以 SQL 寫報表

by

Triton Ho 

理想與現實

- 如果你想知道怎去分析以 TB 為單位的數據，你來錯地方了啦
- 對一般中小型公司.....
  - 資料庫很少 > 100GB 的
  - DBA 是誰？能吃的嗎？
  - Replication 是什麼？
  - 專門來作報表的伺服器？你想太多了吧

正確觀念

正確觀念

- 別把低價值資料放到 RDBMS
- 善用 Sampling
- 系統優先度
- 定義「報表」
- 低 consistency 報表
- 善用 Materialization
- 報表和 Seq Scan
- Seq Scan 和 Caching

別把低價值資料放到 RDBMS

- RDBMS 很貴的
  - 頂級 CPU + RAM + SSD + network
  - 在很多系統， RDBMS 都只有一台 Master
- 低價值資料 = 輕微誤差也死不了人的數據
  - 例子：網站的每秒流量

善用 Sampling

- 做報表前，先問清楚用戶報表目的
- 例子：公司的總營業額
  - 用來報稅的，一定要完全準確
  - 用來做市場計劃 / 趨勢分析的，不一定需要完全準確
- 大數定律

系統優先度

- 一切行動，以會付錢（不管是現在還是未來）的用戶為最優先
  - 在網上商店，用戶等 5 秒便會說永別
  - 老闆等報表一小時不會死
    - tmux 是好工具 ~
- < 高 Consistency ，高效能，低開發時間 >
  三者最多你只能要兩個

定義「報表」

- 最廣義的「報表」
  - 一切非 CRUD 的行動
  - 例子：網上討論區的「熱門文章」和「最近文章」
  - 遊戲中的「玩家排名」

低 consistency 報表

- Consistency~= 數據正確性 + 數據即時性
  - 「熱門文章」和「最近文章」便是典型例子
- 邪惡想法：只要用戶不覺得有問題，系統不一定需要 100% Consistent
- 延伸思考：把「熱門文章」和「最近文章」的 ResultSet 放到 Redis ？

善用 Materialization

- 用在高 Consistency + 不能等待報表上
- 就是把報表要用的 data set 預先準備好，預先貯在起來
- 在資料改動時，也需要改動這份 data set
- 手動例子：在網上商店有 Orders 這個 table ，然後再有 UnFinishedOrder<Order_id> 這個特殊 table
  - Partial Index 也是另外一個方案

報表和 Seq Scan

- 如果你的報表需要拿出大量 Records ，使用 Full table scan 是不能避免的
  - 例子：把 2015 年稅務年度的 purchase_orders 作結算
  - 一個 Seq Scan 比多個不連續的 Random Read 更快
  - Index 在這情況沒有幫助
  - 你抄一個 1G 的 A 片比較快，還是 10000 張 A 圖比較快？

Seq Scan 和 Caching

- 如果 RDBMS 有 4GB RAM 能用，而現在要 Seq Scan 一個 5GB 的 table 。第二次的 Seq Scan 的 hit rate 會是多少？
  - 在 LRU 下，答案是 0%
  - Oracle 和 PostgreSQL 是故意把 Seq Scan 的資料不留在 Buffer 的
- 增加 RAM 能讓報表變快，是因為報表常常使用 index join
  - Caching 對 Random Access 有幫助

SQL 實戰

SQL 實戰

- 避免使用 select *
- 避免使用迴圈
- 善用 temp table
- 淺談 subquery
- 善用 window-function
- 善用 with-clause
- 手動的 predicate pushing
- 善用 Conditional function

避免使用 select *

- SQL 是宣告式語言，你應該專心你的 business logic ，讓 RDBMS 自動找出最佳的 execution plan
  - 同理 1 ：除非你家的D B A 在偷懶 / 你的 table schema 出錯，你不應該寫 SQL hints
  - 同理 2 ：宣告式語言 = tell-the-truth ，如果你只需要 ColA 和 ColB ，你應該用上 select ColA, ColB from tableX ……
    - 越清楚地告訴 RDBMS 你需要什麼， RDBMS 便越有可能找到最佳的方案
- 請參看： Index-Only Scans

避免使用迴圈

- 不是同一 subnet 下， network latency 可以非常大
  - 你可以用你的筆電連到 Amazon RDS 試試 ~
- 不是每一家 RDBMS 都有 execution plan caching 的

Saleman	PO_ID	Unit_Price	Total_Amount
	PO00001	21	210
	PO00011	21	105
Amy	PO00034	34	136
	PO00002	10	80
Betty	PO00022	18	36
	PO00045	5	50
	PO00101	9	900
Susan	PO00102	11	99

迴圈例子

- 你是否會寫成：

```
select name from salemen;
select * from orders
where saleman = 'Amy' order by id;
select * from orders
where saleman = 'Betty' order by id;
……
```

- 正確寫法：

```
select * from orders
order by saleman, id;
```

然後再對報表的 Saleman column 作 post-processing ，把相同的名字合併

善用 Temp table

- optimizer 所需要的時間，跟 Query 內的 tables 成 super-linear 關係
  - 如果有很多 tables ， optimizer 會使用 heuristic search（中文翻譯：猜猜樂）
- 所以，如果你需要執行超級複雜 Query ，請考慮把它分成數個 query ，並且使用 Temp table 來暫存資料
- Temp Table 只有單一 Session 能見到， Session 關掉後所有資料也會消失，所以他的效能遠遠比正常 table 快

淺談 subquery

- 例子：

```
select * from orders
where saleman in (
    select name from team
    where team_name = 'teamA'
)
```

- 有些新人會以為： subquery 會在 Outer query 的每一個 Record 都執行一次
  - 用屁股想也知道這是不可能嘛！

Uncorrelated subquery

- 跟外層沒有任何關係，不會用上外層任何資料
- subquery 內容只需要跑一次，所以效能非常快
- 例子：

```
select * from orders
where saleman in (
    select name from office_staff
    where team_name = 'teamA'
)
```

correlated subquery

- 就是說，subquery 會引用外層的資料
- 例子：

```
select * from orders
where exists (
    select 1 from office_staff
    where team_name = 'teamA'
    and orders.saleman = office_staff.name
)
```

- 最智障的 execution plan ：為每一個 orders record 跑一次 subquery 然後作對比
- 現實中，大部份情況 optimizer 會使用 join 來執行
  - 詳情請看 Query transformation 技術
  - optimizer 實際能力： Oracle > > PostgreSQL > MariaDB > MySQL

善用 window-function

- 還沒有 SQL:2003 前，如果你需要 pagination
  - 例子：

```
select * from students
where …… order by id limit 50
```

- window-function 是 SQL:2003 是標準語法，不過 MySQL 不支援，而且能造到遠比 pagination 強大的功能
  - 例子：

```
select * from (
    select *, row_number() over (order by score desc) as row_num
    from students
    where ……
) t
where t.row_num between 11 and 20
```

window-function 進階例子

- 我想要每班成績最高的首三個學生

```
Select * from (
    SELECT class_id, student_id,
    rank() over (
    partition by class_id order by score desc
    ) as ranking
    FROM students
) t
where t.ranking <= 3
```

- 這個例子的 window function ，讓你不用為每班作一次 select ，效能好了很多

善用 with-clause

- With-clause 是 SQL:1999 標準語法，不過 MySQL 和 MariaDB 都不支援
- 例子：我想要知道第二組別學生中，每班成績首三名的學生中，他們之間有兄弟姐妹關係的人
- 小心一點： postgreSQL 會對 with-clause 內的 subquery 進行 materialization

沒有使用 with-clause

```
Select * from (
    Select * from (
    SELECT class_id, student_id, parent_id
    rank() over (partition by class_id order by score desc) as ranking
    FROM students
    where students.group_id = 2
    ) t1
    where t1.ranking <= 3
) wtf1
where exists (
    Select 1 from (
    SELECT class_id, student_id, parent_id
    rank() over (partition by class_id order by score desc) as ranking
    FROM students
    where students.group_id = 2
    ) t2
    where t2.ranking <= 3
    and wtf1.student_id != t2.student_id and wtf1.parent_id = t2.parent_id
)
```

使用 with-clause 後

```
With top3students as (
    select * from (
        SELECT class_id, student_id, parent_id
        rank() over (partition by class_id order by score desc) as ranking
        FROM students
        where group_id = 2
    ) t1
    where t1.ranking <= 3
)
Select * from top3students t1
where exists (
    Select 1 from top3students t2
    where t1.student_id != t2.student_id and t1.parent_id = t2.parent_id
)
```

手動的 predicate pushing

- predicate pushing = 把你的 filtering condition，在不改變你的 Query 意思下，從外層推到內層
  - optimizer 會為你自動地工作的
  - Oracle 做得最好，但是不完美
- Materialization 是其中一個 prediate pushing 障礙
- Window function, aggregation 是另一大障礙

欠佳例子 1

```
With top3students as (
    select * from (
        SELECT class_id, student_id, parent_id
        rank() over (partition by class_id order by score desc) as ranking
        FROM students
    ) t1
    where t1.ranking <= 3
)
Select * from top3students t1
where exists (
    Select 1 from top3students t2
    where t1.student_id != t2.student_id and t1.parent_id = t2.parent_id
    and t1.group_id = t2.group_id
)
and t1.group_id = 2
```

欠佳例子 2

```
Select group_id, avg(score)
from students
group by group_id
having group_id = 2
```

理想例子

```
With top3students as (
    select * from (
        SELECT class_id, student_id, parent_id
        rank() over (partition by class_id order by score desc) as ranking
        FROM students
        where group_id = 2
    ) t1
    where t1.ranking <= 3
)
Select * from top3students t1
where exists (
    Select 1 from top3students t2
    where t1.student_id != t2.student_id and t1.parent_id = t2.parent_id
)
```

善用 Conditional function

- 現在你在寫 Candy crush ，每八小時給玩家一顆糖。如果玩家持續不消耗，最多他能有１００顆糖

```
select id, least(100, candy +
trunc(extract(epoch from current_timestamp -
last_candy_time) / 60 / 8)
) as candy
from players
```

例子： Case when 語法

- 讓你能在 Query 中使用 if-then-else 和 switching
- 例子：我想在學生報表中，首先輸出女生，然後再輸出男生，其中女生以身高順序排序，男生以身高倒序排序

```
Select * from students
order by gender,
(
    case
    when gender = 'female' then height
    when gender = 'male' then height * -1
    end
)
```

對 SQL 語言感想

- 跟很多 functional-programming language 一樣，他能以極短和優美的做到強大功能
  - Triton 親身經驗：相同工作， pl/SQL 程式碼只會是 Java 行數的 1 / 5
- 但是.....有誰看出以下 SQL 有什麼問題？

```
Select ……
from tableA
left join tableB on tableA.XXX = tableB.YYY
where tableB.ZZZ = ?
```

結語

- SQL optimizer 很可愛，但也很麻煩
- 管太多（使用 optimizer hints ），只會累死自己，而且大多數情況也沒有最佳結果
- 盡可能對她坦白，把所有你知道的告訴她，她便會乖乖的

完