Architecture & Performance Tuning

Surendra Panpaliya

Prerequisites

Technical Skills

Strong SQL querying (SELECT, JOIN, GROUP BY)

Basic indexing knowledge

Familiarity with SQL Server Management Studio (SSMS)

Basic understanding of OLTP workloads

Surendra Panpaliya, DTM

Founder & CEO, GKTCS Innovations – building future-ready enterprises.

Empowered 35,000+ IT professionals through training, mentoring, and consulting.

Partnered with 300+ multinational corporations to accelerate business growth.

Day 1

SQL Server Internals & Query Execution (Foundation)

Module 1: SQL Server Architecture & Execution Lifecycle

Module 2: Execution Plans & Query Tuning Deep Dive

Day 2

Indexing, Statistics & Concurrency

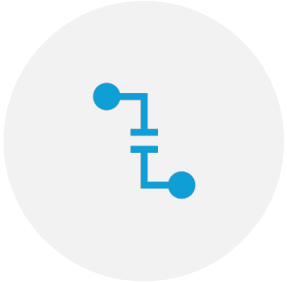
Module 3: Index & Statistics Tuning (4 Hours)

Module 4: Wait Stats, Blocking & Concurrency (4 Hours)

DAY 3

Memory, TempDB, CPU, I/O & Cloud Performance

Module 5: TempDB & Memory Tuning



Module 6: Parallelism, CPU & I/O Tuning

Module 7: Query Store, HA & Azure SQL Performance

What is SQL Server?

A **database software**

made by Microsoft.

What is SQL Server?

Store data safely

Allow you to search data

Update data

Protect data

Handle many users at the same time

Real-life example

Think of SQL Server like a **bank locker system**:

Data = your valuables

Tables = lockers

SQL Server = bank system managing lockers

SQL = language to talk to the bank system

SQL Server 2025 – What's New

Faster performance

Better memory handling

Smarter Query Store

Improved cloud & hybrid support

More automation for DBAs

Module 1: SQL Server Architecture & Execution Lifecycle

What Happens When You Run a SQL Query?

```
SELECT *  FROM orders WHERE order_id = 10;
```

What Happens When You Run a SQL Query?

- SQL Server does **NOT** immediately read the data.
- Instead, it goes through **multiple internal steps**, like a human thinking before acting.

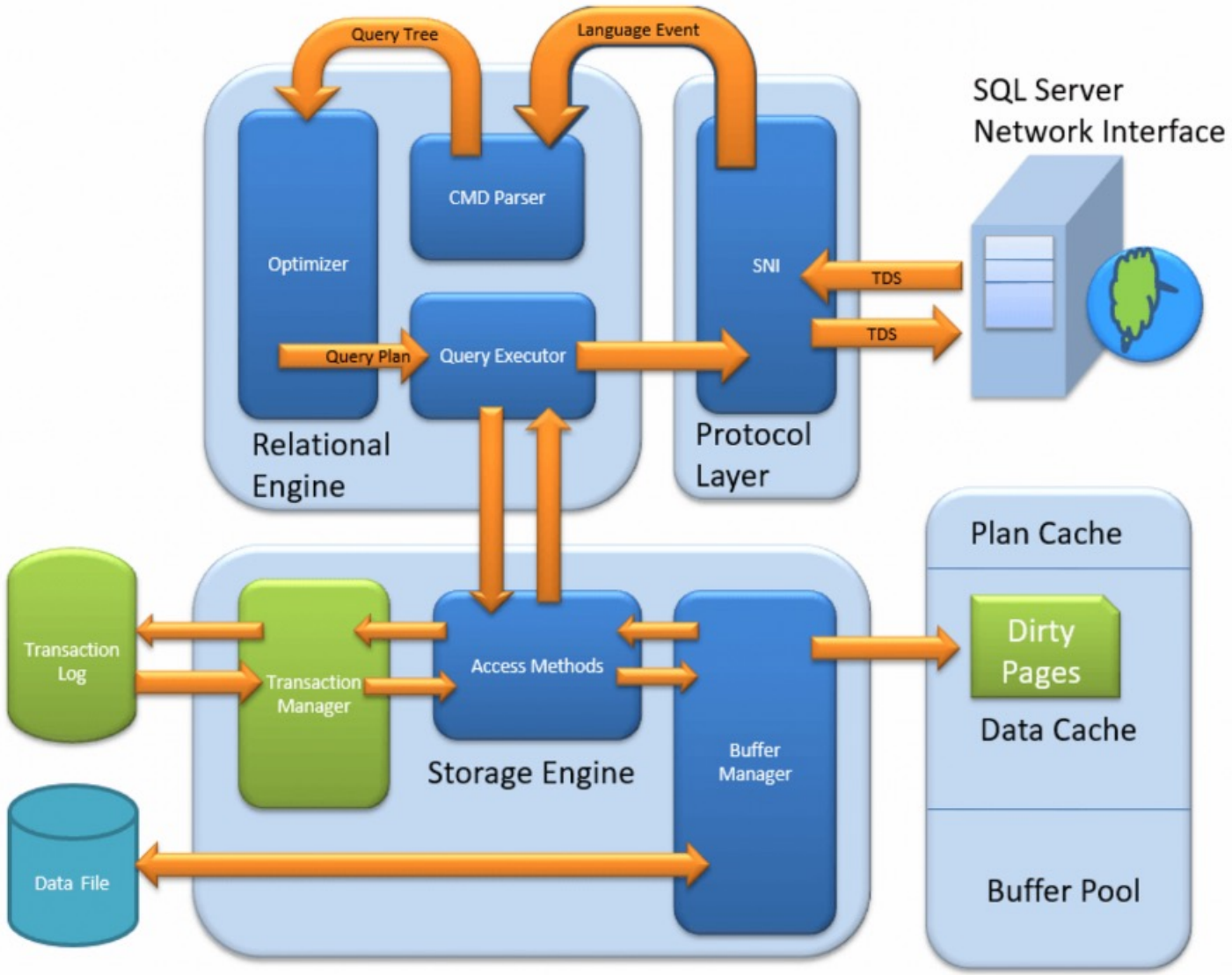
To understand these steps, let's understand **SQL Server Architecture**.

SQL Server Architecture

- Think of SQL Server like
- a **smart factory**
  - You give an instruction (SQL query)
  - One part **thinks**
  - Another part **does the physical work**

That’s why SQL Server has **two main engines**:

SQL SERVER ARCHITECTURE



2.1 Relational Engine (The Brain 🧠)

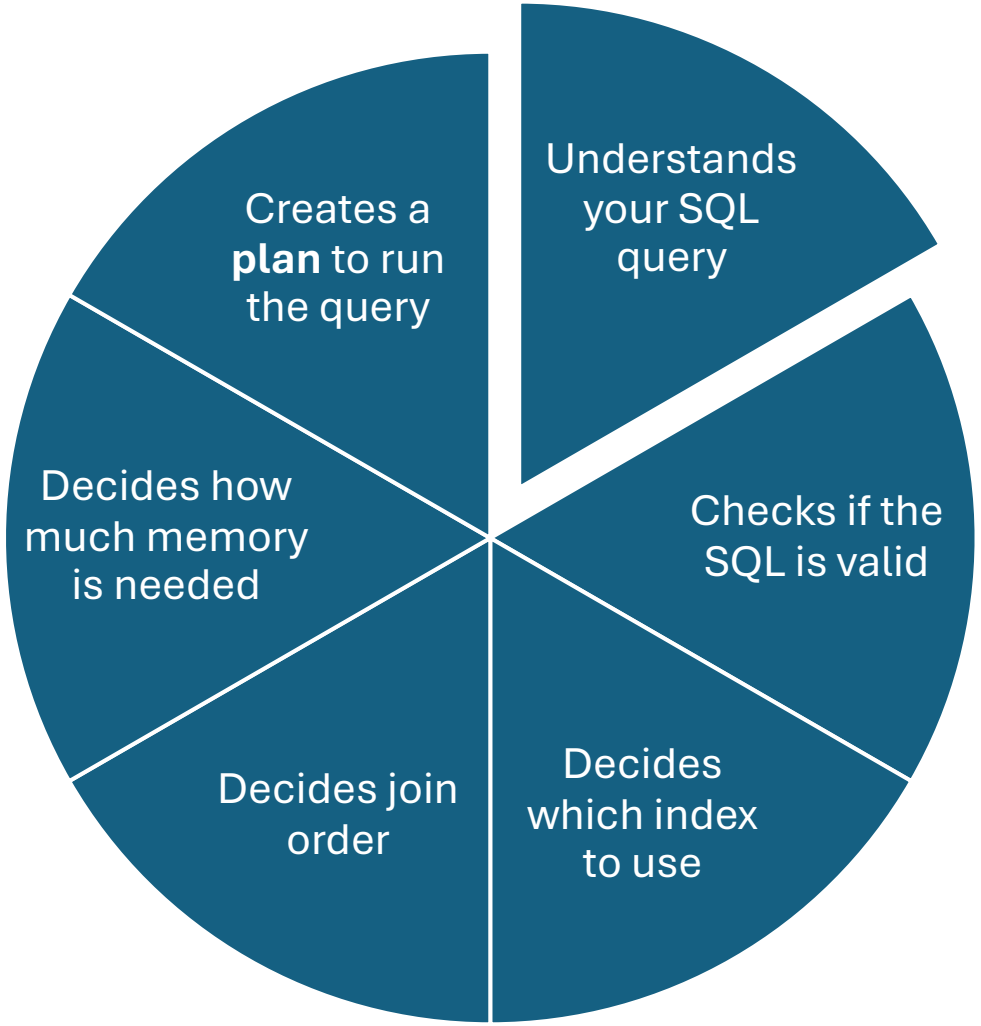
What is the Relational Engine?

The **Relational Engine** is the part of SQL Server that **THINKS and DECIDES**.

It does NOT touch the disk directly.

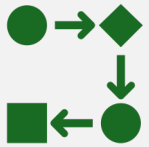
2.1 Relational Engine (The Brain 🧠)

What does it do?



2.1 Relational Engine (The Brain 🧠) Simple analogy

Relational Engine = Planner / Manager



“How should this work be done efficiently?”

2.2 Storage Engine (The Worker)

What is the Storage Engine?

The **Storage Engine** is the part that **DOES THE ACTUAL WORK**

2.2 Storage Engine (The Worker)

What does it do?

Reads data from disk

Loads data into memory (RAM)

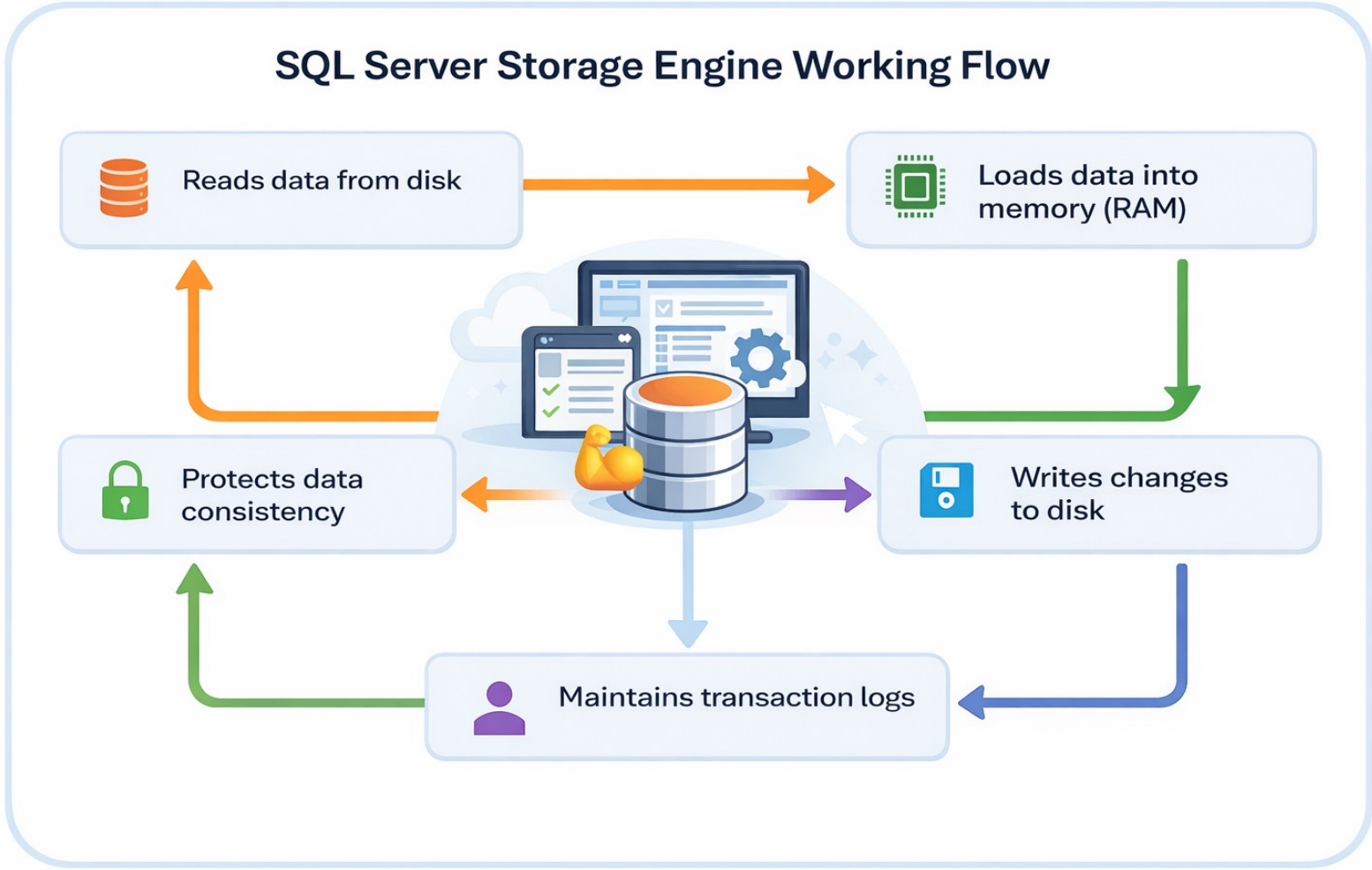
Writes changes to disk

Maintains transaction logs

Protects data consistency

2.2 Storage Engine (The Worker)

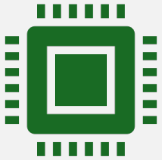
What does it do?



2.2 Storage Engine (The Worker)

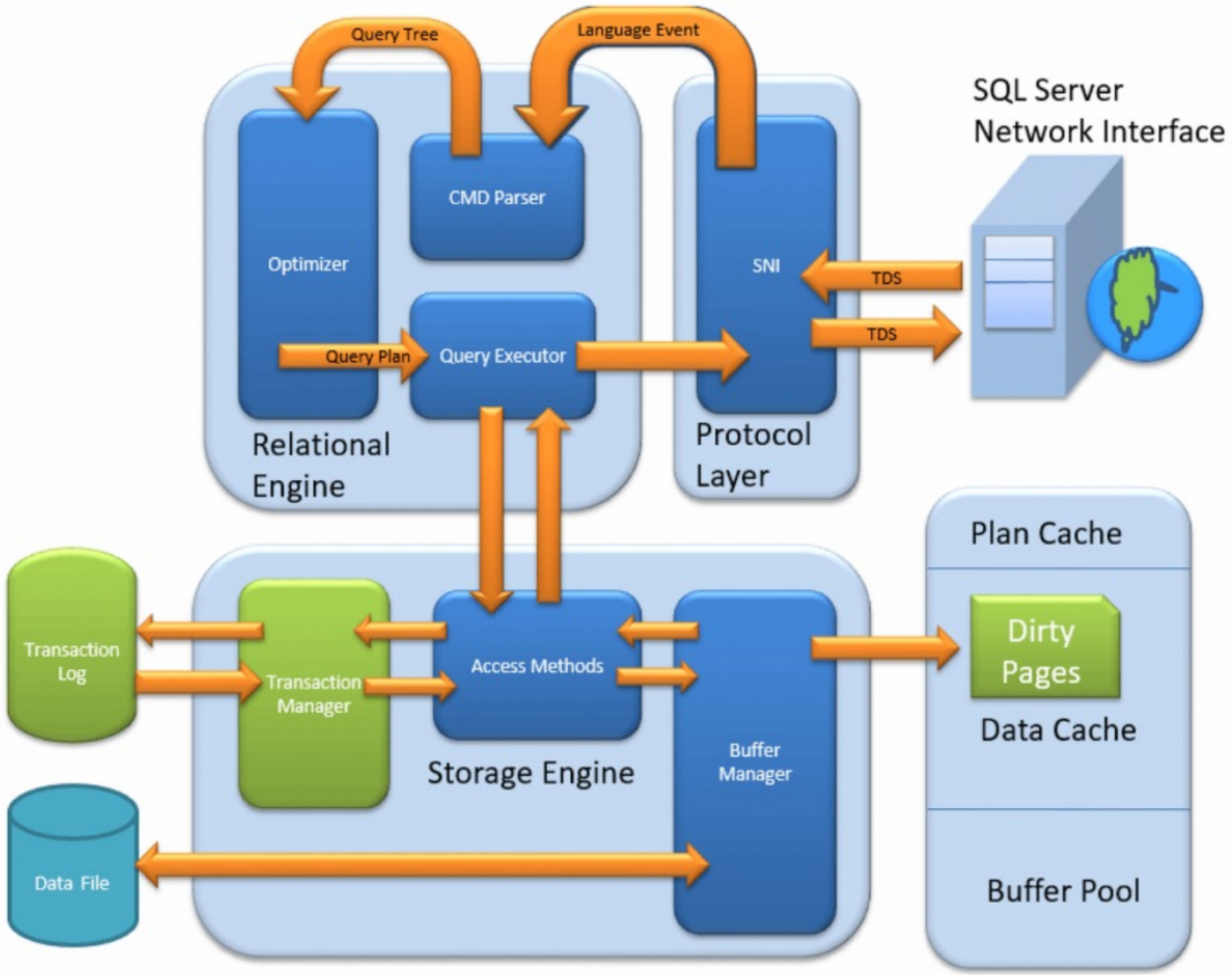
Simple analogy

Storage Engine = Worker



“Fetch this data, store this data, save it safely.”

SQL SERVER ARCHITECTURE



Key Difference (Very Important)

Engine	Role
Relational Engine	Thinks & decides
Storage Engine	Reads & writes data

3. Query Execution Lifecycle (Step-by-Step)

This explains **what happens internally** from the moment you press **Enter**.

3.1 Parsing

What is Parsing?

Parsing means:

SQL Server checks **syntax**

SQL Server checks spelling of SQL keywords

SQL Server confirms query is written correctly

3.1 Parsing

Example

SELEC * FROM orders;

❌ This fails during **parsing** because SELEC is wrong.

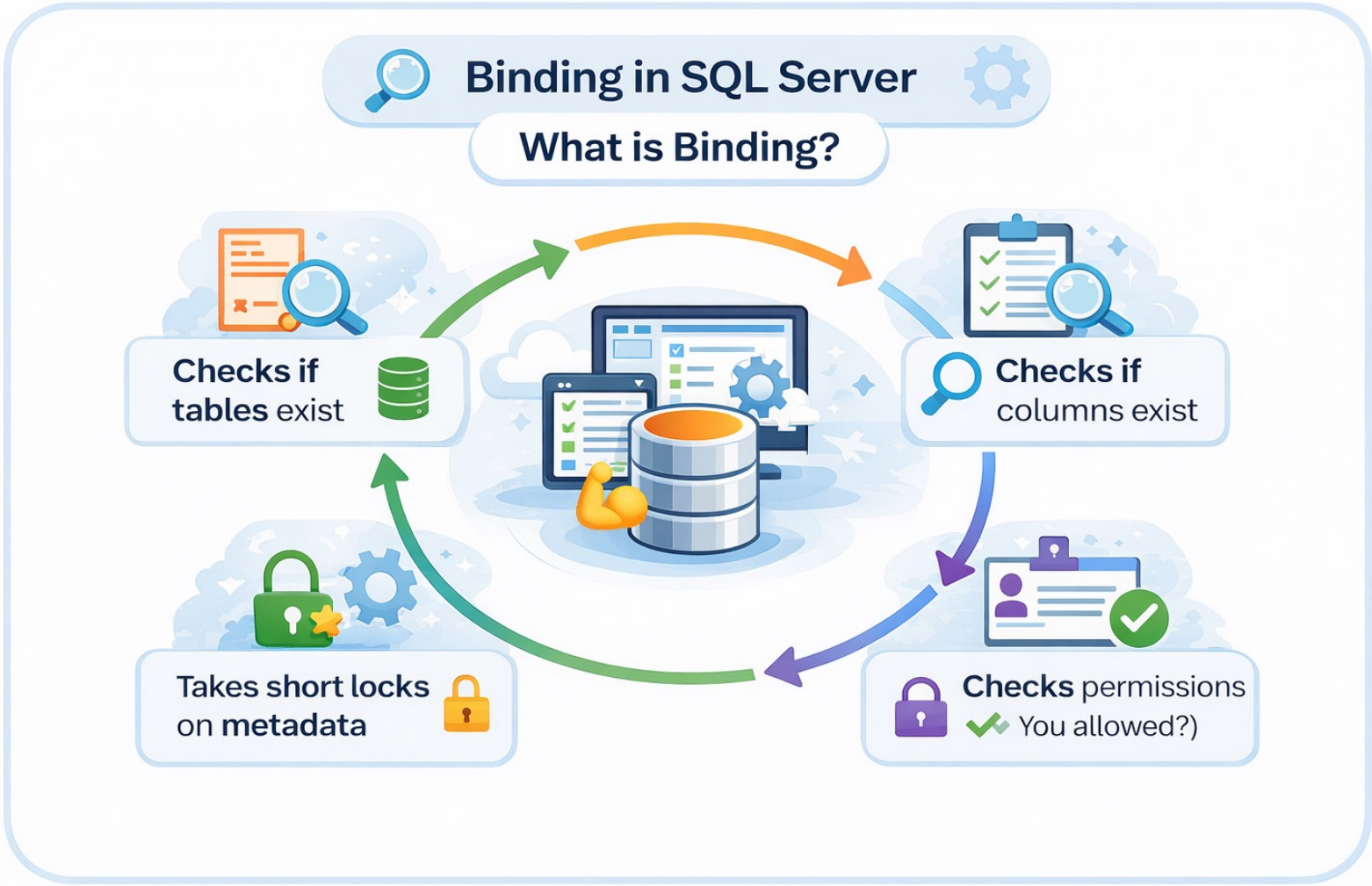
Important point

Parsing does NOT check data

Parsing is only about **language correctness**

3.2 Binding

What is Binding?



3.2 Binding

What is Binding?

Binding means:

SQL Server checks if tables exist

Checks if columns exist

Checks permissions (are you allowed?)

Takes short locks on metadata

3.2 Binding

Example

SELECT price FROM orders;

- If price column does not exist → error during **binding**

3.2 Binding

Simple analogy

- 📋 Binding = “Do all required objects exist and are you allowed to use them?”

3.3 Optimization

What is Optimization?

This is the **most important step**.

- SQL Server decides:
- Which index to use
- Scan or seek?
- Join type (Nested Loop, Hash, Merge)
- How much memory is required
- Parallel or single-threaded execution

3.3 Optimization

Important fact

SQL Server may create **multiple possible plans** and choose the cheapest one.

Why this matters

A query can be fast today and slow tomorrow because **a different plan was chosen**.

3.4 Execution

What is Execution?

Execution is when SQL Server:

- Assigns a worker thread
- Uses CPU
- Uses memory
- Reads rows
- Returns results

3.4 Execution

This is where:

- CPU usage increases
- Memory is consumed
- Disk reads may happen

4. Schedulers, Workers & Threads (CPU Basics)

4.1 What is a Scheduler?

A **Scheduler** is SQL Server's way of managing CPU.

- One scheduler ≈ one CPU core
- Scheduler decides which task runs next

Analogy

 Scheduler = Traffic police for CPU

4.2 What is a Worker?

A **Worker** is a thread that executes a query.

- Each running query needs at least one worker
- Parallel queries need multiple workers

Analogy

👷 Worker = Person doing the work

4.3 What is a Thread?

- A **Thread** is the actual execution unit used by Windows/Linux.
- SQL Server manages threads internally
- Too many threads = waiting

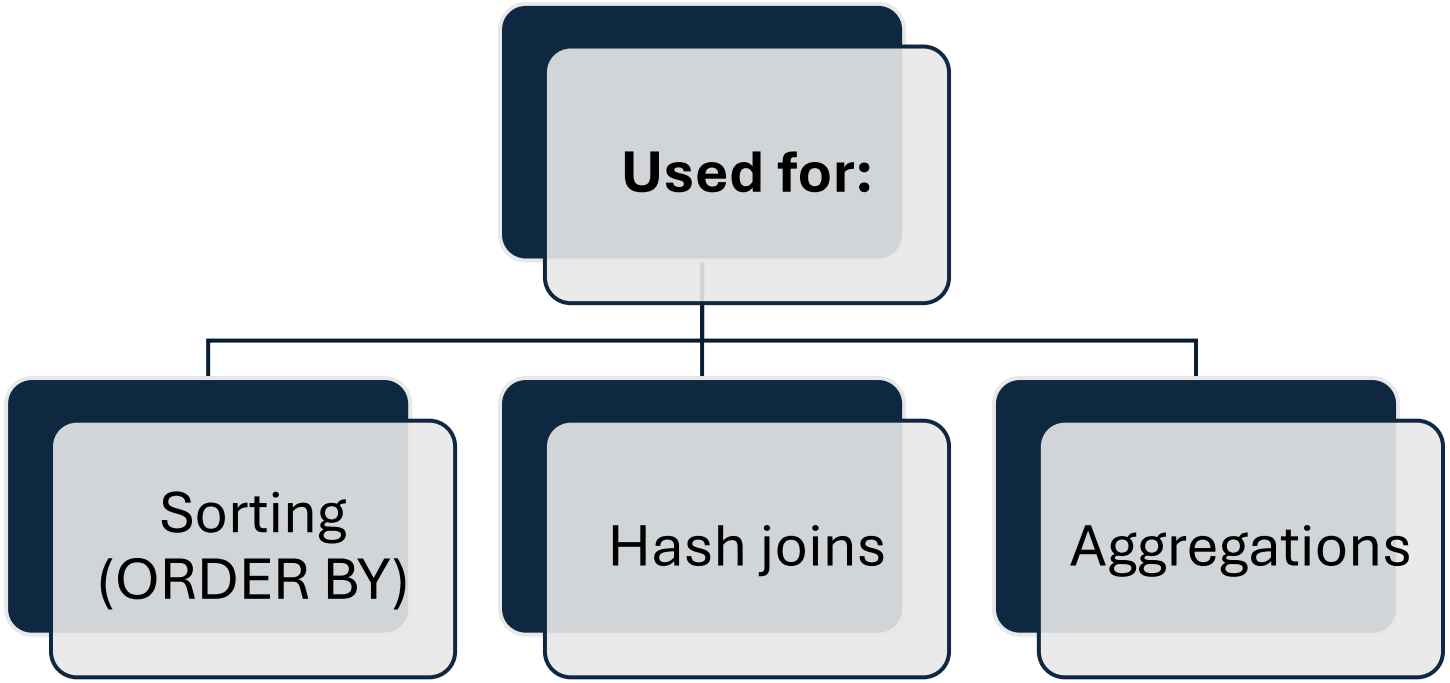
5. Memory Grants

5. Memory Grants

What is a Memory Grant?

- Before execution, SQL Server asks:
- “How much memory does this query need?”
- This reserved memory is called a **memory grant**.

5. Memory Grants



What Can Go Wrong?

Situation	Result
Too little memory	Data spills to TempDB (slow)
Too much memory	Other queries wait
Many big grants	System slowdown

6. Latches vs Locks vs Waits

6.1 Locks

What are Locks?

- Locks protect **data correctness**.

Example:

- One user updates a row
- Another user must wait

📌 Locks are **logical protection**

6.2 Latches

What are Latches?

- Latches protect **memory structures** inside SQL Server.

Example:

- Multiple threads trying to modify same memory page

📌 Latches are **internal and short-lived**

6.3 Waits
What are Waits?

WAITS TELL YOU:

“WHAT IS SQL SERVER
WAITING FOR RIGHT NOW?”

6.3 Waits

Examples:

- Waiting for CPU
- Waiting for memory
- Waiting for disk
- Waiting for a lock

📌 Waits are **symptoms**, not problems themselves

7. Hands-on Labs

7.1 Trace Query Lifecycle

Shows:

Running queries

Their CPU usage

What they are waiting for

7.1 Trace Query Lifecycle

Tool:

sys.dm_exec_requests

```
SELECT
   session_id,
   status,
   command,
   cpu_time,
   wait_type
FROM sys.dm_exec_requests;
```

7.2 Identify Workers & Schedulers

Tool:

sys.dm_os_schedulers

```
SELECT
   scheduler_id,
   runnable_tasks_count,
   active_workers_count
FROM sys.dm_os_schedulers
WHERE status = 'VISIBLE ONLINE';
```

7.2 Identify Workers & Schedulers

Shows:

CPU pressure

Worker availability

7.3 Observe Memory Grants

Tool:

sys.dm_exec_query_memory_grants

```
SELECT
   session_id,
   requested_memory_kb,
   granted_memory_kb,
   used_memory_kb
FROM sys.dm_exec_query_memory_grants;
```

7.3 Observe Memory Grants

Shows:

How much memory a query requested

How much it actually got

8. Simple Mental Model

🧠 Relational Engine → Think

💪 Storage Engine → Work

Scheduler → CPU control

👷 Worker → Executes query

8. Simple Mental Model

 Memory Grant → Reserved RAM

 Locks → Data safety

 Latches → Memory safety

 Waits → What SQL Server is waiting for

PART 1: SQL Server Architecture

Concept 1: Relational Engine vs Storage Engine

Real-Life Example 🏫 (School Analogy)

- **Teacher** decides *what* to teach → Relational Engine
- **Students** write notes → Storage Engine
- Teacher never writes notes for students.
- Students don’t decide syllabus.

LAB 1.1 – Observe Query Thinking vs Working

```
SELECT *
FROM sys.dm_exec_requests
WHERE session_id > 50;
```

What to Observe

- cpu_time → thinking + execution
- reads → data fetching
- wait_type → delays

Beginner Insight

- High CPU but low reads = **thinking problem**
- High reads = **data fetching problem**

What to Check Next (Real Incident)

- CPU high? → Look at execution plans
- Reads high? → Look at indexes

PART 2: QUERY EXECUTION LIFECYCLE

Concept 2: Parsing

Real-Life Example ✍️

Before sending an email:

- Spell check
- Grammar check

SQL Server does the same with your SQL.

LAB 2.1 – Parsing Error Demo

```
SELEC * FROM sys.objects;
```

❌ Error occurs immediately → **Parsing phase**

Learning

- SQL Server stops early
- No CPU, no disk used

Concept 3: Binding

Real-Life Example 

- You enter a restaurant:
- Menu must exist
- Dish must exist
- You must be allowed to order

LAB 2.2 – Binding Error Demo

```
SELECT price FROM sys.objects;
```

❌ Column does not exist → **Binding failure**

Learning

- SQL Server checks tables, columns, permissions
- Still no data read

Concept 4: Optimization

Real-Life Example 🚗

Google Maps:

- Tries multiple routes
- Picks fastest route
- SQL Server does the same with query plans.

LAB 2.3 – Optimization in Action

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

Now run:

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

LAB 2.3 – Optimization in Action

What to Observe

- Same table
- Different plans (seek vs scan)

Learning

- SQL Server chooses **different strategies**
- Data distribution matters

Concept 5: Execution

Real-Life Example 🏃

Planning done → actual work starts.

LAB 2.4 – Execution Observation

Run this query and keep it running:

```
SELECT *
FROM Sales.SalesOrderDetail a
CROSS JOIN Sales.SalesOrderDetail b;
```

LAB 2.4 – Execution Observation

In another window:

```
SELECT
   session_id,
   status,
   cpu_time,
   wait_type
FROM sys.dm_exec_requests;
```

LAB 2.4 – Execution Observation

Learning

Execution consumes CPU

Long execution = resource usage

PART 3: SCHEDULERS, WORKERS & THREADS

Concept 6: Scheduler

Real-Life Example 🚦

Traffic signal:

Only one direction moves at a time

Scheduler controls CPU access.

LAB 3.1 – Observe CPU Pressure

```
SELECT
    scheduler_id,
    runnable_tasks_count,
    active_workers_count
FROM sys.dm_os_schedulers
WHERE status = 'VISIBLE ONLINE';
```

How to Read

runnable_tasks_count > 0 → CPU pressure

More runnable tasks = waiting for CPU

Concept 7: Worker Threads

Real-Life Example 👷

Workers in a factory:

- Limited workers
- Too many jobs → waiting

LAB 3.2 – Worker Consumption

Run multiple heavy queries simultaneously and observe:

```
SELECT
   scheduler_id,
   active_workers_count
FROM sys.dm_os_schedulers;
```

LAB 3.2 – Worker Consumption

Learning

- Too many queries = thread pressure
- Adding CPU does not always help

PART 4: MEMORY GRANTS

Concept 8: Memory Grant

Real-Life Example 🪑

Conference hall:

- Chairs reserved before meeting
- Too many chairs → others wait
- Too few chairs → people stand

LAB 4.1 – Memory Grant Observation

```
SELECT
   session_id,
   requested_memory_kb,
   granted_memory_kb,
   used_memory_kb,
   wait_time_ms
FROM sys.dm_exec_query_memory_grants;
```

How to Read

wait_time_ms > 0 → waiting for memory

granted >> used → wasted memory

PART 5: LATCHES vs LOCKS vs WAITS

Concept 9: Locks

Real-Life Example 🔐

ATM machine:

- One person at a time
- Locks protect **data correctness**.

Concept 10: Latches

Real-Life Example 🖇️

Two people trying to write on the same whiteboard

Latches protect **memory structures**.

Concept 11: Waits

Real-Life Example ⏳

Standing in line:

- Waiting for food
- Waiting for bus
- Waiting for ticket

Waits tell **WHY SQL Server is waiting**.

LAB 5.1 – Observe Waits

```
SELECT
    session_id,
    wait_type,
    wait_time,
    blocking_session_id
FROM sys.dm_exec_requests
WHERE wait_type IS NOT NULL;
```

Learning

Waits are **symptoms**

Root cause must be identified

SUMMARY

Concept	Simple Meaning
Relational Engine	Thinks
Storage Engine	Works
Parsing	SQL syntax check
Binding	Object & permission check
Optimization	Best plan selection
Execution	Actual work

SUMMARY

Concept	Simple Meaning
Scheduler	CPU manager
Worker	Executes query
Memory Grant	Reserved RAM
Locks	Data safety
Latches	Memory safety
Waits	Delay reason

What is a DMV in SQL Server?

Dynamic Management View

Like a live CCTV camera inside SQL Server

Shows what is happening RIGHT NOW

What is a DMV in SQL Server?

- When users say: “Report is slow”
- Dynamic Management View answer:
- Which query?
- Why slow?
- Waiting for what?
- CPU, memory, IO, or locks?

Key DMV Concepts

- **Dynamic**
- Data changes **every second**
- Shows **current / recent activity**
- Not stored permanently

Key DMV Concepts

- **“Management” mean**
- Helps in:
- Monitoring
- Troubleshooting
- Performance tuning
- Capacity planning

Key DMV Concepts

- **“View” mean**
- You query it like a table:
- SELECT * FROM sys.dm_exec_requests;
- But:
- You **cannot INSERT/UPDATE/DELETE**
- Read-only system views

Core DMV Categories

Category	Purpose
Execution	What queries are running
OS	CPU, memory, schedulers
Memory	Memory grants, usage
Waits	Why queries are waiting
Index	Missing / unused indexes

Surendra Panpaliya

Founder and CEO

GKTCS Innovations

https://www.gktcs.com