Ottimizzare le Queries - Explain

- EXPLAIN [EXTENDED] SELECT ....
- Metodo empirico di ottimizzazione delle query e delle tabelle.
- Dà statistiche sul numero di righe visionate per ogni select, join, sub select e union
- Funziona solo su query che ritornano risultati

Explain - Esempi

EXPLAIN SELECT Country.Name FROM Country ;

```
+----+-------------+---------+--------+----------------+--------+-----------+------+------+-------+
| id | select_type | table   | type   | possible_keys  | key    | key_len   | ref  | rows | Extra |
+----+-------------+---------+--------+----------------+--------+-----------+------+------+-------+
|  1 | SIMPLE      | Country | ALL    | NULL           | NULL   | NULL      | NULL |  239 |       |
+----+-------------+---------+--------+----------------+--------+-----------+------+------+-------+
```

EXPLAIN SELECT Country.Code FROM Country ;

```
+----+-------------+---------+-------+---------------+---------+---------+------+------+-------------+
| id | select_type | table   | type  | possible_keys | key     | key_len | ref  | rows | Extra       |
+----+-------------+---------+-------+---------------+---------+---------+------+------+-------------+
|  1 | SIMPLE      | Country | index | NULL          | PRIMARY | 3       | NULL |  239 | Using index |
+----+-------------+---------+-------+---------------+---------+---------+------+------+-------------+
```

Explain - Output

I campi:

table - ci mostra quale da tabella considerata al momento (join)

type - ci dice che tipo di join viene usato:
dal migliore al peggiore: system, const, eq_ref, ref, range, index, all

possible_keys - ci indica quali possibili indici vengono applicati alla tabella.

key - quale indice e' usato

key_len - la lunghezza della chiave. Più piccola è meglio è.

ref - ci dice quale colonna o costante è usata

rows - numero di righe che MySQL ha esaminato per ottenere i dati

Extra - altre informazioni riguardo la query: (le peggiori sono "using temporary" and "using filesort")

Ottimizzare le Queries – Explain cont.

- E' utile per:
  - Ottimizzare gli indici.
  - Ottimizzare i tipi di variabile.
  - Ottimizzare le proprie query

Explain – Ottimizzare gli Indici

- Capiamo l'importanza degli indici e delle chiavi con EXPLAIN.
- Proviamo a togliere le chiavi a World.City e lanciamo una semplice SELECT.
- ALTER TABLE `World`.`City` MODIFY COLUMN `ID` INTEGER NOT NULL,
  DROP PRIMARY KEY;
- EXPLAIN SELECT ID FROM City WHERE ID=1

Explain – Chiavi Primarie

- ALTER TABLE `World`.`City` MODIFY COLUMN `ID` INTEGER NOT NULL AUTO_INCREMENT, ADD PRIMARY KEY(`ID`);
- EXPLAIN SELECT ID FROM City WHERE ID=1
- Ora viene visionata una sola riga

Explain - Indici

EXPLAIN Select Name from City
where Name='New York'

- Le righe visionate sono tutte.
- Indicizzando:
  - ALTER TABLE `World`.`City` ADD INDEX `new_index` (`Name`);

Explain - Indici Numerici

- EXPLAIN SELECT * FROM City C where C.`Population`/2 > 500000
- Se indicizzo Population la precedente query non si avvantaggerà del'indice.
- EXPLAIN SELECT * FROM City C where C.`Population` > 500000*2

Explain – Ottimizzare i Tipi

- La comparazione tra Campi di tipo o lunghezza diversi non permette le ottimizzazioni.
- Se conosciamo i campi che devono essere comparati, diamo loro tipo uguale e/o lunghezza uguale.

Explain – Ottimizzare le Query

- Riscrivere SubQuery come Inner Join
- EXPLAIN SELECT Country.Name FROM Country WHERE Country.Code IN (SELECT CountryCode FROM City);
- EXPLAIN SELECT DISTINCT Country.Name FROM Country,City WHERE Country.Code=CountryCode;

Explain – Ottimizzare le Query 2

- Riscrivere SubQuery come Outer Join
- EXPLAIN SELECT Country.Name FROM Country LEFT JOIN City On Country.Code=CountryCode WHERE CountryCode IS NULL;
- EXPLAIN SELECT Country.Name FROM Country WHERE Country.Code NOT IN (SELECT CountryCode FROM City);

EXPLAIN - Subquery

- Le SubQuery possono performare meglio?
- **EXPLAIN SELECT** Country.Code,Country.Name **FROM** Country **WHERE** Country.Code **IN** ('ITA','DEN','USA') **AND EXISTS** (**SELECT NULL FROM** City **WHERE** City.CountryCode>'ITA' **AND** Code=CountryCode) **ORDER BY** Country.Name;
- **EXPLAIN SELECT DISTINCT** Country.Code,Country.Name **FROM** Country,City **WHERE** Country.Code **IN** ('ITA','DEN','USA') **AND** City.CountryCode>'ITA' **AND** Code=CountryCode **ORDER BY** Country.Name;

Ottimizzare le Tabelle

- Analizza e riscrive la distribuzione delle chiavi:
  - ANALYZE TABLE tablename;
- Ricompatta (deframmenta) la tabella:
  - OPTIMIZE TABLE tablename

Le Variabili in MySQL

- Le variabili sono utulizzate nel seguente modo:
- SET @v:='valore'; -- <- assegnare un valore
- SELECT @v:=Campo FROM tabella; -- <- i.c.s
- SELECT @v; -- <- visualizzarlo

Caching e Prepared Statement

- MySQL e gli altri DBMS possono velocizzare le query attraverso il **caching**.
- Diamo due volte una stessa query e vedremo che la seconda volta ci metterà meno tempo.
- Il massimo del caching si ha attraverso l'utilizzo dei **prepared statement**.

Prepared Statement

- Sintassi:
- PREPARE stmt_name FROM preparable_stmt;
- EXECUTE stmt_name [USING @var_name [, @var_name] ...];
- DROP PREPARE stmt_name;

Prepared Statements

- PREPARE stmt1 FROM 'SELECT SQRT(POW(?,2) + POW(?,2)) AS hypotenuse';
- SET @a = 3;
- SET @b = 4;
- EXECUTE stmt1 USING @a, @b;
- **Vantaggi:**
  - **Sono sempre cachati.**
  - **Non si devono riscrivere.**
  - **Sono più 'sicuri' negli applicativi.**

InnoDB – Applichiamo le FK

- [CONSTRAINT symbol] FOREIGN KEY [id]
  (index_col_name, ...) REFERENCES tbl_name
  (index_col_name, ...) [ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}] [ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]

- Entrambe le tabelle devono essere tabelle InnoDB e non devono essere temporary tables.
- Nella tabella referenziata le colonne definite nella FK devono essere indicizzate nello stesso ordine
- I tipi BLOB e TEXT non possono essere FK.
- Il simbolo CONSTRAINT deve essere unico nel DB. Se non viene esplicitato viene creato automaticamente

FK - Azioni

- CASCADE: Cancella o aggiorna la riga della tabella genitore e automaticamente cancella o aggiorna le righe nella tabella figlio. Non si possono definire più UPDATE CASCADE che agiscono sulle stesse colonne figlie o genitrici
- SET NULL: Cancella o aggiorna la riga dal genitore e setta la colonna figlia a NULL (non devono essere Not Null).
- NO ACTION: In MySQL 5.0, InnoDB rifiuterà la cancellazione o l'aggiornamento per la tabella genitore.

FK – Azioni - 2

- RESTRICT: V. NO ACTION
- SET DEFAULT: Riconosciuta dal parser ma non usata.