SCHOOL OF
COMPUTER AND SECURITY SCIENCE

AUSTRALIA ECU EDITH COWAN UNIVERSITY

CSG1207/CSI5135: Systems and Database Design
Lab 07

Introduction

This lab allows you to practise the SQL covered in this week's lecture. Do your best to answer the following questions and write the specified queries. You are encouraged to experiment with SQL – it is a very flexible language, so if you can think of something that would be useful to achieve in a query, it can probably be done. This lab uses the "company" database, which you can create by running the script file Module 5 of the unit materials. It is assumed that you are working in SQL Server Management Studio, also covered in Module 5.

If you are having trouble writing an SQL query, read any error messages and try to fix the error. Search for examples in the unit materials, and ask your tutor for assistance if needed. Contact your tutor if you spot something which appears to be incorrect in any of the labs.

Lab Tasks

Q1. Using Management Studio, examine the tables in the "company" database. Identify the primary and foreign keys of all tables, and compare them to the ERD of the company database in the lecture of Module 5. Has the database been implemented correctly?

Q2. In the Object Explorer of Management Studio, right click on the "Database Diagrams" folder inside the "company" database and select "New Database Diagram". Create support objects if you are prompted to do so, and then select all the tables in the list and press "Add". Press "Close" and look at the ERD created by Management Studio. Does this ERD match the one in the lecture of Module 5?
(You may wish to save the ERD so you can return to it at a later stage)

Q3. If you have noticed anything missing or incorrect in the "company" database, write an ALTER TABLE statement to correct the problem. Use the steps in the first two tasks of this lab to check that the database now matches the ERD in the lecture of Module 5.

Q4. In addition to the primary and foreign keys, two CHECK constraints and a UNIQUE constraint have been defined in the "employee" table. What names have they been given, what columns do they effect, and what do they all do/ensure/enforce?

Q5. Write a CREATE TABLE statement to create the table specified in below:

"project" table				
Column Name	**Data Type & Length**	**Null**	**Constraints**	**Other**
project_id	INT	NOT NULL	PK	IDENTITY
project_name	VARCHAR(50)	NOT NULL	UNIQUE	
project_desc	TEXT	NULL		
creation_date	SMALLDATETIME	NOT NULL		DEFAULT GETDATE()

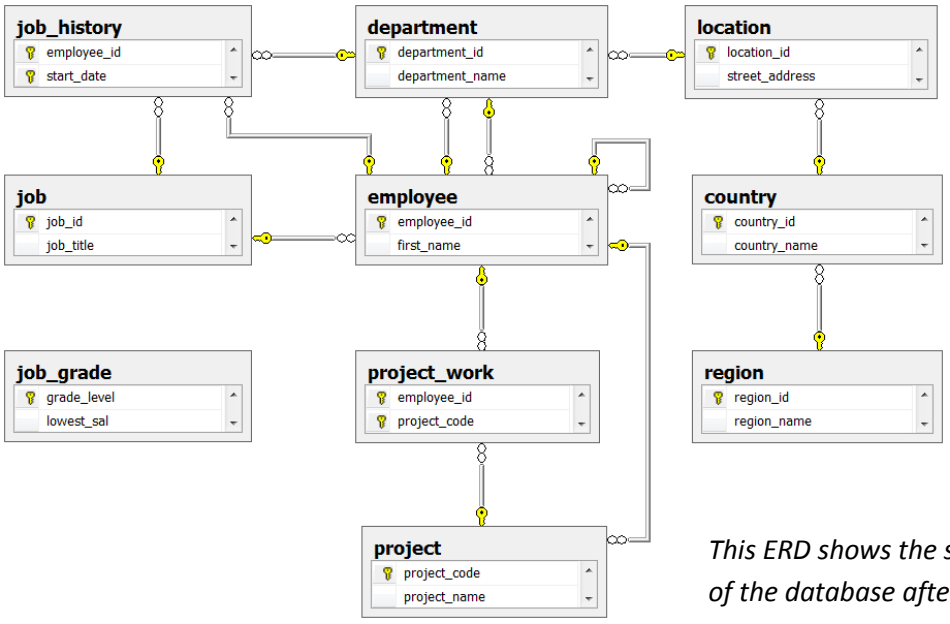
Q6. Alter the "project" table to add a "project_leader" column – INT, NOT NULL. Give it a foreign key constraint, referencing the "employee_id" column of the "employee" table.

Q7. Rather than using an auto-incrementing integer as the primary key, the company wishes to use a 5 letter project code. Alter the existing table to meet these new requirements. Since the "project_id" column has a primary key constraint, this involves several steps:

- An ALTER TABLE statement to *drop the primary key constraint*
- An ALTER TABLE statement to *drop the project_id column*
- An ALTER TABLE statement to *add the project_code column, CHAR(5), and give it a primary key constraint*

Q8. Rewrite the CREATE TABLE statement from task 5 of this lab so that it incorporates the changes made in tasks 6 and 7. You will need to drop the "project" table before you can create it again.

Q9. To record which employees are working in which project, create a "project_work" table with an employee_id column and a project_code column. Make sure the data types of the columns match the columns that they are referencing, and give them appropriate foreign key constraints. The primary key of the table should be a compound key using both of the columns. (This table is essentially an intermediary table resolving a M:M)



This ERD shows the structure of the database after task 9.

Challenge Query!

The following query is more difficult than the previous queries, and may involve SQL syntax which has not yet been covered in the unit.

Q10. Create a table named "item" with the following specifications:

- An "item_id" column, INT, with a primary key constraint. Using IDENTITY, make it auto-incrementing starting at 100 and incrementing by 20 each time.

- An "item_name" column, VARCHAR(50) and NOT NULL, with a unique constraint and a default value of "No name".

- An "item_desc" column, VARCHAR(250) and NOT NULL, with a CHECK constraint that ensures that the description is at least 40 characters long. Use the "LEN()" function to determine the length of the description.
    - For an example, see slide 21 of the Module 7 lecture.

- A "initial_stock" column and a "reorder_level" column, both SMALLINT and NOT NULL, with defaults of 100 and 25 respectively.
    - Include a CHECK constraint that ensures that the reorder level is less than the initial stock.

- A "created_by" column, INT and NULL, with a foreign key constraint referencing the item_id in the same table (i.e. a self-referencing relationship).

Once you've created the table and think all of the constraints are correct, test it by trying to insert some data into it. Inserting data is covered in Module 8, but here is an example of an insert statement you can modify and run to test your table:

```
INSERT INTO item
VALUES
(
  'This Is The Item Name',
  'This is an item description which is long enough.',
  75,
  15,
  NULL
);
```

Modify and run the INSERT statement as many times as you need to test all of your constraints. You can see the content of the table with "`SELECT * FROM item;`".