

DATA BASE SYSTEMS ARCHITECTURE

Databases III

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

Lecturer
Department of Computer Engineer
School of Engineering
Universidad Distrital Francisco José de Caldas

2025-I



Outline

1 Database System Administration



2 Record Storage



3 DBMS Architecture



4 Transactional System



5 Query Execution



6 Concurrency Control



7 Failure Recovery



Outline

1 Database System Administration

2 Record Storage

3 DBMS Architecture

4 Transactional System

5 Query Execution

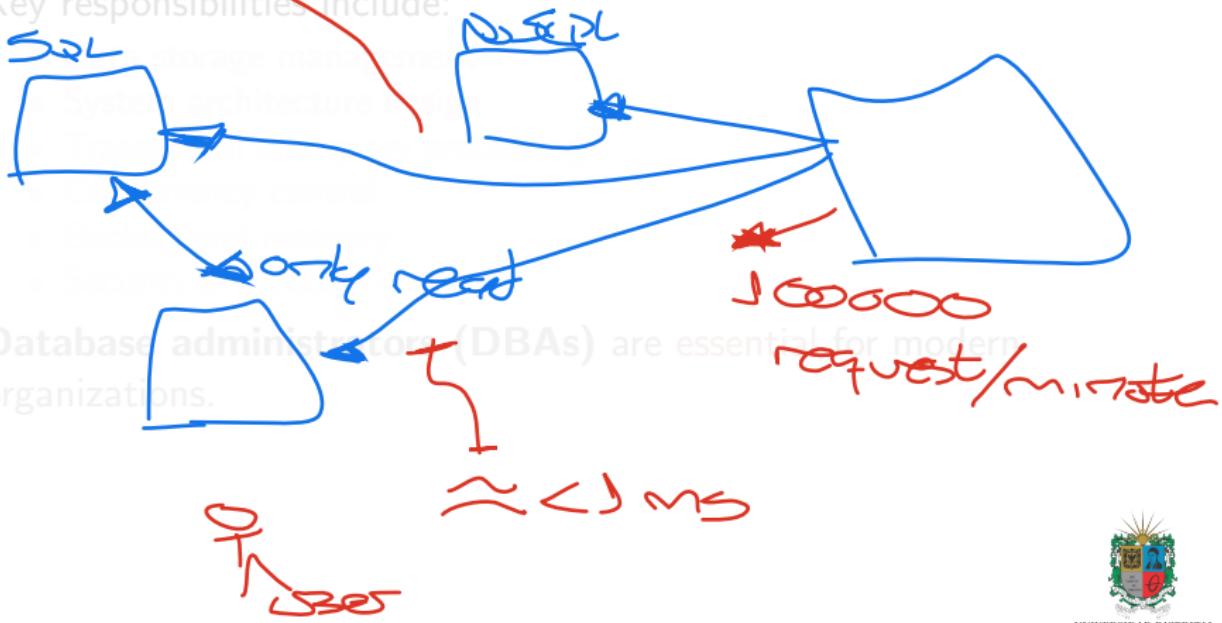
6 Concurrency Control

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:



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

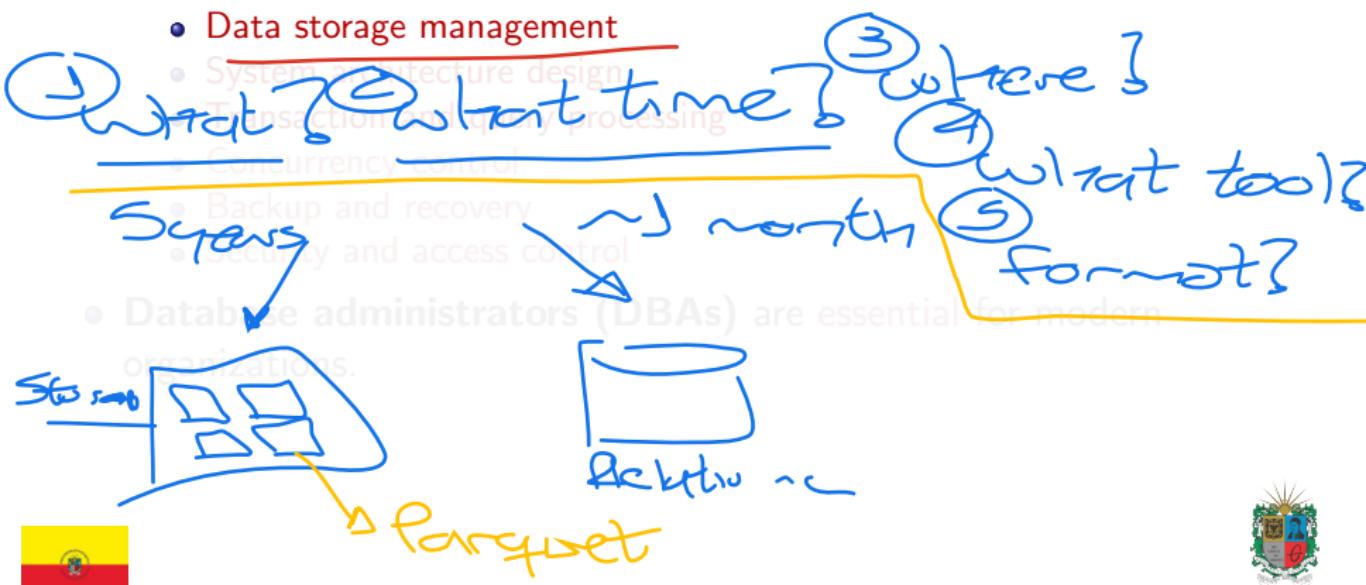
- Transaction processing

- Concurrency control

- Backup and recovery

- Security and access control

- Database administrators (DBAs) are essential for modern



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

physical
infrastructure

on-premises? cloud? hybrid?

logical architecture → SQL? NoSQL?
mirrors? sync? scaling?
ETL-ELT? data lakes?
data warehouses? data mart?



Database System Administration

- **Database system administration** is the discipline of **managing**, **configuring**, and **maintaining database systems** to ensure their reliability, performance, security, and availability.

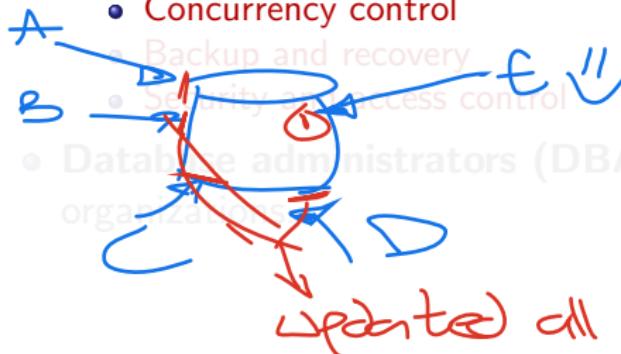
Key responsibilities include:

- Data storage management
 - System architecture design
 - Transaction and **query processing**
 - Concurrency control
 - Backup and recovery
 - Security and access control
- response time? Concurrency?
- Order in queries? optimization?
- 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



- 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.

incremental



exit tolerant?
copy -> availability
self-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.

SSH

Grants?

ZFA → authZ

authenticator

protocols



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.



Outline

1 Database System Administration

2 Record Storage

3 DBMS Architecture

4 Transactional System

5 Query Execution

6 Concurrency Control

7 Failure Recovery



Record Storage Concepts

- A **record** (or row/tuple) is the basic unit of data storage in a database table.
- Efficient record storage is useful for fast data retrieval and update.
- Storage techniques
 - Plain text file
 - Metadata
 - Sparse matrix



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.

SELECT
FROM

1. Operation with most executions
2. Cost ↗
 - ↳ money
 - ↳ resources

Create
Update
Delete
Save (row)



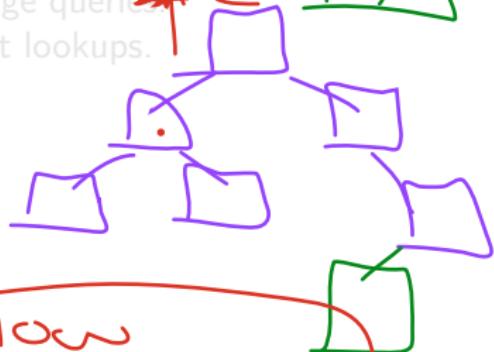
Record Storage Concepts

- 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.



Grant &
Freeze



Slow
Search

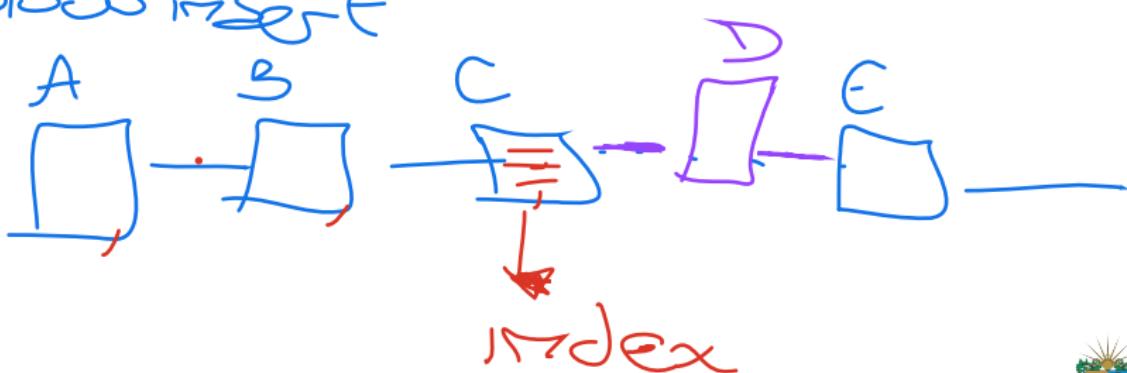


Record Storage Concepts

- 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.

Slow insert



Record Storage Concepts

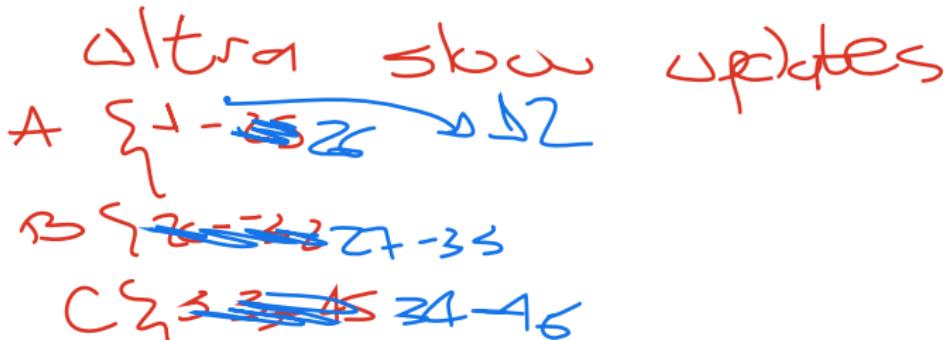
- A **record** (or row/tuple) is the **basic unit** of data storage in a *database table*.
- **Efficient record storage** is crucial for **fast** data retrieval and **update**.
- Storage techniques:
 - **Heap storage**: Unordered, fast inserts.
 - **Sequential storage**: Ordered, fast range queries.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

ultra slow updates

A { 1 - 25 2 → 12

B { 2 - 23 27 - 35

C { 3 - 25 24 - 46



Block and Page Organization

- Data is stored in **blocks** (pages) on disk.
- Block size and layout affect I/O performance.
- Records may be packed, or 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.

video



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 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 span multiple blocks.
- Free space management is important for updates and inserts.
- Fragmentation affects performance.
- Data compression can also improve storage efficiency.

+ storage

+ hard to migrate



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 **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.

2 G → J. S G
Free



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 **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

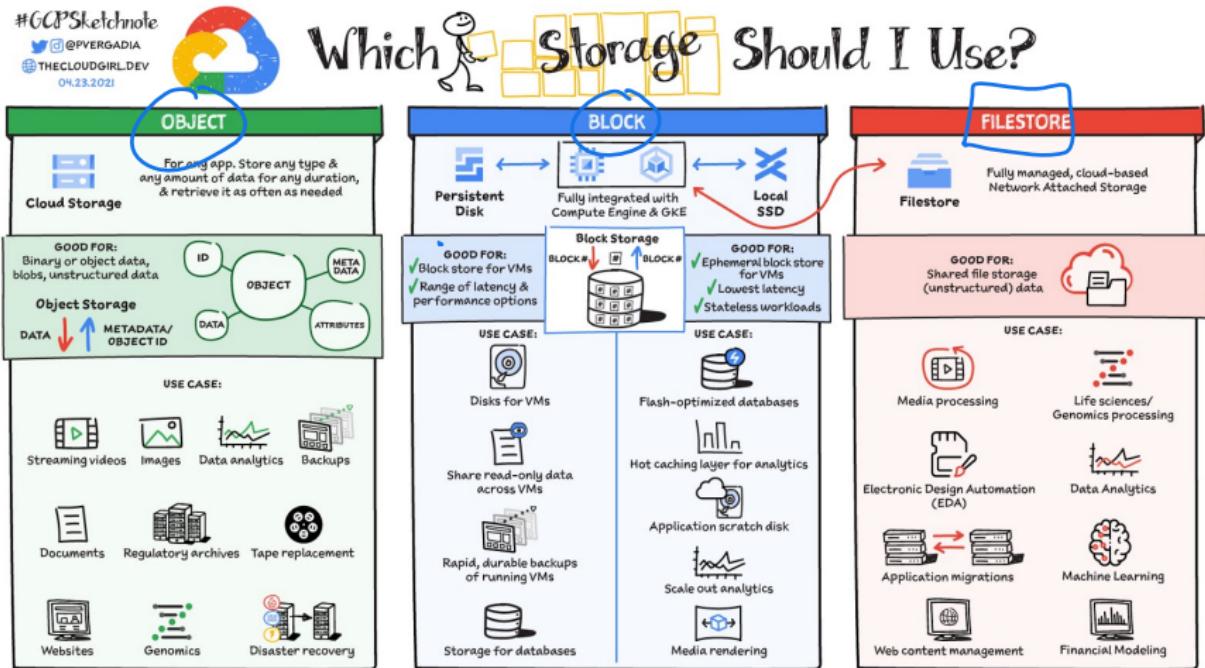
- Data is stored in **blocks** (pages) on disk.
- **Block size** and **layout** affect **I/O performance**.
- Records may be **packed**, **slotted**, or **span** multiple blocks.
- **Free space management** is important for **updates** and **inserts**.
- **Fragmentation management** ensures **efficient**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

|
- Redundant
+ Fast load



Record Storage: Image

#GAPSketchnote
 @PVERGADIA
THECLOUDGIRL.DEV
 04.23.2021



Outline

1 Database System Administration

2 Record Storage

3 DBMS Architecture

4 Transactional System

.

5 Query Execution

6 Concurrency Control

7 Failure Recovery



DBMS Architecture Overview

- A Database Management System (**DBMS**) is organized in layers:
 - **Storage Manager**: Handles data storage, file organization, and access methods.
 - **File Manager**: Manages files and provides interfaces for data retrieval.
 - **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:

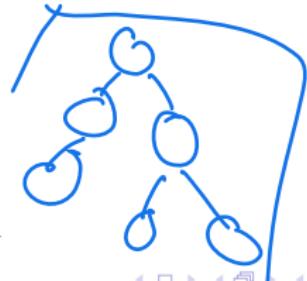
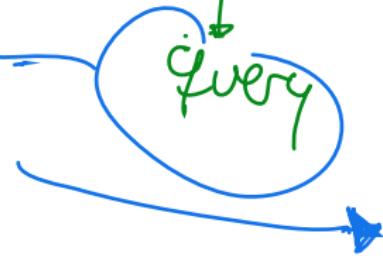
- Storage Manager:** Handles *data storage, file organization, and access methods.*
- Query Processor:** Parses, optimizes, and executes *SQL queries.*
- Transaction Manager:** Ensures *ACID properties for transactions.*
- Concurrency Control Manager:** Manages simultaneous operations without conflicts.
- Recovery Manager:** Handles failures and restores data consistency.
- Query Optimizer:** Selects the most efficient strategy for executing queries.

Declarative → what

for raw in tableA:
data.append(row)

- Each component is responsible for a **specific aspect** of the system.

Functional
Programming



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.

Atomicity → Durability, Consistency.

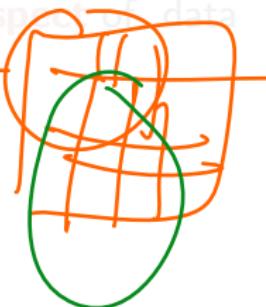
Isolated

Durability

request

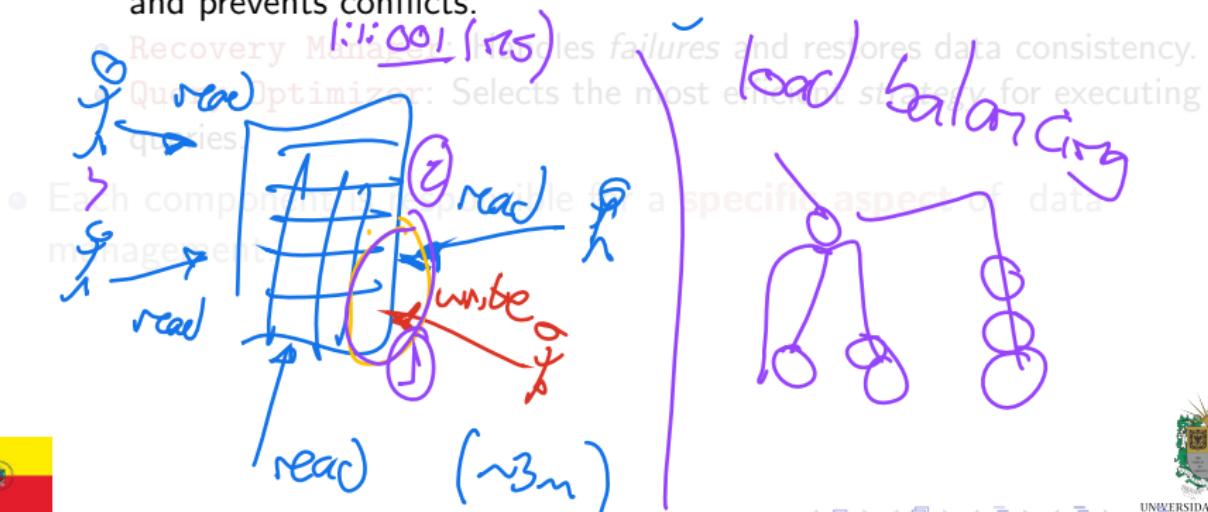
Queue

execute



DBMS Architecture Overview

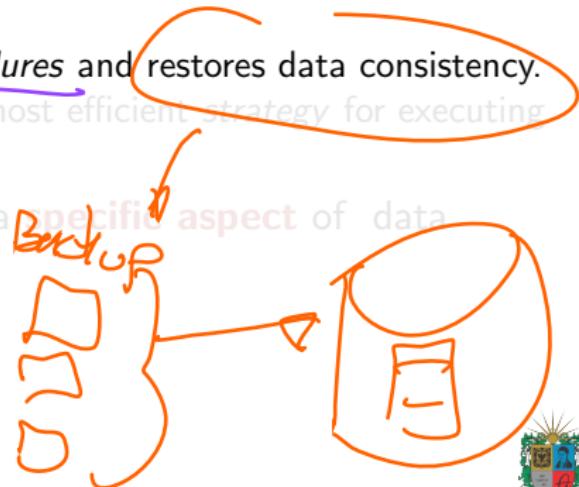
- A Database Management System (DBMS) is organized in **layers**:
 - **Storage Manager**: Handles *data storage, file organization, and access methods*.
 - **Query Processor**: Parses, optimizes, and executes *SQL queries*.
 - **Transaction Manager**: Ensures *ACID properties* for transactions.
 - **Concurrency Control Manager**: Manages *simultaneous operations* and prevents conflicts.



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.

If I need money
 for restart
 copy A → B
 en Crypt



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.

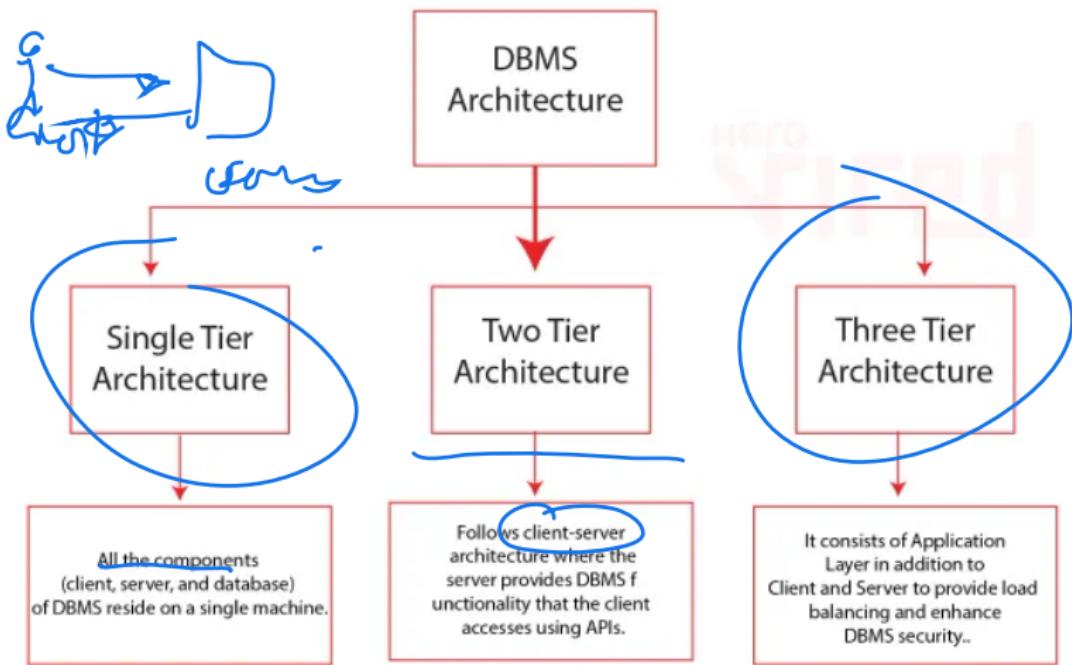


DBMS Architecture Overview

- 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 N-Tier

DBMS Architecture



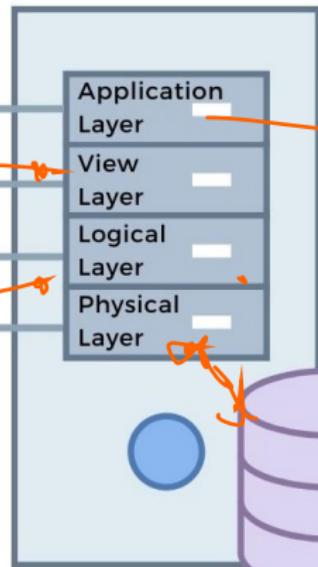
DatabaseTown.com

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.

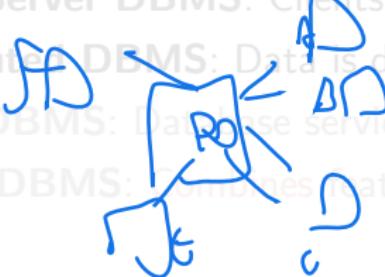
It is responsible for managing the physical storage of data on disk.



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 natures 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:

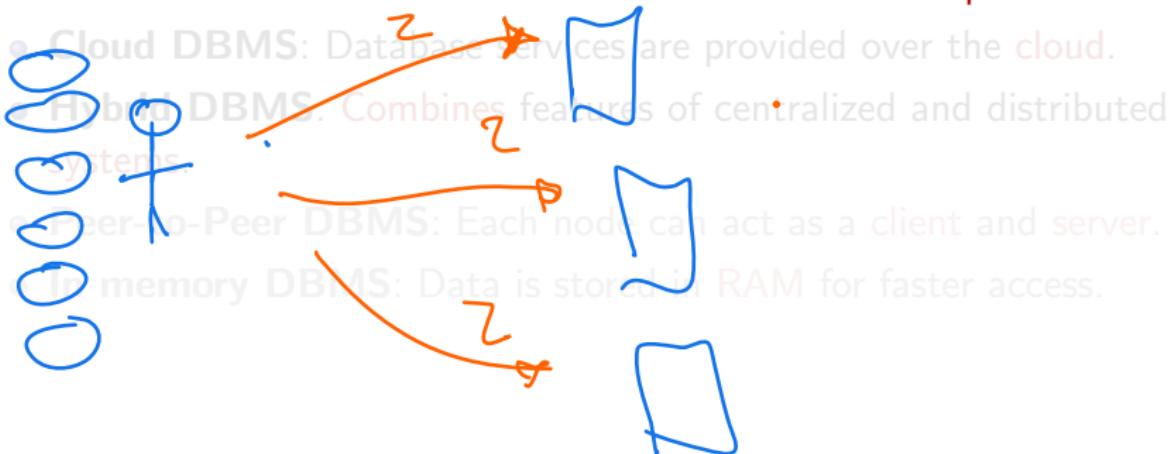
- **Centralized DBMS**: All components are on a **single server**.
- **Client-Server DBMS**: Clients **access** the database through a **server**.
- **Distributed DBMS**: Data is distributed across **multiple servers**.
- **Cloud DBMS**: Database services are provided over the **cloud**.
- **Hybrid DBMS**: 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**.



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.

\$ downside



Types of DBMS Architecture

There are several types of DBMS architectures:

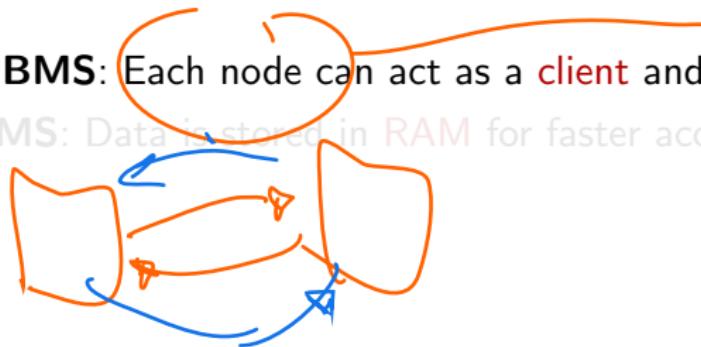
- **Centralized DBMS**: All components are on a **single server**.
- **Client-Server DBMS**: Clients **access** the database through a **server**.
- **Distributed DBMS**: Data is distributed across **multiple servers**.
- **Cloud DBMS**: Database services are provided over the **cloud**.
- **Hybrid DBMS**: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



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.

Cadé
↳ Redis



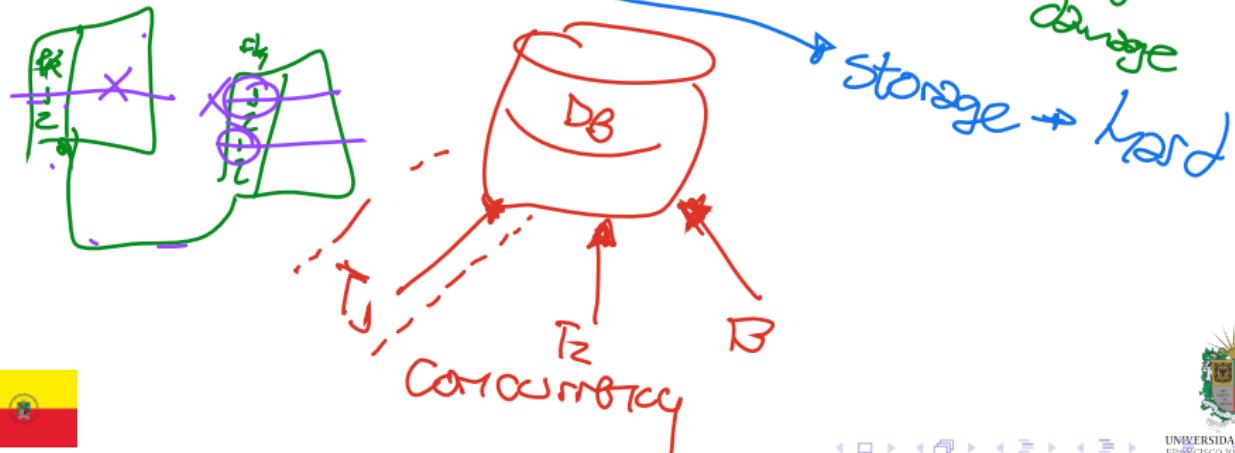
Outline

- 1 Database System Administration
- 2 Record Storage
- 3 DBMS Architecture
- 4 Transactional System
- 5 Query Execution
- 6 Concurrency Control
- 7 Failure Recovery



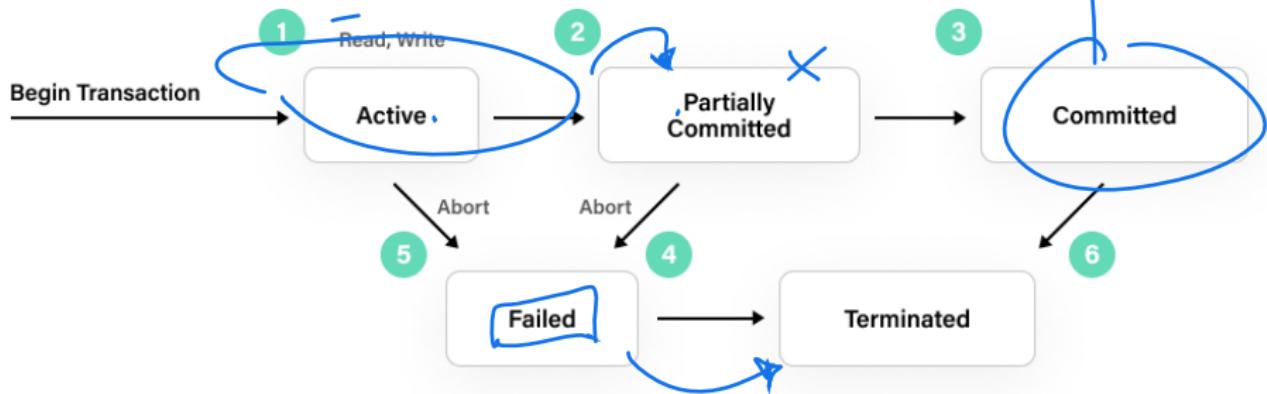
Transactional System Concepts

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



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.



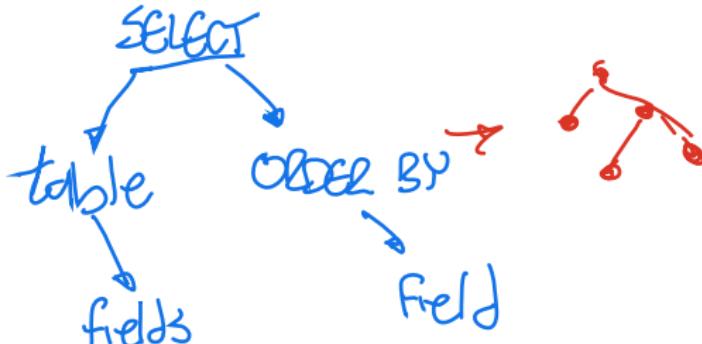
Outline

- 1 Database System Administration
- 2 Record Storage
- 3 DBMS Architecture
- 4 Transactional System
- 5 Query Execution
- 6 Concurrency Control
- 7 Failure Recovery



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.

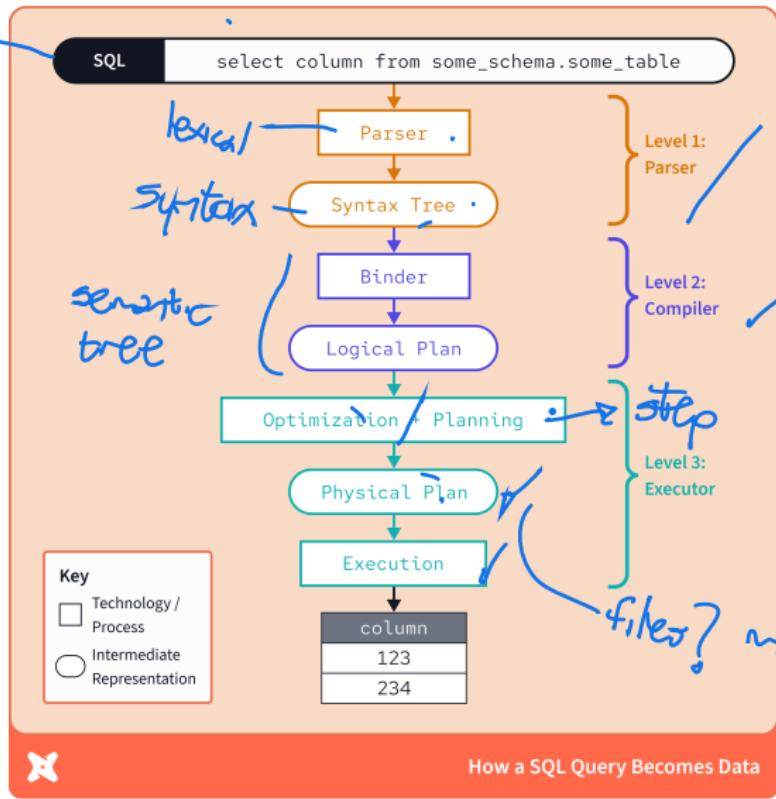


how? → orgine
 for (...)
 if (...) words
 lexical
 ↓
 syntax → order

~~INTO~~ ~~INSERT~~
 INSERT INTO



Query Execution Flow: Full Transaction



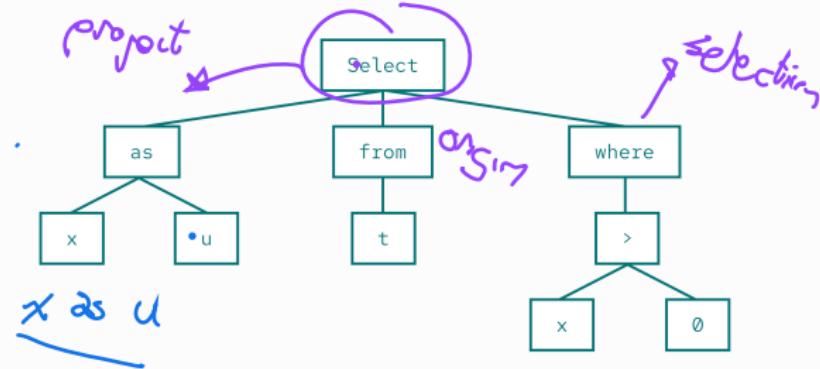
Query Execution Flow: Syntax Tree

SQL

select x as u from t where x > 0

Parser

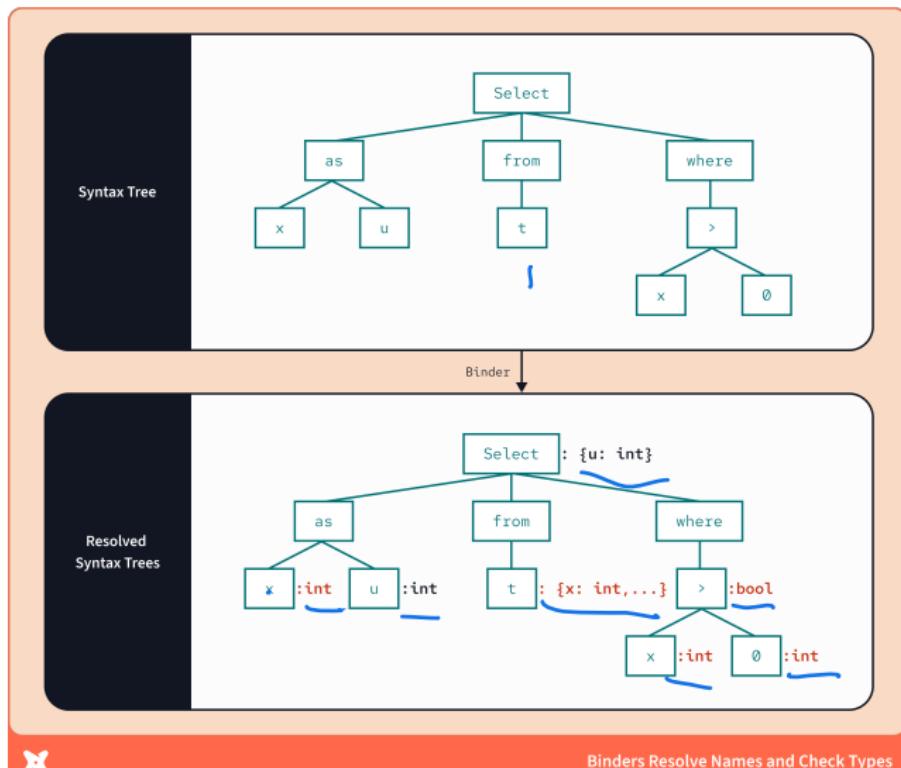
Syntax Tree



Parsers Recognize the Structure of the Query

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Query Execution Flow: Compilation

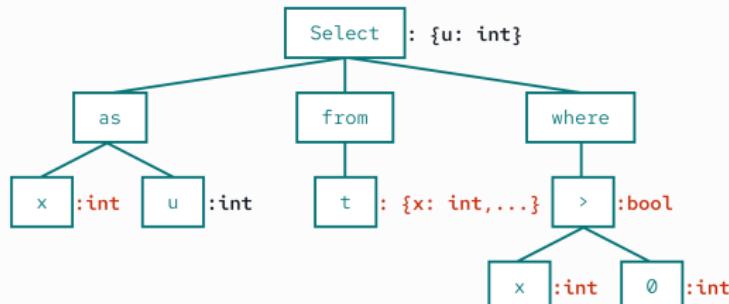


Binders Resolve Names and Check Types



Query Execution Flow: Logical Plan

Resolved
Syntax Trees



Logical Plan

1. TableScan: $t \rightarrow$ for condition filter
 2. Filter: $t.x > 0 \rightarrow$ condition for filter
 3. Projection: $t.x \text{ as } u \rightarrow$ for t and



Compilation Produces an Executable Plan from a Resolved Syntax Tree



Query Optimization

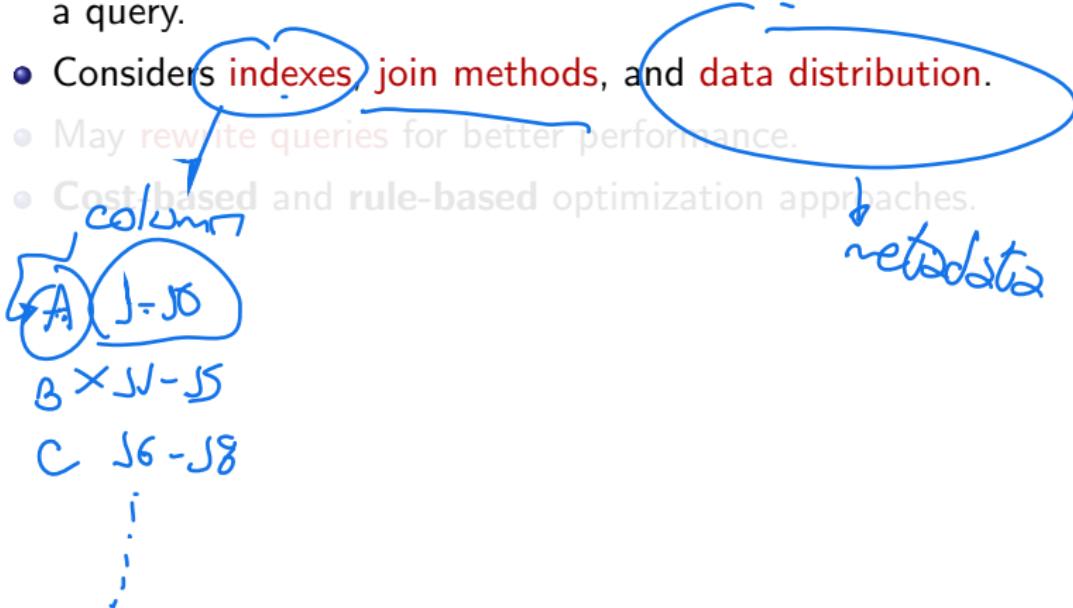
- The query optimizer selects the most efficient strategy for executing a query.
- Considers indexes, join methods, and data distribution.
- May rewrite queries for better performance.
 - ~~re-write~~ *processor*
 - *amount & order*

plan



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.

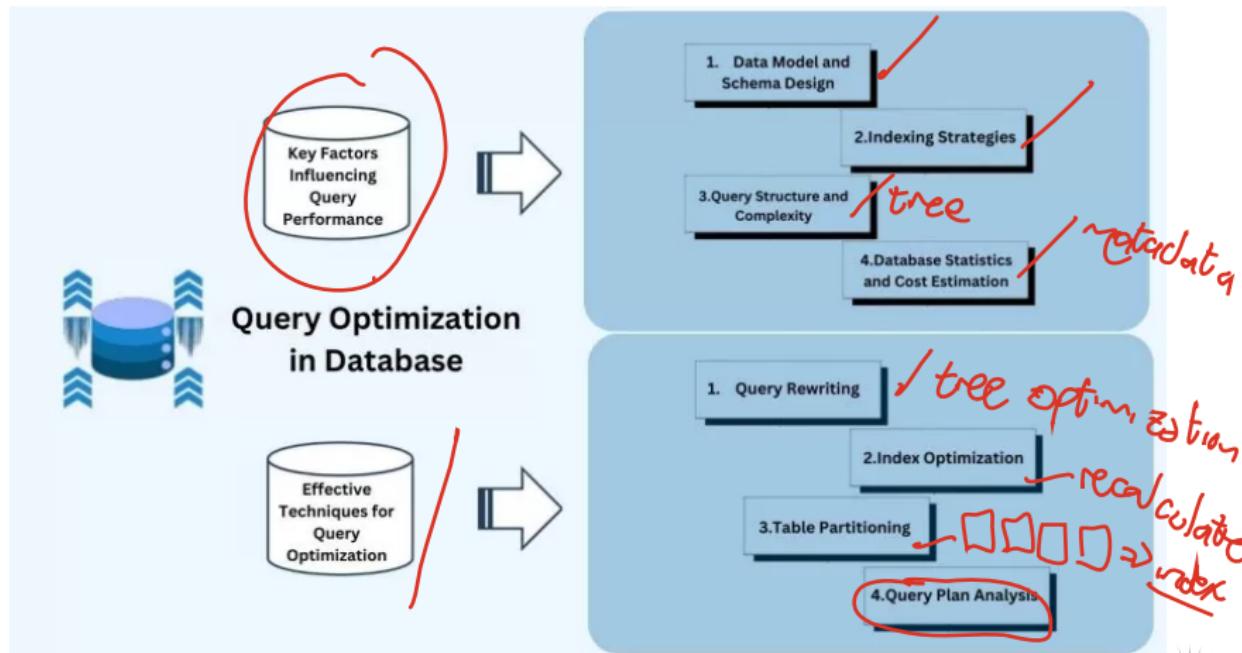


Query Optimization

- 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



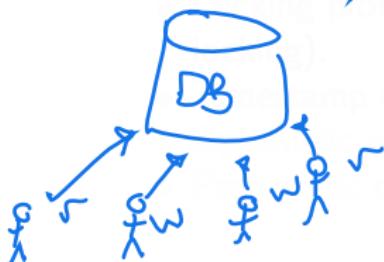
Outline

- 1 Database System Administration
- 2 Record Storage
- 3 DBMS Architecture
- 4 Transactional System
- 5 Query Execution
- 6 Concurrency Control
- 7 Failure Recovery

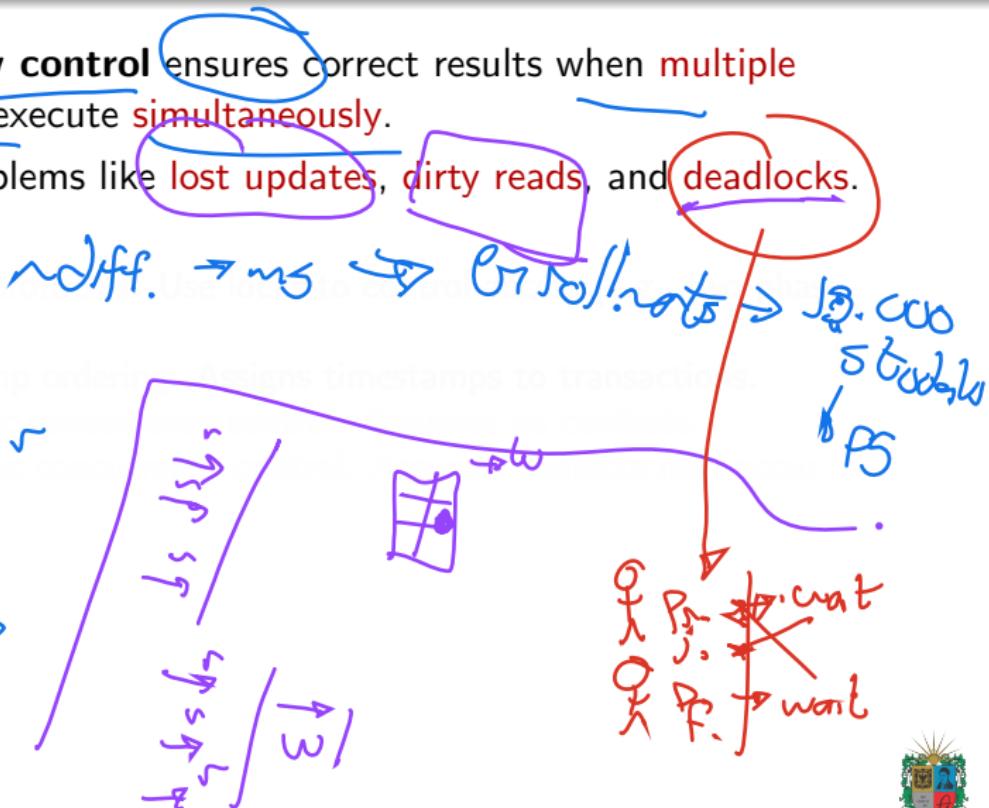


Concurrency Control Concepts

- **Concurrency control** ensures correct results when multiple transactions execute simultaneously.
- Prevents problems like lost updates, dirty reads, and deadlocks.
- Techniques:



Decisions?



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.

Consistency + Integrity

Write → lock read

read → lock write

- waiting time

- update

- memory check

- queues



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.

$T_1: J: 1.00$ $T_2: J: 1.00$ \rightarrow *Fast*

Dirty Reads



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.

→ Parallel without locks

- ↳ reduce queues
- ↳ memory

High response time



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 + timestamp



Locking and Deadlocks

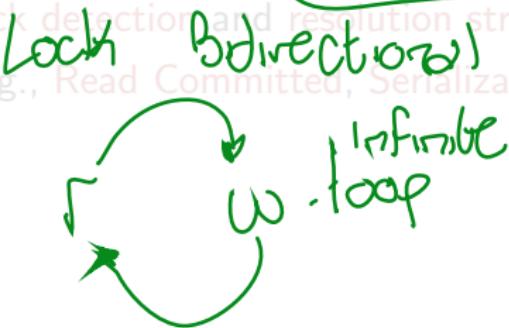


- **Locks** prevent conflicting operations on the same data.
- Deadlocks occur when transactions wait indefinitely for each other.
- DBMS uses deadlock detection and resolution strategies.
- Isolation levels (e.g., Read Committed, Serializable) control visibility of changes.



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.



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.

Loops?



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.

ACID



Outline

- 1 Database System Administration
- 2 Record Storage
- 3 DBMS Architecture
- 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 fails
actions are lost

fast → good
healthy

- Recovery techniques:

→ last stable version } methods



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:

• Checkpointing
• Log-based recovery (REDO/UNDO)

domino effect

process close

ACID

• Checkpointing



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.



Plain-best



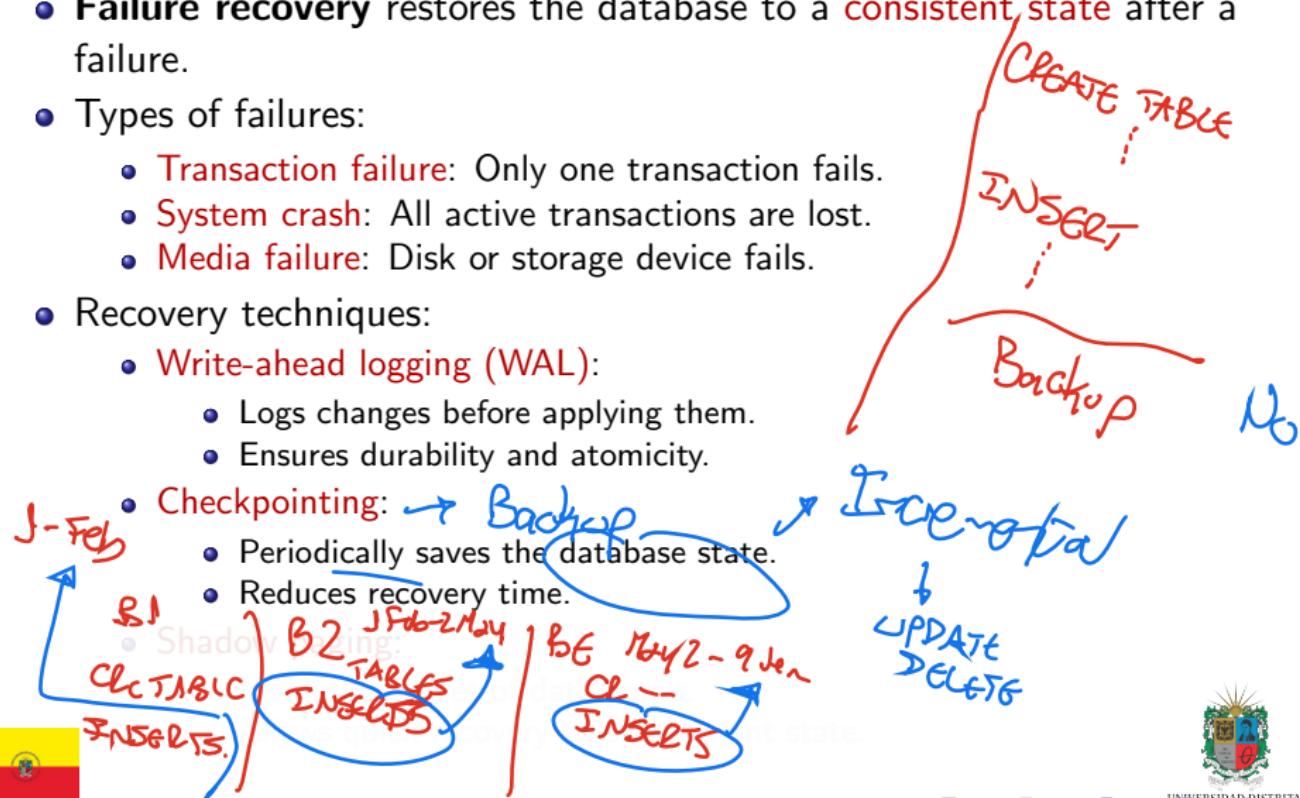
hour/date over action

} CEGATE
 UPDATE
 DELETE



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.



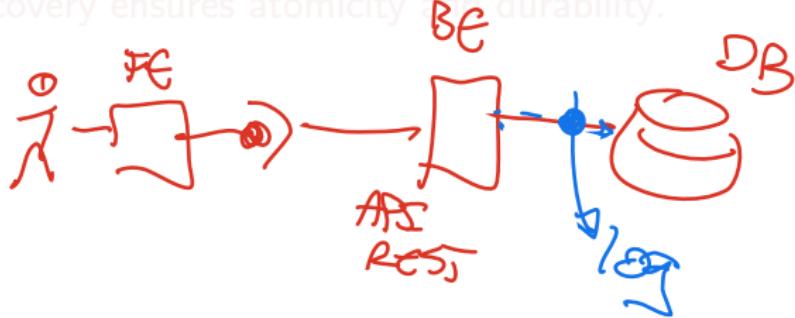
Failure Recovery Concepts

- **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~~ **Storage** periodically ~~save the database state to speed up recovery.~~ **+Format**
- ~~Shadowing~~ **ACID**: Maintains copies of data changes. **→ Open**, **→ log** tech
- 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.

Automatic

monthly
day → hour
10 minutes



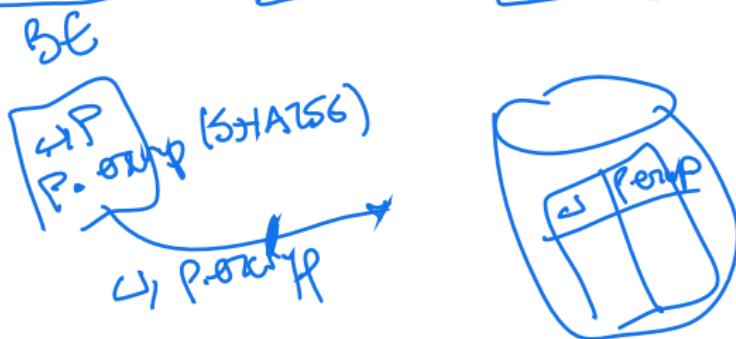
Write-Ahead Logging and Checkpoints

- **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

- 2 weeks
-
- 1 Database System Administration
 - 2 Record Storage
 - 3 DBMS Architecture
 - 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>

