# Module 7: Creating and Maintaining Indexes

# Overview

- Creating Indexes
- Creating Index Options
- Maintaining Indexes
- Introduction to Statistics
- Querying the sysindexes Table
- Setting Up Indexes Using the Index Tuning Wizard
- Performance Considerations

# Creating Indexes

- Creating and Dropping Indexes
- Creating Unique Indexes
- Creating Composite Indexes
- Creating Indexes on Computed Columns
- Obtaining Information on Existing Indexes

# Creating and Dropping Indexes

- **Using the CREATE INDEX Statement**
  - Indexes are created automatically on tables with PRIMARY KEY or UNIQUE constraints
  - Indexes can be created on views if certain requirements are met

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

- **Using the DROP INDEX Statement**

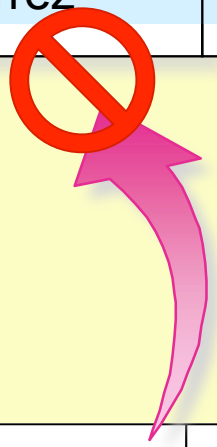
```
USE Northwind
DROP INDEX employees.CL_lastname
```

# Creating Unique Indexes

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

**Customers**

| *CustomerID* | *CompanyName* | *ContactName* | … |
|---|---|---|---|
| QUICK | QUICK-Stop | Horst Kloss | |
| BONAP | Bon app' | Laurence Lebihan | |
| RANCH | Rancho grande | Sergio Gutiérrez | |



**Duplicate key values are not allowed when a new row is added to the table**

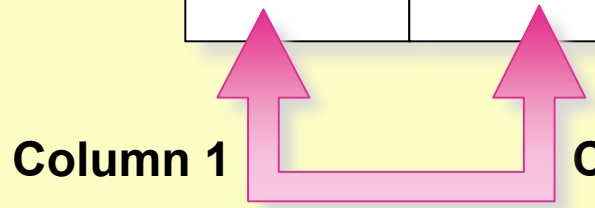
| | | | |
|---|---|---|---|
| RANCH | Santé Gourmet | Jonas Bergulfsen | … |

# Creating Composite Indexes

```
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 |




**Composite Key**

# Creating Indexes on Computed Columns

- **You Can Create Indexes on Computed Columns When:**
  - Computed_column_expression is deterministic and precise
  - ANSI_NULL connection-level option is ON
  - Computed column cannot evaluate to the **text**, **ntext**, or **image** data types
  - Required SET options are set ON when you create the index and when INSERT, UPDATE, or DELETE statements change the index value
  - NUMERIC_ROUNDABORT option is set OFF
- **Query Optimizer May Ignore an Index on a Computed Column**

# Obtaining Information on Existing Indexes

- Using the sp_helpindex System Stored Procedure

```
USE Northwind
EXEC sp_helpindex Customers
```

- Using the sp_help *tablename* System Stored Procedure

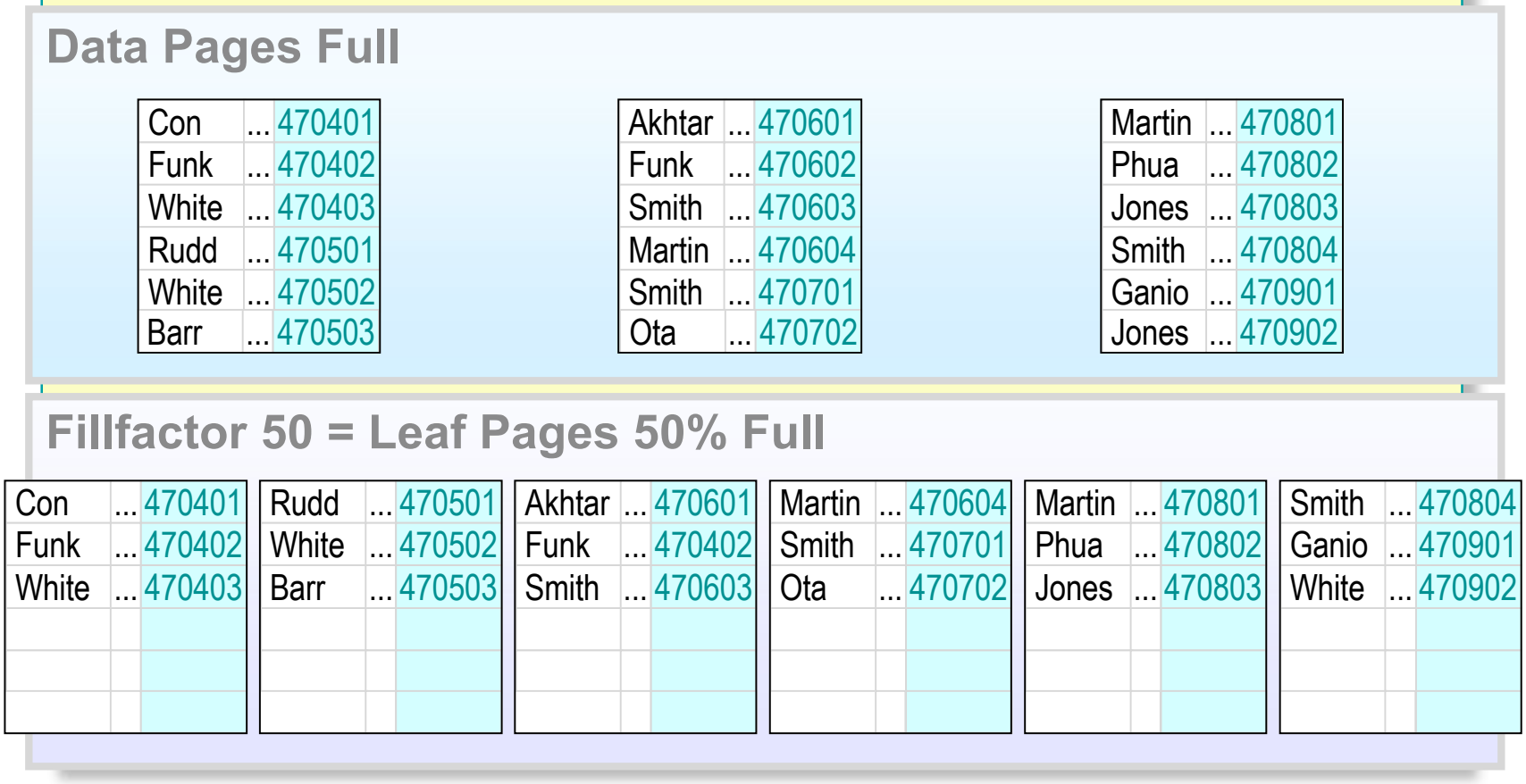
# Creating Index Options

- Using the FILLFACTOR Option
- Using the PAD_INDEX Option

- Info on Clustered vs Non-Clustered Indexes in SQL Server – See http://www.sql-server-performance.com/gv_index_data_structures.asp .

# Using the FILLFACTOR Option

- Specifies How Much to Fill the Page
- Impacts Leaf-Level Pages



# Using the PAD_INDEX Option

- The PAD_INDEX Option Applies to Non-Leaf-Level Index Pages
- If PAD_INDEX Is Not Specified, the Default Leaves Space for One Row Entry in Non-Leaf-Level Pages
- Number of Rows on Non-Leaf-Level Pages Is Never Less Than Two
- PAD_INDEX Uses the Fillfactor Value

```
USE Northwind
CREATE INDEX OrderID_ind
       ON Orders(OrderID)
       WITH PAD_INDEX, FILLFACTOR=70
```

# Maintaining Indexes

- Data Fragmentation
- DBCC SHOWCONTIG Statement
- DBCC INDEXDEFRAG
- DROP_EXISTING Option

# Data Fragmentation

- **How Fragmentation Occurs**
  - SQL Server reorganizes index pages when data is modified
  - Reorganization causes index pages to split
- **Methods of Managing Fragmentation**
  - Drop and recreate an index and specify a fillfactor value
  - Rebuild an index and specify a fillfactor value
- **Business Environment**
  - Data fragmentation can be good for OLTP environment
  - Data fragmentation can be bad for Analysis Services environment

# DBCC SHOWCONTIG Statement

- **What DBCC SHOWCONTIG Determines**
  - Whether a table or index is heavily fragmented
  - Whether data and index pages are full
- **When to Execute**
  - If tables have been heavily modified
  - If tables contain imported data
  - If tables seem to cause poor query performance

# DBCC INDEXDEFRAG Statement

- **DBCC INDEXDEFRAG**
  - Defragments the leaf level of an index
  - Arranges leaf-level pages so that the physical order of the pages matches the left-to-right logical order
  - Improves index-scanning performance
- **Index Defragmenting vs. Index Rebuilding**

# DROP_EXISTING Option

- **Rebuilding an Index**
  - Reorganizes leaf pages
  - Removes fragmentation
  - Recalculates index statistics
- **Changing Index Characteristics**
  - Type
  - Index columns
  - Options

```
CREATE UNIQUE NONCLUSTERED INDEX U_OrdID_ProdID
ON [Order Details] (OrderID, ProductID)
WITH DROP_EXISTING, FILLFACTOR=65
```

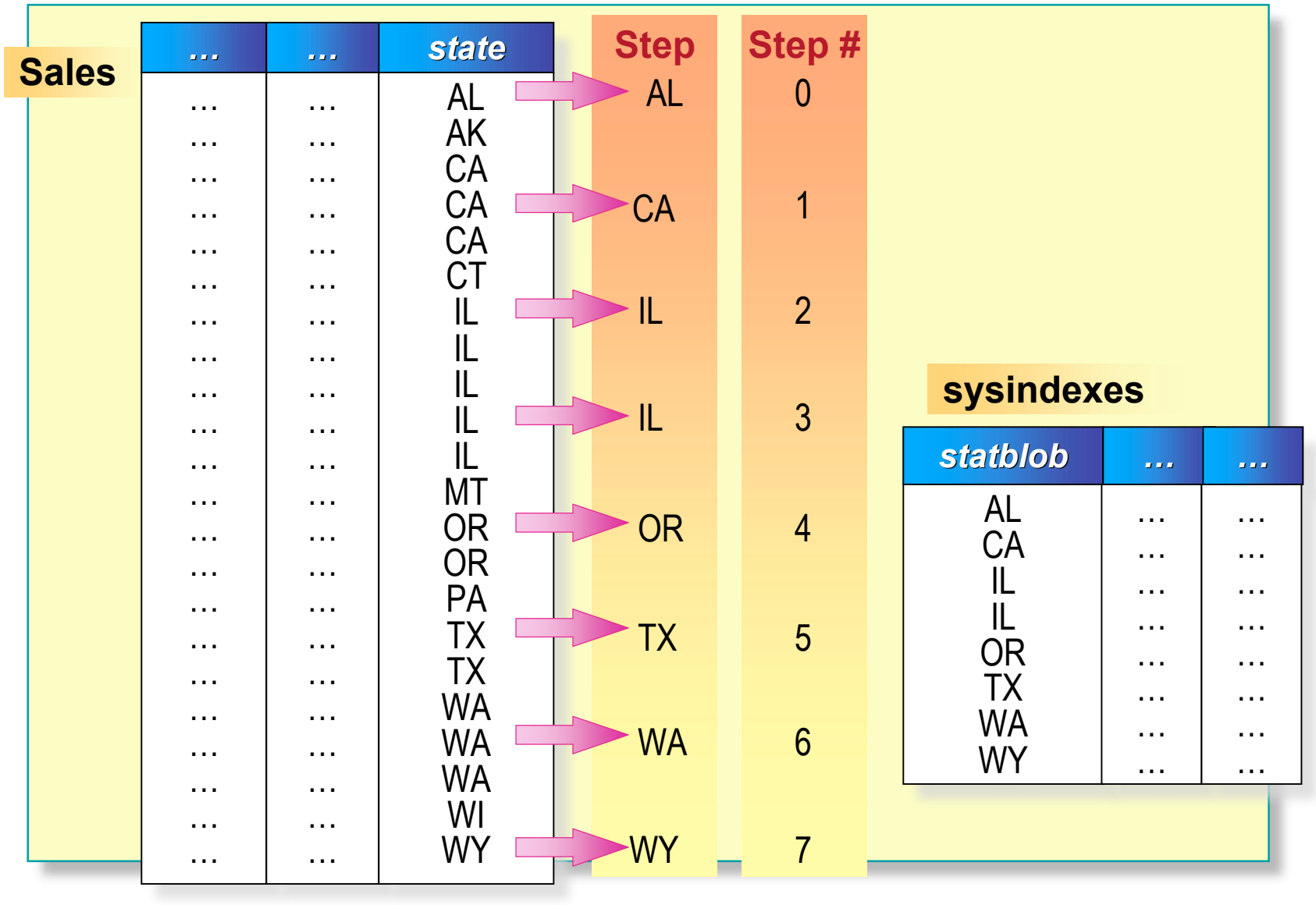
# Introduction to Statistics

- How Statistics Are Gathered
- How Statistics Are Stored
- Creating Statistics
- Updating Statistics
- Viewing Statistics

# How Statistics Are Gathered

- **Reads Column Values or a Sampling of Column Values**
  - Produces an evenly distributed sorted list of values
- **Performs a Full Scan or Sampling of Rows**
  - Dynamically determines the percentage of rows to be sampled based on the number of rows in the table
- **Selects Samplings**
  - From the table or from the smallest nonclustered index on the columns
  - All of the rows on the data page are used to update the statistical information

# How Statistics Are Stored



# Creating Statistics

- **Automatically Creating Statistics**
  - Indexed columns that contain data
  - Non-indexed columns that are used in a join predicate or a WHERE clause
- **Manually Creating Statistics**
  - Columns that are not indexed
  - All columns other than the first column of a composite index

# Updating Statistics

- **Frequency of Updating Statistics**
- **Automatically Updating Statistics**
- **Manually Updating Statistics**
  - If you create an index before any data is put into the table
  - If a table is truncated
  - If you add many rows to a table that contains minimal or no data, and you plan to immediately query against that table

# Viewing Statistics

- **The DBCC SHOW_STATISTICS Statement Returns Statistical Information in the Distribution Page for an Index or Column**
- **Statistical Information Includes:**
  - The time when the statistics were last updated
  - The number of rows sampled to produce the histogram
  - Density information
  - Average key length
  - Histogram step information

# Querying the sysindexes Table

- **Stores Table and Index Information**
  - Type of index (**indid**)
  - Space used (**dpages**, **reserved**, and **used**)
  - Fillfactor (**OrigFillFactor)**
- **Stores Statistics for Each Index**

# Setting Up Indexes Using the Index Tuning Wizard

- **Use the Index Tuning Wizard to:**
  - Recommend or verify optimal index configuration
  - Provide cost analysis reports
  - Recommend ways to tune the database
  - Specify criteria when a workload is evaluated
- **Do Not Use the Index Tuning Wizard on:**
  - Tables referenced by cross-database queries that do not exist
  - System tables, PRIMARY KEY constraints, unique indexes

# Performance Considerations

- Create Indexes on Foreign Keys
- Create the Clustered Index Before Nonclustered Indexes
- Consider Creating Composite Indexes
- Create Multiple Indexes for a Table That Is Read Frequently
- Use the Index Tuning Wizard

# Recommended Practices

- Use the FILLFACTOR Option to Optimize Performance
- Use the DROP_EXISTING Option for Maintaining Indexes
- Execute DBCC SHOWCONTIG to Measure Fragmentation
- Allow SQL Server to Create and Update Statistics Automatically
- Consider Creating Statistics on Nonindexed Columns to Enable More Efficient Execution Plans

# Review

- Creating Indexes
- Creating Index Options
- Maintaining Indexes
- Introduction to Statistics
- Querying the sysindexes Table
- Setting Up Indexes Using the Index Tuning Wizard
- Performance Considerations