SQL: Triggers and stored Procedures

Martin Read

Triggers

- Special routines that run when something happens
  - Triggered automatically
- Very useful for reacting to insert or delete events
  - E.g. Trigger an update to the quantity of a product after a sale happens
  - If a quantity of a product falls below a certain amount then alerting to reorder that product should happen
- Could be done manually, but let automation do the heavy lifting

Triggers

- Activated before/after a row is updated, deleted or inserted
- Is associated with a specific database table
- Database tables can have more than one trigger
  - But think about the sequence
- Trigger executed within the transaction that triggered it
- Triggers cannot have parameters (unlike stored procedures)
- Used for audit trails

Trigger example

- Reduce quantity of product after sales

```
CREATE TRIGGER ChangeStock ON OrderDetails
AFTER INSERT
AS
BEGIN
  IF UPDATE (Quantity)
  BEGIN
        UPDATE Products
        SET Products.Quantity = Products.Quantity -
        inserted.Quantity
        FROM inserted
        WHERE Products.ProductId = inserted.ProductId
  END
END;
```

Stored Procedures

- A SQL script that is saved in the Database
  - has a name
- Encapsulates specific actions required by the application
- Can reduce network traffic and so increase the performance
- Provides security – use parameters for input
- Decreases code duplication by allowing code sharing

Stored Procedure Syntax

- The stored procedure itself is called from the interface you would be creating
- For now we use EXEC

```
CREATE PROCEDURE ProductList AS
BEGIN
   SELECT ProductId, Product_Details, Price,
Quantity
   FROM Products
   ORDER BY ProductID
END;
```

```
EXEC ProductList
```

Sub-queries

- What if..

```
CREATE PROCEDURE ProductList AS
BEGIN
  SELECT p.ProductId, p.Product_Details, p.Price,
 p.Quantity,
     (SELECT sum(od.quantity) FROM OrderDetails
 od
   WHERE p.ProductId = od.ProductId
   Group By od.ProductId) as Sales
  FROM Products as p
  ORDER BY ProductID
END;
```

- sales for each item - calculate the values

Parameters

- Stored procedures take values in and can output values
  - Parameters
  - Indicated with @ symbol
- Parameters must be surrounded with opening and closing brackets
- They also have the data type specified
- Executing the procedure requires passing the parameter to it

Example

- Parameters are ‘bound’ to the procedure in the interface
- For now we call it like this

```
CREATE PROCEDURE FindProduct(@ProductId as INT) AS
BEGIN
  SELECT Product_Details, Price, Quantity
  FROM Products
  WHERE ProductId = @ProductId
END;
```

```
EXEC FindProduct 2
```

Multiple Parameters

- Can have more than one
- List them
- When calling them,
  - use Named parameters

```
CREATE PROCEDURE FindProductsInPriceRa
nge(
    @TopRangePrice as FLOAT,
    @BottomRangePrice AS FLOAT
    ) AS
BEGIN
  SELECT Product_Details, Price, Quantity
  FROM Products
  WHERE Price >= @BottomRangePrice
  AND Price <= @TopRangePrice
  ORDER BY Price
END;
```

```
EXEC FindProductsInPriceRange @TopRangePrice = 0.4,
 @BottomRangePrice = 0.1
```


University of Plymouth
Advancing knowledge, transforming lives