Q28)

A transaction is a sequence of operations performed as a single logical unit of work, and it must comply the ACID properties(atomicity, consistency, isolation, and durability).

The SQL Server Database Engine mainly uses two mechanisms to ensure the integrity of transactions, maintain the consistency of databases when multiple users are accessing data at the same time, and avoid bad concurrency effects. The first mechanism is locking, in which each transaction requests locks of different types on the resources on which the transaction is dependent. The lock blocks other transactions from modifying the resources and it will be freed when the transaction no longer has a dependency on the locked resources. The second mechanism is row versioning, in which the SQL Server Database Engine maintains versions of each row that is modified and applications can specify that a transaction use the row versions to view data as it existed at the start of the transaction or query instead of protecting all reads with locks.

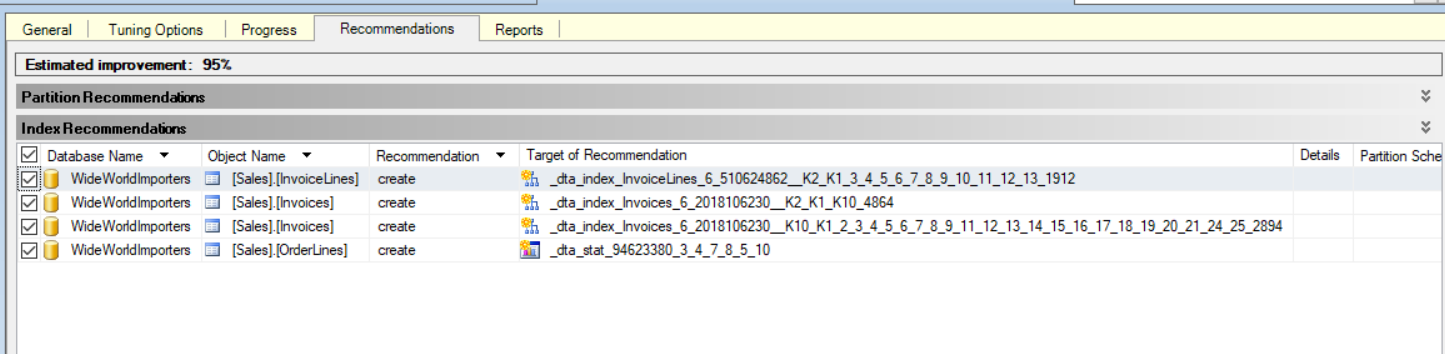
Transactions specify an isolation level that defines the degree to which one transaction must be isolated from resource or data modifications made by other transactions. The isolation level controls whether locks are taken when data is read and what type of locks are requested, how long the read locks are held, and the actions when a read operation referencing rows modified by another transaction. There are four isolation levels, which are, from the lowest to the highest, Read Uncommitted, Read Committed, Repeatable Read, Serializable. Choosing the appropriate isolation level depends on balancing the data integrity requirements of the application against the overhead of each isolation level.

Locking is a mechanism used to synchronize access by multiple users to the same piece of data at the same time. Before a transaction acquires a dependency on the current state of a piece of data, such as by reading or modifying the data, it must protect itself from the effects of another transaction modifying the same data. The transaction does this by requesting a lock on the piece of data. Locks have different modes, including shared, update, exclusive, which are used in different operations. The lock mode defines the level of dependency the transaction has on the data. No transaction can be granted a lock that would conflict with the mode of a lock already granted on that data to another transaction. If a transaction requests a lock mode that conflicts with a lock that has already been granted on the same data, the instance of the SQL Server Database Engine will pause the requesting transaction until the first lock is released. When a transaction modifies a piece of data, it holds the lock protecting the modification until the end of the transaction. How long a transaction holds the locks acquired to protect read operations depends on the transaction isolation level setting. All locks held by a transaction are released when the transaction completes.

Q29)

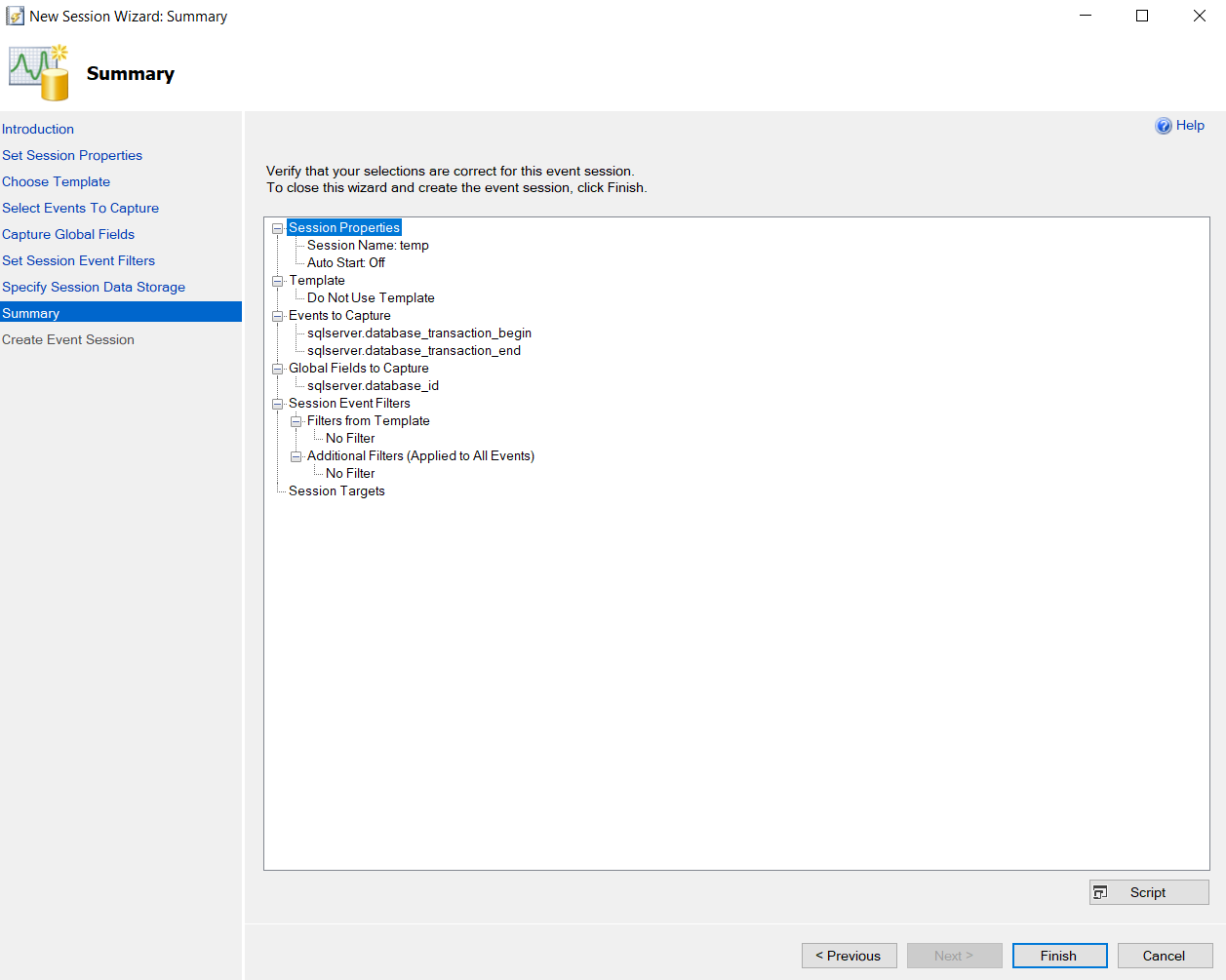
Tuning Advisor analyzes databases and makes recommendations that we can use to optimize query performance. We can use it to select and create an optimal set of indexes, indexed views, or table partitions without having an expert understanding of the database structure or the internals of SQL Server. It can help us to perform the tasks including troubleshoot the performance of a specific problem query, tune a large set of queries across one or more databases, perform an exploratory what-if analysis of potential physical design changes, and manage storage space.

In the following picture, we could see that the recommendations given by the Tuning Advisor could give us an estimated improvement of 95%.



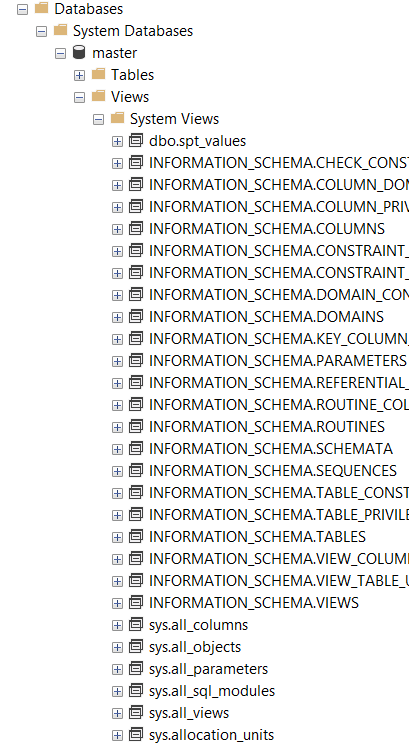
Extended events is a performance monitoring system that enables users to collect data needed to monitor and troubleshoot problems in SQL Server. By using extended events, we can see details about the inner operations of the SQL system and the application. We could select which occurrences we are interested in and how we want the system to report the data.

The following picture shows how we could create a new session to monitor transaction-related events.

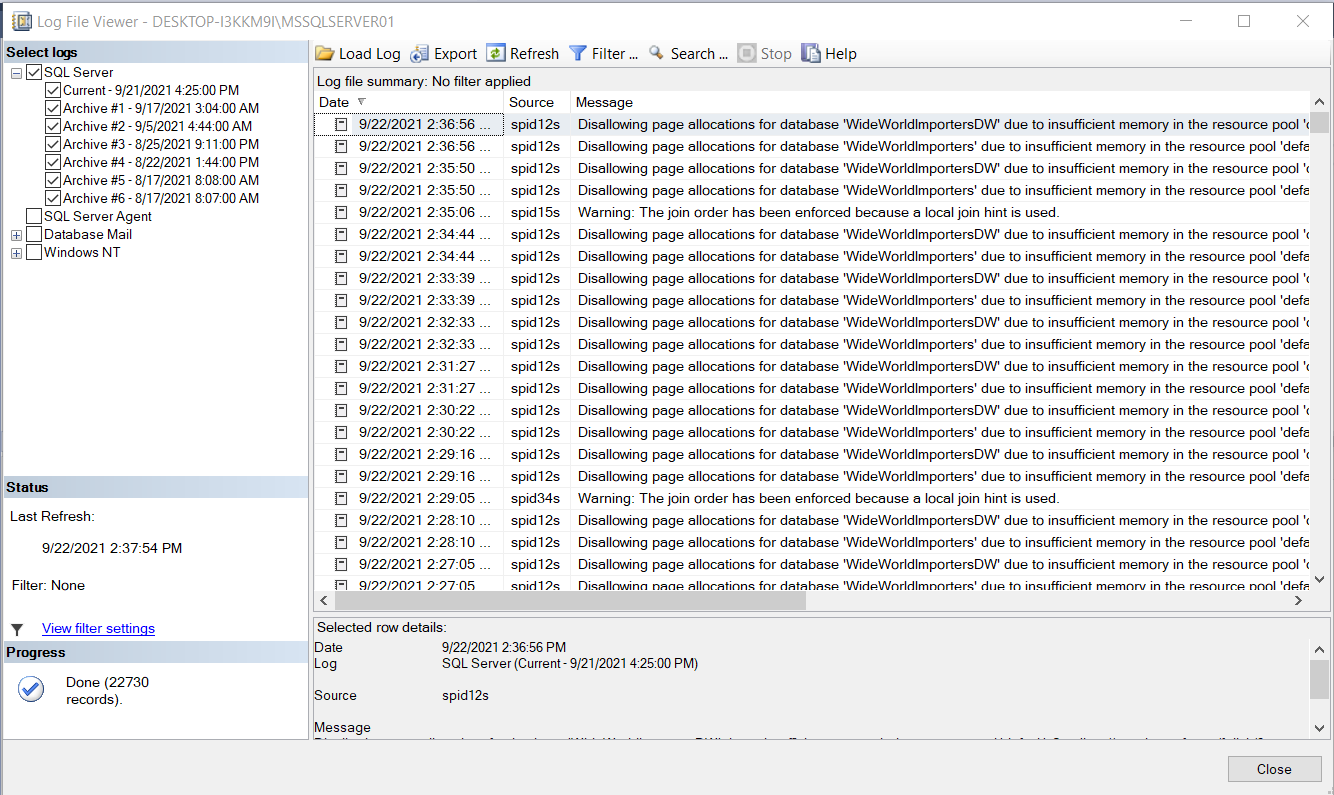


Dynamic management views(DMV) return server state information that can be used to monitor the health of a server instance, diagnose problems, and tune performance. There are several DMVs that provide data about query stats, execution plans, and recent queries. We could use them to monitor the performance by, for example, finding the queries that use the most reads, writes, etc.

Some of the DMVs are shown in the picture below.



The SQL Server error log contains user-defined events and certain system events we can use for troubleshooting. With the Log File Viewer, we could obtain the log files for the whole server.



We could use Execution plans to determine the most efficient way to complete an operation. An execution plan consists of the sequence in which the source tables are accessed, the methods used to extract data from each table, the methods used to compute calculations, and how to filter, aggregate, and sort data from each table. SSMS has three options to display execution plans: The Estimated Execution Plan, The Actual Execution Plan, The Live Query Statistics. The Actual Execution Plan is available after the query execution is completed. It includes actual runtime information such as execution warnings, the elapsed and CPU time used during execution.

The following picture shows how we could compare the query cost of two queries that perform the same operation, then we could choose to go with the query with less cost.

