ITWS 4250/6250 — Database Applications and Systems
Fall 2020 Take-Home Exam

Overview

- This exam consists of a "theory" component, and a "practical" component.
- You have until **9:30 P.M.** on December 16 to submit your solutions to both components.
- This exam will count as 10% of your final course grade.
- All work must be done individually.
- This exam is open book and open notes. That includes any online documentation needed for the practical component.

The "theory" component of the exam should be completed manually, either by printing this document, adding your solutions by hand, and submitting a scan or photo of your hand-written solutions, or by creating a document of some kind with your solutions. Please limit your submissions to text documents, pdf documents, or images (in other words, please refrain from submitting `.doc` or `.docx` type documents). The "theory" component is worth **40 points**.

The "practical" component of the exam is described in the `readme.md` document included in the same directory of the course git repo as this file. The "practical" component is worth **60 points**.

For the purposes of this exam, assume you've been hired to work on a software application that will be used by a store selling used records and CD's.

You've been informed that the database will need to use the following relations:

- $Album(\underline{albumId}, artistId, title, year, publisher)$
- $Song(\underline{title}, trackNumber, \underline{albumId}, length, genre, composer)$
- $Artist(\underline{artistId}, name, birthdate)$
- $Inventory(albumId, numberInStock, price, upc)$
- $Streaming(songTitle, albumId, provider, \underline{url})$

Album describes information related to an individual record or CD. Assume that 127 characters is sufficient for the title.

Song describes the individual song tracks on an Album. Length should be in seconds. Genre will be between one and four genre labels, separated by commas.

Artist describes the artist who performed the songs on the Albums. Assume that 255 characters might be needed for their names.

Inventory describes the inventory currently available in the store. *upc* is the "Universal Product Code"–it's the numerical representation of the barcode you sometimes see on labels. The number is effectively random (there's no ordering to the system). An Integer won't be sufficient to hold it.

Streaming describes where the songs may be found (legally) online, should patrons be interested.

Album.albumId and *Artist.artistId* are both 32 bit integers.

1. (a) (3 points) Because the primary key for *Song* is $(title, albumId)$, Postgres automatically creates a B-Tree index for it. Assume that our database is running on a system with a block size of 16384 bytes and an address length of 8 bytes. What is the maximum number of pointers that can be contained in an interior node of the B-Tree (remember that `VARCHAR` fields require an additional character for null-termination)?

   (b) (5 points) Why is it preferable to define the key as `PRIMARY KEY (title, albumId)` rather than `PRIMARY KEY(albumId, title)`?

2. One of your coworkers wrote a Python web application to allow customers to find out what's in stock based on album title and year. Part of the application sets up a connection to the database, creates a cursor on that connection, then passes the cursor to the following function along with values for the year and the name of the album (taken from user input on a webpage):

```
def find_albums(year, album_title, cursor):
query = \
"SELECT a.title, i.numberInStock, i.price" \
"FROM Album a join Inventory i on a.albumId = i.albumId" \
"WHERE a.title ilike '%" + album_title + "%'" \
"AND a.year > %s"
cursor.execute(query, (year,))
return [
(title, number, price) for title, number, price in cursor.fetchall()
]
```

   (a) (3 points) Which parameter of the `find_albums` function is vulnerable to SQL Injection?

   (b) (5 points) Explain how you would fix your coworker's code

3. Assume your database uses Redo logging for transaction management. (Postgres actually uses a form of Redo logging called "Write-Ahead Logging," but you should assume the simplified version covered in class and the textbook).

   After a series of sales and processing of new inventory, a janitor accidentally unplugs the computer hosting the database server. When the system is powered back up, the *Inventory* relation and the log exist as defined below (*upc* and *price* have been intentionally omitted for simplicity). Note that the log update statements all have the form $\langle t, a, n \rangle$ where t is the transaction id, a the address for the block containing the tuple with the corresponding album id, and n the value for a.

Inventory

albumId	numberInStock
10	6
15	2
22	5
18	4
30	6

Log

Start T1
$\langle T1, 10, 3 \rangle$
$\langle T1, 18, 5 \rangle$
Start T2
$\langle T1, 22, 1 \rangle$
$\langle T2, 15, 3 \rangle$
Commit T1
$\langle T2, 30, 7 \rangle$
Start T3
Start T4
Commit T2
$\langle T4, 15, 4 \rangle$
Start T5
$\langle T5, 15, 2 \rangle$
$\langle T3, 22, 6 \rangle$
$\langle T3, 10, 6 \rangle$
Start T6
Abort T4
$\langle T5, 22, 5 \rangle$
$\langle T6, 18, 4 \rangle$
Commit T3
$\langle T6, 30, 6 \rangle$

   (a) (3 points) Which transactions should have their actions reflected in the *Inventory* relation after recovery is complete?

   (b) (10 points) After the system uses the log to reconstruct the *Inventory* relation, what will the proper values be?

For the last two questions, don't assume any connection to the Record Store scenario.

4. (4 points) Assume relations R and S stored on disk. R is clustered and fits in 1024 blocks. S is also clustered and fits in 8192 blocks. Your physical query plan requires a set union between R and S ($R \cup S$). You have 256 blocks of memory available for your algorithm. How many Disk I/Os are required for the operation? Please show any work involved in arriving at your answer.

5. Assume the existence of a database for the purposes of gathering Census data.

   (a) (4 points) Alice creates a schema including a table named `Housing`. She then delegates privileges to Bob by executing `GRANT SELECT ON Housing TO Bob WITH GRANT OPTION;`
   What happens when Bob runs `GRANT SELECT ON Housing TO Carol`?

   (b) (3 points) Bob leaves to take a job elsewhere, so Alice runs `REVOKE SELECT ON Housing FROM Bob RESTRICT;`
   What happens, and why?