Capgemini

RDBMS - SQL Server

Lesson 06 : Database Objects: Indexes & Views

Lesson Objectives

- In this lesson, you will learn:
  - Creating Indexes
  - Querying the sysindexes Table
  - Performance Considerations
  - Creating Views

Index – An Overview

- Database systems generally use indexes to provide fast access to relational data
- An index is a separate physical data structure that enables queries to access one or more data rows fast
- This structure is known as B-Tree Structure
- Proper tuning of index is therefore a key for query performance
- Database Engine uses index to find the data just like one uses index in a book
- When a table is dropped , indexes also get dropped automatically
- Only the owner of the table can create indexes
- SQL Server supports two types of indexes
  - Clustered
  - Non clustered

How SQL Server access data?

- SQL Server accesses data in one of two ways:
- By scanning all the data pages in a table, which is called a table scan. When SQL Server performs a table scan, it:
  - Starts at the beginning of the table
  - Scans from page to page through all the rows in the table
  - Extracts the rows that meet the criteria of the query
- By using indexes. When SQL Server uses an index, it:
  - Traverses the index tree structure to find rows that the query requests
  - Extracts only the needed rows that meet the criteria of the query

Clustered Index

- A clustered index determines the physical order of the data in a table
- Database Engine allows the creation of a single clustered index per table
- If a clustered index is defined for a table, the table is called a clustered table
- A Unique Clustered index is built by default for each table, for which you define the primary key using the primary key constraint
- Also, each clustered index is unique by default that is, each data value can appear only once in a column for which the clustered index is defined

Non-Clustered Index

- A Non-Clustered index has the same index structure as a clustered index
- A Non-Clustered index does not change the physical order of the rows in the table
- A table can have more than non clustered index
- Unique Non-Clustered index will be created automatically when you create unique key on a column to enforce uniqueness of key value

Creating and Dropping Indexes

- Indexes are created automatically on tables with PRIMARY KEY or UNIQUE constraints
  - Indexes can also be created using the CREATE INDEX Statement

```
USE Northwind
CREATE CLUSTERED INDEX CL_lastname
ON employees(lastname)
```

  - Indexes can be dropped using the DROP command

```
USE Northwind
DROP INDEX employees.CL_lastname
```

Creating and Dropping Indexes

- To create non clustered index ncl_deptno

```
USE Northwind
    CREATE NON CLUSTERED INDEX NCL_deptno
    ON employees(deptno)
```

- Using the DROP INDEX Statement

```
USE Northwind
DROP INDEX employees.NCL_deptno
```

Creating Unique Indexes

- Unique index can be non clustered or clustered
- Unique non clustered index is automatically created when a column has UNIQUE constraint
- Unique Clustered index is automatically created when column has a PRIMARY KEY constraint
- Ensures column(s) have unique value
- There is no difference in the way Unique constraint and Unique index work, except for syntax

```
USE Northwind
CREATE UNIQUE NONCLUSTERED INDEX U_CustID
ON customers(CustomerID)
```

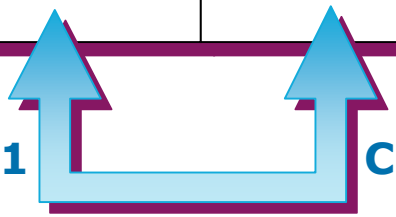
Creating Composite Indexes

- Index of two /more columns are said to be composite

```
USE Northwind
CREATE UNIQUE NONCLUSTERED INDEX
U_OrdID_ProdID
ON [Order Details] (OrderID, ProductID)
```

Order Details

OrderID	ProductID	UnitPrice	Quantity	Discount
10248	11	14.000	12	0.0
10248	42	9.800	10	0.0
10248	72	34.800	5	0.0



Column 1 **Column 2**

Composite Key

Columnstore Indexes

- SQL Server 2012 introduces ColumnStore Indexes.

Benefits of using SQL Server ColumnStore Indexes

- Faster query performance for common data warehouse queries as only required columns/pages in the query are fetched from disk
- Data is stored in a highly compressed form to reduce the storage space
- Frequently accessed columns (pages that contains data for these columns) remain in memory because a high ratio of compression is used in the pages and less pages are involved

Columnstore Indexes

```
CREATE NONCLUSTERED COLUMNSTORE INDEX
idx_colSale
ON myTable (OrderDate, ProductID, SaleAmount)
```

Obtaining information on Indexes

- Using the sp_helpindex System Stored Procedure

```
USE Northwind
EXEC sp_helpindex Customers
```

- Using the sp_help tablename System Stored Procedure

Indexes – Performance Considerations

- Create indexes on foreign keys
- Create the clustered index before nonclustered indexes
- Consider before creating composite indexes
- Create multiple indexes for a table that is read frequently
- Use the index tuning wizard get statics of index usage

Demo

- Creating Indexes

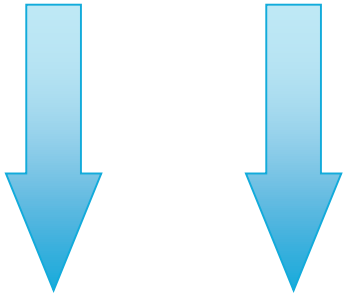
Views – An Overview

- Views are Virtual tables, which provides access to a subset of columns from one or more tables
- Created from one or more base tables or other views
- Internally Views are stored queries
- Views are created when
  - To hide the complexity of the underlying database schema, or customize the data and schema for a set of users.
  - To control access to rows and columns of data.
- Objective of creating views is Abstraction , not performance

Views – An Overview

Employees

EmployeeID	LastName	Firstname	Title
1	Davolio	Nancy	~~~
2	Fuller	Andrew	~~~
3	Leverling	Janet	~~~



```
USE Northwind
GO
CREATE VIEW dbo.EmployeeView
AS
SELECT LastName, Firstname
FROM Employees
```

EmployeeView

Lastname	Firstname
Davolio	Nancy
Fuller	Andrew
Leverling	Janet

User's View

Views – Advantages

- Focus the Data for Users
  - Focus on important or appropriate data only
  - Limit access to sensitive data
- Mask Database Complexity
  - Hide complex database design
  - Simplify complex queries, including distributed queries to heterogeneous data
- Simplify Management of User Access on Data

Views – Types

- Standard Views
- Indexed Views
- Partitioned Views

Defining Views

- Creating views
- Altering and dropping views
- Locating view definition information
- Hiding view definitions

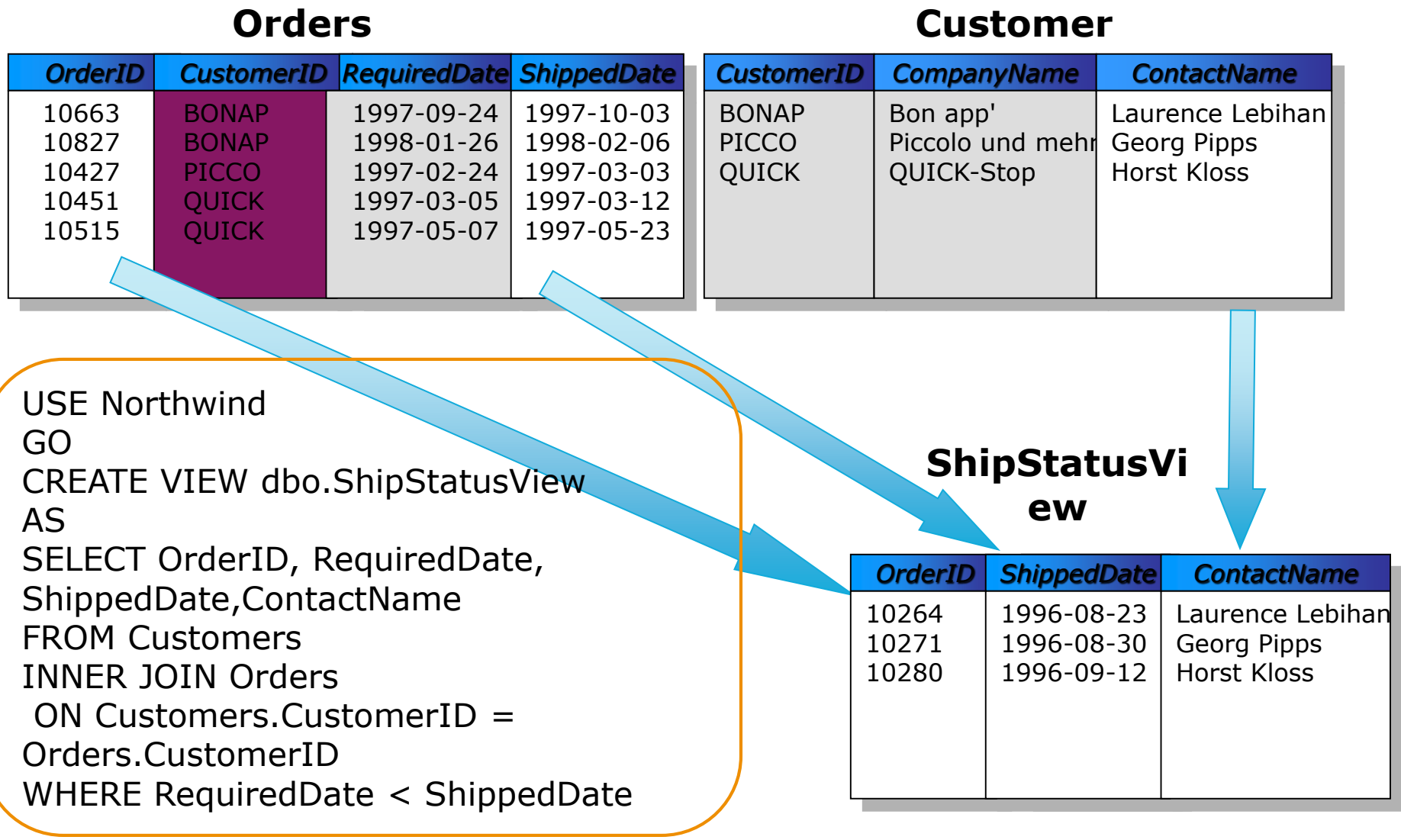
Creating Views

- Creating a View

```
CREATE VIEW dbo.OrderSubtotalsView (OrderID, Subtotal)
AS
SELECT OD.OrderID,
 SUM(CONVERT(money,(OD.UnitPrice*Quantity*(1-
Discount)/100))*100)
FROM [Order Details] OD
GROUP BY OD.OrderID
GO
```

- Restrictions on View Definitions
  - Cannot include ORDER BY clause
  - Cannot include INTO keyword

Example – Views with Join Query



Altering & Dropping Views

- Altering Views
  - Retains assigned permissions
  - Causes new SELECT statement and options to replace existing definition

```
USE Northwind
GO
ALTER VIEW dbo.EmployeeView
AS
SELECT LastName, FirstName, Extension
FROM Employees
```

- Dropping Views

```
DROP VIEW dbo.ShipStatusView
```

Locating View Definition Information

- Locating View Definitions
  - Not available if view was created using WITH ENCRYPTION option
- Locating View Dependencies
  - Lists objects upon which view depends
  - Lists objects that depend on a view

Hiding View Definition

- Use the WITH ENCRYPTION Option
- Do not delete entries in the syscomments table

```
USE Northwind
GO
CREATE VIEW dbo.[Order Subtotals]
   WITH ENCRYPTION
AS
SELECT OrderID,
 Sum(CONVERT(money, (UnitPrice * Quantity * (1 - Discount) /
100)) * 100) AS Subtotal
FROM [Order Details]
GROUP BY OrderID
GO
```

Modifying Data through View

- Cannot affect more than one underlying table
- Cannot be made to columns having aggregation
- Depends on the constraints placed on the base tables
- Are verified if the WITH CHECK OPTION has been specified

Views – Recommended Practices

- Use a Standard Naming Convention
- dbo Should Own All Views
- Verify Object Dependencies Before You Drop Objects
- Never Delete Entries in the syscomments Table
- Carefully Evaluate Creating Views Based on Views

Demo

- Working with Views

Summary

- In this lesson, you have learnt:
  - Creating Indexes
  - Types of indexes
    - Clustered Index, Non clustered index ,Filtered Indexes ,Column store Indexes
  - Creating and modifying Views

Review Question

- Question 1: -------- Gets created automatically for Primary key constrain
  - clustered index
  - Unique clustered index
  - Unique Non clustered index
- Question 2: -------- option with views will not stored base query of views in syscomments table
- Question 3: A table can have multiple unique non clustered index
  - True/False