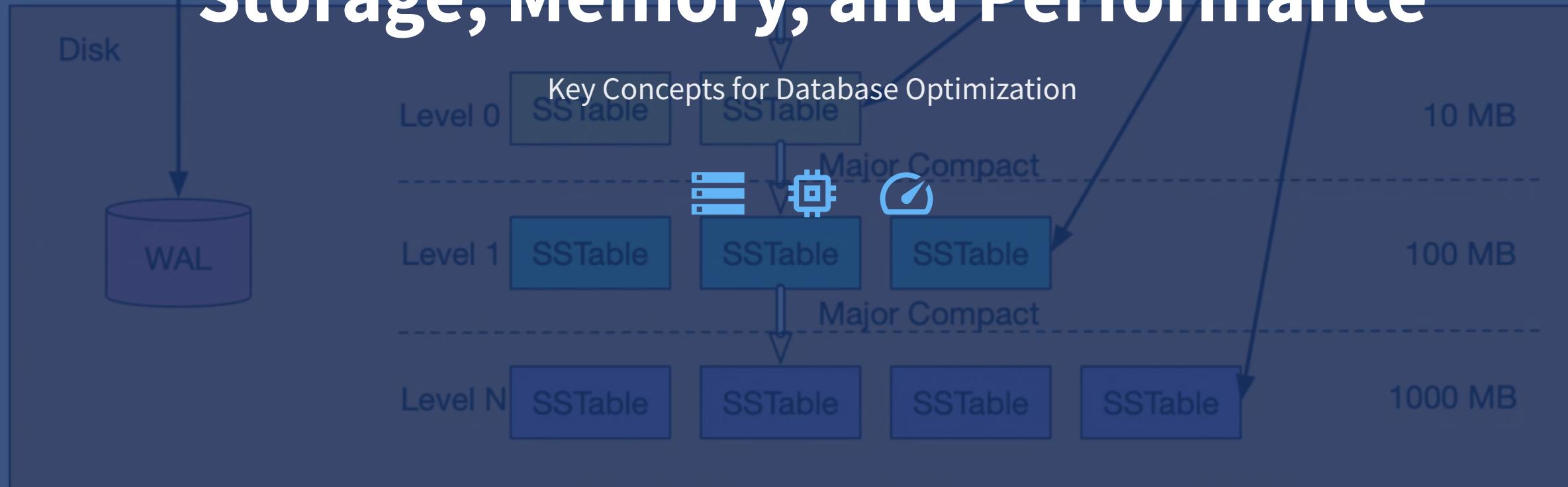


# Database Management Systems: Storage, Memory, and Performance



# Overview

- 1**  Storage Engines and File Structures
- 2**  Memory Management (Buffer Pools, Cache)
- 3**  Parameter Tuning for Performance

# Storage Engines and File Structures

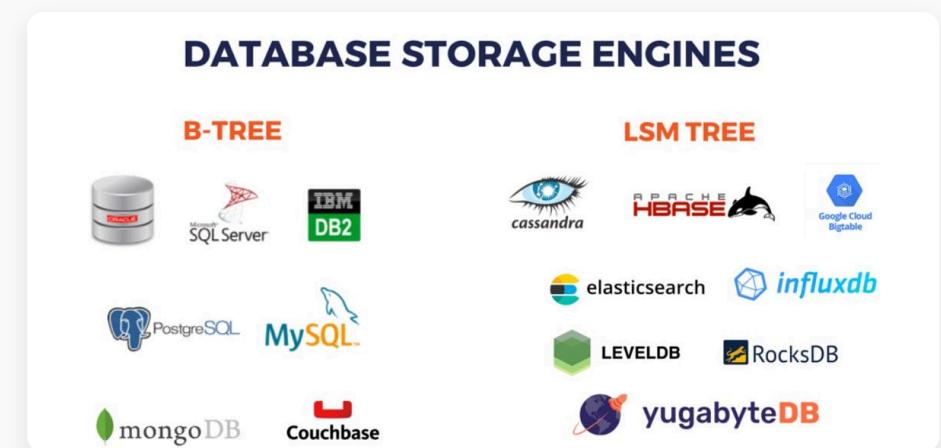
A **storage engine** is the underlying software component that a database management system uses to **store, read, update, and delete** data in memory and storage systems.

## B-Tree Based

Optimized for read performance

## LSM-Tree Based

Optimized for write performance



# B-Tree vs. LSM-Tree Storage Engines

## ■ B-Tree Based

- ✓ **Self-balancing** tree data structure
- ✓ Keeps data **sorted**
- ✓ Operations in **logarithmic time**

### 👍 Pros

- High throughput reads
- Low latency reads
- Efficient for range queries

### 👎 Cons

- Poor write performance
- Random writes expensive
- Read-modify-write penalty

## Real-World Usage

Oracle DB MS SQL Server

IBM DB2

MySQL (InnoDB)

PostgreSQL

## ↗ LSM-Tree Based

- ✓ **Log-structured** merge-tree
- ✓ Defers and **batches** index changes
- ✓ Memory → Disk **cascade**

### 👍 Pros

- Fast sequential writes
- Optimized for high write volume
- Efficient for tiered storage

### 👎 Cons

- Higher read amplification
- More CPU resources during reads
- Takes more memory/disk storage

## Real-World Usage

Apache Cassandra

Elasticsearch

Google Bigtable

Apache HBase

RocksDB

## DATABASE STORAGE ENGINES

### B-TREE



### LSM TREE



# Memory Management in Databases

The DBMS is responsible for **managing memory** and moving data back-and-forth from disk. Data cannot be directly operated on from disk, so databases must efficiently move data from disk to memory.

## ⌚ Buffer Pool

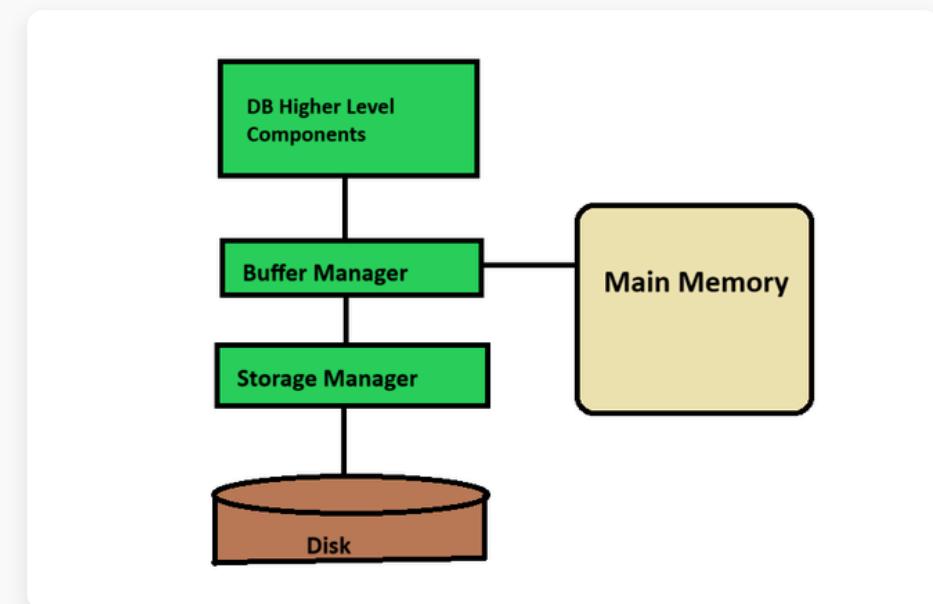
In-memory cache of pages read from disk, organized as an array of fixed-size frames

## ↑ Spatial Control

Where pages are physically written on disk to keep related pages together

## ⌚ Temporal Control

When to read pages into memory and when to write them to disk



# Buffer Pool Architecture and Optimization

## Buffer Pool Organization

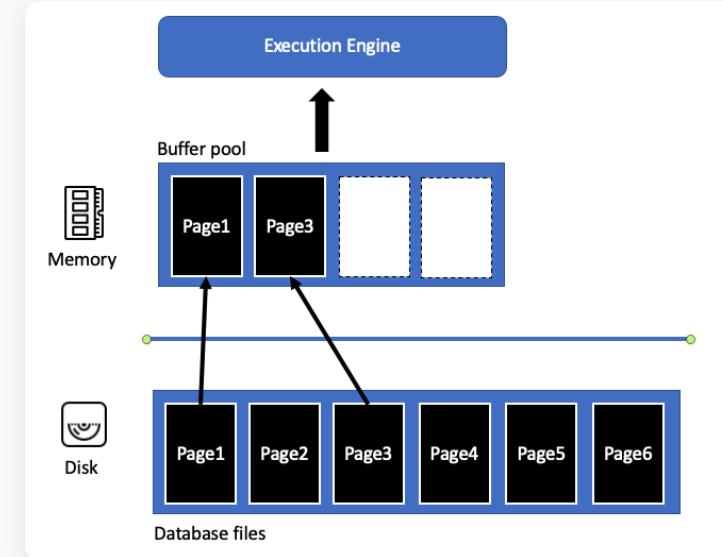
- Array of frames
- Dirty flags tracking
- Page table mapping
- Pin counters for access

## Memory Allocation Policies

- Global - entire workload
- Local - per query/transaction

## Optimization Techniques

- Multiple Buffer Pools  
Per-database or per-page type
- Pre-fetching  
Based on query plan execution
- Scan Sharing  
Multiple queries reuse data
- Buffer Pool Bypass  
Local memory for sequential scans



## Buffer Replacement Policies

- LRU - Least Recently Used
- CLOCK - Approximation of LRU

## Dirty Page Management

- Drop non-dirty pages
- Write back dirty pages

- Background writing periodic

# Parameter Tuning for Performance

**Parameter tuning** involves adjusting configuration settings in a database system to optimize performance, resource utilization, and overall efficiency based on specific workload requirements.



## Enhanced Efficiency

Faster query execution, reduced resource utilization, better hardware exploitation



## Improved Response Times

Swifter results, reduced latency, more responsive user experience



## Reduced Resource Usage

Less server burden, efficient hardware use, potential cost savings



## Scalability

Seamlessly accommodate increased workloads without performance degradation

## ! Why It Matters

Proper parameter tuning ensures databases operate at peak efficiency, providing optimal performance for applications while maximizing resource utilization and minimizing operational costs.

# Common Performance Issues in Databases

## Slow Queries

Suboptimal design, missing indexes, large datasets

## Inefficient Indexing

Too many indexes, redundant indexing, wrong columns

## Locking Problems

Excessive locking, lengthy transactions, conflicts

## Table Fragmentation

Data scattered, suboptimal disk I/O operations

## Improper Configuration

Insufficient resources, not optimized for workload

## Resource Contention

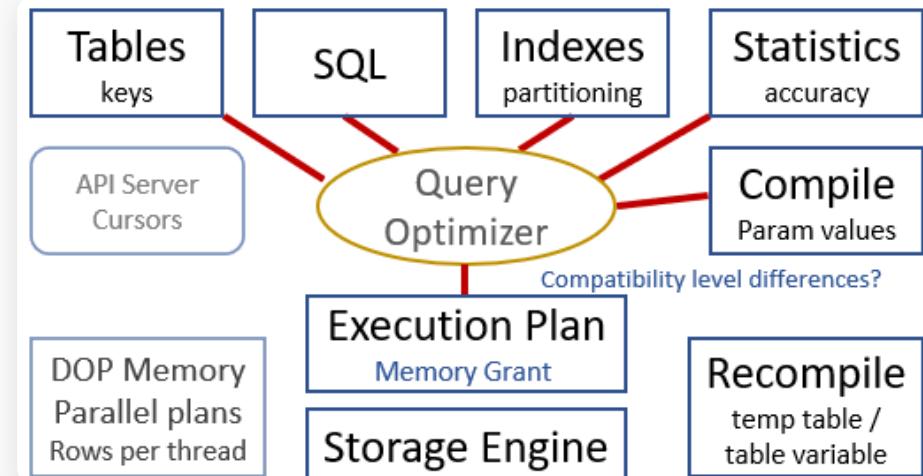
Operations competing for CPU, memory, disk I/O

## Inefficient Query Design

SELECT \*, unnecessary complexity, poor joins

## Inadequate Hardware

Insufficient CPU, memory, or storage capacity



## Performance Tuning Impact

Proper parameter tuning can address these issues by optimizing resource allocation, improving query execution plans, and balancing system workload.

# Key Parameters and Optimization Strategies

## ↔️ Query Optimization

### ✍️ Query Rewriting

Eliminate subqueries, simplify joins, optimize WHERE clauses

### ☰ Indexing

Create indexes on columns in WHERE clauses or JOIN operations

### 📊 Query Plan Analysis

Identify bottlenecks, suboptimal joins, missing indexes

## 💻 Resource Utilization

### 🕒 CPU Usage

Monitor for high processing load from poorly optimized queries

### ☰ Memory Usage

Balance memory consumption with query response times

### 💾 Disk I/O

Optimize queries to reduce disk reads or upgrade storage

## ⚙️ Configuration Settings

### ☰ InnoDB Configuration

Buffer pool size, thread concurrency, transaction isolation

### ⟳ Caching Mechanisms

Query caching, buffer pools, memory allocation strategies

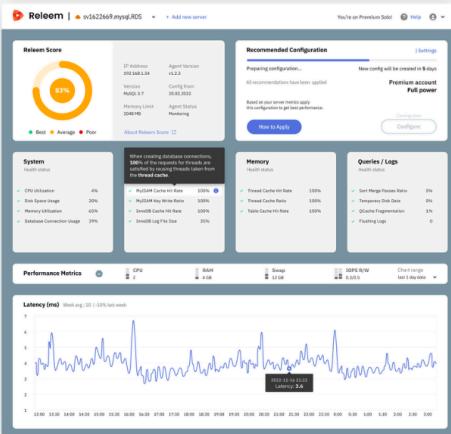


### Easy-to-Use MySQL Performance Tuning and Monitoring Tool

⌚ Simple Installation

🛡️ Open source agent

⚙️ Automation



## ≡ Key Parameters to Monitor

🕒 Buffer Pool Size

⌚ Thread Concurrency

🔒 Transaction Isolation

☰ Query Cache Size

☰ Sort Buffer Size

⌚ Join Buffer Size

💾 Table Open Cache

⌚ Query Timeout

# Summary and Key Takeaways



## Storage Engines

- ▣ **B-Tree** - Optimized for read performance
- ▢ **LSM-Tree** - Optimized for write performance
- ✓ Choose based on workload requirements



## Memory Management

- **Buffer Pool** - In-memory cache of pages
- ⌚ **Optimization** - Multiple pools, pre-fetching
- ⇄ **Replacement** - LRU, CLOCK policies



## Parameter Tuning

- ↔ **Query Optimization** - Rewriting, indexing
- 💻 **Resource Monitoring** - CPU, memory, I/O
- ⚙️ **Configuration** - Buffer pool, caching



## Final Thought

Understanding storage engines, memory management, and parameter tuning is essential for effective database administration and optimal system performance.