CIS 315 - Exam 2 Review Suggestions

last modified: 10-27-10

* You are responsible for material covered in lectures and labs, and especially anything that's been on a homework or lab exercise; 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 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 you are **not** responsible for union/intersect/minus, update, delete, SQL sequences, or SQL*Loader (Week 10 Lab) on Exam #2 (although that will be included in the material for the Final Exam).
* 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.
* Note that you are also responsible for knowing -- and following -- the course SQL style standards and the course ERD notation.
* primary foci for Exam #2:
    * modeling supertype/subtypes, functional dependencies, superkey/minimal key/candidate key/primary key definitions, normalization, conversion of a data model into a relational database design/schema (including converting supertype/subtypes into appropriate relations)
    * SQL and SQL*Plus-related: sub-selects/nested selects, concatenating columns, projecting literals, EXISTS/NOT EXISTS, correlated queries, order-by, group-by, having
    * with a few exceptions, the focus of most of the Exam #2 questions will be on the new material covered since Exam #1. However, since much of the new material is "built" upon previous material, that previous material will still be involved; much of this is cumulative.

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

More on Modeling

* what is a recursive relationship? how is it depicted in an ER diagram?
* what is a weak entity? how is it depicted in an ER diagram?
* what is a supertype entity class? what is a subtype entity class?
    * how is a supertype/subtype relationship depicted in an ER diagram? (remember to follow class style standards for these)

* what is meant by having a `d` in the circle in depicting supertype/subtypes entity classes? ...having an `o` in that circle? ...having a `u` in that circle?

Normalization

* what is a partial dependency?
* what is a transitive dependency?

* what is **normalization**? what is its purpose?

* what are modification anomalies?
  * ...deletion anomalies?
  * ...insertion anomalies

  * how does normalization reduce/get rid of some of these?

* be aware of the tradeoffs involved in normalization;

* remember: normalization does not **eliminate** data redundancies;
  * it tries to **reduce/eliminate UNNECESSARY** data redundancies;

  * it produces controlled data redundancy that lets us link/join database tables as needed.

* also note: when you normalize, you generally introduce referential integrity constraints.
  * (what are referential integrity constraints? how can they be handled/implemented in Oracle SQL?)

Normal Forms

* 1NF, 2NF, 3NF (first normal form, second normal form, third normal form)
  * EXPECT to have to normalize sets of relations to these normal forms;
  * could also be questions about them in general, also;

  * what does it mean if a set of relations is in 1NF? 2NF? 3NF?
    * what kinds of anomalies are reduced/eliminated when a set of relations is in each form?

    * what can no longer exist, if in each of these forms?

* BCNF, 4NF, 5NF, 6NF (Boyce-Codd Normal Form, fourth normal form, fifth normal form, sixth normal form)
  * only need to know that they exist, and how they "relate" to one another (a set of relations in 2NF is also in 1NF; a set of relations in 3NF is also in 1NF and 2NF; ... a set of relations in 6NF is also in 1NF, 2NF, 3NF, BCNF4NF, and 5NF);

  * I won't ask you to normalize sets of relations into these forms.

* what is **denormalization**? Why would we do that? Why do we not always normalize "to the max"?

Converting an ER Model into a Database Design/Schema

* recall: a **database schema/design** defines a database's structure:
    * its **tables**, (which includes each table's attributes and primary key),
    * **relationships**,
    * **domains**, and
    * **business rules**

* what is (should be) the database development process? (come up with a data model, **THEN** convert that data model into a database schema/design!)
    * and then normalize further if necessary --- this is a good "double-check" on the model/design process, making sure that your relations are in 1st, then 2nd, and then 3rd normal form;

    * BUT note that, if you have really modeled the entities in your scenario, you may not have to do much in this normalization. The "themes" that are separated in normalization should/could correspond to entities...

* remember: an entity is NOT equivalent to a table or relation!! (eventually, each entity will *result* in *one or more* corresponding tables/relations in the database schema/design that we develop from a model;)

* how do maximum cardinalities affect the eventual design?
* how do minimum cardinalities affect the eventual design?

* EXPECT to have to convert models to appropriate sets of relations;
    * what are the considerations when deciding upon primary keys for each "base" table?

    * how are multi-valued attributes handled?

    * if you know that a single-valued attribute should always have a value for some entity class, what SQL feature would you want to use in defining its (physical) domain within a create table statement?

    * be sure you know how to handle 1:1, 1:N, M:N relationships;
        * how do you change the relations involved?

        * when are additional relations necessary?

        * what are the primary keys?

        * which attributes need to be foreign keys reflecting referential integrity constraints?

    * why should you sometimes view mandatory 1:1 relationships with suspicion?
    * when does it matter which "base" table gets the foreign key in handling a 1:1 relationship? When does it not matter?

* how do you tell which is the "parent", and which is the "child", in a 1:N relationship?
  * need to know which one will get the primary key of the other placed in it as a foreign key;

* what is an intersection table? When is one needed in a design/schema? What does an intersection table include, and how is it implemented?

* how does one convert a superclass-subclass relationship into appropriate tables?

* how do you implement a foreign key, a primary key in SQL? (this had better be review! BUT it could very well be on the test.)

LAB-RELATED TOPICS:

* EXPECT IT --- you will HAVE to write at least one **join** using SQL.
  * that join might involve more than two tables, of course;

  * IF a FROM clause contains N tables --- how many join conditions should there be, to make that Cartesian product of N tables into an equi- or natural join?

* EXPECT IT --- you will be required to read AND write proper syntax SQL and SQL*Plus statements.
  * (by "read", I mean that I may give you a statement and ask you questions about it; I could also give you various table contents, and ask you what the results of running a given statement would be;)

  * and, of course, I could ask you to write a SQL statement that would perform a specified action or query;

  * (even through IN, AND, OR, and NOT, and the comparator operators <, >, =, etc., were fair game for Exam #1, they are also very likely to play a part in Exam #2 as well. There are important aspects to how they are used with **nested selects**, for example.)

Sub-selects/nested selects

* EXPECT to have to read AND write nested queries on this exam;

* EXPECT to have to write a given "question" in SQL in different ways --- one part might ask you to answer it using a join, another might ask you to answer it using a nested query, another might ask you it answer it using EXISTS or NOT EXISTS, etc.

* (if, however, I do not specify that a certain style or feature be used, then you may answer it however you like (as long as it correctly answers the question and follows class style guidelines, of course).)

IN operator

* covered in Exam #1, but it is so important/useful used with sub-selects that it will be covered on this exam, also.

* why is IN sometimes the better/more correct choice with a subquery instead of =?

Projecting a constant literal

* what does it mean to project a literal constant string, such as 'a'?

```
select 'a'
from   empl;
```

* how can this be useful, especially when combined with concatenation (||)?

EXISTS and NOT EXISTS predicates

* expect at least one question reading these, one question writing these;

* what's the "big" thing to remember? (that the subquery that is the right-hand-side of the exists/not exists predicate had better be **correlated** to the outer query!)

* what is a **correlated query**? what is a **correlation condition**?
  * how does a correlated query differ from a "plain" nested query?

  * how does a correlation condition differ from a join condition? How can you tell the difference between them? (this is important when dealing with EXISTS, NOT EXISTS)

ORDER BY clause

* EXPECT to have to read and write SELECT statements including this clause;

* what does it do?

* how do you order rows in increasing order? in descending order?

* what does it mean if there is more than one attribute in an ORDER BY clause?
  * remember to specify if desc *separately* for EACH attribute that is to be in descending order;

* remember: it CANNOT be used in a sub-select, only in an outermost select.

GROUP BY clause

* EXPECT to have to read and write SELECT statements including this clause;

* remember: GROUP BY lets you group the rows in a table based on equal values within specified columns;

* if there is no GROUP BY clause, how many rows will ALWAYS be in the result of a select statement projecting an aggregate function result? How many can there be if there *is* a GROUP BY

clause?

* what is allowed in the SELECT clause when you have a GROUP BY clause within a SELECT statement?

* what does it mean if there is more than one attribute in a GROUP BY clause?

* is grouping done before or after any row selection specified?

* be careful not to confuse ORDER BY, GROUP BY; remember that GROUP BY does not infer any ORDER that the resulting groups will be displayed --- if you want that, you SHOULD include an ORDER BY clause also.

HAVING clause

* EXPECT to have to read and write SELECT statements including this clause;

* remember: a HAVING clause must be used in conjunction with a GROUP BY clause;

* it lets you limit which GROUPS you'll see in the result;

* so, HAVING is to groups what WHERE is to rows, kind of;
    * be careful to understand the difference between WHERE and HAVING.