Architecture & Performance Tuning

Surendra Panpaliya

Day 1

SQL Server Internals & Query Execution (Foundation)

Module 1: SQL Server Architecture & Execution Lifecycle

Module 2: Execution Plans & Query Tuning Deep Dive

Module 2: Execution Plans & Query Tuning Deep Dive

Big Picture

Think of SQL Server like **Google Maps** 🗺️

- Your SQL query = destination
- Execution plan = route chosen
- Different routes = different performance
- Wrong route = slow query

Query tuning = helping SQL Server choose a better route

1. What is an Execution Plan?

An **execution plan** is:

- SQL Server’s **step-by-step instruction list** for running your query.

It answers:

- Which table to read first?
- Which index to use?
- How to join tables?
- How much memory to use?

Real-Life Example 🍳

Cooking Maggi noodles:

- Boil water
- Add noodles
- Add masala
- Serve

Different order → bad result

Same in SQL Server.

Lab 1: See an Execution Plan

```
SELECT *
FROM Sales.SalesOrderDetail
WHERE ProductID = 870;
```

In SSMS:

- Click **Include Actual Execution Plan**
- Run the query
- You will see **icons connected with arrows** → this is the execution plan.

2. Logical vs Physical Execution Plans

2.1 Logical Plan

2.2 Physical Plan

2.1 Logical Plan (WHAT to do)

Logical plan describes:

- Join tables
- Filter rows
- Aggregate data

It does **NOT** say *how* exactly.

- **Example**
- “Get orders → join customers → filter by date”

2.2 Physical Plan (HOW to do)

Physical plan decides:

- Index Seek or Scan?
- Hash Join or Nested Loop?
- Parallel or Serial?

2.2 Physical Plan

Real Life Example

Plan Type	Example
Logical	“Deliver package”
Physical	“Bike / Car / Truck”

Simple Lab 2: Spot Physical Decisions

```
SELECT *
FROM Sales.SalesOrderDetail
WHERE ProductID > 800;
```

Look at operators:

- **Index Scan**
- **Index Seek**

These are **physical decisions**.

3. Operators, Cost & Cardinality Estimation

3.1 What is an Operator?

- An **operator** is one step in the plan.

Examples:

- Index Seek
- Index Scan
- Hash Join
- Sort

3.2 What is Operator Cost?

Simple Meaning

- Cost = SQL Server’s **estimated effort**, not time.
- CPU cost
- I/O cost
- Relative number

⚠️ Cost is **not seconds**

3.3 Cardinality Estimation

Meaning

- Cardinality = **how many rows SQL Server thinks will come**
- Estimated rows
- Actual rows

3.3 Cardinality Estimation

Real-Life Example 🎟️

You book chairs for a meeting:

- Expected people = 10
- Actual people = 100

Chaos 😄

Same happens in SQL Server

Lab 3: Bad Estimates

```
SELECT *
FROM Sales.SalesOrderDetail
WHERE OrderQty = 1;
```

In execution plan:

- Hover on operator
- Compare:
    - **Estimated Rows**
    - **Actual Rows**

If difference is huge → **bad estimate**

4. Join Types & Join Order Problems

4.1 Join Types

4.2 Join Order

4.1 Join Types

Join	When SQL Server Uses It	Real-Life Example
Nested Loop	Small + Indexed	Checking roll numbers
Hash Join	Large data	Bucket matching
Merge Join	Sorted data	Two sorted lists

4.2 Join Order

SQL Server decides:

- Which table first
- Which table next

Wrong order = heavy cost

Lab 4: Join Change

```
SELECT *
FROM Sales.SalesOrderHeader h
JOIN Sales.SalesOrderDetail d
ON h.SalesOrderID = d.SalesOrderID;
```

- Observe join type.
- Then add filter on one table and rerun.

Learning:

- **Join type changes based on data size**

5. Parameter Sniffing

What is Parameter Sniffing?

When a query runs **first time**, SQL Server:

- “Sniffs” parameter value
- Creates plan based on it
- Reuses plan later

5. Parameter Sniffing

Real-Life Example 🍕

- First customer orders **1 pizza** → kitchen prepares small setup
- Next customer orders **100 pizzas** → same setup → disaster

Lab 5: Parameter Sniffing Demo

```
CREATE PROCEDURE GetOrders
@OrderQty INT
AS
SELECT *
FROM Sales.SalesOrderDetail
WHERE OrderQty = @OrderQty;
```

Lab 5: Parameter Sniffing Demo

Run:

```
EXEC GetOrders 1;
EXEC GetOrders 100;
```

- Same plan reused → may be bad.

Simple Fix Options

- Recompile
- Rewrite query
- Use Query Store

6. Bad Plans Due to Stale Statistics

What are Statistics?

- Statistics tell SQL Server:
- Data distribution
- How many rows exist

- If stats are old → wrong estimates.

6. Bad Plans Due to Stale Statistics

Real-Life Example 📊

Old population data → wrong city planning.

Lab 6: Stats Impact

```
SELECT *
FROM Sales.SalesOrderDetail
WHERE ProductID = 870;
```

Update stats and compare plan:

```
UPDATE STATISTICS Sales.SalesOrderDetail;
```

7. Query Rewriting Techniques

Why Rewrite Queries?

- Sometimes SQL Server **cannot guess your intention**.
- Rewriting helps optimizer.

7. Query Rewriting Techniques

Example 1: Avoid SELECT *

```
SELECT SalesOrderID
FROM Sales.SalesOrderDetail;
```

Less data → faster.

7. Query Rewriting Techniques

Example 2: Move filters early

Bad:

```
SELECT *
FROM A JOIN B
ON A.id = B.id
WHERE A.status = 'Active';
```

7. Query Rewriting Techniques

Example 2: Move filters early

Better:

```
SELECT *
FROM (SELECT * FROM A WHERE status='Active') A
JOIN B ON A.id = B.id;
```

8. Hints vs Plan Forcing

8.1 Hints

Hints **force SQL Server** to behave.

Examples:

- INDEX hint
- JOIN hint

Problem

- Hints **freeze optimizer intelligence**.

8.2 Plan Forcing

- Query Store allows:
- Capture good plan
- Force it safely
- Revert easily

8.2 Plan Forcing

Real-Life Example 🧠

- Instead of forcing driver,
- use **GPS history** to choose better route.

Lab 7: Query Store Plan Forcing

- Enable Query Store
- Identify good plan
- Force it
- Observe improvement

9. Tools Used

Execution Plans

Visual map of query execution.

Query Store

History of:

- Queries
- Plans
- Performance

9. Tools Used

sys.dm_exec_query_stats

```
SELECT *
FROM sys.dm_exec_query_stats;
```

Shows:

- CPU used
- Execution count
- Total time

FINAL MEMORY CHEAT SHEET 🧠

Concept	Simple Meaning
Execution Plan	SQL Server’s recipe
Logical Plan	What to do
Physical Plan	How to do
Operator	One step
Cardinality	Row count guess
Join Type	How tables connect

FINAL MEMORY CHEAT SHEET 🧠

Concept	Simple Meaning
Parameter Sniffing	First value bias
Statistics	Data knowledge
Query Rewrite	Help optimizer
Hints	Hard force (danger)
Query Store	Safe plan control

LAB 1 – Understanding Execution Plans

Step 1: Run a Simple Query

```
SELECT *
FROM Sales.SalesOrderDetail
WHERE ProductID = 870;
```

✓ Enable **Include Actual Execution Plan**

LAB 1 – Understanding Execution Plans

What to Observe

- Operators (icons)
- Arrows (data flow)
- Estimated vs Actual rows

LAB 1 – Understanding Execution Plans

Real-Life Analogy 🗺️

- Execution plan = Google Maps route
- Operators = turns on the road
- Arrows = traffic volume

LAB 2 – Logical vs Physical Plans

Goal

Understand **WHAT vs HOW** difference.

Query

```
SELECT *
FROM Sales.SalesOrderDetail
WHERE OrderQty > 5;
```

LAB 2 – Logical vs Physical Plans

What to Learn

- Logical intent: “Filter rows”
- Physical choice: “Index Scan” or “Seek”

Key Learning

- Logical plan stays same
- Physical plan changes based on data

LAB 3 – Cardinality Estimation

Goal

Understand **row estimation mistakes**.

Query

```
SELECT *
FROM Sales.SalesOrderDetail
WHERE OrderQty = 1;
```

LAB 3 – Cardinality Estimation

Inspect Execution Plan

Hover on the operator and check:
- Estimated Rows
- Actual Rows

Real-Life Analogy 🎟️

Booking seats for 10 people, 100 arrive.

LAB 4 – Table Scan vs Index Seek

Goal

- Understand **why scans happen.**

Query 1 (Seek)

```
SELECT *
FROM Sales.SalesOrderDetail
WHERE SalesOrderDetailID = 1;
```

LAB 4 – Table Scan vs Index Seek

Query 2 (Scan)

```
SELECT *
FROM Sales.SalesOrderDetail
WHERE OrderQty > 1;
```

LAB 4 – Table Scan vs Index Seek

Learning

- Seek = targeted lookup
- Scan = reading many rows

Rule to Remember

- Scan is not always bad
- Scan on large table frequently = problem

LAB 5 – Join Types & Join Order

Goal

Understand **why join types change**.

Query

```
SELECT *
FROM Sales.SalesOrderHeader h
JOIN Sales.SalesOrderDetail d
ON h.SalesOrderID = d.SalesOrderID;
```

LAB 5 – Join Types & Join Order

Observe

- Join operator type
- Estimated rows on each side

Real-Life Analogy 🧩

- Nested Loop → one-by-one matching
- Hash Join → bucket matching
- Merge Join → sorted list matching

LAB 6 – Parameter Sniffing

Goal

Understand **why same query behaves differently.**

Create Procedure

```
CREATE OR ALTER PROCEDURE GetOrders
   @Qty INT
AS
SELECT *
FROM Sales.SalesOrderDetail
WHERE OrderQty = @Qty;
```

LAB 6 – Parameter Sniffing

Execute

```
EXEC GetOrders 1;
EXEC GetOrders 100;
```

Observation

- Same execution plan reused
- Performance may differ

LAB 6 – Parameter Sniffing

Real-Life Analogy 🍕

Kitchen prepared for 1 order, suddenly 100 arrive.

Learning

First execution influences future executions

LAB 7 – Stale Statistics & Bad Plans

Goal

Understand **why statistics matter**.

Query

```
SELECT *
FROM Sales.SalesOrderDetail
WHERE ProductID = 870;
```

LAB 7 – Stale Statistics & Bad Plans

Update Statistics

```
UPDATE STATISTICS Sales.SalesOrderDetail;
```

Rerun Query & Compare Plan

Learning

- Statistics guide optimizer
- Old stats = wrong decisions

LAB 8 – Query Rewriting

Goal

Learn **how small changes help SQL Server**.

Bad Practice

```
SELECT *
FROM Sales.SalesOrderDetail;
```

LAB 8 – Query Rewriting

Better Practice

```
SELECT SalesOrderID, ProductID
FROM Sales.SalesOrderDetail;
```

Learning

- Less data = less work
- Clear intent = better plan

LAB 9 – Hints vs Query Store

Safe vs Dangerous

Goal

Understand **why hints are risky.**

LAB 9 – Hints vs Query Store

Hint Example (DO NOT USE IN REAL LIFE)

```
SELECT *
FROM Sales.SalesOrderDetail
WITH (INDEX(IX_ProductID));
```

Query Store (Recommended)

- Capture multiple plans
- Identify good plan
- Force safely
- Revert anytime

LAB 9 – Hints vs Query Store

Real-Life Analogy 🚦

- Hints = locking traffic signal forever
- Query Store = learning from past traffic

MODULE 2 TAKEAWAY

Concept	One-Line Meaning
Execution Plan	SQL Server’s recipe
Cardinality	Row count guess
Join Type	How tables connect
Scan vs Seek	How data is read
Parameter Sniffing	First execution bias

MODULE 2 TAKEAWAY

Concept	One-Line Meaning
Statistics	Optimizer’s knowledge
Query Rewrite	Helping SQL Server
Hints	Dangerous force
Query Store	Safe tuning

Surendra Panpaliya

Founder and CEO

GKTCS Innovations

https://www.gktcs.com