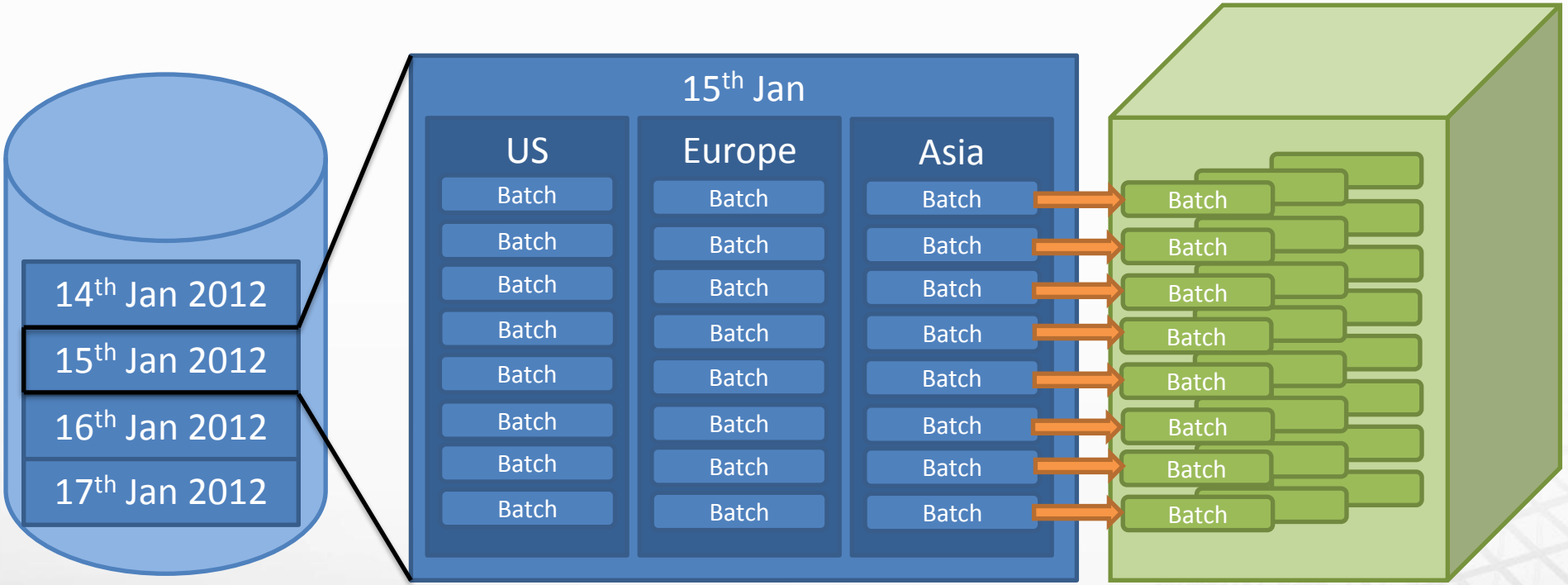
ROLAP with Column Store Index – Deep Dive

Alexei Khalyako
SQL CAT Program Manager
alexeik@microsoft.com

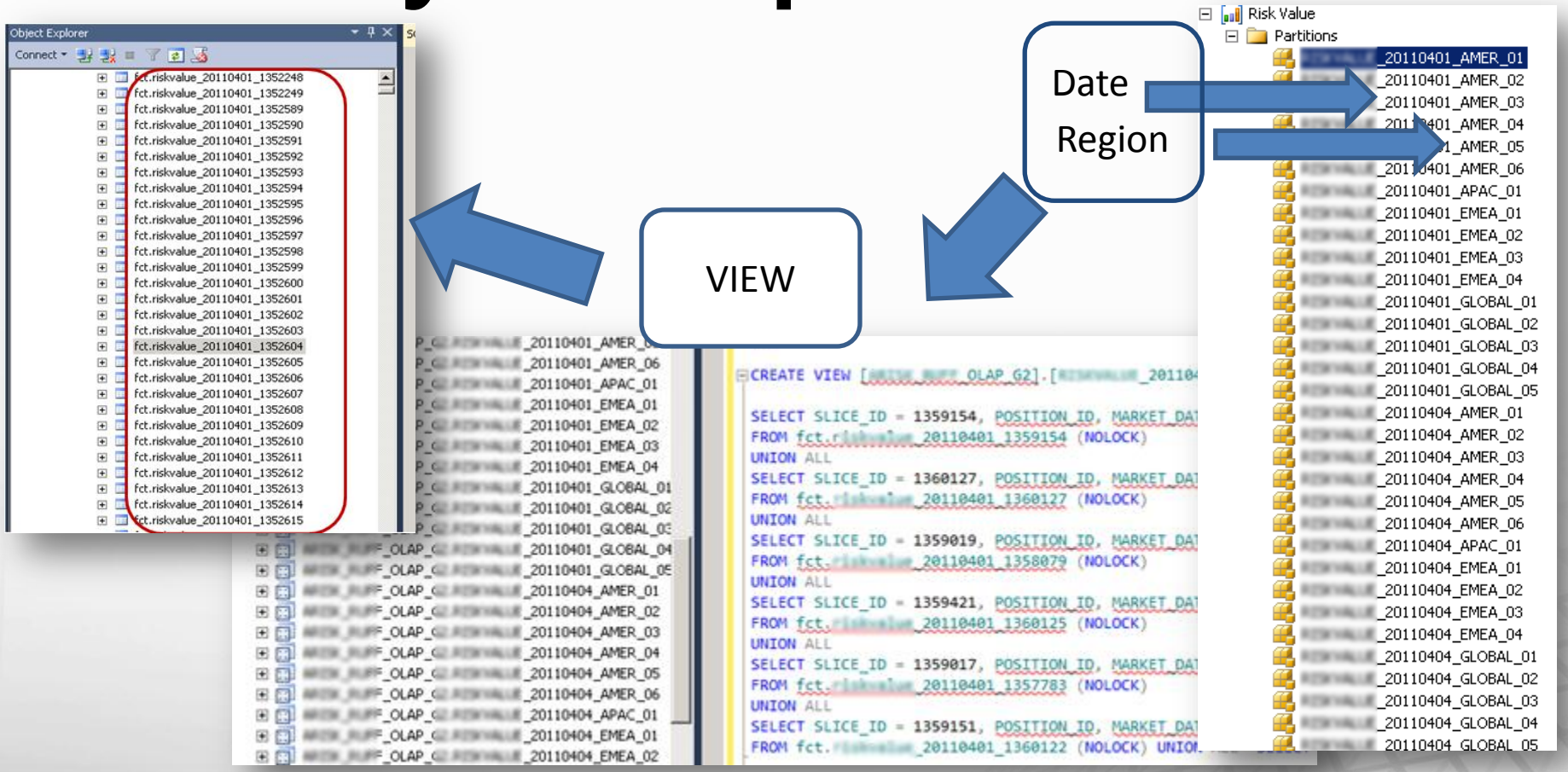
What are we doing *now*?

- ❑1000 of concurrent users
- ❑ 4TB Cube
- ❑High speed loading

Current design



Physical implementation



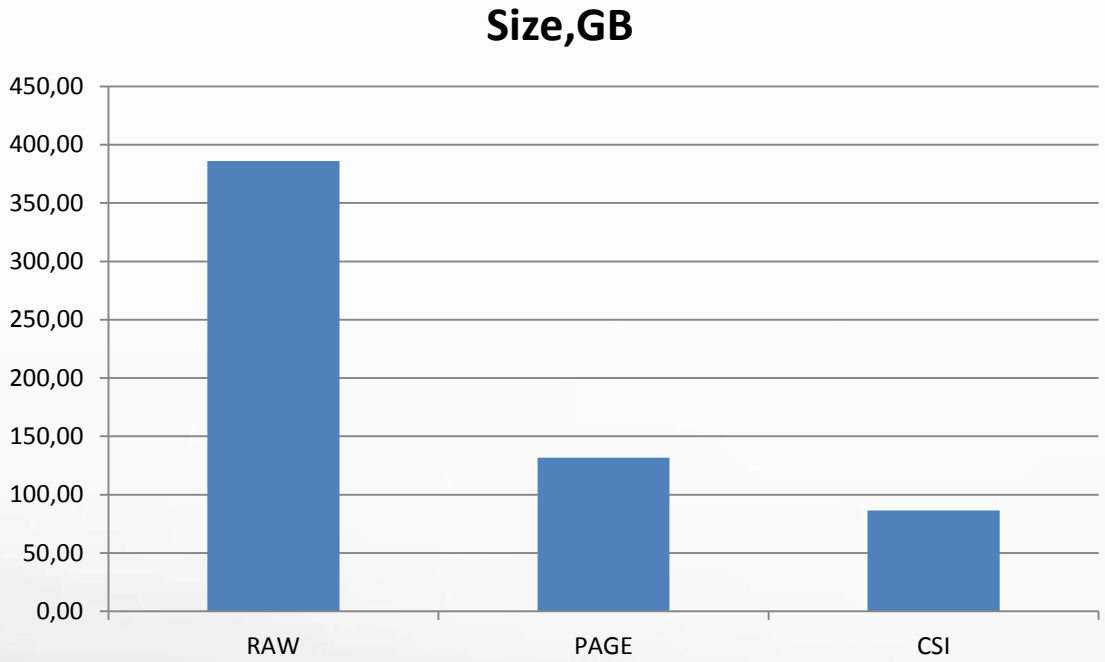
MOLAP

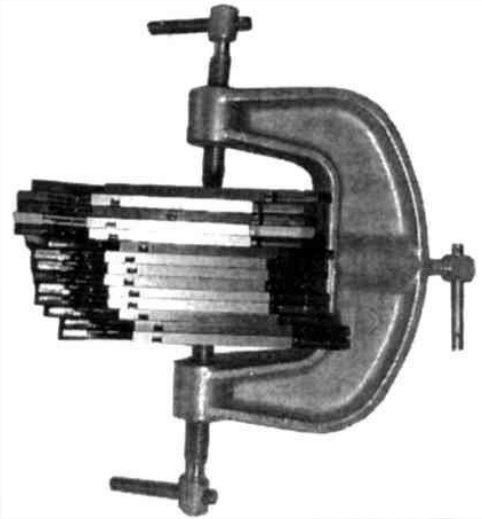
- Move data 2x times
- Partitioning complex
- Bitmap Index

ROLAP

- Data loaded one time
- Partitioning simple
- Column Store Index

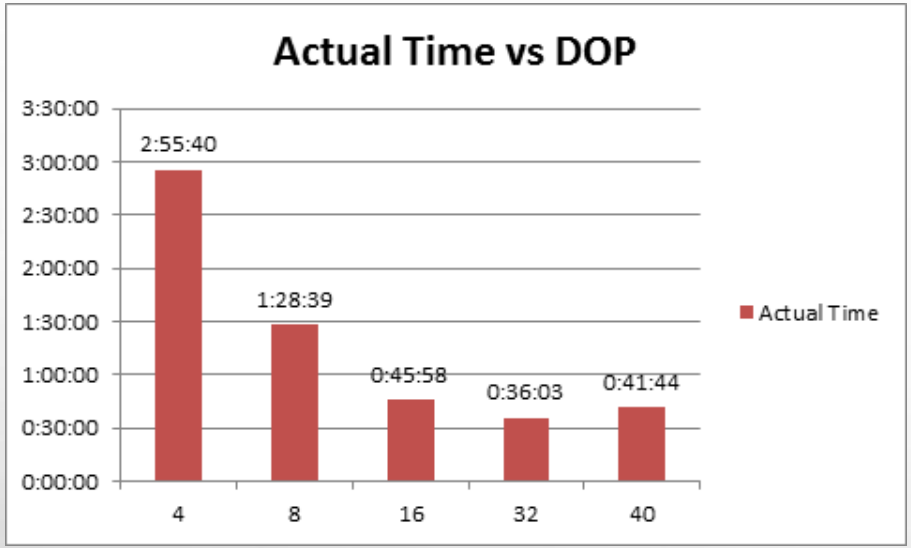
Good to know: Compression

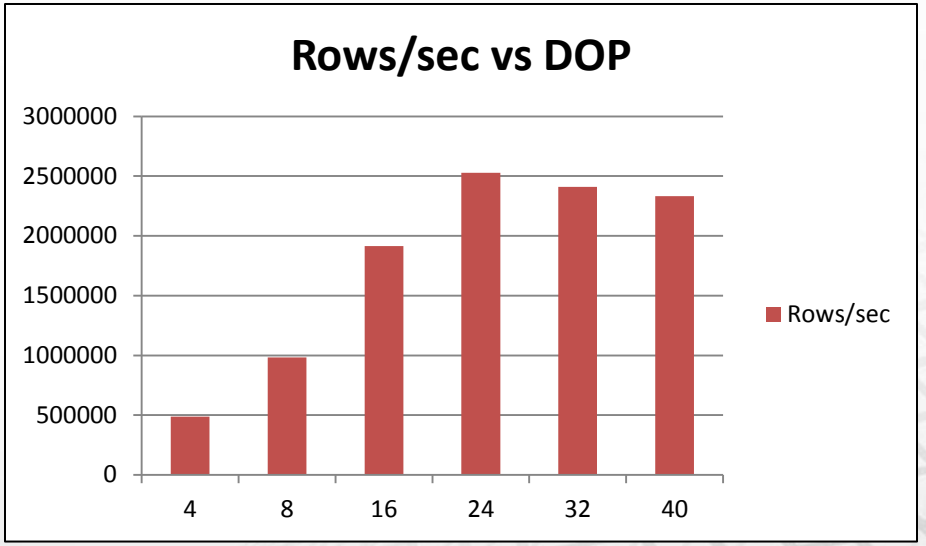




Good to know: Scalability of Index Build

- Index building time scales well up to 32 cores
- Reached 2,5 million rows/sec



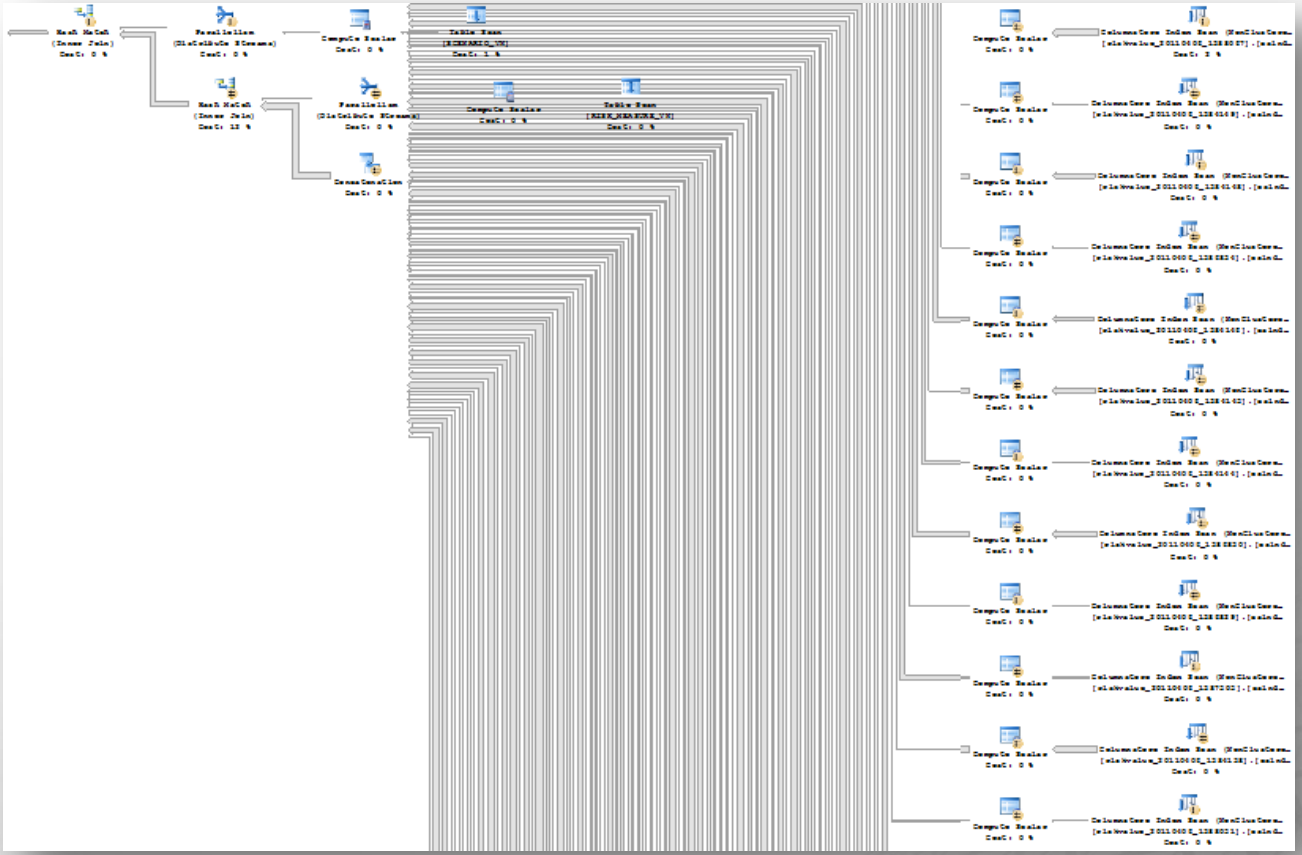


It should just work

Let’s do the *dumb* thing:

- one-to-one conversion

Duh!



IS there a better way?

MOLAP

1 Table Per Day

… Per region

…. Per Slice

55 K fact tables

Remove table to delete

ROLAP

1 Partition Per day

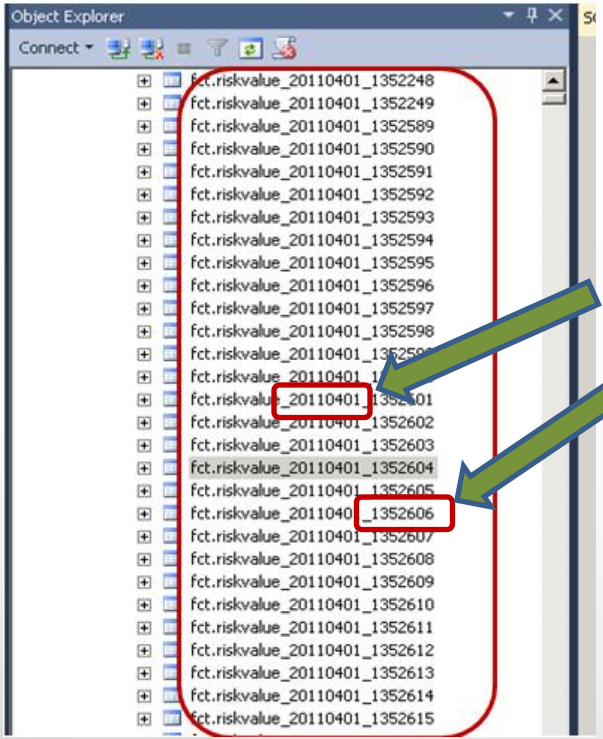
… 8 ‘hash’ tables for the fact

8 Partitioned Fact tables

SWITCH to delete

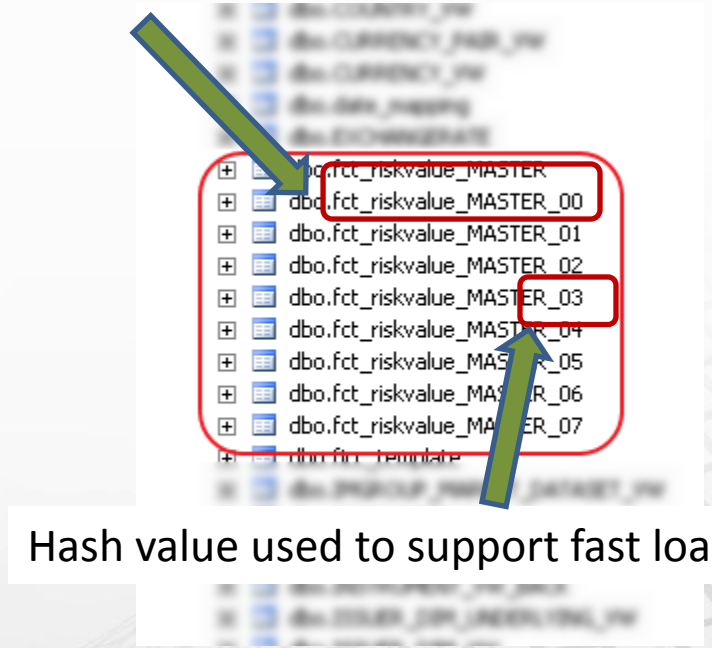
New partitioning

Before



After

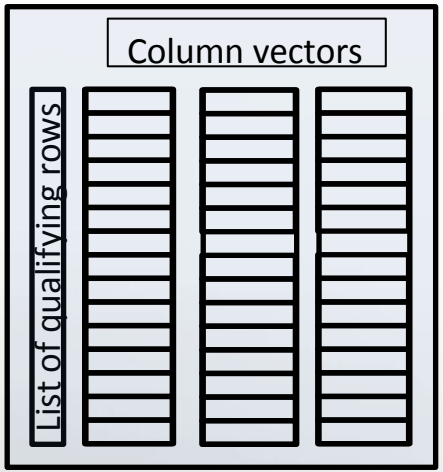
One table partitioned by ‘day’



“All we need is.. “ BATCH

.. Or ‘vector’ processing

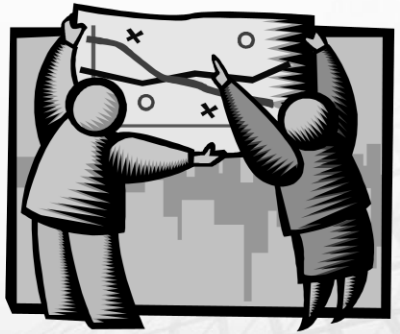
Batch object



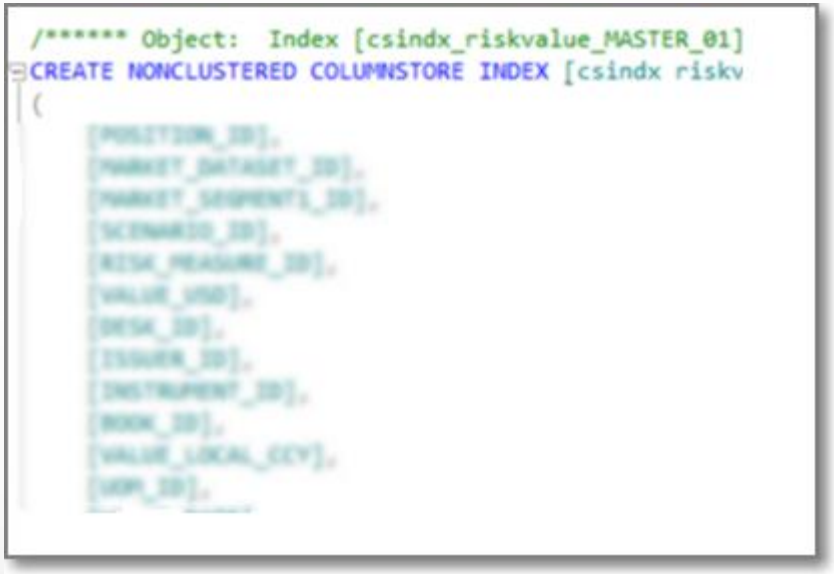
- Process a batch of rows at a time
  - Batch stored in vector form
- Vector mode operators implemented
  - Filter, hash join, hash aggregation
- Greatly reduced CPU time

Indexing strategy

- We build a column store index on the fact table
- Dimensions stay HEAP

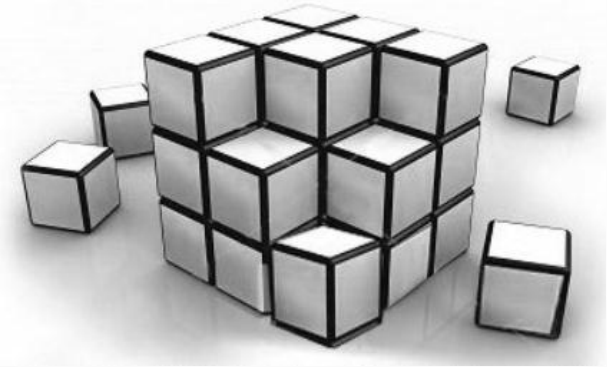


Indexing Fact table



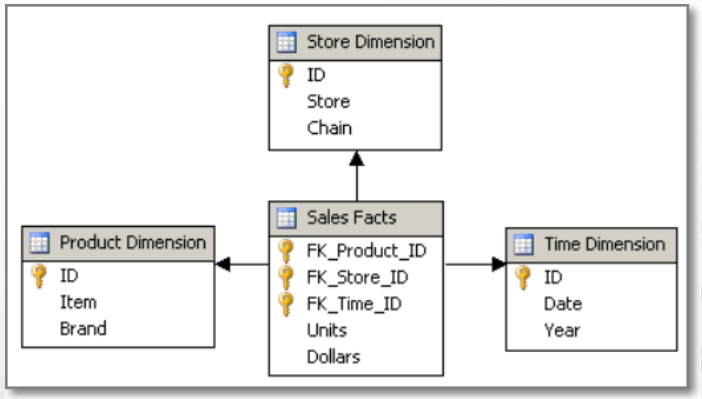
- Include ***all*** columns
- Do not forget to align with partition function

Small mistakes matter

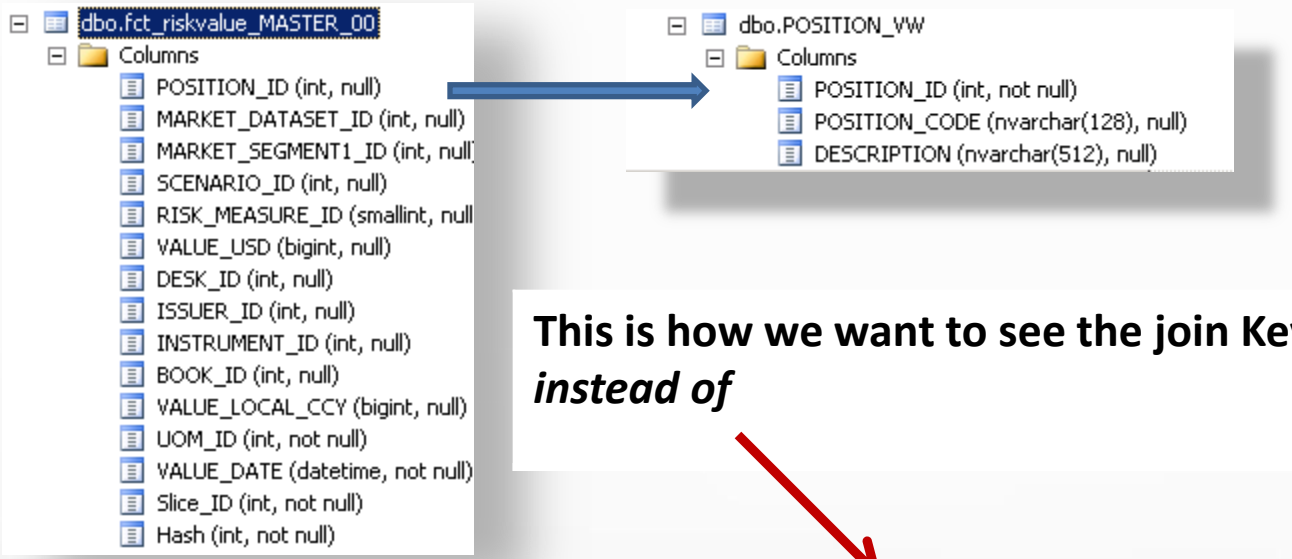


MOLAP developer :
”It is not very accurate, BUT! It is not big problem, that sometimes join columns data types do not match if this gives right results’

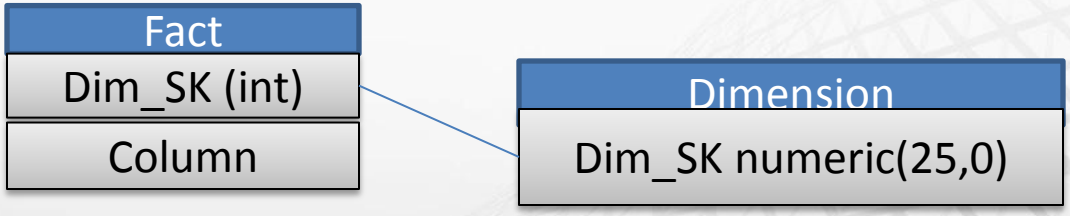
Relational guy:
”Well, this is not Kimball, this burns CPU, query will suffer from that..’



Join Key

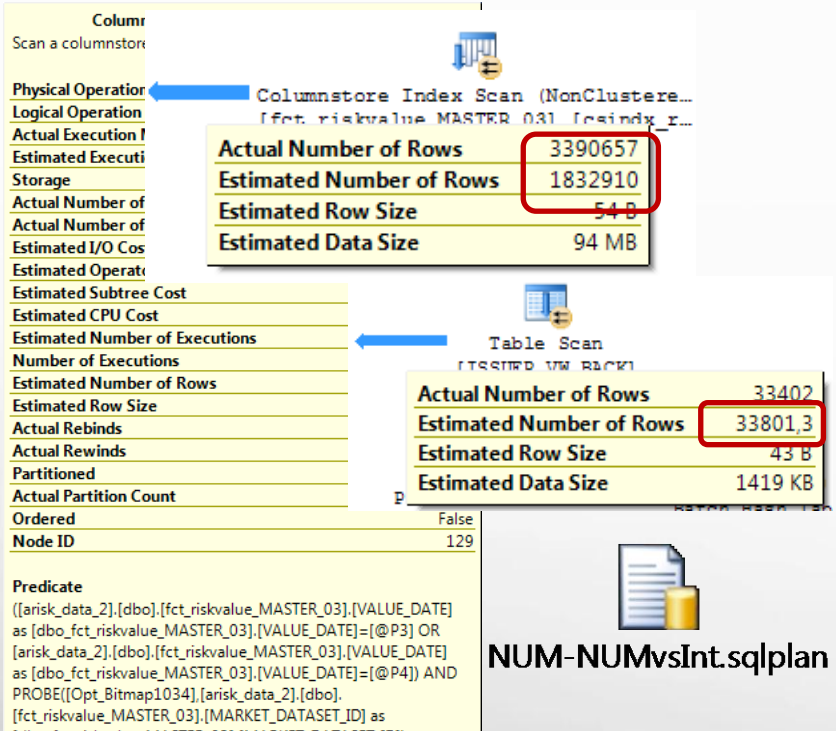


This is how we want to see the join Key to look a like ,
instead of



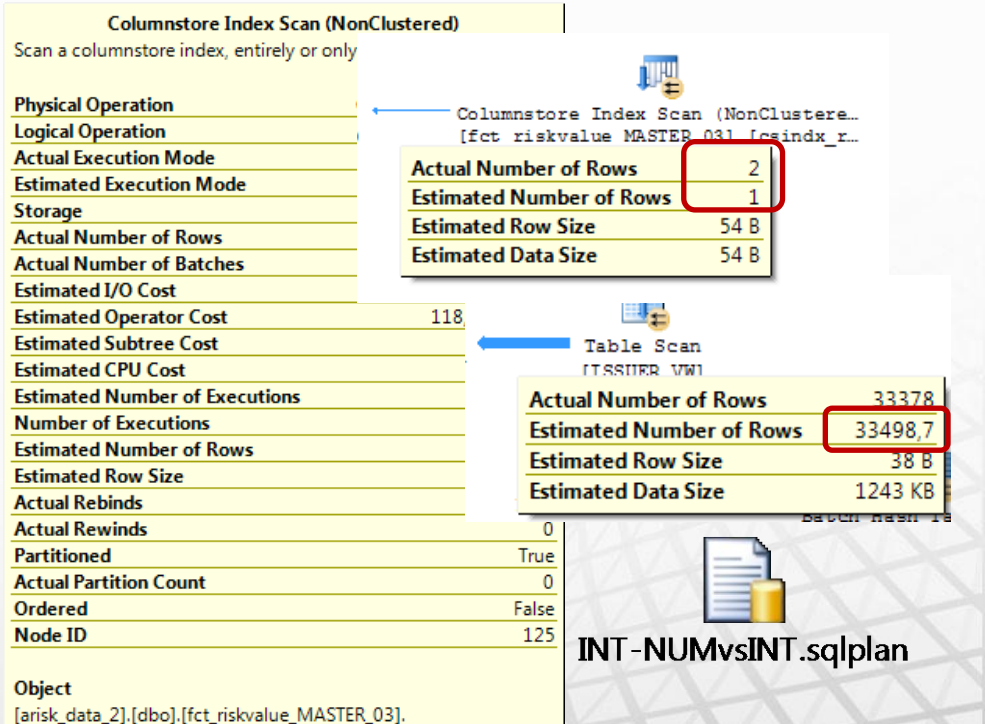
INT/ NUMERIC(18,0)

- Runtime: 20 sec
- Execution mode: BATCH



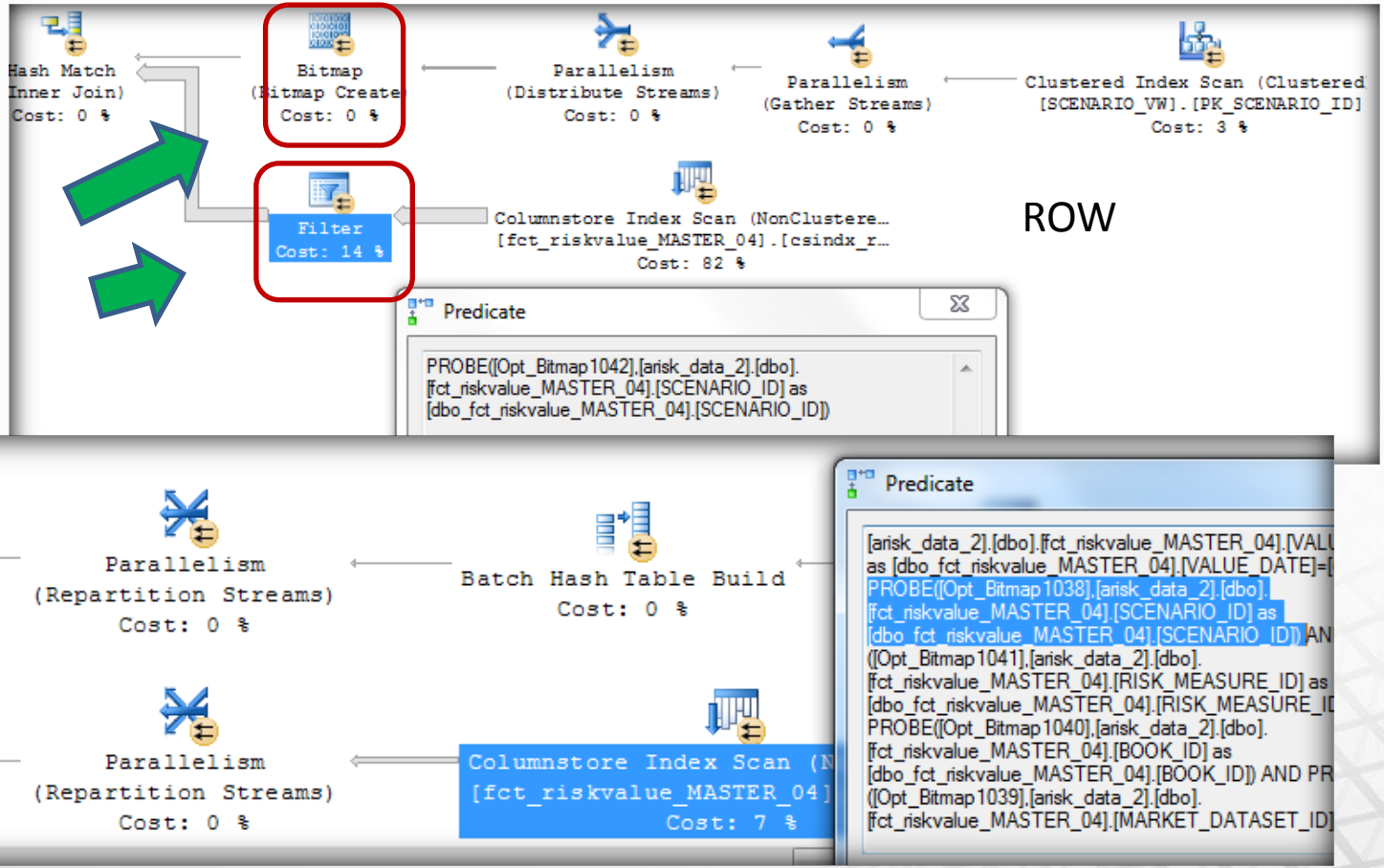
INT/INT

- Runtime:1 sec
- Execution mode: BATCH

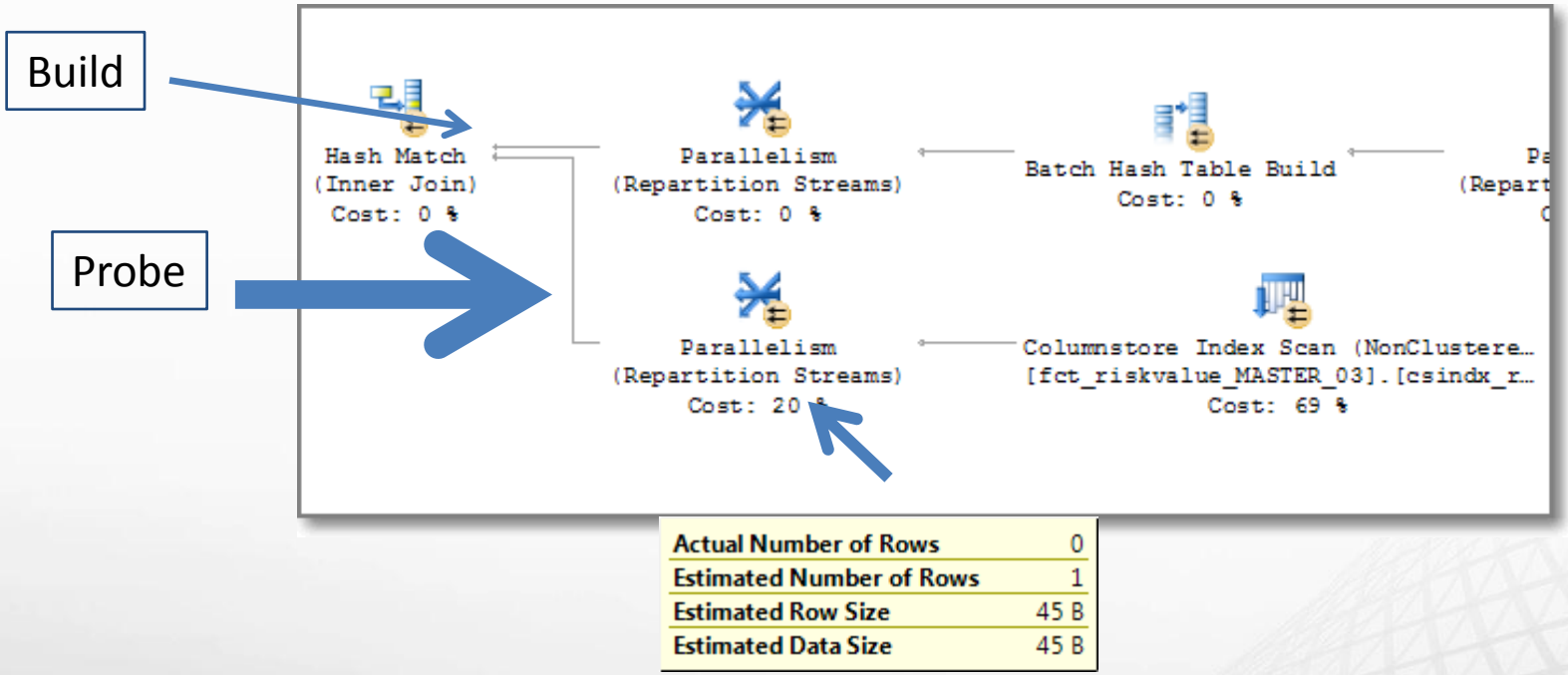


Beware of row execution

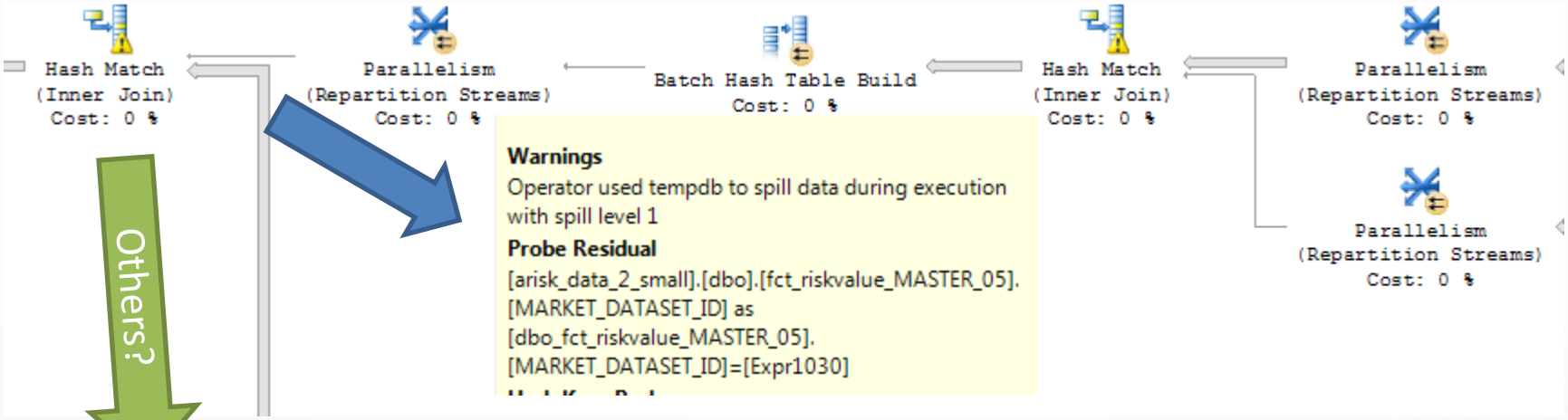
Row execution indicator !



Recap: Hash Joins



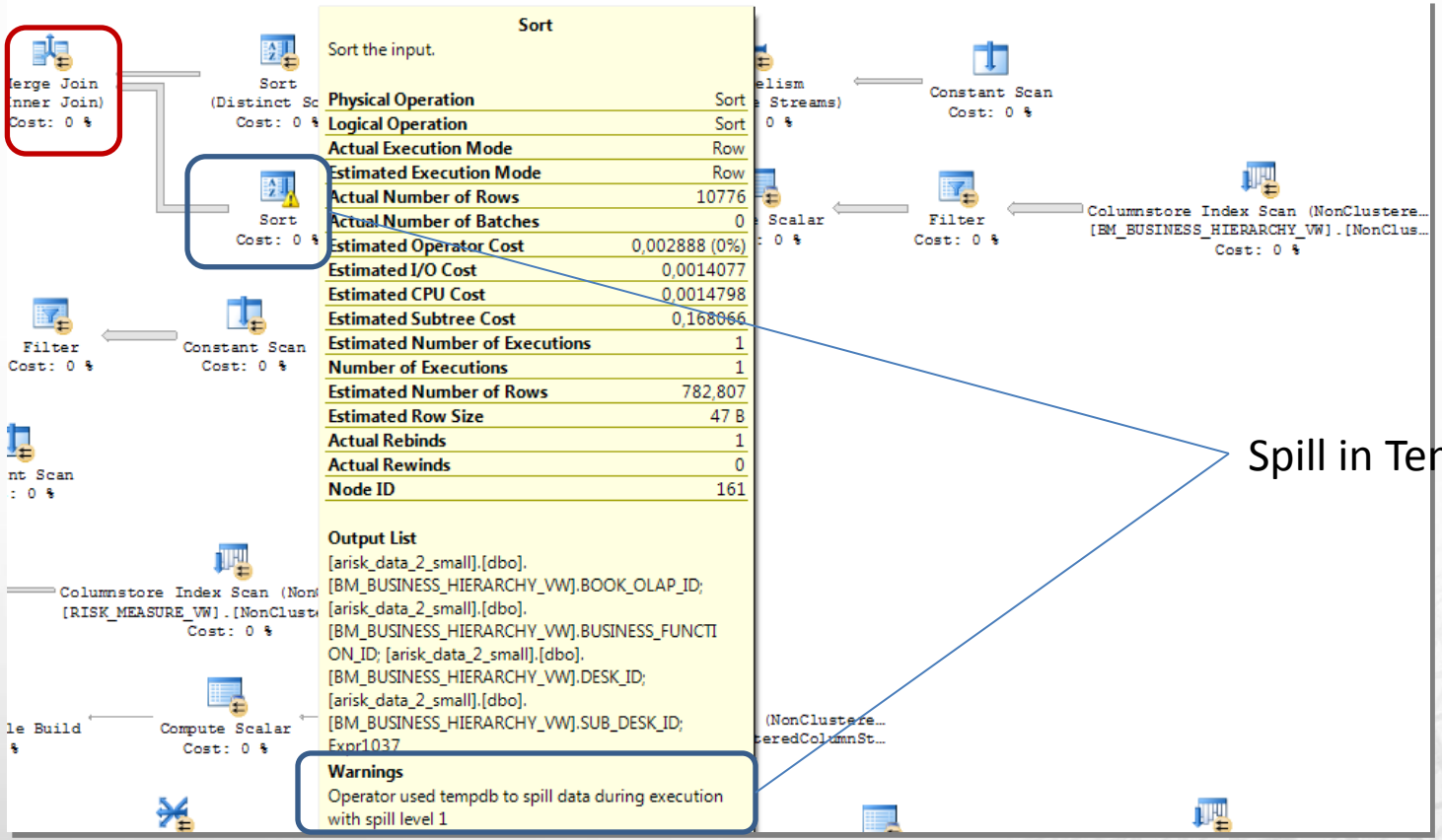
Spill on the Hash Build Side




```
SELECT session_id,requested_memory_kb
,granted_memory_kb,used_memory_kb
from sys.dm_exec_query_memory_grants
```

session_id	requested_memory_kb	granted_memory_kb	used_memory_kb
51	438536	~~438536~~	42527248

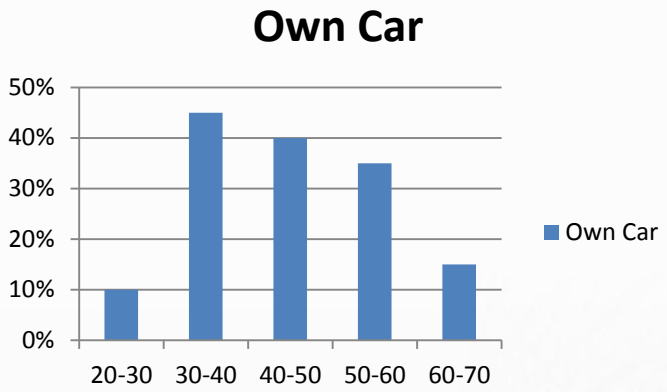
Spill on the Dim Search Side

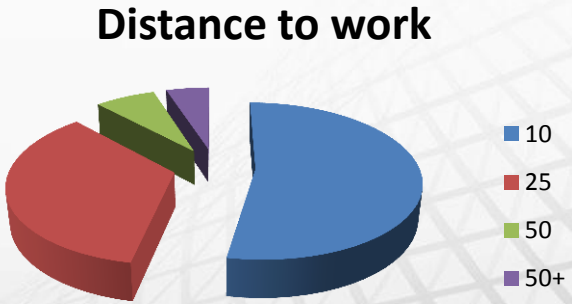


Spill in TempDB

Young people do not have cars

```
SELECT <columns> From Customer_Dim
WHERE
[Distince_To_Work] BETWEEN 10 and 20
AND
[Age] BETWEEN 25 and  30
And
[Private_Car_Owns] ='Yes'
```





We can help SQL

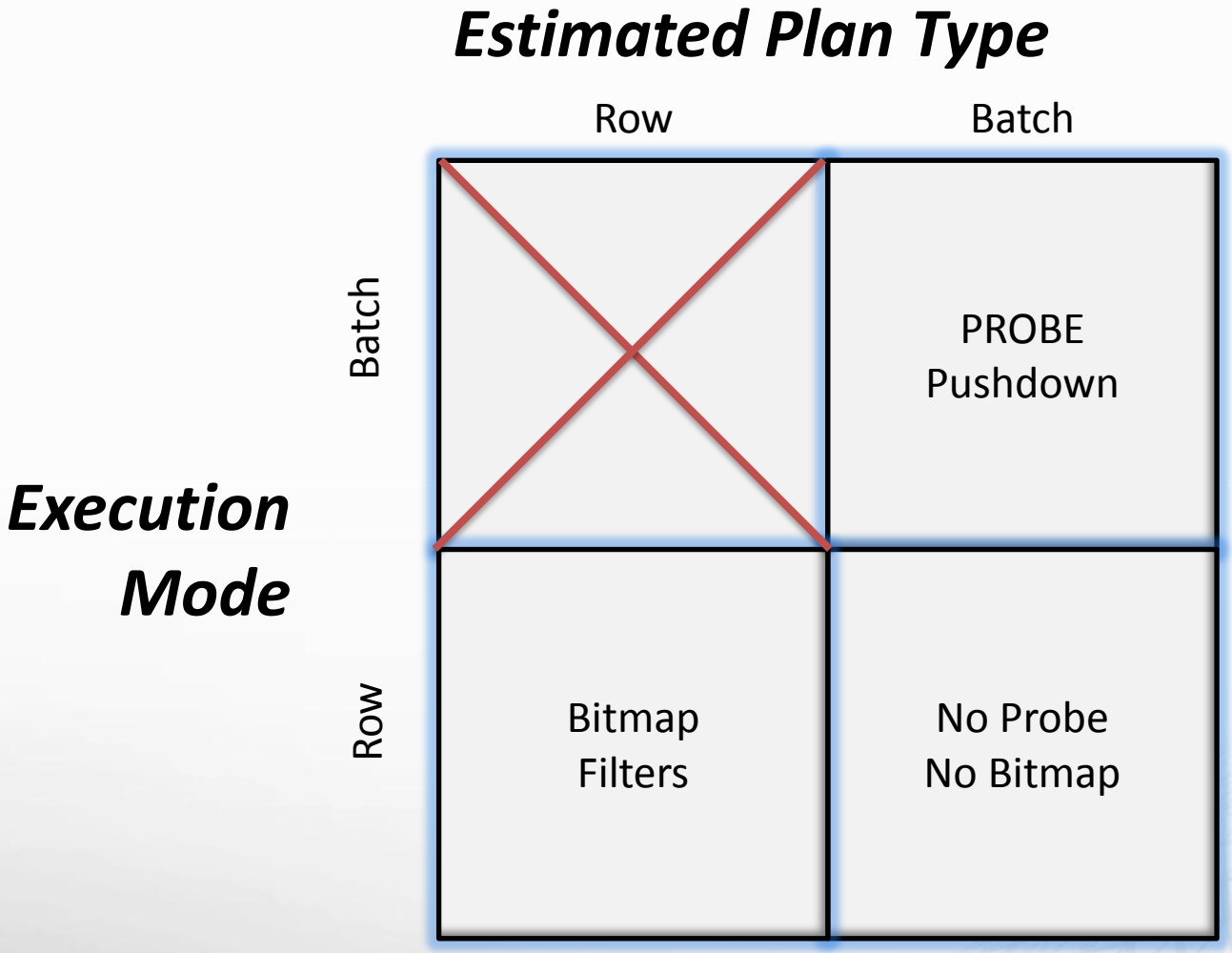
- Create filtered composite statistics

```
Create statistics YoungCarOwners
On Customer_Dim
      ([Distince_To_Work],
      [Age],
      [Private_Car_Owns])
Where
      [Age]> 25 and [Age] <30
      And [Private_Car_Owns] ='Yes'
```

Composite statistics in action

- Added two *filtered* stats on one dim

query_id	test_name	time/sec	test_name	time/sec	% Improvemnt
15	ROLAP_NO_CUSTOM_STAT	103,78	ROLAP_WITH_CUSTOM_STAT	48,69	53,08%
99	ROLAP_NO_CUSTOM_STAT	84,09	ROLAP_WITH_CUSTOM_STAT	40,36	52,00%
24	ROLAP_NO_CUSTOM_STAT	94,93	ROLAP_WITH_CUSTOM_STAT	51,87	45,36%
72	ROLAP_NO_CUSTOM_STAT	6901,83	ROLAP_WITH_CUSTOM_STAT	4401,19	36,23%
9	ROLAP_NO_CUSTOM_STAT	4623,60	ROLAP_WITH_CUSTOM_STAT	3095,50	33,05%
6	ROLAP_NO_CUSTOM_STAT	157,78	ROLAP_WITH_CUSTOM_STAT	122,13	22,59%
45	ROLAP_NO_CUSTOM_STAT	7882,40	ROLAP_WITH_CUSTOM_STAT	6186,38	21,52%
93	ROLAP_NO_CUSTOM_STAT	944,48	ROLAP_WITH_CUSTOM_STAT	753,84	20,18%

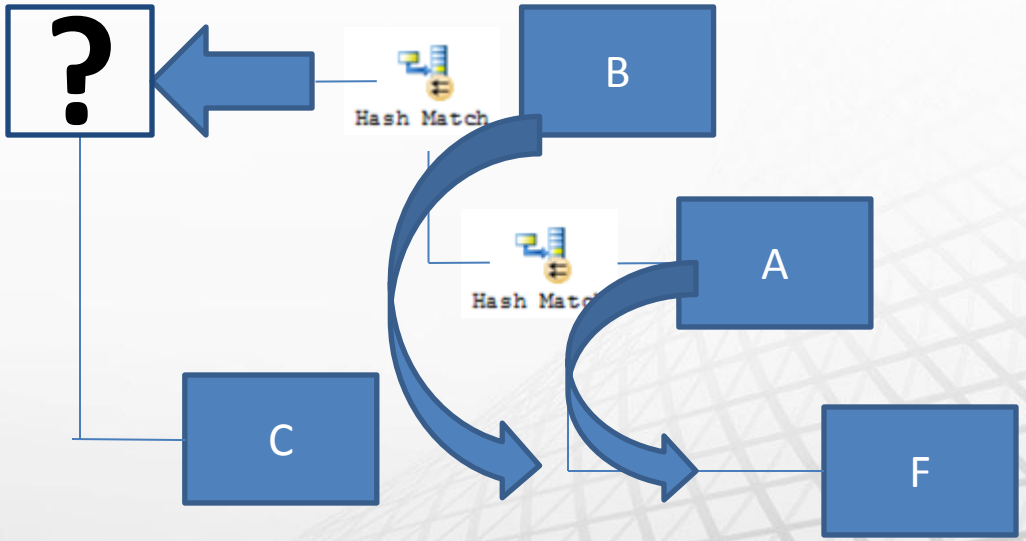

Estimated Plan Type
Row
Batch
Execution Mode
Batch
Row
PROBE Pushdown
Bitmap Filters
No Probe No Bitmap

Query pattern

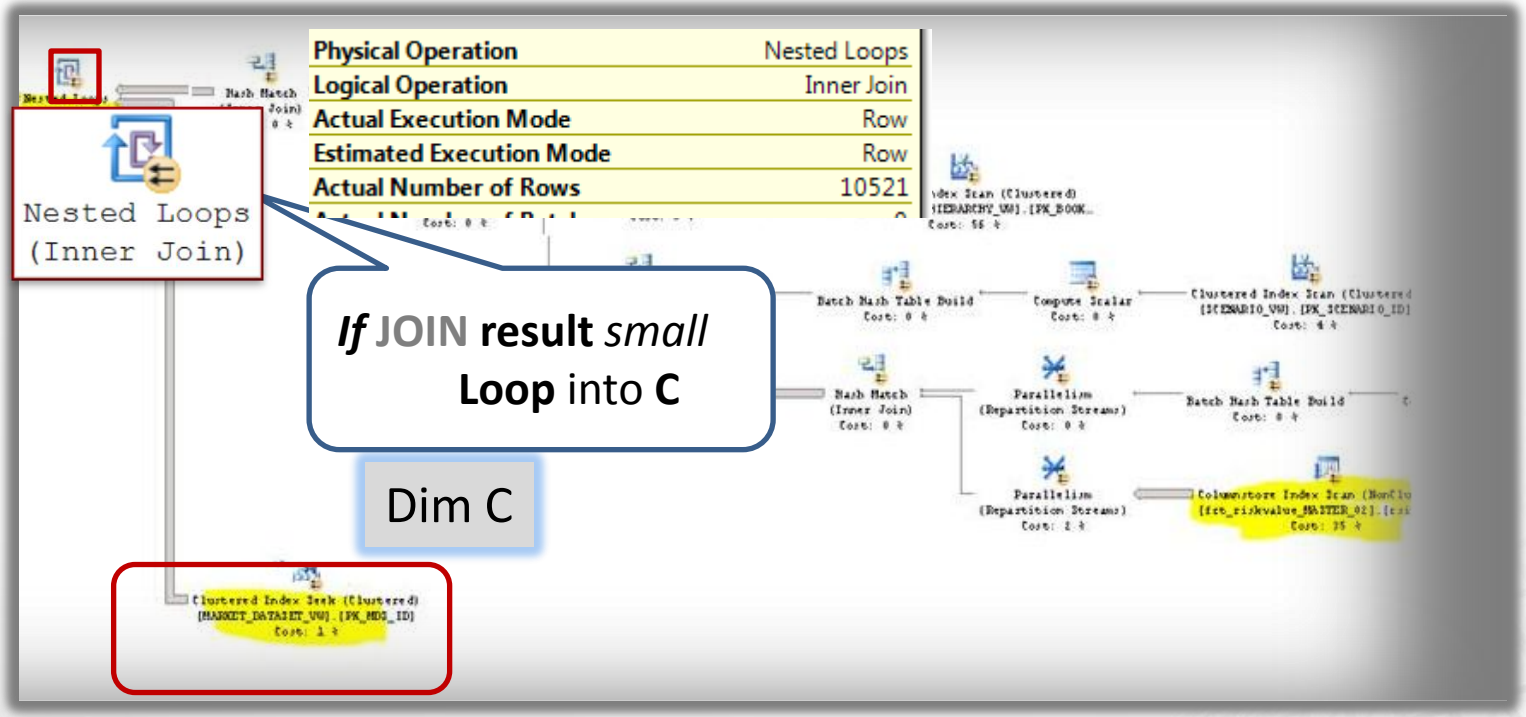
```
SELECT column1, column2
FROM Fact F
   JOIN Dim A
   JOIN Dim B
   JOIN Dim C-- (this is big)
WHERE
         DimA.Col1 = value1
And      DimB.Col = Val1
Group By DimC.Col1
```

If JOIN result small
➡ Loop into C
If JOIN result big
➡ Build Hash on C

Can we make this fast?

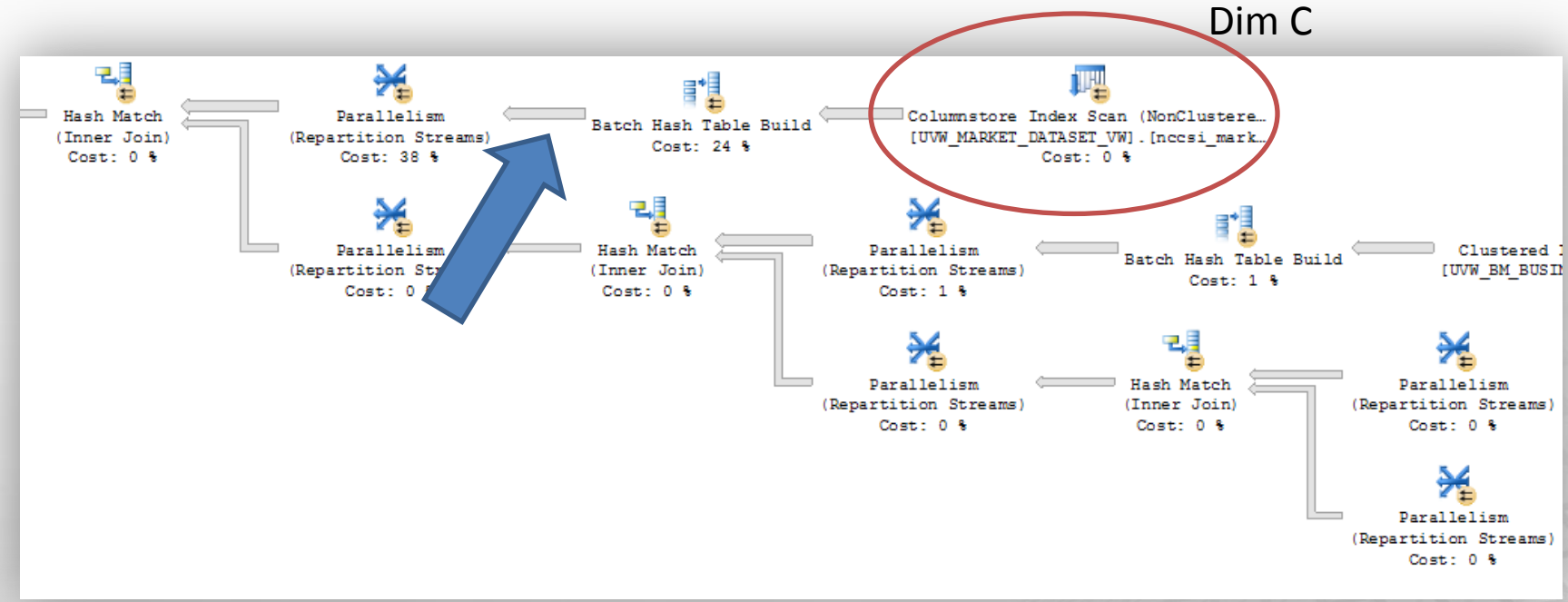


Expensive Loop



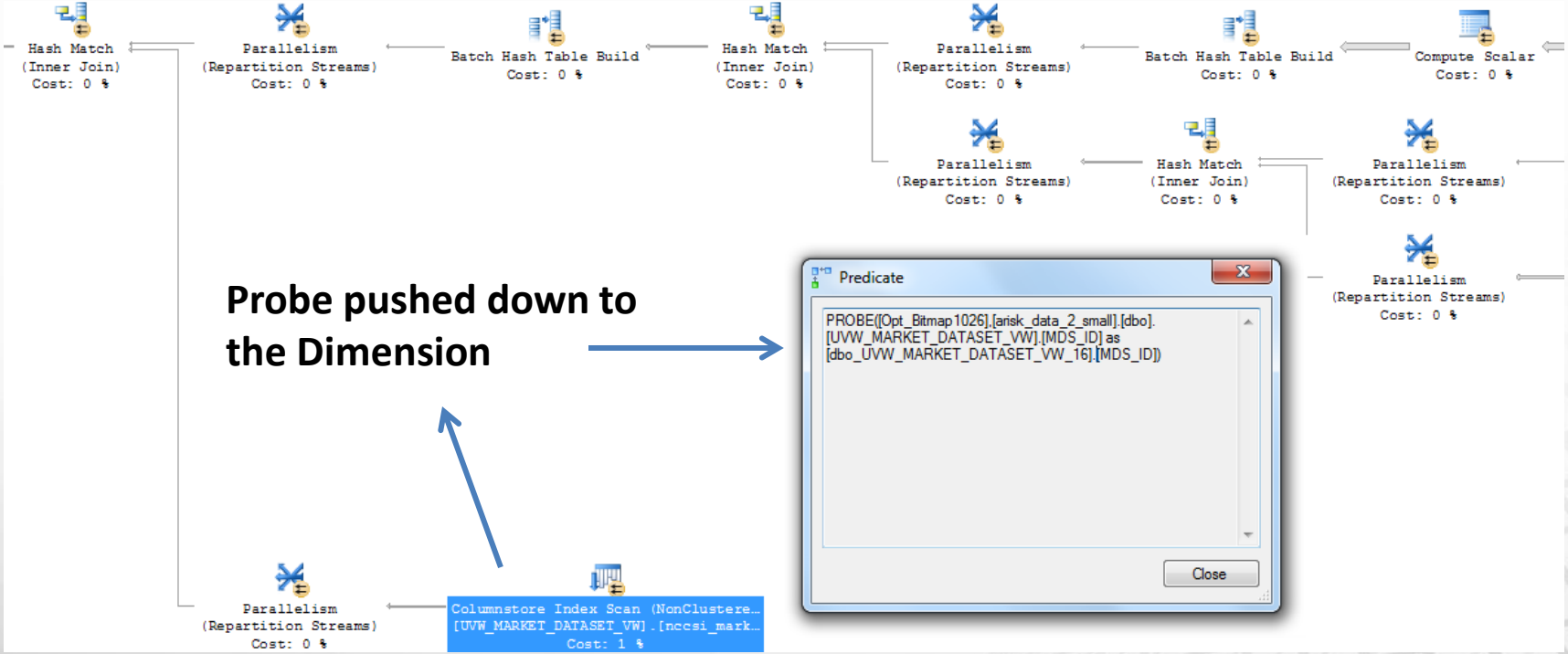
CSI on the Dimension

- All great as long as the estimates are correct

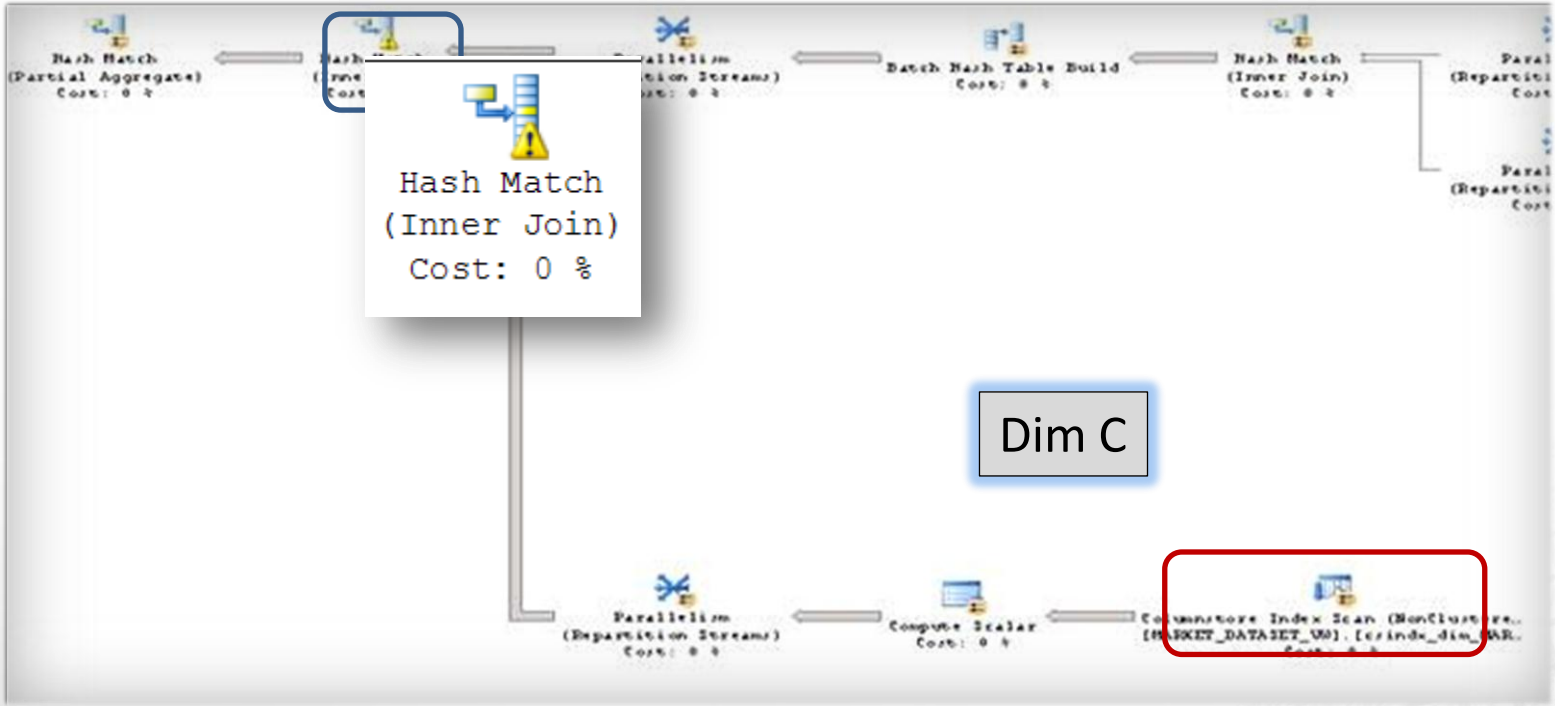


Also good plan

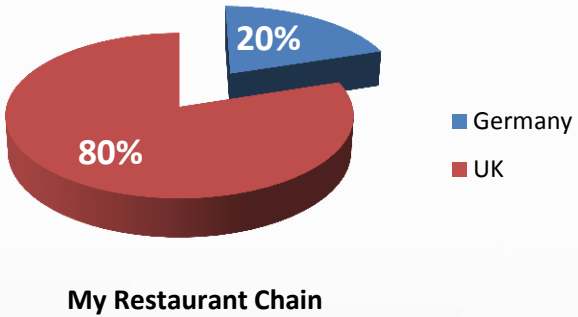
As long as there are no misestimates..

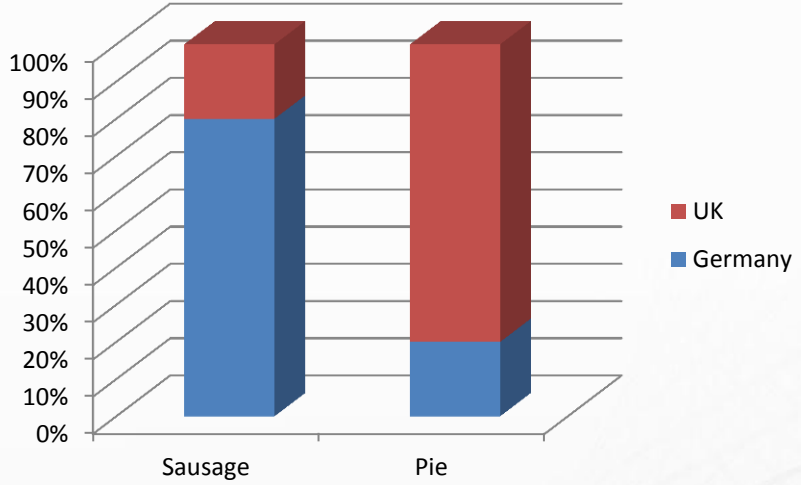


Misestimate of the JOIN result



Composite statistics on the fact table



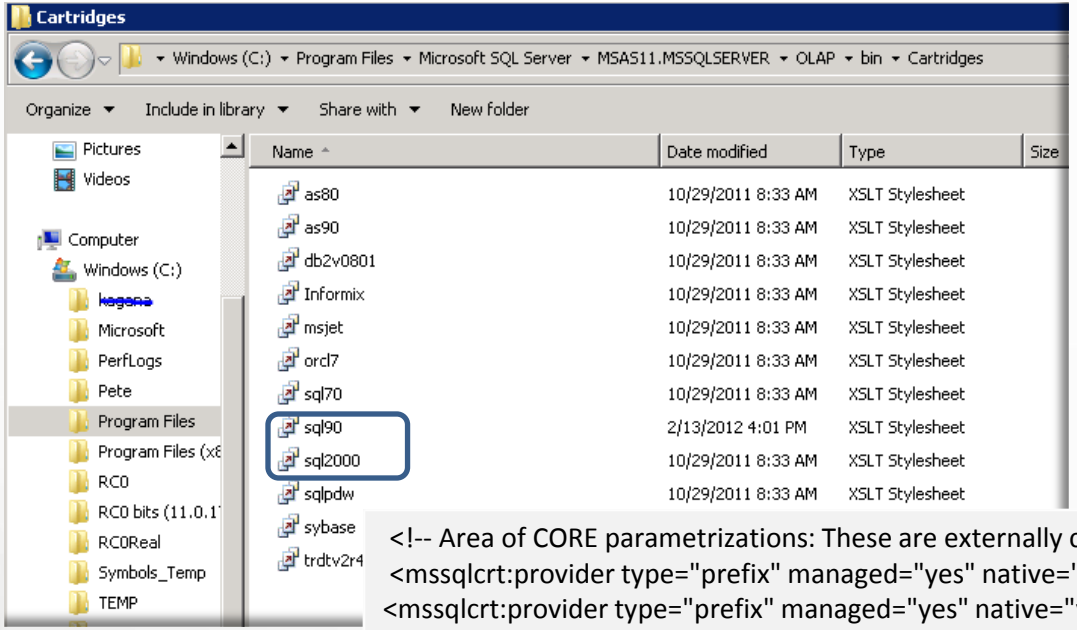



```
SELECT <columns> FROM Fact.MyRestaurants
Join on Dim.Country
Join on Dim.Products
WHERE Fact.Country = 'Germany'
  AND Fact.Product = 'Sausage'
```

Intelligent HINTS

- OPTION (HASH JOIN, Nested LOOP)
  - We know how to do that in relational, but query sent from MOLAP

Hacking SSAS




```
<!-- Area of CORE parametrizations: These are externally checked     -->
 <mssqlcrt:provider type="prefix" managed="yes" native="yes">Microsoft SQL Server.09</mssqlcrt:provider>
<mssqlcrt:provider type="prefix" managed="yes" native="yes">Microsoft SQL Server.10</mssqlcrt:provider>
<mssqlcrt:provider type="prefix" managed="yes" native="yes">Microsoft SQL Server.11</mssqlcrt:provider>
<mssqlcrt:parameter-style native="unnamed" managed="named"/>
```

```
<xsl:param name="post-select-query-hint"> OPTION (HASH JOIN, NESTED LOOP JOIN) </xsl:param>
```

Before Hinting

- Runtime: 29sec

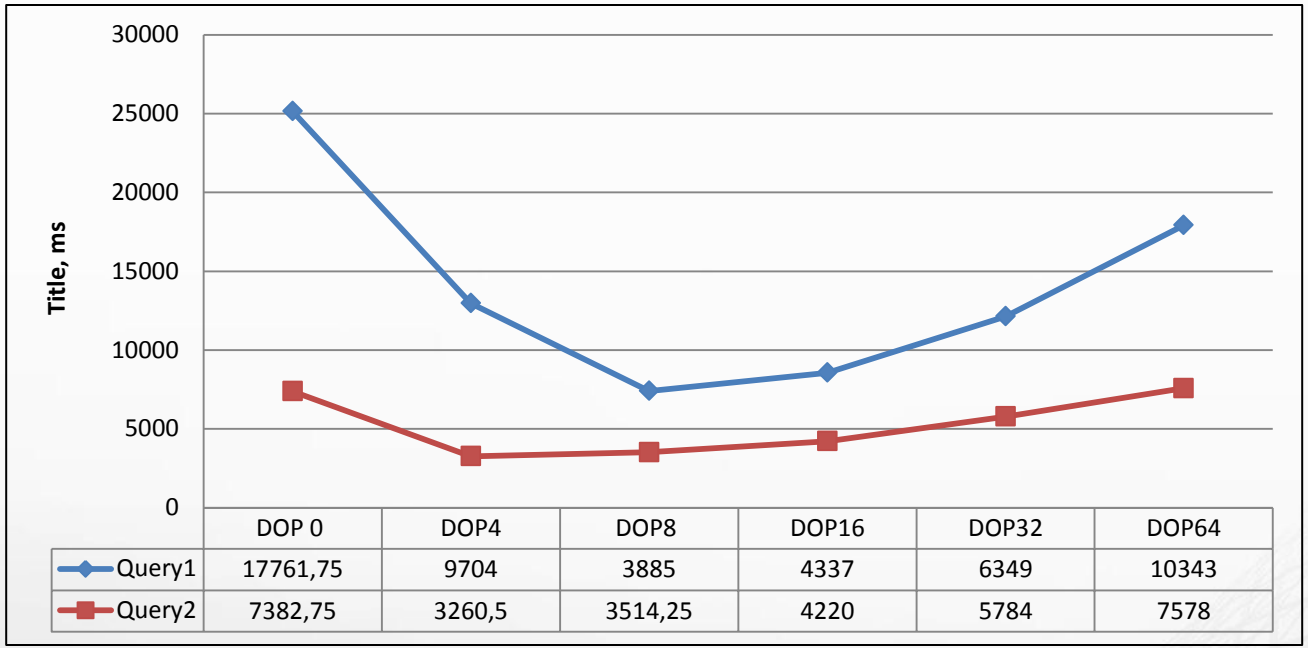
Option -- HASH.sqlplan

After

- Runtime: 14 sec

Option HASH.sqlplan

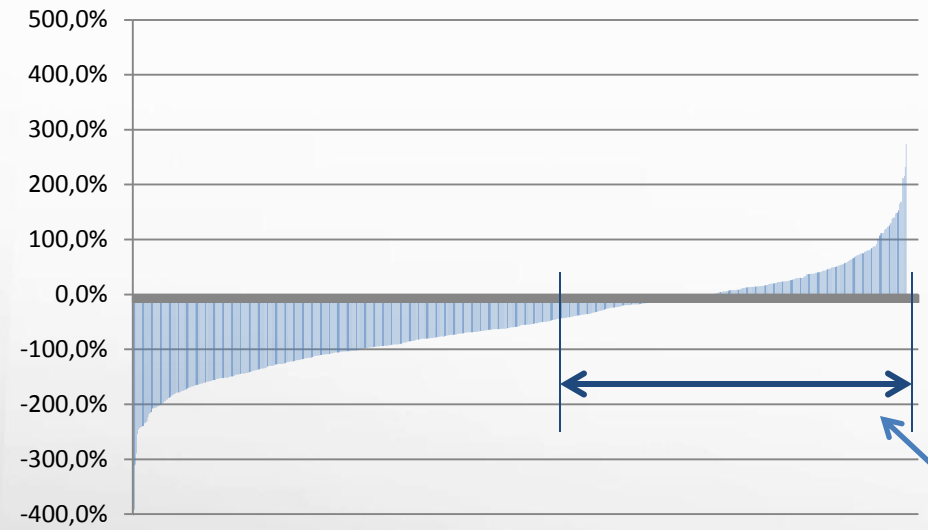
Impact of the Degree of Parallelism

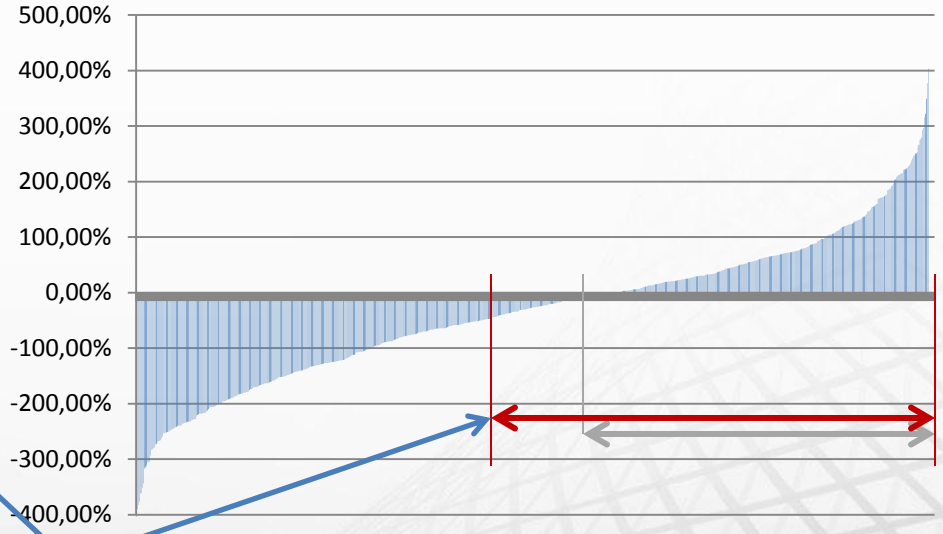


	DOP 0	DOP4	DOP8	DOP16	DOP32	DOP64
Query1	17761,75	9704	3885	4337	6349	10343
Query2	7382,75	3260,5	3514,25	4220	5784	7578

Compare with MOLAP

- Before Optimizations
- After Optimizations





ROLAP is Ok

Check List

- Dimensional Modeling
- Join columns data types matching
- Up-to-date or custom statistics
  - No spills on the disk
- Batch mode column store index scan
- Column store on the big dimensions

?

!

Additional Resources

- Resource list in columnstore FAQ:
  - http://social.technet.microsoft.com/wiki/contents/articles/sql-server-columnstore-index-faq.aspx
  - Includes tuning guide, white paper, video talk, video demo, and in-depth technical paper
- Column Store tuning guide
  - Great resource for SIs & consultants

Microsoft®

Your potential. Our passion.®

Microsoft®

Your potential. Our passion.®

Motivation

- VERY BIG customer
  - Estimates 4 TB of data in the SSAS cube
  - Near real time (XX TB/sec, which is ..million rows/sec)
- Demand:
  - Store more
  - Fresh data

Why ROLAP?

“ROLAP stands for **R**elational **O**n**l**ine **A**nalytical **P**rocessing.

ROLAP is an alternative to the MOLAP (Multidimensional OLAP) technology. While both ROLAP and MOLAP analytic tools are designed to allow analysis of data through the use of a multidimensional data model, ROLAP differs significantly in that it does not require the pre-computation and storage of information. Instead, ROLAP tools access the data in a relational database and generate SQL queries to calculate information at the appropriate level when an end user requests it. With ROLAP, it is possible to create additional database tables (*summary tables* or *aggregations*) which summarize the data at any desired combination of dimensions.

While ROLAP uses a relational database source, generally the database must be carefully designed for ROLAP use. A database which was designed for OLTP will not function well as a ROLAP database. Therefore, ROLAP still involves creating an additional copy of the data. However, since it is a database, a variety of technologies can be used to populate the database.”

Thoughts

- Relational engine scales better
- Loaded once we don’t need to do anything else – query will need simply go’n grab data
- Column store index:
- *‘This new index, combined with enhanced query optimization and execution features, improves data warehouse query performance by* ***hundreds to thousands*** *of times in some cases, and can routinely give a tenfold speedup for a broad range of queries fitting the scenario for which it was designed. ‘* - source: ‘Columnstore Indexes for Fast Data Warehouse Query Processing in SQL Server 11.0 ‘

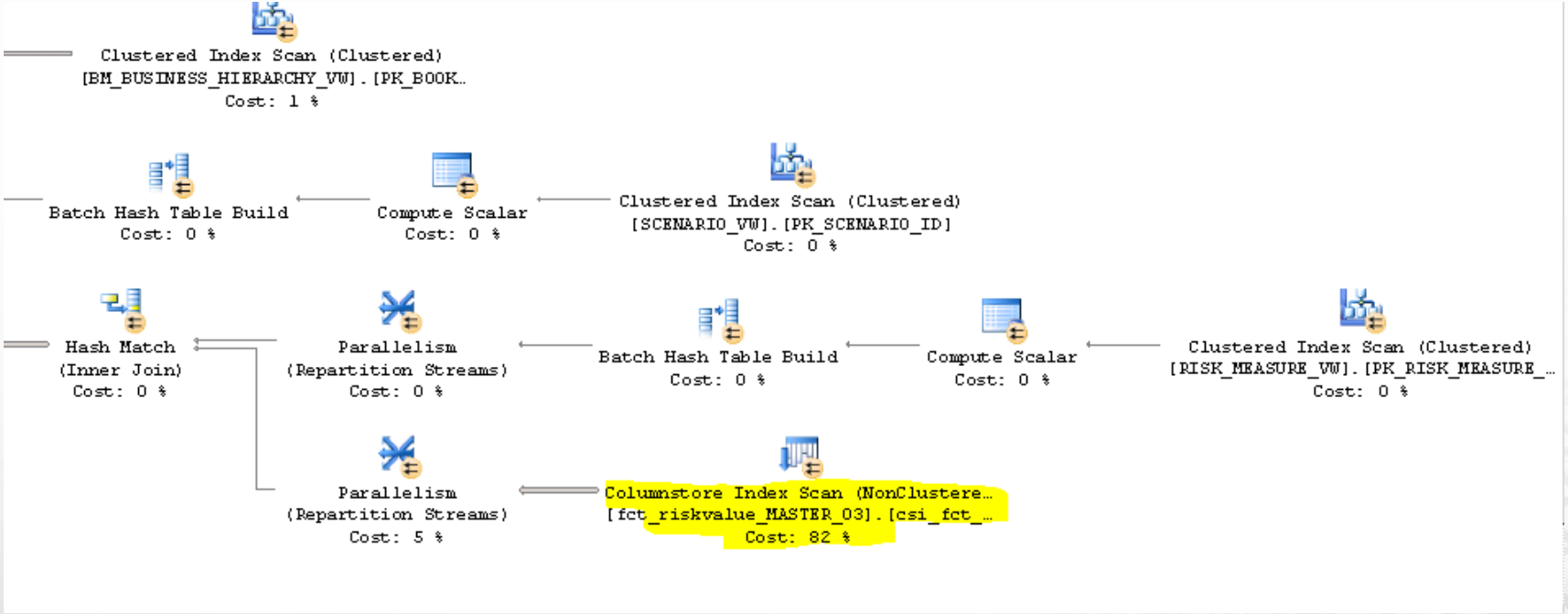
```
exec sp_executesql N'
SELECT  SUM ( [dbo_fct_riskvalue_MASTER_03].[VALUE_USD] )
 AS [dbo_fct_riskvalue_MASTER_03VALUE_USD0_0], SUM ( [dbo_fct_riskvalue_MASTER_03].[VALUE_LOCAL_CCY] )
 AS [dbo_fct_riskvalue_MASTER_03VALUE_LOCAL_CCY0_1],[dbo_fct_riskvalue_MASTER_03].[VALUE_DATE] AS
[dbo_fct_riskvalue_MASTER_03VALUE_DATE0_5],[dbo_UVW_BM_BUSINESS_HIERARCHY_VW_15].[SUB_DESK_ID] AS
[dbo_UVW_BM_BUSINESS_HIERARCHY_VWSUB_DESK_ID2_0],[dbo_UVW_MARKET_DATASET_VW_16].[CURVE_QUALITY] AS [dbo_UVW_MARKET_DATASET_VWCURVE_QUALITY3_0]
FROM [dbo].[fct_riskvalue_MASTER_03] AS [dbo_fct_riskvalue_MASTER_03],[dbo].[UVW_BM_BUSINESS_HIERARCHY_VW] AS
[dbo_UVW_BM_BUSINESS_HIERARCHY_VW_15],[dbo].[UVW_MARKET_DATASET_VW] AS [dbo_UVW_MARKET_DATASET_VW_16],[dbo].[UVW_MARKET_SEGMENT_VW] AS
[dbo_UVW_MARKET_SEGMENT_VW_1],[dbo].[UVW_RISK_MEASURE_VW] AS [dbo_UVW_RISK_MEASURE_VW_2],[dbo].[UVW_SCENARIO_VW] AS [dbo_UVW_SCENARIO_VW_3]
WHERE
(

(
[dbo_fct_riskvalue_MASTER_03].[MARKET_SEGMENT1_ID]=[dbo_UVW_MARKET_SEGMENT_VW_1].[MS_ID]
)
AND
(
[dbo_fct_riskvalue_MASTER_03].[RISK_MEASURE_ID]=[dbo_UVW_RISK_MEASURE_VW_2].[RISK_MEASURE_ID]
)
AND
(
[dbo_fct_riskvalue_MASTER_03].[SCENARIO_ID]=[dbo_UVW_SCENARIO_VW_3].[SCENARIO_ID]
)
AND
(
[dbo_fct_riskvalue_MASTER_03].[BOOK_ID]=[dbo_UVW_BM_BUSINESS_HIERARCHY_VW_15].[BOOK_ID]
)
AND
(
[dbo_fct_riskvalue_MASTER_03].[MARKET_DATASET_ID]=[dbo_UVW_MARKET_DATASET_VW_16].[MDS_ID]
)
AND
(
[dbo_UVW_RISK_MEASURE_VW_2].[RISK_MEASURE_NAME]=
@P1
```

Nested Loop Join query

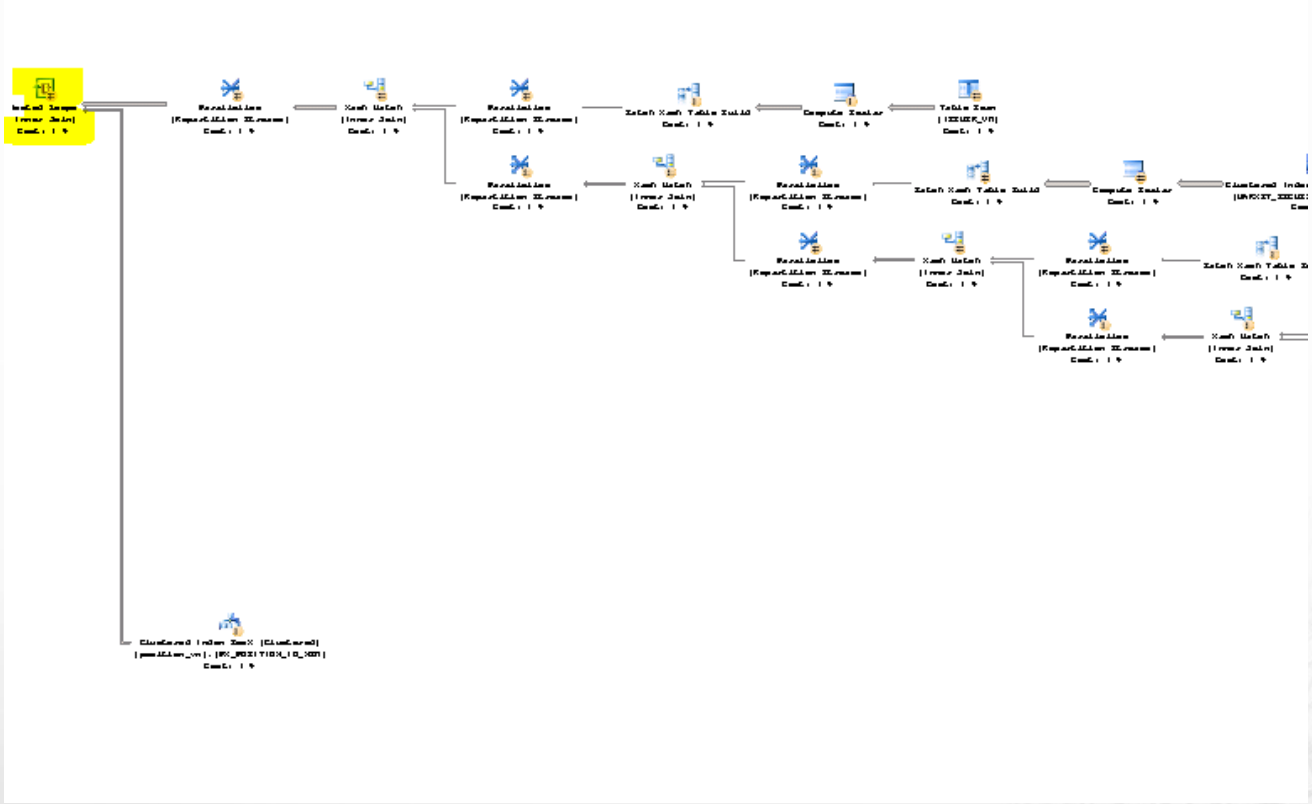
```
exec sp_executesql N'
SELECT  SUM ( [dbo_fct_riskvalue_MASTER_03].[VALUE_USD] )
 AS [dbo_fct_riskvalue_MASTER_03VALUE_USD0_0], SUM ( [dbo_fct_riskvalue_MASTER_03].[VALUE_LOCAL_CCY] )
 AS [dbo_fct_riskvalue_MASTER_03VALUE_LOCAL_CCY0_1],[dbo_fct_riskvalue_MASTER_03].[VALUE_DATE] AS [dbo_fct_riskvalue_MASTER_03VALUE_DATE0_6],[dbo_UVW_POSITION_VW_5].[PRODUCT_NAME] AS [dbo_UVW_POSITION_VWPRODUCT_NAME1_0],[dbo_UVW_BM_BUSINESS_HIERARCHY_VW_15].[SUB_DESK_ID] AS [dbo_UVW_BM_BUSINESS_HIERARCHY_VWSUB_DESK_ID3_0]
FROM [dbo].[fct_riskvalue_MASTER_03] AS [dbo_fct_riskvalue_MASTER_03],[dbo].[UVW_POSITION_VW] AS [dbo_UVW_POSITION_VW_5],[dbo].[UVW_BM_BUSINESS_HIERARCHY_VW] AS [dbo_UVW_BM_BUSINESS_HIERARCHY_VW_15],[dbo].[UVW_MARKET_SEGMENT_VW] AS [dbo_UVW_MARKET_SEGMENT_VW_1],[dbo].[UVW_RISK_MEASURE_VW] AS [dbo_UVW_RISK_MEASURE_VW_2],[dbo].[UVW_SCENARIO_VW] AS [dbo_UVW_SCENARIO_VW_3],[dbo].[UVW_ISSUER_VW] AS [dbo_UVW_ISSUER_VW_14],[dbo].[UVW_MARKET_DATASET_VW] AS [dbo_UVW_MARKET_DATASET_VW_16]
WHERE
(

(
[dbo_fct_riskvalue_MASTER_03].[MARKET_SEGMENT1_ID]=[dbo_UVW_MARKET_SEGMENT_VW_1].[MS_ID]
)
AND
(
[dbo_fct_riskvalue_MASTER_03].[RISK_MEASURE_ID]=[dbo_UVW_RISK_MEASURE_VW_2].[RISK_MEASURE_ID]
)
AND
(
[dbo_fct_riskvalue_MASTER_03].[SCENARIO_ID]=[dbo_UVW_SCENARIO_VW_3].[SCENARIO_ID]
)
AND
(
[dbo_fct_riskvalue_MASTER_03].[POSITION_ID]=[dbo_UVW_POSITION_VW_5].[POSITION_ID]
)
AND
(
[dbo_fct_riskvalue_MASTER_03].[ISSUER_ID]=[dbo_UVW_ISSUER_VW_14].[CURVE_ID]
)
AND
(
[dbo_fct_riskvalue_MASTER_03].[BOOK_ID]=[dbo_UVW_BM_BUSINESS_HIERARCHY_VW_15].[BOOK_ID]
```

Plan

- First look – all is nice



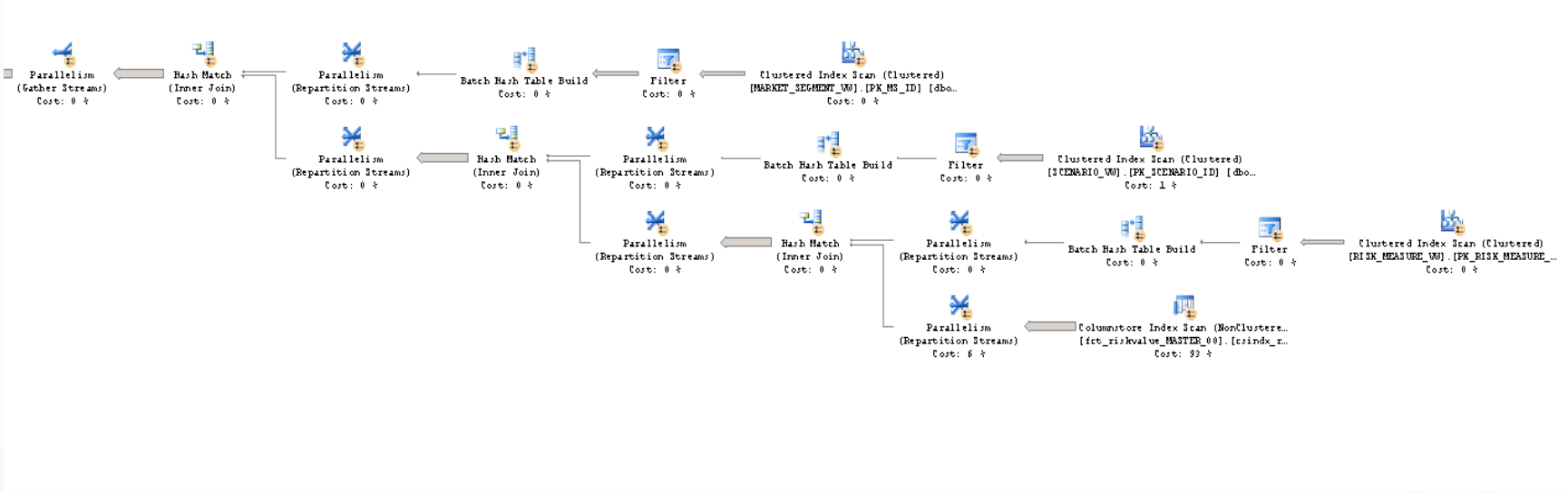
Zooming out



Query to show the benefits of the data type matches

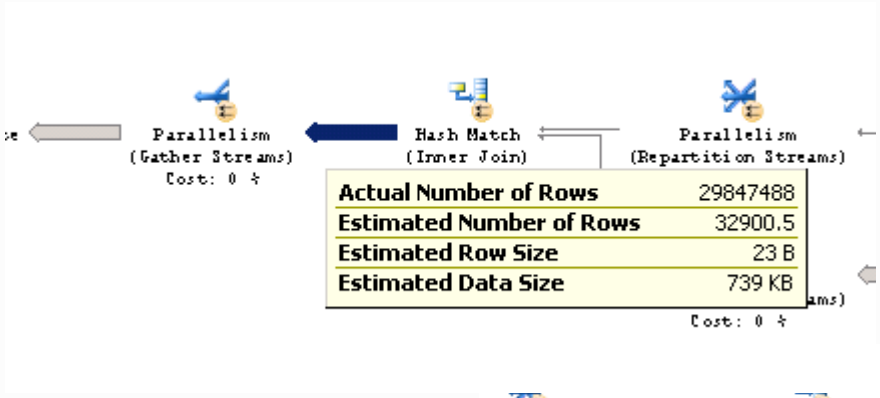
```
--SET STATISTICS TIME ON
--UPDATE STATISTICS dbo.fct_riskvalue_AMER_01_hashed
Declare @Param1 nvarchar(max) = 'TV', @Param2 nvarchar(max) = 'CentreState', @Param3 nvarchar(max) = 'FO'


 SELECT  SUM ( [ARISK_RUFF_OLAP_G2_RISK_VALUE_MODEL_SQL].[VALUE_USD] )AS [ARISK_RUFF_OLAP_G2_RISK_VALUE_MODEL_SQLVALUE_USD0_0],
 SUM ( [ARISK_RUFF_OLAP_G2_RISK_VALUE_MODEL_SQL].[VALUE_LOCAL_CCY] )AS [ARISK_RUFF_OLAP_G2_RISK_VALUE_MODEL_SQLVALUE_LOCAL_CCY0_2]
        FROM
            (

SELECT  *
      FROM dbo.fct_riskvalue_MASTER_00



            )
         AS [ARISK_RUFF_OLAP_G2_RISK_VALUE_MODEL_SQL],
  [dbo].MARKET_SEGMENT_VW AS [dbo_UVW_MARKET_SEGMENT_VW_1],
  [dbo].RISK_MEASURE_VW AS [dbo_UVW_RISK_MEASURE_VW_2],
  [dbo].SCENARIO_VW AS [dbo_UVW_SCENARIO_VW_3]
        WHERE
        (

        (
            [ARISK_RUFF_OLAP_G2_RISK_VALUE_MODEL_SQL].[MARKET_SEGMENT1_ID]      =     [dbo_UVW_MARKET_SEGMENT_VW_1].[MS_ID]
        )
                AND
        (
            [ARISK_RUFF_OLAP_G2_RISK_VALUE_MODEL_SQL].[RISK_MEASURE_ID]     =     [dbo_UVW_RISK_MEASURE_VW_2].[RISK_MEASURE_ID]
        )
                AND
        (
```

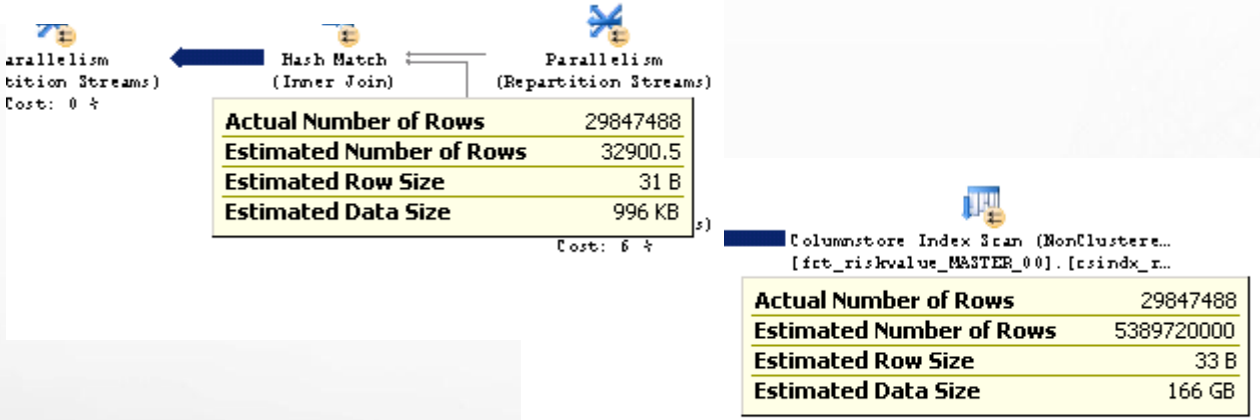
All looks good from the first look



- Execution time is 2 min 33 sec
- Instead 30 sec

But something look strange





DOP vs Execution time