Developing SQL Databases

Lab 4 – Creating Indexes

Overview

A table named Opportunity has recently been added to the DirectMarketing schema within the database, but it has no constraints in place. In this lab, you will implement the required constraints to ensure data integrity and, if you have time, test that constraints work as specified.

Before starting this lab, you should view **Module 4 – Creating Indexes** in the course *Developing SQL Databases*. Then, if you have not already done so, follow the instructions in the **Getting Started** document for this course to set up the lab environment.

If you find some of the challenges difficult, don't worry – you can find suggested solutions for all of the challenges in the **Lab Solution** folder for this module.

What You'll Need

To complete the labs, you will need the following:

- An Azure SQL Database instance with the AdventureWorksLT sample database. Review the Getting Started document for information about how to provision this.
- The lab files for this course

Setup

1. Using SQL Server Management Studio, connect to the AdventureWorksLT database.

2. Open Lab4Setup.sql from the Setup folder for this course and run the following Transact-SQL:

```
CREATE TABLE SalesLT.MediaOutlet (

MediaOutletID INT NOT NULL,

MediaOutletName NVARCHAR(40),

PrimaryContact NVARCHAR (50),

City NVARCHAR (50)

);


GO

CREATE TABLE SalesLT.PrintMediaPlacement ( PrintMediaPlacementID INT NOT NULL,

MediaOutletID INT,

PlacementDate DATETIME,

PublicationDate DATETIME,

RelatedProductID INT,

PlacementCost DECIMAL(18,2)

);
```

Challenge 1: Create Clustered Indexes

The sales department has begun using the new tables, and are finding that, when trying to query the data, the performance is unacceptable. They have asked you to make any database changes you can to improve performance.

Add a Clustered Index to SalesLT.MediaOutlet

1. Consider which column is best suited to an index.
2. Using Transact-SQL statements, add a clustered index to that column on the SalesLT.MediaOutlet table.
3. Consider implementing the index by creating a unique constraint.
4. Use Object Explorer to check that the index was created successfully.

Add a Clustered Index to SalesLT.PrintMediaPlacement

1. Consider which column is best suited to an index.
2. Using Transact-SQL statements, add a clustered index to that column on the SalesLT.PrintMediaPlacement table.
3. Consider implementing the index by creating a unique constraint.
4. Use Object Explorer to check that the index was created successfully.

Challenge 2: Create a Covering Index

The sales team has found that the performance improvements that you have made are not working for one specific query. You have been tasked with adding additional performance improvements to handle this query.

Add Some Test Data

1. Open the **Setup** folder.
2. Run the Transact-SQL in InsertDummyData.sql to insert test data into the two tables.

Run the Poor Performing Query

1. Switch on Include Actual Execution Plan.
2. Open the **Setup** folder.
3. Run the Transact-SQL in **SalesQuery.sql**.
4. Examine the Execution Plan.
5. Note the missing index warning in SQL Server Management Studio.

Create a Covering Index

1. On the **Execution Plan** tab, right-click the green **Missing Index** and click **Missing Index Details**.
2. Use the generated Transact-SQL to create the missing covering index with a name of **NCI_PrintMediaPlacement**.
3. Use Object Explorer to check that the index was created successfully.

Check the Performance of the Sales Query

1. Re-run the sales query.
2. Check the Execution Plan and ensure the database engine is using the new **NCI_PrintMediaPlacement** index.
3. Close SQL Server Management Studio without saving any changes