DB Restore Types in SQL Server:

In SQL Server, restoring a database involves multiple phases that work together to bring the database to a consistent state as it was at the time of the backup. The process generally includes the following stages:

1. Restore Phase 1: Restoring the Database (Full Backup)

- **Step 1: Initial Restore (Full Backup)**
  - The first phase of the restore process is to restore the full backup of the database. This provides the foundation for the restoration and restores the database to the state at the time the backup was created.
  - The **RESTORE DATABASE** command is used to restore the full backup.
  - The database remains in a *RESTORING* state after this phase, meaning the database is not yet accessible for users.

Key Concepts:

- A full backup includes all the data, the full transaction log, and the structures like tables and indexes.
- At this point, the database is in a recovery state where no user can access the data.

2. Restore Phase 2: Restoring Transaction Logs (Transaction Log Backup)

- **Step 2: Apply Transaction Log Backups (Differential or Log Backups)**
  - After the full backup has been restored, transaction log backups are applied to bring the database to a specific point in time (e.g., the last log backup before a crash or disaster).
  - These backups apply the changes that have been made to the database since the full backup.
  - The **RESTORE LOG** command is used for this.
  - A database can have multiple transaction log backups, and they are applied sequentially to roll forward all transactions.

Key Concepts:

- Transaction log backups contain a record of all the transactions that occurred after the last backup (either full or differential).
- You apply transaction log backups in the order they were taken to ensure the database is brought to a consistent state.
- The database remains in a *RESTORING* state after this phase too.

3. Restore Phase 3: Restoring the Final Transaction Log or Marking the Database as Recoverable

- **Step 3: Mark Database as Operational (Final Restore)**
  - Once all transaction log backups are restored, the database is still in a *RESTORING* state, which prevents user access.
  - The final step is to apply the **WITH RECOVERY** option, either by explicitly issuing a `RESTORE DATABASE` command or by simply not applying any further transaction logs.
  - This step completes the restore process and marks the database as fully recovered, and user access is enabled again.

    - The **RESTORE DATABASE** command with the `WITH RECOVERY` option is used here to indicate the completion of the restore operation.

Key Concepts:

- The database will become accessible to users after this phase, as it transitions from the *RESTORING* state to the *ONLINE* state.
- You should not restore any further transaction log backups once you've done the final restore with the `WITH RECOVERY` option.

4. Restore Phase 4: Point-in-Time Recovery (Optional)

- **Step 4: Point-in-Time Recovery (If needed)**
    - In some cases, you might need to restore the database to a specific point in time (e.g., to a point just before a critical error occurred).
    - You can do this by restoring a full backup and then applying transaction log backups up until the exact point in time you require. This is often done by using the `STOPAT` clause in the `RESTORE LOG` command.
    - The **RESTORE LOG WITH STOPAT** command allows you to roll back the database to a specific time, undoing any transactions after that point.
    - After this, you would still need to complete the restore with the `WITH RECOVERY` option.

Key Concepts:

- Point-in-time recovery allows you to recover from scenarios like accidental data deletion or corruption.
- The point-in-time recovery requires a consistent set of transaction log backups and the full backup.

5. Restore Phase 5: Recovery with No Recovery (Differential Restore)

- **Step 5: Applying Differential Backups (Optional)**
    - Differential backups are backups that include all the changes made since the last full backup. You can restore a full backup first, followed by applying a differential backup.
    - Differential backups help to reduce the amount of time and storage needed as compared to applying multiple transaction log backups.

Key Concepts:

- Differential backups don't require applying all previous transaction logs since the last full backup, making them faster to restore than applying individual transaction log backups.
- After applying a differential backup, you would still need to follow up with transaction log backups (if necessary).

Summary of the Phases

1. **Restore Full Backup**: Restore the full backup of the database to start the restoration process.
2. **Restore Transaction Logs**: Apply the transaction log backups in order to roll forward all transactions.
3. **Mark Database as Available**: Apply the `WITH RECOVERY` option to complete the restore process and bring the database online.
4. **Point-in-Time Recovery**: Optionally, restore to a specific point in time using `STOPAT`.
5. **Apply Differential Backup**: Optionally, apply differential backups if available to reduce the time taken to restore the database.

Final Notes:

- **Transaction Log Backup** is critical in a full database restore. Without transaction log backups, restoring to a point-in-time is not possible.
- **Recovery Models** (Full, Bulk-Logged, Simple) influence the way backups are managed and restored. In the **Full** recovery model, all transaction logs need to be restored after a full backup.
- **WITH NORECOVERY** keeps the database in a restoring state for subsequent log restores, while **WITH RECOVERY** completes the restoration and brings the database online.

Let me know if you need more detailed explanations about each step!