job State and history.
2. The schemas for the supported RDBMS can be found in the spring-batch-core jar under the package org.springframework.batch.core.
3. There is no support for NonSQL persistence support such as MongoDB.
4. The API provides a non-persistent, in-memory implementation using MapJobRepositoryFactoryBean to create a SimpleJobRepository. However, per the documentation it is really intended for testing and prototyping. It is not suited for multi-threaded jobs with splits
5. The API provides Listener interfaces which can be used to log Life Cycle Events.
6. The API provides JobExplorer, JobOperator objects which can be used to retrieve Job

history from the persistence store.

7. The embedded database Spring APIs can be used to set up in-memory Persistence store. However, for H2, the URL is hard-coded to use the in-memory implementation, without any option to write out the database to file.

Solution:

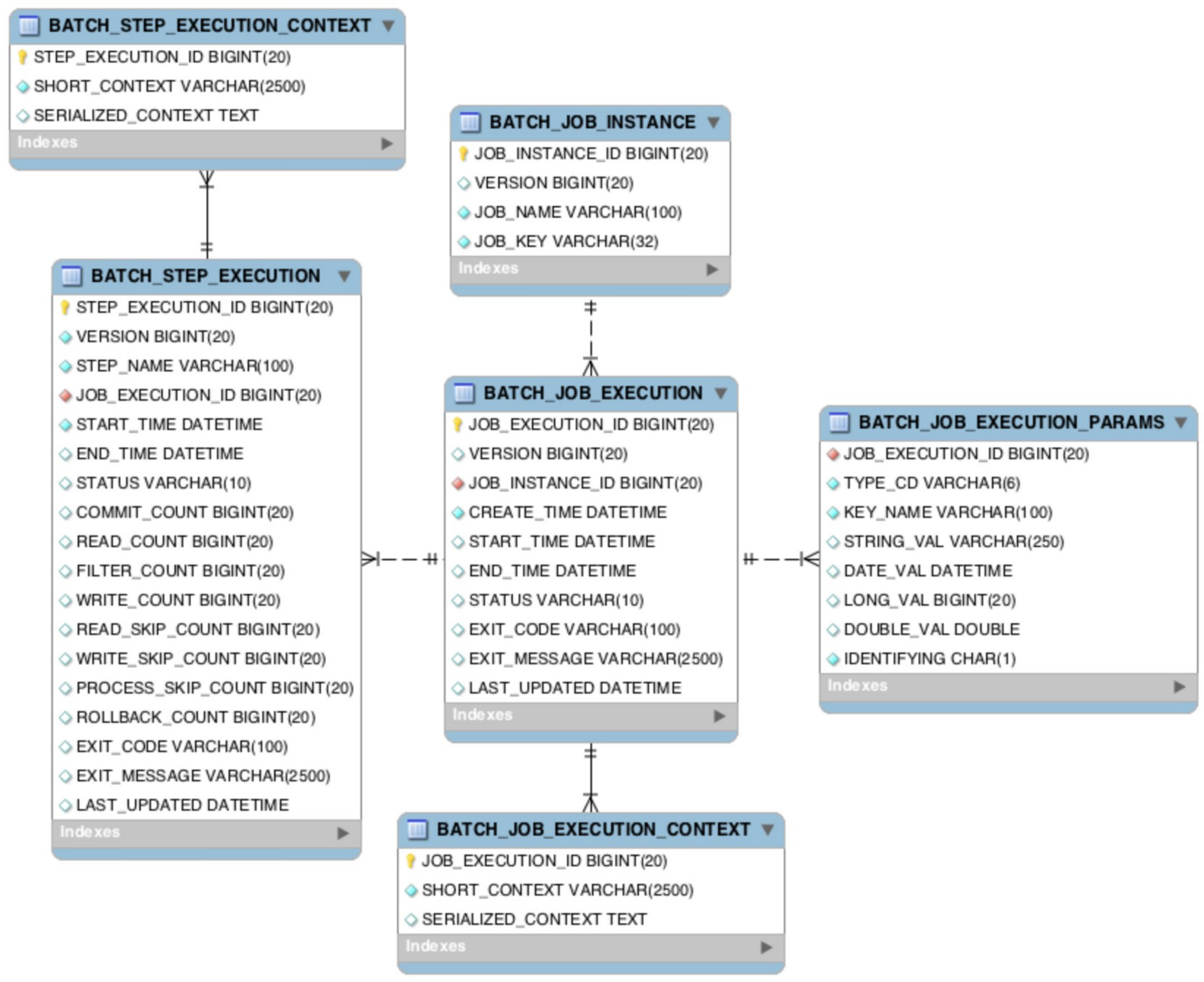
A H2 in-memory database with an external file seems is the recommended solution for the following reasons:

1. Avoids creation of DB in the restricted environments where Database creation/requests are taking a long time.
2. In-memory H2 with externalized DB file provides access to the files for use via JobExplorer, JobOperator APIs. The DB file can also be loaded to a docker H2 volume, for SQL based analysis.

Appendices:

Database Schema

The following diagram shows the ERP data model used by the Spring Batch framework.



API SQL

The following SQLs were collected from the logs for a simple batch Job with a single step. The Job read a file and used two writers to Write the files back to file system. There were about 40 SQL calls made for the execution of this Job. Further there were some SQLs that were executed prior to Job starting and after Job ending.

H2 database running in server mode (external in it's own jvm) was used as the data store.

This example demonstrates the usage of the DB by the batch job.

```
SELECT JOB_INSTANCE_ID, JOB_NAME from BATCH_JOB_INSTANCE where JOB_NAME = 'importProducts' and JOB_KEY
= 'be91bcf190233b3e100a6f885be87717'

SELECT JOB_INSTANCE_ID, JOB_NAME from BATCH_JOB_INSTANCE where JOB_NAME = 'importProducts' and JOB_KEY
= 'be91bcf190233b3e100a6f885be87717'

SELECT JOB_INSTANCE_ID, JOB_NAME from BATCH_JOB_INSTANCE where JOB_NAME = 'importProducts' and JOB_KEY
= 'be91bcf190233b3e100a6f885be87717'

select BATCH_JOB_SEQ.nextval from dual

INSERT into BATCH_JOB_INSTANCE(JOB_INSTANCE_ID, JOB_NAME, JOB_KEY, VERSION) values (17,
'importProducts', 'be91bcf190233b3e100a6f885be87717', 0)

INSERT into BATCH_JOB_PARAMS(JOB_INSTANCE_ID, KEY_NAME, TYPE_CD, STRING_VAL, DATE_VAL, LONG_VAL,
DOUBLE_VAL) values (17, 'targetDirectory', 'STRING','/target/test-classes/data/', '12/31/1969
19:00:00.000', 0, 0.0)

INSERT into BATCH_JOB_PARAMS(JOB_INSTANCE_ID, KEY_NAME, TYPE_CD, STRING_VAL, DATE_VAL, LONG_VAL,
DOUBLE_VAL) values (17, 'inputFile', 'STRING', 'products.txt', '12/31/1969 19:00:00.000', 0, 0.0)

INSERT into BATCH_JOB_PARAMS(JOB_INSTANCE_ID, KEY_NAME, TYPE_CD, STRING_VAL, DATE_VAL, LONG_VAL,
DOUBLE_VAL) values (17, 'outputFile', 'STRING', 'products-out.txt', '12/31/1969 19:00:00.000', 0, 0.0)

INSERT into BATCH_JOB_PARAMS(JOB_INSTANCE_ID, KEY_NAME, TYPE_CD, STRING_VAL, DATE_VAL, LONG_VAL,
DOUBLE_VAL) values (17, 'timestamp', 'LONG', '', '12/31/1969 19:00:00.000', 1579031711440, 0.0)

select BATCH_JOB_EXECUTION_SEQ.nextval from dual

INSERT into BATCH_JOB_EXECUTION(JOB_EXECUTION_ID, JOB_INSTANCE_ID, START_TIME, END_TIME, STATUS,
EXIT_CODE, EXIT_MESSAGE, VERSION, CREATE_TIME, LAST_UPDATED) values (17, 17, NULL, NULL, 'STARTING',
'UNKNOWN', '', 0, '01/14/2020 14:55:11.655', '01/14/2020 14:55:11.655')

INSERT INTO BATCH_JOB_EXECUTION_CONTEXT (SHORT_CONTEXT, SERIALIZED_CONTEXT, JOB_EXECUTION_ID)
VALUES('{"map":""}', NULL, 17)

SELECT COUNT(*) FROM BATCH_JOB_EXECUTION WHERE JOB_EXECUTION_ID = 17

UPDATE BATCH_JOB_EXECUTION set START_TIME = '01/14/2020 14:55:11.691', END_TIME = NULL, STATUS =
'STARTED', EXIT_CODE = 'UNKNOWN', EXIT_MESSAGE = '', VERSION = 1, CREATE_TIME = '01/14/2020
14:55:11.655', LAST_UPDATED = '01/14/2020 14:55:11.692' where JOB_EXECUTION_ID = 17 and VERSION = 0
```

```
1499  INFO  e.b.s.EIMBatchMonitoringListener - Started job JobExecution: id=17, version=1,
startTime=Tue Jan 14 14:55:11 EST 2020, endTime=null, lastUpdated=Tue Jan 14 14:55:11 EST 2020,
status=STARTED, exitStatus=exitCode=UNKNOWN;exitDescription=, job=[JobInstance: id=17, version=0,
JobParameters=[{targetDirectory=./target/test-classes/data/, inputFile=products.txt,
outputFile=products-out.txt, timestamp=1579031711440}], Job=[importProducts]]
```

```
SELECT JOB_EXECUTION_ID, START_TIME, END_TIME, STATUS, EXIT_CODE, EXIT_MESSAGE, CREATE_TIME,
LAST_UPDATED, VERSION from BATCH_JOB_EXECUTION where JOB_INSTANCE_ID = 17 order by JOB_EXECUTION_ID
desc
```

```
SELECT STEP_EXECUTION_ID, STEP_NAME, START_TIME, END_TIME, STATUS, COMMIT_COUNT, READ_COUNT,
FILTER_COUNT, WRITE_COUNT, EXIT_CODE, EXIT_MESSAGE, READ_SKIP_COUNT, WRITE_SKIP_COUNT,
PROCESS_SKIP_COUNT, ROLLBACK_COUNT, LAST_UPDATED, VERSION from BATCH_STEP_EXECUTION where
JOB_EXECUTION_ID = 17 order by STEP_EXECUTION_ID

SELECT JOB_EXECUTION_ID, START_TIME, END_TIME, STATUS, EXIT_CODE, EXIT_MESSAGE, CREATE_TIME,
LAST_UPDATED, VERSION from BATCH_JOB_EXECUTION where JOB_INSTANCE_ID = 17 order by JOB_EXECUTION_ID
desc

SELECT STEP_EXECUTION_ID, STEP_NAME, START_TIME, END_TIME, STATUS, COMMIT_COUNT, READ_COUNT,
FILTER_COUNT, WRITE_COUNT, EXIT_CODE, EXIT_MESSAGE, READ_SKIP_COUNT, WRITE_SKIP_COUNT,
PROCESS_SKIP_COUNT, ROLLBACK_COUNT, LAST_UPDATED, VERSION from BATCH_STEP_EXECUTION where
JOB_EXECUTION_ID = 17 order by STEP_EXECUTION_ID

select BATCH_STEP_EXECUTION_SEQ.nextval from dual

INSERT into BATCH_STEP_EXECUTION(STEP_EXECUTION_ID, VERSION, STEP_NAME, JOB_EXECUTION_ID, START_TIME,
END_TIME, STATUS, COMMIT_COUNT, READ_COUNT, FILTER_COUNT, WRITE_COUNT, EXIT_CODE, EXIT_MESSAGE,
READ_SKIP_COUNT, WRITE_SKIP_COUNT, PROCESS_SKIP_COUNT, ROLLBACK_COUNT, LAST_UPDATED) values(17, 0,
'readWriteProducts', 17, '01/14/2020 14:55:11.728', NULL, 'STARTING', 0, 0, 0, 0, 'EXECUTING', '', 0,
0, 0, 0, '01/14/2020 14:55:11.729')

INSERT INTO BATCH_STEP_EXECUTION_CONTEXT (SHORT_CONTEXT, SERIALIZED_CONTEXT, STEP_EXECUTION_ID)
VALUES('{"map":""}', NULL, 17)

UPDATE BATCH_STEP_EXECUTION set START_TIME = '01/14/2020 14:55:11.742', END_TIME = NULL, STATUS =
'STARTED', COMMIT_COUNT = 0, READ_COUNT = 0, FILTER_COUNT = 0, WRITE_COUNT = 0, EXIT_CODE =
'EXECUTING', EXIT_MESSAGE = '', VERSION = 1, READ_SKIP_COUNT = 0, PROCESS_SKIP_COUNT = 0,
WRITE_SKIP_COUNT = 0, ROLLBACK_COUNT = 0, LAST_UPDATED = '01/14/2020 14:55:11.743' where
STEP_EXECUTION_ID = 17 and VERSION = 0

SELECT VERSION FROM BATCH_JOB_EXECUTION WHERE JOB_EXECUTION_ID=17

UPDATE BATCH_STEP_EXECUTION_CONTEXT SET SHORT_CONTEXT = '{"map":{"entry":
[{"string":"FlatFileItemWriter.current.count","long":2552},
{"string":"FlatFileItemWriter.written","long":0},
{"string":"FlatFileItemReader.read.count","int":0}]}}', SERIALIZED_CONTEXT = NULL WHERE
STEP_EXECUTION_ID = 17

UPDATE BATCH_STEP_EXECUTION_CONTEXT SET SHORT_CONTEXT = '{"map":{"entry":
[{"string":"FlatFileItemWriter.current.count","long":2665},
{"string":"FlatFileItemWriter.written","long":3},
{"string":"FlatFileItemReader.read.count","int":3}]}}', SERIALIZED_CONTEXT = NULL WHERE
STEP_EXECUTION_ID = 17

UPDATE BATCH_STEP_EXECUTION set START_TIME = '01/14/2020 14:55:11.742', END_TIME = NULL, STATUS =
'STARTED', COMMIT_COUNT = 1, READ_COUNT = 3, FILTER_COUNT = 0, WRITE_COUNT = 3, EXIT_CODE =
'EXECUTING', EXIT_MESSAGE = '', VERSION = 2, READ_SKIP_COUNT = 0, PROCESS_SKIP_COUNT = 0,
WRITE_SKIP_COUNT = 0, ROLLBACK_COUNT = 0, LAST_UPDATED = '01/14/2020 14:55:11.881' where
STEP_EXECUTION_ID = 17 and VERSION = 1

SELECT VERSION FROM BATCH_JOB_EXECUTION WHERE JOB_EXECUTION_ID=17

UPDATE BATCH_STEP_EXECUTION_CONTEXT SET SHORT_CONTEXT = '{"map":{"entry":
[{"string":"FlatFileItemWriter.current.count","long":2778},
{"string":"FlatFileItemWriter.written","long":6},
{"string":"FlatFileItemReader.read.count","int":6}]}}', SERIALIZED_CONTEXT = NULL WHERE
STEP_EXECUTION_ID = 17

UPDATE BATCH_STEP_EXECUTION set START_TIME = '01/14/2020 14:55:11.742', END_TIME = NULL, STATUS =
'STARTED', COMMIT_COUNT = 2, READ_COUNT = 6, FILTER_COUNT = 0, WRITE_COUNT = 6, EXIT_CODE =
'EXECUTING', EXIT_MESSAGE = '', VERSION = 3, READ_SKIP_COUNT = 0, PROCESS_SKIP_COUNT = 0,
WRITE_SKIP_COUNT = 0, ROLLBACK_COUNT = 0, LAST_UPDATED = '01/14/2020 14:55:11.903' where
STEP_EXECUTION_ID = 17 and VERSION = 2

SELECT VERSION FROM BATCH_JOB_EXECUTION WHERE JOB_EXECUTION_ID=17

UPDATE BATCH_STEP_EXECUTION_CONTEXT SET SHORT_CONTEXT = '{"map":{"entry":
[{"string":"FlatFileItemWriter.current.count","long":2871},
{"string":"FlatFileItemWriter.written","long":8},
{"string":"FlatFileItemReader.read.count","int":9}]}}', SERIALIZED_CONTEXT = NULL WHERE
STEP_EXECUTION_ID = 17

UPDATE BATCH_STEP_EXECUTION set START_TIME = '01/14/2020 14:55:11.742', END_TIME = NULL, STATUS =
'STARTED', COMMIT_COUNT = 3, READ_COUNT = 8, FILTER_COUNT = 0, WRITE_COUNT = 8, EXIT_CODE =
'EXECUTING', EXIT_MESSAGE = '', VERSION = 4, READ_SKIP_COUNT = 0, PROCESS_SKIP_COUNT = 0,
WRITE_SKIP_COUNT = 0, ROLLBACK_COUNT = 0, LAST_UPDATED = '01/14/2020 14:55:11.921' where
STEP_EXECUTION_ID = 17 and VERSION = 3

SELECT VERSION FROM BATCH_JOB_EXECUTION WHERE JOB_EXECUTION_ID=17

UPDATE BATCH_STEP_EXECUTION_CONTEXT SET SHORT_CONTEXT = '{"map":{"entry":
[{"string":"FlatFileItemWriter.current.count","long":2871},
{"string":"FlatFileItemWriter.written","long":8},
{"string":"FlatFileItemReader.read.count","int":9}]}}', SERIALIZED_CONTEXT = NULL WHERE
STEP_EXECUTION_ID = 17

UPDATE BATCH_STEP_EXECUTION set START_TIME = '01/14/2020 14:55:11.742', END_TIME = '01/14/2020
14:55:11.941', STATUS = 'COMPLETED', COMMIT_COUNT = 3, READ_COUNT = 8, FILTER_COUNT = 0, WRITE_COUNT =
8, EXIT_CODE = 'COMPLETED', EXIT_MESSAGE = '', VERSION = 5, READ_SKIP_COUNT = 0, PROCESS_SKIP_COUNT =
0, WRITE_SKIP_COUNT = 0, ROLLBACK_COUNT = 0, LAST_UPDATED = '01/14/2020 14:55:11.942' where
STEP_EXECUTION_ID = 17 and VERSION = 4

SELECT VERSION FROM BATCH_JOB_EXECUTION WHERE JOB_EXECUTION_ID=17

UPDATE BATCH_JOB_EXECUTION_CONTEXT SET SHORT_CONTEXT = '{"map":""}', SERIALIZED_CONTEXT = NULL WHERE
JOB_EXECUTION_ID = 17


1755  INFO  e.b.s.EIMBatchMonitoringListener - Ended job JobExecution: id=17, version=1, startTime=Tue
Jan 14 14:55:11 EST 2020, endTime=Tue Jan 14 14:55:11 EST 2020, lastUpdated=Tue Jan 14 14:55:11 EST
2020, status=COMPLETED, exitStatus=exitCode=COMPLETED;exitDescription=, job=[JobInstance: id=17,
version=0, JobParameters=[{targetDirectory=./target/test-classes/data/, inputFile=products.txt,
outputFile=products-out.txt, timestamp=1579031711440}], Job=[importProducts]]


SELECT COUNT(*) FROM BATCH_JOB_EXECUTION WHERE JOB_EXECUTION_ID = 17

UPDATE BATCH_JOB_EXECUTION set START_TIME = '01/14/2020 14:55:11.691', END_TIME = '01/14/2020
14:55:11.962', STATUS = 'COMPLETED', EXIT_CODE = 'COMPLETED', EXIT_MESSAGE = '', VERSION = 2,
CREATE_TIME = '01/14/2020 14:55:11.655', LAST_UPDATED = '01/14/2020 14:55:11.964' where
JOB_EXECUTION_ID = 17 and VERSION = 1
```

References:

https://docs.spring.io/spring-batch/docs/4.1.x/reference/html/index.html
http://www.h2database.com/html/main.html