Database Administration Security

Cybersecurity Department

Course Code: ***CBS 214***

Theoretical Lecture 2: Database Security Principles

Halal Abdulrahman Ahmed

Lecture Outlines

- **Database Security Introduction**
- **ACID Principles** (Atomicity, Consistency, Isolation, Durability)
- **BASE Principles** (Basically Available, Soft State, Eventual Consistency)
- **Data States** (Data at Rest vs. Data in Transit)
- **Designing a Secure Database**

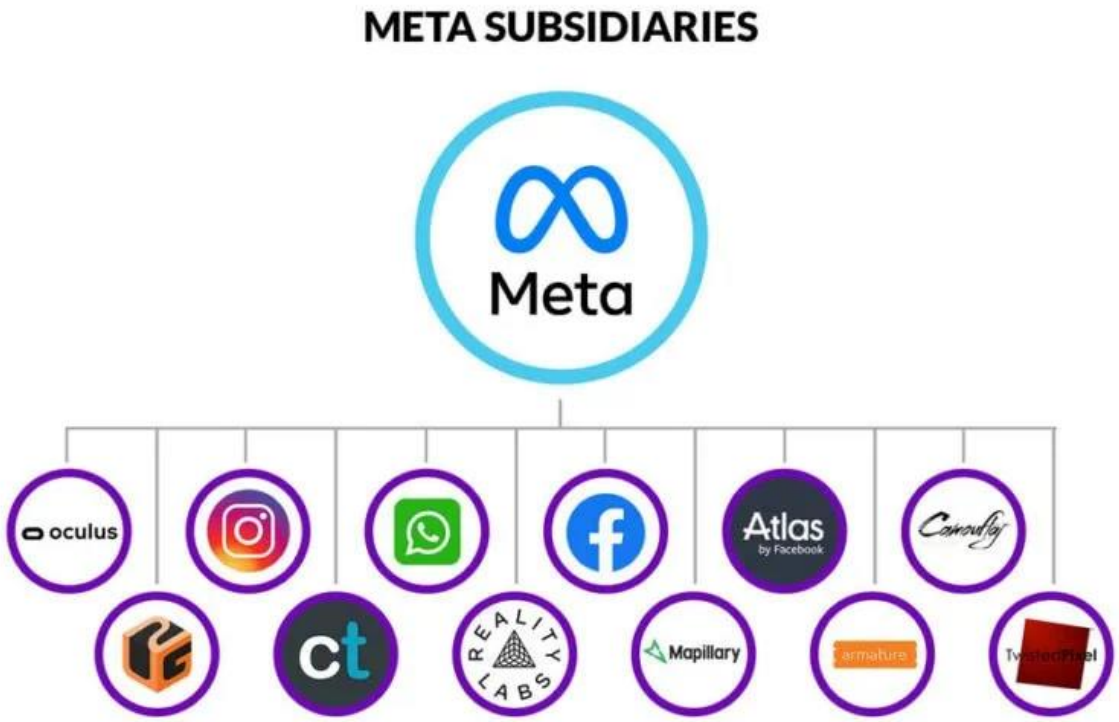
Learning Outcomes

By the end of this lecture, students will be able to:

- **Understand the CIA Triad:** Explain how Confidentiality, Integrity, and Availability apply specifically to database systems.
- **Analyze Transaction Models:** Differentiate between **ACID** properties for data integrity and **BASE** properties for high-scale distributed systems.
- **Differentiate Data States:** Identify the unique security risks and defenses for **Data at Rest** versus **Data in Transit**.
- **Apply Security Design:** Implement a multi-layered defense strategy, including physical isolation, network hardening, and real-time auditing.

Database Security Introduction

Security is important for databases and their applications, the transactions that happen on a DB must conform to the basic security triad principles of CIA (confidentiality, integrity, and availability.



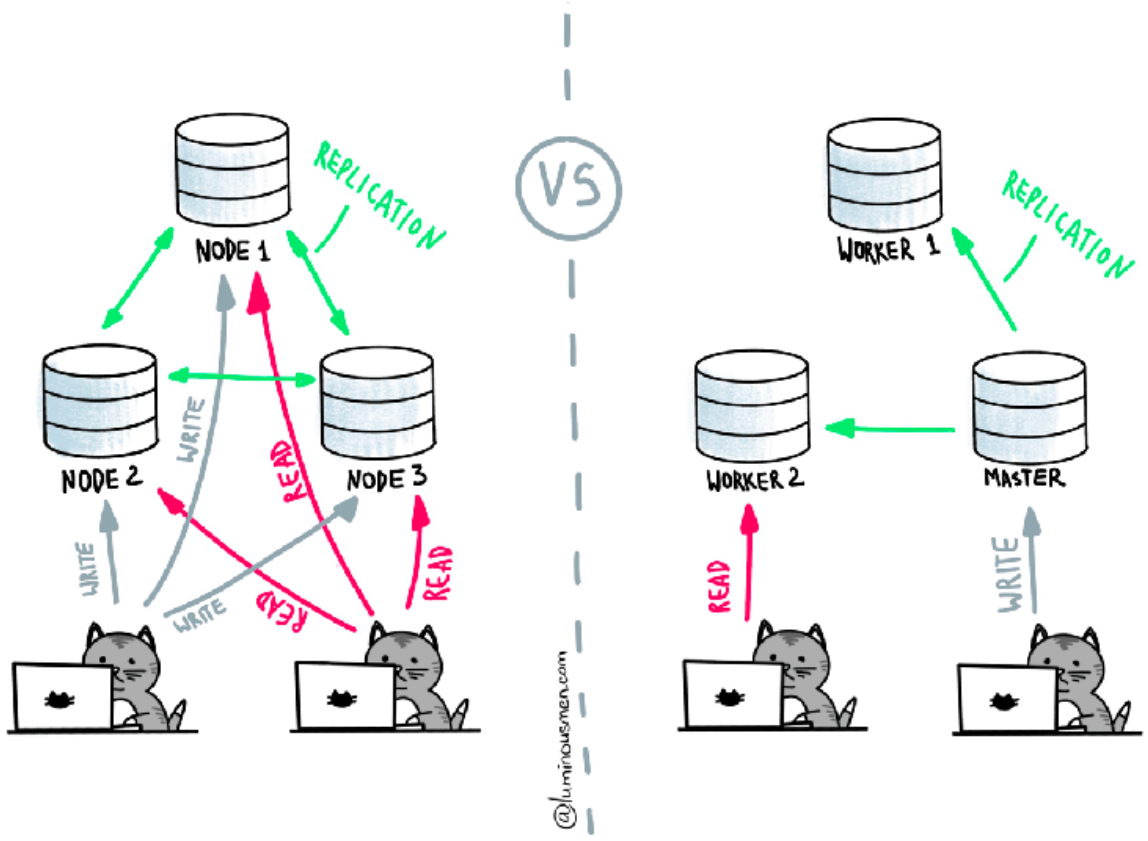
Facebook’s "Big Data" Scale Today

Every single minute, Facebook processes a massive amount of information. Here is what happens in just **60 seconds**:

- **Users:** Over **2 million people** log in.
- **Likes:** Users click the "Like" button **4 million times**.
- **Photos:** More than **136,000 photos** are uploaded.
- **Video:** Users watch **138.9 million Reels** (this is the fastest-growing category).
- **Total Reach:** Facebook now has **3.07 billion** monthly users, the first social platform to reach this size.

Assume that everything is stored in a database, each transaction has to be discrete, transactions should not interfere with other transactions, and all data should be available when required to needed users. This is where the basic ACID principles of a DB spring to action.

ACID and BASE



ACID - Atomicity (All-or-Nothing)

- **Atomicity** means a transaction is treated as **one complete unit**
- Either **all steps succeed**, or **none of them are applied**
- The database will **never keep a half-finished transaction**
- If any step fails, the DB undoes the changes using a **rollback**
- This protects **data integrity**, because partial updates can leave data in a wrong state.

ACID - Atomicity (All-or-Nothing)

Think of a transaction like one package: it must be delivered fully, or not delivered at all. If something goes wrong halfway, SQL Server rolls everything back so the database stays correct.

ACID - Consistency (Rules Always Hold)

- **Consistency** means the database must always follow its **rules and constraints**
- A transaction should move the database from **one valid state** to **another valid state**
- Even if a transaction fails, the database should stay **stable** (no corruption/crash)
- Any inserted/updated data must satisfy **all defined rules**, such as:
    a. data types
    b. PRIMARY KEY / FOREIGN KEY
    c. NOT NULL / CHECK constraints
    d. business rules

ACID - Consistency (Rules Always Hold)

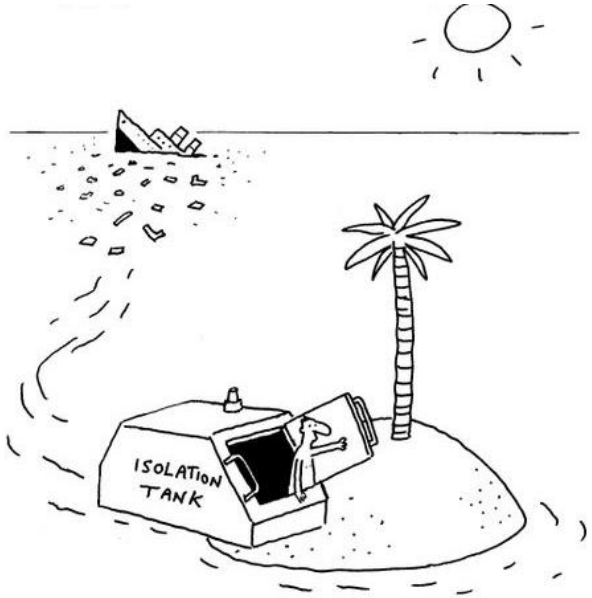
Consistency means the database is always ‘correct’ according to its rules. If someone tries to store invalid data, SQL Server should reject it, and the database remains in a valid state.

ACID - Isolation (Transactions Don’t Interfere)

- **Isolation** means each transaction runs as if it is **alone** in the database
- Multiple transactions can run at the same time, but they should **not conflict** or produce incorrect results
- One transaction should not see **unfinished changes** made by another transaction
- If a transaction triggers or starts another transaction, each must still complete in a **controlled, separate** way

ACID - Isolation (Transactions Don’t Interfere)

Isolation prevents ‘collisions’ when many users work at the same time. It avoids problems like reading temporary data or overwriting someone else’s update. In SQL Server, isolation levels help control how much one transaction can see from another.



ACID - Durability (Committed = Saved Permanently)

- **Durability** means once a transaction is **committed**, its results are **permanent**
- Even if the system crashes (power failure, server restart), the committed changes **must not be lost**
- The database must be able to **recover** the committed data after it comes back online
- In practice, durability is supported by mechanisms like:
  - transaction logging
  - recovery/restore processes

ACID – Durability (Committed = Saved Permanently)

Durability is the promise that ‘Commit means done.’ If SQL Server says the transaction is committed, then even after a crash, the data should still be there when the database recovers.

SAVE

BASE

- **BASE = a design approach for high-scale databases**
- Common in **distributed / NoSQL** systems, BASE is used when the system must keep working even during network delays or node issues. Consistency may be delayed.
- Often prioritizes **availability and speed** over immediate consistency
- Data may be temporarily different across nodes, then it synchronizes and **BASE stands for:**
    a. **B**asically **A**vailable
    b. **S**oft state
    c. **E**ventual consistency

B - Basically Available

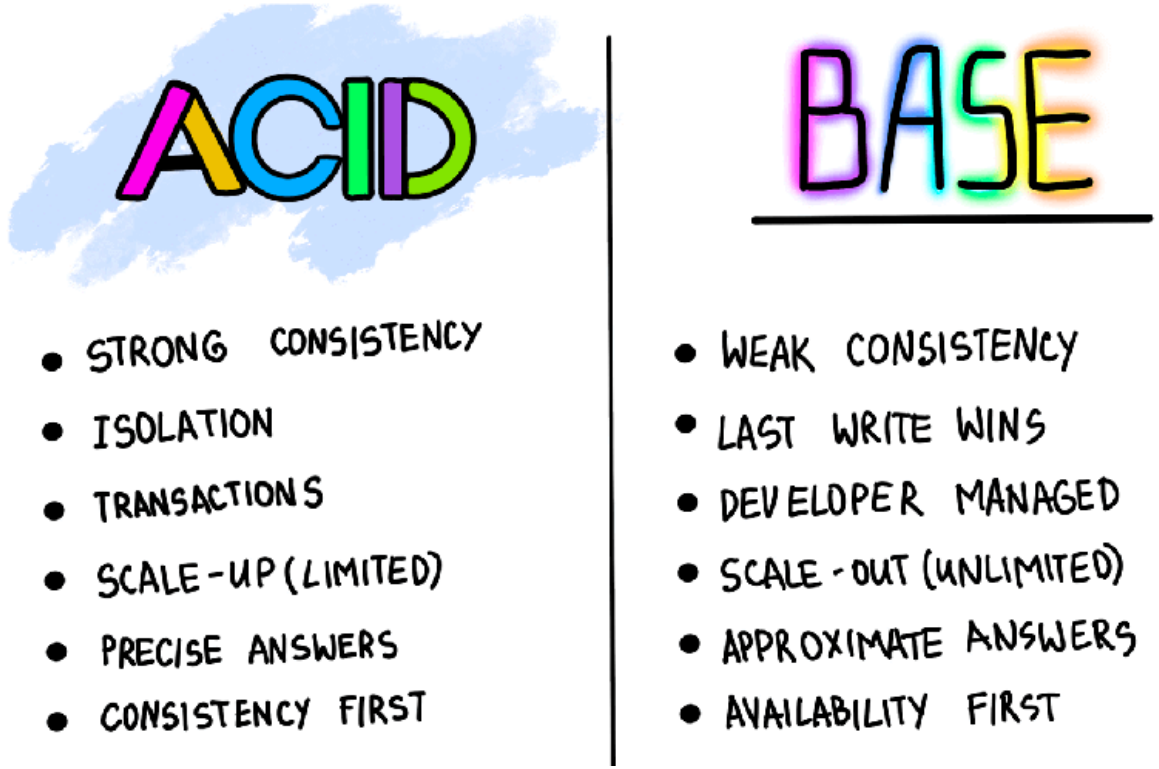
- Reads/writes are generally **available (**“Available” does not mean ‘always perfectly consistent)
- But **freshness is not guaranteed** at the exact moment you read
- You might read an **older value** while another node already has a newer value

S - Soft State

- The database state can **change over time** due to background syncing
- At a given moment, the system may not show the latest global truth
- Soft state means the system is still converging.

E - Eventual Consistency

- Data becomes consistent **over time** across nodes/clusters
- The exact time is **not fixed**; it depends on network and replication delays
- Immediately after a transaction, different users may see different results



ACID	BASE
ATOMICITY: A database transaction is completed in full or it is discarded or rolled back. There are no partial transactions. Also known as "All or none."	BASICALLY AVALABLE: NoSQL databases. Availability of data is achieved by spreading data across the nodes of database cluster.
CONSISTENCY: Database remains in a consistent state. If the database is consistent before a transaction, it should remain consistent after the transaction. Also known as "no adverse effects."	Read/Write functionality is available for all transactions, but it is possible to have data that is not persistent, or the data read by a query is not up to date.
ISOLATION: Each database transaction works in isolation and does not interfere with the other transactions. Also known as "parallel transactions."	SOFT STATE: Lacks immediate consistency. Data values may change over time. Data consistency is the responsibility of developers.
DURABILITY:A committed database transaction holds even if the system fails or restarts. Data saved is durable. Also known as "permanent."	EVENTUALLY CONSISTENT: Immediate consistency is not guaranteed, but at some point the database is consistent after data goes through all clusters and nodes. It's possible to read data before that final state is reached.

Data in Transit, Data at Rest

Data in Transit (or Data in Motion)

- It is data that is **traveling** from one location to another over a network.
- **Where it lives:** Being sent in an email, uploaded to a website, or moving between a user's phone and a server.
- **Security Focus:** To protect data in transit, we use **Secure Protocols** like **HTTPS** or **SSL/TLS**. This ensures that if someone "sniffs" or intercepts the data while it’s moving through the air (Wi-Fi) or through cables, they cannot see the content.

Data at Rest

- It is a data that is **stored** and not currently moving. It is sitting on a physical or digital storage device. It is stored in Hard drives, cloud storage (like Google Drive or iCloud), databases, and backup tapes.

- **Security Focus:** To protect data at rest, we use **Encryption** and **Access Controls** (passwords/permissions). If a hacker steals a hard drive, they shouldn't be able to read the files because they are encrypted.

Feature	Data at Rest	Data in Transit
State	Inactive / Stored	Active / Moving
Example	A file on your Desktop	An email you just sent
Main Risk	Physical theft or server hack	Interception or "Eavesdropping"
Primary Defense	Disk Encryption & Firewalls	HTTPS, SSL, and VPNs

Designing a Secure Database

- Based on standard database management principles designing a secure database is about building multiple layers of defense. It isn't just about a strong password; it involves how the data is stored, moved, and accessed.
- Designing a secure database is not just about a password. It is a combination of **Physical Security** (locks), **Network Security** (IP lists), **Software Hardening** (closing ports), and **Continuous Monitoring** (scripts).

Physical Security & Infrastructure Isolation

- **Server Separation**: Databases should be installed on standalone servers. This creates a "firebreak"; if a virus attacks a web application, it cannot easily spread to the database.
- **Performance Benefits**: Separate servers mean the database doesn't compete with other software for resources, making it faster.
- **Physical Guarding**: Database machines must be kept in a room under lock and key to prevent "insider threats" (disgruntled or curious employees).

Physical Security & Infrastructure Isolation

- **Availability (The 24/7 Rule)**: The room requires backup power and climate control (HVAC) to ensure the database stays online 365 days a year.
- **Device Restriction**: No electronic devices, such as phones or tablets, should be allowed inside the secure server room to prevent data theft or unauthorized recording.

Network Controls & Connection Security

- **Invited IP Address Lists**: Use the database software to create a list of "Trusted IPs".
- **Static vs. Dynamic**: Transactions should only be allowed from these static, pre-approved IP addresses.
- **Denying Connections**: The system should be designed to automatically block any connection attempt coming from an IP address not on the invited list.

System Hardening (Ports and Default Settings)

- **Closing Unused Ports**: Hackers scan for open "ports" (entry points). Any port that is not actively needed for a specific service must be closed.
- **Default Port Risks**: Common software suites (Oracle, SQL Server) come with standard, well-known ports. Hackers know exactly where to look.
- **Configuration**: You must disable these default ports or change them during setup to ensure they cannot be used to gain unauthorized entry.

Real-Time Monitoring & Auditing

- **Internal vs. External Threats**: Breaches happen via outside hackers, disgruntled employees, or negligent "insiders" who make a mistake in judgment.
- **The Human Factor (Social Engineering)**: Even a technically secure database can be compromised if an employee is tricked into giving away their credentials.
- **Event Monitoring Scripts**: You can write SQL scripts to monitor the database automatically. These scripts should alert the admin for:
  - Excessive logins or failed login attempts.
  - Password changes or account lockouts/unlocks.

Real-Time Monitoring & Auditing (cont.)

- **Auditing "Out of Zone" Activity**: Implementation of auditing identifies exactly which user is causing trouble or going "out of zone".
- **Termination Protocol**: When an employee leaves or is fired, their credentials must be suspended **immediately** to prevent them from exploiting or stealing information.

Next Lecture

- Continue the Introduction of Database Security

Research Task

1. What are the basic principles on which modern databases are built?
2. What does the isolation property of databases indicate?
3. What is meant by soft state of a database in the BASE database model?
4. In what states can data exist?
5. What are the possible ways of backing up data?”

"Normally, this project would require weeks of research and verification...but luckily there's an app for it."

References

- **Danturthi, S.** (2019). *Database and application security: A practitioner's guide*. Apress.


Any
Question