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

Note:

You are not required to complete the processes, tasks, activities, or steps presented in this example. The various samples provided are for illustrative purposes only and it's likely that if you try this out you will encounter issues in your system.

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.

2. 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]
```

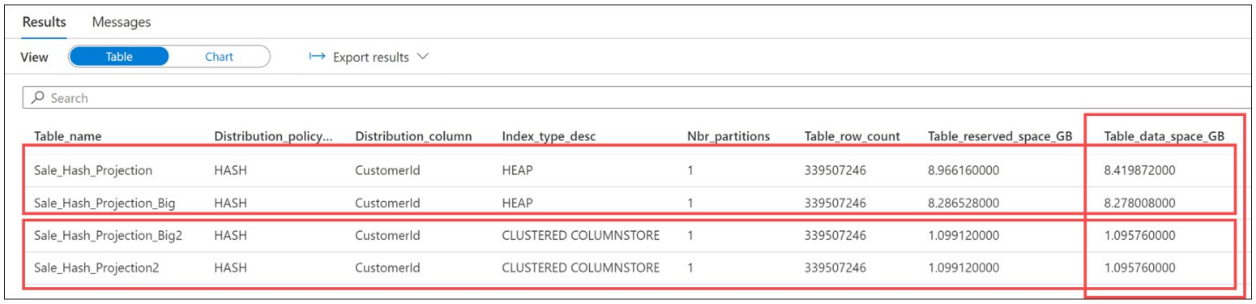
3. Verify that the four tables have the same number of rows (there should be 339,507,246 rows in each):

```
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]
```

4. 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
```

5. Analyze the results:



Table_name	Distribution_policy...	Distribution_column	Index_type_desc	Nbr_partitions	Table_row_count	Table_reserved_space_GB	Table_data_space_GB
Sale_Hash_Projection	HASH	CustomerId	HEAP	1	339507246	8.966160000	8.419872000
Sale_Hash_Projection_Big	HASH	CustomerId	HEAP	1	339507246	8.286528000	8.278008000
Sale_Hash_Projection_Big2	HASH	CustomerId	CLUSTERED COLUMNSTORE	1	339507246	1.099120000	1.095760000
Sale_Hash_Projection2	HASH	CustomerId	CLUSTERED COLUMNSTORE	1	339507246	1.099120000	1.095760000

Data type selection impact on table storage

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 almost 1 GB (0.8941 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.