# SET OPERATORS

The Transact-SQL programming language provides several SET statements that change the current session handling of specific information.

## Date Time Statements

#### SET DATEFIRST

Sets the first day of the week to a number from 1 through 7.

SET DATEFIRST { number | @number\_var }

To see the current setting of SET DATEFIRST, use the @@DATEFIRST function.

#### SET DATEFORMAT

Sets the order of the month, day, and year date parts for interpreting **date**, **smalldatetime**, **datetime**, **datetime2** and **datetimeoffset** character strings.

SET DATEFORMAT { format | @format\_var }

## Locking Statements

#### DEADLOCK\_PRIORITY

Specifies the relative importance that the current session continue processing if it is deadlocked with another session.

SET DEADLOCK\_PRIORITY { LOW | NORMAL | HIGH | <numeric-priority> | @deadlock\_var | @deadlock\_intvar }

<numeric-priority> ::= { -10 | -9 | -8 | … | 0 | … | 8 | 9 | 10 }

#### SET LOCK\_TIMEOUT

Specifies the number of milliseconds a statement waits for a lock to be released.

SET LOCK\_TIMEOUT timeout\_period

*timeout\_period*

Is the number of milliseconds that will pass before Microsoft SQL Server returns a locking error. A value of -1 (default) indicates no time-out period (that is, wait forever).

When a wait for a lock exceeds the time-out value, an error is returned. A value of 0 means to not wait at all and return a message as soon as a lock is encountered.

## Miscellaneous statements

#### SET CONCAT\_NULL\_YIELDS\_NULL

Controls whether concatenation results are treated as null or empty string values.

SET CONCAT\_NULL\_YIELDS\_NULL { ON | OFF }

When SET CONCAT\_NULL\_YIELDS\_NULL is ON, concatenating a null value with a string yields a NULL result. For example, SELECT 'abc' + NULL yields NULL . When SET CONCAT\_NULL\_YIELDS\_NULL is OFF, concatenating a null value with a string yields the string itself (the null value is treated as an empty string). For example,

SELECT 'abc' + NULL yields abc .

#### SET CONTEXT\_INFO

Associates up to 128 bytes of binary information with the current session or connection.

SET CONTEXT\_INFO { binary\_str | @binary\_var }

#### SET CURSOR\_CLOSE\_ON\_COMMIT

Controls the behavior of the Transact-SQL COMMIT TRANSACTION statement. The default value for this setting is OFF. This means that the server will not close cursors when you commit a transaction.

SET CURSOR\_CLOSE\_ON\_COMMIT { ON | OFF }

When SET CURSOR\_CLOSE\_ON\_COMMIT is ON, this setting closes any open cursors on commit or rollback in compliance with ISO. When SET CURSOR\_CLOSE\_ON\_COMMIT is OFF, the cursor is not closed when a transaction is committed.

**NOTE:** SET CURSOR\_CLOSE\_ON\_COMMIT to ON will not close open cursors on rollback when the rollback is applied to a savepoint\_name from a SAVE TRANSACTION statement.

#### SET FIPS\_FLAGGER

Specifies checking for compliance with the FIPS 127-2 standard. This is based on the ISO standard.

SET FIPS\_FLAGGER ( 'level' | OFF )

Level must me one of the following value

|  |  |
| --- | --- |
| **VALUE** | **DESCRIPTION** |
| ENTRY | Standards checking for ISO entry-level compliance. |
| FULL | Standards checking for ISO full compliance. |
| INTERMEDIATE | Standards checking for ISO intermediate-level compliance. |
| OFF | No standards checking. |

#### SET FMTONLY

Returns only metadata to the client. Can be used to test the format of the response without actually running the query.

SET FMTONLY { ON | OFF }

No rows are processed or sent to the client because of the request when SET FMTONLY is turned ON.

#### SET IDENTITY\_INSERT

Allows explicit values to be inserted into the identity column of a table.

SET IDENTITY\_INSERT [ database\_name . [ schema\_name ] . ] table { ON | OFF }

#### SET LANGUAGE

Specifies the language environment for the session. The session language determines the datetime formats and system messages.

SET LANGUAGE { [ N ] 'language' | @language\_var }

[**N**]**'***language***'** | **@***language\_var*

Is the name of the language as stored in sys.syslanguages.

#### SET OFFSETS

Returns the offset (position relative to the start of a statement) of specified keywords in Transact-SQL statements to DB-Library applications.

SET OFFSETS keyword\_list { ON | OFF }

*keyword\_list*

Is a comma-separated list of Transact-SQL constructs including SELECT, FROM, ORDER, TABLE, PROCEDURE, STATEMENT, PARAM, and EXECUTE.

#### SET QUOTED\_IDENTIFIER

Causes SQL Server to follow the ISO rules regarding quotation mark delimiting identifiers and literal strings. Identifiers delimited by double quotation marks can be either Transact-SQL reserved keywords or can contain characters not generally allowed by the Transact-SQL syntax rules for identifiers.

SET QUOTED\_IDENTIFIER { ON | OFF }

When SET QUOTED\_IDENTIFIER is ON, identifiers can be delimited by double quotation marks, and literals must be delimited by single quotation marks. When SET QUOTED\_IDENTIFIER is OFF, identifiers cannot be quoted and must follow all Transact-SQL rules for identifiers.

Literals can be delimited by either single or double quotation marks.

SET QUOTED\_IDENTIFIER OFF

GO

-- An attempt to create a table with a reserved keyword as a name

-- should fail.

CREATE TABLE "select" ("identity" INT IDENTITY NOT NULL, "order" INT NOT NULL);

GO

SET QUOTED\_IDENTIFIER ON;

GO

-- Will succeed.

CREATE TABLE "select" ("identity" INT IDENTITY NOT NULL, "order" INT NOT NULL);

GO

SELECT "identity","order" FROM "select" ORDER BY "order"; -- Succeed

GO

SET QUOTED\_IDENTIFIER OFF;

GO

SET QUOTED\_IDENTIFIER OFF;

GO

CREATE TABLE dbo.Test (ID INT, String VARCHAR(30)) ;

GO

-- Literal strings can be in single or double quotation marks.

INSERT INTO dbo.Test VALUES (1, "'Text in single quotes'");

INSERT INTO dbo.Test VALUES (2, '''Text in single quotes''');

INSERT INTO dbo.Test VALUES (3, 'Text with 2 '''' single quotes');

INSERT INTO dbo.Test VALUES (4, '"Text in double quotes"');

INSERT INTO dbo.Test VALUES (5, """Text in double quotes""");

INSERT INTO dbo.Test VALUES (6, "Text with 2 """" double quotes");

GO

SET QUOTED\_IDENTIFIER ON;

GO

-- Strings inside double quotation marks are now treated

-- as object names, so they cannot be used for literals.

INSERT INTO dbo."Test" VALUES (7, 'Text with a single '' quote');

GO

-- Object identifiers do not have to be in double quotation marks

-- if they are not reserved keywords.

SELECT ID, String FROM dbo.Test;

O/P:

1 'Text in single quotes'

2 'Text in single quotes'

3 Text with 2 '' single quotes

4 "Text in double quotes"

5 "Text in double quotes"

6 Text with 2 "" double quotes

7 Text with a single ' quote

SET QUOTED\_IDENTIFIER OFF;

GO

## Query Execution Statements

#### SET ARITHABORT

Terminates a query when an overflow or divide-by-zero error occurs during query execution.

SET ARITHABORT { ON | OFF }

You should always set ARITHABORT to ON in your logon sessions. Setting ARITHABORT to OFF can negatively impact query optimization leading to performance issues.

Warning: The default ARITHABORT setting for SQL Server Management Studio is ON. Client applications setting ARITHABORT to OFF can receive different query plans making it difficult to troubleshoot poorly performing queries. That is, the same query can execute fast in management studio but slow in the application. When troubleshooting queries with Management Studio always match the client ARITHABORT setting.

* If SET ARITHABORT is ON and SET ANSI WARNINGS is ON, these error conditions cause the query to terminate.
* If SET ARITHABORT is ON and SET ANSI WARNINGS is OFF, these error conditions cause the batch to terminate. If the errors occur in a transaction, the transaction is rolled back. If SET ARITHABORT is OFF and one of these errors occurs, a warning message is displayed, and NULL is assigned to the result of the arithmetic operation.
* If SET ARITHABORT is OFF and SET ANSI WARNINGS is OFF and one of these errors occurs, a warning message is displayed, and NULL is assigned to the result of the arithmetic operation.

#### SET ARITHIGNORE

Controls whether error messages are returned from overflow or divide-by-zero errors during a query.

SET ARITHIGNORE { ON | OFF }

* The SET ARITHIGNORE setting only controls whether an error message is returned. SQL Server returns a NULL in a calculation involving an overflow or divide-by-zero error, regardless of this setting.
* The SET ARITHABORT setting can be used to determine whether the query is terminated. This setting does not affect errors occurring during INSERT, UPDATE, and DELETE statements.
* If either SET ARITHABORT or SET ARITHIGNORE is OFF and SET ANSI\_WARNINGS is ON, SQL Server still returns an error message when encountering divide-by-zero or overflow errors.

#### SET NOCOUNT

Stops the message that shows the count of the number of rows affected by a Transact-SQL statement or stored procedure from being returned as part of the result set.

SET NOCOUNT { ON | OFF }

When SET NOCOUNT is ON, the count is not returned. When SET NOCOUNT is OFF, the count is returned. The @@ROWCOUNT function is updated even when SET NOCOUNT is ON.

#### SET NOEXEC

Compiles each query but does not execute it.

SET NOEXEC { ON | OFF }

When SET NOEXEC is ON, SQL Server compiles each batch of Transact-SQL statements but does not execute them.

The execution of statements in SQL Server has two phases: compilation and execution. This setting is useful for having SQL Server validate the syntax and object names in Transact-SQL code when executing.

#### SET NUMERIC\_ROUNDABORT

Specifies the level of error reporting generated when rounding in an expression causes a loss of precision.

SET NUMERIC\_ROUNDABORT { ON | OFF }

When SET NUMERIC\_ROUNDABORT is ON, an error is generated after a loss of precision occurs in an expression. When OFF, losses of precision do not generate error messages and the result is rounded to the precision of the column or variable storing the result.

|  |  |  |
| --- | --- | --- |
| **SETTING** | **SET NUMERIC\_ROUNDABORT ON** | **SET NUMERIC\_ROUNDABORT OFF** |
| SET ARITHABORT ON | Error is generated; no result set returned | No errors or warnings; result is rounded. |
| SET ARITHABORT OFF | Warning is returned; expression returns NULL | No errors or warnings; result is rounded. |

SET NUMERIC\_ROUNDABORT ON;

SET ARITHABORT ON;

GO

DECLARE @result DECIMAL(5, 2), @value\_1 DECIMAL(5, 4), @value\_2 DECIMAL(5, 4);

SET @value\_1 = 1.1234; SET @value\_2 = 1.1234 ;

SELECT @result = @value\_1 + @value\_2;

SELECT @result;

-- ERROR : Arithmetic overflow error converting numeric to data type numeric.

GO

-- SET NUMERIC\_ROUNDABORT to ON and SET ARITHABORT to OFF.

SET NUMERIC\_ROUNDABORT ON;

SET ARITHABORT OFF;

GO

DECLARE @result DECIMAL(5, 2), @value\_1 DECIMAL(5, 4), @value\_2 DECIMAL(5, 4);

SET @value\_1 = 1.1234; SET @value\_2 = 1.1234 ;

SELECT @result = @value\_1 + @value\_2;

SELECT @result;

--Error : Arithmetic overflow error converting numeric to data type numeric.

GO

-- SET NUMERIC\_ROUNDABORT to OFF and SET ARITHABORT to ON.

SET NUMERIC\_ROUNDABORT OFF;

SET ARITHABORT ON;

GO

DECLARE @result DECIMAL(5, 2), @value\_1 DECIMAL(5, 4), @value\_2 DECIMAL(5, 4);

SET @value\_1 = 1.1234; SET @value\_2 = 1.1234 ;

SELECT @result = @value\_1 + @value\_2;

SELECT @result;

--O/P: 2.25

GO

-- SET NUMERIC\_ROUNDABORT to OFF and SET ARITHABORT to OFF.

SET NUMERIC\_ROUNDABORT OFF;

SET ARITHABORT OFF;

DECLARE @result DECIMAL(5, 2), @value\_1 DECIMAL(5, 4), @value\_2 DECIMAL(5, 4);

SET @value\_1 = 1.1234; SET @value\_2 = 1.1234;

SELECT @result = @value\_1 + @value\_2;

SELECT @result;

-- O/p: 2.25

#### SET PARSEONLY

Examines the syntax of each Transact-SQL statement and returns any error messages without compiling or executing the statement.

SET PARSEONLY { ON | OFF }

When SET PARSEONLY is ON, SQL Server only parses the statement. When SET PARSEONLY is OFF, SQL Server compiles and executes the statement.

#### SET QUERY\_GOVERNOR\_COST\_LIMIT

Overrides the currently configured **query governor cost limit** value for the current connection.

SET QUERY\_GOVERNOR\_COST\_LIMIT value

*value*

Is a numeric or integer value specifying the longest time in which a query can run. Values are rounded down to the nearest integer. Negative values are rounded up to 0.

#### SET ROWCOUNT

Causes SQL Server to stop processing the query after the specified number of rows are returned.

SET ROWCOUNT { number | @number\_var }

*number* | @*number\_var*

Is the number, an integer, of rows to be processed before stopping the specific query.

Setting the SET ROWCOUNT option causes most Transact-SQL statements to stop processing when they have been affected by the specified number of rows.

#### SET TEXTSIZE

Specifies the size of **varchar(max)**, **nvarchar(max)**, **varbinary(max)**, **text**, **ntext**, and **image** data returned by a SELECT statement.

SET TEXTSIZE { number }

*number*

Is the length of **varchar(max)**, **nvarchar(max)**, **varbinary(max)**, **text**, **ntext**, or **image** data, in bytes. *number* is an integer with a maximum value of 2147483647 (2 GB). A value of -1 indicates unlimited size. A value of 0 resets the size to the default value of 4 KB.

## ISO Settings statements

#### SET ANSI\_DEFAULTS

Controls a group of SQL Server settings that collectively specify some ISO standard behavior.

SET ANSI\_DEFAULTS { ON | OFF }

#### SET ANSI\_NULL\_DFLT\_OFF

Alters the behavior of the session to override default nullability of new columns when the ANSI null default option for the database is **true**.

SET ANSI\_NULL\_DFLT\_OFF { ON | OFF }

* This setting only affects the nullability of new columns when the nullability of the column is not specified in the CREATE TABLE and ALTER TABLE statements. By default, when SET ANSI\_NULL\_DFLT\_OFF is ON, new columns that are created by using the ALTER TABLE and CREATE TABLE statements are NOT NULL if the nullability status of the column is not explicitly specified. SET ANSI\_NULL\_DFLT\_OFF does not affect columns that are created by using an explicit NULL or NOT NULL.
* Both SET ANSI\_NULL\_DFLT\_OFF and SET ANSI\_NULL\_DFLT\_ON cannot be set ON at the same time. If one option is set ON, the other option is set OFF. Therefore, either ANSI\_NULL\_DFLT\_OFF or SET ANSI\_NULL\_DFLT\_ON can be set ON, or both can be set OFF.

SET ANSI\_NULL\_DFLT\_OFF ON;

CREATE TABLE t2 (a TINYINT);

-- NULL INSERT Fails beacuse table column created as NOT NULL by default

INSERT INTO t2 (a) VALUES (NULL);

SET ANSI\_NULL\_DFLT\_OFF OFF;

CREATE TABLE t3 (a TINYINT) ;

-- NULL INSERT should succeed. because by default column created as NULL

INSERT INTO t3 (a) VALUES (NULL);

Note use above two SQL statements in two different sessions not in same session.

ALTER DATABASE AdventureWorks2012 SET ANSI\_NULL\_DEFAULT OFF;

#### SET ANSI\_NULL\_DFLT\_ON

Modifies the behavior of the session to override default nullability of new columns when the **ANSI null default** option for the database is **false**.

SET ANSI\_NULL\_DFLT\_ON {ON | OFF}

* This setting only affects the nullability of new columns when the nullability of the column is not specified in the CREATE TABLE and ALTER TABLE statements. When SET ANSI\_NULL\_DFLT\_ON is ON, new columns created by using the ALTER TABLE and CREATE TABLE statements allow null values if the nullability status of the column is not explicitly specified. SET ANSI\_NULL\_DFLT\_ON does not affect columns created with an explicit NULL or NOT NULL.
* Both SET ANSI\_NULL\_DFLT\_OFF and SET ANSI\_NULL\_DFLT\_ON cannot be set ON at the same time. If one option is set ON, the other option is set OFF. Therefore, either ANSI\_NULL\_DFLT\_OFF or ANSI\_NULL\_DFLT\_ON can be set ON, or both can be set OFF. If either option is ON, that setting (SET ANSI\_NULL\_DFLT\_OFF or SET ANSI\_NULL\_DFLT\_ON) takes effect.

ALTER DATABASE AdventureWorks2012 SET ANSI\_NULL\_DEFAULT OFF;

#### SET ANSI\_NULLS

Specifies ISO compliant behavior of the Equals (=) and Not Equal To (<>) comparison operators when they are used with null values.

SET ANSI\_NULLS { ON | OFF }

* When SET ANSI\_NULLS is ON, a SELECT statement that uses WHERE *column\_name* = **NULL** returns zero rows even if there are null values in *column\_name*. A SELECT statement that uses WHERE *column\_name* <> **NULL** returns zero rows even if there are nonnull values in *column\_name*.
* When SET ANSI\_NULLS is OFF, the Equals (=) and Not Equal To (<>) comparison operators do not follow the ISO standard. A SELECT statement that uses WHERE *column\_name* = **NULL** returns the rows that have null values in *column\_name*. A SELECT statement that uses WHERE *column\_name* <> **NULL** returns the rows that have nonnull values in the column. Also, a SELECT statement that uses WHERE *column\_name* <> *XYZ\_value* returns all rows that are not *XYZ\_value* and that are not NULL(NULL values are not returned).

#### SET ANSI\_PADDING

Controls the way the column stores values shorter than the defined size of the column, and the way the column stores values that have trailing blanks in **char**, **varchar**, **binary**, and **varbinary** data.

SET ANSI\_PADDING { ON | OFF }

Columns defined with **char**, **varchar**, **binary**, and **varbinary** data types have a defined size.

This setting affects only the definition of new columns. After the column is created, SQL Server stores the values based on the setting when the column was created. Existing columns are not affected by a later change to this setting.

**SET ANSI\_PADDING ON :** Pad original value (with trailing blanks for **char** columns and with trailing zeros for **binary** columns) to the length of the column.

#### SET ANSI\_WARNINGS

Specifies ISO standard behavior for several error conditions.

SET ANSI\_WARNINGS { ON | OFF }

SET ANSI\_WARNINGS affects the following conditions:

* When set to ON, if null values appear in aggregate functions, such as SUM, AVG, MAX, MIN, STDEV,STDEVP, VAR, VARP, or COUNT, a warning message is generated. When set to OFF, no warning is issued.
* When set to ON, the divide-by-zero and arithmetic overflow errors cause the statement to be rolled back and an error message is generated. When set to OFF, the divide-by-zero and arithmetic overflow errors cause null values to be returned.
* The behavior in which a divide-by-zero or arithmetic overflow error causes null values to be returned occurs if an INSERT or UPDATE is tried on a **character**, Unicode, or **binary** column in which the length of a new value exceeds the maximum size of the column. If SET ANSI\_WARNINGS is ON, the INSERT or UPDATE is canceled as specified by the ISO standard. Trailing blanks are ignored for character columns and trailing nulls are ignored for binary columns. When OFF, data is truncated to the size of the column and the statement succeeds.

**NOTE:** When truncation occurs in any conversion to or from **binary** or **varbinary** data, no warning or error is issued, regardless of SET options.

**NOTE:** ANSI\_WARNINGS is not honored when passing parameters in a stored procedure, user-defined function, or when declaring and setting variables in a batch statement. For example, if a variable is defined as **char(3)**, and then set to a value larger than three characters, the data is truncated to the defined size and the INSERT or UPDATE statement succeeds.

## Statistics statements

#### SET FORCEPLAN

When FORCEPLAN is set to ON, the SQL Server query optimizer processes a join in the same order as the tables appear in the FROM clause of a query. In addition, setting FORCEPLAN to ON forces the use of a nested loop join unless other types of joins are required to construct a plan for the query, or they are requested with join hints or query hints.

SET FORCEPLAN { ON | OFF }

SET FORCEPLAN essentially overrides the logic used by the query optimizer to process a Transact-SQL SELECT statement.

#### SET SHOWPLAN\_ALL

Causes Microsoft SQL Server not to execute Transact-SQL statements. Instead, SQL Server returns detailed information about how the statements are executed and provides estimates of the resource requirements for the statements.

SET SHOWPLAN\_ALL { ON | OFF }

When SET SHOWPLAN\_ALL is ON, SQL Server returns execution information for each statement without executing it, and Transact-SQL statements are not executed.

#### SET SHOWPLAN\_TEXT

Causes Microsoft SQL Server not to execute Transact-SQL statements. Instead, SQL Server returns detailed information about how the statements are executed.

SET SHOWPLAN\_TEXT { ON | OFF }

SET SHOWPLAN\_TEXT is intended to return readable output for Microsoft Win32 command prompt applications such as the osql utility. SET SHOWPLAN\_ALL returns more detailed output intended to be used with programs designed to handle its output.

SET SHOWPLAN\_TEXT and SET SHOWPLAN\_ALL cannot be specified in a stored procedure. They must be the only statements in a batch

#### SET SHOWPLAN\_XML

Causes SQL Server not to execute Transact-SQL statements. Instead, SQL Server returns detailed information about how the statements are going to be executed in the form of a well-defined XML document.

SET SHOWPLAN\_XML { ON | OFF }

SET SHOWPLAN\_XML is intended to return output as nvarchar(max) for applications such as the sqlcmd utility, where the XML output is subsequently used by other tools to display and process the query plan information.

#### SET STATISTICS IO

Causes SQL Server to display information regarding the amount of disk activity generated by Transact-SQL statements.

SET STATISTICS IO { ON | OFF }

|  |  |
| --- | --- |
| **OUTPUT ITEM** | **MEANING** |
| **Table** | Name of the table. |
| **Scan count** | Number of seeks/scans started after reaching the leaf level in any direction to retrieve all the values to construct the final dataset for the output.  Scan count is 0 if the index used is a unique index or clustered index on a primary key and you are seeking for only one value.  Scant count is 1 when you are searching for one value using a non-unique clustered index which is defined on a non-primary key column.  Scan count is N when N is the number of different seek/scan started towards the left or right side at the leaf level after locating a key value using the index key |
| **logical reads** | Number of pages read from the data cache. |
| **physical reads** | Number of pages read from disk. |
| **read-ahead reads** | Number of pages placed into the cache for the query. |
| **lob logical reads** | Number of **text, ntext, image**, or large value type (**varchar(max), varchar(max), varbinary(max))** pages read from the data cache. |
| **lob physical reads** | Number of **text, ntext, image** or large value type pages read from disk. |
| **lob read-ahead reads** | Number of **text, ntext, image** or large value type pages placed into the cache for the query. |

#### SET STATISTICS PROFILE

Displays the profile information for a statement. STATISTICS PROFILE works for ad hoc queries, views, and stored procedures.

SET STATISTICS PROFILE { ON | OFF }

When STATISTICS PROFILE is ON, each executed query returns its regular result set, followed by an additional result set that shows a profile of the query execution.

|  |  |
| --- | --- |
| **COLUMN NAME** | **DESCRIPTION** |
| **Rows** | Actual number of rows produced by each operator |
| **Executes** | Number of times the operator has been executed |

#### STATISTICS TIME

Displays the number of milliseconds required to parse, compile, and execute each statement.

SET STATISTICS TIME { ON | OFF }

#### STATISTICS XML

Causes Microsoft SQL Server to execute Transact-SQL statements and generate detailed information about how the statements were executed in the form of a well-defined XML document.

SET STATISTICS XML { ON | OFF }

SET STATISTICS XML returns output as **nvarchar(max)** for applications, such as the **sqlcmd** utility, where the XML output is subsequently used by other tools to display and process the query plan information. SET STATISTICS XML returns information as a set of XML documents.

## Transactions statements

#### SET IMPLICIT\_TRANSACTIONS

Sets the BEGIN TRANSACTION mode to *implicit*, for the connection.

SET IMPLICIT\_TRANSACTIONS { ON | OFF }

When ON, the system is in *implicit* transaction mode. This means that if @@TRANCOUNT = 0, any of the following Transact-SQL statements begins a new transaction. It is equivalent to an unseen BEGIN TRANSACTION being

executed first:

**ALTER TABLE FETCH REVOKE**

**BEGIN TRANSACTION GRANT SELECT**

**CREATE INSERT TRUNCATE TABLE**

**DELETE OPEN UPDATE DROP**

When OFF, each of the preceding T-SQL statements is bounded by an unseen BEGIN TRANSACTION and an unseen COMMIT TRANSACTION statement. When OFF, we say the transaction mode is *autocommit*.

#### SET REMOTE\_PROC\_TRANSACTIONS

Specifies that when a local transaction is active, executing a remote stored procedure starts a Transact-SQL

distributed transaction managed by Microsoft Distributed Transaction Coordinator (MS DTC).

SET REMOTE\_PROC\_TRANSACTIONS { ON | OFF }

When ON, a Transact-SQL distributed transaction is started when a remote stored procedure is executed from a local transaction. When OFF, calling remote stored procedures from a local transaction does not start a Transact- SQL distributed transaction.

#### SET TRANSACTION ISOLATION LEVEL

Controls the locking and row versioning behavior of Transact-SQL statements issued by a connection to SQL Server.

SET TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED

| REPEATABLE READ | SNAPSHOT | SERIALIZABLE }

#### SET XACT\_ABORT

NOTE : The **THROW** statement honors **SET XACT\_ABORT RAISERROR** does not. New Applications should use **THROW** instead of **RAISERROR**.

**Syntax:** SET XACT\_ABORT { ON | OFF }

When SET XACT\_ABORT is ON, if a Transact-SQL statement raises a run-time error, the entire transaction is terminated and rolled back.

When SET XACT\_ABORT is OFF, in some cases only the Transact-SQL statement that raised the error is rolled back and the transaction continues processing. Depending upon the severity of the error, the entire transaction may be rolled back even when SET XACT\_ABORT is OFF. OFF is the default setting.

Compile errors, such as syntax errors, are not affected by SET XACT\_ABORT.

XACT\_ABORT must be set ON for data modification statements in an implicit or explicit transaction against most OLE DB providers, including SQL Server. The only case where this option is not required is if the provider supports nested transactions.

CREATE TABLE t1 (a INT NOT NULL PRIMARY KEY);

CREATE TABLE t2 (a INT NOT NULL REFERENCES t1(a));

GO

INSERT INTO t1 VALUES (1);

INSERT INTO t1 VALUES (3);

INSERT INTO t1 VALUES (4);

INSERT INTO t1 VALUES (6);

GO

SET XACT\_ABORT OFF;

GO

BEGIN TRANSACTION;

INSERT INTO t2 VALUES (1);

INSERT INTO t2 VALUES (2); -- Foreign key error.

INSERT INTO t2 VALUES (3);

COMMIT TRANSACTION;

GO

SET XACT\_ABORT ON;

GO

BEGIN TRANSACTION;

INSERT INTO t2 VALUES (4);

INSERT INTO t2 VALUES (5); -- Foreign key error.

INSERT INTO t2 VALUES (6);

COMMIT TRANSACTION;

GO

-- SELECT shows only keys 1 and 3 added.

-- Key 2 insert failed and was rolled back, but

-- XACT\_ABORT was OFF and rest of transaction

-- succeeded.

-- Key 5 insert error with XACT\_ABORT ON caused

-- all of the second transaction to roll back.

SELECT \* FROM t2;

O/P:

1

3

GO