Huy Bui
Information Technology
T5616SN

DATABASE
Simple queries
5 February 2018

1 Query in SQL

The purpose of this article is to lay out the basic structure and use of SQL SELECT queries and statements. These statements are part of Transact-SQL (T-SQL) language specification and are central to the use of Microsoft SQL Server.

Once we have data in the database, we want to look at the data and analyze it in an endless number of different ways, constantly varying the filtering, sorting, and calculations that we apply to the raw data. The SQL SELECT statement is what we use to choose, or select, the data that we want returned from the database to our application.

1.1 The SELECT ... FROM clause

The most basic SELECT statement has only 2 parts: (1) what columns you want to return and (2) what table(s) those columns come from. If we want to retrieve all of the information about all of the customers in the Employees table, we could use the asterisk (*) as a shortcut for all of the columns, and our query looks like

```
1    SELECT * FROM Employees
```

If we want only specific columns (as is usually the case), we can/should explicitly specify them in a comma-separated list, as in

```
1    SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees
```

which results in the specified fields of data for all of the rows in the table:

EmployeeID	FirstName	LastName	HireDate	City
1	Nancy	Davolio	1/5/1992 12:00:00 AM	Seattle
2	Andrew	Fuller	14/8/1992 12:00:00 AM	Tacoma
3	Janet	Leverling	1/4/1992 12:00:00 AM	Kirkland
4	Margaret	Peacock	3/5/1993 12:00:00 AM	Redmond
5	Steven	Buchanan	17/10/1993 12:00:00 AM	London
6	Michael	Suyama	17/10/1993 12:00:00 AM	London
7	Robert	King	2/1/1994 12:00:00 AM	London
8	Laura	Callahan	5/3/1994 12:00:00 AM	Seattle
9	Anne	Dodsworth	15/11/1994 12:00:00 AM	London

Figure 1: Select multiple columns

1.2 The WHERE clause

The next thing we want to do is to start limiting, or filtering, the data we fetch from the database. By adding a WHERE clause to the SELECT statement, we add one (or more) conditions that must be met by the selected data. This will limit the number of rows that answer the query and are fetched. In many cases, this is where most of the "action" of a query takes place. We can continue with our previous query, and limit it to only those employees living in London:

```
1    SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees
2    WHERE City = 'London'
```

EmployeeID	FirstName	LastName	HireDate	City
5	Steven	Buchanan	17/10/1993 12:00:00 AM	London
6	Michael	Suyama	17/10/1993 12:00:00 AM	London
7	Robert	King	2/1/1994 12:00:00 AM	London
9	Anne	Dodsworth	15/11/1994 12:00:00 AM	London

Figure 2: List all employees living in London using WHERE clause

1.3 The ORDER BY clause

Until now, we have been discussing filtering the data: that is, defining the conditions that determine which rows will be included in the final set of rows to be fetched and returned from the database. Once we have determined which columns and rows will be included in the results of our SELECT query, we may want to control the order in which the rows appear—sorting the data. To sort the data rows, we include the ORDER BY clause. The ORDER BY clause includes one or more column names that specify the sort order. If we return to one of our first SELECT statements, we can sort its results by City with the following statement:

```
SELECT EmployeeID, FirstName, LastName, HireDate, City FROM Employees
ORDER BY City
```

EmployeeID	FirstName	LastName	HireDate	City
3	Janet	Leverling	1/4/1992 12:00:00 AM	Kirkland
5	Steven	Buchanan	17/10/1993 12:00:00 AM	London
6	Michael	Suyama	17/10/1993 12:00:00 AM	London
7	Robert	King	2/1/1994 12:00:00 AM	London
9	Anne	Dodsworth	15/11/1994 12:00:00 AM	London
4	Margaret	Peacock	3/5/1993 12:00:00 AM	Redmond
1	Nancy	Davolio	1/5/1992 12:00:00 AM	Seattle
8	Laura	Callahan	5/3/1994 12:00:00 AM	Seattle
2	Andrew	Fuller	14/8/1992 12:00:00 AM	Tacoma

Figure 3: Sort results by City

1.4 The GROUP BY clause

The GROUP BY clause groups the SELECT statement results according to the values in a list of one or more column expressions. For example, consider the Sales table ().

Country	**Region**	**Sales**
Canada	Alberta	100
Canada	British Columbia	200
Canada	British Columbia	300
United States	Montana	100

This next query groups Country and Region and returns the aggregate sum for each combination of values.

```
SELECT Country, Region, SUM(sales) AS TotalSales
FROM Sales
GROUP BY Country, Region;
```

The GROUP BY clause also interacts with other statements, such as COUNT, AVG, HAVING. As for HAVING clause, it searchs condition for a group or an aggregate. COUNT returns the number of items in a group. And AVG returns the average of the values in a group.

2 Practical part

The following query lists all staff with age greater than 50 years including at columns FirstName, FamilyName, and DateOfBirth from Staff table (Figure 4).

```
SELECT FirstName, FamilyName,
FORMAT(DateOfBirth, 'D', 'en-US') AS Birthday
FROM Staff
WHERE DateOfBirth < '1968-2-2';
```

	FirstName	FamilyName	Birthday
1	John	White	Monday, October 1, 1945
2	Justin	Case	Saturday, November 1, 1958
3	Ann	Beech	Thursday, November 10, 1960
4	David	Ford	Monday, March 24, 1958
5	April	Showers	Thursday, January 24, 1963
6	Susan	Brand	Monday, June 3, 1940
7	Julie	Lee	Sunday, June 13, 1965
8	Cris	Cross	Saturday, September 23, 1967
9	Talisha	Ditton	Monday, October 5, 1959
10	Pacey	Hailstone	Tuesday, April 12, 1955

Figure 4: Older-than-50 staffs and their birthday

The following query produces a list of monthly salary for all staff, showing first and last names, and salary details, and annual salary information in table Staff (Figure 5).

```
SELECT FirstName, FamilyName,
CAST(Salary/12 AS DECIMAL(8,2)) AS 'Monthly sSalary'
FROM Staff
```

	FirstName	FamilyName	Monthly sSalary
1	John	White	2507.55
2	Justin	Case	2107.58
3	Mary	Howe	750.64
4	Lee	Walker	1586.22
5	Ann	Beech	1002.54
6	David	Ford	1504.23
7	April	Showers	1670.93

Figure 5: Show monthly salary for all staff

The following query find all owners who live in Glasgow. Include at least FirstName, FamilyName, and City from PrivateOwner table in your result table (Figure 6).

```
SELECT FirstName, FamilyName, City
FROM PrivateOwner
WHERE City='Glasgow'
```

	FirstName	FamilyName	City
1	Carol	Farrel	Glasgow
2	Tina	Murphy	Glasgow
3	Tony	Shaw	Glasgow

Figure 6: Show all staffs who live in Glasgow

The following query produces a list of all staff, arranged in alphabetical order of first name (Figure 7).

```
SELECT FirstName, FamilyName
FROM Staff
ORDER BY FirstName
```

	FirstName	FamilyName
1	Abbey	Felton
2	Ann	Beech
3	April	Showers
4	Chrystal	Cameron
5	Cris	Cross
6	David	Ford
7	Hadley	Bailor

Figure 7: Show staffs' names in alphabetical order

The following query produces a list of all staff, arranged in order of age (Figure 8).

```
SELECT FirstName, FamilyName,
FORMAT(DateOfBirth, 'yyyy', 'en-US') AS YearOfBirth,
CAST( YEAR(GETDATE()) - YEAR(DateOfBirth) AS INT) AS Age
FROM Staff
ORDER BY Age
```

	FirstName	FamilyName	YearOfBirth	Age
1	Octavia	Callaway	1999	19
2	Hadley	Bailor	1998	20
3	Lakeisha	Mallet	1988	30
4	Ubon	Edginton	1986	32
5	Abbey	Felton	1981	37
6	Chrystal	Cameron	1979	39
7	Vadin	Justice	1979	39
8	Hana	Alburg	1977	41
9	Lee	Walker	1972	46

Figure 8: Show staffs' ages in ascending order

The following query produces a list of properties for rent in order of property type (major sort key) in ascending alphabetic order, and within property type, in descending order of rent (minor sort key) (Figure 9).

```
SELECT Street, City, Postcode, TypeOfProperty, NumberOfRooms, Rent
FROM PropertyForRent
ORDER BY TypeOfProperty DESC, Rent DESC
```

The following query lists the number of properties in each property type (Figure 10).

```
SELECT TypeOfProperty, COUNT(*) AS "Number of properties"
FROM PropertyForRent
GROUP BY TypeOfProperty
HAVING COUNT(*) > 1
ORDER BY COUNT(*) DESC
```

The following query lists the number of properties managed by each staff member (Figure 11).

```
SELECT StaffId, COUNT(*) AS "Number of properties"
FROM PropertyForRent
GROUP BY StaffId
HAVING COUNT(*) > 1
```

The following query lists the number of properties viewed more than once as well as the number of viewing (Figure 12).

```
SELECT PropertyForRentId, COUNT(*) AS "Number of Viewings"
FROM Viewing
GROUP BY PropertyForRentId
HAVING COUNT(*) > 1
```

The following query shows average rent for each property type (Figure 13).

```
SELECT TypeOfProperty,
COUNT(*) AS "Number of properties",
```

Street	City	Postcode	TypeOfProperty	NumberOfRooms	Rent
Chiswick High Street 15	London	T44 3CK	Villa	5	1450.00
Prince Consort Road 26	London	V65 6KK	Villa	5	1365.00
Uxbridge Road 55	London	S34 2CX	Villa	6	1170.00
Eaton Terrace 99	London	T34 6CJ	Villa	4	1155.00
Slippery Lane 16	London	B52 5CD	Villa	5	1135.00
Wretham Street 6	Bristol	W93 2BG	Villa	5	940.00
3 Argyll Street	London	RW5 7EN	Villa	7	900.00
Rickman Drive 9	Bristol	D44 5YY	Villa	6	810.00
Lytham Croft 14	Bristol	L71 8UU	Villa	5	780.00
Stoke Hill	Bristol	Q16 9JX	Villa	7	750.00
Froghart Terrace 26	Aberdeen	T52 4JS	House	4	850.00
Crimon Plaza 65	Aberdeen	S99 1KK	House	4	810.00
Urguhart Road 12	Aberdeen	K88 6CV	House	4	780.00

Figure 9: Sort list in order of property type and rent

TypeOfProperty	Number of properties
Villa	10
Bungalow	6
House	6
Flat	4
Cottage	3

Figure 10: Sort list in order of property type and rent

StaffId	Number of properties
NULL	3
2	3
4	4
7	8
10	6
20	5

Figure 11: Sort list in order of property type and rent

PropertyForRentId	Number of Viewings
5	2
6	2
7	2
24	2
25	3

Figure 12: Sort list in order of property type and rent

```
3 CAST(AVG(Rent) AS DECIMAL(8,2)) AS "Average rent"
4 FROM PropertyForRent
5 GROUP BY TypeOfProperty
6 HAVING COUNT(*) > 1
```

TypeOfProperty	Number of properties	Average rent
Bungalow	6	1019.17
Cottage	3	498.33
Flat	4	393.75
House	6	723.33
Villa	10	1045.50

Figure 13: Sort list in order of property type and rent