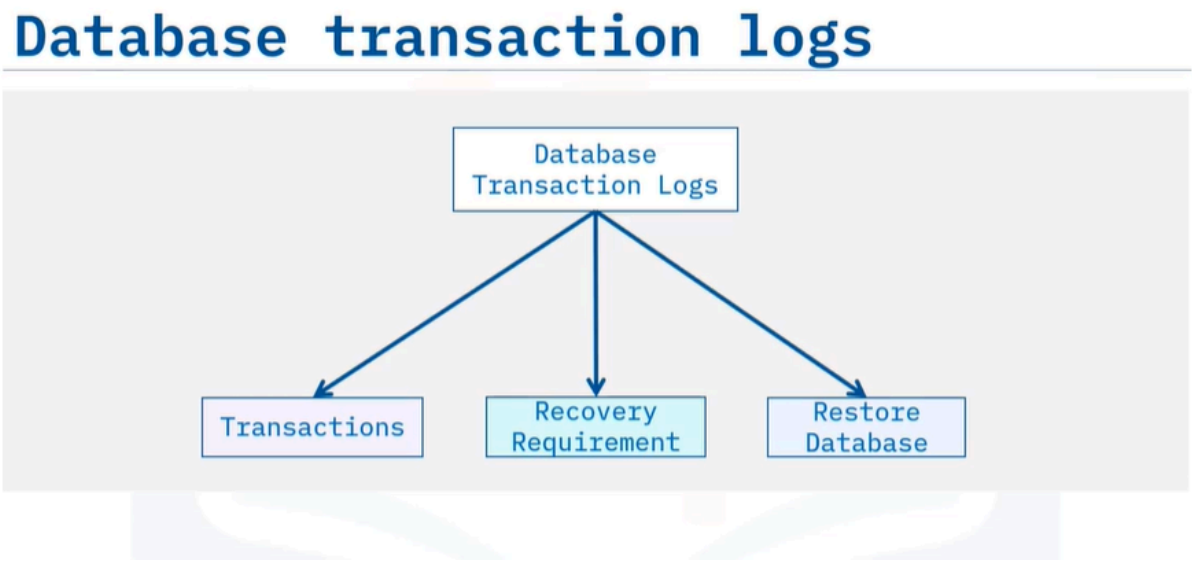
Using Database Transaction Logs for Recovery

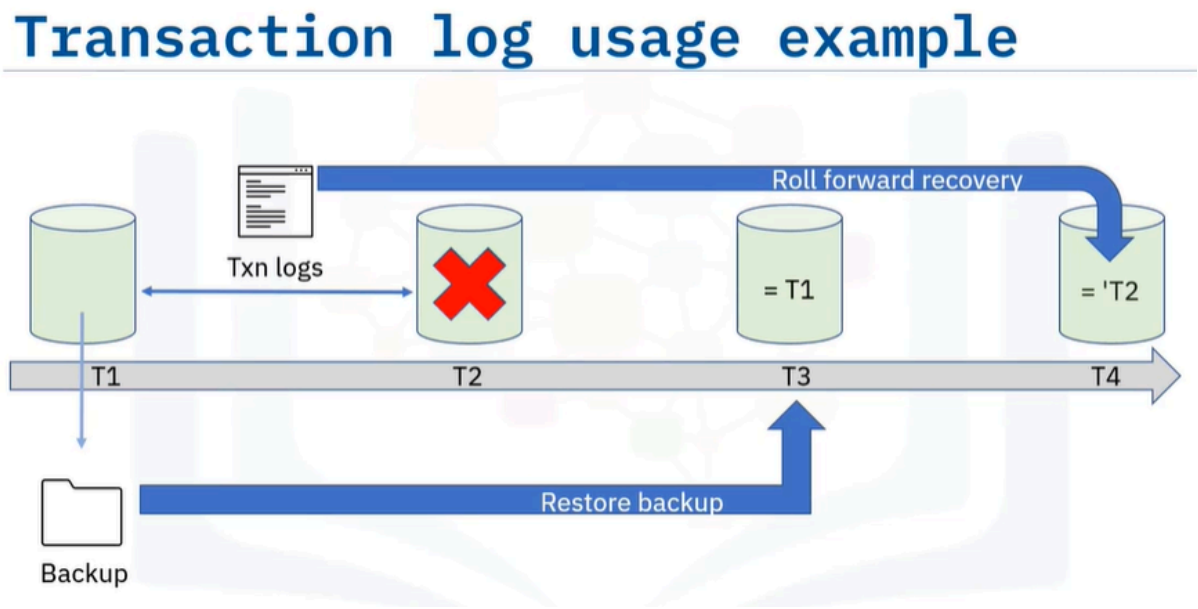
Sure! Let's talk about database transaction logs in simple terms.

Database transaction logs are like a diary for a database. They keep a record of everything that happens to the data, such as when information is added, changed, or deleted. Imagine you have a notebook where you write down every time you make a change to your room, like moving furniture or adding new decorations. If something goes wrong, like a spill or a mess, you can look back at your notes to remember what you did and fix it. Similarly, if a database has a problem, the transaction logs help restore it to a previous state, ensuring that no important information is lost.

For example, let's say you backed up your database at a certain time (like taking a picture of your room). If something goes wrong later, you can use that backup to return to the way things were at that moment. Then, you can look at the transaction logs to see what changes happened after that backup and apply those changes to get everything back to normal. This process is like cleaning up your room by first going back to the picture and then making the necessary adjustments based on your notes.



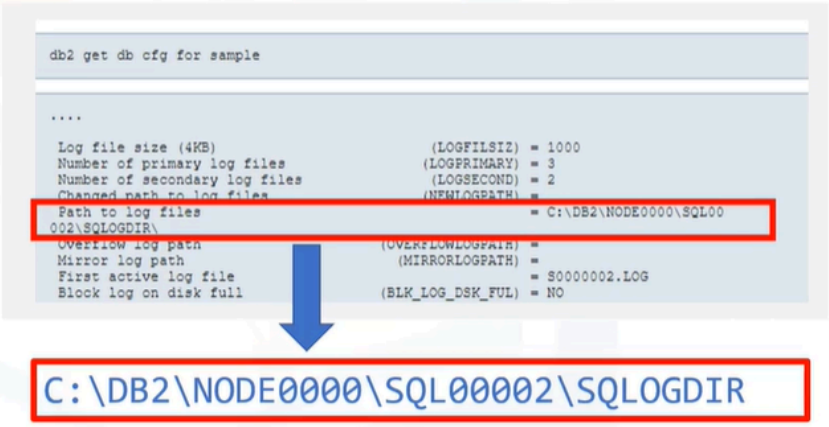
- Separate from the diagnostic log, a database management system (DBMS) uses transaction logs to keep track of all transactions that change or modify the database. The information stored in these transaction logs can be used for recovery purposes when data is accidentally deleted, or a system or hardware failure such as a disk crash occurs. In case of such events, the transaction log, in conjunction with a database backup, can help restore your database back to a consistent state at a specific point in time.



- For example: In this timeline, at time T1 you issue the BACKUP command to make a full copy of your database. Users then continue working, and the information about their activities is kept in transaction logs. At time T2, there is a problem that damages the database. So, at time T3 you issue a RESTORE command using the backup image you took at time T1. This allows you to get the database back to the point in time that the backup was taken. Then, at time T4 you apply the transaction logs for the period of time between T1 and T2 to the restored database. This process of applying committed transactions from transaction logs is called roll forward recovery, and it returns your database to the point just before the crash happened.

Storing transaction log files

- Configure log file location
- Db2:
  - sqlogdir
- PostgreSQL:
  - pg_xlog
- Write-ahead log (WAL)



- You typically specify where you want transaction log files to be stored using the database configuration. Here is an example location of a db2 transaction log file directory in a sqlogdir subdirectory on a Windows system. In PostgreSQL, the log files default to being stored in the pg_xlog sub-directory. In PostgreSQL, the transaction log file is referred to as a write-ahead log (WAL), because before the database manager writes the database changes or transactions to data files, the DBMS first writes them to the WAL.

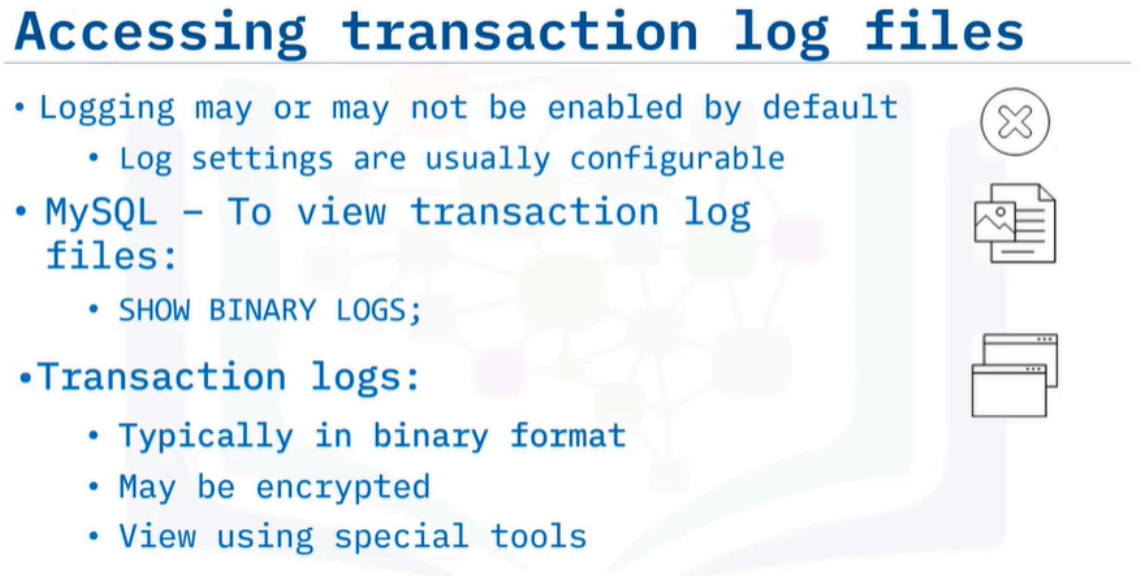
Storing transaction log files

- Isolate logs on different volumes from data
  - Performance
  - Recoverability

- Log mirroring – store second copy of log files in an alternative location

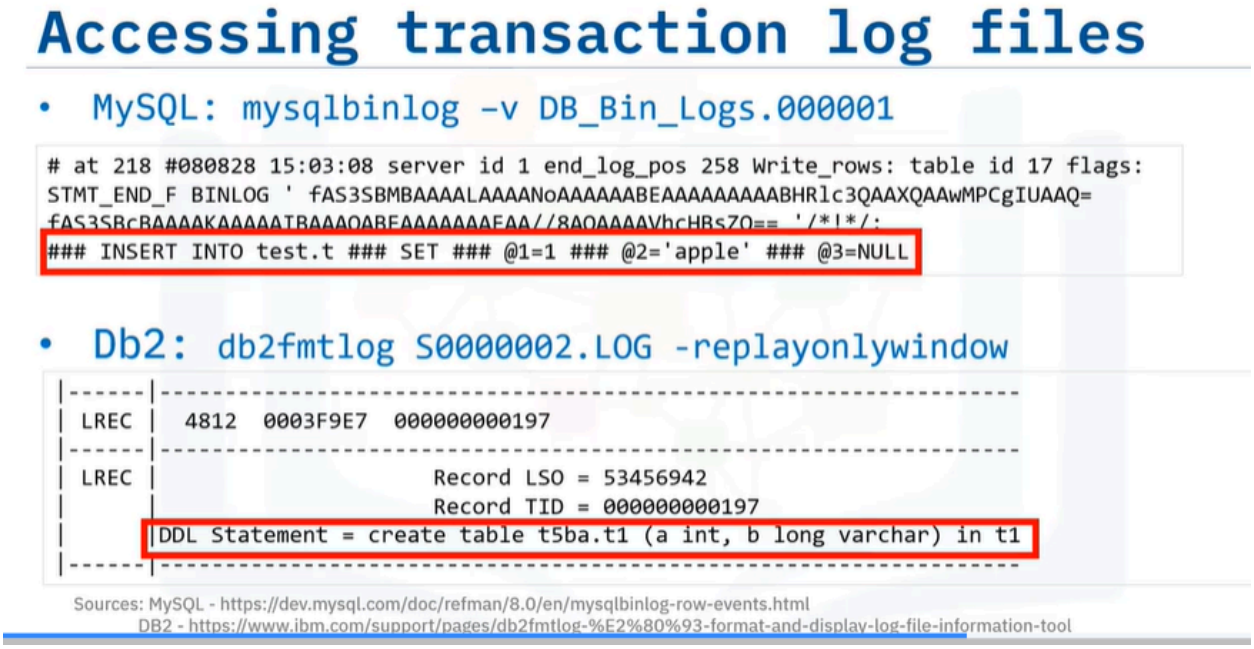
- Log shipping – copy and send logs to replica or standby servers

- A recommended best practice, at least for production databases, is to place the transaction log files on storage volumes that are separate from where the database objects like tablespaces are stored. This not only helps with performance, so database writes are not competing with log writes, but also improves recoverability since it isolates logs from crashes or the corruption of data volumes. For even more enhanced recoverability in business-critical databases, some relational database management systems (RDBMSs) provide the ability to automatically mirror logs to a second set of storage devices. Or you can write scripts to copy or ship logs to remote replicas or standby systems in a process called log shipping.

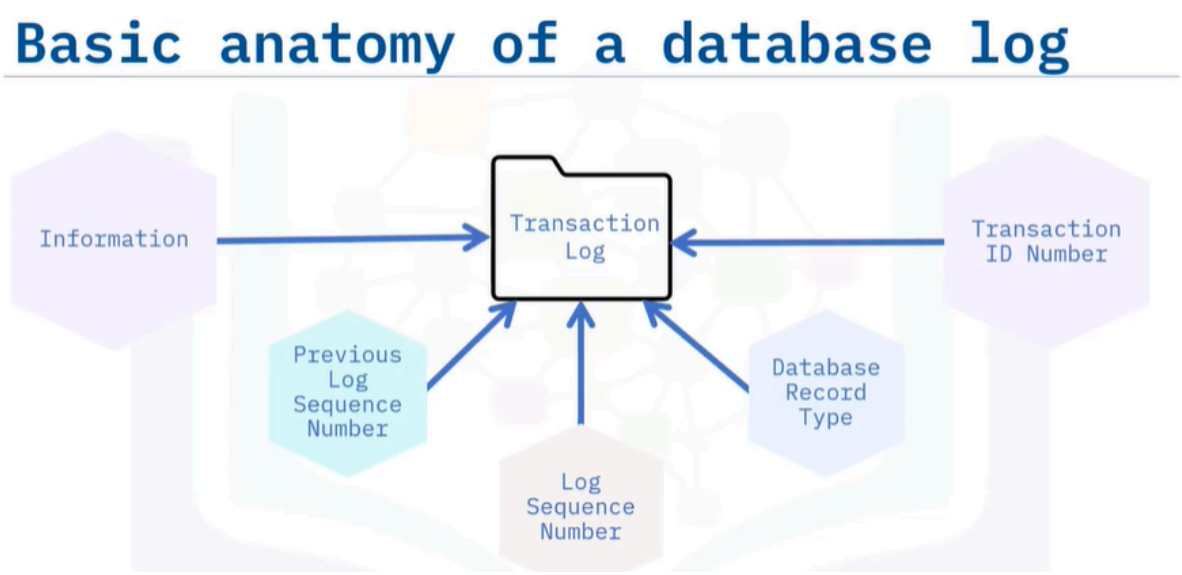


- In many RDBMSs, like Db2 and SQL Server, transaction logging is enabled by default, but you still may want to change the default settings for improved recoverability and performance. In other RDBMSs, like MySQL, transaction logging is disabled by default and may need to be manually configured. For MySQL, you can run the SHOW BINARY LOGS command to check if logging is enabled and if it is, you will see a list of the log files. Unlike other types of logs, such as diagnostic and error logs that are mostly in text format and human readable, the database transaction logs are typically in binary formats that are

sometimes encrypted and require specialized tools to format and display contents.



- For example, in MySQL, you use the mysqlbinlog tool to view the contents of MySQL transaction log files, also known as binary logs. The sample output shows a log file entry for an INSERT statement. And for Db2, you use the db2fmtlog tool for formatting and displaying log file information. The sample output shows a record for a DDL statement to create a table.



- A log file contains multiple entries, and the information they contain may vary depending on the RDBMS. For example, a typical database log record is made up of: A log transaction ID number, which is a unique ID for the log record. The database record type, which describes the type of database log record. A log sequence number, which is a reference to the database transaction generating the log record. A previous log sequence number is a link to the last log record. This implies that database logs are constructed in linked list form. And information which details the actual changes that triggered the log record to be written.

How can you use transaction logs for recovery?

You can use transaction logs for recovery in a few key steps:

1. **Backup**: First, you create a full backup of your database. This is like taking a snapshot of your data at a specific point in time.
2. **Identify the Issue**: If something goes wrong, such as data being accidentally deleted or a system failure, you need to identify when the problem occurred.

3. **Restore the Backup**: You restore the database using the backup you created earlier. This brings the database back to the state it was in at the time of the backup.
4. **Apply Transaction Logs**: After restoring the backup, you then apply the transaction logs. These logs contain a record of all changes made to the database after the backup was taken. By applying these logs, you can bring the database up to date, incorporating all the changes that occurred after the backup.
5. **Roll Forward Recovery**: This process of applying the transaction logs to the restored database is called roll forward recovery. It allows you to recover the database to the point just before the issue occurred.

In summary, transaction logs are essential for recovering a database because they provide a detailed record of changes that can be used to restore the database to a consistent state after a failure or data loss.

What is the structure of a database transaction log?

The structure of a database transaction log typically includes several key components:

1. **Log Transaction ID**:
    - A unique identifier for each log record. This helps in tracking and referencing specific transactions.
2. **Database Record Type**:
    - This describes the type of operation that was logged, such as an insert, update, or delete action.
3. **Log Sequence Number (LSN)**:
    - A reference number that indicates the order of the transaction. It helps in maintaining the sequence of operations.
4. **Previous Log Sequence Number**:

- This links to the last log record, creating a chain of log entries. It allows the database to understand the order of transactions.

5. **Change Details**:
    - This section contains the actual information about the changes made to the database, such as the data that was inserted, updated, or deleted.

These components work together to create a comprehensive record of all transactions that modify the database, allowing for effective recovery and auditing.

What would happen if transaction logs were not used in databases?

If transaction logs were not used in databases, several issues could arise:

1. **Data Loss**:
    - Without transaction logs, any changes made to the database would not be recorded. In the event of a system failure or crash, all unsaved changes could be lost, leading to significant data loss.
2. **Difficult Recovery**:
    - Recovery from failures would be much more challenging. Without logs, it would be hard to determine what changes occurred before the failure, making it difficult to restore the database to a consistent state.
3. **Inconsistent Data**:
    - Transaction logs help maintain data integrity. Without them, there could be inconsistencies in the database, as some transactions might be partially completed or not recorded at all.
4. **No Audit Trail**:

- Transaction logs provide a record of all changes, which is essential for auditing and compliance. Without them, tracking changes and understanding the history of data modifications would be impossible.

5. **Increased Downtime**:
    - In the event of a failure, the time required to recover the database would increase significantly, leading to longer downtimes and potential loss of business.

In summary, transaction logs are crucial for ensuring data integrity, enabling recovery, and maintaining a reliable database system. If they were not used, the risks to data and system reliability would be much higher.

How can transaction logs improve database recovery processes?

Transaction logs improve database recovery processes in several important ways:

1. **Detailed Change Tracking**:
    - Transaction logs record every change made to the database, including inserts, updates, and deletes. This detailed tracking allows for precise recovery of data to a specific point in time.
2. **Point-in-Time Recovery**:
    - With transaction logs, you can restore the database to a specific moment before a failure or data loss occurred. This is particularly useful for recovering from accidental deletions or corruption.
3. **Roll Forward Recovery**:
    - After restoring a backup, transaction logs can be applied to bring the database up to date. This process, known as roll forward recovery, ensures that all committed transactions are included, minimizing data loss.
4. **Isolation of Changes**:

- Transaction logs allow for the isolation of changes made during transactions. If a transaction fails, only the changes related to that transaction can be rolled back, leaving the rest of the database intact.

5. **Improved Performance**:
    - By separating transaction logging from the main database operations, performance can be enhanced. This separation allows for more efficient writing of logs without impacting the speed of database transactions.
6. **Automated Recovery**:
    - Many database management systems can automatically use transaction logs to recover from failures without manual intervention, streamlining the recovery process and reducing downtime.

In summary, transaction logs enhance the recovery process by providing a reliable and detailed record of changes, enabling point-in-time recovery, and allowing for efficient and automated recovery methods.