CS 325 - Exam 1 Review Suggestions - Fall 2024

last modified: 2024-10-02

- 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 1 if you do the following:
  - set up and/or attend an exam study session with a group of AT LEAST 3 students (but it can involve as many 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 1**) 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 1.
- You are responsible for material covered in lectures and labs, and especially anything that's been on a homework or lab exercise or slides; BUT, here's a quick overview of especially important material.
  - It is strongly advised that you study the materials below, 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, and you are to work **individually**.
- This will be a pencil-and-paper exam; you will be reading and/or 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.

Introduction to DBMSs

- What is a DBMS? What is an example of a DBMS? What are some typical capabilities of a DBMS?
- What is a database? What is a relational database?
  - What are the 4 main elements of a database? (user data, metadata, indexes, & application metadata)
- What are some of the important limitations of file-processing systems?
  - What are some of the advantages of the database approach over these file-processing systems?

- What is metadata?
- What is a relation?
  - How do you represent a relation in relation structure form? ...in tabular form? ...in SQL `create table` statement form?
  - (possible question: here's a table/relation. Write a relation structure for it.)
  - What are the important "restrictions"/features of a relation?
  - (possible question: is the following a relation? Why not?)
  - What is a primary key? How do you indicate a relation's primary key in relation structure form?
- DBMS - database management system
  - What are some typical capabilities of a DBMS?
  - What is meant by DDL, DML, and DCL? What does each of these do?
  - What are some capabilities that a (high-end) DBMS might provide?
- You should know that a database design/schema defines a database's structure, and typically includes:
  - its tables,
  - relationships,
  - domains, and
  - business rules
- What are business rules?
- How do you create tables in Oracle SQL? How do you define relationships between tables?

Introduction to the relational model and relational operations

- Who developed it? When? (Codd, 1970) Why was it first resisted?
- What is a relation? (including: What are the restrictions?)
  - Be comfortable with relation/table, tuple/row/record, attribute/column/field terminology
  - Single-valued cells, no duplicate rows, order of rows/columns not important, column entries all of same "kind"/from the same domain; must have a primary key

Functional Dependencies and Key Definitions

- What is a functional dependency? What does it mean for one attribute to be functionally dependent on another?
  - Understand the -> notation;
  - What is the determinant in A -> B ?
  - Given a relation in relation structure form, what are some functional dependencies that you can assume?
  - Does a determinant have to be a primary key? Why or why not?
  - Does a primary key have to be a determinant? Why or why not?
  - if (A, B) --> C, does it logically follow that A --> C and B --> C?

- if A --> (B, C), does it logically follow that A --> B and A --> C?

- What is a superkey? ...a minimal key? ...a candidate key? ...a primary key?
  - How does one indicate a primary key in a relation structure?
  - How does one indicate a primary key in a SQL `create table` statement?
  - How many attributes may be in a primary key?

Relational Operations

- The set-theoretic relational operations include union, difference, intersection, Cartesian product -- we have discussed only Cartesian product thus far;
- The relation-theoretic relational operations include rename, selection, projection, equi-join/natural join/other joins, and division -- we have discussed only selection, projection, equi-join, and natural join so far;
- At this point, you are expected to know and understand selection, projection, equi-join, natural join, and Cartesian product;
  - Expect to have to show that you can perform some or all of these operations on example relations; (note that I may describe the desired relational operation in words OR give it in SQL form);
  - What is an equi-join? How does it differ from a natural join?
  - Know the `relation_name.attribute_name` notation;
  - You should be able to express queries as combinations of relational algebra operations.

Introduction to the Entity-Relationship Model

- What is (should be) the database development process? What is a database model? What are some of the general strategies for developing a database model?
  - Important: the idea is that you come up with a database model *before* you come up with tables! Why should you develop a data model/database model before starting to create database tables?
  - Note: there is more than one type of database model; we are focusing on the most commonly-used, the entity-relationship model
- Remember: you are responsible for the entity-relationship diagram (ERD) notation given in class and in the course handouts;
- What are the elements of the E-R model? (entities, attributes, identifying attributes, relationships, cardinalities)
  - (Note that "entity" is often used interchangeably for both entity class and entity instance; if you are unsure whether I mean entity class or entity instance in an exam question, please ask me during the exam!)
- What is an entity class? How is it depicted in an ERD? What is/are an entity class's identifying attribute(s)?
  - 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;)
- What is a relationship? How is it depicted in an ERD?

- What is an attribute? What is its domain? How is an attribute depicted in an ERD (according to our course ERD standards)?
  - Make sure it is clear to you what is *not* an attribute in an ERD as well: attribute lists in an ERD should include *no* relationship-related information. *Only* the relationship lines in the ERD show the relationships between entity classes!
- What are maximum cardinalities of a relationship?
  - What are the possible maximum cardinality values (typically)? (one and many)
  - Based on the maximum cardinalities, what are the 3 (4) "kinds" of relationships? (1:1, 1:N, N:M)
  - Given the pertinent information about a scenario, you should be able to determine which maximum cardinalities are appropriate for a relationship;
  - According to our course ERD standards, how are maximum cardinalities depicted in an ERD? You should be able to read and understand an ERD's maximum cardinalities, and you should be able to create an ERD with appropriate maximum cardinalities;
- What are minimum cardinalities of a relationship?
  - What are the possible minimum cardinality values (typically)? (0 and 1)
  - Given the pertinent information about a scenario, you should be able to determine which minimum cardinalities are appropriate for a relationship;
  - According to our course ERD standards, how are minimum cardinalities depicted in an ERD? You should be able to read and understand an ERD's minimum cardinalities, and you should be able to create an ERD with appropriate minimum cardinalities;
- 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?
- It is **very likely** that you will be asked questions about a given ERD, to see if you can read it correctly;
- It is **very likely** that you will be asked to either draw or complete an ERD given scenario information;

Basics of Oracle SQL and Oracle SQL*Plus

- You will be required to read *and/or* 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, it is **likely** that I will ask you to write a SQL statement that would perform a specified action or query;
- How do you start up SQL*Plus on nrs-projects?
- How do you create a table using SQL?
  - How do you define attributes? What are some of the common data types?
  - Be comfortable with further ways you can restrict the domains of attributes (`not null`, `default`, `check`)

- How do you define a table's primary key in SQL?
- What is a foreign key? How do you define a foreign key in SQL?
- What kind of integrity checking do you get "automatically" in Oracle when you make an attribute a foreign key? ...a primary key?

- How do you insert rows into a table? (know both variants)
  - Which version of `insert` do you need to use to make sure you get any `default` values for attributes that have them?
- How do you use a `select` statement to show the contents of a table?
- What is a SQL script? How is it created? How is it run?
  - How can you write a SQL comment?
- You should be familiar with the SQL*Plus commands we have discussed so far, especially:
  - Which command can be used to list the column definitions for a table? (`describe`)
  - Which can be used to start spooling results to a file (and which can stop such spooling)?
  - Which can be used to execute a SQL script?
  - Which can be used to output specified characters to the screen?
- How can you delete a table?
- What is the basic syntax and basic semantics of the SQL `select` statement?

Writing "pure" relational operations using a SQL `select` statement

- How can you write a (pure) relational **projection** using a SQL `select` statement?
  - What SQL keyword can mean the difference between a "pure" relational projection and a not-so-"pure" one? What is the effect of this keyword? Where must it be placed?
- How can you write a relational **selection** using a SQL `select` statement?
- How can you write a relational **Cartesian product** using a SQL `select` statement?
- How can you write a relational **equi-join** using a SQL `select` statement? a relational **natural join**?
- How can these relational operations be combined within a single SQL `select` statement?
  - How can you express a query/question as a **combination** of relational algebra operations?
  - How can you express a query/question as a SQL `select` statement?

More on the basic SQL `select` statement

- Be familiar with the `where` clause possibilities discussed so far:
  - `=   <   >   <=   >=   <>   !=`
  - `in`
  - `is null`, `is not null`
  - `and`, `or`, `not`
  - `between`
  - `date functions (e.g., TO_DATE() or EXTRACT())`
  - `like`, `%`, `_` (underscore)

- aliases
  - What is a table alias (within the `from` clause)? what is a column alias (within the `select` clause)?
  - Why is a table alias useful?
  - Why is a column alias useful?
- Computed columns
  - You should be able to read and write queries that project computed columns;
  - Handle NULL in Computations using COALESCE() or NVL()
  - Make sure that you understand: whatever computations you choose to project from a `select` statement, projecting those computations does **not** change the contents of the database!
- Aggregate functions
  - `avg`, `min`, `max`, `sum`, `count`
  - `ROUND() function`
  - Expect to have to read and/or write some of these;
  - Where can these be used within a `select` statement?
  - What effect do `null` values have with regard to these?
  - In a basic `select` statement (with just `from` and `where` clauses), either zero or how many rows will **always** be in the result of a `select` statement projecting an aggregate function?
- Sub-selects/nested selects
  - 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?
  - it is **likely** that I will ask you to read AND write (or answer questions related to) nested queries on this exam;
  - it is **likely** that I will ask you 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.
  - know the ANSI join notation, know how to join multiple tables
  - 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
  - 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 =?