MYSQL 튜닝

업무에 바로 쓰는 SQL 튜닝

목차

DB 튜닝과 필요성

DB 튜닝 필요성

SQL 튜닝의 필요성

- 데이터의 수가 많아지면 테이블을 풀스캔을 통해 조회하면 시간이 오래걸린다.
- SQL 튜닝은 최소한의 자원으로 상대적으로 빠른 시간 내에 데이터 CRUD 작업을 할 수 있도록 튜닝하는 것을 말한다.
- 결론 : 튜닝은 주어진 환경을 통해 처리량과 응답속도를 개선하기 위해서 필요하다.

실행 계획(EXPLAIN)

실행 계획(EXPLAIN)

MySQL의 실행 계획

- SQL문 앞에 EXPLAIN 키워드를 입력하고 실행하면 옵티마이저가 만든 실행계획이 출력됩니다. (DESCRIBE, DEC 도 가능)

```
mysql> EXPLAIN
    -> SELECT *
    -> FROM 사원
    -> WHERE 사원번호 BETWEEN 100001 AND 200000;
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
| id | select_type | table  | partitions | type  | possible_keys | key     | key_len | ref  | rows  | filtered | Extra       |
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
|  1 | SIMPLE      | 사원   | NULL       | range | PRIMARY       | PRIMARY | 4       | NULL | 20080 |   100.00 | Using where |
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

실행 계획(EXPLAIN)

Id

- 실행 순서를 표시하는 숫자입니다. ID가 작을수록 먼저 수행된 것이고, ID가 같은 값이라면 두 개 테이블의 조인이 이루어 진것 입니다.

```
mysql> EXPLAIN
    -> SELECT 사원.사원번호, 사원.이름, 사원.성, 급여.연봉,
    ->        (SELECT MAX(부서번호)
    ->         FROM 부서사원_매핑 as 매핑 WHERE 매핑.사원번호 = 사원.사원번호) 카운트
    -> FROM 사원, 급여
    -> WHERE 사원.사원번호 = 10001
    -> AND 사원.사원번호 = 급여.사원번호;
+----+--------------------+--------+------------+-------+---------------+---------+---------+-------+------+----------+-----------------------------+
| id | select_type        | table  | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtered | Extra                       |
+----+--------------------+--------+------------+-------+---------------+---------+---------+-------+------+----------+-----------------------------+
|  1 | PRIMARY            | 사원   | NULL       | const | PRIMARY       | PRIMARY | 4       | const |    1 |   100.00 | NULL                        |
|  1 | PRIMARY            | 급여   | NULL       | ref   | PRIMARY       | PRIMARY | 4       | const |   17 |   100.00 | NULL                        |
|  2 | DEPENDENT SUBQUERY | NULL   | NULL       | NULL  | NULL          | NULL    | NULL    | NULL  | NULL |     NULL | Select tables optimized away |
+----+--------------------+--------+------------+-------+---------------+---------+---------+-------+------+----------+-----------------------------+
3 rows in set, 2 warnings (0.01 sec)
```

실행 계획(EXPLAIN)

select_type

- SQL문을 구성하는 SELECT 문의 유형을 가르키는 항목입니다.

TYPE	설명
SIMPLE	UNION이나 내부쿼리가 없는 단순한 SELECT 구문
PRIMARY	서브쿼리나 UNION이 포함된 SQL문에서 첫번째 SELECT문
DERIVED	FROM절에 작성된 서브쿼리
SUBQUERY	SELECT 하위 쿼리의 첫번째
UNION	두번째 이후 SELECT의 UNION 구문
UNION RESULT	UNION의 결과
DEPENDENT SUBQUERY	외부 쿼리의 SELECT 하위 쿼리 첫번째
DEPENDENT UNION	외부 쿼리에 종속된 두번째 이후의 SELECT의 UNION
UNCACHEABLE SUBQUERY	결과를 재사용 할 수 없는 서브쿼리, 사용자정의 함수/변수, RAND() UUID() 가 포함된 것
MATERIALIZED	IN절에 연결된 서브쿼리로 임시테이블 생성해서 수행합니다.

```
mysql> EXPLAIN
    -> SELECT * FROM 사원 WHERE 사원번호 = 100000;
+----+-------------+--------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
| id | select_type | table  | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtered | Extra |
+----+-------------+--------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | 사원   | NULL       | const | PRIMARY       | PRIMARY | 4       | const |    1 |   100.00 | NULL  |
+----+-------------+--------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

실행 계획(EXPLAIN)

Table

- 출력 행이 참조하는 테이블의 이름

```
EXPLAIN
SELECT 사원.사원번호, 급여.연봉
FROM 사원,
      (SELECT 사원번호, MAX(연봉) as 연봉
       FROM 급여
       WHERE 사원번호 BETWEEN 10001 AND 20000 ) as 급여
WHERE 사원.사원번호 = 급여.사원번호;

+----+-------------+------------+------------+--------+---------------+---------+---------+-------+--------+----------+-------------+
| id | select_type | table      | partitions | type   | possible_keys | key     | key_len | ref   | rows   | filtered | Extra       |
+----+-------------+------------+------------+--------+---------------+---------+---------+-------+--------+----------+-------------+
|  1 | PRIMARY     | <derived2> | NULL       | system | NULL          | NULL    | NULL    | NULL  |      1 |   100.00 | NULL        |
|  1 | PRIMARY     | 사원       | NULL       | const  | PRIMARY       | PRIMARY | 4       | const |      1 |   100.00 | Using index |
|  2 | DERIVED     | 급여       | NULL       | range  | PRIMARY       | PRIMARY | 4       | NULL  | 184756 |   100.00 | Using where |
+----+-------------+------------+------------+--------+---------------+---------+---------+-------+--------+----------+-------------+
3 rows in set, 1 warning (0.03 sec)
```

실행 계획(EXPLAIN)

Partitions

- 쿼리와 일치하는 레코드가 있는 파티션 입니다.
- 사전에 정의한 전체 파티션 중 특정 파티션에 선택적으로 접근하는 것이 SQL 성능면에서 유리합니다. 너무 많은 파티션에 접근하는 것이 확인된다면 파티션 정의를 튜닝 해야 합니다.

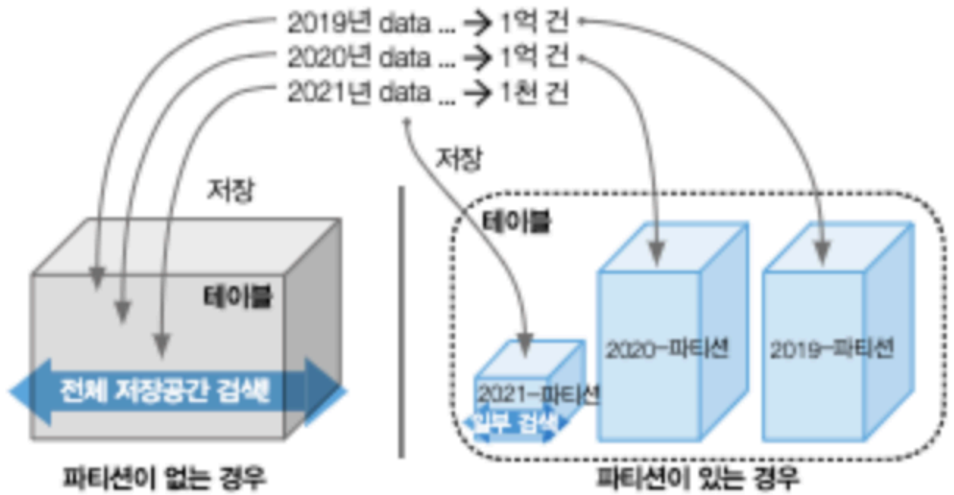


그림 3-39 파티션 유무에 따른 액세스 범위 비교

실행 계획(EXPLAIN)

Type

- 테이블의 데이터를 어떻게 찾을 것인가

TYPE	설명
system	테이블에 데이터가 없거나 하나의 행만 있는 경우 성능상 최상의 type
const	조회되는 데이터가 하나만 있는 경우
eq_ref	테이블 조합에 대해 조회되는 데이터가 하나만 있는 경우
ref	테이블 조합에 대해 여러 데이터를 조회하는 경우
fulltext	텍스트 검색을 빠르게 처리하기 위해 전문 인덱스를 사용해서 접근하는 방식
ref_or_null	ref과 유사하지만 IS NULL 에 추가 검색을 수행
index_merge	결합된 인덱스로 접근하는 경우
range	테이블 내의 연속된 범위를 조회하는 유형 =, <, >
Index	인덱스 풀 스캔
all	테이블 풀 스캔

```
mysql> EXPLAIN
    -> SELECT *
    -> FROM 사원
    -> WHERE 사원번호 BETWEEN 100001 AND 200000;
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
| id | select_type | table  | partitions | type  | possible_keys | key     | key_len | ref  | rows  | filtered | Extra       |
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
|  1 | SIMPLE      | 사원   | NULL       | range | PRIMARY       | PRIMARY | 4       | NULL | 20080 |   100.00 | Using where |
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

실행 계획(EXPLAIN)

possible_keys

- MySQL의 옵티마이저가 이 테이블에 사용 할 수 있는 인덱스 목록
- 실제 사용하는 인덱스가 아닌 후보군을 보여주는 것

Key

- MySQL이 실제 사용하기로 결정한 키(인덱스)

실행 계획(EXPLAIN)

```
mysql> EXPLAIN
    -> SELECT 사원번호
    -> FROM 직급
    -> WHERE 직급명 = 'Manager';
+----+-------------+--------+------------+-------+---------------+---------+---------+------+--------+----------+--------------
| id | select_type | table  | partitions | type  | possible_keys | key     | key_len | ref  | rows   | filtered | Extra
+----+-------------+--------+------------+-------+---------------+---------+---------+------+--------+----------+--------------
|  1 | SIMPLE      | 직급   | NULL       | index | PRIMARY       | PRIMARY | 159     | NULL | 441891 |    10.00 | Using where; Us
+----+-------------+--------+------------+-------+---------------+---------+---------+------+--------+----------+--------------
1 row in set, 1 warning (0.02 sec)

mysql> EXPLAIN
    -> SELECT * FROM 사원;
+----+-------------+--------+------------+------+---------------+------+---------+------+--------+----------+-------+
| id | select_type | table  | partitions | type | possible_keys | key  | key_len | ref  | rows   | filtered | Extra |
+----+-------------+--------+------------+------+---------------+------+---------+------+--------+----------+-------+
|  1 | SIMPLE      | 사원   | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 299512 |   100.00 | NULL  |
+----+-------------+--------+------------+------+---------------+------+---------+------+--------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

실행 계획(EXPLAIN)

key_len

- MySQL이 사용하기로 결정한 키의 길이(byte)

```
mysql> EXPLAIN
    -> SELECT *
    -> FROM 사원
    -> WHERE 사원번호 BETWEEN 100001 AND 200000;
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
| id | select_type | table  | partitions | type  | possible_keys | key     | key_len | ref  | rows  | filtered | Extra       |
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
|  1 | SIMPLE      | 사원   | NULL       | range | PRIMARY       | PRIMARY | 4       | NULL | 20080 |   100.00 | Using where |
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

실행 계획(EXPLAIN)

ref

- Reference의 약자로 테이블 조인을 수행할 때 어떤 조건으로 조인 했는지 알려줍니다.

```
mysql> EXPLAIN
    -> SELECT 사원.사원번호, 직급.직급명
    -> FROM 사원, 직급
    -> WHERE 사원.사원번호 = 직급.사원번호
    -> AND 사원.사원번호 BETWEEN 10001 AND 10100;
+----+-------------+--------+------------+-------+---------------+---------+---------+--------------------------+------+-----
| id | select_type | table  | partitions | type  | possible_keys | key     | key_len | ref                      | rows | filt
+----+-------------+--------+------------+-------+---------------+---------+---------+--------------------------+------+-----
|  1 | SIMPLE      | 사원   | NULL       | range | PRIMARY       | PRIMARY | 4       | NULL                     |  100 |   100
|  1 | SIMPLE      | 직급   | NULL       | ref   | PRIMARY       | PRIMARY | 4       | tuning.사원.사원번호     |    1 |   100.00
+----+-------------+--------+------------+-------+---------------+---------+---------+--------------------------+------+-----
2 rows in set, 1 warning (0.01 sec)
```

실행 계획(EXPLAIN)

row

- SQL문을 수행하고자 접근하는 데이터의 모든 행의 수를 나타낸다.

filtered

- 테이블 조건으로 필터링된 테이블 행의 예상 백분율

실행 계획(EXPLAIN)

Extra

- SQL문을 어떻게 수행할 것인지에 대한 정보

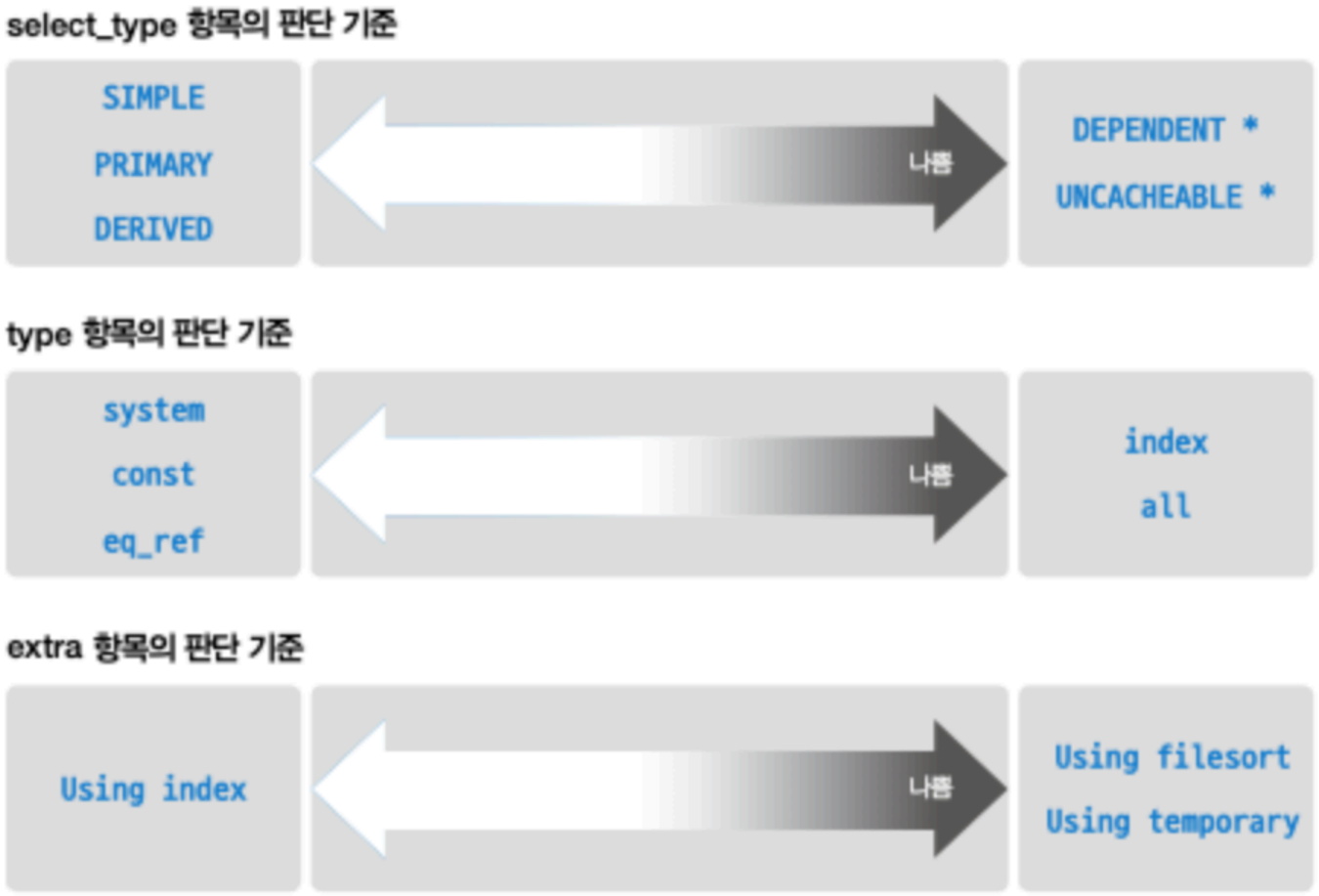
TYPE	설명
Distinct	중복이 제거되어 유일한 값을 찾을 때 출력되는 정보
Using where	테이블에서 행을 가져온 후 where절의 검색조건을 적용해 해의 범위를 축소한 것
Using temporary	정렬등의 이유로 중간 결과를 저장하고자 하는 임시 테이블을 생성하는 경우
Using index	테이블에 접근하지 않고 인덱스만을 읽어서 처리 할수 있는 경우
Using filesort	Order by인덱스로 해결하지 못하고 filesort(MySQL의 quick sort)로 행을 정렬한 것
Using join buffer	조인을 수행하기위해 조인 버퍼를 사용
Not exist	하나의 일치하는 행을 찾으면 추가로 검색하지 않는 유형

```
mysql> EXPLAIN
    -> SELECT *
    -> FROM 사원
    -> WHERE 사원번호 BETWEEN 100001 AND 200000;
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
| id | select_type | table  | partitions | type  | possible_keys | key     | key_len | ref  | rows  | filtered | Extra       |
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
|  1 | SIMPLE      | 사원   | NULL       | range | PRIMARY       | PRIMARY | 4       | NULL | 20080 |   100.00 | Using where |
+----+-------------+--------+------------+-------+---------------+---------+---------+------+-------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

좋고 나쁨을 판단하는 기준

SQL 튜닝 대상 판별

- 튜닝이 필요한 작업을 명확히 선을 그어 구분하긴 어렵습니다.
- 상황에 따라 달라 어느 쪽이 좋다/나쁘다 늘 단언하기 어렵습니다.
- 하지만 SQL 튜닝 대상을 검토 할 때 selecy_type, type, extra을 참조 할수 있습니다.



실습

실습

따라하기

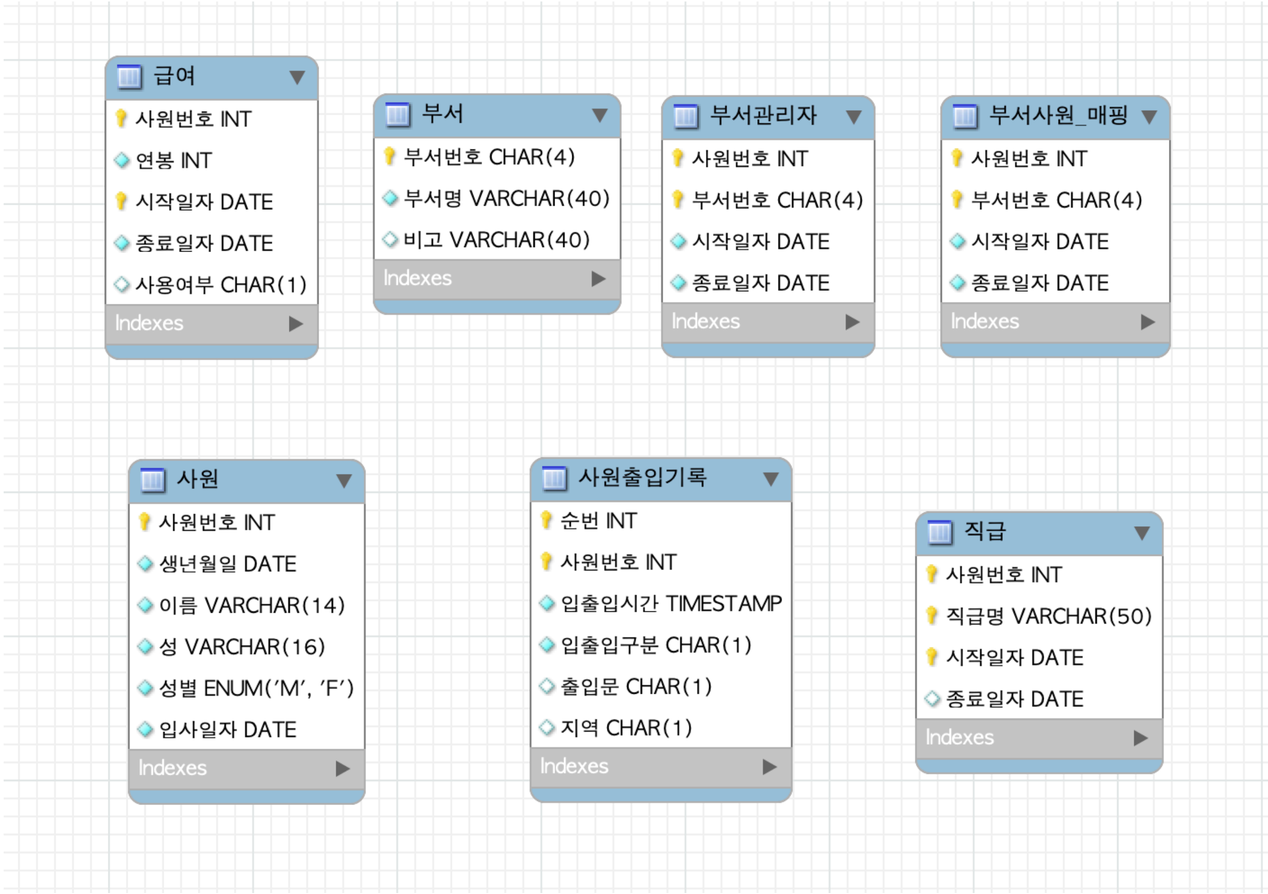
- https://github.com/7ieon/SQLtune
- 다운로드 -> 압축해제
- 해당 폴더로 이동

- Mysql -uroot -p —port 포트번호 < data_setting.sql

테이블 소개

테이블의 데이터 건수

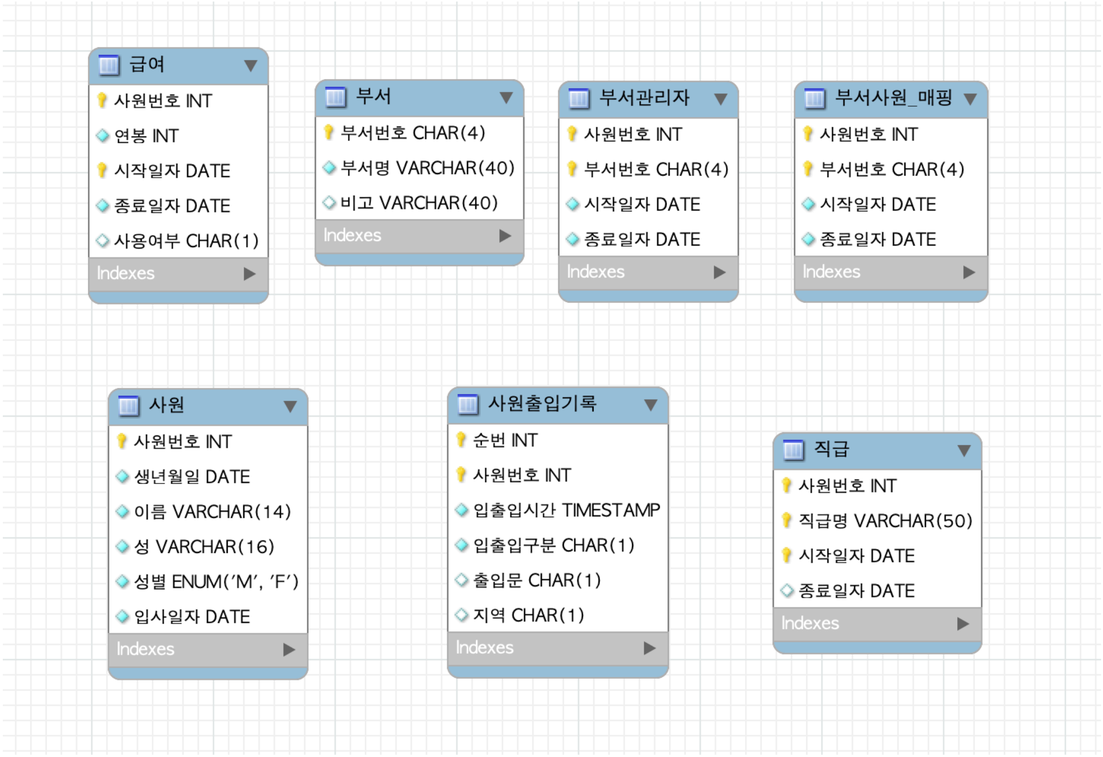
테이블명	데이터 건수
급여	2,844,047
부서	9
부서관리자	24
부서사원_매핑	331,603
사원	300,024
사원출입기록	660,000
직급	443,308



테이블 소개

테이블의 인덱스

테이블명	키 유형	키명	키_구성열
급여	PK	PRIAMRY KEY	사원번호 + 시작일자
	INDEX	I_사용여부	사용여부
부서	PK	PRIAMRY KEY	부서번호
	UNIQUE INDEX	UI_부서명	부서명
부서관리자	PK	PRIAMRY KEY	사원번호 + 부서번호
	INDEX	I_부서번호	부서번호
부서사원_매핑	PK	PRIAMRY KEY	사원번호 + 부서번호
	INDEX	I_부서번호	부서번호
사원	PK	PRIAMRY KEY	사원번호
	INDEX	I_입사일자	입사일자
	INDEX	I_성별_성	성별 + 성
사원출입기록	PK	PRIAMRY KEY	순번 + 사원번호
	INDEX	I_출입문	출입문
	INDEX	I_지역	지역
	INDEX	I_시간	입출입시간
직급	PK	PRIAMRY KEY	사원번호 + 직급명 + 시작일자



show index from 급여;

```
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_c
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------
| 급여  |          0 | PRIMARY  |            1 | 사원번호    | A         |      298323 |     NULL | NULL   |      | BTREE      |         |
| 급여  |          0 | PRIMARY  |            2 | 시작일자    | A         |     2838731 |     NULL | NULL   |      | BTREE      |         |
| 급여  |          1 | I_사용여부 |          1 | 사용여부    | A         |           1 |     NULL | NULL   | YES  | BTREE      |         |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------
3 rows in set (0.01 sec)
```

==========

show index from 부서;

```
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_c
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------
| 부서  |          0 | PRIMARY  |            1 | 부서번호    | A         |           9 |     NULL | NULL   |      | BTREE      |         |
| 부서  |          0 | UI_부서명 |           1 | 부서명      | A         |           9 |     NULL | NULL   |      | BTREE      |         | YE
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------
2 rows in set (0.00 sec)
```

==========

show index from 부서관리자;

SQL문 단순 수정 튜닝

SQL문 단순 수정 튜닝

형변환으로 인덱스를 활용하지 못하는 나쁜 SQL 문

```
SELECT COUNT(1)
FROM 급여
WHERE 사용여부 = 1;
```

```
+----------+
| COUNT(1) |
+----------+
|    42842 |
+----------+
1 row in set (0.58 sec)
```

```
+----+-------------+-------+-----------+------+---------------+---------------+---------+------+---------+----------+-------------------------+
| id | select_type | table | partitions | type | possible_keys | key           | key_len | ref  | rows    | filtered | Extra                   |
+----+-------------+-------+-----------+------+---------------+---------------+---------+------+---------+----------+-------------------------+
|  1 | SIMPLE      | 급여  | NULL       | index | I_사용여부    | I_사용여부    | 4       | NULL | 2838731 |    10.00 | Using where; Using index |
+----+-------------+-------+-----------+------+---------------+---------------+---------+------+---------+----------+-------------------------+
1 row in set, 3 warnings (0.01 sec)
```

- key 항목이 I_사용여부 로 출력되므로 해당 인덱스를 사용
- type 항목이 index이므로 인덱스 풀 스캔으로 수행
- filtered 항목이 10.00이므로 MySQL 엔진으로 가져온 데이터 중 10%를 추출해서 최종 데이터를 출력함

SQL문 단순 수정 튜닝

형변환으로 인덱스를 활용하지 못하는 나쁜 SQL 문

```
+-------------+---------+------+-----+---------+-------+
| Field       | Type    | Null | Key | Default | Extra |
+-------------+---------+------+-----+---------+-------+
| 사원번호    | int     | NO   | PRI | NULL    |       |
| 연봉        | int     | NO   |     | NULL    |       |
| 시작일자    | date    | NO   | PRI | NULL    |       |
| 종료일자    | date    | NO   |     | NULL    |       |
| 사용여부    | char(1) | YES  | MUL |         |       |
+-------------+---------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

```
SELECT 사용여부, COUNT(1)
FROM 급여
GROUP BY 사용여부;
```

```
+-------------+----------+
| 사용여부    | COUNT(1) |
+-------------+----------+
| 0           |  2801205 |
| 1           |    42842 |
+-------------+----------+
2 rows in set (0.41 sec)
```

- 사용 여부 열은 문자형이 char(1)인 데이터 유형이지만, where 조건에선 숫자 유형으로 데이터에 접근
=> DBMS 내부의 묵시적 형변환 발생
=> 전체 데이터를 스캔하는 FULL SCAN 발생

SQL문 단순 수정 튜닝

형변환으로 인덱스를 활용하지 못하는 나쁜 SQL 문

```
SELECT COUNT(1)
FROM 급여
WHERE 사용여부 = '1';
```

```
+----------+
| COUNT(1) |
+----------+
|    42842 |
+----------+
1 row in set (0.03 sec)
```

```
+----+-------------+-------+-----------+------+---------------+---------------+---------+-------+-------+----------+-------------+
| id | select_type | table  | partitions | type | possible_keys  | key            | key_len | ref   | rows  | filtered | Extra       |
+----+-------------+-------+-----------+------+---------------+---------------+---------+-------+-------+----------+-------------+
|  1 | SIMPLE      | 급여   | NULL       | ref  | I_사용여부     | I_사용여부     | 4       | const | 82824 |   100.00 | Using index |
+----+-------------+-------+-----------+------+---------------+---------------+---------+-------+-------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

SQL문 단순 수정 튜닝

인덱스 고려 없이 열을 사용하는 나쁜 SQL 문

```
+-----------------+--------+-----------+
| 성              | 성별   | 카운트    |
+-----------------+--------+-----------+
| Aamodt          | M      |       120 |
| Aamodt          | F      |        85 |
| Acton           | M      |       108 |
… 중략 …
| Zyda            | F      |        72 |
| Zykh            | M      |        87 |
| Zykh            | F      |        61 |
+-----------------+--------+-----------+
3274 rows in set (0.56 sec)
```

```
SELECT 성, 성별, COUNT(1) as 카운트
  FROM 사원
 GROUP BY 성, 성별;
```

```
+----+-------------+-------+------------+------+---------------+-------------+---------+-----+--------+----------
+---------------------------+
| id | select_type | table  | partitions | type  | possible_keys | key          | key_len | ref  | rows   | filtered |
Extra                        |
+----+-------------+-------+------------+------+---------------+-------------+---------+-----+--------+----------
+---------------------------+
|  1 | SIMPLE      | 사원   | NULL       | index | I_성별_성     | I_성별_성    | 51      | NULL | 299512 |   100.00 |
Using index; Using temporary |
+----+-------------+-------+------------+------+---------------+-------------+---------+-----+--------+----------
+---------------------------+
1 row in set, 1 warning (0.00 sec)
```

- I_성별_성 index를 사용하고, 임시테이블을 생성하여 성 / 성별을 grouping 해 count() 연산 수행
- I_성별_성 index의 구성 열이 group by 절에 포함되므로, 테이블 접근 없이 인덱스만 사용하는 커버링 인덱스(Using Index)로 수행

SQL문 단순 수정 튜닝

인덱스 고려 없이 열을 사용하는 나쁜 SQL 문

```
+--------+------------+----------------+--------------+-------------+-----------+-------------+----------+--------+------+---
| Table  | Non_unique | Key_name       | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | In
+--------+------------+----------------+--------------+-------------+-----------+-------------+----------+--------+------+---
| 사원   |          0 | PRIMARY        |            1 | 사원번호    | A         |      299512 |     NULL |   NULL |      | BTREE
| 사원   |          1 | I_입사일자     |            1 | 입사일자    | A         |        4718 |     NULL |   NULL |      | BTREE
| 사원   |          1 | I_성별_성      |            1 | 성별        | A         |           1 |     NULL |   NULL |      | BTREE
| 사원   |          1 | I_성별_성      |            2 | 성          | A         |        3360 |     NULL |   NULL |      | BTREE
+--------+------------+----------------+--------------+-------------+-----------+-------------+----------+--------+------+---
3274 rows in set (0.56 sec)

+----+-------------+-------+------------+------+---------------+-------------+---------+------+--------+----------+
+----------------------------+
| id | select_type | table  | partitions | type  | possible_keys | key          | key_len | ref  | rows   | filtered |
Extra                         |
+----+-------------+-------+------------+------+---------------+-------------+---------+------+--------+----------+
+----------------------------+
|  1 | SIMPLE      | 사원   | NULL       | index | I_성별_성     | I_성별_성    | 51      | NULL | 299512 |   100.00 |
Using index; Using temporary |
+----+-------------+-------+------------+------+---------------+-------------+---------+------+--------+----------+
+----------------------------+
1 row in set, 1 warning (0.00 sec)
```

- 인덱스를 활용하는데도 메모리나 디스크에 임시테이블을 꼭 생성해야 하는지?
- > I_성별_성 index는 성별 컬럼 기준으로 정렬 후 성 컬럼으로 정렬되었다는 의미이므로, 인덱스 순서 활용 가능

SQL문 단순 수정 튜닝

인덱스 고려 없이 열을 사용하는 나쁜 SQL 문

```
SELECT 성, 성별, COUNT(1) as 카운트
  FROM 사원
 GROUP BY 성별, 성;
```

```
+------------------+--------+-----------+
| 성                | 성별    | 카운트     |
+------------------+--------+-----------+
| Aamodt           | M      |       120 |
| Aamodt           | F      |        85 |
| Acton            | M      |       108 |
| Acton            | F      |        81 |
… 중략 …
| Zyda             | F      |        72 |
| Zykh             | M      |        87 |
| Zykh             | F      |        61 |
+------------------+--------+-----------+
3274 rows in set (0.10 sec)

+----+-------------+-------+------------+------+---------------+-------------+---------+-----+--------+----------
+-------------+
| id | select_type | table  | partitions | type  | possible_keys | key          | key_len | ref  | rows   | filtered |
Extra       |
+----+-------------+-------+------------+------+---------------+-------------+---------+-----+--------+----------
+-------------+
|  1 | SIMPLE      | 사원   | NULL       | index | I_성별_성     | I_성별_성    | 51      | NULL | 299512 |   100.00 |
Using index |
+----+-------------+-------+------------+------+---------------+-------------+---------+-----+--------+----------
+-------------+
1 row in set, 1 warning (0.00 sec)
```

SQL문 단순 수정 튜닝

동등 조건으로 인덱스를 사용하는 나쁜 SQL 문

```
SELECT *
FROM 사원출입기록
WHERE 출입문 = 'B';
```

```
+---------+----------+---------------------+-----------+--------+------+
| 순번    | 사원번호 | 입출입시간          | 입출입구분 | 출입문 | 지역 |
+---------+----------+---------------------+-----------+--------+------+
|  983026 |   110022 | 2020-05-26 11:16:28 |           | B      | b    |
|  983027 |   110085 | 2020-09-05 01:20:57 |           | B      | b    |
|  983028 |   110183 | 2020-03-09 20:55:22 |           | B      | b    |
... 중략 ...
| 983033 |   111400 | 2020-08-03 08:41:13 |           | B      | b    |
| 983034 |   111692 | 2020-07-12 04:42:28 |           | B      | b    |
| 983035 |   110114 | 2020-11-15 21:28:40 |           | B      | b    |
+--------+----------+---------------------+-----------+--------+------+
300000 rows in set (0.45 sec)

+----+-------------+-------------+------------+------+---------------+---------+---------+-------+--------+----------+-------+
| id | select_type | table       | partitions | type | possible_keys | key     | key_len | ref   | rows   | filtered | Extra |
+----+-------------+-------------+------------+------+---------------+---------+---------+-------+--------+----------+-------+
|  1 | SIMPLE      | 사원출입기록 | NULL       | ref  | I_출입문      | I_출입문 | 4       | const | 329467 |   100.00 | NULL  |
+----+-------------+-------------+------------+------+---------------+---------+---------+-------+--------+----------+-------+
1 row in set, 1 warning (0.02 sec)
```

SQL문 단순 수정 튜닝

동등 조건으로 인덱스를 사용하는 나쁜 SQL 문

```
SELECT 출입문, COUNT(1)
FROM 사원출입기록
GROUP BY 출입문;
```

* 결과

```
+--------+----------+
| 출입문 | COUNT(1) |
+--------+----------+
| A      |   250000 |
| B      |   300000 |
| C      |    10000 |
| D      |   100000 |
+--------+----------+
```

- 사원출입기록 테이블 중 출입문 B는 총 66만건 데이터 중 30만건을 차지하므로 전체 데이터의 50%
=> 앞의 실행계획에 따르면 I_출입문 index로 index scan을 수행하는데, 전체 데이터의 약 50%에 달하는 데이터를 조회하기 위해 인덱스를 활용하는 것은 효율적이지 않을 수 있다.

SQL문 단순 수정 튜닝

동등 조건으로 인덱스를 사용하는 나쁜 SQL 문

```
SELECT *
FROM 사원출입기록  IGNORE INDEX(I_출입문)
WHERE 출입문 = 'B';
```

```
+--------+---------+-------------------+-----------+--------+------+
| 순번    | 사원번호 | 입출입시간         | 입출입구분 | 출입문 | 지역 |
+--------+---------+-------------------+-----------+--------+------+
|  983026 |   110022 | 2020-05-26 11:16:28 | |       | B    | b   |
|  983027 |   110085 | 2020-09-05 01:20:57 | |       | B    | b   |
|  983028 |   110183 | 2020-03-09 20:55:22 | |       | B    | b   |
... 중략 ...
| 983033 |   111400 | 2020-08-03 08:41:13 | |       | B    | b   |
| 983034 |   111692 | 2020-07-12 04:42:28 | |       | B    | b   |
| 983035 |   110114 | 2020-11-15 21:28:40 | |       | B    | b   |
+--------+---------+-------------------+-----------+--------+------+
300000 rows in set (0.25 sec)
```

```
+----+-------------+-------------------+------------+------+---------------+------+---------+------+--------
+----------+-------------+
| id | select_type | table               | partitions | type | possible_keys | key  | key_len | ref  | rows   |
filtered | Extra       |
+----+-------------+-------------------+------------+------+---------------+------+---------+------+--------
+----------+-------------+
|  1 | SIMPLE      | 사원출입기록        | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 658935 |    10.00 |
Using where |
+----+-------------+-------------------+------------+------+---------------+------+---------+------+--------
+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

SQL문 조인 설정 변경 튜닝

SQL문 조인 설정 변경 튜닝

작은 테이블이 먼저 조인에 참여하는 나쁜 SQL문

```
+--------------+--------------+
| 사원번호      | 부서번호      |
+--------------+--------------+
|        11732 | d009         |
|        14179 | d009         |
|        16989 | d009         |
... 중략 ...
|       483592 | d007         |
|       487945 | d007         |
|       488117 | d007         |
+--------------+--------------+
1341 rows in set (0.49 sec)
```

```
SELECT 매핑.사원번호, 부서.부서번호
FROM 부서사원_매핑 매핑, 부서
WHERE 매핑.부서번호 = 부서.부서번호
AND 매핑.시작일자 >= '2002-03-01';
```

```
+----+-------------+--------+------------+-------+----------------+----------------+---------+-----------------------------+-------+----------+-------------+
| id | select_type | table  | partitions | type  | possible_keys  | key            | key_len | ref                         | rows  | filtered | Extra       |
+----+-------------+--------+------------+-------+----------------+----------------+---------+-----------------------------+-------+----------+-------------+
|  1 | SIMPLE      | 부서   | NULL       | index | PRIMARY        | UI_부서명      | 122     | NULL                        |     9 |   100.00 | Using index |
|  1 | SIMPLE      | 매핑   | NULL       | ref   | I_부서번호     | I_부서번호     | 12      | tuning.부서.부서번호        | 41392 |    33.33 | Using where |
+----+-------------+--------+------------+-------+----------------+----------------+---------+-----------------------------+-------+----------+-------------+
2 rows in set, 1 warning (0.00 sec)
```

- 부서 테이블에 먼저 접근 후 UI_부서명 index를 활용해 index full scan
- 상대적으로 큰 크기의 부서사원매핑 테이블은 I_부서번호 index로 index scan 수행(4만건의 행을 인덱스 스캔을 하고 랜덤 엑세스로 테이블에 접근)

SQL문 조인 설정 변경 튜닝

작은 테이블이 먼저 조인에 참여하는 나쁜 SQL문

```
mysql> SELECT COUNT(1) FROM 부서사원_매핑;
+----------+
| COUNT(1) |
+----------+
|   331603 |
+----------+
1 row in set (0.09 sec)

mysql> SELECT COUNT(1) FROM 부서;
+----------+
| COUNT(1) |
+----------+
|        9 |
+----------+
1 row in set (0.01 sec)

mysql> SELECT COUNT(1) FROM 부서사원_매핑 WHERE 시작일자>='2002-03-01';
+----------+
| COUNT(1) |
+----------+
|     1341 |
+----------+
1 row in set (0.08 sec)
```

- 드리븐 테이블에서 대량 데이터에 랜덤 액세스시 비효율적
- 부서사원_매핑 테이블에 30만 건 이상의 데이터를 MySQL 엔진으로 가져온 모든 데이터에 대해 where 절의 필터 조건(매핑.시작일자) 수행하기 때문에 **매핑.시작일자 조건절을 먼저 적용할 수 있다면 조인 시 비교 대상 줄어듬**

SQL문 조인 설정 변경 튜닝

작은 테이블이 먼저 조인에 참여하는 나쁜 SQL문

```
+--------------+--------------+
| 사원번호      | 부서번호      |
+--------------+--------------+
|        11732 | d009         |
|        14179 | d009         |
|        16989 | d009         |
|        20686 | d009         |
... 중략 ...
|       483592 | d007         |
|       487945 | d007         |
|       488117 | d007         |
+--------------+--------------+
1341 rows in set (0.09 sec)
```

```
SELECT /*+ JOIN_FIXED_ORDER () */
매핑.사원번호, 부서.부서번호
FROM 부서사원_매핑 매핑, 부서
WHERE 매핑.부서번호 = 부서.부서번호
AND 매핑.시작일자 >= '2002-03-01';
```

```
+----+-------------+--------+------------+--------+----------------+---------+---------+---------------------------+--------+----------+-------------+
| id | select_type | table  | partitions | type   | possible_keys  | key     | key_len | ref                       | rows   | filtered | Extra       |
+----+-------------+--------+------------+--------+----------------+---------+---------+---------------------------+--------+----------+-------------+
|  1 | SIMPLE      | 매핑   | NULL       | ALL    | I_부서번호     | NULL    | NULL    | NULL                      | 331143 |    33.33 | Using where |
|  1 | SIMPLE      | 부서   | NULL       | eq_ref | PRIMARY        | PRIMARY | 12      | tuning.매핑.부서번호      |      1 |   100.00 | Using index |
+----+-------------+--------+------------+--------+----------------+---------+---------+---------------------------+--------+----------+-------------+
2 rows in set, 1 warning (0.00 sec)
```

SQL문 조인 설정 변경 튜닝

작은 테이블이 먼저 조인에 참여하는 나쁜 SQL문

```
+--------------+--------------+
| 사원번호      | 부서번호      |
+--------------+--------------+
|        11732 | d009         |
|        14179 | d009         |
|        16989 | d009         |
|        20686 | d009         |
... 중략 ...
|       483592 | d007         |
|       487945 | d007         |
|       488117 | d007         |
+--------------+--------------+
1341 rows in set (0.09 sec)
```

```
SELECT  매핑.사원번호, 부서.부서번호
FROM 부서사원_매핑 매핑
LEFT JOIN 부서
On 매핑.부서번호 = 부서.부서번호
WHERE 매핑.시작일자 >= '2002-03-01';
```

```
+----+-------------+--------+------------+--------+---------------+---------+---------+---------------------------+--------+----------+-------------+
| id | select_type | table  | partitions | type   | possible_keys | key     | key_len | ref                       | rows   | filtered | Extra       |
+----+-------------+--------+------------+--------+---------------+---------+---------+---------------------------+--------+----------+-------------+
|  1 | SIMPLE      | 매핑   | NULL       | ALL    | NULL          | NULL    | NULL    | NULL                      | 331143 |    33.33 | Using where |
|  1 | SIMPLE      | 부서   | NULL       | eq_ref | PRIMARY       | PRIMARY | 12      | tuning.매핑.부서번호      |      1 |   100.00 | Using index |
+----+-------------+--------+------------+--------+---------------+---------+---------+---------------------------+--------+----------+-------------+
2 rows in set, 1 warning (0.01 sec)
```

SQL문 조인 설정 변경 튜닝

메인 테이블에 계속 의존하는 나쁜 SQL문

```
+--------------+---------------+------------------+
| 사원번호     | 이름          | 성               |
+--------------+---------------+------------------+
|       450025 | Dharmaraja    | Marrevee         |
|       450040 | Iara          | Falby            |
|       450044 | Steen         | Broder           |
... 중략 ...
|       499980 | Gino          | Usery            |
|       499986 | Nathan        | Ranta            |
|       499988 | Bangqing      | Kleiser          |
+--------------+---------------+------------------+
3155 rows in set (0.36 sec)
```

```
SELECT 사원.사원번호, 사원.이름, 사원.성
  FROM 사원
 WHERE 사원번호 > 450000
   AND ( SELECT MAX(연봉)
           FROM 급여
          WHERE 사원번호 = 사원.사원번호
       ) > 100000;
```

```
+----+--------------------+--------+------------+-------+---------------+---------+---------+---------------------------+--------+----------+-------------+
| id | select_type        | table  | partitions | type  | possible_keys | key     | key_len | ref                       | rows   | filtered | Extra       |
+----+--------------------+--------+------------+-------+---------------+---------+---------+---------------------------+--------+----------+-------------+
|  1 | PRIMARY            | 사원   | NULL       | range | PRIMARY       | PRIMARY | 4       | NULL                      | 104330 |   100.00 | Using where |
|  2 | DEPENDENT SUBQUERY | 급여   | NULL       | ref   | PRIMARY       | PRIMARY | 4       | tuning.사원.사원번호      |      9 |   100.00 | NULL        |
+----+--------------------+--------+------------+-------+---------------+---------+---------+---------------------------+--------+----------+-------------+
2 rows in set, 2 warnings (0.00 sec)
```

- 사원테이블은 primary key를 활용해서 index range scan 수행, 그 후 급여 테이블은 외부 테이블인 사원테이블로부터 조건을 전달받아 수행하는 의존 서브 쿼리(dependent subquery)로 수행

SQL문 조인 설정 변경 튜닝

메인 테이블에 계속 의존하는 나쁜 SQL문

```
mysql> SELECT COUNT(1) FROM 사원;
+----------+
| COUNT(1) |
+----------+
|   300024 |
+----------+
1 row in set (0.12 sec)

mysql> SELECT COUNT(1) FROM 급여;
+----------+
| COUNT(1) |
+----------+
|  2844047 |
+----------+
1 row in set (0.47 sec)

mysql> SELECT COUNT(1)
    -> FROM 사원
    -> WHERE 사원번호 > 450000;
+----------+
| COUNT(1) |
+----------+
|    49999 |
+----------+
1 row in set (0.03 sec)
```

```
mysql> show index from 사원;
+--------+------------+----------------+--------------+-------------+-----------+-------------+----------+---
| Table  | Non_unique | Key_name       | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Pac
+--------+------------+----------------+--------------+-------------+-----------+-------------+----------+---
| 사원   |          0 | PRIMARY        |            1 | 사원번호    | A         |      299512 |     NULL |   NULL
| 사원   |          1 | I_입사일자     |            1 | 입사일자    | A         |        4718 |     NULL |   NULL |
| 사원   |          1 | I_성별_성      |            1 | 성별        | A         |           1 |     NULL |   NULL
| 사원   |          1 | I_성별_성      |            2 | 성          | A         |        3360 |     NULL |   NULL
+--------+------------+----------------+--------------+-------------+-----------+-------------+----------+---
4 rows in set (0.04 sec)

mysql> show index from 급여;
+--------+------------+----------------+--------------+-------------+-----------+-------------+----------+---
| Table  | Non_unique | Key_name       | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Pac
+--------+------------+----------------+--------------+-------------+-----------+-------------+----------+---
| 급여   |          0 | PRIMARY        |            1 | 사원번호    | A         |      314970 |     NULL |   NULL
| 급여   |          0 | PRIMARY        |            2 | 시작일자    | A         |     2838731 |     NULL |   NULL
| 급여   |          1 | I_사용여부     |            1 | 사용여부    | A         |           1 |     NULL |   NULL |
+--------+------------+----------------+--------------+-------------+-----------+-------------+----------+---
3 rows in set (0.02 sec)
```

- select_type 항목에 DEPENDENT 키워드가 있으면, 외부 테이블에서 조건절을 받은 뒤 처리되어야 하므로 튜닝 대상으로 고려
- 외부 테이블인 사원 테이블의 사원 정보를 서브쿼리 대신 조인으로 변경 (조인이 서브쿼리보다 성능이 좋을 가능성이 높다.)

SQL문 조인 설정 변경 튜닝

메인 테이블에 계속 의존하는 나쁜 SQL문

```
+--------------+---------------+------------------+
| 사원번호     | 이름          | 성               |
+--------------+---------------+------------------+
|       450025 | Dharmaraja    | Marrevee         |
|       450040 | Iara          | Falby            |
|       450044 | Steen         | Broder           |
... 중략 ...
|       499980 | Gino          | Usery            |
|       499986 | Nathan        | Ranta            |
|       499988 | Bangqing      | Kleiser          |
+--------------+---------------+------------------+
3155 rows in set (0.17 sec)
```

```
SELECT 사원.사원번호, 사원.이름, 사원.성
  FROM 사원, 급여
 WHERE 사원.사원번호 > 450000
   AND 사원.사원번호 = 급여.사원번호
 GROUP BY 사원.사원번호
HAVING MAX(급여.연봉) > 100000;
```

```
+----+-------------+--------+------------+-------+-----------------------------------+---------+---------+--------------------------+--------+----------+-
| id | select_type | table  | partitions | type  | possible_keys                     | key     | key_len | ref                      | rows   | filtered | E
+----+-------------+--------+------------+-------+-----------------------------------+---------+---------+--------------------------+--------+----------+-
|  1 | SIMPLE      | 사원   | NULL       | range | PRIMARY,I_입사일자,I_성별_성      | PRIMARY | 4       | NULL                     | 104330 |   100.00 | Using
|  1 | SIMPLE      | 급여   | NULL       | ref   | PRIMARY                           | PRIMARY | 4       | tuning.사원.사원번호     |      9 |   100.00 | NULL
+----+-------------+--------+------------+-------+-----------------------------------+---------+---------+--------------------------+--------+----------+-
2 rows in set, 1 warning (0.01 sec)
```

- 드라이빙 테이블은 급여 테이블, 드리븐 테이블은 사원 테이블
- 급여 테이블에 먼저 접근하기 위한 범위 축소 조건은 사원.사원번호 > 450000 절을 통해 급여.사원번호 > 450000 조건절로 변형되어 적용
- DEPENDENT 방식은 삭제되고, 사원 테이블과 급여 테이블이 조인하는 방식으로 변경

SQL문 조인 설정 변경 튜닝

불필요한 조인을 수행하는 나쁜 SQL문

```
SELECT COUNT(DISTINCT 사원.사원번호) as 데이터건수
  FROM 사원,
       ( SELECT 사원번호
           FROM 사원출입기록 기록
          WHERE 출입문 = 'A'
       ) 기록
 WHERE 사원.사원번호 = 기록.사원번호;
```

```
+-----------------+
| 데이터건수      |
+-----------------+
|          150000 |
+-----------------+
1 row in set (0.43 sec)
```

```
+----+-------------+--------+------------+--------+---------------+------------+---------+---------------------------+--------+----------+-------------+
| id | select_type | table  | partitions | type   | possible_keys | key        | key_len | ref                       | rows   | filtered | Extra       |
+----+-------------+--------+------------+--------+---------------+------------+---------+---------------------------+--------+----------+-------------+
|  1 | SIMPLE      | 기록   | NULL       | ref    | I_출입문      | I_출입문   | 4       | const                     | 329467 |   100.00 | Using index |
|  1 | SIMPLE      | 사원   | NULL       | eq_ref | PRIMARY       | PRIMARY    | 4       | tuning.기록.사원번호      |      1 |   100.00 | Using index |
+----+-------------+--------+------------+--------+---------------+------------+---------+---------------------------+--------+----------+-------------+
2 rows in set, 1 warning (0.00 sec)
```

- 사원출입기록 테이블은 값이 'A'인 상수와 직접 비교하므로 ref 항목이 const, 인덱스를 사용한 동등 비교 수행하므로 type 항목이 ref로 표시
- 사원 테이블은 primary key를 사용해서 조인 조건절인 사원번호 컬럼으로 데이터 비교하기 때문에 type 항목에 eq_ref로 표시

SQL문 조인 설정 변경 튜닝

불필요한 조인을 수행하는 나쁜 SQL문

```
mysql> DESC 사원;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| 사원번호     | int           | NO   | PRI | NULL    |       |
| 생년월일     | date          | NO   |     | NULL    |       |
| 이름         | varchar(14)   | NO   |     | NULL    |       |
| 성           | varchar(16)   | NO   |     | NULL    |       |
| 성별         | enum('M','F') | NO   | MUL | NULL    |       |
| 입사일자     | date          | NO   | MUL | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

- from 절의 인라인 뷰는 옵티마이저에 의해 view merging으로 최적화 되어 위의 SQL문처럼 수행됨
- select 절의 최종 결과는 사원번호에서 중복을 제거한 건수만 알면 됨

=> 사원출입기록의 사원번호는 중복제거를 하기 보단 사원 테이블과 조인 과정 중 값의 존재 여부만 확인해도 됨

=> **EXISTS 사용 가능**

SQL문 조인 설정 변경 튜닝

불필요한 조인을 수행하는 나쁜 SQL문

```
SELECT COUNT(1) as 데이터건수
  FROM 사원
 WHERE EXISTS (SELECT 1
                 FROM 사원출입기록 기록
                WHERE 출입문 = 'A'
                AND 기록.사원번호 = 사원.사원번호);
```

```
+-----------------+
| 데이터건수      |
+-----------------+
|          150000 |
+-----------------+
1 row in set (0.14 sec)
```

```
+----+--------------+-------------+------------+-------+---------------------+---------------------+---------+---------------------------+--------+----------
| id | select_type  | table       | partitions | type  | possible_keys       | key                 | key_len | ref                       | rows   | filtered
+----+--------------+-------------+------------+-------+---------------------+---------------------+---------+---------------------------+--------+----------
|  1 | SIMPLE       | 사원        | NULL       | index | PRIMARY             | I_입사일자          | 3       | NULL                      | 299512 |   100.00 | Us
|  1 | SIMPLE       | <subquery2> | NULL       | eq_ref | <auto_distinct_key> | <auto_distinct_key> | 4       | tuning.사원.사원번호      |      1 |   100.00 | NU
|  2 | MATERIALIZED | 기록        | NULL       | ref   | I_출입문            | I_출입문            | 4       | const                     | 329467 |   100.00 | Usi
+----+--------------+-------------+------------+-------+---------------------+---------------------+---------+---------------------------+--------+----------
3 rows in set, 2 warnings (0.00 sec)
```

- 사원출입기록 테이블의 데이터는 최종결과에 사용하지 않고 존재 여부만 파악하면 되므로 EXISTS 구문으로 변경
- 실행계획을 보면, id가 1인 사원 테이블은 드라이빙 테이블이고, <subquery2>는 id = 2인 사원출력기록 테이블
- 사원출력기록 테이블은 EXISTS 연산자로 데이터 존재 여부를 파악하기 위해 임시 테이블을 생성하는 MATERIALIZED로 표기

SQL문 재작성으로 착한 쿼리 만들기

SQL문 재작성으로 착한 쿼리 만들기

처음부터 모든 데이터를 가져오는 나쁜 SQL문

```
SELECT 사원.사원번호,
       급여.평균연봉,
       급여.최고연봉,
       급여.최저연봉
 FROM 사원,
      ( SELECT 사원번호,
          ROUND(AVG(연봉),0) 평균연봉,
               ROUND(MAX(연봉),0) 최고연봉,
               ROUND(MIN(연봉),0) 최저연봉
         FROM 급여
         GROUP BY 사원번호
      ) 급여
WHERE 사원.사원번호 = 급여.사원번호
AND 사원.사원번호 BETWEEN 10001 AND 10100;
```

```
+----------+----------+----------+----------+
| 사원번호 | 평균연봉 | 최고연봉 | 최저연봉 |
+----------+----------+----------+----------+
|    10001 |    75389 |    88958 |    60117 |
|    10002 |    68855 |    72527 |    65828 |
| …        | ….       | …        |          |
|    10099 |    83902 |    98538 |    68781 |
|    10100 |    64537 |    74957 |    54398 |
+----------+----------+----------+----------+

100 rows in set (1.08 sec)
```

```
+----+-------------+------------+------------+-------+-----------------------+-------------+---------+--------------------------+---------+----------+-----
| id | select_type | table      | partitions | type  | possible_keys         | key         | key_len | ref                      | rows    | filtered | Extra
+----+-------------+------------+------------+-------+-----------------------+-------------+---------+--------------------------+---------+----------+-----
|  1 | PRIMARY     | 사원       | NULL       | range | PRIMARY               | PRIMARY     | 4       | NULL                     |     100 |   100.00 | Using
|  1 | PRIMARY     | <derived2> | NULL       | ref   | <auto_key0>           | <auto_key0> | 4       | tuning.사원.사원번호     |      10 |   100.00 | NULL
|  2 | DERIVED     | 급여       | NULL       | index | PRIMARY,I_사용여부    | PRIMARY     | 7       | NULL                     | 2838731 |   100.00 | NULL
+----+-------------+------------+------------+-------+-----------------------+-------------+---------+--------------------------+---------+----------+-----
3 rows in set, 1 warning (0.01 sec)
```

- 사원테이블과 인라인뷰가 join 되고있다.
- 급여테이블이 인덱스 풀 스캔을 하고 있다.

SQL문 재작성으로 착한 쿼리 만들기

처음부터 모든 데이터를 가져오는 나쁜 SQL문

```
mysql> SELECT COUNT(1) FROM 급여;
+----------+
| COUNT(1) |
+----------+
|  2844047 |
+----------+
1 row in set (0.17 sec)

mysql> SELECT COUNT(1) FROM 사원;
+----------+
| COUNT(1) |
+----------+
|   300024 |
+----------+
1 row in set (0.11 sec)
```

- 사원 정보는 30만 중 100개만 가지고 수행
- 급여 정보는 거의 완전히 사용되고 있음
- 필요한 급여정보만을 가져온다면

SQL문 재작성으로 착한 쿼리 만들기

처음부터 모든 데이터를 가져오는 나쁜 SQL문

```
SELECT 사원.사원번호,
       ROUND(AVG(연봉),0) 평균연봉,
       ROUND(MAX(연봉),0) 최고연봉,
       ROUND(MIN(연봉),0) 최저연봉
 FROM 사원, 급여
WHERE 사원.사원번호 = 급여.사원번호
AND 사원.사원번호 BETWEEN 10001 AND 10100
GROUP BY 사원번호;
```

```
+----------+----------+----------+----------+
| 사원번호 | 평균연봉 | 최고연봉 | 최저연봉 |
+----------+----------+----------+----------+
|    10001 |    75389 |    88958 |    60117 |
|    10002 |    68855 |    72527 |    65828 |
…  ….  …
|    10099 |    83902 |    98538 |    68781 |
|    10100 |    64537 |    74957 |    54398 |
+----------+----------+----------+----------+

100 rows in set (0.01 sec)
```

```
+----+-------------+--------+------------+-------+---------------------------------------+---------+---------+---------------------------+------+----------+----
| id | select_type | table  | partitions | type  | possible_keys                         | key     | key_len | ref                       | rows | filtered | Ext
+----+-------------+--------+------------+-------+---------------------------------------+---------+---------+---------------------------+------+----------+----
|  1 | SIMPLE      | 사원   | NULL       | range | PRIMARY,I_입사일자,I_성별_성          | PRIMARY | 4       | NULL                      |  100 |   100.00 | Using wh
|  1 | SIMPLE      | 급여   | NULL       | ref   | PRIMARY                               | PRIMARY | 4       | tuning.사원.사원번호      |    9 |   100.00 | NULL
+----+-------------+--------+------------+-------+---------------------------------------+---------+---------+---------------------------+------+----------+----
2 rows in set, 1 warning (0.01 sec)
```

- 인라인뷰 대신 group by를 밖으로

SQL문 재작성으로 착한 쿼리 만들기

처음부터 모든 데이터를 가져오는 나쁜 SQL문

```
+----------+----------+----------+----------+
| 사원번호 | 평균연봉 | 최고연봉 | 최저연봉 |
+----------+----------+----------+----------+
|    10001 |    75389 |    88958 |    60117 |
|    10002 |    68855 |    72527 |    65828 |
| …        | ….       | …        |          |
|    10099 |    83902 |    98538 |    68781 |
|    10100 |    64537 |    74957 |    54398 |
+----------+----------+----------+----------+

100 rows in set (0.01 sec)
```

```
SELECT 사원.사원번호,
    ( SELECT ROUND(AVG(연봉),0)
      FROM 급여 as 급여1
      WHERE 사원번호 = 사원.사원번호
    ) AS 평균연봉,
    ( SELECT ROUND(MAX(연봉),0)
      FROM 급여 as 급여2
      WHERE 사원번호 = 사원.사원번호
    ) AS 최고연봉,
    ( SELECT ROUND(MIN(연봉),0)
      FROM 급여 as 급여3
      WHERE 사원번호 = 사원.사원번호
    ) AS 최저연봉
  FROM 사원
 WHERE 사원.사원번호 BETWEEN 10001 AND 10100;
```

```
+----+--------------------+--------+------------+-------+---------------------------------------+---------+---------+--------------------------+------+-------
+-------------------------+
| id | select_type        | table  | partitions | type  | possible_keys                         | key     | key_len | ref                      | rows | filter
|
+----+--------------------+--------+------------+-------+---------------------------------------+---------+---------+--------------------------+------+-------
+-------------------------+
|  1 | PRIMARY            | 사원   | NULL       | range | PRIMARY,I_입사일자,I_성별_성          | PRIMARY | 4       | NULL                     |  100 |   100.00 |
|  4 | DEPENDENT SUBQUERY | 급여3  | NULL       | ref   | PRIMARY                               | PRIMARY | 4       | tuning.사원.사원번호     |    9 |   100.00 |
|  3 | DEPENDENT SUBQUERY | 급여2  | NULL       | ref   | PRIMARY                               | PRIMARY | 4       | tuning.사원.사원번호     |    9 |   100.00 |
|  2 | DEPENDENT SUBQUERY | 급여1  | NULL       | ref   | PRIMARY                               | PRIMARY | 4       | tuning.사원.사원번호     |    9 |   100.00 |
+----+--------------------+--------+------------+-------+---------------------------------------+---------+---------+--------------------------+------+-------
+-------------------------+
4 rows in set, 4 warnings (0.00 sec)
```

- 인라인뷰 대신 사원번호마다 평균연봉, 최고연봉, 최저연봉을 계산

SQL문 재작성으로 착한 쿼리 만들기

대량의 데이터를 가져와 조인하는 나쁜 SQL문

```
SELECT DISTINCT 매핑.부서번호
        FROM 부서관리자 관리자,
             부서사원_매핑 매핑
       WHERE 관리자.부서번호 = 매핑.부서번호
       ORDER BY 매핑.부서번호;
```

```
+--------------+
| 부서번호     |
+--------------+
| d001         |
| d002         |
| d003         |
| d004         |
| d005         |
| d006         |
| d007         |
| d008         |
| d009         |
+--------------+
9 rows in set (0.29 sec)
```

```
+----+-------------+----------+------------+-------+-----------------------+---------------+---------+---------------------------+--------+----------+----
| id | select_type | table    | partitions | type  | possible_keys         | key           | key_len | ref                       | rows   | filtered | Extr
+----+-------------+----------+------------+-------+-----------------------+---------------+---------+---------------------------+--------+----------+----
|  1 | SIMPLE      | 매핑     | NULL       | index | PRIMARY,I_부서번호    | I_부서번호    | 12      | NULL                      | 331143 |   100.00 | Using ind
|  1 | SIMPLE      | 관리자   | NULL       | ref   | I_부서번호            | I_부서번호    | 12      | tuning.매핑.부서번호      |      2 |   100.00 | Using index;
+----+-------------+----------+------------+-------+-----------------------+---------------+---------+---------------------------+--------+----------+----
2 rows in set, 1 warning (0.00 sec)
```

- 부서관리자 테이블과 부서사원_매핑 테이블을 부서번호로 조인
- 중복제거
- 정렬

SQL문 재작성으로 착한 쿼리 만들기

대량의 데이터를 가져와 조인하는 나쁜 SQL문

```
SELECT 매핑.부서번호
  FROM ( SELECT DISTINCT 부서번호
           FROM 부서사원_매핑 매핑
       ) 매핑
 WHERE EXISTS (SELECT 1
                 FROM 부서관리자 관리자
                WHERE 부서번호 = 매핑.부서번호)
 ORDER BY 매핑.부서번호;
```

```
+--------------+
| 부서번호     |
+--------------+
| d001         |
| d002         |
| d003         |
| d004         |
| d005         |
| d006         |
| d007         |
| d008         |
| d009         |
+--------------+
9 rows in set (0.01 sec)
```

```
+----+-------------+-----------+------------+-------+-----------------------+---------------+---------+-------------------------------+------+----------+---
| id | select_type | table     | partitions | type  | possible_keys         | key           | key_len | ref                           | rows | filtered | Ex
+----+-------------+-----------+------------+-------+-----------------------+---------------+---------+-------------------------------+------+----------+---
|  1 | PRIMARY     | 관리자    | NULL       | index | I_부서번호            | I_부서번호    | 12      | NULL                          |   24 |    37.50 | Using in
|  1 | PRIMARY     | <derived2> | NULL       | ref   | <auto_key0>           | <auto_key0>   | 12      | tuning.관리자.부서번호        |    2 |   100.00 | Using
|  2 | DERIVED     | 매핑       | NULL       | range | PRIMARY,I_부서번호    | I_부서번호    | 12      | NULL                          |    9 |   100.00 | Using in
+----+-------------+-----------+------------+-------+-----------------------+---------------+---------+-------------------------------+------+----------+---
3 rows in set, 2 warnings (0.01 sec)
```

- 미리 중복을 제거
- 조인하지 않고 있는지 검사