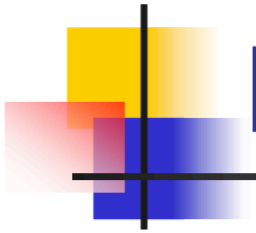
08. Ottimizzare l’uso degli indici

Introduzione

- Indici e selettività
- Query per prefisso
- Indici e distribuzione dei dati
- Indici e funzioni
- Only indexed columns
- Indici e DML
- Indici e SORT



Nota

- La tabella EMP utilizzata negli esempi successivi ha un elevato numero di righe (circa 100.000), mentre la tabella DEPT è di dimensioni ridotte (< 100).

Indici vs Full Table Scan (1)

- Accesso per indice ➔ random disk reads
- Full table scan ➔ sequential disk reads

- Effettuare delle “random disk reads” è notevolmente più lento che effettuare delle “sequential disk reads”, a causa del “seek time”.

Indici vs Full Table Scan (2)

- Per tabelle di dimensioni ridotte o per query che ritornano un elevate numero di righe è (in genere) più efficiente un accesso in full-table-scan.
- Eccezione: tabelle INNODB e query sulla PK

Indici e selettività: esempio (1)

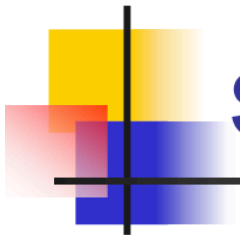
```
SELECT * FROM t4 WHERE sex='F';
```

- L'utilizzo di un (eventuale) indice che ritorna in media il 50% delle righe è meno efficiente di un accesso in full-table-scan.

```
SELECT MAX(empno)
FROM t4 WHERE sex='M';
1 row in set (9.63 sec)

(con indice !!!)
```

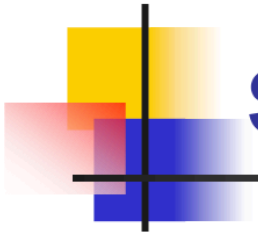
```
DROP INDEX t4_IX1;

SELECT MAX(empno)
FROM t4 WHERE sex='M';
1 row in set (6.28 sec)

(senza indice !!!)
```

Ottimizzare l'accesso alla singola tabella (1)



- Un accesso per indice è preferibile se la quantità dei dati selezionati è inferiore al 10% (del totale), altrimenti è preferibile un "full table scan".
- La percentuale varia a secondo della "engine". Per MyIsam è conveniente un accesso per indice fino al 30% delle righe selezionate.

Ottimizzare l'accesso alla singola tabella (2)



- Per tabelle con poche righe è, normalmente, preferibile un accesso in full table scan.
- Se necessario/conveniente aggiungere degli indici per selezionare e ordinare i dati.

Ottimizzare l'accesso alla singola tabella: un esempio

```
SELECT * FROM emp WHERE
ename LIKE 'nome14045%';
+-------+-----------+--------+
| empno | ename     | deptno |
+-------+-----------+--------+
| 14045 | nome14045 |      5 |
+-------+-----------+--------+
1 row in set (1.76 sec)
```

```
EXPLAIN SELECT * FROM emp WHERE
ename LIKE 'nome14045%' \G

*********** 1. Row ****
           id: 1
  select_type: SIMPLE
        table: emp
         type: ALL
...
```

```
CREATE INDEX IX_ENAME
ON emp(ename);
```

Ottimizzare l'accesso alla singola tabella: un esempio

```
SELECT * FROM emp WHERE
ename LIKE 'nome14045%';
+-------+-----------+--------+
| empno | ename     | deptno |
+-------+-----------+--------+
| 14045 | nome14045 |      5 |
+-------+-----------+--------+
1 row in set (0.33 sec)
```

```
EXPLAIN SELECT * FROM emp WHERE
ename LIKE 'nome14045%' \G

*********** 1. Row ****
           id: 1
  select_type: SIMPLE
        table: emp
         type: range
possible_keys: IX_ENAME
          key: IX_ENAME
```

- Spiegazione: la creazione di un indice sulla colonna ENAME, velocizza le query che utilizzano l'indice in modo SELETTIVO (ossia che recuperano poche righe).

```
SELECT count(*) FROM emp WHERE
ename LIKE 'nome%';
+-----------+
| count(*)  |
+-----------+
|    128000 |
+-----------+
1 row in set (1.32 sec)
```

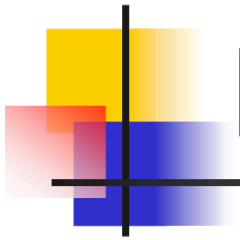
```
EXPLAIN SELECT count(*) FROM emp WHERE
ename LIKE 'nome%' \G

*********** 1. Row ****
           id: 1
  select_type: SIMPLE
        table: emp
         type: range
possible_keys: IX_ENAME
          key: IX_ENAME
```

```
DROP INDEX IX_ENAME ON emp;
```

```
SELECT count(*) FROM emp WHERE
ename LIKE 'nome%';
+-----------+
| count(*)  |
+-----------+
|    128000 |
+-----------+
1 row in set (0.88 sec)
```

```
EXPLAIN SELECT count(*) FROM emp WHERE
ename LIKE 'nome%' \G

*********** 1. Row ****
           id: 1
  select_type: SIMPLE
        table: emp
         type: ALL
```

- Spiegazione: la creazione di un indice sulla colonna ENAME, rallenta le query che utilizzano l'indice in modo NON SELETTIVO (ossia che recuperano molte righe tramite l'indice).



Indici e query per prefisso

- Un indice multicolonna definito sulle colonne (t1,t2, … , tn) può essere utilizzate per query che hanno una condizione di where su un qualsiasi prefisso delle colonne indicizzate.

Indici e prefisso: esempio (1)

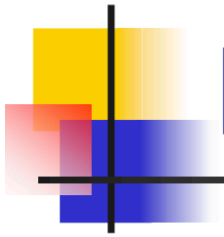
```
CREATE TABLE t5
(a int, b date, c varchar(10));

CREATE INDEX t5_ix1 ON t5(a,b);
```

Indici e prefisso: esempio (2)

```
EXPLAIN
SELECT * FROM t5 WHERE a=1 and b=2;

   select_type: SIMPLE
         table: T5
          type: ref
possible_keys: t5_ix1
           key: t5_ix1
```

```
EXPLAIN
SELECT * FROM t5 WHERE a=1;

   select_type: SIMPLE
         table: T5
          type: ref
possible_keys: t5_ix1
           key: t5_ix1
```

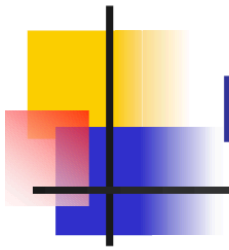
```
EXPLAIN
SELECT * FROM t5 WHERE b=2;

   select_type: SIMPLE
         table: t5
          type: ALL
possible_keys: NULL
           key: NULL
```



Indici e valori NULL

- Mysql memorizza all'interno di un indice anche i valori NULL.
- Query che utilizzano condizioni del tipo "IS NULL" o "IS NOT NULL" possono essere indicizzate.
- (*) Altri rdbms non indicizzano i valori NULL.

Indici e distribuzione dei dati non uniforme (1)



- Nel caso in cui una colonna indicizzata ha una distribuzione non uniforme dei dati, la convenienza nell'uso dell'indice dipende dal valore selezionato.

- E' consigliabile "guidare" l'ottimizzatore ad utilizzare o meno l'indice in base ad i valori selezionati.

Indici e distribuzione dei dati non uniforme: esempio (1)

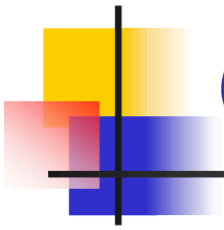
```
SELECT deptno, count(*) FROM t4
GROUP BY deptno;

+--------+----------+
| deptno | count(*) |
+--------+----------+
|     10 |  1063910 |
|     20 |       10 |
+--------+----------+
```

```
SELECT MAX(empno) FROM t4
WHERE deptno=10;

+-------------+
| max(empno)  |
+-------------+
|     1063910 |
+-------------+
1 row in set (10.41 sec)
```

```
EXPLAIN SELECT MAX(empno) FROM t4
WHERE deptno=10 \G
*********** 1. Row **************
           id: 1
  select_type: SIMPLE
        table: t4
         type: ref
possible_keys: IX4
          key: IX4
```

```
SELECT MAX(empno) FROM t4
WHERE deptno=20;

+-------------+
| max(empno)  |
+-------------+
|     1063920 |
+-------------+
1 row in set (0.00 sec)
```

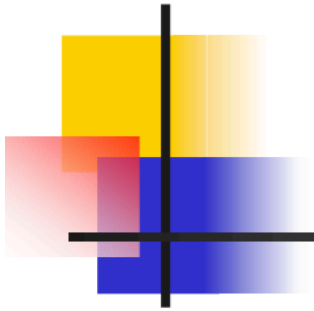
```
EXPLAIN SELECT MAX(empno) FROM t4
WHERE deptno=20 \G
*********** 1. Row **************
           id: 1
  select_type: SIMPLE
        table: t4
         type: ref
possible_keys: IX4
          key: IX4
```



Gli Hint (1)

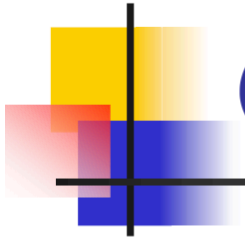
- E’ un suggerimento all’ottimizzatore per guidarlo verso la scelta di un piano di esecuzione.

- Non è uno standard SQL, ogni rdbms ha una sintassi proprietaria.
- Permettono di ridurre il tempo di PARSE.
- Permettono (se corretti) di impostare il piano di esecuzione ottimale.



I principali HINT

- STRAIGHT_JOIN
- USE INDEX
- IGNORE INDEX
- FORCE INDEX



Gli hint INDEX

- Forza l'ottimizzatore a valutare (USE), ignorare (IGNORE) o utilizzare (FORCE) l'uso di un indice.
- Non sono validi per forzare il piano di esecuzione di ORDER BY o GROUP BY.

Hint INDEX: la sintassi

- SINTASSI:

  SELECT … FROM <table>

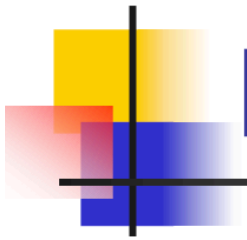
  [/*! {USE INDEX | IGNORE INDEX | FORCE INDEX} (index-list) */]

  …

- E’ fortemente consigliabile racchiudere gli HINT tra /*! HINT */
- La maggior parte degli rdbms considerano tutto quello che si trova tra /* e */ un commento, quindi non segnalano un errore sintattico.

```
SELECT MAX(empno) FROM t4 /*! IGNORE
INDEX (IX4) */ WHERE deptno=10;
+-------------+
| max(empno)  |
+-------------+
|     1063910 |
+-------------+
1 row in set (6.33 sec)
```

```
EXPLAIN SELECT MAX(empno) FROM t4 /*!
IGNORE INDEX (IX4) */ WHERE deptno=10\G
*********** 1. row ***************
           id: 1
  select_type: SIMPLE
        table: t4
         type: ALL
possible_keys: NULL
          key: NULL
```



ESEMPIO: hint INDEX (1)

```
SELECT count(*) FROM emp
WHERE ename like 'nome%';

......
1 row in set (1.57 sec)
```

```
EXPLAIN
SELECT count(*) FROM emp
WHERE ename like 'nome%' \G
```

```
*********** 1. row *****
           id: 1
  select_type: SIMPLE
        table: emp
         type: range
possible_keys: ix_ename
          key: ix_ename
...
```

```
SELECT count(*) FROM emp
/*! IGNORE INDEX (ix_ename) */
WHERE ename like 'nome%';

.....
1 row in set (0.40 sec)
```

```
EXPLAIN
SELECT count(*) FROM emp
/*! IGNORE INDEX (ix_ename) */
WHERE ename like 'nome%' \G
```

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

Indici e funzioni

- L'utilizzo di una qualsiasi funzione su una colonna indicizzata NON permette l'uso di un eventuale indice definito sulla colonna stessa.

Indici e funzioni: esempio (1)

```
EXPLAIN SELECT * FROM emp
WHERE LOWER(ename)='xyz' \G
*************** 1. row **********
           id: 1
  select_type: SIMPLE
        table: emp
         type: ALL
possible_keys: NULL
          key: NULL
```

Indici e funzioni: esempio (2)

```
EXPLAIN SELECT * FROM emp
WHERE ename='xyz' \G
*************** 1. row ***********
           id: 1
  select_type: SIMPLE
        table: emp
         type: ref
possible_keys: emp_ix
          key: emp_ix
```

Indici e conversione implicita

- E’ un caso particolare di funzioni applicate a colonne indicizzate.
- Se si confronta una colonna di tipo stringa con un numero, mysql implicitamente applica la funzione CAST alla colonna, impedendo in questo modo l’utilizzo di un indice.

```
EXPLAIN SELECT * FROM emp
WHERE ename=10 \G
*************** 1. row **********
             id: 1
    select_type: SIMPLE
          table: emp
           type: ALL
possible_keys: emp_ix
            key: NULL
```

```
EXPLAIN SELECT * FROM emp
WHERE ename='10' \G
*************** 1. row **********
             id: 1
    select_type: SIMPLE
          table: emp
           type: ref
possible_keys: emp_ix
            key: emp_ix
```

Only indexed columns

- Se tutte le colonne selezionate da una query appartengono all'indice, mysql non accederà alla tabella ma solamente all'indice.

```
EXPLAIN SELECT ename,sex
FROM emp
WHERE ename='xyz' \G
```

```
   select_type: SIMPLE
         table: emp
          type: ref
possible_keys: emp_ix
           key: emp_ix
       key_len: 11
           ref: const
          rows: 1
         Extra: Using where
```

```
CREATE INDEX ename_sex_ix
ON
emp(ename,sex);
```

```
   select_type: SIMPLE
         table: emp
          type: ref
possible_keys: ename_sex_ix
           key: ename_sex_ix
       key_len: 11
           ref: const
          rows: 1
         Extra: Using where; Using index
```

Indici e DML

- La presenza di indici definiti su una tabella rende meno efficienti le operazioni di insert/update/delete (DML) sulla tabella stessa.
- Le operazioni DML “frammentano” l’indice. Aumenta l’occupazione di spazio e le query sono meno efficienti.

SORT in memoria o su disco

- Se lo spazio di memoria richiesto per effettuare un sort non è sufficiente, è necessario effettuare il sort su disco.
- I sort su disco sono uno dei fattori più degradanti per le performance.

Istruzioni che richiedono un SORT (1)

Le seguenti clausole utilizzate all'interno di una query richiedono un sort:

- DISTINCT
- GROUP BY
- UNION
- ORDER BY

Le seguenti DDL richiedono un sort:

- CREATE INDEX

Indici e SORT

- Gli indici B-tree sono una struttura ordinata.
- L'utilizzo di un apposito indice può evitare la necessità di effettuare l'operazione di SORT.

```
EXPLAIN
SELECT DISTINCT ename
FROM emp\G
```

```
   select_type: SIMPLE
         table: emp
          type: index
possible_keys: NULL
           key: emp_ix
       key_len: 11
           ref: NULL
          rows: 2
         Extra: Using index
```

```
DROP INDEX emp_ix ON emp;

EXPLAIN
SELECT DISTINCT ename
FROM emp\G
```

```
   select_type: SIMPLE
         table: emp
          type: ALL
possible_keys: NULL
           key: NULL
       key_len: NULL
           ref: NULL
          rows: 2
         Extra: Using temporary
```

Indici multicolumn e SORT

- In genere è necessario un indice “multicolumn” per evitare il SORT.
- Le colonne da indicizzare sono quelle utilizzate nella WHERE + quelle utilizzate nella ORDER BY.
- Attenzione all’ordine delle colonne.

```
CREATE INDEX emp_ix ON emp(ename);
CREATE INDEX deptno_ix on emp(deptno);

EXPLAIN
SELECT * FROM emp
WHERE deptno=10
ORDER BY ename\G
```

```
   select_type: SIMPLE
         table: emp
          type: ref
possible_keys: deptno_ix
           key: deptno_ix
       key_len: 5
           ref: const
          rows: 1
      Extra: Using where; Using filesort
```

```
CREATE INDEX deptno_ename_ix
ON emp(deptno,ename);

EXPLAIN
SELECT * FROM emp
WHERE deptno=10
ORDER BY ename\G
```

```
   select_type: SIMPLE
         table: emp
          type: ref
possible_keys: deptno_ix,deptno_ename_ix
           key: deptno_ename_ix
       key_len: 5
           ref: const
          rows: 1
         Extra: Using where
```

Indici e MIN/MAX

- Gli indici B-tree sono una struttura ordinata.
- L'utilizzo di un apposito indice permette di effettuare velocemente un'operazione di MIN/MAX su una colonna.

```
SELECT MAX(ename) FROM emp;

+-------------+
| max(ename)  |
+-------------+
| nome99999   |
+-------------+
1 row in set (0.39 sec)
```

```
EXPLAIN SELECT MAX(ename) FROM emp \G
*********** 1. row ****
           id: 1
  select_type: SIMPLE
        table: emp
         type: ALL
possible_keys: NULL
```

```
CREATE INDEX ename_ix
ON emp(ename);
```

```
SELECT MAX(ename) FROM emp;

+-------------+
| max(ename)  |
+-------------+
| nome99999   |
+-------------+
1 row in set (0.03 sec)
```

```
EXPLAIN SELECT MAX(ename) FROM emp \G
*********** 1. row ********
           id: 1
  select_type: SIMPLE
        table: NULL
         type: NULL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: NULL
        Extra: Select tables optimized away
```

Ordinamento fisico delle righe

- Se si interroga frequentemente una tabella per i valori contenuti in una certa colonna è consigliabile ordinare fisicamente le righe della tabella in base alla colonna.
- L'accesso per indice sarà, in questo modo, più efficiente perché si dovranno recuperare meno blocchi dal disco.

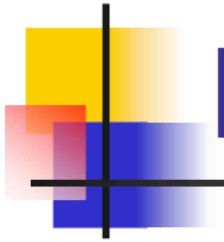
Esempio (1)

```
CREATE TABLE T15 TYPE=MYISAM
AS SELECT * FROM T14
ORDER BY deptno;
```

- Su entrambe le tabelle è definito un indice sulla colonna deptno.

```
SELECT MAX(empno) FROM T14
WHERE deptno=2;

+-------------+
| max(empno)  |
+-------------+
|           2 |
+-------------+
1 row in set (8.20 sec)
```

```
SELECT MAX(empno) FROM T15
WHERE deptno=2;

+-------------+
| max(empno)  |
+-------------+
|           2 |
+-------------+
1 row in set (0.95 sec)
```



Riepilogo

- Indici e selettività
- Query per prefisso
- Indici e distribuzione dei dati
- Indici e funzioni
- Only indexed columns
- Indici e DML
- Indici e SORT