

DATA BASE 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



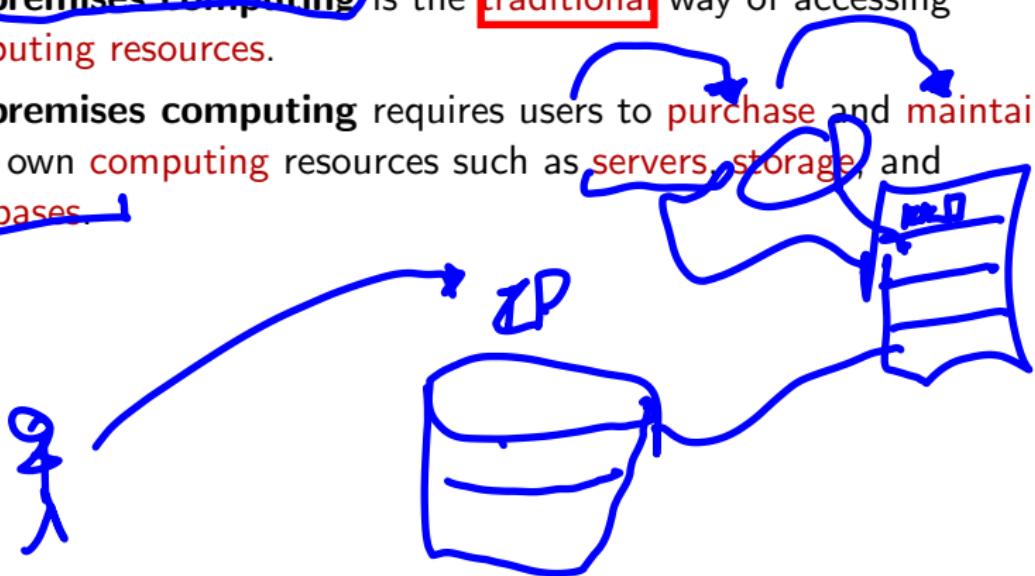
Outline

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



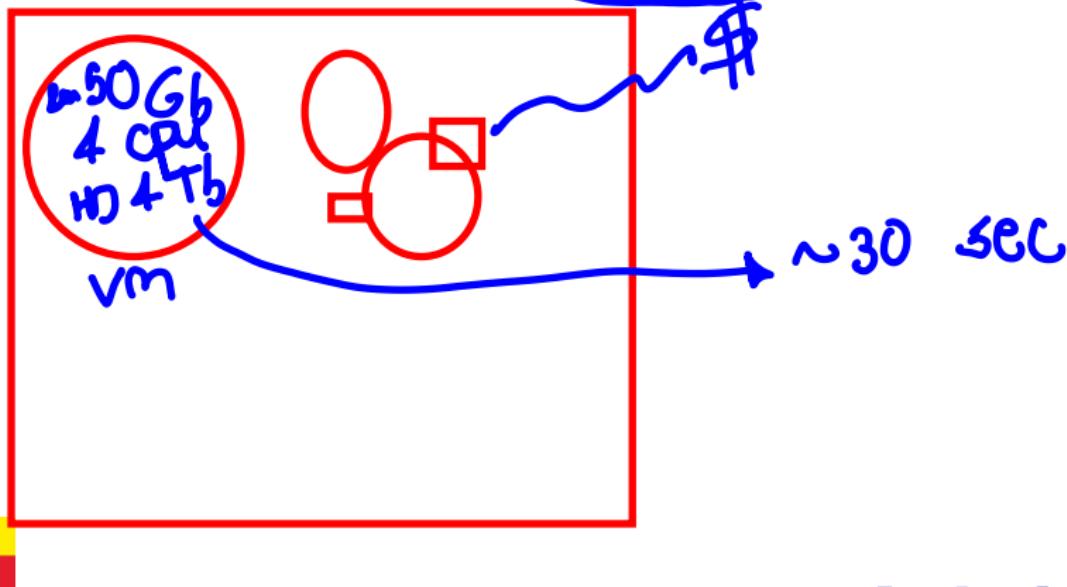
What is On-Premises Computing?

- **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**.

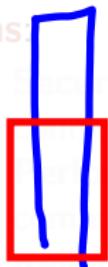


Pros & Cons of Cloud Computing

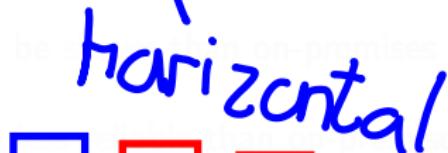
- Pros:

- **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:



vertical



horizontal

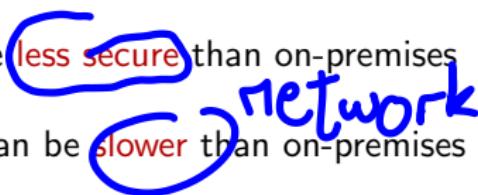


Pros & Cons of Cloud Computing

- **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.

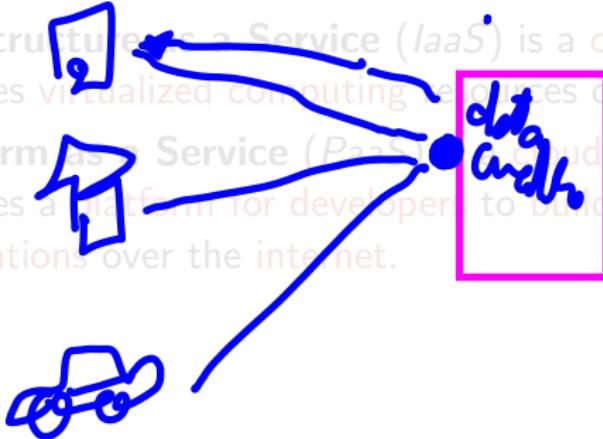
- **Cons:**

- **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

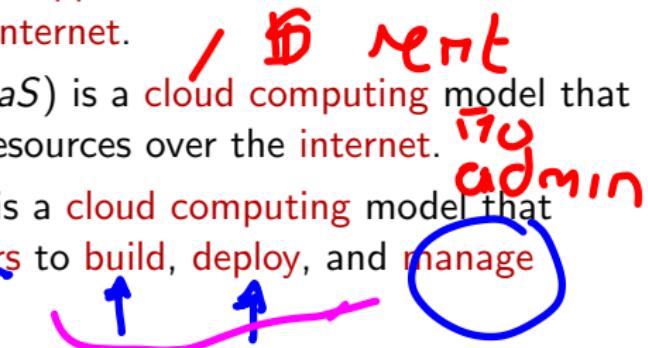
- **Software as a Service (SaaS)** is a **software distribution** model in which a **third-party** provider **hosts** applications and makes them available to customers over the **internet**.
- **Infrastructure as a Service (IaaS)** is a **cloud computing** model that provides **virtualized computing**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

IaC → terraform



SaaS Vs. IaaS Vs. PaaS

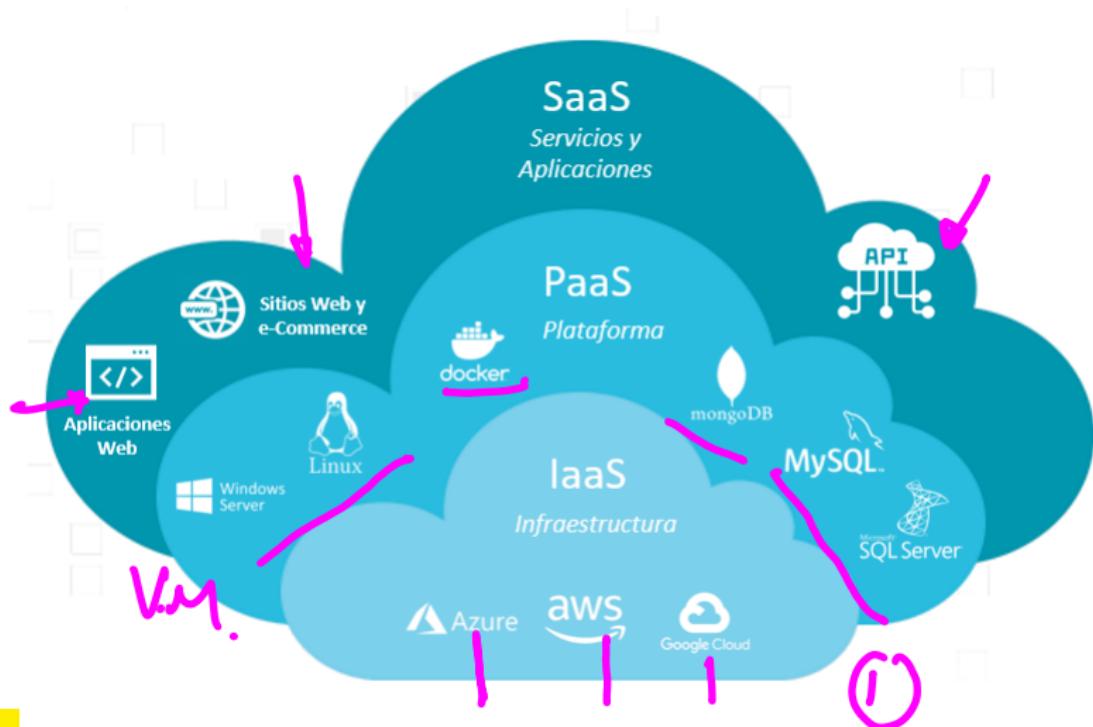
- **Software as a Service (SaaS)** is a **software distribution** model in which a **third-party** provider **hosts** applications and makes them available to customers over the **internet**. */ \$ rent*
- **Infrastructure as a Service (IaaS)** is a **cloud computing** model that provides **virtualized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



Docker - K8s
Artifacts
ML/AI



Cloud Levels



DataBases as a Service

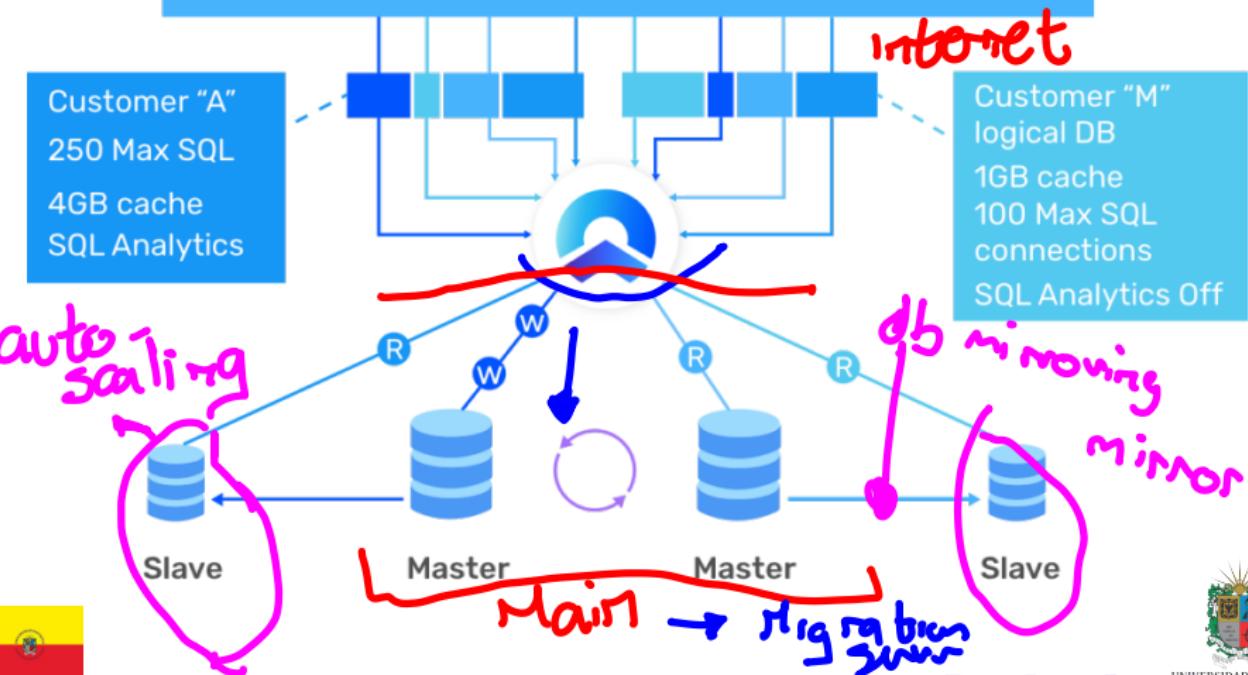
Database as a Service (DBaaS) is a cloud computing model that provides database services over the internet.

- ① DBMS { NoSQL } ready to use
- ② Security : Firewall, Duty, Logs, Policies
- ③ Migration (Snowball → mobile)



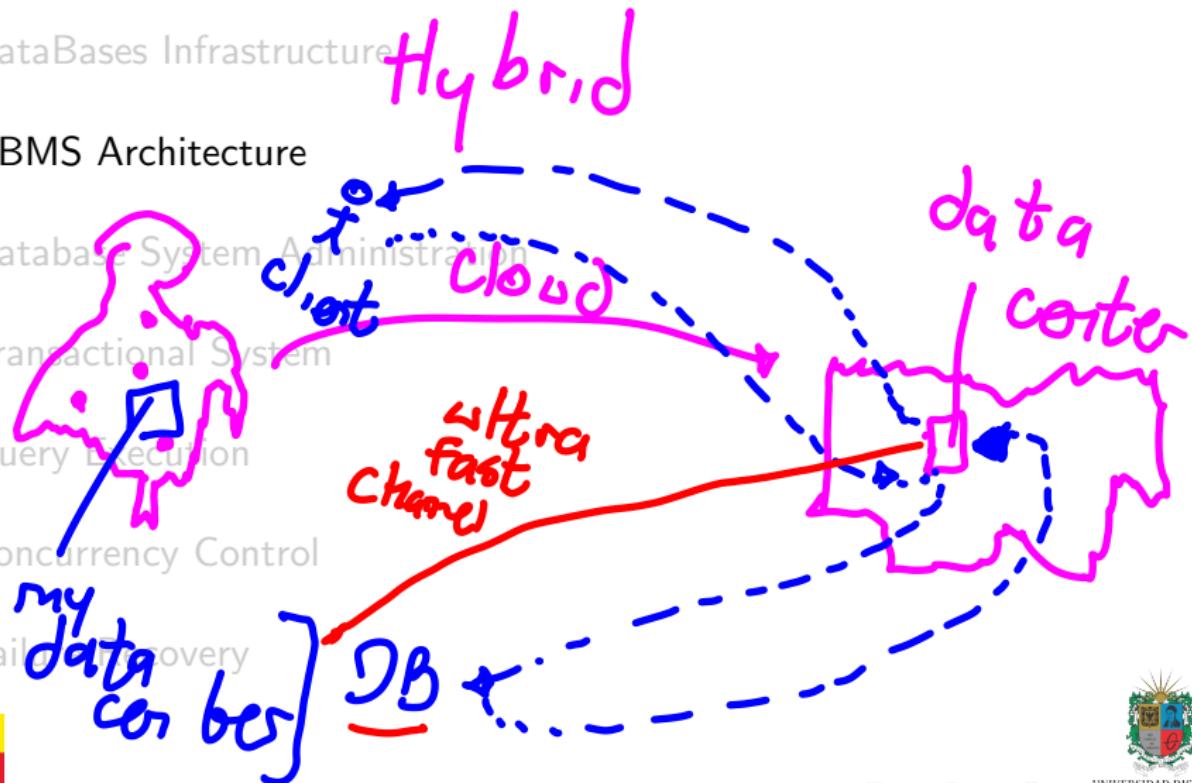
Case Study: DBaaS Custom for Clients

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



Outline

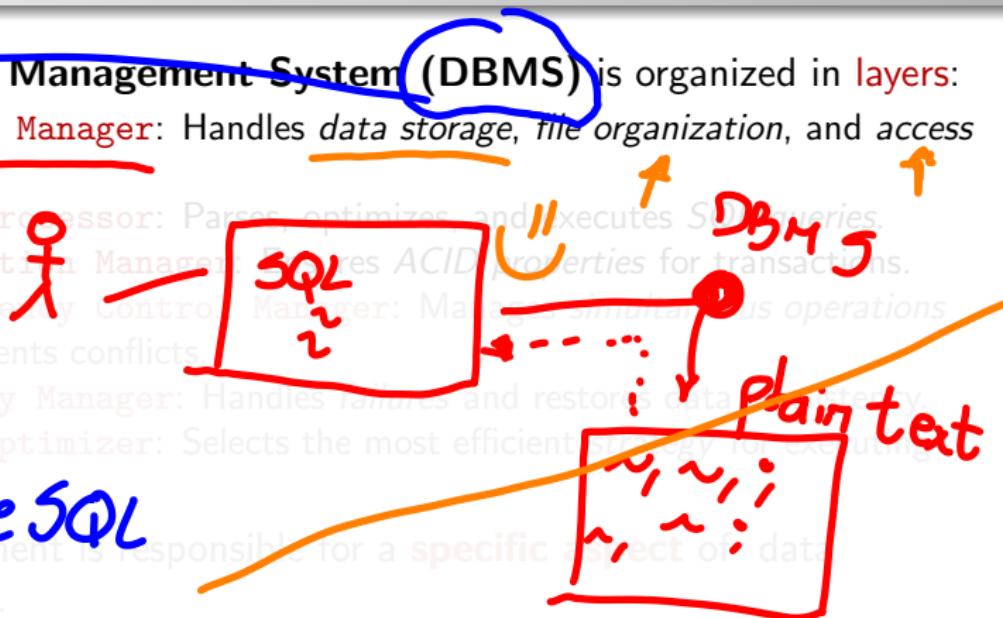
- 1 DataBases Infrastructure
- 2 DBMS Architecture
- 3 Database System Administration
- 4 Transactional System
- 5 Query Execution
- 6 Concurrency Control
- 7 Failure Recovery



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 integrity.
 - Optimizer:** Selects the most efficient query execution plan for executing queries.

MySQL
PostgreSQL
Oracle



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



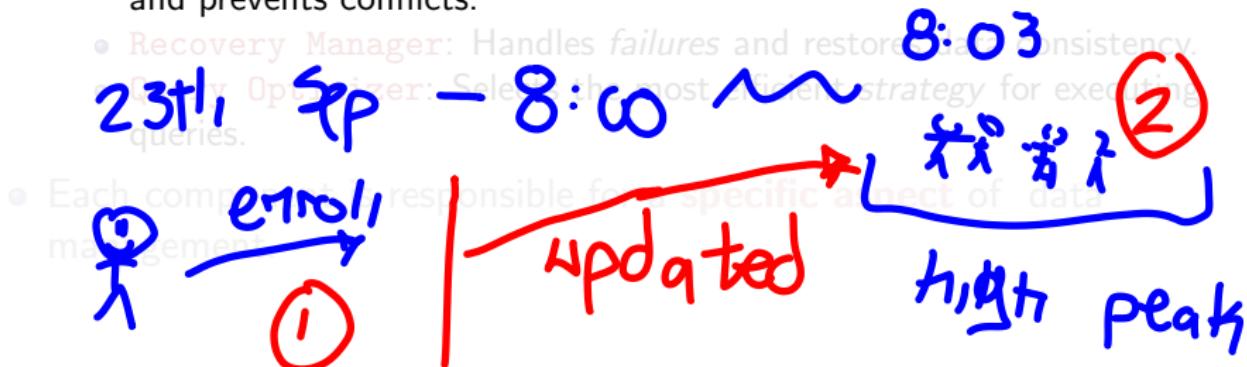
DBMS Architecture Overview

- 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.
 - **Synchronization 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.



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 ~~specific aspect~~ *inherent* aspect of data management.

high level

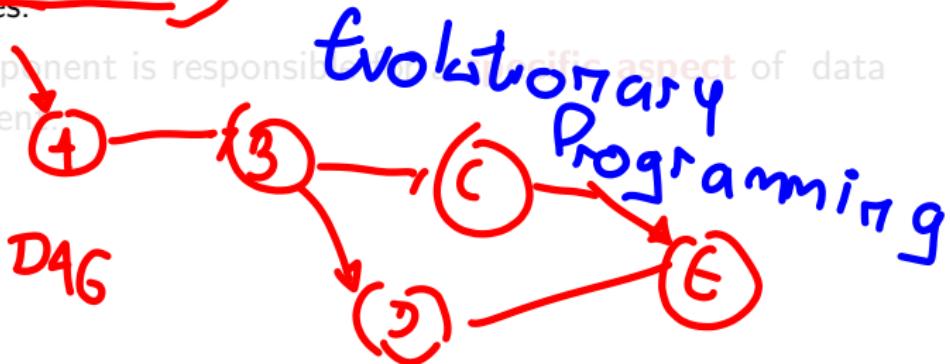
inherent

Backup



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

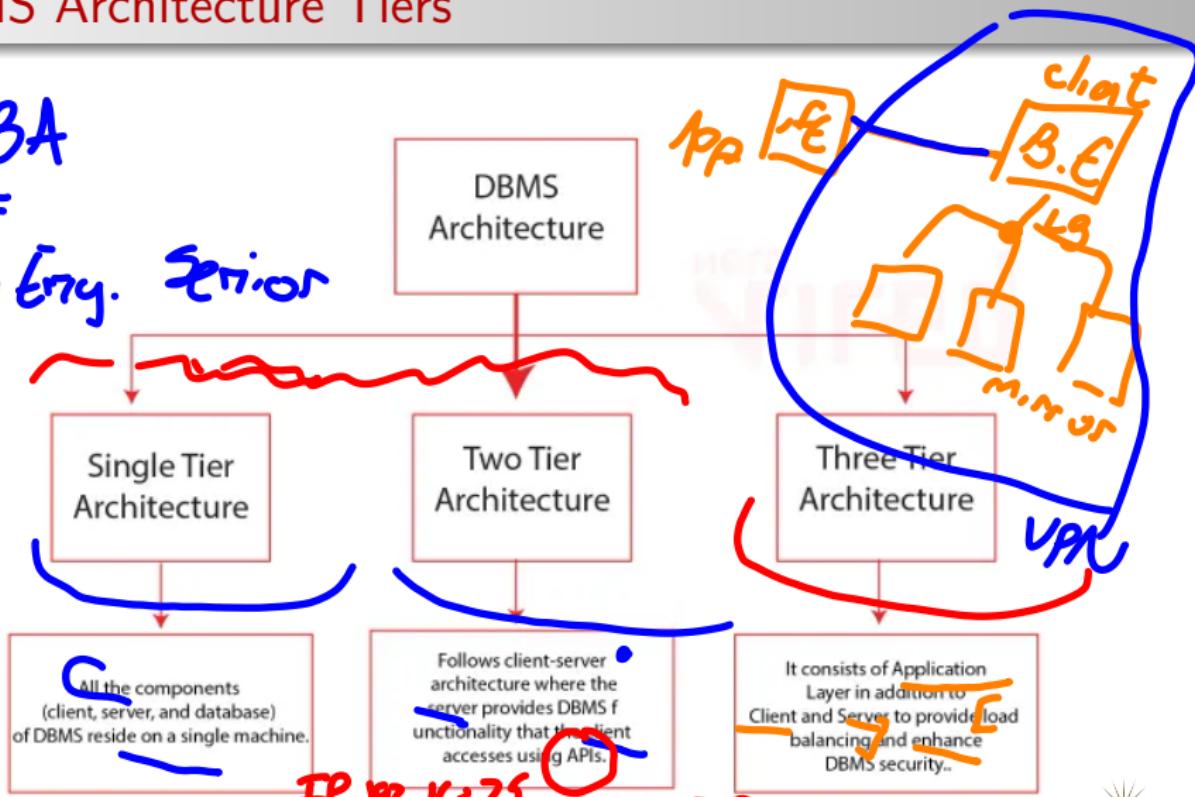
- A **Database Management System (DBMS)** is organized in **layers**:
 - **Storage Manager**: Handles *data storage, file organization, and access methods*.
 - **Query Processor**: Parses, optimizes, and executes *SQL queries*.
 - **Transaction Manager**: Ensures *ACID properties* for transactions.
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

all



DBMS Architecture Tiers

DBA
= Data Engr. Senior



DBMS Architecture N-Tier

DBMS Architecture



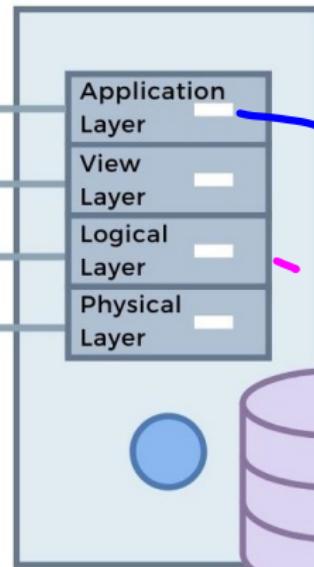
DatabaseTown.com

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.



form (insert, update, delete)
GUI report
SQL query (SELECT)
component

Backend

Relational Algebra



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.

1-tier
developers



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 stored in RAM for faster access.

parallelism

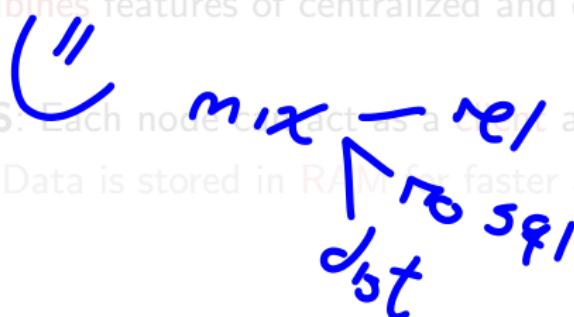
to replicas



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 **acts** 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 both 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.

DB - 100-200ms

Cache - 5-10 ms

Cache
Lifetime (sec-min)

max 30
hours



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:
 - everything it works
 - memory processor time
 - profiling
 - grants
 - backups
 - accessible
 - scalability
- 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:



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
- **Indexes**
- **Improve DB model**
- **Material 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 responsible for managing databases in modern organizations.
- Sync*
- replicas / only read*



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

Plan

Backup and recovery

Security and access control

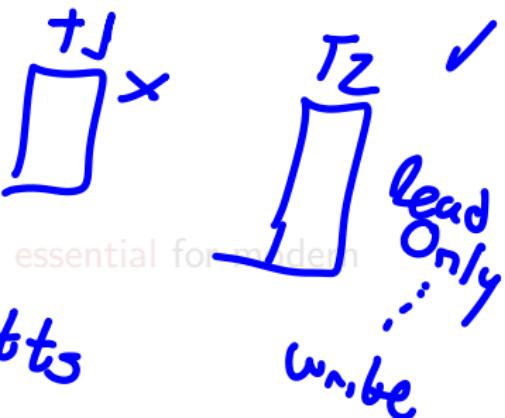
- Database administrators (DBAs) are **essential** for modern organizations.

today



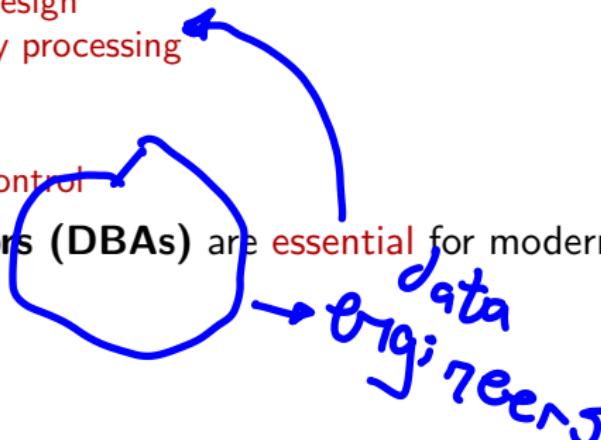
Database System Administration

- 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

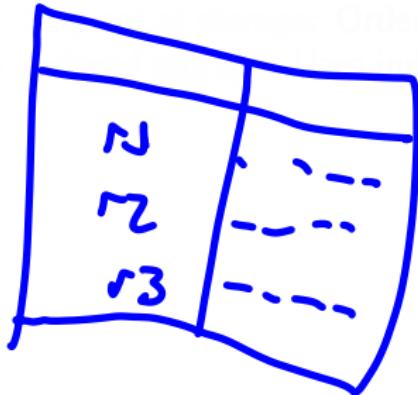
- **Database system administration** is the discipline of **managing, configuring, and maintaining database systems** to ensure their reliability, performance, security, and availability.
- Key responsibilities include:
 - Data storage management
 - System architecture design
 - Transaction and query 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

save variables



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 storage.
 - Sequential storage: Ordered, fast range queries.
 - Hash 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

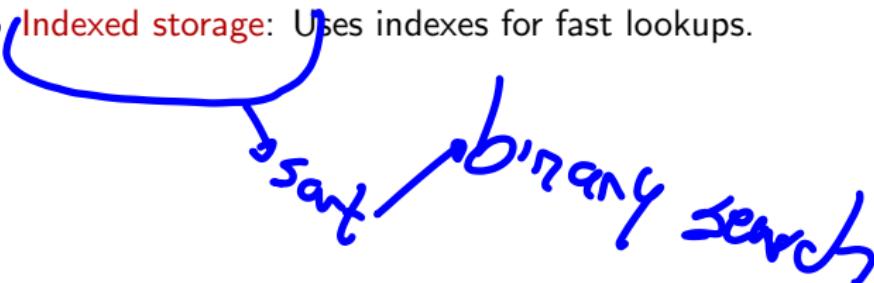
- A **record** (or **row/tuple**) is the **basic unit** of data storage in a *database table*.
- **Efficient record storage** is crucial for **fast** data retrieval and **update**.
- Storage techniques:
 - **Heap storage**: Unordered, fast inserts.
 - **Sequential storage**: Ordered, fast range queries.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

say binary search



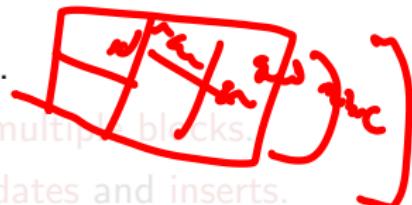
Block and Page Organization

- 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**.



Teacher

```

SELECT s.name career 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;
    
```

Jo
A hand-drawn diagram of a page organization. It shows a large rectangular page divided into smaller rectangular sections, each labeled with a number (1, 2, 3, etc.). Some sections contain vertical lines, representing slots for records. An arrow points from the text "Jo" to this diagram.



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

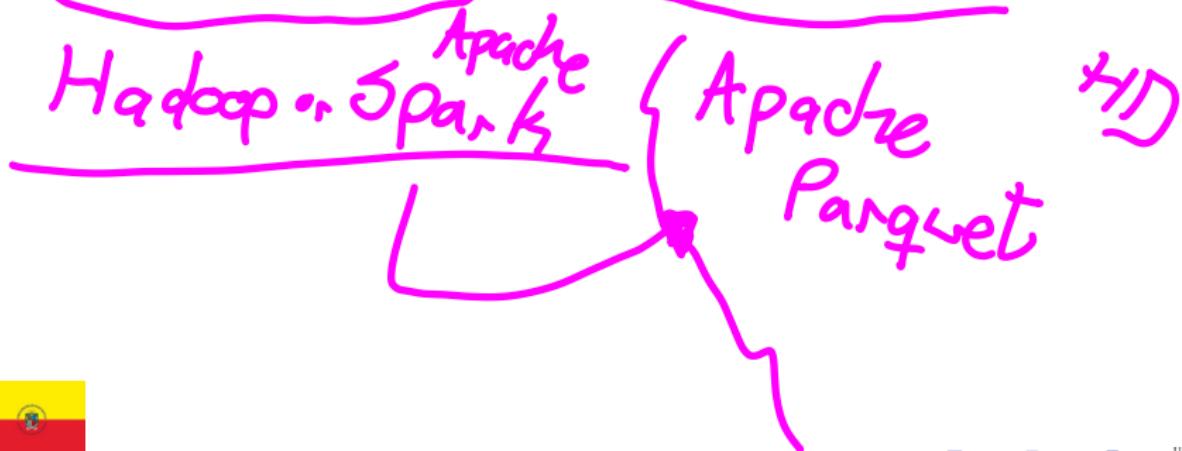
- Data is stored in **blocks** (pages) on disk.
- **Block size** and **layout** affect **I/O performance**.
- Records may be **packed**, **slotted**, or may **span multiple blocks**.
- **Free space management** is important for **updates and inserts**.
- **Fragmentation management** ensures efficient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

Distrubuted
Data Sysem



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.

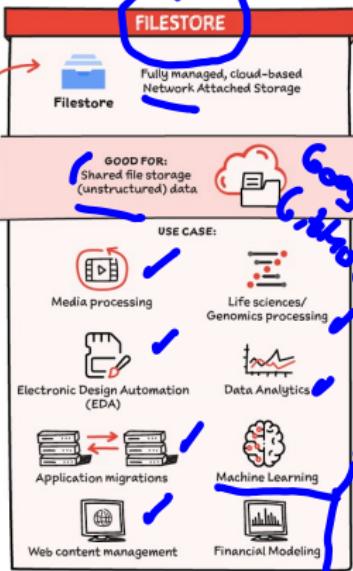
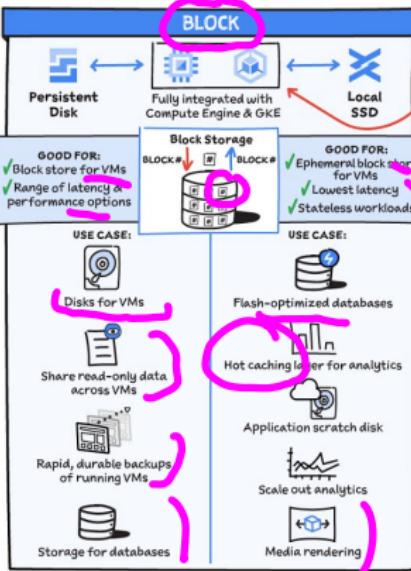
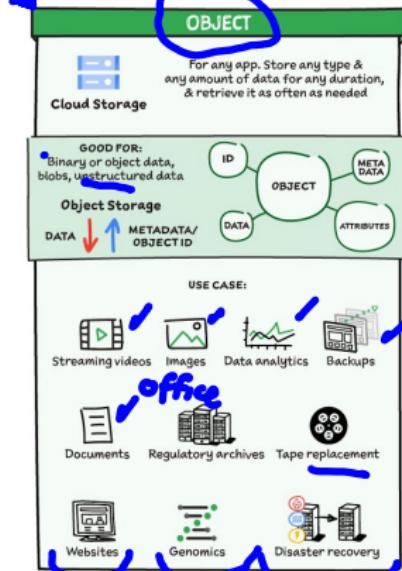


Record Storage: Use Cases

#GAPSketchnote
 @PVERGADIA
 THECLOUDGIRL.DEV
 04.23.2021



Which Storage Should I Use?



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.

C
E
V
D

queue

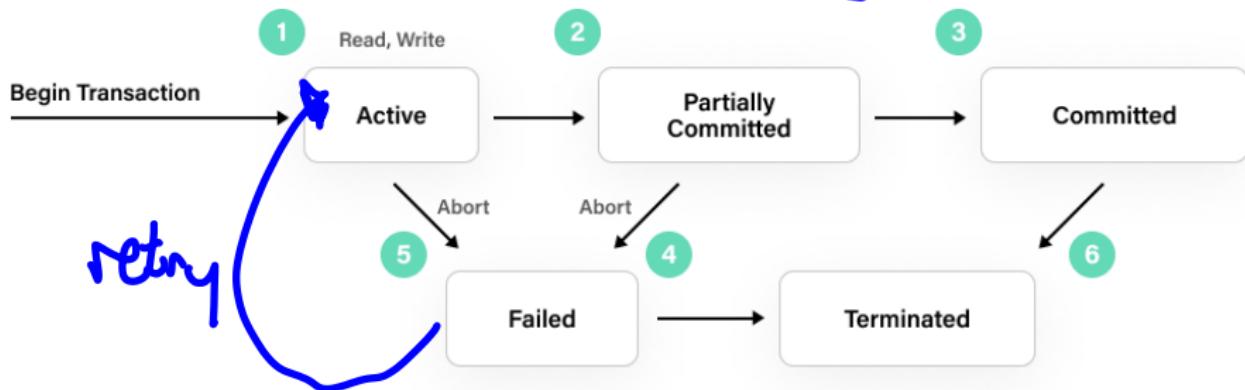
queries

concurrency



Transaction Lifecycle

- **Begin**: Transaction starts. → enter
- **Read/Write**: Operations are performed. → memory
- **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



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**.

Snowflake

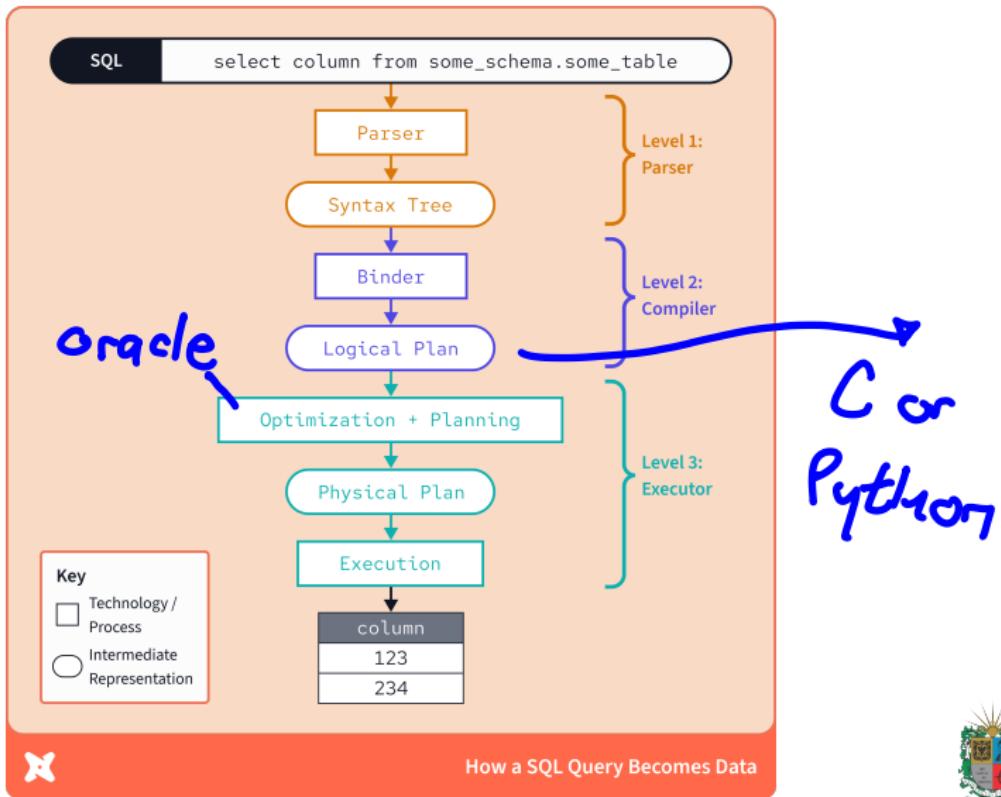
Compiler

SQL
functions

I
declarative



Query Execution Flow: Full Transaction



How a SQL Query Becomes Data



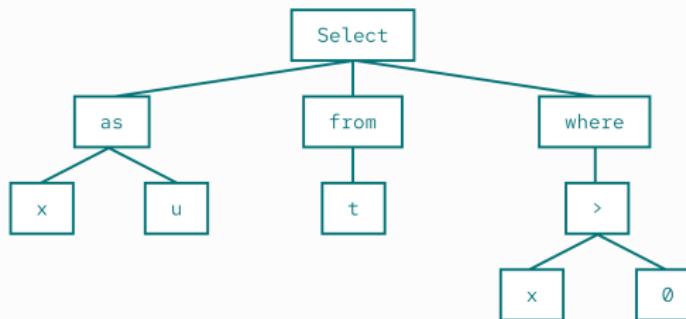
Query Execution Flow: Syntax Tree

SQL

select x as u from t where x > 0

Parser

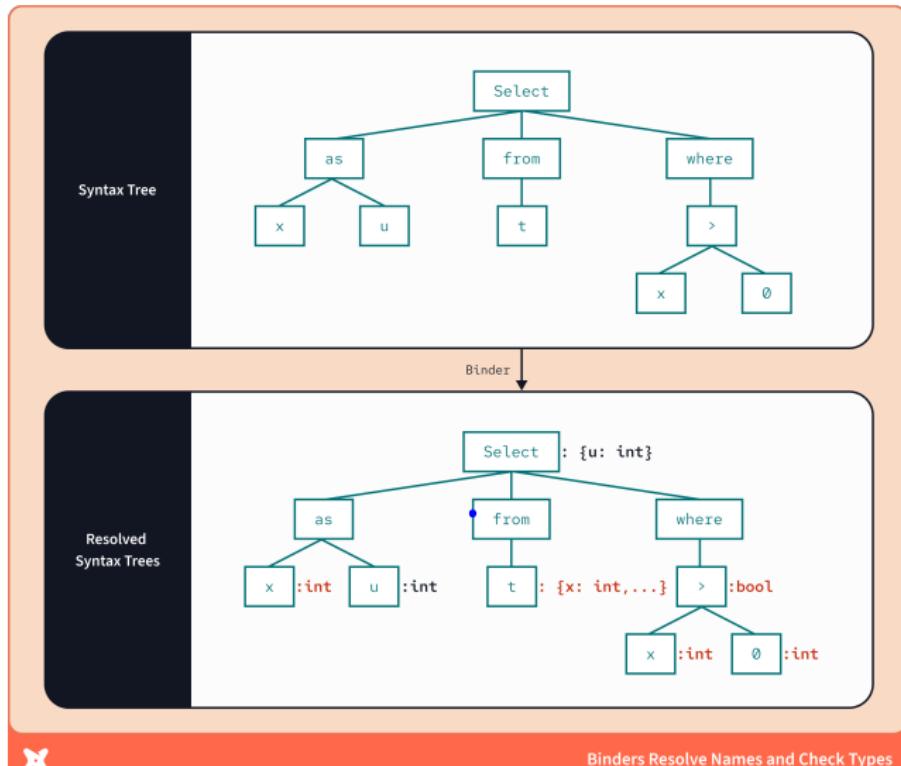
Syntax Tree



Parsers Recognize the Structure of the Query



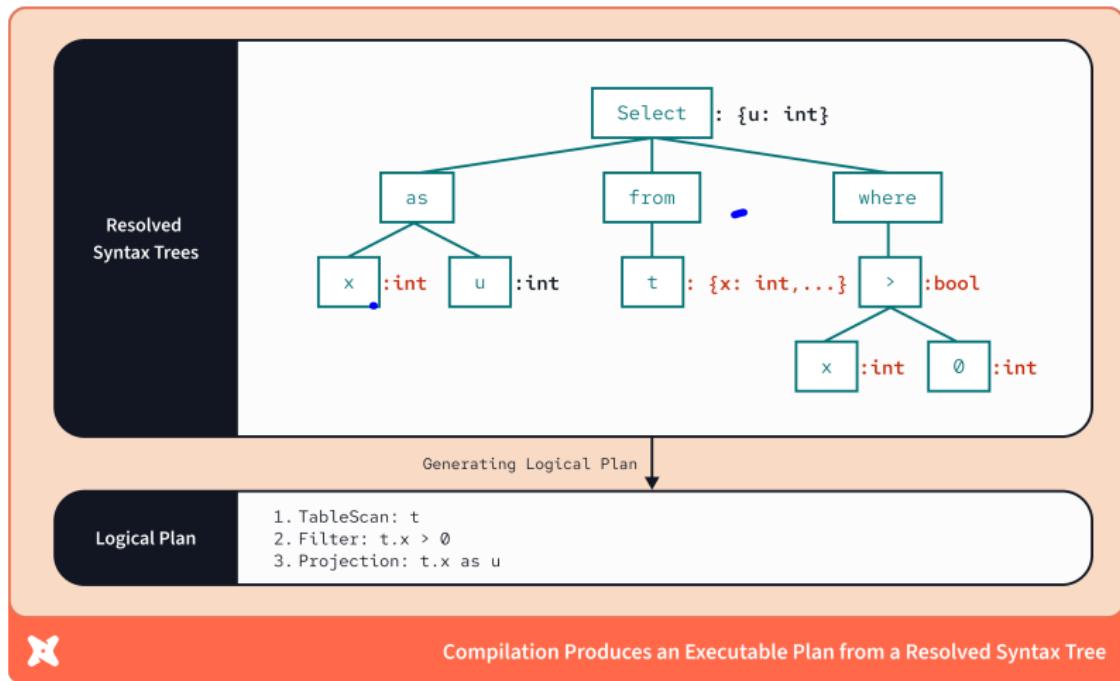
Query Execution Flow: Compilation



Binders Resolve Names and Check Types

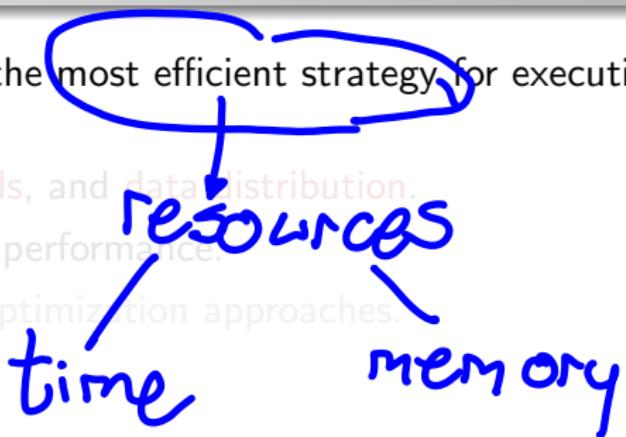


Query Execution Flow: Logical Plan



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.
- May use **cost-based** and **rule-based** optimization approaches.

order
↓
**yellow
pages**

join methods
↓
**cartesian
product**

data distribution
↓

**Distributed
Databases**



Query Optimization

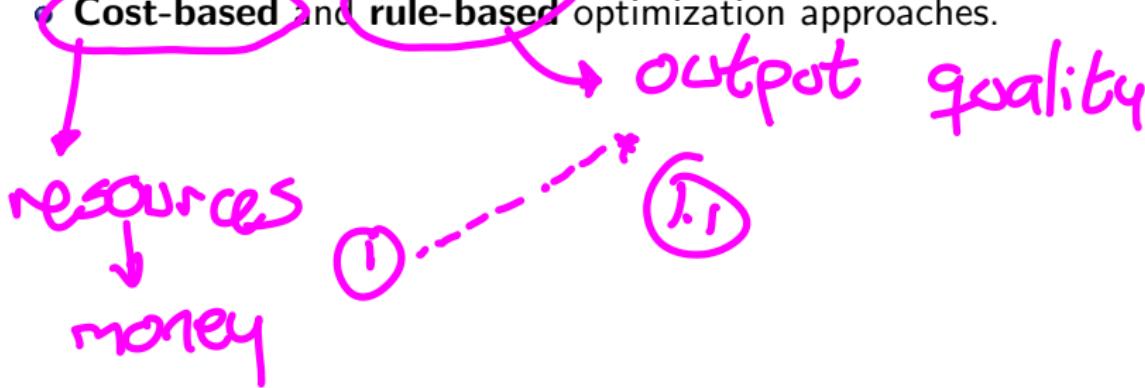
- The **query optimizer** selects the most efficient strategy for executing a query.
- Considers **indexes**, join methods, and **data distribution**.
- May **rewrite queries** for better performance.
- Cost-based and rule-based optimization approaches.

internal → no the user

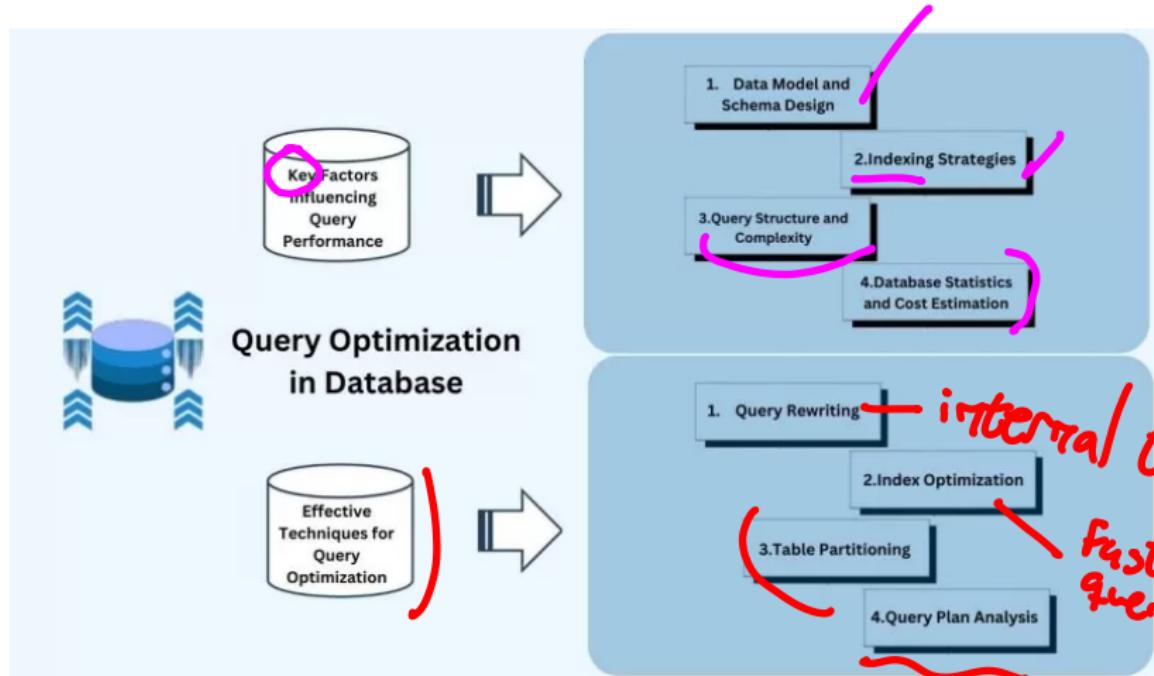


Query Optimization

- 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 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.



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.

Fast



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.



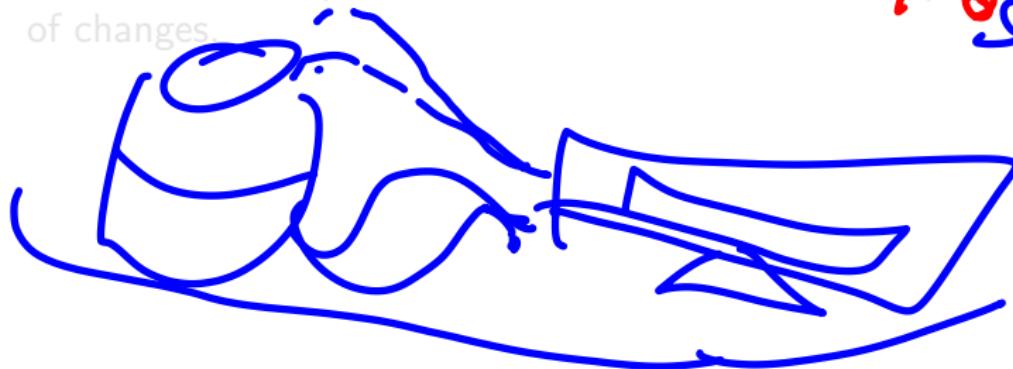
Locking and Deadlocks

- 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
- processes*



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)

• Write-ahead logging (WAL) is a technique used for database recovery. It involves logging all write operations to a log file before they are applied to the database. This allows the system to roll back incomplete transactions and redo completed ones if a failure occurs.

• WAL is used in conjunction with other recovery techniques like checkpointing and redo logs.

• WAL is a common technique used in modern databases like MySQL, PostgreSQL, and Oracle.

• WAL is also used in distributed systems like Apache Hadoop and Apache Flink.

• WAL is a reliable and efficient technique for database 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:

• 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.
 - **Checkpointing**
 - Periodically saves 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:**

→ Snapshot
↓
incremental
query

Maintaining consistent images.

Allowing 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.

serv. tareas



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>

