

資料庫管理 (114-1)

作業四

作業設計：孔令傑
國立臺灣大學資訊管理學系

繳交作業時，請至 NTU COOL 下載本作業題目的地方上傳一個 PDF 檔。在生成這個 PDF 檔時，可以用打字的也可以用手寫的，但不管怎樣，請務必注意繳交的文件的專業程度（通常透過排版、文字圖片表格方程式的清晰程度、用字遣詞等面向呈現），如果專業程度不夠，也會被酌量扣分。每位學生都要上傳自己寫的解答。不接受紙本繳交；不接受遲交。可以用中文或英文作答。這份作業的截止時間是 **11 月 19 日早上 08:00:00**，遲交在 12 小時內者扣 10 分、在 12 到 24 小時內者扣 20 分、超過 24 小時的這份作業將得不到分數。

相關規定與提醒

1. 關於上網查詢與 AI 工具：

任何一份作業都可以被用任何方式完成，包括上網搜尋和使用各種 AI 工具。如果你想用，請留意以下幾件事。首先，抄襲還是不被允許的，如果我們發現抄襲（包括抄襲網路上的答案，或是抄襲同學的答案），都還是會給予嚴厲的懲罰（視情節輕重而定，通常是該份作業算零分，或者不予通過這門課）。只要沒有抄襲，那我們就只根據你交上來的答案的品質給分，不論你是自己想出來的，還是有利用 AI 工具。如果某甲善用了 AI 工具後寫得很好，某乙自己努力寫但寫得不好，那某甲會得到比較高分。其次，如大家所知，AI 工具給的答案可能會錯，也可能不合適。使用 AI 工具是學生的自由，但確認 AI 工具的答案是否合適、是否需要調整則是學生的任務。請務必自行確認答案的正確性與合適性。最後還是想提醒大家，學到多少東西都是自己的，如果一時困難用 AI 工具度過難關那是無妨，但之後建議還是花時間把東西學起來，對自己比較好。

2. 關於「專業」：

在我們這門課及許多課程中，都需要繳交各種報告。一份報告如果要達成他的效果（例如成功募資、完成溝通的任務等等），除了需要好的內容，也需要「專業」，而顯得專業通常需要「長得好看」以及「看起來用心」，這在沒有標準答案的任務上更是如此。有鑑於此，在這門課的作業和專案，我們都會要求報告的格式和美觀，並且納入評分標準。為此，我們提供報告格式參考指南「DB_reportFormatGuideline.pdf」，上面列舉了一份格式良好的報告的最低標

準。在作業一我們會請助教就違反參考指南的地方標出來讓大家知道（我們理解那份指南並不是最完美的，但如果完全沒有標準，同學們容易無所適從，所以我們還是設計一份當標準），但不會扣分，只是提醒大家；從作業二起就會有部分分數是報告專業度分數。之所以要求這些不是想要找大家麻煩，而是大家離出社會也不是太遠了，確實應該要開始被要求報告的可讀性和易讀性，所以我們願意花一些時間要求大家，但不會刁難大家，也請大家理解和盡力嘗試了。

3. 關於「批改」：

如課程大綱所述，為了不要累死助教，每次作業可能只有部分題目會被批改和給予回饋，但每一題的參考解答都會在作業截止後公佈。如果有一題沒被批改，那所有有寫那一題的學生都會得到那一題的全部分數，但沒寫或期限前沒交作業的自然就不會拿到那一題的分數。最後，請注意是「可能」，換言之也有可能是所有題目都被批改。

題目

1. (10 分) 上課的時候有說，PostgreSQL 裡面預設的交易隔離等級是「Read committed」，而且 PostgreSQL 裡的「Read uncommitted」跟「Read committed」其實是一樣的。換句話說，PostgreSQL 應該怎樣也不會讓你的程式發生 dirty read。

為了檢驗和體驗這件事，在本題中請寫兩個交易，各用一個 client 去執行。請明確地寫出這兩個交易的每一個作業 (operation)，也明確地寫出你用怎樣的順序執行這些作業，然後說明你如何在一個交易中有 uncommitted 的 update 去更新某筆資料，在 commit 該 update 前另一個交易中的某個作業讀取了那筆資料，而在這次讀取中又得到什麼。最終請說明 PostgreSQL 是否確實有阻止 dirty read。

你可以用純粹 SQL 語法去構成你的交易，也可以用 PHP、Python、Javascript 或其他還算大眾化的語言。請直接複製貼上你的程式碼繳交、寫前面要求的說明，並且用螢幕截圖說明所有作業開始前的資料狀態，以及讀取後得到的資料狀態。

2. (30 分) 在表 1 和表 2 展示了兩個交易 T_1 、 T_2 ，各有七和六個作業 (operation)。為了方便大家作答，我們用 O_{ij} 代表交易 T_i 的第 j 個作業。

- (a) (5 分) 請寫出這兩個交易中所有互相衝突的作業 (operation) 配對。作答時請直接使用 O_{ij} 作答即可。

提示：印出資料的作業不牽涉到資料庫，自然也不可能跟任何作業衝突。

作業編號	作業
O_{11}	read_item(X)
O_{12}	read_item(Y)
O_{13}	read_item(Z)
O_{14}	$Z \leftarrow X + Y$
O_{15}	write_item(Z)
O_{16}	$X \leftarrow 100$
O_{17}	write_item(X)

表 1: 交易 T_1

作業編號	作業
O_{21}	read_item(X)
O_{22}	$X \leftarrow X + 10$
O_{23}	write_item(X)
O_{24}	read_item(Z)
O_{25}	$Z \leftarrow Z + X$
O_{26}	read_item(Z)

表 2: 交易 T_2

(b) (5 分) 假設實際上是 T_2 先發生然後 T_1 才發生，請寫下這個前提下的序列式行程 (serial schedule)。接著請寫出任意一個和此序列式行程等價 (equivalent) 的非序列式行程 (non-serial schedule)，並說明它們為何等價。請從作業間的關係說明，不要用執行結果相同與否說明。如果你認為沒有任何一個非序列式行程可以跟先 T_2 再 T_1 的序列式行程等價，請就寫沒有並且說明你這麼認為的原因。在寫序列式和非序列式行程時，你只要列出資料存取 (包含讀取和寫入) 作業即可。

(c) (5 分) 假設實際上是 T_1 先發生然後 T_2 才發生，請寫下這個前提下的序列式行程 (serial schedule)。接著請寫出任意一個和此序列式行程等價 (equivalent) 的非序列式行程 (non-serial schedule)，並說明它們為何等價。請從作業間的關係說明，不要用執行結果相同與否說明。如果你認為沒有任何一個非序列式行程可以跟先 T_1 再 T_2 的序列式行程等價，請就寫沒有並且說明你這麼認為的原因。在寫序列式和非序列式行程時，你只要列出資料存取 (包含讀取和寫入) 作業即可。

(d) (10 分) 請用兩階段鎖定 (two-phase locking) 的機制幫 T_1 和 T_2 這兩個交易加上合適但盡可能少和輕量的讀取鎖 (read lock) 和寫入鎖 (write lock)，讓資料庫在排程時，不會排出不可序列化 (non-serializable) 的非序列化行程 (non-serial schedule)。請使用 $\text{read_lock}(X)$ 代表幫 X 加讀取鎖、 $\text{write_lock}(X)$ 代表幫 X 加寫入鎖、 $\text{unlock}(X)$ 代表幫 X 解鎖。如果你曾經對資料 X 加過讀取鎖，當你想要升級成寫入鎖時，請直接寫 $\text{write_lock}(X)$ ；如果你曾經對資料 X 加過寫入鎖，當你想要降級成讀取鎖時，請直接寫 $\text{read_lock}(X)$ 。請寫下加上跟鎖相關的作業之後的兩個交易，並且說明為什麼資料庫不會排出不可序列化的非序列化行程。

在回答本題時，大家幫資料上鎖的程度應該愈小愈好。具體來說，如果用兩

個鎖就能達成，就不要用三個鎖；如果用讀取鎖就能達成，就不要用寫入鎖；如果晚點加鎖仍能達成，就不要早加鎖；如果早點解鎖就能達成，就不要晚解鎖；依此類推。

- (e) (5 分) 承上題，請針對你在前一小題寫下的包含加解鎖作業的交易，寫出任意一個會發生死鎖 (deadlock) 的非序列式行程 (可以寫到發生死鎖為止即可，不一定要寫完)。請說明為什麼會發生死鎖。如果你認為不可能發生死鎖，請說明為什麼不可能。
3. (10 分) 上課時介紹過，我們可以透過幫每個交易用一樣的規則加解鎖去防止 uncommitted read、unrepeatable read 或 phantom read。請設計一套加解鎖的規則去防止 lost update。你得到的規則可能跟課本上的四個規則的其中之一一模一樣，也可能都不一樣。在說明你的規則後，請再說明為什麼它可以防止 lost update。
4. (25 分) 針對上課曾使用過的 THSR 資料庫，請考慮這句 SQL 指令：

```
SELECT
    p1.trip_id,
    p1.depart_time AS departure_time,
    p2.arrive_time AS arrival_time,
    (
        SELECT COUNT(*)
        FROM RESERVED_TICKET
        WHERE trip_id = p1.trip_id
            AND depart_station_id = 1030
            AND arrive_station_id = 1000
            AND travel_date BETWEEN '2023-08-01'
            AND '2023-08-31'
    ) AS reserved_sales
FROM PASS p1
    JOIN PASS p2 ON p1.trip_id = p2.trip_id
WHERE p1.station_id = 1030
    AND p2.station_id = 1000
    AND p1.depart_time > '06:00:00'
    AND p1.depart_time < p2.arrive_time
ORDER BY p1.depart_time
LIMIT 10;
```

注意：本題有許多小題都要求大家去觀察 query plan 和 estimated total cost，但不同電腦、不同 RDBMS 的 estimated total cost 都可能不一樣，甚至 query plan 也可能不一樣。

- (a) (5 分) 請用自己的話說明這句指令在做什麼，包含該指令使用哪些資料表、做那些合併、基於什麼做合併、做哪些彙總和篩選、最終輸出什麼。
 - (b) (5 分) 在沒有任何 index 的情況下，請截圖貼上你的 RDBMS 產出的 query plan。這個 query plan 的 estimated total cost 是多少？
 - (c) (5 分) 請依序建立三個 single-column index (用預設最陽春的方式建立就好，且不需要用到 INCLUDE)，對象分別是 `trip_id`、`depart_station_id` 和 `arrive_station_id`。請完整地寫下你建立這三個 index 的語法，然後逐一檢視在只有該 index 存在時的 query plan。請截圖貼上你的 RDBMS 產出的三個 query plan，並且比較它們的 estimated total cost。使用個別 index 得到的 estimated total cost 有沒有大小差別？不論有或沒有，你認為是為什麼？請用自己的話提出解釋。
 - (d) (5 分) 請幫前一小題的三個欄位建立一個 multi-column 的 index (用預設最陽春的方式建立就好，且不用到 INCLUDE)；由於順序有差，請讓 `trip_id` 最優先、`depart_station_id` 次之、`arrive_station_id` 最後。請完整地寫下你建立這個 index 的語法，然後檢視在只有該 index 存在時的 query plan (換言之，你應該移除在前一小題建立的 index)。請截圖貼上你的 RDBMS 產出的 query plan，並且寫下它的 estimated total cost。以上我們一共有五個執行該 SQL 指令的方法 (沒有 index、三個 single-column index、一個 multi-column index)，請比較這五種方法的 estimated total cost。五種方法得到的 estimated total cost 有沒有大小差別？cost 小的是為什麼小，cost 大的又是為什麼大？請用自己的話提出解釋。
 - (e) (5 分) 請在只有各 index 存在時手動執行這句 SQL 指令並且觀察執行時間。estimated total cost 小的 query plan 是否有小的執行時間？不論你觀察到什麼，請用自己的話提出解釋。
5. (15 分) 請考慮以下數列：800, 400, 600, [700], 900, 650, 750, 200, 300, [350], 375, 380, 420, 500, 410, 415, 416, [418]。
- (a) (3 分) 把這個數列由左到右依序插入到一個原本是空的、一個 inner node 最多向下分支數為 4、一個 leaf node 最多存 3 個 key 的 B+ tree 中，並且把最終結果畫出來。你可以用老師上課教的那個網頁沒問題，也可以在完成後直接螢幕截圖做繳交。

- (b) (3 分) 請用你的話解釋，插入數列中框起來的那個 700 的時候，這棵 B+ tree 是怎麼調整它自己的。你可以把插入 700 前和後甚至過程中的 B+ tree 畫出來，方便你做解釋。你可以螢幕截圖，但你必須用自己的話說明這個流程。
- (c) (3 分) 請用你的話解釋，插入數列中框起來的那個 350 的時候，這棵 B+ tree 是怎麼調整它自己的。你可以把插入 350 前和後甚至過程中的 B+ tree 畫出來，方便你做解釋。你可以螢幕截圖，但你必須用自己的話說明這個流程。
- (d) (3 分) 請用你的話解釋，插入數列中框起來的那個 418 的時候，這棵 B+ tree 是怎麼調整它自己的。你可以把插入 418 前和後甚至過程中的 B+ tree 畫出來，方便你做解釋。你可以螢幕截圖，但你必須用自己的話說明這個流程。
- (e) (3 分) 請把這個數列排序，然後用 bottom-up 的方式建構一棵 B+ tree，這棵 B+ tree 的 inner node 的 order 為 4 (最多指向 4 個 child node)、leaf node 的 order 為 3 (每個 leaf node 最多放 3 個 key 值)。建構時，請讓每個 leaf node 一律都含有 2 個元素 (但如果數列中的數字有奇數個，則最後一個 leaf node 裡面請放 3 個元素)，而除了最後面的 inner node 可以只指向 2 個 child node，前面的每個 inner node 都是指向 3 個 child node。請把完成後的完整的 B+ tree 畫出來。

注意：這樣的題目在期末考也可能會出現。我們固然接受大家在寫作業時用工具輔助作答，但請大家確保自己確實理解觀念（進而在只有紙筆的環境下也有辦法作答）。

6. (10 分) 承第四題，請問上課教過的各種 index 中，還有沒有什麼 index 至少在原則上可以進一步加速第四題的 SQL 查詢？如果你認為沒有，請說明你的原因；如果你認為有，請說明是哪種 index，並且說明應該怎麼建立那樣的 index (例如直接寫下建立該 index 的 SQL 語法)，並且說明為什麼那樣的 index 至少原則上可以進一步加速。

本題所說的「各種 index」是指 multi-column index、covering index、partial index 之類的，不是指 B+ tree index、hash index、bitmap index 這個層級的。