How to Crash Rock with Lava, SQL, and Workflows

Slide Deck

https://github.com/courtneycooksey

Who Am I?

Software Engineer
Savannah, MO
The Crossing, EPC

What is this about?

How can we find and fix performance issues in Lava, SQL, and Workflows?

It's Sunday Morning

A thousand people are hitting your website to watch the livestream

People are lined up to check their kids into childcare

New families are waiting at the desk to get their children's information added to Rock

Every page load and button press is taking 10 seconds

Issues Can Arise From Work Happening in the Moment or the Accumulation of Work

Processing at that Moment

- SQL reports that display current check-in information for staff
- Workflows we use at the desk to add new families to Rock

System-wide Loads

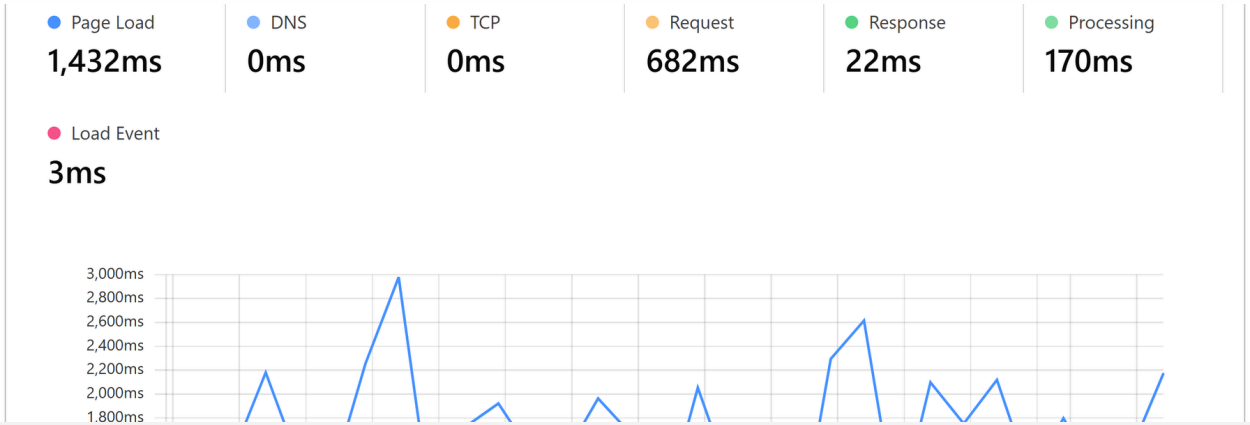
- Lava Templates that make up your public website
- Insufficient Caching on your public site
- Persisted Dataviews

Identifying Issues

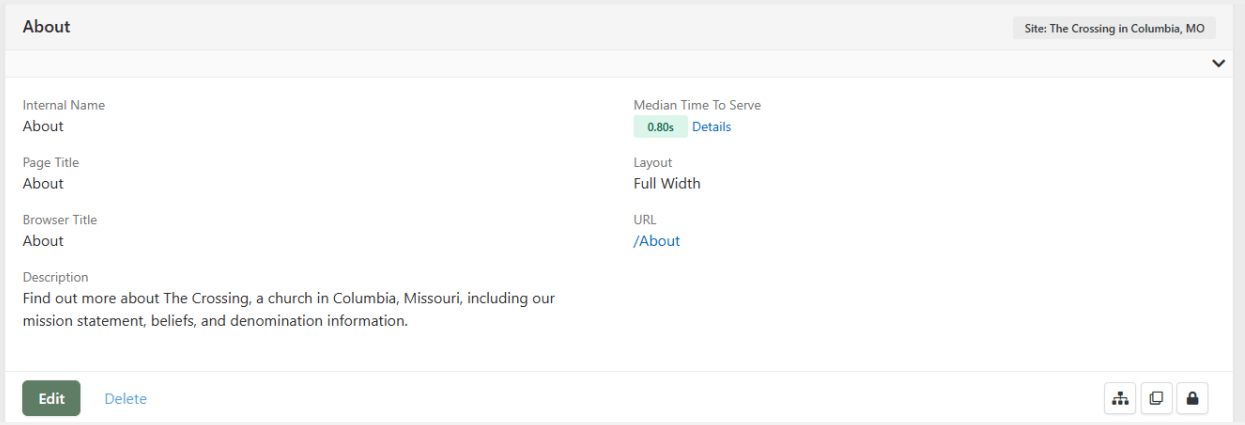
Locating pages and blocks with performance issues

Identifying Page Load Times

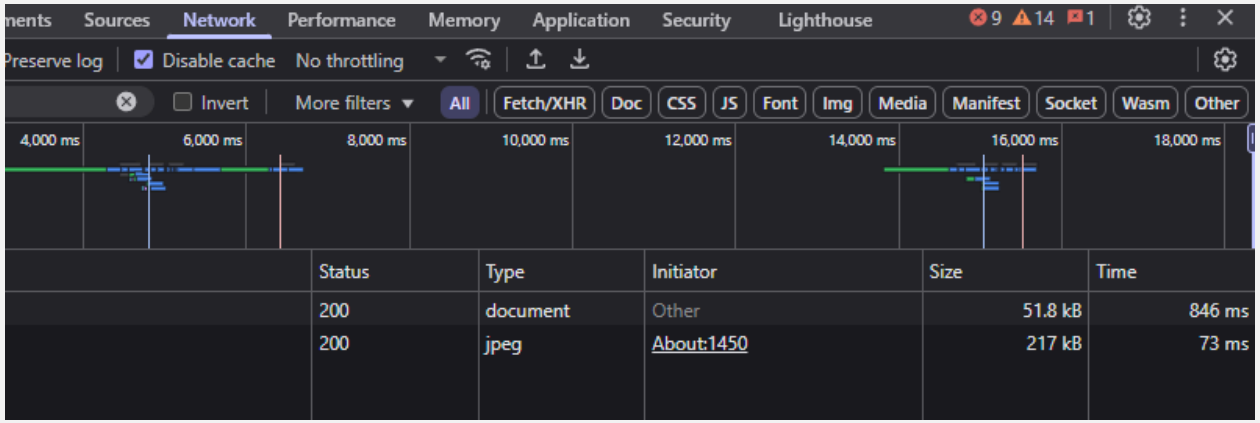
Cloudflare Web Analytics (Or Similar Tool)



Rock Page Median Time to Serve



Browser Developer Console



3rd Party Tools

- Page Speed Insights
- Lighthouse

Using the ShowDebugTimings Query Parameter

```
?ShowDebugTimings=true
```

32.75 ms	Sticky Nav Bar (Sticky Nav Bar)	93.28 ms
126.03 ms	Canonical URL (HTML Content)	33.25 ms
159.28 ms	Lucky Orange (HTML Content)	3.35 ms
162.63 ms	Location (HTML Content)	2.82 ms
165.45 ms	Service Times (HTML Content)	3.30 ms
168.75 ms	Most Recent Sermon (Content Channel View)	64.70 ms
233.45 ms	Featured Events (Featured Events)	6,510.91 ms
6,744.36 ms	Footer (HTML Content)	0.05 ms

A Quick Note for Custom Blocks

```
protected override void OnLoad( EventArgs e )
{
    //Custom Logic
    base.OnLoad( e );
}
```

The base.OnLoad method calculates these times, if this method call is before your custom logic for the block it will list your custom block's load time as belonging to the next block in the list.

When Do You Need to Optimize Your Code?

Report a Staff Member Uses Twice a Year	Report Small Group Leaders Use Weekly	Home Page of Your Website
5 second load time is acceptable	Maybe consider refactoring/ optimizing	5 second load time is unacceptable

Debugging Lava

Identifying Problematic Lava

Quick Ways to Verify Problematic Lava

- **Remove the Section of Lava**
  Verify the page/block loads quickly without the suspect Lava
- **Test in Another Environment**
  Rule out other factors by testing the same Lava in another environment
- **Adjust Cache Settings**
  Pages and Blocks like Content Channel View and Content Channel Item View have different settings for caching

Performance in Lava Hinges on Two Things

Number of Times the Database is Hit

- Talking to the database takes time
- A lot of Lava that talks to the database is likely making multiple calls and getting more data than we need

Amount of Data You're Processing in Memory

- Data rendered does not equal data processed
- Without caching, you're processing all your data on every page load

Most Likely Causes of Lava Performance Issues

- Inefficient SQL in SQL Tag
- Complex Entity Expressions
- Serialization and Deserialization
- Lava Field Type
- Commands that Write to the Database (Interaction Write, Workflow Activate)

Lava Tuning

Fixing the Problem

Leadership Team Staff Page

7 members in the group, how many calls to the database does this lava make?

8? 15? 27

```
{% groupmember where:'GroupId == 500' %}
  <div class="row">
    {% for gm in groupmemberItems %}
      <div class="col-xs-12 col-md-4">
        <img src="/GetImage.ashx?Guid={{gm.Person.Photo.Guid}}"
/>
        {{gm.Person.FullName}} - {{gm.Person |
Attribute:'Position' }}
      </div>
    {% endfor %}
  </div>
```

Block Takes 687ms to Load

Every time the page is loaded

Server Block OnLoad	0.05 ms	
Staff (HTML Content)	687.92 ms	

Additional Entity Command Options

Disable Attribute Prefetch

Starting in v15 Rock will automatically load all attributes for the collection of entities.

```
{% entity disableattributeprefetch:'true' %}
```

Prefetch Attributes

To limit the attributes returned, provide a comma separated list of attribute keys

```
{% person prefetchattributes:'Employer' %}
```

Select

Only returns the desired fields

```
{% group select:'new ( Id, Members.Count() AS MemberCount )' %}
```

Expression

Allows any valid LINQ for advanced filtering

```
{% person expression:'PhoneNumbers.Count() > 1' %}
```

Disable Attribute Prefetch

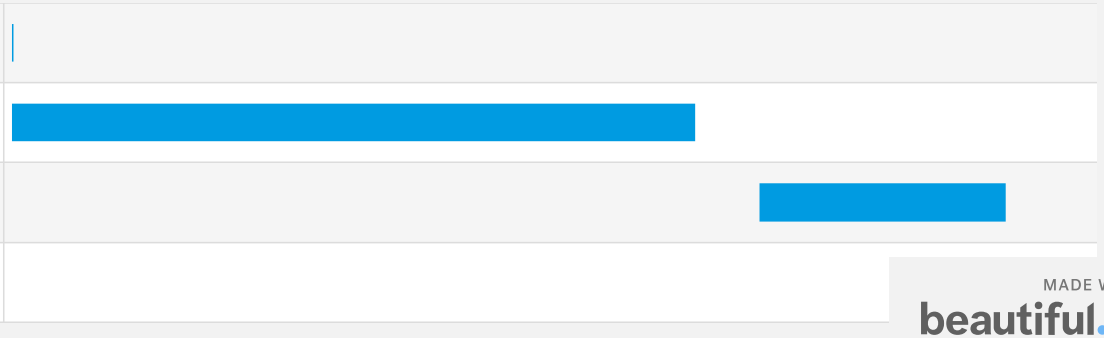
```
{% groupmember where:'GroupId ==
500'disableattributeprefetch:'true'%}
  <div class="row">
    {% for gm in groupmemberItems %}
      //...
    {% endfor %}
  </div>
{% endgroupmember %}
```

Server Block OnLoad	0.05 ms	
Staff (HTML Content)	687.92 ms	
Attribute (HTML Content)	285.64 ms	

Person Entity With Expression and Prefetch

```
{% person expression:'Members.Any(GroupId == 500)'
   prefetchattributes:'Position' %}
   {% for p in personItems %}
      //...
   {% endfor %}
{% endperson %}
```

Server Block OnLoad	0.05 ms
Staff (HTML Content)	687.92 ms
Attribute (HTML Content)	285.64 ms
Person Entity (HTML Content)	64.81 ms



Lava Cache Tag

```
{% cache key:'staff-members' duration:'86400' %}
  {% groupmember where:'GroupId == 500' %}
     //...
  {% endgroupmember %}
{% endcache %}
```

Cache Tag (HTML Content)	3.12 ms

Lava Cache Tag

Use **twopass** and the **raw** tag for dynamic elements

```
{% cache key:'welcome-message' duration:'3600' twopass:'true' %}
  {% raw %}Welcome, {{CurrentPerson.NickName}}!{% endraw %}
  Upcoming Events <br/>
  {% calendarevents calendarid:'1' audienceids:'151,152' %}
    {% for item in EventScheduledInstances %}
        {{ item.Name }} on {{ item.Date }} at {{ item.Time }}
    {% endfor %}
  {% endcalendarevents %}
{% endcache %}
```

Persisted Datasets

Use Lava to build a JSON data structure

Name •

Events - 20s

Active

Access Key •

twenties_events

Description

Data about events for the ministry.

Build Script

```
{% assign audience = 2121 %}
{% assign calendar = 1 %}
{% sql return:'occurrences' %}
DECLARE @EIOEntityTypeId int = 283;
DECLARE @ECIEntityTypeId int = 270;
DECLARE @EventCalendarId int = 1;

DECLARE @EIOPivot varchar(MAX) = (SELECT STRING_AGG(CONCAT('[', [Key], ']'), ', ')
                                  FROM (SELECT DISTINCT EntityTypeId, [Key]
                                        FROM AttributeValue
                                                 INNER JOIN Attribute ON AttributeValue.AttributeId = Attribute.Id
                                        WHERE EntityTypeId = @EIOEntityTypeId) AS EventItemOccAttrs
                                  GROUP BY EntityTypeId);
DECLARE @EIOSelect varchar(MAX) = (SELECT STRING_AGG(CONCAT( 'MAX([', [Key], ']) AS [', [Key], ']'), ', ')
                                  FROM (SELECT DISTINCT EntityTypeId, [Key]
                                        FROM AttributeValue
                                                 INNER JOIN Attribute ON AttributeValue.AttributeId = Attribute.Id
```

Persisted Datasets

```
{% assign data = 'twenties_events' | PersistedDataset %}
```

Page can pull the JSON data at one time instead of redoing the calculation for every person that visits the page.

Replacing Entity Commands with SQL

Occasionally it is more efficient to write SQL instead of using an Entity Command

- Large Dataset Filtered on Related Entity
- Required Data is an Aggregate
- Very Few Columns Are Required

Two Blocks Print the Same Content

Workflow for Web Requests (Bugs, Changes, New Features)

Lava

62	0	62
Bug Reports Total	Bug Reports Open	Bug Reports Resolved

SQL

62	0	62
Bug Reports Total	Bug Reports Open	Bug Reports Resolved

Server Block OnLoad	0.06 ms
Lava (HTML Content)	971.29 ms
SQL (HTML Content)	25.46 ms

Lava Version Loads All Workflows of Type

```
{% assign workflows = '[]' | FromJSON %}
{% assign activeBugReports = '[]' | FromJSON %}
{% workflow where:'WorkflowTypeId == 272 && Status == "Completed"' %}
    {% for wrkflw in workflowItems %}
        {% assign type = wrkflw | Attribute:'RequestType','RawValue' %}
        {% if type == 'bug' %}
            {% assign workflows = workflows | AddToArray:wrkflw %}
            {% assign status = wrkflw | Attribute:'RequestStatus','RawValue' %}
            {% if status == 'Open' %}
                {% assign activeBugReports = activeBugReports | AddToArray:wrkflw %}
            {% endif %}
        {% endif %}
    {% endfor %}
{% endworkflow %}
```

SQL Version Loads Required Stats Only

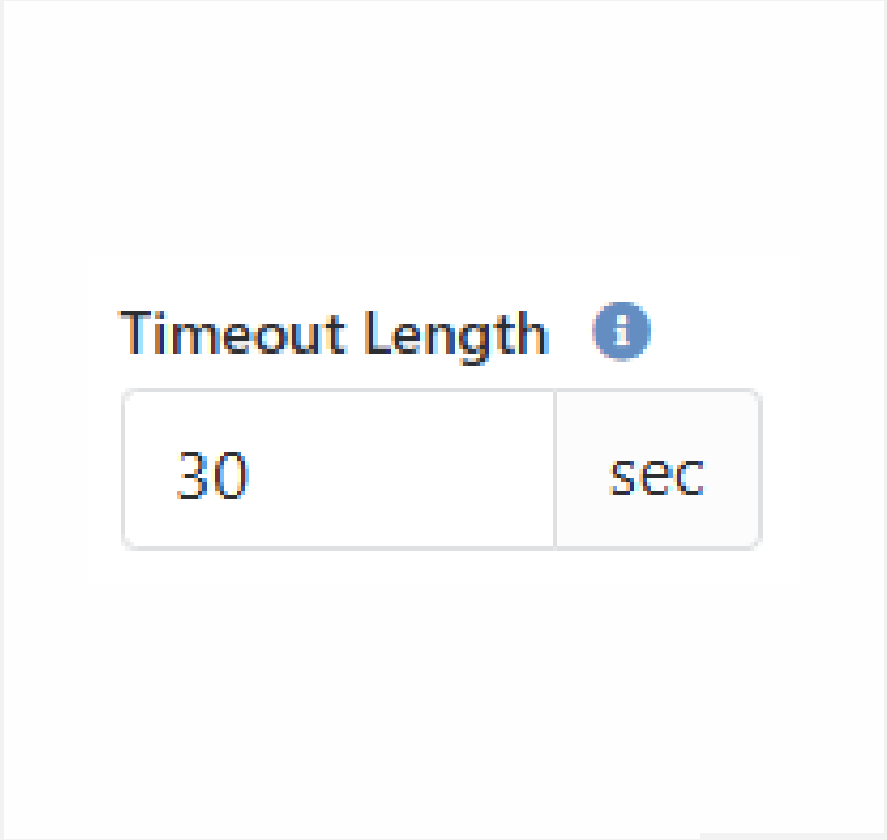
```
SELECT WorkflowTypeId, RequestStatus.Value, COUNT(*) AS Ct
FROM Workflow
   INNER JOIN AttributeValue RequestStatus ON RequestStatus.AttributeId = 19919 AND
                                              RequestStatus.EntityId = Workflow.Id
   INNER JOIN AttributeValue RequestType   ON RequestType.AttributeId = 19716 AND
                                              RequestType.EntityId = Workflow.Id
WHERE WorkflowTypeId = 272
  AND Status = 'Completed'
  AND RequestType.Value = 'bug'
GROUP BY WorkflowTypeId, RequestStatus.Value
```

Debugging SQL

Tools for Identifying Performance Issues in SQL

High Timeout Lengths Can Indicate SQL Performance Issues

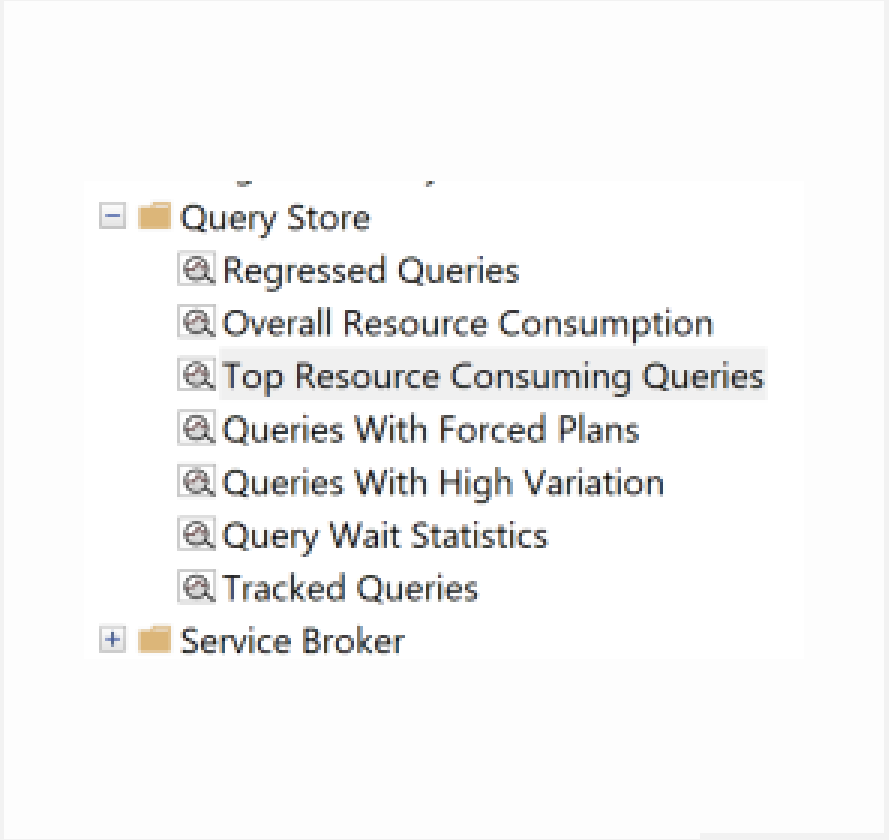
If your Dynamic Data block requires a timeout length greater than a minute, it could indicate that your SQL is not optimized. There are some exceptions for complex queries running calculations on large tables.



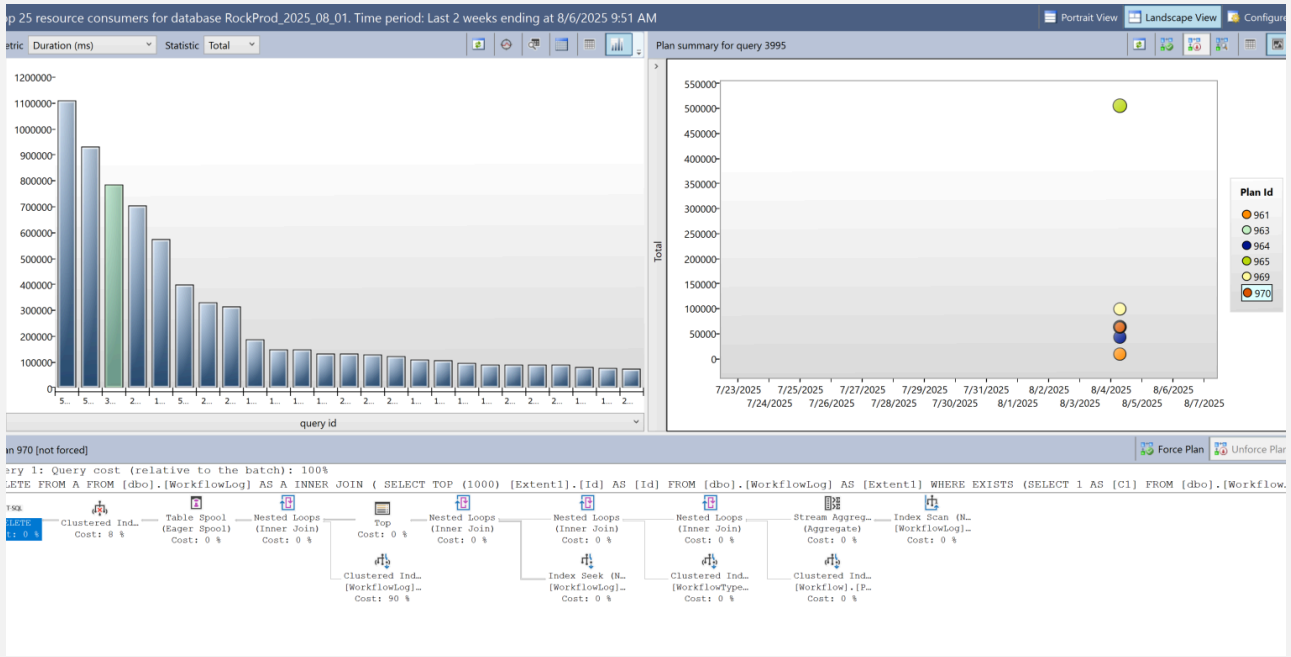
Use SSMS Query Store to Identify Top Resource Consuming Queries

SQL Server Management Studio will list your top resource consuming queries and help you identify if some are using an inefficient query plan or if you need to re-write your SQL.

Update the configuration to your desired timeframe.



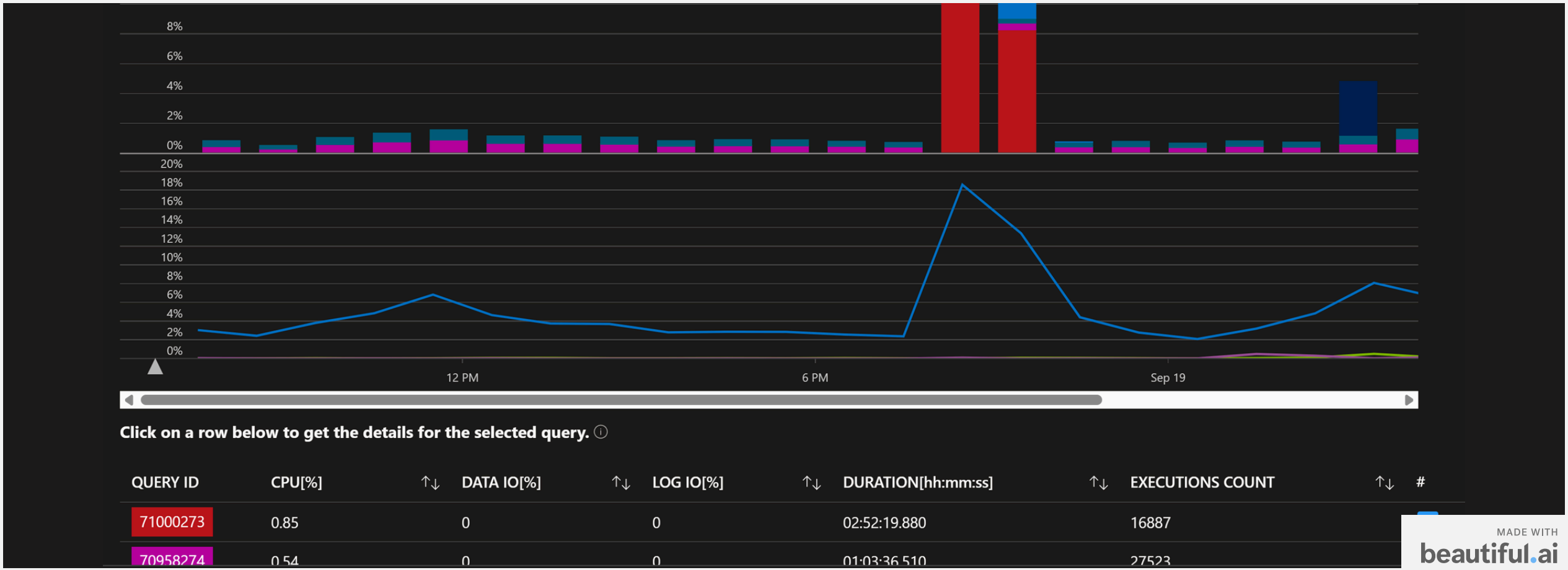
You Can Force a Query Plan to Improve Performance



When a query has multiple plans and there is high variation in them, we can force a specific plan that performs better than others.
Be careful when doing this because the dynamic parts of your query could be better served using different query plans.

Azure's Query Performance Insight Provides the Same Information

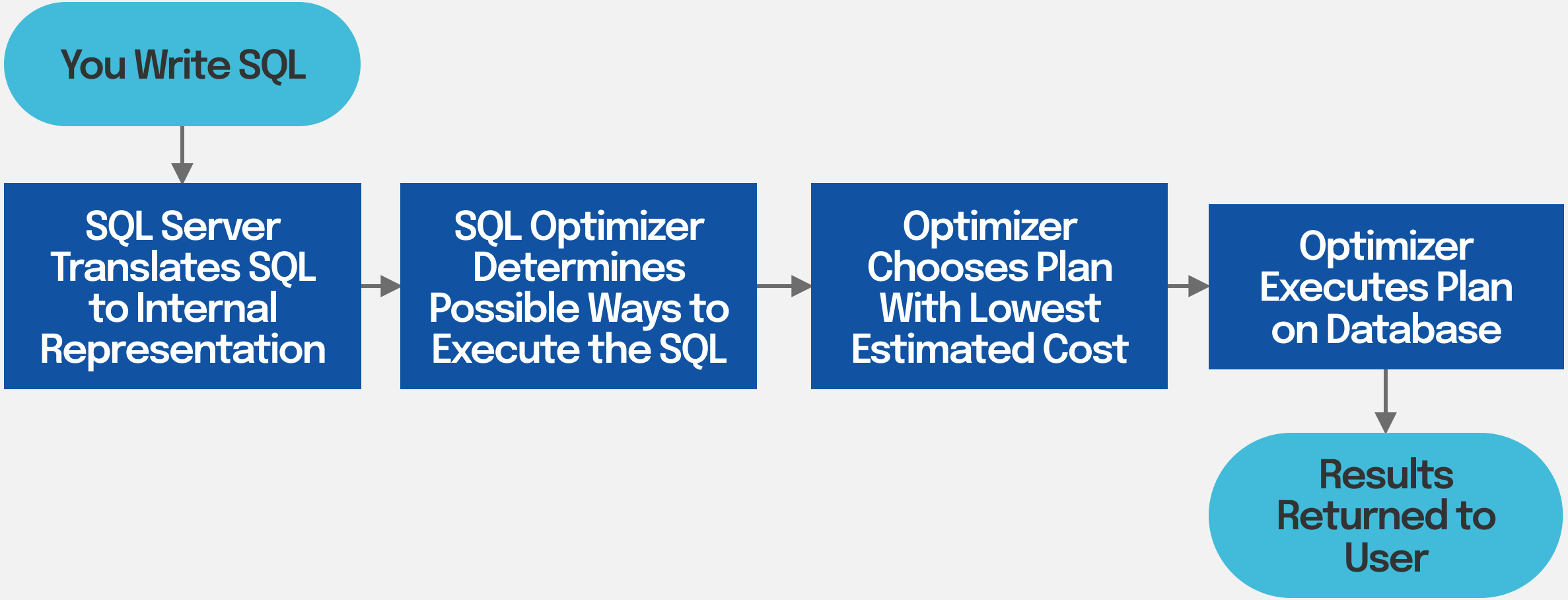
Query Store must be configured and running on your database.



SQL Tuning

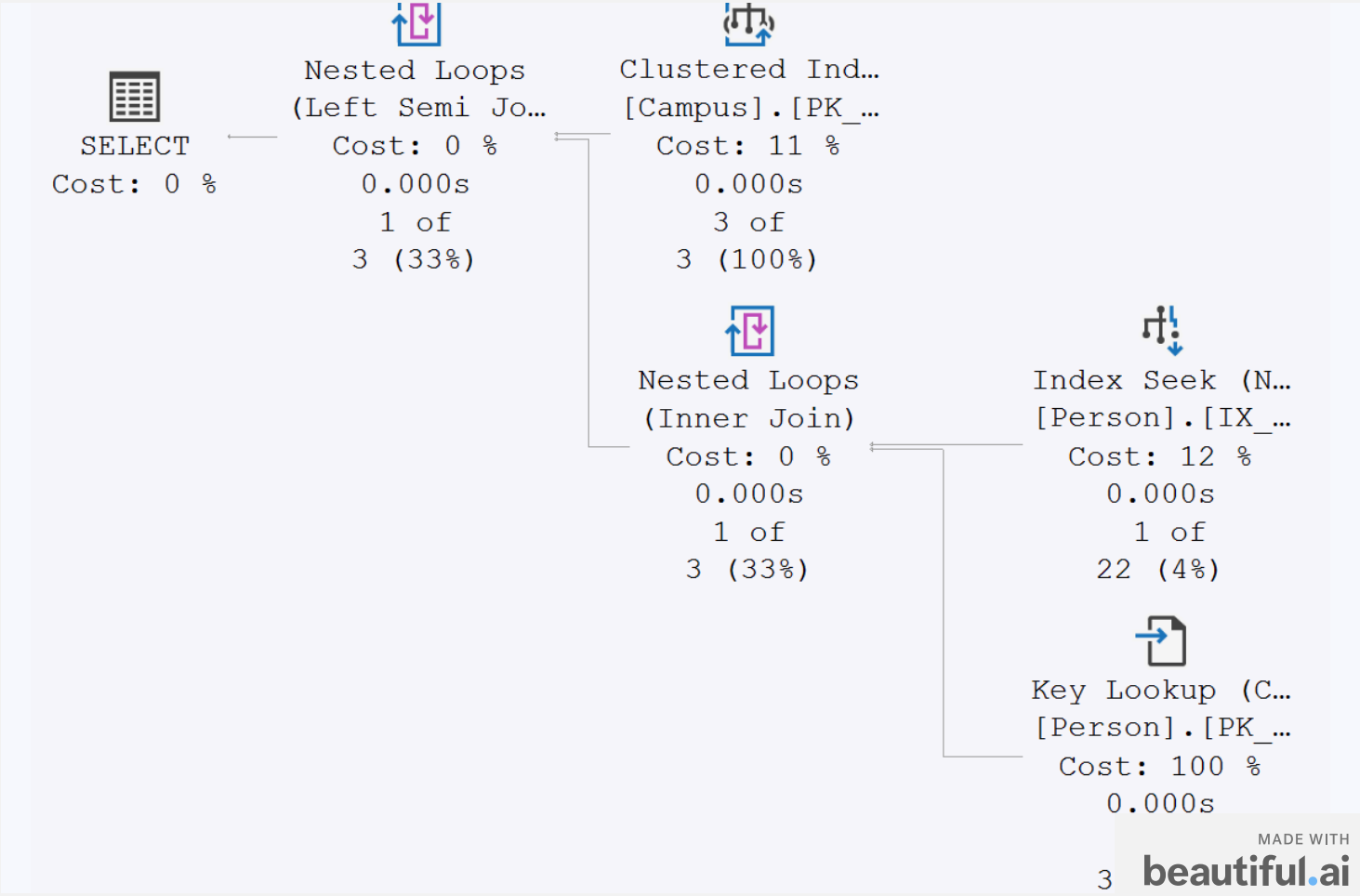
Controlling the Execution Plan

How SQL is Processed

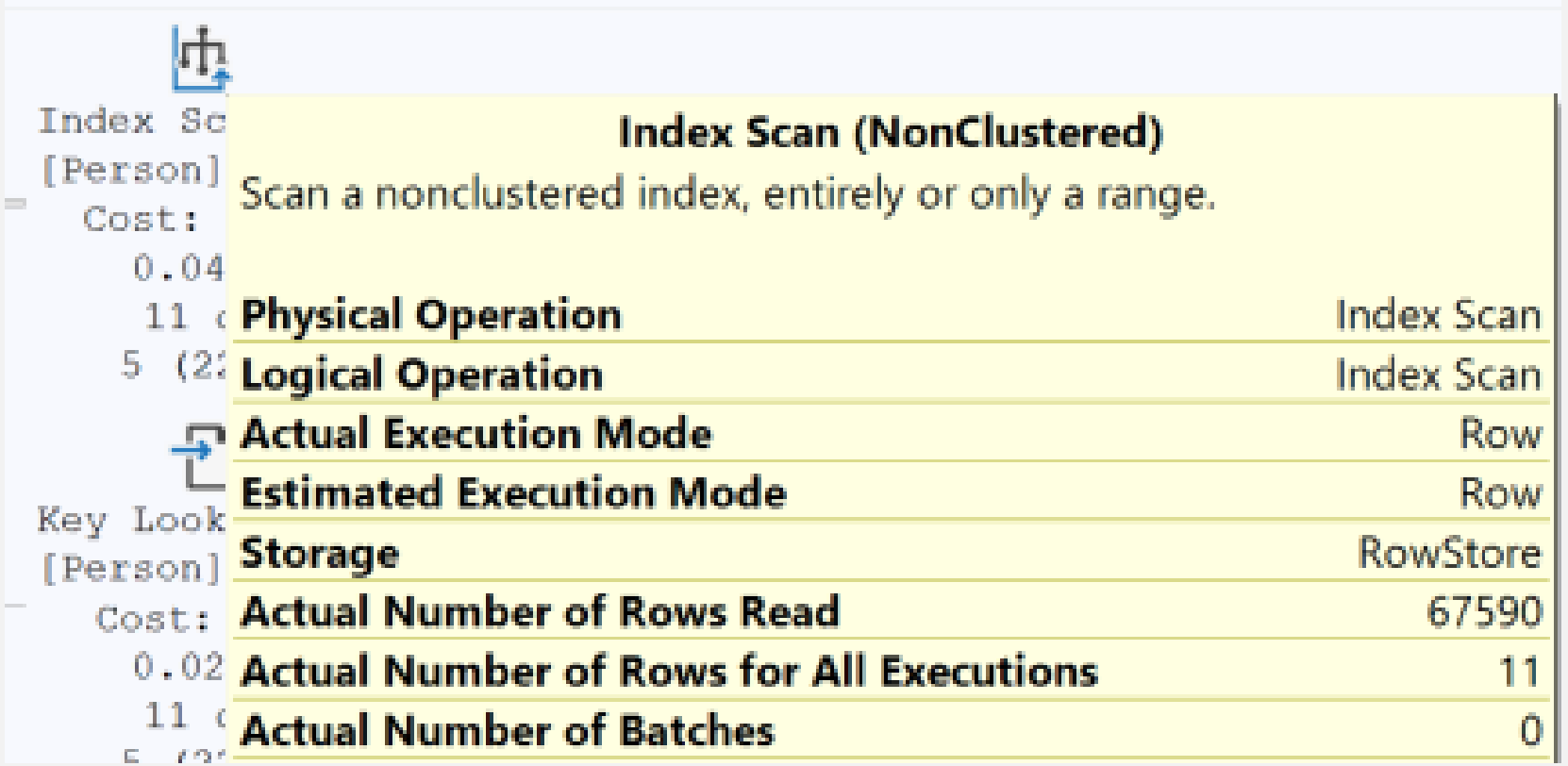


The Execution Plan Defines How a Query Will Be Executed

- **Data flow is read bottom right to top left**
  Call order is read left to right
- **Arrow size indicates data size**
- **Running multiple queries allows you to compare estimated cost of execution**



Hover Over a Step in the Plan to View Details of the Execution



Next Three Slides Cut for Time but Included for Reference

They define the different operations you will see when viewing SQL Execution Plans

Overview of Possible Join Operations

Nested Loops

For each row in the top data set, perform one search of the other data set for matching values.

Hash Match

Using each row in the top (build) data set, create a hash table, which will then be probed using the rows from the bottom (probe) data set to find any matching value.

Merge Join

Read data from both inputs simultaneously and merge the two inputs, joining each matching row value. This requires both inputs to be sorted on the join column(s).

Overview of Possible Data Reading Operations

Index Scan

The query reads every row in the index to decide what to return

Index Seek

The query uses the index to locate the exact records to return

Key Lookup

Occurs when data not covered by the index is requested

Table Scan

The query reads every row in the table to decide what to return

Only occurs when there is no clustered index, all Rock tables have a clustered index

Other Common Operations

- **Aggregate**
  Calculates Aggregate Expressions (MIN, MAX, AVG, etc.)
- **Assert**
  Verifies rows meet conditions of a CHECK or FOREIGN KEY constraint
- **Bitmap Filter**
  Reduces the rows passed into the build set of a Hash Match to speed up Parallelism
- **Compute Scalar**
  Calculates a new value for a row (CAST)
- **Concatenation**
  Joins rows from a UNION ALL
- **Sort**
  Sorts the rows, usually for an ORDER BY but also can be done before a Merge Join

There Are Two Types of Indexes

- **Clustered Index**
  Stores the table in sorted order based on the values of the index column(s), should be the PK.

- **Non-clustered Index**
  Additional data structure with the key values for the index and a pointer (locator) to the data row.

Evaluating the Operation

Rock Person Table: 67995 Rows

```
SELECT * FROM Person
WHERE LastName = 'Cooksey'
```

Actual Number of Rows Read	67995
Actual Number of Rows for All Executions	11

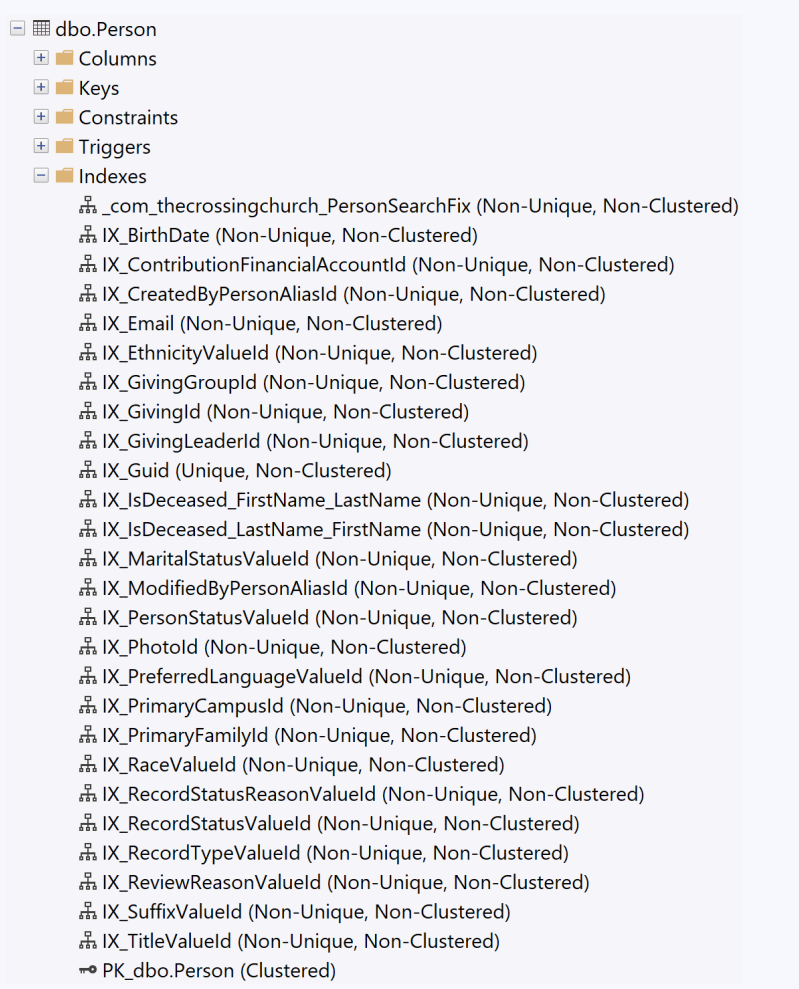
Evaluating the Operation

Rock Person Table: 67995 Rows

```
SELECT * FROM Person
WHERE LastName = 'Cooksey'
  AND (IsDeceased = 0 OR IsDeceased = 1)
```

Actual Number of Rows Read	11
Actual Number of Rows for All Executions	11

Why Did Adding IsDeceased Change the Performance?




```sql
SELECT * FROM Person
WHERE LastName = 'Cooksey'
```

IX_ISDECEASED_FIRSTNAME_LASTNAME

```sql
SELECT * FROM Person
WHERE LastName = 'Cooksey'
  AND (IsDeceased = 0 OR IsDeceased = 1)
```

IX_ISDECEASED_LASTNAME_FIRSTNAME

Common Issues With SQL Queries

1. Execution Plan Uses an Index that Performs Poorly
2. Arithmetic Expressions or Functions in the Filter Clause
3. Query/Statement Locks the Table
4. Execution Plan Uses an Inefficient Join Order
5. Implicit Data Type Conversions are Occurring
6. Extraneous Data is Pulled

To Enable the Use of an Index

Provide a reasonably selective condition on the leading column of the index

```
SELECT * FROM Attribute
WHERE EntityTypeQualifierColumn = 'BlockTypeId'
```

```
SELECT * FROM Attribute WHERE EntityTypeId = 9
AND EntityTypeQualifierColumn = 'BlockTypeId'
```

To Enable the Use of an Index

Provide a reasonably selective condition on the leading column of the index

```
SELECT * FROM Attribute
WHERE EntityTypeQualifierColumn = 'BlockTypeId'
```

Results in a Clustered Index Scan (PK_Attribute)

```
SELECT * FROM Attribute WHERE EntityTypeId = 9
AND EntityTypeQualifierColumn = 'BlockTypeId'
```

To Enable the Use of an Index

Provide a reasonably selective condition on the leading column of the index

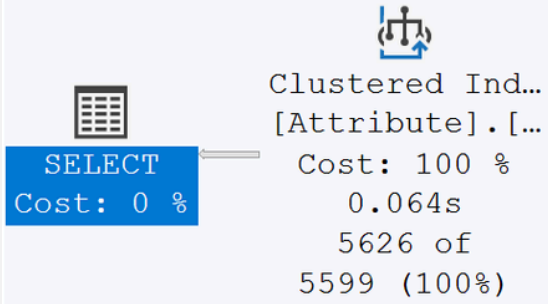
```
SELECT * FROM Attribute
WHERE EntityTypeQualifierColumn = 'BlockTypeId'
```

Results in a Clustered Index Scan (PK_Attribute)

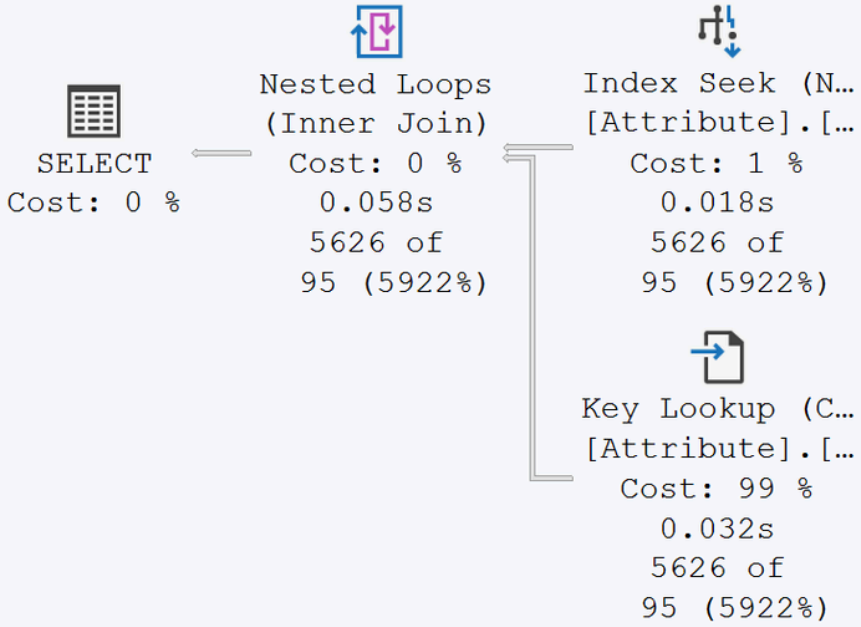
```
SELECT * FROM Attribute WHERE EntityTypeId = 9
AND EntityTypeQualifierColumn = 'BlockTypeId'
```

Results in an Index Seek since the leading field of the index is used (IX_EntityTypeId_EntityTypeQualifierColumn_EntityTypeQualifierValue_Key)

Query 1: Query cost (relative to the batch): 80%
SELECT * FROM [Attribute] WHERE [EntityTypeQualifierColumn]=@1



Query 2: Query cost (relative to the batch): 20%
SELECT * FROM [Attribute] WHERE [EntityTypeQualifierColumn]=@1 AND [EntityTypeId]=@2



A Condition That is Not Selective Enough Will Cause the Optimizer to Abandon the Index

```
SELECT * FROM Person
WHERE LastName LIKE 'C%'
  AND (IsDeceased = 0 OR IsDeceased = 1)
```

Actual Number of Rows Read	67995
Actual Number of Rows for All Executions	4594

Exercise Caution When Changing Indexes in Rock

You likely have full control over your Rock DB, don't destroy it.

- **The Current Indexes Exist for a Reason**
  It is not recommended that you delete or modify any indexes created by the Rock App Code
- **Creating New Indexes Effects All Database Operations**
  The data structure for an index has to be modified any time data is created, updated, or deleted in the table
- **Other Queries May Begin to Use a New Index**
  Introducing a new index could cause SQL that once ran effectively to use the new index which could be sub-optimal for its performance
- **Additional Indexes Can Break Migrations**

Monitor the Database Maintenance Job to Assert Indexes are Properly Maintained

Database Maintenance	9/19/2025 1:05 AM	9/19/2025 1:05 AM	5m 1s	Success	Summary:

- Rebuild [__MigrationHistory].[PK_dbo.__MigrationHistory] (49ms)
- Rebuild [AttributeValue].[IX_ValueChecksum] (1,960ms)
- Rebuild [Auth].[IX_CreatedByPersonAliasId] (22ms)
- Rebuild [Block].[PK_dbo.Block] (69ms)
- Rebuild [ConnectionRequestWorkflow].[IX_ConnectionWorkflowId] (25ms)
- Rebuild [ContentChannelItem].[nci_wi_ContentChannelItem_86627917B6BF611523074114A8003670] (113ms)
- Rebuild [ContentChannelItem].[nci_wi_ContentChannelItem_FDE8C34797B59F628D4F422FCD8045A6] (2,100ms)
- Rebuild [EntitySetItem].[IX_CreatedByPersonAliasId] (14ms)
- Rebuild [EntitySetItem].[IX_EntitySetId] (12ms)
- Rebuild [EntitySetItem].[IX_EntitySetId_EntityId] (12ms)

Avoid Arithmetic Expressions and Functions in Filter Clauses

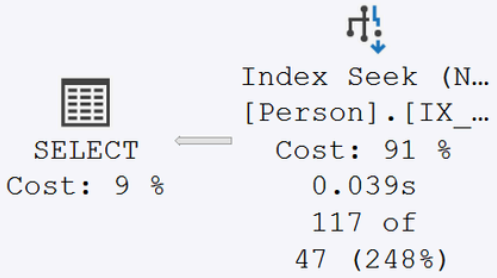
A Calculation in the WHERE, ON, or HAVING Clauses
Can Cause the Optimizer to Abandon the Index

```
SELECT FirstName, LastName FROM Person WHERE (IsDeceased IS
NULL OR IsDeceased = 0) AND 'Cook' = LEFT(LastName, 4)
```

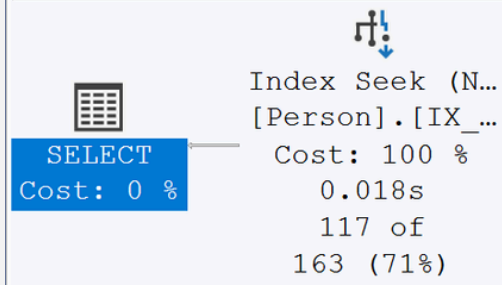
```
SELECT FirstName, LastName FROM Person WHERE (IsDeceased IS
NULL OR IsDeceased = 0) AND LastName LIKE 'Cook%'
```

Avoid Arithmetic Expressions and Functions in Filter Clauses

```
Query 1: Query cost (relative to the batch): 99%
SELECT FirstName, LastName FROM Person WHERE (IsDeceased IS NULL OR IsDeceased = 0) AND 'Cook' = LEFT(LastName, 4)
```




```
Query 2: Query cost (relative to the batch): 1%
SELECT FirstName, LastName FROM Person WHERE (IsDeceased IS NULL OR IsDeceased = 0) AND LastName LIKE 'Cook%'
```



Avoid Arithmetic Expressions and Functions in Filter Clauses

Actual Number of Rows Read	67411
Actual Number of Rows for All Executions	117

Uses the IX_IsDeceased_FirstName_LastName Index and Reads 99.7% of the Rows in the Table

Actual Number of Rows Read	117
Actual Number of Rows for All Executions	117

Uses the IX_IsDeceased_LastName _FirstName Index and Only Reads the Relevant Rows

A Query/Statement That Locks Rows or the Table Will Block Other Queries

Read Locks

Read locks cooperate with other read locks and do not block each other.

Modifying Locks

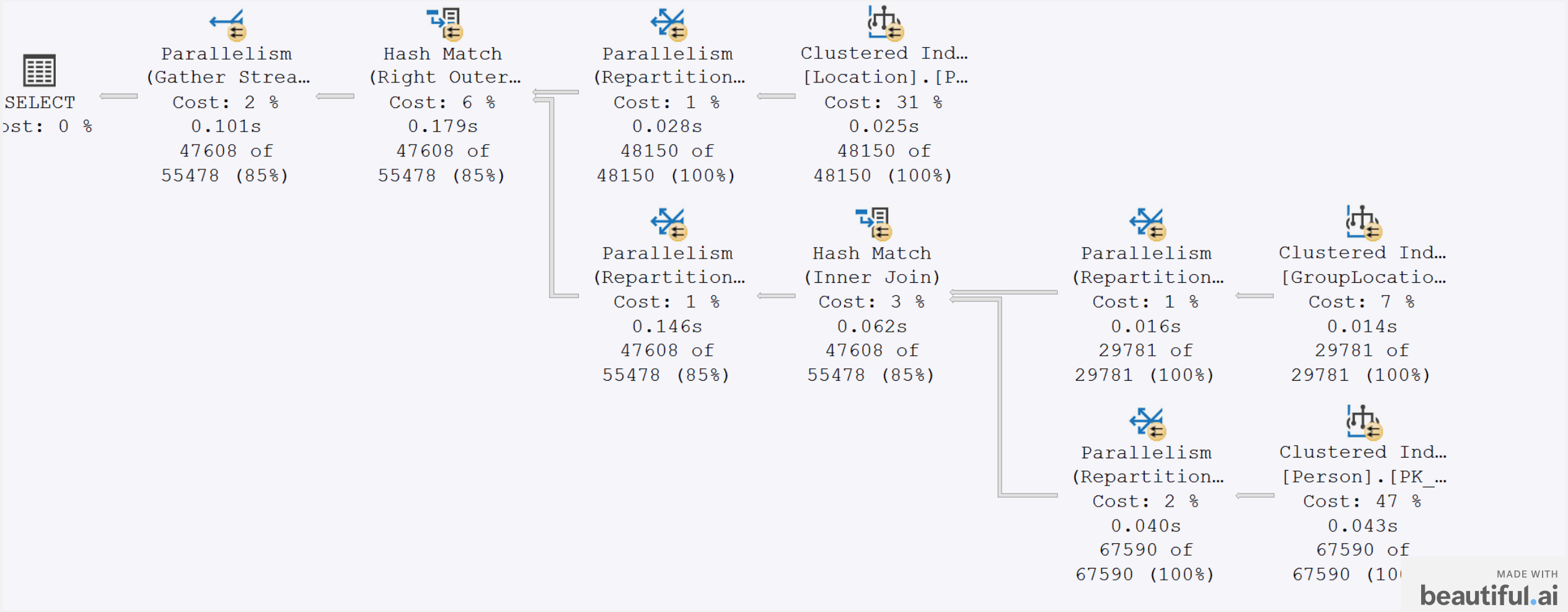
When two sessions attempt to modify the same piece of data, one will be first and lock the data. The other session will be blocked until the operation is complete.

Forcing the Join Order You Want

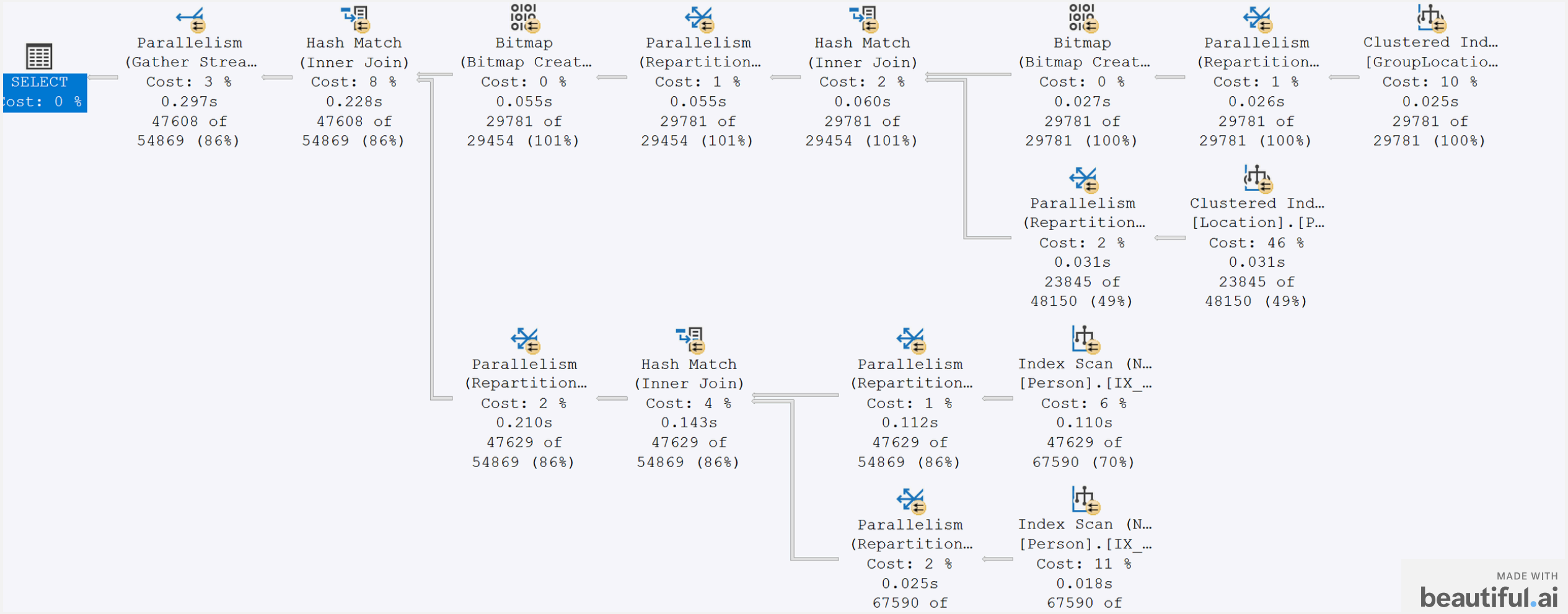
```
SELECT L.State, L.City, P.LastName, P.FirstName
  FROM Person P
      LEFT OUTER JOIN GroupLocation GL ON PrimaryFamilyId = GL.GroupId
      LEFT OUTER JOIN Location L ON GL.LocationId = L.Id
  WHERE GroupLocationTypeValueId = 19
```

```
SELECT L.State, L.City, P.LastName, P.FirstName
  FROM Person P
      INNER JOIN GroupLocation GL ON PrimaryFamilyId = GL.GroupId
      INNER JOIN Location L ON GL.LocationId = L.Id
  WHERE GroupLocationTypeValueId = 19
```

First Query Starts With Entire Person Table Join to Group Location to Location



Second Query Starts With Group Location Table Join to Locations to Person Indexes



Use a Program Like Plan Explorer to Confirm Actual Statistics

sults

	Est Cost %	Compile Time	Duration	UDF Duration	CPU	UDF CPU	Est CPU Cost %	Reads
JOI...	60.0%	52	335	0	825	0	43.6%	7,308
Grou...	40.0%	37	315	0	613	0	56.4%	4,799

SOLARWINDS®

Compare Against the Same Datatype to Avoid Implicit Conversions

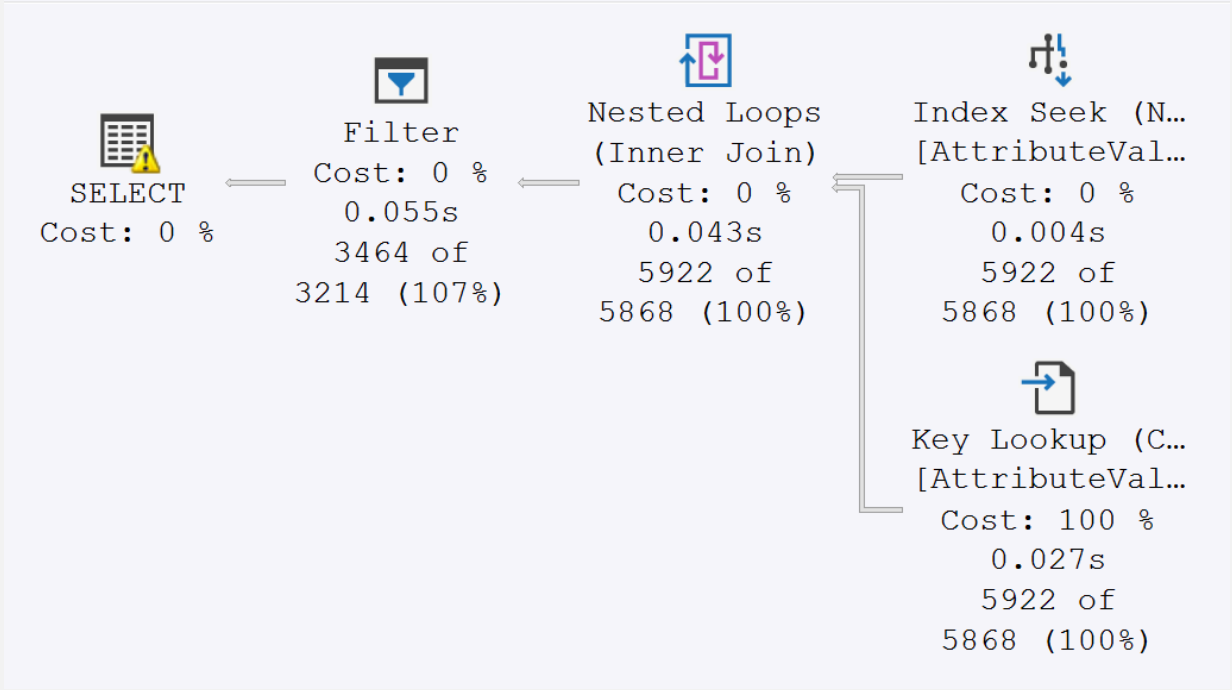
```
SELECT Id, EntityId, Value, AttributeId FROM AttributeValue
WHERE AttributeId = 36301 AND Value > 18
```

```
SELECT Id, EntityId, Value, AttributeId FROM AttributeValue
WHERE AttributeId = 36301 AND ValueAsNumeric > 18
```

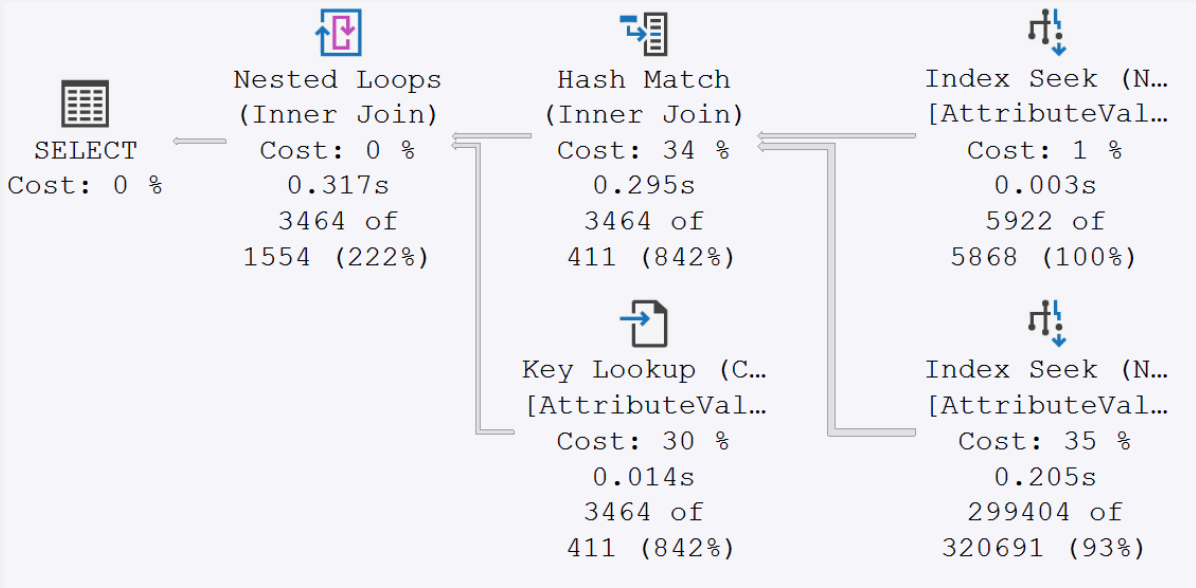
Statistics Comparing the Queries Not Quite Expected

Filter Column	Estimated Cost%	Compile Time	Duration	CPU	Reads	Rows
Value	85.8%	9	44	44	25,235	825
ValueAsNumeric	14.2%	7	200	200	4,639	825

Index Seek for ValueAsNumeric Is Passing a Lot of Data



Value



ValueAsNumeric

Knowing the Data, Value Should Never Be Greater Than 30

Value is a Person Attribute saving the number of Sundays attended kids ministry in last 6 months.

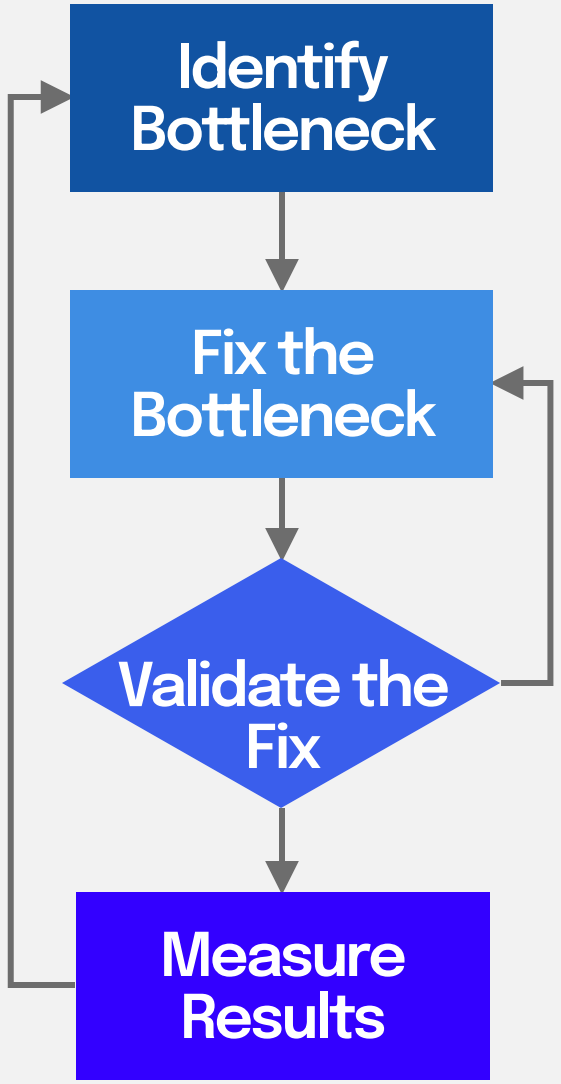
```
SELECT Id, EntityId, Value, AttributeId FROM AttributeValue
WHERE AttributeId = 36301 AND Value > 18 AND Value < 30
```

```
SELECT Id, EntityId, Value, AttributeId FROM AttributeValue
WHERE AttributeId = 36301 AND ValueAsNumeric > 18
  AND ValueAsNumeric < 30
```

Now Statistics Match What We Expected

Filter Column	Estimated Cost%	Compile Time	Duration	CPU	Reads	Rows
Value	97.9%	9	47	47	25,235	825
ValueAsNumeric	2.1%	10	31	31	3,677	825

Tuning SQL is a Process



Keep Result Sets Small

```
SELECT * FROM Person
```

```
SELECT * FROM Person WHERE IsDeceased = 0 AND LastName LIKE
'Cook%'
```

```
SELECT Id, FirstName, LastName FROM Person
WHERE IsDeceased = 0 AND LastName LIKE 'Cook%'
```

Comparing the Actual Costs

Imagine if this query were the innermost subquery in a complex report

Query	Estimated Cost%	Compile Time	Duration	CPU	Reads	Rows
All Columns, No Filter	84.8%	21	1,319	1,296	3,883	67,590
All Columns, Filtered	15.1%	31	3	3	404	117
Required Columns, Filtered	0.1%	8	0	0	5	117

SQL Tuning

Best Practices for Designing Efficient Queries

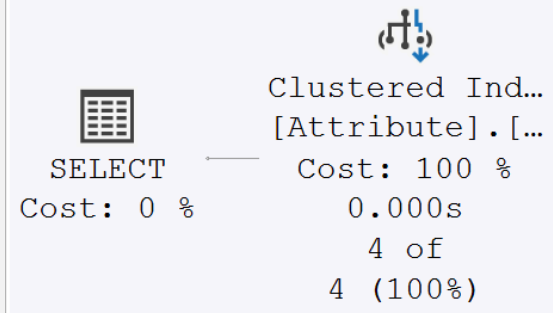
Evaluate Different Filter Operators To Find Best Result

```
SELECT * FROM Attribute WHERE Id IN (1271, 1272, 1273, 1274)
```

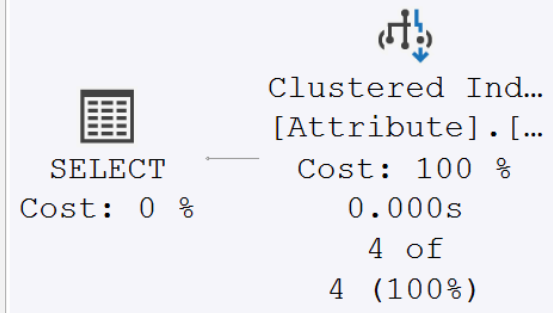
```
SELECT * FROM Attribute
WHERE Id = 1271 OR Id = 1272 OR Id = 1273 OR Id = 1274
```

```
SELECT * FROM Attribute WHERE Id BETWEEN 1271 AND 1274
```

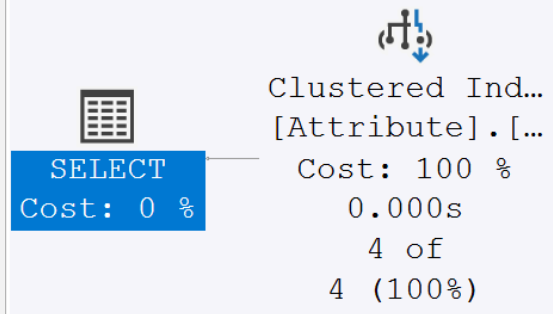
Query 1: Query cost (relative to the batch): 33%
SELECT * FROM Attribute WHERE Id IN (1271, 1272, 1273, 1274)



Query 2: Query cost (relative to the batch): 33%
SELECT * FROM Attribute WHERE Id = 1271 OR Id = 1272 OR Id = 1273 OR Id = 1274



Query 3: Query cost (relative to the batch): 33%
SELECT * FROM [Attribute] WHERE [Id]>=@1 AND [Id]<=@2



Execution Plans Appear to Be the Same Except for How the Optimizer Handled Them

Seek Predicates
[1] Seek Keys[1]: Prefix: [RockProd_2025_08_01].[dbo].[Attribute].Id = Scalar Operator((1271)), [2] Seek Keys[1]: Prefix: [RockProd_2025_08_01].[dbo].[Attribute].Id = Scalar Operator((1272)), [3] Seek Keys[1]: Prefix: [RockProd_2025_08_01].[dbo].[Attribute].Id = Scalar Operator((1273)), [4] Seek Keys[1]: Prefix: [RockProd_2025_08_01].[dbo].[Attribute].Id = Scalar Operator((1274))

IN and OR

Seek Predicates
Seek Keys[1]: Start: [RockProd_2025_08_01].[dbo].[Attribute].Id >= Scalar Operator(CONVERT_IMPLICIT(int,[@1],0)), End: [RockProd_2025_08_01].[dbo].[Attribute].Id <= Scalar Operator (CONVERT_IMPLICIT(int,[@2],0))

BETWEEN

There is No Rule for IN, OR, or BETWEEN Performing the Best

Always Test Your Queries and Apply the Best Condition

Filter Operator	Estimated Cost%	Compile Time	Duration	CPU	Reads	Scans	Rows
IN	33.3%	1	2	16	12	4	4
OR	33.3%	1	1	0	12	4	4
BETWEEN	33.3%	2	1	0	5	1	4

Avoid Non-sargable Search Conditions

Search ARGument ABLE - Optimizer can perform an efficient SEEK

Type	Search Conditions
Sargable	Inclusion conditions =, >, >=, <, <=, BETWEEN, some LIKE conditions ('<literal>%')
Non-sargable	Exclusion conditions <>, !=, !>, !<, NOT EXISTS, NOT IN, NOT LIKE, some LIKE conditions ('%<literal>' or '%<literal>%')

Temporary Tables and Table Variables Are Almost Identical

```
SELECT Id, FirstName, LastName, Age INTO #TempFiveYrOlds FROM Person WHERE Age = 5
```

```
DECLARE @TempFiveYrOlds TABLE
                        (
                            Id        INT PRIMARY KEY,
                            FirstName VARCHAR(100),
                            LastName  VARCHAR(100),
                            Age       INT
                        );

INSERT INTO @TempFiveYrOlds SELECT Id, FirstName, LastName, Age
FROM Person WHERE Age = 5
```

Differences Between Temp Table and Table Variable

- **Temporary Table**

- **Has Statistics**
  Inserting data into temp table causes stats to recompile

- **Use Across Scopes**
  Can create in one procedure and use in another

- **Table Variable**

- **No Statistics**
  No stats, no recompile
  Can be inefficient when joining

- **Single Scope**
  Can only use where it is defined

Additional Tips

- **Every Query Will Update the DB Cache to Minimize Physical I/O**

  Working on a local copy of a DB can make a query seem like it performs well because it's reading the same data from the cache

- **Views Are Not Tables**

  Joining complex queries with views inserts the view query into your query making it even more complex

- **Explicitly Define an Object's Owner so the Optimizer Doesn't Have to**

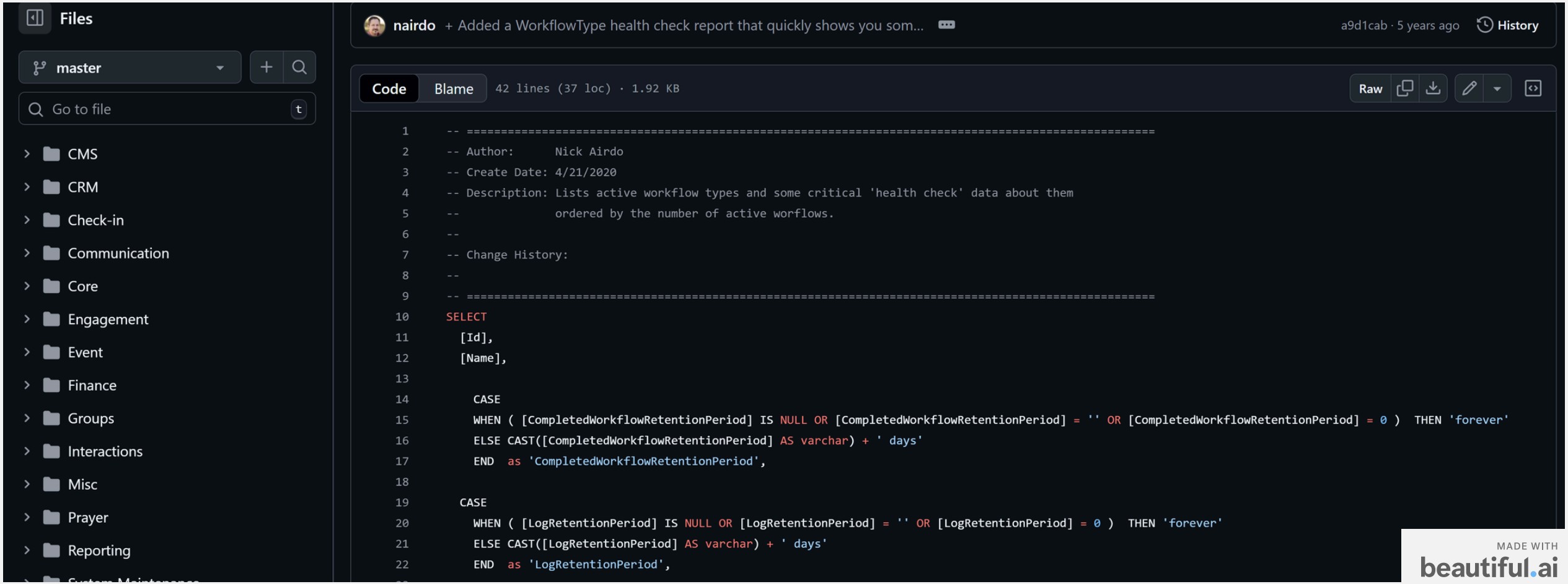
Alternatives to Dynamic Data Blocks and SQL Lava Tag

- **Persisted Datasets**
  Does the data really need to be up to the minute fresh or could it be built every few hours?
- **Metrics**
  For calculating high level stats, consider storing the data as a metric with a SQL source that is calculated on a regular schedule.
- **Job and Entity Attribute Value**
  Instead of calculating a new value for an entity, consider a job running at off-peak times to regularly update a static value for the entity.

Locating Problematic Workflow Types

Rock-SQL-Library Workflow Types Health Check

https://github.com/SparkDevNetwork/Rock-SQL-Library



Identify Workflow Types With a High Number of Concurrent Active Workflows

```
SELECT WorkflowTypeId, Name, ActiveWorkflows
FROM WorkflowType
    INNER JOIN (SELECT WorkflowTypeId, COUNT(*) AS
ActiveWorkflows
                    FROM Workflow
                    WHERE CompletedDateTime IS NULL
                    GROUP BY WorkflowTypeId
    ) AS Workflows ON WorkflowTypeId = WorkflowType.Id
ORDER BY ActiveWorkflows DESC
```

Identify Workflow Types With Longest Lifespan of Workflows

```
DECLARE @today DATETIME = GETDATE();
SELECT WorkflowTypeId, Name, ShortestLifespan, LongestLifespan, AverageLifespan, StandardDeviation
FROM WorkflowType
    INNER JOIN (SELECT WorkflowTypeId,
                MIN(Lifespan)          AS ShortestLifespan,
                MAX(Lifespan)          AS LongestLifespan,
                AVG(Lifespan)          AS AverageLifespan,
                MAX(StandardDeviation) AS StandardDeviation
        FROM (SELECT WorkflowTypeId, ActivatedDateTime, CompletedDate,
                DATEDIFF(DAY, ActivatedDateTime, CompletedDate) AS Lifespan,
                STDEV(DATEDIFF(DAY, ActivatedDateTime, CompletedDate)) OVER (PARTITION BY WorkflowTypeId) AS
StandardDeviation
                FROM (SELECT WorkflowTypeId, ActivatedDateTime,
                             IIF(CompletedDateTime IS NULL, @today, CompletedDateTime) AS CompletedDate
                FROM Workflow) AS Workflows) AS WorkflowLifespan
        GROUP BY WorkflowTypeId) AS WorkflowStats ON WorkflowTypeId = Id
ORDER BY AverageLifespan DESC
```

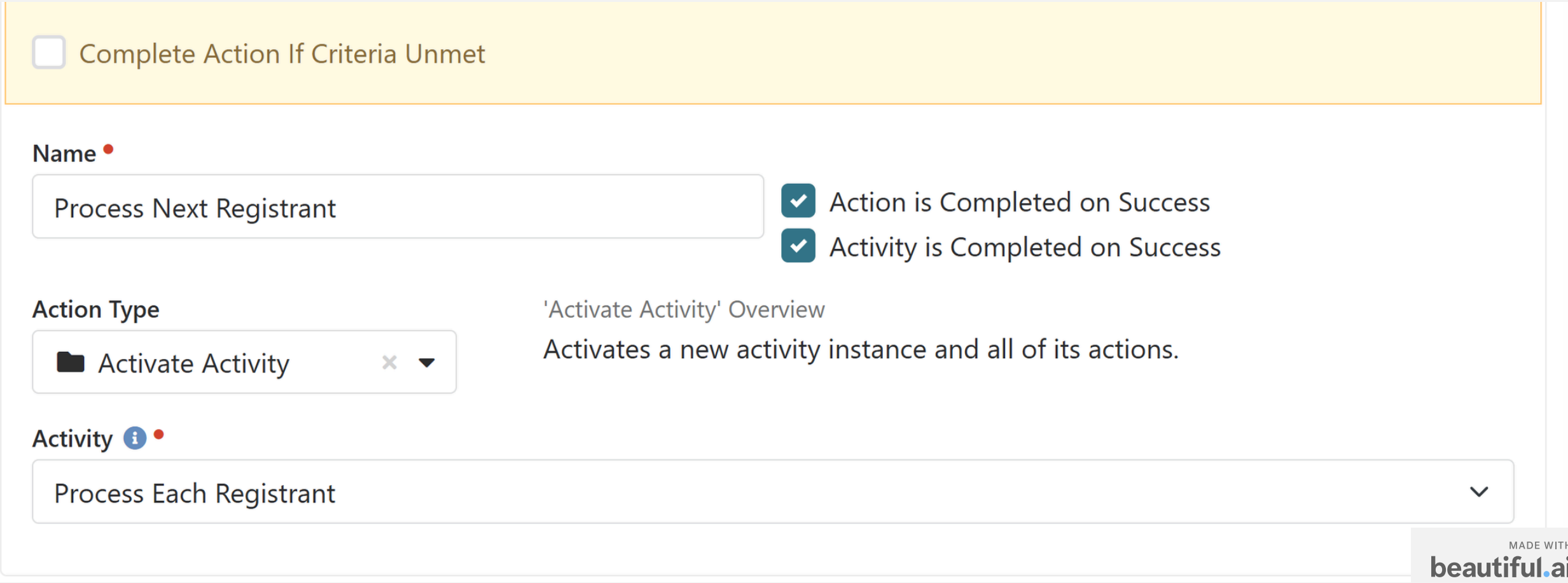
Monitor the Process Workflows Job

Check the Job History, Clean Up Errors in Workflows

```
SELECT ServiceJobId,
       Status,
       AVG(Elapsed)  AS AvgTimeToRun,
       MAX(Elapsed)  AS MaxTimeToRun,
       SUM(HadError) AS RunsWithError,
       COUNT(*)      AS Runs
FROM (SELECT ServiceJobId,
             Status,
             DATEDIFF(SECOND, StartDateTime, StopDateTime) AS Elapsed,
             IIF(StatusMessage LIKE '%error%', 1, 0)       AS HadError
      FROM ServiceJobHistory
      WHERE ServiceJobId = 8) AS JobHistory
GROUP BY ServiceJobId, Status
```

The Activate Activity Action Allows You To Create a Loop in a Workflow

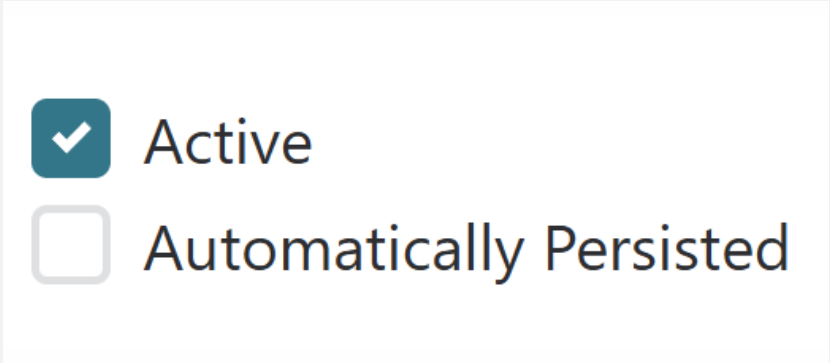
An Infinite Loop Will Crash Rock

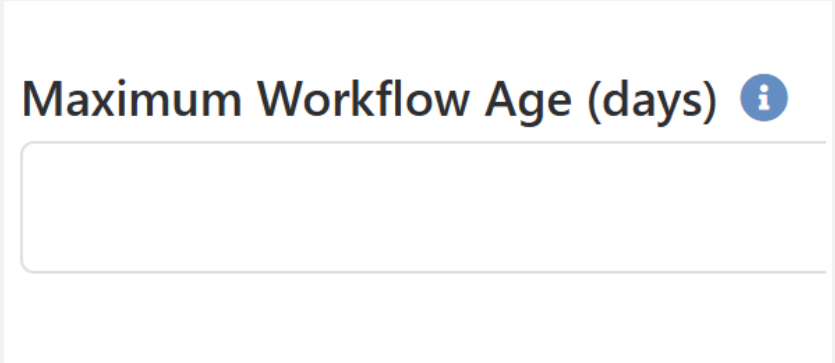


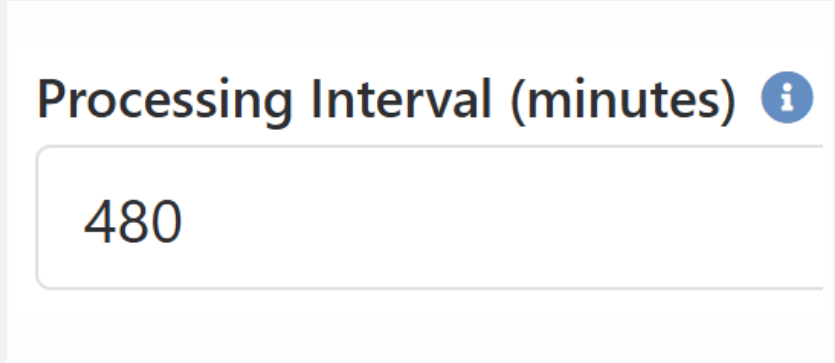
Common Causes of Performance Issues in Workflows

- **Workflow Logging**
  Writing to the DB is one of the most expensive DB Operations
- **Workflow Triggers**
  Performs logic on every entity of the specified type
- **Over Processing**
  Workflows shouldn't live longer than necessary
- **Run SQL Action**
  The issue might be with SQL, not the Workflow itself
- **Run Lava Action**
  The issue might be with Lava, not the Workflow itself

Only Process Workflows When Necessary







Turn off Automatically Persist

Some workflows might not need to ever be saved. Persist workflows only when required.

Set Lowest Max Age Possible

Set all workflows to automatically inactivate as quickly as possible for the process.

Set a High Processing Interval

How often does the process workflow job need to verify there is no action to take on the workflow?

Resources

Red Gate SQL Book

https://assets.red-gate.com/community/books/sql-server-execution-plans-3rd-edition.pdf

SQL Shack Plan Operators

https://www.sqlshack.com/sql-server-execution-plan-operators-part-3/#ToC

SQL Server Performance Tuning

https://www.oreilly.com/library/view/sql-server-2022/9781484288917/

Solarwinds Plan Explorer

https://www.solarwinds.com/free-tools/plan-explorer

Rock SQL Library

https://github.com/SparkDevNetwork/Rock-SQL-Library/tree/master

Slide Deck

https://github.com/courtney