Microsoft

Cardinality Estimation & Statistics

Agenda

- New CE Model
- Understanding Statistics
- Managing Statistics

What is Cardinality Estimation?

"The process of estimating the number of rows flowing through the operators of a query execution plan"

Cardinality Estimation in PDW 2012

- New Cardinality Estimation (CE) model
- Shipped in SMP SQL Server in 2014
- More robust model
- Stabilising plan choice
- Simplified algorithm
- Improved estimations on average

Understanding New CE Model

Cardinality Model Assumptions

1. Containment
2. Uniformity
3. Independence

057

Containment

Containment

- Joins: values exist in both LEFT & RIGHT tables
- Filters: constant value exists in table

Base Containment

- Leaf level only
- Joins and filters evaluated separately

Uniformity

Statistical histogram contains up to 200 steps

Assumptions

- When value exists in a bucket range it is assumed to exist evenly in the bucket and with the same frequency of occurrence
- When value does not exist in any bucket range then it is assumed to exist evenly across all the buckets
- Distinct values are assumed to be spread evenly across a bucket

Independence

Assumptions

- Columns in the **same** table are correlated to some degree
- Columns in **different** tables are **not** correlated in any way

Different approaches used

- Filters
- Group by

Exponential Back-off

Fulfils the relaxed independence assumption for columns in the same table

- Multiple filters must exist on the same table
- Max 4 filters sorted by increasing selectivity i.e. most selective first

p0*SQRT(p1)*SQRT(SQRT(p2))*SQRT(SQRT(SQRT(p3)))

- Algorithm softens the estimate by taking larger square roots

Average Frequency

Designed to fix Ascending Key problems

- Assumes that the queried values exist in the data set even if the value falls out of the range of the histogram

Equi-Joins

New CE Model uses “Coarse alignment”

4 Step process

1. Distinct Count of left & right tables
2. Average frequency of left & right tables
3. Join frequency
4. Join cardinality

Calculating Equi-Join Cardinality

1. Distinct count of left & right table
   - Distinct count of rows for columns used
   - Retain the Lesser of the two Distinct Counts for step 4
2. Avg. frequency of left & right tables
   - # Rows (Total Cardinality) / Distinct Count for each side
3. Join frequency
   - Avg. Frequency LEFT Table * Avg. Frequency Right Table
4. Join cardinality
   - Join Frequency * Lesser Distinct Count

Simple Example

LEFT TABLE A

- 1,000,000 Rows

RIGHT TABLE B

50,000 Rows

Step 1:
Distinct Count

Left	100,000
Right	50,000

Step 2:
Average Frequency

Left	10
Right	1

Step 3:
Join Frequency
10

Step 4:
Join Cardinality
500,000

Join cardinality =
Join Frequency
* Lesser of Distinct Counts from Step 1

Complex Joins – Equality & Inequality

New CE Model assumes

- A 1:M relationship between a small table and a large table
- Each row in the large table matches exactly one row in the small table

Forces the estimated size of the larger input for join cardinality

Ambient Cardinality for Group By

Affects GROUP BY calculation

- Caps the size of the group by estimation
- Looks at the columns used in GROUP BY
- Calculates distinct columns based on smallest number of joins to satisfy GROUP BY clause
- If the ambient cardinality estimate is significantly different to the distinct count value then it will be used
- Otherwise favour independence

White Paper: New CE Model

New white paper published for SQL 2014

Relevant for PDW but note the following

- Most troubleshooting options aren’t available
- Fixed database compatibility
- No Trace Flag support
- No XEvents

Understanding Statistics

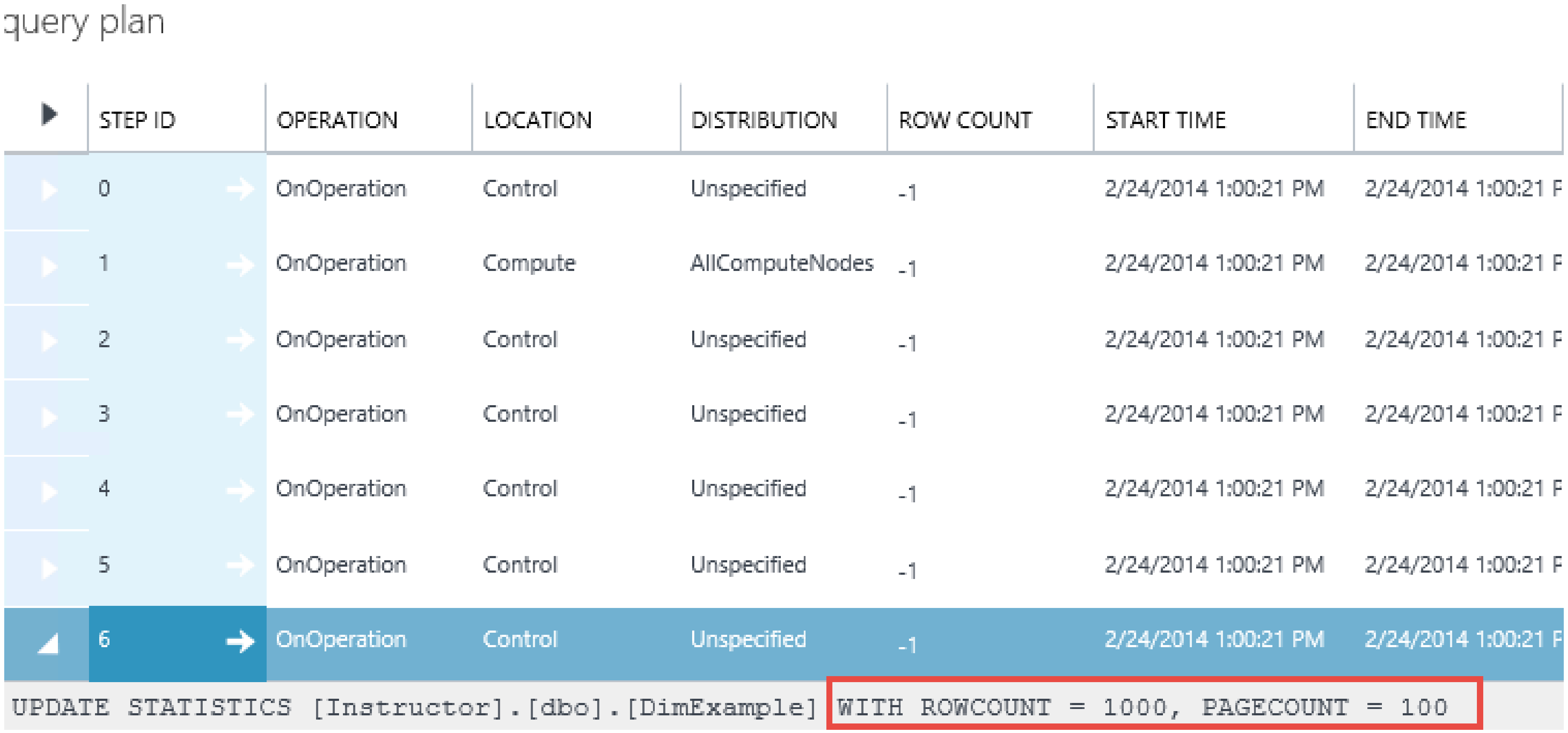
What are Statistics

- Database objects
- Contain statistical information about the distribution of values in a column
  - The statistics object includes a histogram to show the distribution of values in the first column
- Multi-column statistics can also be generated
  - These also hold density information i.e. the correlation of values between columns

Table Level Statistics

- Statistical information also held on the table
- Row count
- Page Count
- Used to estimate the size of a given table

Table Level Statistics – Create Table



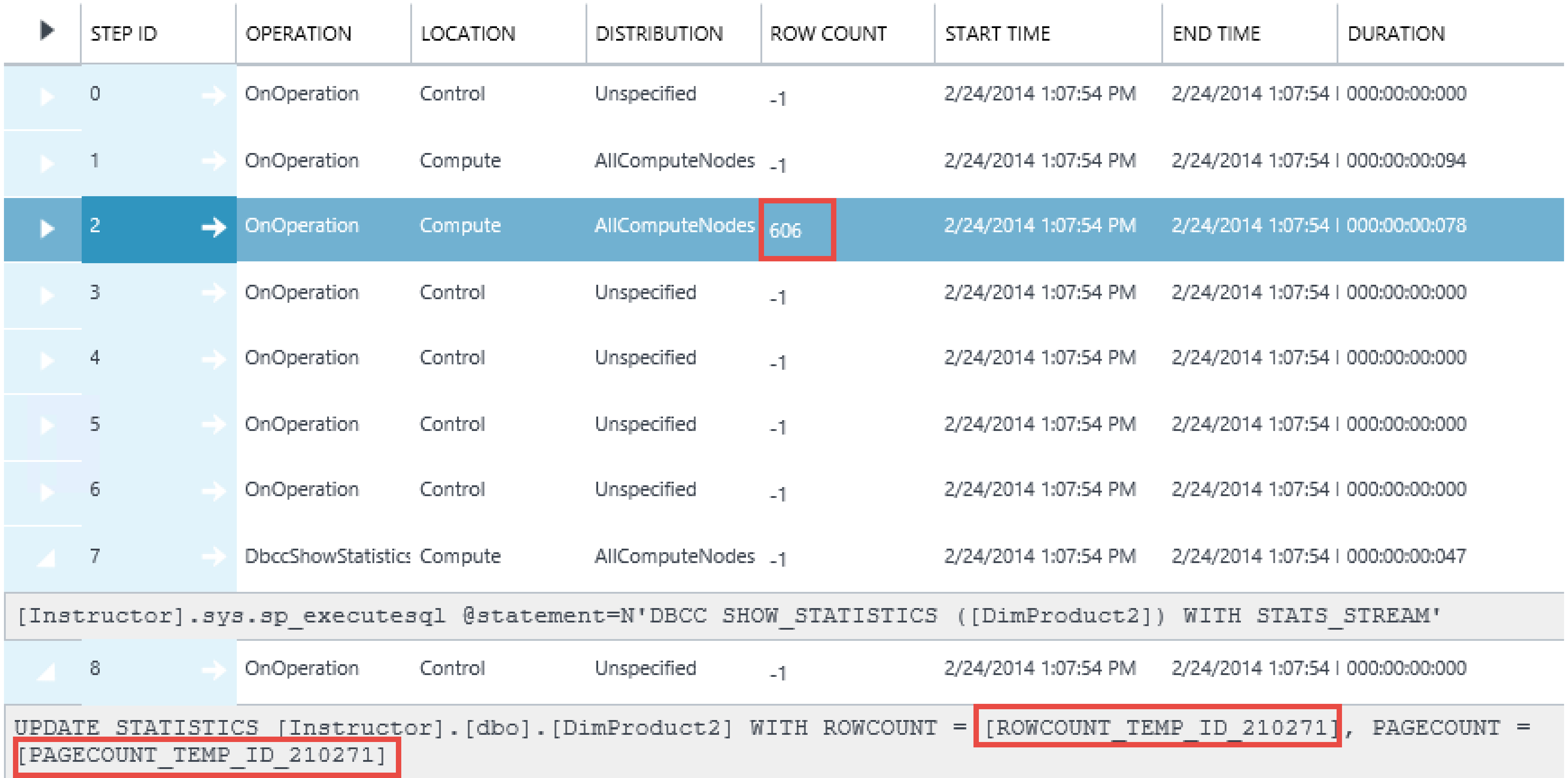
query plan

	STEP ID	OPERATION	LOCATION	DISTRIBUTION	ROW COUNT	START TIME	END TIME
	0	OnOperation	Control	Unspecified	-1	2/24/2014 1:00:21 PM	2/24/2014 1:00:21 F
	1	OnOperation	Compute	AllComputeNodes	-1	2/24/2014 1:00:21 PM	2/24/2014 1:00:21 F
	2	OnOperation	Control	Unspecified	-1	2/24/2014 1:00:21 PM	2/24/2014 1:00:21 F
	3	OnOperation	Control	Unspecified	-1	2/24/2014 1:00:21 PM	2/24/2014 1:00:21 F
	4	OnOperation	Control	Unspecified	-1	2/24/2014 1:00:21 PM	2/24/2014 1:00:21 F
	5	OnOperation	Control	Unspecified	-1	2/24/2014 1:00:21 PM	2/24/2014 1:00:21 F
	6	OnOperation	Control	Unspecified	-1	2/24/2014 1:00:21 PM	2/24/2014 1:00:21 F

```
UPDATE STATISTICS [Instructor].[dbo].[DimExample] WITH ROWCOUNT = 1000, PAGECOUNT = 100
```

Table Level Statistics - CTAS

query plan



	STEP ID	OPERATION	LOCATION	DISTRIBUTION	ROW COUNT	START TIME	END TIME	DURATION
	0	OnOperation	Control	Unspecified	-1	2/24/2014 1:07:54 PM	2/24/2014 1:07:54 I	000:00:00:000
	1	OnOperation	Compute	AllComputeNodes	-1	2/24/2014 1:07:54 PM	2/24/2014 1:07:54 I	000:00:00:094
	2	OnOperation	Compute	AllComputeNodes	606	2/24/2014 1:07:54 PM	2/24/2014 1:07:54 I	000:00:00:078
	3	OnOperation	Control	Unspecified	-1	2/24/2014 1:07:54 PM	2/24/2014 1:07:54 I	000:00:00:000
	4	OnOperation	Control	Unspecified	-1	2/24/2014 1:07:54 PM	2/24/2014 1:07:54 I	000:00:00:000
	5	OnOperation	Control	Unspecified	-1	2/24/2014 1:07:54 PM	2/24/2014 1:07:54 I	000:00:00:000
	6	OnOperation	Control	Unspecified	-1	2/24/2014 1:07:54 PM	2/24/2014 1:07:54 I	000:00:00:000
	7	DbccShowStatistics	Compute	AllComputeNodes	-1	2/24/2014 1:07:54 PM	2/24/2014 1:07:54 I	000:00:00:047

```
[Instructor].sys.sp_executesql @statement=N'DBCC SHOW_STATISTICS ([DimProduct2]) WITH STATS_STREAM'
```

	STEP ID	OPERATION	LOCATION	DISTRIBUTION	ROW COUNT	START TIME	END TIME	DURATION
	8	OnOperation	Control	Unspecified	-1	2/24/2014 1:07:54 PM	2/24/2014 1:07:54 I	000:00:00:000

```
UPDATE STATISTICS [Instructor].[dbo].[DimProduct2] WITH ROWCOUNT = [ROWCOUNT_TEMP_ID_210271], PAGECOUNT =
[PAGECOUNT_TEMP_ID_210271]
```

What is in a Statistic Object

- Header
  - Number of rows sampled
  - Date of sampling
- Histogram
  - Used to represent the distribution and frequency of occurrence for distinct values in the table
  - Limited to 200 steps or boundary points
- Density Vector
  - Measures the uniqueness of a column or set of columns
  - The lower the density value the higher the uniqueness in the table

Why use Statistics

Accurately estimate the cardinality in the query

- SELECT * FROM TBL
  - Table level “row-count” statistics used to estimate num of rows & table size
- SELECT * FROM TBL WHERE height = 60
  - Single column statistic on height has an approximation for number of rows
- SELECT * FROM TBL WHERE height = 60 AND weight = 150
  - Multi-column statistic with histogram on height and density correlation on height and weight estimates the number of rows

How Statistics Are Used Logically

By PDWEngine on Control node

- Queries the shell database on Control node
- Estimate the cardinality of rows from the query
- Generate efficient D-SQL plans
- Minimise Movement

How Statistics Are Used Physically

By SQL Server on Compute Nodes

- Estimate the cardinality of rows from the query
- Generate efficient SMP plans

How Statistics Influence the Query Plan

The query optimizer uses statistics to make choices about the query plan.

- MPP Plan Choices
  - DMS moves: Broadcast, Shuffle, or Partition?
  - Hadoop: Map job or PDW job?
- SMP Plan Choices
  - Table operators: Index Seek or Scan?
  - JOIN operators: Hash, Merge, or Nested Loop?
  - Parallel or Serial?

Stats In Action

User Issues Query

PDW Engine

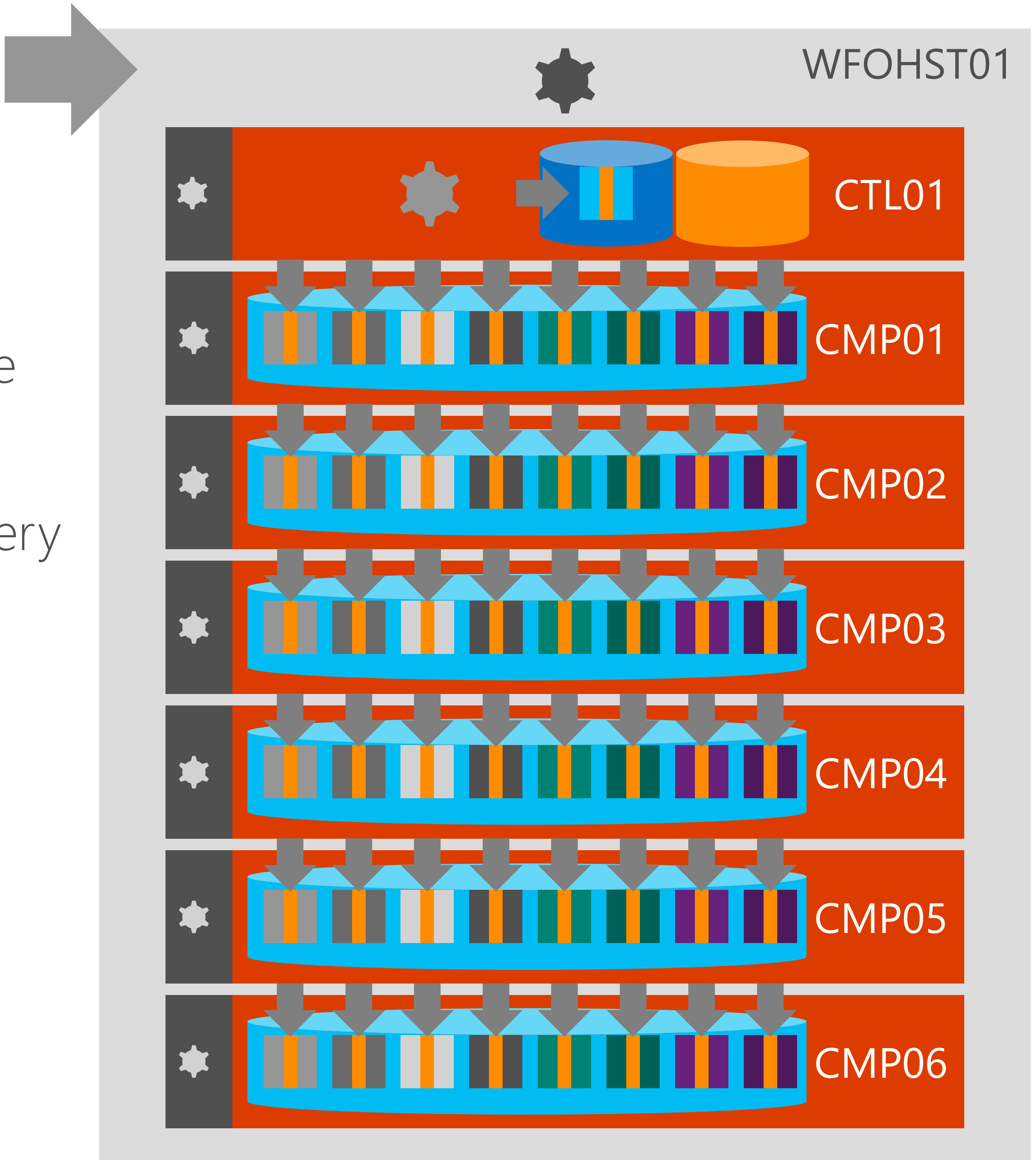
- Queries Shell database on the Control node
- Uses statistics to generate the search space
- Generates the MPP Plan
- Orchestrates a series of steps to resolve query

SQL Server

- Uses statistics held on all Compute nodes
- Generate SMP plans for each query

PDW Engine

- Returns result to user



What if Statistics are Inaccurate?

Inaccurate statistics result in bad cardinality estimates and poor query plan choices

Example: What if cardinality estimate is 1000 rows, but the actual result is 1,000,000,000 rows?

- Estimated query plan will not match the actual query execution
- Query might not complete.
- You could miss opportunities to tune the query before executing it.

Generating Statistics

Types of Statistics Objects

- User generated
- System generated

System Generated Statistics

- Not seen on Control node
- Generated on Compute nodes only
- Single column only

Not visible via

- sys.stats
- sys.stats_columns

- Generated if no statistic exists on the column
  - (AUTO_CREATE_STATISTICS ON)
  - Generated as required to improve SMP cardinality estimates
- Updated automatically
  - (AUTO_UPDATE_STATISTICS ON)
  - Standard 20% change required to trigger auto stat update
- Updated asynchronously
  - (AUTO_UPDATE_STATISTICS_ASYNC ON)
- Cannot be dropped

User Defined Statistics

- Created on the Control node <u>and</u> the Compute nodes
- Can be single or multi-column
- Visible via sys.stats / sys.stats_columns
- Can be created on a column that has a system-generated statistic
- Manually created
- Can be dropped

Updating User Defined Statistics

At the Control node

- Must be manually updated
- UPDATE STATISTICS DDL

The user defined statistic at the Control node does not receive any of the automatic updates

Only UPDATE STATISTICS will refresh the statistics held on the control node

On the Compute nodes

- Automatically & asynchronously updated
- Treated in the same way as a system generated stat

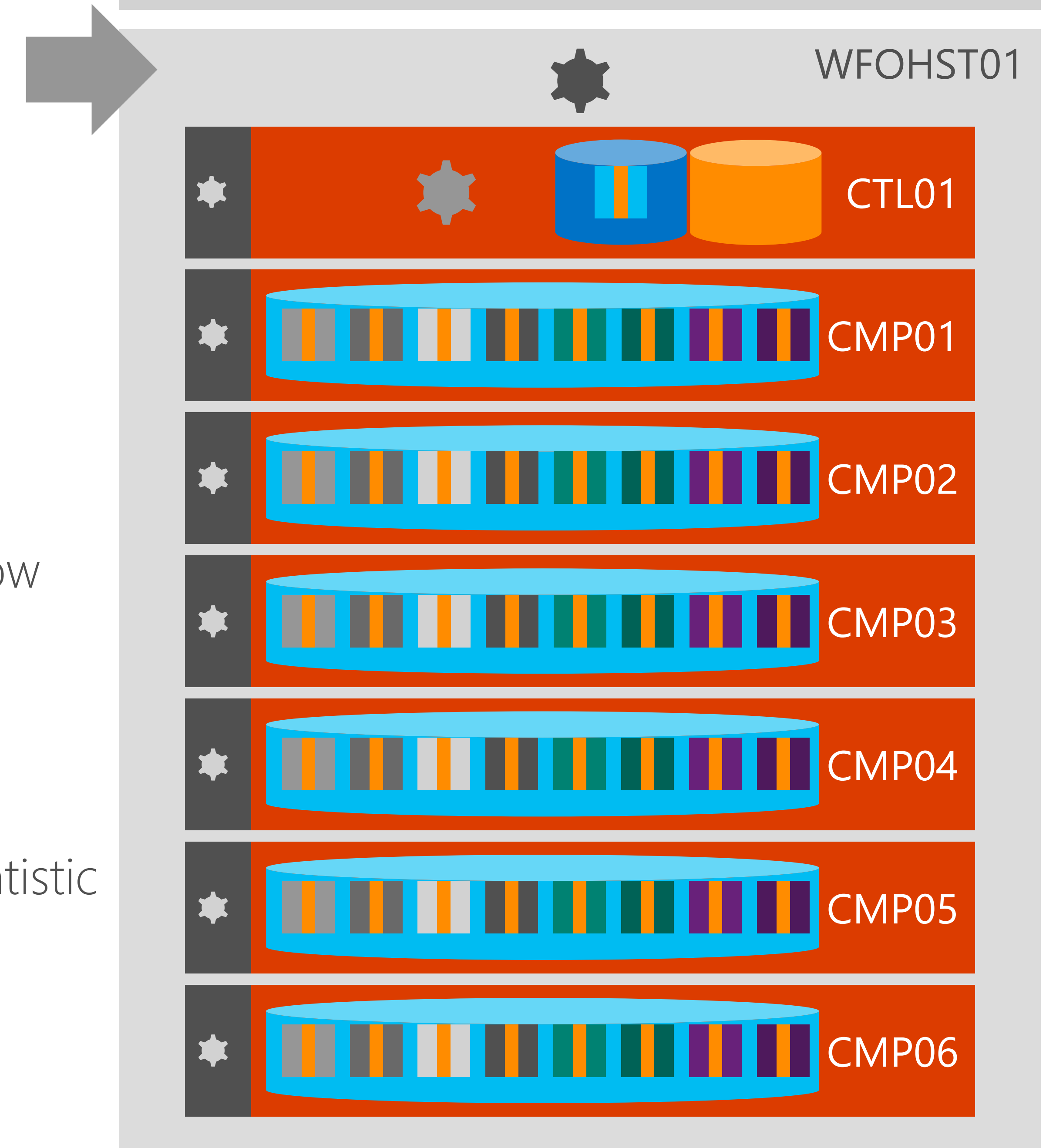
Automatically updated stats at the Compute node level only assist SMP plans; they do not improve D-SQL plans

Stats In Action

User Issues CREATE STATISTICS DDL
PDW Engine

- Checks permissions
- Creates stat on all distributions
- Retrieves table level stat info from all distributions
- Updates Control node logical table with row and page count information
- Creates statistics on Control node Shell db
- Retrieves column level stat info from distributions
- Updates Control node statistics object
- Creates extended property for physical statistic object name

```
DBCC SHOW_STATISTICS
(<table_name>,<stat_name>) WITH STATS_STREAM
```



Feature Support

Supported

- CREATE STATISTICS
- UPDATE STATISTICS
- DROP STATISTICS
- sys.stats
- sys.stats_columns
- sp_statistics

Unsupported

- Trace flags
- Legacy CE Model
- User executed DBCC SHOW_STATISTICS

Create Statistics DDL

Sample DDL

```
CREATE STATISTICS stats_name
ON
database_name.dbo.table_name
(   column_name
,   column2_name
)
WITH FULLSCAN
;
```

Unsupported Features

- Filtered statistics
- Sample size changes
- DBCC SHOW_STATISTICS

Update Statistics DDL

Sample DDL

```
UPDATE STATISTICS
database_name.dbo.table_name
(   statistics_name
|   index_name
)
WITH FULLSCAN
;
```

Unsupported Features

- Sample size changes
- NORECOMPUTE
- RESAMPLE
- ALL | COLUMNS | INDEX
- `EXEC sp_updatestats;`
- `STATS_DATE()`

Drop Statistics DDL

```
DROP STATISTICS database_name.dbo.table_name
;
```

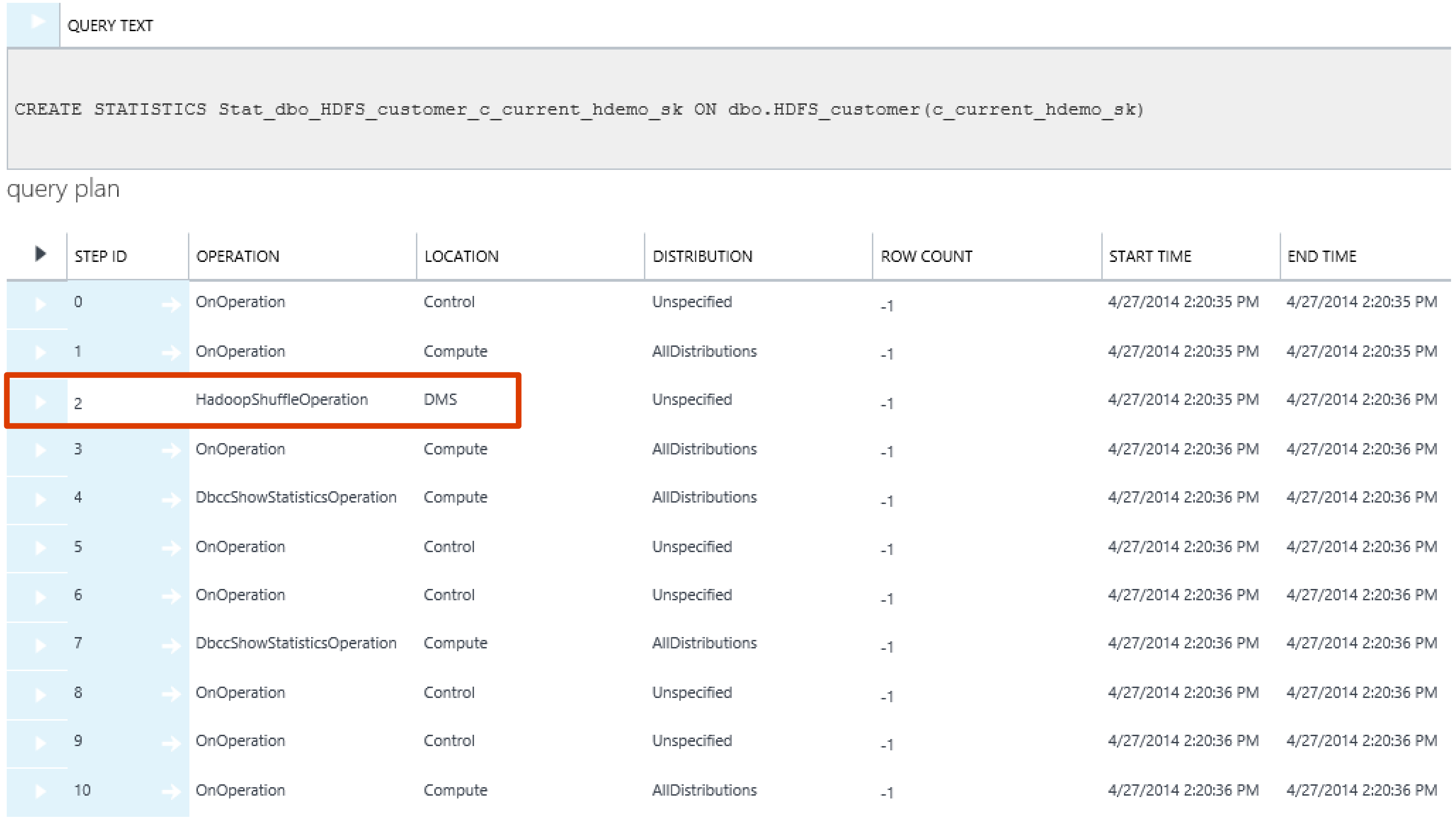
- Requires ALTER permission on the table
- Can only drop user generated stats

Statistics on External Tables – Table Level

Table level Statistics are created at creation time

- CREATE EXTERNAL TABLE (CET)
  - ROWCOUNT estimated using metadata based on ROW SIZE / FILE SIZE in Hadoop
  - PAGECOUNT is derived from estimated ROWCOUNT and ROW SIZE
- CREATE EXTERNAL TABLE AS SELECT (CETAS)
  - ROWCOUNT is known after performing the export
  - PAGECOUNT is calculated from ROWCOUNT and ROW SIZE
- CETAS more precise compared to CET

Column Statistics on External Tables

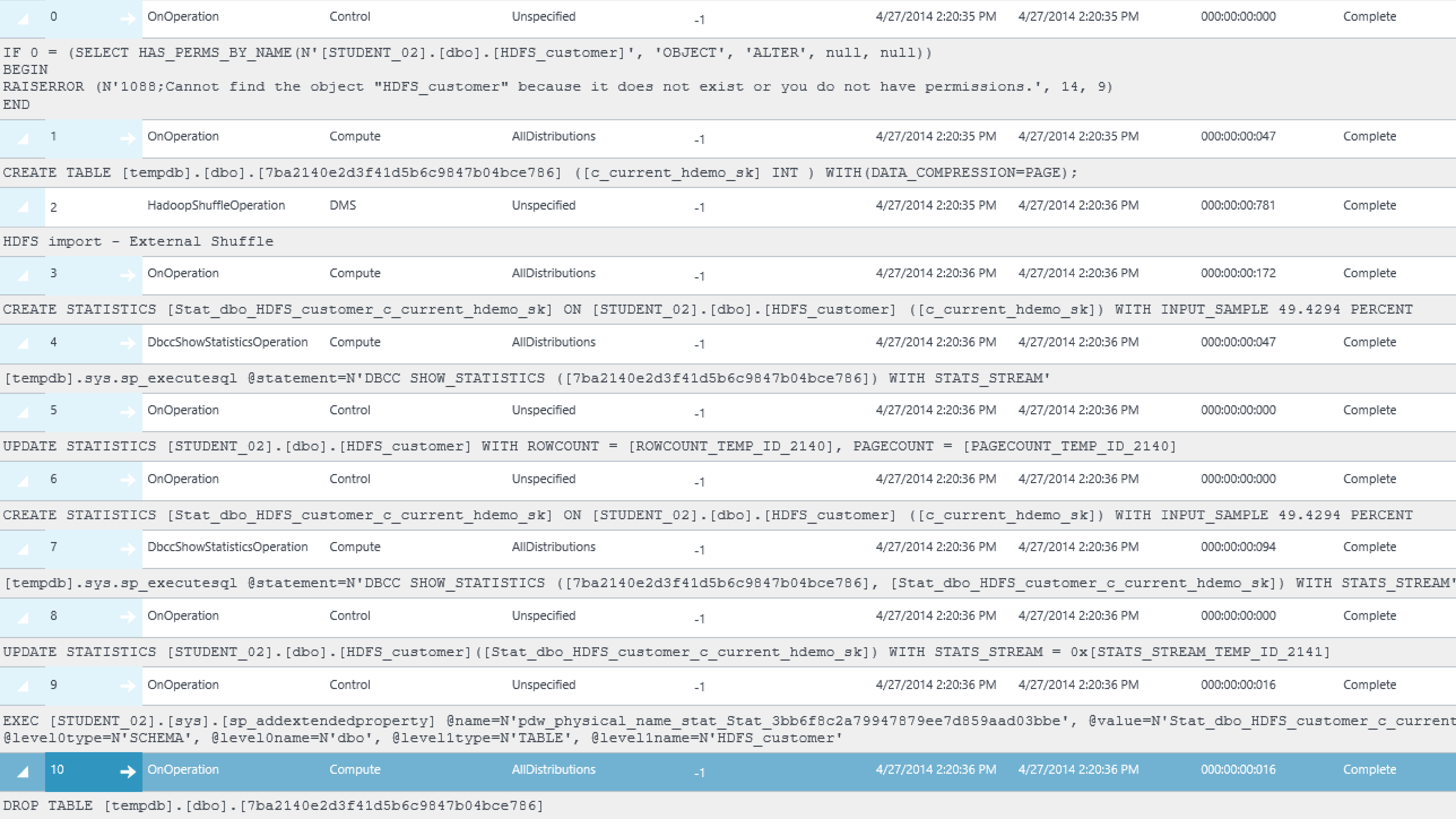


QUERY TEXT

```
CREATE STATISTICS Stat_dbo_HDFS_customer_c_current_hdemo_sk ON dbo.HDFS_customer(c_current_hdemo_sk)
```

query plan

STEP ID	OPERATION	LOCATION	DISTRIBUTION	ROW COUNT	START TIME	END TIME
0	OnOperation	Control	Unspecified	-1	4/27/2014 2:20:35 PM	4/27/2014 2:20:35 PM
1	OnOperation	Compute	AllDistributions	-1	4/27/2014 2:20:35 PM	4/27/2014 2:20:35 PM
2	HadoopShuffleOperation	DMS	Unspecified	-1	4/27/2014 2:20:35 PM	4/27/2014 2:20:36 PM
3	OnOperation	Compute	AllDistributions	-1	4/27/2014 2:20:36 PM	4/27/2014 2:20:36 PM
4	DbccShowStatisticsOperation	Compute	AllDistributions	-1	4/27/2014 2:20:36 PM	4/27/2014 2:20:36 PM
5	OnOperation	Control	Unspecified	-1	4/27/2014 2:20:36 PM	4/27/2014 2:20:36 PM
6	OnOperation	Control	Unspecified	-1	4/27/2014 2:20:36 PM	4/27/2014 2:20:36 PM
7	DbccShowStatisticsOperation	Compute	AllDistributions	-1	4/27/2014 2:20:36 PM	4/27/2014 2:20:36 PM
8	OnOperation	Control	Unspecified	-1	4/27/2014 2:20:36 PM	4/27/2014 2:20:36 PM
9	OnOperation	Control	Unspecified	-1	4/27/2014 2:20:36 PM	4/27/2014 2:20:36 PM
10	OnOperation	Compute	AllDistributions	-1	4/27/2014 2:20:36 PM	4/27/2014 2:20:36 PM



```
0   OnOperation                  Control   Unspecified       -1   4/27/2014 2:20:35 PM   4/27/2014 2:20:35 PM   000:00:00:000   Complete
IF 0 = (SELECT HAS_PERMS_BY_NAME(N'[STUDENT_02].[dbo].[HDFS_customer]', 'OBJECT', 'ALTER', null, null))
BEGIN
RAISERROR (N'1088;Cannot find the object "HDFS_customer" because it does not exist or you do not have permissions.', 14, 9)
END
1   OnOperation                  Compute   AllDistributions  -1   4/27/2014 2:20:35 PM   4/27/2014 2:20:35 PM   000:00:00:047   Complete
CREATE TABLE [tempdb].[dbo].[7ba2140e2d3f41d5b6c9847b04bce786] ([c_current_hdemo_sk] INT ) WITH(DATA_COMPRESSION=PAGE);
2   HadoopShuffleOperation       DMS       Unspecified       -1   4/27/2014 2:20:35 PM   4/27/2014 2:20:36 PM   000:00:00:781   Complete
HDFS import - External Shuffle
3   OnOperation                  Compute   AllDistributions  -1   4/27/2014 2:20:36 PM   4/27/2014 2:20:36 PM   000:00:00:172   Complete
CREATE STATISTICS [Stat_dbo_HDFS_customer_c_current_hdemo_sk] ON [STUDENT_02].[dbo].[HDFS_customer] ([c_current_hdemo_sk]) WITH INPUT_SAMPLE 49.4294 PERCENT
4   DbccShowStatisticsOperation  Compute   AllDistributions  -1   4/27/2014 2:20:36 PM   4/27/2014 2:20:36 PM   000:00:00:047   Complete
[tempdb].sys.sp_executesql @statement=N'DBCC SHOW_STATISTICS ([7ba2140e2d3f41d5b6c9847b04bce786]) WITH STATS_STREAM'
5   OnOperation                  Control   Unspecified       -1   4/27/2014 2:20:36 PM   4/27/2014 2:20:36 PM   000:00:00:000   Complete
UPDATE STATISTICS [STUDENT_02].[dbo].[HDFS_customer] WITH ROWCOUNT = [ROWCOUNT_TEMP_ID_2140], PAGECOUNT = [PAGECOUNT_TEMP_ID_2140]
6   OnOperation                  Control   Unspecified       -1   4/27/2014 2:20:36 PM   4/27/2014 2:20:36 PM   000:00:00:000   Complete
CREATE STATISTICS [Stat_dbo_HDFS_customer_c_current_hdemo_sk] ON [STUDENT_02].[dbo].[HDFS_customer] ([c_current_hdemo_sk]) WITH INPUT_SAMPLE 49.4294 PERCENT
7   DbccShowStatisticsOperation  Compute   AllDistributions  -1   4/27/2014 2:20:36 PM   4/27/2014 2:20:36 PM   000:00:00:094   Complete
[tempdb].sys.sp_executesql @statement=N'DBCC SHOW_STATISTICS ([7ba2140e2d3f41d5b6c9847b04bce786], [Stat_dbo_HDFS_customer_c_current_hdemo_sk]) WITH STATS_STREAM'
8   OnOperation                  Control   Unspecified       -1   4/27/2014 2:20:36 PM   4/27/2014 2:20:36 PM   000:00:00:000   Complete
UPDATE STATISTICS [STUDENT_02].[dbo].[HDFS_customer]([Stat_dbo_HDFS_customer_c_current_hdemo_sk]) WITH STATS_STREAM = 0x[STATS_STREAM_TEMP_ID_2141]
9   OnOperation                  Control   Unspecified       -1   4/27/2014 2:20:36 PM   4/27/2014 2:20:36 PM   000:00:00:016   Complete
EXEC [STUDENT_02].[sys].[sp_addextendedproperty] @name=N'pdw_physical_name_stat_Stat_3bb6f8c2a79947879ee7d859aad03bbe', @value=N'Stat_dbo_HDFS_customer_c_current
@level0type=N'SCHEMA', @level0name=N'dbo', @level1type=N'TABLE', @level1name=N'HDFS_customer'
10  OnOperation                  Compute   AllDistributions  -1   4/27/2014 2:20:36 PM   4/27/2014 2:20:36 PM   000:00:00:016   Complete
DROP TABLE [tempdb].[dbo].[7ba2140e2d3f41d5b6c9847b04bce786]
```

Managing Statistics

Updating All Statistics on a Table

UPDATE STATISTICS <TableName>

- Easy to implement BUT is a serial operation
- Updates each statistic on a table in series
- Includes all system generated statistics

Updating Named Statistics

UPDATE STATISTICS <Table_name>(<stat_name>)

- Fine grained control
- Minimizes duration of update
- More onerous to implement
- Only updates user-defined statistics

Creating Stats – Best Practices

- Create stats immediately after table creation
- Avoids system generated stats being created
- Maximises control
- Maximises consistency and predictability

Creating Stats – Opportunity Cost

If you don't create stats straight away…

- SQL Server may generate system statistics objects
  - Satisfy any queries fired at tables in the interim
- You can end up with a user generated stat and a system generated stat on the same column
  - CREATE STATISTICS will create another named statistic for a column even if a system generated stat already exists
- You won't know which statistic is used to resolve the SMP query
  - It will depend on which is the freshest stat but that could be different for each distribution
- UPDATE STATISTICS <Table_name> updates statistics in series
  - Worst case (all columns having both system and user generated statistics) this process will now take twice as long
- You cannot delete the system generated statistic
  - To get rid of system generated stats you must CTAS to a new table

Statistics Strategies – Single Column

Optimistic

Used for controlled workloads

Cover columns used by following clauses

- WHERE
- JOIN
- GROUP BY
- ORDER BY
- DISTINCT

Pessimistic

Used for ad-hoc workloads

One stat object on every column

Statistics Strategies – Multi-Column

Good For

- Composite Joins
  - equality join predicates only
  - Avoids nested loop plans on SMP
- Composite Group By
  - Improves Cardinality Estimate of Aggregation
- Distinct Count
  - Reduces Cardinality Estimate for large distinct counts

Not Used for

- Composite filters

Even when the filters are on the same table the new CE model will not use a multi-column stat.

Already being considered for v.Next

Statistics Management

- Statistics need to be pro-actively managed
- Common requirement to extend ELT frameworks to cover this
- Remember to include creation of stats for the ELT process itself
    - Staging tables
    - Temporary tables

Create “Missing” Statistics

```
WITH T
AS
(
SELECT    t.name                    AS Table_name
,         s.name                    AS [Table_Schema_name]
,         c.name                    AS [Column_name]
FROM      sys.tables t
JOIN      sys.schemas s                 ON  t.Schema_ID = s.Schema_ID
JOIN      sys.columns c                 ON  t.Object_ID = c.Object_ID
LEFT JOIN sys.stats_columns l           ON  l.Object_ID = c.Object_ID
                                        AND l.Column_ID = c.Column_ID
                                        AND l.stats_column_ID = 1

WHERE  l.object_ID IS NULL
)
SELECT 'CREATE STATISTICS Stat_'+Table_Schema_name+'_'+Table_Name+'_'+Column_name+' ON
'+Table_Schema_name+'.'+Table_Name+'('+Column_name+')' AS CreateStat
,      'CREATE STATISTICS Stat_'+Table_Schema_name+'_'+Table_Name+'_'+Column_name+' ON
'+Table_Schema_name+'.'+Table_Name+'('+Column_name+') WITH FULLSCAN' AS CreateStatFull
FROM T
```

Microsoft