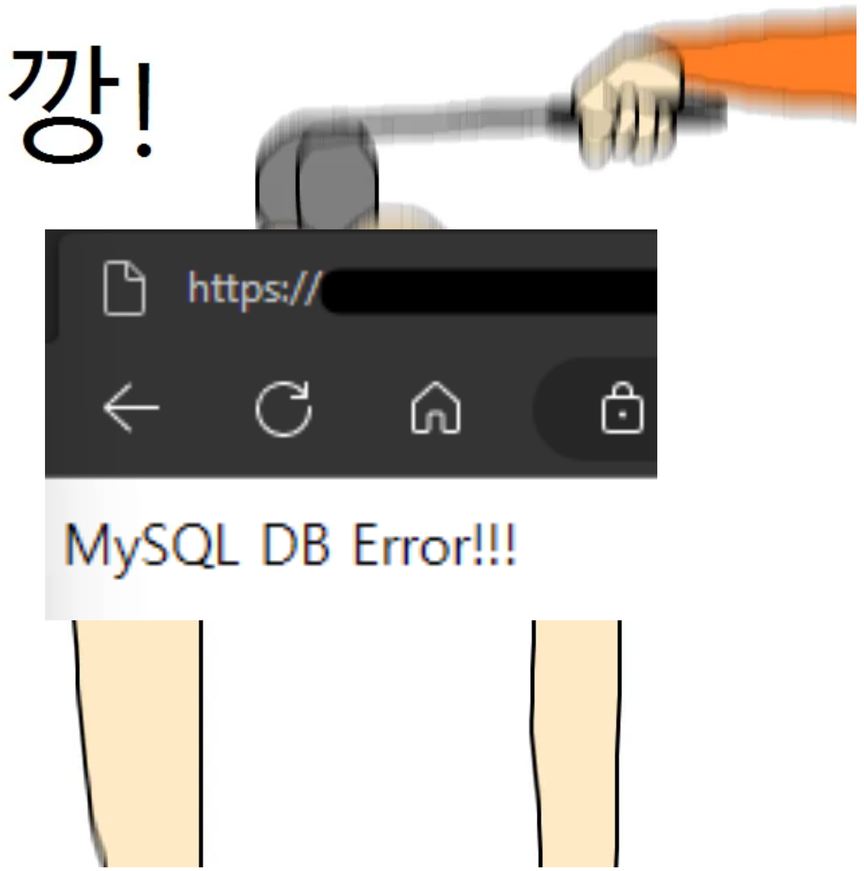
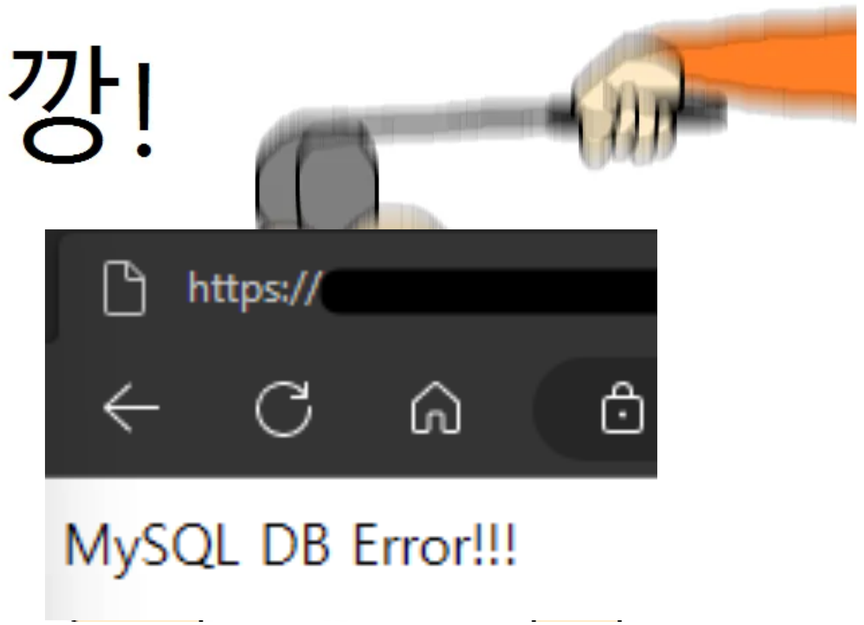


테이블 스키마 무중단으로 변경하기

백엔드 7기 모코



(있어보이게)

테이블 스키마 온라인 마이그레이션

Table Scheme Online Migration

백엔드 7기 모코

4. DB 데이터를 drop 하지 않는다

- 이건 정말 중요합니다! 실제 운영에서는:
  - 테이블 구조 바꿔야 해도 DROP 하지 않습니다 (마이그레이션)
  - 데이터 지워야 해도 DROP 하지 않습니다 (soft delete)
  - 스키마 변경도 무중단으로 해야 합니다
  - "잠깐 서비스 중단하고 DB 재설치할게요" ← 망하는 지름길
- 왜? 고객 데이터 = 회사의 생명입니다
- 실수 방지 시스템을 만드세요
  - 실수로 DROP 명령어를 써도 막을 수 있어야겠죠?
  - 아니면 DB 계정에서 권한을 빼세요.

스키마 마이그레이션이 무중단으로 가능한가?

- 기존 테이블의 스키마를 변경하려면 테이블 락이 걸릴텐데
- 그럼 그동안 데이터 접근(Read, Write)이 안될텐데
- 그럼 그동안 정상적인 서비스 운영이 안될텐데
- 오.. 근데 테이블 락을 막을 수 있다면 스키마 무중단 마이그레이션이 가능할 것 같아.

근데 그게 가능해?

MySQL에서 테이블 스키마를 “무중단”으로 변경해보자!!

Posted on 2012-05-22 (Last modified on 2025-08-19) | 7 minutes | gywndi

1. **임시 테이블 생성**
    - Insert/Delete 시 변경 사항을 저장할 테이블 (TAB01_INSERT, TAB01_DELETE)
    - 데이터를 임시로 저장할 테이블
2. **트리거 생성**
    - 원본 테이블에 Delete 발생 시 해당 ROW를 임시 테이블에 저장
    - 원본 테이블에 Insert 발생 시 해당 ROW를 임시 테이블에 저장
3. **원본 테이블 export/import**
    - export -> 테이블명 변경 -> import
    - SED로 테이블 명을 임시 테이블 명으로 변경
4. **원본 테이블과 임시 테이블 RENAME**
    - 원본 테이블 : TAB01 -> TAB01_OLD
    - 임시 테이블 : TAB01_TMP -> TAB01
5. **테이블 변경 분 저장**
    - TAB01_INSERT에 저장된 데이터 Insert
    - TAB01_DELETE에 저장된 데이터 Delete
6. **트리거, 임시테이블 제거**
    - 트리거 : Insert트리거, Delete 트리거
    - 테이블 : TAB01_INSERT, TAB01_DELETE

MySQL에서 테이블 스키마를 “무중단”으로 변경해보자!!

Posted on 2012-05-22 (Last modified on 2025-08-19) | 7 minutes | gywndi

1. **임시 테이블 생성**
    - Insert/Delete 시 변경 사항을 저장할 테이블 (TAB01_INSERT, TAB01_DELETE)
    - 데이터를 임시로 저장할 테이블
2. **트리거 생성**
    - 원본 테이블에 Delete 발생 시 해당 ROW를 임시 테이블에 저장
    - 원본 테이블에 Insert 발생 시 해당 ROW를 임시 테이블에 저장
3. **원본 테이블 export/import**
    - export -> 테이블명 변경 -> import
    - SED로 테이블 명을 임시 테이블 명으로 변경
4. **원본 테이블과 임시 테이블 RENAME**
    - 원본 테이블 : TAB01 -> TAB01_OLD
    - 임시 테이블 : TAB01_TMP -> TAB01
5. **테이블 변경 분 저장**
    - TAB01_INSERT에 저장된 데이터 Insert
    - TAB01_DELETE에 저장된 데이터 Delete
6. **트리거, 임시테이블 제거**
    - 트리거 : Insert트리거, Delete 트리거
    - 테이블 : TAB01_INSERT, TAB01_DELETE

메타데이터 락

(매우 짧은 시간, 테이블 크기와 무관)

MySQL에서 테이블 스키마를 “무중단”으로 변경해보자!!

Posted on 2012-05-22 (Last modified on 2025-08-19) | 7 minutes | gywndi

1. **임시 테이블 생성**
    - Insert/Delete 시 변경 사항을 저장할 테이블 (TAB01_INSERT, TAB01_DELETE)
    - 데이터를 임시로 저장할 테이블
2. **트리거 생성**
    - 원본 테이블에 Delete 발생 시 해당 ROW를 임시 테이블에 저장
    - 원본 테이블에 Insert 발생 시 해당 ROW를 임시 테이블에 저장
3. **원본 테이블 export/import**
    - export -> 테이블명 변경 -> import
    - SED로 테이블 명을 임시 테이블 명으로 변경
4. **원본 테이블과 임시 테이블 RENAME**
    - 원본 테이블 : TAB01 -> TAB01_OLD
    - 임시 테이블 : TAB01_TMP -> TAB01

메타데이터 락

(매우 짧은 시간, 테이블 크기와 무관)

- **공식 문서가 아닌 블로그를 완전히 신뢰해도 괜찮은 것일까?**
- **이 방법이 정말 최선이라면 왜 공식 기능으로 제공하지 않고 있을까?**
- **포스팅 일자가 2012년(작성일 기준 13년 전)이다. 그 사이에 변경된 것이 많을 수 있다.**

시대가 바뀌었넹

- mysql 5.6에서 **Online DDL** 도입
  - 게시글은 5.5 버전에서 작성됨

MySQL 5.6 이전

1.스키마 변경 사항이 반영된 **임시 테이블을 생성**한다.

2.대상 테이블에 **Read Lock**을 걸고 **데이터를 복사**한다.

3.임시 테이블에 대상 테이블 **데이터를 복제**한다.

4.임시 테이블을 대상 테이블과 **교체**한다.

MySQL 5.6 이후

DDL이 **가벼운 Online DDL**과 **무거운 Offline DDL**로 나뉨.

- Offline DDL (기존의 DDL 처리 방식)
  - DML을 거부하고 대상 테이블에 **Read(&Write) Lock**을 건다.
- Online DDL
  - DDL 적용 과정에서 DML(INSERT, UPDATE, DELETE)를 대체로 **허용**

ALGORITHM

- **COPY**
  - Offline DDL
  - 대상 테이블을 복사하여 임시 테이블을 만들어 작업한다.
  - 테이블 데이터는 대상 테이블으로부터 행 단위로 복사된다.
  - 동시 DML을 허용하지 않는다.
- **INPLACE**
  - Online DDL(MySQL 5.6+ 제공)
  - 임시 테이블을 생성하지 않지만, 경우에 따라 테이블 리빌딩이 발생할 수 있다.
  - 대부분 동시 DML을 허용한다.
- **INSTANT**
  - Online DDL(MySQL 8.0.12+ 제공)
  - Data Dictionary에서 메타 데이터만 수정한다.
  - 테이블 데이터의 영향을 받지 않으므로 작업이 즉시 진행된다.
  - 동시 DML을 허용한다.

스키마 중단 변경
(Scheme Offline Migration)

스키마 무중단 변경
(Scheme Online Migration)

“Did the Online Schema Migration go smoothly?” (스키마 무중단 변경이 순조롭게 진행되었나요?)

Me (DBA):

Online DDL은 이미 무중단 변경이구나!
그럼.. Offline DDL은 어떡하지?

Offline DDL의 Online Migration 방법

- **프로시저와 트리거** 기반 스키마 변경
- **트리거** 기반 스키마 변경 도구
- **바이너리 로그** 기반 스키마 변경 도구

프로시저와 트리거 기반 스키마 변경

커스텀 쿼리 작성

1. 스키마 변경을 적용한 **임시 테이블을 생성**한다.
2. 스키마 변경 간 적용되는 **데이터 변경 내역을 관리할 테이블**을 추가로 생성한다.
3. 대상 테이블에서 변경이 일어날 시 변경 내역 테이블을 호출하도록 **트리거를 생성**한다.
4. 임시 테이블로 **데이터를 복사**한다.
5. 변경사항을 임시 테이블에 반영하고 **테이블 이름을 스왑**한다.
6. 변경사항을 다시 임시 테이블에 반영하고 **트리거 및 변경 내역 테이블을 제거**한다.

프로시저와 트리거 기반 스키마 변경

커스텀 쿼리 작성

- **프로시저**를 만들어두면 **커스텀 쿼리 생성을 자동화**할 수 있다.
- 하지만 **복잡한 DDL**에 대해서는 문제가 발생할 수 있고 프로시저는 **유지보수가 어렵다.**

프로

커스텀

- 프로
- 하지

생

다.

```
DELIMITER $$
DROP PROCEDURE print_rb_query $$
CREATE PROCEDURE print_rb_query(IN P_DB VARCHAR(255), IN P_TAB VARCHAR(255), IN P_ALTER_STR
BEGIN
    ## Declare Variables
    SET @qry = '\n';
    SET @ai_num = 0;
    SET @ai_name = '';
    SET @col_list = '';

    ## Get AUTO_INCREMENT Number for Temporary Table
    SELECT CAST(AUTO_INCREMENT*1.01 AS UNSIGNED) INTO @ai_num
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_SCHEMA = P_DB
    AND TABLE_NAME = P_TAB;

    ## Get auto_increment Column Name
    SELECT COLUMN_NAME INTO @ai_name
    FROM INFORMATION_SCHEMA.COLUMNS
    WHERE TABLE_SCHEMA = P_DB
    AND TABLE_NAME = P_TAB
    AND EXTRA = 'auto_increment';

    SET @qry = CONCAT(@qry, '>> Step 1>  Prepare : SQL &lt;&lt;\n');
    SET @qry = CONCAT(@qry, 'DROP TABLE IF EXISTS ',P_DB,'.',P_TAB,'_INSERT;\n');
    SET @qry = CONCAT(@qry, 'CREATE TABLE ',P_DB,'.',P_TAB,'_INSERT LIKE ',P_TAB,';\n');
    SET @qry = CONCAT(@qry, 'DROP TABLE IF EXISTS ',P_DB,'.',P_TAB,'_DELETE;\n');
    SET @qry = CONCAT(@qry, 'CREATE TABLE ',P_DB,'.',P_TAB,'_DELETE LIKE ',P_TAB,';\n');
    SET @qry = CONCAT(@qry, 'DROP TABLE IF EXISTS ',P_DB,'.',P_TAB,'_TMP;\n');
    SET @qry = CONCAT(@qry, 'CREATE TABLE ',P_DB,'.',P_TAB,'_TMP LIKE ',P_TAB,';\n');
    SET @qry = CONCAT(@qry, 'ALTER TABLE ',P_DB,'.',P_TAB,'_TMP AUTO_INCREMENT = ',@ai_num,'
    SET @qry = CONCAT(@qry, 'ALTER TABLE ',P_DB,'.',P_TAB,'_TMP ',P_ALTER_STR,'\n');

    ## Change Delimiter
    SET @qry = CONCAT(@qry, 'DELIMITER $$\n');

    ## Get Column List for Delete Trigger
    select GROUP_CONCAT('\n    OLD.',COLUMN_NAME) into @col_list
    from INFORMATION_SCHEMA.COLUMNS
    where TABLE_SCHEMA = P_DB
    and table_name = P_TAB
    order by ORDINAL_POSITION;

    SET @qry = CONCAT(@qry, 'DROP TRIGGER IF EXISTS ',P_DB,'.TRG_',P_TAB,'_DELETE$$\n');
    SET @qry = CONCAT(@qry, 'CREATE TRIGGER ',P_DB,'.TRG_',P_TAB,'_DELETE\n');
    SET @qry = CONCAT(@qry, 'AFTER DELETE ON ',P_DB,'.',P_TAB,'\n');
    SET @qry = CONCAT(@qry, 'FOR EACH ROW\n');
    SET @qry = CONCAT(@qry, 'BEGIN\n');
    SET @qry = CONCAT(@qry, 'INSERT INTO ',P_DB,'.',P_TAB,'_DELETE VALUES(');
    SET @qry = CONCAT(@qry, @col_list);
    SET @qry = CONCAT(@qry, ');\n');
    SET @qry = CONCAT(@qry, 'END$$\n');
```

```
    ## Get Column List for Insert Trigger
    select GROUP_CONCAT('\n    NEW.',COLUMN_NAME) into @col_list
    from INFORMATION_SCHEMA.COLUMNS
    where TABLE_SCHEMA = P_DB
    and table_name = P_TAB
    order by ORDINAL_POSITION;
    SET @qry = CONCAT(@qry, 'DROP TRIGGER IF EXISTS ',P_DB,'.TRG_',P_TAB,'_INSERT$$\n');
    SET @qry = CONCAT(@qry, 'CREATE TRIGGER ',P_DB,'.TRG_',P_TAB,'_INSERT\n');
    SET @qry = CONCAT(@qry, 'AFTER INSERT ON ',P_DB,'.',P_TAB,'\n');
    SET @qry = CONCAT(@qry, 'FOR EACH ROW\n');
    SET @qry = CONCAT(@qry, 'BEGIN\n');
    SET @qry = CONCAT(@qry, 'INSERT INTO ',P_DB,'.',P_TAB,'_INSERT VALUES(');
    SET @qry = CONCAT(@qry, @col_list);
    SET @qry = CONCAT(@qry, ');\n');
    SET @qry = CONCAT(@qry, 'END$$\n');

    ## Change Delimiter
    SET @qry = CONCAT(@qry, 'DELIMITER ;\n');
    SET @qry = CONCAT(@qry, '\n\n');

    ## Insert Data
    SET @qry = CONCAT(@qry, '>> Step 2>  Data Copy : Shell Script &lt;&lt;\n');
    SET @qry = CONCAT(@qry, 'mig_dif_tab.sh ',P_DB,' ',P_TAB,' ',P_TAB,'_TMP\n');
    SET @qry = CONCAT(@qry, '\n\n');
    SET @qry = CONCAT(@qry, '>> Step 3>  Final Job : SQL &lt;&lt;\n');

    ## Insert Data into Temporary Table
    SET @qry = CONCAT(@qry, 'INSERT IGNORE INTO ',P_DB,'.',P_TAB,'_TMP SELECT * FROM ',P_DB,

    ## Delete Data from Temporary Table
    SET @qry = CONCAT(@qry, 'DELETE A FROM ',P_DB,'.',P_TAB,'_TMP A INNER JOIN ',P_DB,'.',P_

    ## Swap table names
    SET @qry = CONCAT(@qry, 'RENAME TABLE ',P_DB,'.',P_TAB,' TO ',P_DB,'.',P_TAB,'_OLD, ',P_

    ## Drop Triggers
    SET @qry = CONCAT(@qry, 'DROP TRIGGER IF EXISTS ',P_DB,'.TRG_',P_TAB,'_DELETE;\n');
    SET @qry = CONCAT(@qry, 'DROP TRIGGER IF EXISTS ',P_DB,'.TRG_',P_TAB,'_INSERT;\n');

    ## Insert Data
    SET @qry = CONCAT(@qry, 'INSERT IGNORE INTO ',P_DB,'.',P_TAB,' SELECT * FROM ',P_DB,'.',

    ## Delete Data
    SET @qry = CONCAT(@qry, 'DELETE A FROM ',P_DB,'.',P_TAB,' A INNER JOIN ',P_DB,'.',P_TAB,

    ## Drop Temporary Tables
    SET @qry = CONCAT(@qry, 'DROP TABLE IF EXISTS ',P_DB,'.',P_TAB,'_INSERT;\n');
    SET @qry = CONCAT(@qry, 'DROP TABLE IF EXISTS ',P_DB,'.',P_TAB,'_DELETE;\n');
    SET @qry = CONCAT(@qry, '\n\n');
    SELECT @qry;
END$$
DELIMITER ;
```

트리거 기반 스키마 변경 도구

pt-online-schema-change

- Perocona Toolkit에서 2011년 출시한 **Online MySQL Scheme Migration 도구**
- **Offline DDL을 online으로 실행**할 수 있다. (ALTER tables without locking them.)
- MySQL 내부의 테이블 변경 방식을 모방
- 대상 테이블의 **복사본에서 작업**하여 원본 테이블 락을 최소화
  - DDL 실행 간 **Read/Write 지원**

트리거 기반 스키마 변경 도구

pt-online-schema-change

1. 대상 테이블을 복사하여 **임시 테이블을 생성**한다.
2. 임시 테이블에 **스키마 변경을 적용**한다.
3. 대상 테이블에 대한 **트리거를 생성**한다.
4. 대상 테이블 **데이터를** 임시 테이블로 **청크 단위로 복사**한다.
5. 복사하는 동안 대상 테이블의 변경사항이 **트리거에 의해** 임시 테이블에 실시간 반영된다.
6. 복사 완료 후 잔여 변경사항을 **최종 동기화**한다.
7. 대상 테이블 데이터 복사 완료 후 **임시 테이블과 교체**한다.(RENAME TABLE, 원자성 보장)
8. 대상 테이블을 **삭제**한다.

바이너리 로그 기반 스키마 변경 도구

gh-ost

- 깃허브에서 2016년 출시한 **Online MySQL Scheme Migration 도구**
- **Offline DDL을 online으로 실행**할 수 있다.
- gh-ost는 스키마 대신 **바이너리 로그 스트림**(Binary Log Stream)을 사용함으로써 트리거로 인한 제약과 위험을 회피한다.
  - 기존 온라인 스키마 변경 도구들은 트리거를 사용

바이너리 로그 기반 스키마 변경 도구

gh-ost

1. 대상 테이블을 복사하여 **임시(고스트) 테이블을 생성**한다.
2. 대상 테이블 데이터를 임시 테이블로 **점진적 복사**한다.
3. **바이너리 로그 스트림**을 통해 원본 테이블의 변경사항을 캡처하고
   임시 테이블에 **비동기적으로 반영**한다.
4. 데이터 복사 및 변경사항 동기화가 완료되면 **대상 테이블과 임시 테이블을 교체**한다.

pt-online-scheme-change vs gh-ost

기능	gh-ost	pt-online-schema-change
트리거	**없음** (바이너리 로그 사용)	**있음** (DML 작업에 추가)
데이터 동기화	바이너리 로그를 통한 비동기	트리거를 통한 동기
마스터 부하	낮음 (복제본 사용)	높음 (마스터에 트리거 작동)
FK 지원	**없음** (드롭 후 재생성 필요)	**있음** (`--alter-foreign-keys-method` 옵션)
복제	RBR 필요, 복제본 선호	SBR/RBR 모두 작동, 마스터에서 실행
안전성	**고급** (훅, 스로틀링, 체크)	양호 (스로틀링, 기본 체크)
컷오버	수동, 제어 가능, 짧은 락	즉시 이름 변경, 제어 부족
재개 가능	예	부분적 (아티팩트가 남아있는 경우)
MySQL 버전	5.7 이상만	5.5 이상 (구 버전 지원)
스로틀링	**고급** (지연/훅을 통한 일시정지/재개)	기본 (복사 스로틀링만)
제한사항	JSON, 파티션, FK, 트리거	적음; PXC는 일부 제약 사항 있음
복제본 테스트	**정확함** (바이너리 로그 기반)	제한적 (트리거가 작동하지 않음)

최종 정리

- MySQL은 **Online DDL**에 한해 **이미 스키마 무중단 변경**을 제공한다.
- **Offline DDL**은 진행 간 테이블에 **Read Lock**이 걸려 서비스에 영향을 미친다.
- 이를 방지하기 위해 **각종 도구가 출시**되었고 현재도 꾸준히 사용되고 있다.
  - **트리거** 기반 마이그레이션: pt-online-schema-change
  - **바이너리 로그** 기반 마이그레이션: gh-ost
- 하지만 내부적으로 **복제 테이블**을 만들어 작업하는 특성 상 **용량을 많이 차지**하며 **DB 성능을 저하**시킨다. (스로틀링을 제공하나 기본적으로 저하시키는 건 맞음)

결론

그냥 중단 변경 하자. 무슨 무중단이여

- Offline DDL의 Online Migration은 여러 단점이 있다.
  - **DB의 부하**를 일으킨다. (성능, 용량)
  - **롤백**이 번거롭다.
- 무중단이라고 무조건 좋은 게 아니다.
  - 위의 사이드이펙트를 감수하고 할 정도로 시급한 게 아니라면
    그냥 **점검 시간**(Maintenance Window)을 가지고 **Offline Migration을 진행**하자.

감사합니다

출처: 모코의 블로그
https://velog.io/@songsunkook/테이블-스키마-무중단으로-변경하기