DBMS – Module – 5

Locking Techniques

Timestamp Ordering Protocol

- The Timestamp Ordering Protocol is used to order the transactions based on their Timestamps. The order of transaction is nothing but the ascending order of the transaction creation.
- The priority of the older transaction is higher that's why it executes first. To determine the timestamp of the transaction, this protocol uses system time or logical counter.
- The lock-based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But Timestamp based protocols start working as soon as a transaction is created.
- Let's assume there are two transactions T1 and T2. Suppose the transaction T1 has entered the system at 007 times and transaction T2 has entered the system at 009 times. T1 has the higher priority, so it executes first as it is entered the system first.
- The timestamp ordering protocol also maintains the timestamp of last 'read' and 'write' operation on a data.

Basic Timestamp ordering protocol works as follows:

1. Check the following condition whenever a transaction Ti issues a **Read (X)** operation:

- If W_TS(X) >TS(Ti) then the operation is rejected.
- If W_TS(X) <= TS(Ti) then the operation is executed.
- Timestamps of all the data items are updated.

2. Check the following condition whenever a transaction Ti issues a **Write(X)** operation:

- If TS(Ti) < R_TS(X) then the operation is rejected.
- If TS(Ti) < W_TS(X) then the operation is rejected and Ti is rolled back otherwise the operation is executed.

Where,

TS(TI) denotes the timestamp of the transaction Ti.

R_TS(X) denotes the Read time-stamp of data-item X.

W_TS(X) denotes the Write time-stamp of data-item X.

Advantages and Disadvantages of TO protocol:

- TO protocol ensures serializability since the precedence graph is as follows:

Image: Precedence Graph for TS ordering

- TS protocol ensures freedom from deadlock that means no transaction ever waits.
- But the schedule may not be recoverable and may not even be cascade- free.

Validation Based Protocol

Validation phase is also known as optimistic concurrency control technique. In the validation based protocol, the transaction is executed in the following three phases:

1. **Read phase:** In this phase, the transaction T is read and executed. It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database.
2. **Validation phase:** In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability.
3. **Write phase:** If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back.

Here each phase has the following different timestamps:

Start(Ti): It contains the time when Ti started its execution.

Validation (T_i): It contains the time when Ti finishes its read phase and starts its validation phase.

Finish(Ti): It contains the time when Ti finishes its write phase.

- This protocol is used to determine the time stamp for the transaction for serialization using the time stamp of the validation phase, as it is the actual phase which determines if the transaction will commit or rollback.
- Hence TS(T) = validation(T).
- The serializability is determined during the validation process. It can't be decided in advance.
- While executing the transaction, it ensures a greater degree of concurrency and also less number of conflicts.
- Thus it contains transactions which have less number of rollbacks.

Multiple Granularity

Let's start by understanding the meaning of granularity.

Granularity: It is the size of data item allowed to lock.

Multiple Granularity:

- It can be defined as hierarchically breaking up the database into blocks which can be locked.

- The Multiple Granularity protocol enhances concurrency and reduces lock overhead.
- It maintains the track of what to lock and how to lock.
- It makes easy to decide either to lock a data item or to unlock a data item. This type of hierarchy can be graphically represented as a tree.

For example: Consider a tree which has four levels of nodes.

- The first level or higher level shows the entire database.
- The second level represents a node of type area. The higher level database consists of exactly these areas.
- The area consists of children nodes which are known as files. No file can be present in more than one area.
- Finally, each file contains child nodes known as records. The file has exactly those records that are its child nodes. No records represent in more than one file.
- Hence, the levels of the tree starting from the top level are as follows:
    1. Database
    2. Area
    3. File
    4. Record

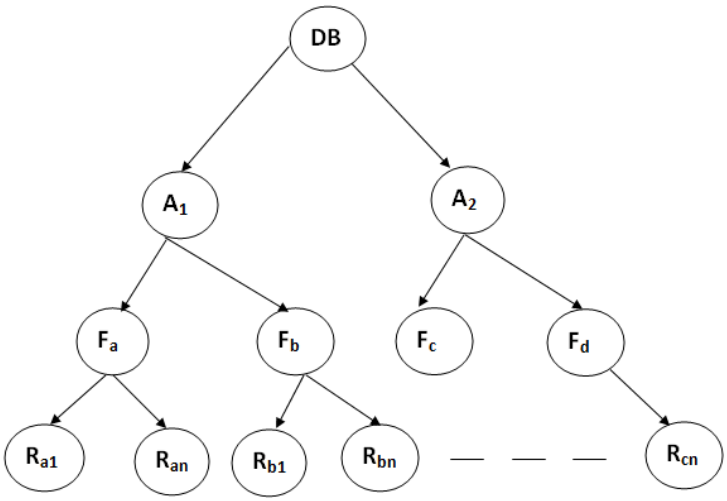

Figure: Multi Granularity tree Hierarchy

In this example, the highest level shows the entire database. The levels below are file, record, and fields.

There are three additional lock modes with multiple granularity:

Intention Mode Lock

Intention-shared (IS): It contains explicit locking at a lower level of the tree but only with shared locks.

Intention-Exclusive (IX): It contains explicit locking at a lower level with exclusive or shared locks.

Shared & Intention-Exclusive (SIX): In this lock, the node is locked in shared mode, and some node is locked in exclusive mode by the same transaction.

Compatibility Matrix with Intention Lock Modes: The below table describes the compatibility matrix for these lock modes:

	IS	IX	S	SIX	X
IS	✓	✓	✓	✓	X
IX	✓	✓	X	X	X
S	✓	X	✓	X	X
SIX	✓	X	X	X	X
X	X	X	X	X	X

It uses the intention lock modes to ensure serializability. It requires that if a transaction attempts to lock a node, then that node must follow these protocols:

- Transaction T1 should follow the lock-compatibility matrix.
- Transaction T1 firstly locks the root of the tree. It can lock it in any mode.
- If T1 currently has the parent of the node locked in either IX or IS mode, then the transaction T1 will lock a node in S or IS mode only.
- If T1 currently has the parent of the node locked in either IX or SIX modes, then the transaction T1 will lock a node in X, SIX, or IX mode only.
- If T1 has not previously unlocked any node only, then the Transaction T1 can lock a node.
- If T1 currently has none of the children of the node-locked only, then Transaction T1 will unlock a node.

Observe that in multiple-granularity, the locks are acquired in top-down order, and locks must be released in bottom-up order.

- If transaction T1 reads record R_{a9} in file F_a, then transaction T1 needs to lock the database, area A_1 and file F_a in IX mode. Finally, it needs to lock R_{a2} in S mode.
- If transaction T2 modifies record R_{a9} in file F_a, then it can do so after locking the database, area A_1 and file F_a in IX mode. Finally, it needs to lock the R_{a9} in X mode.
- If transaction T3 reads all the records in file F_a, then transaction T3 needs to lock the database, and area A in IS mode. At last, it needs to lock F_a in S mode.
- If transaction T4 reads the entire database, then T4 needs to lock the database in S mode.

Log-Based Recovery

- The log is a sequence of records. Log of each transaction is maintained in some stable storage so that if any failure occurs, then it can be recovered from there.
- If any operation is performed on the database, then it will be recorded in the log.
- But the process of storing the logs should be done before the actual transaction is applied in the database.

Let's assume there is a transaction to modify the City of a student. The following logs are written for this transaction.

- When the transaction is initiated, then it writes 'start' log.

1. <Tn, Start>

- When the transaction modifies the City from 'Noida' to 'Bangalore', then another log is written to the file.

1. <Tn, City, 'Noida', 'Bangalore' >

- When the transaction is finished, then it writes another log to indicate the end of the transaction.

1. <Tn, Commit>

There are two approaches to modify the database:

1. Deferred database modification:

- The deferred modification technique occurs if the transaction does not modify the database until it has committed.
- In this method, all the logs are created and stored in the stable storage, and the database is updated when a transaction commits.

2. Immediate database modification:

- The Immediate modification technique occurs if database modification occurs while the transaction is still active.
- In this technique, the database is modified immediately after every operation. It follows an actual database modification.

Recovery using Log records

When the system is crashed, then the system consults the log to find which transactions need to be undone and which need to be redone.

1. If the log contains the record <Ti, Start> and <Ti, Commit> or <Ti, Commit>, then the Transaction Ti needs to be redone.
2. If log contains record<T_n, Start> but does not contain the record either <Ti, commit> or <Ti, abort>, then the Transaction Ti needs to be undone.

Database Security

Security of databases refers to the array of controls, tools, and procedures designed to ensure and safeguard confidentiality, integrity, and accessibility. This tutorial will concentrate on confidentiality because it's a component that is most at risk in data security breaches.

Security for databases must cover and safeguard the following aspects:

- The database containing data.
- Database management systems (DBMS)
- Any applications that are associated with it.
- Physical database servers or the database server virtual, and the hardware that runs it.
- The infrastructure for computing or network that is used to connect to the database.

Security of databases is a complicated and challenging task that requires all aspects of security practices and technologies. This is inherently at odds with the accessibility of databases. The more usable and accessible the database is, the more susceptible we are to threats from security. The more vulnerable it is to attacks and threats, the more difficult it is to access and utilize.

Why Database Security is Important?

According to the definition, a data breach refers to a breach of data integrity in databases. The amount of damage an incident like a data breach can cause our business is contingent on various consequences or elements.

- **Intellectual property that is compromised:** Our intellectual property--trade secrets, inventions, or proprietary methods -- could be vital for our ability to maintain an advantage in our industry. If our intellectual property has been stolen or disclosed and our competitive advantage is lost, it could be difficult to keep or recover.
- **The damage to our brand's reputation:** Customers or partners may not want to purchase goods or services from us (or deal with our business) If they do not feel they can trust our company to protect their data or their own.
- **The concept of business continuity (or lack of it):** Some businesses cannot continue to function until a breach has been resolved.
- **Penalties or fines to be paid for not complying:** The cost of not complying with international regulations like the Sarbanes-Oxley Act (SAO) or Payment Card Industry Data Security Standard (PCI DSS) specific to industry regulations on data privacy, like HIPAA or regional privacy laws like the European Union's General Data Protection Regulation (GDPR) could be a major problem with fines in worst cases in excess of many million dollars for each violation.
- **Costs for repairing breaches and notifying consumers about them:** Alongside notifying customers of a breach, the company that has been breached is required to cover the investigation and forensic services such as crisis management, triage repairs to the affected systems, and much more.

Common Threats and Challenges

Numerous software configurations that are not correct, weaknesses, or patterns of carelessness or abuse can lead to a breach of security. Here are some of the most prevalent kinds of reasons for security attacks and the reasons.

Insider Dangers

An insider threat can be an attack on security from any three sources having an access privilege to the database.

- A malicious insider who wants to cause harm
- An insider who is negligent and makes mistakes that expose the database to attack. vulnerable to attacks
- An infiltrator is an outsider who acquires credentials by using a method like phishing or accessing the database of credential information in the database itself.

Insider dangers are among the most frequent sources of security breaches to databases. They often occur as a consequence of the inability of employees to have access to privileged user credentials.

Human Error

The unintentional mistakes, weak passwords or sharing passwords, and other negligent or uninformed behaviours of users remain the root causes of almost half (49 percent) of all data security breaches.

Database Software Vulnerabilities can be Exploited

Hackers earn their money by identifying and exploiting vulnerabilities in software such as databases management software. The major database software companies and open-source databases management platforms release regular security patches to fix these weaknesses. However, failing to implement the patches on time could increase the risk of being hacked.

SQL/NoSQL Injection Attacks

A specific threat to databases is the infusing of untrue SQL as well as other non-SQL string attacks in queries for databases delivered by web-based apps and HTTP headers. Companies that do not follow the safe coding practices for web applications and conduct regular vulnerability tests are susceptible to attacks using these.

Buffer Overflow is a way to Exploit Buffers

Buffer overflow happens when a program seeks to copy more data into the memory block with a certain length than it can accommodate. The attackers may make use of the extra data, which is stored in adjacent memory addresses, to establish a basis for they can begin attacks.

DDoS (DoS/DDoS) Attacks

In a denial-of-service (DoS) attack in which the attacker overwhelms the targeted server -- in this case, the database server with such a large volume of requests that the server is unable to meet no longer legitimate requests made by actual users. In most cases, the server is unstable or even fails to function.

Malware

Malware is software designed to exploit vulnerabilities or cause harm to databases. Malware can be accessed via any device that connects to the databases network.

Attacks on Backups

Companies that do not protect backup data using the same rigorous controls employed to protect databases themselves are at risk of cyberattacks on backups.

The following factors amplify the threats:

- **Data volumes are growing:** Data capture, storage, and processing continue to increase exponentially in almost all organizations. Any tools or methods must be highly flexible to meet current as well as far-off needs.
- **The infrastructure is sprawling:** Network environments are becoming more complicated, especially as companies shift their workloads into multiple clouds and hybrid cloud architectures and make the selection of deployment, management, and administration of security solutions more difficult.
- **More stringent requirements for regulatory compliance:** The worldwide regulatory compliance landscape continues to increase by complexity. This makes the compliance of every mandate more challenging.

Best use of Database Security

As databases are almost always accessible via the network, any security risk to any component or part of the infrastructure can threaten the database. Likewise, any security attack that impacts a device or workstation could endanger the database. Therefore, security for databases must go beyond the limits of the database.

In evaluating the security of databases in our workplace to determine our organization's top priorities, look at each of these areas.

- **Security for physical security:** If the database servers are on-premises or the cloud data centre, they should be placed in a secure, controlled climate. (If our server for database is located in a cloud-based data centre, the cloud provider will handle the security on our behalf.)
- **Access to the network and administrative restrictions:** The practical minimum number of users granted access to the database and their access rights should be restricted to the minimum level required to fulfil their tasks. Additionally, access to the network is limited to the minimum permissions needed.
- **End security of the user account or device**: Be aware of who has access to the database and when and how data is used. Monitoring tools for data can notify you of data-related activities that are uncommon or seem to be dangerous. Any device that connects to the network hosting the database must be physically secured (in the sole control of the appropriate person) and be subject to security checks throughout the day.
- **Security:** ALL data--including data stored in databases, as well as credential information should be secured using the highest-quality encryption when in storage and while in transport. All encryption keys must be used in accordance with the best practices guidelines.
- **Security of databases using software:** Always use the most current version of our software to manage databases and apply any patches immediately after they're released.
- **Security for web server applications and websites:** Any application or web server that connects to the database could be a target and should be subjected to periodic security testing and best practices management.
- **Security of backups:** All backups, images, or copies of the database should have the identical (or equally rigorous) security procedures as the database itself.
- **Auditing:** Audits of security standards for databases should be conducted every few months. Record all the logins on the server as well as the operating system. Also, record any operations that are made on sensitive data, too.

Data protection tools and platforms

Today, a variety of companies provide data protection platforms and tools. A comprehensive solution should have all of the following features:

- **Discovery:** The ability to discover is often needed to meet regulatory compliance requirements**.** Look for a tool that can detect and categorize weaknesses across our databases, whether they're hosted in the cloud or on-premises. It will also provide recommendations to address any vulnerabilities that are discovered.
- **Monitoring of Data Activity:** The solution should be capable of monitoring and analysing the entire data activity in all databases, whether our application is on-premises, in the cloud, or inside a container. It will alert us to suspicious activity in real-time to allow us to respond more quickly to threats. It also provides visibility into the state of our information through an integrated and comprehensive user interface. It is also important to choose a system that enforces rules that govern policies, procedures, and the separation of duties. Be sure that the solution we select is able to generate the reports we need to comply with the regulations.
- **The ability to Tokenize and Encrypt Data:** In case of an incident, encryption is an additional line of protection against any compromise. Any software we choose to use must have the flexibility to protect data cloud, on-premises hybrid, or multi-cloud environments. Find a tool with volume, file, and application encryption features that meet our company's regulations for compliance. This could require tokenization (data concealing) or advanced key management of security keys.
- **Optimization of Data Security and Risk Analysis:** An application that will provide contextual insights through the combination of security data with advanced analytics will allow users to perform optimizing, risk assessment, and reporting in a breeze. Select a tool that is able to keep and combine large amounts of recent and historical data about the security and state of your databases. Also, choose a solution that provides data exploration, auditing, and reporting capabilities via an extensive but user-friendly self-service dashboard.

Database Access Control

• Assumption: The underlying computer system (housing the database) has authenticated the user to access the system as well as granted access to the database.

• A Database Access Control System provides a specific capability that controls access to portions of the database.

• A DBMS can support a range of administrative policies:

– Centralized admin: A small number of privileged users may grant and revoke access rights

– Ownership-based admin: The owner

• Access rights for a DBMS:

– Create, Insert, Delete, Update, Read, and Write

• Access rights can be at different levels of granularity

– Entire database, individual tables, selected rows or columns within a table.

• Access rights can be determined based on the contents of a table entry.

– E.g., In a personnel database, a department manager may only be allowed to view salary info for employees in his/her department.

SQL-based Access Definition

• SQL provides two commands for managing access rights: GRANT and

REVOKE.

• GRANT command

– Used to grant one or more access rights or can be used to assign a user to a role.

– For access rights, the command can optionally specify that it applies only to a specified table.

– The TO clause specifies the user or role to which the rights are granted.

– A PUBLIC value indicates that any user has the specified access rights.

– The optional IDENTIFIED BY clause specifies a password that must be used to revoke the access rights of this GRANT command.

– The GRANT OPTION indicates that the grantee can grant this access right to other users, with or without the grant option.

GRANT	{ privileges | role }
[ON	table]
TO	{ user | role | PUBLIC }
[IDENTIFIED BY	password]
[WITH	GRANT OPTION]

Example: GRANT SELECT ON ANY TABLE TO kenny This statement permits user ‘kenny’ to query any table in the database.

Inference

• Inference is the process of performing authorized queries and deducing unauthorized information from the legitimate responses received.

• The inference problem arises when a combination of a number of data items is more sensitive than the individual items, or when a combination of data items can be used to infer data of a higher sensitivity.

• The attackers may use non-sensitive data and metadata (knowledge about correlations or dependencies among data items)

• The information transfer path by which unauthorized data is obtained is referred to as an inference channel.

• Two techniques to derive additional information:

– Analyzing functional dependencies between attributes within a table or across tables

– Merging views with the same constraints

Inference Example

Employee Table

Name	Position	Salary ($)	Department	Location
Andrew	Programmer Analyst	$80,000	Software	Jackson, MS
Robert	Quality Control	$65,000	Software	Memphis, TN
Sheela	Software Developer	$90,000	Software	Atlanta, GA
Victor	Systems Engineer	$75,000	Systems	San Jose, CA
Mary	Systems Administrator	$95,000	Systems	Seattle, WA
Ryan	Network Engineer	$87,000	Systems	Boston, MA

View V1

Name	Position
Andrew	Programmer Analyst
Robert	Quality Control
Sheela	Software Developer

View V2

Salary ($)	Location
$80,000	Jackson, MS
$65,000	Memphis, TN
$90,000	Atlanta, GA

```
CREATE View V1 AS
SELECT Name, Position
FROM Employee
WHERE Department = "Software"
```

```
CREATE View V2 AS
SELECT Salary, Location
FROM Employee
WHERE Department = "Software"
```

Statistical Database (SDB)

• SDB provides data of a statistical nature such as counts and averages

• **Two types:**

– Pure statistical database: Only stores statistical data (like census database)

– Ordinary database with statistical access: Contains individual entries

• Access using DAC, MAC and RBAC models

• Permit statistical queries based on the underlying raw data.

• The access control objective of an SDB is to provide users with the aggregate information without compromising the confidentiality of any individual entity present in the database.

– The security problem is "inference" through one or a series of statistical queries.

SDB: Characteristic Formula

• Statistics are derived from a database by means of a logical Boolean formula (referred as Characteristic formula) over the values of attributes.

• A Characteristic formula uses the operators OR, AND, and NOT (+, *, ~), written here in the increasing order of priority.

– E.g., (Sex = Male) * ((Major = CS) + (Major = EE)) specifies all male students majoring in either CS or EE

– For numerical attributes, relational operators may be used. E.g., (GP > 3.7) specifies all students whose grade point average is above 3.7.

• For simplicity, we may omit the attribute names if they are clear from context. E.g., Male * (CS + EE)

SDB Example

Database with Statistical Access with 13 Students

Name	Sex	Major	Class	SAT	GPA
Allen	Female	CS	1980	600	3.4
Baker	Female	EE	1980	520	2.5
Cook	Male	EE	1978	630	3.5
Davis	Female	CS	1978	800	4.0
Evans	Male	Bio	1979	500	2.2
Frank	Male	EE	1981	580	3.0
Good	Male	CS	1978	700	3.8
Hall	Female	IT	1979	580	2.8
Iles	Male	CS	1981	600	3.2
Jones	Female	Bio	1979	750	3.8
Kline	Female	IT	1981	500	2.5
Lane	Male	EE	1978	600	3.0
Moore	Male	CS	1979	650	3.5

Attribute A_j	Possible Values	$\lvert A_j \rvert$
Sex	Male, Female	2
Major	Bio, CS, EE, IT,...	30
Class	1978, 1979, 1980, 1981	4
SAT	310, 320, 330, ..., 790, 800	50
GPA	0.0, 0.1, 0.2, ..., 3.9, 4.0	41

Source: Table 5.3:
W. Stallings: Computer Security: Principles and Practice, 2nd ed.