Project for CSC 540: Database-Management Concepts and Systems

This is the description of the task for the main project for CSC 540. The details may change over time, so please visit the online version again. The project will be based on the case study presented in the **narrative**. Your main task is to design and develop a database system. In order to do this systematically, you will need to perform *all and only* the subtasks described in terms of the three project reports. *(Please note that you will not receive extra credit if you do more than is required; if in doubt, please see the instructor or TA before the due date for each report.)*

Implementation Plan

The way this project is envisioned you will carry out the following steps.

1. Understand the problem.
2. Create the local E/R diagrams (see item 7 in project report 1).
3. Create the local relational schemas (see item 9 in project report 1).
4. Create the database schema (see item 1 in project report 2).
5. Input the schema into the DBMS and populate the base relations (see item 3 in project report 2).
6. Design SQL queries and updates to realize the operations in the **narrative**, and test the queries (see item 4 in project report 2).
7. For each operation, create application code to exercise the operations appropriately (see items 2 and 3 in project report 3).
8. Test by hand.
9. Give a demo to the TA and the instructor (see item 5 in project report 3).

Organization

The project is recommended to be a team effort, each team consisting ideally of (and never more than) 4 students. Please choose your project partners by the announced date. There is no extra credit for forming a team of fewer than 4 people, but such teams are allowed. In such cases, the number of application programs required for the assignment is scaled down, but the remaining tasks are not scaled down.

The team will have the following functional roles. Every member will have to do some database design and application programming, so the roles can be rotated among the members. Each team member will write the same number of application programs. The prime and backup application programmers will do this first.

Software Engineer (*prime* and *backup*)
: Responsible for designing the software architecture, including transaction logic.

Database Designer / Administrator (*prime* and *backup*)
: Responsible for designing E/R diagrams and relational schemas, based on the informal specifications given in the **narrative**. Also responsible for database-schema maintenance and analysis of query execution. Develops advanced transactions in collaboration with the software engineer.

Application Programmer (*prime* and *backup*)
: Responsible for designing and implementing the actual applications (operations) that use the designed database.

Test Plan Engineer (*prime* and *backup*)

Ensures code quality by designing and implementing test plans for the applications in the project. Also responsible for documentation.

Grading

The programming project will be implemented in a system and described in reports. The reports will include E/R diagrams, relational schemas and their justifications, physical design and its justification, description of the queries and transactions (embedded SQL and program logic). The grading for the project will be totaled over four modules as follows. Some of these categories correspond to specific items in project reports (as listed). The others are spread over several items.

Section	Category	Max Points
G	**General**	**340 total**
	Problem description (1.1-1.6)	200
	Documentation (1.8, 1.10, 3.4)	140
S	**Schemas**	**520 total**
	E/R diagrams (1.7)	160
	Relational schemas (1.9, 2.1, 2.3)	240
	Integrity constraints, etc. (2.2)	120
D	**Data access logic**	**240 total**
	SQL queries (2.4)	240
A	**Applications**	**500 total**
	Programs (3.2)	60
	Advanced transactions (3.3)	120
	Demo (3.5)	320
P	**Peer evaluations** (1.11, 2.5, 3.6)	**Will be reflected in your total**
T	**Total**	**1600 plus peer evaluations**

Project Reports

- Project report #1 (520 points plus peer evaluations) *Please include one or more separate pages with a complete list of your assumptions about the project domain (as described in the narrative). Your entire report can be graded only with respect to your assumptions, and is not going to be graded in their absence.*
    1. (10 points) Describe the problem concisely (fewer than about 250 words). Why is a database (as opposed to, say, a simple set of files) a good idea for this task? Submit your description and explanation.
    2. (10 points) Describe the intended classes of users of your database system. (One possible class of users might be "staff who do billing".) Submit your descriptions.
    3. (10 points) Identify 5 main "things" about which you will need to keep information, and the

information you will need to keep. Submit the names of the 5 "things" and the information you need to keep about them.

4. (10 points) Describe realistic situations where using your database system will require handling any two of the operations (see Tasks and Operations in the **narrative**); in each situation, you may consider one or more operations. (The assignment is to describe a *total* of two operations, using a *total* of either one or two situations.) Describe each situation in about 50 words. Submit your descriptions.
5. (80 points) Sketch the APIs (application program interfaces; for this project, an API of an operation is just inputs and outputs for the operation) required for each of the operations listed in Tasks and Operations in the **narrative**. (In some cases the output will be just a confirmation.) Submit the APIs. Note: A lot of points will be taken off if you miss some of the operations in the narrative.
6. (80 points) Give short descriptions (no more than 50 words each) of the views of the data that correspond to the intended classes of users, one view per user class. Each view should reflect all and only the **data** (rather than operations) in the database that is relevant to all operations for the given class of users. For example, the hotel-receptionist view may reflect all the check-in and check-out information about the guests, and may represent the entire database as just one table for guests, rooms, and check-ins. You do not need to give all the details in your descriptions. Be careful not to include users' operations (from Tasks and Operations in the **narrative**) in your views. Submit your descriptions, one per user class.
7. (160 points) Construct local E/R diagrams, one for each view in item 6. In each diagram, you need to reflect all database information that is relevant to all operations for the given class of users. (See item 6 above and the **narrative**). Submit all the diagrams.
8. (40 points) Document the local E/R diagrams. Highlight any design decisions - specifically, describe why you have the entities and relationships that you do. Submit your documentation.
9. (80 points) Derive a local relational schema from each of the above local E/R diagrams (see item 7). Please do not be creative in your translations - points will be taken off if your translations from E/R diagrams to relation schemas cannot be done mechanically as explained in the textbook. Submit all the relational schemas - one per user class.
10. (40 points) Document the local relational schemas. Highlight any design decisions - specifically, explain why you decided to make the relations you did and how each entity and relationship in the E/R diagram is captured in the relational schema. Submit your documentation.
11. Peer evaluations (via submit board). For each project report, if you (individually) wish to get a grade for the report, you must submit your own peer evaluation of each member of your team. Please submit a single plain-text file *per student* via the submit board, with your own evaluation of all members of your team including yourself, using the values (such as "excellent", "ordinary", or "superficial", to grade your team members' *participation in the teamwork*) listed in the peer-evaluation form. No signature is needed in your submit-board submission.

- Project report #2 (520 points plus peer evaluations) *Please include one or more separate pages with a complete list of your assumptions about the project domain (as described in the narrative). Your entire report can be graded only with respect to your assumptions, and is not going to be graded in their absence.*
    1. (80 points) Derive a global relational database schema from the schemas you obtained at step 9 in project report 1. Normalize to at least 3NF. Submit your database schema and the explanations why your relation schemas are in at least 3NF. Note: to conclude that your relation schemas satisfy (at least) 3NF (and to get full credit for this item), you need to consider and discuss *all* functional dependencies that the instructor can assume might hold on your schemas, rather than

only those dependencies that are obvious and do hold on your schemas.

2. (120 points) Describe any design decisions for the global schema. Identify and explain all integrity constraints of the following types: NOT NULL, key, and referential integrity. Describe which attributes are allowed to be NULL, why, and what a NULL value means for each attribute on which it is allowed. Submit all your descriptions and explanations.
3. (80 points) Using the terminal for MariaDB (see https://www.csc.ncsu.edu/techsupport/technotes/mysql.php), for all your relation schemas create base relations with the right attribute domains and with *all* the integrity constraints you listed in item 2 of this report. Populate the base relations with 4-8 rows each. Show what is in each table by printing out, for each table, the answer to the "SELECT * FROM" query. Submit the printouts of (1) all your "CREATE TABLE" statements, of (2) all your "SELECT *" queries, and of (3) the answers to all the queries on the terminal for MariaDB.
4. (240 points) Write interactive SQL queries for each operation in the **narrative**, and test the queries on the terminal for MariaDB:

   4.1 (140 points). Make assumptions to justify the constants in your queries: for example, for the operation "show all staff members who are receptionists" assume that the database has information about three receptionists. Use these fictitious constants to ask specific SQL queries and to make specific updates on your stored relations. For the queries, choose the constants so that a non-empty answer set is returned; you may need to add additional rows to do so. For each operation in Tasks and Operations in the **narrative**, submit the query or update (whichever is appropriate) for the operation and the printout of the response to the query/update on the terminal for MariaDB. Note: A lot of points will be taken off if you miss some of the operations in the narrative.

   4.2 (70 points). Use the EXPLAIN directive in MariaDB (see course-project web page) to study execution plans for your queries (from item 4.1 above). Find two queries for which EXPLAIN shows full table scans; print the EXPLAIN outputs for these queries. For these two queries, create appropriate indexes; print the EXPLAIN output that shows the use of the indexes. For each query, submit the printouts of: (1) the SQL query, (2) the execution plan (EXPLAIN) for the query which shows at least one full table scan, (3) your index-creation statement in SQL, and (4) the execution plan (EXPLAIN) for the query where the full table scan has been replaced with a use of your index.

   4.3 (30 points). For any two of your SQL queries ("select" statements only, rather than "insert/delete/update") **with joins**, explain why the queries are correct - one explanation per query. If you come up with erroneous solutions prior to the correct one, also include those as part of the explanation. For each query, submit the query itself and your explanation. Your explanations will consist of two parts:

   - the corresponding relational algebra expression, **and**
   - a *correctness proof* that shows that the query answer always corresponds to the query specification;
     for example, a correctness proof of the query
     "SELECT e.Lname, d.DeptName FROM Employee e, Department d WHERE e.DeptNo = d.DeptNo,"
     with specification
     "return the last names of all employees together with the names of their departments,"

can be as follows:
"Suppose *e* is any tuple in the Employee relation, and *d* is any tuple in the Department relation, such that the value *e.DeptNo* is the same as the value *d.DeptNo*. Each such combination of tuples *(e, d)* gives personal information about one employee, together with all information on the department for that employee. For each such combination *(e, d)*, the query returns the value of Lname and the value of DeptName. These values are the last name of the employee and the name of the department for the employee. But this is exactly what our query should return; see the specification."

5. Peer evaluations (via submit board). For each project report, if you (individually) wish to get a grade for the report, you must submit your own peer evaluation of each member of your team. Please submit a single plain-text file *per student* via the submit board, with your own evaluation of all members of your team including yourself, using the values (such as "excellent", "ordinary", or "superficial", to grade your team members' *participation in the teamwork*) listed in the peer-evaluation form. No signature is needed in your submit-board submission.

- Project report #3 (560 points plus peer evaluations). *Please include one or more separate pages with a complete list of your assumptions about the project domain (as described in the narrative). Your entire report can be graded only with respect to your assumptions, and is not going to be graded in their absence.*
    1. Submit, in hardcopy, revised versions of the previous reports - you will get credit for the improvements, scaled by 50%. You will need to submit both the relevant *numbered* pages of the original reports and your revisions; the revisions should (1) mention the item and page numbers in the original report and (2) have the improved parts highlighted.
    2. (60 points) Write all the required applications and test them appropriately. Submit all source code to the submit board.
    3. (120 points) In at least two of the applications (see item 2 above), use transactions to ensure that the applications work correctly even if they encounter unexpected events. (Example: a credit-card authorization fails because a staff member has entered an invalid credit-card number.) Document the program logic of the transactions in the applications. Submit in hardcopy (1) the parts of the code that contain the transactions (points will be taken off by the grader if these parts of the code are not easy to locate), and (2) your documentation. Make sure that your transactions use the COMMIT and ROLLBACK statements.
    4. (60 points) Document your programs. The documentation should be part of the code (submitted via the submit board), but will be graded in addition. In a separate hardcopy submission, highlight *high-level* design decisions, which are any choices/decisions that you had to make when designing your database and applications. As part of the documentation submitted in hardcopy, explain who in your team played what functional role (e.g., software engineer, database designer/administrator, etc, see above under Organization) in each part (1 through 3) of the project. Submit the documentation.
    5. (320 points) Demo. Graded on the functionality & robustness of your programs. (You are *not* required to implement a graphical user interface or a web-based interface, and will *not* get extra credit for doing so.)
    6. Peer evaluations (via submit board). For each project report, if you (individually) wish to get a grade for the report, you must submit your own peer evaluation of each member of your team. Please submit a single plain-text file *per student* via the submit board, with your own evaluation of all members of your team including yourself, using the values (such as "excellent", "ordinary", or "superficial", to grade your team members' *participation in the teamwork*) listed in the peer-

evaluation form. No signature is needed in your submit-board submission.

Rada Chirkova (chirkova AT csc ncsu edu)