Lecture 8

COMP207

Survey

- Since I mistakenly did not upload it last week I wanted to go over the comments that came in by yesterday so that it is not too slow
- Two comments so far:
  - Positive:
    - I like the structure of the lectures, videos and the examples given in the lectures themselves. I think the videos do well to explain how commands work, and the lectures are good at putting it into context.
    - Learn more about SQL.
  - Negative:
    - When giving demonstrations in SQL sometimes it's hard to read what is being written and hard to view the output.
    - Feel a little bit difficult about the course this week.

Survey response

- Positive: Glad to hear that the structure is nice! I am also happy to hear that you liked learning more about SQL (I am aware that this is two different students)
  - Response:
    - First negative comment (trouble seeing in MySQL): I hope it helps when I zoom in on the input text and will try to keep doing that and I will try to use the magnifier glass (Windows accessibility tool) to help with the output
    - I assume the second one (harder to follow) is about ACID (even if the corresponding positive one is about SQL - if it was because of GROUP BY, I understand that, but try to focus on how to use it instead of just what it is): If so, I hope it will get easier this week and next week as we cover more how we can implement them

This weeks topic

- This week we are talking about locks and logs

Basic Locks

- Basic locks on an item consist of locks and unlocks on that item
  - Each item has a padlock
  - If you ask the schedule for the padlock, it will give it to you (if it has it – otherwise you must wait until it comes back) and you must have it to read/write to the item
  - You must give up the lock afterwards
- In short-hand notation: $l_1(X)$ and $u_1(X)$ for locking and unlocking of X in transaction 1
- (Go over example from last time)

Shared and exclusive locks

- There are different locks for reading (shared lock) and writing (exclusive lock)
  - You can read with the write lock also though – they are just better
  - You can get a shared lock for the item as long as none else has an **exclusive lock** on it
  - You can get an exclusive lock for the item as long as none else has a **lock** on it
- In short-hand notation: $sl_1(X)$ and $xl_1(X)$ for shared-locking and exclusive-locking of X in transaction 1
- (Go over example from last time again)

Also: Update locks

- Third type of lock: Update lock
- **Change:** When using those, you can't have a shared-lock and an exclusive-lock on an item
- Update locks are mostly like shared lock, except:
  - You can get an update-lock AND an exclusive-lock
  - The rules for granting locks for shared and update locks is that none else can have an **update-lock** or an **exclusive-lock**
    - This often confuses people: It is NOT symmetric

Scenario: Person with many friends on Facebook

- **Disclaimer:** Facebook is not that public with these kind of things and I am not sure if this is how they do it
- Imagine you have a lot of friends and people might get information from your page quite often
  - Each time, your page would be read
- If you want to write something you could get an upgrade lock and then ask for an exclusive lock
  - Because of the assymmetry, you would just need to wait for the last person that started reading before you got the upgrade lock
- If you went for an exclusive lock initially, you could wait forever (if nothing is done to prevent it): This is called starvation

Multiple granulairty locks

- What is an item?
  - You could use items as whole tables
    - Positive: Only a few locks are needed even for large queries
    - Negative: You often have to wait even if you are just using a small query
  - You could use items as individual entries
    - Positive: A small query rarely has to wait much
    - Negative: A big query requires many locks
  - Multiple granularity locks are trying to do both fast:
    - Can (shared/exclusive)-lock a table
    - Can (shared/exclusive)-lock an entry
    - But how do you tell if someone has a lock on an entry of your table? Do you need to look at all entries?!?

Intention locks to the rescue

- To allow both locks on tables and entries (or even more levels), we have intention locks
- Intention (shared/exlusive) locks means that you want an (shared/exclusive) lock on a subitem

Transaction requests lock of type …

	Shared (S)	Exclusive (X)	IS	IX
Shared (S)	yes	no	yes	no
Exclusive (X)	no	no	no	no
IS	yes	no	yes	yes
IX	no	no	yes	yes

Grant if the only types of locks held by *other* transactions are those with a "yes"

2PL

- In each transaction, each lock must come before every unlock
  - You do not need to unlock instantly, just some time after you finish using the item
- This ensures conflict-serializability

What would MySQL do?

- Has shared/exclusive locks
- Has multiple granularity locks
  - Specifically: table, gap (like interval) and individual records
- Does use 2PL
- No update locks
  - I think – the part of the MySQL manual about locks does not mention it but it is hard to search for

- Let's see some examples!

Do exam questions from two years ago