跟 MySQL 說再見
加入 PostgreSQL 吧

By Triton Ho
Software Engineer of Mar's Potato

感謝各位台灣朋友的熱情幫助
沒有你們提供免費場地，住宿，活動統籌，便不會有這次活動

在今天的淺談中

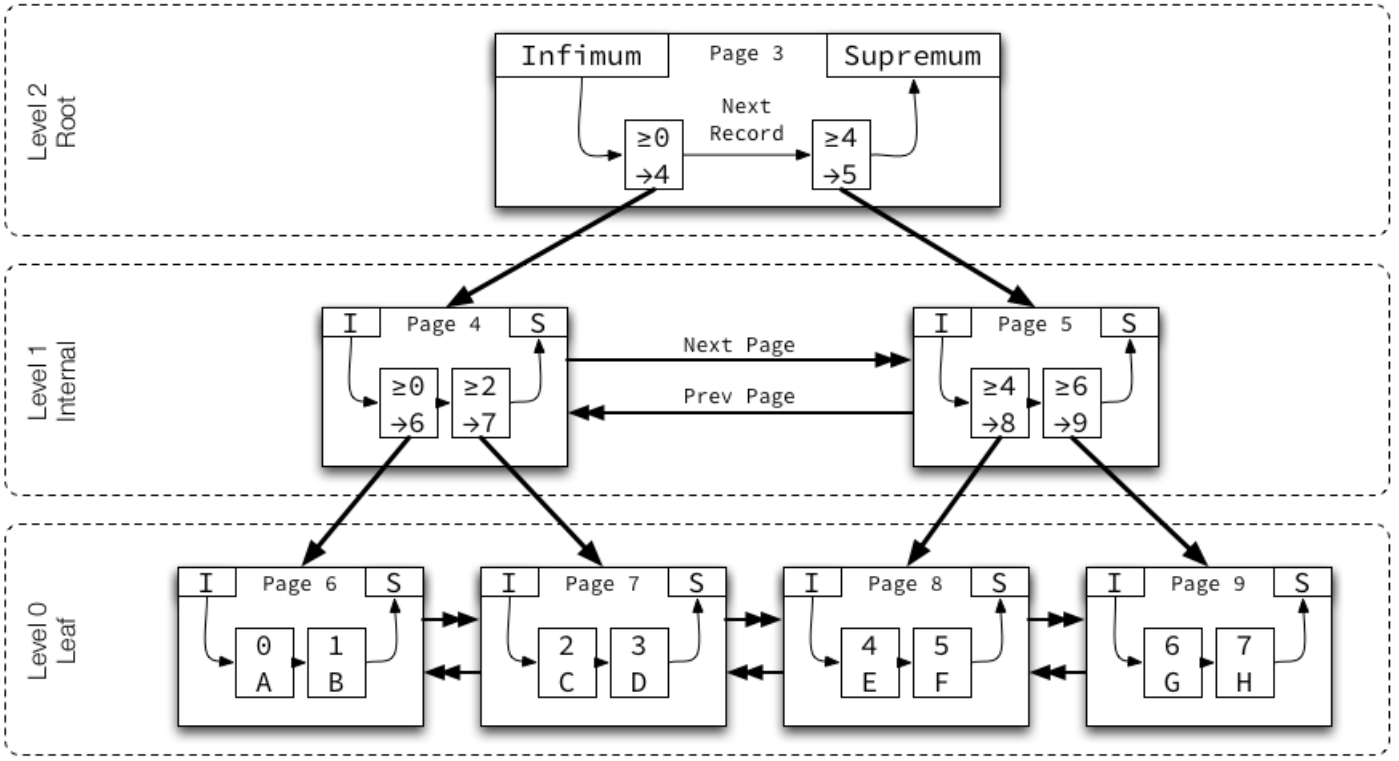
- MySQL = MySQL 5.7(innoDB)
- PostgreSQL = PostgreSQL 9.3
- Oracle = Oracle 11g(11.2.0.3)

大綱

- 淺談 Index-Organized Table(IOT)
- 淺談 Heap Table
- 對比 MySQL 和 PostgreSQL

淺談 Index-Organized Table

- 正如其名， Table 本身是一個巨大的 B+ tree index



Index-Organized Table 優點

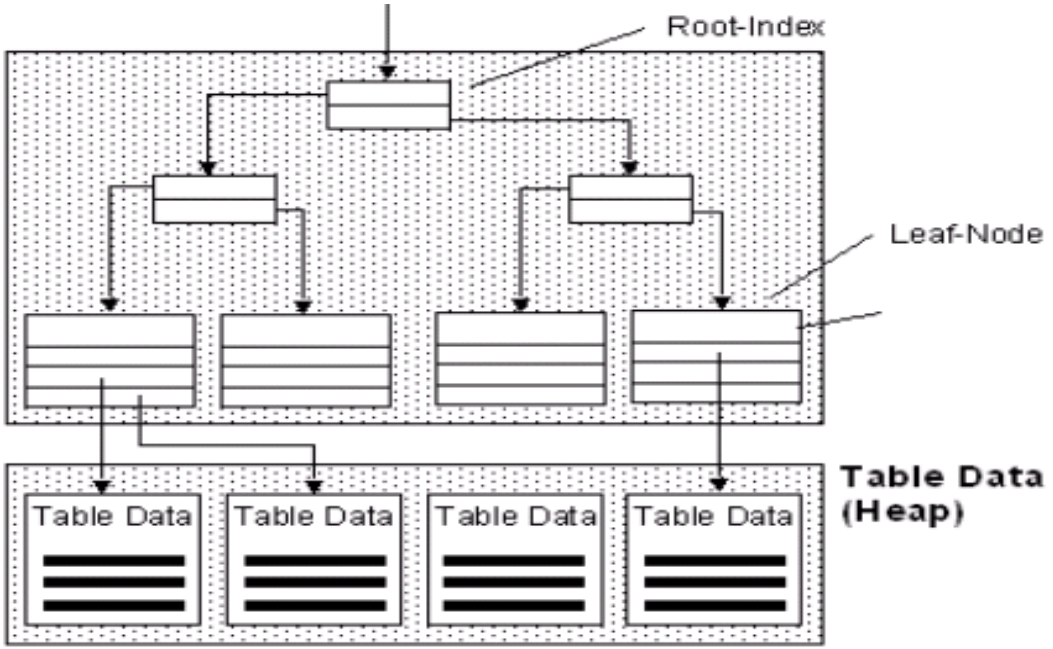
- 因為 rows 全都順序整理好了，在 range scan 時非常快
- 能省下 sorted by Primary Key 的時間（之後簡稱 PK ）
- 只需要一次（邏輯上） READ 便能拿到資料，需要較少 Storage IO

Index-Organized Table 缺點

- 因為整個 rows 都存放於 B+ tree 的 leaf node ，每個 leaf node 能存的 rows 有限
  - 因此，經常會發生 leaf node 的 splitting / merging
  - 因為 leaf node 的 splitting / merging ，引發 rows 要移動
- 如果 PK 是有規律的（例子： auto-increment ），讀取 / 寫入很容易集中在少量的 leaf-node 中，引發 hot-spot 問題
  - 負責這些 hot-spot 的 storage device 會忙死了，但是其他的 storage device 卻很閒
  - 因為 hot-spot 問題而發生意想不到 blocking ，常常接下來引發更多的 blocking
  - 同一時期產生的 rows 常常會在相近的時間 delete ，這樣會 rows 不平均地放到 leaf node 中，進而需要 leaf node merging

淺談 Heap Table

- Heap != priority list
- 最簡單來說， row data 隨便找一個 data block 存放。 PK 獨立放在一個 B+ tree index ，並且在 index leaf node 儲存指向 row data 的 pointer



Heap Table 優點

- 因為 index leaf node 只存放 (PK + pointer)
  - 一個 leaf node 可以存放更多的 rows ， leaf node splitting/merging 自然大減
  - 即使發生 leaf node splitting/merging ，也不會令 row data 需要移動位置
- 因為 index 本身體積小，即使降低 fill_factor 也不會讓 index 變大太多
  - 降低 fill_factor 能有效迴避 hot spot 問題
- Row data 能存放到 heap 中任何一個 data block ，沒有指定位置
  - 即使 PK 是用上 auto-increment ，相近 PK 的 row 會散落到整個 heap 之內，先天性不容易發生 hot spot
  - 在 insert new rows 時， rows data 能找一個沒有正被改動中的 data block 來寫入。不容易發生 blocking

Heap Table 缺點

- Range Scan on Pk 一般需要整個 table 都作一次 scaning ，極吃 CPU
- 需要兩次（邏輯上） READ 才能拿到資料，需要較多 Storage IO

先總結一下

- IOT 在 Read 的速度比較高
- OLTP 極少作 Range scan ，而且 Reporting 能在 read-only slave 上作， IOT 的優勢在大型 OTAP 系統中沒有意義
- 大量同時進行的 WRITE 之下， heap table 比較快，也不容易產生 hot spot

對比 MySQL 和 PostgreSQL
還有 Oracle 耶

對比 MySQL, postgreSQL and Oracle

	MySQL	PostgreSQL	Oracle
Table structure	IOT	Heap table	Heap table(default), IOT
Data block size	16kB	8kB	8kB
Free data block management	N.A.	Free Space Map (balanced binary tree)	Free list Group (a group of link list)
Old rows location	Original datablock, pointed by garbage pointer	Original datablock	REDO log
Old rows removal time	Periodic background purge job	Daily VACUUM job	After all TX starttime >= old row timestamp
Index write lock	Page level when no page splitting/merging	Page level	Page level

Table structure 解說

- RDBMS 的 READ 能用 replication 解決， read-only slave 是可以無限增加的
- WRITE 只能在 Master。所以， RDBMS 的「快」從來只看 WRITE 效能
- CPU ， storage ， storage IO 只會越來越不值錢，未來（現在？）效能更看是否發生 blocking
- 因為 IOT 在 READ 和 low concurrent WRITE 比較快，所以跑分一流
  - benchmark 冠軍是 MSSQL 唷 ~

Data Block Size 解說

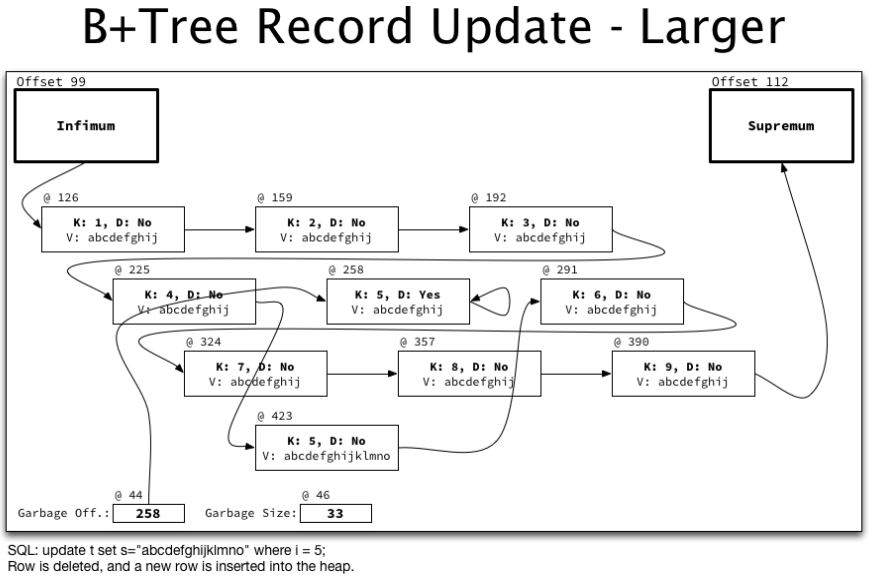
- 越大的 data block ， structural overhead (header, checksum, remaining space) 比率越低
  - structural overhead 比率低，便能減少 Storage IO
- 越小的 data block ，能儲存的 rows 數量便變小
  - 所以，同一時間多個 TX 同時改動同一 data block 的內的 rows 可能性便變小
  - 所以， hot spot 可能性變小
- 越小的 data block ， index leaf node splitting / merging 時影響到的 rows 便越小
  - 雖然 splitting / merging 變得更頻密，但每次的 splitting / merging 的痛苦度下降
- Oracle / PostgreSQL 重視 hot spot 問題，所以只用 8KB data block。 MySQL 重視 storage IO ，所以用 16KB data block

Free data block management 解說

- Free data block = 還有空間剩下，可以用來儲放 new rows 的 data block
- 就是說新 insert 的資料，應該怎樣找到有空間的 data block 儲存
- MySQL 採用 IOT ，所以 insert 是根據 PK 來決定位置，無需 Free data block management
- Oracle 用多個 linked list 存放 Free data block 的 pointer ，每次 insert new row 時從 list 中拿取 data block
- PostgreSQL 用特殊的 complete binary tree

MySQL update/delete 解說

- MySQL 採用 lazy deleteion
  在 update 比本來的大 / delete row 時，便用 garbage pointer 指向本來 row 位置
- 垃圾會在由 periodic background job 清理



SQL: update t set s="abcdefghijklmno" where i = 5;
Row is deleted, and a new row is inserted into the heap.

PostgreSQL update/delete 解說

- PostgreSQL 使用 lazy deletion ，所有 old version rows 留在本來位置，直到 Daily VACCUM job 清掉
- 新的 row 不一定存放於相同的 data block

Oracle update/delete 解說

- 在 update / delete 時， Oracle 會把 old version rows 放到 UNDO log ，然後寫在本來的 data block 上
- 新的 row 一定存放於相同的 data block
  - Data block 位置不足，便會把 data 放到其他 data block ，然後在原來 data block 上加上新位置的 pointer
  - 這個現象叫 row chaining ，並且會引起額外的 storage IO
  - 所以 Oracle 也需要定期的 table reorganize

Update / Delete 小結

- 任何的 Update / Delete ，都無可避免的讓 table structure 變爛
- 三個主流的 database ，都有事後的 background job ，在 database 不繁忙時把 table structure 變好
- 不用害怕 PostgreSQL 的 Daily VACUUM / Oracle 的 table reorganization ，這只是讓你自行決定什麼時候執行的 background task

Index Write Index 解說

- PostgreSQL / Oracle 的 B+ tree 在 page splitting / merging 時，會把改動中的 leaf node 和其上一層的 node 加上 WRITE_LOCK
- MySQL 的 B+ tree 在 page splitting / merging 時，整棵 B+ tree 都會加上 WRITE_LOCK
  - 因為 MySQL 是用 IOT 的，就是說：效果等同全個 table 被鎖掉耶 ~

總結： MySQL 的種種問題 (1)

- MySQL 的用上傳統的 Isolation ，用上 IOT ， 16KB data block ，其設計理念是想盡量省下 CPU 和 storage IO
  - 所以，低寫入量的系統， MySQL 效能比 Oracle / PostgreSQL 更好
  - 可是，一旦寫入量變大， Blocking / deadlock 會讓 MySQL 陷入等待，無法用盡全部的 CPU / Storage IO
- 而 PostgreSQL 先天性不容易發生 blocking / deadlock。即使 PostgreSQL 每件工作用的 CPU / storage IO 比 MySQL 多，但是 PostgreSQL 每件工作卻不容易互相干擾，讓系統的 CPU / Storage IO 被充份使用。所以其總效能比 MySQL 遠遠更好！

總結： MySQL 的種種問題 (2)

- MySQL 的 B+ tree 在 page splitting / merging 時，會讓整棵 B+ tree 被加上 WRITE_LOCK
  - ~~喵的，設計根本有問題啦！~~
  - 這問題要等到 5.7 正式出來後才能被修正
  - http://dev.mysql.com/worklog/task/?id=6326
- MySQL 不肯徹底重寫其架構。結果 MySQL 5.7 同時支持 READ_LOCK 和（有 ~~feature~~/bug ）的 MVCC
  - MSSQL 2008 也有相似問題。（而且更加嚴重！）

總結： MySQL 的種種問題 (3)

- MySQL 語法停留在 SQL:1999（ PostgreSQL 全面支持 SQL:2003 ）
  - Recursive-select
  - inline view
  - Rank()
- 這些超好用的功能， MySQL 都 ~ 沒 ~ 有 ~
- MySQL 的 Query optimizer ，也是三者中最弱的

第二節完
發問時間