**Question 28**

**Transaction**

A transaction is a sequence of operations performed as single logical unit of work. It is usually important to maintain transaction in a database system with many users, otherwise, performance and efficiency problems occur which cause serious trouble for the system and database. Transactions in a database management system should meet four properties, atomicity, consistency, isolation and durability. With these four properties, a transaction can efficiently modify or manipulate the database and ensure database integrity.

Database programmers are responsible for starting and ending transactions at points that enforce the logical consistency of the data. Database system is responsible for ensuring the physical integrity of each transaction. The transaction management components of a database system allow application users to focus on writing correct transaction statements without worrying about the concurrency and all other problems.

There are three different types of transactions, explicit transaction, autocommit transaction and autocommit transaction. Explicit transaction is one in which users explicitly define both the start and end of the transaction. Autocommit transaction is the default mode, it completes after a statement is executed either successfully or failed. Implicit transaction do not need to specify the start of the transaction, another transaction (all DML statements) automatically starts after the former transaction finishes.

**Locking**

Each transaction requests locks of different types on the resources, such as rows, pages, or tables. The locks block other transactions from modifying the resources. The database system uses locking to ensure the integrity of transactions and maintains the consistency of database when multiple users accessing data at the same time. A few important locks, including shared lock, update lock, and exclusive lock. Locks are used in concurrency control, often called pessimistic concurrency control. It is mainly used in situations where the cost of protecting the data is more than the cost of rolling back the data.

**Isolation levels**

Transaction isolation level defines the level of protection for the transaction from modifications made by other transactions. A lower isolation level increases the ability of many users to access data at the same time but increases the number of concurrency side effects. Whereas a higher isolation level increases the restriction of users to access data but decreases the concurrency side effects.

Isolation level for locks

Read Uncommitted, Read Committed, Repeatable read, Serializable

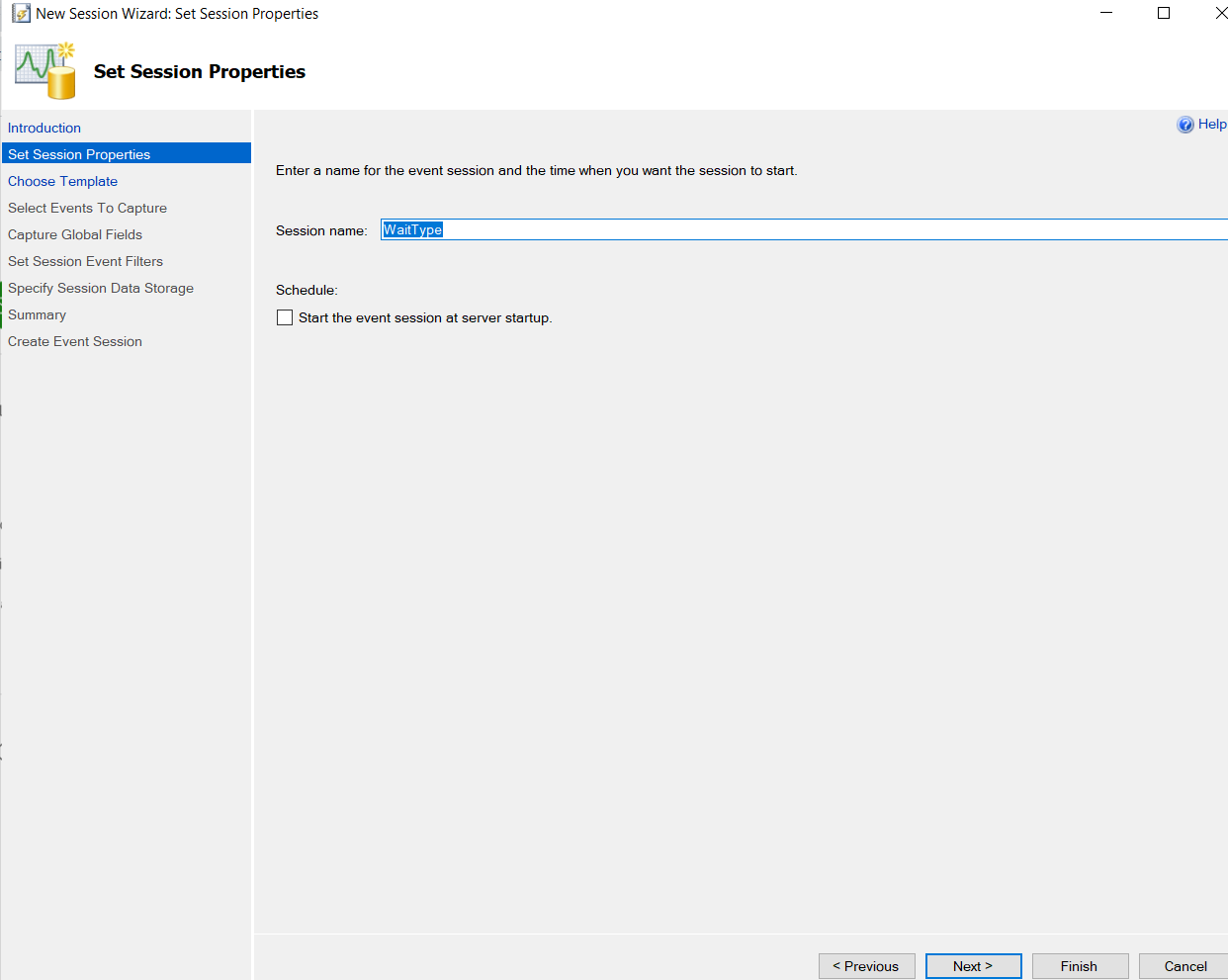
Isolation level for row-versioning: Read Committed Snapshot, Snapshot, in memory olpt

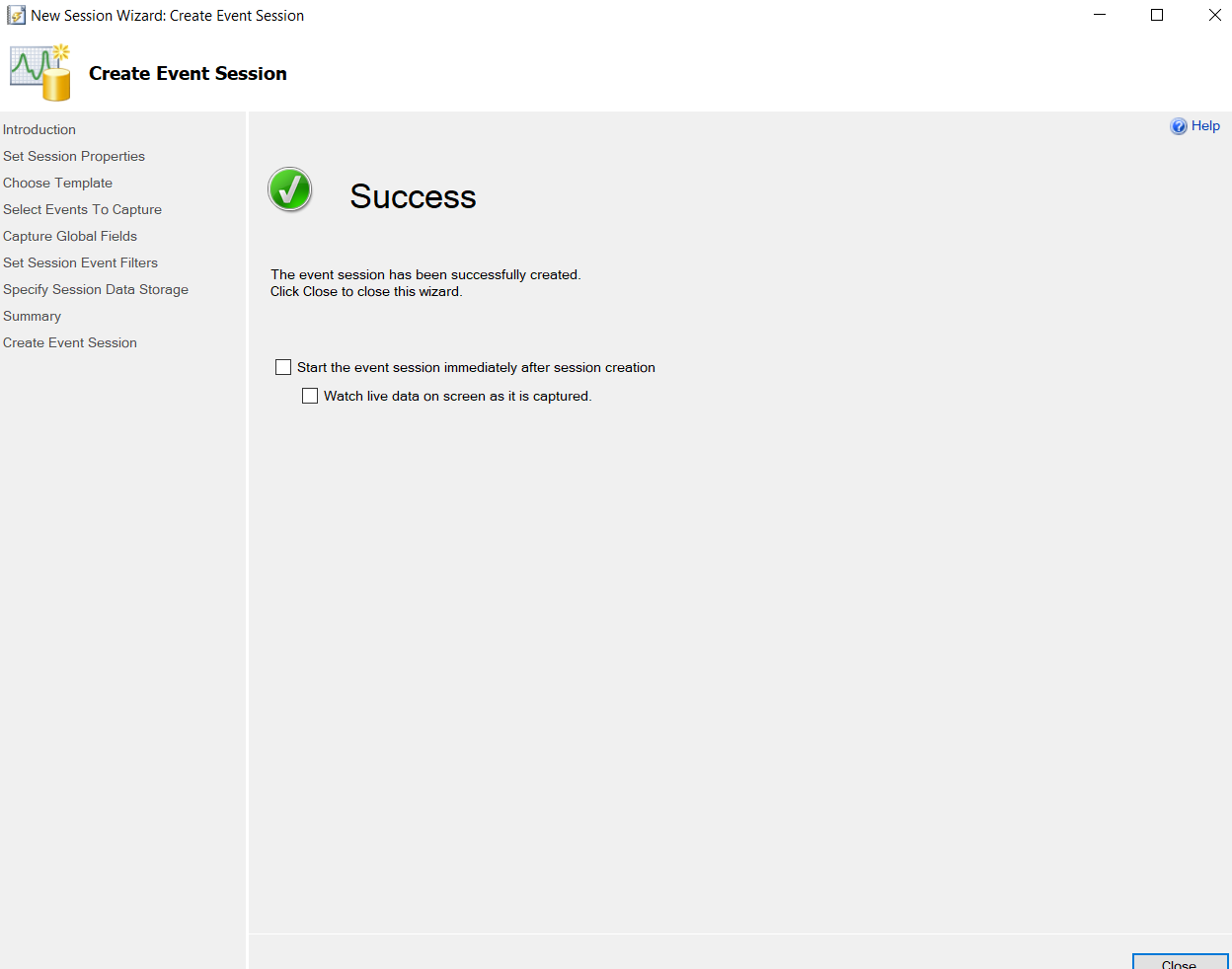
In optimistic concurrency control, users do not lock data when they read it. When a user updates data, the system checks to see if another user changed the data after it was read. If another user updated the data, an error is raised. Typically, the user receiving the error rolls back the transaction and starts over.

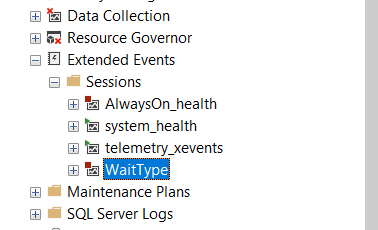
**Question 29**

As the size of the database increases, the speed of query may decrease. This is when performance monitor and tuning come into play. Performance tuning is the process of ensuring that the SQL statements run in the fastest possible time. QL Server provides various of tools for monitoring and improving performance. Those tools include Dynamic Management Views (DMV), Extended Events, Logs, DBCC (Trace Flags), System stored procedures and Execution plan. Besides, we as programmer can also improve query performance by taking advantage of or avoiding certain behaviors, such as improving table indexing, avoiding using correlated subquery and temp tables and so on.

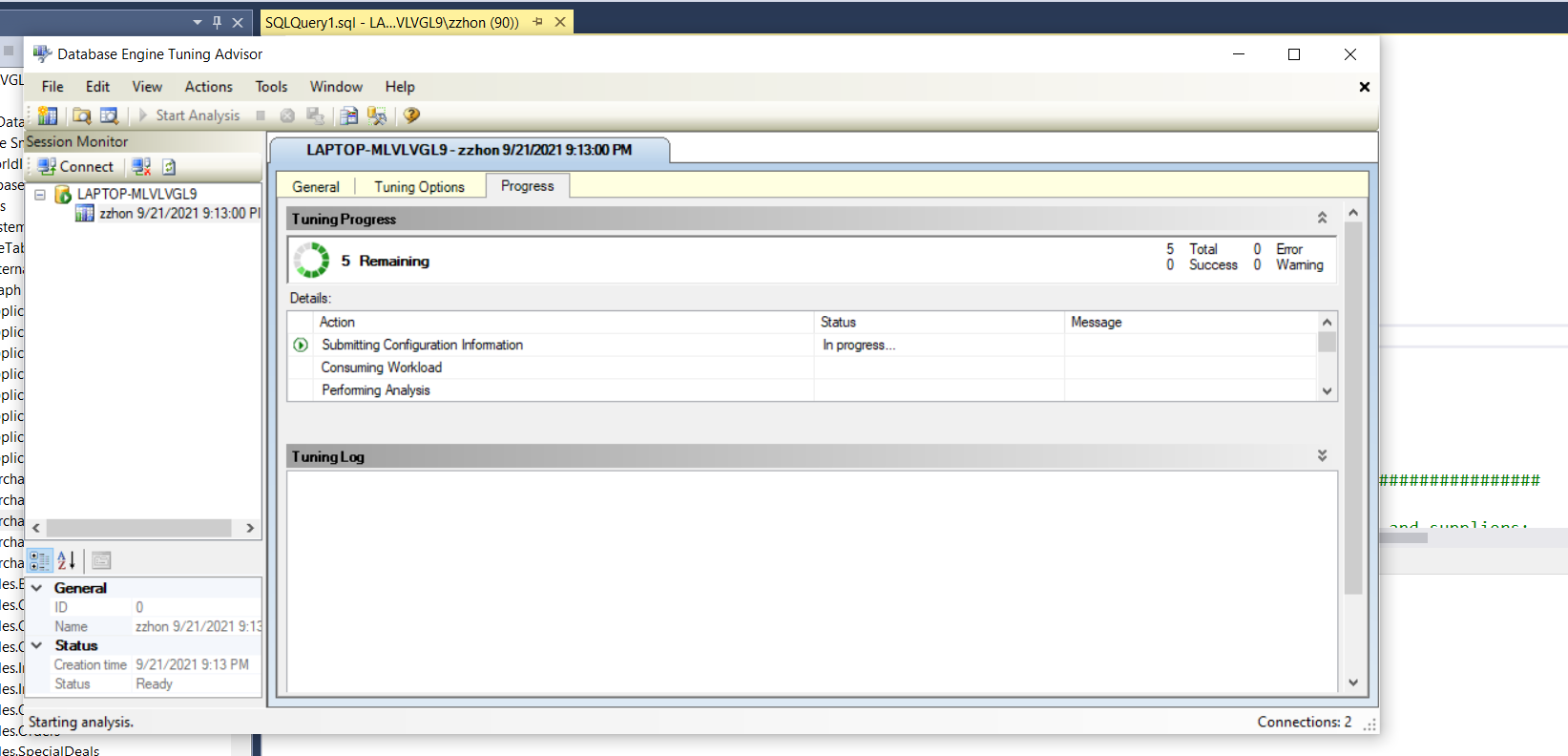
Extended events, which is a lightweight performance monitor system that enables users to collect data needed to monitor and troubleshooting problems in SQL Server. By using extended events, we can see details about the inner operations of the SQL system and your application. We tell the system what occurrences we are interested in and how we want the system to return the data to us.





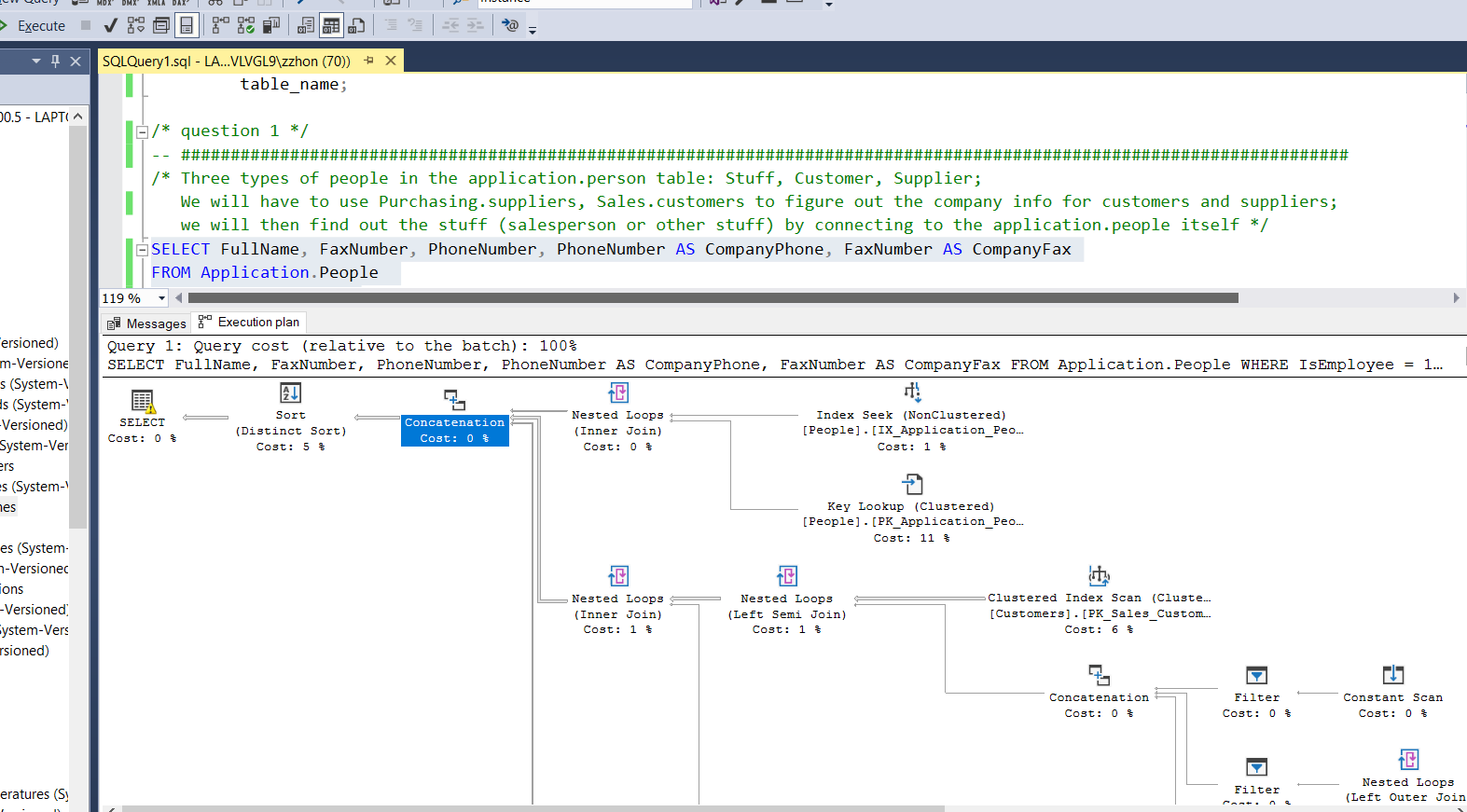


Tuning advisor makes optimize performance easier by analyzing the current query plan cache or by analyzing a workload of transact-sql queries that you create and recommending an appropriate physical design.



Dynamic management views return server state information that can be used to monitor the health of a server instance, diagnose problems and turn performance.

An execution plan shows the processing strategy used by the query optimizer to execute a query. It helps a user to determine why queries are taking more time to execute and based on the investigation, users can update their queries for better performance. Query Analyzer has an option, called “Display Estimated Execution Plan” (located on the Query drop-down menu). If this option is turned on, it will display a query execution plan in a separate window when the query runs again.



The Windows application event log provides an overall picture of events occurring on the Windows Server and Windows operating systems as a whole, as well as events in SQL Server, SQL Server Agent, and full-text search. It contains information about events in SQL Server that is not available elsewhere. You can use the information in the error log to troubleshoot SQL Server-related problems. Log and data files need to be stored on separate drives as the writing and accessing of log files is sequential, whereas writing and accessing of data files is non-sequential. Storing log and data files separately onto other physical drives can enhance the system’s performance levels.