School of Computer Science, McGill University

COMP-421 Database Systems, Winter 2024

Project 2: Creating and Working with Your Database

Due Date February 23, 12:00 NOON

In this deliverable you are going to refine your schema, create your database using DB2 and write and run some SQL queries. **We have created a UNIX and a DB2 group account for each registered group. Information on group names and passwords can be found on myCourses.** ~~It is important to reset the default password for your group's unix account.~~ ~~**You will lose 10 points if you do not**~~ ! ~~Make sure all members in your group are aware of the password.~~

Please note, that for this and the next project deliverable you might frequently need to work with the DB2 online information (link from myCourses) in order to figure out how things work in the database system. It is an essential part of the project to learn how to find the needed information in the online help. To make life a bit easier at the beginning, some extra links/documents are given on myCourses. They describe the most essential things.

In this assignment, while you can use either the `db2` command line client or the IntelliJ IDEA IDE when you start development (some are shortly introduced in myCourses and/or demonstrated during lectures.), you must use the `db2` command line utility (through the template scripts given to you) to run your finalized project files and submit the output. A template, `project2.tar.gz` is included with this assignment to help with this.

WARNING !! **Do not insert more than 500 records per table. Violators will be penalized.** But make sure that you insert sufficient records into the tables such that your queries produce at least a couple of records for the queries you create.

1 Assignment

Some of the solutions to the below questions have to be provided in a document **project2.pdf** whereas others will be independent files of their own (described next to each question).

You are given a template, `project2.tar.gz` that you **must** use to execute and turn in these extra submissions.

Where required to produce a screenshot of a command execution, it must be executed from the DB2 command line from an ssh terminal logged into the `winter2024-comp421.cs.mcgill.ca` server. Make sure that the screenshots include both the command/SQL and the corresponding outputs, even when you are providing the SQL separately in the submission. We need to know that the SQL you provided us is what you actually executed to get the output you got. Where necessary, you may truncate part of the output to fit the screen (try to enlarge your ssh terminal when you can). It is important that ALL of the SQL statement is visible in the screenshot and at least a good part of the results (if all of the result does not fit in the screen).

Please note that while each of the deliverables below can be somewhat developed in parallel, proper testing of points 5-9 can only happen once points 3 and 4 have been performed. Keep this in mind in your time management considerations.

1. **(0 Points)** Make sure you are using the DB2 database provided in the winter2024-comp421 server for your project (we have mechanisms to verify this). Not doing this and using another database (including PostgreSQL), using databases installed in your own personal computer, etc., will result in **20 points penalty**.

2. (0 Points) Include the relational schema of Project #1 according to the feedback given on it and that you are using for this phase of the project - even if you have not made any changes from project 1. Include this in the document **project2.pdf** under the section "**Relational Schema**". Not doing this will result in **5 point penalty**! Turning in your modified ER schema is not required, but you are free to do so.

3. (20 Points) Write a SQL database schema for the relational schema you have designed using the `CREATE TABLE` command and enter it in the database. Choose suitable data types for your attributes. Indicate primary keys, foreign keys or any other integrity constraints that you can express with the commands learnt. Indicate the constraints you cannot express. The Online Information contains detailed information about data types, and the `CREATE TABLE` statement.

Once you have figured out the DDLs, you can write them into the `createtbl.sql` file and have it executed using the `createtbl.sh`. Verify that all the tables got created correctly in the log file. and turn in the `createtbl.log` produced by the script along with the `createtbl.sql` file.

Next, write the corresponding `DROP TABLE` statements into the `droptbl.sql` file and have it executed using the `droptbl.sh`. Verify that all the tables got dropped correctly in the log file. and turn in the `createtbl.log` produced by the script along with the `droptbl.sql` file.

Once properly written, the above scripts will be very handy for you to "reset" the database quickly throughout your development time if you make mistakes or want to make changes, etc.

To be turned in:
~~Turn in the `createtbl.sql` and `droptbl.sql` files as well as the `createtbl.log` and `createtbl.log` files that were produced when you run the `createtbl.sh` resp. `droptbl.sh` scripts.~~ Under a section "**Pending constraints**" in the **project2.pdf**, indicate any constraints (use your ER and relational translation notes to check) that you cannot express in your database implementation.

cities
participation
constraint

4. **(20 Points)** Next we will load data (you should have the tables created in the database for this). Make sure that you have at least 5 (unless limited by the actual possible data values of its domain) records in each table that you have created. 5-20 records might be sufficient for several of the tables. **Do not insert more than 500 records in any given table!**

   When generating larger number of records, you can use clever techniques like using shell scripts, a spreadsheet (like Excel, google, etc.) and writing some formulae to convert a table of values in a spreadsheet into corresponding SQL insert statements. This can minimize your manual effort and chance of making mistakes. There are also some websites that help you generate random data (investigate and explore!).

   Once you have figured out your insert statements, write them into the sql file `loaddata.sql` and execute them using the `loaddata.sh`. Verify the log file `loaddata.log`.

   If properly utilized, this script, along with the scripts from the previous question can make your development "birth pangs" mild, where you have to constantly keep changing structures and data until you "stabilize" your system.

   I recommend that you read forward to the rest of the questions to help you get an understanding of the kind of data that you will need to write some of the SQL queries that are being asked and plan accordingly.

   To be turned in:
   Turn in the `loaddata.sql` file as well as the log file `loaddata.log` that was produced when running the `loaddata.sh` script.

5. (20 Points) Write five queries on your project database, using the select-from-where construct of SQL. The queries should be typical queries of the application domain. Simple queries over one table that select some attributes and have only basic conditions in the WHERE clause are not allowed. Instead, each query should exhibit some advanced features: queries over at least three tables, aggregation together with group by over at least two tables, using WITH, subqueries, combination of set and join operators, etc.

   To be turned in:
   Create a section "**SQL Queries**" in your **project2.pdf** with subsections **"Query1"** to **"Query5"**. Each of these subsections describes one of the queries you have designed. You have to present (a) in plain English what information the query exactly returns, (b) the SQL query statement as plain text, (c) the screenshot of the query being executed in DB2 using the command line utility that shows its output. Your screenshot may truncate off some part of the output if it is not possible to fit all of it into a single screen but it should include ALL of the SQL statement.

6. (8 Points) Write two data modification commands for your application. Both commands should be "interesting," in the sense that they involve some complex feature, such as inserting the result of a query, updating several tuples at once, or deleting a set of tuples that is more than one but less than all the tuples in a relation.

   To be turned in:
   Create a section "**SQL Modifications**" in your **project2.pdf** with subsections **"Mod1"** and **"Mod5"**. Each of these subsections describes one of the modification commands you have designed. You have to present (a) in plain English what modification the statement exactly performs, (b) the SQL update/delete/insert statement as plain text, (c) the screenshot of the statement being executed in DB2 using the command line utility that shows both the input statement and the output.

7. (14 Points) Create two views on top of your database schema, show how to use them and what happens when you try to update the views.One of the views should be over a single table with only simple constraints and returning some of the attributes of the table.

   To be turned in:
   Create a section **"Views"** in your **project2.pdf** with a subsection for each of the views (**"View1"** and **"View2"**). For each view, you have to present (a) in plain English what the data of the view represents, (b) the CREATE VIEW statement, (c) a screenshot of the statement being executed in DB2 using the command line utility that shows both the input statement and the output, (d) a screenshot of a SQL query that selects everything from the view truncated to just 5 records, (e) a screenshot of when you insert a new record into the view that has valid domain values for all attributes of the view together with the result this insert causes, (f) an explanation of the result of the insert you attempted (Indicate which explanation given in the DB2 manual is the one that explains the result for the insert you attempted).

8. (8 Points) Add two `CHECK` constraints that are important in the context of this database. Maybe you had indicated a constraint during your specification that you would like to ensure and that you can express with the options we discussed in class.

   To be turned in:
   Create a section *"Check Constraints"* in *project2.pdf* with a subsection for each of the views (**"Check1"** and **"Check2"**). Under this section, (a) describe what your constraint achieves, (b) show the constraint statement, (c) a screenshot of running this command on the database, (d) a screenshot of the execution trying to insert a record that violates this constraint including the error result the database returns.

9. (10 Points). **Creativity Points:**
   The purpose of this is to encourage students to try innovative approaches in their solutions. You can implement either one of it. Report this in a section **"Creativity"** in **project2.pdf**.

   - **Real data sets:** Generate for at least one table meaningful data for all its attributes. For example, instead of names like 'customer001', 'customer002' etc, it should be like 'Mike McCarthy', 'Katie Lohan'. The table must have a foreign key to another table or another table must have a foreign key to this table. The table must have a minimum of 100 records.
     You may use data provided by other websites etc. for this purpose, but references need to be provided.
     WARNING !! Do not generate more than 500 records per table.
     To be turned in:
     Turn in a description of which table has the meaningful data and how you created it. If you have used programs and scripts, attach them as extra documents and explain their content within the **"Creativity"** section in **project2.pdf**.
   - **Advanced SQL Features:** Use advanced SQL features, NOT covered in class. This includes Recursion, OLAP SQL Functions (Such as Window Functions), etc. For this, you can use internet/Manuals to find lot of examples and information on syntax.
     To be turned in:
     Turn in (a) a brief description of the purpose of the query, (b) the SQL statement, (c) a screenshot of running the command in the database and the output (possibly truncated).
   - **Complex Analytical Queries:** Come up with a complex business requirement (to read data) and develop SQL queries to substantiate it. That is, implement one advanced analytics task. It might need more than one query.

     Analytical queries are characterized by the fact that they work over many tables, and a larger portion of records from those tables and provides a very selective or summarized output. Aggregation operators are often an integral part of such queries. Some examples are given below.
     – Give a list of customers between the ages of 30 and 40 who has been depositing more than $500 into their savings account every month for the last one year and has not enrolled in any pension plans. (This is useful in finding a potential list of customers with savings/habits who will make a good target for marketing pension plans).

     – Find the average insurance claims by various age-categories, marital status, income, make of primary car, ownership of home, etc. for the customers in PQ. (This is useful in determining what are the 'risk' categories, that can be charged higher premium).

– Find customers who used to spent more than $50 on average per visit and has made at least 10 visits in the last three years, but has not been visiting in the past six months and has address in ‘Montreal’. (This is good target to sent some discount vouchers to bring back old customers).

To be turned in:
Turn in (a) a brief description of the purpose of the analytical task, (b) the SQL statement(s) that solve the task, (c) a screenshot of running the command(s) in the database and the output (possibly truncated).

10. (0) Points. Indicate in one paragraph how you worked together (how many meetings) and how each of the team members contributed to the project deliverable. If you do not describe this, you will get 10 penalty points.