Table of Contents

[Binary Large Object (Blob) Data 19](#_Toc510809813)

[Options for Storing Blobs 19](#_Toc510809814)

[FILESTREAM (SQL Server) 19](#_Toc510809815)

[FileTables (SQL Server) 19](#_Toc510809816)

[Remote Blob Store (RBS) 19](#_Toc510809817)

[Why RBS? 19](#_Toc510809818)

[Optimized database storage and performance 19](#_Toc510809819)

[Efficient BLOB management 19](#_Toc510809820)

[Comparing FILESTREAM and FileTable 20](#_Toc510809821)

[When to Use FILESTREAM 20](#_Toc510809822)

[FILESTREAM Storage 21](#_Toc510809823)

[Accessing BLOB Data with Transact-SQL and File System Streaming Access 21](#_Toc510809824)

[Transact-SQL Access 21](#_Toc510809825)

[Inserting a Row That Contains FILESTREAM Data 22](#_Toc510809826)

[Benefits of the FileTable Feature 24](#_Toc510809827)

[What Is a FileTable? 25](#_Toc510809828)

[Collations 28](#_Toc510809829)

[Definition: 28](#_Toc510809830)

[SQL Server Configuration Manager 29](#_Toc510809831)

[Configure on Windows 30](#_Toc510809832)

[Lock Pages in Memory (LPIM) 31](#_Toc510809833)

[Cursors 32](#_Toc510809834)

[Data Collection 35](#_Toc510809835)

[What is Data Collector? 35](#_Toc510809836)

[Terminology 35](#_Toc510809837)

[System Data Collection Set Reports 37](#_Toc510809838)

[Data Compression 38](#_Toc510809839)

[Database Mail 40](#_Toc510809840)

[Benefits of using Database Mail 40](#_Toc510809841)

[Reliability 40](#_Toc510809842)

[Scalability 40](#_Toc510809843)

[Security 40](#_Toc510809844)

[Supportability 41](#_Toc510809845)

[Database Mail Architecture 41](#_Toc510809846)

[Database Mail Configuration Object Relationship 42](#_Toc510809847)

[Database Mail Account 42](#_Toc510809848)

[Database Mail Profile 43](#_Toc510809849)

[Databases 44](#_Toc510809850)

[System Databases 44](#_Toc510809851)

[Data Types 45](#_Toc510809852)

[Data type categories 45](#_Toc510809853)

[Exact numerics 45](#_Toc510809854)

[Approximate numerics 46](#_Toc510809855)

[Date and time 46](#_Toc510809856)

[Character strings 46](#_Toc510809857)

[Unicode character strings 46](#_Toc510809858)

[Binary strings 46](#_Toc510809859)

[Other data types 46](#_Toc510809860)

[Constants 46](#_Toc510809861)

[Data type conversion 47](#_Toc510809862)

[Implicit and explicit conversion 47](#_Toc510809863)

[Data type precedence 47](#_Toc510809864)

[Bit: 48](#_Toc510809865)

[Cursor : 48](#_Toc510809866)

[Date and time types 48](#_Toc510809867)

[Date : 48](#_Toc510809868)

[DateTime: 48](#_Toc510809869)

[DateTime2: 49](#_Toc510809870)

[DateTimeOffset: 49](#_Toc510809871)

[Smalldatetime: 49](#_Toc510809872)

[Time: 49](#_Toc510809873)

[Numeric types 49](#_Toc510809874)

[decimal and numeric 49](#_Toc510809875)

[float and real 49](#_Toc510809876)

[int, bigint, smallint, and tinyint 50](#_Toc510809877)

[money and smallmoney 50](#_Toc510809878)

[Rowversion 50](#_Toc510809879)

[String and binary 50](#_Toc510809880)

[binary and varbinary 50](#_Toc510809881)

[char and varchar 50](#_Toc510809882)

[nchar and nvarchar 51](#_Toc510809883)

[ntext, text, and image 51](#_Toc510809884)

[Spatial geography 52](#_Toc510809885)

[Spatial geometry 52](#_Toc510809886)

[sql\_variant 52](#_Toc510809887)

[Table 52](#_Toc510809888)

[Uniqueidentifier 52](#_Toc510809889)

[XML 53](#_Toc510809890)

[DBCC (Database Console Commands) 54](#_Toc510809891)

[Extended Events 55](#_Toc510809892)

[Benefits of SQL Server Extended Events 55](#_Toc510809893)

[FULLTEXT CATALOG 56](#_Toc510809894)

[FULLTEXT INDEX 56](#_Toc510809895)

[Full-text Search 58](#_Toc510809896)

[Compare Full-Text Search queries to the LIKE predicate 58](#_Toc510809897)

[Full-Text Search architecture 59](#_Toc510809898)

[SQL Server process 59](#_Toc510809899)

[Filter Daemon Host process 61](#_Toc510809900)

[Full-Text Search processing 61](#_Toc510809901)

[Full-Text indexing process 61](#_Toc510809902)

[Full-Text querying process 62](#_Toc510809903)

[Full-text index architecture 62](#_Toc510809904)

[Differences between full-text indexes and regular SQL Server indexes:. 63](#_Toc510809905)

[Full-Text search linguistic components and language support 63](#_Toc510809906)

[Set up full-text search in two steps 64](#_Toc510809907)

[Choose options for a full-text index 65](#_Toc510809908)

[Choose a language 65](#_Toc510809909)

[Choose a filegroup 65](#_Toc510809910)

[Choose a full-text catalog 65](#_Toc510809911)

[Associate a unique index 65](#_Toc510809912)

[Associate a stoplist 66](#_Toc510809913)

[A stoplist is a list of stopwords, also known as noise words. A stoplist is associated with each full-text index, and the words in that stoplist are applied to full-text queries on that index. By default, the system stoplist is associated with a new full-text index. You can create and use your own stoplist too 66](#_Toc510809914)

[CREATE FULLTEXT STOPLIST myStoplist FROM SYSTEM STOPLIST; 66](#_Toc510809915)

[ALTER FULLTEXT STOPLIST myStoplist ADD 'en' LANGUAGE 'Spanish'; 66](#_Toc510809916)

[Update a full-text index 66](#_Toc510809917)

[Query with Full-Text Search 66](#_Toc510809918)

[Simple examples of each predicate and function 66](#_Toc510809919)

[Graph processing with SQL Server 69](#_Toc510809920)

[What is a graph database? 69](#_Toc510809921)

[When to use a graph database 69](#_Toc510809922)

[SQL Graph Architecture 69](#_Toc510809923)

[Sample Schema 70](#_Toc510809924)

[Sample Script 71](#_Toc510809925)

[Hierarchical Data 73](#_Toc510809926)

[Key Properties of hierarchyid 73](#_Toc510809927)

[Limitations of hierarchyid 73](#_Toc510809928)

[When to Use Alternatives to hierarchyid 74](#_Toc510809929)

[Parent/Child 74](#_Toc510809930)

[Indexing Strategies for Hierarchical Data 74](#_Toc510809931)

[hierarchyid methods 75](#_Toc510809932)

[GetAncestor 75](#_Toc510809933)

[GetDescendant 75](#_Toc510809934)

[GetLevel 75](#_Toc510809935)

[GetRoot 75](#_Toc510809936)

[IsDescendantOf (Database Engine) 75](#_Toc510809937)

[Parse (Database Engine) 75](#_Toc510809938)

[Read (Database Engine) 76](#_Toc510809939)

[GetReparentedValue (Database Engine) 76](#_Toc510809940)

[ToString 76](#_Toc510809941)

[Write (Database Engine) 76](#_Toc510809942)

[Import and export data from SQL Server and Azure SQL Database 77](#_Toc510809943)

[Methods for importing and exporting data 77](#_Toc510809944)

[Use Transact-SQL statements 77](#_Toc510809945)

[Import a JSON document into a single column 79](#_Toc510809946)

[Import multiple JSON documents 79](#_Toc510809947)

[Use BCP from the command prompt 79](#_Toc510809948)

[Use the Import Flat File Wizard 81](#_Toc510809949)

[Feature in SQL 2017 from Database -> Task -> Import flat file 81](#_Toc510809950)

[Use the SQL Server Import and Export Wizard 81](#_Toc510809951)

[In-Memory OLTP (In-Memory Optimization) 82](#_Toc510809952)

[INDEXES 87](#_Toc510809953)

[Overview 87](#_Toc510809954)

[Heaps (Tables without Clustered Indexes) 88](#_Toc510809955)

[Clustered and Nonclustered Indexes Described 89](#_Toc510809956)

[Clustered 89](#_Toc510809957)

[Nonclustered 90](#_Toc510809958)

[Create Unique Indexes 90](#_Toc510809959)

[Create Filtered Indexes 91](#_Toc510809960)

[Create Indexes with Included Columns 92](#_Toc510809961)

[Delete an Index 93](#_Toc510809962)

[Modify an Index 93](#_Toc510809963)

[Move an Existing Index to a Different Filegroup 94](#_Toc510809964)

[Indexes on Computed Columns 94](#_Toc510809965)

[SORT\_IN\_TEMPDB Option for Indexes 94](#_Toc510809966)

[Phases of Index Building 94](#_Toc510809967)

[Disable Indexes and Constraints 95](#_Toc510809968)

[Enable Indexes and Constraints 95](#_Toc510809969)

[Rename Indexes 96](#_Toc510809970)

[Set Index Options 96](#_Toc510809971)

[Disk Space Requirements for Index DDL Operations 96](#_Toc510809972)

[Index Operations That Require No Additional Disk Space 96](#_Toc510809973)

[Index Operations That Require Additional Disk Space 96](#_Toc510809974)

[Temporary Disk Space for Sorting 97](#_Toc510809975)

[Temporary Disk Space for Online Index Operations 97](#_Toc510809976)

[Transaction Log Disk Space for Index Operations 97](#_Toc510809977)

[Index Disk Space Example 98](#_Toc510809978)

[Reorganize and Rebuild Indexes 98](#_Toc510809979)

[Specify Fill Factor for an Index 98](#_Toc510809980)

[Performance Considerations 99](#_Toc510809981)

[Page Splits 99](#_Toc510809982)

[Adding Data to the End of the Table 99](#_Toc510809983)

[Perform Index Operations Online 99](#_Toc510809984)

[How Online Index Operations Work 100](#_Toc510809985)

[Source and pre-existing indexes 100](#_Toc510809986)

[Target 100](#_Toc510809987)

[Temporary mapping index 100](#_Toc510809988)

[Guidelines for Online Index Operations 101](#_Toc510809989)

[Configure Parallel Index Operations 102](#_Toc510809990)

[Index Properties F1 Help 103](#_Toc510809991)

[Columnstore indexes 103](#_Toc510809992)

[Index Design Guidelines 106](#_Toc510809993)

[Clustered Index Design Guidelines 110](#_Toc510809994)

[Clustered Index Architecture 110](#_Toc510809995)

[Query Considerations 111](#_Toc510809996)

[Column Considerations 112](#_Toc510809997)

[Nonclustered Index Design Guidelines 113](#_Toc510809998)

[Nonclustered Index Architecture 113](#_Toc510809999)

[Database Considerations 114](#_Toc510810000)

[Query Considerations 115](#_Toc510810001)

[Column Considerations 115](#_Toc510810002)

[Use Included Columns to Extend Nonclustered Indexes 116](#_Toc510810003)

[Performance Considerations 117](#_Toc510810004)

[Unique Index Design Guidelines 117](#_Toc510810005)

[SET OPERATORS 121](#_Toc510810006)

[Date Time Statements 121](#_Toc510810007)

[Locking Statements 121](#_Toc510810008)

[Miscellaneous statements 121](#_Toc510810009)

[Query Execution Statements 124](#_Toc510810010)

[ISO Settings statements 128](#_Toc510810011)

[Statistics statements 130](#_Toc510810012)

[Transactions statements 133](#_Toc510810013)

[PARTITIONS 135](#_Toc510810014)

[Benefits OF Partitioning: 135](#_Toc510810015)

[Components and Concepts 135](#_Toc510810016)

[Performance Guidelines 136](#_Toc510810017)

[PERFORMANCE 138](#_Toc510810018)

[Configuration Options for Performance 138](#_Toc510810019)

[Query Performance Options 139](#_Toc510810020)

[Performance Center for SQL Server Database Engine and Azure SQL Database 141](#_Toc510810021)

[Configuring Storage Spaces with a NVDIMM-N write-back cache 141](#_Toc510810022)

[Adaptive query processing in SQL databases 141](#_Toc510810023)

[How to enable adaptive query processing 142](#_Toc510810024)

[Batch mode adaptive joins 143](#_Toc510810025)

[Interleaved execution 143](#_Toc510810026)

[Interleaved execution for multi-statement table valued functions 143](#_Toc510810027)

[Automatic tuning 143](#_Toc510810028)

[*What is SQL plan choice regression*? 144](#_Toc510810029)

[Automatic index management 144](#_Toc510810030)

[Cardinality Estimation 145](#_Toc510810031)

[Monitor and Tune for Performance 145](#_Toc510810032)

[Monitoring and tuning databases for performance 145](#_Toc510810033)

[Monitor SQL Server Components 145](#_Toc510810034)

[Performance Monitoring and Tuning Tools 146](#_Toc510810035)

[Establish a Performance Baseline 147](#_Toc510810036)

[Isolate Performance Problems 148](#_Toc510810037)

[Identify Bottlenecks 148](#_Toc510810038)

[Server Performance and Activity Monitoring 148](#_Toc510810039)

[Live Query Statistics 149](#_Toc510810040)

[Monitoring Performance By Using the Query Store 149](#_Toc510810041)

[ALTER DATABASE AdventureWorks2012 SET QUERY\_STORE = ON; 150](#_Toc510810042)

[Using the Query Store with In-Memory OLTP 150](#_Toc510810043)

[Best Practice with the Query Store 150](#_Toc510810044)

[Query Store Usage Scenarios 150](#_Toc510810045)

[How Query Store Collects Data 152](#_Toc510810046)

[Database Tuning Advisor 153](#_Toc510810047)

[Database Engine Tuning Advisor Benefits 153](#_Toc510810048)

[Limitations and Restrictions 154](#_Toc510810049)

[Plan Guides 154](#_Toc510810050)

[ Create a New Plan Guide 155](#_Toc510810051)

[ Create a Plan Guide for Parameterized Queries 155](#_Toc510810052)

[ Specify Query Parameterization Behavior by Using Plan Guides 155](#_Toc510810053)

[ Apply a Fixed Query Plan to a Plan Guide 155](#_Toc510810054)

[ Attach Query Hints to a Plan Guide 155](#_Toc510810055)

[ View Plan Guide Properties 155](#_Toc510810056)

[ Use SQL Server Profiler to Create and Test Plan Guides 155](#_Toc510810057)

[ Validate Plan Guides After Upgrade 155](#_Toc510810058)

[ Delete a Plan Guide 155](#_Toc510810059)

[ Enable or Disable a Plan Guide 155](#_Toc510810060)

[ Monitor Resource Usage (System Monitor) 155](#_Toc510810061)

[Policy-based management 156](#_Toc510810062)

[Three Policy-Based Management components 156](#_Toc510810063)

[Terms 157](#_Toc510810064)

[PolyBase 157](#_Toc510810065)

[Why use PolyBase? 158](#_Toc510810066)

[PolyBase scale-out groups 159](#_Toc510810067)

[Heead node 159](#_Toc510810068)

[Compute node 160](#_Toc510810069)

[Distributed query processing 160](#_Toc510810070)

[Showplan Logical and Physical Operators 162](#_Toc510810071)

[Operator Descriptions 162](#_Toc510810072)

[Sequence Numbers 180](#_Toc510810073)

[Using Sequences 180](#_Toc510810074)

[Limitations 180](#_Toc510810075)

[Statistics 182](#_Toc510810076)

[Histogram 182](#_Toc510810077)

[When to create statistics 182](#_Toc510810078)

[Queries that use statistics effectively 183](#_Toc510810079)

[Improving cardinality estimates for expressions 183](#_Toc510810080)

[Improving cardinality estimates for variables and functions 183](#_Toc510810081)

[Improving cardinality estimates with query hints 183](#_Toc510810082)

[Improving cardinality estimates with Plan Guides 184](#_Toc510810083)

[Create Statistics 184](#_Toc510810084)

[DROP STATISTICS 184](#_Toc510810085)

[View Statistics Properties 184](#_Toc510810086)

[Rename Statistics 184](#_Toc510810087)

[Update Statistics 184](#_Toc510810088)

[Stored Procedures 185](#_Toc510810089)

[Benefits of Using Stored Procedures 185](#_Toc510810090)

[Types of Stored Procedures 186](#_Toc510810091)

[Synonyms 188](#_Toc510810092)

[System Catalog Views 189](#_Toc510810093)

[System Compatibility Views 189](#_Toc510810094)

[System Dynamic Management Views 189](#_Toc510810095)

[System Functions for Transact-SQL 190](#_Toc510810096)

[System Information Schema Views 205](#_Toc510810097)

[System Stored Procedures 205](#_Toc510810098)

[System Tables 205](#_Toc510810099)

[System views 205](#_Toc510810100)

[Tables 205](#_Toc510810101)

[Track changes 205](#_Toc510810102)

[Table Variable 206](#_Toc510810103)

[Overview: 206](#_Toc510810104)

[Benefits: 206](#_Toc510810105)

[Limitations and restrictions 206](#_Toc510810106)

[Triggers 208](#_Toc510810107)

[Logon Triggers 208](#_Toc510810108)

[DDL Triggers 208](#_Toc510810109)

[DDL Trigger Scope 209](#_Toc510810110)

[Use the EVENTDATA Function 209](#_Toc510810111)

[DML Triggers 210](#_Toc510810112)

[DML Trigger Benefits 210](#_Toc510810113)

[Types of DML Triggers 211](#_Toc510810114)

[Create Nested Triggers 211](#_Toc510810115)

[Specify First and Last Triggers 213](#_Toc510810116)

[Use the inserted and deleted Tables 213](#_Toc510810117)

[Delete or Disable DML Triggers 214](#_Toc510810118)

[User-defined functions 215](#_Toc510810119)

[User-defined functions 215](#_Toc510810120)

[Types of functions 215](#_Toc510810121)

[Deterministic and Nondeterministic Functions 216](#_Toc510810122)

[Create User-defined Functions 216](#_Toc510810123)

[Limitations and restrictions 216](#_Toc510810124)

[Scalar Functions 217](#_Toc510810125)

[Table-Valued Functions 217](#_Toc510810126)

[SELECT \* FROM <Function\_Name> (602); 217](#_Toc510810127)

[Create CLR Functions 218](#_Toc510810128)

[Create User-defined Aggregates 218](#_Toc510810129)

[Views 219](#_Toc510810130)

[Types of Views 219](#_Toc510810131)

[Updatable Views 220](#_Toc510810132)

[XML 222](#_Toc510810133)

[XML Data 222](#_Toc510810134)

[XML Data Type and Columns 222](#_Toc510810135)

[Relational or XML Data Model 222](#_Toc510810136)

[Reasons for Storing XML Data in SQL Server 222](#_Toc510810137)

[XML Storage Options 223](#_Toc510810138)

[Granularity of XML Data 224](#_Toc510810139)

[Limitations of the xml Data Type 224](#_Toc510810140)

[sp\_refreshview 312](#_Toc510810141)

[Quick Check 269](#_Toc510810142)

[Quick Check 269](#_Toc510810143)

[Quick Check 270](#_Toc510810144)

[Quick Check 271](#_Toc510810145)

[Quick Check 271](#_Toc510810146)

[Quick Check 272](#_Toc510810147)

[Overview of locking 283](#_Toc510810148)

[**Types of Concurrency Problems** **Error! Bookmark not defined.**](#_Toc510810149)

# Functions

## System

# XACT\_STATE:

# Is a scalar function that reports the user transaction state of a current running request. XACT\_STATE indicates whether the request has an active user transaction, and whether the transaction is capable of being committed.

Syntax: XACT\_STATE()

XACT\_STATE returns the following values.

| Return value | Meaning |
| --- | --- |
| 1 | The current request has an active user transaction. The request can perform any actions, including writing data and committing the transaction. |
| 0 | There is no active user transaction for the current request. |
| -1 | The current request has an active user transaction, but an error has occurred that has caused the transaction to be classified as an uncommittable transaction. The request cannot commit the transaction or roll back to a savepoint; it can only request a full rollback of the transaction. The request cannot perform any write operations until it rolls back the transaction. The request can only perform read operations until it rolls back the transaction. After the transaction has been rolled back, the request can perform both read and write operations and can begin a new transaction. |

Both the XACT\_STATE and @@TRANCOUNT functions can be used to detect whether the current request has an active user transaction. @@TRANCOUNT cannot be used to determine whether that transaction has been classified as an uncommittable transaction. XACT\_STATE cannot be used to determine whether there are nested transactions.

-- SET XACT\_ABORT ON will render the transaction uncommittable

-- when the constraint violation occurs.

SET XACT\_ABORT ON;

BEGIN TRY

BEGIN TRANSACTION;

-- A FOREIGN KEY constraint exists on this table. This

-- statement will generate a constraint violation error.

DELETE FROM Production.Product WHERE ProductID = 980;

-- If the delete operation succeeds, commit the transaction. The CATCH

-- block will not execute.

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

-- Test XACT\_STATE for 0, 1, or -1.

-- If 1, the transaction is committable.

-- If -1, the transaction is uncommittable and should be rolled back.

-- XACT\_STATE = 0 means there is no transaction and

-- a commit or rollback operation would generate an error.

-- Test whether the transaction is uncommittable.

IF (XACT\_STATE()) = -1

BEGIN

PRINT 'The transaction is in an uncommittable state. Rolling back transaction.'

ROLLBACK TRANSACTION;

END;

-- Test whether the transaction is active and valid.

IF (XACT\_STATE()) = 1

BEGIN

PRINT 'The transaction is committable.Committing transaction.'

COMMIT TRANSACTION;

END;

END CATCH;

GO

# ALTER TABLE computed\_column\_definition

column\_name AS computed\_column\_expression

[ PERSISTED [ NOT NULL ] ]

[

[ CONSTRAINT constraint\_name ]

{ PRIMARY KEY | UNIQUE }

[ CLUSTERED | NONCLUSTERED ]

[ WITH FILLFACTOR = fillfactor ]

[ WITH ( <index\_option> [, ...n ] ) ]

[ ON { partition\_scheme\_name ( partition\_column\_name ) | filegroup

| "default" } ]

| [ FOREIGN KEY ]

REFERENCES ref\_table [ ( ref\_column ) ]

[ ON DELETE { NO ACTION | CASCADE } ]

[ ON UPDATE { NO ACTION } ]

[ NOT FOR REPLICATION ]

| CHECK [ NOT FOR REPLICATION ] ( logical\_expression )

]

PERSISTED  
Specifies that the Database Engine will physically store the computed values in the table, and update the values when any other columns on which the computed column depends are updated. Marking a computed column as PERSISTED allows an index to be created on a computed column that is deterministic, but not precise.

**ALTER TABLE Inventory ADD TotalItems AS ItemsInStore + ItemsInWarehouse PERSISTED**

CREATE TABLE Sales.OrderDetails (

ListPrice money not null,

Quantity int not null,

LineItemTotal as (ListPrice \* Quantity) PERSISTED)

Need to create an audit record only when either the MobileNumber or HomeNumber column

is updated. Which Transact-SQL query should you use?

CREATE TRIGGER TrgPhoneNumberChange

ON Customers FOR UPDATE

AS

IF UPDATE (HomeNumber) OR UPDATE (MobileNumber)

- - Create Audit Records

You develop a Microsoft SQL Server 2012 database that has two tables named SavingAccounts and

LoanAccounts. Both tables have a column named AccountNumber of the nvarchar data type. You

use a third table named Transactions that has columns named TransactionId AccountNumber, Amount, and TransactionDate. You need to ensure that when multiple records are inserted in the

Transactions table, only the records that have a valid AccountNumber in the SavingAccounts or

LoanAccounts are inserted. Which Transact-SQL statement should you use?

CREATE TRIGGER TrgValidateAccountNumber

ON Transactions

INSTEAD OF INSERT

AS

BEGIN

INSERT INTO Transactions

SELECT TransactionID,AccountNumber,Amount,TransactionDate FROM inserted

WHERE AccountNumber IN

(SELECT AccountNumber FROM LoanAccounts

UNION SELECT AccountNumber FROM SavingAccounts))

END

indexed view.

You need to ensure that users can update only the phone numbers and email addresses (Only two columns out of 6 columns Partitioned view) by using this view. What should you do?

Create an INSTEAD OF UPDATE trigger on the view.

SCHEMABINDING Clause in views

You need to ensure that users are able to modify data by using the view (it has SCHEMABINDING clause and joining of the two tables and sum /Aggregate column). What should you do?

Create an INSTEAD OF UPDATE trigger on the view.

You have an XML schema collection named Sales.InvoiceSchema. You need to declare a variable of

the XML type named XML1. The solution must ensure that XML1 is validated by using

Sales.InvoiceSchema. Which code segment should you use?

Correct Answer:

DECLARE @XML1 XML(Sales.InvoiceSchema)

CROSS APPLY

GROUPING SETS

ROLLUP

CUBE

NOT IN <>ALL (NOT IN , <>ALL both are same )

EXECUTE AS OWNER

EXECUTE AS CALLER

.WRITE in SQL server

CREATE SEQUENCE CustomerSequence AS int

START WITH 1

INCREMENT BY 1

MINVALUE 1

MAXVALUE 100

CYCLE

UPDATE Customers SET PartitionNumber = NEXT VALUE FOR CustomerSequence

DROP SEQUENCE CustomerSequence

UPDATE SYNTAX:

[ WITH <common\_table\_expression> [...n] ]

UPDATE

[ TOP ( expression ) [ PERCENT ] ]

{ { table\_alias | <object> | rowset\_function\_limited

[ WITH ( <Table\_Hint\_Limited> [ ...n ] ) ]

}

| @table\_variable

}

SET

{ column\_name = { expression | DEFAULT | NULL }

| { udt\_column\_name.{ { property\_name = expression

| field\_name = expression }

| method\_name ( argument [ ,...n ] )

}

}

| column\_name { .WRITE ( expression , @Offset , @Length ) }

| @variable = expression

| @variable = column = expression

| column\_name { += | -= | \*= | /= | %= | &= | ^= | |= } expression

| @variable { += | -= | \*= | /= | %= | &= | ^= | |= } expression

| @variable = column { += | -= | \*= | /= | %= | &= | ^= | |= } expression

} [ ,...n ]

[ <OUTPUT Clause> ]

[ FROM{ <table\_source> } [ ,...n ] ]

[ WHERE { <search\_condition>

| { [ CURRENT OF

{ { [ GLOBAL ] cursor\_name }

| cursor\_variable\_name

}

]

}

}

]

[ OPTION ( <query\_hint> [ ,...n ] ) ]

[ ; ]

<object> ::=

{

[ server\_name . database\_name . schema\_name .

| database\_name .[ schema\_name ] .

| schema\_name .

]

table\_or\_view\_name}

**.**WRITE **(**expression**,**@Offset**,**@Length**)**

Specifies that a section of the value of column\_name is to be modified.

expression replaces @Length units starting from @Offset of column\_name. Only columns of **varchar(max)**, **nvarchar(max)**, or **varbinary(max)** can be specified with this clause. column\_name cannot be NULL and cannot be qualified with a table name or table alias.

IF (XACT\_STATE ( ) ) = 1

-- the estimated rows do not match the actual rows (So need to update statistics.)

Use index column as original, don’t use functions, convert, cast etc on index column. If you are using all these the index will not seek .The index will be scan for entire table.

The filter in the where clause it is written as a SARG (search argument). If you include the indexed column as part of an expression or as part of a function argument the query optimizer wont be able to use the index defined over the column, and this most likely lead to an inefficient execution plan.

PIVOT

Enable the optimize for ad hoc workloads option.

There is high contention between readers and writers on several tables used by your transaction.

You need to minimize the use of the tempdb space. You also need to prevent reading queries from

blocking writing queries. Which isolation level should you use?

READ COMMITTED SNAPSHOT

# THROW

Raises an exception and transfers execution to a CATCH block of a TRY…CATCH construct

THROW [ { error\_number | @local\_variable },

{ message | @local\_variable },

{ state | @local\_variable } ]

[ ; ]

The statement before the THROW statement must be followed by the semicolon (;) statement terminator.

If the THROW statement is specified without parameters, it must appear inside a CATCH block. This causes the caught exception to be raised. Any error that occurs in a THROW statement causes the statement batch to be terminated.

## Differences Between RAISERROR and THROW

The following table lists differences between the RAISERROR and THROW statements.

| RAISERROR statement | THROW statement |
| --- | --- |
| If a msg\_id is passed to RAISERROR, the ID must be defined in sys.messages. | The error\_number parameter does not have to be defined in sys.messages. |
| The msg\_str parameter can contain **printf** formatting styles. | The message parameter does not accept **printf** style formatting. |
| The severity parameter specifies the severity of the exception. | There is no severity parameter. The exception severity is always set to 16. |

Alter Table DBo.Book ADD BOOKGUID UNIQUEIDENTIFIER NOT NULL Constraint DF\_BookGUID Default newid() WITH VALUES

structure. Other services include exception handling, hosting for external components like Common Language Runtime, CLR etc.

**Design Guidance**

**Data loading Guidance**

**What's new**

**Query performance**

**Real-time operational analytics**

**Data Warehouse**

**Defragment**

**Index Options**

**AGGREGATE**

**APPLICATION ROLE**

**ASSEMBLY**

**ASYMMETRIC KEY**

**AVAILABILITY GROUP**

**BROKER PRIORITY**

**CERTIFICATE**

**COLUMNSTORE INDEX**

**COLUMN ENCRYPTION KEY**

**COLUMN MASTER KEY**

**CONTRACT**

**CREDENTIAL**

**CRYPTOGRAPHIC PROVIDER**

**DATABASE**

**DATABASE (Azure SQL Database)**

**DATABASE (Azure SQL Data Warehouse)**

**DATABASE (Parallel Data Warehouse)**

**DATABASE AUDIT SPECIFICATION**

**DATABASE ENCRYPTION KEY**

**DATABASE SCOPED CREDENTIAL**

**DEFAULT**

**ENDPOINT**

**EVENT NOTIFICATION**

**EVENT SESSION**

**EXTERNAL DATA SOURCE**

**EXTERNAL LIBRARY**

**EXTERNAL FILE FORMAT**

**EXTERNAL RESOURCE POOL**

**EXTERNAL TABLE**

**EXTERNAL TABLE AS SELECT**

**FULLTEXT STOPLIST**

**FUNCTION**

**FUNCTION (SQL Data Warehouse)**

**INDEX**

**LOGIN**

**MASTER KEY**

**MESSAGE TYPE**

**PARTITION FUNCTION**

**PARTITION SCHEME**

**PROCEDURE**

**QUEUE**

**REMOTE SERVICE BINDING**

**REMOTE TABLE AS SELECT (Parallel Data Warehouse)**

**RESOURCE POOL**

**ROLE**

**ROUTE**

**RULE**

**SCHEMA**

**SEARCH PROPERTY LIST**

**SECURITY POLICY**

**SELECTIVE XML INDEX**

**SEQUENCE**

**SERVER AUDIT**

**SERVER AUDIT SPECIFICATION**

**SERVER ROLE**

**SERVICE**

**SPATIAL INDEX**

**STATISTICS**

**SYMMETRIC KEY**

**SYNONYM**

**TABLE TABLE (Azure SQL Data Warehouse)**

**TABLE (SQL Graph)**

**TABLE AS SELECT (Azure SQL Data Warehouse)**

**TABLE IDENTITY (Property)**

**TRIGGER**

**TYPE**

**USER**

**WORKLOAD GROUP**

**XML INDEX**

**XML INDEX (Selective XML Indexes)**

**XML SCHEMA COLLECTION**

**Overview**

Computed Column

Persisted Computed column

A CLR User-Defined Aggregate

CLR User defined type

Full-text search

CONTAINS PREDICATE

FREE TEXT PREDICATE

CONTAINSTABLE FUNCTION

FREETEXTTABLE FUNCTION

data model that implements Table-per-Hierarchy inheritance

data model that implements Table-per-Type inheritance

data model that includes a complex type

data model that implements a single entity with multiple associations

BISM Model

Table level check constraint

Column level check constraint

Output class

Dynamic Cursor

Partitioned View

Textbox sortExpression property in ssrs

Data tap in ssis

ON Error Event handler in ssis

Run the package by using the dtexecui.exe utility and the SQL Log provider.

SSISDB.[catalog].[executions] view

SSISDB.[catalog].[event\_messages] view

GenerateAndPersistNewIndex Fuzzy Lookup transformation option

TransactionOption property of the package

EventHandlerClass property

IsolationLevel transaction property of a Data Flow task

The catalog.create\_environment\_reference stored procedure.

SorABCeyPosition property for a data flow.

Key Granularity attributes

A CHECK constraint is used to restrict the data that is

allowed for a column to specific values. A CHECK constraint consists of a Boolean expression that evaluates to

either TRUE or FALSE. If the expression evaluates to TRUE, the value is allowed for the column, and if the

expression evaluates to FALSE, the value is not allowed for the column. CHECK constraints can be defined at

the table level or column level, but only CHECK constraints defined at the table level can use columns other

than the constrained column in the constraint expression.

SELECT statement that includes the OUTPUT clause. When performing DML

operations, you can use the OUTPUT clause to obtain and display information about the rows affected by the DML operation. The OUTPUT clause can display this information to the user, insert the

data into another permanent or temporary table or table variable using an INTO clause, or pass the data to a nested DML statement for processing.

Cursors can be used to operate on underlying data on a row-by-row basis.

A dynamic cursor allows scrolling

forward and backward, and all data changes to the underlying table are visible. Cursors are sometimes

unavoidable, but should be replaced with

set-based operations or other methods when possible because they can often degrade performance.

Partitioned views are used when you have similar data stored in

multiple tables and want to create a view to

allow access to all of the data as if it were stored in a single table. Partitioned views are implemented by

creating a view that queries several tables

and combines the results using the UNION ALL operator. A partitioned view can improve performance and

increase availability

Why are you changing, What is the most memorable situation you had in your job, Why should we take you, what is your approach to the new job ?

By using update statistics we can eliminate difference between actual number of rows and estimate number of rows.

Plan Caching and recompiling SQL server White Paper.

retain Same Connection is true

Delay Validation True

Data Compression

----> Row Compression ---- Reduces fixed data type column to variable format.

--->Page Compression

---> Unicode Compression

Unicode compressions apply automatically when you applied Row or page compression.

There are three new catalog views you can use to gather information about columnstore

indexes:

■ sys.column\_store\_index\_stats

■ sys.column\_store\_segments

■ sys.column\_store\_dictionaries

filter non cluster index: A filtered

remember that the majority of DW

queries involve scans over large amounts of data. As a general best practice, you should use

as few nonclustered indexes in your data warehouse as possible.

In a DW, you should not use many nonclustered indexes.

■ Use small, integer surrogate columns for clustered primary keys.

■ Use indexed views.

■ Use columnstore indexes and exploit batch processing.

column store index

batch Processing

Hash joins can use batch processing.

Merge joins do not use batch processing.

Scan operators can benefit from batch processing.

Nested loops joins do not use batch processing.

Filter operators use batch processing as well.

Indexed views are especially useful for speeding up queries that aggregate data.

Indexed views can also speed up queries that perform multiple joins.

Phantom Read: when one transaction executes a query twice in side transaction and it gets a different number of rows in the result set each time.

Non Repeatable Read: Non Repeatable read happens when one transaction reads the same data twice and another transaction updates that data in between first and second read of transaction one. Here we are not repeating same read twice .

Locks are:

Shared

Update

Exclusive

Schema (modification and stability)

Bulk Update Intent (shared, update, exclusive)

Key Range (shared, insert, exclusive)

# Memory Management Architecture Guide

Tables in a Star or

Snowflake schema are divided into dimension tables (commonly known as *dimensions*) and

fact tables.

IMPPROV\_IOWAIT

Error Redirecting.

You cannot change the value of a parameter while a SSIS package is running.

**Synonym**

**quick check**

■ When are property expressions evaluated as a package is running?

**quick check answer**

■ Unlike parameters that are read at the start of package execution, property

expressions are updated when the property is accessed by the package during

package execution. a property expression can change the value of a property in

the middle of package execution, so that the new value is read when the property

is needed by the package.

Locking in the Database Engine

Customizing Locking and Row Versioning

Lock Modes

Lock Compatibility

Row Versioning-based Isolation Levels in the Database Engine

Controlling Transactions (Database Engine)

How to monitor blocking in SQL Server

**Notes:**

1. All isolation levels except for read uncommitted protect against dirty reads.
2. The repeatable read, serializable, and snapshot isolation levels protect a transaction from non-repeatable reads.
3. Only the serializable and snapshot isolation levels protect a transaction from phantom reads.
4. You can use views, stored procedures, and functions to retrieve data. Choose the proper type of object according to a particular situation’s requirements and needs.
5. Use a standard view when you want to package a T-SQL query as a unit for security, deployment, and reusability.
6. Stored procedures are suitable for scenarios that require the input coming from external values and/or when the expected result might not be a result set but a scalar value.
7. User-defined functions (UDFs) enable you to reuse their results in more flexible ways than stored procedures (for instance, in the FROM clause of a SELECT statement, in the Column list section of a SELECT statement, and in a WHERE or HAVING clause of a SELECT statement).
8. UDFs cannot be used to perform actions that modify the database state, and they can only declare input parameters.

* COLUMNSTORE  
  COLUMNSTORE is the default and specifies to compress with the most performant columnstore compression. This is the typical choice.
* COLUMNSTORE\_ARCHIVE  
  COLUMNSTORE\_ARCHIVE further compresses the table or partition to a smaller size. Use this option for situations such as archival that require a smaller storage size and can afford more time for storage and retrieval.

# sp\_refreshview

**Get comma separated values as ta table result set**

**how to remove first and last character from string in sql server**

select @Definition as string, left (right (@Definition, len (@Definition)-1), len (@Definition)-2) as AfterRemovalFirstAndLastCharacter;

Set the Merge agent on the problem subscribers to use the slow link agent profile.

You administer a database that includes a table named Customers that contains more than 750 rows. You

create a new column named PartitionNumber of the int type in the table. You need to assign a PartitionNumber

for each record in the Customers table. You also need to ensure that the PartitionNumber satisfies the following

conditions:

Always starts with 1.

Starts again from 1 after it reaches 100.

Which Transact-SQL statement should you use?

You create a stored procedure named dbo.ModifyData that can modify rows. You need to ensure that when the transaction fails, dbo.ModifyData meets the following requirements:

Does not return an error

Closes all opened transactions

Which Transact-SQL statement should you use?

BEGIN TRANSACTION

BEGIN TRY

EXEC dbo.ModifyData

COMMIT TRANSACTION

END TRY

BEGIN CATCH

IF @@ERROR != 0

ROLLBACK TRANSACTION;

END CATCH

New rows are inserted into the tables in the SalesDB database and updates to existing rows occur on a

high frequency. The inserts and updates often blocked by queries retrieving and reading data

You should make use of the SERIALIZABLE ISOLATION LEVEL.

ABC.com users report that ABCApp3 is functioning sluggishly. You discover that concurrent updates are

causing blockages on the SalesDB database.

SNAPSHOT ISOLATION LEVEL.

**Difference between Read Commit and Read commit snapshot and snapshot Isolation**

# SQL Server Transaction Log Architecture and Management Guide

# Ghost cleanup process guide

## Ghost records

Records that are deleted from a leaf level of an index page aren't physically removed from the page - instead, the record is marked as 'to be deleted', or ghosted. This means that the row stays on the page but a bit is changed in the row header to indicate that the row is really a ghost. This is to optimize performance during a delete operation. Ghosts are necessary for row-level locking, but are also necessary for snapshot isolation where we need to maintain the older versions of rows.

## Ghost record cleanup task

Records that are marked for deletion, or ghosted, are cleaned up by the background ghost cleanup process. This background process runs sometime after the delete transaction is committed, and physically removes ghosted records from pages. The ghost cleanup process runs automatically on an interval (every 5 seconds for SQL Server 2012+, every 10 seconds for SQL Server 2008/2008R2) and checks to see if any pages have been marked with ghost records. If it finds any, then it goes and deletes the records that are marked for deletion, or ghosted, touching at most 10 pages with each execution.

When a record is ghosted, the database is marked as having ghosted entries, and the ghost cleanup process will only scan those databases. The ghost cleanup process will also mark the database as 'having no ghosted records' once all ghosted records have been deleted, and it will skip this database the next time it runs. The process will also skip any databases it is unable to take a shared lock on, and will try again the next time it runs.

The below query can identify how many ghosted records exist in a single database.

SQLCopy

SELECT sum(ghost\_record\_count) total\_ghost\_records, db\_name(database\_id)

FROM sys.dm\_db\_index\_physical\_stats (NULL, NULL, NULL, NULL, NULL)

group by database\_id

order by total\_ghost\_records desc

## Disable the ghost cleanup

On high-load systems with many deletes, the ghost cleanup process can cause a performance issue from keeping pages in the buffer pool and generating IO. As such, it is possible to disable this process with the use of trace flag 661. More information about this can be found in [Tuning options for SQL Server when running high performance workloads](https://support.microsoft.com/help/920093/tuning-options-for-sql-server-when-running-in-high-performance-workloa). However, there are performance implications from disabling the process.

Disabling the ghost cleanup process can cause your database to grow unnecessarily large and can lead to performance issues. Since the ghost cleanup process removes records that are marked as ghosts, disabling the process will leave these records on the page, preventing SQL Server from reusing this space. This forces SQL Server to add data to new pages instead, leading to bloated database files, and can also cause [page splits](https://docs.microsoft.com/en-us/sql/relational-databases/indexes/specify-fill-factor-for-an-index?view=sql-server-2017). Page splits lead to performance issues when creating execution plans, and when doing scan operations.

Once the ghost cleanup process is disabled, some action needs to be taken to remove the ghosted records. One option is to execute an index rebuild, which will move data around on pages. Another option is to manually run [sp\_clean\_db\_free\_space](https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-clean-db-free-space-transact-sql?view=sql-server-2017) (to clean all database data files) or [sp\_clean\_db\_file\_free\_space](https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-clean-db-file-free-space-transact-sql?view=sql-server-2017) (to clean a single database datafile), which will delete ghosted records.

Warning

Disabling the ghost cleanup process is not generally recommended. Doing so should be tested thoroughly in a controlled environment before being implemented permanently in a production environment.

**Internal objects** that are created by the database engine. These include:

* Work tables to store intermediate results for spools, cursors, sorts, and temporary large object (LOB) storage.
* Work files for hash join or hash aggregate operations.
* Intermediate sort results for operations such as creating or rebuilding indexes (if SORT\_IN\_TEMPDB is specified), or certain GROUP BY, ORDER BY, or UNION queries.

## Restrictions

The following operations cannot be performed on the **tempdb** database:

* Adding filegroups.
* Backing up or restoring the database.
* Changing collation. The default collation is the server collation.
* Changing the database owner. **tempdb** is owned by **sa**.
* Creating a database snapshot.
* Dropping the database.
* Dropping the **guest** user from the database.
* Enabling change data capture.
* Participating in database mirroring.
* Removing the primary filegroup, primary data file, or log file.
* Renaming the database or primary filegroup.
* Running DBCC CHECKALLOC.
* Running DBCC CHECKCATALOG.
* Setting the database to OFFLINE.
* Setting the database or primary filegroup to READ\_ONLY.