5주차

7. 스키마 조작(DDL)

DBMS 서버의 모든 오브젝트를 생성하거나 변경하는 쿼리

스키마를 변경하는 작업은 시간이 오래 걸리고 MySQL 서버에 많은 부하를 발생 시키는 작업들이 있으므로 주의해야 한다.

7.1 온라인 DDL

테이블 구조가 변경되는 동안 다른 커넥션에서 DML을 실행할 수 없었지만 8.0 부터 내장 온라인 DDL 기능으로 처리가 가능해졌다.

1. 온라인 DDL 알고리즘

`older_alter_table` 시스템 변수를 이용해 `ALTER TABLE` 명령이 온라인 DDL로 동작할지 여부를 설정 (기본값은 OFF로 온라인 DDL 활성화)

`ALTER TABLE` 명령 실행 시 알고리즘 선택 순서(우선순위가 낮을수록 더 큰 잠금과 작업이 필요)

1. `ALGORITHM=INSTANT`로 스키마 변경이 가능한지 확인 후, 가능하다면 선택
2. `ALGORITHM=INPLACE`로 스키마 변경이 가능한지 확인 후, 가능하다면 선택
3. `ALGORITHM=COPY` 알고리즘 선택

- `INSTANT`
  - 테이블 데이터는 변경하지 않고 메타데이터만 변경하고 작업 완료
  - 스키마 변경 도중 테이블의 읽고 쓰기는 대기하지만 스키마 변경 시간이 매우 짧음
- `INPLACE`
  - 임시 테이블로 데이터를 복사하지 않고 스키마 변경 실행
  - 스키마 변경 도중 테이블의 읽기 쓰기가 가능하지만 테이블의 모든 레코드를 리빌드해야 할 수 있기 때문에 테이블의 크기에 따라 많은 시간이 소요될 수 있음
  - 최초 시작 시점과 마지막 종료 시점에 매우 짧게 테이블의 읽고 쓰기가 불가능
- `COPY`
  - 변경된 스키마를 적용한 임시 테이블을 생성해 모든 레코드를 복사하고 최종적으로 임시 테이블을 `RENAME`하여 스키마 변경
  - 테이블 읽기만 가능(`INSERT`, `UPDATE`, `DELETE`는 실행 불가)

온라인 DDL 명령은 알고리즘과 함께 잠금 수준도 명시 가능

```
ALTER TABLE salaries
    CHANGE to_date end_date DATE NOT NULL,
    ALGORITHM = INPLACE,
    LOCK = NONE;
```

`INSTANT` 알고리즘은 메타데이터 잠금만 필요 ⇒ `LOCK` 옵션 명시 불가

`LOCK` 종류

- `NONE`: 잠금 없음
- `SHARED`: 읽기 잠금 (스키마 변경 중 `INSERT`, `UPDATE`, `DELETE` 불가)
- `EXCLUSIVE`: 쓰기 잠금 (스키마 변경 중 읽기 쓰기 불가)

> `INPLACE` 알고리즘은 대부분 `NONE`으로 설정 가능하지만 가끔 `SHARED` 수준까지 설정해야 할 수 있다. `EXCLUSIVE`는 예전 MySQL 서버의 전통적인 `ALTER TABLE`과 동일하므로 굳이 `LOCK`을 명시할 필요가 없다.

PK를 추가하는 경우와 같이 테이블 레코드의 리빌드가 필요한 경우를 MySQL 서버 매뉴얼에서는 Data Reorganizing 또는 Table Rebuild라고 명명

`INPLACE` 알고리즘을 사용하는 경우

- 테이블 리빌드가 필요한 경우: 잠금을 필요로 하지 않기 때문에 읽고 쓰기가 가능하지만 여전히 테이블의 레코드 건수에 따라 상당히 많은 시간이 소요될 수 있다.
- 테이블 리빌드가 필요하지 않은 경우: `INPLACE` 알고리즘을 사용하지만 `INSTANT` 알고리즘과 비슷하게 매우 빨리 작업이 완료될 수 있다.

2. 온라인 처리 가능한 스키마 변경

MySQL :: MySQL 8.0 Reference Manual :: 17.12.1 Online DDL Operations

Online support details, syntax examples, and usage notes for DDL operations are provided under the following topics in this section.

https://dev.mysql.com/doc/refman/8.0/en/innodb-online-ddl-operations.html

`ALTER TABLE` 문장에 `LOCK`과 `ALGORITHM`을 명시함으로써 온라인 스키마 변경 처리 알고리즘을 강제할 수 있다. (하지만 무조건 그 알고리즘으로 처리되는 것은 아니지만 명시된 알고리즘으로 온라인 DDL 처리가 불가하다면 에러가 발생한다.)

3. INPLACE 알고리즘

1. `INPLACE` 스키마 변경이 지원되는 스토리지 엔진의 테이블인지 확인
2. `INPLACE` 스키마 변경 준비(스키마 변경에 대한 정보를 준비해 온라인 DDL 작업 동안 변경되는 데이터 추적 준비)
3. 테이블 스키마 변경 및 새로운 DML 로깅(실제 스키마 변경을 수행하는 과정. 다른 커넥션의 DML 작업이 대기하지는 않음. 다른 스레드에서 사용자에 의해 발생한 DML들에 대해 별도 로그 기록)

4. 로그 적용(온라인 DDL 작업 동안 수집된 DML 로그를 테이블에 적용)
5. `INPLACE` 스키마 변경 완료(`COMMIT`)

2번과 4번 단계에서는 잠깐의 Exclusive lock이 필요하며 이 시점에 다른 커넥션의 DML들이 잠깐 대기한다.

온라인 스키마 변경이 진행되는 동안 새로 유입된 DML 쿼리들에 의해 변경되는 데이터를 **Online alter log**라는 메모리 공간에 쌓아두었다가 스키마 변경 완료 시 로그 내용을 실제 테이블로 일괄 적용한다. Online alter log는 디스크가 아닌 메모리에만 생성되며, 메모리 공간의 크기는 `innodb_online_alter_log_max_size` 시스템 변수로 설정한다.(기본값 128MB)

4. 온라인 DDL 실패 케이스

- `ALTER TABLE` 명령이 장시간 실행되고 동시에 다른 커넥션에서 DML이 많이 실행되는 경우, 또는 Online alter log의 공간이 부족한 경우
- `ALTER TABLE` 명령이 실행되는 동안 이전 버전 테이블에는 문제가 안 되지만, 새로운 테이블 구조에 적합하지 않은 레코드가 `INSERT` 되거나 `UPDATE` 되는 경우
- 스키마 변경을 위해서 필요한 잠금 수준보다 낮은 잠금 옵션이 사용된 경우
- 온라인 스키마 변경 작업의 처음과 마지막 과정에서 잠금이 필요한데 이 잠금을 획득하지 못하고 타임 아웃이 발생하는 경우
- 온라인으로 인덱스를 생성하는 작업의 경우 정렬을 위해 `tmpdir` 시스템 변수에 설정된 디스크의 임시 디렉터리를 사용하는데 이 공간이 부족한 경우

> 온라인 스키마 변경에 필요한 잠금은 테이블 수준의 메타데이터 잠금이다. 메타데이터 잠금에 대한 타임 아웃은 `lock_wait_timeout` 시스템 변수에 의해 결정된다.(기본값 315,360,000초) `inoodb_lock_wait_timeout`과 별개

5. 온라인 DDL 진행 상황 모니터링

모든 `ALTER TABLE` 명령은 `performance_schema`를 통해 진행 상황을 모니터링할 수 있다.MySQL 서버의 `PERFORMACE_SCHEMA` 옵션을 활성화 한 후 `performance_schema` 옵션(Instrument와 Consumer 옵션)이 활성화되어야 한다.

```
SET GLOBAL PERFORMANCE_SCHEMA = ON;

UPDATE performance_schema.setup_instruments
SET ENABLED = 'YES',
    TIMED   = 'YES'
WHERE NAME LIKE 'stage/innodb/alter%';

UPDATE performance_schema.setup_consumers
SET ENABLED = 'YES'
WHERE NAME LIKE '%stages%';
```

> `Variable 'performance_schema' is a read only variable`
> 구성 파일을 수정해 변경해야 하는 듯

스키마 변경 작업의 진행 상황은 `performance_schema.events_stages_current` 테이블을 통해 확인 가능하며 실행 중인 스키마 변경 종류에 따라 기록 내용이 조금씩 달라진다.

```
SELECT EVENT_NAME, WORK_COMPLETED, WORK_ESTIMATED
FROM performance_schema.events_stages_current;
```

온라인 DDL은 단계별로 `EVENT_NAME` 컬럼의 값을 달리 보여준다.

`WORK_ESTIMATED`와 `WORK_COMPLETED` 컬럼 값을 비교하면 `ALTER TABLE`의 진행 상황을 예측할 수 있다.

7.2 데이터베이스 변경

MySQL에서는 스키마 == 데이터베이스

데이터베이스에 설정할 수 있는 옵션은 기본 문자 집합이나 콜레이션을 설정하는 정도 뿐

1. 데이터베이스 생성

```
CREATE DATABASE [IF NOT EXISTS] employees;
```

기본 문자 집합과 콜레이션으로 employees 데이터베이스 생성. 기본은 MySQL 서버의 character_set_server 시스템 변수에 정의된 문자 집합 사용

```
CREATE DATABASE [IF NOT EXISTS] employees CHARACTER SET utf8mb4;

CREATE DATABASE [IF NOT EXISTS] employees
    CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
```

특정 문자 집합과 콜레이션이 지정된 데이터베이스 생성

2. 데이터베이스 목록

```
SHOW DATABASES;

SHOW DATABASES LIKE '%emp%';
```

접속된 MySQL 서버가 가지고 있는 데이터베이스 목록 나열. 권한을 가지고 있는 데이터베이스 목록만 표시하고, SHOW DATABASES 권한이 있어야 명령 실행 가능

3. 데이터베이스 선택

```
USE employees;
```

기본 데이터베이스 선택 명령. SQL 문장에서 별도로 데이터베이스를 명시하지 않고 테이블 이름이나 프로시저 이름만 명시하면 현재 커넥션의 기본 데이터베이스에서 검색

4. 데이터베이스 속성 변경

```
ALTER DATABASE employees CHARACTER SET = euckr;
ALTER DATABASE employees CHARACTER SET = euckr COLLATE = euckr_korean_ci;
```

데이터베이스를 생성할 때 지정한 문자 집합이나 콜레이션을 변경

> `CHARACTER SET`: 데이터베이스, 테이블, 컬러멩서 사용 가능한 문자 집합
>
> `COLLATE`: 문자열 비교와 정렬 시 사용하는 규칙

5. 데이터베이스 삭제

```
DROP DATABASE [IF EXISTS] employees;
```

데이터베이스 삭제

7.3 테이블 스페이스 변경

MySQL 서버에는 전통적으로 테이블별로 전용의 테이블스페이스를 사용했다. InnoDB 스토리지 엔진의 시스템 테이블 스페이스(ibdata1 파일)만 제너럴 테이블스페이스(General Tablespace)를 사용했는데 제너럴 테이블스페이스는 여러 테이블의 데이터를 한꺼번에 저장하는 테이블스페이스를 의미

8.0 부터 사용자 테이블을 제너럴 테이블스페이스로 저장하는 기능이 추가되고 관리하는 DDL 명령들이 추가되었다.

제약 사항

- 파티션 테이블은 제너럴 테이블스페이스를 사용하지 못함
- 복제 소스와 레플리카 서버가 동일 호스트에서 실행되는 경우 `ADD DATAFILE` 문장은 사용 불가
- 테이블 암호화(TDE)는 테이블스페이스 단위로 설정됨
- 테이블 압축 가능 여부는 테이블스페이스의 블록 사이즈와 InnoDB 페이지 사이즈에 의해 결정됨
- 특정 테이블을 삭제(`DROP TABLE`)해도 디스크 공간이 운영체제로 반납되지 않음

장점

- 제너럴 테이블스페이스를 사용하면 파일 핸들러(Open file descriptor)를 최소화
- 테이블스페이스 관리에 필요한 메모리 공간을 최소화

테이블의 개수가 매우 많은 경우 장점이 유용하게 적용된다.

`innodb_file_per_table` 시스템 변수를 사용해 개별 테이블스페이스를 사용할지 여부를 제어할 수 있다.(기본값이 ON으로 개별 테이블스페이스를 사용)

7.4 테이블 변경

1. 테이블 생성

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tb_test
(
    member_id       BIGINT [UNSIGNED] [AUTO_INCREMENT],
    nickname        CHAR(20) [CHARACTER SET 'utf8'] [COLLATE 'utf8_general_ci']
    home_url        VARCHAR(200) [COLLATE 'latin1_general_cs'],
    birth_year      SMALLINT [(4)] [UNSIGNED] [ZEROFILL],
    member_point    INT [NOT NULL] [DEFAULT 0],
    registered_dttm DATETIME [NOT NULL],
    modified_ts     TIMESTAMP [NOT NULL] [DEFAULT CURRENT_TIMESTAMP],
    gender          ENUM ('Female','Male') [NOT NULL],
    hobby           SET ('Reading','Game','Sports'),
    PROFILE         TEXT [NOT NULL],
    session_data    BLOB,
    PRIMARY KEY (member_id),
    UNIQUE INDEX ux_nickname (nickname),
    INDEX ix_registereddttm (registered_dttm)
) ENGINE = INNODB;
```

`TEMPORARY` : 해당 데이터베이스 커넥션에서만 사용 가능한 임시 테이블을 생성

`IF NOT EXISTS` : 동일한 이름의 테이블이 존재하는 경우 에러를 무시

`ENGINE` : 테이블이 사용할 스토리지 엔진(기본이 InnoDB)

모든 컬럼은 초깃값을 설정하는 `DEFAULT` 와 `NULL` 저장 여부를 설정하는 `NULL` • `NOT NULL` 제약 명시 가능

문자열 타입은 타입 뒤에 반드시 컬럼에 최대한 저장할 수 있는 문자 수를 명시해야 한다.

숫자 타입은 선택적으로 길이를 설정할 수 있지만 실제 컬럼에 저장될 값의 길이가 아닌 단순히 값을 표시할 때 보여줄 길이를 지정하는 것이다. 또한 음수 양수 저장 여부를 `UNSIGNED` 키워드로 설정 가능. `ZEROFILL` 은 숫자 값의 왼쪽에 0을 패딩할지 결정하는 옵션

5.6 부터 `DATE` 와 `DATETIME` , `TIMESTAMP` 타입 모두 `DEFAULT` 명시 가능

2. 테이블 구조 조회

`SHOW CREATE TABLE` 명령과 `DESC` 명령

- `SHOW CREATE TABLE` : 테이블의 CREATE TABLE 문장을 표시. 최초 사용자가 실행한 내용을 그대로 보여주는 것은 아님. 메타 정보를 읽어 재작성해서 보여주는 것
- `DESC` : `DESCRIBE` 의 약어 형태 명령어로 `SHOW CREATE TABLE` 과 동일한 결과. 컬럼 정보를 표 형태로 표시. 인덱스 컬럼의 순서나 외래키, 테이블 자체 속성은 보여주지 않음

3. 테이블 구조 변경

`ALTER TABLE` 사용

MySQL :: MySQL 8.0 Reference Manual :: 15.1.9 ALTER TABLE Statement

ALTER TABLE changes the structure
of a table. For example, you can add or delete columns, create or
destroy indexes, change the type of existing columns, or rename

https://dev.mysql.com/doc/refman/8.0/en/alter-table.html

```
ALTER TABLE employees
    CONVERT TO CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci,
    ALGORITHM = INPLACE,
    LOCK = NONE;
```

테이블의 기본 문자 집합과 콜레이션을 변경하고 테이블의 모든 컬럼과 기존 데이터의 문자 셋과 콜레이션을 변경하는 쿼리

```
ALTER TABLE employees
    ENGINE = InnoDB,
    ALGORITHM = INPLACE,
    LOCK = NONE;
```

테이블의 스토리지 엔진을 변경하는 쿼리. 내부적인 테이블 저장소를 변경하는 것이라 항상 테이블의 모든 레코드를 복사하는 작업이 필요하다. `ALTER TABLE` 문장에 명시된 `ENGINE`이 기존과 동일하더라도 테이블의 데이터를 복사하는 작업은 실행되므로 주의.

엔진 변경 뿐 아니라 테이블 데이터 리빌드 용도로도 사용한다. (레코드의 삭제가 자주 발생해 데이터가 저장되지 않은 빈 공간인 Fragmentation을 제거해 디스크 사용 공간을 줄이는 역할을 한다. 동일한 기능으로 `OPTIMIZE TABLE` 명령어도 있다.

4. 테이블 명 변경

`RENAME TABLE` 사용

테이블 이름을 변경하거나 데이터베이스 이동

```
RENAME TABLE table1 TO table2;
RENAME TABLE db1.table1 TO db2.table2;
```

데이터베이스를 변경하는 경우 메타 정보뿐 아니라 테이블에 저장된 파일까지 다른 디렉토리로 이동해야 한다.

기존에 사용 중인 테이블과 다른 테이블을 교체해야 하는 경우 쿼리를 순서대로 실행하면 기존에 사용하던 테이블이 잠시동안 사라지는 문제가 발생하기 때문에 여러 테이블의 `RENAME` 명령을 하나의 문장으로 묶어 실행 가능하다.

```
RENAME TABLE batch TO batch_old,
    batch_new TO batch;
```

`RENAME TABLE`에 명시된 모든 테이블에 대해 잠금을 걸고 테이블 이름 변경 작업을 실행한다.

5. 테이블 상태 조회

`SHOW TABLE STATUS`

```
SHOW TABLE STATUS LIKE '';
```

> MySQL :: MySQL 8.0 Reference Manual :: 15.7.7.38 SHOW TABLE STATUS Statement
> SHOW TABLE STATUS works likes
> SHOW TABLES, but provides a lot
> of information about each non-TEMPORARY
> https://dev.mysql.com/doc/refman/8.0/en/show-table-status.html

이때 출력되는 레코드 건수나 평균 크기는 예측값이기 때문에 테이블이 너무 작거나 너무 크면 오차가 커질 수 있다.

> 쿼리 마지막에 `\G`를 붙이면 레코드의 컬럼을 라인당 하나씩만 표현하게 하는 옵션이다.

```
SELECT *
FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'employees'
  AND TABLE_NAME = 'employees';
```

`information_schema` 데이터베이스에는 MySQL 서버가 가진 스키마들에 대한 메타 정보를 가진 딕셔너리 테이블이 관리된다. 실제 존재하는 테이블이 아닌 서버가 시작되면서 데이터베이스와 테이블 등에 대한 다양한 메타 정보를 모아 메모리에 올려두고 사용자가 참조할 수 있는 테이블이다.

information_schema 데이터베이스 정보

- 데이터베이스 객체에 대한 메타 정보
- 테이블과 컬럼에 대한 간략한 통계 정보
- 전문 검색 디버깅을 위한 뷰(view)
- 압축 실행과 실패 횟수에 대한 집계

> MySQL :: MySQL Information Schema
> This is the MySQL Information Schema extract from the MySQL
> 5.7 Reference Manual.
> https://dev.mysql.com/doc/mysql-infoschema-excerpt/5.7/en/

6. 테이블 구조 복사

`SHOW CREATE TABLE` 명령을 이용하면 내용을 조금 변경해야 할 수 있고, `CREATE TABLE … AS SELECT … LIMIT 0`은 인덱스가 생성되지 않는 단점이 있다.

데이터는 복사하지 않고 테이블의 구조만 동일하게 복사하는 `CREATE TABLE … LIKE`를 사용하면 구조가 같은 테이블을 쉽게 생성할 수 있다.

> `CTAS` 구문은 리두 로그를 기록하지 않기 때문에 성능이 빠르다는 이야기는 아직 MySQL서버에는 해당하지 않는다. `CTAS` 구문은 `CREATE TABLE`과 `INSERT … SELECT …` 문장으로 나누어 실행하는 것과 성능 면에서 차이가 없다.

7. 테이블 삭제

`DROP TABLE`

테이블 삭제는 다른 테이블의 DML이나 쿼리를 직접 방해하지 않는다. 용량이 매우 큰 테이블을 삭제하는 작업은 부하가 크다. 그래서 테이블 크기가 크다면 서비스 도중에 삭제 작업은 수행하지 않는 게 좋다.

> MySQL 서버의 데이터 파일이 리눅스의 ext3 파일 시스템을 사용하는 경우 파일의 조각이 디스크의 이곳저곳에 분산되어 저장된다. 이로 인해 대용량 테이블 삭제 작업은 디스크 읽기 쓰기 작업을 상당히 많이 발생시킬 수 있다. 최근에 사용되는 리눅스는 대부분 ext4나 xfs를 사용하기 때문에 많이 줄었다.

InnoDB 스토리지 엔진의 어댑티브 해시 인덱스(Adaptive hash index)는 버퍼 풀의 각 페이지가 가진 레코드에 대한 해시 인덱스 기능을 제공한다. 어댑티브 해시 인덱스가 활성화되어 있는 경우 테이블이 삭제되면 어댑티브 해시 인덱스 정보도 모두 삭제해야 하는데 정보가 많다면 인덱스 삭제 작업으로 인해 부하가 높아질 수 있으므로 주의해야 한다. 테이블 스키마 변경에도 영향을 미칠 수 있다.

7.5 컬럼 변경

`ALTER TABLE`

1. 컬럼 추가

8.0 부터 테이블 컬럼 추가는 대부분 `INPLACE` 알고리즘을 사용하는 온라인 DDL 처리가 가능하다.

테이블의 가장 마지막 컬럼으로 추가하는 경우에는 `INSTANT` 알고리즘으로 즉시 처리된다. 그래서 테이블이 큰 경우 가능하다면 컬럼을 테이블의 마지막 컬럼으로 추가하는 것이 좋다.

2. 컬럼 삭제

컬럼을 삭제하는 작업은 항상 테이블의 리빌드가 필요하기 때문에 `INSTANT` 알고리즘을 사용할 수 없어 항상 `INPLACE` 알고리즘으로만 컬럼 삭제가 가능하다.

```
ALTER TABLE employees DROP COLUMN emp_telno, ALGORITHM = INPLACE, LOCK = NONE;
```

`COLUMN` 키워드는 입력하지 않아도 됨

3. 컬럼 이름 및 컬럼 타입 변경

```
ALTER TABLE salaries
    CHANGE to_date end_date DATE NOT NULL,
    ALGORITHM = INPLACE,
    LOCK = NONE;
```

컬럼 이름 변경 작업. `INPLACE` 알고리즘을 사용하지만 실제 데이터 리빌드 작업은 필요하지 않으므로 `INSTANT` 알고리즘과 같이 빠르게 작업이 완료된다.

```
ALTER TABLE salaries
    MODIFY salary VARCHAR(20),
    ALGORITHM = COPY,
    LOCK = SHARED;
```

컬럼 타입 변경. 완전히 다른 타입으로 변경하거나 `VARCHAR` 타입의 길이를 축소하는 경우 `COPY` 알고리즘을 사용해야 한다. `VARCHAR` 타입의 길이를 확장하는 경우에는 `INPLACE` 알고리즘을 사용하며 테이블의 리빌드가 필요할 수 있다.

`VARCHAR`나 `VARBINARY`타입의 경우 컬럼의 최대 허용 사이즈는 메타데이터에 저장되지만 실제 컬럼이 가지는 값의 길이는 데이터 레코드의 컬럼 헤더에 저장된다.컬럼값의 길이 저장용 공간은 컬럼의 값이 최대 가질 수 있는 바이트 수가 255 이하인 경우 1바이트만 사용하며 256 바이트 이상인 경우 2바이트를 사용한다. 동일한 값이어도 `VARCHAR(10)` 컬럼에 저장될 때보다 `VARCHAR(1000)` 컬럼에 저장될 때는 1바이트 더 사용

`INPLACE` 알고리즘으로 변경할 때 UTF8MB4 문자 셋을 사용하는 경우, 크기가 10에서 64로 변경하면 테이블 리빌드가 필요

7.6 인덱스 변경

8.0 부터는 대부분 인덱스 변경 작업이 온라인 DDL로 처리가 가능하도록 개선되었다.

> 전문 검색 인덱스와 공간 검색 인덱스를 제외하면 나머지 인덱스는 모두 B-Tree 자료 구조를 사용한다. `USING HASH` 절은 MySQL Cluster (NDB)를 위한 옵션이지 MySQL 서버의 InnoDB나 MyISAM 스토리지 엔진을 위한 옵션이 아니다.

1. 인덱스 추가

`ALTER TABLE ADD INDEX`

```
ALTER TABLE employees
    ADD PRIMARY KEY (emp_no),
    ALGORITHM = INPLACE,
    LOCK = NONE;

ALTER TABLE employees
    ADD UNIQUE INDEX ux_empno (emp_no),
    ALGORITHM = INPLACE,
    LOCK = NONE;

ALTER TABLE employees
    ADD INDEX ix_lastname (last_name),
    ALGORITHM = INPLACE,
    LOCK = NONE;
```

```
ALTER TABLE employees
    ADD FULLTEXT INDEX fx_firsname_lastname (first_name, last_name),
    ALGORITHM = INPLACE,
    LOCK = SHARED;

ALTER TABLE employees
    ADD SPATIAL INDEX fx_loc (last_location),
    ALGORITHM = INPLACE,
    LOCK = SHARED;
```

전문 검색 인덱스와 공간 검색 인덱스는 `INPLACE` 알고리즘으로 인덱스 생성이 가능하지만 `SHARED` 잠금이 필요하다. 나머지 B-Tree 자료 구조를 사용하는 인덱스 추가는 PK라고 하더라도 `INPLACE` 알고리즘에 잠금 없이 온라인으로 인덱스 생성이 가능

2. 인덱스 조회

`SHOW INDEXES`, `SHOW CREATE TABLE`

> MySQL :: MySQL 8.0 Reference Manual :: 15.7.7.22 SHOW INDEX Statement
> SHOW INDEX returns table index
> information. The format resembles that of the
> SQLStatistics call in ODBC. This statement
> https://dev.mysql.com/doc/refman/8.0/en/show-index.html

- `Key_name`: 인덱스 이름
- `Seq_in_index`: 인덱스에서 해당 컬럼의 위치
- `Cardinality`: 인덱스에서 해당 컬럼까지의 유니크한 값의 개수

3. 인덱스 이름 변경

5.7 부터는 `ALTER TABLE … RENAME INDEX … TO …`를 사용해 인덱스 이름 변경 가능

인덱스 이름을 변경하는 작업은 `INPLACE` 알고리즘을 사용하지만 실제 테이블 리빌드를 필요로 하지 않기 때문에 응용 프로그램에서 힌트로 해당 인덱스의 이름을 사용 중이어도 짧은 시간에 인덱스를 교체할 수 있다.

4. 인덱스 가시성 변경

인덱스 삭제는 `ALTER TABLE DROP INDEX` 명령으로 즉시 완료되지만 한 번 삭제된 인덱스를 새로 생성하는 것은 매우 많은 시간이 걸릴 수 있다.

8.0 부터는 쿼리를 실행할 때 해당 인덱스를 사용할 수 있게 할지 말지 결정하는 인덱스 가시성 제어 기능이 추가되었다.

```
ALTER TABLE employees ALTER INDEX ix_firstname INVISIBLE;
```

MySQL 옵티마이저는 상태가 `INVISIBLE`인 인덱스는 없는 것으로 간주하고 실행 계획을 수립한다. 해당 명령은 메타데이터만 변경하기 때문에 온라인 DDL로 실행되는지 여부를 고려하지 않아도 된다.

최초 인덱스 생성 시에도 가시성을 설정할 수 있다.

> `optimizer_switch` 시스템 변수에 `use_invisible_indexes` 옵션이 ON으로 설정된 경우 옵티마이저는 쿼리가 `INVISIBLE` 상태의 인덱스도 사용할 수 있게 한다.(기본값은 OFF)

4. 인덱스 삭제

`ALTER TABLE … DROP INDEX …`

인덱스 삭제는 일반적으로 매우 빨리 처리된다. 세컨더리 인덱스 삭제 작업은 `INPLACE` 알고리즘을 사용하지만 실제 테이블 리빌드를 필요로 하지는 않는다. PK 삭제 작업은 모든 세컨더리 인덱스의 리프 노드에 저장된 PK 값을 삭제해야 하기 때문에 임시 테이블로 레코드를 복사해 테이블을 재구축해야 한다.

```
ALTER TABLE employees
    DROP PRIMARY KEY,
    ALGORITHM = COPY,
    LOCK = SHARED;

ALTER TABLE employees
    DROP INDEX ux_empno,
    ALGORITHM = INPLACE,
    LOCK = NONE;

ALTER TABLE employees
    DROP INDEX fx_loc,
    ALGORITHM = INPLACE,
    LOCK = NONE;
```

PK 삭제는 `COPY` 알고리즘, `SHARED` 락 사용

7.7 테이블 변경 묶음 실행

여러 인덱스 생성을 개별로 실행하는 것 보다 하나로 묶어서 처리하는 것이 효율적이다. 인덱스를 생성할 때마다 테이블 레코드를 풀 스캔 해서 인덱스를 생성하게 된다. 하나로 묶어 처리하면 한 번만 풀 스캔을 하고 여러 인덱스를 생성할 수 있다.

서로 다른 알고리즘을 사용하는 스키마 변경 작업들은 굳이 묶어서 실행할 필요는 없다. 가능하면 동일한 알고리즘을 사용하는 스키마 변경 작업은 묶어서 실행하자.

7.8 프로세스 조회 및 강제 종료

`SHOW PROCESSLIST` 명령어를 사용해 MySQL 서버에 접속된 사용자 목록이나 각 클라이언트 사용자가 현재 어떤 쿼리를 실행하고 있는지 확인할 수 있다.

접속된 클라이언트의 요청을 처리하는 스레드 수만큼 레코드가 표시

- `Id`: MySQL 서버의 스레드 아이디이며, 쿼리나 커넥션을 강제 종료할 때는 이 값을 식별자로 사용
- `User`: 클라이언트가 인증에 사용한 사용자 계정

- `Host`: 클라이언트의 호스트명이나 IP 주소
- `db`: 클라이언트가 기본으로 사용하는 데이터베이스 이름
- `Command`: 해당 스레드가 현재 처리 중인 작업
- `Time`: `Command` 컬럼에 표시된 작업이 얼마나 실행되었는지 시간
- `State`: `Command` 컬럼에 표시된 작업이 해당 스레드가 처리하고 있는 작업의 큰 분류라면 State는 소분류 작업 내용
- `Info`: 해당 스레드가 실행 중인 쿼리 문장

`Command` 컬럼 값이 `Query` 이면서 `Time`이 상당히 큰 값을 가지면 쿼리가 상당히 장시간 실행되고 있는 것이다.

`State` 컬럼의 내용이 중요하다. 대표적으로 `Copying …`, `Sorting …` 으로 시작하는 값들이 표시될 때는 주의 깊게 살펴봐야 한다.

> MySQL :: MySQL 8.0 Reference Manual :: 10.14 Examining Server Thread (Process) Information
> To ascertain what your MySQL server is doing, it can be helpful to
> examine the process list, which indicates the operations currently
> being performed by the set of threads executing within the server.
> https://dev.mysql.com/doc/refman/8.0/en/thread-information.html

```
KILL QUERY 4228;
```

Id가 4228인 스레드가 실행 중인 쿼리 종료(커넥션은 유지)

```
KILL 4228;
```

Id가 4228인 스레드가 실행하고 있는 쿼리뿐만 아니라 해당 커넥션까지 강제 종료

7.9 활성 트랜잭션 조회

트랜잭션이 오랜 시간 완료되지 않고 활성 상태로 남아있는 것도 MySQL 서버의 성능에 영향을 미칠 수 있다.

`information_schema.innodb_trx` 테이블을 통해 트랜잭션 목록 확인

`trx_rows_modified` 컬럼과 `trx_rows_locked` 컬럼의 값을 참조해보면 변경 레코드 개수와 잠금 레코드 개수를 볼 수 있다. 어떤 레코드를 잠그고 있는지는 `performance_schema.data_locks` 테이블을 참조하면 된다.

쿼리만 강제 종료하면 커넥션이나 트랜잭션은 여전히 활성 상태로 남는다. application에서 쿼리의 에러를 감지하여 트랜잭션을 롤백하게 되어 있다면 쿼리만 종료하면 된다. 하지만 그렇지 않으면 커넥션 자체를 종료시키는 것이 안정적일 수 있다.

8. 쿼리 성능 테스트

8.1 쿼리 성능에 영향을 미치는 요소

여러 종류의 버퍼나 캐시가 쿼리 실행 성능에 큰 영향을 미치는 요소이다.

1. 운영체제 캐시

MySQL 서버는 운영체제 파일 시스템 관련 기능(system call)을 이용해 데이터 파일을 읽어온다. 보통 운영체제는 한 번 읽은 데이터는 운영체제가 관리하는 캐시 영역에 보관하고, 이후에 동일한 데이터가 요청되면 캐시 내용을 바로 MySQL 서버로 반환한다. InnoDB 스토리지 엔진은 일반적으로 파일 시스템의 캐시나 버퍼를 거치지 않는 Direct I/O를 사용하므로 운영체제의 캐시에 큰 영향을 받지 않지만, MyISAM 스토리지 엔진은 영향을 받는다.

그래서 쿼리 성능을 테스트하려면 운영체제의 캐시 삭제 명령을 실행하고 테스트하는 것이 좋다.

2. MySQL 서버의 버퍼 풀(InnoDB 버퍼 풀과 MyISAM의 키 캐시)

MySQL 서버에서도 데이터 파일의 내용을 페이지(또는 블록) 단위로 캐시하는 기능을 제공한다.

MySQL의 캐시

- InnoDB
  - 버퍼 풀
  - 인덱스 페이지, 데이터 페이지 캐시
  - 쓰기 작업을 위한 버퍼링 작업까지 겸해서 처리
- MyISAM
  - 키 캐시
  - 읽기를 위한 캐시 역할
  - 제한적으로 인덱스 변경만을 위한 버퍼 역할 수행

MySQL 서버가 한 번 시작되면 InnoDB의 버퍼 풀과 MyISAM 키 캐시 내용을 강제로 퍼지(Purge, 삭제)할 수 있는 방법이 없다. 서버 재시작이 방법!