Deadlock Handling

What is Deadlock?

The Deadlock is a condition in a multi-user database environment where transactions are unable to the complete because they are each waiting for the resources held by other transactions. This results in a cycle of the dependencies where no transaction can proceed.

Basically, **Deadlocks occur when two or more transactions wait indefinitely for resources held by each other.** Also, mastering how to detect and resolve deadlocks is vital for database efficiency.

In database management systems (DBMS) a deadlock occurs when two or more transactions are unable to the proceed because each transaction is waiting for the other to the release locks on resources. This situation creates a cycle of the dependencies where no transaction can continue leading to the standstill in the system. The Deadlocks can severely impact the performance and reliability of a DBMS making it crucial to the understand and manage them effectively.

Deadlock Handling

Characteristics of Deadlock

- Mutual Exclusion: Only one transaction can hold a particular resource at a time.
- Hold and Wait: The Transactions holding resources may request additional resources held by others.
- No Preemption: The Resources cannot be forcibly taken from the transaction holding them.
- Circular Wait: A cycle of transactions exists where each transaction is waiting for the resource held by the next transaction in the cycle.

Deadlock Handling

DBMSs often use various techniques to detect and resolve deadlocks automatically. These techniques include timeout mechanisms, where a transaction is forced to release its locks after a certain period of time, and deadlock detection algorithms, which periodically scan the transaction log for deadlock cycles and then choose a transaction to abort to resolve the deadlock.

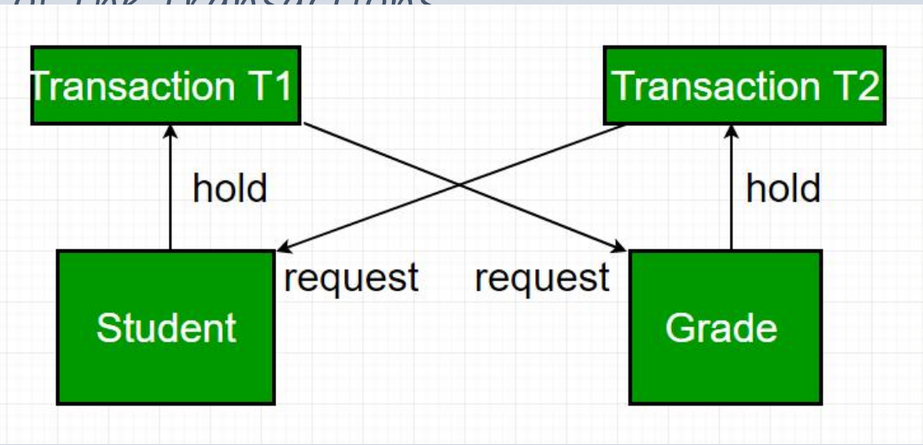
It is also possible to prevent deadlocks by careful design of transactions, such as always acquiring locks in the same order or releasing locks as soon as possible. Proper design of the database schema and application can also help to minimize the likelihood of deadlocks.

In a database, a deadlock is an unwanted situation in which two or more transactions are waiting indefinitely for one another to give up locks. Deadlock is said to be one of the most feared complications in DBMS as it brings the whole system to a Halt.

Deadlock Handling

Example – let us understand the concept of deadlock suppose, Transaction T1 holds a lock on some rows in the Students table and **needs to update** some rows in the Grades table. Simultaneously, Transaction **T2 holds** locks on those very rows (Which T1 needs to update) in the Grades table **but needs** to update the rows in the Student table **held by Transaction T1.**

Now, the main problem arises. Transaction T1 will wait for transaction T2 to give up the lock, and similarly, transaction T2 will wait for transaction T1 to give up the lock. As a consequence, All activity comes to a halt and remains at a standstill forever unless the DBMS detects the deadlock and aborts one of the transactions



Deadlock Handling

What is Deadlock Prevention?

For a large database, the deadlock prevention method is suitable. A deadlock can be prevented if the resources are allocated in such a way that a deadlock never occurs. The DBMS analyzes the operations whether they can create a deadlock situation or not, If they do, that transaction is never allowed to be executed.

Deadlock prevention mechanism proposes two schemes:

- **Wait-Die Scheme:** In this scheme, If a transaction requests a resource that is locked by another transaction, then the DBMS simply checks the timestamp of both transactions and allows the older transaction to wait until the resource is available for execution.
Suppose, there are two transactions T1 and T2, and Let the timestamp of any transaction T be TS (T). Now, If there is a lock on T2 by some other transaction and T1 is requesting resources held by T2, then DBMS performs the following actions:Checks if TS (T1) < TS (T2) – if T1 is the older transaction and T2 has held some resource, then it allows T1 to wait until resource is available for execution. That means if a younger transaction has locked some resource and an older transaction is waiting for it, then an older transaction is allowed to wait for it till it is available. If T1 is an older transaction and has held some resource with it and if T2 is waiting for it, then T2 is killed and restarted later with random delay but with the same timestamp. i.e. if the older transaction has held some resource and the younger transaction waits for the resource, then the younger transaction is killed and restarted with a very minute delay with the same timestamp.

Deadlock Handling

- **Wound Wait Scheme:** In this scheme, if an older transaction requests for a resource held by a younger transaction, then an older transaction forces a younger transaction to kill the transaction and release the resource. The younger transaction is restarted with a minute delay but with the same timestamp. If the younger transaction is requesting a resource that is held by an older one, then the younger transaction is asked to wait till the older one releases it.

The following table lists the differences between Wait – Die and Wound -Wait scheme prevention schemes:

Wait – Die	Wound -Wait
It is based on a non-preemptive technique.	It is based on a preemptive technique.
In this, older transactions must wait for the younger one to release its data items.	In this, older transactions never wait for younger transactions.
The number of aborts and rollbacks is higher in these techniques.	In this, the number of aborts and rollback is lesser.

Challenges of database security in DBMS

1. **Data quality** –

•The database community basically needs techniques and some organizational solutions to assess and attest the quality of data. These techniques may include the simple mechanism such as quality stamps that are posted on different websites. We also need techniques that will provide us more effective integrity semantics verification tools for assessment of data quality, based on many techniques such as record linkage.

•**2. Intellectual property rights** – As the use of Internet and intranet is increasing day by day, legal and informational aspects of data are becoming major concerns for many organizations. To address this concerns watermark technique are used which will help to protect content from unauthorized duplication and distribution by giving the provable power to the ownership of the content. Traditionally they are dependent upon the availability of a large domain within which the objects can be altered while retaining its essential or important properties. However, research is needed to access the robustness of many such techniques and the study and investigate many different approaches or methods that aimed to prevent intellectual property rights violation.

3. Database survivability – Database systems need to operate and continued their functions even with the reduced capabilities, despite disruptive events such as information warfare attacks A DBMS in addition to making every effort to prevent an attack and detecting one in the event of the occurrence should be able to do the following:

•**Confident:** We should take immediate action to eliminate the attacker's access to the system and to isolate or contain the problem to prevent further spread.

Differences between Grant and Revoke commands:

S.NO	Grant	Revoke
1	This DCL command grants permissions to the user on the database objects.	This DCL command removes permissions if any granted to the users on database objects.
2	It assigns access rights to users.	It revokes the useraccess rights of users.
3	For each user you need to specify the permissions.	If access for one user is removed; all the particular permissions provided by that users to others will be removed.
4	When the access is decentralized granting permissions will be easy.	If decentralized access removing the granted permissions is difficult.

Role-based Access Control

Only the administrator should have complete access to the network while the other employees like junior network engineer need not full access to the network device. A junior-level engineer generally requires only to crosscheck the configuration of the device, not to add or delete any configuration so why should give full access to that employee? For these types of scenarios, the administrator defines access to the devices according to the roles of the user.

Role-based Access Control – The concept of Role-based Access Control is to create a set of permissions and assign these permissions to a user or group. With the help of these permissions, only limited access to users can be provided therefore level of security is increased. There are different ways to perform RBAC such as creating custom privilege levels or creating views.

Custom level privilege – When we take a console of the router, we enter into the user-level mode. The user-level mode has privilege level 1. By typing enable, we enter into a privileged mode where the privilege level is 15. A user with privilege level 15 can access all the commands that are at level 15 or below. By creating a custom privilege level (between 2 and 14) and assigning commands to it, the administrator can provide subset of commands to the user.

Public Key Infrastructure

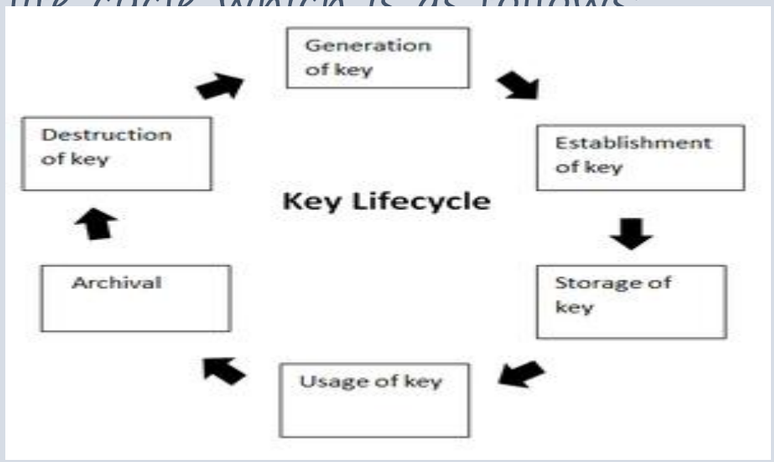
Public key infrastructure or PKI is the governing body behind issuing digital certificates. It helps to protect confidential data and gives unique identities to users and systems. Thus, it ensures security in communications.

The public key infrastructure uses a pair of keys: the public key and the private key to achieve security. The public keys are prone to attacks and thus an intact infrastructure is needed to maintain them.

Managing Keys in the Cryptosystem:

The security of a cryptosystem relies on its keys. Thus, it is important that we have a solid key management system in place. The 3 main areas of key management are as follows:

- A cryptographic key is a piece of data that must be managed by secure administration.
- It involves managing the key life cycle which is as follows:



Challenges of database security in DBMS

Seeing the vast increase in volume and speed of threats to databases and many information assets, research efforts need to be consider to the following issues such as data quality, intellectual property rights, and database survivability. Let's discuss them one by one.

1. Data quality –

•The database community basically needs techniques and some organizational solutions to assess and attest the quality of data. These techniques may include the simple mechanism such as quality stamps that are posted on different websites. We also need techniques that will provide us more effective integrity semantics verification tools for assessment of data quality, based on many techniques such as record linkage.

•We also need application-level recovery techniques to automatically repair the incorrect data.

•The ETL that is extracted transform and load tools widely used for loading the data in the data warehouse are presently grappling with these issues.

•**2. Intellectual property rights** – As the use of Internet and intranet is increasing day by day, legal and informational aspects of data are becoming major concerns for many organizations. To address this concerns watermark technique are used which will help to protect content from unauthorized duplication and distribution by giving the provable power to the ownership of the content.

•**Encryption:** Data encryption is an effective way to protect sensitive data in transit and at rest. However, it can also be a challenge to implement and manage encryption keys and ensure that

Challenges of database security in DBMS

Access Control: Access control involves regulating the access to data within the database. It can be challenging to implement access control mechanisms that allow authorized users to access the data they need while preventing unauthorized users from accessing it.

Auditing and Logging: DBMS must maintain an audit trail of all activities in the database. This includes monitoring who accesses the database, what data is accessed, and when it is accessed. This can be a challenge to implement and manage, especially in large databases.

Database Design: The design of the database can also impact security. A poorly designed database can lead to security vulnerabilities, such as SQL injection attacks, which can compromise the confidentiality, integrity, and availability of data.

Malicious attacks: Cyberattacks such as hacking, malware, and phishing pose a significant threat to the security of databases. DBMS must have robust security measures in place to prevent and detect such attacks.

Physical Security: Physical security of the database is also important, as unauthorized physical access to the server can lead to data breaches.