**CONCURRENT EXECUTION OF TRANSACTION**

In the transaction process, a system usually allows executing more than one transaction simultaneously. This process is called a concurrent execution.

**Advantages of concurrent execution of a transaction**

1. Decrease waiting time or turnaround time.

2. Improve response time

3. Increased throughput or resource utilization.

**Problems with Concurrent Execution**

In a database transaction, the two main operations are READ and WRITE operations. So, there is a need to manage these two operations in the concurrent execution of the transactions as if these operations are not performed in an interleaved manner, and the data may become inconsistent. So, the following problems occur with the Concurrent Execution of the operations: 1: Lost Update Problems (W - W Conflict)

2. Dirty Read Problems (W-R Conflict)

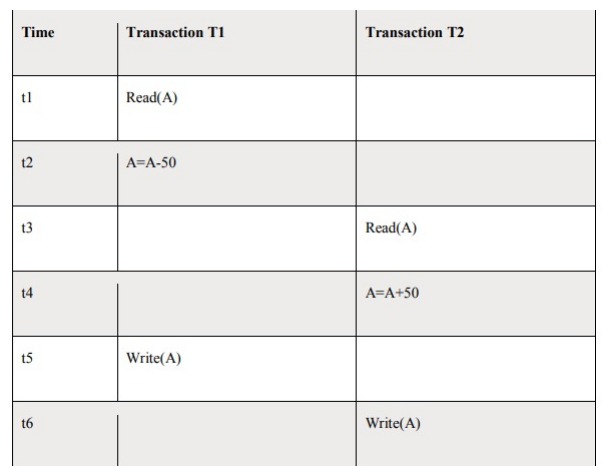
3. Unrepeatable Read Problem (W-R Conflict)

**1. Lost update problem (Write – Write conflict)**

This type of problem occurs when two transactions in database access the same data item and have their operations in an interleaved manner that makes the value of some database item incorrect.

If there are two transactions T1 and T2 accessing the same data item value and then update it, then the second record overwrites the first record.

Example: Let’s take the value of A is 100



Here,

At t1 time, T1 transaction reads the value of A i.e., 100.

• At t2 time, T1 transaction deducts the value of A by 50.

• At t3 time, T2 transactions read the value of A i.e., 100.

• At t4 time, T2 transaction adds the value of A by 150.

• At t5 time, T1 transaction writes the value of A data item on the basis of value seen at time t2 i.e., 50.

• At t6 time, T2 transaction writes the value of A based on value seen at time t4 i.e., 150.

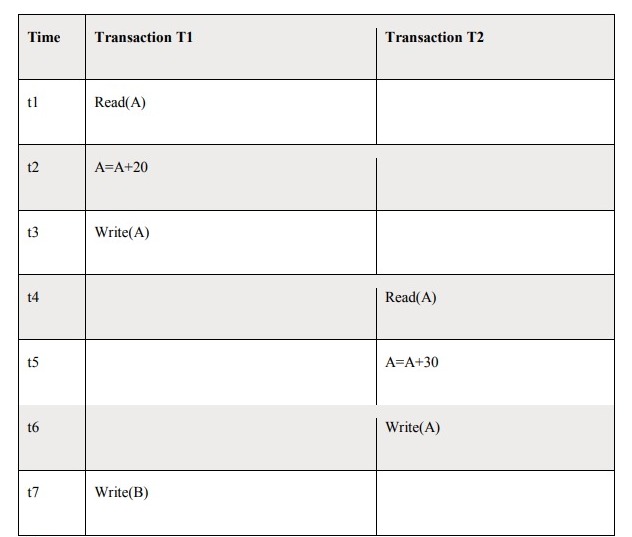
• So at time T6, the update of Transaction T1 is lost because Transaction T2 overwrites the value of A without looking at its current value.

• Such type of problem is known as the Lost Update Problem.

**Dirty read problem (W-R conflict)**

This type of problem occurs when one transaction T1 updates a data item of the database, and then that transaction fails due to some reason, but its updates are accessed by some other transaction. Example:

Let’s take the value of A is 100



Here,

At t1 time, T1 transaction reads the value of A i.e., 100.

• At t2 time, T1 transaction adds the value of A by 20.

• At t3 time, T1transaction writes the value of A (120) in the database.

• At t4 time, T2 transactions read the value of A data item i.e., 120.

• At t5 time, T2 transaction adds the value of A data item by 30.

• At t6 time, T2transaction writes the value of A (150) in the database.

• At t7 time, a T1 transaction fails due to power failure then it is rollback according to atomicity property of transaction (either all or none).

• So, transaction T2 at t4 time contains a value which has not been committed in the database.

The value read by the transaction T2 is known as a dirty read.

**Unrepeatable read (R-W Conflict)**

It is also known as an inconsistent retrieval problem. If a transaction T1 reads a value of data item twice and the data item is changed by another transaction T2 in between the two read operation. Hence T1 access two different values for its two read operation of the same data item. Example: Let’s take the value of A is 100

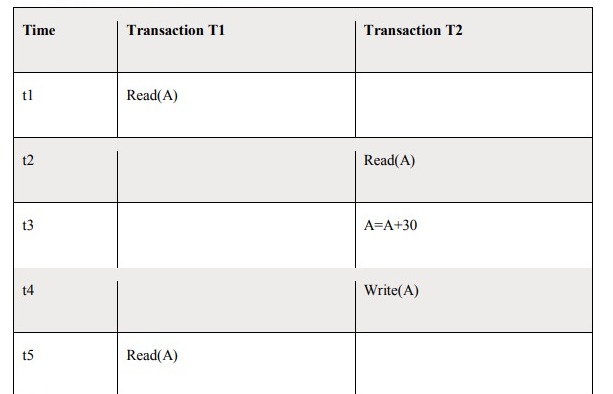
Here,

At t1 time, T1 transaction reads the value of A i.e., 100.

• At t2 time, T2transaction reads the value of A i.e., 100.

• At t3 time, T2 transaction adds the value of A data item by 30.

• At t4 time, T2 transaction writes the value of A (130) in the database.



• Transaction T2 updates the value of A. Thus, when another read statement is performed by transaction T1, it accesses the new value of A, which was updated by T2.

Such type of conflict is known as R-W conflict.

**CONCURRENCY CONTROL**

Concurrency Control is the working concept that is required for controlling and managing the concurrent execution of database operations and thus avoiding the inconsistencies in the database. Thus, for maintaining the concurrency of the database, we have the concurrency control protocols.

**Concurrency Control Protocols**

The concurrency control protocols ensure the atomicity, consistency, isolation, durability and serializability of the concurrent execution of the database transactions. Therefore, these protocols are categorized as:

* Lock Based Concurrency Control Protocol
* Time Stamp Concurrency Control Protocol
* Validation Based Concurrency Control Protocol

**Lock-Based Protocol**

In this type of protocol, any transaction cannot read or write data until it acquires an appropriate lock on it. There are two types of lock:

**1. Shared lock:**

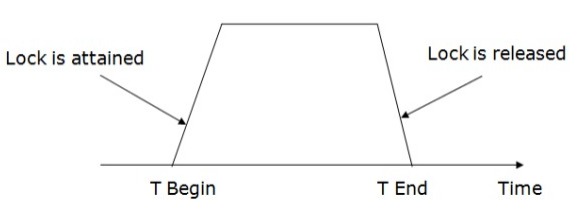
* It is also known as a Read-only lock. In a shared lock, the data item can only read by the transaction.
* It can be shared between the transactions because when the transaction holds a lock, then it can't update the data on the data item.

**2. Exclusive lock:**

* In the exclusive lock, the data item can be both reads as well as written by the transaction.
* This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously.

**TWO-PHASE LOCKING (2PL)**

* The two-phase locking protocol divides the execution phase of the transaction into three parts. o In the first part, when the execution of the transaction starts, it seeks permission for the lock it requires.
* In the second part, the transaction acquires all the locks. The third phase is started as soon as the transaction releases its first lock.
* In the third phase, the transaction cannot demand any new locks. It only releases the acquired locks.



There are two phases of 2PL:

**Growing phase:**

In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released.

**Shrinking phase:**

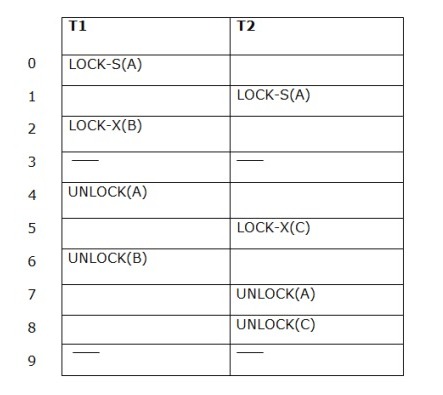
In the shrinking phase, existing lock held by the transaction may be released, but no new locks can be acquired.

In the below example, if lock conversion is allowed then the following phase can happen:

1. Upgrading of lock (from S (a) to X (a)) is allowed in growing phase.

2. Downgrading of lock (from X (a) to S (a)) must be done in shrinking phase.

Example:



The following way shows how unlocking and locking work with 2-PL.

Transaction T1:

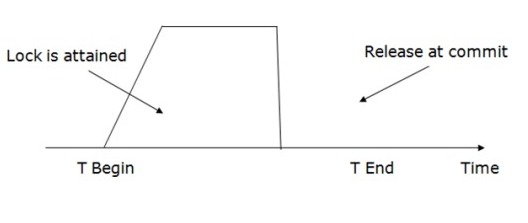
* Growing phase: from step 1-3
* Shrinking phase: from step 5-7
* Lock point: at 3

Transaction T2:

* Growing phase: from step 2-6
* Shrinking phase: from step 8-9
* Lock point: at 6

**4. Strict Two-phase locking (Strict-2PL)**

* The first phase of Strict-2PL is similar to 2PL. In the first phase, after acquiring all the locks, the transaction continues to execute normally.
* The only difference between 2PL and strict 2PL is that Strict-2PL does not release a lock after using it.
* Strict-2PL waits until the whole transaction to commit, and then it releases all the locks at a time.
* Strict-2PL protocol does not have shrinking phase of lock release



**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.

**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) denote 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.

**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 (Ti):** 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.

**THOMAS WRITE RULE**

Thomas Write Rule provides the guarantee of serializability order for the protocol. It improves the Basic Timestamp Ordering Algorithm. The basic Thomas write rules are as follows:

* If TS(T) < R\_TS(X) then transaction T is aborted and rolled back, and operation is rejected.
* If TS(T) < W\_TS(X) then don't execute the W\_item(X) operation of the transaction and continue processing.
* If neither condition 1 nor condition 2 occurs, then allowed to execute the WRITE operation by transaction Ti and set W\_TS(X) to TS(T).

**MULTIPLE GRANULARITY**

Let's start by understanding the meaning of granularity.

**Granularity:**

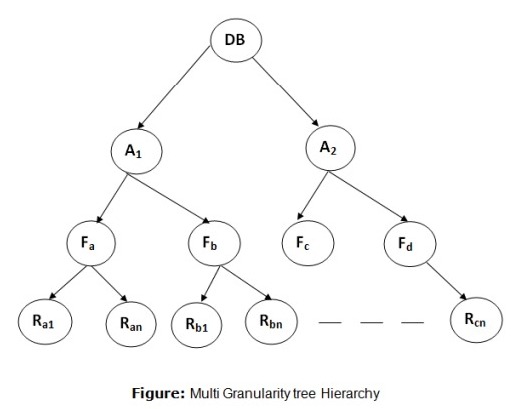
It is the size of data item allowed to lock.

**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.
* 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:

* Database
* Area
* File
* Record



**Recovery and Atomicity:**

When a system crashes, it may have several transactions being executed and various files opened for them to modify the data items. But according to ACID properties of DBMS, atomicity of transactions as a whole must be maintained, that is, either all the operations are executed or none. Database recovery means recovering the data when it gets deleted, hacked or damaged accidentally.

Atomicity is must whether is transaction is over or not it should reflect in the database permanently or it should not affect the database at all.

When a DBMS recovers from a crash, it should maintain the following −

* It should check the states of all the transactions, which were being executed.
* A transaction may be in the middle of some operation; the DBMS must ensure the atomicity of the transaction in this case.
* It should check whether the transaction can be completed now or it needs to be rolled back.
* No transactions would be allowed to leave the DBMS in an inconsistent state.

There are two types of techniques, which can help a DBMS in recovering as well as maintaining the atomicity of a transaction −

* Maintaining the logs of each transaction, and writing them onto some stable storage before actually modifying the database.
* Maintaining shadow paging, where the changes are done on a volatile memory, and later, the actual database is updated.

**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.

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 with Concurrent Transactions**

Concurrency control means that multiple transactions can be executed at the same time and then the interleaved logs occur. But there may be changes in transaction results so maintain the order of execution of those transactions. During recovery, it would be very difficult for the recovery system to backtrack all the logs and then start recovering. Recovery with concurrent transactions can be done in the following four ways.

1. Interaction with concurrency control

2. Transaction rollback

3. Checkpoints

4. Restart recovery

**Interaction with concurrency control:**

In this scheme, the recovery scheme depends greatly on the concurrency control scheme that is used. So, to rollback a failed transaction, we must undo the updates performed by the transaction.

**Transaction rollback:**

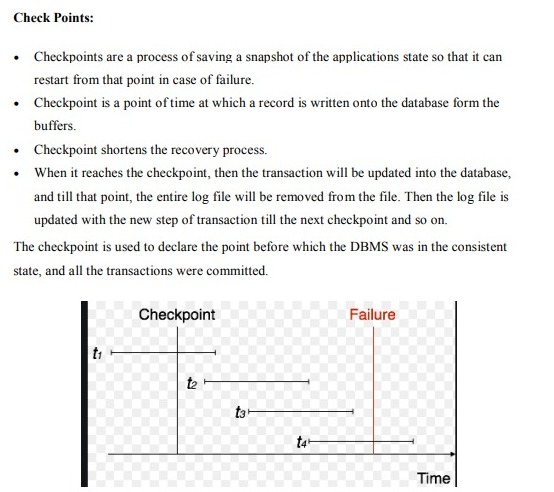
In this scheme, we rollback a failed transaction by using the log. The system scans the log backward a failed transaction, for every log record found in the log the system restores the data item.

**Checkpoints:**

* A checkpoint is a process of saving a snapshot of the applications state so that it can restart from that point in case of failure.
* Checkpoint is a point of time at which a record is written onto the database form the buffers. Checkpoint shortens the recovery process.
* When it reaches the checkpoint, then the transaction will be updated into the database, and till that point, the entire log file will be removed from the file. Then the log file is updated with the new step of transaction till the next checkpoint and so on.
* The checkpoint is used to declare the point before which the DBMS was in the consistent state, and all the transactions were committed.

**Restart recovery:**

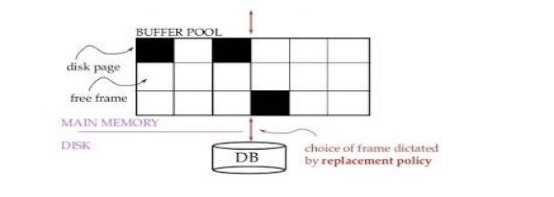
* When the system recovers from a crash, it constructs two lists.
* The undo-list consists of transactions to be undone, and the redo-list consists of transaction to be redone.
* The system constructs the two lists as follows: Initially, they are both empty. The system scans the log backward, examining each record, until it finds the first checkpoint record.



**BUFFER MANAGEMENT**

The buffer manager is the software layer that is responsible for bringing pages from physical disk to main memory as needed. The buffer manages the available main memory by dividing the main memory into a collection of pages, which we called as buffer pool. The main memory pages in the buffer pool are called frames.

* Data must be in RAM for DBMS to operate on it!
* Buffer manager hides the fact that not all data is in RAM.



**Buffer Manager**

* A Buffer Manager is responsible for allocating space to the buffer in order to store data into the buffer.
* If a user request a particular block and the block is available in the buffer, the buffer manager provides the block address in the main memory.
* If the block is not available in the buffer, the buffer manager allocates the block in the buffer.
* If free space is not available, it throws out some existing blocks from the buffer to allocate the required space for the new block.
* The blocks which are thrown are written back to the disk only if they are recently modified when writing on the disk.
* If the user requests such thrown-out blocks, the buffer manager reads the requested block from the disk to the buffer and then passes the address of the requested block to the user in the main memory.
* However, the internal actions of the buffer manager are not visible to the programs that may create any problem in disk-block requests. The buffer manager is just like a virtual machine

**Failure with Loss of Nonvolatile Storage**

**Loss of Volatile Storage**

A volatile storage like RAM stores all the active logs, disk buffers, and related data. In addition, it stores all the transactions that are being currently executed. What happens if such a volatile storage crashes abruptly? It would obviously take away all the logs and active copies of the database. It makes recovery almost impossible, as everything that is required to recover the data is lost.

Following techniques may be adopted in case of loss of volatile storage −

* We can have checkpoints at multiple stages so as to save the contents of the database periodically.
* A state of active database in the volatile memory can be periodically dumped onto a stable storage, which may also contain logs and active transactions and buffer blocks.
* Dump can be marked on a log file, whenever the database contents are dumped from a non-volatile memory to a stable one.

**Recovery**

* When the system recovers from a failure, it can restore the latest dump.
* It can maintain a redo-list and an undo-list as checkpoints.
* It can recover the system by consulting undo-redo lists to restore the state of all transactions up to the last checkpoint.

**ARIES Algorithm:**

Algorithm for Recovery and Isolation Exploiting Semantics (ARIES) is based on the Write Ahead Log (WAL) protocol. Every update operation writes a log record which is one of the following:

**1. Undo-only log record:**

Only the before image is logged. Thus, an undo operation can be done to retrieve the old data.

**2. Redo-only log record:**

Only the after image is logged. Thus, a redo operation can be attempted.

**3. Undo-redo log record:**

Both before images and after images are logged.

* In it, every log record is assigned a unique and monotonically increasing log sequence number (LSN).
* Every data page has a page LSN field that is set to the LSN of the log record corresponding to the last update on the page.
* WAL requires that the log record corresponding to an update make it to stable storage before the data page corresponding to that update is written to disk.
* For performance reasons, each log write is not immediately forced to disk. A log tail is maintained in main memory to buffer log writes.
* The log tail is flushed to disk when it gets full. A transaction cannot be declared committed until the commit log record makes it to disk.
* Once in a while the recovery subsystem writes a checkpoint record to the log. The checkpoint record contains the transaction table and the dirty page table.
* A master log record is maintained separately, in stable storage, to store the LSN of the latest checkpoint record that made it to disk.
* On restart, the recovery subsystem reads the master log record to find the checkpoint’s LSN, reads the checkpoint record, and starts recovery from there on.

The recovery process actually consists of 3 phases:

**1. Analysis:**

The recovery subsystem determines the earliest log record from which the next pass must start. It also scans the log forward from the checkpoint record to construct a snapshot of what the system looked like at the instant of the crash.

**2. Redo:**

Starting at the earliest LSN, the log is read forward and each update redone.

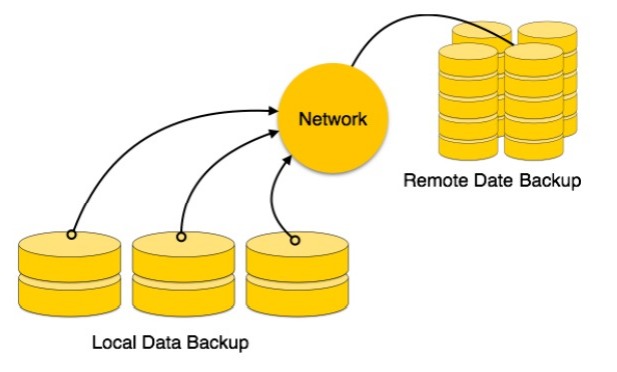
**3. Undo:**

The log is scanned backward and updates corresponding to loser transactions are undone.

**Remote Backup**

Remote backup provides a sense of security in case the primary location where the database is located gets destroyed. Remote backup can be offline or real-time or online. In case it is offline, it is maintained manually.

Online backup systems are more real-time and lifesavers for database administrators and investors. An online backup system is a mechanism where every bit of the real-time data is backed up simultaneously at two distant places. One of them is directly connected to the system and the other one is kept at a remote place as backup.



As soon as the primary database storage fails, the backup system senses the failure and switches the user system to the remote storage. Sometimes this is so instant that the users can’t even realize a failure.

**File –**

A file is named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tables and optical disks.

**What is database security?**

Database security refers to the range of tools, controls and measures designed to establish and preserve database confidentiality, integrity and availability. Confidentiality is the element that’s compromised in most data breaches.

Database security must address and protect the following:

* The data in the database.
* The database management system (DBMS).
* Any associated applications.
* The physical database server or the virtual database server and the underlying hardware.
* The computing or network infrastructure that is used to access the database.

Database security is a complex and challenging endeavor that involves all aspects of information security technologies and practices. It’s also naturally at odds with database usability. The more accessible and usable the database, the more vulnerable it is to security threats; the more invulnerable the database is to threats, the more difficult it is to access and use. This paradox is sometimes referred to as [Anderson’s Rule](https://en.wikipedia.org/wiki/Anderson%27s_rule_(computer_science)).

**Why is it important?**

By definition, a data breach is a failure to maintain the confidentiality of data in a database. How much harm a data breach inflicts on your enterprise depends on various consequences or factors:

* **Compromised intellectual property:** Your intellectual property—trade secrets, inventions, and proprietary practices—can be critical to your ability to maintain a competitive advantage in your market. If that intellectual property is stolen or exposed, your competitive advantage can be difficult or impossible to maintain or recover.
* **Damage to brand reputation:** Customers or partners might be unwilling to buy your products or services (or do business with your company) if they don’t feel they can trust you to protect your data or theirs.
* **Business continuity**(**or lack thereof):**Some businesses cannot continue to operate until a breach is resolved.
* **Fines or penalties for non-compliance:**The financial impact for failing to comply with global regulations such as the Sarbanes-Oxley Act (SAO) or Payment Card Industry Data Security Standard (PCI DSS), industry-specific data privacy regulations such as HIPAA, or regional data privacy regulations, such as Europe’s General Data Protection Regulation (GDPR) can be devastating, with fines in the worst cases exceeding several million dollars *per violation*.
* **Costs of repairing breaches and notifying customers:** In addition to the cost of communicating a breach to customer, a breached organization must pay for forensic and investigative activities, crisis management, triage, repair of the affected systems and more.

**Common threats and challenges**

Many software misconfigurations, vulnerabilities or patterns of carelessness or misuse can result in breaches. The following are among the most common types or causes of database security attacks.

**Insider threats**

An insider threat is a security threat from any one of three sources with privileged access to the database:

* A malicious insider who intends to do harm.
* A negligent insider who makes errors that make the database vulnerable to attack.
* An infiltrator, an outsider who somehow obtains credentials via a scheme, such as phishing or by gaining access to the credential database itself.

Insider threats are among the most common causes of database security breaches and are often the result of allowing too many employees to hold privileged user access credentials.

**Human error**

Accidents, weak passwords, password sharing and other unwise or uninformed user behaviors continue to be the cause of [nearly half (49%) of all reported data breaches](https://www.ibm.com/account/reg/signup?formid=urx-46542).

**Exploitation of database software vulnerabilities**

Hackers make their living by finding and targeting vulnerabilities in all kinds of software, including database management software. All major commercial database software vendors and open source database management platforms issue regular security patches to address these vulnerabilities, but failure to apply these patches in a timely fashion can increase your exposure.

**SQL or NoSQL injection attacks**

A database-specific threat, these involve the insertion of arbitrary SQL or [non-SQL](https://www.ibm.com/topics/nosql-databases) attack strings into database queries that are served by web applications or HTTP headers. Organizations that don’t follow secure web application coding practices and perform regular vulnerability testing are open to these attacks.

**Buffer overflow exploitation**

Buffer overflow occurs when a process attempts to write more data to a fixed-length block of memory than it is allowed to hold. Attackers can use the excess data, which is stored in adjacent memory addresses, as a foundation from which to start attacks.

**Malware**

Malware is software that is written specifically to take advantage of vulnerabilities or otherwise cause damage to the database. Malware can arrive via any endpoint device connecting to the database’s network.

**Attacks on backups**

Organizations that fail to protect backup data with the same stringent controls that are used to protect the database itself can be vulnerable to attacks on backups.

These threats are exacerbated by the following:

* **Growing data volumes:** Data capture, storage and processing continues to grow exponentially across nearly all organizations. Any data security tools or practices need to be highly scalable to meet near and distant future needs.
* **Infrastructure sprawl**: [Network environments](https://www.ibm.com/topics/networking) are becoming increasingly complex, particularly as businesses move workloads to [multi cloud](https://www.ibm.com/products/cloud-pak-for-aiops) or [hybrid cloud](https://www.ibm.com/hybrid-cloud) architectures, making the choice, deployment and management of security solutions ever more challenging.
* **Increasingly stringent regulatory requirements:** The worldwide regulatory compliance landscape continues to grow in complexity, making adhering to all mandates more difficult.
* **Cyber security skills shortage:** Experts predict there might be as many as [8 million unfilled cyber security positions by 2022](https://www.ibm.com/consulting/hr-talent-transformation).

**Denial of service (DoS and DDoS) attacks**

In a denial of service (DoS) attack, the attacker deluges the target server—in this case the database server—with so many requests that the server can no longer fulfill legitimate requests from actual users, and, often, the server becomes unstable or crashes.

In a distributed denial of service attack (DDoS), the deluge comes from multiple servers, making it more difficult to stop the attack.

**Best practices**

Because databases are network-accessible, any security threat to any component within or portion of the network infrastructure is also a threat to the database, and any attack impacting a user’s device or workstation can threaten the database. Thus, database security must extend far beyond the confines of the database alone.

When evaluating database security in your environment to decide on your team’s top priorities, consider each of the following areas:

* **Physical security:** Whether your database server is on-premises or in a cloud data center, it must be located within a secure, climate-controlled environment. If your database server is in a cloud data center, your cloud provider takes care of this for you.
* **Administrative and network access controls:** The practical minimum number of users should have access to the database, and their permissions should be restricted to the minimum levels necessary for them to do their jobs. Likewise, network access should be limited to the minimum level of permissions necessary.
* **User account and device security:** Always be aware of who is accessing the database and when and how the data is being used. Data monitoring solutions can alert you if data activities are unusual or appear risky. All user devices connecting to the network housing the database should be physically secure (in the hands of the right user only) and subject to security controls at all times.
* **Encryption:** All data, including data in the database and credential data, should be protected with best-in-class encryption while at rest and in transit. All encryption keys should be handled in accordance with best practice guidelines.
* **Database software security:** Always use the latest version of your database management software, and apply all patches when they are issued.
* **Application and web server security:** Any application or web server that interacts with the database can be a channel for attack and should be subject to ongoing security testing and best practice management.
* **Backup security:** All backups, copies or images of the database must be subject to the same (or equally stringent) security controls as the database itself.
* **Auditing:** Record all logins to the database server and operating system, and log all operations that are performed on sensitive data as well. Database security standard audits should be performed regularly.

**Controls and policies**

In addition to implementing layered security controls across your entire network environment, database security requires you to establish the correct controls and policies for access to the database itself. These include:

* **Administrative controls** to govern installation, change and configuration management for the database.
* **Preventive controls** to govern access, encryption, tokenization and masking.
* **Detective controls** to monitor database activity monitoring and data loss prevention tools. These solutions make it possible to identify and alert on anomalous or suspicious activities.

Database security policies should be integrated with and support your overall business goals, such as protection of critical intellectual property and your cyber security policies and [cloud security policies](https://www.ibm.com/services/cloud-security). Ensure that you have designated responsibility for maintaining and auditing security controls within your organization and that your policies complement those of your cloud provider in shared responsibility agreements. Security controls, security awareness training and education programs, and penetration testing and vulnerability assessment strategies should all be established in support of your formal security policies.

Data protection tools and platforms

Today, a wide array of vendors offers data protection tools and platforms. A full-scale solution should include all of the following capabilities:

* **Discovery:**Look for a tool that can scan for and classify vulnerabilities across all your databases—whether they’re hosted in the cloud or on-premises—and offer recommendations for remediating any vulnerabilities that are identified. Discovery capabilities are often required to conform to regulatory compliance mandates.
* **Data activity monitoring:**The solution should be able to monitor and audit all data activities across all databases, regardless of whether your deployment is on-premises, in the cloud, or in a [container](https://www.ibm.com/topics/containers). It should alert you to suspicious activities in real-time so that you can respond to threats more quickly. You’ll also want a solution that can enforce rules, policies and separation of duties and that offers visibility into the status of your data through a comprehensive and unified user interface. Make sure that any solution you choose can generate the reports you need to meet compliance requirements.
* **Encryption and tokenization capabilities:** Upon a breach, encryption offers a final line of defense against compromise. Any tool that you choose should include flexible encryption capabilities that can safeguard data in on-premises, cloud, hybrid or multicloud environments. Look for a tool with file, volume and application encryption capabilities that conform to your industry’s compliance requirements, which might demand tokenization (data masking) or advanced security key management capabilities.
* **Data security optimization and risk analysis:** A tool that can generate contextual insights by combining data security information with advanced analytics will enable you to accomplish optimization, risk analysis and reporting with ease. Choose a solution that can retain and synthesize large quantities of historical and recent data about the status and security of your databases, and look for one that offers data exploration, auditing and reporting capabilities through a comprehensive but user-friendly self-service dashboard.