CMPT 354
Database Systems

Simon Fraser University
Spring 2017

Instructor: Oliver Schulte

Assignment 1: Entity-Relationship Modeling. The Relational Model. MS SQL Server.

Instructions: Check the instructions in the syllabus. The university policy on academic dishonesty and plagiarism (cheating) will be taken very seriously in this course. *Everything submitted should be your own writing or coding.* You must not let other students copy your work. On your assignment, put down your **name**, the number of the assignment and the number of the course. Spelling and grammar count.

Group Work: Discussions of the assignment is okay, for example to understand the concepts involved. If you work in a group, put down the name of all members of your group. There should be no group submissions. Each group member should write up their own solution to show their own understanding.

For the due date please see our course management server https://courses.cs.sfu.ca .

Detailed submission instructions are provided at the end of the file.

Note on ER exercises: ER modelling is as much a science as an art, and not a rote procedure. The purpose of this assignment is to give you a bit of practice in capturing informal descriptions in an ER diagram. Just like in real life, if you feel that not all the details that you need to know have been completely specified, use your common sense. If in doubt, explain explicitly what assumptions you are making and how they are shaping your model. Another good strategy is to follow standard examples, like those in the books and lectures, unless you have a reason for doing something different.

Part I: Database Design for a Company Domain

Draw a single ER diagram that represents the specifications listed below. To help you do this, we break down the drawing for you into separate steps.

1. A company database needs to store information about employees (identified by *ssn*, with *salary* and *phone* as attributes), departments (identified by *dno*, with *dname* and *budget* as attributes), and children of employees (with *name* and *age* as attributes). Draw an ER diagram to represent these entities and their attributes.
2. Extend your ER diagram with relationships to represent the following information.
   a. Employees *work* in departments.
   b. Each department is *managed by* an employee (exactly one).
   c. A child must be identified uniquely by name when the parent (who is an employee) is known. We are not interested in information about a child once the parent leaves the company.
3. Write SQL statements that translate your ER model for part I into the relational model (that is, the SQL statements should create tables corresponding to the entities and relationships in the ER model). Chapter 7.6 and the lecture notes give you several examples of this process. You may want to first translate the ER model into a relational schema, then the relational schema into SQL commands.

Important Notes:

1. *You are not required to write down the relational schema in this format*:
   Students(sid: string, name: string, login: string, age: integer, gpa: real).
   We won't grade you on the schema, only on the SQL commands. We just think the schema will help you.

2. *Be sure to incorporate as many constraints mentioned in parts 1 and 2 as possible.* If there is a constraint mentioned in part I that you cannot capture in SQL statements (using the basis SQL data description constructs, that is, no ASSERT or CHECK commands), describe what the constraint is and explain why it cannot be captured.

Marking Criteria:

1. Total Marks for ER diagram (Parts 1 and 2): 100 Marks
   （1） Technical Correctness (50 Marks)
   （2） Notational Correctness (20 Marks)
   （3） Clarity of Diagram (20 Marks)
   （4） Presentation (including quality of explanations, if any) (10 Marks)

2. Total Marks for SQL statements (Part 3): 100 Marks
   （1） Technical Correctness (including match between specifications and design) (60 Marks)
   （2） Notational Correctness (20 Marks)
   （3） Presentation (including quality of explanations, if any) (20 Marks)

Part II. Database Design for an Airport Domain

Draw a single ER diagram that represents the specifications listed below. To help you do this, we break down the drawing for you into separate steps.

1. Consider a model of an airport with planes, models of planes, test of planes, and technicians. Draw an ER diagram to represent these entities and the following attributes.

   Planes have a unique registration number. Airplane **Models** are each identified by a model number (e.g. DC-80), and each have a capacity and a weight. A number of **technicians** work at the airport. You need to store for each the name, phone number, address, and salary. The airport has a number of **tests** that are used regularly to ensure that airplanes are safe. Each test has a unique FAA number, a name, and a maximum possible score. Extend your ER diagram with relationships to represent the following information.

   a. Each airplane is of one specific model.

   b. Each technician is an expert on one or more plane models. His or her expertise may overlap with that of other technicians.

   c. Transport Canada requires the airport to keep track of each time a given airplane is tested using a given test. For each testing event, the information needed is the date, the number of hours spent doing the test, and the score that the airplane received on the test.

2. Write SQL statements that translate your ER model for part I into the relational model (that is, the SQL statements should create tables corresponding to the entities and relationships in the ER model). Chapter 3.5 and the lecture notes give you several examples of this process. You may want to first translate the ER model into a relational schema, then the relational schema into SQL commands. We won't grade you on the schema, only on the SQL commands. We just think the schema will help you. *Be sure to incorporate as many constraints mentioned in parts 1 and 2 as possible.* If there is a constraint mentioned in part I that you cannot capture in SQL statements (using the basis SQL data description constructs, that is, no ASSERT or CHECK commands), describe what the constraint is and explain why it cannot be captured.

Marking Criteria:

1. Total Marks for ER Diagram (Parts 1 and 2): 150 Marks.
   （1）Technical Correctness (including match between specifications and design) (60 marks)
   （2）Notational Correctness (40 Marks)
   （3）Clarity of Diagram (30 Marks)
   （4）Presentation (including quality of explanations, if any) (20 Marks)

2. Total Marks for SQL statements (Part 3): 110 Marks
   （1）Technical Correctness (including match between specifications and design) (60 Marks)
   （2）Notational Correctness (30 Marks)
   （3）Presentation (including quality of explanations, if any) (20 Marks)

Part III: Programming.

7. This part requires installing SQL Server. Please see the installation instructions. Using CSIL SQL Server, it may be necessary to cite the database's name in the queries below. For example, for part a), the query should be
"SELECT COUNT(*) FROM AdventureWorksLT.SalesLT.Address" instead of "SELECT COUNT(*) FROM SalesLT.Address".

Execute the following commands and write down the results. We will cover the details and meanings of the commands later on in the course. For the moment, I want to verify that everyone successfully set up the AdventureWorks database, so do not worry about understanding how the commands work. [1 point each]

a) Tell me the number of records in the Address table.

```
SELECT COUNT(*) FROM SalesLT.Address
```

b) Tell me the average order quantity sold.

```
SELECT AVG(OrderQty) FROM SalesLT.SalesOrderDetail
```

c) Find the Product Models. Tell me how many there are (i.e., how many tuples/records are returned as a result of this query?)

```
SELECT DISTINCT(Name) FROM SalesLT.ProductModel
```

d) Tell me the number of addresses in Bellevue.

```
SELECT COUNT(*) FROM SalesLT.Address WHERE City = 'Bellevue'
```

e) Tell me the amount of the largest Sales Order.

```
SELECT MAX(TotalDue) FROM SalesLT.SalesOrderHeader
`
```

Total Marks for Part IV: 10

Marking Criteria:
1.Each query answer (1 Marks)
2.Setting up the system (5 Marks)

Installation Help.

This section is meant to assist you with getting a working version of SQL Server and with setting up the AdventureWorks database on it. You have two basic options: To use the CSIL lab installation, or to install SQL Server on your own system. *The only setup we are committed to supporting is in the CSIL lab*.

CSIL lab setup. You can run SQL Server in the CSIL lab. Instructions have been emailed. You can connect to CSIL remotely and run SQL Server that way; please see CSIL instructions. You should have read access to the AdventureWorks/AWorks database. For this assignment, read access is sufficient. The CSIL staff is preparing scripts to load the AdventureWorks tables into your CSIL database.

Personal SQL Server Copy. This involves two steps.

1. Downloading SQL Server and installing it on your system. Download SQL Server 2012 via SFU's "DreamSpark" authentication at this page: https://services.cs.sfu.ca/msdnaa/
2. Importing the AdventureWorks database into your copy.

See the course web page for detailed instructions.

Submission Details.

The main purpose of the submission instruction is to have a uniform submission format. Uniform submissions means faster grading time means you get your grades back more quickly! A secondary purpose is to give you a bit of experience with the ways in which SQL Server, and other database systems, allow you to explore your database design in the GUI.

1. For ER Modeling.

1. I strongly recommend using a drawing program, not hand-drawing, for the following reasons.
   a) The result will be more legible. You will lose points for a diagram that's hard to read.
   b) By hand-drawing you discourage yourself from improving your diagram because it's quite difficult to redo parts of a hand-drawn figure without redoing everything.
   c) Knowing how to use a drawing program is a good skill for life and work.
   d) It may be reasonable to mix electronic and hand drawing, for instance to get dashed lines for weak entities.
2. You can use any program you like, for example: PowerPoint, Visio from Microsoft, or yed (http://en.wikipedia.org/wiki/YEd) or check out http://en.wikipedia.org/wiki/Er_diagram#ER_diagramming_tools. (The trickiest part seems to be making partial keys for weak entities. For instance in yed, you have to make a node with text ---- and no circle, then move the node so that the --- appears in the right place.)
3. If you feel unsure about some of your design choices, you can add explanations. This is an option only; you can get full marks without writing explanations of your diagram.

Upload your diagram + optional explanations as a single pdf file called "part1-ER.pdf" (for question in part1) or "part2-ER.pdf" (for question in part2)

2. For Translate ER into relational models.

1. Submit a single SQL script that contains all your create statements. This should be executable in SQL Server. Call the file "xxx (question number).sql" .
2. After you execute the create statements to make tables for a database, you can use SQL Server to create a database diagram that shows the foreign key dependencies. Output the diagram to a pdf file called "xxx (question number)-diagram.pdf".
3. For each table that you created, show a screenshot of the table design. You can display the table design in SQL Server by right-clicking on the table and selecting "Design". Combine all the table design screenshots into a file called "xxx (question number)-design.pdf".
4. You may add optional explanations (e.g., why some constraints cannot be represented in SQL in your opinion). Please write these to another pdf file called "explanation-xxx (question number).pdf".

Upload your submissions as a single archive file called "part1-translate.zip" (for question in part1) or "part2- translate.zip" (for question in part2)

3. For Creating Relational Schemas in SQL

Same as in Part II, repeated here for clarity.

1. Submit a single SQL script that contains all your create statements. This should be executable in SQL Server. Call the file "xxx (question number).sql" .
2. After you execute the create statements to make tables for a database, you can use SQL Server to create a database diagram that shows the foreign key dependencies. Output the diagram to a pdf file called "xxx (question number)-diagram.pdf".
3. For each table that you created, show a screenshot of the table design. You can display the table design in SQL Server by right-clicking on the table and selecting "Design". Combine all the table design screenshots into a file called "xxx(question number)-design.pdf".
4. You may add optional explanations (e.g., why some constraints cannot be represented in SQL in your opinion). Please write these to another pdf file called "explanation- xxx(question number).pdf".

Upload your submissions as a single archive file.

4. For programming.

For each of the five queries, include a screenshot that shows how you execute the query and what answer you got. Upload the screenshots into a single pdf file called "part3.pdf".