스프링 배치 메타테이블

잉어잉어킹 · 어제

팔로우 ♥ 0

> **스프링 배치(Batch)란?**
> Batch (명): 일괄, 집단, 무리

대용량 데이터 처리를 위해 설계된 프레임워크이다.
주기적으로 대량의 데이터를 일괄적으로 처리 할 때 유용하게 사용할 수 있다고 한다.

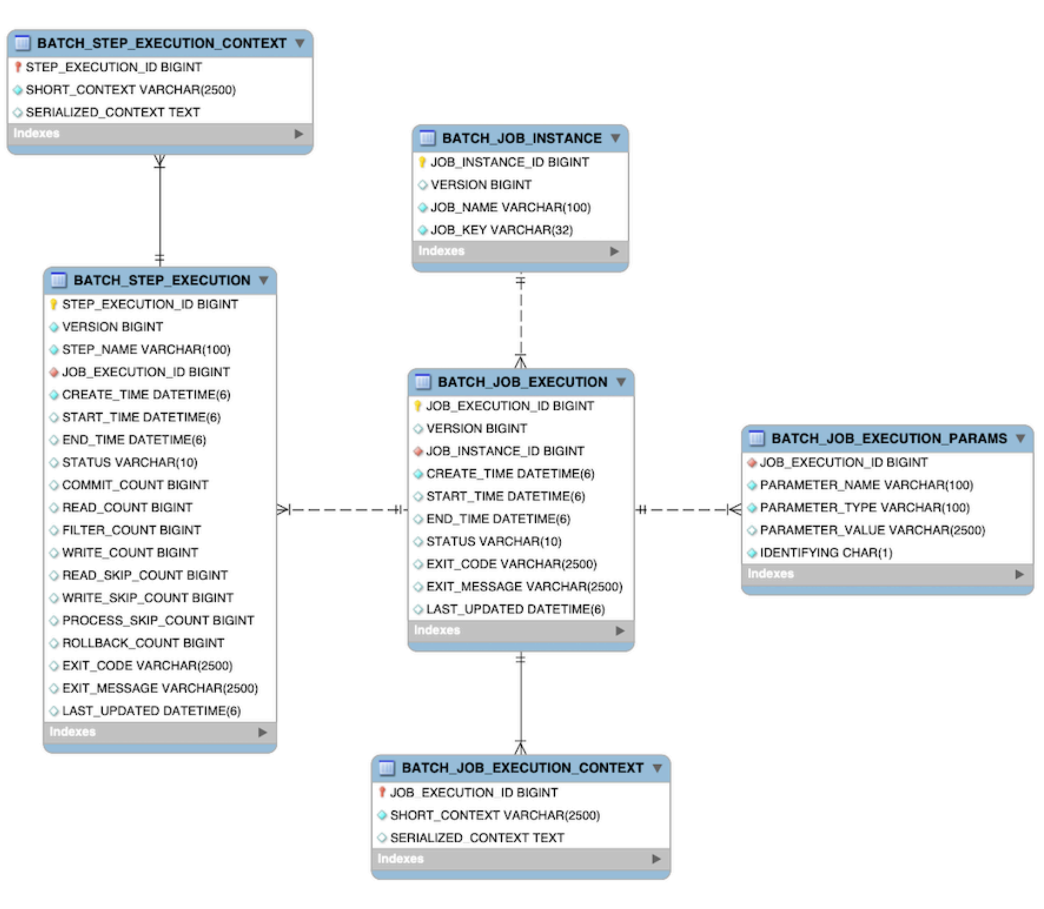
Job, Step, Job Parameter
스프링 배치를 처음 생성하고 프로젝트를 시작하면, 총 6개의 메타테이블이 자동으로 생성되는 것을 볼 수 있다. 각각 job, step, jop parameter를 다루는 테이블들이다.

- Job
  배치에서 수행되는 일련의 배치 프로세스 전체를 나타내는 단위이다.
  예시) 파일을 읽고 DB에 저장하는 작업을 배치로 수행한다면, 이것 자체가 Job이 된다.

- Step
  Job을 구성하는 작은 단위이다.
  ItemReader(읽기), ItemProcessor(처리), ItemWriter(쓰기)
  예시) 파일에서 데이터를 읽기, 데이터 가공하기, DB에 저장하기

- Job Parameter
  Job 실행 시 외부에서 주입되는 파라미터
  예시) 특정 날짜의 데이터를 처리하도록 하는 Job일 때, 그날의 날짜

> **메타 테이블 구조**



BATCH_JOB_INSTANCE
실행한 Job의 인스턴스에 대한 정보를 저장한다.
이때 동일한 Job Parameter로 실행된 Job일 경우는 새로운 레코드가 생성되지 않는다.

job_inctance_id	Job의 고유 번호(PK)
version	버전정보, update 시마다 1씩 증가
job_name	job 이름
job_key	Job Parameter의 값, 다른 잡과 구분하기 위해 사용

BATCH_JOB_EXECUTION
Job 실행에 관한 정보를 저장한다.
Job이 실행될 때, JobExecution이라는 새로운 객체가 있으며, 이 테이블에 새로운 레코드로 생성된다.

job_execution_id	Job 실행의 고유 번호(PK)
version	버전정보
job_instance_id	BATCH_JOB_INSTANCE 테이블의 기본키(FK)
create_time	execution 생성 시간
start_time	execution 시작 시간
end_time	execution 종료 시간, 성공/실패 모두 남게 됨
status	execution의 현재 상태
exit_code	execution의 종료 코드
exit_message	Job이 종료될 경우 어떻게 종료되었는지
last_updated	execution이 마지막으로 지속된 시간

end_time의 값이 비어 있다면, 특정 유형의 오류가 발생해 프레임워크가 실패하기 전 마지막 저장을 수행할 수 없음을 뜻한다.

BATCH_JOB_EXECUTION_PARAMS
Job Parameter에 대한 정보를 저장한다.
하나 이상의 key/value 쌍으로 Job에 전달되며, Job이 실행될 때 전달된 파라미터 정보를 해당 테이블에 저장하게 된다.
IDENTIFYING 라는 컬럼이 true로 설정되면 해당 Job Parameter가 Job Instance의 유니크성을 위해 사용되었다는 뜻이다.

job_execution_id	Job 실행 고유 번호(FK, 기본키 아님)
parameter_name	파라미터 이름
parameter_type	파라미터 타입
parameter_value	파라미터 값
IDENTIFYING	파라미터가 Job Instance의 유니크성을 위해 사용된 파라미터라면 ture로 세팅

BATCH_STEP_EXECUTION
StepExecution과 관련된 정보를 저장한다.

step_execution_id	step 실행 고유 번호(PK)
version	버전정보
step_name	step 이름
job_execution_id	BATCH_JOB_EXECUTION의 기본키(FK)
start_time	execution 시작 시간
end_time	execution 종료 시간
status	execution의 상태
commit_count	execution 동안 트랜잭션 커밋된 카운트 나열
read_count	실행 동안 읽어들인 아이템 수
filter_count	실행 동안 필터된 아이템 수
write_count	실행 동한 쓰기된 아이템 수
read_skip_count	실행 동안 읽기 시 스킵된 아이템 수
write_skip_count	실행 동안 쓰기 시 스킵된 아이템 수
process_skip_count	실행 동안 프로세서가 스킵된 아이엠
rollback_count	실행 동안 롤백된 아이템 수
exit_code	실행 동안 종료된 문자열
exit_message	Job이 어떻게 종료되었는지
last_updated	execution이 마지막으로 지속된 시간

end_time의 값이 비어 있다면, 특정 유형의 오류가 발생해 프레임워크가 실패하기 전 마지막 저장을 수행할 수 없음을 뜻한다.

BATCH_JOB_EXECUTION_CONTEXT
Job의 ExcutionContext에 대한 정보를 저장한다.
Job Execution 하나당 하나씩의 Job Execution Context를 갖게 되며, 해당 Context 정보를 통해 동일한 Job Scope 내에서 데이터를 공유할 수 있다.
실패 후 중단된 부분부터 다시 시작될 수 있도록, 검색해야하는 상태를 제공한다.

job_execution_id	Job 실행 ID(FK)
short_context	serialized 컨텍스트의 문자로 된 버전
serialized_context	전체 컨텍스트

BATCH_STEP_EXECUTION_CONTEXT
Step의 ExcutionContext에 대한 정보를 저장한다.

step_execution_id	Step 실행 ID(FK)
short_context	serialized 컨텍스트의 문자로 된 버전
serialized_context	전체 컨텍스트

> **시퀀스 테이블**

Job과 Step 실행의 각 기록마다 고유한 ID가 필요한데, 이때 시퀀스 테이블이 고유한 값을 생성해 준다.

BATCH_JOB_SEQ
Job에 대한 시퀀스 테이블이다.

```
id:
    bigint
    배치 job의 기본키

unigue key:
    char(1)
    배치 Job 시퀀스를 구별하는 유니크 PK
```

BATCH_JOB_EXECUTION_SEQ
Job Execution의 시퀀스 테이블이다.

```
id:
    bigint
    배치 Job Execution의 기본키

unigue key:
    char(1)
    배치 Job Execution 시퀀스를 구별하는 유니크 PK
```

BATCH_STEP_EXECUTION_SEQ
STEP Execution의 시퀀스 테이블이다.

```
id:
    bigint
    배치 Step Execution의 기본키

unigue key:
    char(1)
    배치 Step Execution 시퀀스를 구별하는 유니크 PK

해당 시퀀스를 통해
Batch_Job_Instance, Batch_Execution, Batch_Step_Execut
의 시퀀스를 배치가 할당하며, 이 값은 중복될 수 없다.
```