글로벌 룰

스터디 시간 : 매주 일요일 20:00

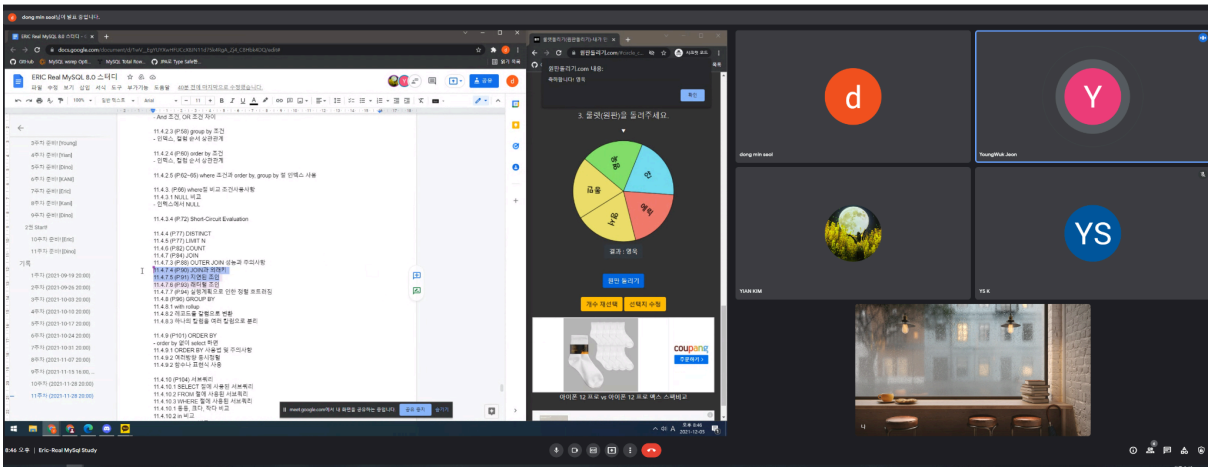
스터디 장소 : 구글 밋 (https://meet.google.com/yag-cjhg-vom?authuser=0)

책: Real MySQL 8.0

- 1권 https://book.naver.com/bookdb/book_detail.nhn?bid=20877661
- 2권 https://book.naver.com/bookdb/book_detail.nhn?bid=20877663

1. 매주 진행자는 정해진 분량의 책 내용중 질문거리를 준비해 온다.
2. 스터디 시간에는 진행자가 질문을 가지고 스터디원에게 질문을 한다.
3. 부족하거나 모르는 부분이 있다면 찾아보고 기록한다.
4. 질문할 거리가 있다면 [기록]탭에 미리 질문을 남겨보자!
5. 최소 스터디 분량은 50페이지.

정해진 파트를 돌림판을 돌려서 나온 스터디원이 설명해주어야 합니다!



스터디 자료

초기 데이터 및 스키마세팅된 docker mysql 이미지를 만들어두었습니다.

yyy9942/eric-mysql 받아 사용하시면 되며 루트계정의 비밀번호는 eric입니다.

스터디 멤버

- Eric
- Young
- Yian
- Kani
- Dino

발걸음

1권 Start!

1주차 준비!

- 글로벌 스터디 룰 정의
- 스터디 사전 미팅
- 책을 준비해주세요!

2주차 준비! [Eric]

- 책 1~75p (1, 2, 3장)
- 예제 DB 생성은 https://github.com/wikibook/realmysql80 을 참고하여 생성 가능
- MySQL 설치해주세요! [Docker | Cloud | Local]

3주차 준비! [Young]

- 책 76p ~153p (4장)

4주차 준비! [Yian]

- 책 154 ~ 212
    - 5장 트랜잭션과 잠금
    - 6장 데이터 압축
    - 7장 데이터 암호화

5주차 준비! [Dino]

- 책 213 ~ 281

        - 8장 인덱스

6주차 준비! [KANI]

- 책 282 ~ 345
    - 9장 옵티마이저와 힌트

7주차 준비! [Eric]

- 책 345 ~ 392
    - 9장 옵티마이저와 힌트 끝까지

8주차 준비! [Kani]

- 책 393 ~ 447
    - 10장 실행계획 - 1

9주차 준비! [Dino]

- 책 448 ~ 490 (끝까지)
    - 10장 실행계획 - 2

2권 Start!

10주차 준비! [Eric]

- 책 1 ~ 52까지

11주차 준비! [Dino]

- 책 52 ~ 148

12주차 준비! [Young]

- 148 ~ 210 - 11장 쿼리작성 및 최적화 끝

13주차 준비! [KANI]

- 211 ~ 261 12장 확장 검색

14주차 준비! [YIAN]

- 262 ~ 354 13장 파티션, 14장 스토어드 프로그램

15주차 준비! [Eric]

- 355 ~ 427 15장 데이터 타입

16주차 준비! [KANI]

- 428 ~ 554 16장 복제

17주차 준비! [Eric]

- 555 ~ 645 17장 InnoDB 클러스터

18주차 준비! [Dino]

- 책거리 합니다! 강남역 커피빈 3층 16시에 만나요~
- 646 ~ 끝까지

기록

1주차 (2021-09-19 20:00)

- 스터디 진행 방식 설명
- 매주 진행자를 뽑는 방식 지정
- 진행자가 해야하는 일 리스트업
    - [ ] 책 범위에 맞는 질문을 준비해야 함
    - [ ] 질문 리스트로 질문해야함
    - [ ] 진행 스터디 범위를 지정해야함
- 참여자의 사전지식은 어느정도인가요?
    - [ ] Eric: DML을 다룰 수 있으며 가-끔 쿼리 실행계획을 분석하기도 하는 초급 레벨. DDL 및 권한관련해서는 모름. MVCC같은거에 관심 많음
    - [ ] YIAN: DDL, DML을 다룰 수 있으며, 테이블 정규화, 테이블 조인, 인덱스 설정과 쿼리 플랜 분석 가능한 초급 수준
    - [ ] zugum: 기본적인 DDL, DML등을 다룰 수 있으며, 간단한 인덱스 힌트를 걸거나 슬로우 쿼리를 확인하는 정도
    - [ ] KANI: SELECT 쿼리 전에 EXPLAIN을 통해서 쿼리 실행 계획을 파악한 후 어떤 인덱스를 타는지 확인할 수 있는 수준. MySQL의 자료 구조 중 극히 일부 내용만 알고 있음. MySQL이 Repeatable Read라는 비교적 낮은 수준의 Isolation Level을 가지고 있지만 MVCC(Multiversion Concurrency Control)라는 개념을 통해서 어느 정도 극복하고 있다고 대충 알고만 있는데 확실히 알고 싶습니다.
    - [ ] young: 기본적인 ddl dml dcl 을 다뤄봄. mvcc등에 대해서 어느정도 알고있음. 간단하게 쿼리 실행계획 보는정도고 전략적으로 튜닝해본 경험은 많지 않음(인덱스 걸리는것만 신경씀…)

- 진행 예제는 2주차에 에릭이 진행하며 보여드림ㄷㄷ

2주차 (2021-09-26 20:00)

2장. 설치와 설정

- 왜 MySQL을 써야할까?
    - 오라클은 비싸기 때문에

        - NoSQL은 용도에 맞지 않음. 다른 DBMS는 안정성이 떨어짐
- 세가지 명령어의 차이점은?
    - mysql -uroot -p --host=localhost --socket=/tmp/mysql.sock
    - mysql -uroot -p --host=127.0.0.1 --port=3306
    - mysql -uroot -p
- MySQL 서버 업그레이드에 대해 설명해보자
    - 인플레이스 업그레이드
        - 데이터를 두고 MySQL 버전을 업그레이드
        - 한번에 올릴 수 있는 버전이 제약됨
    - 논리적 업그레이드
        - 덤프떠서 새로운 버전 DB에 다시 덤프
- MySQL 5.7 -> 8.0 업그레이드에서 어떤 점이 바뀌었을까? (P.33)
- mysql의 설정파일인 my.cnf파일의 탐색 디렉터리 우선순위는 어떻게 알 수 있을까? (p36)
    - mysqld --verbose --help
- 시스템 변수 (38p)
    - 글로벌 변수와 세션 변수
    - 정적 변수와 동적 변수
        - SET PERSIST
        - RESET PERSIST
        - my.cnf

3장. 사용자 및 권한

- 사용자 식별에 사용되는 것은?
- 시스템계정과 일반계정의 차이점
- 계정 생성 옵션
    - IDENTIFIED WITH
    - REQUIRE
    - PASSWORD EXPIRE
    - PASSWORD HISTORY
    - PASSWORD REUSE INTERVAL
    - PASSWORD REQUIRE
    - ACCOUNT LOCK / UNLOCK

- 비밀번호 정책 설정
    - validate로 글자조합 지정 또는 금칙어 설정
- 이중 비밀번호? (RETAIN CURRENT PASSWORD)
- 글로벌권한과 객체권한
- 정적 권한과 동적 권한
- 컬럼단위 권한을 잘 사용하지 않는 이유?
- 권한을 유저에 부여한 후 활성화하는 방법
- 계정과 권한의 차이?

3주차 (2021-10-03 20:00)

- MySQL의 전체 구조는?
    - MySQL 엔진
    - 스토리지 엔진
    - 핸들러 API
- MySQL의 스레딩 구조?
    - 포그라운드 스레드
    - 백그라운드 스레드
- 메모리 구조
    - 글로벌 메모리 영역
    - 로컬 메모리 영역
- 플러그인 모델과 컴포넌트 모델 ?
- 쿼리 실행 구조
    - 쿼리 파서
    - 전처리기
    - 옵티마이저
    - 실행 엔진
    - 핸들러
- InnoDB 스토리지 엔진 아키텍처
    - 클러스터링 인덱스
    - MVCC (Multi Version Concurrency Control)

        - 잠금 없는 일관된 읽기 (Non-Locking Consistent Read)로 인해 생길수 있는 문제점?
        - 자동 데드락 감지
- InnoDB 버퍼풀
    - 크기 설정
    - 버퍼풀의 페이지 크기 조각을 관리하기 위한 3가지 자료구조
        - LRU 리스트
        - 플러시 리스트
        - 프리 리스트
- InnoDB 스토리지 엔진에서 데이터를 찾는 과정
- 버퍼풀과 리두로그
- Double Write Buffer
- 언두 로그
- 체인지 버퍼
- 리두 로그
- 어댑티브 해시 인덱스
- MySQL 로그
    - 에러 로그
    - 슬로우 쿼리 로그

4주차 (2021-10-10 20:00)

5장 트랜잭션과 잠금

- 잠금과 트랜잭션, 트랜잭션의 격리수준
- MySQL에서 트랜잭션이란?
- 트랜잭션 사용 시 주의사항
- MySQL 엔진의 잠금 (Lock) 4가지가 각각의 특징과 어떤 경우에 사용되는지
    - 글로벌 락
    - 테이블 락
    - 네임드 락
    - 메타데이터 락

	X	IX	S	IS
X	Conflict	Conflict	Conflict	Conflict
IX	Conflict	Compatible	Conflict	Compatible
S	Conflict	Conflict	Compatible	Compatible
IS	Conflict	Compatible	Compatible	Compatible

- InnoDB 스토리지 엔진 잠금 (Lock) 각각의 특징
    - 레코드 락
    - 갭 락
    - 넥스트 키 락
    - 자동 증가 락
- 인덱스와 잠금의 특징
- 레코드 수준의 잠금 확인 및 해제 특징, 잠금과 잠금 대기 순서 확인 방법
- MySQL의 격리 수준 각각의 특징과 조회 결과
    - READ UNCOMMITTED
    - READ COMMITTED
    - REPEATABLE READ
    - SERIALIZABLE

6장 데이터 압축

- 페이지 압축 개념, 특징, 문제점
- 테이블 압축
- KEY_BLOCK_SIZE 결정

7장 데이터 암호화

- MySQL 서버의 데이터 암호화
    - 2단계 키 관리
    - 암호화와 성능, 복제
- 응용 프로그램 암호화와의 비교
- 언두 로그 및 리두 로그 암호화
- 바이너리 로그 암호화 키 관리와 변경 방법

5주차 (2021-10-17 20:00)

8장. 인덱스 (p214)

8.1. 디스크 읽기방식

1. 랜덤 I/O 와 순차 I/ O

8.2. 인덱스란

1. 프라이머리 키 & 세컨더리 인덱스란?

- 프라이머리키는 레코드를 대표하는 칼럼의 값으로 만들어진 인덱스
- 프라이머리키를 제외한 모든 인덱스를 세컨더리 인덱스라고 부른다

2. 유니크 인덱스의 장점

- 항상 1개만 존재한다는것을 옵티마이저에게 보장해주기 때문에 쿼리 실행계획에서 이점이 있다.

3. B-Tree vs Hash 차이

- 해시인덱스는 빠르다는 장점이 있지만, 범위검색 또는 일부일치검색을 할 때 사용할 수 없다.

8.3. B-Tree 인덱스

8.3.1. B-Tree 구조 및 특성

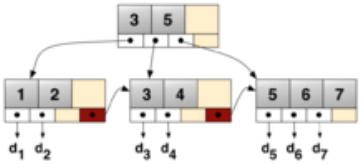
1. B-TREE 구조

- 루트노드 - 브랜치노드 - 리프노드 - 데이터파일 구조로 이루어져있음
- 리프노드에는 실제 데이터를 찾아가기위한 PK값이 있고, 이 PK를통해 실제 레코드를 가져오는 구조

2. B-TREE 상에서 MyISAM, InnoDB 세컨터리 인덱스 차이

- MyISAM은 세컨더리 인덱스가 InnoDB와 저장되는 방식이 다름

참고- B+트리



8.3.2. B-Tree 인덱스 CRUD

1. 인덱스 키 추가, 삭제, 변경, 검색

- 입력, 수정, 삭제작업은 unique index를 제외하고는 change buffer를 통한 lazy 처리
- 수정은 (삭제 -> 입력) 형태로 이루어져 있음
- 검색에서는 100% 일치 또는 값의 앞부분만 일치하는 경우에 사용 가능
  (like %text는 안됨)

8.3.3. B-Tree 인덱스 사용 영향

1. 인덱스 키 값의 크기와 인덱스 깊이

- 인덱스의 키 값이 커진다면?

2. 선택도, 기수성

- 기수성이 높거나, 낮으면?

8.3.2. B-Tree 인덱스를 통한 데이터 읽기

1. 인덱스 레인지 스캔

- 인덱스 레인지 스캔 단계

- 커버링 인덱스란

2. 인덱스 풀 스캔

- 인덱스 풀 스캔 vs 테이블 풀 스캔

3. 루스 인덱스 스캔 & 인덱스 스킵 스캔

- MySQL에서 루스 인덱스 스캔과 인덱스 스킵 스캔 차이

=> 루스 인덱스 스캔은 GROUP BY에서만 사용 가능, 하지만 인덱스 스킵은 WHERE절에도 사용이 가능하다. 단, 커버링인덱스로 검색을 했을 때에만 사용이 가능하다.

4. 다중 칼럼 인덱스

- 두 개 이상의 컬럼으로 구성된 인덱스
- 인덱스 순차적으로 정렬하며, 하위 컬럼은 상위컬럼에 종속되어 정렬된다.

8.3.3. B-Tree 인덱스 정렬 & 스캔 & 가용성

1. 인덱스 정렬

- MySQL 5.7 & 8.0 오름차순, 내림차순 설정차이

2. 인덱스 스캔 방향

- InnoDB에서 인덱스 역순 스캔이 인덱스 정순스캔에 비해 느린 이유

=> 레코드는 단방향 연결구조이며 페이지잠금이 정순스캔에 적합한 구조이기 때문

3. 인덱스 가용성

8.4. R-TREE 인덱스

1. MBR 이란

8.5. 전문검색 인덱스

1. MeCab과 n-gram

2. 전문 검색 인덱스 가용성 조건

8.6 함수 기반 인덱스

1. 가상칼럼 인덱스

2. 함수기반 인덱스

8.7. 멀티 벨류 인덱스

8.8. 클러스터링 인덱스

1. 클러스터링 인덱스란

2. 장단점 및 주의 사항

8.9 유니크 인덱스

1. 인덱스 읽기 & 쓰기

2. 사용시 주의사항

8.10 외래키

1. 외래키 특성

6주차 (2021-10-24 20:00)

9장. 옵티마이저와 힌트 part1(p283 ~ p345)

- 쿼리 실행 절차
    1. SQL 파싱 : SQL 파서 모듈로 처리
    2. 테이블 및 인덱스 선택 : 옵티마이저에서 처리
    3. 데이터 읽기 : MySQL 엔진과 스토리지 엔진이 동시에 참여해서 처리
- 옵티마이저의 종류
    - 규칙 기반 최적화 : 옵티마이저의 내장 우선순위를 사용
    - 비용 기반 최적화 : 통계 정보를 활용

9.2. 기본 데이터 처리

1. 풀 테이블 스캔과 풀 인덱스 스캔(p285)
    - 풀 테이블 스캔이 사용되는 경우
        - 테이블의 내용이 적을 경우
        - 필터링이 마땅치 않은 경우
        - 옵티마이저가 판단해서 일치 레코드가 많은 경우
    - 풀 인덱스 스캔이 사용되는 경우
        - SELECT 조건이 인덱스가 있는 경우
2. 병렬 처리(p287)
    - WHERE 조건이 없이 단순히 테이블의 전체 건수를 가져오는 쿼리만 병렬로 가능

9.2.3. ORDER BY 처리(Using filesort)

- 인덱스를 이용한 정렬
    - 삽입/수정/삭제 쿼리가 실행될 때 이미 인덱스가 정렬되어 있으므로 빠르다.
    - 인덱스를 위한 추가 공간이 필요하다. (디스크 공간, InnoDB 버퍼풀)
- Filesort를 이용한 정렬

- 인덱스를 이용한 방식의 단점이 장점.
- 정렬 작업이 쿼리 실행 시 처리되므로 레코드 대상 건수가 많아질수록 응답 속도가 느림

1. 소트 버퍼(p289)
2. 정렬 알고리즘(p291)
    ■ 정렬 방식
        - <sort_key, rowid>
        - <sort_key, additional_fields>
        - <sort_key, packed_additional_fields>
    ■ 싱글 패스 정렬 방식
    ■ 투 패스 정렬 방식
3. 정렬 처리 방법(p295)
    ■ 인덱스를 이용한 정렬
    ■ 조인의 드라이빙 테이블만 정렬
    ■ 임시 테이블을 이용한 정렬
    ■ 성능 비교
        - 스트리밍 방식
        - 버퍼링 방식 *jdbc 라이브러리
4. 정렬 관련 상태 변수(p304)

9.2.4. GROUP BY 처리

1. 인덱스 스캔을 이용하는 GROUP BY(타이트한 인덱스 스캔)(p306)
2. 루스 인덱스 스캔을 이용하는 GROUP BY(p307)
3. 임시 테이블을 사용하는 GROUP BY(p308)

9.2.5 DISTINCT 처리

1. SELECT DISTINCT ...(p310)
2. 집합 함수와 함께 사용된 DISTINCT(p311)

9.2.6 내부 임시 테이블 활용

1. 내부 임시 테이블의 의미(p314)
2. 메모리 임시 테이블과 디스크 임시 테이블(p316)
3. 임시 테이블이 필요한 쿼리(p316)
4. 임시 테이블 관련 상태 변수(p317)

9.3 고급 최적화

9.3.1 옵티마이저 스위치 옵션

1. MRR과 배치 키 엑세스(mrr & batched_key_access)(p320)
2. 블록 네스티드 루프 조인(block_nested_loop)(p320)
3. 인덱스 컨디션 푸시다운(index_condition_pushdown)(p324)
4. 인덱스 확장(use_index_extensions)(p327)
5. 인덱스 머지(index_merge)(p329)
6. 인덱스 머지 - 교집합(index_merge_intersection)(p329)
7. 인덱스 머지 - 합집합(index_merge_union)(p332)
8. 인덱스 머지 - 정렬 후 합집합(index_merge_sort_union)(p334)
9. 세미 조인(semijoin)(p335)
10. 테이블 풀-아웃(Table Pull-out)(p337)
11. 퍼스트 매치(firstmatch)(p339)
12. 루스 스캔(loosescan)(p341)
13. 구체화(Materialization)(p343)
14. 중복 제거(Duplicated Weed-out)(p345) => To be continue...

7주차 (2021-10-31 20:00)

9.3.1.14 중복제거 (Duplicated Weed-out)

- 중복제거 알고리즘은 세미조인을 어떻게 변환시켜 처리할까?
    - 두 테이블을 조인한다
    - 임시테이블에 결과를 저장한다
    - 중복된 결과를 제거한다 (Group by)

9.3.1.15 컨디션 팬아웃 (condition fanout filter)

특정 컬럼의 분포도를 더 정밀하게 예측하여 옵티마이저의 최적화를 도와준다.

- 컨디션 팬아웃 필터가 조건을 만족하는 레코드를 계산하는 방법은?
    - Range optimizer 예측
    - 히스토그램 예측
    - 인덱스 통계 예측
    - 추측기반 예측

9.3.1.16 파생 테이블 머지 (derived_merge)

- AS-IS 파생테이블 쿼리 처리방법은?

- 파생테이블을 머지할 수 없는 조건은?
    - 집계함수, GROUP BY, LIMIT, UNION, SELECT절에 사용되는 서브쿼리 등

9.3.1.17 인비저블 인덱스(use_invisible_indexes)

- 인덱스를 삭제하지 않고도 인덱스를 비활성화할 수 있는 옵션. 인덱스 삭제하는것도 비용이기 때문에 이런 옵션을 사용함

9.3.1.18 스킵 스캔(skip_scan)

- (A, B, C)인덱스에서 (B) 인덱스를 조회할 때에도 제한적으로 인덱스를 사용하는 스캔
- (A, B, C) 인덱스에서 스킵스캔을 사용할 수 있는 조건은?
    - (B, C)로스캔하는 경우 + 유니크값이 적어야 함
- 인덱스 스킵 스캔이 효율적으로 동작하는 조건은?
    - 유니크값이 적어야 함 (카디널리티가 낮아야 함)

9.3.1.19 해시 조인(hash_join)

빠르다고 해서 기대하고있었지만, 생각보다 아직은 효율적이지 않음. (개선이 필요함) 네스티드 루프조인이 비효율적이거나, 사용하기 어려운 경우에만 보조적으로 사용되는 조인

빌드단계 - 레코드 건수가 적은 테이블을 메모리에 해시테이블로 생성,프로브단계 - 나머지 테이블의 레코드를 읽어서 해시테이블의 일치 레코드를 찾음

- 네스티드 루프조인과의 차이점은?
    - 해시조인은 첫 레코드를 찾는데는 시간이 많이 걸리지만, 최종레코드를 찾을때까지 시간이 짧게 걸림
    - 네스티드 루프조인은 첫 레코드를 빨리 찾지만 최종레코드를 찾을때는 시간이 걸림
- 해시조인은 항상 빠를까?
    - 응답속도는 네스티드 루프조인이 빠름
    - 스루풋은 해시조인이 빠를'수'도 있음
- 옵티마이저가 해시조인을 사용하는 조건은?
    - 네스티드 루프조인을 사용하는것이 비효율적일 때
        - 조인 대상 테이블의 레코드가 적을 때
        - 조인을 인덱스 없는 컬럼으로 할 때
- 해시조인의 동작 방식은?
    - 빌드단계 [레코드 건수가 적은 테이블을 메모리에 해시테이블로 생성]
    - 프로브단계 [나머지 테이블의 레코드를 읽어서 해시테이블의 일치 레코드를 찾음]
    - 레코드건수가 많으면 청크단위로 쪼개서 처리함

9.3.1.20 인덱스 정렬 선호 (prefer_ordering_index)

- 인덱스 정렬 선호가 등장한 이유는?

9.3.2 조인 최적화 알고리즘

9.3.2.1 Exhaustive 검색 알고리즘

- 문제점
    - 가능한 모든 조합을 다 조합하기 때문에, 테이블이 늘어날수록 팩토리얼로 늘어난다

9.3.2.2 Greedy 검색 알고리즘

- 동작방식은?
    - 조인조합을 생성한 후, 최소비용 실행계획을 산정
    - 산정한 조합의 첫 테이블을 첫 테이블로 선정
    - 첫 테이블을 기준으로 또 조인조합 생성
    - ...반복

9.4 쿼리힌트

9.4.1 인덱스 힌트

- STRAIGHT_JOIN
- USE INDEX
- FORCE INDEX
- IGNORE INDEX
- SQL_CALC_FOUND_ROWS

9.4.2 옵티마이저 힌트

- 힌트 작성법
- MAX_EXECUTION_TIME
- SET_VAR
- SEMIJOIN & NO_SEMIJOIN
- SUBQUERY
- BNL & NO_BNL & HASHJOIN & NO_HASHJOIN
- JOIN_FIXED_ORDER & JOIN_ORDER & JOIN_PREFIX & JOIN_SUFFIX
- MERGE & NO_MERGE
- INDEX_MERGE & NO_INDEX_MERGE
- NO_ICP
- SKIP_SCAN & NO_SKIP_SCAN
- INDEX & NO_INDEX

8주차 (2021-11-07 20:00)

10장. 실행 계획 part1(p393 ~ p449)
10.1. 통계 정보(p394)
- 히스토그램(Histogram) 정보 도입
10.1.1. 테이블 및 인덱스 통계 정보
1. MySQL 서버의 통계 정보(p395)
- innodb 테이블을 통한 통계 관리

10.1.2. 히스토그램
1. 히스토그램 정보 수집 및 삭제(p399)
- Singleton Histogram
- Equi-Height Histogram
2. 히스토그램의 용도(p404)
3. 히스토그램과 인덱스(p407)
- Index Dive

10.1.3. 코스트 모델(Cost Model)(p408)
- server_cost
- engine_cost

10.2. 실행 계획 확인
10.2.1. 실행 계획 출력 포맷(p412)
- FORMAT 옵션
10.2.2. 쿼리의 실행 시간 확인(p414)
- EXPLAIN ANALYZE 명령

10.3. 실행 계획 분석
10.3.1. id 칼럼(p417)

10.3.2. select_type 칼럼(p420)
1. SIMPLE(p420)
2. PRIMARY(p420)
3. UNION(p420)
4. DEPENDENT UNION(p421)

5. UNION RESULT(p422)
6. SUBQUERY(p423)
7. DEPENDENT SUBQUERY(p424)
8. DERIVED(p425)
9. DEPENDENT DERIVED(p426)
10. UNCACHEABLE SUBQUERY(p427)
11. UNCACHEABLE UNION(p428)
12. MATERIALIZED(p428)

10.3.3. table 칼럼(p429)
10.3.4. partitions 칼럼(p432)

10.3.5. type 칼럼(p434)
1. system(p435)
2. const(p436)
3. eq_ref(p438)
4. ref(p439)
5. fulltext(p440)
6. ref_or_null(p442)
7. unique_subquery(p442)
8. index_subquery(p443)
9. range(p443)
10. index_merge(p444)
11. index(p445)
12. ALL(p446)

10.3.6. possible_keys 칼럼(p448)
10.3.7. key 칼럼(p448)
10.3.8. key_len 칼럼(p449)

9주차 (2021-11-15 16:00, 오프라인)

10.3.9. ref 컬럼

- 컬럼값이 func 일 경우

10.3.10. rows 컬럼

10.3.11. filtered 컬럼

- Rows 컬럼과 filtered 컬럼 차이

10.3.12. Extra 컬럼

- 10.3.12.1 const row not found
- 10.3.12.2 deleting all rows
- 10.3.12.3 distinct
- 10.3.12.4 FirstMatch
- 10.3.12.5 Full scan on Null key
- 10.3.12.6 Impossible HAVING
- 10.3.12.7 Impossible WHERE
- 10.3.12.8 LooseScan
- 10.3.12.9 No Matching min/max row
- 10.3.12.10 no matching row in const table
- 10.3.12.11 No matching rows after partition pruning
- 10.3.12.12 No table Used
- 10.3.12.13 Not exists
- 10.3.12.14 Plan isn't ready yet
- 10.3.12.15 Range checked for each record
- 10.3.12.16 Recursive
- 10.3.12.17 Rematerialized
- 10.3.12.18 Select tables optimized away
- 10.3.12.19 Start temporary, End temporary
- 10.3.12.20 Unique row not found
- 10.3.12.21 Using filesort
- 10.3.12.22 Using index
- 10.3.12.23 Using index condition
- 10.3.12.24 Using index for group by
  Tight index-scan vs Loose index-scan
- 10.3.12.25 Using Index for skip scan

- 10.3.12.26 Using join buffer
- 10.3.12.27 Using MRR
- 10.3.12.28 Using sort_union
- 10.3.12.29 Using temporary
- 10.3.12.30 Using Where
- 10.3.12.31 Zero Limit

10주차 (2021-11-28 20:00)

11.1.1 SQL 모드

- MySQL 8.0 이전에서 8.0버전으로 업그레이드할 경우 주의해야하는 옵션과 그 이유는?

11.1.2 영문 대소문자 구분

11.1.3 MySQL 예약어

11.2 매뉴얼의 SQL 문법 표기를 읽는 방법

- 링크

13.2.6.1 INSERT ... SELECT Statement

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [(col_name [, col_name] ...)]
    {SELECT ... | TABLE table_name}
    [ON DUPLICATE KEY UPDATE assignment_list]

value:
    {expr | DEFAULT}

assignment:
    col_name = value

assignment_list:
    assignment [, assignment] ...
```

11.3.1 리터럴 표기법 문자열

- 문자열
- 숫자
- 날짜
- 불리언
    - tinyint로 사용

11.3.2 MySQL 연산자

- 동등 비교(=, <=>)
- 부정 (<>, !=)
- Not (!)
- AND(&&), OR(||)
- 나누기(/, DIV), 나머지(%, MOD)
- REGEXP
- LIKE
- BETWEEN
- IN

11.3.3 MySQL 내장 함수

- NULL 값 비교 및 대체 (IFNULL, ISNULL)
- 현재 시각 조회(NOW, SYSDATE)
    - NOW,SYSDATE의 차이점은?
- 날짜와 시간의 포맷(DATE_FORMAT, STR_TO_DATE)
- 날짜와 시간의 연산(DATE_ADD, DATE_SUB)
- 타임스탬프 연산(UNIX_TIMESTAMP, FROM_UNIXTIME)
- 문자열 처리(RPAD, LPAD/RTRIM, LTRIM, TRIM)
- 문자열 결합(CONCAT)
- GROUP BY 문자열 결합
- 값의 비교와 대체(CASE WHEN THEN END)
- 타입의 변환(CAST, CONVERT)
- 이진값과 16진수 문자열(Hex String)변환 (HEX, UNHEX)
- 암호화 및 해시 함수(MD5, SHA, SHA2)
- 처리 대기(SLEEP)
- 벤치마크(BENCHMARK)
- IP 주소 변환(INET_ATON, INET_NTOA)
- JSON 포맷(JSON_PRETTY)
- JSON 필드 크기(JSON_STORAGE_SIZE)
- JSON 필드 추출(JSON_EXTRACT)
- JSON 오브젝트 포함 여부 확인(JSON_CONTRAINS)
- JSON 오브젝트 생성(JSON_OBJECT)
- JSON 칼럼으로 집계(JSON_OBJECTAGG & JSON_ARRAYAGG)
- JSON 데이터를 테이블로 변환(JSON_TABLE)

11주차 (2021-12-05 20:00)

11.4.1 (P.53) SELECT 절 처리순서
11.4.2 (P.55) where, group by, order by 인덱스사용

11.4.2.1 인덱스 기본 규칙
- 인덱스를 타기 힘든 경우

11.4.2.2 (P.57) where절 인덱스 사용
- 인덱스, 컬럼 순서 상관관계
- And 조건, OR 조건 차이

11.4.2.3 (P.58) group by 조건
- 인덱스, 컬럼 순서 상관관계

11.4.2.4 (P.60) order by 조건
- 인덱스, 컬럼 순서 상관관계

11.4.2.5 (P.62~65) where 조건과 order by, group by 절 인덱스 사용

11.4.3. (P.66) where절 비교 조건사용사항
11.4.3.1 NULL 비교
- 인덱스에서 NULL

11.4.3.4 (P.72) Short-Circuit Evaluation

11.4.4 (P.77) DISTINCT
11.4.5 (P.77) LIMIT N
11.4.6 (P.82) COUNT
11.4.7 (P.84) JOIN
11.4.7.3 (P.88) OUTER JOIN 성능과 주의사항
11.4.7.4 (P.90) JOIN과 외래키
11.4.7.5 (P.91) 지연된 조인
11.4.7.6 (P.93) 래터럴 조인
11.4.7.7 (P.94) 실행계획으로 인한 정렬 흐트러짐
11.4.8 (P.96) GROUP BY
11.4.8.1 with rollup
11.4.8.2 레코드를 칼럼으로 변환
11.4.8.3 하나의 칼럼을 여러 칼럼으로 분리

11.4.9 (P101) ORDER BY
- order by 없이 select 하면
11.4.9.1 ORDER BY 사용법 및 주의사항
11.4.9.2 여러방향 동시정렬
11.4.9.2 함수나 표현식 사용

11.4.10 (P104) 서브쿼리
11.4.10.1 SELECT 절에 사용된 서브쿼리
11.4.10.2 FROM 절에 사용된 서브쿼리
11.4.10.3 WHERE 절에 사용된 서브쿼리
11.4.10.1 동등, 크다, 작다 비교
11.4.10.2 in 비교
11.4.10.3 NOT IN 비교

11.4.11 (P114) CTE (Common Table Expression)
11.4.11.1 (P115) 비 재귀 CTE
11.4.11.2 (P118) 재귀 CTE
11.4.12 (P122) 재귀 CTE 활용
11.4.12.3 (P133) 윈도우 함수
11.4.12.4 (P137) 윈도우 함수와 성능
11.4.13 (P139) 잠금을 사용하는 SELECT
11.4.13.2 (P142) NOWAIT & SKIP LOCKED

12주차 (2021-12-12 12:00)

11.5.1 INSERT 고급 옵션
- INSERT IGNORE
    - 에러가 나는 INSERT를 무시하고 계속 실행한다. NULL -> 기본값, 중복키 -> 무시
    -
- INSERT ON DUPLICATE KEY UPDATE
    - 중복되는 키가 있을 때 INSERT 대신 UPDATE를 실행한다

11.5.2 LOAD DATA 명령 주의 사항
- 단일 스레드/단일 트랜잭션으로 실행되어 데이터가 많을 경우 UNDO로그의 양이 많아지며, 실행시간도 길어져서 성능상으로 문제가 발생할 여지가 있다. 여러 파일로 나누어 실행하는것이 대안책

11.5.3.1 대량 INSERT성능
- 정렬된 PK로 INSERT시 더 빠른 성능을 보여준다. 클러스터링된 상태로 INSERT될 경우 디스크에 랜덤 액세스를 해야하기 때문이다.
- 세컨더리 인덱스가 많을수록 (당연하지만) INSERT 성능이 떨어진다

11.5.3.2 프라이머리 키 선정
11.5.3.3 Auto-Increment 칼럼
- PK가 Auto-Increment 컬럼일 경우 가장 빠른 INSERT를 보장한다

11.6.1 UPDATE ... ORDER BY ... LIMIT n
- PK로 정렬하지 않는 경우 레플리카서버에서도 동일한 row를 업데이트하는걸 보장하지 않기 때문에 주의해야한다.

11.6.2 JOIN UPDATE
- OLTP 환경에서는 데드락 유발 가능성이 있어 주의가 필요하다

11.6.3 여러 레코드 UPDATE

11.6.4 JOIN DELETE
- DELETE를 조인을 통해 여러 테이블에서 연관된 row를 한번에 삭제할 수 있음

11.7.1.1 온라인 DDL 알고리즘
- MySQL 8.0부터는 내장 온라인 DDL기능이 제공된다

11.7.1.2 온라인 처리 가능한 스키마 변경

11.7.1.3 INPLACE 알고리즘
11.7.1.4 온라인 DDL의 실패 케이스
11.7.1.5 온라인 DDL 진행 상황 모니터링

11.7.2 데이터베이스 변경
11.7.3 테이블 스페이스 변경
11.7.4 테이블 변경

11.7.5 칼럼 변경
11.7.6 인덱스 변경

11.7.7 테이블 변경 묶음 실행
11.7.8 프로세스 조회 및 강제 종료
11.7.9 활성 트랜잭션 조회

11.8.1 쿼리의 성능에 영향을 미치는 요소

13주차 (2021-12-19 20:00)

12. 확장 검색

12.1. 전문 검색(p212)

- 전문 검색이란?

12.1.1. 전문 검색 인덱스의 생성과 검색(p213)

- 형태소 분석

- n-gram 파서

12.1.2. 전문 검색 쿼리 모드(p220)

12.1.2.1. 자연어 검색(NATURAL LANGUAGE MODE)

12.1.2.2. 불리언 검색(BOOLEAN MODE) (p222)

12.1.2.3. 검색어 확장(QUERY EXPANSION) (p225)

12.1.3. 전문 검색 인덱스 디버깅(p226)

12.2. 공간 검색(p229)

- OGC

- OpenGIS

- SRC와 GCS, PCS

- SRID와 SRS-ID

- WKT와 WKB

- MBR과 R-TREE

12.2.2. SRS(Spatial Reference System)(p230)

12.2.3. 투영 좌표계와 평면 좌표계(p236)

12.2.4. 지리 좌표계(p242)

12.2.4.1. 지리 좌표계 데이터 관리

12.2.4.2. 지리 좌표계 주의 사항(p250)

12.2.4.2.1. 정확성 주의사항

12.2.4.2.2. 성능 주의사항(p254)

12.2.4.2.3. 좌표계 변환(p256)

14주차 (2021-12-26 20:00)

13. 파티션

13.1 개요

13.1.1 파티션을 사용하는 이유

- 책 p.263~265에 나와 있는 3가지 경우 중 2가지를 선택해서 설명해 주세요.

13.1.2 MySQL 파티션의 내부 처리

- 파티션 테이블을 검색할 때 성능에 미치는 조건은 무엇일까요?
  그리고 그 조건의 조합이 어떻게 실행되는지 설명해 주세요.
- 파티션 프루닝(Partition pruning)에 대해서 설명해 주세요.

13.2 주의사항

- MySQL 서버의 파티션 기능의 제약 사항 중 5가지 이상 설명해 주세요.
  그리고 가장 크게 영향을 미치는 제약 사항은 무엇인가요?
- 유니크 인덱스 사용 시 파티션 키 사용 방법과 주의사항을 예제를 들어 설명해 주세요.

13.3 MySQL 파티션의 종류

13.3.3.1 레인지 파티션

13.3.3.2 리스트 파티션

13.3.3.3 해시 파티션

13.3.3.4 키 파티션

- MySQL에서는 4가지 기존 파티션 기법을 제공하고 있습니다. 각 파티션의 용도와 방법, 주의사항에 대해서 설명해 주세요.
  - 방법에 대한 내용은 테이블 생성, 분리와 병합에 대한 내용을 다루고 있으니 선택해서 간략하게 설명해 주세요.

	레인지 파티션	리스트 파티션	해시 파티션	키 파티션
용도				
방법				
주의사항				

13.3.5 리니어 해시 파티션/리니어 키 파티션

- 리니어 해시 파티션과 리니어 키 파티션에서 사용하고 있는 알고리즘은 무엇인가요? 그리고 주의사항이 어떤게 있는지 설명해 주세요.

14. 스토어드 프로그램

- 스토어드 프로그램은 무엇일까요?

14.1 스토어드 프로그램의 장단점

14.1.1 스토어드 프로그램의 장점

- 스토어드 프로그램의 장점 중 2가지 이상 설명해 주세요.

14.1.2 스토어드 프로그램의 단점

- 스토어드 프로그램의 단점을 설명해 주세요.

14.2 스토어드 프로그램의 문법

- 스토어드 프로그램의 종류 중 테이블의 레코드가 저장되거나 변경될 때 미리 정의해둔 작업을 자동으로 실행해주는 스토어드 프로그램은 무엇일까요?

14.2.2 스토어드 프로시저

- 스토어드 프로시저를 생성할 때 주의사항은 어떤게 있을까요?
- 스토어드 프로시저와 스토어드 함수의 큰 차이점은 무엇일까요?
- 스토어드 프로시저의 커서 반환에 대해 설명해 주세요.

14.2.3 스토어드 함수

- 스토어드 함수는 무엇인가요? 그리고 스토어드 함수가 스토어드 프로시저와 다른 부분에 대해 설명해 주세요.
- 스토어드 함수를 다음 예제처럼 CALL 명령어로 실행하면 “PROCEDURE ~ does no exit” 에러가 발생합니다. 스토어드 함수를 실행할 수 있는 명령어로 바꿔주세요.

예제 명령어	수정내용
CALL sf_sum(1,2) AS sum;	SELECT sf_sum(1,2) AS sum;

14.2.4 트리거

- 트리거에 대한 설명과 트리거 생성 방법에 대해 설명해 주세요.

14.2.5 이벤트

- 일회성 이벤트와 반복성 이벤트를 생성하는 방법에 대해 예제를 들어 설명해 주세요.
- 다음 명령어를 사용해서 확인할 수 있는 정보는 어떤게 있을까요? 2가지 이상 설명해 주세요.

```
SELECT * FROM information_schema.EVENTS \G
```

14.2.6 스토어드 프로그램 본문**(Body)** 작성

14.2.6.3 제어문

- 스토어드 프로그램에서 제어문 사용방법에 대해 설명해 주세요.
    - IF … ELSEIF … ELSE … AND IF
    - CASE WHEN … THEN … ELSE … END CASE
    - 반복 루프

14.2.6.4 핸들러와 컨디션을 이용한 에러 핸들링

- 책 p.326~334에서는 SQLSTATE와 에러 번호(Error No), 핸들러, 컨디션, 컨디션을 사용하는 핸들러 정의에 대한 내용을 다루고 있습니다. 기억나는대로 설명해 주세요.

14.2.6.5 시그널을 이용한 예외 발생

- 스토어드 프로그램의 BEGIN … END 블록에서 SIGNAL을 사용하는 방법에 대해 설명해 주세요.

14.2.6.6 커서

- 커서와 센서티브 커서, 인센서티브 커서에 대해 설명해 주세요.

14.3 스토어드 프로그램의 보안 옵션

- 책에 4가지 옵션이 나오는데, 각 4가지 옵션이 무엇인가요?
- 각 옵션에 대해 설명해 주세요.

14.4 스토어드 프로그램의 참고 및 주의사항

- MySQL 클라이언트에서 한글처리를 하려면 어떻게 해야 할까요?

- 스토어드 프로그램에서 사용자 변수 보다 로컬 변수를 사용해야 하는 이유는 무엇일까요?
- 스토어드 프로그램에서 재귀 호출을 막기 위한 방법을 설명해 주세요.

15주차 (2022-01-02 20:00)

15.1 문자열 (CHAR, VARCHAR)
15.1.1 저장공간 (356p)
Q. VARCHAR타입이 CHAR보다 성능적으로 좋지 않을 수 있는 이유는?
A) 가변타입은 사이즈 기록이 추가적으로 필요하다. 어디까지 읽어와야하는지를 추가적으로 알아야 한다. 데이터가 업데이트될 때 길이가 늘어나면, 데이터를 분리/이동시켜야 하기 때문에 더 느릴 수 있다.
15.1.2 저장공간과 스키마 변경 (360p)
15.1.3 문자집합(캐릭터 셋) (362p)
15.1.4 콜레이션 (367p)
Q. WHERE 조건에서 인덱스를 효율적으로사용하기 위해서는 어떤것이 일치해야할까?
1. 타입명
2. 타입 길이
3. 콜레이션
15.1.5 비교 방식(378p)
Q. 다음중 참인 것은? [pad_attribute=PAD SPACE]
1. SELECT ‘ ABC’ LIKE ‘ABC’ [redacted]
2. SELECT ‘ABC’ = ‘ABC ‘ [redacted]
3. SELECT ‘ABC ‘ LIKE ‘ABC ‘ [redacted]
4. SELECT ‘ ABC ’ = ‘ ABC ‘ [redacted]
15.1.6 문자열 이스케이프 처리(381p)

15.2 숫자
15.2.1 정수 (384p)
15.2.2 부동 소수점 (384p)
15.2.3 DECIMAL (386p)
15.2.4 정수 타입의 칼럼을 생성할 때의 주의사항 (386p)
15.2.5 자동 증가(AUTO_INCREMENT) 옵션 사용 (387p)

15.3 날짜와 시간 (389p)
15.3.1 자동 업데이트 (395p)

15.4 ENUM과 SET
15.4.1 ENUM (396p)
15.4.2 SET
Q. ENUM과 SET의 차이는?

15.5 TEXT와 BLOB (403p)

15.6 공간 데이터 타입 (408p)
15.6.1 공간 데이터 생성 (409p)

* SRID(spatial reference identifier)는 평면 지구 매핑 또는 둥근 지구 매핑에 사용되는 특정 타원면을 기준으로 하는 공간 참조 시스템
15.6.2 공간 데이터 조회 (411p)

15.7 JSON 타입
15.7.1 저장방식 (415p)
15.7.2 부분 업데이트 성능(418p)
15.7.3 JSON 타입 콜레이션과 비교(422p)
15.7.4 JSON 칼럼 선택 (422p)
Q. 정규화 vs JSON 장단점

15.8 가상 칼럼(파생 칼럼) (425p)

16주차 (2022-01-09 20:00)

16. 복제

16.1 개요(p429)
16.2 복제 아키텍처(p431)
16.3 복제 타입(p434)
16.3.1 바이너리 로그 파일 위치 기반 복제(p434)
16.3.1.1 바이너리 로그 파일 위치 기반의 복제 구축(p435)
16.3.1.2 바이너리 로그 파일 위치 기반의 복제에서 트랜잭션 건너뛰기(p442)

16.3.2 글로벌 트랜잭션 아이디(GTID) 기반 복제(p444)
16.3.2.1 GTID의 필요성(p445)
16.3.2.2 글로벌 트랜잭션 아이디(p448)
16.3.2.3 글로벌 트랜잭션 아이디 기반의 복제 구축(p452)

16.3.2.4 글로벌 트랜잭션 아이디 기반 복제에서 트랜잭션 건너뛰기(p459)
16.3.2.5 Non-GTID 기반 복제에서 GTID 기반 복제로 온라인 변경(p462)
16.3.2.6 GTID 기반 복제 제약 사항(p468)

16.4 복제 데이터 포맷
16.4.1 Statement 기반 바이너리 로그 포맷(p469)
16.4.2 Row 기반 바이너리 로그 포맷(p471)
16.4.3 Mixed 포맷(p472)
16.4.4 Row 포맷의 용량 최적화(p473)
16.4.4.1 바이너리 로그 Row 이미지(p473)
16.4.4.2 바이너리 로그 트랜잭션 압축(p475)

16.5 복제 동기화 방식
16.5.1 비동기 복제(Asynchronous replication)(p485)
16.5.2 반동기 복제(Semi-synchronous replication)(p486)
16.5.2.1 반동기 복제 설정 방법(p490)

16.6 복제 토폴로지

16.6.1 싱글 레플리카 복제 구성(p495)
16.6.2 멀티 레플리카 복제 구성(p496)
16.6.3 체인 복제 구성(p497)
16.6.4 듀얼 소스 복제 구성(p501)
16.6.5 멀티 소스 복제 구성(p503)
16.6.5.1 멀티 소스 복제 동작(p504)

16.6.5.2 멀티 소스 복제 구축(p506)

16.7 복제 고급 설정
16.7.1 지연된 복제(Delayed Replication)(p515)
16.7.2 멀티 스레드 복제(Multi-threaded Replication)(p519)
16.7.2.1 데이터베이스 기반 멀티 스레드 복제(p521)

16.7.2.2 LOGICAL CLOCK 기반 멀티 스레드 복제(p524)
16.7.2.3 멀티 스레드 복제와 복제 포지션 정보(p537)

16.7.3 크래시 세이프 복제(Crash-safe Replication)(p540)
16.7.3.1 서버 장애와 복제 실패(p540)
16.7.3.2 복제 사용 형태별 크래시 세이프 복제 설정(p543)
16.7.4 필터링된 복제(Filtered Replication)(p548~554)

17주차 (2022-01-16 20:00)

17.1 InnoDB 클러스터 아키텍처 [557p]
17.2 그룹 복제(Group Replication) [559p]
17.2.1 그룹복제 아키텍처 [562p]

17.2.2 그룹 복제 모드 [564p]
17.2.2.1 싱글 프라이머리 모드
17.2.2.2 멀티 프라이머리 모드
17.2.3 그룹 멤버관리 [568p]

17.2.4 그룹 복제에서의 트랜잭션 처리 [572p]
17.2.4.1 트랜잭션 일관성 수준
17.2.4.1.1 EVENTUAL 일관성 수준
17.2.4.1.2 BEFORE_ON_PRIMARY_FAILOVER 일관성 수준
17.2.4.1.3 BEFORE 일관성 수준
17.2.4.1.4 AFTER 일관성 수준
17.2.4.1.5 BEFORE_AND_AFTER 일관성 수준

17.2.4.2 흐름제어 [580p]
17.2.5 그룹복제의 자동 장애 감지 및 대응[584p]

17.2.6 그룹 복제의 분산 복구[588p]
17.2.6.1 분산 복구 방식

17.2.6.2 분산 복구 프로세스
17.2.6.3 분산 복구 설정
17.2.6.4 분산 복구 오류 처리

17.2.7 그룹 복제 요구사항 [595p]
17.2.8 그룹 복제 제약 사항 [597p]
17.3 MySQL 셸 [598p]
17.4 MySQL 라우터 [600p]

17.5 InnoDB 클러스터 구축 [602p]
17.5.1 InnoDB 클러스터 요구사항
17.5.2 InnoDB 클러스터 생성
17.5.2.1 사전준비
17.5.2.2 InnoDB 클러스터 생성
17.5.2.3 InnoDB 클러스터 인스턴스 추가

17.5.2.4 MySQL 라우터 설정 [612p]

17.6 InnoDB 클러스터 모니터링 [621p]
17.7 InnoDB 클러스터 작업[625p]
17.7.1 클러스터 모드 변경
17.7.2 프라이머리 변경
17.7.3 인스턴스 제거
17.7.4 클러스터 해체

17.7.5 클러스터 및 인스턴스 설정 변경[631p]
17.7.5.1 빌트인 태그

17.8 InnoDB 클러스터 트러블슈팅[639p]
17.8.1 클러스터 인스턴스 장애
17.8.2 클러스터의 정족수 손실
17.9 InnoDB 클러스터 버전 업그레이드
17.10 InnoDB 클러스터 제약 사항

18주차 (2022-01-23 16:00)

18.1. Performance 스키마란? (P647)
18.2. Performance 스키마 구성 (P648)

18.2.1. Setup 테이블 (P649)
18.2.2. Instance 테이블 (P650)
18.2.3. Connection 테이블 (P650)
18.2.4. Variable 테이블 (P651)
18.2.5. Event 테이블 (P652)

Event 테이블 구성