CHAPTER 5

LESSENING THE LOAD ON DATABASE SERVER

5.1 INTRODUCTION

Information technology operations are a crucial aspect of most organizational operations. One of the main concern is business continuity; companies rely on their information systems to run their operations. If a system becomes unavailable, company operations may be impaired or stopped completely. It is necessary to provide a reliable infrastructure for IT operations, in order to minimize any chance of disruption or disaster and to sustain the adequate performance of the systems.

These days, Information technology operations are facing various typical challenges. They are,

- Availability of Critical data at all times
- Huge cost involved to recover the systems in case of disaster
- Performance hindrances due to growth of historical data

In this methodology, following steps are implemented in disaster recovery mechanism for reducing the load on data base server to improve the performance of web applications.
- Separating the OLAP and OLTP.
- Optimizing Queries and Stored Procedures.

5.2 SEPERATING THE OLAP AND OLTP

Web applications usages are wider since they can access over the network. Performance hindrances on the web applications are major challenges faced by IT solution providers. In three tier (App, Web and Database) architecture, Web application performance mostly suffers to get the reliable response from the database Server. Web application performance behaves poorly due to the surplus congestion on database server. Congestion on database server due to more OLTP and OLAP related tasks that lead to poor performance of web applications. The intention of this methodology is to separate OLAP, Reports workloads into standby server and OLTP workloads on primary database server not to impede each other. Ultimately, web application performance is improved by getting reliable response from database server by balancing the load on production database servers. Based on this methodology, OLTP and Reports related requests from web server have been processed effectively by Database Server.

5.2.1 Existing Disaster Recovery Mechanism in SQL Server

Importance of Disaster Recovery Planning is, as IT systems have become increasingly critical to the smooth operation of a company, and arguably the economy as a whole, the importance of ensuring the continued operation of those systems, or the rapid recovery of the systems, has increased.
It is estimated that most large companies spend between 2% and 4% of their IT budget on disaster recovery planning, with the aim of avoiding larger losses in the event that the business cannot continue to function due to loss of IT infrastructure and data.

**Log shipping** is one of the database disaster recovery techniques which are available in the SQL Server. It is used to synchronize the distributed database server and it can improve the application performance and availability of database. Synchronize the database by copying transaction logs, backing up, and restoring data. SQL server used SQL server job agents for making those processes automatic. Log shipping does not involve automatic transfer of server if there is any failure. If the primary server fails, it will not redirect the application to a standby server. This has to be done manually.

![Log shipping architecture](image)

**Figure 5.1 Log shipping architecture**

The main functions of Log Shipping from Figure 5.1 are

- Backing up the transaction log of the primary database.
- Copying the transaction log backup to each standby server.
- Restoring the transaction log backup on the standby database.

After implementing the DR (Disaster Recovery) mechanism (Primary and Standby Server Utilizations in the DR Mechanism), normally, web application users are accessing primary database server through requesting web server as shown in Figure 5.2. And Online Transactional Processing (OLTP), Online Analytical Processing (OLAP) and Reports related transactions are processing at primary database server. Ultimately, primary database server resources (Memory, IO and Processor) are utilized heavily.

All the SQL procedures which related to the OLTP, OLAP and Reports execution happens on the primary database server. Primary server database and tables are heavily accessed.

Figure 5.2 Web application users accessing database server through requesting web server
Standby server resources are not efficiently utilized apart from the operation of database synchronization. Primary database server contains mixing of OLTP, OLAP, and Reporting workloads hugely. OLTP workloads are characterized by many small transactions, with an expectation of very quick response time from the user. OLAP and Reporting workloads are characterized by a few long-running operations that might consume more resources and cause more contention. The long-running operations are caused by locking and by the underlying physical sub-system hence, cause the performance hindrances of web application. Due to this, response from database server to web server takes plenty amount of time.

5.2.2 Design and Implementation of SQL Server Load Balancing

Purpose of this design is to separate OLAP, Reports and OLTP workloads on one database server not to impede each other and ultimately to improve the web application performance. OLAP and reporting workloads tend to be characterized by infrequent, long-running queries. Web application users are rarely waiting impatiently for the queries to complete. OLTP workloads tend to be characterized by lots of small transactions that return something to the user in less than a second. Long-running queries for analysis, reports, or ad-hoc queries may block inserts and other transactions in the OLTP workload until the OLAP and reporting query completes (Figure 5.3).
In order to enhance the production database server performance on the primary server, ultimately we need to minimize the load on the primary server. In order to balance the load on primary server, the standby (secondary) server which built for DR aspects can be efficiently utilized.

To minimize the load on primary server, need to separate OLAP and Reports workloads on standby server have to be designed. To achieve this and to setup a database load balancing; need to do the following.

- Log shipping must be implemented.
- Primary and standby database must have FULL recovery mode.
- Standby server database has to be in standby/read only mode.
- **Linked server** of standby server to be created on the primary server.
5.2.3 Linked Server

Linked server enables SQL server to connect and access the database created in the remote server (Standby Server). Linked server is a mechanism that enables a SQL server to access data from remote data sources. Linked servers can be used to query data, update data, and to execute procedure calls residing on a remote server. Remote data sources can be other SQL server instances or other relational database products such as Oracle, Access, and Sybase. Excel, text, and XML files can also be data sources. Connections to remote data sources are established via an OLEDB provider. OLEDB is a set of common object model interfaces, created by Microsoft, that provide a consistent method of accessing different data sources. Creating a linked server requires that the correct OLEDB provider is specified for the particular data source.

When running distributed queries on a linked server, if the linked server has the same character set and sort order (collation) as the local SQL server, then can reduce overhead and boost performance if the SP_SERVEROPTION “collation compatible” option is set to true. This setting is telling SQL server to assume that all columns and character sets on the remote server are compatible with the local server. This same option can also be turned on for a linked server using Enterprise Manager or Management Studio.

If this option is not selected, then the distributed query being executed on the remote server must return the entire table to the local server in order for the WHERE clause to be applied. As imagined, this could potentially return a lot of unnecessary data over the network, slowing it down.
5.2.4 Implementation of Linked Server

After implementing the linked server in primary database server, need to manually enable the following code on the SQL procedures which are required and related to the Reports and OLAP processes. Primary intent is to redirect the SQL Procedures execution at standby server.

```
SET XACT_ABORT ON
EXEC [LinkedServerName].[DBname].[dbo].[Procedurename]
SET XACT_ABORT OFF
```

To execute a distributed query against a linked server, it is must to include a fully qualified, four-part table name for each data source to query. This four-part name should be in the form `linked_server_name.catalog.schema.object_name`.

Example:

```
EXEC [LinkedServerName].[DatabaseName].[dbo].[Procedurename]
```

As using the linked servers for distributed transactions, XACT_ABORT must be set ON for data modification statements in an implicit or explicit transaction against most OLEDB providers, including SQL Server. And this statement specifies whether SQL server automatically rolls back the current transaction when a Transact-SQL statement raises a run-time error. Hence include the first statement as SET XACT_ABORT ON in the procedure and specify SET XACT_ABORT OFF at the end of the SQL procedure.

After including these redirect statements in the SQL procedures, Execution happens at standby servers. Finally, SQL server load balancing technique has been implemented and realizes the following areas.
• OLTP transaction from web application users’ execution happens in the primary server.

• Reports and OLAP related stored procedures from web application users are rerouted to the standby servers via linked server.

5.2.5 Performance Measurement

Following are the performance observations of primary database server after implementing the load balancing in SQL server.

Method of Performance Measured

SQL server profiler tool used to capture the duration of the procedures before and after implementing the load balancing code changes. For this, few procedures have been considered for the processes that are taken long time for execution in the few areas in application like Finance, Production and sales. Using the filter objects in the SQL server profiler, to capture the events respectively before and after the load balancing implementation.

Environment Measured

Database Size : 850 GB

Concurrent users : 120

Results Obtained

Based on the Figures 5.4 and 5.5 observations, tremendous performance improvement in the web applications and web servers are seen.
Figure 5.4 Performance improvements in the web servers

Figure 5.5 Performance improvements in the web applications
Following benefits are observed in different perspectives after implementing the SQL Server load balancing.

- **Web Application Perspective**
  - Web application performance is improved.
  - Web application users are experienced optimal navigations of pages.
  - OLAP and OLTP transactions went fine without any locks and blocks.

- **Primary database Server Perspective**
  - The load is balanced in the primary database server.
  - Responses from web server are improved.
  - Locks and blocks are reduced in the primary database and tables.
  - Utilization of resources like CPU, RAM and IO which causes sluggishness in the primary database server is reduced, hence getting reliable response to the web server.

- **Standby server Perspective**
  - Serves as a report server.
  - Used for OLAP (On-line Analytical processing) towards data warehousing.
  - Standby servers are utilized efficiently.
  - Reports are launched quickly irrespective of the volume of data as cache that was built first time will remain unaffected.
– In case of any failure of primary database server, Web application can continue to access the standby server with minimum change

5.3 OPTIMIZING QUERIES AND STORED PROCEDURES

In the SQL server system, CPU, disk, memory and network are the major subsystems than can be affected by bottlenecks when a query is run. CPU is the most common bottleneck and most of the times; it is so because of bottlenecks in other sub systems such as memory and disk. For example, memory or disk pressure can result in excessive kernel mode operations, there by leading to higher CPU consumption.

In old decades, Query performance was suffered due to many reasons, such as lack of indexes on the tables, poor joining order with tables, recompilation that causes Query Execution plan generation every time and excessive use of temporary tables in tempdb.

TempDB is the global resource for the SQL server system. It is one of the primary system databases that extremely decide the reasonable performance of the SQL server instance and the Server. Impact in terms of heavy load on the TempDB leads to the performance hindrances of entire system due to the resource crunch.

Intention of this methodology is to reduce the load on TempDB and improve the system performance by writing a query in optimized way in the user databases.

TempDB Database In SQL Server

The TempDB system database is very similar to a user database. The main difference is that data in TempDB does not persist after SQL server
shuts down. SQL server uses **TempDB** to store internal objects such as the intermediate results of a query. It is a workspace for holding temporary objects or intermediate result sets.

All the production user databases in the SQL server instance will access the same TempDB database only. Hence, if any problematic query from any one user database may acquire more resources (IO, Memory, and CPU) on TempDB which lead poor performance of other databases, applications and entire system in the same SQL instance. Therefore, need to be more cautious on designing the user Queries on the user databases with respect to the TempDB performance.

In the SQL server, Query Execution and Processing the Query acquires the CPU, memory and disk resources.

**Management of TempDB**

- Temporary user objects that are explicitly created, such as: global or local temporary tables, temporary stored procedures, table variables, or cursors.

- Internal objects that are created by the SQL server database engine, for example, work tables to store intermediate results for spools or sorting.

- Row versions that are generated by data modification transactions for features, such as: online index operations, Multiple Active Result Sets (MARS), and AFTER triggers.

- **TempDB** is re-created every time SQL server is started so that the system always starts with a clean copy of the database.

**Temporary Database usage Architecture in SQL Server Instance**
Figure 5.6 illustrates the architecture of temporary database in SQL server.

Figure 5.6 Architecture of temporary database

How TempDB Database is affecting the Server Performance

- Since it is a global resource of the SQL server system, the entire user created databases and users can have access to this TempDB database.

- Log and data files growth on TempDB may lead to the server running out of space.

- Responsible for page and extent allocations and DE allocations for all type of objects in TempDB and this may
lead to the DML operations (INSERT, UPDATE, DELETE, SELECT) contention.

- Normally, DML operations are more CPU intensive as this causes page allocation/deallocation on Tempdb. But, Schema (DDL) changes are not more CPU intensive as like DML operations.

- Responsible for manipulation of metadata when user objects in TempDB are created or dropped and this may lead to DDL operations contentions.

- There are many production user databases in the SQL server instance will access the same TempDB database only. Hence, if any problematic query from any one user database may acquires more resources (IO, Memory, and CPU) on TempDB which leads to poor performance of other databases, applications and entire system in the same SQL instance.

**Primary Operations that are Lead to Heavy Load on TempDB**

- Heavy usage of temporary tables (Local and Global).
- Heavy usage of temporary variables.
- Heavy usage of CTE (Common table Expressions).

All the above options are used for reports related procedures and SQL’s as they can be used in multi session and they are used most often to provide workspace for the intermediate results when processing data within a batch or procedure. During peak hours, this workload on TempDB may affect the entire OLTP transactions of the system.

**Temporary Tables**
Creating temporary tables (#temp) in the user databases will be created and dropped on TEMPDB only. Creating temporary tables with large result set may acquire more space in TempDB and lead more resources (IO, Memory and CPU) contention on the TempDB.

Temporary table objects have created and dropped as many times as multiple users have been accessing the same.

Temporary table is defined within the execution scope (Local Temp) or session (Global temp) of procedures.

Temporary tables were using primarily on the procedures where usage on multi session by several users.

Frequent access of large set temporary tables by concurrent users may lead to contention on allocation tables in TempDB.

Temporary table operations are logged in TempDB which leads to log file growth if incase of table contains large result sets.

Temporary tables are likely to remain cached in memory, but only if they are frequently-used: same as with a base table. TempDB operates a system called temporary object reuse, which will cache a portion of the temporary objects with the plan, if there is sufficient memory

**Table Variables**

Creating table variables (@T table) in the user databases will be created and dropped on TEMPDB only. Creating table variables with large result set may acquire more space in
TempDB and lead more resources (IO, Memory and CPU) contention on the TempDB.

- Also, table variable doesn’t have feature to add indexes on it. Hence, on the user databases itself it will slow down the process.

- Frequent access of large set table variables by concurrent users may lead to contention on allocation tables in TempDB.

- Table variable operations are logged in TempDB which leads to log file growth if incase of table contains large result sets.

CTE (Common Table Expression)

- CTE (common table expression) is used to have a temporary result set that is defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement.

- A CTE is similar to a derived table in that it is not stored as an object and lasts only for the duration of the query.

- Unlike a derived table, a CTE can be self-referencing and can be referenced multiple times in the same query.

- When the query plan for a common table expression query uses a spool operator to save intermediate query results, the database engine creates a work table in TempDB to support this operation.

- Frequent access of large set CTE by concurrent users may lead to contention on allocation tables in TempDB.
Above mentioned operations are highly TempDB database intensive and cause over load to TempDB and lead to severe server performance hindrances. Commonly, allocate and deallocate pages due to temporary tables, table variables and CTE may lead to TempDB overload.

**Optimization Process**

System performance hindrances may arise when dealing with massive databases and huge number of concurrent users. Therefore, need to be more cautious on designing the user Queries and SQL statements on the user databases with respect to the TempDB performance.

We can address above all the problems with replacing temporary tables with **Permanent tables** with **SPID** column.

SPID in SQL server is a server process ID. These process ID’s are essentially sessions in SQL server. Every time an application connects to SQL server, a new connection (or SPID) is created. This connection has a defined scope and memory space and cannot interact with other SPIDs. The term SPID is synonymous with connection, or session.

SPID’s can have their own transaction isolation levels defined globally. When set within a connection, all proceeding executions maintain the same transaction isolation.

**Design Permanent Tables Instead of Temporary Tables**

- Used permanent tables with putting an appropriate index is solved the TempDB load issue in the database.
• Created tables permanently with SPID column and SPID column must be indexed as cluster.

• Identified the Stored Procedures which have temporary tables (#temp) and took more time to execute hence utilizing more resources. Replaced those stored Procedures with permanent tables with SPID column check.

• Server Process Identification (SPID) number column should be added into the tables. It should be checked with @@SPID (Current SPID) on all the SQL statements in the WHERE conditions. This is to ensure the data integrity and consistency.

• SPID check should be apt for multi session and concurrent usage.

**Strategies to be Followed during of Permanent Tables Implementation**

• Use permanent tables instead of temporary tables when the query is dealing with massive records.

• If the particular SQL query is used for multi session with concurrent access, then SPID column should be added in to the permanent table.

• SPID check with current @@SPID is mandatory for all SQL statements that are used in SQL procedures. This is to ensure the data integrity and consistency.

• SPID check condition should made in the all possible WHERE clause as this is ensure no deadlock happens while multi session transactions.
Cluster index should be created on SPID column on permanent tables. This is to speed up the entire transactions and data retrieval easier and faster.

If the Stored Procedures and SQL statements are going to be utilized on single session at a time then straightly go on permanent table without having SPID column.

Also for these permanent tables are created for temporary purpose, hence deletion based on SPID frequently may lead to table fragmentation in order to defragment them these should be truncated in periodic basis.

**Illustration of Code Implementation against Temporary Tables**

- Sample procedure with Temporary table for Concurrent transaction in Multi Session

```sql
// Creation of #temporary table within the procedure scope
CREATE PROCEDURE PROCEDURENAME (PARAMETERS)
AS
BEGIN
//#TEMP table will be created on TEMPDB
CREATE TABLE #TEMP (col1 VARCHAR (20), col2 VARCHAR (3), col3 VARCHAR (10))
//Allocated space on Tempdb for #TEMP table will be created on TEMPDB
INSERT INTO #TEMP (col1, col2, col3)
SELECT col1, col2, col3
FROM TABLE1, TABLE2 //Massive tables which has more records
WHERE (CONDITION)
/* Load of Massive records influencing on Global Resource TempDB database itself*/
```
SELECT * FROM #TEMP
Drop table #TEMP
//#TEMP table will be created and dropped on TEMPDB
END

// Creation of permanent table on user database with Session Identification number column
CREATE TABLE TEMP (col1 VARCHAR (20), col2 VARCHAR (3), col3 VARCHAR (10), SPID smallint)

// Creation of Clustered Index table with Session Identification number column
CREATE CLUSTER INDEX CIX on TEMP (SPID)

//use the created permanent table on the procedure based on the requirement
CREATE PROCEDURE PROCEDURENAME (PARAMETERS)
AS
BEGIN
/* SPID check is to ensure the data reliability for multi session*/
/*Entire load of Massive records influencing on User database itself*/
DELETE FROM TEMP WHERE SPID=@@SPID
INSERT INTO TEMP (col1, col2, col3, SPID)
SELECT col1, col2, col3,@@SPID
FROM
TABLE1, TABLE2 //Massive tables which has more records
WHERE (CONDITION)
/* SPID check is to ensure the data reliability and used in concurrent transactions during multi session*/
SELECT * FROM TEMP WHERE SPID=@@SPID
END

5.3.1 Performance Measurement
In the production database server, there are many ERP applications databases like Account Payable, Account Receivable, Shipping, Production, Finance, HRMS and Payroll.

Observed that payroll database application procedures and queries are taking more time and utilizing more server resources (CPU, IO and Memory). It leads to other application databases performance also very poor due to this server resource crunch.

Subsequently, 35 important Procedures are identified in the payroll module which causes more TempDB database load. Tuned or optimized them as per the method illustrated above.

Following are the observations of production database server performance after optimization of Queries and Stored Procedures in SQL server Instance.

**Method of Performance Measured**

SQL server Profiler tool is used to capture the duration, CPU and physical read of the procedures before and after optimizing procedures.

For this, few procedures are considered for the processes that are taken long time for execution in Payroll Module. The filter of those objects in the SQL Server profiler to capture the events respectively before and after the Query tuning is also given. Also, SQL server 2005 has the Dash board report which gives the CPU and IO intensive Queries and Stored Procedures.

**Environment Measured**

Database Size : 650 GB

Concurrent users : 120
Results Obtained

Figure 5.7 shows the performance improvement of TempDB after optimizing Queries and Stored Procedures.
Following Benefits are observed after optimizing the Queries and Stored Procedures:

- Observed the improvement of entire system performance.
- Identified and converted using this method for major stored procedures while using Temp tables, table variables and CTE’s across all the DB’s.
- OLTP related and Reports related performances have improved graciously.
- Controlled the TempDB data file growth that lead to running out of disk space and server restart.

**Figure 5.7 Performance of TempDB**
Other applications like AP, AR, Shipping, Finance and HRMS OLTP performance were improved drastically.

CPU heavy utilization has come down drastically and improved overall performance of the server.

Eliminated the cost towards the hardware resource addition to the server to obtain reasonable performance.

Restarting production server occasions has come down due to high CPU and memory utilization with respect to the TempDB.

### 5.4 SUMMARY

Implemented the load balancing on the database servers, achieved the tremendous performance improvement of web applications. Response from database server to web server has improved tremendously. The OLTP performance of the web application is improved and overall cost savings due to reduction in resource addition in web server. Web application users are experiencing optimal navigations of pages. Standby server will be utilized efficiently and it is used for data warehousing purposes for various web application users. Also, the standby server is still used for Disaster Recovery aspects and web server can be rerouted.

Since the Stored Procedures and Queries which causing more TempDB load are optimized, the OLTP performance of the production database server is improved and overall cost savings due to reduction in resource addition in the database server. CPU heavy utilization has come down drastically and improved performance of the server and other applications in the server were improved drastically.