The Benefits of Stored Procedures

There are a number of ways to access data in SQL Server, or any enterprise DBMS. There are lots of books that discuss getting data in and out of databases and the best ways to do that. Many advocate the use of stored procedures to ensure the safety of the data.

The tree main benefits that I see on stored procedures are:

- Abstraction
- Security
- Performance

Stored Procedures add an extra layer of abstraction in to the design of a software system. This means that, so long as the interface on the stored procedure stays the same, then the underlying table structure can change with no noticable consequence to the application that is using the database.

For instance, if the database has to be denormalised to get a little extra performance in certain situations then the stored procedures can handle the additional updates and inserts necessary to ensure the integrity of the data across the tables. Without this the each of the callers would have ensure that these changes had taken place. Of course, the use of stored procedures does not in anyway grant a waiver from properly designing the data model, but it can help if the perfect normalised model has to give way for performance improvements.

This layer of abstraction also helps put up an extra barrier to would be intruders. If access to the data in SQL Server is only ever permitted via stored procedures then permission does not need to be explicitly set on any of the tables. Therefore none of the tables should ever need to be exposed directly to outside applications. For an outside application to modify the database, it must go through stored procedures.

Stored procedures can be written to validate any input that is sent to them to ensure the integrity of the data beyond the simple constraints otherwise available on the tables. Parameters can be checked for valid ranges. Information can be cross checked with data in other tables.

Even if it is thought that someone attempting to crack into a website will never get this far in, from a security perspective, anything that can reduce the attack surface is beneficial.

Performance can be improved by the use of stored procedures. They are precompiled so when they are run there is no

additional lag as the SQL is parsed, compiled, execution plans drawn up and then run, they just run because all that extra work is done at the time the `CREATE PROCEDURE` or `ALTER PROCEDURE` commands are run rather than when procedures themselves are run.

Another area in which stored procedures improve performance is that is pushes all the work onto the server in one go. A stored procedure can perform a series of queries and return many tables in, what is to the outside world, one operation. This saves the calling process from making many requests and the additional time of several network roundtrips. It also means that, if the contents of one set of data being returned is dependent on the results of a previous set of data that is being retrieved through the same stored procedure, that the data only has to flow from the database server to the application. If stored procedures were not being used it would mean that the data from the first database call has to get sent back to the database for the second call in order for it to continue retrieving the information needed by the application.

For instance. Lets say that Northwind traders send out a quarterly statement to its customers, and that for each statement certain information needs to be extracted from the database. The tables Customer, Order and Order Details are used. This information could be retrieved in several steps by calling the database for each set of information as it is needed to generate the statements. First with a `SELECT * FROM Customers WHERE CustomerID = @CustomerID`. This gets the details for the head of the statement. Then a `SELECT * FROM Orders WHERE CustomerID = @CustomerID AND OrderDate>=@StartDate AND OrderDate<=@EndDate` to get the details for each individual order by that customer. And finally a series of calls (one for each of the Order records that were retrieved) like `SELECT * FROM [Order Details] WHERE OrderID = @OrderID`

Assuming that the customer in question is "Rattlesnake Canyon Grocery" and the period for the statement is Q1 1998 then that is 5 roundtrips to the database and 5 times the database has to parse some SQL. This could be done by a single stored procedure that takes only one trip to the database and is precompiled.

```
CREATE PROCEDURE GetQuarterlyStatement
@CustomerID nvarchar(5),
@StartDate datetime,
@EndDate datetime
AS
SELECT * FROM Customers
    WHERE CustomerID=@CustomerID
SELECT * FROM Orders
```

```
    WHERE CustomerID=@CustomerID
    AND OrderDate>=@StartDate
    AND OrderDate<=@EndDate
    ORDER BY OrderDate DESC
SELECT [Order Details].* FROM [Order Details]
    INNER JOIN Orders ON [Order Details].OrderID =
Orders.OrderID
    WHERE CustomerID=@CustomerID
    AND OrderDate>=@StartDate
    AND OrderDate<=@EndDate
    ORDER BY OrderDate DESC
GO
```

The stored procedure is now doing in one trip what previously took 5 trips. Of course, this example is somewhat contrived for brevity, in a real application there would be joins to the product tables, and the columns would be listed rather than using `SELECT *` and so on.