

Module 5: SQL and Finance

- Indexes
- Stored procedures



SQL

Objectives

In this module, we will focus on indexes and stored procedures.

Trainer Notes

Indexes and stored procedures will be critical in understanding the day to day of databases that you will be running/supporting.

Indexes

An index can be applied to a table in a database with the purpose of speeding up retrieval of rows from the table or the view.

It is composed of keys which are built on one or more of the columns in the table or the view.

The keys are stored in a B-tree structure so that the server can find the row or rows associated with it quickly.

Indexes are automatically created when primary key and unique constraints are defined on table columns.

Trainer/Trainee Notes

Real life production databases can be huge. And with the requirement of storing data for several years for regulatory reasons it is important that you can be efficient as possible in your queries on the database. Indexes will help with retrieval.

How Do Indexes Work?

Reduced I/O disk operations

- The query optimizer will evaluate the query when running it and decide which method is the most efficient for retrieving the data
- This could be a table scan or scanning one or more indexes

Table Scan

- In this method, the query optimizer reads all the rows in the table and extracts the relevant ones
- Can be resource intensive
- Likely used when the result is a high percentage of rows from the table

Index retrieval

- The optimizer searches the index key columns and finds the storage location of the rows needed by the query
- Usually faster as an index will contain very few columns and rows are stored in order

Trainer/Trainee Notes

Reducing the number of disk accesses will speed up queries and also reduce the impact on system resources

Think about indexes like the back of a book index

When Not To Use Indexes

While indexes speed up SELECT statements and the use of WHERE clauses the presence of an index will also slow down INSERT or UPDATE statements.

Indexes are also not appropriate on all tables. The below shows where they are not always suitable:

Small Tables

Tables with frequent large updates or inserts

Columns that are frequently touched in some way

Columns with large numbers of nulls

Trainer/Trainee Notes

Unless you are developing databases, I would not expect production support staff to have to create indexes but it is important to know about their existence.

Creating Indexes

So if we did want to create an index how would we do it?

```
CREATE INDEX index_name ON table_name  
(column1, column2....);
```

In this index, duplicate
values are allowed

```
CREATE UNIQUE INDEX index_name ON  
table_name (column1, column2....);
```

In this index, no duplicate
values are allowed

And dropping an index:

Command	Database
DROP INDEX table.index;	SQL Server
DROP INDEX index_name;	Oracle/DB2
ALTER TABLE table DROP INDEX index;	MySQL

Trainer/Trainee Notes

Here you can see that creating an index is very easy on the column you wish, and dropping the index is also pretty simple across the various database types.

SQL Stored Procedures

Stored procedures are used to group one or more SQL statements into logical units.

When you call a stored procedure for the first time, the server will create an execution plan which is then stored in the cache.

Once the execution plan is saved, the server will reuse this plan to execute the stored procedure every time it is run.

If your database changes over time, there is potential that your saved execution plan is not the most optimal way of retrieving data anymore and can lead to slow performing queries.

Trainer/Trainee Notes

Execution plans not performing are often the root cause of performance issues on a database. A DBA will have tools to analyse the query plans to understand where the inefficiencies lie. A execution plan can be deleted which will force the engine to recalculate and hopefully create a more efficient query and result.

Creating and Running Stored Procedures

How would a stored procedure be created?

```
CREATE PROCEDURE spName  
AS  
BEGIN  
SELECT columns FROM table ORDER BY column;  
END;
```

This is compiled as a normal query on the database. You should then be able to see it in the object explorer. It can be run in the following ways:

EXECUTE sp_name;

EXEC sp_name;

Trainer/Trainee Notes

Having a high level understanding of stored procedures and execution plans is important.

Modifying and Deleting a Stored Procedure

Modifying and deleting a stored procedure is fairly simple:

```
ALTER PROCEDURE spName  
AS  
BEGIN  
SELECT columns FROM table ORDER BY column;  
END;
```

Deleting:

```
DROP PROCEDURE sp_name;
```

```
DROP PROC sp_name;
```

Trainer/Trainee Notes

Check with the database version you are running what the correct syntax is. Also be very careful deleting a stored procedure or altering it as multiple people may well be using it as well as the application itself.

Query Optimization

SQL server is a cost based optimizer.

The lowest cost plan will always be taken into consideration and CPU/IO and memory are some of the parameters taken into consideration.

Finding an optimal plan

- The optimizer has to find the balance between what plan it wants to select and maximising the effectiveness of that plan pre execution
- It has to make this decision before actually executing the task itself (i.e., in a finite amount of time)
- The estimated execution plan uses stats and many other parameters based on what is available on the server

Trainer/Trainee Notes

Statistics are important here – if the statistics on a database are stale, it will potentially cause the wrong plan to be chosen and executed. Often DBAs will need to refresh the statistics if performance issues are seen