사용 방법

```
EXPLAIN [EXTENDED] SELECT ... FROM ... WHERE ...
```

MySQL Explain 결과의 각 항목 별 의미

구분	설명
id	select 아이디로 SELECT를 구분하는 번호
table	참조하는 테이블
select_type	select에 대한 타입
type	조인 혹은 조회 타입
possible_keys	데이터를 조회할 때 DB에서 사용할 수 있는 인덱스 리스트
key	실제로 사용할 인덱스
key_len	실제로 사용할 인덱스의 길이
ref	Key 안의 인덱스와 비교하는 컬럼(상수)
rows	쿼리 실행 시 조사하는 행 수립
extra	추가 정보

id

행이 어떤 SELECT 구문을 나타내는 지를 알려주는 것으로 구문에 서브 쿼리나 UNION이 없다면 SELECT는 하나밖에 없기 때문에 모든 행에 대해 1이란 값이 부여되지만 이외의 경우에는 원 구문에서 순서에 따라 각 SELECT 구문들에 순차적으로 번호가 부여된다.

table

행이 어떤 테이블에 접근하는 지를 보여주는 것으로 대부분의 경우 테이블 이름이나 SQL에서 지정된 별명 같은 값을 나타낸다.

select_type

구분	설명
SIMPLE	단순 SELECT (Union 이나 Sub Query 가 없는 SELECT 문)
PRIMARY	Sub Query를 사용할 경우 Sub Query의 외부에 있는 쿼리(첫번째 쿼리), UNION 을 사용할 경우 UNION의 첫 번째 SELECT 쿼리
UNION	UNION 쿼리에서 Primary를 제외한 나머지 SELECT
DEPENDENT_UNION	UNION 과 동일하나, 외부쿼리에 의존적임 (값을 공급 받음)
UNION_RESULT	UNION 쿼리의 결과물
SUBQUERY	Sub Query 또는 Sub Query를 구성하는 여러 쿼리 중 첫 번째 SELECT문
DEPENDENT_SUBQUERY	Sub Query 와 동일하나, 외곽쿼리에 의존적임 (값을 공급 받음)
DERIVED	SELECT로 추출된 테이블 (FROM 절 에서의 서브쿼리 또는 Inline View)
UNCACHEABLE SUBQUERY	Sub Query와 동일하지만 공급되는 모든 값에 대해 Sub Query를 재처리. 외부쿼리에서 공급되는 값이 동이라더라도 Cache된 결과를 사용할 수 없음
UNCACHEABLE UNION	UNION 과 동일하지만 공급되는 모든 값에 대하여 UNION 쿼리를 재처리

type

구분	설명
system	테이블에 단 한개의 데이터만 있는 경우
const	SELECT에서 Primary Key 혹은 Unique Key를 ~~상수~~ 상수로 조회하는 경우로 많아야 한 건의 데이터만 있음
eq_ref	조인을 할 때 Primary Key
ref	조인을 할 때 Primary Key 혹은 Unique Key가 아닌 Key로 매칭하는 경우
ref_or_null	ref 와 같지만 null 이 추가되어 검색되는 경우
index_merge	두 개의 인덱스가 병합되어 검색이 이루어지는 경우
unique_subquery	다음과 같이 IN 절 안의 서브쿼리에서 Primary Key가 오는 특수한 경우 SELECT * FROM tab01 WHERE col01 IN (SELECT Primary Key FROM tab01);
index_subquery	unique_subquery와 비슷하나 Primary Key가 아닌 인덱스인 경우 SELECT * FROM tab01 WHERE col01 IN (SELECT key01 FROM tab02);
range	특정 범위 내에서 인덱스를 사용하여 원하는 데이터를 추출하는 경우로, 데이터가 방대하지 않다면 단순 SELECT 에서는 나쁘지 않음
index	인덱스를 처음부터 끝까지 찾아서 검색하는 경우로, 일반적으로 인덱스 풀스캔이라고 함
all	테이블을 처음부터 끝까지 검색하는 경우로, 일반적으로 테이블 풀스캔이라고 함

possible_keys

쿼리에서 접근하는 컬럼 들과 사용된 비교 연산자들을 바탕으로 어떤 인덱스를 사용할 수 있는지를 표시해준다.

key

테이블에 접근하는 방법을 최적화 하기 위해 어떤 인덱스를 사용하기로 결정했는 지를 나타낸다.

key_len

MySQL이 인덱스에 얼마나 많은 바이트를 사용하고 있는 지를 보여준다. MySQL에서 인덱스에 있는 컬럼들 중 일부만 사용한다면 이 값을 통해 어떤 컬럼들이 사용되는 지를 계산할 수 있다.

ref

키 컬럼에 나와 있는 인덱스에서 값을 찾기 위해 선행 테이블의 어떤 컬럼이 사용되었는 지를 나타낸다.

rows

Oracle 실행계획의 'card'와 흡사함.

원하는 행을 찾기 위해 얼마나 많은 행을 읽어야 할 지에 대한 예측값을 의미한다.

extra

구분	설명
using index	커버링 인덱스라고 하며 인덱스 자료 구조를 이용해서 데이터를 추출
using where	where 조건으로 데이터를 추출. type이 ALL 혹은 Indx 타입과 함께 표현되면 성능이 좋지 않다는 의미
using filesort	데이터 정렬이 필요한 경우로 메모리 혹은 디스크상에서의 정렬을 모두 포함. 결과 데이터가 많은 경우 성능에 직접적인 영향을 줌
using temporary	쿼리 처리 시 내부적으로 temporary table이 사용되는 경우를 의미함

MySQL Explain 상 일반적으로 데이터가 많은 경우 Using Filesort 와 Using Temporary 상태는 좋지 않으며 쿼리 튜닝 후 모니터링이 필요하다.

MySQL 5.6 Command Line Client

```
mysql> EXPLAIN SELECT zc1.id
    -> FROM zipcode zc1, zipcode2 zc2
    -> WHERE zc1.id = zc2.id;
+----+-------------+-------+--------+---------------+---------+---------+-----------------+--------+-------------+
| id | select_type | table | type   | possible_keys | key     | key_len | ref             | rows   | Extra       |
+----+-------------+-------+--------+---------------+---------+---------+-----------------+--------+-------------+
|  1 | SIMPLE      | zc2   | index  | PRIMARY       | PRIMARY | 4       | NULL            | 580203 | Using index |
|  1 | SIMPLE      | zc1   | eq_ref | PRIMARY       | PRIMARY | 4       | posting2.zc2.id |      1 | Using index |
+----+-------------+-------+--------+---------------+---------+---------+-----------------+--------+-------------+
2 rows in set (0.00 sec)

mysql>
```