**We are going to discuess about below concept:**

**Compare storage requirements between optimal and sub-optimal column data types**

Perform the following steps to compare storage requirements between optimal and suboptimal column data types

1. Use the following query to create two tables (Sale\_Hash\_Projection and Sale\_Hash\_Projection2) which contain a subset of the columns from Sale\_Heap:

CREATE TABLE [wwi\_perf].[Sale\_Hash\_Projection]

WITH

(

DISTRIBUTION = HASH ( [CustomerId] ),

HEAP

)

AS

SELECT

[CustomerId]

,[ProductId]

,[Quantity]

FROM

[wwi\_perf].[Sale\_Heap]

CREATE TABLE [wwi\_perf].[Sale\_Hash\_Projection2]

WITH

(

DISTRIBUTION = HASH ( [CustomerId] ),

CLUSTERED COLUMNSTORE INDEX

)

AS

SELECT

[CustomerId]

,[ProductId]

,[Quantity]

FROM

[wwi\_perf].[Sale\_Heap]

The query should finish execution in a few minutes.

1. Use the following query to create two additional tables (Sale\_Hash\_Projection\_Big and Sale\_Hash\_Projection\_Big2) that have the same columns, but with different (sub\_optimal) data types:

CREATE TABLE [wwi\_perf].[Sale\_Hash\_Projection\_Big]

WITH

(

DISTRIBUTION = HASH ( [CustomerId] ),

HEAP

)

AS

SELECT

[CustomerId]

,CAST([ProductId] as bigint) as [ProductId]

,CAST([Quantity] as bigint) as [Quantity]

FROM

[wwi\_perf].[Sale\_Heap]

CREATE TABLE [wwi\_perf].[Sale\_Hash\_Projection\_Big2]

WITH

(

DISTRIBUTION = HASH ( [CustomerId] ),

CLUSTERED COLUMNSTORE INDEX

)

AS

SELECT

[CustomerId]

,CAST([ProductId] as bigint) as [ProductId]

,CAST([Quantity] as bigint) as [Quantity]

FROM

[wwi\_perf].[Sale\_Heap]

1. Verify that the four tables have the same number of rows (there should be 339,507,246 rows in each):

SELECT 'Sale\_Hash\_Projection', COUNT\_BIG(\*) FROM [wwi\_perf].[Sale\_Hash\_Projection]

UNION

SELECT 'Sale\_Hash\_Projection2', COUNT\_BIG(\*) FROM [wwi\_perf].[Sale\_Hash\_Projection2]

UNION

SELECT 'Sale\_Hash\_Projection\_Big', COUNT\_BIG(\*) FROM [wwi\_perf].[Sale\_Hash\_Projection\_Big]

UNION

SELECT 'Sale\_Hash\_Projection\_Big2', COUNT\_BIG(\*) FROM [wwi\_perf].[Sale\_Hash\_Projection\_Big2]

1. Run the following query to compare the storage requirements for the three tables:

SELECT

database\_name

, schema\_name

, table\_name

, distribution\_policy\_name

, distribution\_column

, index\_type\_desc

, COUNT(distinct partition\_nmbr) as nbr\_partitions

, SUM(row\_count) as table\_row\_count

, SUM(reserved\_space\_GB) as table\_reserved\_space\_GB

, SUM(data\_space\_GB) as table\_data\_space\_GB

, SUM(index\_space\_GB) as table\_index\_space\_GB

, SUM(unused\_space\_GB) as table\_unused\_space\_GB

FROM

[wwi\_perf].[vTableSizes]

WHERE

schema\_name = 'wwi\_perf'

and table\_name in ('Sale\_Hash\_Projection', 'Sale\_Hash\_Projection2',

'Sale\_Hash\_Projection\_Big', 'Sale\_Hash\_Projection\_Big2')

GROUP BY

database\_name

, schema\_name

, table\_name

, distribution\_policy\_name

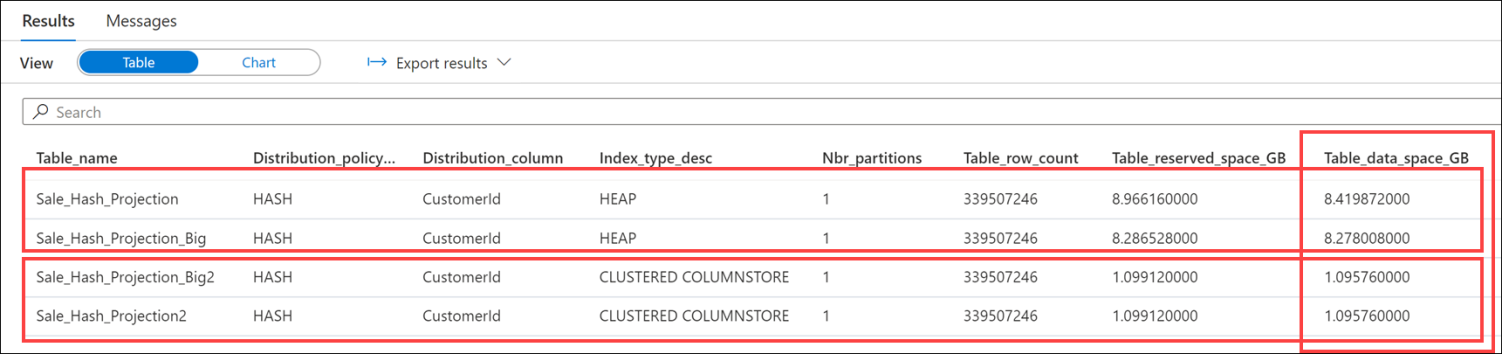
, distribution\_column

, index\_type\_desc

ORDER BY

table\_reserved\_space\_GB desc

1. Analyze the results:



There are two important conclusions to draw here:

* + In the case of HEAP tables, the storage impact of using BIGINT instead of SMALLINT(for ProductId) and TINYINT (for QUANTITY) is 0.141864 GB. We're talking here about only two columns and a moderate number of rows (2.9 billion).
  + Even in the case of CLUSTERED COLUMNSTORE tables, where compression will offset some of the differences, there is still a difference of 12.7 MB.