DataBase Systems Architecture

Databases III

Author: Eng. Carlos Andrés Sierra, M.Sc.
cavirguezs@udistrital.edu.co

Full-time Adjunct Professor
Computer Engineering Program
School of Engineering
Universidad Distrital Francisco José de Caldas

2025-III

Outline

1. DataBases Infrastructure
2. DBMS Architecture
3. Database System Administration
4. Transactional System
5. Query Execution
6. Concurrency Control
7. Failure Recovery

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Outline

1. DataBases Infrastructure
2. DBMS Architecture
3. Database System Administration
4. Transactional System
5. Query Execution
6. Concurrency Control
7. Failure Recovery

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

What is On-Premises Computing?

- **On-premises computing** is the traditional way of accessing computing resources.
- **On-premises computing** requires users to purchase and maintain their own computing resources such as servers, storage, and databases.

What is the Cloud Computing?

- **Cloud computing** is the delivery of computing services over the internet.
- **Cloud computing** allows users to access computing resources such as servers, storage, and databases **on demand**.

Pros & Cons of Cloud Computing

- **Pros**:
  - **Cost-Effective**: Cloud computing is a cost-effective way to access computing resources.
  - **Scalable**: Cloud computing is a scalable way to access computing resources.
  - **Flexible**: Cloud computing is a flexible way to access computing resources.
- **Cons**:
  - **Security**: Cloud computing can be less secure than on-premises computing.
  - **Performance**: Cloud computing can be slower than on-premises computing.
  - **Reliability**: Cloud computing can be less reliable than on-premises computing.

Pros & Cons of Cloud Computing

- **Pros**:
  - **Cost-Effective**: Cloud computing is a cost-effective way to access computing resources.
  - **Scalable**: Cloud computing is a scalable way to access computing resources.
  - **Flexible**: Cloud computing is a flexible way to access computing resources.
- **Cons**:
  - **Security**: Cloud computing can be less secure than on-premises computing.
  - **Performance**: Cloud computing can be slower than on-premises computing.
  - **Reliability**: Cloud computing can be less reliable than on-premises computing.

SaaS Vs. IaaS Vs. PaaS

- **Software as a Service** (*SaaS*) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the internet.
- **Infrastructure as a Service** (*IaaS*) is a cloud computing model that provides virtualized computing resources over the internet.
- **Platform as a Service** (*PaaS*) is a cloud computing model that provides a platform for developers to build, deploy, and manage applications over the internet.

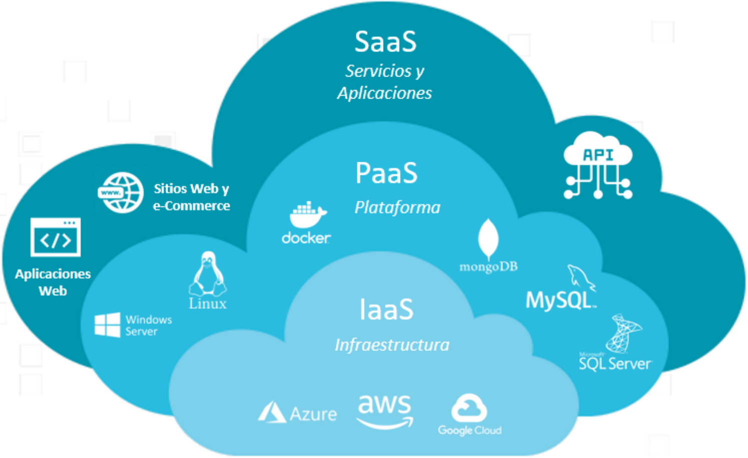
SaaS Vs. IaaS Vs. PaaS

- **Software as a Service** (*SaaS*) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the internet.
- **Infrastructure as a Service** (*IaaS*) is a cloud computing model that provides virtualized computing resources over the internet.
- **Platform as a Service** (*PaaS*) is a cloud computing model that provides a platform for developers to build, deploy, and manage applications over the internet.

SaaS Vs. IaaS Vs. PaaS

- **Software as a Service** (*SaaS*) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the internet.
- **Infrastructure as a Service** (*IaaS*) is a cloud computing model that provides virtualized computing resources over the internet.
- **Platform as a Service** (*PaaS*) is a cloud computing model that provides a platform for developers to build, deploy, and manage applications over the internet.

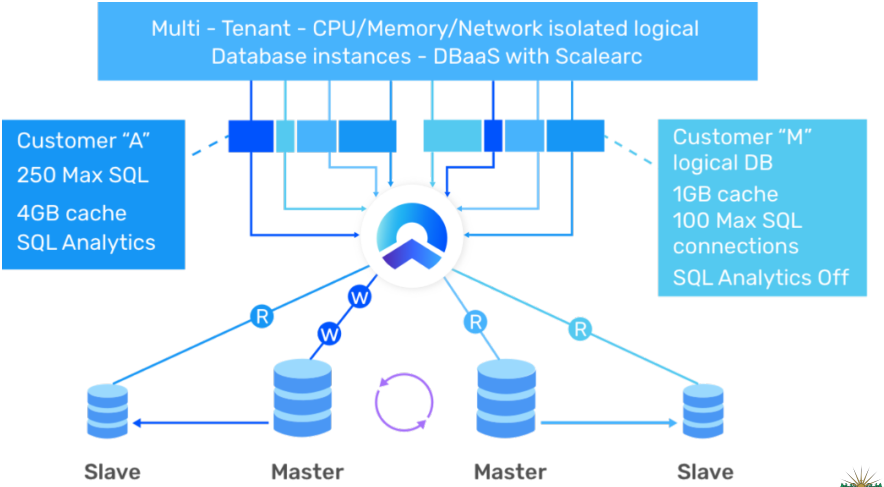
Cloud Levels



DataBases as a Service

Database as a Service (DBaaS) is a cloud computing model that provides database services over the internet.

Case Study: DBaaS Custom for Clients



Multi - Tenant - CPU/Memory/Network isolated logical Database instances - DBaaS with Scalearc

Customer "A"
250 Max SQL
4GB cache
SQL Analytics

Customer "M"
logical DB
1GB cache
100 Max SQL connections
SQL Analytics Off

Outline

1. DataBases Infrastructure
2. DBMS Architecture
3. Database System Administration
4. Transactional System
5. Query Execution
6. Concurrency Control
7. Failure Recovery

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

DBMS Architecture Overview

- A **Database Management System (DBMS)** is organized in layers:
  - `Storage Manager`: Handles *data storage*, *file organization*, and *access methods*.
  - `Query Processor`: Parses, optimizes, and executes *SQL queries*.
  - `Transaction Manager`: Ensures *ACID properties* for transactions.
  - `Concurrency Control Manager`: Manages *simultaneous operations* and prevents conflicts.
  - `Recovery Manager`: Handles *failures* and restores data consistency.
  - `Query Optimizer`: Selects the most efficient *strategy* for executing queries.
- Each component is responsible for a **specific aspect** of data management.

DBMS Architecture Overview

- A **Database Management System (DBMS)** is organized in layers:
  - `Storage Manager`: Handles *data storage*, *file organization*, and *access methods*.
  - `Query Processor`: Parses, optimizes, and executes *SQL queries*.
  - `Transaction Manager`: Ensures *ACID properties* for transactions.
  - `Concurrency Control Manager`: Manages *simultaneous operations* and prevents conflicts.
  - `Recovery Manager`: Handles *failures* and restores data consistency.
  - `Query Optimizer`: Selects the most efficient *strategy* for executing queries.
- Each component is responsible for a **specific aspect** of data management.

DBMS Architecture Overview

- A **Database Management System (DBMS)** is organized in layers:
  - `Storage Manager`: Handles *data storage*, *file organization*, and *access methods*.
  - `Query Processor`: Parses, optimizes, and executes *SQL queries*.
  - `Transaction Manager`: Ensures *ACID properties* for transactions.
  - `Concurrency Control Manager`: Manages *simultaneous operations* and prevents conflicts.
  - `Recovery Manager`: Handles *failures* and restores data consistency.
  - `Query Optimizer`: Selects the most efficient *strategy* for executing queries.
- Each component is responsible for a **specific aspect** of data management.

DBMS Architecture Overview

- A **Database Management System (DBMS)** is organized in layers:
  - `Storage Manager`: Handles *data storage*, *file organization*, and *access methods*.
  - `Query Processor`: Parses, optimizes, and executes *SQL queries*.
  - `Transaction Manager`: Ensures *ACID properties* for transactions.
  - `Concurrency Control Manager`: Manages *simultaneous operations* and prevents conflicts.
  - `Recovery Manager`: Handles *failures* and restores data consistency.
  - `Query Optimizer`: Selects the most efficient *strategy* for executing queries.
- Each component is responsible for a **specific aspect** of data management.

DBMS Architecture Overview

- A **Database Management System (DBMS)** is organized in layers:
  - Storage Manager: Handles *data storage*, *file organization*, and *access methods*.
  - Query Processor: Parses, optimizes, and executes *SQL queries*.
  - Transaction Manager: Ensures *ACID properties* for transactions.
  - Concurrency Control Manager: Manages *simultaneous operations* and prevents conflicts.
  - Recovery Manager: Handles *failures* and restores data consistency.
  - Query Optimizer: Selects the most efficient *strategy* for executing queries.
- Each component is responsible for a **specific aspect** of data management.

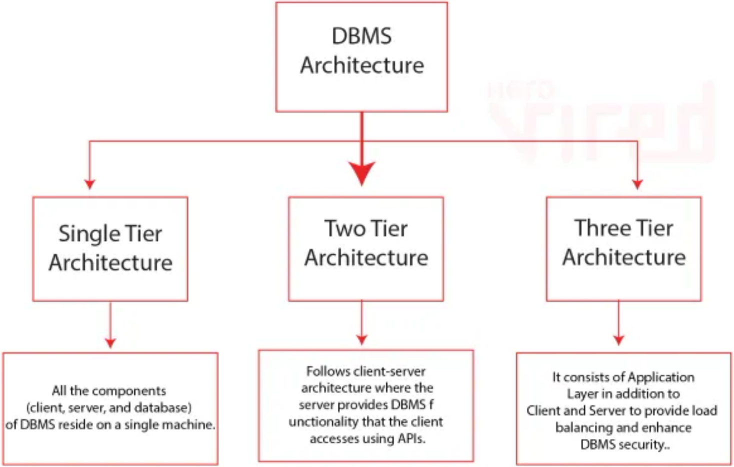
DBMS Architecture Overview

- A **Database Management System (DBMS)** is organized in layers:
  - `Storage Manager`: Handles *data storage*, *file organization*, and *access methods*.
  - `Query Processor`: Parses, optimizes, and executes *SQL queries*.
  - `Transaction Manager`: Ensures *ACID properties* for transactions.
  - `Concurrency Control Manager`: Manages *simultaneous operations* and prevents conflicts.
  - `Recovery Manager`: Handles *failures* and restores data consistency.
  - `Query Optimizer`: Selects the most efficient *strategy* for executing queries.
- Each component is responsible for a **specific aspect** of data management.

DBMS Architecture Overview

- A **Database Management System (DBMS)** is organized in layers:
  - `Storage Manager`: Handles *data storage*, *file organization*, and *access methods*.
  - `Query Processor`: Parses, optimizes, and executes *SQL queries*.
  - `Transaction Manager`: Ensures *ACID properties* for transactions.
  - `Concurrency Control Manager`: Manages *simultaneous operations* and prevents conflicts.
  - `Recovery Manager`: Handles *failures* and restores data consistency.
  - `Query Optimizer`: Selects the most efficient *strategy* for executing queries.
- Each component is responsible for a **specific aspect** of data management.

DBMS Architecture Tiers



DBMS Architecture

Single Tier Architecture

All the components (client, server, and database) of DBMS reside on a single machine.

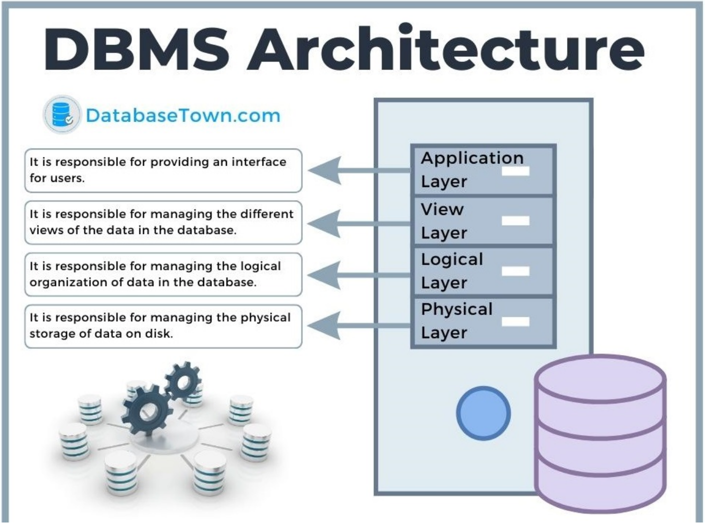
Two Tier Architecture

Follows client-server architecture where the server provides DBMS f unctionality that the client accesses using APIs.

Three Tier Architecture

It consists of Application Layer in addition to Client and Server to provide load balancing and enhance DBMS security..

DBMS Architecture N-Tier



It is responsible for providing an interface for users.

It is responsible for managing the different views of the data in the database.

It is responsible for managing the logical organization of data in the database.

It is responsible for managing the physical storage of data on disk.

Types of DBMS Architecture

There are several types of DBMS architectures:

- **Centralized DBMS**: All components are on a single server.
- **Client-Server DBMS**: Clients access the database through a server.
- **Distributed DBMS**: Data is distributed across multiple servers.
- **Cloud DBMS**: Database services are provided over the cloud.
- **Hybrid DBMS**: Combines features of centralized and distributed systems.
- **Peer-to-Peer DBMS**: Each node can act as a client and server.
- **In memory DBMS**: Data is stored in RAM for faster access.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Types of DBMS Architecture

There are several types of DBMS architectures:

- **Centralized DBMS**: All components are on a single server.
- **Client-Server DBMS**: Clients access the database through a server.
- **Distributed DBMS**: Data is distributed across multiple servers.
- **Cloud DBMS**: Database services are provided over the cloud.
- **Hybrid DBMS**: Combines features of centralized and distributed systems.
- **Peer-to-Peer DBMS**: Each node can act as a client and server.
- **In memory DBMS**: Data is stored in RAM for faster access.

Types of DBMS Architecture

There are several types of DBMS architectures:

- **Centralized DBMS**: All components are on a single server.
- **Client-Server DBMS**: Clients access the database through a server.
- **Distributed DBMS**: Data is distributed across multiple servers.
- **Cloud DBMS**: Database services are provided over the cloud.
- **Hybrid DBMS**: Combines features of centralized and distributed systems.
- **Peer-to-Peer DBMS**: Each node can act as a client and server.
- **In memory DBMS**: Data is stored in RAM for faster access.

Types of DBMS Architecture

There are several types of DBMS architectures:

- **Centralized DBMS**: All components are on a single server.
- **Client-Server DBMS**: Clients access the database through a server.
- **Distributed DBMS**: Data is distributed across multiple servers.
- **Cloud DBMS**: Database services are provided over the cloud.
- **Hybrid DBMS**: Combines features of centralized and distributed systems.
- **Peer-to-Peer DBMS**: Each node can act as a client and server.
- **In memory DBMS**: Data is stored in RAM for faster access.

Types of DBMS Architecture

There are several types of DBMS architectures:

- **Centralized DBMS**: All components are on a single server.
- **Client-Server DBMS**: Clients access the database through a server.
- **Distributed DBMS**: Data is distributed across multiple servers.
- **Cloud DBMS**: Database services are provided over the cloud.
- **Hybrid DBMS**: Combines features of centralized and distributed systems.
- **Peer-to-Peer DBMS**: Each node can act as a client and server.
- **In memory DBMS**: Data is stored in RAM for faster access.

Types of DBMS Architecture

There are several types of DBMS architectures:

- **Centralized DBMS**: All components are on a single server.
- **Client-Server DBMS**: Clients access the database through a server.
- **Distributed DBMS**: Data is distributed across multiple servers.
- **Cloud DBMS**: Database services are provided over the cloud.
- **Hybrid DBMS**: Combines features of centralized and distributed systems.
- **Peer-to-Peer DBMS**: Each node can act as a client and server.
- **In memory DBMS**: Data is stored in RAM for faster access.

Types of DBMS Architecture

There are several types of DBMS architectures:

- **Centralized DBMS**: All components are on a single server.
- **Client-Server DBMS**: Clients access the database through a server.
- **Distributed DBMS**: Data is distributed across multiple servers.
- **Cloud DBMS**: Database services are provided over the cloud.
- **Hybrid DBMS**: Combines features of centralized and distributed systems.
- **Peer-to-Peer DBMS**: Each node can act as a client and server.
- **In memory DBMS**: Data is stored in RAM for faster access.

Outline

1 DataBases Infrastructure

2 DBMS Architecture

3 Database System Administration

4 Transactional System

5 Query Execution

6 Concurrency Control

7 Failure Recovery

Database System Administration

- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their `reliability`, `performance`, `security`, and `availability`.
- Key responsibilities include:
  - Data storage management
  - System architecture design
  - Transaction and query processing
  - Concurrency control
  - Backup and recovery
  - Security and access control
- **Database administrators (DBAs)** are essential for modern organizations.

Database System Administration

- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their `reliability`, `performance`, `security`, and `availability`.
- Key responsibilities include:
    - Data storage management
    - System architecture design
    - Transaction and query processing
    - Concurrency control
    - Backup and recovery
    - Security and access control
- **Database administrators (DBAs)** are essential for modern organizations.

Database System Administration

- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their `reliability`, `performance`, `security`, and `availability`.
- Key responsibilities include:
  - Data storage management
  - System architecture design
  - Transaction and query processing
  - Concurrency control
  - Backup and recovery
  - Security and access control
- **Database administrators (DBAs)** are essential for modern organizations.

Database System Administration

- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their reliability, performance, security, and availability.
- Key responsibilities include:
  - Data storage management
  - System architecture design
  - Transaction and query processing
  - Concurrency control
  - Backup and recovery
  - Security and access control
- **Database administrators (DBAs)** are essential for modern organizations.

Database System Administration

- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their `reliability`, `performance`, `security`, and `availability`.
- Key responsibilities include:
  - Data storage management
  - System architecture design
  - Transaction and query processing
  - Concurrency control
  - Backup and recovery
  - Security and access control
- **Database administrators (DBAs)** are essential for modern organizations.

Database System Administration

- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their `reliability`, `performance`, `security`, and `availability`.
- Key responsibilities include:
  - Data storage management
  - System architecture design
  - Transaction and query processing
  - Concurrency control
  - Backup and recovery
  - Security and access control
- **Database administrators (DBAs)** are essential for modern organizations.

Database System Administration

- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their `reliability`, `performance`, `security`, and `availability`.
- Key responsibilities include:
  - Data storage management
  - System architecture design
  - Transaction and query processing
  - Concurrency control
  - Backup and recovery
  - Security and access control
- **Database administrators (DBAs)** are essential for modern organizations.

Database System Administration

- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their `reliability`, `performance`, `security`, and `availability`.
- Key responsibilities include:
  - Data storage management
  - System architecture design
  - Transaction and query processing
  - Concurrency control
  - Backup and recovery
  - Security and access control
- **Database administrators (DBAs)** are essential for modern organizations.

Record Storage Concepts

- A **record** (or `row/tuple`) is the basic unit of data storage in a *database table*.
- **Efficient record storage** is crucial for fast data retrieval and update.
- `Storage techniques:`
  - Heap storage: Unordered, fast inserts.
  - Sequential storage: Ordered, fast range queries.
  - Indexed storage: Uses indexes for fast lookups.

Record Storage Concepts

- A **record** (or `row/tuple`) is the basic unit of data storage in a *database table*.
- **Efficient record storage** is crucial for fast data retrieval and update.
- `Storage techniques:`
    - Heap storage: Unordered, fast inserts.
    - Sequential storage: Ordered, fast range queries.
    - Indexed storage: Uses indexes for fast lookups.

Record Storage Concepts

- A **record** (or `row/tuple`) is the basic unit of data storage in a *database table*.
- **Efficient record storage** is crucial for fast data retrieval and update.
- `Storage techniques`:
  - Heap storage: Unordered, fast inserts.
  - Sequential storage: Ordered, fast range queries.
  - Indexed storage: Uses indexes for fast lookups.

Record Storage Concepts

- A **record** (or `row/tuple`) is the basic unit of data storage in a *database table*.
- **Efficient record storage** is crucial for fast data retrieval and update.
- `Storage techniques`:
    - Heap storage: Unordered, fast inserts.
    - Sequential storage: Ordered, fast range queries.
    - Indexed storage: Uses indexes for fast lookups.

Record Storage Concepts

- A **record** (or `row/tuple`) is the basic unit of data storage in a *database table*.
- **Efficient record storage** is crucial for fast data retrieval and update.
- `Storage techniques`:
  - Heap storage: Unordered, fast inserts.
  - Sequential storage: Ordered, fast range queries.
  - Indexed storage: Uses indexes for fast lookups.

Block and Page Organization

- Data is stored in **blocks** (pages) on disk.
- Block size and layout affect **I/O performance**.
- Records may be packed, slotted, or may span multiple blocks.
- **Free space management** is important for updates and inserts.
- **Fragmentation management** ensures efficient use of space.
- **Data compression** can also improve storage efficiency.

Block and Page Organization

- Data is stored in **blocks** (pages) on disk.
- Block size and layout affect **I/O performance**.
- Records may be packed, slotted, or may span multiple blocks.
- **Free space management** is important for updates and inserts.
- **Fragmentation management** ensures efficient use of space.
- **Data compression** can also improve storage efficiency.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Block and Page Organization

- Data is stored in **blocks** (pages) on disk.
- Block size and layout affect **I/O performance**.
- Records may be packed, slotted, or may span multiple blocks.
- **Free space management** is important for updates and inserts.
- **Fragmentation management** ensures efficient use of space.
- **Data compression** can also improve storage efficiency.

Block and Page Organization

- Data is stored in **blocks** (pages) on disk.
- Block size and layout affect **I/O performance**.
- Records may be packed, slotted, or may span multiple blocks.
- **Free space management** is important for updates and inserts.
- **Fragmentation management** ensures efficient use of space.
- **Data compression** can also improve storage efficiency.

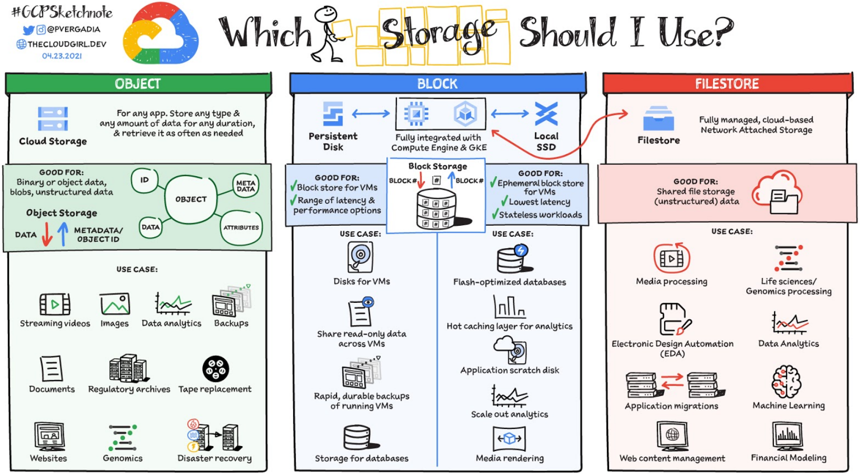
Block and Page Organization

- Data is stored in **blocks** (pages) on disk.
- Block size and layout affect **I/O performance**.
- Records may be packed, slotted, or may span multiple blocks.
- **Free space management** is important for updates and inserts.
- **Fragmentation management** ensures efficient use of space.
- **Data compression** can also improve storage efficiency.

Block and Page Organization

- Data is stored in **blocks** (pages) on disk.
- Block size and layout affect **I/O performance**.
- Records may be packed, slotted, or may span multiple blocks.
- **Free space management** is important for updates and inserts.
- **Fragmentation management** ensures efficient use of space.
- **Data compression** can also improve storage efficiency.

Record Storage: Use Cases



#GCPSketchnote
@PVERGADIA
THECLOUDGIRL.DEV
04.23.2021

Which Storage Should I Use?

OBJECT

Cloud Storage

For any app. Store any type & any amount of data for any duration, & retrieve it as often as needed

GOOD FOR:
Binary or object data, blobs, unstructured data

Object Storage

DATA ↓ ↑ METADATA/ OBJECT ID

USE CASE:

Streaming videos · Images · Data analytics · Backups · Documents · Regulatory archives · Tape replacement · Websites · Genomics · Disaster recovery

BLOCK

Persistent Disk ⟷ Fully integrated with Compute Engine & GKE ⟷ Local SSD

Block Storage

GOOD FOR:
✓ Block store for VMs
✓ Range of latency & performance options

USE CASE:

Disks for VMs · Share read-only data across VMs · Rapid, durable backups of running VMs · Storage for databases

GOOD FOR:
✓ Ephemeral block store for VMs
✓ Lowest latency
✓ Stateless workloads

USE CASE:

Flash-optimized databases · Hot caching layer for analytics · Application scratch disk · Scale out analytics · Media rendering

FILESTORE

Filestore

Fully managed, cloud-based Network Attached Storage

GOOD FOR:
Shared file storage (unstructured) data

USE CASE:

Media processing · Life sciences/ Genomics processing · Electronic Design Automation (EDA) · Data Analytics · Application migrations · Machine Learning · Web content management · Financial Modeling

Outline

1 DataBases Infrastructure

2 DBMS Architecture

3 Database System Administration

4 Transactional System

5 Query Execution

6 Concurrency Control

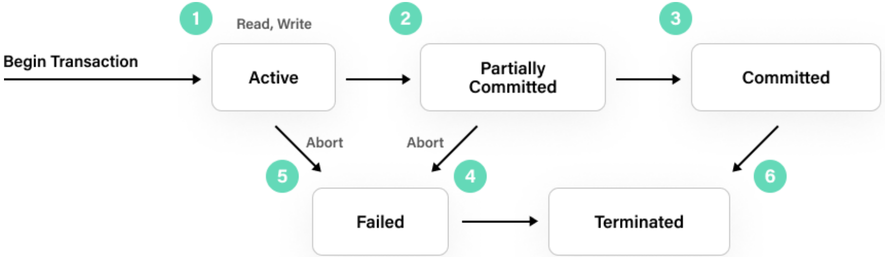
7 Failure Recovery

Transactional System Concepts

- A **transaction** is a sequence of operations performed as a single logical unit of work.
- Transactions must satisfy the **ACID** properties:
  - Atomicity: All or nothing.
  - Consistency: Preserves database integrity.
  - Isolation: Transactions do not interfere.
  - Durability: Results persist after completion.

Transaction Lifecycle

- **Begin**: Transaction starts.
- **Read/Write**: Operations are performed.
- **Commit**: Changes are made permanent.
- **Rollback**: Changes are undone if an error occurs.
- Savepoints can be used for partial rollbacks.



Outline

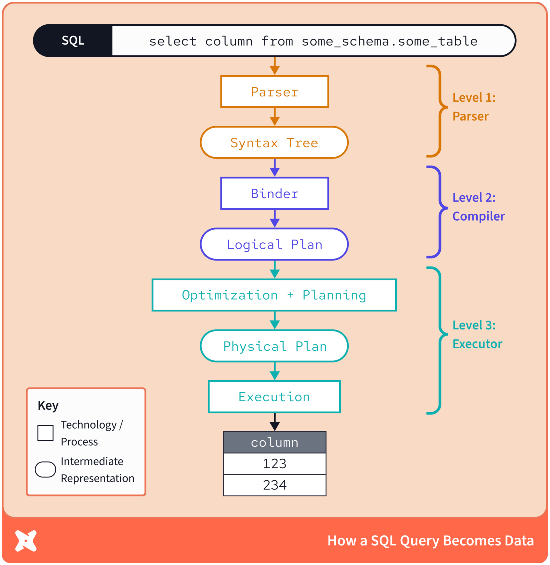
1. DataBases Infrastructure
2. DBMS Architecture
3. Database System Administration
4. Transactional System
5. Query Execution
6. Concurrency Control
7. Failure Recovery

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

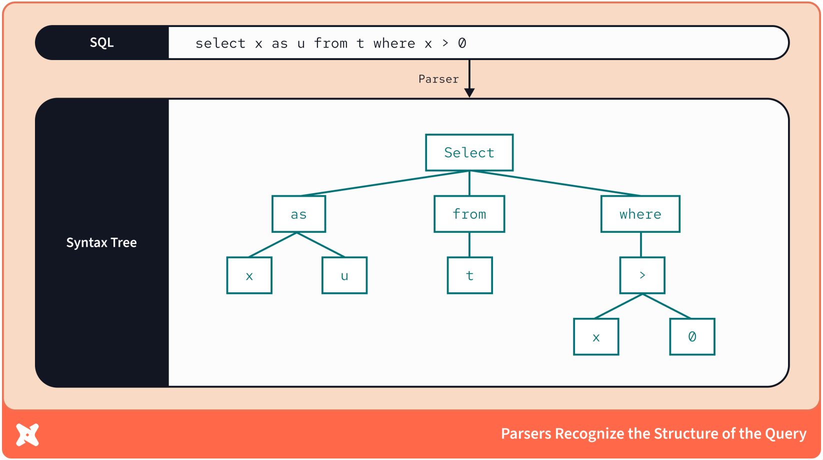
Query Execution Process

- **Query execution** is the process of interpreting and running database queries.
- Steps:
  - Parsing: Analyzing query syntax.
  - Optimization: Choosing the best execution plan.
  - Execution: Retrieving and processing data.
- **Efficient execution** is critical for performance.

Query Execution Flow: Full Transaction



Query Execution Flow: Syntax Tree

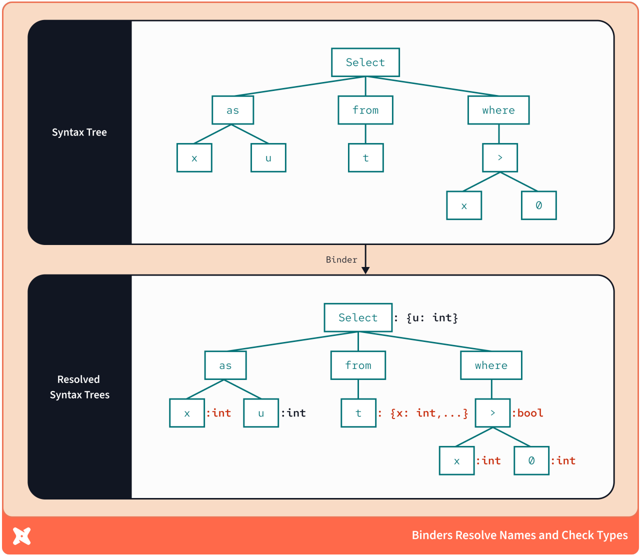


SQL

```
select x as u from t where x > 0
```

Parsers Recognize the Structure of the Query

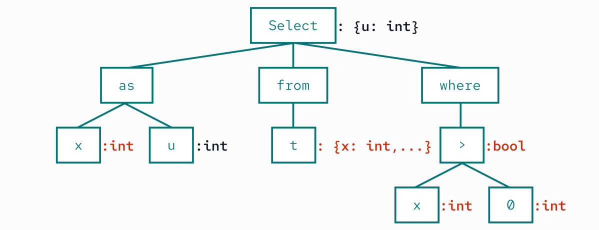
Query Execution Flow: Compilation



Binders Resolve Names and Check Types

Query Execution Flow: Logical Plan

Resolved Syntax Trees



Generating Logical Plan

Logical Plan

```
1. TableScan: t
2. Filter: t.x > 0
3. Projection: t.x as u
```

Compilation Produces an Executable Plan from a Resolved Syntax Tree

Query Optimization

- The **query optimizer** selects the most efficient strategy for executing a query.
- Considers indexes, join methods, and data distribution.
- May rewrite queries for better performance.
- **Cost-based** and **rule-based** optimization approaches.

Query Optimization

- The **query optimizer** selects the most efficient strategy for executing a query.
- Considers indexes, join methods, and data distribution.
- May rewrite queries for better performance.
- **Cost-based** and **rule-based** optimization approaches.

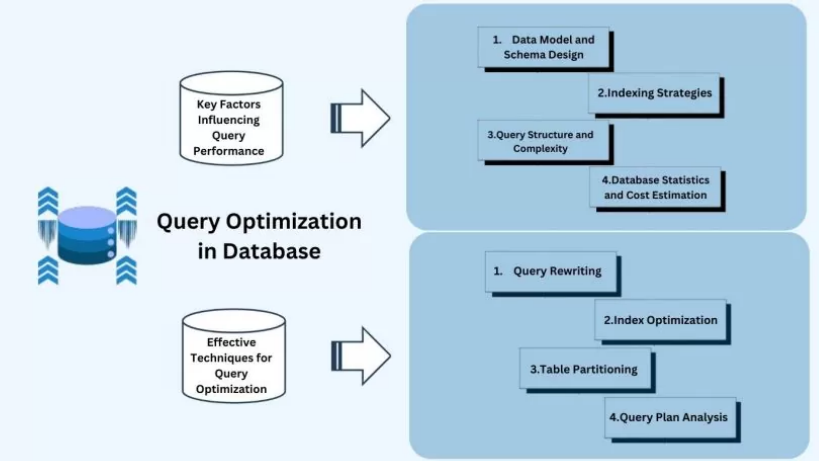
Query Optimization

- The **query optimizer** selects the most efficient strategy for executing a query.
- Considers indexes, join methods, and data distribution.
- May rewrite queries for better performance.
- **Cost-based** and **rule-based** optimization approaches.

Query Optimization

- The **query optimizer** selects the most efficient strategy for executing a query.
- Considers indexes, join methods, and data distribution.
- May rewrite queries for better performance.
- **Cost-based** and **rule-based** optimization approaches.

Query Optimization Factors


Key Factors Influencing Query Performance
1. Data Model and Schema Design
2.Indexing Strategies
3.Query Structure and Complexity
4.Database Statistics and Cost Estimation
Query Optimization in Database
Effective Techniques for Query Optimization
1. Query Rewriting
2.Index Optimization
3.Table Partitioning
4.Query Plan Analysis

Outline

1. DataBases Infrastructure
2. DBMS Architecture
3. Database System Administration
4. Transactional System
5. Query Execution
6. Concurrency Control
7. Failure Recovery

Concurrency Control Concepts

- **Concurrency control** ensures correct results when multiple transactions execute simultaneously.
- Prevents problems like lost updates, dirty reads, and deadlocks.
- Techniques:
    - Locking protocols: Use locks to control access (e.g., two-phase locking).
    - Timestamp ordering: Assigns timestamps to transactions.
    - Optimistic concurrency control: Assumes no conflicts.
    - Pessimistic concurrency control: Assumes conflicts may occur.

Concurrency Control Concepts

- **Concurrency control** ensures correct results when multiple transactions execute simultaneously.
- Prevents problems like lost updates, dirty reads, and deadlocks.
- Techniques:
  - Locking protocols: Use locks to control access (e.g., two-phase locking).
  - Timestamp ordering: Assigns timestamps to transactions.
  - Optimistic concurrency control: Assumes no conflicts.
  - Pessimistic concurrency control: Assumes conflicts may occur.

Concurrency Control Concepts

- **Concurrency control** ensures correct results when multiple transactions execute simultaneously.
- Prevents problems like lost updates, dirty reads, and deadlocks.
- Techniques:
  - Locking protocols: Use locks to control access (e.g., two-phase locking).
  - Timestamp ordering: Assigns timestamps to transactions.
  - Optimistic concurrency control: Assumes no conflicts.
  - Pessimistic concurrency control: Assumes conflicts may occur.

Concurrency Control Concepts

- **Concurrency control** ensures correct results when multiple transactions execute simultaneously.
- Prevents problems like lost updates, dirty reads, and deadlocks.
- Techniques:
  - Locking protocols: Use locks to control access (e.g., two-phase locking).
  - Timestamp ordering: Assigns timestamps to transactions.
  - Optimistic concurrency control: Assumes no conflicts.
  - Pessimistic concurrency control: Assumes conflicts may occur.

Concurrency Control Concepts

- **Concurrency control** ensures correct results when multiple transactions execute simultaneously.
- Prevents problems like lost updates, dirty reads, and deadlocks.
- Techniques:
  - Locking protocols: Use locks to control access (e.g., two-phase locking).
  - Timestamp ordering: Assigns timestamps to transactions.
  - Optimistic concurrency control: Assumes no conflicts.
  - Pessimistic concurrency control: Assumes conflicts may occur.

Locking and Deadlocks

- **Locks** prevent conflicting operations on the same data.
- **Deadlocks** occur when transactions wait indefinitely for each other.
- DBMS uses deadlock detection and resolution strategies.
- **Isolation levels** (e.g., Read Committed, Serializable) control visibility of changes.

Locking and Deadlocks

- **Locks** prevent conflicting operations on the same data.
- **Deadlocks** occur when transactions wait indefinitely for each other.
- DBMS uses deadlock detection and resolution strategies.
- **Isolation levels** (e.g., Read Committed, Serializable) control visibility of changes.

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Locking and Deadlocks

- **Locks** prevent conflicting operations on the same data.
- **Deadlocks** occur when transactions wait indefinitely for each other.
- DBMS uses deadlock detection and resolution strategies.
- **Isolation levels** (e.g., Read Committed, Serializable) control visibility of changes.

Locking and Deadlocks

- **Locks** prevent conflicting operations on the same data.
- **Deadlocks** occur when transactions wait indefinitely for each other.
- DBMS uses deadlock detection and resolution strategies.
- **Isolation levels** (e.g., Read Committed, Serializable) control visibility of changes.

Outline

1. DataBases Infrastructure
2. DBMS Architecture
3. Database System Administration
4. Transactional System
5. Query Execution
6. Concurrency Control
7. Failure Recovery

Failure Recovery Concepts

- **Failure recovery** restores the database to a consistent state after a failure.
- Types of failures:
  - Transaction failure: Only one transaction fails.
  - System crash: All active transactions are lost.
  - Media failure: Disk or storage device fails.
- Recovery techniques:
  - Write-ahead logging (WAL):
  - Checkpointing:
  - Shadow paging:

Failure Recovery Concepts

- **Failure recovery** restores the database to a consistent state after a failure.
- Types of failures:
  - Transaction failure: Only one transaction fails.
  - System crash: All active transactions are lost.
  - Media failure: Disk or storage device fails.
- Recovery techniques:
  - Write-ahead logging (WAL):
  - Checkpointing:
  - Shadow paging:

Failure Recovery Concepts

- **Failure recovery** restores the database to a consistent state after a failure.
- Types of failures:
  - Transaction failure: Only one transaction fails.
  - System crash: All active transactions are lost.
  - Media failure: Disk or storage device fails.
- Recovery techniques:
  - Write-ahead logging (WAL):
    - Logs changes before applying them.
    - Ensures durability and atomicity.
  - Checkpointing:
    - Periodically saves the database state.
    - Reduces recovery time.
  - Shadow paging:
    - Maintains copies of data pages.
    - Allows quick recovery to a consistent state.

Failure Recovery Concepts

- **Failure recovery** restores the database to a consistent state after a failure.
- Types of failures:
  - Transaction failure: Only one transaction fails.
  - System crash: All active transactions are lost.
  - Media failure: Disk or storage device fails.
- Recovery techniques:
  - Write-ahead logging (WAL):
    - Logs changes before applying them.
    - Ensures durability and atomicity.
  - Checkpointing:
    - Periodically saves the database state.
    - Reduces recovery time.
  - Shadow paging:
    - Maintains copies of data pages.
    - Allows quick recovery to a consistent state.

Failure Recovery Concepts

- **Failure recovery** restores the database to a consistent state after a failure.
- Types of failures:
  - Transaction failure: Only one transaction fails.
  - System crash: All active transactions are lost.
  - Media failure: Disk or storage device fails.
- Recovery techniques:
  - Write-ahead logging (WAL):
    - Logs changes before applying them.
    - Ensures durability and atomicity.
  - Checkpointing:
    - Periodically saves the database state.
    - Reduces recovery time.
  - Shadow paging:
    - Maintains copies of data pages.
    - Allows quick recovery to a consistent state.

Write-Ahead Logging and Checkpoints

- **Write-ahead logging (WAL)**: All changes are logged before being applied to the database.
- **Checkpoints**: Periodically save the database state to speed up recovery.
- **Shadow paging**: Maintains copies of data pages.
- Recovery ensures atomicity and durability.

Write-Ahead Logging and Checkpoints

- **Write-ahead logging (WAL)**: All changes are logged before being applied to the database.
- **Checkpoints**: Periodically save the database state to speed up recovery.
- **Shadow paging**: Maintains copies of data pages.
- Recovery ensures atomicity and durability.

Write-Ahead Logging and Checkpoints

- **Write-ahead logging (WAL)**: All changes are logged before being applied to the database.
- **Checkpoints**: Periodically save the database state to speed up recovery.
- **Shadow paging**: Maintains copies of data pages.
- Recovery ensures atomicity and durability.

Write-Ahead Logging and Checkpoints

- **Write-ahead logging (WAL)**: All changes are logged before being applied to the database.
- **Checkpoints**: Periodically save the database state to speed up recovery.
- **Shadow paging**: Maintains copies of data pages.
- Recovery ensures atomicity and durability.

Outline

1. DataBases Infrastructure
2. DBMS Architecture
3. Database System Administration
4. Transactional System
5. Query Execution
6. Concurrency Control
7. Failure Recovery

Thanks!

Questions?

Repo: `https://github.com/EngAndres/ud-public/tree/main/courses/databases-ii`