CS 325 - Final Exam Review Suggestions - Fall 2021

last modified: 2021-12-06

- You can receive (a maximum) ***5 POINTS BONUS**** on the Final Exam if you do the following:
  - Make a **hand-written** Final Exam study sheet.
  - Submit a photo or scan of it saved as a .pdf, .png, .jpg, or .tiff to Canvas **by **8:00 am** on Monday, December 13** such that I can read at least some significant **post-Exam-2** CS 325 Final Exam material on it
  - Please let me know if you have any questions about this, and I hope it helps you in reviewing course concepts more effectively before the Final Exam.
  - You are **encouraged** to have this available as you are taking the Final Exam.
- The Final Exam will be given in Canvas.
  - It will be **available from 8:00 - 10:00 am on Monday, December 13**.
  - It will be set up such that **you may only attempt the Final Exam once**.
  - That said, remember that **I will be available in the regular Zoom CS 325 session from 8:00 - 10:00 am on Monday, December 13.**
    - This will be set up so all microphones are muted, and you will only be able to chat with me.
    - **You are encouraged to LET ME KNOW about technical difficulties, so we can make provisions as needed!**
    - **And, I will have a small displayed window in the Zoom session with any typos or announcements that have come up so far during the Final Exam.**
  - The Final Exam will be set up such that **you will be shown one question at a time**,

    BUT there will be a list of question-links on the right-hand-side of the Canvas screen, and you can go back and forth between questions during that **one** exam attempt.
  - You are expected to work **individually** on the exam -- it is not acceptable during the exam to discuss anything on the exam with anyone else.
  - You may look up information from your Final Exam study sheet, from course reading packets, from posted examples, etc., during the exam, but note that if you take too long looking up material, you may have trouble completing the exam during the time period.
  - I expect there will be a few multiple-choice questions, and the rest will be short-answer questions.
    - BUT given the Canvas quiz question options, for those short-answer questions, you will be shown a larger area for your answer than you will need!
    - You will be reading and writing SQL statements and SQL*Plus commands.
    - You will be answering questions about concepts as well.
- A link to a packet of references and additional instructions - intended for use with the Final Exam - will be posted from 8:00 to 10:00 am on the course Canvas site, linked from the Final Exam Instructions.
  - So, you can have it open in another browser window while you are taking the Final Exam.

  - This is intended both for reference and for use directly in some exam questions.
- You are responsible for material covered in lectures and labs, and especially anything that's been on a homework, lab exercise or the course project; BUT, here's a quick overview of especially important material.
- Your studying should include careful study of posted examples and notes as well as the lab exercises and homeworks thus far.
- Note that you are also responsible for knowing -- and following -- the course SQL style standards and the course ERD notation.
- The Final Exam is **cumulative**.
  - If it was fair game for Exam 1 or Exam 2, it is fair game for the Final Exam.
  - So, studying the review suggestion handouts for Exam 1 and Exam 2 would be wise; (they're still available from the course web page, under "Homeworks and Handouts").
  - Expect quite a bit of SQL,

    ...and note that you will be asked to write at least one query **involving** a join. (It might be an equi-join or a query involving further projection or selection from an equi-join.)

DBMS Support for Various Integrities

- EXPECT a question related to DBMS support for **entity integrity**, **domain integrity**, **referential integrity**, and **transaction integrity**.
  - For example, be able to answer questions such as those in **Homework 10 - Problem 3**.

Transaction Management and Concurrency Control

Transactions

- What *is* a **transaction**?
- What are the 5 main database transaction properties (to ensure database integrity in the presence of concurrency)? (Hint: ACIDS)
  - Expect a question trying to determine if you know what is means for a transaction to be ATOMIC --- what does it mean? (What is atomicity?)
  - Serializability --- what is that? why is it desirable? what are several ways of achieving it? etc.
  - Database durability --- what is that? why is it desirable? what are some means for attempting to provide this? etc.
  - What is meant by a consistent database state?
  - What is meant by isolation?
  - How does SQL support transactions? (`commit`, `rollback`)

    The exam could include fragments of code that include `commit;` and `rollback;` statements, and then ask questions to see if you know their effects.
  - For example, be able to answer questions such as those in **Homework 10, Problem 4**.

Database Recovery

- What is **database recovery**?
  - Transaction logs are one way of implementing/providing support for database recovery.
  - How can they be used in such recovery?
  - What do the concepts of rollforward, rollback mean in such recovery?
  - Be able to answer database recovery questions such as those in **Homework 10, Problem 2**.

Concurrency Control

- **concurrency control** - how do you handle MULTIPLE, concurrent transactions?
  - how can LOCKS and TIMESTAMPS help to do this?
- what kinds of problems can occur with transactions?
  - lost updates, inconsistent reads, etc.
  - (as well as "effects" of concurrency control such as deadlock, starvation, etc.)

Locks

- what are they? how can they be used?
- **binary** locks
- **shared/exclusive** locks (also called **read/write** locks)
  - what are the differences between these in terms of memory? overhead? potential concurrency? (TRADE-OFFS between them...)
- what do we mean by lock **granularity**?
  - lock granularity is how much is being locked at a time;
  - what are the tradeoffs there?
  - (notice the tradeoffs --- smaller granularity --> more potential concurrency, but also more overhead (space for locking bits, time to handle locks)
- **two-phased locking** is needed with locks to achieve **serializability**
  - you have a GROWING phase and a SHRINKING phase
  - that is, a transaction can obtain locks as needed (growing phase) BUT once it gives up a lock, it CANNOT obtain any more (shrinking phase)
  - (also discussed a more restrictive (and easier to implement) version of this in which the system doesn't release ANY locks until the transaction is committed or rolled back -- the "growing" phase is the whole transaction's "lifetime"! More draconian, but easier to implement)
- locking CAN lead to **deadlocks** --- what are they, what are some ways of dealing with this problem?

Timestamps

- timestamps are an ALTERNATIVE to locking for concurrency control

- know BASICS of this approach, as discussed.
- basic idea: EACH transaction gets a unique timestamp, CONFLICTING operations must be done in TIMESTAMP order
  - (so, if a transaction wishes to perform a conflicting action that would VIOLATE timestamp order, it is TERMINATED, rolled back, and restarted (so it gets a BIGGER timestamp))
  - note tradeoffs, though --- could take more space (depending on granularity, of course) than locking, because EVERY (unit) has a read timestamp and a write timestamp...

Optimistic Methods

- talked BRIEFLY about optimistic methods, too --- know just the BASIC idea there
  - you only have to know this to the extent discussed in DB Reading Packet 10 - "Transaction management, part 2"

Databases, society, and ethics

- What are some of the issues related to databases, society, and ethics?

LAB-RELATED TOPICS:

- note: **EXPECT** to write at least one `update` statement and `delete` statement
- note: **EXPECT** to write at least one query **involving** a join. (It might be an equi-join or a query involving further projection or selection from an equi-join.)
- note that, at this point, you should be able to answer questions about, for example:
  - what order tables must be created,
  - what order rows must be inserted, and
  - what the effects of referential integrity constraints are --
  - those are from earlier in the semester, but **creating** and **populating** your project tables should have made these clearer!

SQL `rollback` and `commit`

- what do `rollback;` and `commit;` provide basic support for?
- when are automatic commits done in SQL*Plus?
- should be able to read, write, use these; given a sequence of statements, you should be able to describe the effects of these.

SQL*Plus statements supporting ASCII reports

- what does `clear` do, in SQL*Plus? What are things you might want to `clear` before a report?
- what is meant by `feedback`? how can you set it to a certain level, or turn it off?
- set `pagesize`, `linesize`, `newpage` - what do these mean?
- `ttitle`, `btitle` - what do these do? Be able to read, write them;

- be able to read, write, use, understand their purpose for all of the above;
- what should you be careful to do at the beginning of a SQL script used to create a report? what should you be careful to do at the end of a SQL script used for creating a report?
- what Oracle SQL function can be used to project dates in different ways? Why might this be useful/desired for reports?
  - what are some other Oracle SQL date-related functions?
- what are some of the Oracle SQL string functions? How might they be useful/desired for reports?
- what are some conversion functions that Oracle's implementation of SQL provides?
- be comfortable with expectations for reports, such as (this is not a complete list!):
  - someone can understand their meaning just by looking at them
  - nice titles
  - meaningful, pretty, consistent, well-formatted, not-truncated column headings
  - well-formatted column contents
  - specific row ordering as befits the purpose of the report
  - using concatenation and computations to make more readable
  - avoiding ugly row-wrapping

`COLUMN` command (`col`)

- what can this SQL*Plus command be used for? What is its effect? Why/When might you use it?
- be comfortable reading this command; be able to write column commands to achieve specific effects;
  - be able to use it with formatting both numeric and alphameric columns;

`BREAK` command

- what can this SQL*Plus command be used for? What is its effect? Why/When might you use it?
- be comfortable reading this command, be able to write column commands to achieve specific effects;
- be careful that you do not confuse this with the effects of the `SELECT` statement `group by` clause -- be aware of the differences between these!
- what `SELECT` statement clause NEEDS to be used with `break` to achieve reasonable results?

`COMPUTE` command

- what can this SQL*Plus command be used for? What is its effect? Why/When might you use it?
  - `compute` NEEDS to be used with what other SQL*Plus command?
- be comfortable reading this command, be able to write column commands to achieve specific effects.