1 Stored Procedures in T-SQL

Stored Procedures in any database work mostly the same—and in most cases, look and 'feel' the same. For example, when writing stored procedures in T-SQL, we're doing nearly identical things as in PL/SQL, with slight differences. What this means is that whatever you learn from one database is—with slight syntactical modifications—is transferable to another database.

2 T-SQL

Transact-SQL is the SQL flavor used by Microsoft SQL Server. As far as basic SQL goes, it is similar to Oracle, etc., as far as stored procedures are concerned, it is slightly different, but still similar. For example, a stored procedure to select all "Person" records may look like:

```
CREATE PROCEDURE GetAllPeople
AS
    SELECT *
    FROM Person
```

Just like in PL/SQL, you can group many statements with a `BEGIN` and `END` keywords—for situations where you want to do more than one statement.

2.1 Parameters

Parameters are passed to stored procedures by listing them right after the procedure name. For example, a stored procedure that looks for a person by their first name and last name, might look like this:

```
CREATE PROCEDURE [FindPerson] ( @qstr VARCHAR(32) )
AS
    SELECT *
    FROM [Person]
    WHERE fname like '%'+@qstr+'%' OR lname like '%'+@qstr+'%'
```

Obviously you can pass more than one parameter and name them anything you like. For example, a procedure to add a person might look like this:

```
CREATE PROCEDURE [CreatePerson]
    ( @fname VARCHAR(32), @lname VARCHAR(32), @email VARCHAR(64) )
AS
INSERT INTO [Person] ( [fname], [lname], [email] )
    VALUES ( @fname, @lname, @email )
RETURN @@IDENTITY
```

2.2 Quoting

There are many reserved words in T-SQL (or at least in Microsoft SQL Server), so the need to escape or quote strings becomes important. It is usually a good idea to quote table and column names, as well as procedure names and parameters. You quote things by putting it in the `[` and `]` brackets. For example, to create a person table, you'd do something like:

```
CREATE TABLE [Person] (
    [id] INT IDENTITY(1,1),
    [fname] VARCHAR(32),
    [lname] VARCHAR(32),
    [email] VARCHAR(64),
    PRIMARY KEY([id])
);
```

Obviously, the same applies to stored procedures.

2.3 Flow Control

T-SQL has the obvious flow control structures, which are only slightly different from the ones in PL/SQL. For example, and `IF` statement is just:

```
IF condition
    statement
```

Where condition is some logical condition, possibly involving variables or queries, and statement is some T-SQL, possibly involving `BEGIN` and `END` to include a whole block of statements. For example:

```
IF (SELECT COUNT(*) FROM [Person] WHERE [fname]='John') > 5
    PRINT 'There are more than 5 Johns in the database.'
```

There is also the usual `WHILE` loop:

```
WHILE (SELECT AVG(salary) FROM [Employee]) < 40000
    UPDATE [Employee] SET salary=salary * 0.05;
```

Which would raise everyone's salary by 5% until the average is over $40k. Again, you can use the `BEGIN` and `END` to use more than one statement, etc.

There are many other fancier things you can do, but I'll let you discover them on your own. The Microsoft SQL Server documentation is a great tool in figuring out how to do something.