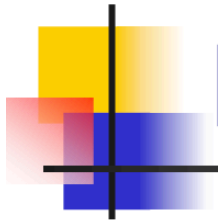
07. Ottimizzare le istruzioni SQL

La UNION (1)

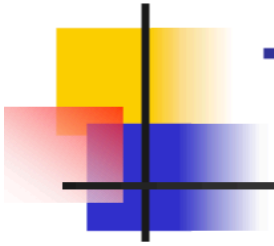
- L'operazione di UNION esegue l'unione insiemistica di 2 query aventi le stesse colonne (numero e tipo).
- La UNION DISTINCT (è il default), ritorna l'unione dei due insiemi senza duplicati.
- La UNION ALL, ritorna l'unione dei due insiemi, senza effettuare la DISTINCT.



Il Join: la sintassi

Esistono 2 sintassi (largamente diffuse ed utilizzate) per scrivere una operazione di join:

- Theta-style
- ANSI-style



Theta-style join: un ESEMPIO

```
SELECT  e.ename, d.descr
FROM    emp e, dept d
WHERE   e.deptno=d.deptno AND
        e.ename like 'B%';
```

- NON permette l'outer join (se non tramite estensioni non ANSI).

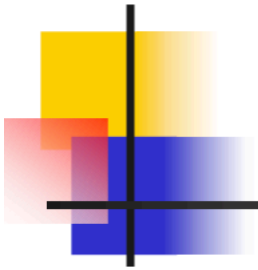
Ansi-style join: un ESEMPIO

```
SELECT  e.ename, d.descr
FROM    emp e JOIN dept d
ON      e.deptno=d.deptno
WHERE   e.ename like 'B%';
```

- E' supportato da tutti i maggiori rdbms (alcuni hanno estensioni proprietarie).
- Permette tutti i tipi di join.

L’esecuzione di istruzioni SQL

- Per essere eseguita un’istruzione SQL deve essere “compilata” e deve essere prodotto “il codice eseguibile” (detto piano di esecuzione).



Il piano di esecuzione

- E' l'insieme dei passi elementari che permettono l'esecuzione di una istruzione SQL.
- Ogni istruzione ha diversi possibili piani di esecuzione.

L’ottimizzatore (1)

- E’ il modulo dell’rdbms che si occupa di determinare il miglior piano di esecuzione possibile di uno statement SQL.

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.
- Altri rdbms hanno una implementazione diversa del comando EXPLAIN.
- Il comando EXPLAIN contiene dettagli che dipendono sia dall'rdbms che dalla versione scelta.

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.

Scrivere istruzioni SQL “semplici”

- Istruzioni SQL con una logica complessa sono difficili da ottimizzare e da manutenere.
- Query complesse sono il risultato di una progettazione fisica non corretta (mancata denormalizzazione).

Verificare i piani di esecuzione

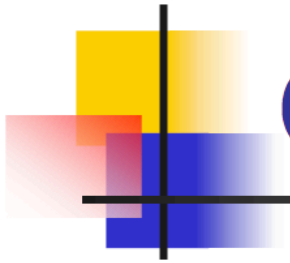
- E’ fondamentale, per ogni query, verificare i piani di esecuzione per ottimizzare gli accessi.
- I piani di esecuzione possono essere forzati mediante l’uso di HINT.

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).

Ottimizzare l'ordine dei passi (1)

- La scelta dell'ordine in cui le tabelle sono messe in join è fondamentale se si usa l'algoritmo di Nested Loop Join.
- E' particolarmente importante quando si effettua il join di molte tabelle.



Ottimizzare l'ordine dei passi (2)

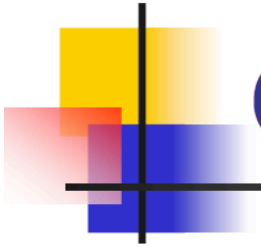
- In caso di "inner" join un qualsiasi ordine di join è possibile, l'ordine NON è determinato dall'ordine in cui le tabelle appaiono nella clausola FROM.
- In caso di "outer" join l'ordine è fissato dalla condizione di outer-join.

Ottimizzare l'ordine dei passi (3)

- In genere è conveniente scegliere come driving-table (outer-table) la tabella con la condizione di filtro più selettiva.
- La selettività deve essere considerata anche rispetto al join con le altre tabelle.

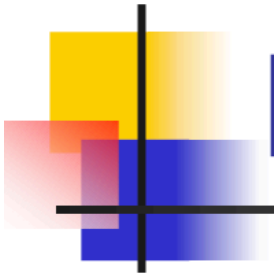
Ottimizzare l'ordine dei passi (4)

- La driving-table può essere acceduta in full-table-scan o tramite un indice selettivo sulle colonne di filtro.
- La "inner-table" deve essere acceduta tramite un indice, preferibilmente univoco, sulle colonne di join.



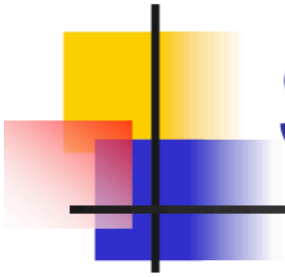
Ottimizzare l'ordine dei passi (5)

- Nell'ottimizzare gli accessi al disco, si deve considerare la cache dei dati/indici.
- Se la inner-table è "piccola" rispetto alle dimensioni della cache ci si può aspettare che i blocchi della tabella (e dei suoi indici) siano prevalentemente in cache.



L’hint STRAIGHT_JOIN

- Forza l’ottimizzatore ad effettuare il JOIN nella sequenza specificata nella clausola FROM.



STRAIGHT_JOIN: la sintassi

- SINTASSI:

  SELECT /*! STRAIGHT_JOIN */

  <column-list>

  FROM …

```
EXPLAIN
SELECT /*! STRAIGHT_JOIN */ *
FROM emp e, dept d
WHERE e.deptno=d.deptno \G
```

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: e
...
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: d
...
```

Accede prima alla tabella EMP (E) e alla DEPT (d)

Ottimizzare l'ordine dei passi: un esempio (1)

```
SELECT e.ename, d.descr
FROM   emp e, dept d
WHERE
e.deptno=d.deptno AND
e.ename like 'nome12345%';


......
11 rows in set (3.05 sec)
```

Ottimizzare l'ordine dei passi: un esempio (2)

```
EXPLAIN
SELECT e.ename, d.descr
FROM   emp e, dept d
WHERE
e.deptno=d.deptno AND
e.ename like 'nome12345%'\G
```

```
*********** 1. row ***
           id: 1
  select_type: SIMPLE
        Table: d
         type: ALL
...
*********** 2. row ****
           id: 1
  select_type: SIMPLE
        Table: e
         type: ref
possible_keys: FK_EMP_DEPT
          key: FK_EMP_DEPT
...
```

```
SELECT /*! STRAIGHT_JOIN */
e.ename, d.descr
FROM emp e, dept d
WHERE
e.deptno=d.deptno and
e.ename like 'nome12345%';


......
11 rows in set (0.40 sec)
```

```
EXPLAIN
SELECT /*! STRAIGHT_JOIN */
e.ename, d.descr
FROM emp e, dept d
WHERE
e.deptno=d.deptno AND
e.ename like 'nome12345%'\G
```

```
*********** 1. row ***
           id: 1
  select_type: SIMPLE
        Table: e
         type: ALL

…
*********** 2. row ****
           id: 1
  select_type: SIMPLE
        Table: d
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
```

- Spiegazione: L'accesso tramite un indice (per ogni riga della tabella DEPT) alla tabella EMP è un accesso casuale (➔ alto seek time). E' preferibile accedere prima alla tabella EMP (che ha un numero elevato di righe) in modo sequenziale e poi alla tabella DEPT tramite la sua PK.

- Nota: Se la condizione sulla colonna “ename” è selettiva, si può ottimizzare ulteriormente la query creando un indice sulla colonna “ename” della tabella EMP ed evitando in questo modo il full-table-scan sulla tabella stessa.

Ottimizzare l'ordine dei passi: un esempio (9)

```
SELECT e.ename, d.descr
FROM   emp e, dept d
WHERE
e.deptno=d.deptno AND
d.descr like 'ufficio 6%' AND
e.ename like 'nome12345%';
......
3 rows in set (0.05 sec)
```

```
EXPLAIN
SELECT e.ename, d.descr
FROM   emp e, dept d
WHERE
e.deptno=d.deptno AND
d.descr like ‘ufficio 6%’ AND
e.ename like 'nome12345%‘\G
```

```
*************** 1. row ***
           id: 1
  select_type: SIMPLE
        Table: d
         type: ALL
...
*************** 2. row ****
           id: 1
  select_type: SIMPLE
        Table: e
         type: ref
possible_keys: FK_EMP_DEPT
          key: FK_EMP_DEPT
...
```

```
SELECT /*! STRAIGHT_JOIN */
e.ename, d.descr
FROM emp e, dept d
WHERE
e.deptno=d.deptno and
d.descr like 'ufficio 6%' AND
e.ename like 'nome12345%';
......
11 rows in set (0.35 sec)
```

```
EXPLAIN
SELECT /*! STRAIGHT_JOIN */
e.ename, d.descr
FROM emp e, dept d
WHERE
e.deptno=d.deptno AND
d.descr like ‘ufficio 6%’ AND
e.ename like 'nome12345%‘\G
```

```
*************************** 1. row ***
           id: 1
  select_type: SIMPLE
        table: e
         type: ALL
...
*************************** 2. row ****
           id: 1
  select_type: SIMPLE
        table: d
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
...
```

- Spiegazione: L'utilizzo della tabella DEPT come driving-table è conveniente se il filtro applicato (ossia le righe selezionate dalla DEPT) comporta l'uso selettivo dell'indice definito sulla FK della tabella EMP.

- Nota: Nell'esempio riportato l'uso della tabella DEPT come driving-table è conveniente perché "poche" righe della tabella EMP sono legate ad un "dipartimento" la cui descrizione inizia con la stringa "ufficio 6".

Riscrivere la query

- Spesso le query possono essere scritte in diversi modi, tra loro equivalenti.
- E' preferibile scrivere la query in modo opportuno, l'ottimizzatore non sempre riesce ad individuare la query più performante.

Riscrivere la query: un esempio (1)

```
SELECT * FROM emp WHERE
(ename like 'nome12345%') OR
(empno = 10000);


......
12 rows in set (0.81 sec)
```

```
EXPLAIN
SELECT * FROM emp WHERE
(ename like 'nome12345%') OR
(empno = 10000) \G
```

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

Riscrivere la query: un esempio (4)

```
SELECT * FROM emp WHERE
(ename like 'nome12345%')
UNION
SELECT * FROM emp WHERE
(empno = 10000);
......
12 rows in set (0.01 sec)
```

```
EXPLAIN
SELECT * FROM emp WHERE
(ename like 'nome12345%')
UNION
SELECT * FROM emp WHERE
(empno = 10000) \G
```

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

*************** 2. row ******
           id: 2
  select_type: UNION
        table: emp
         type: const
possible_keys: PRIMARY
          key: PRIMARY
```

```
*********** 3. row *****
           id: NULL
  select_type: UNION RESULT
        table: <union1,2>
         type: ALL
possible_keys: NULL
          key: NULL
```

- Spiegazione: MySql utilizza un solo indice per query (nella versione < 5.0). Se non può utilizzare nessun indice per soddisfare tutte le condizione effettua un full table scan. Se la query viene “divisa” in più union ha la possibilità di utilizzare un indice per ogni union.