CMSI 486 DATABASE SYSTEMS

Dr. Stephanie E. August

Enterprise Model Project

Enterprise Model Project

Introduction

Over the course of the term, you will be designing and implementing a database as described in appendix A. This project is broken into a number of deliverables as listed in table 1. The due date for each deliverable is listed in appendix A.

Table 1. Project deliverables.

Deliverable	Assignment
1	Describe the enterprise
2	Define the environment
3	Develop logical and conceptual models of your enterprise
4	Define the database and formulate queries
5	Consider database integrity and security
6	Implement the database
7	Lessons learned
final	Completed report and files containing design, tables, queries, traces

You are upload each of your deliverables to the relevant project component on MyLMUConnect. There should be a distinct and separately page-numbered document for each of the sections listed in table 2 below. A detailed description of each section follows.

There are nine sections in all.

Pages are to be numbered consecutively within each section. For example, the first page of the enterprise description (section IV) should be numbered IV-1, the second page of the enterprise description should be numbered IV-2, and the *n*th page of the description should be IV-*n*. If the design section (section V) spans 10 pages they should be numbered V-1 through V-10.

Label all tables and figures. Table labels appear before the table, centered. Figure labels appear after the figure, centered. Observe the examples in this document and follow a similar format.

As each interim assignment is reviewed, you are expected to incorporate into your next version any corrections, modifications, or additions which I have noted on your project, and upload those revisions to the relevant project component. The previous version of each section with corrections/modifications/additions indicated, must remain available on MyLMU|Connect for the duration of the semester. This is used as reference when reviewing subsequent submissions. Your final report should consist of one document for each project section, including the title page and comprehensive table of contents.

The title page and table of contents should be current with each deliverable, except for deliverable 1. The table of contents should include at least the top three levels of headings and relevant page numbers for each section of your project report.

Table 2. Project report sections.

Part of Deliverable #	Belongs in Section #	Section Contents
3	I.	title page [see sample in appendix C]
3	II.	Table of Contents [three levels of headings with page numbers]
1	III.	Description of the Enterprise
2	IV.	Definition of the Environment IV.1. Input and report forms. A list of input and report forms, with itemized data items. IV.2. Assumptions. List of assumptions for your enterprise database. IV.3. User-oriented data dictionary. IV.4. Cross-reference table.
3	V.	Enterprise Database Design V.1. Logical model of the enterprise. V.1.1. List of entities and attributes. V.1.2. List of relationships and attributes. V.1.3. Entity-Relationship diagram of the enterprise. V.2. Conceptual model of the enterprise. V.3. Table dictionary. V.4. Attribute dictionary.
4	VI.	Database and Query Definition VI.1. Database Definition. SQL DDL for your database objects. VI.2. Database Queries. English version of 10+ database queries, and the SQL DML for each database query VI.3. Review sign-off sheet . VI.4. Design Tradeoffs and Limitations. Discussion of the limitations of your design.
5	VII.	Database Integrity and Security VII.1 Functional Dependencies. A list of the functional dependencies that hold on your database. VII.2. Adjustments for Normalization. An explanation of the changes needed to normalize your database. VII.3. Integrity and Security. A list (in English) of the integrity and security constraints which are to hold on your database.
6	VIII.	Implementation Notes VIII.1. Indices. A list of the indices used by your database, with a justification for each. VIII.2. Data. The data used to populate your database. VIII.3. Query Trace. A trace of the execution of each of your queries. VIII.4. Implementation Assessment. An assessment of how smoothly your implementation went.
7	IX.	Lessons Learned

The remainder of this document issues a caveat to developers, then describes the assignments associated with each deliverable. Appendix A provides a project problem description. Appendix B includes a template that can be used to develop your project report. Appendix C includes a sample title page. A copy of the review signoff sheet required in section VI can be found in Appendix D.

You are expected to be familiar with the required contents, format, and due date of each section of your project report.

Notes to the Wise

Plan ahead, work all during term. You can't do this project in a single night. You can't even do an entire deliverable in a single night for the most part!!!

Remember the five *P*s: *P*roper *p*lanning *p*revents *p*oor *p*erformance.

Doing a good job on the project requires spending 8 to10 hr/week on the course.

The deliverables in this project make sense and guide you through the development process if you follow them in order. The steps will appear redundant otherwise. For best results, complete each deliverable in sequence, and revise with each deliverable.

Deliverable 1: Describe the enterprise.
Turn in: Section III. Description of the Enterprise: Textual description of your enterprise.

Write a textual description of the enterprise (500 words). Describe the purpose of the enterprise, the people involved, and the information processing performed. Include descriptions of the records that need to be maintained, the entities to which they relate, and the relationships that exist among the entities. At this point, the goal is to understand the enterprise that you will be modeling. This description will evolve, and, perhaps, become more detailed, over the course of the project. *Do not include any "techtalk" in your description!* That is, avoid database-specific terms such as entity, relationship, or cardinality. Gear the description to a non-technical manager. Use the description in Appendix A (if provided) as a *starting* point, tailoring it to your specific enterprise model and adding details and questions/queries as needed.

Deliverable 2: Define the environment.
Turn in: Title Page
Table of Contents
Section III. Description of the Enterprise (revised as needed)
New: **Section IV.1. Input and report forms:** List of input and report forms, with itemized data items.
Section IV.2. Assumptions: List of assumptions for your enterprise database.
Section IV.3 User-oriented data dictionary.
Section IV.4 Cross-reference table.

In this step you will take a closer look at the environment in which your database will exist. Assume that you are having or have had a series of meetings and interviews with users of the proposed system, to determine their data needs and preferences. In real life, you will be working with the end users from day one, to ensure that the final product will meet their needs, as well as to gain their support for and confidence in the project.

First, consider how information will be put into the system, and retrieved from it. If you were actually working with an enterprise to design and implement an information processing system, you would have to identify and write out the format for each input document and database report, and for each input and output screen for every routine transaction to be performed against the database. For example, suppose we were developing a DBMS for a software consulting firm.

One form we might use would be the consultant application form shown in figure 1. A screen listing potential consultants for a job appears in figure 2. In a large enterprise, the data analysis or business systems analysis staff would most likely perform this deliverable.

Without actually designing the layout of the input/output screens and database reports, carefully think through daily operations. *Draw up a list of the input form and report forms, and itemize the data values utilized on each.* For example, corresponding to figure 1 we would have the list shown in figure 3.

Next, ***write out a list of assumptions*** **for your environment** that you would have picked up during your customer meetings. You will add to this list as the project progresses. Some assumptions from the consulting enterprise are shown in figure 4.This example is by no means exhaustive.

Now, to help refine your system, you develop two documents:

- A *user-oriented data dictionary*, consisting of an alphabetical list of every data item referenced in any document, report, or routine transaction and an informal definition for each item. The dictionary should be created in table form, with two columns: Datum, and Informal Definition. An example of the dictionary can be found in table 3.

  Users will reference this document independently from accessing the database itself to better understand the enterprise semantics and write appropriate queries and database programs.

SoftWare Consultants, Inc.
Application Form

Date of Application _____
Name _____
Address _____
_____
Telephone _____ Email _____ Fax _____
Social Security Number _____
Date of Birth* _____ Sex* _____
Skills -- For each of the following, indicate your experience level:

	None	Some	Extensive
C			
C++			
Windows			
Object-Oriented Programming			
Relational DBMS			
CLIPS			
Lisp			

Date Available to Work: _____
Restrictions on Work Dates: _____
_____
To be filled in by interviewer:
Date Hired _____ Reason not hired _____
* SoftWare Consultants, Inc. does not discriminate of the basis of age or sex of applicant.

Figure 1. Input form: Consultant application.

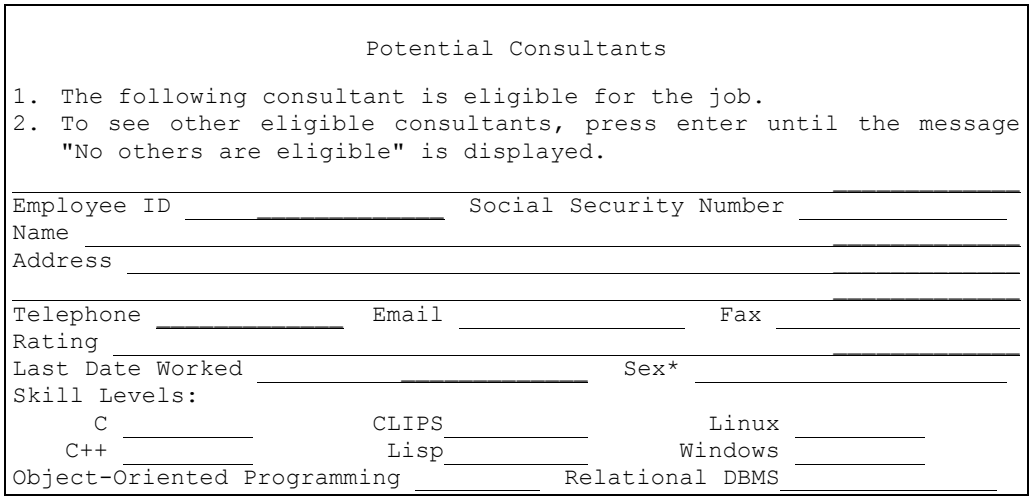

Figure 2. Output Screen: Potential consultants for a job.

- A *cross-reference table* showing which items appear on the various documents, reports, or transactions you have already identified. Table 4 shows how the data would map to the documents, reports, and transactions of the consulting enterprise.

  This document allows us to trace requirements to data items, forms, reports, and database transactions. If we make a change to any of these artifacts, this document will enable us to understand the impact the change will have on other artifacts.

SoftWare Consultants, Inc. -- Application Form

- Date of Application
- Name
- Address
- Telephone
- Email
- Fax
- Social Security Number
- Date of Birth
- Sex
- Skill experience level (None/Some/Extensive)
  - C
  - C++
  - Windows
  - Object-Oriented Programming
  - Relational DBMS
  - CLIPS
  - Lisp
- Date Available to Work
- Restrictions on Work Dates
- Date Hired
- Reason not hired

Figure 3. List corresponding to the consultant application input form of figure 1.

1. Prospective consultants fill out an application form, and are interviewed by both a regional consulting manager and a current consultant to ascertain interests of consultants and confirm their skill levels.
2. Consultants are assigned to only one job at a time.
3. Both skill level and requirements of the job determine a consultant's wage.
4. Clients are charged a daily rate and are billed weekly.
5. Assignments can be from one day to several months duration.
6. A consultant might work at the customer site, an office of SoftWare Consultants, Inc., or at the consultant's home, depending upon the requirements of the job.

Figure 4 Some assumptions for the SoftWare Consultants, Inc., enterprise.

Table 3. User-oriented data dictionary.

Datum[1]	**Information Definition**
consultantName	The name of a consultant in the form <last>, <first and rest>. Example: Doe, Jane Marie Smith
consultantAddress	The address of a consultant. Includes street number, street name, unit number, city, state, postal code if relevant, and country, if outside the U.S. Example A: Example B: 123 N. Main Street, Suite 14 O'Donnell Road Central, CA 99999 Doolan, County Clare, Ireland
consultantPhone	The telephone number of a consultant. Includes area code (if US) or country code and city code (if not US) and phone number.
potentialConsultantName	The name of a prospective consultant. Refer to *conName* for format.
potentialConsultantAddress	The address of a prospective consultant. Refer to *conAddr* for format.
potentialConsultantPhone	The telephone number of a prospective consultant. Refer to conPhone for format.
software_skill	Software skill possessed by a person or required by a job. Examples: C#, LISP, CLIPS, Java, Linux, Windows, relational DBMS
software_skill_level	Level of expertise of a skill either possessed by a consultant or required by a project. Allowed values: *None, some, extensive.*
...	...

Table 4. Mapping of data to forms and transactions.

[1] Avoid using all capitals for your attribute names; it is easier to read names with mixed capitalization or lowercase with underscores (_) between words.

Datum	Form or screen								
	Prospective Consultant Application	Prospective Client Application	Client Information	Consultant	⋮				
consultantName				X					
consultantAddress				X					
consultantPhone				X					
potentialConsultantName	X								
potentialConsultantAddress	X								
potentialConsultantPhone	X								
software_skill	X			X					
software_skill_level	X			X					
. . .									. . .

Deliverable 3: Develop logical and conceptual models of your enterprise.

Turn in: **Title Page**
Table of Contents (revised as needed)
Section III. Description of the Enterprise (revised as needed)
Section IV. Definition of the Environment (revised as needed)

New: **Section V.1. Logical model of the enterprise.**
V.1.1. List of entities and attributes.
V.1.2. List of relationships and attributes.
V.1.3. Entity-relationship diagram of the enterprise.
Section V.2. Conceptual model of the enterprise.
Section V.3. Table dictionary.
Section V.4. Attribute dictionary.

Now you are going to develop both the logical and conceptual models of your enterprise. First, you develop a logical or semantic model of your enterprise. This is a three step process:

- *Make a list of all entities and their associated attributes.*
  This may take several attempts, and different designers will arrive at different solutions. In identifying entities, you will examine the data dictionary you developed in Section V.3. Think about the enterprise, and enumerate the persons, places, events, objects, or concepts that you need to keep information about. The original data dictionary may have some items that you need not store in the database. They can be dropped from the list of attributes.

- *Make a list of relationships to be represented and any descriptive attributes for them.*
  At this point you may decide not to store some of the items from the original data dictionary. For example, information such as *payroll totals* might only be needed for a periodic payroll report. It might make more sense to calculate it when needed, than to

explicitly store it in the database. Document any changes you make, and retain this information in the history section.

For the SoftWare Consultants, Inc., example we might have the entity sets:

CONSULTANT[2]: consultantName, consultantPhone, consultantAddress, consultantID, SS#, DoB

SKILLS: software_skill

and the relationship set:

CONSULTANT_SKILLS: consultantID, software_skill, software_skill_level

- *Draw an E-R diagram to represent the enterprise.*
  Be sure to identify relationship cardinalities, and any weak entity sets. Use generalization and aggregation as necessary to express relationships.

  This document provides a picture of the entire enterprise model. People working with the database will use the ERD to navigate a database the way we use a road map to navigate through our cities and states.

Pointers:

- The textual description of the enterprise you are modeling should reflect the semantics of your entity-relationship diagram *accurately*, including entity sets, relationship sets, cardinalities, and attributes.
- Each entity set should represent a single concept -- don't confuse *order* and *product*, for instance, or *order* and *customer*.
- Explicitly represent relationships between/among entity sets in the E-R diagram.
- Each attribute should have a unique name!
- Indicate the primary key (PK) for each entity set.
- Indicate which attributes are candidate keys (CK) in each attribute. Remember that the PK is always a CK!
- Do not include foreign keys in the ERD. They are implicit in the relationships represented in the diagram. If you include FKs in spite of this request, you must explicitly identify each foreign key attribute as a foreign key in each relation and indicate its source or parent table.

[2] Entity and relationship names are capitalized here to make them stand out; use mixed case or lower case with underscores in the actual database.

- Indicate each discriminator of a weak set, where such a discriminating attribute exists.

Next, *produce a conceptual model of your enterprise*, by reducing the E-R diagram to tables in the relational model, as we did in class. You can do this using either the

table-name-R(*attribute-1*, ..., *attribute-n*)
CK: *attribute-1*, *attribute-i*
FK: *attribute-j* references *table-name-S.attribute-k*

notation, or the

table-name-R		
attribute-1	...	*attribute-n*
PK CK		 CK FK references *table-name-S.attribute-k*

notation. Be sure to indicate in each table which attributes participate in the primary key, which attributes are parts of candidate keys, and which attributes are foreign keys. Remember to indicate the source of each foreign key.

The final steps of this deliverable are to create a *revised* data dictionary in two parts. The first part, the "table dictionary", will consist of *a three-column table* listing *each table* to be included in the database, the *attributes* that are in the table, and *an informal definition of the table*. This will provide an easy reference guide to your database. The second part of the data dictionary, the "attribute dictionary", will consist of a revised version of the user-oriented data dictionary described in section IV.3 above. *To that document, add a column specifying the table in which each attribute is used*. If an attribute has been renamed and used as a foreign key in a table, add it to the attribute dictionary list, specifying the table in which the renamed version is used. In the definition portion of the entry, indicate the attribute which the foreign key attribute references.

Deliverable 4: Define the database and formulate queries.
Turn in: **Title Page**
Table of Contents (revised as needed)
Section III. Description of the Enterprise (revised as needed)
Section IV. Definition of the Environment (revised as needed)
Section V. Enterprise Database Design (revised as needed)
New: **Section VI.1. Database Definition:** SQL DDL for your database objects.
Section VI.2. English version of 10+ database queries, and the SQL DML for each database query.
Section VI.3. Review sign-off sheet.
Section VI.4. Design Limitations: Discussion of the limitations of your design.

Write the SQL DDL statements to create all tables needed to implement the design completed in Deliverable 3. Include relevant inetgrity constraints for foreign keys, that is, for each foreign key declared, specify in the DDL the action that should be taken if the related valued in the parent table is deleted or updated. Use ALTER TABLE statements to include integrity constraints

rather than declaring them in the DDL for each table. Include your DDL in one or more `.sql` files, with an accompanying database build file that will load the SQL statements.

Compose English language queries that are needed to process at least ten (10) nonroutine requests for information from the database just created. These ten queries should be OLAP-oriented, rather than OLTP-oriented. For each, write the request in English, followed by the corresponding SQL command. At least 5 of these should be "difficult" queries, that is, queries involving multiple tables and/or complex operations on a single table.

Team with another student to review designs, schemes. A Review Sign-off Sheet is found in appendix D. Have the other student sign off on the design using this sheet, and include evidence of the sign-off in the report. The comment section should have something more meaningful than "Looks OK to me." You will receive two grades for this sheet. First, you be graded on the quality of the review and comments that you provide for *some other* student. Second, you will be graded on whether or not a completed sheet (with your reviewer's comments) is included with your report.

Identify the limitations of your design. Include a discussion of the kinds of information which are difficult to extract from your database, due to its content and structure and/or due to the limitations of SQL. Propose modifications to your enterprise model or the query language which would make it easier to extract the information.

Deliverable 5: Consider database integrity and security.

Turn in: **Title Page**
Table of Contents (revised as needed)
Section III. Description of the Enterprise (revised as needed)
Section IV. Definition of the Environment (revised as needed)
Section V. Enterprise Database Design (revised as needed)
Section VI. Database and Query Definition (***revised to incorporate the DDL to support the constraint listed in Section VII.3***, and other revisions as needed)

New: **Section VII.1. Functional Dependencies:** A list of the functional dependencies that hold on your database.
Section VII.2. Adjustments for Normalization: An explanation of the changes needed to normalize your database.
Section VII.3. Integrity and Security: A list (in English) of the integrity and security constraints which are to hold on your database.

At this point you need to concern yourself with integrity and security issues related to your database. First, let us consider normalization. This will be a two-step process:

- *Identify all of the functional dependencies that hold on the database.*
- *Normalize each relation* identified in the preceding deliverable. Be sure that every attribute listed in the first step above appears in at least one table. Then decide whether the table should be implemented in the highest normal form. If not, explain why.

Modify your environment definition and cross-reference table from deliverable 2 to reflect any changes resulting from the normalization process.

Revise your table and attribute dictionaries from section V.3 and V.4 as needed.

Revise your entity-relationship diagram as needed.

List the integrity and security constraints that should hold on your database. Consider how the use of foreign keys, assertions, triggers, and *grant* statements support integrity and security in your database. Your writeup should clearly identify classes of users and the privileges accorded to each. Justify each constraint employed. Identify and define any views required. Any views created need to be added to both parts of the data dictionary. Write the SQL statements needed to support these constraints, wherever possible. If SQL does not support integrity and security constraints which you would like to maintain on your DB, explain where it falls short, and how you can compensate for the shortcoming(s). (You can suggest application programs or extensions to SQL to shore up the constraint support mechanism.) Refer to the textbook for ideas.

Deliverable 6: Implement the database.

Turn in: **Title Page**
Table of Contents (revised as needed)
Section III. Description of the Enterprise (revised as needed)
Section IV. Definition of the Environment (revised as needed)
Section V. Enterprise Database Design (revised as needed)
Section VI. Database and Query Definition (***revised to include the SQL DDL for index declarations listed in Section VIII.1***, and other revisions as needed)
Section VII. Database Integrity and Security

New: **Section VIII.1. Indices:** A list of the indices used by your database, with a justification for each.
Section VIII.2. Data: The data used to populate your database.
Section VIII.3. Query Trace: A trace of the execution of each of your queries.
Section VIII.4. Implementation Assessment: An assessment of how smoothly your implementation went.

Implement your database. Make any needed adjustments to your DDL files. Indicate where in your database an index needs to be added, and explain why it should be added. Add the DDL to generate each index, as needed.

Populate your database. Create a file containing the *insert* statements needed to perform your initial data load.

Implement, and test, and save each of the queries developed in Deliverable 4. If possible, test your implementation on more than one DBMS, and describe the differences you encountered, if any. If you use ANSI standard SQL, your code should run on any ANSI-compliant DBMS.

Create a trace of the creation and load of your database and the test of each query.

Deliverable 7:Lessons learned.
Turn in: Section IX. Lessons Learned.
Sections I-VIII (revised as needed)

Describe what you learned during the course of this project. This discussion should be about 500 words in length. Your lessons learned section should at a minimum address the following questions: What would you do differently next time? What database resources should you have used, or do you wish had been available? How many hours did you put in? How useful was the project in helping you learn the course material. What would facilitated completing the project? How do you like ORACLE or the DBMS engine you used on the project?

Final Deliverable

#1. Upload to MyLMU Connect:

Zip file containing .sql files with your DDL and DML and a README file that provides an inventory of the files included in the zip

#2. Turn in completed report:

Includes complete project report, lessons learned, SQL, English queries and SQL implementation, query trace, and an inventory of all files includes uploaded to myLMUConnect.

#3. Demo your project:

Schedule a one-on-one demo with instructor, to be held between Wednesday, 4 December, and Thursday, 12 December. If your project is in good shape, this should entail a 5-minute meeting during which you demonstrate that your three most complex queries run and return anticipated results. The queries will be selected at the time of the demo. More extensive testing will be done when the projects are reviewed. The demo will take longer if the selected queries do not appear to run properly.

Note:

Some of the ideas for this project were taken from:

Ricardo, Catherine M. *Database Systems: Principles, design, and implementation.* New York, Macmillan, 1990. 0-02-399665-X