



### **Index Concepts**

Guarantees order in which data is sorted

Think of a phonebook

Data sorted by last name

Designing indexes takes time and planning

- More indexes generally leads to faster querying
- More indexes generally leads to slower modifications
- Data in an index must be kept up to date

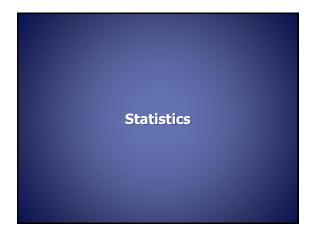
Indexes must be maintained

Administrator must defragment indexes

TRAINSIGNAL

# Index Types Clustered index - Order in which the data in the table is stored - Page number in a book Nonclustered index - Copy of data sorted for queries - Index in the back of a book · Contains a pointer back to the original content

### **Index Example** Customers CustomerID Name JoinDate Christopher 11/8/2011 42 43 Karin 12/1/2011 44 12/5/2011 45 Susan 12/7/2011 Christopher 42 Dave 45 43 SELECT Name Susan 44 , JoinDate FROM Customers WHERE Name LIKE 'Chris%'; TRAINSIGNAL



### **Query Optimizer**

Determines how to execute a query

- Generates an execution plan

### Goals

- Best way to execute a query
- Determine the execution plan as quickly as possible



TRAINSIGNAL

### **Statistics**

Stores information about data in indexes and tables

- "Cliff's Notes"

Used to determine how to execute a query SQL Server automatically maintains indexes

- Administrators can control this manually

TRAINSIGNAL

**Query Plans** 

# Query Plans See what's going on behind the scenes Determine the execution plan used by SQL Server Uses: Optimizing queries Determine which indexes were used Two options to view: Actual execution plan

- Requires the query be executed
- Estimated execution plan
  - May not be correct.

TRAINSIGNAL

### **Data Access Methods**

### Scan

- Examines every row in the table or index
- Imagine searching for all people named Christopher in the phonebook

### Seek

- Able to follow path directly to the desired data
- Imagine searching for all people with a last name of Harrison in the phonebook

TRAINSIGNAL

### **Join Algorithms**

### Merge Join

- Data is presorted
- Most efficient join

### Loop Join (Nested Loops)

- One table is much smaller than the other
- Smaller table is searched for values that match larger table

### Hash Join

- Large tables and unsorted data
- Slowest type of join

TRAINSIGNAL

1	
-	



### **Query Hint Concepts**

Control how SQL Server will execute a query

- Which index to use
   How to lock data

Typically avoided

TRAINSIGNAL

### **Locking Hints**

### ROWLOCK

Force locks at the row level

### PAGELOCK

- Force locks at the page level

### TABLOCK

- Force locks at the table level

TRAINSIGNAL

# **Index and Parameter Hints FORCESCAN** Scans index or table FORCESEEK Seeks at the index or table OPTIMIZE FOR UNKNOWN Tells SQL Server to create a query plan where the value for the parameter will change frequently TRAINSIGNAL **Dynamic Management Objects Dynamic Management Objects** Used to see what's going on behind the scenes - Find out how SQL Server is using resources Identify problem queries Views are treated like tables Functions accept parameters Parameters are often other database objects TRAINSIGNAL

## **Index Dynamic Management Objects** sys.dm\_db\_index\_usage\_stats Determine which indexes are (or aren't) being used sys.dm\_db\_mising\_index\_details Determine what indexes SQL Server thinks should be added $sys.dm\_db\_index\_physical\_stats$ - Determine how an index is using disk space TRAINSIGNAL **Transaction Dynamic Management Objects** sys.dm\_tran\_database\_transactions - Something $sys.dm\_tran\_session\_transactions$ Something sys.dm\_tran\_locks Something else TRAINSIGNAL **Dynamic SQL**

Dynamic SQL	
Allows for the creation of SQL on the fly  - Can enable a more flexible application	
Concerns  - Performance  • Something  - Security  • SQL Injection Attacks  - Allows an attacker to inject their on SQL statements  - Use parameters	
TRAINSIGNAL	
Executing Dynamic SQL Statements	]
<pre>EXECUTE sp_executesql <sql>     , <parameter definitions="">     , <parameter values=""></parameter></parameter></sql></pre>	
TRAINSIGNAL	