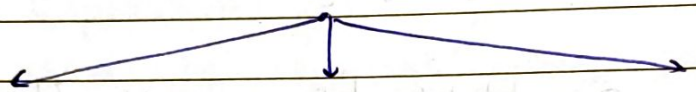
Page :
Date :

* Concurrency Control Protocols.

- Concurrency control protocols are the set of rules which are maintained in order to Solve the concurrency control problems in the database.
- It ensures atomicity, consistency, isolation, durability and serializability of the concurrent execution of database transactions.



Lock based Concurrency Control Protocol | Timestamp Concurrency Control Protocol. | Validation Based Concurrency Control Protocol.

1> Lock based Protocol

- In locked based prototocol, each transaction needs to acquire locks before they start accessing or modifying the data items.

Types of Lock.

i] Shared lock: Shared lock is also known as read lock which allows multiple transactions to read the data simultaneously.
The transaction which is holding a shared lock can only read the data item but it can not modify the data item.

ii] Exclusive lock: Exclusive lock is also known as the write lock. Exclusive lock allows a transaction to update a data item. Only one transaction can hold the exclusive lock on a data item at a time. While a transaction is holding an exclusive lock on a data item, no other transaction is allowed to acquire a shared/exclusive lock on the same data item.

	S	X
S	Yes	No
X	No	No

Request

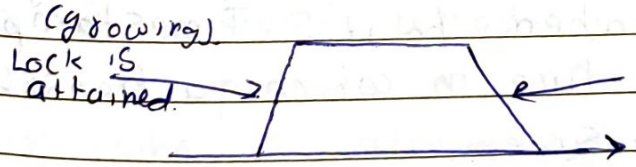
There are two kind of lock based protocol mostly used in database.

i Two Phase locking (2PL):

- It ensure strict ordering of lock acquisition and release.
- It ensure serializability by dividing transaction's execution into two phase

1] Growing phase: In this phase, the transaction start acquiring locks before performing any modification on the data items
Once a transaction acqures a lock, that lock can not be released until the transaction reaches end of the execution.

2] Shrinking Phase: In this phase, the transaction releases all the acquired locks once it performs all the modificatons on the data item.
Once the transaction starts releasing locks, it can not acquire any locks further.



ii Strinct Two-Phase locking (Strict-2PL)

- It is similar to 2PL but in this protocol the transaction cannot release lock before it commits.
- It should satisfy the basic 2PL and all exclusive locks should hold until commit or abort
- It removes irrecoverability and Cascading Problems.

* Disadvantages of 2PL:
- It may not be free from irrecoverability
- It may not be free from deadlocles
- It may not be ^ starvation
- It may not be free from cascading roll backs
- Longer wait times.
- Reduced Concurrency

iii) Rigourous 2PL:
- It is more Strinter than (Strict 2PL and 2PL)
- It should satisfy basic 2PL and all exclusive and shared locks should hold until commit or abort.
- Locks are ~~held~~ released only at the end of the transaction.
- It still has deadlock & starvation problem

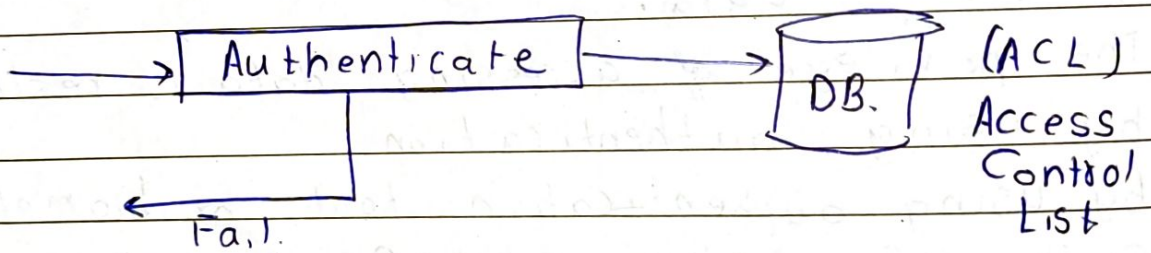
2) Timestamp Ordering Protocol
- In this protocol each transaction has a timestamp attached to it. Timestamp is nothing but the time in which a transaction enters into the system.
- Timestamp for each transaction is unique.
- The conflicting pairs of operations can be resolved by the timestamp ordering protocol therefor guaranteeing that the transactions take place in the correct order.
- Read_TS (RTS) is latest or last transaction's unique timestamp which performs read successfully.

- Write_TS (WTS) is lastest or last transactions unique timestamp (Number) which perform write successfully

* Database Security:
It means keeping sensitive information safe and prevent loss of data.
* Control measures:
* Authentication:
It is the process of confirmation that whether the user log in only according to the rights provided to him to perform the activities of database.
A particular user can login only up to his privilege but he can't access the other sensitive data.
- The privilege of accessing data is restricted by using authentication.
- By using authentication tools or biometrics such as retina and finger prints can prevent database from unauthorized useds.
- It is the process of confirming that a user who is attempting to access database is authorised to do so
- It has two types.
i] One factor Authentication: This method relies on a single type of credential to authenticate users. It is easy to implement but less Secure.
ex. Username and password.

ii] Two Step factor Authentication:
This method enhances security by requiring two independent factors to verify user's identity.
ex. User ID Pass and Otp
ex. User ID pass and biometric.

2 ~~Access Control~~ Authorization:
• Authorization is the process of determining whether a user, system, or application has permission to perform specific actions or access specific resources within a database.
• While authentication verifies identity, authorization governs access rights based on that identity.



3 Audit Control / trial
• It refers to mechanisms and policies implemented to monitor, log and review actions performed within a database System.
• It records who, when and what has been accessed from the database.
• It is a Chronological record of database activities
• Audit trail provides a clear sequence of events to support forensic investigations, compliance audits

Camlin

and security monitoring.

4 Database Encryption.

Database encryption involves converting data into a format that is unreadable without a decryption key, ensuring unauthorized users cannot access sensitive information.

Hello World! ⟶ #Ax1bto Xml12S ⟶ Hello World!
Encryption. decryption.

5 Encryption Algorithms

i) Symmetric Encryption: Uses a single key for encryption and decryption eg AES, DES.

ii) Asymmetric Encryption: Uses a public-private key pair eg. RSA.

iii) Hashing Algorithm: For one way encryption e.g (SHA-256, MD5)

If any hacker (Unauthorised User) get access to database, he is unable to utilize it.

It helps to save from cyber attacks.

Authorized users are given a decoding key to decode data.

* CIA Triad in DBMS

It is a foundational principle for securing database.

- Confidentiality: Protects sensitive data from unauthorized access.
- Integrity: Maintains the accuracy and consistency of data over its lifecycle.
- Availability: Ensures that the data and database are accessible to authorized users when needed.

Camlin

Date :

6 Granting and Revoking Privileges (DBA)
- Granting privileges allows a user with appropriate permissions to assign specific rights to other or roles.
- Privileges gives rights to user to work on or read certain portion of database.
- Syntax: GRANT privilege_list ON object to user;
- ex GRANT insert, select on table1 to user9;
- It is used to grant access rights based on rules.

- Revoking privileges removes previously granted access rights from users or roles to prevent unauthorized operations.
- Syntax REVOKE privilege_list ON object FROM user_or_roles
ex. REVOKE SELECT, INSERT ON employees FROM user1;

7. Access Control:
- It is a Security mechanism which will decide who can use Specific resource and what actions they can perform on that resource
- They are rules which tells users what type of access and to which data and what actions they can perform

* Access Control Models
Access control models define how access is granted or denied.

Page :
Date :

A Discretionary Access Control (DAC)

- In this the owner of data (resource) decides the roles to manupulate the data and the access control to the data.
- Owner determines who can access Specific resource.
- It restricts access to objects based on the identity of Subjects or groups to which the belong.
- It is an identy based access control.
- It is less Secure
- It is easy to implement.
- Users are provided access based on their identity and not using levels.
- Data owner can transfer ownership to other
- Data owner can determine the type of access given to other users.

B Mandatory Access Control (MAC)

- It is a Security model that enforces access policies for System resources based on Security labels assigned to objects.
- Access rights are regulated by a central authority based on multiple level of Security.
- It is more Secure to use.
- Users are restricted based on their power and level of hierarchy.
- It is Strict version of DAC.
- MAC is appropriate for extremely secure information Such as military or mission critical data applications.
- It may uses Access Control List (ACL) to determine which user is allowed or denied access to Specific object.

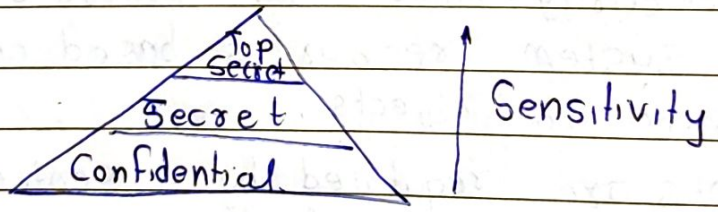
Camlin

Three levels in MAC

i] Confidential: The lowest level of classified information, Information at this level is Sensitive but not critical.
Unauthorised disclosure could cause minal damage.

ii] Secret: A mid-level classification, Information marked as Secret requires significant protection as its unauthorized disclosure could cause serious damage to national security.

iii] Top Secret: The highest level classification level. Information at this level requires the most stringent protecton.
Unauthorized disclosure can cause exceptional grave damage.



c Role Based Access Control (RBAC)

- It is highly popular and flexible security model used in organizations.
- It provides access control based on Jobtitle.
- It assigns roles to users, each rol is associated with specific permissions to perform certain tasks

- ex. In IT company different roles of users may include users such as project manager, team leader etc.
- These members will have different level of access in order of their role
- In this an administrator assigns each user one or more roles. Each role represents a set of permissions or privileges for the use.
- Large organizations with many employees use this.
- ~~El kes mar~~

* Advantages.
- Simple Management: Instead of managing permissions for individual, you manage them at the role level.
- Enhanced Security.
- Scalability

* Intrusion Detection System (IDS)
- An intrusion detection system is a security tool that monitors a computer network or systems for malicious activities or policy violations.
- It helps detect unauthorized access, potential threats, and abnormal activities by analyzing traffic and alerting administrators to take actions.
- An IDS is crucial for maintaining network security and protecting sensitive data from cyber attacks.

· The intrusion detector learning task is to build a predictive model capable of distinguishing between bad connections and good connections.
· It helps in protecting databases from attacks like SQL injection, privilege abuse, data theft and unauthorized data modifications.

* Working.
· An IDS monitors traffic on a computer network
· It analyzes the data flowing through network to look for patterns and signs of abnormal behavior.
· IDS compares the network activity to a set of predefined rules and patterns to identify any activity that might indicate an attack or intrusion.
· If IDS detects something matching rules or patterns, it sends an alert to the system admin.
· Administrator can then investigate the alert and take action to prevent any damage.

* Types of IDS
1. SBIDS (Signature Based IDS):
· It monitors all the data (Packets) traversing a network and compares them against a database signature
· It compares activities against database of known attack signatures or pattern.
ex: Detecting a specific string associated with known malware or SQL Injection.

Date :

Polymer is a long molecule formed by joing together thousands of small units by chemical bonds.

- This is useful and accurate for known attacks.
- It is ineffective against new or unknown threats
- SIDS continously monitors network traffic or system activity.

2. N IDS - Network Intrusion detection System
- It monitors inbound and outbound traffics on the network.
- It is deployed at strategic points within an organization's network to monitor incoming and outgoing traffic.
- It monitors and detects malicious and suspicious traffic coming to and going from all devices connected to the network.
- Provides real-time network wide monitoring.
- It struggles with encrypted traffic or high speed networks.
- ex. Snort, Suricata., Zeek

3 H IDS (Host IDS)
- It is able to detect anomalies network packets that originate from inside organization.
- It is installed on individual devices that are connected to internet and an organizations itsenal network.

• It detects packets that come from inside the organization that are malicious.
• It can also discover malicious threats coming from host, such as a host being infected.
• It provides detailed monitoring at System level
• It can detect attacks like privilege escalation or file tampering.
• It is resource intensive.

4. ~~Anomaly based (ABIDS~~
4. ABIDS (Anomaly based)
• It compares database against an established base line
• It monitors for deviations from established normal behaviour.
• It focuses on identifying unusual patterns that do not match System's learned baseline unlike SBIDS.
• Metrics used include network traffic volume, system usage patterns, login times etc.
• It can potentially identify unknown attacks
• Detects wide range of unexpected network activity
• Requires extensive training time to create baseline.
ex. Splunk, Wazuh, OSSEC.
ex. An organization with consistent daily traffic (eg. 500 GB per day) Suddenly experiences 2 tb outbound traffic. alerts DBA

* SQL Injection.

- SQL Injection is a security flow in web applications where attackers insert harmful SQL code through user inputs.
- This can allow them to access sensitive data, change database contents or even take control of the system.
- It occurs when malitious SQL commonds are inserted into form fields.
- SQL injection also allows attackers to perfor denial of service (DoS) attacks by overloading the server requests.

ex. txtSQL = "Select * from Users where Id = " + Id ;
attackers can concater at input like 1=1.

Select * from Users where Id = 105 or 1 = 1

It will return all users instead of 105 leaking other Users data.

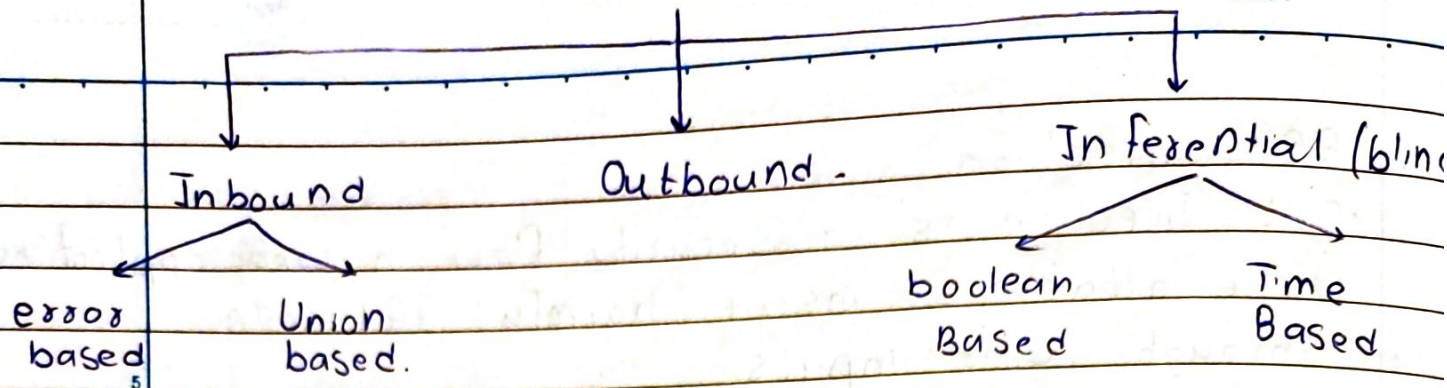
- It manupulates SQL queries through user inputs

* Prevension:

- Practice regular audid and keep monitoring the database.
- Input Validation : reject dangerous charters like '; -- etc.
- Parameterized Queries.
- Tools or Software like sqlmap for penetration testing or automated Scanning.
- Web application firewall (WAF).
- Usng limiting Previledges.

Camlin

SQL Injection



★ Inbound SQL Injection:

- It refers to attacks where malicious SQL code is injected directly into the application's input fields or parameters through application interface.
- The results of injection are reflected within the application itself.
- The attacker observes the application's response immediately.
- It allows attackers to extract sensitive information or modify database itself

i) Error based SQL Injection:

- It relies on database error messages to infer information about the database schema or structure.
- Attackers exploit error messages generated by the web applicaton to gain access to confidential data or modify the database.

ii) Union-Based SQL Injection:

- Combines the results of multiple Select queries into a single output, often used to extract data from other tables.

- It leverages the UNION operator
- This allows attackers to add their own commands to original query executing malicious code.

ex. Select a,b from table1;
↳ (modified with malisious)
Select a,b from table1 Union Select c,d from table 2;

- Both the queries must return same number of columns with same datatypes.

Outbound SQL Injection:
- It refers to attacks that extract information outside the application by forcing the database to send the data elsewhere, such as through DNS or HTTP requests.
- It uses different channel to communicate with the database instead of Application interface.
- It allows attackers to exfiltrate sensitive data from the database
- It is used in restricted enviroment where data cannot be retrieved through direct queries.
- In this data is sent to a remote server controlled by th

Inferential- ~~based~~ (blind) SQL Injection
- The attacker does not directly see the database's output or error messages but infers

information by analyzing the application's behavior or response time.
- By analyzing application's behavior, attackers can determine the success of the query.
- In this injection, attacker never see the output of the SQL queries
- It is complex and time consuming.

i] Boolean-Based Blind SQL Injection.
- In this attacker send queries that result in true or false.
ex. Select * from users where Id=1 AND 1=1;
Select * from users where Id=1 AND 1=0;
- The attacker observes whether application behaves differently (eg page loads correctly or fails.

ii] Time-Based Blind SQL Injection:
- It Exploits database functions that delay responses to infer information.
- ex. Select * from users AND Sleep(5);
- Attacks notes response time to determine whether result was true or false.
ex. Select IF(1=1, SLEEP(5), 0);
delay response by 5 sec if true.
- This is very slow as attacker has to enumerate each character by character.

Page :
Date :

* Database Recovery:
• Database Systems are subjected to failures but the data stored in them must be available as and when required.
When a database fails it must possess the facilities for fast recovery.
• Database recovery techniques are used in database management systems to restore a database to a consistent state after a failure or error has occurred.
• The main goal of recovery techniques is to ensure data integrity and consistency.

i] Rollback / Undo Recovery Technique:
The rollback / undo recovery technique is based on the principle of backing out or undoing the effects of a transaction that has not been completed successfully due to a system failure or error.
• This technique is accomplished by undoing the changes made by the transaction using the log records stored in the transaction log.
• The transaction log contains a record of all the transactions that have been performed on the database.
• System goes to the previous state.

ii] Commit Redo Recovery Technique:
The commit / redo recovery technique is based on the principle of reapplying the changes made by a transaction that has been...

completed successfully to the database.
- This technique is accomplished by using the log records stored in the transaction log to redo the changes made by the transaction that was in progress at the time of the failure or error.
- The system uses the log records to reapply the changes made by the transaction and restore the database to most recent consistent state.

iii] Checkpoint Recovery Technique:
Checkpoint Recovery is a technique used to improve data integrity and system stability, especially in databases and distributed systems.
- It entails preserving the system states at regular intervals, known as checkpoints, at which all ongoing transactions are either completed or not initiated.
- These states is kept in stable, non-volatile storage so that it can withstand crashes.
- In the event of breakdown, the system can be restored to the most recent checkpoint, which reduces data loss and down time.
- The frequency of checkpoint formation is carefully regulated to decrease system overhead.

★ Log Based Recovery: (~~Immediate Database~~)
A log is a sequential file where every database operation is recorded before being executed on the database.

Page :
Date :

each log entry typically includes.
- Transaction ID
- Operation.
- Affected Dat
- Old value
- New Value.

1] Immediate Database Update (Modification).
- In this approach, changes made by a transaction are written to the database immediately, even before the transaction commits.
- If a failure occurs, recovery is based on log
  - * Undo : Rollback any uncommitted transactions.
  - * Redo : Reapply changes made by commited transactions.

DB.

A=100
B=200

logs example :

T_1
R(A)
A = A + 100
w(A)
R(B)
B = B + 200
w(B)
commit

< T_1, Start>
< T_1, A, 100, 200>
< T_1, B, 200, 400>
< T_1, Commit >.

2] Deferred Database Modification:
In this approach, changes made by a transaction are not immediately applied to the database.
Instead
- Changes are recorded in logs
- The database is only update on transaction commits.

• It is also called No Undo/Redo recovery.
• Recovery involves only redoing commited transactions, as no uncommitted changes are written to the database
• Old data is not stored in deferred

ex

T_1	$<T_1, Start>$
R(A)	$<T_1, A, 200>$
A = A+100	$<T_1, B, 400>$
W(A)	$<T_1$ Commit$>$
R(B)	
B = B+200	
W(B)	

New Value

* Hashing:
• Hashing is a technique to quickly locate a data record in a database irrespective of the size of the database.
• For large databases indexing takes lot of time so hashing is used.
• The hashing technique utilizes an auxiliary hash table to store the data records using a hash function.

There are 2 key components in hashing.

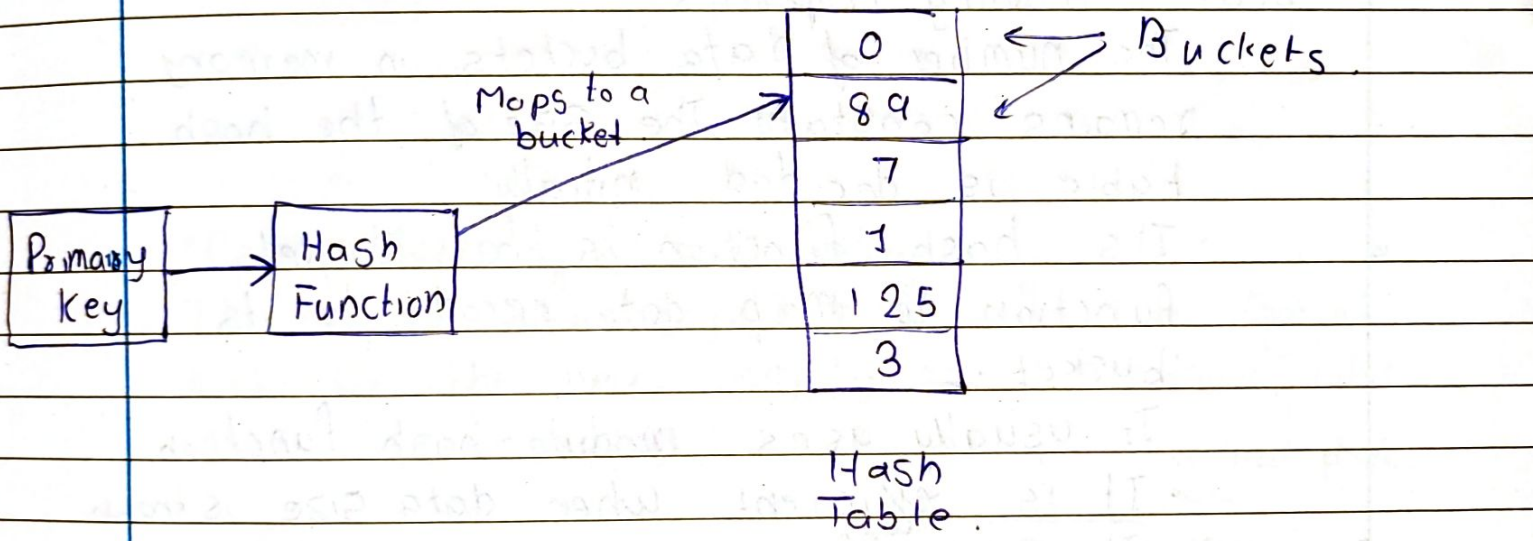
* Hash table: A hash table is an array or data structure and its size is determined by the total volume of data records present inthe database.

Each memory location in a hash-table is called a bucket or hash indice. and

Page :
Date :

* Bucket : A bucket is a memory location in the hash table that stores a data record's exact location and can be accessed through a hash function.

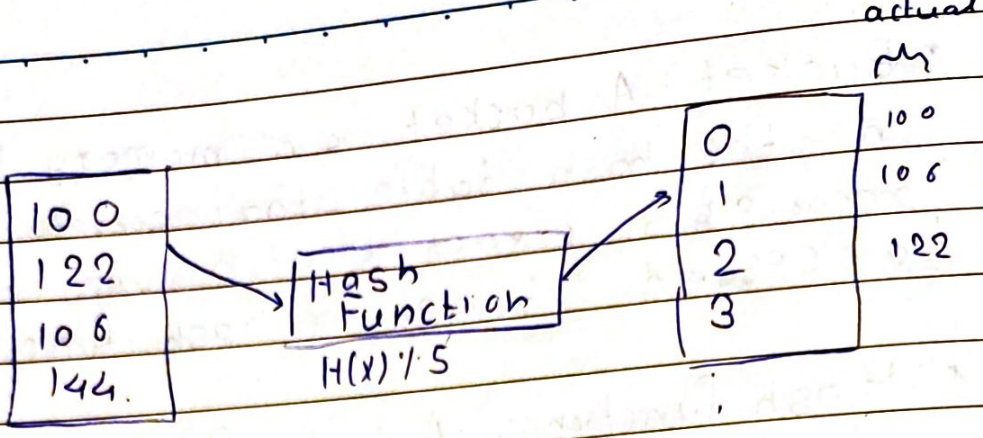
* Hash function : A hash function is a mathematical equation or algorithm that takes one data record's primary key as input and computes the hash index as output.



Hash Table.

* Types of Hashing
There are two primary hashing techniques in DBMS.

1. Static Hashing :
In static hashing, the hash function always generates the same bucket's address.
For ex, if we have a data record for employee id = 106, the hash function is mod-5 which is $H(X) \% 5$ where X is id.
Then $H(106) \% 5 = 1$.
⇒ Item must be stored or searched in 1st bucket in hash table.



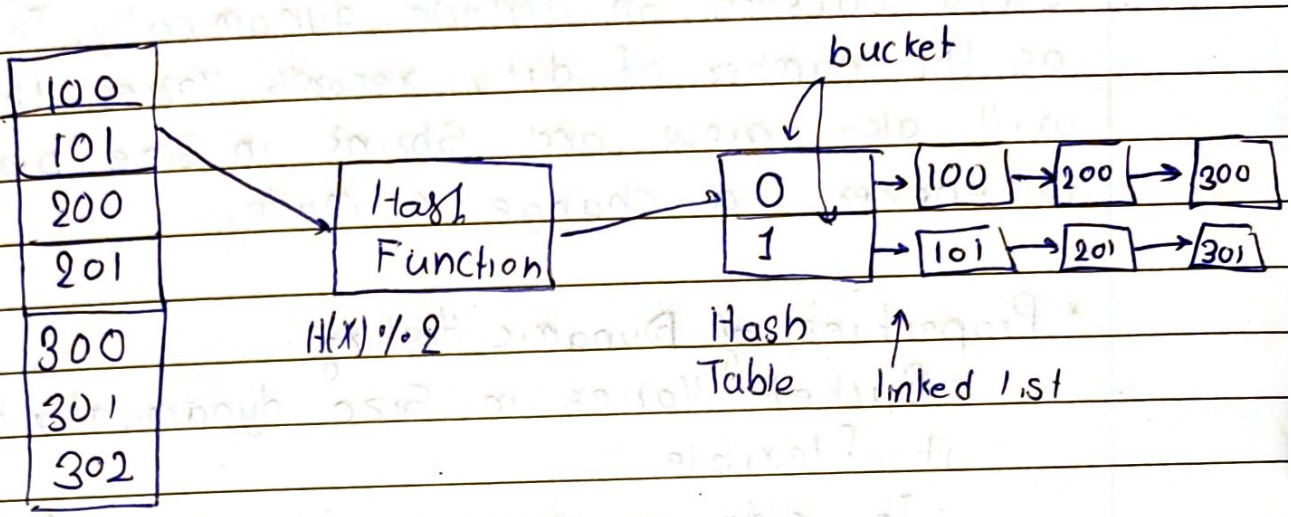
- The primary key is used as the input to the hash function and the hash function generates the output as the hash index.

* Static Hashing Properties.
  - The number of data buckets in memory remains constant. The size of the hash table is decided initially
  - Its hash function is the simplest function to map data records to its bucket.
    It usually uses modulo-hash function.
  - It is efficient when data size is known.
  - It is inefficient and inaccurate due to limited space and collision.

To resolve bucket overflow, techniques such as chaining and open addressing are used.

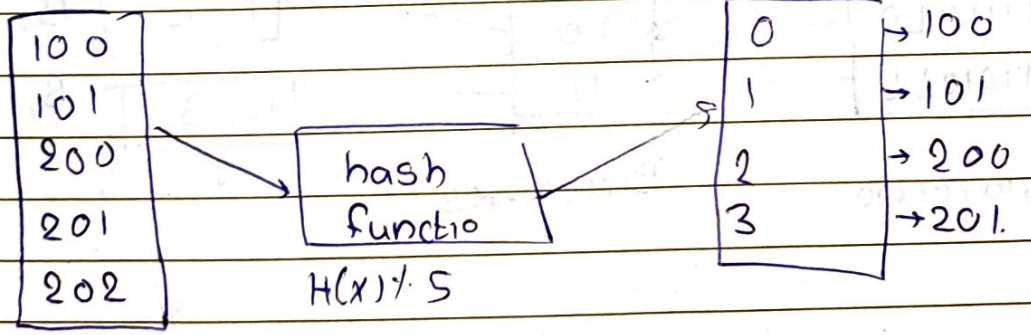
- Chaining: In this mechanism hash table is implemented using an array of type node, where each bucket is of node type and can contain a long chain of linked list to store the data records.
- So, even if a hash function generates same value, no collision is caused and

data is stored by adding new node.



- Open Addressing / Closed Hashing:

It aims to solve the problem of collision by looking out for the next empty slot available which can store data.



100 % 5 = 0 → Allot
101 % 5 = 1 → Allot
200 % 5 = 0 → Already full (Allot empty Slot 2)
201 % 5 = 1 → Already full (Allot empty Slot 3)

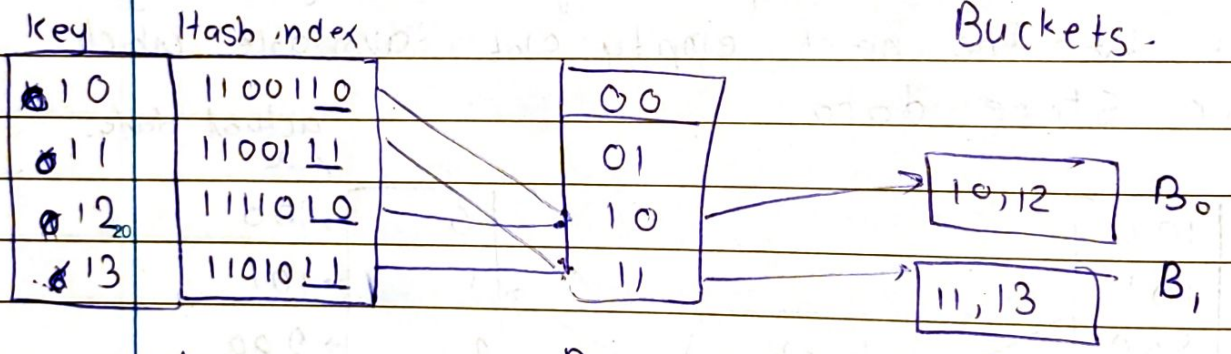
* Dynamic Hashing:

- It is also known as extendible hashing, used to handle database that frequently changes data.

Page :
Date :

sets.
This method offers us a way to add and remove data buckets on demand dynamically. This way as the number of data records varies, the buckets will also grow and shrink in size periodically whenever a change is made.

* Properties of Dynamic Hashing:
  - Bucket varies in size dynamically making it flexible
  - It aids in improving overall performance by minimizing or completely preventing collisions.
  - It consists directories which contain pointer to bucks.



* Global depth: It indicates bits used in directory.

6

It minimizes collision by expand table or Splits buckets.

- Efficient memory management

* Transaction Processing:

Transaction is a set of operations used to perform logical unit of work. It generally represent change in database

For work = transfer money from A to b (transfer 500)
Operations will be read write commit.

A = 1000
B = 2000 ⟩ -500.

R(A)
A = A - 500
w(A)
R(B)
B = B + 500
w(B)
Commit

After Commit
A = 500
B = 1500

* ACID properties

A Atomicity
C Consistency
I Isolation
D Durability.

Atomicity is either all or none. A failed transaction connot be resume but it would restart.

restart → R(A)
A = A - 500
Fail → rollback w(A)
R(B)

Page :
Date :

Consistency:
· Before transaction starts and after the transaction complete the sum of money or total amount must be same

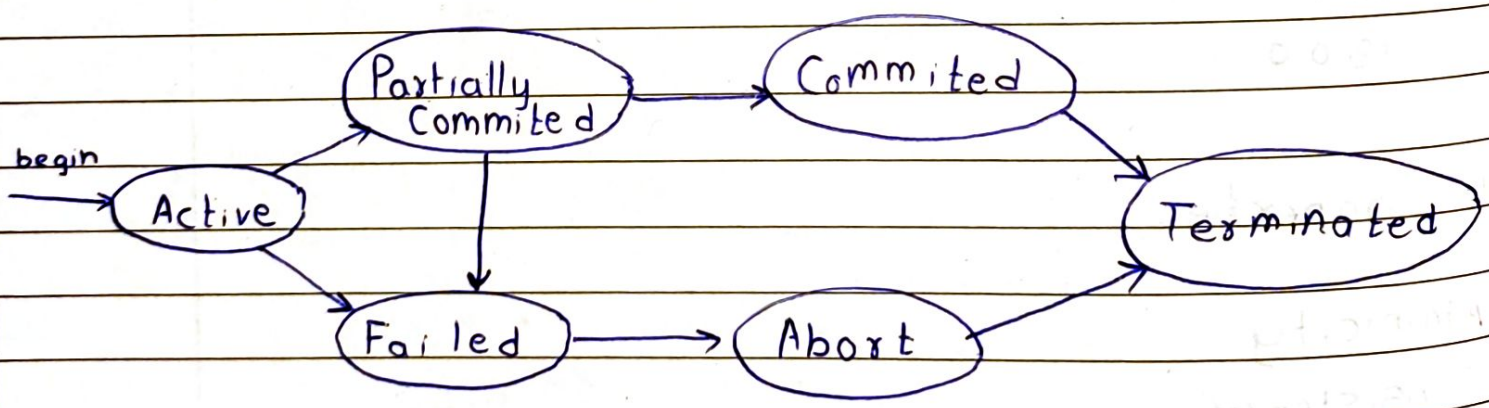
A = 1000
B = 2000
} = 3000

A = 500
B = 2500
} = 3000

Isolation:
Conversion of parallel transaction into serial Scheduling. Here conflict and view Serializability is used.

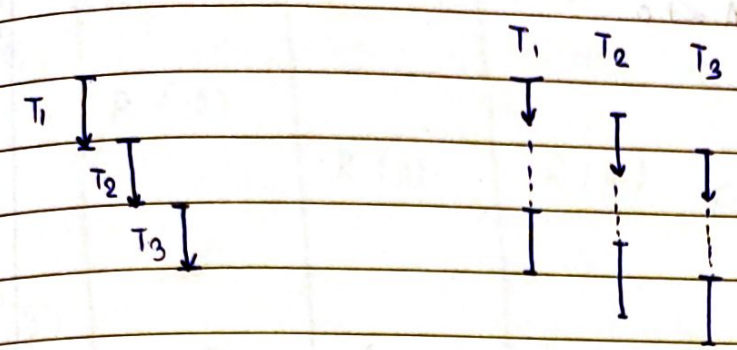
Durability:
Changes in the database should be permanent

* Transaction States:



* Scheduling
It is a chronological execution sequence of multiple transactions mainly of two types

Serial Schedule and parallel Schedule.



Serial

- Consistency ↑
- waiting time ↑
- Performance ↓

ex. Atm

Parallel

- waiting time ↓
- Performance ↑
- Consistency ↓

ex. Online banking

*Conflicts in Scheduling:

Conflict in parallel Scheduling

R	R	
R	W	Conflicts
W	R	Conflicts
W	W	Conflicts

1. Write Read Conflict (Problem) (dirty Read Problem):

Suppose database has A = 100

T_1	T_2
R (A)	
A = A - 50	
W (A)	
	R (A)
	A = A - 20
	W (A)
Commit	
	Commit

R (A)
A = A - 20
W (A)
Commit

Page :
Date :

Read Write Conflict (Unrepeatable read):

A = 10

T_1	T_2
R(A)	
	R(A)
	A = A - 10
	w(A)
	commit → A = 0
R(A).	

* Irrecoverable Schedule

A = 100

T_1	T_2	
R(A)		
w(A)		
	R(A)	A = 50
	w(A)	A = A - 20
	commit	→ A = 30 → A = 100
w(A)		
:		
fail.		

A = A - 50

} lost (Do irreco

If T_1 gets failed at last it gets restarted by T_2 progress is lost.

* Cascading VS Cascadless Schedule.

S

T_1	T_2	T_3	T_4
R(A)			
W(A)			
:	R (A)		
:	:	R (A)	R (A)
		:	:
R (B)			
W(B)			
:			
:			
Fail		abort	

S

T_1	T_2	T_3	T_4
R(A)			
W(A)			
	R(A) ✗	R(B) ✓	
commit		R(C) ✓	
	R(A) ✓		

We we read after write on same data and initial task gets failed the the task using read is aborted too.

In Cascadless if another task wants to use the same data item which is already being used in write by some other task then it is not allowed until it is commited.

Cascade

If there is a read after write operation on same account but in different transaction then after failure you have to abort all the transaction which comes after write. Here CPU utilization is not proper and performance is less.

Cascadless

You are not allowed to read data after write on same account in different transaction until it commits or fail or abort.

T_1	T_2
R(A)	
	R(A)
w(A)	
	w(A)

write operation after write causes conflict still it is allowed in cascadless Schedule

* Strict Schedule
In this after write operation the transaction must commit or fail.

T_1	T_2
R(A)	
	R(A)
w(A)	
commit	

* Serializability:
It is checking Schedules has ability to become Serializable or not

T_1	T_2
R(A)	
w(A)	
	R(A)
	w(A)

T_1	T_2
	R(A)
	w(A)
R(A)	
w(A)	

Serial

T_1	T_2
R(A)	
	R(A)
	W(A)
W(A)	

Parallel

* Conflict and View Serializability are two methods to convert parallel into Serial

S

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	

S1

T_1	T_2
R(A)	
W(A)	
R(B)	
	R(A)
	W(A)

Check if they are conflict equivalent

R(A) R(A) → No Conflict

R(A) W(A)
W(A) R(A)
W(A) W(A)
} Conflict

R(A) R(B) → NC.

R(A) W(B)
W(A) R(B)
W(A) W(B)
} NC.

In conflict serializability method the adjacent pairs of transaction are non conflict they we can swap their position

S

T_1	T_2
R (A)	
W (A)	
	R (A)
	W (A)
R (B)	

← A.C So swap.

S

T_1	T_2
R (A)	
W (A)	
	R (A)
R (B)	
	W (A)

← Non Conflict So swap

S

T_1	T_2
R (A)	
W (A)	
R (B)	
	R (A)
	W (A)

$\Rightarrow S \equiv S'$

Q

T_1	T_2	T_3
R (x)		
	R (y)	
	R (x)	
R (y)		
R (z)		
		W (y)
		W (z)

Question 5.

T_1	T_2	T_3
R(x)		R(y)
		R(x)
	R(y)	
	R(z)	
		w(y)
	w(z)	
R(z)		
w(x)		
w(z).		

Check whether the given Schedule is conflict Serializable or not. If yes in what flow it is Serializable.

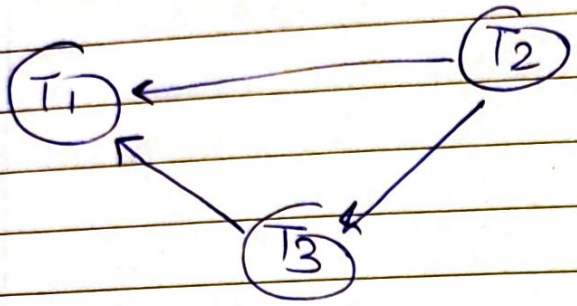
S_1)

T_1	T_2	T_3
		R(y)
R(x)		
		R(x)
	R(y)	
	R(z)	
		w(y)
	w(z)	
R(z)		
w(z)		
w(z).		

S_2)

T_1	T_2	T_3
		R(y)
		R(x)
R(x)		

Step 1: Precedence graph:
Condition: To draw precidence graph
• Check Conflict pairs in other transaction and draw edge.



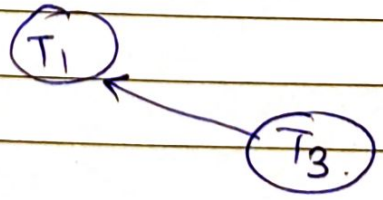
Check if the graph is cycle or not
There is not cycle (loop) in precedence graph.
⇒ It is conflict Serializable. or it can be converted into Serial

* Rule to check flow.
Check graph in-degree = 0 first.

Trans	Indegree
T_2	0
T_3	1
T_1	2

Now remove T_2.

$T_2 \rightarrow T_3 \rightarrow T_1$



Tran	In degree
T_3	0
T_1	1

$T_2 \rightarrow T_3 \rightarrow T_1$

It is the flow of this Scedule after converting

If loop is found we cant say if it is Serializable or not. We can just say it is not conflict Serializable.

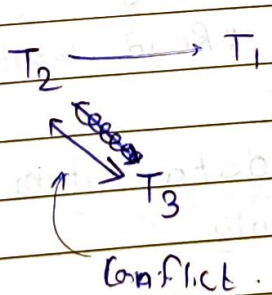
When loop occurs in conflict Serializability then we cant say it is serializable or not.
After loop occurs we have to check by view serializability.

* View Serializability.

T_1	T_2	T_3
R(A)		
	R(A)	
		R(B)
W(A)		
	R(C)	
	R(B)	
	W(B)	
		W(C)

	A	B	C
1. Initial read	T_1, T_2	T_3, T_2	T_2
2. Update	T_1	T_2	T_3
3. Final Update.	T_1	T_2	T_3

A = $T_2 \longrightarrow T_1$
B = $T_3 \longrightarrow T_2$
C = $T_2 \longrightarrow T_3$



It is not Serializable.

Ex.

T_1	T_2	T_3
R (A)		
	R (A)	
		R (B)
W (A)		
	R (C)	
	R (B)	
	W (B)	
W (C)		

* Concurrency Control Protocol
It states how to make schedule serializable.
It can be achieved by locking protocols.
There are many locking protocols as shared exclusive protocol, 2PL, Strict 2PL, regourous 2PL, and timestamp ordering protocol.
From this shared exclusive is basic and important. as well as time stamp

* Shared Exclusive locking
- Shared Lock (S) : Read
- Exclusive Lock (X) : Read, write

* Shared Lock
If transaction locks data item in shared mode then allow to read only

* Exclusive Lock
It is denoted by X
If transaction locks data item in exclusive mode then read and write both are allowed

Grant	Request S	Request X
S	Yes	No
X	No	No

* Drawbacks of all protocol

- May not be sufficient to produce only serializable schedule
- May not free from irrecoverability.
- May not free from deadlock.
- May not free from starvation.

* 2PL : 2 Phase Locking protocol :

- It solves the drawback of

- Growing : In growing phase locks are acquired and no locks are released.
- Shrinking : In shrinking phase locks are released and no locks are acquired

	T_1	T_2
growing ↓	X (A)	
	R (A)	
	W (A)	
	:	X (A) - waiting.
	:	:
	:	S (C)
	S (C)	
	:	
	:	
shrinking ↓	U (A)	

Lock Point:

It ensures Serializability with the help of lock point which is between last lock and unlock. position.

Drawbacks (Problems in 2PL)
- may not free from irrecoverability.
- may not free from deadlocks
- may not free from Starvation.
- may not fre from cascading rollback.

* Strict 2PL:
It should satisfy the basic 2PL and all exclusive locks should hold untill commit or abort
It removes irrecoverability and coscading problem

* Rigourous 2PL:
- It is more stricter
- It should satisfy basic 2PL and all shared and exclusive locks should hold untill commit or abort
- Both have deadlock and starvation problem.

* Time Stamp Ordering Protocal:
- Timestamp ordering protocol, here unique value is assigned to every transaction. It tells the order when transaction enter into System
- Read TS (RTS) is latest or last transaction number which perform read successfully.
- Write TS (WTS) is latest or last transaction number which perform write successfully.

Database recovery → Shadow paging
→ log based

Timestamp	100	200	300
	T_1	T_2	T_3
	R(A)		
		R(A)	
			R(A
		w(A)	
			w(A)
	w(A)		

RTS(A) = 300

WTS(A) = 100

★ Rules

1. Transaction T_i issues a read (A) operation
   A. if WTS(A) > TS(T_i), Rollback T_i
   B. otherwise execute R(A) operation
      set RTS(A) = max {RTS(A), TS(T_i)}

2. Transaction T_i issues a write (A) operation.
   A. if RTS(A) > TS(T_i) then rollback T_i
   B. if WTS(A) > TS(T_i) then rollback T_i
   C. otherwise execute write(A) operation
      set WTS(A) = TS(T_i)

100	200
T_1	T_2
	R(A)
W(A)	

Not Allowed rollback (as per rule)

100	200
T_1	T2
R(A)	
	W(A) ✓ Allowed

Older executing first

Younger executing first

	100	200	300
	T_1	T_2	T_3
1 →	R(A)		
2		R(B)	
3	W(C)		
4			R(B)
5	R(C)		
6		W(B)	
7			W(A)

	A	B	C
RTS	100	~~0~~ 200 300	~~0~~ 100
WTS	~~0~~ 300	~~0~~ 300	~~0~~ 100

Check $WTS(B) > TS(T_1)$

Step 1: $0 \not> 100$, no rollback.
execute R(A)

Step 2: $WTS(B) > TS(T_2)$
$0 \not> 200$, no rollback
execute R(B)

Step 3: $RTS(C) > TS(T_1) \Rightarrow 0 > 100$ ✗
$WTS(C) > TS(T_1) \Rightarrow 0 > 100$ ✗
$\Rightarrow$ execute W(C).

Step 4: $WTS(B) > TS(T_3) \Rightarrow 0 > 300$ ✗
$\Rightarrow$ execute R(B)

Step 5: $WTS(C) > TS \Rightarrow 100 > 100$ ✗
execute R(C).

Step 6: $RTS(A) > TS(T_2) \Rightarrow 300 > 200$ ✓

Rollback T2.

Step 7: $RTS(A) > TS(T_3) \Rightarrow 100 > 300$ ✗

$WSS(A) > TS(T_3) \Rightarrow$

* Database Security:

- Authentication
- Authorisation
- Audit control / trial
- Database encryption
- Granting & Revoking priviledges
- Access control.

* Database Authentication:

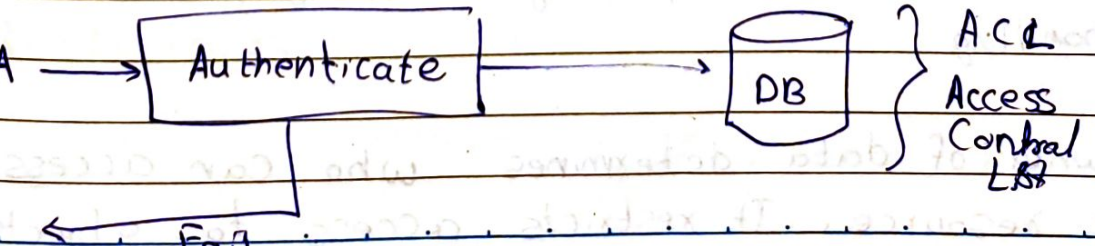
It is a process of confirming that a user who is attempting to access into a database is authorised to do so

- two way.

It contains Single factor authentication and two-Step factor authentication.

* Database Authorisation:

It is a process where database manager gets information about authenticated users.



Audit control

- It is record showing who, when and what has been accessed from database for Security relevant data.
- It is chronological.

Database encryption

CIA

Confidentiality — Integrity — Availability.

*Granting and revoking priviledges.

* Access Control: It is a Security mechanism which will decide who can use specific resource and what actions they can perform on that resource.
- They are rules which user has what type of access to which data and what action they are authorised to perform.

* Access Control Model

1. DAC: Discretionary Access Control.
2. MAC: Mandatory Access Control.
3. RBAC: Role based Access Control.

DAC: The owner of data decides the roles to manupulate the data and the access control to the
- This model is used specially in confidencial data handling

- The owner of data determines who can access specific resource. It restricts access to objects based on the identity of subjects or groups to which they belong,

camlin

- Data owner can transfer ownership to other
- Data owner can determine the type of access given to other users.

* MAC : • Access rights are regulated by a central authority based on multiple level of Security.
- It is a Strict version of DAC.
- MAC is appropriate for extremely Secure information such as military applications, or mission critical data application.
- Three levels of
  - Secret
  - Top Secret
  - Confidential.

* RBAC - Access based on Job title
ex. in it company different roles of users may include Such as project manger team leader, etc.
- These members will have different level of access in order to perform functions. based on their role.

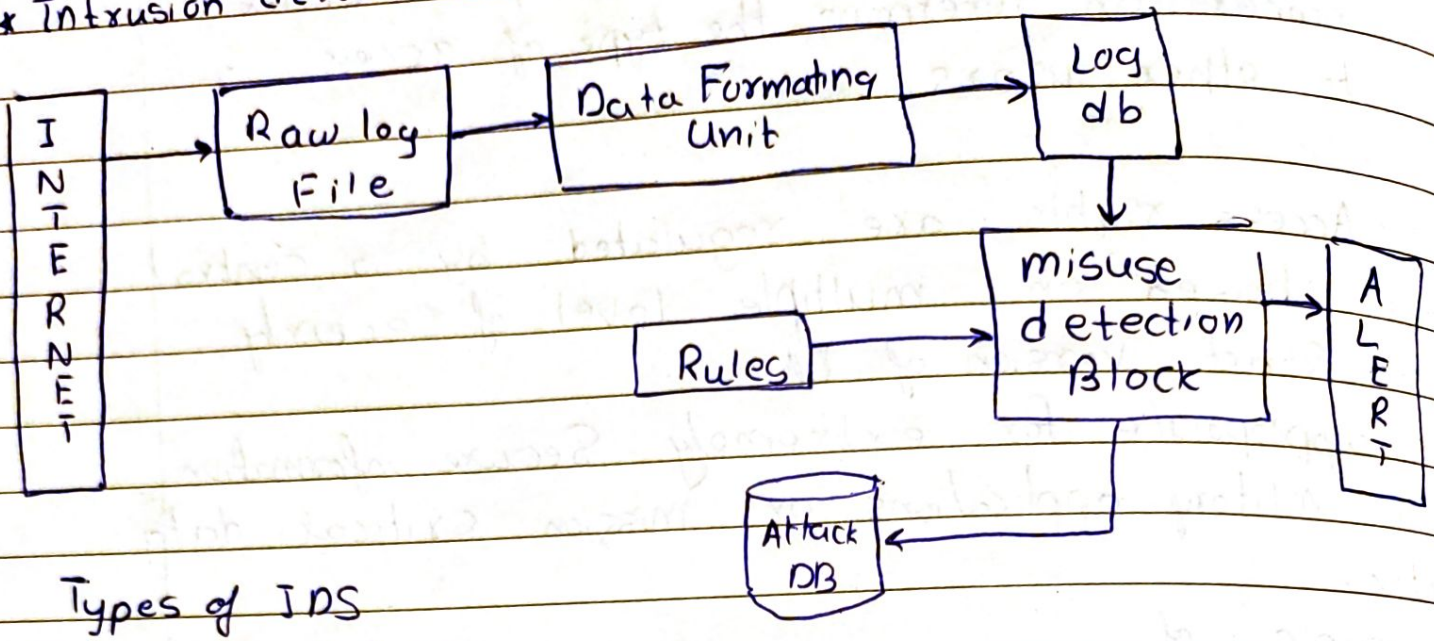
* Intrusion detection System.

Intruder ⟶ Intrusion ⟶ IDS

↓

Outsider (Masquerader)
Insider (Misfeasor)

* Intrusion detection (IDS)



Types of IDS

There are 4 types

· SBIDS, HIDS, NIDS, ABIDS

* SBIDS : Signature Based Intrusion detection System
* NIDS - Network Intrusion detection System
* HIDS - Host intrusion detection System.
* ABIDS - Anomaly based Intrusion detection System.

Signature Based IDS: It monitors all the data packets traversing a network and compares them against a database signature.

~~Net~~

Network IDS : It can monitor inbound and outbound traffic on the network

Host IDS : It is able to detect anomalies network packets that originate from inside the organ

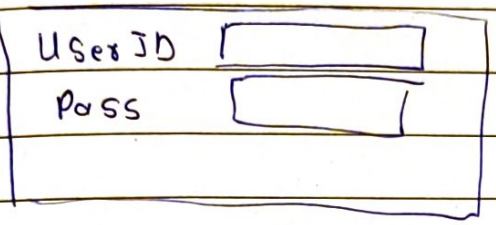
Anomaly based.

It compares database against an establish base line

SQL Injection :

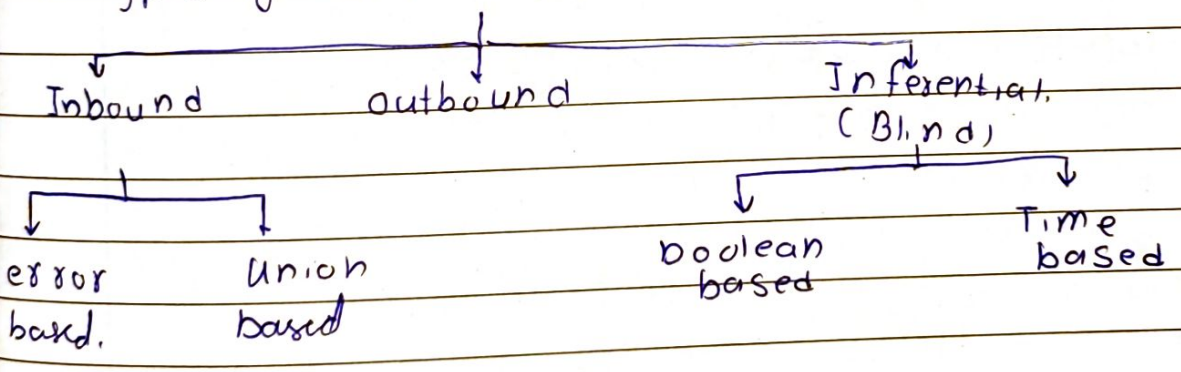
It is one type of web attack, where attacker is trying to insert Something to damage or misuse or distroy or alter Valuable data into the database.

Occurs when malitious SQL statements are inserted into form fields.



* Prevension:
- Practice regular audit and keep monitering the database
- Tools and Software to prevent

* Types of SQL injection.



SQL query, relational algebra, normalization, lossy lossless decomposition, Indices, b tree, Serializability, locking based protocols.
- Hashing, Recovery.