CIS 315 - Final Exam Review Suggestions

last modified: 12-08-10

* 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.
    * It is strongly advised that you study posted examples and notes, and make sure you can do exercises such as those on homeworks and lab exercises;
    * (Note that if your understanding of the material is not strong enough, you may have difficulty completing the exam within its time limit.)
* You must work individually on the exam.
* You are permitted to bring into the exam a single piece of paper (8.5" by 11") on which you have handwritten whatever you wish on one or both sides. This paper must include your name, it must be handwritten by you, and it will not be returned.
    * Other than this piece of paper, the exam is closed-note, closed-book, and closed-computer.
* Note that Final Exams are *not* returned. I will retain them for at least 2 years, and you may come by my office to see them if you wish after grades are posted to the course Moodle site.
* Note that you are **not** responsible for SQL*Loader, since we were unable to actually use it.
* This will be a pencil-and-paper exam; you will be reading and writing SQL*Plus commands and SQL statements in this format, as well as answering questions about concepts we have discussed.
* Remember 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", if you've misplaced your earlier copies);
    * Studying Exam 1 and Exam 2 would also be wise.
    * Expect quite a bit of SQL,

      ... and note that you will be asked to write at least one query involving a join.

DBMS Support for Various Integrities

* There *will* be a question related to DBMS support for **entity integrity**, **domain integrity**, **referential integrity**, and **transaction integrity**;
    * For example, could you answer questions such as those in Homework 7, Problem 1-1?

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`)
        * Could give a fragment of code that includes `commit`, `rollback`, and see if you know their effects;

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 a recovery question like that given in Homework 7, Problem 1-2.

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...)

* Be able to do problems like those on Homework 8, Problem 1

* 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 of the ways of dealing with this problem?

Timestamps

* timestamps are an ALTERNATIVE to locking for concurrency control
* know BASICS of this approach, as discussed in lecture
  * (be sure you can do problems like on Homework 8, Problem 1)

* 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 tradoffs, 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

A few words on OODBMS and ORDBMS

* what are the two categories of things that make up an object?
* what is the purpose of an object-oriented database management system (OODBMS)? (or, what MIGHT it be?)
* how does an OODBMS differ from a relational DBMS (RDBMS)?
* be familiar with some potential advantages, disadvantages of an OODBMS (as compared to an RDBMS)
* what is an Object-Relational Database Management System (ORDBMS)?

* considering OODBMS, ORDBMS, and RDBMS --- which has the strongest underlying mathematical theory?
* some think that the move towards OODBMS's jeopardizes the progress made in separating ... what? Why is this separation considered beneficial (at least by some)?

Databases, society, and ethics

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

LAB-RELATED TOPICS:

* note: EXPECT to have to write at least one `update` statement and at least one `delete` statement on the Final Exam;
* ...as well as the join you've already been "promised";
* note that I could also ask questions about, for example, what order tables must be created, and what order rows must be inserted, and what the effects of referential integrity constraints are --- those are from earlier in the semester, but populating your project tables probably made such issues especially clear!

Set-theoretic operations: UNION, INTERSECT, MINUS

* why are these called set-theoretic operations? What are relations sets of?
* be able to read/write SELECT statements using these operations;
* these require that the relations involved be UNION-COMPATIBLE; what does that mean?
* what is the difference between **union** and **union all**?

The "full" SELECT statement

* study this enough to comfortable reading, writing `select` statements even when they use many of the possible clauses;
* this includes understanding the "semantics" of this, as well... (if you have both `group by` and `where` clauses, for example, in what order will they be done? etc.)
* (no, I won't ask you to write out the full `select` statement syntax diagram... 8-))

Update, Delete

* EXPECT to have to read and write these;
* be comfortable with how referential integrity enforcement can affect these;
* what is the difference between `delete` and `drop table`?
* what is the difference between `insert` and `update`?
    * be comfortable with how referential integrity, domain integrity, and entity integrity enforcement by the DBMS can all affect `insert`, `update`, and `delete`

Sequences

* what are they? how can they be created? used?
* how do you obtain the next value from a given sequence? the current value?
* be able to read, write, and use sequences

SQL Support for further restriction of domains

* be comfortable with further ways you can restrict the domains of attributes (**not null**, **default**, **check**)
    * be able to read, understand, and write statements combining all of the above features;

SQL Views

* what is a SQL view? How does a view differ from a table? What are the potential advantages of using views? How do you use a view, once it is created?
* how can a view be used together with the SQL `grant` command to possibly enhance database security?
* how do you create a view?
* how can you specify column names for a view? (know both ways, their advantages and disadvantages)
* if a view's definition involves computations or function calls, what must be done with that projected result within the view?
* make sure your views follow course style standards (there are at least two things you want to avoid...)

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 commands, 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 typical 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 (keep it from appearing)?
* 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 (nice titles, meaningful, pretty, consistent, and well-formatted column headings, well-formatted column contents, specific row ordering as befits the purpose of the report, using concatenation and computations to make more readable reports, avoiding ugly row-wrapping, etc.)

COLUMN command (col)

* what can this SQL*Plus command be used for? What is its effect? Why/When might you want to use it?

* be comfortable reading this command, be able to write column commands to achieve specific effects;
  * be able to use it with both numeric and alphameric columns;

BREAK command

* what can this SQL*Plus command be used for? What is its effect? Why/When might you want to 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 in conjunction with **break** to achieve reasonable results?
* since a SQL*Plus command is only supposed to be on one line -- how can you TELL SQL*Plus that you'd like to continue the current line? (to go to the next line but pretend it is continuing the same line)?

COMPUTE command

* what can this SQL*Plus command be used for? What is its effect? Why/When might you want to use it?

    * **compute** must be used in conjunction with what other SQL*Plus command?
* be comfortable reading this command, be able to write column commands to achieve specific effects;

Outer joins

* you should be able to read, write an **outer join**;
* how does an **outer join** differ from natural join and equi-join?
* when would you want to use an outer join?

Intro to PL/SQL and to PL/SQL triggers

* what is PL/SQL? What features does it include that SQL lacks?
* what is a trigger? for what might it be useful?
* you might be asked to write a simple trigger -- you could definitely be asked to read a simple trigger, and answer questions about it, including:
    * when it would be executed (note that there are 3-4 aspects of this you must be able to indicate: before or after, what action, on what table, and is this trigger executed for each row affected?)
    * what are the trigger's effects?
* how do you declare variables in PL/SQL? how can you use a `select` statement to set a variable? how can you write `if`-statements in PL/SQL?
* what are `:new.col_name`, `:old.col_name`, and how can they be used in a trigger?
* how can you write something to the screen in PL/SQL? What SQL*Plus setting is necessary for such writing to actually be seen?
* what must follow a trigger for it to be compiled? What SQL*Plus command can you type to see the compiler error messages if a trigger compiles with errors?