

**Permanent**  
select into/bulkcopy must be set

**Temporary**  
requires dboption

**() \*/% +/-**  
(int, smallint, tinyint, float, real, money, smallmoney)

**Eliminates Duplicates**

```
SELECT [ALL | DISTINCT] {operators} {functionname} select_list
[INTO {new_table_name}]
[FROM {table_name | view_name} {(optimizer_hints)}
[[, {table_name2 | view_name2} {(optimizer_hints)}
[... {table_name16 | view_name16} {(optimizer_hints)}]]
[WHERE clause] [LIKE wildcards]
[GROUP BY clause]
[HAVING clause]
[ORDER BY clause]
[COMPUTE clause]
[FOR BROWSE]
```

**Sort Result Set**

**%**  
**\_**  
**[^]**

**Join Operators**

=  
>  
<  
>=  
<=  
|=  
|>  
|<  
|=+  
|>+  
|<+

**Aggregate Functions (Summary)**

SUM(expression)  
AVG(expression)  
COUNT(expression)  
MAX(expression)  
MIN(expression)

```
CREATE PROC[EDURE] procedure_name [number]
[(@parameter data_type) [VARYING] [= default] [OUTPUT]
[...n] table_name WITH (<table_hint_limited> [...n])
[view_name
| rowset_function_limited]
[WITH
{
[RECOMPILE
| ENCRYPTION
| RECOMPILE, ENCRYPTION]
{ [(column_list)]
[VALUES ( { [DEFAULT
| NULL
| expression]
[...n]
| derived_table
| execute_statement]
}
] [FOR REPLICATION]
AS
sql_statement [...n]
```

**DDL: Data Definition Language**  
**DML: Data Manipulation Language**

```
UPDATE
{
table_name WITH (<table_hint_limited> [...n])
| view_name
| rowset_function_limited
}
SET
{column_name = {expression | DEFAULT |
NULL}
| @variable = expression
| @variable = column = expression } [...n]
}
[[FROM {<table_source>} [...n]]
]
[WHERE
<search_condition>]
]
[WHERE CURRENT OF
{ { [GLOBAL] cursor_name }
| cursor_variable_name }
]
]
[OPTION (<query_hint> [...n])]
```

```
DELETE
[FROM]
{
table_name WITH (<table_hint_limited> [...n])
| view_name
| rowset_function_limited
}
[FROM {<table_source>} [...n]]
]
[WHERE
{<search_condition>
| { [CURRENT OF
{ { [GLOBAL] cursor_name }
| cursor_variable_name }
]
}
]
]
[OPTION (<query_hint> [...n])]
```

**Error Handling**

**RAISERROR**  
BEGIN  
RAISERROR ('The job\_id %d must be between %d and %d.',  
16, 1, @@JOB\_ID, @@MIN\_LVL, @@MAX\_LVL)  
ROLLBACK TRANSACTION  
END  
**severity**  
Is the user-defined severity level associated with this message.  
Severity levels from 0 through 18 can be used by any user.  
- level 17: insufficient resources  
- level 18: non fatal internal error  
Severity levels 19 through 25 are used only by members of the sysadmin fixed server role. For severity levels 19 through 25, the WITH LOG option is required. 20-25 are fatal.

**View**

- SQL allows 32 nesting levels
- Cannot include:  
ORDER BY, COMPUTE, COMPUTE BY or INTO
- Cannot associate triggers or DEFAULT.
- WITH CHECK OPTION allows changes to view data only if it is within the bounds of the view criteria

**Joins**

**INNER** ... Specifies all matching pairs of rows are returned. Discards unmatched rows from both tables. This is the default if no join type is specified.

**FULL [OUTER]** ... Specifies that a row from either the left or right table that does not meet the join condition is included in the result set, and output columns that correspond to the other table are set to NULL. This is in addition to all rows usually returned by the INNER JOIN.

**LEFT [OUTER]** ... Specifies that all rows from the left table not meeting the join condition are included in the result set, and output columns from the other table are set to NULL in addition to all rows returned by the inner join.

**RIGHT [OUTER]** ... Specifies all rows from the right table not meeting the join condition are included in the result set, and output columns that correspond to the other table are set to NULL, in addition to all rows returned by the inner join.

**Numeric Functions**

ABS, ACOS, ARB, ATAN, ATN2  
CEILING  
COS, COT  
DEGREES  
EXP  
FLOOR  
LOG, LOG10  
PI  
POWER  
RADIANS  
RAND  
ROUND  
SIGN  
SIN  
SQRT  
TAN

**Character Functions**

ASCII  
CHAR, CHARINDEX  
DIFFERENCE  
LOWER  
LTRIM  
PATINDEX  
REPLICATE  
REVERSE  
RIGHT  
RTRIM  
SOUNDEX  
SPACE  
STR  
STUFF  
SUBSTRING  
UPPER

**Date Functions**

DATETIMEFROMPART (number, date)  
DATETIMEPART (datepart, date1, date2)  
DATETIMEOFFSET (datepart, date)  
DATETIMEPART (datepart, date)  
GETDATE

**Conversion Functions**

CAST  
CONVERT  
COL\_LENGTH  
COL\_NAME  
DATALength  
DELETED  
DE\_NAME  
GETANNULL  
HOST\_ID  
HOST\_NAME  
IDENTIFIER  
IDENTIFIER  
INDEX\_COL  
ISNULL  
NULLIF  
OBJECT\_ID  
OBJECT\_NAME  
STAT\_DATE  
USER\_ID  
USER\_NAME  
USER\_NAME

**ACID**

**Atomicity**  
A transaction must be an atomic unit of work; either all of its data modifications are performed, or none of them is performed.

**Consistency**  
When completed, a transaction must leave all data in a consistent state. In a relational database, all rules must be applied to the transaction's modifications to maintain all data integrity. All internal data structures, such as B-tree indexes or doubly linked lists, must be correct at the end of the transaction.

**Isolation**  
Modifications made by concurrent transactions must be isolated from the modifications made by any other concurrent transactions. A transaction either sees data in the state it was in before another concurrent transaction modified it, or it sees the data after the second transaction has completed, but it does not see an intermediate state. This is referred to as serializability because it results in the ability to reload the starting data and replay a series of transactions to end up with the data in the same state it was in after the original transactions were performed.

**Durability**  
After a transaction has completed, its effects are permanently in place in the system. The modifications persist even in the event of a system failure.

**PAD\_INDEX**  
... Free space on non-leaf pages. Requires FILLFACTOR.

**FILLFACTOR** ...  
% definition of how much to fill leaf level pages. Low = few page splits, but large index.

**Summary Data**

**GROUP BY** ... organise rows into groups and summarise groups.  
**HAVING** ... restrict GROUPING  
**CUBE** ... GROUP BY summary is returned for every possible combination of group and subgroup, displayed as NULL.  
**ROLLUP** ... Summary rows are added into the result set.



**INDEXES** → **COMPOSITE**  
... 1-16 columns

**CLUSTERED** ... Key values same as physical order of cor. rows.  
**NON-CLUSTERED** ... Physical store does not match physical order.

... WITH INDEX(0) >> clustered index, table scan if no CIndex  
... WITH INDEX(1) >> clustered index, fail if no CIndex

**Isolation Levels**

**READ COMMITTED** ... Specifies that shared locks are held while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in nonrepeatable reads or phantom data. This option is the SQL Server default.

**READ UNCOMMITTED** ... Implements dirty read, or isolation level 0 locking, which means that no shared locks are issued and no exclusive locks are honored. When this option is set, it is possible to read uncommitted or dirty data; values in the data can be changed and rows can appear or disappear in the data set before the end of the transaction. This option has the same effect as setting NOLOCK on all tables in all SELECT statements in a transaction. This is the least restrictive of the four isolation levels.

**REPEATABLE READ** ... Locks are placed on all data that is used in a query, preventing other users from updating the data, but new phantom rows can be inserted into the data set by another user and are included in later reads in the current transaction. Because concurrency is lower than the default isolation level, use this option only when necessary.

**SERIALIZABLE** ... Places a range lock on the data set, preventing other users from updating or inserting rows into the data set until the transaction is complete. This is the most restrictive of the four isolation levels. Because concurrency is lower, use this option only when necessary. This option has the same effect as setting HOLDLOCK on all tables in all SELECT statements in a transaction.

**SCHEMA** ... Used when an operation dependent on the schema of a table is executing. The types of schema locks are: schema modification (Sch-M) and schema stability (Sch-S).

**SQL Cursors**

**Static**  
Fixes the result set when the cursor is opened. READ-ONLY! Static cursors detect few or no changes but consume relatively few resources while scrolling, although they store the entire cursor in tempdb.

**Dynamic**  
Opposite of static. Changes are reflected! Dynamic cursors detect all changes but consume more resources while scrolling, although they make the lightest use of tempdb. READ\_ONLY keyword makes the dynamic cursor read-only.

**Key-Set Driven**  
Controlled by a unique set of identifiers. Keyset-driven cursors lie in between, detecting most changes but at less expense than dynamic cursors. Changes made by owner can be seen, others not.

**Forward-only** ... fetch rows from start to end. FORWARD\_ONLY keyword

**Scrolling** ... up and down

**Normalisation**

**1NF** Every row-by-column position in a table must have only one value, no repeating columns.  
**2NF** Store only data that relates to one entity ("thing"), described by the PK.  
**3NF** All non-key columns must be mutually independent. Obvious example of a dependency is a calculated column, which is a no-no.

**Denormalisation**

Process of defining a table that is not normalized to enhance performance and simplify queries. Recommended when normalization causes many queries to join in excess of four (4) ways.

**Transact-SQL Batch**

```
DECLARE @dbname varchar(25)
DECLARE @tblname varchar(25)
SET @dbname = 'Sales'
SET @tblname = 'Customer'
EXEC ('USE ' + @dbname +
'SELECT * FROM ' + @tblname)
```

**Denormalised**

Orderid	Customerid	Items
1	4	4 5 hammer, 3 screwdriver, 6 monkey wrench
2	23	1 hammer
3	15	2 deluxe garden hose, 2 economy nozzle
4	2	15 10' 2x4 untreated pine board
5	23	1 screwdriver
6	2	5 key

Orderid	Customerid	Orderitem#	Quantity	Item
1	4	1	5	hammer
1	4	2	3	screwdriver
1	4	3	6	monkey wrench
2	23	1	1	hammer
3	15	1	2	deluxe garden hose
3	15	2	2	economy nozzle
4	2	1	15	10' 2x4 untreated pine board
5	23	1	1	screwdriver
6	2	1	5	key

Orderid	Customerid	OrderDate
1	1	5/1/94
2	3	5/9/94
3	1	7/4/94
4	2	8/1/94
5	1	8/2/94
6	2	8/2/94

Orderid	Orderitem#	Quantity	Productid
1	1	5	32
1	2	3	2
2	1	1	32
3	1	2	113
3	2	2	121
4	1	15	1024
5	1	1	2
6	1	5	52

Productid	ProductDescription
2	screwdriver
32	hammer
52	key
113	deluxe garden hose
121	economy nozzle
1024	10' 2x4 untreated pine boards

**1NF**  
**2NF**  
**3NF**

**Data Integrity**

Domain (column) integrity	NULL, DEFAULT, CHECK	Datatypes, Rules, Defaults, Triggers, SPs
Entity (table) integrity	PK, NULL, UNIQUE	UNIQUE indexes, New Identity datatype
Referential integrity	FK, REFERENCES	Triggers, SPs
User-defined integrity	FK, CHECK, REFERENCES	Rules, Triggers, SPs

Stored procedure: EXEC update\_prices -or- Sales.dbo.update\_prices

**Triggers**

**INSERT** ... rows added to the table and an in memory table called "inserted".  
**UPDATE** ... original rows moved to an in-memory table "deleted" and new rows added to the "inserted" table.  
**DELETE** ... rows deleted are moved to the "deleted" table.

**Nested Triggers** ... Triggers can be nested up to 32 levels. If a trigger changes a table on which there is another trigger, the second trigger is activated and can then call a third trigger, and so on. If any trigger in the chain sets off an infinite loop, the nesting level is exceeded and the trigger is canceled. To disable nested triggers, set the nested\_triggers option of sp\_configure to 0 (off). The default configuration allows nested triggers. If nested triggers is off, recursive triggers is also disabled, regardless of the recursive\_triggers setting of sp\_dboption.

**Data Types**

**bit** Integer data with either a 1 or 0 value.

**bigint** 64-bit Integer (whole number) data from -2^63 (-9,223,372,036,854,775,807) through 2^63 (9,223,372,036,854,775,807).

**int** 32-bit Integer (whole number) data from -2^31 (-2,147,483,648) through 2^31 - 1 (2,147,483,647).

**smallint** 16-bit Integer data from 2^15 (-32,768) through 2^15 - 1 (32,767).

**tinyint** Integer data from 0 through 255.

**decimal** Fixed precision and scale numeric data from -10^38 -1 through 10^38 -1.

**numeric** A synonym for decimal.

**money** Monetary data values from -2^63 (-922,337,203,685,477.5808) through 2^63 - 1 (+922,337,203,685,477.5807), with accuracy to a ten-thousandth of a monetary unit.

**smallmoney** Monetary data values from -214,748.3648 through +214,748.3647, with accuracy to a ten-thousandth of a monetary unit.

**float** Floating precision number data from -1.79E + 308 through 1.79E + 308.

**real** Floating precision number data from -3.40E + 38 through 3.40E + 38.

**datetime** Date and time data from January 1, 1753, to December 31, 9999, with an accuracy of three-hundredths of a sec, or 3.33 millisecond.

**smalldatetime** Date and time data from January 1, 1900, through June 6, 2079, with an accuracy of one minute.

**cursor** A reference to a cursor.

**timestamp** A database-wide unique number.

**uniqueidentifier** A globally unique identifier (128-bit GUID).

**char** Fixed-length non-Unicode character data with a maximum length of 8,000 chars.

**nchar** Fixed-length Unicode data with a maximum length of 4,000 characters.

**varchar** Variable-length non-Unicode data with a maximum of 8,000 characters.

**nvarchar** Variable-length Unicode data with a maximum length of 4,000 characters. sysname is a system-supplied user-defined data type that is a synonym for nvarchar(128) and is used to reference database object names.

**text** Variable-length non-Unicode data with a maximum length of 2^31 - 1 (2,147,483,647) characters.

**ntext** Variable-length Unicode data with a maximum length of 2^30 - 1 (1,073,741,823) characters.

**binary** Fixed-length binary data with a maximum length of 8,000 bytes.

**varbinary** Variable-length binary data with a maximum length of 8,000 bytes.

**image** Variable-length binary data with a maximum length of 2^31 - 1 (2,147,483,647) bytes.

