CS 325 - Project Handout - Fall 2021

When studying the design and implementation of databases, it greatly helps understanding to actually experience the process being studied. So, during this semester you will be modeling, designing, and implementing a database for a scenario. (Think of it as modeling, designing, and implementing a **demonstrable prototype** of a database for that scenario.)

This project will use the Oracle DBMS (Database Management System) on campus. The project consists of **several milestones**, with pieces turned in for each.

Project Milestones

- the overall project grade -- the sum of the parts below -- makes up **20%** of the final course grade, as noted in the syllabus. (this sum is multiplied by 0.20 to compute the project portion of your semester course grade.)

Project Milestone	Milestone's Worth	Additional Information
Scenario Selection	up to 5 points	due **11:59 pm on Friday, October 15**
Project Model Draft	up to 17.5 points	due **11:59 pm on Sunday, October 24**
Project Design Draft	up to 15 points	due **11:59 pm on Friday, November 19**
Project Population	up to 7.5 points	due **11:59 pm on Sunday, December 5**
Project Presentation	up to 5 points	given **during Lab on Friday, December 10**
Project Final Milestone	up to 50 points	up to 40 pts baseline, and (see below) up to 10 points for doing more than the minimum; due **11:59 pm on Friday, December 10**

Project EARLY BONUS

To provide some encouragement for you to not leave the finishing touches of your final project milestone until the last minute, here is a project BONUS opportunity.

- IF you present your project in lab on **Friday, December 10**,

  ******* AND!! *******

- you submit ALL files for the *final* milestone before **11:59 pm** on **Thursday, December 9**,

  ...it will be considered **early**, and will receive **a 5 point BONUS** added to the overall project grade.

(Note that if you **do not** present your project in lab on December 10, you **CANNOT** receive this bonus.)

Project Grading Comments

Your final project should be robust enough to **demonstrate** as a **prototype** -- do not attempt to implement a "production quality" system! (And if you cannot implement everything you hoped to, implement as much as you can, aiming for an **interesting, demonstrable prototype**.) A project that meets all of the minimum requirements mentioned in this handout, and that has met all of the minimum requirements all down the line (throughout the milestones), and that does so well, would receive a grade of **90**.

The other **10** points I will reward based on merit -- did you come up with a particularly interesting, original proposal? is a database particularly well-designed and implemented? Is something about a project above-and-beyond the minimal requirements, or exemplary? Were useful extra features included, or was some major aspect particularly well-done? Did something about a project just stand out, or make a strong impression?

Especially for the project model, but also for the project design and population, you **should** likely be **improving** them over the course of the project. So, you usually are required to submit the **latest** versions of files from **earlier** milestones as part of **later** milestones. This is a requirement, and your grade will be affected if you do not include these.

Project minimum structural requirements

Because an important topic in this course is database modeling and design, your project is required to meet certain criteria to help ensure that it is at least somewhat interesting in model and structural senses. **Final projects that do not meet these minimum criteria will be severely penalized.**

The **database model** for your database must include:

- at least 5 **distinct**, **significant** entity classes
  - a (non-union)-superclass entity along with all of its subclass entities count as **one** combined entity toward the five-entity-class minimum, unless some of the subtypes have relationships in which only that subtype participates. Be careful, and **ask me** if you have concerns about this.
- at least 4 **distinct**, **significant** relationship classes, at least one that is **1:N** and at least one that is **M:N**
- at least one **multi-valued** attribute

And, the corresponding **database design/schema** must eventually **correctly implement** all of the above.

Project Scenario Selection (worth: up to 5 points)

One important aspect of this project is that you are not just "building a database" -- you are **modeling** a **scenario** in which a database could be useful. This distinction is important, and needs to be kept in mind.. You need a scenario with multiple categories-of-actors interacting in various ways.

There is a handout posted with this Project handout, "Ideas for CS 325 Project Scenarios". You may:

- choose one of these
- choose and modify one of these
- use one of these as a starting point, **OR**
- describe your own custom scenario in a **similar style** to how these are described.

Hopefully these will help you to select or come up with a scenario you will find interesting to model and build a prototype database for.

Project Scenario Selection milestone (up to 5 points)

Due:

by 11:59 pm on **Friday, October 15**

How to submit:

Submit the file `325scenario.pdf` to Canvas.

What to submit:

In a file `325scenario.pdf`:

- include your name and the last-modified date
- give your scenario selection as follows:
  - Give a name for your scenario. (e.g., Loving Care Pet Boarding, Rockin' It Rock Climbers, Humboldt Bowling Club, Arcata Art Emporium, etc.)
  - Include at least one paragraph **describing** your **scenario** (in the the style of those included in the "Ideas for CS 325 Project Scenarios" handout).
    - It is fine for you to simply use one of the paragraphs from the "Ideas for CS 325 Project Scenarios" handout (although include:

      (from the "Ideas for CS 325 Project Scenarios" handout)

      ...if so, since attributing sources is good practice).
    - It is fine for you to simply modify/customize one or more of the paragraphs from the "Ideas for CS 325 Project Scenarios" handout (although include:

      (adapted from the "Ideas for CS 325 Project Scenarios" handout)

...if so, since attributing sources is good practice).

- And it is fine for you to write your description paragraph(s) if you would like, also. But be sure to include at least the level of description given in the provided examples.

- NOW consider -- what are some **QUESTIONS** people might ask within your chosen scenario? (For example, for an old-fashioned video rental store scenario, some example questions might be:
  - Does customer X currently have any unreturned rentals?
  - In what formats is movie Y available for rental?
  - What are the titles of Comedy movies for which we have copies?

  Come up with **at least 5** such example questions for your chosen scenario, and include a **numbered list** of these questions.

- ALSO come up with **at least 3 initial business rules** for your scenario. (For example, for an old-fashioned video rental store scenario, some example business rules might be:
  - While a store manager may authorize a different rental return date than the usual, all rentals must have a return date specified at the time of the rental.
  - A customer with an overdue rental may not rent any more videos until taking care of that overdue rental.
  - A customer may indicate a favorite movies category, but they may only indicate one such favorite category.

  Come up with at least 3 initial reasonable business rules for your chosen scenario, and include **a numbered list** of these initial business rules. (Note that you will be **expanding** this in a **separate** file as the project continues.)

Checklist: files to submit for this milestone:

△ `325scenario.pdf` (make sure it has your scenario name AND description, **AND** your list of at-least-5-questions, **AND** your list of at-least-3-business rules)

Project Model Draft Milestone (worth: up to 17.5 points

NOTE: you should **not** be mentioning tables or relations at this point **at all**, nor should you be mentioning primary or foreign keys. Those are appropriate in the **next** milestone, the Project **Design** Draft Milestone!

Project Model Draft milestone (up to 17.5 points)

Due:

by **11:59 pm** on **Sunday, October 24**

How to submit:

Submit the files noted below to Canvas.

What to submit:

(If desired, you may submit a `.zip` file containing the following files instead of submitting the individual files.)

Submit the latest, current version of your `325scenario.pdf`.

ALSO: Create a PDF file named `325model.pdf` containing the following:

- your name
- CS 325 - Fall 2021
- the date the model was last modified
- your current-as-of-this-point Entity-Relationship Diagram (ERD) for your project, depicting your database model
- make sure this includes the lists of attributes for each entity class, in which multi-valued attributes are indicated, as discussed in lecture
- remember, entity classes are NOT tables or relations. Do NOT refer to them as such, nor include information about tables or relations in the model. The project is NOT at the table/relation stage yet!
  - Likewise, since these are not tables or relations, there are no primary keys or foreign keys yet, either. These are NOT part of the E-R model stage.
- Make sure that your completed model meets the model aspects of the **Project minimum structural requirements** given below.

ALSO: Remember those initial business rules from the Project Scenario Selection milestone?

NOW make a **new, separate file**, named `325biz-rules.pdf`, with the latest/current version of your business rules, that includes:

- your name
- CS 325 - Fall 2021
- the date the business rules were last modified

- a **bulleted** list of your business rules thus far
- (Hint: you SHOULD find yourself adding to your initial set of business rules as part of the modeling process!)

Project minimum structural requirements

Because an important topic in this course is database modeling and design, your project is required to meet certain criteria to help ensure that it is at least somewhat interesting in model and structural senses. **Final projects that do not meet these minimum criteria will be severely penalized.**

The **database model** for your database must include:

- at least 5 **distinct**, **significant** entity classes
  - a (non-union)-superclass entity along with all of its subclass entities count as **one** combined entity toward the five-entity-class minimum, unless some of the subtypes have relationships in which only that subtype participates. Be careful, and **ask me** if you have concerns about this.
- at least 4 **distinct**, **significant** relationship classes, at least one that is **1:N** and at least one that is **M:N**
- at least one **multi-valued** attribute

And, the corresponding database design/schema must eventually correctly implement all of the above.

Checklist: files to submit for this milestone:

△ `325model.pdf`

△ `325biz-rules.pdf`

△ `325scenario.pdf` (latest/current version)

Project Design Draft Milestone (worth: up to 15 points)

Due:

by **11:59 pm** on **Friday, November 19**

How to submit:

Submit the files noted below to Canvas.

What to submit:

(If desired, you may submit a `.zip` file containing the following files instead of submitting the individual files.)

Submit the latest, current version of your `325scenario.pdf`, `325model.pdf`, and `325biz-rules.pdf`.

- **Make sure** that the model you submit corresponds to your submitted design/schema! Your design/schema will be graded on the basis of the model version that you submit along with it.
  - This does **NOT** mean to add table-related aspects to your model! It means, if you change any maximum or minimum cardinalities, for example, or add or remove any entity classes, etc., based either on comments from me or from insights you get during the design process, that you modify the model to reflect them.

ALSO: Create a text file `325design-rs.txt` containing the following:

- your name
- CS 325 - Fall 2021
- the date the design was last modified
- your current-as-of-this-point *partial* database schema/design, in which the structure of your database's tables are expressed as neatly-formatted **relation structures**,

  with SQL-style `references` clauses used to indicated foreign keys
  - (why do I call this *partial*? Because, for example, it does not include any domain information for the database attributes)
  - the purpose of this partial version of your design is for convenient future reference, for example as you are working on example queries and reports for a later milestone

ALSO: Create a **SQL script** named `325design.sql` containing the following:

- your name (in a comment)
- CS 325 - Fall 2021 (in a comment)
- the date the design was last modified (in a comment)
- your current-as-of-this-point less-partial database schema/design, in which the structure of your database's tables are NOW expressed as **nicely-formatted SQL** `create table` **statements**.
  - precede each SQL `create table` statement with a **neat comment** describing the table's **purpose**, explaining any attribute whose meaning is not immediately clear from its name, and elaborating on the

domain of any attribute whose logical domain needs more description than is apparent from its physical domain

– (for convenience later during population, go ahead and also precede each SQL `create table` statement with a corresponding `drop table` statement as well)
– remember: primary keys must be explicitly specified for each table, and all foreign keys necessary should be explicitly specified as well
– (thus, these SQL `create table` statements give us 3 of the 4 components of a database design/schema: the tables' structure, their relationships, and at least some indication of each attribute's physical domain, if not its logical domain.

  And you are already submitting your latest set of business rules along with this, so this milestone's pieces do include a complete database design/schema.)

Additional notes/reminders:

- **PLEASE NOTE**: `insert` statements **DO NOT** belong with the database design. Please save them for the population milestone, and do **NOT** include them here.

Checklist: files to submit for this milestone:

△ `325design-rs.txt`
△ `325design.sql`

△ `325model.pdf` (latest/current version)
△ `325biz-rules.pdf` (latest/current version)
△ `325scenario.pdf` (latest/current version)

Project Population Milestone (worth: up to 7.5 points)

Due:

by **11:59 pm** on **Sunday, December 5**

How to submit:

Submit the files noted below to Canvas.

What to submit:

(If desired, you may submit a `.zip` file containing the following files instead of submitting the individual files.)

Submit the latest, current version of your `325scenario.pdf`, `325model.pdf`, `325biz-rules.pdf`, `325design-rs.txt`, and `325design.sql`.

– (make sure that your submitted model corresponds with your submitted database design/schema!)

ALSO: Create and submit a `325populate.sql` file (that is, a SQL script) that populates your tables with some example data (which can be fictitious!)

*******THIS MUST BE a *SEPARATE* FILE FROM YOUR** `325design.sql` file!!!!!**********

- include your name in a comment in the SQL script
- include `CS 325 - Fall 2021` in a comment in the SQL script
- include the date that the SQL script was last modified in a comment within the SQL script
- this script should **START** with appropriate `delete` statements, **deleting** any current contents in these tables
  - (this is to make **re-**populating the database more straightforward during testing -- you should be able to just re-run `325populate.sql` to restore the database contents to this initial example set of rows)
  - remember that you need to delete rows from child tables FIRST, **before** deleting rows from parent tables that might be referenced by those child tables
  - note that these will show that 0 rows have been deleted from each table the **first** time this population script is run, **and** when this population script happens to be run **right after** `325design.sql` has been run
- **follow** these `delete` statements with appropriate `insert` statements, inserting rows into all of your tables
  - precede **each** table's set of `insert` statements with a `prompt` command to note **which** table is about to have rows inserted -- then, if you see an error message after this, you will know **which** table's insert(s) had problems
  - remember that you need to insert rows into parent tables FIRST, **before** inserting rows into child tables that reference those parent tables
- how much data should you include? The goal is to include enough to make your project at least a **non-trivial, demonstrable prototype**. Trying to give some firmer guidelines:
  - have at least 10 rows per table,

- ...with additional rows as needed to make your database a reasonable prototype. (You'll find that some tables need **additional** rows so that others can have sufficient contents.)
- Again, fictitious data is fine, but make it "look" realistic -- don't use names like `'fg^s&A#'`, for example.
- Don't use "real" data that is sensitive! (**No** "real" social security numbers, credit card numbers, etc.!)

ALSO: create and submit a SQL script `325show-contents.sql` that contains:

- your name (in a comment)
- CS 325 - Fall 2021 (in a comment)
- the date the contents script was last modified (in a comment)
- `spool` commands to output its results to a file `325result-contents.txt`
- one or more `select *` statements for each table
  - precede each `select` statement with a `prompt` command to display the name of the table whose rows are being displayed
  - make sure that the rows are readable, and do not "wrap-around" in the `325result-contents.txt` file. There are several ways to achieve this; for example:
  - you can show a very-wide table's contents using several `select` statements, each projecting the primary key and a few of its other attributes;
  - you can use the `truncate` feature of the SQL*Plus `column` command (ask me if you need help with this)
  - you can increase the number of characters printed per line before wrapping by changing `linesize`

Also submit the file `325result-contents.txt` resulting from running `325show-contents.sql` .

Checklist: files to submit for this milestone:

△ `325populate.sql`

△ `325show-contents.sql`

△ `325result-contents.txt`

△ `325design-rs.txt` (latest/current version)

△ `325design.sql` (latest/current version)

△ `325model.pdf` (latest/current version)

△ `325biz-rules.pdf` (latest/current version)

△ `325scenario.pdf` (latest/current version)

Project Presentation Milestone (worth: up to 5 points)

Due:

by beginning of YOUR lab session on **Friday, December 10**

How to submit:

Present the following during lab on **Friday, December 10**.

What to present:

To provide you with some practice preparing and giving a short live-program-demonstration related to your database project, and to give you encouragement as you polish the final project milestone's example queries and example reports, you are to prepare and present a short presentation meeting the following requirements.

Note that this presentation will ALSO be the Week 15 Lab Exercise.

Remember, also, that doing this presentation is required as part of the early-completion 5-point bonus.

Your presentation must meet the following requirements:

- You should plan for it to be **no more than 3 minutes** in length.
- In your presentation:
  - start by **briefly** describing your scenario
  - then run, **live**, ONE of your example **queries** or **reports** that you believe would be helpful to people within that scenario, including the following parts:
    - first **DISPLAY the code for the query/report already typed within a SQL script**, and then **RUN that SQL script live** to show its result

      (for example,

      `SQL> host more` *your-script*`.sql`

      [briefly describe displayed query/report]

      `SQL> start` *your-script*`.sql`

      [briefly describe displayed query/report results])
    - also **say WHY you think this query/report would be USEFUL** to people within that scenario
- Just to make sure this is clear: this must be a **live-program-demonstration** -- you are expected to **actually run** your selected example script as *part* of your demonstration, **projecting** your script's output.

Checklist: files to submit for this milestone:

△ (none -- this milestone is completed during the Week 15 Lab on Friday, December 10)

Project Final Milestone (worth: up to 40 points baseline + up to 10 points for work above and beyond = up to 50 points)

Due:

by **11:59 pm** on **Friday, December 10**

(**But remember**: the **early completion 5 POINT BONUS** applies to projects that are:

- **completely** submitted by **11:59 pm** on **Thursday, December 9**
- ***** AND *** presented in lab** on **Friday, December 10**)

How to submit:

Submit the files noted below to Canvas.

What to submit:

(If desired, you may submit a `.zip` file containing the following files instead of submitting the individual files.)

Submit the latest, current version of your `325scenario.pdf`, `325model.pdf`, `325biz-rules.pdf`, `325design-rs.txt`, `325design.sql`, `325populate.sql`, `325show-contents.sql`, and `325result-contents.txt`

- (make sure that your submitted model corresponds with your submitted database design/schema! There could be a SUBSTANTIAL PENALTY if these do not correspond/"go together"!)
- ALSO NOTE that there will be SUBSTANTIAL PENALTIES if the **Project minimum structural requirements** given in the Project Model milestone are not met in the submitted model and design

Create and submit a `325queries.sql` file that contains a set of **example queries** using your prototype database, including:

- your name (in a comment)
- CS 325 - Fall 2021 (in a comment)
- the date the example queries script was last modified (in a comment)
- `spool` commands to output its results to a file `325query-results.txt`
- at least **eight** substantially-different and structurally-different representative queries, including at least some "innovative" ones, that meet the following requirements:
  - each query must be **potentially meaningful/useful** to users of your database
  - precede each query statement with a `prompt` command explaining the query's purpose and **numbering** the query
  - include at least one **join**
  - include at least one **appropriately-nested** query
  - include at least one appropriate use of an **aggregate function** (such as `count, min, max, avg,`

  `sum,` etc.)
  - include at least one appropriate use of a `group by` clause
  - include at least one **compound** `where` condition with at least a couple of sub-conditions **other** than join conditions
  - make sure that enough example data is included so that these queries' results are a meaningful demonstration -- you may need to add additional rows for this to be the case.
- These queries could be related to the questions given in your database proposal, but they do **not** have to be, and you are **not** limited to answering those original questions!
- The **most important** criterion is that they show **clearly** how the database could be **useful** to end-users within the scenario.

Also submit the file `325query-results.txt` resulting from running `325queries.sql`

Create and submit one or more files `325report1.sql`, `325report2.sql`, ... that create **example reports** from your prototype database, including:

- your name (in a comment)
- CS 325 - Fall 2021 (in a comment)
- the date that example reports script was last modified (in a comment)
- `spool` commands to output that script's results to a file `325report1-results.txt` (or `325report2-results.txt`, etc.)
- at least three substantially-different and structurally-different representative reports, including at least some "innovative" ones, that meet the following requirements:
  - each report must be **potentially meaningful/useful** to users of your database
  - the code for each report must be preceded by a neat **comment** explaining its purpose
  - each report should be **well-designed** and **well-laid-out**. Human-readability is an important characteristic of a well-designed report.
  - "nice"/"pretty" column formatting -- especially for numeric columns -- is expected
  - "nice"/"pretty" heading formatting is expected
  - concatenation should be used to make reports more pleasant to read -- (for example, instead of having separate first and last name columns, a report should display a single column with the last name concatenated with the first name, or vice versa, depending on the report's purpose)
  - rows should be explicitly ordered in a meaningful way within reports (and this includes appropriate secondary ordering as applicable, etc.)
  - at least a top title is expected (for **each** report)
  - include at least one appropriate `break` command whose results are obvious
  - include at least one appropriate `compute` command whose results are obvious
  - at least two of your reports should be based on queries involving **more than one** table (**or** on a view created from more than one table)
  - at least one report should contain at least one column whose contents are appropriate, meaningful numeric

data with a well-formatted **fractional** part

- `break` should be used to avoid "ugly" repetition in consecutive report rows
- `skip` should be used judiciously to separate results generated using breaks and computes. (Avoid too much skipping of lines, however -- e.g., avoid having a blank line after every row in a report.)
- make sure that enough example data is included so that these reports' results are a meaningful demonstration -- you may need to add additional rows for this to be the case.

- These reports could be related to the questions given in your database proposal, but they do not have to be, and you are not limited to reports that answer those original questions.
- The **most important** criterion is that they show **clearly** how the database could be **useful** to end-users within the scenario.

Also submit the files `325report1-results.txt`, `325report2-results.txt`, ... resulting from running `325report1.sql`, `325report2.sql`, ...

Create and submit a file `325discussion.pdf` that contains:

- your name
- CS 325 - Fall 2021
- the date this discussion was last modified
- a discussion on "**How can this implemented database now be used?**" meeting the following requirements:
  - it should contain **at least 300 words** (I will measure this using the UNIX `wc` command.)
  - discuss how your particular database, now implemented, can be **used** within your proposal's scenario
  - be specific!
  - NOTE that this **could** include discussions of possible **applications** users in the scenario might like to build ON TOP OF/USING this database

Create and submit a file `325readme.pdf` that contains:

- your name
- CS 325 - Fall 2021
- the date this readme file was last modified
- a list containing names and descriptions of all "code" files (SQL, SQL*Plus, PL/SQL, etc.) used in the final version of your submitted project
  - that is, follow **each** file name with a brief description of that file's contents
- a **separate** list of instructions for how to set up and use the database (which scripts are used to create and initially populate the database? etc.)
  - (these two lists are quite common contents in a README file -- they constitute very simple external documentation for "installing" your project)
  - Please note that I may or may not actually run your database -- however, I had better be able to use the instructions above to recreate your database from the files you submit, if necessary.

Remember: if previous milestone(s) required that certain changes or additions be made to the proposal, to the previous set of business rules, to the model, to the design/schema, and/or to the population, then those changes should be reflected in newly-submitted versions for this milestone, or **this** milestone's grade may be affected

FINALLY, you may, if you wish, also create and submit a file `325misc.pdf`, accompanied by additional files as you desire, that contains:

- your name
- CS 325 - Fall 2021
- the date this Miscellaneous file was last modified
- a description/discussion/list of anything else that you wish for me to consider while grading your project -- for example,
  - mention of features you have used in addition to those required (such as sequences, or views, or time/date functions, for example) and in which files those are used
  - lists of additional files you are submitting demonstrating additional features or functionality, such as PL/SQL triggers, files demonstrating that the triggers work, code for forms you have designed or implemented, etc.
  - If you have made some **extra** effort, and you want to **make sure** that I do not overlook it while grading your project, **then mention it/show it off here!**

The final project, if well-done, will be a package you can proudly show off as part of your "portfolio" for interviews, or possibly even use as a reference later for future databases that you design.

Checklist: files to submit for this milestone:

△ `325queries.sql`

△ `325query-results.txt`

△ `325report1.sql`, `325report2.sql`, `325report3.sql`, ...

△ `325report1-results.txt`, `325report2-results.txt`, `325report3-results.txt`,...

△ `325discussion.pdf`

△ `325readme.pdf`

△ `325misc.pdf` (optional, but a good idea to include!)

△ `325populate.sql` (latest/current version)

△ `325show-contents.sql` (latest/current version)

△ `325result-contents.txt` (latest/current version)

△ `325design-rs.txt` (latest/current version)

△ `325design.sql` (latest/current version)

△ `325model.pdf` (latest/current version)

△ `325biz-rules.pdf` (latest/current version)

△ `325scenario.pdf` (latest/current version)