I limiti dell'ottimizzatore

- E' possibile che l'ottimizzatore NON determini il migliore piano di esecuzione.
- Le istruzioni SQL devono essere ottimizzate da chi le scrive e non ci si deve affidare all'ottimizzatore.

Il comando EXPLAIN (1)

- Permette di visualizzare il piano di esecuzione di una SELECT.
- MySql non permette di visualizzare il piano di esecuzione di una DML o di una DDL.

EXPLAIN: le informazioni (1)

- Una query è composta da n passi, ossia da n “query elementari”.
- Data una query il comando EXPLAIN ritorna n righe, una per ogni passo da effettuare per portare a termine la query stessa.
- Ogni riga contiene le informazioni di dettaglio su come il singolo passo viene eseguito.

EXPLAIN: ID

- Identificativo (progressivo) della select all’interno dell’intera query.

EXPLAIN: SELECT_TYPE (1)

- Identifica il ruolo del singolo passo, ossia la sua funzione all'interno della query principale.

- SIMPLE

  query semplice (che non usa UNION o subquery).

- PRIMARY

  query più esterna.

EXPLAIN: SELECT_TYPE (3

- UNION

  seconda (o successiva) select in una UNION.
- DEPENDENT UNION

  seconda (o successiva) select in una UNION, dipendente da una query esterna.
- UNION RESULT

  il risultato di una UNION.

EXPLAIN: SELECT_TYPE (

- SUBQUERY

  subquery “semplice”.
- DEPENDENT SUBQUERY

  subquery dipendente da una query esterna.
- DERIVED

  subquery in una FROM.

```
EXPLAIN
SELECT * FROM emp
WHERE empno=2\G
```

```
*********** 1. row *****
           id: 1
  select_type: SIMPLE
        table: EMP
         type: const
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: const
         rows: 1
        Extra:
```

```
EXPLAIN SELECT * FROM emp
WHERE
deptno=
(SELECT max(deptno) FROM dept)\G
```

```
*********** 1. row *****
           id: 1
  select_type: PRIMARY
        table: emp
...
*********** 2. row *****
           id: 2
  select_type: SUBQUERY
        table: NULL
...
```

```
EXPLAIN
SELECT * FROM emp
UNION ALL
SELECT * FROM manager \G
```

```
*********** 1. row ******
           id: 1
  select_type: PRIMARY
        table: emp
...
*********** 2. row ******
           id: 2
  select_type: UNION
        table: manager
*********** 3. row ******
           id: NULL
  select_type: UNION RESULT
        table: <union1,2>
```

```
EXPLAIN SELECT * FROM emp WHERE
deptno=(SELECT max(deptno)  FROM dept)\G
```

```
*************** 1. row *****
           id: 1
  select_type: PRIMARY
        table: emp
...
*************** 2. row *****
           id: 2
  select_type: SUBQUERY
        table: dept
...
```

EXPLAIN: TYPE (1)

- Identifica il tipo di accesso effettuato sulla tabella del passo n-esimo, ossia informazioni di dettaglio su come la tabella viene acceduta.

- ALL

  si accede a tutte le righe di una tabella, ossia in FULL TABLE SCAN.

- INDEX

  si accede a tutte le chiavi di un indice, ossia in FULL INDEX SCAN.

- RANGE

  si accede alla tabella tramite un indice per un “range” di valori possibili.

- CONST

  una sola riga soddisfa le condizioni. La tabella viene acceduta una sola volta all’interno dell’intera query.

- EQ_REF

  la tabella ha una “matching-row” per ogni riga ritornata al passo precedente (quando in un join la tabella viene acceduta tramite una PK).

- REF

  la tabella ha più “matching-row” per ogni riga ritornata al passo precedente, (quando in un join la tabella viene acceduta tramite un indice non univoco).

- INDEX_MERGE

  si accede alla tabella utilizzando più indici e poi si effettua il “merge” degli stessi per recuperare le righe richieste.

  E’ utilizzato quando all’interno della condizione di WHERE ci sono delle espressioni in AND/OR.

```
EXPLAIN
SELECT distinct(ename) FROM emp \G

*********** 1. row *****
           id: 1
  select_type: SIMPLE
        table: emp
         type: ALL
possible_keys: NULL
          key: NULL
...
```

```
EXPLAIN
SELECT * FROM emp
WHERE empno BETWEEN 10 AND 100 \G

*************** 1. row *****
           id: 1
  select_type: SIMPLE
        table: EMP
         type: range
possible_keys: PRIMARY
          key: PRIMARY
...
```

```
EXPLAIN
SELECT * FROM emp WHERE empno=2\G

*********** 1. row *****
           id: 1
  select_type: SIMPLE
        table: EMP
         type: const
possible_keys: PRIMARY
          key: PRIMARY
...
```

```
EXPLAIN
SELECT * FROM emp
/*! FORCE INDEX (PRIMARY,fk_emp_dept) */
WHERE empno=10 or deptno=20 \G
```

```
*********** 1. row *****
           id: 1
  select_type: SIMPLE
        table: emp
         type: index_merge
possible_keys: PRIMARY,fk_emp_dept
          key: PRIMARY,fk_emp_dept
      key_len: 4,5
          ref: NULL
         rows: 2
        Extra: Using union...
```

EXPLAIN: POSSIBLE_KEYS/KEY

- POSSIBLE_KEYS

  La lista degli indici che possono essere utilizzati per portare a termine la singola query.

- KEY

  La lista degli indici effettivamente utilizzati per portare a termine la singola query.

```
EXPLAIN
SELECT * FROM emp
WHERE empno=10 or deptno=20 \G
```

```
*********** 1. row *****
           id: 1
  select_type: SIMPLE
        Table: emp
         type: ALL
possible_keys: PRIMARY,fk_emp_dept
          key: NULL
      key_len: NULL
          ref: NULL
...
```

EXPLAIN: ROWS

- ROWS

  il numero di righe che mysql “crede” di dover esaminare per eseguire la query.

EXPLAIN: EXTRA (1)

- Contiene informazioni di dettaglio su come viene eseguito il singolo passo.

- USING FILESORT

  Viene utilizzato un file temporaneo per effettuare un ordinamento.

- USING TEMPORARY

  Viene utilizzata una tabella temporanea per memorizzare i risultati parziali.

- USING INDEX

  Viene utilizzato il solo indice per recuperare le informazioni richieste. Non è necessario accedere alla tabella.

- USING WHERE

  Viene utilizzata una condizione di where come filtro per restituire le sole righe richieste.

```
EXPLAIN
SELECT * FROM emp ORDER BY ename\G
```

```
*************** 1. row *****
           id: 1
  select_type: SIMPLE
        table: emp
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 3
        Extra: Using filesort
```

```
EXPLAIN
SELECT DISTINCT ename
FROM emp WHERE deptno=10\G
```

```
*********** 1. row *****
           id: 1
  select_type: SIMPLE
        table: emp
         type: ref
possible_keys: fk_emp_dept
          key: fk_emp_dept
      key_len: 5
          ref: const
         rows: 1
        Extra: Using where; Using
  temporary
```

```
EXPLAIN
SELECT deptno FROM emp
WHERE deptno >= 10\G
```

```
*********** 1. row *****
           id: 1
  select_type: SIMPLE
        Table: emp
         type: index
possible_keys: fk_emp_dept
          key: fk_emp_dept
      key_len: 5
          ref: NULL
         rows: 3
        Extra: Using where; Using index
```

Tuning delle istruzioni SQL

- Scrivere istruzioni SQL “semplici”.
- Verificare i piani di esecuzione.
- Ottimizzare l’accesso alla singola tabella.
- Ottimizzare l’ordine dei passi.
- Riscrivere la query.