Transact-SQL Cheat Sheet

by thecodereaper27 via cheatography.com/66567/cs/16616/

Overview

Transact-SQL is the language used to query data in Microsoft SQL Server and Azure SQL Database.	
SQL is declarative, not procedural.	
Data is stored in tables, which may be related to one another through common key fields (*primary* and *foreign*).	
Schemas are namespaces for database objects.	
Fully-qualified name for an object	`[server_name].[database_name].[schema_name].object_name`
Best practice within database context	`schema_name.object_name`
Always terminate a SQL statement with a semi-colon, `;`.	
Relational Databases	

SQL Statement Types

DML	Data Manipulation Language	Statements for querying and modifying data.	`SELECT, INSERT, UPDATE, DELETE`
DDL	Data Definition Language	Statements for defining database objects.	`CREATE, ALTER, DROP`
DCL	Data Control Language	Statements for assigning permissions.	`GRANT, REVOKE, DENY`

NULL

`NULL`	NULL is used to indicate an unknown or missing value.	NULL is not equivalent to zero or an empty string. Arithmetic or string concatenation operations involving one or more NULL operands return NULL. For example, 12 + NULL = NULL.
`ISNULL(column/variable, value)`	Return *value* if the column or variable is `NULL`.	If you need to compare a value to NULL, use the IS operator instead of the = operator.
`NULLIF(column/variable, value)`	Returns `NULL` if the column or variable is *value*.	The NULLIF function returns NULL when a column or variable contains a specified value.
`COALESCE(column/variable1, column/variable2, ...)`	Returns the *value* of the first non-`NULL` column or variable in the list.	The COALESCE function returns the first non-NULL value in a specified list of columns or variables.

SELECT Statement

Main query clauses in *keyed-in order*		
`SELECT`	<select list>	Defines which columns to return.
`FROM`	<table source>	Defines table(s) to query.
`WHERE`	<search condition>	Filters rows using a predicate.
`GROUP BY`	<group by list>	Arranges rows by groups.
`HAVING`	<search condition>	Filters groups using a predicate.
`ORDER BY`	<order by list>	Sorts the output.
Logical query processing order		

SELECT Statement (cont)

`FROM`

`WHERE`

`GROUP BY`

`HAVING`

`SELECT`

`ORDER BY`

Notes

Use the `SELECT` statement to retrieve a rowset of data from tables and views in a database.

In the `SELECT` clause, you can use `*` to return all columns, but generally you should specify explicit columns.

You can specify expressions in the `SELECT` clause to return the results of calculations.

You can use the `AS` keyword to specify aliases for columns in the rowset returned by the`SELECT` statement.

By default, the SELECT statement returns all rows. If mulitple rows contain the same values for every column, they are duplicated in the results. Using the DISTINCT keyword eliminates duplicates, ensuring that only one row for each distinct combination of column values is returned.

The order of rows in the result of a SELECT statement is not guaranteed unless you explicitly specify one or more columns in an ORDER BY clause. You can specify sort direction as ASC (the default) or DESC.

You can combine the ORDER BY clause with the TOP keyword to retrict the results so that they include only the top n rows (where n is the number or percentage of rows you want to return).

You can implement a query to retrieve a specified "page" of results by using the OFFSET and FETCH keywords with the ORDER BY clause.

Use the WHERE clause to filter the results returned by a SELECT query based on a search condition.

A search condition is composed of one or more predicates.

Predicates include conditional operators (such as =, >, and <), IN, LIKE, and NOT.

You can use AND and OR to combine predicates based on Boolean logic.

Example

```
SELECT country, YEAR(hiredate) AS yearhired, COUNT(*) AS numemployees
FROM HR.Employees
WHERE hiredate >= '20140101'
GROUP BY country, YEAR(hiredate)
HAVING COUNT(*) > 1
ORDER BY country, yearhired DESC;
```

Data Types

Transact-SQL supports a wide range of data types, which can be broadly categorized as exact numeric, approximate numeric, character, date/time, binary, and other (which includes specialized data types for handling data such as XML and spatial data).

Some data types are compatible, and values can be implicitly converted between them. Conversion between other data types requires the use of explicit conversion functions.