

Architecture and Editions of Microsoft SQL Server: A Comprehensive Overview

Hareesh Edupuganti

Aptiva Corp, Oracle PL/SQL Developer, North Brunswick Township, NJ, USA

Abstract

Microsoft SQL Server is one of the most widely used relational database management systems, designed to support enterprise-level applications through robust storage, processing, and security features. Its architecture combines a relational engine for query processing, a storage engine for data management, and supporting components such as SQL OS, memory structures, and transaction logging to ensure ACID properties and system reliability. SQL Server also offers diverse editions—including Enterprise, Standard, Developer, Express, and specialized versions—that cater to different organizational needs ranging from large-scale business intelligence to lightweight embedded applications. Understanding both the internal architecture and the variety of editions is essential for database administrators and organizations in selecting the right configuration for performance, scalability, and cost-efficiency. This paper provides a comprehensive overview of SQL Server's architecture and editions, highlighting their features, advantages, and roles in database management.

Index Terms: SQL Server, database systems, architecture, editions, relational engine

I. WHAT IS SQL SERVER?

- It is a software, developed by Microsoft, which is implemented from the specification of RDBMS.
- It is also an ORDBMS.
- It is platform dependent.
- It is both GUI and command based software.
- It supports SQL (SEQUEL) language which is an IBM product, non-procedural, common database and case insensitive language.

Usage of SQL Server

- To create databases.
- To maintain databases.
- To analyze the data through SQL Server Analysis Services (SSAS).
- To generate reports through SQL Server Reporting Services (SSRS).
- To carry out ETL operations through SQL Server Integration Services (SSIS).

Versions of SQL Server

Version	Year	Code Name
6.0	1995	SQL95
6.5	1996	Hydra
7.0	1998	Sphinx
8.0 (2000)	2000	Shiloh
9.0 (2005)	2005	Yukon
10.0 (2008)	2008	Katmai
10.5 (2008 R2)	2010	Kilimanjaro
11.0 (2012)	2012	Denali
12 (2014)	2014	Hekaton (initially), SQL 14 (current)

II. SQL SERVER COMPONENTS

SQL Server works in client-server architecture, hence it supports two types of components:

(a) Workstation and (b) Server.

□ **Workstation components** are installed in every device/SQL Server operator's machine. These are just interfaces to interact with Server components. Example: SSMS, SSCM, Profiler, BIDS, SQLEM etc.

□ **Server components** are installed in centralized server. These are services. Example: SQL Server, SQL Server Agent, SSIS, SSAS, SSRS, SQL browser, SQL Server full text search etc.

Instance of SQL Server

- An **instance** is an installation of SQL Server.
- An instance is an exact copy of the same software.
- If we install 'n' times, then 'n' instances will be created.
- There are two types of instances in SQL Server a) Default b) Named.
- Only one default instance will be supported in one Server.
- Multiple named instances will be supported in one Server.
- Default instance will take the server name as Instance name.
- Default instance service name is MSSQLSERVER.
- 16 instances will be supported in 2000 version.
- 50 instances will supported in 2005 and later versions.

Advantages of Instances

- ☐ To install different versions in one machine.
- ☐ To reduce cost.
- ☐ To maintain production, development, and test environments separately.
- ☐ To reduce temporary database problems.
- ☐ To separate security privileges.
- ☐ To maintain standby server.

III. SQL SERVER – EDITIONS

SQL Server is available in various editions. This chapter lists the multiple editions with its features.

- ☐ **Enterprise:** This is the top-end edition with a full feature set.
- ☐ **Standard:** This has less features than Enterprise, when there is no requirement of advanced features.
- ☐ **Workgroup:** This is suitable for remote offices of a larger company.
- ☐ **Web:** This is designed for web applications.
- ☐ **Developer:** This is similar to Enterprise, but licensed to only one user for development, testing and demo. It can be easily upgraded to Enterprise without reinstallation.
- ☐ **Express:** This is free entry level database. It can utilize only 1 CPU and 1 GB memory, the maximum size of the database is 10 GB.
- ☐ **Compact:** This is free embedded database for mobile application development. The maximum size of the database is 4 GB.
- ☐ **Datacenter:** The major change in new SQL Server 2008 R2 is Datacenter Edition. The Datacenter edition has no memory limitation and offers support for more than 25 instances.
- ☐ **Business Intelligence:** Business Intelligence Edition is a new introduction in SQL Server 2012. This edition includes all the features in the Standard edition and support for advanced BI features such as Power View and PowerPivot, but it lacks support for advanced availability features like AlwaysOn Availability Groups and other online operations.

Enterprise Evaluation: The SQL Server Evaluation Edition is a great way to get a fully functional and free instance of SQL Server for learning and developing solutions. This edition has a built-in expiry of 6 months from the time that you install it.

IV. SQL SERVER – ARCHITECTURE

We have classified the architecture of SQL Server into the following parts for easy understanding:

- ☐ General architecture
- ☐ Memory architecture
- ☐ Data file architecture
- ☐ Log file architecture

General Architecture

Client: Where the request initiated.

Query: SQL query which is high level language.

Logical Units: Keywords, expressions and operators, etc.

N/W Packets: Network related code.

Protocols: In SQL Server we have 4 protocols.

- ☐ Shared memory (for local connections and troubleshooting purpose).
- ☐ Named pipes (for connections which are in LAN connectivity).
- ☐ TCP/IP (for connections which are in WAN connectivity).
- ☐ VIA-Virtual Interface Adapter (requires special hardware to set up by vendor and also deprecated from SQL 2012 version).

Server: Where SQL Services got installed and databases reside.

Relational Engine: This is where real execution will be done. It contains Query parser, Query optimizer and Query executor.

Query Parser (Command Parser) and Compiler (Translator): This will check syntax of the query and it will convert the query to machine language.

Query Optimizer: It will prepare the execution plan as output by taking query, statistics and Algebraizer tree as input.

Execution Plan: It is like a roadmap, which contains the order of all the steps to be performed as part of the query execution.

Query Executor: This is where the query will be executed step by step with the help of execution plan and also the storage engine will be contacted.

Storage Engine: It is responsible for storage and retrieval of data on the storage system (disk, SAN, etc.), data manipulation, locking and managing transactions.

SQL OS: This lies between the host machine (Windows OS) and SQL Server. All the activities performed on database engine are taken care of by SQL OS. SQL OS provides various operating system services, such as memory management deals with buffer pool, log buffer and deadlock detection using the blocking and locking structure.

Checkpoint Process: Checkpoint is an internal process that writes all dirty pages (modified pages) from Buffer Cache to Physical disk. Apart from this, it also writes the log records from log buffer to physical file. Writing of Dirty pages from buffer cache to data file is also known as Hardening of dirty pages.

It is a dedicated process and runs automatically by SQL Server at specific intervals. SQL Server runs checkpoint process for each database individually. Checkpoint helps to reduce the recovery time for SQL Server in the event of unexpected shutdown or system crash\Failure.

Checkpoints in SQL Server

In SQL Server 2012 there are four types of **checkpoints**:

- **Automatic:** This is the most common checkpoint which runs as a process in the background to make sure SQL Server Database can be recovered in the time limit defined by the Recovery Interval – Server Configuration Option.
- **Indirect:** This is new in SQL Server 2012. This also runs in the background but to meet a user-specified target recovery time for the specific database where the option has been configured. Once the Target_Recovery_Time for a given database has been selected, this will override the Recovery Interval specified for the server and avoid automatic checkpoint on such DB.
- **Manual:** This one runs just like any other T-SQL statement, once you issue checkpoint command it will run to its completion. Manual checkpoint runs for your current database only. You can also specify the Checkpoint_Duration which is optional - this duration specifies the time in which you want your checkpoint to complete.
- **Internal:** As a user you can't control internal checkpoint. Issued on specific operations such as:
 - Shutdown initiates a checkpoint operation on all databases except when shutdown is not clean (shutdown with nowait).
 - If the recovery model gets changed from Full\Bulk-logged to Simple.
 - While taking backup of the database.
 - If your DB is in simple recovery model, checkpoint process executes automatically either when the log becomes 70% full, or based on Server option-Recovery Interval.
 - Alter database command to add or remove a data\log file also initiates a checkpoint.
 - Checkpoint also takes place when the recovery model of the DB is bulk- logged and a minimally logged operation is performed.
 - DB Snapshot creation.

Lazy Writer Process: Lazy writer will push dirty pages to disk for an entirely different reason, because it needs to free up memory in the buffer pool. This happens when SQL server comes under memory pressure. As far as I am aware, this is controlled by an internal process and there is no setting for it.

SQL server constantly monitors memory usage to assess resource contention (or availability); its job is to make sure that there is a certain amount of free space available at all times. As part of this process, when it notices any such resource contention, it triggers Lazy Writer to free up some pages in memory by writing out dirty pages to disk. It employs Least Recently Used (LRU) algorithm to decide which pages are to be flushed to the disk.

If Lazy Writer is always active, it could indicate memory bottleneck.

V. MEMORY ARCHITECTURE

Following are some of the salient features of memory architecture.

- One of the primary design goals of all database software is to minimize disk I/O because disk reads and writes are among the most resource-intensive operations.
- Memory in windows can be called with Virtual Address Space, shared by Kernel mode (OS mode) and User mode (Application like SQL Server).
- SQL Server "User address space" is broken into two regions: MemToLeave and Buffer Pool.
- Size of MemToLeave (MTL) and Buffer Pool (BPool) is determined by SQL Server during startup.
- **Buffer management** is a key component in achieving I/O highly efficiency. The buffer management component consists of two mechanisms: the buffer manager to access and update database pages, and the buffer pool to reduce database file I/O.
- The buffer pool is further divided into multiple sections. The most important ones being the buffer cache (also referred to as data cache) and procedure cache. **Buffer cache** holds the data pages in memory so that frequently accessed data can be retrieved from cache. The alternative would be reading data pages from the disk. Reading data pages from cache optimizes performance by minimizing the number of required I/O operations which are inherently slower than retrieving data from the memory.
- **Procedure cache** keeps the stored procedure and query execution plans to minimize the number of times that query plans have to be generated. You can find out information about the size and activity within the procedure cache using DBCC PROCCACHE statement.

Other portions of buffer pool include:

- **System level data structures** – Holds SQL Server instance level data about databases and locks.
- **Connection context** – Each connection to the instance has a small area of memory to record the current state of the connection. This information includes stored procedure and user-defined function parameters, cursor positions and more.
- **Stack space** – Windows allocates stack space for each thread started by SQL Server.

VI. DATA FILE ARCHITECTURE

Data File architecture has the following components:

File Groups

Database files can be grouped together in file groups for allocation and administration purposes. No file can be a member of more than one file group. Log files are never part of a file group. Log space is managed separately from data space.

There are two types of file groups in SQL Server, Primary and User-defined. Primary file group contains the primary data file and any other files not specifically assigned to another file group. All pages for the system tables are allocated in the primary file group. User-defined file groups are any file groups specified using the file group keyword in create database or alter database statement.

One file group in each database operates as the default file group. When SQL Server allocates a page to a table or index for which no file group was specified when they were created, the pages are allocated from default file group. To switch the default file group from one file group to another file group, it should have db_owner fixed db role.

By default, primary file group is the default file group. User should have db_owner fixed database role in order to take backup of files and file groups individually.

Files

Databases have three types of files - Primary data file, Secondary data file, and Log file. Primary data file is the starting point of the database and points to the other files in the database.

Every database has one primary data file. We can give any extension for the primary data file but the recommended extension is **.mdf**. Secondary data file is a file other than the primary data file in that database. Some databases may have multiple secondary data files. Some databases may not have a single secondary data file. Recommended extension for secondary data file is **.ndf**.

Log files hold all of the log information used to recover the database. Database must have at least one log file. We can have multiple log files for one database. The recommended extension for log file is **.ldf**.

The location of all the files in a database are recorded in both master database and the primary file for the database. Most of the time, the database engine uses the file location from the master database.

Files have two names - Logical and Physical. Logical name is used to refer to the file in all T-SQL statements. Physical name is the OS_file_name, it must follow the rules of OS. Data and Log files can be placed on either FAT or NTFS file systems, but cannot be placed on compressed file systems. There can be up to 32,767 files in one database.

Extents

Extents are basic unit in which space is allocated to tables and indexes. An extent is 8 contiguous pages or 64KB. SQL Server has two types of extents - Uniform and Mixed. Uniform extents are made up of only single object. Mixed extents are shared by up to eight objects.

Pages

It is the fundamental unit of data storage in MS SQL Server. The size of the page is 8KB. The start of each page is 96 byte header used to store system information such as type of page, amount of free space on the page and object id of the object owning the page. There are 9 types of data pages in SQL Server.

- ☐ **Data** - Data rows with all data except text, ntext and image data.
- ☐ **Index** - Index entries.
- ☐ **Text/Image** - Text, image and ntext data.
- ☐ **GAM** - Information about allocated extents.
- ☐ **SGAM** - Information about allocated extents at system level.
- ☐ **Page Free Space (PFS)** - Information about free space available on pages.
- ☐ **Index Allocation Map (IAM)** - Information about extents used by a table or index.
- ☐ **Bulk Changed Map (BCM)** - Information about extents modified by bulk operations since the last backup log statement.
- ☐ **Differential Changed Map (DCM)** - Information about extents that have changed since the last backup database statement.

VII. LOG FILE ARCHITECTURE

The SQL Server transaction log operates logically as if the transaction log is a string of log records. Each log record is identified by Log Sequence Number (LSN). Each log record contains the ID of the transaction that it belongs to.

Log records for data modifications record either the logical operation performed or they record the before and after images of the modified data. The before image is a copy of the data before the operation is performed; the after image is a copy of the data after the operation has been performed.

The steps to recover an operation depend on the type of log record:

- ☐ Logical operation logged.
 - To roll the logical operation forward, the operation is performed again.
 - To roll the logical operation back, the reverse logical operation is performed.
- ☐ Before and after image logged.
 - To roll the operation forward, the after image is applied.
 - To roll the operation back, the before image is applied.

Different types of operations are recorded in the transaction log. These operations include:

- ☐ The start and end of each transaction.
- ☐ Every data modification (insert, update, or delete). This includes changes by system stored procedures or data definition language (DDL) statements to any table, including system tables.
- ☐ Every extent and page allocation or de allocation.
- ☐ Creating or dropping a table or index.

Rollback operations are also logged. Each transaction reserves space on the transaction log to make sure that enough log space exists to support a rollback that is caused by either an explicit rollback statement or if an error is encountered. This reserved space is freed when the transaction is completed.

The section of the log file from the first log record that must be present for a successful database-wide rollback to the last-written log record is called the active part of the log, or the active log. This is the section of the log required to a full recovery of the database. No part of the active log can ever be truncated. LSN of this first log record is known as the minimum recovery LSN (Min LSN).

VIII. CONCLUSION

The Database Engine chooses the size of the virtual log files dynamically while it is creating or extending log files. The Database Engine tries to maintain a small number of virtual files. The size or number of virtual log files cannot be configured or set by administrators. The only time virtual log files affect system performance is if the physical log files are defined by small size and growth_increment values. The size value is the initial size for the log file and the growth_increment value is the amount of space added to the file every time new space is required. If the log files grow to a large size because of many small increments, they will have many virtual log files. This can slow down database startup and also log backup and restore operations.

REFERENCES

- [1] Blaha, Michael R. *A Manager's Guide to Database Technology*. Upper Saddle River, NJ: Prentice Hall (2001). ISBN 0-13-030418-2.
- [2] Connolly, Thomas, and Carolyn Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management*. 4th ed. Harlow, England: Addison-Wesley (2004). ISBN 978-0-321-29401-2.