

frilly / November 12, 2024 04:21PM

[8 種常見 SQL 錯誤用法，你避開了嗎？](#)

1. [1. SQL 語句的執行順序：](#)
2. [2. 1、LIMIT 語句](#)
3. [3. 2、隱式轉換](#)
4. [4. 3、關聯更新、刪除](#)
5. [5. 4、混合排序](#)
6. [6. 5、EXISTS 語句](#)
7. [7. 6、條件下推](#)
8. [8. 7、提前縮小範圍](#)
9. [9. 8、中間結果集下推](#)
10. [10. 總結](#)

1. SQL 語句的執行順序：

FROM
ON
JOIN
WHERE
GROUP BY
HAVING
SELECT
DISTINCT
ORDER BY
LIMIT

2. 1、LIMIT 語句

分頁查詢是最常用的場景之一，但也通常也是最容易出問題的地方。比如對於下面簡單的語句，一般 DBA 想到的辦法是在 type, name, create_time 欄位上加組合索引。這樣條件排序都能有效的利用到索引，效能迅速提升。

```
SELECT *  
FROM operation  
WHERE type = 'SQLStats'  
      AND name = 'SlowLog'  
ORDER BY create_time  
LIMIT 1000, 10;
```

好吧，可能90%以上的 DBA 解決該問題就到此為止。但當 LIMIT 子句變成“LIMIT 1000000,10”時，程式設計師仍然會抱怨：我只取10條記錄為什麼還是慢？

要知道資料庫也並不知道第1000000條記錄從什麼地方開始，即使有索引也需要從頭計算一次。出現這種效能問題，多數情形下是程式設計師偷懶了。

在前端資料瀏覽翻頁，或者大資料分批匯出等場景下，是可以將上一頁的最大值當成引數作為查詢條件的。SQL 重新設計如下：

```

SELECT *
FROM operation
WHERE type = 'SQLStats'
AND name = 'SlowLog'
AND create_time > '2017-03-16 14:00:00'
ORDER BY create_time limit 10;

```

在新設計下查詢時間基本固定，不會隨著資料量的增長而發生變化。

3. 2、隱式轉換

SQL 語句中查詢變數和欄位定義型別不匹配是另一個常見的錯誤。比如下面的語句：

```

mysql> explain extended SELECT *
> FROM my_balance b
> WHERE b.bpn = 14000000123
> AND b.isverified IS NULL ;
mysql> show warnings;
| Warning | 1739 | Cannot use ref access on index 'bpn' due to type or collation conversion on field 'bpn'

```

其中欄位 bpn 的定義為 varchar(20)，MySQL 的策略是將字串轉換為數字之後再比較。函式作用於表字段，索引失效。

上述情況可能是應用程式框架自動填入的引數，而不是程式設計師的原意。現在應用框架很多很繁雜，使用方便的同時也小心它可能給自己挖坑。

4. 3、關聯更新、刪除

雖然 MySQL5.6 引入了物化特性，但需要特別注意它目前僅僅針對查詢語句的最佳化。對於更新或刪除需要手工重寫成 JOIN。

比如下面 UPDATE 語句，MySQL 實際執行的是迴圈/巢狀子查詢 (DEPENDENT SUBQUERY)，其執行時間可想而知。

```

UPDATE operation o
SET status = 'applying'
WHERE o.id IN (SELECT id
FROM (SELECT o.id,
o.status
FROM operation o
WHERE o.group = 123
AND o.status NOT IN ( 'done' )
ORDER BY o.parent,
o.id
LIMIT 1) t);

```

執行計劃：

```

+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+
--+
```

```

| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
--+
| 1 | PRIMARY | o | index | | PRIMARY | 8 | | 24 | Using where; Using temporary
|
| 2 | DEPENDENT SUBQUERY | | | | | | | | Impossible WHERE noticed after reading const tables |
| 3 | DERIVED | o | ref | idx_2,idx_5 | idx_5 | 8 | const | 1 | Using where; Using filesort
|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
--+

```

重寫為 JOIN 之後，子查詢的選擇模式從 DEPENDENT SUBQUERY 變成 DERIVED，執行速度大大加快，從7秒降低到2毫秒。

```

UPDATE operation o
JOIN (SELECT o.id,
        o.status
      FROM operation o
      WHERE o.group = 123
            AND o.status NOT IN ( 'done' )
      ORDER BY o.parent,
              o.id
      LIMIT 1) t
ON o.id = t.id
SET status = 'applying'

```

執行計劃簡化為：

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | | | | | | | | Impossible WHERE noticed after reading const tables |
| 2 | DERIVED | o | ref | idx_2,idx_5 | idx_5 | 8 | const | 1 | Using where; Using filesort
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

5. 4、混合排序

MySQL 不能利用索引進行混合排序。但在某些場景，還是有機會使用特殊方法提升效能的。

```

SELECT *
FROM my_order o
INNER JOIN my_appraise a ON a.orderid = o.id
ORDER BY a.is_reply ASC,
         a.appraise_time DESC
LIMIT 0, 20

```

執行計劃顯示為全表掃描：

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

```
| 1 | SIMPLE | a | ALL | idx_orderid | NULL | NULL | NULL | 1967647 | Using filesort |
| 1 | SIMPLE | o | eq_ref | PRIMARY | PRIMARY | 122 | a.orderid | 1 | NULL |
+---+-----+-----+-----+-----+-----+-----+-----+-----+---+
```

由於 is_reply 只有0和1兩種狀態，我們按照下面的方法重寫後，執行時間從1.58秒降低到2毫秒。

```
SELECT *
FROM ((SELECT *
      FROM my_order o
      INNER JOIN my_appraise a
      ON a.orderid = o.id
      AND is_reply = 0
      ORDER BY appraise_time DESC
      LIMIT 0, 20)
UNION ALL
(SELECT *
      FROM my_order o
      INNER JOIN my_appraise a
      ON a.orderid = o.id
      AND is_reply = 1
      ORDER BY appraise_time DESC
      LIMIT 0, 20)) t
ORDER BY is_reply ASC,
        appraisetime DESC
LIMIT 20;
```

6. 5、EXISTS語句

MySQL 對待 EXISTS 子句時，仍然採用巢狀子查詢的執行方式。如下面的 SQL 語句：

```
SELECT *
FROM my_neighbor n
LEFT JOIN my_neighbor_apply sra
ON n.id = sra.neighbor_id
AND sra.user_id = 'xxx'
WHERE n.topic_status
```