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

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

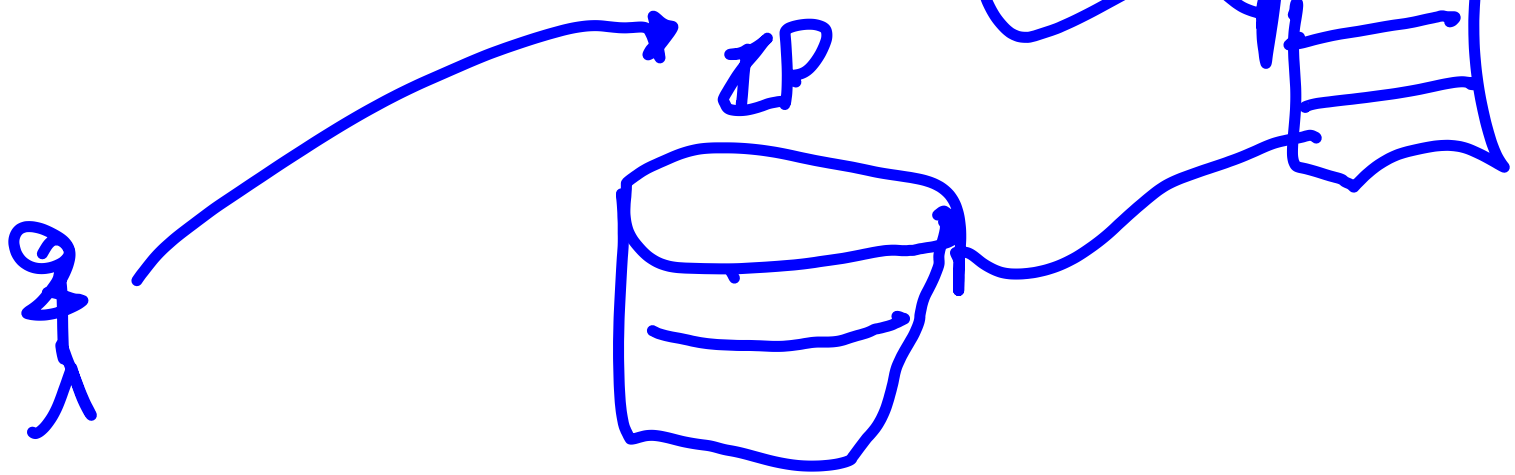
Outline

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

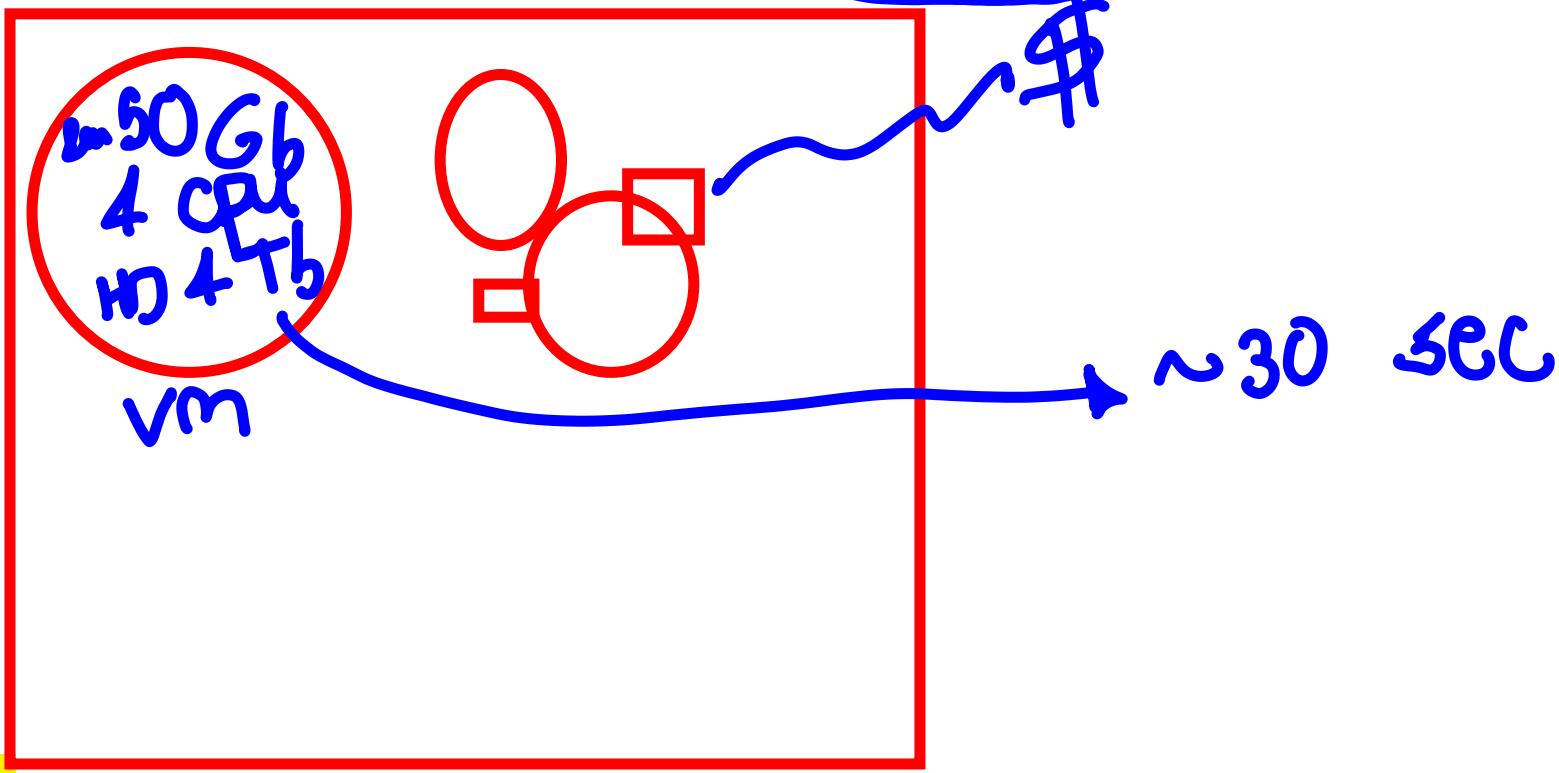
What is On-Premises Computing?

- **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.



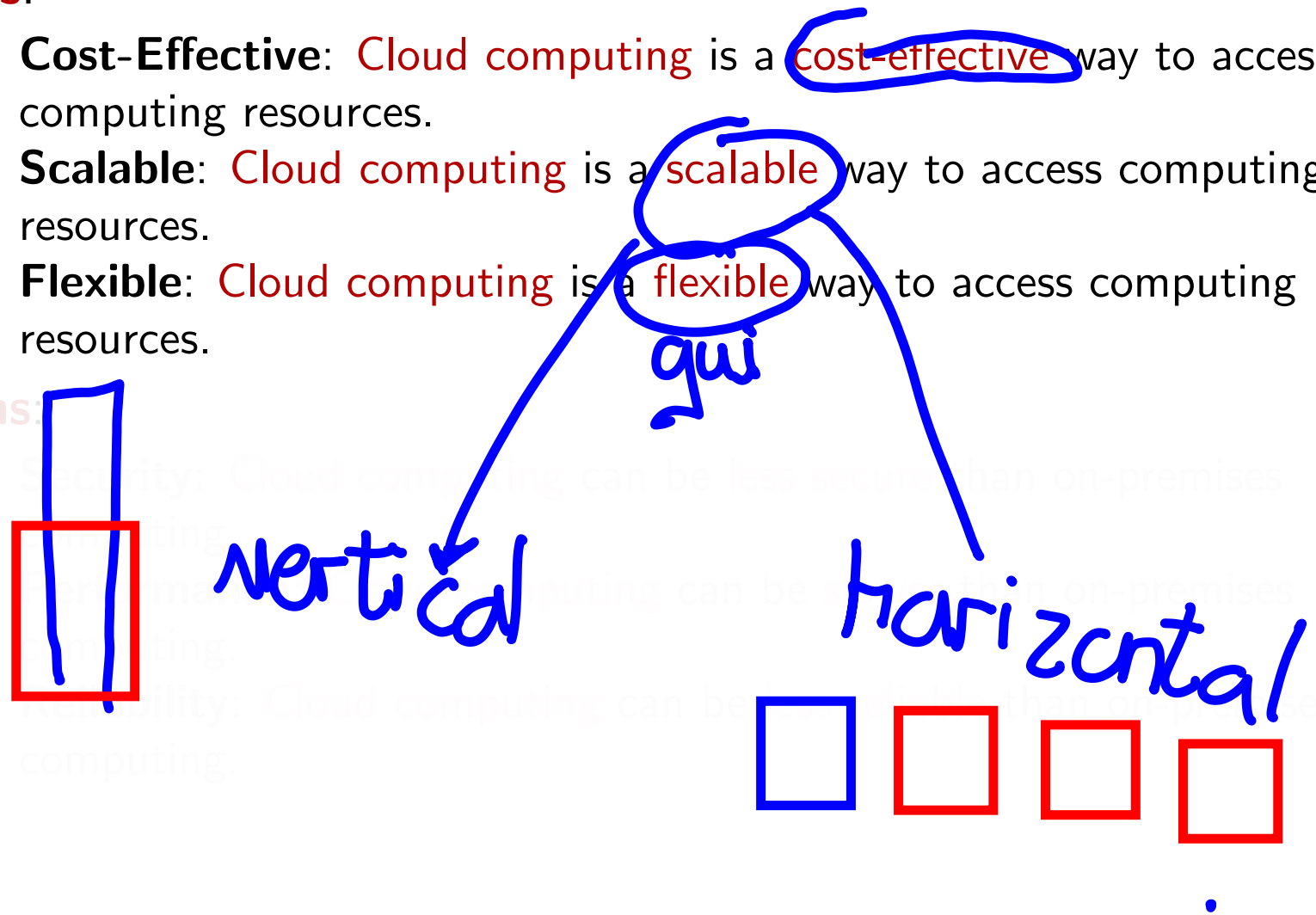
What is the Cloud Computing?

- **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.



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

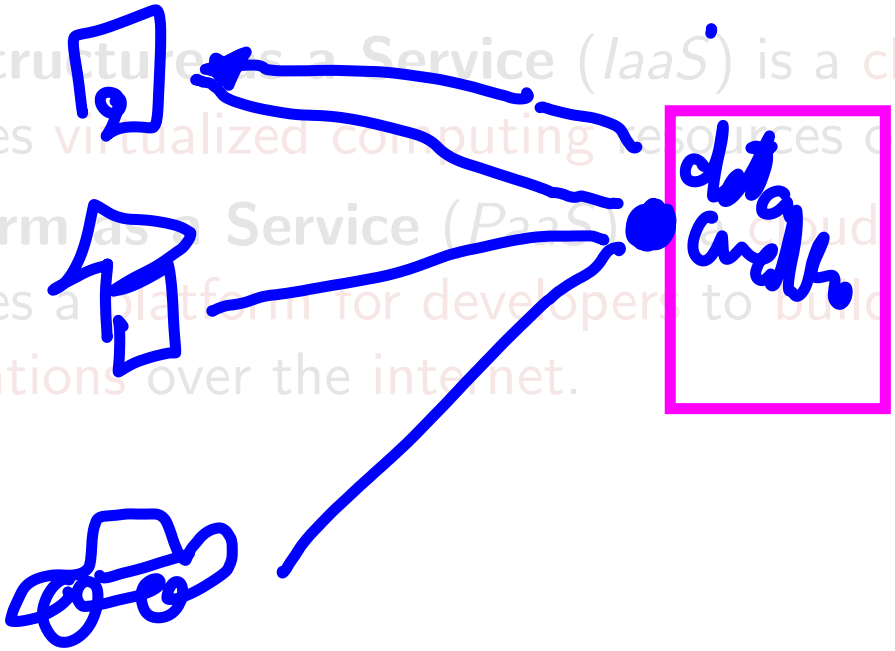
Pros & Cons of Cloud Computing

- **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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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

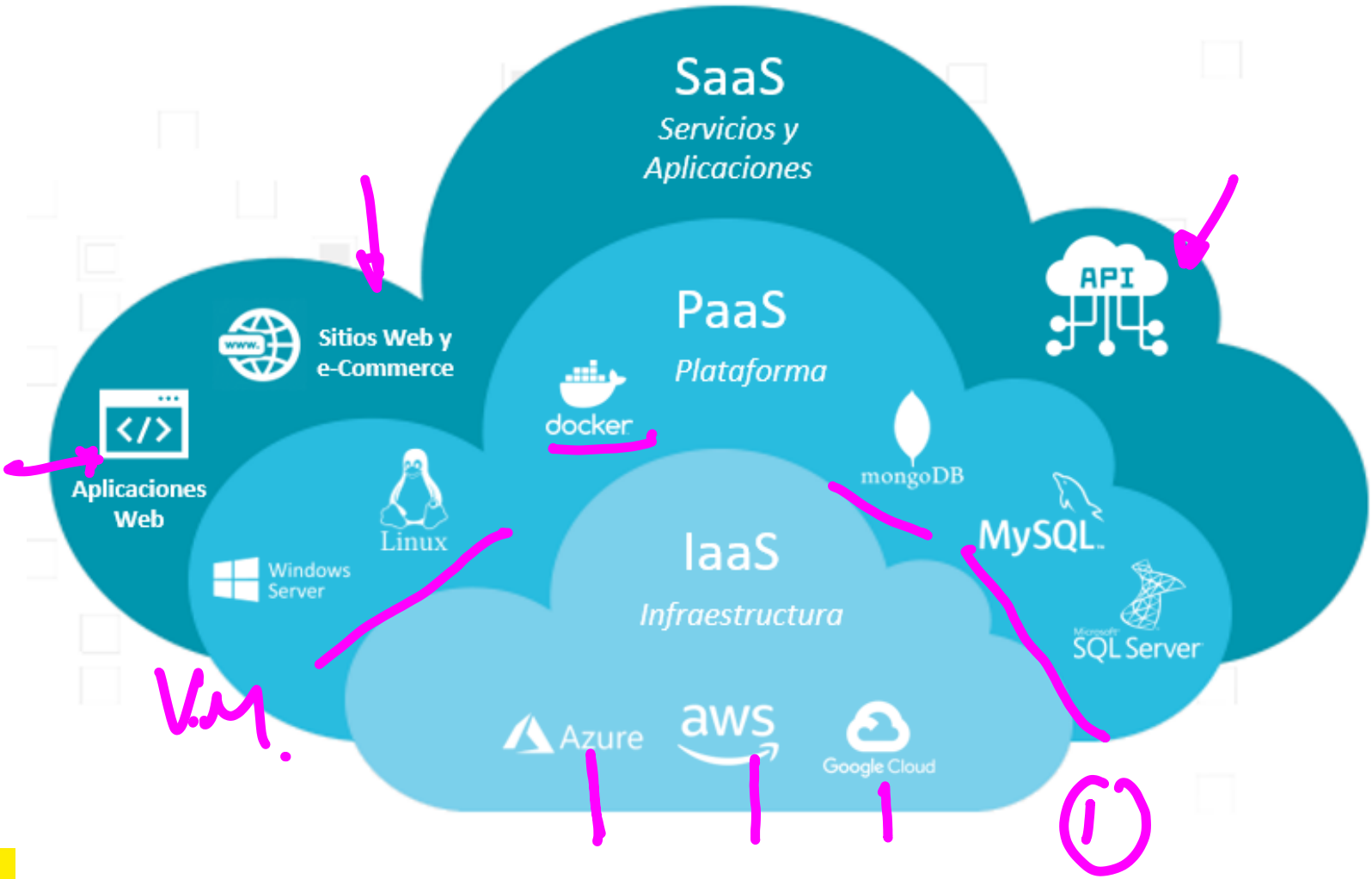
- **Software as a Service** (*SaaS*) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the internet.
- **Infrastructure as a Service** (*IaaS*) is a cloud computing model that provides virtualized computing resources over the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

IaC → terraform

SaaS Vs. IaaS Vs. PaaS

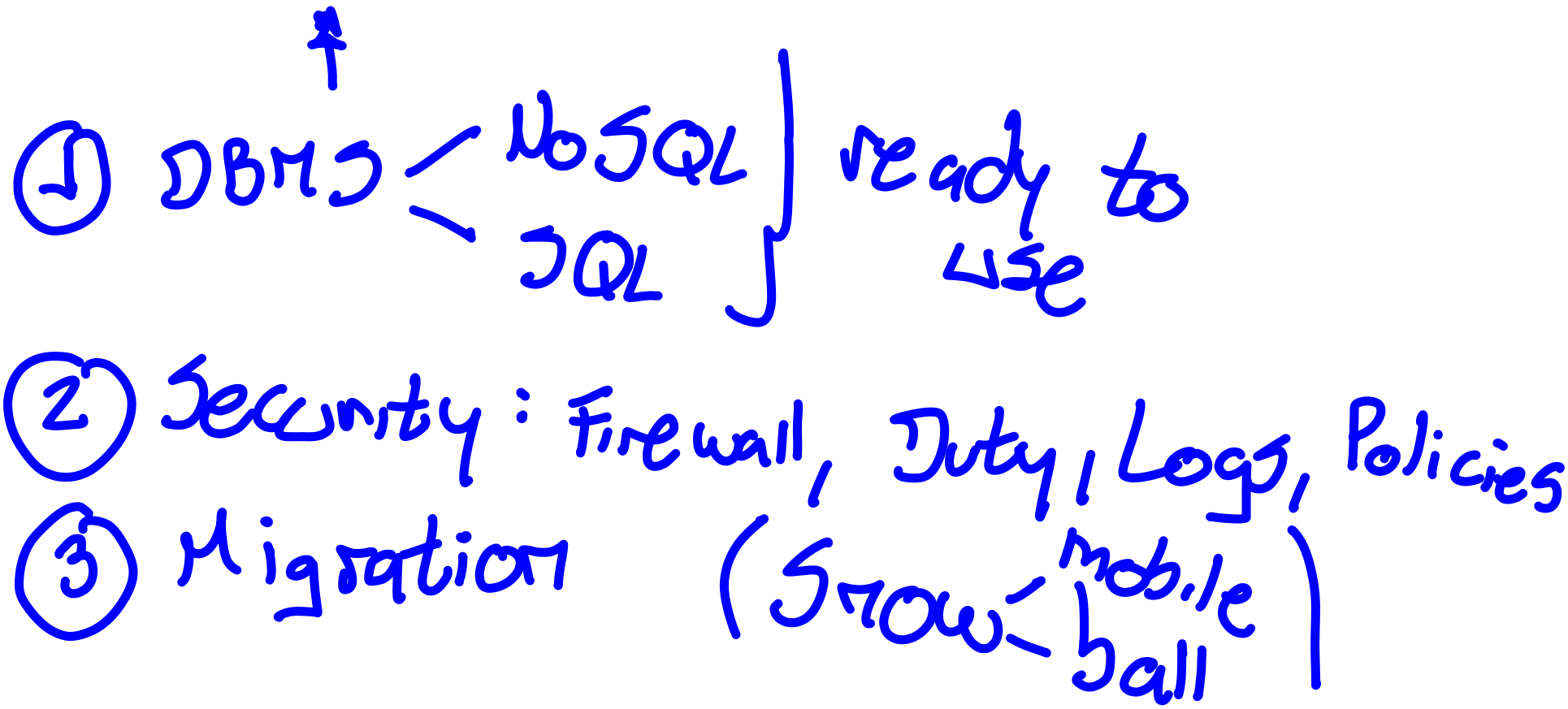
- **Software as a Service** (*SaaS*) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the internet.
- **Infrastructure as a Service** (*IaaS*) is a cloud computing model that provides virtualized computing resources over the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



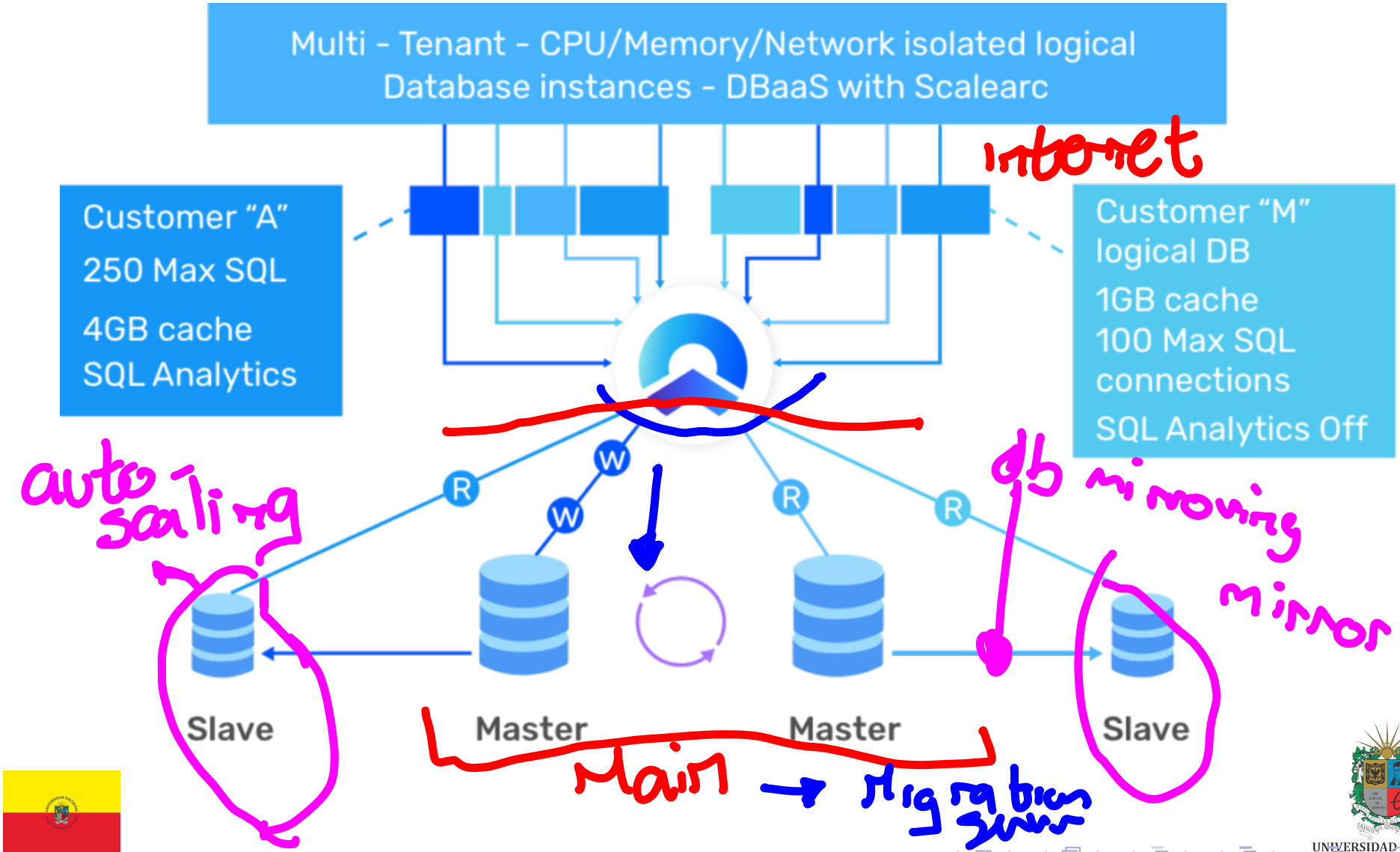
DataBases as a Service

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



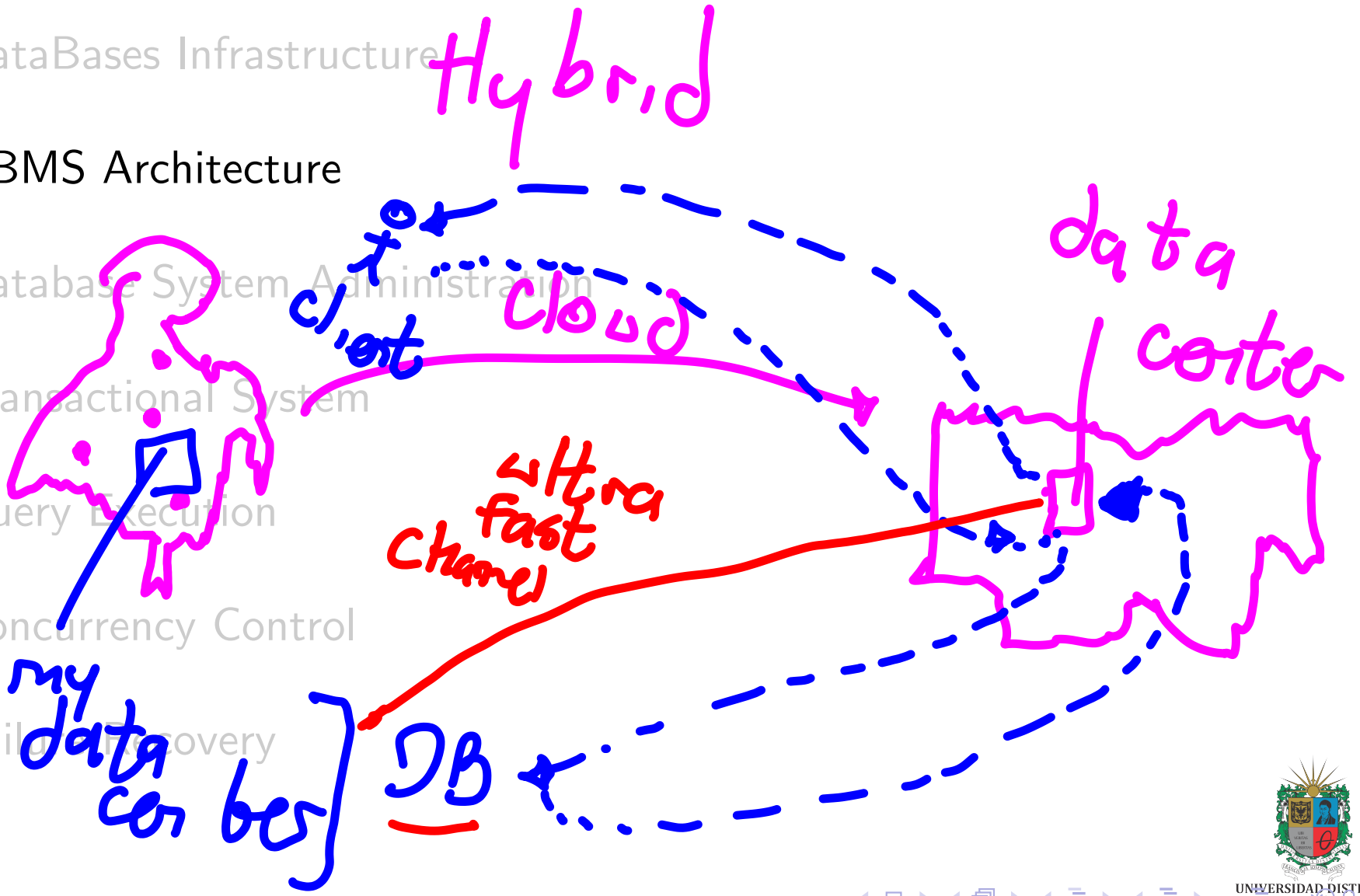
UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Case Study: DBaaS Custom for Clients



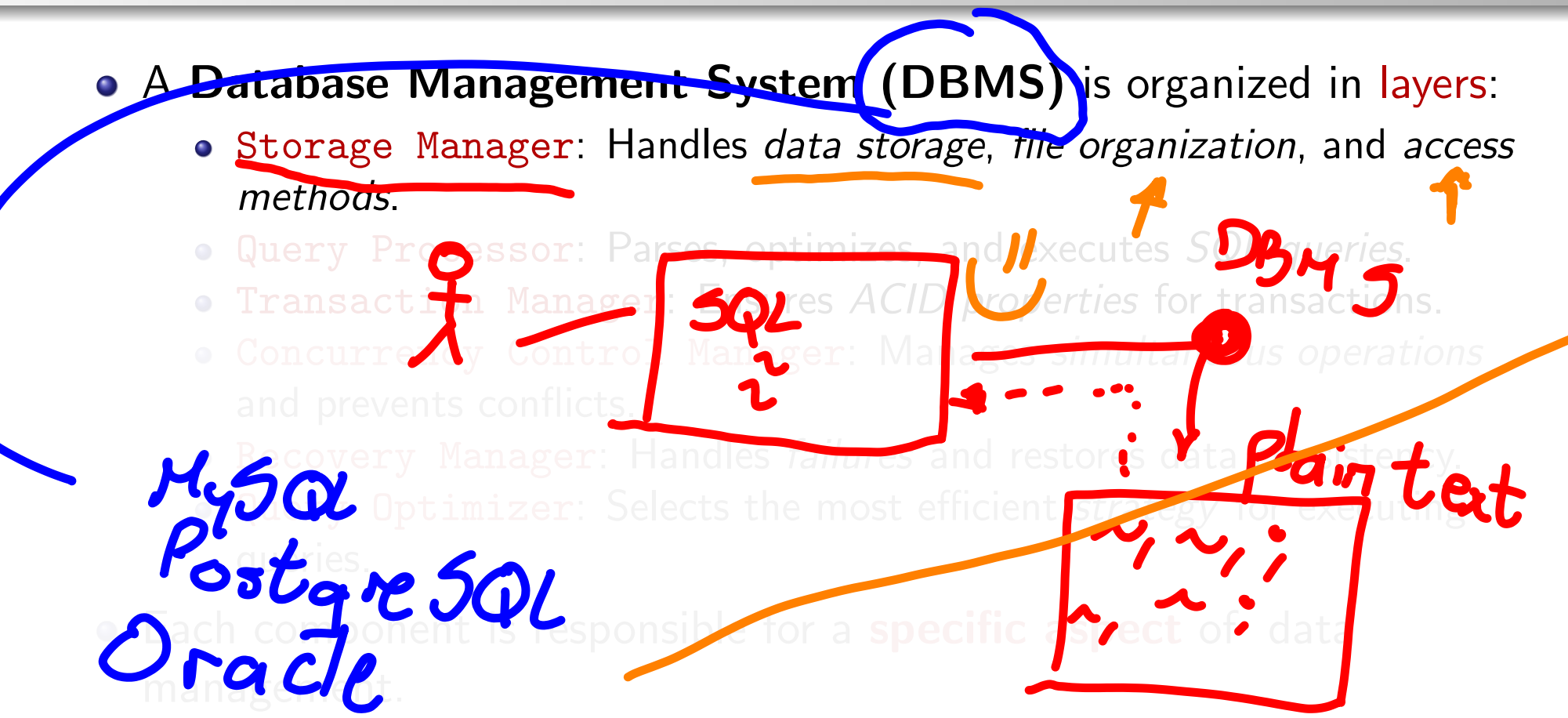
Outline

1. DataBases Infrastructure
2. DBMS Architecture
3. Database System Administration
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*.
  - `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.

graph

compiler

tree

language to interact with storage

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

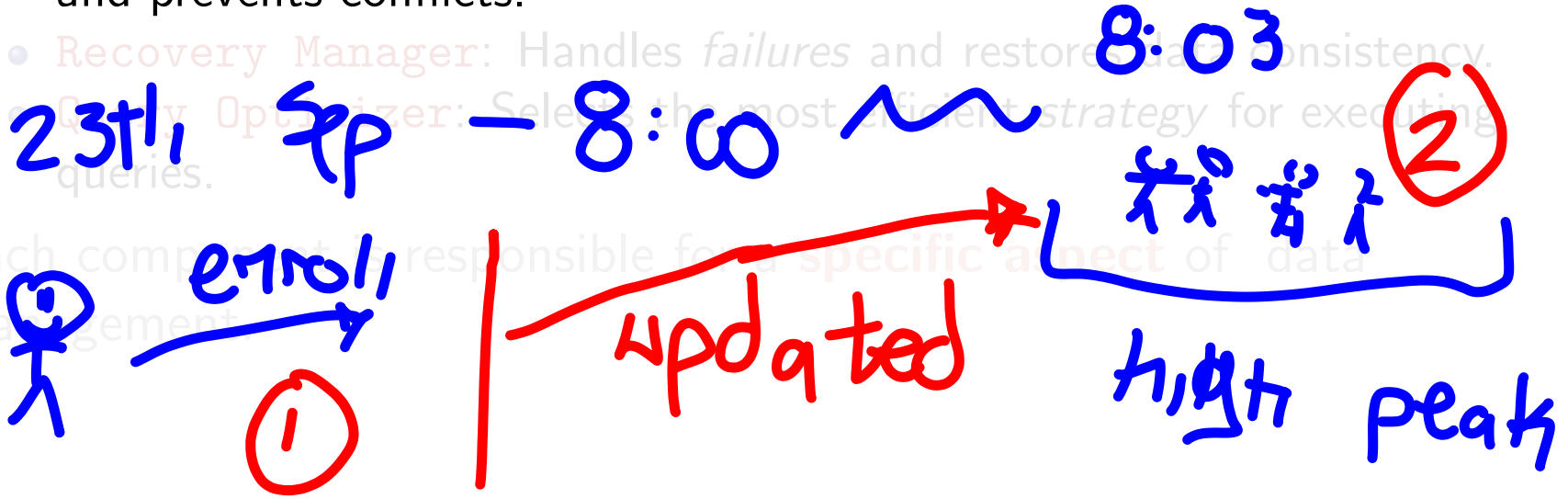
DBMS Architecture Overview

- 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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

DBMS Architecture Overview

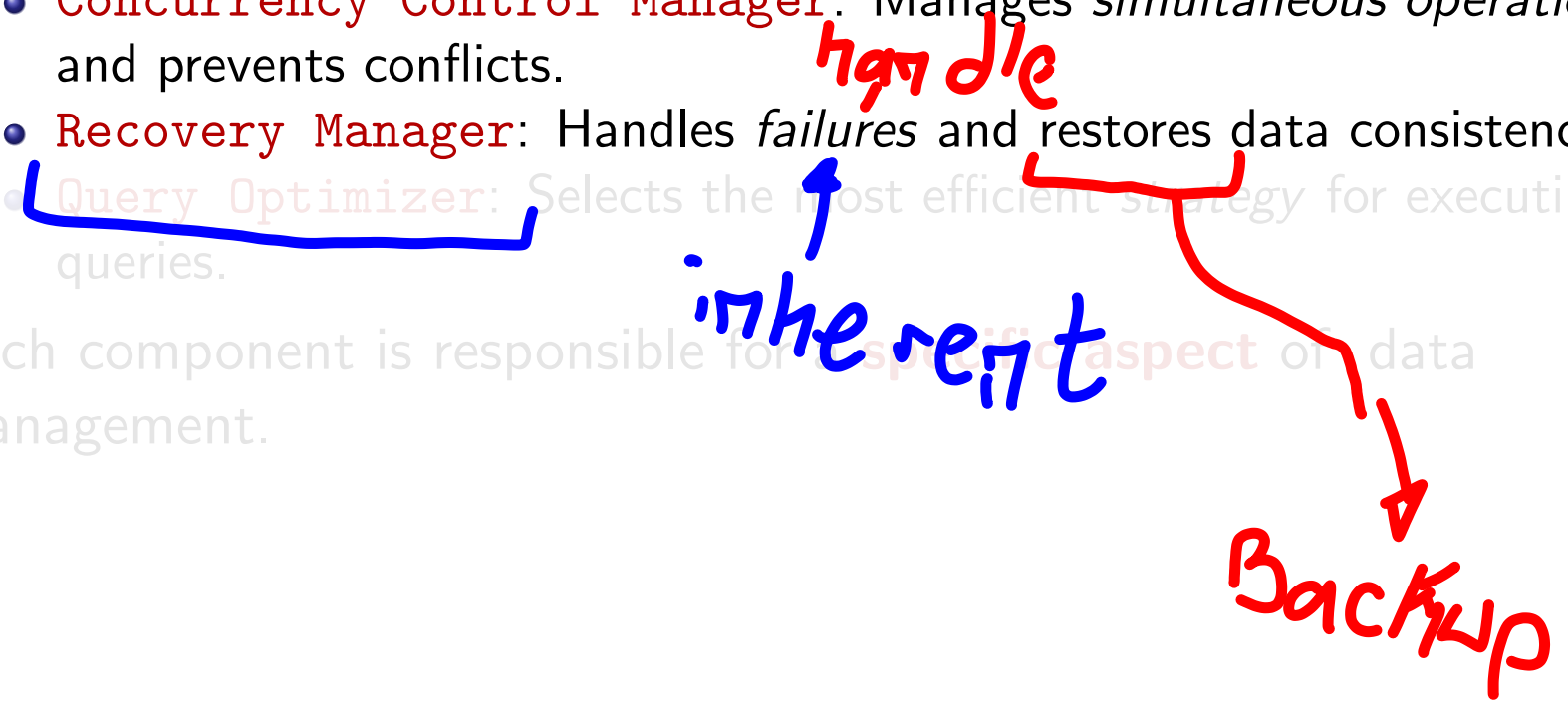
- A **Database Management System (DBMS)** is organized in layers:
  - `Storage Manager`: Handles *data storage*, *file organization*, and *access methods*.
  - `Query Processor`: Parses, optimizes, and executes *SQL queries*.
  - `Transaction Manager`: Ensures *ACID properties* for transactions.
  -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



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

DBMS Architecture Overview

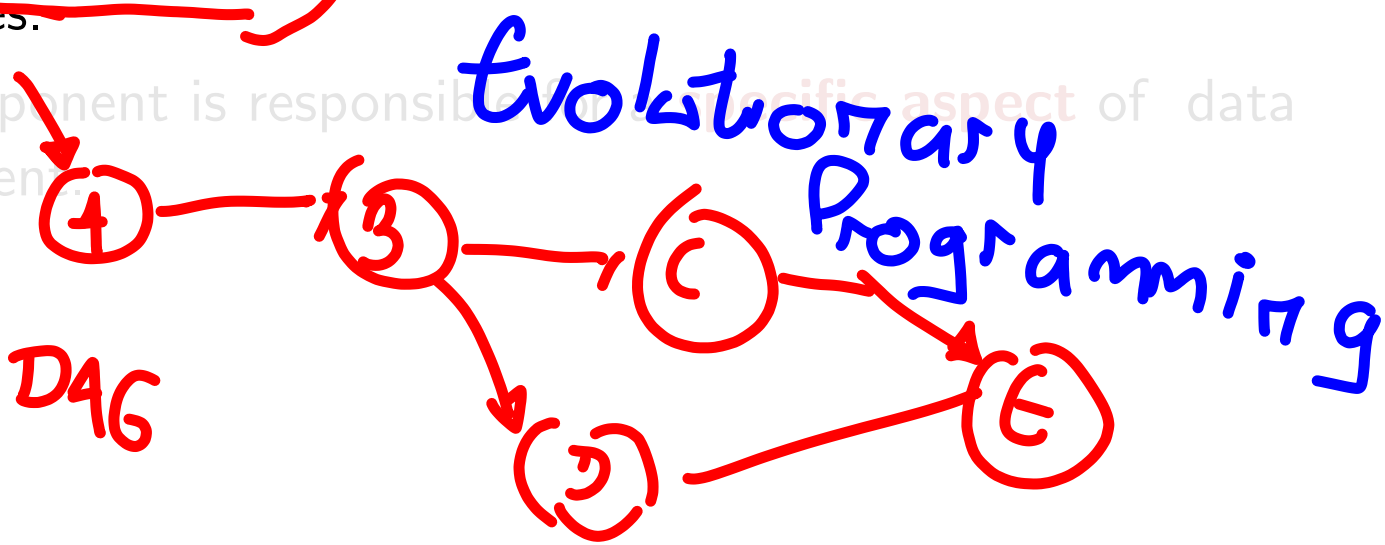
- A **Database Management System (DBMS)** is organized in layers:
  - `Storage Manager`: Handles *data storage*, *file organization*, and *access methods*.
  - `Query Processor`: Parses, optimizes, and executes *SQL queries*.
  - `Transaction Manager`: Ensures *ACID properties* for transactions.
  -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



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.



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

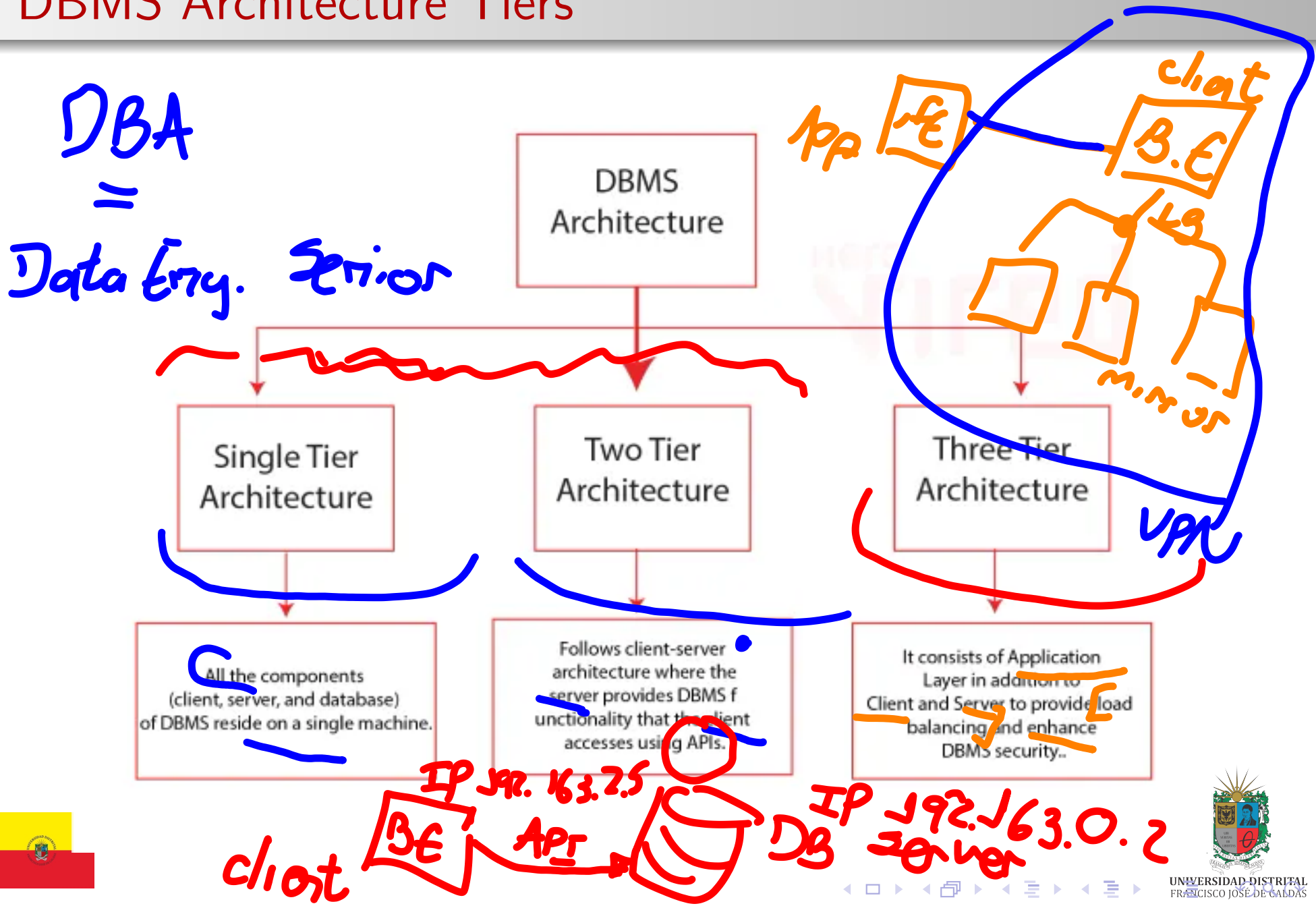
DBMS Architecture Overview

- 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.

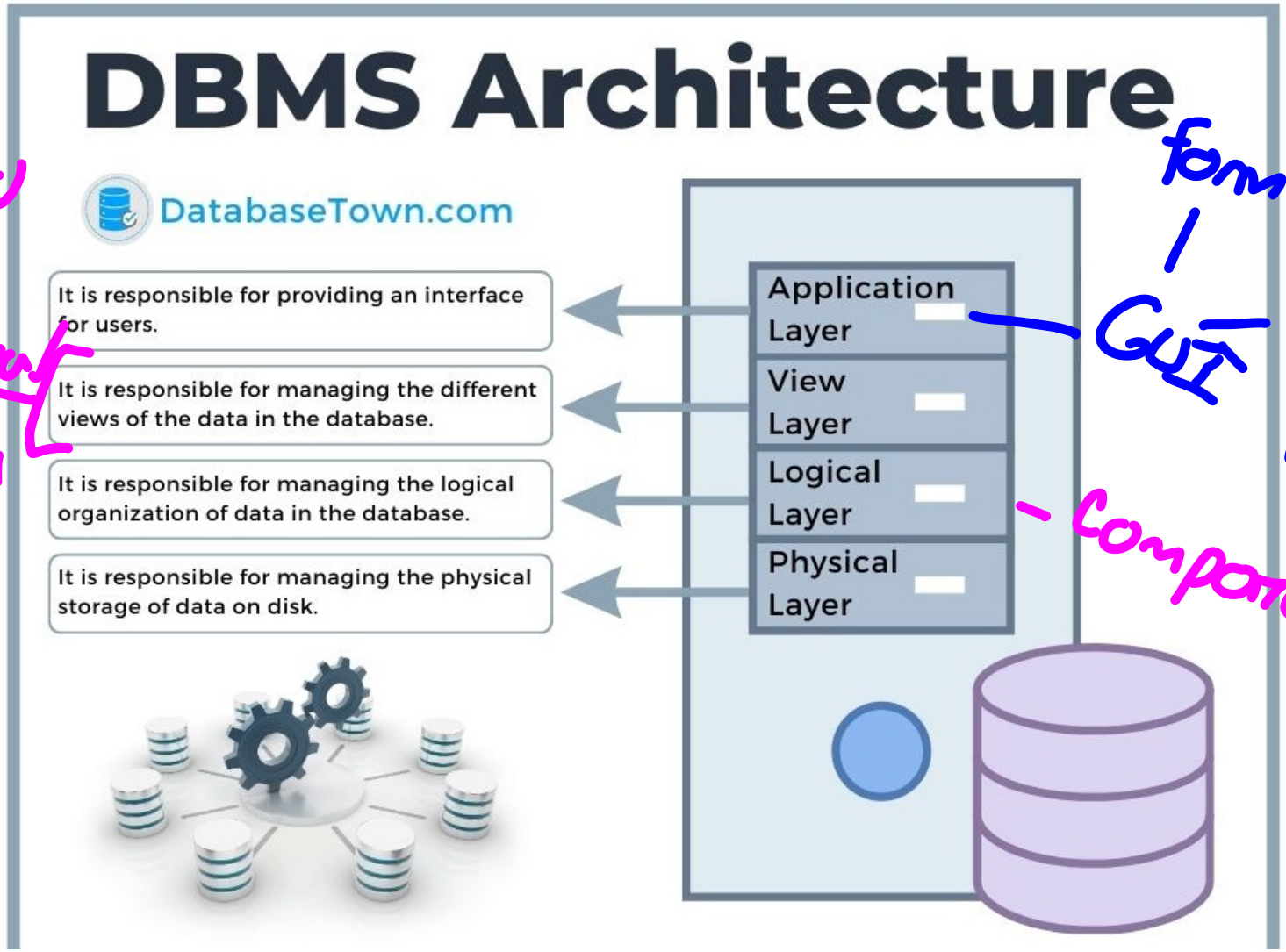
all

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

DBMS Architecture Tiers



DBMS Architecture N-Tier



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**: Combine 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.

1-tier

developers

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Types of DBMS Architecture

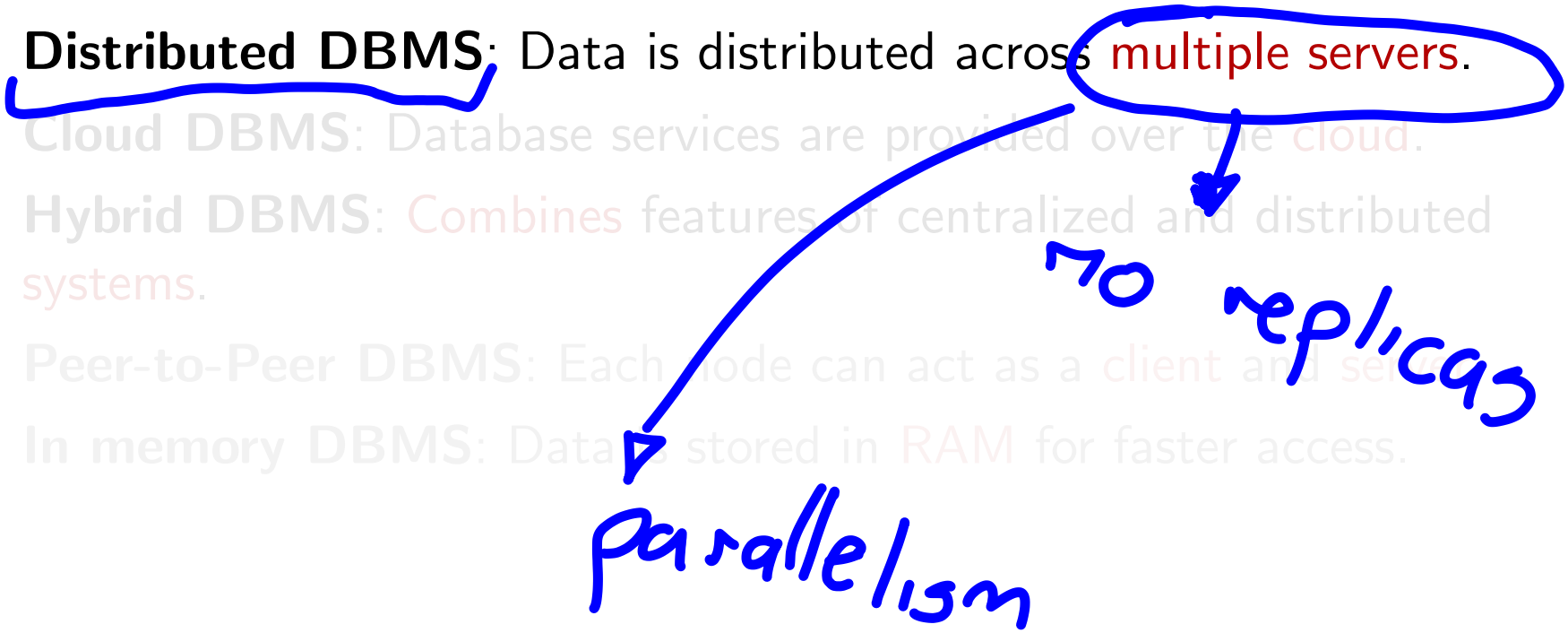
There are several types of DBMS architectures:

- **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.



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

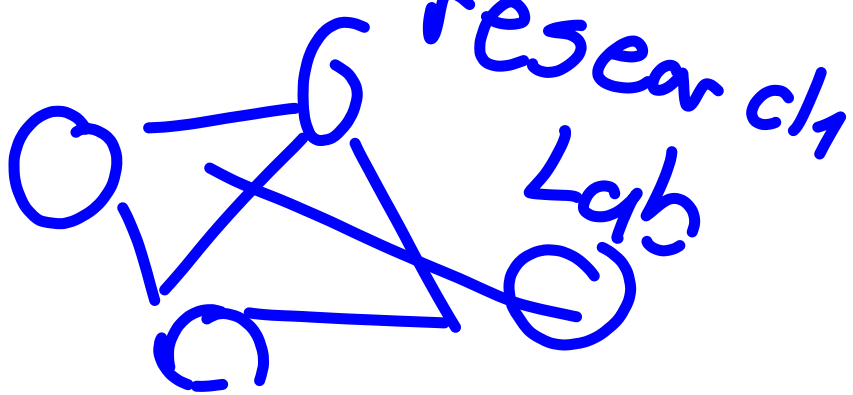
There are several types of DBMS architectures:

- **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.



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.

DB- 100-200ms

Cache - 5-10 ms

Cache

lifetime (sec-min)

Max 30 hours

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

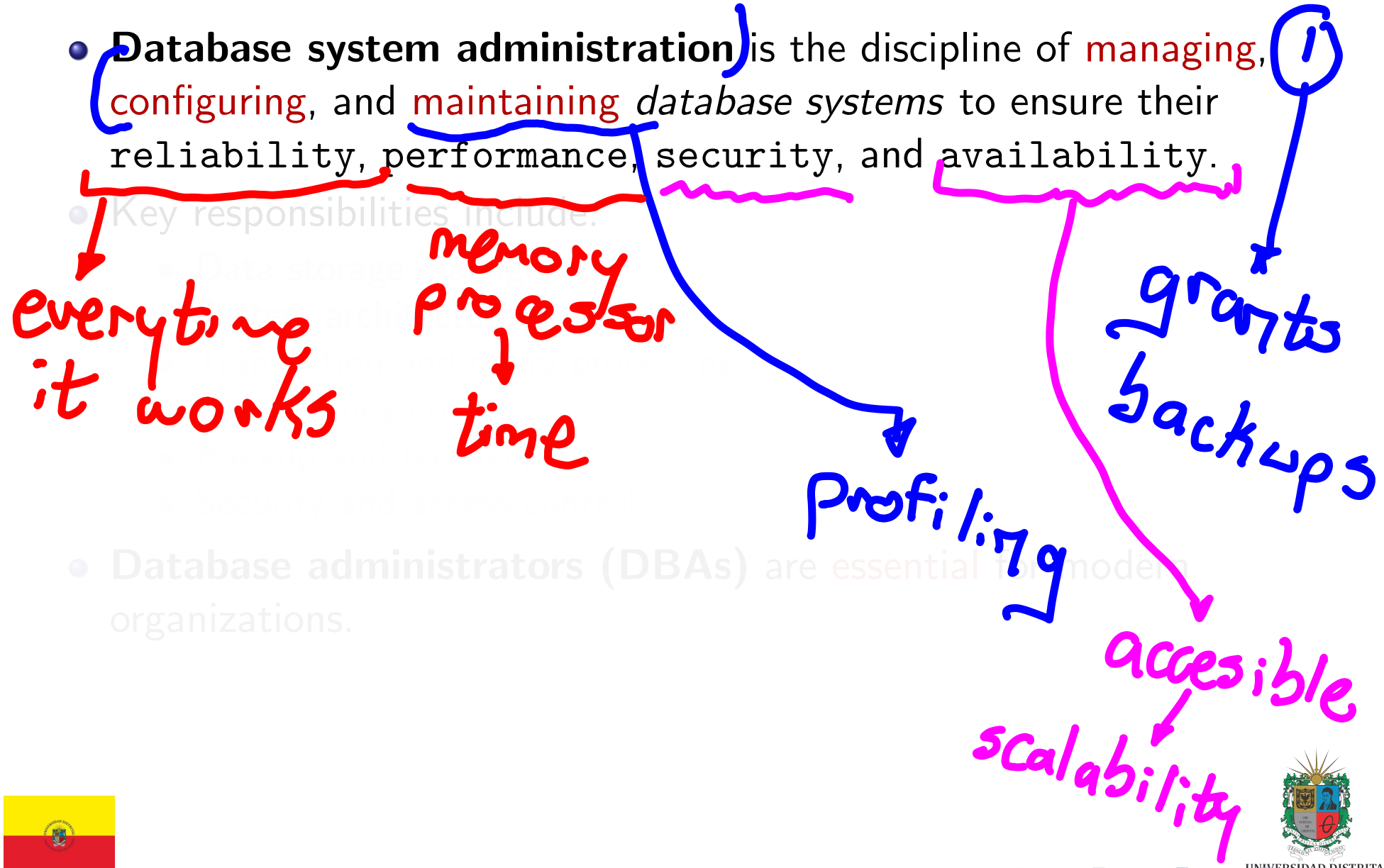
Outline

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

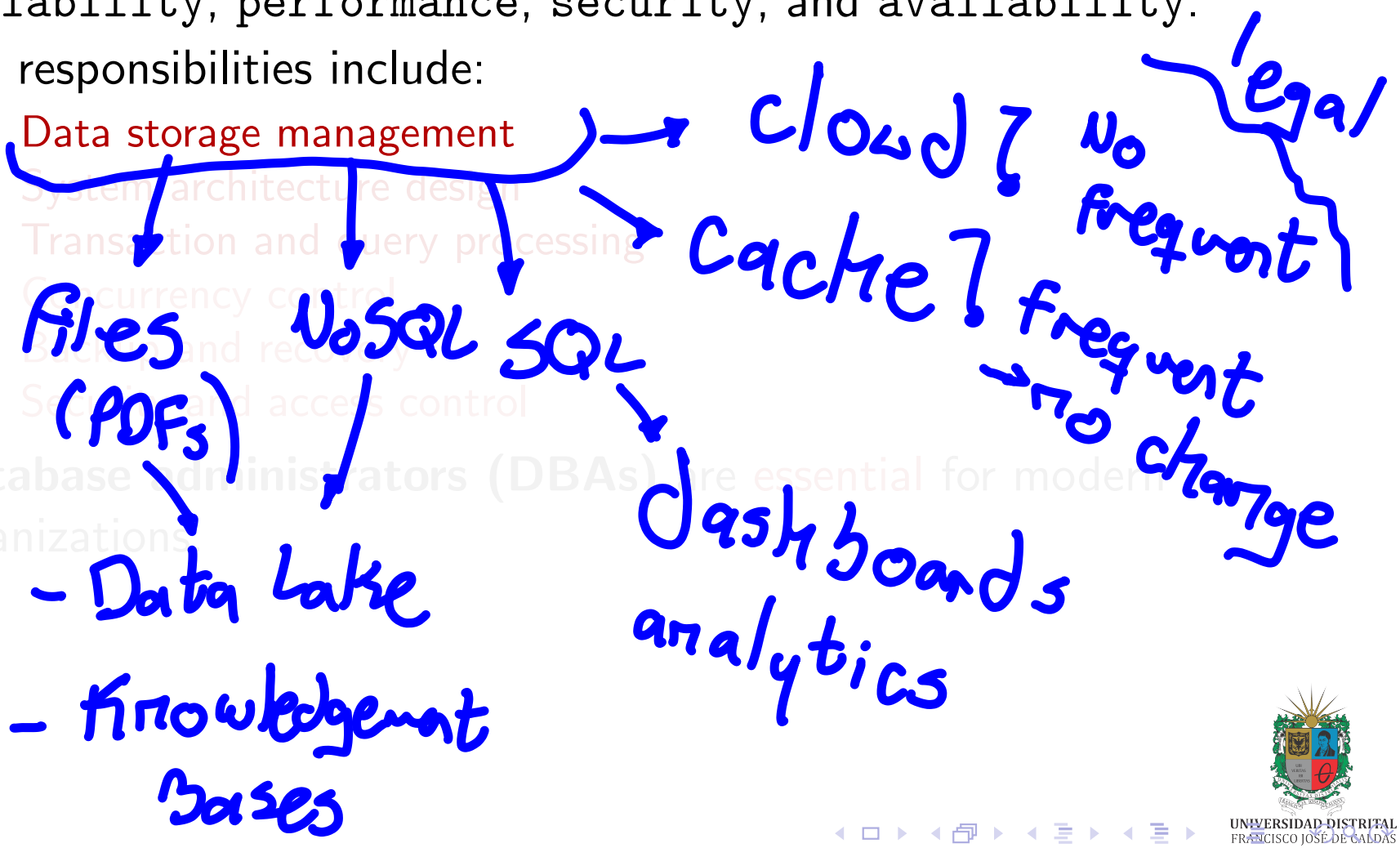
Database System Administration

- **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
  - Backup and recovery planning
  - Performance tuning
  - Security enforcement
- **Database administrators (DBAs)** are essential in modern organizations.



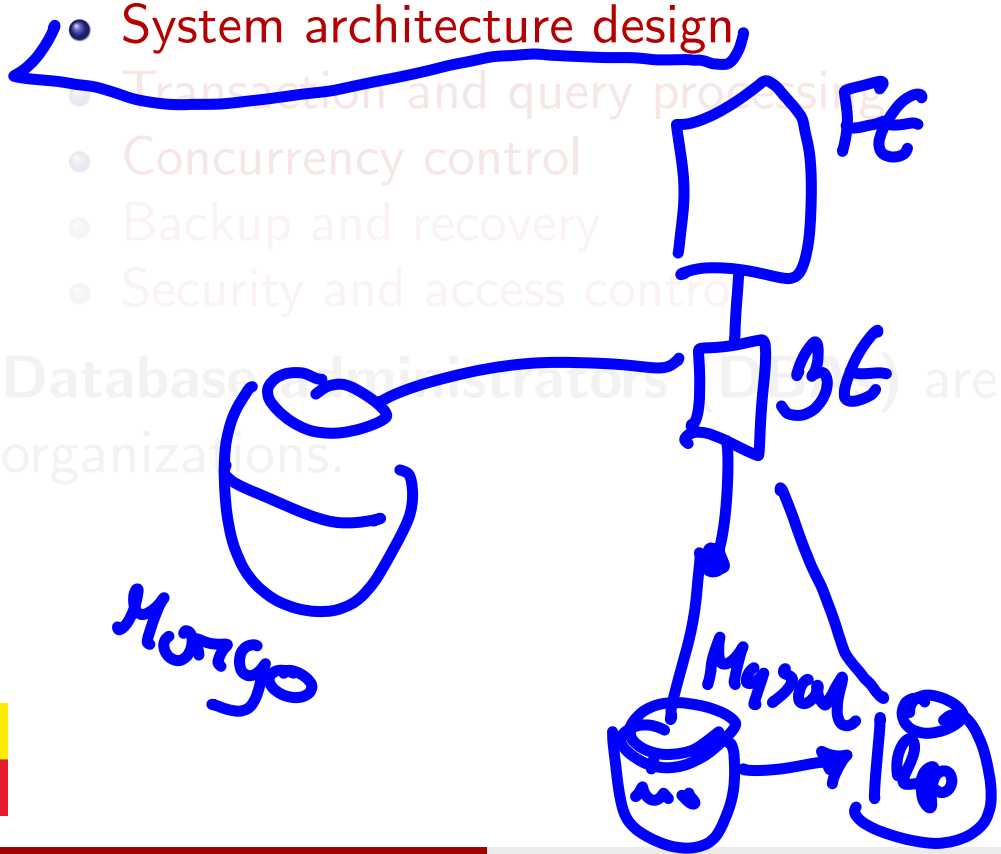
Database System Administration

- **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.

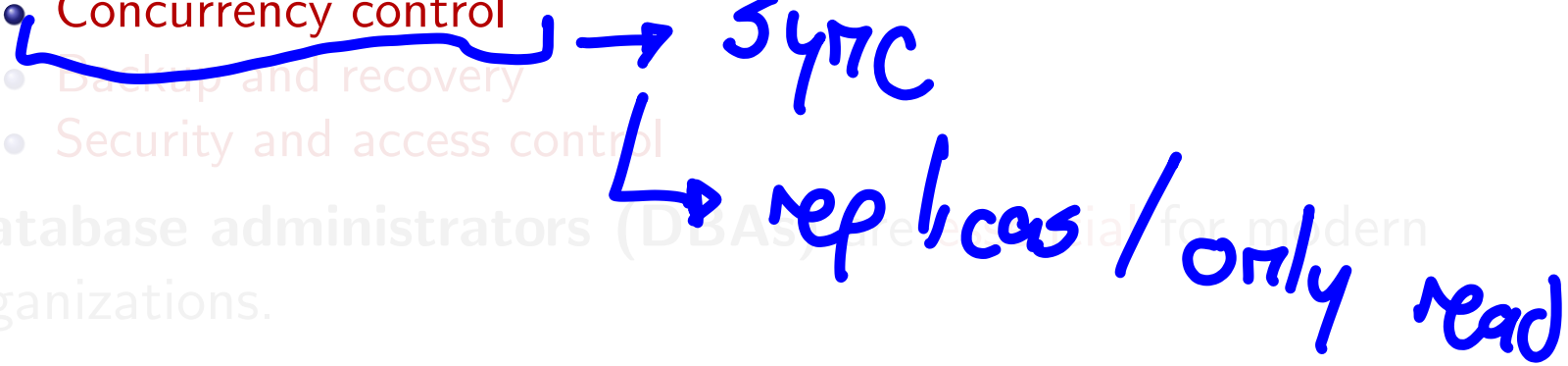
→ Indexes

→ Improve DB model

→ Internal Optimization

Database System Administration

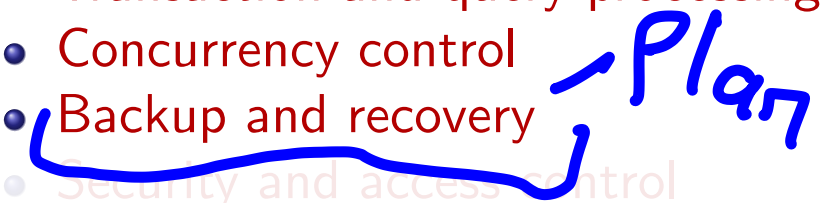
- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their `reliability`, `performance`, `security`, and `availability`.
- Key responsibilities include:
  - Data storage management
  - System architecture design
  - Transaction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

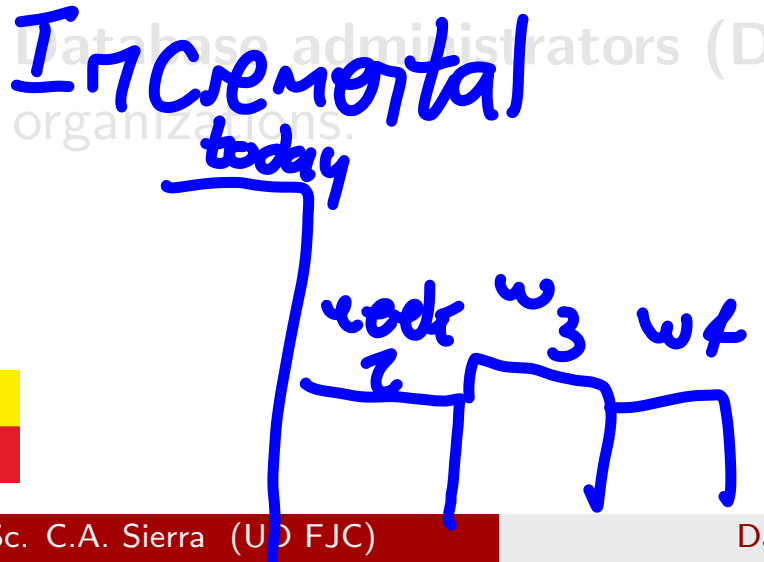


UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Database System Administration

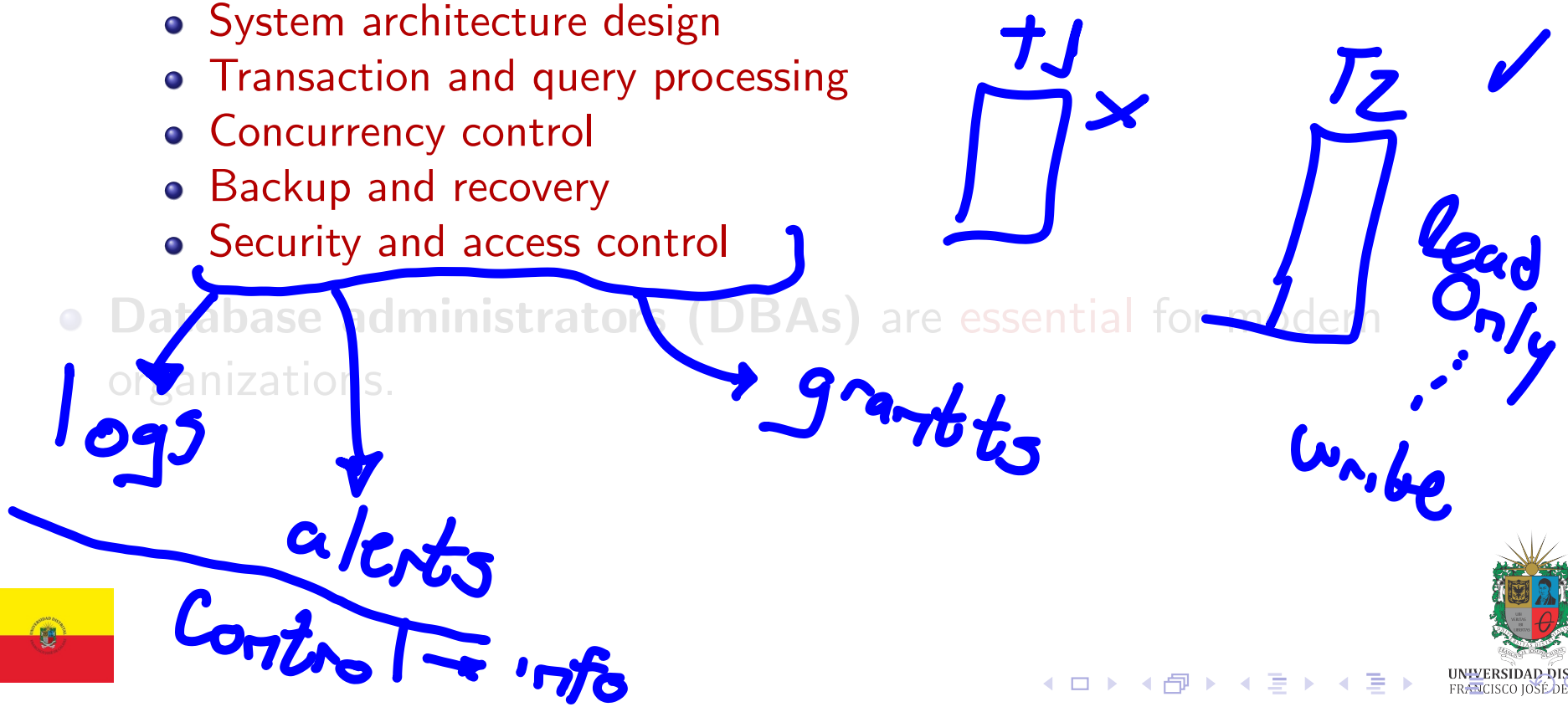
- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their `reliability`, `performance`, `security`, and `availability`.
- Key responsibilities include:
  - Data storage management
  - System architecture design
  - Transaction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





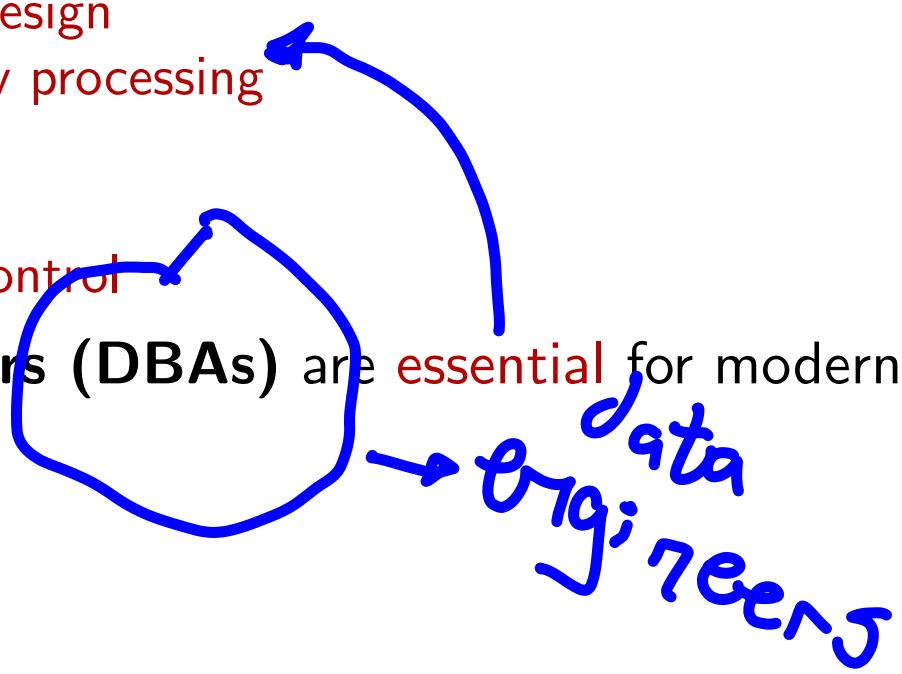
Database System Administration

- **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

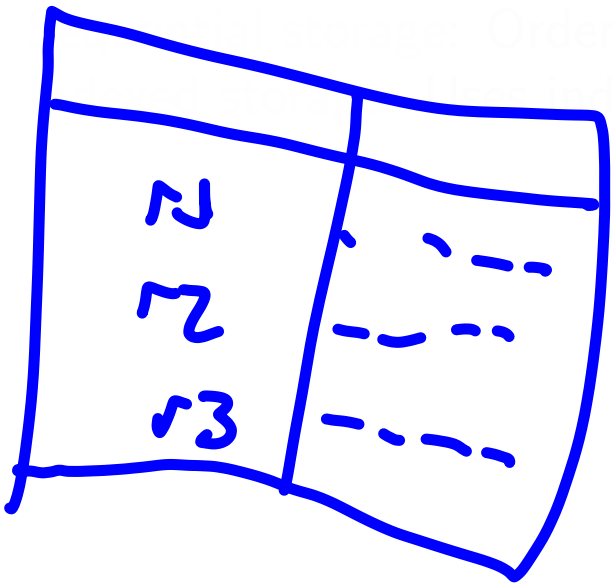
- **Database system administration** is the discipline of managing, configuring, and maintaining *database systems* to ensure their `reliability`, `performance`, `security`, and `availability`.
- Key responsibilities include:
  - Data storage management
  - System architecture design
  - Transaction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



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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.
  - Sorted storage: Ordered, fast range queries.
  - Hash storage: Use 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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

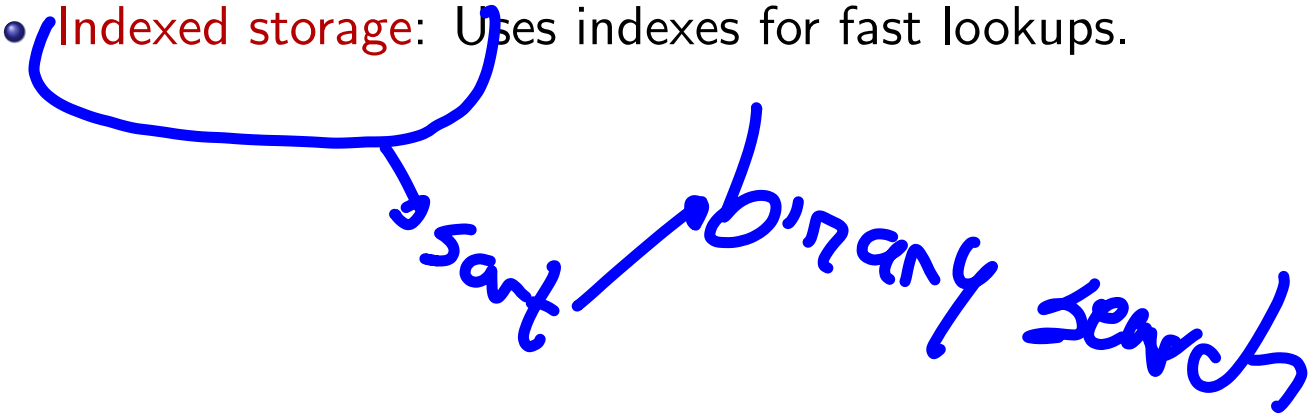
Record Storage Concepts

- 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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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.



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

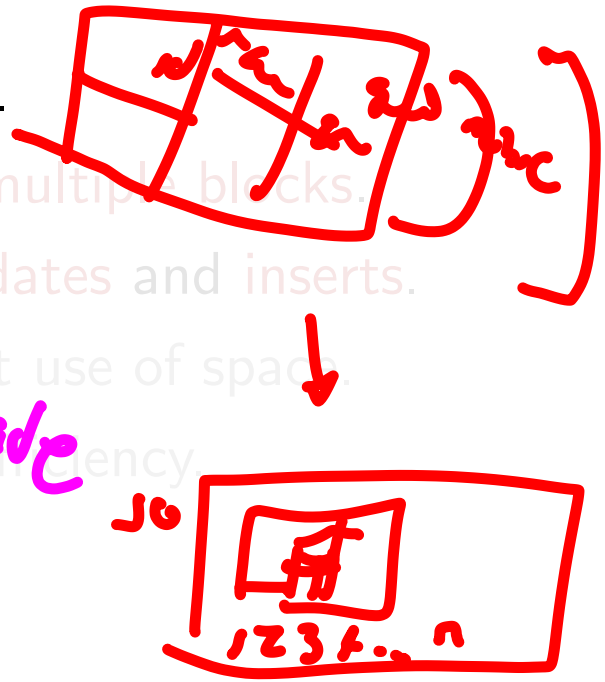
Block and Page Organization

- 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.

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.
- Segmentation management ensures efficient use of space.
- Data compression can also improve storage efficiency.




```
SELECT s.name, c.name as career, s.grade
  FROM student AS s
  JOIN career AS c
    ON s.career_fk = c.id
ORDER BY s.grade DESC
OFFSET (n-1)* 10
 LIMIT 10;
```

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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Block and Page Organization

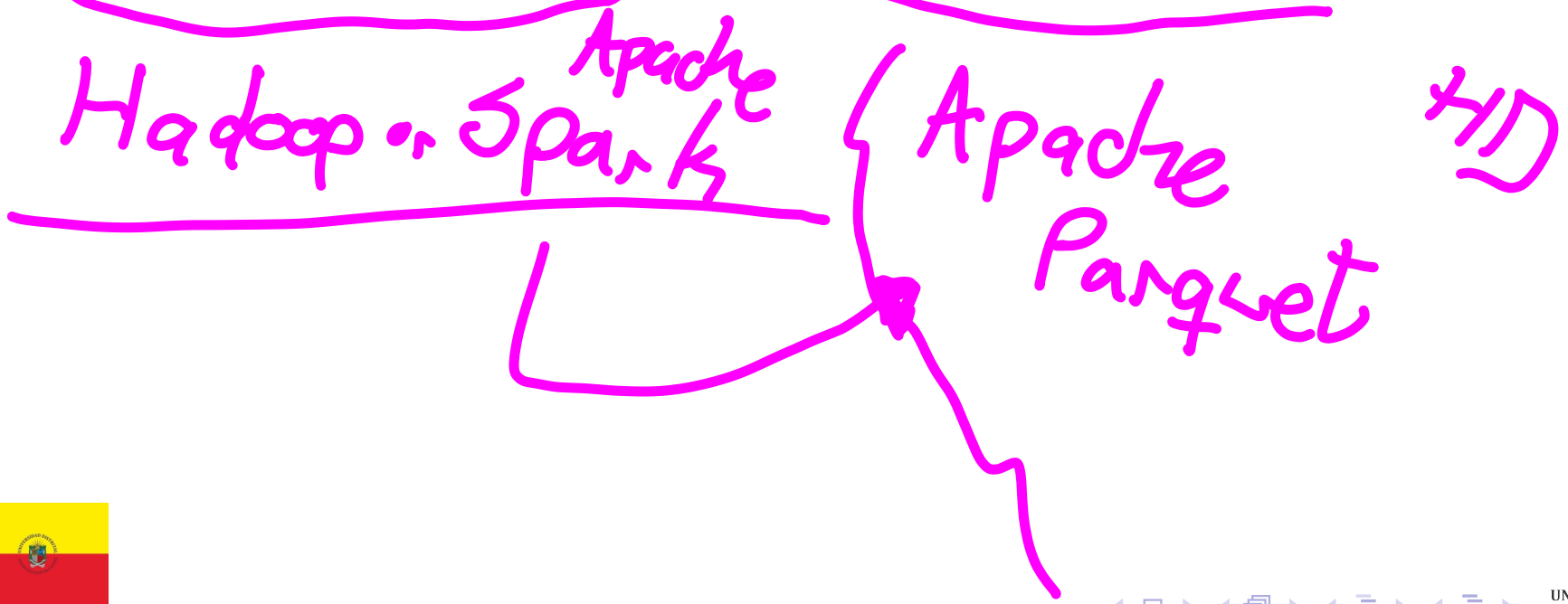
- Data is stored in **blocks** (pages) on disk.
- Block size and layout affect **I/O performance**.
- Records may be packed, slotted, or may span multiple blocks.
- **Free space management** is important for updates and inserts.
- **Fragmentation management** ensures efficient use of space.
- **Data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

Distributed Data Systems

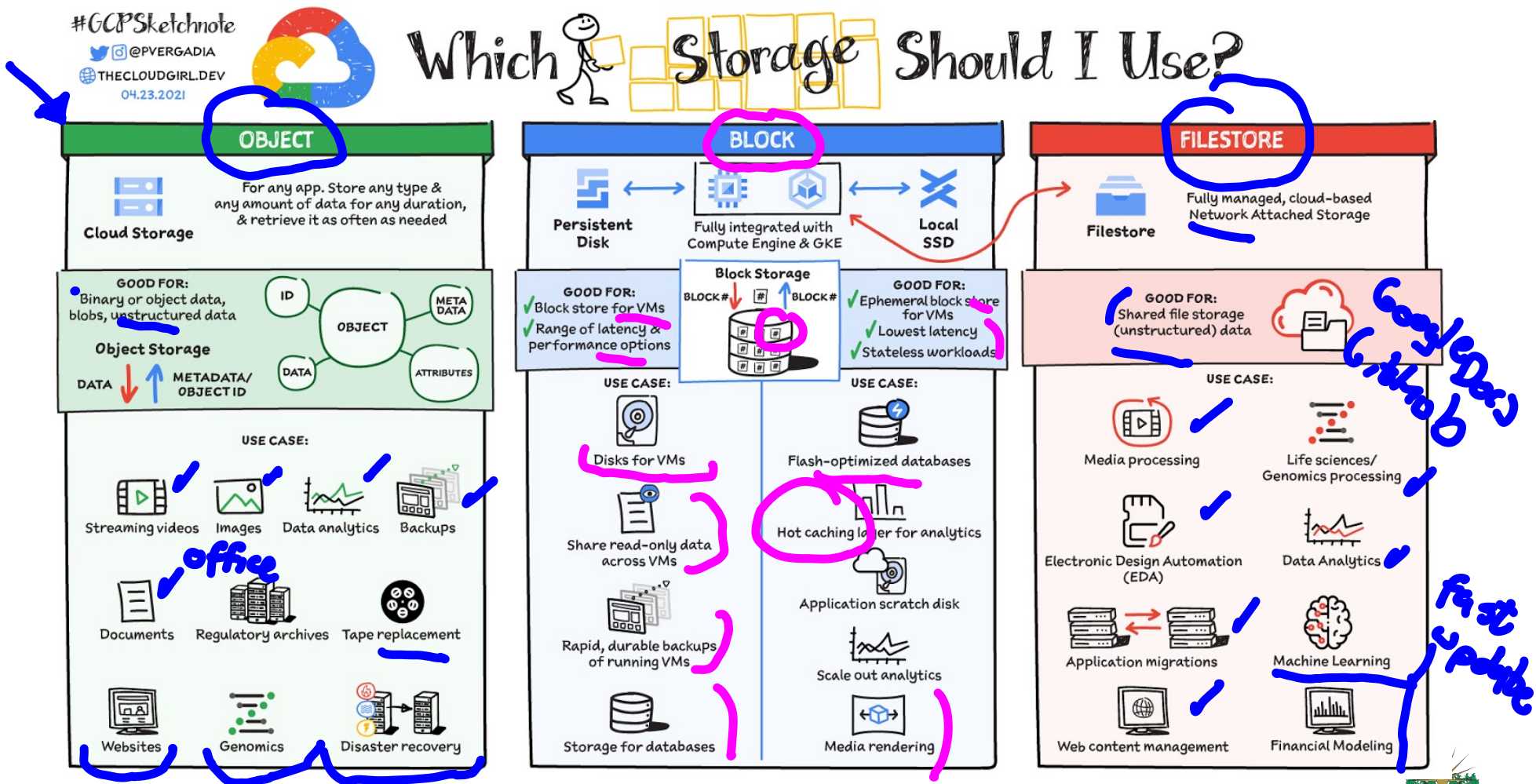
UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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.



Record Storage: Use Cases



Outline

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

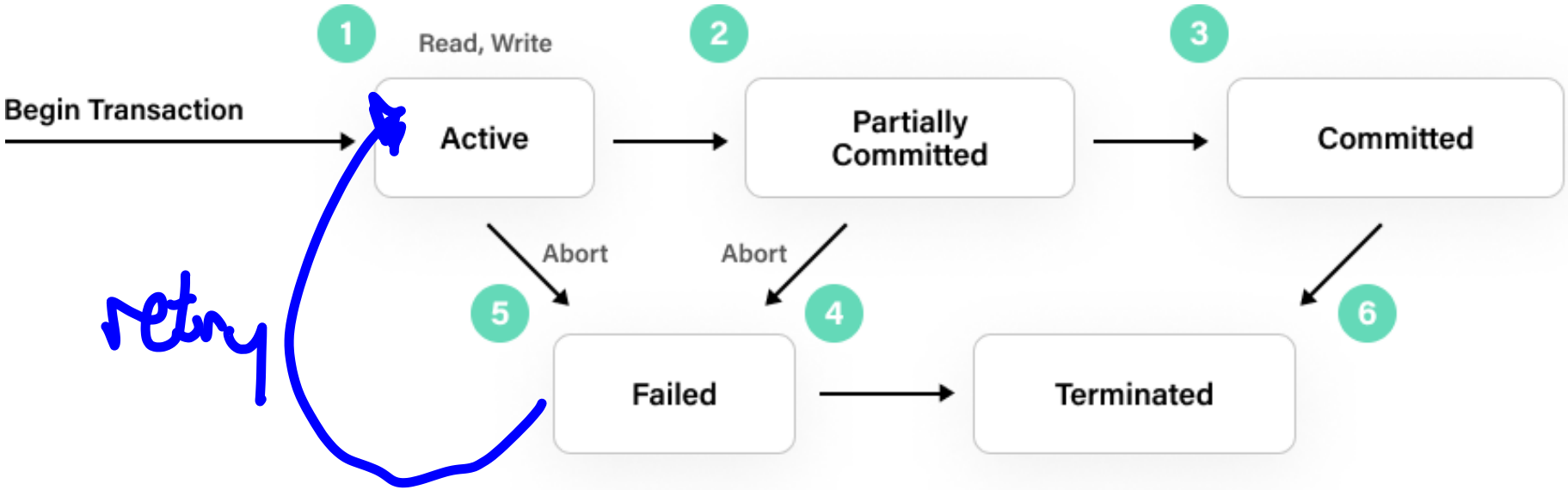
UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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

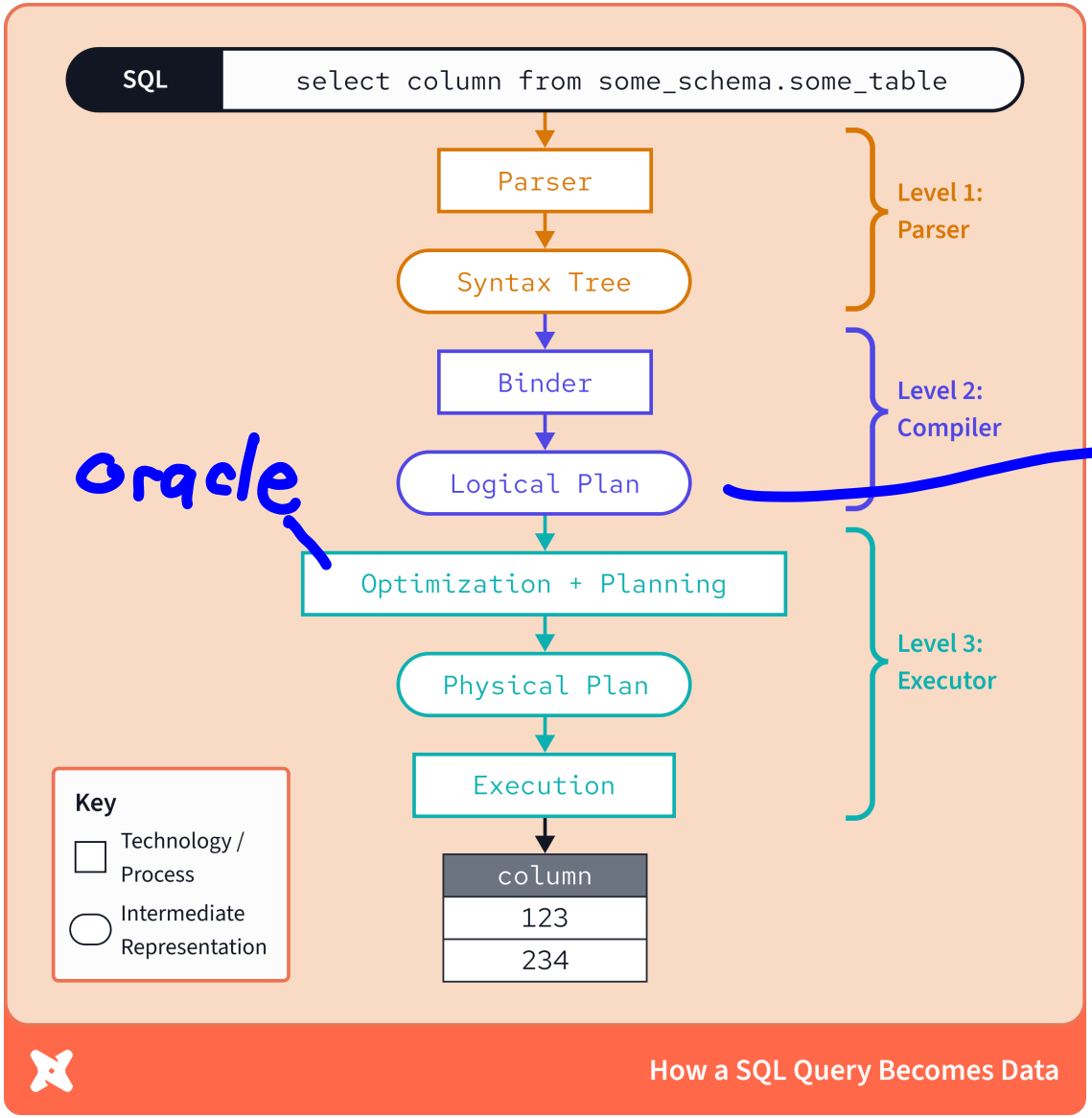
1. DataBases Infrastructure
2. DBMS Architecture
3. Database System Administration
4. Transactional System
5. Query Execution
6. Concurrency Control
7. Failure Recovery

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.

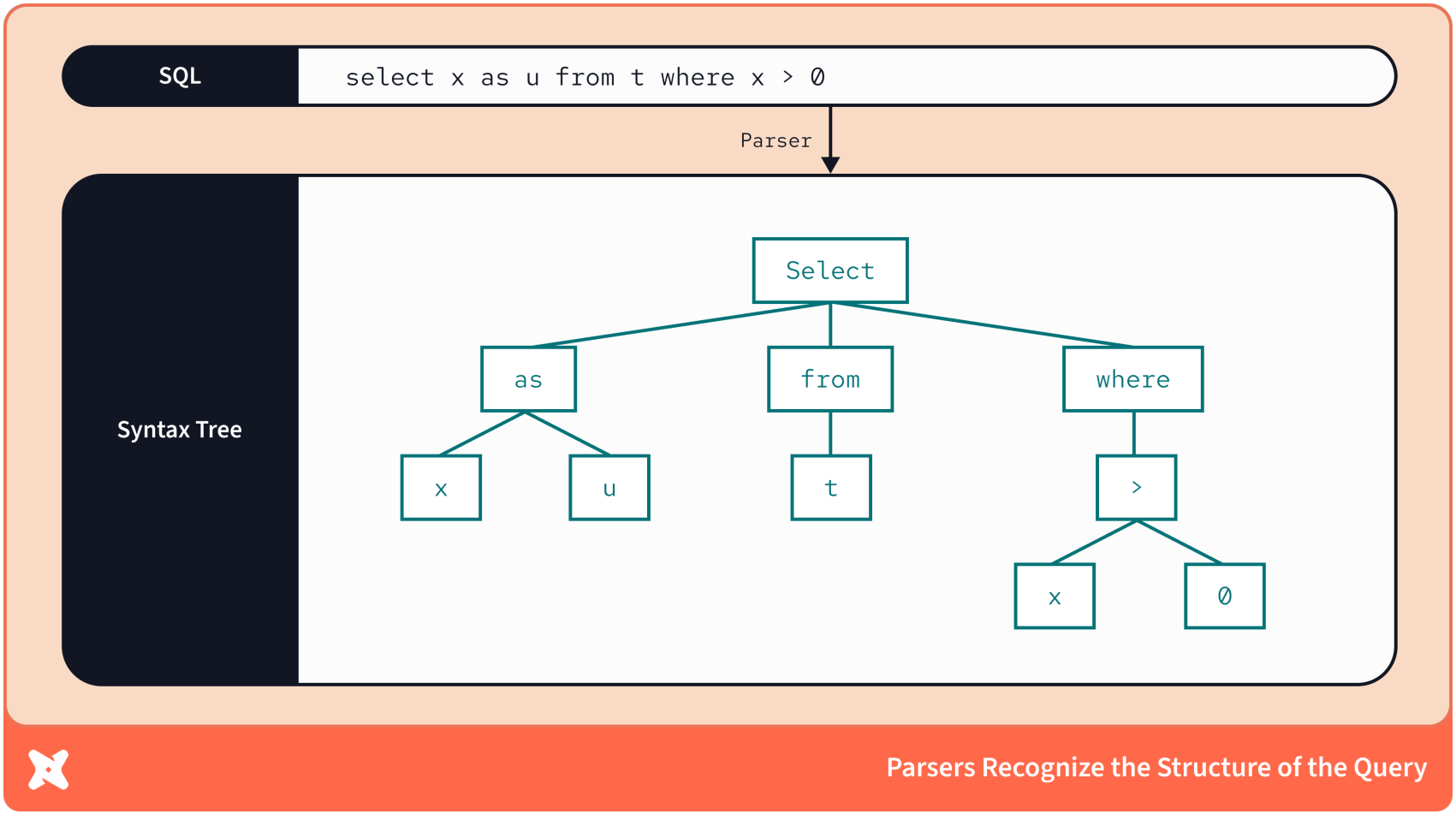
Query Execution Flow: Full Transaction



oracle

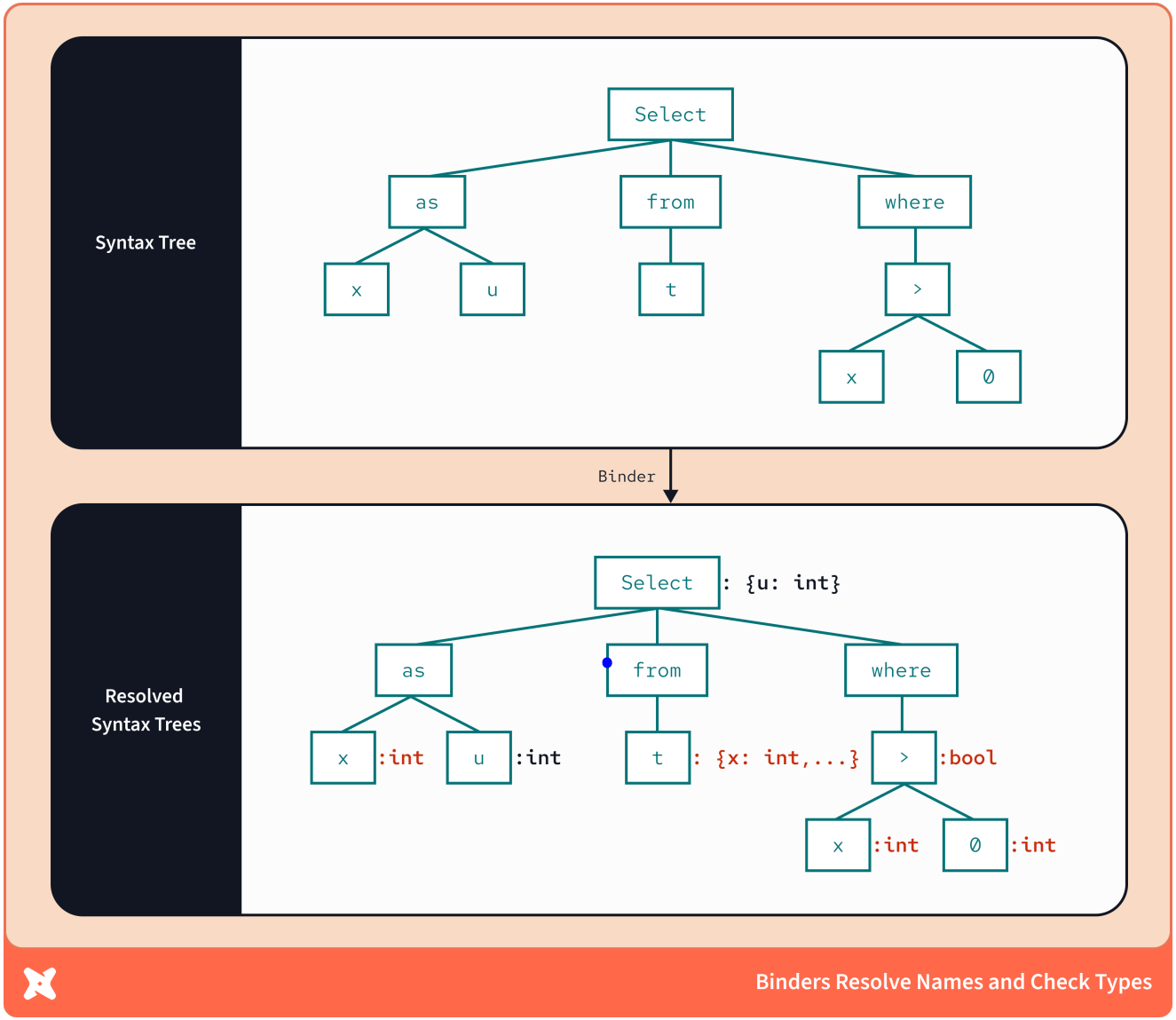
C or Python

Query Execution Flow: Syntax Tree

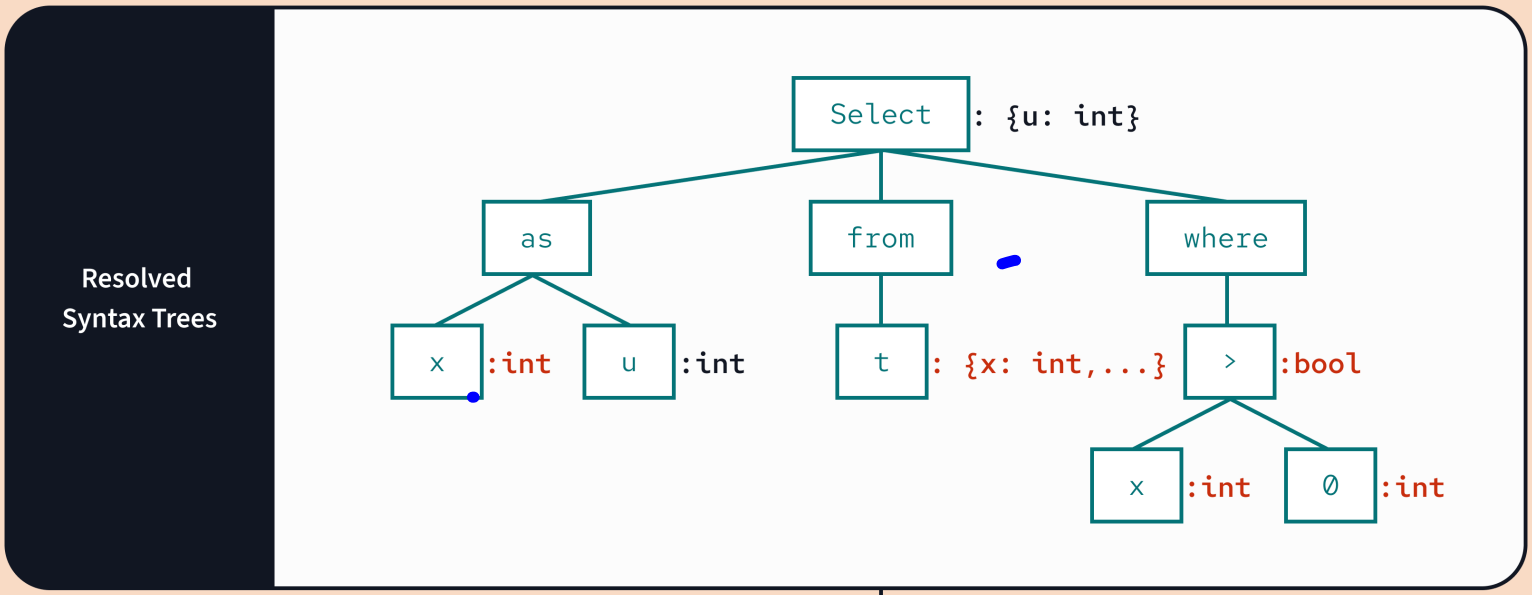


Parsers Recognize the Structure of the Query

Query Execution Flow: Compilation



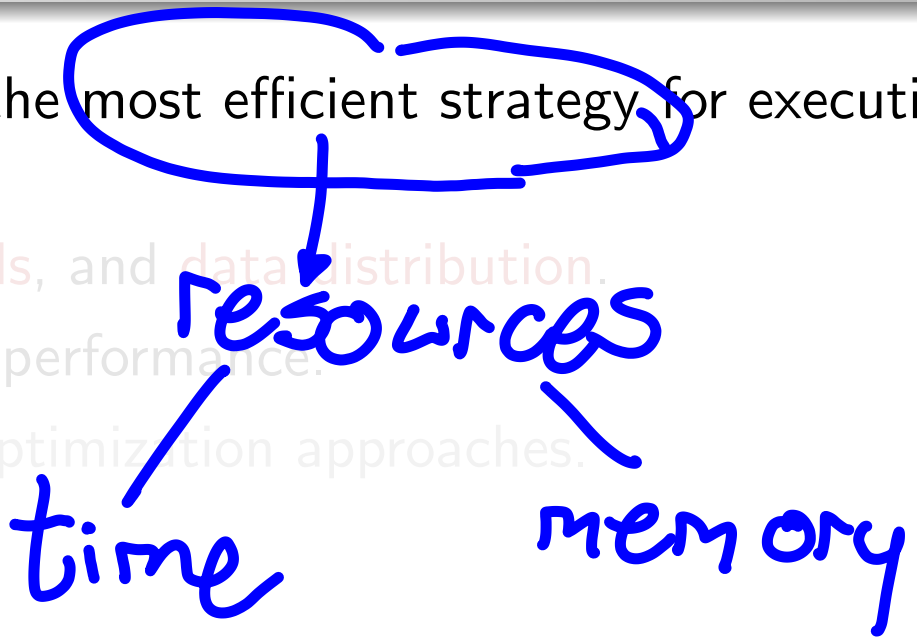
Query Execution Flow: Logical Plan



Compilation Produces an Executable Plan from a Resolved Syntax Tree

Query Optimization

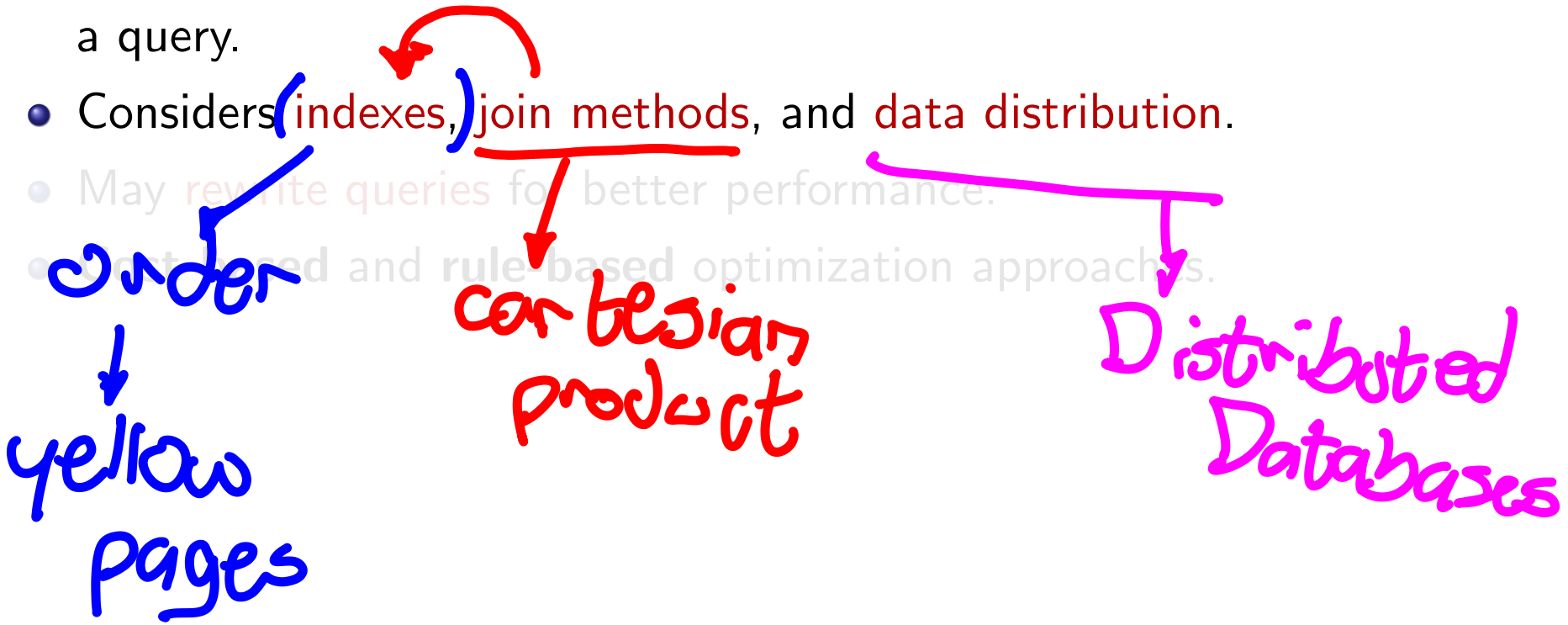
- The **query optimizer** selects the most efficient strategy for executing a query.
- Considers indexes, join methods, and data distribution.
- May rewrite queries for better performance.
- **Cost-based** and **rule-based** optimization approaches.



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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.



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Query Optimization

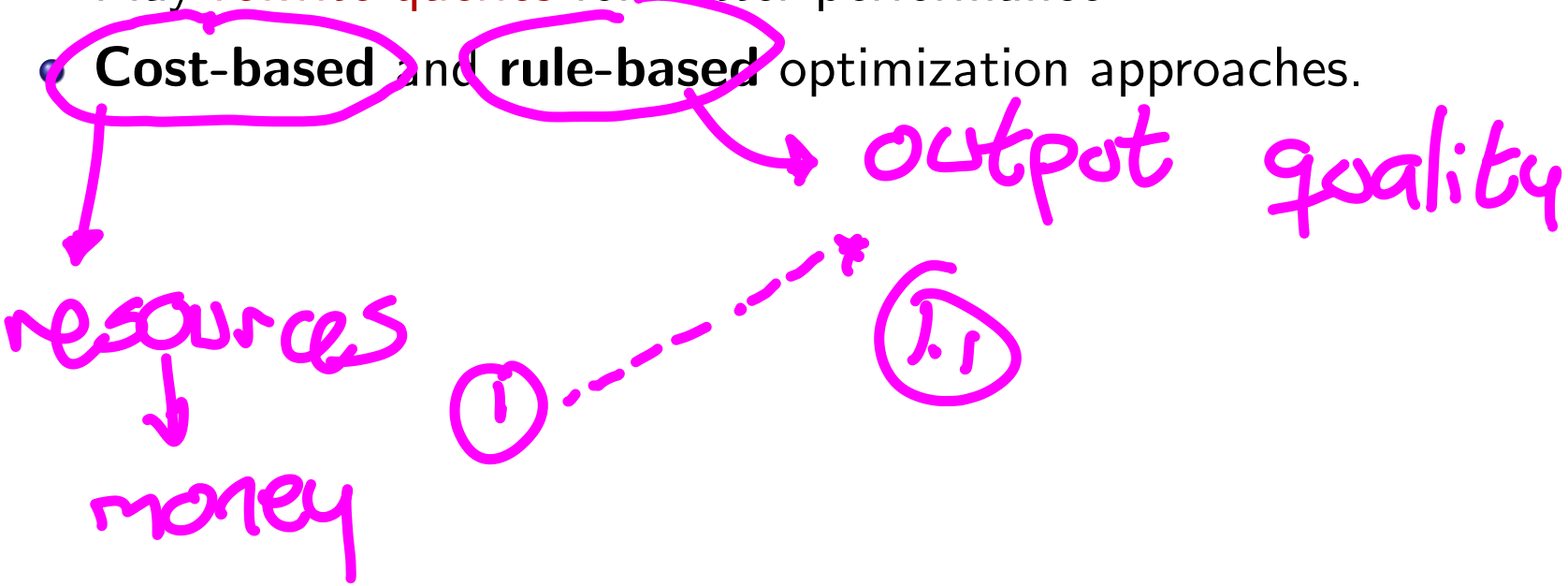
- The **query optimizer** selects the most efficient strategy for executing a query.
- Considers indexes, join methods, and data distribution.
- May rewrite queries for better performance.
- **Cost-based** and **rule-based** optimization approaches.

internal → no the user

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

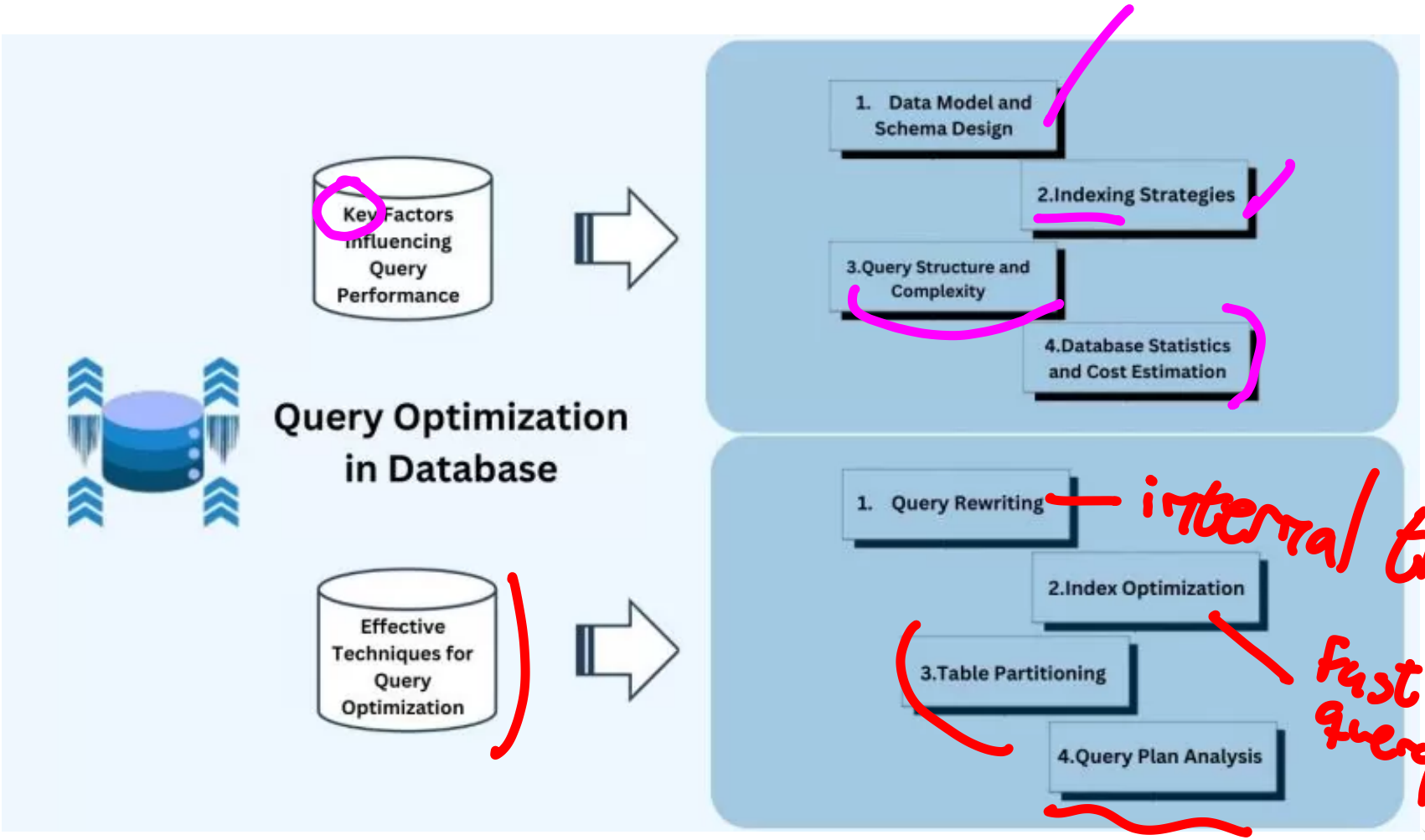
Query Optimization

- 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.



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Query Optimization Factors



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

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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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.

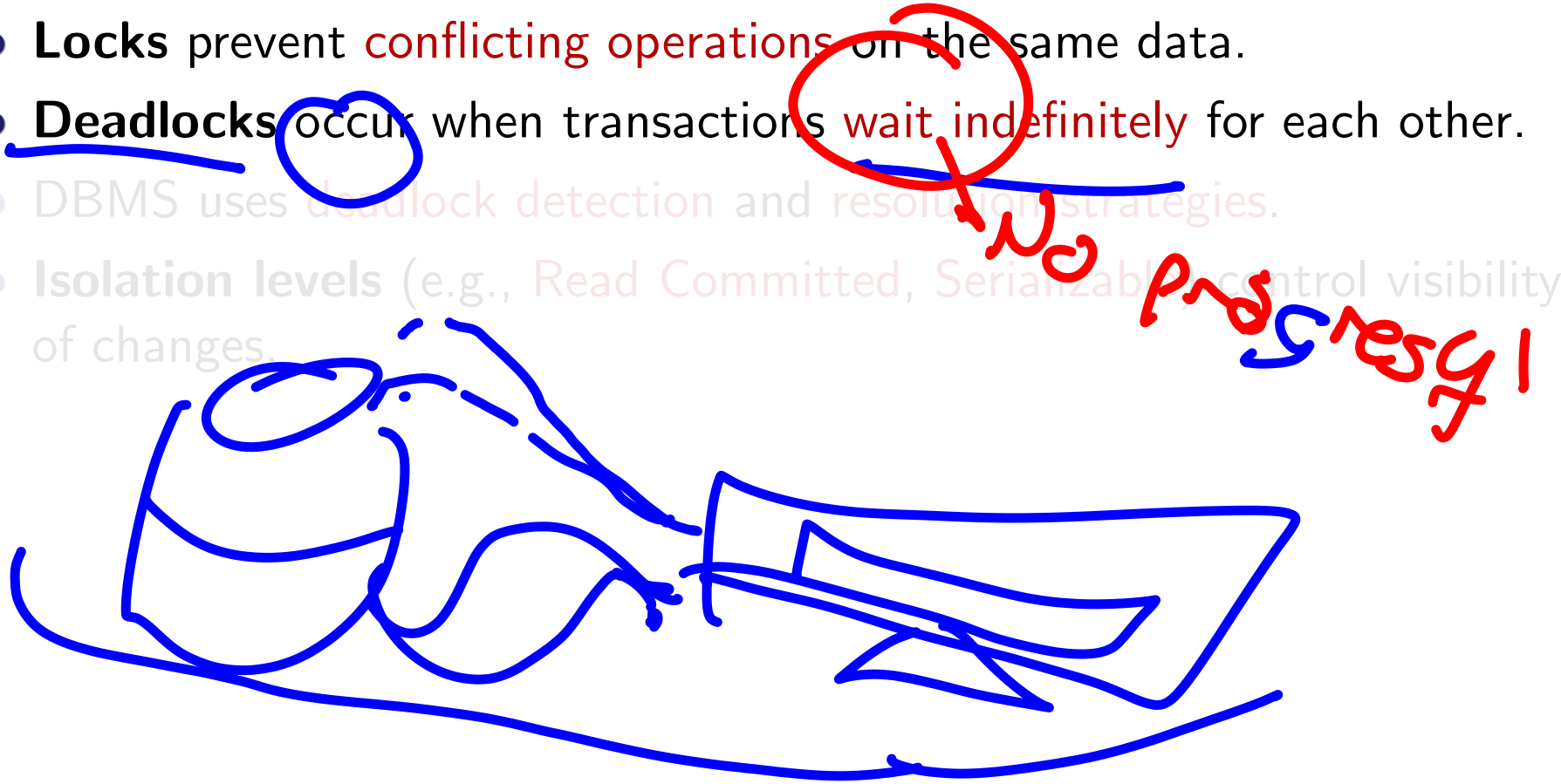
Locking and Deadlocks

- **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.

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.



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.

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.

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

Failure Recovery Concepts

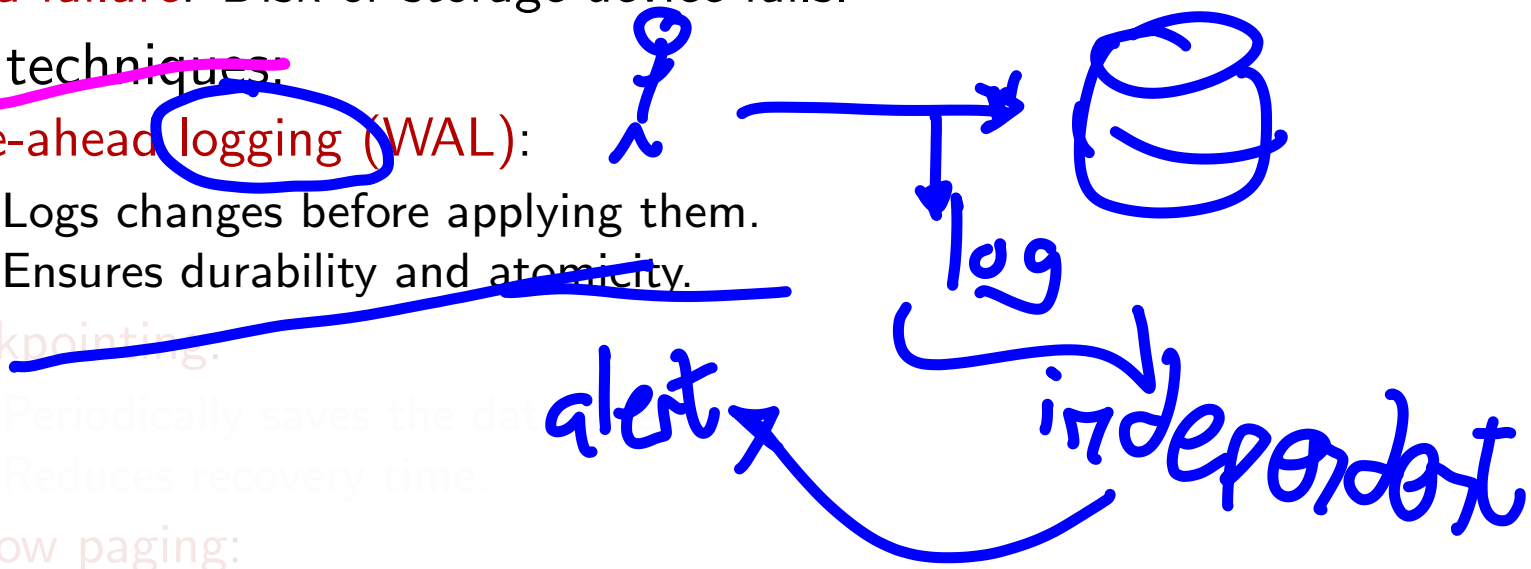
- **Failure recovery** restores the database to a consistent state after a failure.
- Types of failures:
  - Transaction failure: Only one transaction fails.
  - System crash: All active transactions are lost.
  - Media failure: Disk or storage device fails.
- Recovery techniques:
  - Write-ahead logging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



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Failure Recovery Concepts

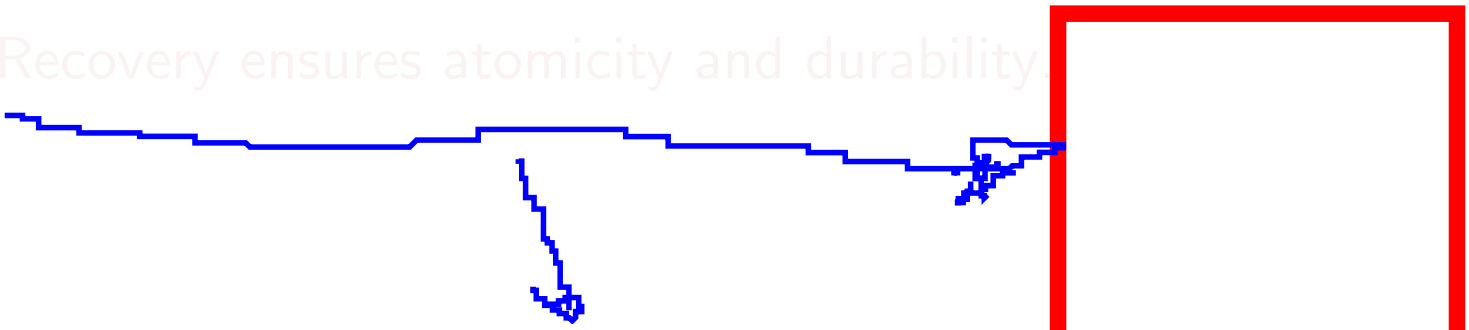
- **Failure recovery** restores the database to a consistent state after a failure.
- Types of failures:
  - Transaction failure: Only one transaction fails.
  - System crash: All active transactions are lost.
  - Media failure: Disk or storage device fails.
- Recovery techniques:
  - Write-ahead logging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



persist into a log

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.

snapshots

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

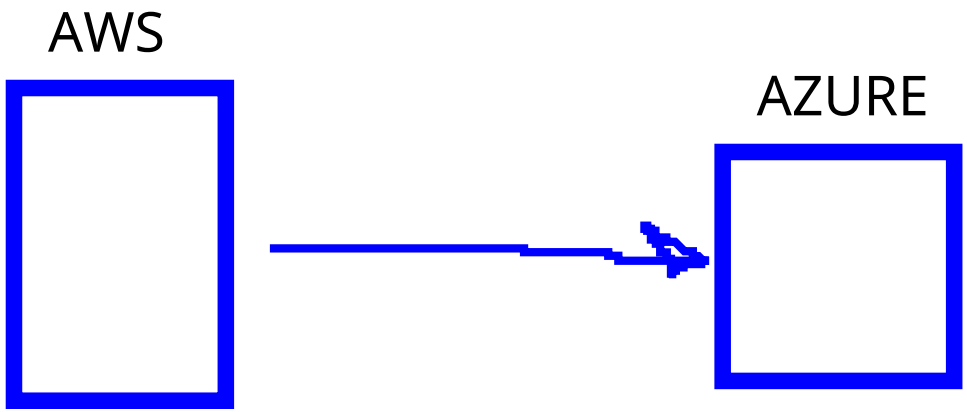
Write-Ahead Logging and Checkpoints

- **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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Thanks!

Questions?

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

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS