# Chapter 1- Creating Data Tables & Referential Integrity

## SQL Server Management Studio practical

## Objectives

The primary objective of this lab is to gain familiarisation with the SQL Server interactive interface and to be able to create some simple tables

## Reference material

This practical is based on material in the chapter.

## Overview

In this exercise you will use a script to create a database that you will spend some time investigating. You will answer some questions designed to make you think like a database developer. You will also add a new table to the database.

## Estimated duration

The estimated duration for this lab is 30 minutes.

## Completed solution

There is no completed solution to this lab.

## Step by step

Please follow the instructions below, reading CAREFULLY at all times as the questions have been thoughtfully worded.

### Exercise 1

Run the following syntax to CREATE a 'review' table:

CREATE TABLE review

(

emp\_no INTEGER NOT NULL,

review\_date DATETIME NOT NULL,

grade CHAR(1) NULL

)

Try running it a second time – what do you expect to happen?

### Exercise 2

Add a 4th column to the ‘review’ table called 'notes'; it is to be a character column of varying length, not to exceed 40 characters in length and is to allow NULLs.

ALTER TABLE review

ADD notes VARCHAR(40) NULL

Try running it again – what do you expect to happen?

### Exercise 3

Inspect the table structure within the system tables using the following SQL SERVER specific code:

sp\_help review

### Exercise 4

INSERT two rows into the table. The values are duplicated, but it will run quite successfully because the 'review' table has no primary key defined and therefore no uniqueness is being enforced.

INSERT INTO review

VALUES (23, '05-12-2006', 'A','He is happy')

INSERT INTO review

VALUES (23, '05-12-2006', 'A','He is still happy')

### Exercise 5

Display the data in the table:

SELECT \*

FROM review

You should get the following:

emp\_no review\_date grade notes

------- ------------------------ ------ ----------------

23 2006-05-12 00:00:00.000 A He is happy

23 2006-05-12 00:00:00.000 A He is still happy

Two reviews for employee 23 have been added. However, they have the same date – this may or may not be something that concerns us.

Also employee 23 does not exist in salesperson. Your database is lacking in 'referential integrity', because we never told the DBMS about any relationship between emp\_no of 'review' and emp\_no of 'salesperson'.

Let's do something about these issues now!

### Exercise 6

Drop the 'review' table:

DROP TABLE review

### Exercise 7

Now we will reCREATE 'review' table this time with an additional constraint.

Firstly, there is to be a composite Primary Key constraint defined on the 'emp\_no' and 'review\_date' columns. The constraint is a separate object in the database which can be 'dropped' independently of the table itself, so it is a good idea to give it a meaningful name like 'pk\_review'.

CONSTRAINT pk\_review   
PRIMARY KEY(emp\_no, review\_date)

Now run the following CREATE table syntax that puts it all together:

CREATE TABLE review

(

emp\_no INTEGER NOT NULL,

review\_date DATETIME NOT NULL,

grade CHAR(1) NULL,

notes VARCHAR(40) NULL,

CONSTRAINT pk\_review   
 PRIMARY KEY(emp\_no, review\_date)

)

### Exercise 8

Try to INSERT two rows into the table using exactly the same code as you used in step 4.

INSERT INTO review

VALUES (23, '05-12-2006', 'A','He is happy')

INSERT INTO review

VALUES (23, '05-12-2006', 'A','He is still happy')

This time the first INSERT will succeed but the second will fail because the 'review' table now has a primary key defined. The database is unhappy because the second insert was attempting to break a referential integrity rule.

The following INSERTS should work however, why?:

INSERT INTO review

VALUES (23, '06-06-2006', 'A','He is happy')

INSERT INTO review

VALUES (24, '05-12-2006', 'A','He is still happy')

They work because the primary key for the review table is a compound (composite) key made up of emp\_no and review\_date. The emp\_no’s can be the same as long as the review\_dates are different and vice versa.

### Exercise 9

Drop the 'review' table:

DROP TABLE review

### Exercise 10

Now we will reCREATE 'review' table this time with an additional constraint.

We are going to define a Foreign Key constraint about any relationship between emp\_no of 'review' and emp\_no of 'salesperson'.

The constraint is a separate object in the database which can be 'dropped' independently of the table itself, so it is a good idea to give it a meaningful name like 'fk\_review\_salesperson'.

CONSTRAINT fk\_review\_salesperson  
FOREIGN KEY(emp\_no)  
REFERENCES salesperson

Now run the following CREATE table syntax that puts it all together:

CREATE TABLE review

(

emp\_no INTEGER NOT NULL,

review\_date DATETIME NOT NULL,

grade CHAR(1) NULL,

notes VARCHAR(40) NULL,

CONSTRAINT pk\_review   
 PRIMARY KEY(emp\_no, review\_date),  
 CONSTRAINT fk\_review\_salesperson  
 FOREIGN KEY(emp\_no)  
 REFERENCES salesperson

)

### Exercise 11

Try to INSERT two rows into the table using exactly the same code as you used in step 4.

INSERT INTO review

VALUES (23, '05-12-2006', 'A','He is happy')

INSERT INTO review

VALUES (23, '05-12-2006', 'A','He is still happy')

They should now both fail as we do not have a salesperson with an emp\_no of 23.

However, if we use values of emp\_no from salesperson we should be able to insert new entries.

INSERT INTO review

VALUES (20, '06-06-2006', 'A','He is happy')

INSERT INTO review

VALUES (30, '05-12-2006', 'B','He is Ok')  
  
INSERT INTO review

VALUES (40, '06-01-2007', 'B','He is very happy')

### Exercise 12

Display the grade for each employee:

SELECT grade, emp\_no

FROM review  
ORDER BY grade

We will now look at what work SQL Server will do to execute this. Choose from the Query menu the option ‘Display estimated execution plan’ (or press Ctrl-L).

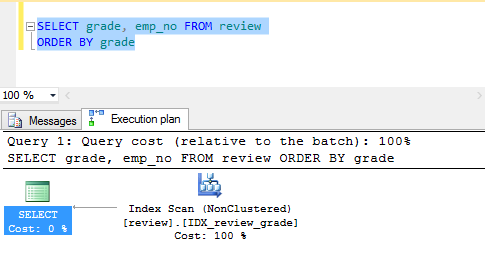
This is showing us that to complete the query the DBMS will use the clustered index pk\_review (our primary key) and then sort the output.

### C:\Users\Andrew\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\New Picture (1).bmpExercise 13

We will now create a new index as this query is pulling out the grade for each employee. We will give the index a name ‘IDX\_review\_grade’.

CREATE INDEX IDX\_review\_grade ON review(grade)

Now we will re-run our query and check the execution plan.



The query will now use the index we have created and will now not sort the data returned. This may (should?) run faster when retrieving data, however, as any row is inserted (or deleted) the index will need to be kept up to date.