CS 325 - Exam 2 Review Suggestions - Fall 2024

last modified: 2024-11-12

- Based on suggestions from Prof. Deb Pires from UCLA: Because of the research-supported learning potential when students study together and explain concepts to one another, you will receive (a maximum) ***5 POINTS BONUS*** on the Exam 2 if you do the following:
  - set up and/or attend an exam study session with your group (but it can involve as many other CS 325 students as you would like!)
  - for **YOU** to receive the bonus, **YOU** submit a word (.docx) file contain the following information on canvas, sent on the **same day** as the study session (which needs to take place **before you take Exam 2**) in which you:
    - include the names of the members of your group who attended the exam study session.
    - **briefly DESCRIBE what** you **covered** at the exam study session.
    - (it needs to mention the COURSE ***TOPIC*** discussed).
    - INCLUDE a **picture** of everyone involved in the study group.
    - (**EACH** person who wants the 5 point bonus **must submit such a word (.docx) file on canvas**)
  - Please let me know if you have any questions about this, and I hope it encourages you to study together for Exam 2.
- 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, and you are to work **individually**.
- 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.
- You are responsible for material covered in class sessions, lab exercises, and homeworks; but, here's a quick overview of especially important material.
  - (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 responsible for the material covered through the end of Week 12 (Friday, November 15th).
- Your studying should include careful study of posted readings and slides 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 (more posted under the “references” section on Canvas course homepage).
- With a few exceptions, the **focus** of **most** of the Exam 2 questions will be on material covered near the end of the Exam 1 content period and 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.
  - Also, note that you **will** be asked to write at least one query involving a join.

Normalization

- 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(although reducing modification anomalies is a more important goal, I think);
  - 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

- Note: because of their importance in understanding normalization and normal forms, the definitions for **functional dependency**, **superkey**, **minimal key/candidate key**, and **primary key** may also be the focus of some Exam 2 questions.
- What is a **partial dependency**?
  - Given relation structures and additional function dependencies,

you should be able to determine if a functional dependency is a partial dependency;

you should be able to say if that set of relations has any partial dependencies.

- What is a **transitive dependency**?
  - Given relation structures and additional function dependencies,

you should be able to determine if a functional dependency is a transitive dependency;

you should be able to say if that set of relations has any transitive dependencies.

- 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?
  - If a set of relations is indeed in 1NF, what can no longer exist? If it is in 2NF...? If it is in 3NF...?
- 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, BCNF, 4NF, 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 successfully modeled the entity classes in your scenario, you may not have to do much in this normalization. The "themes" that are separated in normalization should/could correspond to entity classes...
- remember: an entity class is NOT equivalent to a table or relation!! (eventually, each entity class will *result* in *one or more* corresponding tables/relations in the database schema/design that we develop from a model;)
- how should maximum cardinalities affect the eventual design?
- how can minimum cardinalities sometimes affect the eventual design?
- EXPECT to have to convert models into 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 (that is, that it should not be permitted to be `NULL`), 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:N, M:N, 1:1 relationships;
- Handling 1:N, M:N, 1:1 relationships:
  - how do you change the relations involved?
  - when are additional relations necessary?
  - what are the primary keys of the relations involved?
  - 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" relation gets the foreign key in handling a 1:1 relationship? When might it matter?
  - how do you tell which is the "parent", and which is the "child", in a 1:N relationship? Also need to know which one will get the primary key of the other placed in it as a foreign key;

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

- how do you implement a foreign key in SQL? ...a primary key in SQL? ...a multi-attribute primary key in SQL?

Converting an ER Model into a Database Design/Schema, part 2

- Handling supertype-subtype relationships, handling weak entity classes, handling association entity classes, handling recursive relationships, handling entity classes which have multiple relationships between them
  - how does one convert a supertype-subtype relationship into appropriate tables?
  - what are the slight differences when handling disjoint subtypes? ...overlapping subtypes? ...union subtypes?
  - what are the slight differences when handling weak entity classes?
  - what is an association entity class? What are the (possible) slight differences when handling these -- what are the two options you generally need to choose from? And when might you favor one of these options over the other?
  - how are recursive relationships handled?
  - how are multiple relationships between the same two entity classes handled?

DBMS Support for Various Integrities

- EXPECT a question related to DBMS support for **entity integrity**, **domain integrity**, **referential integrity**, and **transaction integrity**.

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.

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?

LAB-RELATED TOPICS:

- EXPECT IT --- you will HAVE to write at least one join using SQL.
  - 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;
- Note -- 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

- Somewhat covered in Exam 1, but it is so important/useful, and are often the foundation for other query questions
- What is meant by a sub-select/nested select? What are some of the possibilities for where these may be placed within a `select` statement?
- EXPECT to have to read AND write nested queries on this exam;
- EXPECT to have to write queries in SQL that answer a given "question" 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.

`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 for use with a subquery rather than =?

Projecting a constant literal and concatenation

* 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 this clause 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 `desc` *separately* for EACH attribute that is to be in descending order;
- remember: an `ORDER BY` clause SHOULD NOT 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 by a `WHERE` clause?
- be careful not to confuse `ORDER BY` and `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`.

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`?

`update, delete`

- EXPECT to have to read and write these, especially including nested subqueries.
- How can you delete row(s) from a table?
  - what is the difference between `delete` and `drop table`?
- How can you modify an existing row in a table?
  - what is the difference between `insert` and `update`?
- be comfortable with how referential integrity enforcement can affect these;

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 enhance database security?
- how do you create a view?
- how can you specify column names for a view? (know both ways, 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
  - (for example: don't include an `order by` clause in a view! Use the desired `order by` when querying the view, as you do for regular tables.)

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.

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.

Some useful string- and date- and time-related functions

- sysdate, to_char. to_date, next_day, add_months, months_between
- initcap, lower, upper, lpad, rpad, ltrim, rtrim, length, substr