# CS 325 - Homework 2

## Deadline

**11:59 pm on Friday, September 16, 2022**.

**Question 1:** The Reading packet discusses some classic limitations of file-processing systems, especially as the quantity of data increases.

Consider a set of files where one file contains a list of classes (class names, class times), another file contains a list of students (student id numbers, student last names, student first names, and student emails), and a third file contains a list of students and the classes they are enrolled in (student id numbers, each followed by the class names of the classes that student is in).

You now want to know the emails of students who are in classes at 2:00 pm on Tuesdays.

What limitation below is the best "fit" for describing why this question is difficult to answer when the data is stored in such files?

1 unnecessary data duplication

2 separated and isolated data

3 more potential for a central point of failure

4 decreased application program dependence

**Question 2:** Which of the following is the *best* example of *unnecessary* data duplication?

1 one file contains a list of classes (class names, class times), another file contains a list of students and the classes they are enrolled in (student id numbers, each followed by the class names of the classes that student is in)

2 one file contains a list of students (student id numbers, student last names, student first names, student emails), and another file lists advisor information (the last name of an advisor, the first name of an advisor, each followed by the student id numbers of all of the students they advise)

3 one file contains a list of classes (class names, each followed by that class' times and class location), and another file contains a list of students (student id numbers, student last names, student first names, student emails, each followed by the class names, class' times, and class location for all of the classes they are taking)

4 one file contains a list of classes (class names, each followed by that class' times and class location), another file contains a list of students (student id numbers, student last names, student first names, and student emails), and a third file contains a list of students and the classes they are enrolled in (student id numbers, each followed by the class names of the classes that student is in).

**Question 3:** For a database implemented using a DBMS, do database applications interact directly with the actual data files of database data? Select the single most-accurate answer below.

1 YES, database applications on such a database interact directly with the actual data files of that database's data.

2 YES, which is why database applications on such a database must also be implemented using the same DBMS.

3 NO, database applications on such a database interact with the DBMS, which essentially hides the actual data files and their format from database application programs.

4 NO, database applications on such a database must interact using the system calls of the operating system hosting that DBMS.

**Question 4:** Which of the following is an (at least potential) **advantage** of DBMSs described in this reading packet?

1 For a given collection of data, storing that data in files will always take more room than storing them in a database managed by a DBMS.

2 For a given collection of data, performance will be better for an application dealing with a DBMS than one dealing with files.

3 It is easy, inexpensive, and fast to convert data in files into data within a database.

4 It can be easier to represent information from the user's world from their perspective using a well-designed database managed using a DBMS.

**Question 5:** Which of the following was a concern that people had at first to using the relational model?

1 They thought it was too simple to be useful for real scenario.

2 They thought it would not support using the data in flexible ways.

3 They thought it would never be practical, that it would always be too slow.

4 They thought it increased the incidence of unnecessarily-duplicated data, even with a well-designed database.

**Question 6:** What is metadata?

1 data about data -- it is what makes a database self-describing, it is a description of a database's data

2 data which includes elements more than 1 MB in size

3 data that has been stored in a database, rather than in a file

4 data that has been inserted into a database using SQL

**Question 7:** Which of the following is a DIFFERENCE between tabular form and relation structure form for depicting a relation?

1. you can clearly see what the relation's primary key is in tabular form, but you cannot in relation structure form
2. you cannot see the specific domain for each column in tabular form, but you can in relation structure form
3. you can see the basic relation structure in tabular form, but you cannot in relation structure form
4. you can see the table's contents in tabular form, but you cannot in relation structure form

**Question 8:** Which of the answers below is following the relation structure form given in the reading packet for a relation `widget` with two columns, `widget_id` and `widget_name`, whose primary key is `widget_id`?

1. `create table widget(widget_id integer, widget_name varchar2(20));`
2. `widget(WIDGET_ID, widget_name)`
3. `widget(widget_id integer, widget_name varchar2(20))`
4. `create table widget(WIDGET_ID, widget_name);`

**Question 9:** Which of the following is a DIFFERENCE between create-table form and relation structure form for depicting a relation?

1. you cannot clearly see what the relation's primary key is in create-table form, but you can in relation structure form
2. you can see the specific (physical) domain for each column in create-table form, but you cannot in relation structure form
3. you can see the basic relation structure in create-table form, but you cannot in relation structure form
4. you can see the table's contents in create-table form, but you cannot in relation structure form

## Notes:

- You are required to use the HSU Oracle `student` database for Questions 10 to 13.
- **SQL Reading Packet 1**, on the course Canvas site, is a useful reference for this homework.
- Feel free to add additional `prompt` commands to your SQL scripts as desired to enhance the readability of the resulting output.

## Setup for Questions 10-13 onward

Use `ssh` to connect to `nrs-projects.humboldt.edu`, and create a directory named `325hw2` on nrs-projects:

```
mkdir 325hw2
```

...and change this directory's permissions so that only you can read it:

```
chmod 700 325hw2
```

...and change your current directory to that directory (go to that new directory) to do this homework:

```
cd 325hw2
```

Put all of your files for this homework in this directory. (And it is from this directory that you should type `~ma548/325submit` to submit your files each time you would like to submit your work-so-far.)

## Question 10

Use `nano` (or `vi` or `emacs`) to create a file named `325hw2-create.sql`:

```
nano 325hw2-create.sql
```

While within `nano` (or whatever), type in the following within SQL comment(s):

- your name
- `CS 325 Homework 2 - create tables`
- the date this file was last modified

Now, within your file `325hw2-create.sql`:

Consider the following tables/relations, each written in **tabular form** and including additional requirements/constraints.

### *the Client table/relation:*

- NOTE: client ids are always intended to be **exactly** 4 characters long

| Cli_id | Cli_lname | Cli_fname | Cli_phone |
|---|---|---|---|
| 000A | Alpha | Ann | 000-0001 |
| 111B | Beta | Bob | 111-1112 |
| 222B | Beta | Ann | 222-2223 |
| 333C | Carlos | David | 333-3334 |
| 444D | Delta | Edie | 111-1112 |

### *the Video table/relation:*

- NOTE: video ids are always intended to be **exactly** 6 characters long

| Vid_id | Vid_format | Vid_purchase_date | Vid_rental_price | Vid_length |
|---|---|---|---|---|
| 00000D | DVD | 10-JAN-2020 | 1.99 | 73 |
| 11111H | HD-DVD | 20-FEB-2021 | 4.99 | 91 |
| 22222B | BluRay | 30-MAR-2019 | 1.99 | 105 |
| 33333H | HD-DVD | 20-FEB-2021 | 3.99 | 69 |
| 44444B | BluRay | 04-APR-2017 | 0.99 | 91 |

### *the Rental table/relation:*

- NOTE: In this **bizarre** scenario, a client is only allowed to rent a particular video **one time**, **ever**. (That's an example of a **business rule**, by the way!)

-(That is, a client can rent different videos over time, but they can only rent a particular video at most once. And a video can be rented by more than one client over time, but it can only be rented by a particular client at most once.)

| Cli_id | Vid_id |
|---|---|
| 111B | 11111H |
| 222B | 00000D |
| 222B | 22222B |
| 333C | 22222B |
| 333C | 00000D |
| 333C | 11111H |
| 000A | 44444B |

Keeping the above in mind:

- write at least one `prompt` command saying that what follows is for this `question`, and that you are creating tables

-(even though there are no `spool` commands in *this* SQL script, the `prompt` commands will still make your within-`sqlplus` output more readable)

- write SQL `drop table` statements and `create table` statements for `Client`, `Video`, and `Rental`, being sure to:

  -include `cascade constraints` in these `drop table` statements

  -include all of the columns shown, and satisfy all of the requirements given

  -give each column a reasonable, appropriate type, following the class style standards and the stated requirements given above

  -remember which type is more appropriate when a character string column's contents are of **different** lengths, and which is more appropriate when a character string column's contents are

**always** the same length

-note that you may **NOT** use types `char` or `varchar2` for the video purchase date nor the video rental price -- choose more appropriate types instead!

-explicitly set an appropriate **primary key** for each table (note that you may **NOT** add additional columns to any of these tables)

-be sure to declare **foreign keys** as appropriate

Make sure that, when run in `sqlplus`:

```
start 325hw2-create.sql
```

successfully drops and creates these three tables (although remember that the `drop table` commands will fail until you actually manage to create these tables for the first time, since until then there is nothing to drop.)

Submit your script `325hw2-create.sql`.

## Question 11

Use `nano` (or `vi` or `emacs`) to create a file named `325hw2-pop.sql`:

```
nano 325hw2-pop.sql
```

While within `nano` (or whatever), type in the following within SQL comment(s):

- your name
- `CS 325 Homework 2 - populate tables`
- the date this file was last modified

Now -- for this homework, we are **separating** table creation and initial table population into separate SQL scripts, because that can be a useful practice.

BUT, for this to work smoothly during re-population (in the case of debugging, or future modifications, etc.), the population script needs to delete the tables' current rows when it is re-run, to avoid trying to insert the same rows more than once. Which would be fine, except we have not yet discussed the SQL `delete` command!

SO: you are being provided with these commands, this time!

After your opening comment(s) in `325hw2-pop.sql`, add the following comment and `delete` commands:

```
/*========
    in case this is re-run (to get a "fresh" set of initial
    contents), delete any current contents
========*/

delete from rental;
delete from video;
delete from client;
```

Follow the above with at least one `prompt` command noting that what follows is for `Question 11`, and that you are inserting rows.

Then, add SQL `insert` statements to insert the rows shown in Question 10's tables into your versions of these tables.

Make sure that, when run in `sqlplus`:

```
start 325hw2-pop.sql
```

successfully adds the rows shown in Question 10's tables to these three tables.

This script is still not yet complete -- you have more, still, to add to it.

## Question 12

In `325hw2-pop.sql`, now add at least one `prompt` command noting that what follows is for `Question 12`, and that you are inserting YOUR additional rows.

Then, add SQL `insert` statements to insert:

- one additional row of your own choice into the `Client` table
- one additional row of your own choice into the `Video` table
- one additional row having your new client renting your new video into the `Rental` table

Make sure that, when run in `sqlplus`:

```
start 325hw2-pop.sql
```

successfully adds the rows shown in Question 10's tables AND your additional rows to these three tables.

Submit your script `325hw2-pop.sql`.

## Question 13

We didn't use `spool` in `325hw2-create.sql` or `325hw2-pop.sql` to again underscore the point that once tables (and their rows) are created within a database, they **persist**, staying around until they are dropped -- and other SQL scripts can make use of them.

Use `nano` (or `vi` or `emacs`) to create a file named `325hw2-use.sql`:

```
nano 325hw2-use.sql
```

While within `nano` (or whatever), type in the following within SQL comment(s):

- your name
- `CS 325 Homework 2 - use tables`
- the date this file was last modified

Now, within your file `325hw2-use.sql`:

- start spooling to a file `325hw2-out.txt`

-also put a `spool off` command at the BOTTOM/END of this file. Type your answers to the

remainder of the parts below BEFORE this `spool off` command!

- put a `prompt` command that includes YOUR name
- put a `prompt` command that indicates that what follows are the current state of the `client` table,

  -and then put a `describe` command showing `client`'s structure,

  -and then put a `select` statement showing `client`'s contents
- put a `prompt` command that indicates that what follows are the current state of the `video` table,

  -and then put a `describe` command showing `video`'s structure,

  -and then put a `select` statement showing `video`'s contents
- put a `prompt` command that indicates that what follows are the current state of the `rental` table,

  -and then put a `describe` command showing `rental`'s structure,

  -and then put a `select` statement showing `rental`'s contents
- double-check that your `spool off` command is AFTER these `prompt`, `describe`, and `select` statements!

(Do not worry about "ugliness" like chopped-off column headings, or too-long rows that wrap to the next line, repeated column headings, or how values are formatted - we'll discuss how to change how these display later.)

Make sure that, when run in `sqlplus`:

```
start 325hw2-use.sql
```

successfully displays the structure and contents of these three tables.

Also make sure that, when run at the `nrs-projects` prompt (that is, at the UNIX level, NOT in `sqlplus`!):

```
more 325hw2-out.txt
```

has as its contents the output from running the script `325hw2-use.sql` that you just saw within `sqlplus`.

Submit your files `325hw2-use.sql` and `325hw2-out.txt`.