Microsoft

Azure
SQL Data Warehouse

Performance Tuning

- Performance Concepts
- Troubleshooting Tools
- Common Issues

Performance
Basics

Performance Concepts (SELECTS)

- COST BASED OPTIMIZER
- DATA MOVEMENT
- RESOURCE CLASSES
- NODE LEVEL QUERY PLAN

SQL DW Query

T-SQL

```
SELECT foo_x, foo_y, foo_z FROM BAR
```

Control

PDW ENGINE

D-SQL

```
<?xml version="1.0" encoding="utf-8"?>
<dsql_query number_nodes="6" number_distributions="60" number_distributions_per_node="10">
    <sql>SELECT foo_x, foo_y, foo_z FROM BAR OPTION(label = 'foo_query_one')</sql>
    <dsql_operations total_cost="0" total_number_operations="1">
        <dsql_operation operation_type="RETURN">
            <location distribution="AllDistributions"/>
            <select>SELECT [T1_1].[foo_x] AS [foo_x], [T1_1].[foo_y] AS [foo_y], [T1_1].[foo_z] AS [foo_z] FROM [loading_gen2].[dbo].[BAR] AS T1_1  OPTION (MAXDOP 7)</select>
        </dsql_operation>
    </dsql_operations>
</dsql_query>
```

Compute

SQL ENGINE

Key thoughts:

- 2 Cost based optimizers (PDW Engine, SQL Engine)
- T-SQL Query broken down into steps (SQL Query, Data Movement, or Management operations) in PDW Engine
- D-SQL Queries executed on distributions by compute nodes

Side note on Stats

There are two sets of stats!

Compute Node Statistics:

Per distribution SQL Server Statistics objects with auto-create and auto-update on.
Determine Compute Node Execution of D-SQL Statements

Control Node Statistics:

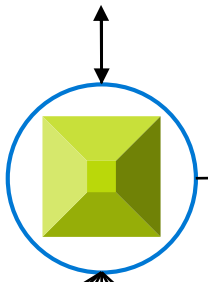
Merge of Compute Node Stats.
Determine D-SQL Query plan

SQL DW Query

T-SQL

```
SELECT AVG(foo_x),foo_y FROM BAR GROUP BY foo_y
```

Control



On Operation

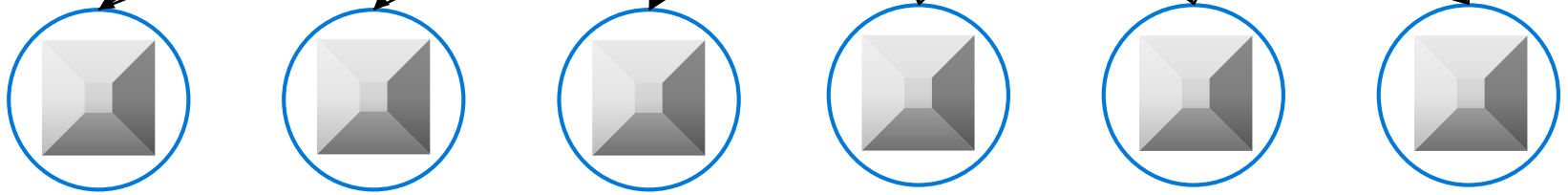
QTable

Return Operation

Step_index	Operation_type	location_type	row_count	Command
0	OnOperation	Control	-1	CREATE TABLE [tempdb].[QTables].[QTable
1	PartitionMoveOperation	Compute	5	SELECT [T1_1].[foo_
2	ReturnOperation	Control	-1	SELECT [T1_1].[col]
3	OnOperation	Control	-1	DROP TABLE [tempc

Partition Move Operation

Compute



Data Size Changes Query Plan

```
--100
SELECT AVG(ArrivalDelay) as [avg_Delay],
       DateId
FROM  [dbo].[Flight_small]
Where CarrierId = 506
GROUP BY DateID
Option(label = 'small_Avg')
```

Step_index	Operation_type	location_type	row_count	Command
0	OnOperation	Control	-1	CREATE TABLE [tempdb].[QTables].
1	PartitionMoveOperation	Compute	29	SELECT [T1_1].[DateId] AS [DateId],
2	ReturnOperation	Control	-1	SELECT [T1_1].[col] AS [col], [T1_1].
3	OnOperation	Control	-1	DROP TABLE [tempdb].[QTables].[Q

```
--133,708,176
SELECT AVG(ArrivalDelay) as [avg_Delay],
       DateId
FROM  [dbo].[Fact_Flight]
Where CarrierId = 506
GROUP BY DateID
Option(label = 'large_Avg')
```

Step_index	Operation_type	location_type	row_count	Command
0	RandomIDOperation	Control	-1	TEMP_ID_10
1	OnOperation	Compute	-1	CREATE TABLE [qtabledb].[dbo].[TEMP_ID_10]
2	ShuffleMoveOperation	Compute	622620	SELECT [T1_1].[DateId] AS [DateId], [T1_1].[col1
3	ReturnOperation	Compute	-1	SELECT [T1_1].[col] AS [col], [T1_1].[DateId] AS
4	OnOperation	Compute	-1	DROP TABLE [qtabledb].[dbo].[TEMP_ID_10]

Key thoughts:

- Assumed that data movement is the bottleneck
- Data moves between distributions to most efficiently satisfy query
- Data movement type determined by Control Stats and Table Design in the PDW engine
- Data movement always lands in temporary tables

Move Operations

Operation Name	Description
Partition Move	Moves data from a distributed table to a single table on the Control node. This operation is used for aggregation operations on the Control node.
Shuffle Move	Redistributes a distributed table. The redistributed table has a different distribution column than the original distributed table. This might be used to when running incompatible joins or incompatible aggregations.
Broadcast Move	Moves distributed data into a replicated table. This operation is frequently used when running a distribution incompatible join.
Trim Move	Moves a replicated table to a distributed table.
Return Operation	Sends query results from the Control node to the user who submitted the query. This is the final operation for a parallel query plan.

Distributed Data Movement (Shuffle)

ProductSales

	AccountID	SalesAmt	…
Node 1:	47	$1,234.36	…
Node 2:	36	$2,345.47	…
Node 3:	14	$3,456.58	…
Node 4:	25	$4,567.69	…
Node 5:	48	$5,678.70	…
Node 6:	37	$6,789.81	…
	…	…	…

SalesAccountTerritory

SATerritoryID	AccountID	…
444	37	…
333	25	
111	36	…
222	47	…
445	14	
334	48	…
…	…	…

Shuffle

SATName	TotalSales
North	$6,789.81
South	$5,678.70
NorthEast	$4,567.69
SouthWest	$3,456.58
East	$2,345.47
West	$1,234.36

	SATName	…
	West	…
	East	
	SouthWest	…
	NorthEast	…
	South	
	North	…
…	…	…

```
CREATE TABLE ProductSales
WITH (DISTRIBUTION=HASH(AccountID))
AS...

CREATE TABLE SalesAccountTerritory
WITH (DISTRIBUTION=HASH(SATerritoryID))
AS...

SELECT TOP 25 a.SalesAccountTerritoryName
      ,TotalSales = SUM(p.SalesAmt)
FROM ProductSales p
JOIN SalesAccountTerritory a
ON   a.AccountID = p.AccountID
GROUP BY a.SalesAccountTerritoryName
ORDER BY 2 DESC
```

Shuffle a little deeper (30,000 DWUc)

Hash Readers

Data

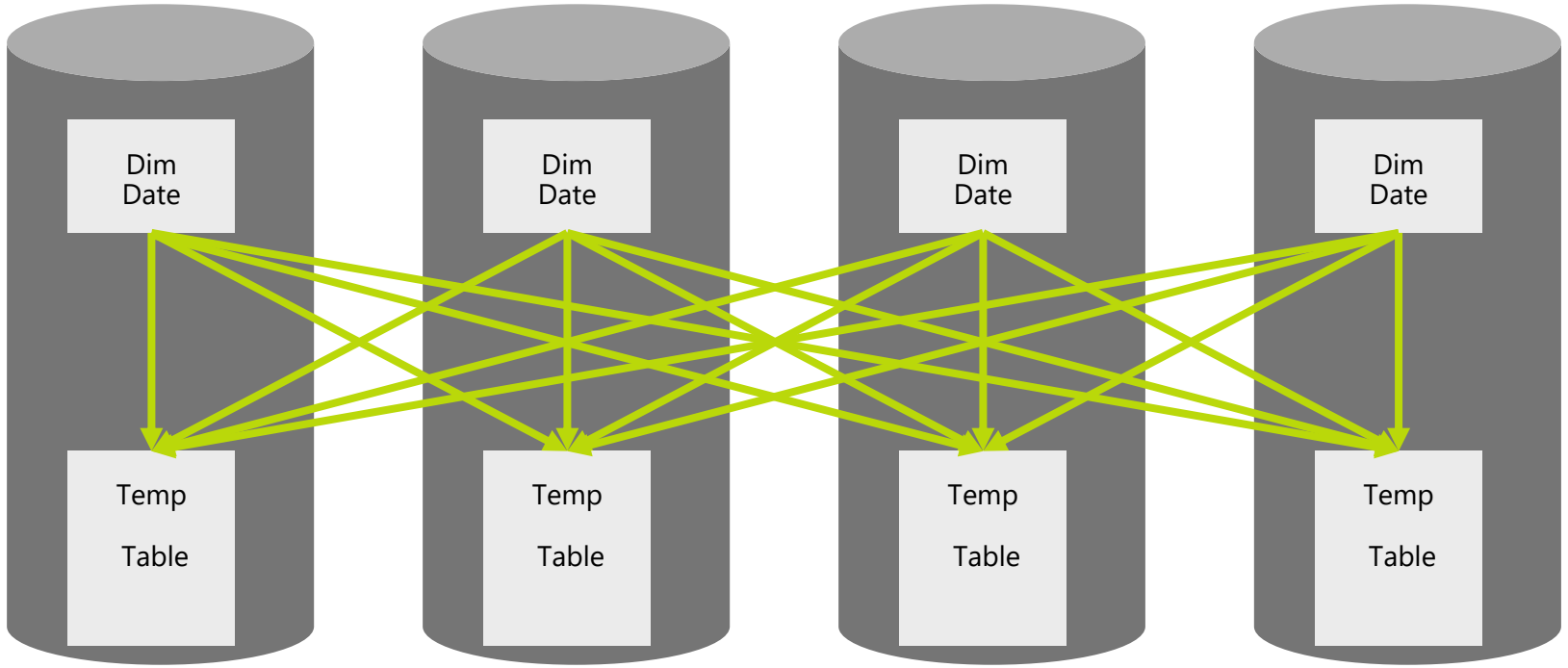
ent

```
SELECT
       [T1_1].[MonthName] AS [MonthName]
       ,[T1_1].[col] AS [col]
   FROM
       (
           SELECT
                   COUNT_BIG (
                       CAST( ( 0 ) AS INT )
                   ) AS [col]
                   ,[T2_1].[MonthName] AS [MonthName]
               FROM
                   [qtabledb].[dbo].[TEMP_ID_17] AS T2_1 INNER JOIN [loading_gen2].[dbo].[Fact_Flight] AS T2_2
                       ON (
                       [T2_1].[ID] = [T2_2].[DateId]
                   )
               GROUP BY
                   [T2_1].[MonthName]
       ) AS T1_1 OPTION (
           MAXDOP 7
           ,MIN_GRANT_PERCENT = 25
           ,DISTRIBUTED_MOVE (
               N ''
           )
       )
```

...

...

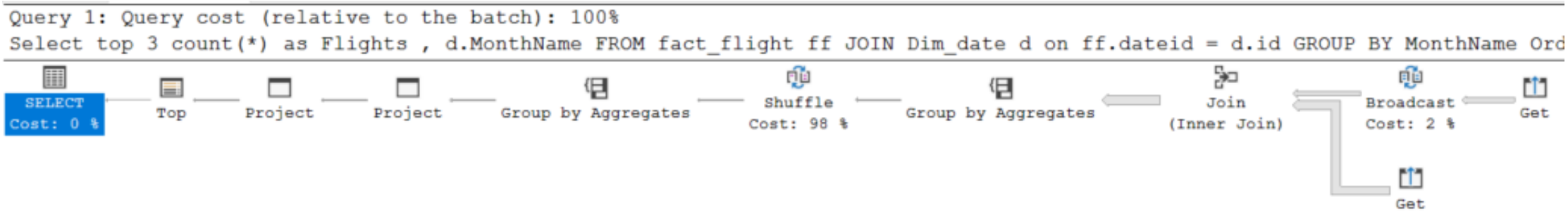
Distributed Data Movement (Broadcast)




```
CREATE TABLE Fact_Flight
WITH( DISTRIBUTION = HASH(flight_id)
AS ...

CREATE TABLE Dim_Date
WITH (DISTRIBUTION = ROUND_ROBIN)
AS ...
```

```
Select top 3 count(*) as Flights
           , d.MonthName
FROM fact_flight ff
JOIN Dim_date  d
on ff.dateid = d.id
GROUP BY MonthName
Order by 1 desc
option(label = 'Broadcast')
```



Compatible joins

Join Type	Left Table	Right	Compatibility
All Join Types			mpatible – No Data Movement
• Inner Join • Right Outer • Cross Join			tible – No Data Movement
• Inner Join • Left Outer Join • Cross Join			le – No Data Movement
All Joins (except cross join) can be compatible!	Dist d	Dis	mpatible IIF: • edicate is an equality join • Predicate joins two distributed columns that have matching data types

BIGINT
< >
INT
< >
SMALLINT
< >
TINYINT

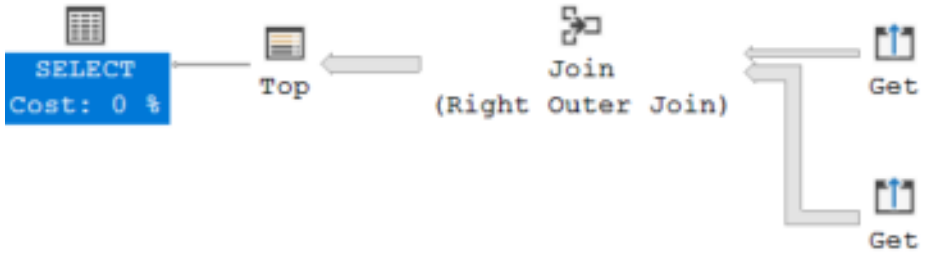
Incompatible joins

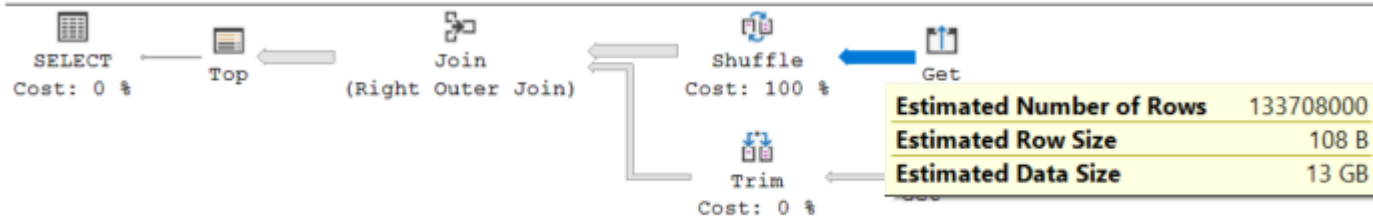
Join Type	Left Table	Right Table	Compatibility
• Left Outer Join • Full Outer Join	Replicated	Distributed	Incompatible – Requires Data Movement
• Right Outer Join • Full Outer Join	Distributed	Replicated	Incompatible – Requires Data Movement
All Joins (except cross join) can be compatible!	Distributed	Distributed	incompatible requires movement unless the join meets the following: • Predicate is an equality join • Predicate joins two distributed columns that have matching data types

Example (left outer join)

```
Select top 10 *
FROM fact_flight  ff
LEFT OUTER JOIN
DIM_DATE_REP  r
ON ff.dateid = r.id
OPTION (label = 'left_outer')
```

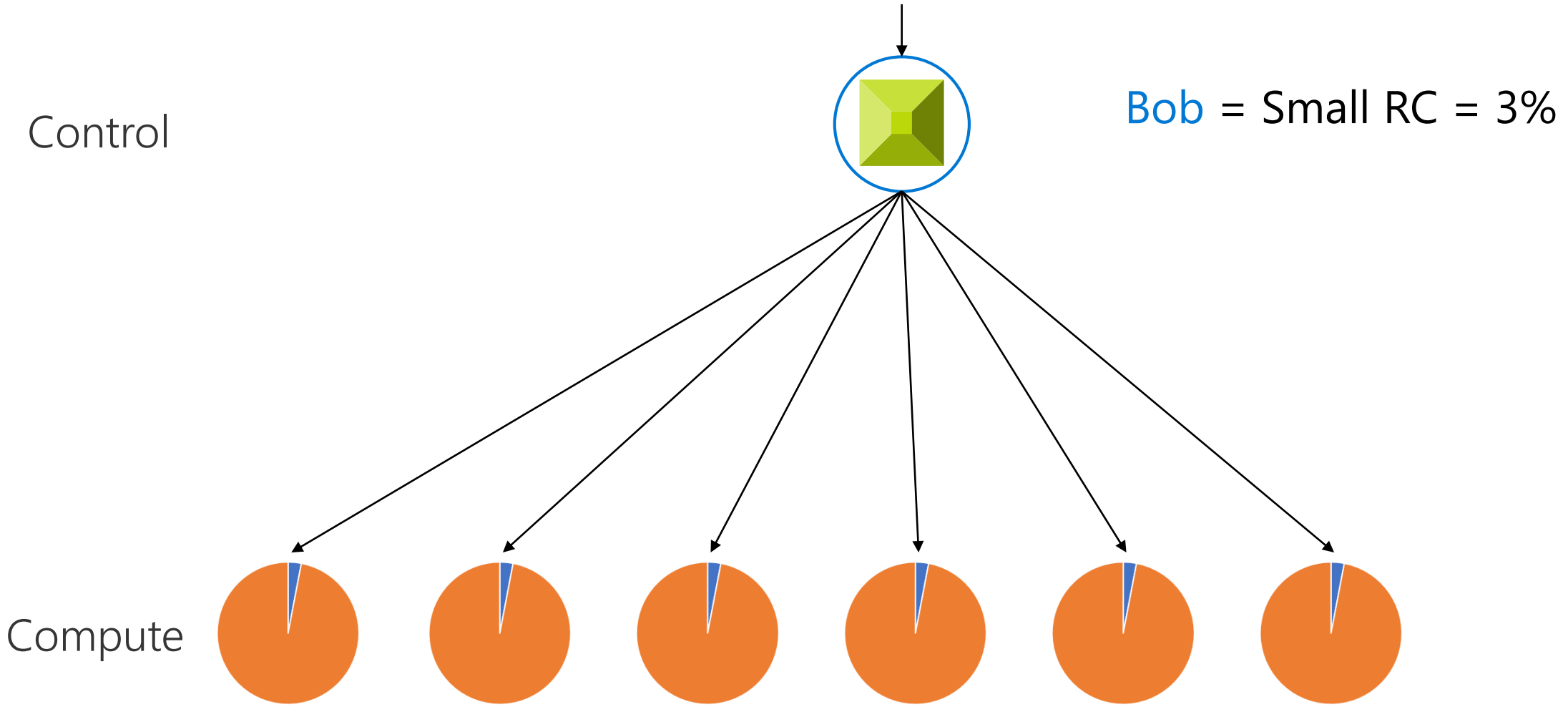
```
Select top 10 *
FROM DIM_DATE_REP  r
LEFT OUTER JOIN
fact_flight  ff
ON r.id = ff.dateid
OPTION (label = 'left_outer')
```





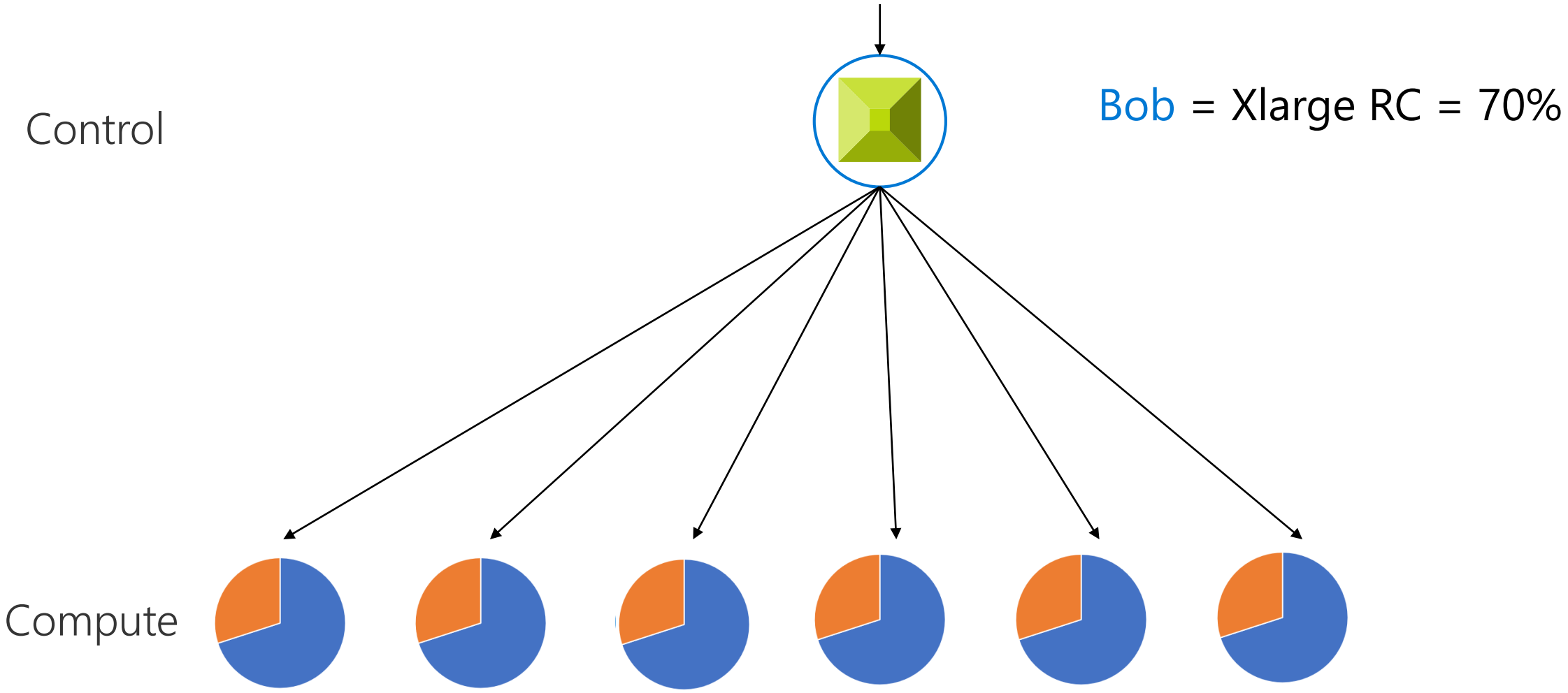
SQL DW Query (RCs)

```
SELECT AVG(foo_x),foo_y FROM BAR GROUP BY foo_y
```



SQL DW Query (RCs)

```
SELECT AVG(foo_x),foo_y FROM BAR GROUP BY foo_y
```



Query Hints

```
-- Syntax for Azure SQL Data Warehouse and Parallel Data Warehouse

OPTION ( <query_option> [ ,...n ] )

<query_option> ::=
    LABEL = label_name |
    <query_hint>

<query_hint> ::=
    HASH JOIN
    | LOOP JOIN
    | MERGE JOIN
    | FORCE ORDER
```

Query Hints in action!

```
-- Execution time = 1 sec
Select top 3 count(*) as Flights
          , d.MonthName
FROM fact_flight ff
JOIN Dim_date  d
on ff.dateid = d.id
GROUP BY MonthName
Order by 1 desc
option(MERGE JOIN)
```

```
Select top 3 count(*) as Flights
          , d.MonthName
FROM fact_flight ff
JOIN Dim_date  d
on ff.dateid = d.id
GROUP BY MonthName
Order by 1 desc
option(LOOP JOIN)
```

Join_type	Step	Operation_type	Elapsed_Time	Rows
MERGE	0	RandomIDOperation	0	-1
MERGE	1	OnOperation	31	-1
MERGE	2	BroadcastMoveOperation	109	10958
MERGE	3	RandomIDOperation	0	-1
MERGE	4	OnOperation	93	-1
MERGE	5	ShuffleMoveOperation	562	720
MERGE	6	ReturnOperation	124	-1
MERGE	7	OnOperation	78	-1
MERGE	8	OnOperation	15	-1

Join_type	Step	Operation_type	Elapsed_Time	Rows
LOOP	0	RandomIDOperation	0	-1
LOOP	1	OnOperation	46	-1
LOOP	2	BroadcastMoveOperation	93	10958
LOOP	3	RandomIDOperation	0	-1
LOOP	4	OnOperation	109	-1
LOOP	5	ShuffleMoveOperation	141273	720
LOOP	6	ReturnOperation	156	-1
LOOP	7	OnOperation	78	-1
LOOP	8	OnOperation	0	-1

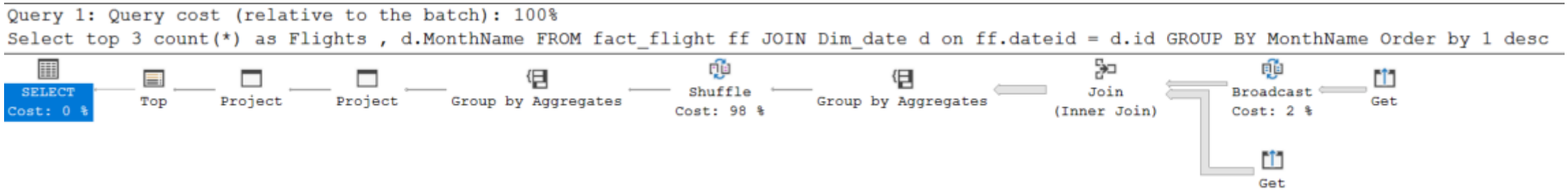
Troubleshooting Tools

Explain Plan

```
EXPLAIN
Select top 3 count(*) as Flights
          , d.MonthName
FROM fact_flight ff
JOIN Dim_date  d
on ff.dateid = d.id
GROUP BY MonthName
Order by 1 desc
```

```
<?xml version="1.0" encoding="utf-8"?>
<dsql_query number_nodes="6" number_distributions="60" number_distributions_per_node="10">
    <sql>Select top 3 count(*) as Flights             , d.MonthName  FROM fact_flight ff   JOIN Dim_date  d   on ff.dateid = d.id  GROUP BY MonthName  Order by 1 desc</sql>
    <dsql_operations total_cost="0" total_number_operations="9">
        <dsql_operation operation_type="RND_ID">
            <identifier>TEMP_ID_79</identifier>
        </dsql_operation>
        <dsql_operation operation_type="ON">
            <location permanent="false" distribution="AllComputeNodes" />
            <sql_operations>
                <sql_operation type="statement">CREATE TABLE [qtabledb].[dbo].[TEMP_ID_79] ([ID] INT NOT NULL, [MonthName] VARCHAR(9) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL )
            </sql_operations>
        </dsql_operation>
        <dsql_operation operation_type="BROADCAST_MOVE">
            <operation_cost cost="26.69183916624" accumulative_cost="26.69183916624" average_rowsize="10.149297" output_rows="10958" GroupNumber="4" />
            <source_statement>SELECT [T1_1].[ID] AS [ID], [T1_1].[MonthName] AS [MonthName] FROM [loading_gen2].[dbo].[Dim_Date] AS T1_1  OPTION (MAXDOP 7, MIN_GRANT_PERCENT = 25, D
            <destination_table>[TEMP_ID_79]</destination_table>
        </dsql_operation>
        <dsql_operation operation_type="RND_ID">
            <identifier>TEMP_ID_80</identifier>
        </dsql_operation>
        <dsql_operation operation_type="ON">
            <location permanent="false" distribution="AllDistributions" />
            <sql_operations>
                <sql_operation type="statement">CREATE TABLE [qtabledb].[dbo].[TEMP_ID_80] ([MonthName] VARCHAR(9) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL, [col] BIGINT ) WITH
            </sql_operations>
        </dsql_operation>
        <dsql_operation operation_type="SHUFFLE_MOVE">
            <operation_cost cost="0.00339583128" accumulative_cost="26.69523499752" average_rowsize="14.149297" output_rows="2.02221666666667" GroupNumber="20" />
            <source_statement>SELECT [T1_1].[MonthName] AS [MonthName], [T1_1].[col] AS [col] FROM (SELECT COUNT_BIG(CAST ((0) AS INT)) AS [col], [T2_1].[MonthName] AS [MonthName] F
            <destination_table>[TEMP_ID_80]</destination_table>
            <shuffle_columns>MonthName;</shuffle_columns>
        </dsql_operation>
        <dsql_operation operation_type="RETURN">
            <location distribution="AllDistributions" />
            <select>SELECT [T1_1].[col] AS [col], [T1_1].[MonthName] AS [MonthName] FROM (SELECT TOP (CAST ((3) AS BIGINT)) [T2_1].[col] AS [col], [T2_1].[MonthName] AS [MonthName]
        </dsql_operation>
        <dsql_operation operation_type="ON">
            <location permanent="false" distribution="AllDistributions" />
            <sql_operations>
                <sql_operation type="statement">DROP TABLE [qtabledb].[dbo].[TEMP_ID_80]</sql_operation>
            </sql_operations>
        </dsql_operation>
        <dsql_operation operation_type="ON">
            <location permanent="false" distribution="AllComputeNodes" />
            <sql_operations>
                <sql_operation type="statement">DROP TABLE [qtabledb].[dbo].[TEMP_ID_79]</sql_operation>
            </sql_operations>
        </dsql_operation>
    </dsql_operations>
</dsql_query>
```

Graphical Estimated Query Plan



Monitoring – T-SQL

Overview

Monitoring via Dynamic Management Views (DMVs)

Detailed view of query execution

Track query lifecycle and resource usage

Monitor in the data warehouse

Benefits

Active query troubleshooting

Identify workload performance bottlenecks

Extremely fine-grained telemetry for deep analysis

Monitor programmatically via T-SQL scripts and stored procedures

DMVs

`sys.dm_pdw_exec_requests`

Current or recently active requests/queries

`sys.dm_pdw_request_steps`

All steps for a given request or query

`sys.dm_pdw_dms_workers`

All workers completing Data Movement Service (DMS) steps

`sys.dm_pdw_waits`

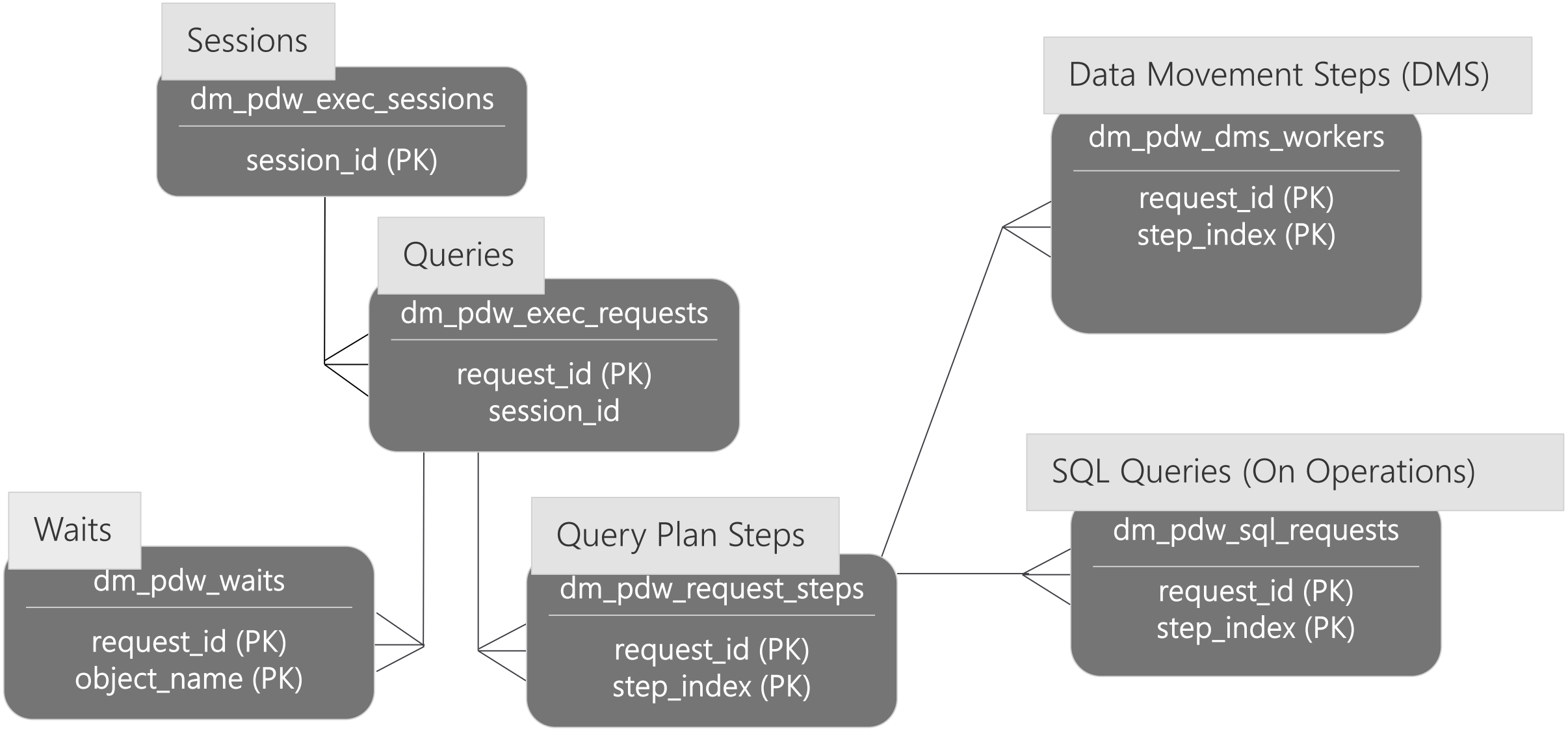
All wait states during a request or query

Locks, waits on queues, etc.

`sys.dm_pdw_sql_requests`

All SQL Server query distributions for each SQL step in a query

Execution DMVs



Common Issues

Common issues

- Out of date stats
- Poor CCI Quality
- Poorly designed table distribution
    - Fact to fact join not aligned
    - Round Robin Fact
    - Very large replicated tables (>2GB)
- Base Table Skew
    - Distributing on Nullable Column!
- Resource class
    - sqladmin = smallrc

Nuanced Issues

Table skew in temporary table created by shuffle

- Null value in distribution key
- Low cardinality, high row count

Multi Column Join

- Multi Column Statistics

SQL Engine picks node bad plan

Questions?