資料庫管理（113-1）
作業五

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。可以用中文或英文作答。這份作業的截止時間是 ***12 月 4 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）幾乎所有現在常用的程式語言都支持交易管理。舉例來說，如果我們使用 PHP 裡面的 `PDO` 架構，程式碼大概會像是[1]：

```php
$pdo = new PDO("pgsql:host=hostname;dbname=database",
               "username", "password");

$pdo->beginTransaction(); // transaction begins

$query = "INSERT INTO table_name (column1, column2)
          VALUES (:value1, :value2)";
$stmt = $pdo->prepare($query);
$stmt->execute(['value1' => $value1, 'value2' => $value2]);

$pdo->commit(); // commit!

// or $pdo->rollBack(); if necessary
```

   在本題中，請用「非 PHP」的任意程式語言改寫投影片「DB113-1_12_transaction.pdf」第 11 頁的程式碼。你寫的程式不會在 SQL 環境中執行 `BEGIN`、`END` 和 if-else 選擇，而是在你選擇的程式語言中執行（類似上面在 PHP 中執行），當然拋出和處理例外的程式碼也應該寫在你選擇的程式語言而非 SQL。請直接複製貼上你的程式碼繳交，並且寫一段簡單的文字說明你的程式碼如何運作；不用繳交任何其他東西（例如不需要註解、不需要繳交運算後的結果等等）。

2. （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，

[1] 當然如果要用 ORM 架構或 PHP 的 `pg_connect` 架構也可以，大家可以自行查詢語法。

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。

請直接複製貼上你的程式碼繳交，並且寫前面要求的說明，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。

3.（20 分）在投影片「DB113-1_12_transaction.pdf」第 52 頁有兩個交易。

(a)（2 分）請把這兩個交易中跟鎖（lock）有關的作業（operation）拿掉，並寫出只包含剩餘作業的兩個原始交易。兩個原始交易應該各包含四個跟鎖無關的資料存取與運算作業。

(b)（2 分）承上題，請寫出所有互相衝突的作業配對。

(c)（2 分）承 (a) 小題，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排程（serial schedule）。你只要幫 (a) 小題的八個資料存取與運作作業排序即可。

(d)（2 分）承上題，請寫出任意一個和 (c) 小題的序列式排程等價（equivalent）的非序列式排程（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排程可以跟 (c) 小題的序列式排程等價，請就寫沒有並且說明你這麼認為的原因。

(e)（2 分）承上題，請寫出任意一個和 (c) 小題的序列式排程不等價的非序列式排程（non-serial schedule），並說明它們為何不等價。請從作業間的關係說明，不要用執行結果相同與否說明。

(f)（4 分）承 (a) 小題，請用兩階段鎖定（two-phase locking）的機制幫 (a) 小題的兩個原始交易加上合適的讀取鎖（read lock）和寫入鎖（write lock），讓每次資料庫讀寫都有受到必要且不過多的鎖定保護。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，請寫 `upgrade_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，請寫 `downgrade_lock(X)`。請寫下加上跟鎖相關的作業之後的兩個交易。

(g)（3 分）承上題，請針對你在 (f) 小題寫下的完整交易，寫出任意一個沒有死鎖（deadlock）和 (c) 小題的序列式排程等價的非序列式排程（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。

(h)（3 分）承 (f) 小題，請針對你在 (f) 小題寫下的完整交易，寫出任意一個有死鎖（deadlock）的非序列式排程（non-serial schedule）。

4.（20 分；每小題 5 分）有一個火車服務，該服務中針對每次列車都只發售張數有上限的無對號車票、必須要是會員才能買票、一個人一次只能買一張票、不可退票。你負責該火車服務的票務系統，其資料庫中有資料表 `REMAINING_TICKET` 跟 `PURCHASE`，分別記錄每天每班車的剩餘票數以及所有會員的購票紀錄。資料表綱要（schema）如下：

```
REMAINING_TICKET(travel_date, train_id, remaining_ticket_qty)
PURCHASE(customer_id, travel_date, train_id, purchase_datetime)
```

(a) 上課有提到交易之間的隔離等級要到什麼程度，是要由系統開發（管理）者決定的，並且舉了上網買車票跟臨櫃買車票的例子。假設該公司對上網買車票的規定流程是：

i. 消費者選擇乘車日期與班次後，查詢得到該日該班次的剩餘票數。如果沒票了就重新選擇，如果還有票則進入下一步驟。

ii. 消費者考慮要不要購票，要的話就輸入會員帳號、會員密碼、驗證碼、信用卡資訊等等，然後按下確定並進入下一步驟；如果不要的話就結束整個流程或回到步驟一。

iii. 系統再次檢查該日該班次是否仍有剩餘車票，如果有則更新 `REMAINING_TICKET` 將該日該班次的剩餘票數減一，並且在 `PURCHASE` 裡面新增一筆相關紀錄，然後給消費者一張電子車票；如果在步驟二的過程中票賣光了，就跟消費者說抱歉，並且不更新資料庫。

請幫步驟一的查詢、步驟三的再次查詢、步驟三的更新和步驟三的新增各寫一句 SQL 指令（一共四句，請自行假設使用者輸入的值與當下的日期與時間）。

(b) 承上題，如果這四句 SQL 指令（和一些寫在某處的 if-else）被包成一個交易，請問該交易是否要防止「unrepeatable read」，為什麼？

(c) 承上題，請在這個交易裡面的合適之處加必要且不過多的鎖，以確保你應該確保的交易完整性與正確性。

(d) 假設該公司對臨櫃買車票的規定流程是：

i. 消費者告知櫃台售票員乘車日期與班次後，售票員查詢得到該日該班次的剩餘票數。系統檢查該日該班次是否仍有剩餘車票，如果有就立刻幫該消費者暫時保留一張票，接著進入下一步驟；如果沒有就顯示沒有，售票員會跟消費者說抱歉，消費者可以離開或重新選擇。

ii. 消費者考慮要不要購票，要的話就在櫃檯的設備上輸入會員帳號、會員密碼等等，然後把信用卡或現金交給售票員，接著進入下一步驟；不要的話就跟售票員說不要，系統就取消幫該消費者暫時保留的車票。

iii. 系統更新 `REMAINING_TICKET` 將該日該班次的剩餘票數減一，並且在 `PURCHASE` 裡面新增一筆相關紀錄，然後售票員給消費者一張電子或實體車票。

如果步驟一的查詢、步驟三的更新和步驟三的新增各對應到一句 SQL 指令，且這三句 SQL 指令（和一些寫在某處的 if-else）被包成一個交易，請在這個交易裡面的合適之處加必要且不過多的鎖，以確保你應該確保的交易完整性與正確性。

5.（10 分）一般來說，關聯式資料庫的弱點是比較不利於分散，亦即把一個關聯式資料庫拆在多臺機器上運行容易效益不彰。請用你自己的話說明為什麼。

6.（15 分；每小題 5 分）NoSQL 和資料庫分片（sharding）是在處理超大量資料時人們會考慮的技術。關於 NoSQL 和分片，請用你自己的話回答以下問題：

(a) 某公司對許多企業提供輿情調查的資訊服務，服務內容如下。首先，該公司每天定期去網路上爬取各新聞媒體、網路論壇、社群媒體等資訊來源的文章，並且儲存在自家的資料庫。該公司的客戶接著就可以針對自己的品牌、自己的商品、競爭者的品牌、競爭者的商品等等做搜尋，該公司的系統就會把搜尋到的所有相關文章呈現給該客戶（如果他想看），也會提供從這些文章彙總得到的相關分析（包含數量、趨勢、聲量比較、情緒傾向等）。

此公司苦於資料量太多（收著收著就有上億篇文章了），因此使用了 NoSQL 而非關聯式資料庫來儲存這些文章。你認為合理嗎？為什麼？請從此應用的性質與關聯式資料庫和 NoSQL 的性質進行分析。

(b) 承上題，此公司希望用分片方式來進一步提升效率。請建議他們該如何分片（例如按客戶分、按時間分、按地區分、用 hash 分等等），並說明原因。

(c) 請考慮你的期末專案的主題。假設在該應用中你要做分片，你會對哪些（個）資料表做分片，你又會怎麼分[2]？為什麼？

7.（15 分；每小題 5 分）資料倉儲（data warehouse）是許多公司建立商業智慧（business intelligence）與決策支援系統的基礎。這些公司首先彙把大量歷史資料儲存在資料倉儲中，接著會定期由商業智慧系統對資料倉儲中的資料做例行性的分析與視覺化（例如呈現在儀表板上給決策者做為決策參考），或者不定期地由分析師對資料倉儲中的資料做特定目的的分析與視覺化。

針對資料倉儲，請用自己的話回答以下問題：

(a) 相較於讓分析師直接進入公司營業用的資料庫做分析，將資料從營業用的資料庫匯入資料倉儲再讓分析師到資料倉儲中做分析，有什麼好處壞處？

(b) 一個資料倉儲中會存放大量資料。考慮到這個以及資料倉儲的應用場警，你會用關聯式資料庫還是 NoSQL 去建立資料倉儲？為什麼？

(c) 資料倉儲中的資料經常被以行式儲存（column store）的方式儲存，而不是列是儲存（row store）。這麼做有什麼好處壞處？請舉一個「該運算在 column store 中會跑得比在 row store 中快」的例子。

特別留意：在臺灣，一個 row 是一列，一個 column 是一行或一欄。

[2]雖然你在期末專案會用關聯式資料庫，但你還是可以考慮分片（sharding）；你可以想成是你改用了 NoSQL 資料庫，或者你使用了關聯式資料庫的分片技術。總之，請不要在意「關聯式資料庫相對較不易在分散式架構運行」這個事實。