Understanding Query Optimization in SQL Server

Introducing Query Optimization in SQL Server

Nikola Ilic
Data Mozart

@DataMozart | www.data-mozart.com

Overview

“Good enough” performance

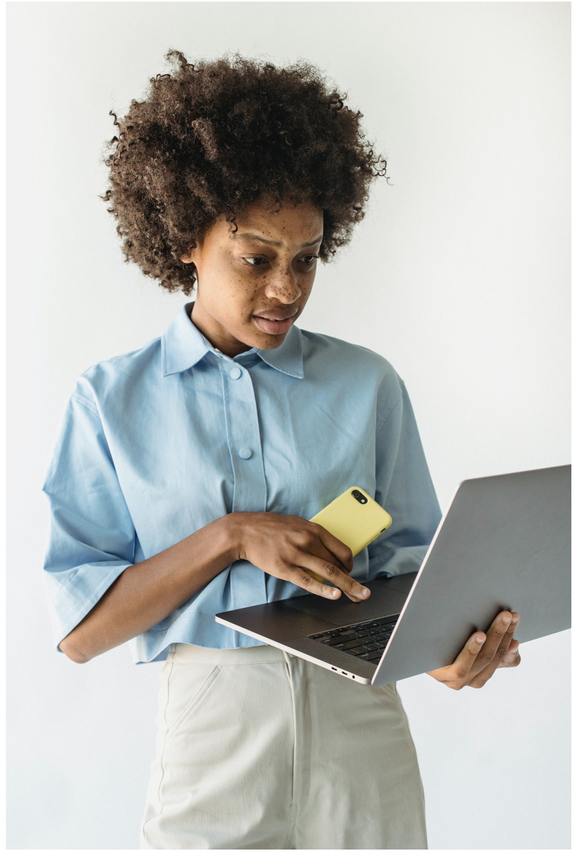
- Performance tuning process
- Most common bottlenecks

SQL Server Query Optimizer

- T-SQL statement evaluation order

Capture query performance metrics

Is 30 seconds a lot or not?



Buying tickets for your favorite music band

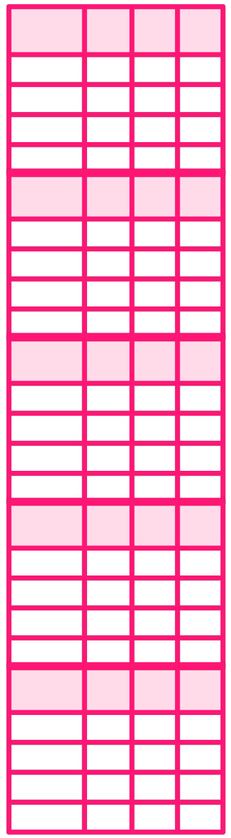
Waiting for the bus

Is 10 seconds a lot or not?

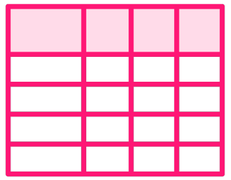
Waiting for breaking news to render

Getting the coffee in Starbucks

Setting the Right Expectations



T-SQL Query



Previous 10 years

Last week

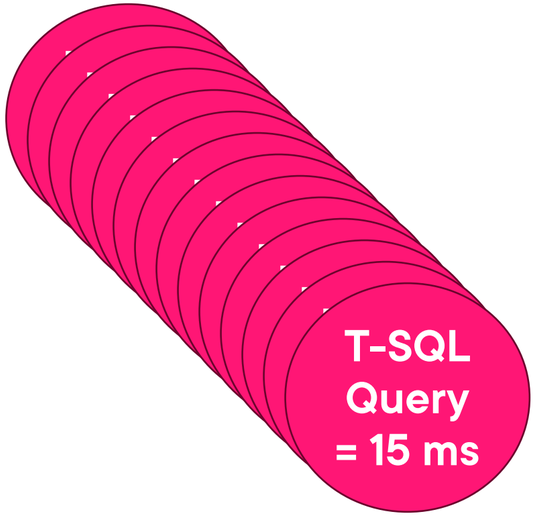
1 billion records

1 thousand records

When should you start tuning queries?

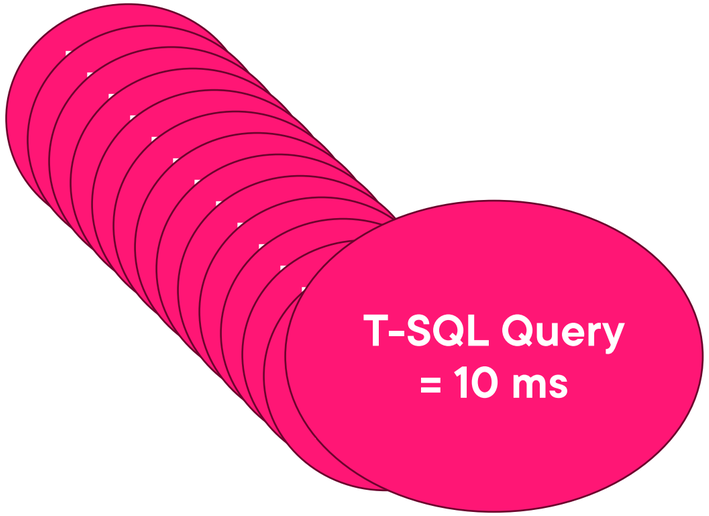
When they are not performing GOOD ENOUGH...

What Is “Good Enough”?



1000x/minute

What Is “Good Enough”?



1000x/minute

T-SQL Query
= 50 ms

5x/hour

Is it worth reducing to 40ms?!

Validation Points in Query Tuning Process

1. Performance of the app or reports that rely on database queries
2. Checking the query plan cache
3. Running queries on non-production system

Validation Points in Query Tuning Process

1. Performance of the app or reports that rely on database queries
2. Checking the query plan cache
3. Running queries on non-production system

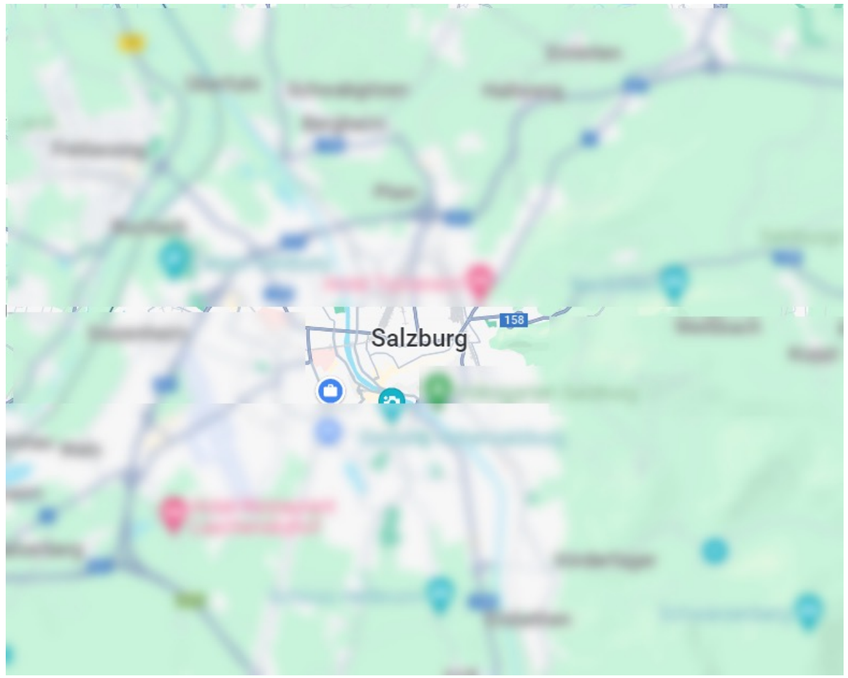
- Time
- CPU
- Disk I/O

The Most Common Performance Bottlenecks

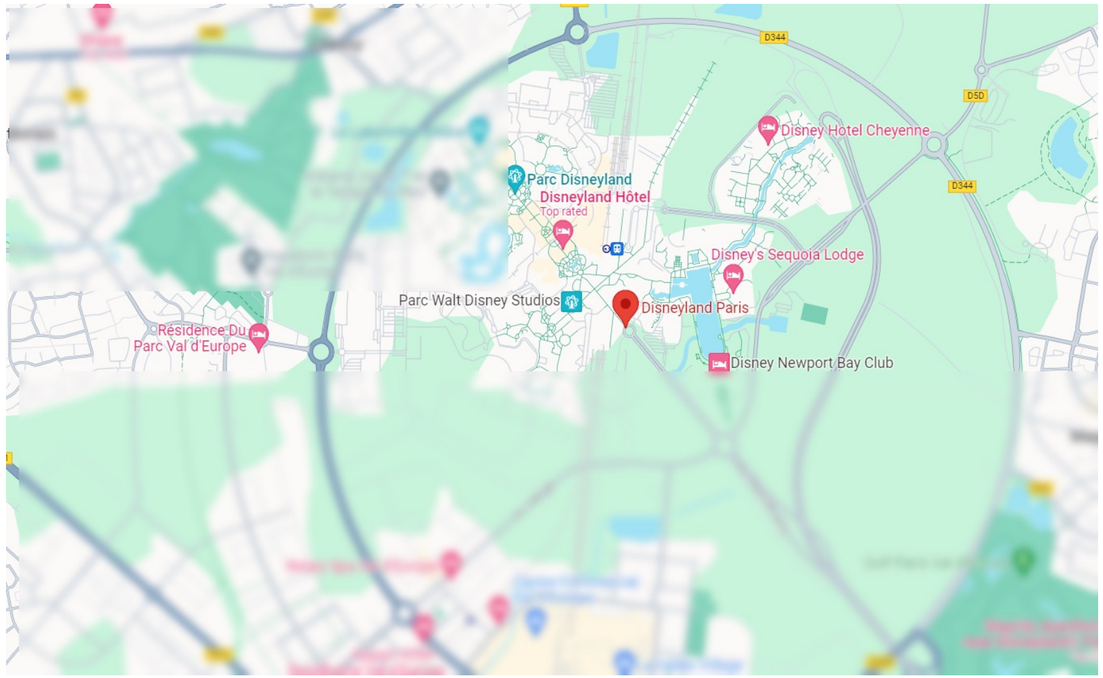
- Inadequate indexing strategy
- Missing/Inaccurate Statistics
- Sub-optimal T-SQL code
- Sub-optimal execution plans
- Sub-optimal execution plan reuse
- Frequent query recompilation

Introducing SQL Server Query Optimizer

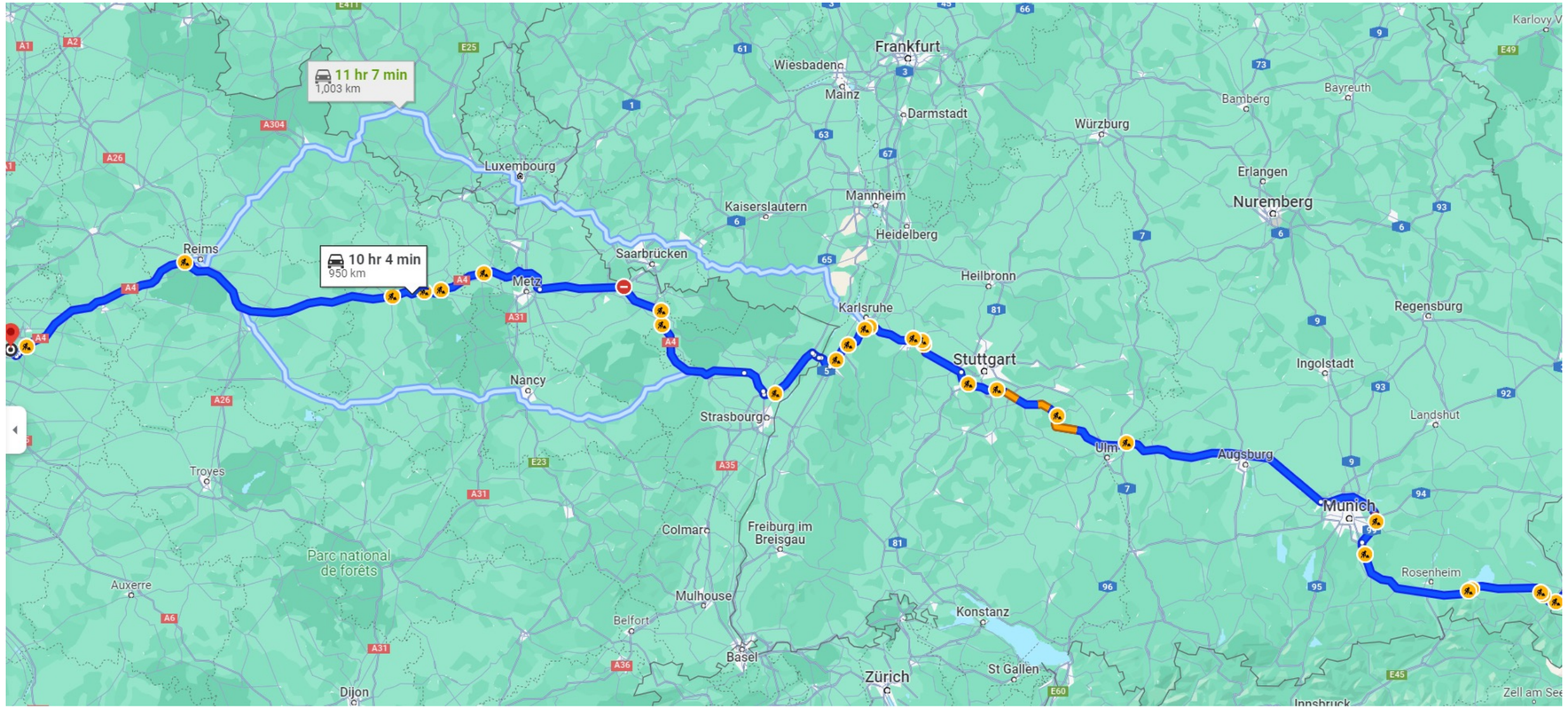
Brain behind SQL Server query performance



Salzburg, Austria



Disneyland, Paris



- ✓ Distance
- ✓ Traffic jams
- ✓ Possible accidents
- ✓ Road works
- ✓ Road type

- ✓ Evaluate criteria
- ✓ Estimate the cost

The “optimal” query plan is not always the most optimal!

T-SQL Statement Evaluation Order

```
SELECT column1
       , column2
       , SUM(column3) AS sumCol
FROM table1 AS t1
INNER JOIN table2 AS t2 ON t1.id = t2.id
WHERE column1 = ‘ABC’
GROUP BY column1
         , column2
HAVING SUM(column3) >= 100
ORDER BY column2 DESC
```

T-SQL Statement Evaluation Order

1 FROM

```
SELECT column1
       , column2
       , SUM(column3) AS sumCol
FROM table1 AS t1
INNER JOIN table2 AS t2 ON t1.id = t2.id
WHERE column1 = ‘ABC’
GROUP BY column1
         , column2
HAVING SUM(column3) >= 100
ORDER BY column2 DESC
```

T-SQL Statement Evaluation Order

```
SELECT column1
       , column2
       , SUM(column3) AS sumCol
FROM table1 AS t1
INNER JOIN table2 AS t2 ON t1.id = t2.id
WHERE column1 = ‘ABC’
GROUP BY column1
         , column2
HAVING SUM(column3) >= 100
ORDER BY column2 DESC
```

1. FROM
2. WHERE

T-SQL Statement Evaluation Order

```
SELECT column1
       , column2
       , SUM(column3) AS sumCol
FROM table1 AS t1
INNER JOIN table2 AS t2 ON t1.id = t2.id
WHERE column1 = 'ABC'
GROUP BY column1
         , column2
HAVING SUM(column3) >= 100
ORDER BY column2 DESC
```

1. FROM
2. WHERE
3. GROUP BY

T-SQL Statement Evaluation Order

```
SELECT column1
       , column2
       , SUM(column3) AS sumCol
FROM table1 AS t1
INNER JOIN table2 AS t2 ON t1.id = t2.id
WHERE column1 = 'ABC'
GROUP BY column1
         , column2
HAVING SUM(column3) >= 100
ORDER BY column2 DESC
```

1. FROM
2. WHERE
3. GROUP BY
4. HAVING

T-SQL Statement Evaluation Order

```
SELECT column1
         , column2
         , SUM(column3) AS sumCol
FROM table1 AS t1
INNER JOIN table2 AS t2 ON t1.id = t2.id
WHERE column1 = ‘ABC’
GROUP BY column1
         , column2
HAVING SUM(column3) >= 100
ORDER BY column2 DESC
```

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT

T-SQL Statement Evaluation Order

```
SELECT column1
       , column2
       , SUM(column3) AS sumCol
FROM table1 AS t1
INNER JOIN table2 AS t2 ON t1.id = t2.id
WHERE column1 = 'ABC'
GROUP BY column1
         , column2
HAVING SUM(column3) >= 100
ORDER BY column2 DESC
```

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

Understanding the Stages of the Query Optimization Process

We only specify which data we want, not the STEPS to get this data

Query Processor Engine in a Nutshell

- ✓ Accepts the query
- ✓ Generates the optimal query plan execution
- ✓ Executes the query plan

Query Optimizer

Execution Engine

Query Processing Stages

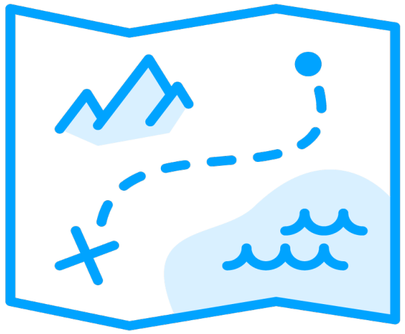
1 Parsing

2 Binding

- ✓ Objects necessary for the query (tables, columns, indexes...)
- ✓ Identifying data types
- ✓ Processing aggregation types for GROUP BY

3 Optimization

Optimization Stage Key Steps



Generating possible execution plans

Number of candidate plans as a series of physical operations

Cost evaluation for each plan

Plan with the lowest cost will be passed to the Execution Engine

Query optimization
=
mapping logical query operations to physical operations

Demo

Capture query performance metrics

- Statistics
- QueryTimeStats
- Querying DMVs
- Query Store

Summary

Is 10ms fast or not?

- Setting the right expectations

The most common query bottlenecks

Similarity between Google Maps and SQL Server query plans

- Get results using multiple different plans

SQL Server Query Optimizer = cost based engine

- Picks the optimal plan among many candidates

Up Next:

Understanding Query Plans