Network Database Applications Assignment 1: Individual Activities

Gary Moore

November 16, 2017

1 Introduction

With the planning completed for the library database system, what remains to be implemented is the database itself. The database is to be implemented using Transactional Structured Query Language which will require a database server, and database management software for writing queries and executing them on the server.

2 Requirements

The following software will be used:

- MSSQL Server 2017 for Docker
- SQLPro management software for macOS

I'll be using the Docker image of SQL Server as it can be deployed on any operating system that can run a virtual machine. It also runs from within a sandboxed container, which makes it measurably more secure than running it on the host machine. I personally needed to use a Docker image as SQL Server is not software that can run natively on macOS.

Docker will likely only be used for the development of the database. When moved to a production environment, the database will likely be exported and run natively on a Windows server.

Typically, SQL Management Studio by Microsoft is used to manage T-SQL databases, but due to the software only being available on one platform I used SQLPro, which has a layout that matches very closely to the Microsoft software.

3 Tables

3.1 tblContact

The first table will be tblContact as tblCustomer and tblEmployee are dependent upon it.

```
CREATE TABLE tblContact
1
2
   (
3
           cnContactID INT IDENTITY (1,1) PRIMARY KEY,
           cnFirstName VARCHAR(30) NOT NULL,
4
5
           cnSurname VARCHAR(30) NOT NULL,
6
           cnPhone PhoneNumber,
7
           cnEmail Email,
           cnAddress1 VARCHAR(30) NOT NULL,
8
9
           cnCity VARCHAR(30) NOT NULL,
10
           cnCounty VARCHAR(30) NOT NULL,
11
           cnPostcode Postcode
12
   )
13
  INSERT INTO tblContact
14
   (cnFirstName, cnSurname, cnPhone, cnEmail, cnAddress1, cnCity, cnCounty,
        \hookrightarrow cnPostcode)
16 VALUES
   ('Gary', 'Moore', '07500564955', 'madmangaz@gmail.com', '49 Woodvale Park', '
        ('Steven', 'Stove', '02887723114', 'stovesteven@gmail.com', '70 Stove Road', '
        \hookrightarrow Stoveland', 'Co Stove', 'BT115DF'),
   ('Dolly', 'Donko', '07545652412', 'donkodolly@gmail.com', '32 Frillo Road', '
        \hookrightarrow Dollyland', 'Co Donko', 'BT323FG'),
   ('Willy', 'Wonko', '07548895621', 'wonkowilly@gmail.com', '49 Wonko Close', '
20
        \hookrightarrow Wonkoworld', 'Co Willy', 'BT63BN'),
   ('Henry', 'Hob', '07502314592', 'hobhenry@gmail.com', '21 Hob Lane', 'Henrytown
21
        \hookrightarrow ', 'Co Hob', 'BT125NA'),
   ('Nobby', 'Noodle', '07504562312', 'noodlenobby@gmail.com', '21 Noodle Way', '
        \hookrightarrow Nobbycity', 'Co Noodle', 'BT426FN'),
   ('Willy', 'Willard', '07501228543', 'willardwilly@gmail.com', '21 Garfield Lane
        \hookrightarrow ', 'Grungetown', 'Co Tables', 'BT124DB'),
   ('Shellie', 'Shirt', '07542655489', 'shirtshellie@gmail.com', '141 Dingdong
        \hookrightarrow Avenue', 'Dingdongcity', 'Co Dole', 'BT126DF'),
    ('Rupert', 'Rung', '02845621453', 'rungrupert@gmail.com', '12 Rung Street', '
        ('Berty', 'Balls', '07512545698', 'ballsberty@gmail.com', '5 Bumble Road', '
        \hookrightarrow Bundo', 'Co Herk', 'BT436BN'),
   ('Jingo', 'Jango', '07512459821', 'jangojingo@gmail.com', '9 Honk Road', 'Linda
        \hookrightarrow City', 'Co Suit', 'BT235HN'),
   ('Will', 'Wharg', '07512324582', 'whargwill@gmail.com', '99 Nine Lane', 'Niner
28
        \hookrightarrow Valley', 'Co Number', 'BT654BA')
29
30 SELECT * FROM tblContact
```

The CREATE TABLE command will create the table tblCustomer, with columns then entered in the bracket delimiters. The first common is an INT, which is shorthand for integer. It is an auto incrementing number that is also the primary key, with IDENTITY (1,1) meaning it increments by one with each new field and PRIMARY KEY denoting that this is a fields identifying column. Each primary key is automatically enforced with constraints to ensure that a null value isn't used, and that each field must have a unique primary key.

Almost every key after the primary key uses the VARCHAR datatype, which can contain a string, and is of variable length. A column of this data type that's notable is cnPostcode. A post code is typically considered to be fixed length, however there's a fairly common edge case of a 6-character long post code, for example BT7 5NA. This is why VARCHAR is used instead of CHAR. Another notable aspect of these columns is most of them use the constraint NOT NULL, which means that a null value cannot be entered into this column.

CHAR of length 11 is used for the phone number as an integer isn't appropriate here because we don't want to perform any calculations on a phone number, and it is possible to enter a value in excess of the 32bit integer limit. CHAR is used instead of VARCHAR because all UK phone numbers are of length 11.

To insert fields into this table we first use the INSERT INTO command followed by the table name tblContact. Next we declare what columns we wish to input information into in delimiters separated by commas. On a new line you write the VALUES keyword followed by delimited entries on a new line with strings surrounded by single quotes and numbers entered without quotes separated with a comma.

The command SELECT * FROM tblContact will display all fields in the table.

Result

	cnContactID	cnFirstName	cnSumame	cnPhone	cnEmail	cnAddress1	cnCity	cnCounty	cnPostcode
1	1	Gary	Moore	07500564955	madmangaz@gmail.com	49 Woodvale Park	Dungannon	Co Tyrone	BT716DB
2	2	Steven	Stove	02887723114	stovesteven@gmail.com	70 Stove Road	Stoveland	Co Stove	BT115DF
3	3	Dolly	Donko	07545652412	donkodolly@gmail.com	32 Frillo Road	Dollyland	Co Donko	BT323FG
4	4	Willy	Wonko	07548895621	wonkowilly@gmail.com	49 Wonko Close	Wonkoworld	Co Willy	BT63BN
5	5	Henry	Hob	07502314592	hobhenry@gmail.com	21 Hob Lane	Henrytown	Co Hob	BT125NA
6	6	Nobby	Noodle	07504562312	noodlenobby@gmail.com	21 Noodle Way	Nobbycity	Co Noodle	BT426FN
7	7	Willy	Willard	07501228543	willardwilly@gmail.com	21 Garfield Lane	Grungetown	Co Tables	BT124DB
8	8	Shellie	Shirt	07542655489	shirtshellie@gmail.com	141 Dingdong Avenue	Dingdongcity	Co Dole	BT126DF
9	9	Rupert	Rung	02845621453	rungrupert@gmail.com	12 Rung Street	Rungland	Co Ring	BT47FG
10	10	Berty	Balls	07512545698	ballsberty@gmail.com	5 Bumble Road	Bundo	Co Herk	BT436BN
11	11	Jingo	Jango	07512459821	jangojingo@gmail.com	9 Honk Road	Linda City	Co Suit	BT235HN
12	12	Will	Wharg	07512324582	whargwill@gmail.com	99 Nine Lane	Niner Valley	Co Number	BT654BA

3.2 tblCustomer

This is a small table as most of the details are contained in tblContact. The two tables use a one to many relationship avoiding redundant data.

```
1 CREATE TABLE tblCustomer
2 (
3 cuCustomerID INT IDENTITY (1,1) PRIMARY KEY,
4 cnContactID INT NOT NULL FOREIGN KEY REFERENCES tblContact(cnContactID)
5 )
```

```
6
7
   INSERT INTO tblCustomer
8
   (cnContactID)
9 VALUES
10 (8),
11 (9),
12
  (10),
13
  (11),
14
   (12)
15
16 SELECT * FROM tblCustomer
```

This table is the first case where a foreign key is used, and a different syntax is used to create a foreign key compared to a primary key. Instead of declaring it with PRIMARY KEY, we write FOREIGN KEY REFERENCES table (column). The purpose of a foreign key is to link this table with another table that it references data from.

We must associate each customer with its corresponding entry in tblContact, and to do this we insert the corresponding foreign key value into tblCustomer.

Result

	cuCustomerID	cnContactID
1	1	8
2	2	9
3	3	10
4	4	11
5	5	12

3.3 tblEmployee

Similarly to tblCustomer we must initialise a foreign key, however there are extra fields for this table with employee specific information.

```
1
   CREATE TABLE tblEmployee
2
   (
3
           emEmployeeID INT identity(1,1) PRIMARY KEY,
4
           emPassword VARCHAR(30) NOT NULL,
5
           emPosition VARCHAR(20) NOT NULL,
6
           emPermissions VARCHAR(10) NOT NULL,
7
           cnContactID INT NOT NULL FOREIGN KEY REFERENCES tblContact(cnContactID)
8
   )
9
10
   INSERT INTO tblEmployee
11
   (emPassword, emPosition, emPermissions, cnContactID)
12 VALUES
13 ('password1', 'andy.admin', 'root', 1),
14 ('password2', 'billy.bob', 'user', 2),
15 ('password3', 'willy.wonka', 'user', 3),
16 ('password4', 'robert.rub', 'user', 4),
```

```
17 ('password5', 'henry.hob', 'user', 5),
18 ('password6', 'willard.wong', 'user', 6),
19 ('password7', 'guest', 'restricted', 7)
20
21 SELECT * FROM tblEmployee
```

Every employee will need a password to get access to the database system and appropriate permissions that will define what that employee can change in the database. For example an administrator will be able to drop and add tables, where as a user may only be able to alter or add data to the database.

Result

	emEmployeeID	emPassword	emPosition	emPermissions	cnContactID
1	1	password1	andy.admin	root	1
2	2	password2	billy.bob	user	2
3	3	password3	willy.wonka	user	3
4	4	password4	robert.rub	user	4
5	5	password5	henry.hob	user	5
6	6	password6	willard.wong	user	6
7	7	password7	guest	restricted	7

3.4 tblAuthor

This table will have a many to many relationship with tblBook. A junction table will be used between the two tables, more information will be in section 3.6.

```
1
   CREATE TABLE tblAuthor (
2
       auAuthorID INT IDENTITY(1,1) PRIMARY KEY,
3
       auFirstName VARCHAR(30) NOT NULL,
       auSurname VARCHAR(30) NOT NULL
4
5
   )
6
   INSERT INTO tblAuthor
7
8
   (auFirstName, auSurname)
9
   VALUES
10
   ('Arty', 'Author'),
   ('Steven', 'Writer'),
11
   ('Henry', 'Scribe'),
12
   ('Barry', 'Book'),
   ('Terry', 'Tree'),
14
   ('Sabrina', 'Scroll')
15
16
   SELECT * FROM tblAuthor
```

For any new books that are added to the database system, the author names will go into a second document to avoid redundant data, as authors can write more than one book.

Result

	auAuthorID	auFirstName	auSumame
1	1	Arty	Author
2	2	Steven	Writer
3	3	Henry	Scribe
4	4	Barry	Book
5	5	Terry	Tree
6	6	Sabrina	Scroll

3.5 tblBook

This table has a many to many relationship with tblAuthor.

```
1
   CREATE TABLE tblBook (
2
       bkISBN VARCHAR (13) PRIMARY KEY NOT NULL,
3
       bkTitle VARCHAR (50) NOT NULL,
4
       bkGenre VARCHAR (30) NOT NULL
5
   )
6
7
   INSERT INTO tblBook
8
   (bkISBN, bkTitle, bkGenre)
9 VALUES
10 (1235785642842, 'The Big Thing', 'SciFi'),
   (7512354894217, 'Blast Off from the Crescent Moon', 'Farty Tale'),
12 (4216548789512, 'The Big One', 'Romance'),
13 (4865423157, 'Back in my Day', 'Memoir'),
  (4562147895444, 'Dingo the Dog', 'Childrens'),
15 (7825123786215, 'Dirty Dan', 'Spaghetti Western')
16
  SELECT * FROM tblBook
17
```

A unique aspect of this table; it uses the ISBN number of the book as its primary key. Because the number is so big, we use VARCHAR instead of the INT data type, which can be used as the primary key so long as it's unique. We're using a variable data type because an ISBN can be 10 or 13 digits long.

We also create columns for the title and genre of the book and use INSERT INTO to fill out some values.

Result

	bkISBN	bkTitle	bkGenre
1	1235785642842	The Big Thing	SciFi
2	4216548789512	The Big One	Romance
3	4562147895444	Dingo the Dog	Childrens
4	4865423157	Back in my Day	Memoir
5	7512354894217	Blast Off from the Crescent Moon	Farty Tale
6	7825123786215	Dirty Dan	Spaghetti Western

3.6 tblBookAuthor

This is a junction table between tblBook and tblAuthor. The purpose of this table is to create a link between these two tables, as a many to many relationship cannot exist directly between these two tables.

```
CREATE TABLE tblBookAuthor (
1
2
       auAuthorID INT NOT NULL FOREIGN KEY REFERENCES tblAuthor(auAuthorID),
3
       bkISBN VARCHAR(13) NOT NULL FOREIGN KEY REFERENCES tblBook(bkISBN),
4
       PRIMARY KEY (auAuthorID, bkISBN)
5
   )
6
7
   INSERT INTO tblBookAuthor
8
   (auAuthorID, bkISBN)
9
   VALUES
10
   (1, '1235785642842'),
   (2, '4216548789512'),
11
   (3, '4562147895444'),
12
   (4, '4865423157'),
13
   (5, '7512354894217'),
15
   (6, '7825123786215')
16
   SELECT * FROM tblBookAuthor
17
```

This is the first instance where a composite key is created. To create one of these, we create two foreign keys from the tables tblBook and tblAuthor and then join these together using PRIMARY KEY (auAuthorID, bkISBN). This joins the two together to create a unique key.

Result

	auAuthorID	bkISBN
1	1	1235785642842
2	2	4216548789512
3	3	4562147895444
4	4	4865423157
5	5	7512354894217
6	6	7825123786215

3.7 tblLeasing

This table establishes three relationships, one with tblCustomer, tblBookLeasing and a junction table.

```
1 CREATE TABLE tblLeasing (
2 lsLeasingID INT IDENTITY (1,1) PRIMARY KEY,
3 cuCustomerID INT NOT NULL FOREIGN KEY REFERENCES tblCustomer(cuCustomerID),
4 emEmployeeID INT NOT NULL FOREIGN KEY REFERENCES tblEmployee(emEmployeeID)
```

```
5
  )
7
  INSERT INTO tblLeasing
   (cuCustomerID, emEmployeeID)
8
9 VALUES
10 (1, 4),
   (2, 4),
11
12
   (3, 2),
13
   (4, 3),
14
   (5, 5)
15
   SELECT * FROM tblLeasing
16
```

This table has some relatively complex transactions going on. To begin with, it is interacting with transactional and non-transactional tables, with transactions happening in a junction table.

Result

	IsLeasingID	cuCustomerID	emEmployeeID
1	1	1	4
2	2	2	4
3	3	3	2
4	4	4	3
5	5	5	5

3.8 tblBookLeasing

This is a transactional table that is a junction between tblBook and tblLeasing. It is responsible for documenting the checkout and return date

```
CREATE TABLE tblBookLeasing (
1
2
       bkISBN VARCHAR (13) NOT NULL FOREIGN KEY REFERENCES tblBook(bkISBN),
3
       lsLeasingID INT NOT NULL FOREIGN KEY REFERENCES tblLeasing(lsLeasingID),
4
       lsLeaseDate DATE NOT NULL,
       lsReturnDate DATE NOT NULL,
5
6
       PRIMARY KEY (lsLeasingID, bkISBN)
7
   )
8
9
  INSERT INTO tblBookLeasing
10
   (bkISBN, lsLeasingID, lsLeaseDate, lsReturnDate)
   VALUES
11
   ('1235785642842', 1, '2018-04-01', '2018-04-08'),
12
   ('7512354894217', 2, '2018-04-02', '2018-04-09'),
   ('4865423157', 3, '2018-04-03', '2018-04-10'),
   ('4865423157', 4, '2018-04-04', '2018-04-11'),
15
   ('7825123786215', 5, '2018-04-05', '2018-04-12')
16
17
18 SELECT * FROM tblBookLeasing
```

This table makes use of a composite key, the creation of which is described in 3.6. tblBookLeasing is also the first use of the DATE data type, which is of course used for storing date information. When a customer decides to check out a book, the transaction will eventually make its way to this table, where the lease and return date are stored in lsLeaseDate and lsReturnDate.

Result

	bkISBN	IsLeasingID	IsLeaseDate	IsRetumDate
1	1235785642842	1	2018-04-01	2018-04-08
2	7512354894217	2	2018-04-02	2018-04-09
3	4865423157	3	2018-04-03	2018-04-10
4	4865423157	4	2018-04-04	2018-04-11
5	7825123786215	5	2018-04-05	2018-04-12

3.9 Notable Transactions

Adding a book

When an employee wants to add a book to the database, they will use a form that will inner join tblAuthor, tblBook and tblBookAuthor. This data is inserted into the table through this form.

Removing a book

The process or removing a book involves some conditional logic. If the book is the only one in the system by that author, the author can be removed too. If the book has other books by the same author, the author will not be deleted. This can be done with an IF...ELSE statement.

Leasing a book

The procedure for checking out a book uses a network of tables to carry out the transaction, again to make sure there's no data redundancy. The tables tblCustomer, tblLEasing, tblEmployee, tblBook and tblBookLeasing will be used to create a record of a lease with tblContact queried to get the customer number from any of the customers details.

4 Constraints

I will make use of constraints to ensure that only valid data is entered into the database. These constraints can be created at CREATE TABLE, or they can be appended to the column definitions using ALTER TABLE.

A constraint is a system in a database that applies limitations to what information can be input into a particular column in a table. An example of a constraint is NOT NULL, which doesn't allow the user to enter a null value into their table. Another example is UNIQUE, which only allows a unique value compared to other values entered into the same column to exist. Violation of a constraint will result in an error which is displayed in the SQL management software's console.

I've created constraints for tblContact; a constraint for email, postcodes and phone numbers.

To add these constraints we need to alter the tblContact table and then use the ADD command followed by the CONSTRAINT keyword on a new line with the name that we'll use for the constraint. We use CHECK to make sure that input values abide by the defined rule in the parenthesis. A boolean value of true is generated if the value matches the pattern, allowing the database to add that value to the table.

4.1 Phone constraint

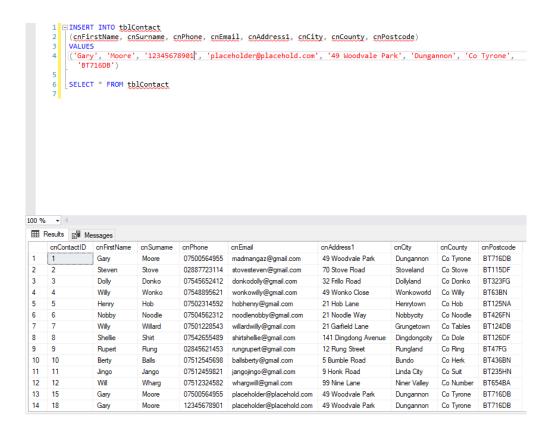
Check line 3 of the example in Chapter 4. The requirements for the phone constraint are quite simple. Only digits can be accepted, therefore I've used the [0-9] wildcard to denote only digits. I've used this 11 times as the wildcard only represents a single character.

Testing

In the first image I'm checking to see if the constraint works when I try to add a phone number with less than 11 digits. The constraint successfully prevents this from happening, throwing an error.

In the second image, I am trying to use letters as a phone number, however the constraint catches this, and throws out an error.

In the third image, I pass a valid phone number to test if the constraint will allow valid data to be inserted into the table. The constraint passes the test.



4.2 Email constraint

Check line 4 of the example in Chapter 4. This check is slightly more complicated as i'm using the AND statement, which means values must pass two checks in order to be inserted into the table. First it will check if the correct email formatting is used, that is the requirement of &, period, username, domain name and tld. I don't want an email to look like '&.com', and the % wildcard will allow any string with no characters up to exist there, so I have to do a second check that requires the length of the value to be greater than 6, which is the minimum for there to be information in the username and domain name. This isn't perfect however, as it could still look like this 'aa@.com'. This constraint should however weed out the most troublesome data validation errors.

Another way to do data validation that I haven't looked at is to use a Regular Expression, however this requires integrating a CLR¹ function into the database solution.

Testing

In the first image I am testing to see if validation works when the input is not long enough to pass. An error message is displayed and the data is not inserted into the table as it did not pass the test.

¹Common Language Runtime

In the second image I am testing to see if validation works when I only input a username without the , period, tld or domain name. The data fails the test and isn't inserted into the table.

In the third image, I am testing valid data to ensure that it passes the test. It does, and the data is inserted into the table.

```
(cnFirstName, cnSurname, cnPhone, cnEmail, cnAddress1, cnCity, cnCounty, cnPostcode)
                   ', 'Moore', '07500564955', 'placeholder@placehold.com', '49 Woodvale Park', 'Dungannon', 'Co Tyrone',
           SELECT * FROM tblContact
100 % -
 Results Messages
      cnContactID
                                                                                                         cnCity
                                                                                                                       cnCounty
                                             07500564955
                                                                                     49 Woodvale Park
                                                                                                                       Co Tyrone
                                                                                                                                   BT716DB
                   Gary
                                 Moore
                                                          madmangaz@gmail.com
                                                                                                          Dungannon
                    Steven
                                 Stove
                                             02887723114
                                                           stovesteven@gmail.com
                                                                                     70 Stove Road
                                                                                                                       Co Stove
                                                                                                                                   BT115DF
                                             07545652412
                                                                                    32 Frillo Road
                    Dolly
                                 Donko
                                                          donkodolly@gmail.com
                                                                                                          Dollyland
                                                                                                                       Co Donko
                                             07548895621
                    Willy
                                 Wonko
                                                           wonkowilly@gmail.com
                                                                                     49 Wonko Close
                                                                                                          Wonkoworld
                                                                                                                       Co Willy
                                                                                                                                   BT63BN
                    Henry
                                 Hob
                                             07502314592
                                                          hobhenry@gmail.com
                                                                                    21 Hob Lane
                                                                                                          Henrytown
                                                                                                                       Co Hob
                                 Noodle
                                             07504562312
                                                                                    21 Noodle Way
                                                                                                          Nobbycity
                                                                                                                       Co Noodle
                                                                                                                                   BT426FN
                    Nobby
                                                          noodlenobby@gmail.com
                                 Willard
                                             07501228543
                                                          willardwilly@gmail.com
                                                                                    21 Garfield Lane
                                                                                                                       Co Tables
                                                                                                                                   BT124DB
                    Willy
                                                                                                          Grungetown
      8
                    Shellie
                                 Shirt
                                             07542655489
                                                          shirtshellie@gmail.com
                                                                                    141 Dingdong Avenue
                                                                                                          Dingdongcity
                                                                                                                       Co Dole
                                                                                                                                   BT126DF
                                             02845621453 rungrupert@gmail.com
                                                                                    12 Rung Street
                                                                                                          Rungland
                                                                                                                       Co Ring
                                                                                                                                   BT47FG
                    Rupert
                                 Rung
 10
      10
                                             07512545698
                                                                                    5 Bumble Road
                                                                                                                                   BT436BN
                                                          ballsberty@gmail.com
                                                                                                          Bundo
                                                                                                                       Co Herk
                    Berty
                                 Balls
                                                                                    9 Honk Road
                                                                                                                                   BT235HN
 11
      11
                                            07512459821 jangojingo@gmail.com
                                                                                                                       Co Suit
                                 Jango
                                                                                                          Linda City
                    Jingo
 12
                                            07512324582
                                                                                                                                   BT654BA
      12
                    Will
                                                                                    99 Nine Lane
                                                                                                          Niner Valley
                                 Wharg
                                                           whargwill@gmail.com
                                                                                                                       Co Number
 13
                                            07500564955 placeholder@placehold.com
      15
                                                                                    49 Woodvale Park
                                                                                                                                   BT716DB
                   Gary
                                 Moore
                                                                                                         Dungannon
                                                                                                                       Co Tyrone
```

4.3 Postcode constraint

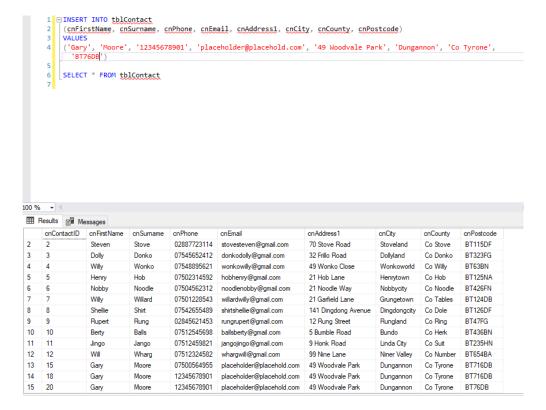
Check line 5 of the example in Chapter 4. A postcode can be 6 or 7 characters long, and the typical format is [Char, Char, Num, Num, Num, Char, Char]. The constraint needs to ensure that this formatting is used, and also that it is of length 6 or 7.

In order to do this I used the wildcards [A-Z] and [0-9] where appropriate. I also used an OR statement that results in true if either one of the conditional statements is true.

Testing

In the first image I am testing to ensure that the constraint rejects data that is too short to be a valid postcode. The constraint works as intended, and an error message is displayed with the data not being inserted into the table.

In the second image I am testing to ensure that a 6 character postcode get successfully inserted into the table. The test shows that the validation passed, and the data goes into the table.



5 Locking

Without locking, users may experience one more issues when interacting with a database system. These are called concurrency side effects, and can range from inconvenient, to destructive.

This is an issue that occurs when two users try to alter the same value, at the same time. What can happen is both users will receive a message confirming a successful edit, however one users data will be overwritten when they read it back again.

This is avoided by introducing locks to the query that prevents two transactions from happening at the same time by making transactions wait until the current one in progress has finished. Users will have to wait until a transaction has completed until they can update the value.

Lock types

Shared lock This is for all intents and purposes a read only locking mechanism. It allows for concurrent access but that data is immutable.

Exclusive lock This only allows one user/transaction to modify a value at any one time. No other type of lock can be placed on an exclusive lock.

Update lock This lock will only effect the row that a transaction is modifying values on. When the user runs a transaction modifying information in a row, that row is locked, and no other transactions can modify any values on this row. This is similar to an exclusive lock, however the difference is a shared lock can still read data from a row that is seeing a transaction with an update lock concurrently.

Examples

An example of locking is shown in section 5.1.

5.1 Rollback

The ROLLBACK function is a very useful feature that protects against accidental data loss by creating save points that you can restore to in a transaction.

For example, in a business your boss may ask you to delete client information in a table that contains 300 records. Without the use of ROLLBACK, all this information will be lost, and your boss will ask you to spend your weekend re-entering this information from hard copies.

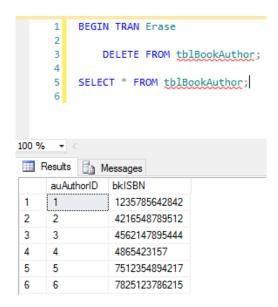
If you used the SAVE TRANSACTION command, you can simply use the command ROLLBACK to almost instantly restore all the data to its prior state, and you can spend the weekend at home with your feet up.

Examples

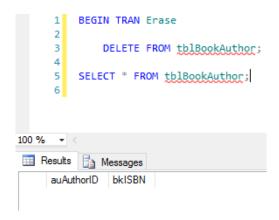
Here is the code we are going to execute.

```
1 BEGIN TRAN Erase
2 
3 DELETE FROM tblBookAuthor;
4 
5 SELECT * FROM tblBookAuthor;
```

And here is our table before the code is executed.



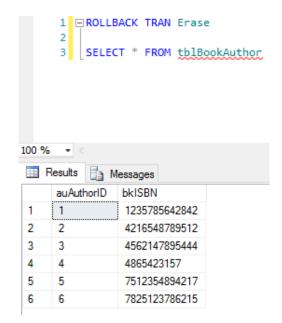
Now I'll delete the data.



I'll restore the data using ROLLBACK TRAN.

1 ROLLBACK TRAN Erase

This image shows that the restore operation was successful.



Feedback to the user can be integrated by using the PRINT statement after the appropriate command is executed. This function is used like this; PRINT 'string'.

You can also include multiple restore points by using the following structure:

```
BEGIN TRAN Delete_Tables
1
2
      SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
3
      SAVE TRAN SP1;
4
          DELETE FROM tblEmployee;
5
      SAVE TRAN SP2;
6
          DELETE FROM tblAuthor;
7
      SAVE TRAN SP3;
8
          DELETE FROM tblBook;
      PRINT 'Records successfully deleted.';
```

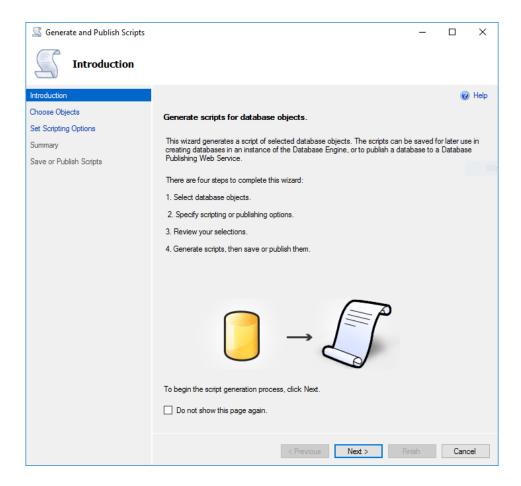
You can then roll back to any of these save points, for example ROLLBACK TRAN SP2 will recover the data right up to the point before deleting tblAuthor.

6 Backup

A good data backup system is essential to avoid wasted time and money. Backups ensure that there is a snapshot to restore from should something catastrophic happen to the data stored in the database system.

Databases are at the backbone of most good web services and websites, so if you don't backup your database systems at all, a failure could mean that your business is wiped out.

The process of backing up in Microsoft SQL Management Studio is fairly simple. To backup data you go to Tasks → Generate Scripts. You will see the following dialogue:



Continue through this dialogue making sure to change 'Schema Only' to 'Schema and Data'. You will eventually get the option to save a script that will recreate the database on a fresh system.

7 Evaluation

Requirements Just to remind myself, the purpose of this database is to process transactions for a library. The customer transaction must be able to take a book, and put it though the system creating a reciept for the customer who leased the book with a return date, book title as well as what librarian served the customer that day.

The second transaction that occurs is when the logged in librarian adds or removes a book, with conditional logic necessary for this as a many to many relationship is used, and the database system must be able to intelligently know when to delete the author or not depending on if the author has other published works in the library.

Effectiveness I think this database system meets the standards of a small library pretty well, however I am not sure if this solution will be able to scale very well. Normalisation was carried out appropriate, creating 8 tables in total with relationships established

through foreign keys, allowing for the use of inner joins in queries. User defined data types were used in order to make use of constraints which aids in data validation. Problem columns such as postcodes, phone numbers and email addresses validate properly, however there remains small problems with the email constraint that could be ironed out with a more advanced regular expression function. More extensive use of user defined data types would have also been helpful for the sake of reusable constraints, but this is not essential.