Summary queries

UNIT 6

Introduction

- A summary query is a query that obtains totals from one or several tables.

- We will learn about its specific features:
  - Aggregate functions,
  - GROUP BY clause.
  - HAVING clause.
  - specific rules

Introduction

```
SELECT Oficina, Region, Ventas
FROM Oficinas
ORDER BY Region
```

```
SELECT Region, SUM (Ventas)
FROM Oficinas
GROUP BY Region
```

Oficina	Region	Ventas
24	centro	150.000 Pts.
23	centro	
28	este	0 Pts.
13	este	368.000 Pts.
12	este	735.000 Pts.
11	este	693.000 Pts.
26	norte	
22	oeste	185.000 Pts.
21	oeste	836.000 Pts.

Region	Suma de Ventas
centro	150.000
este	1.796.000
norte	
oeste	1.021.000

Normal query
One row from one source row

Summary query
One row from several source rows.

Aggregate functions

- Aggregate functions perform a calculation on a set of values and return a single value.
- All have the same structure:

  FunctionName([ALL|DISTINCT] expression)

- Expression is a column name or an expression based on columns (of course, source columns) which indicates the set of values.
- The WHERE clause is performed before the functions.
- Except for COUNT, aggregate functions ignore null values.
- With DISTINCT duplicated values are considered only once.
- Aggregate functions cannot be nested.

COUNT() / COUNT_BIG()

COUNT([ALL|DISTINCT] expression)

Returns the number of items in a group, except NULL items.
Expression is an expression of any type except text, image, ntext.
For return values greater than 2^{31}-1, COUNT produces an error
→ Use COUNT_BIG instead.

COUNT(*)

Counts the number of rows returned after the WHERE clause.

```
SELECT COUNT(region)
FROM oficinas;

SELECT COUNT(DISTINCT region)
FROM oficinas;
```

```
SELECT COUNT(numemp)
FROM empleados;

SELECT COUNT(*)
FROM empleados;
```

MAX() / MIN()

MAX([ALL|DISTINCT] expression)
MIN([ALL|DISTINCT] expression)

MAX Returns the maximum value in the expression.
MIN Returns the minimum value in the expression

expression can be a numeric, character, uniqueidentifier, or datetime column, but not a bit column.

DISTINCT is not meaningful with MAX/MIN (the max value will be the same considering or not duplicated values) and is available for ISO compatibility only .

Ex:

```
SELECT MAX(objetivo) AS MayorObjetivo
FROM oficinas;

SELECT MIN(contrato) AS PrimerContrato
FROM empleados;
```

SUM() / AVG()

SUM([ALL|DISTINCT] expression)
AVG([ALL|DISTINCT] expression)

SUM Returns the sum of all the values.
AVG Returns the average of the values.

SUM /AVG can be used with numeric columns only.
Null values are ignored.
The result has the same datatype and can have greater precision.
An overflow error may occur.

Ex:

```
SELECT SUM(ventas) AS VentasTotales, MAX(objetivo) AS MayorObjetivo
FROM oficinas;
```

Statistical functions

VAR([ALL|DISTINCT] expression)

Returns the statistical variance of all values in the specified expression.

VARP([ALL|DISTINCT] expression)

Returns the statistical variance for the population for all values in the specified expression.

STDEV([ALL|DISTINCT] expression)

Returns the statistical standard deviation of all values in the specified expression.

STDEVP([ALL|DISTINCT] expression)

Returns the statistical standard deviation for the population for all values in the specified expression.

They can be used with numeric values only.

GROUPING

GROUPING(ColumnNa)

GROUPING is used to distinguish the null values that are returned by ROLLUP or CUBE from standard null values.

- GROUPING returns 1 for aggregated null in the result set.
- or 0 for not aggregated.

GROUPING can be used only when GROUP BY is specified with CUBE or ROLLUP.

ColumnNa must be a column in the GROUP BY clause.

First study the GROUP BY clause and then you will understand this function.

GROUP BY

GROUP BY [ALL] groupby_expression [,...*n*]
[WITH { CUBE | ROLLUP }]

Groups a selected set of rows into a set of summary rows by the values of one or more columns or expressions based on columns.

One row is returned for each group.

When using several columns in a GROUP BY clause, the same rules as the ORDER BY clause are applied.

groupby_expression can be:

- a source column (returned by the FROM clause).
- an expression
  - based on a source column ,
  - aggregated functions are not permitted,
  - of any datatype except text, image, ntext.

It is also known as a grouping column.

GROUP BY

When using CUBE o ROLLUP a maximum of 10 grouping columns are allowed.

When not using CUBE o ROLLUP the maximum number of grouping columns depends on the number and size of the grouping columns.

If a grouping column contains null values, all null values are considered equal, and they are put into a single group.

IN THE SELECT LIST THERE CANNOT BE A SIMPLE COLUMN (COLUMN WITHOUT AGGREGATE FUNCTION) IF IT IS NOT A GROUPING COLUMN.

GROUP BY

```
SELECT rep, clie, count(numpedido) AS [Número de pedidos],
MAX(importe) AS [Importe máximo]
FROM pedidos
WHERE YEAR(fechapedido) = 1997
GROUP BY  rep, clie
ORDER BY rep, clie;
```

Result:

Rep	clie	Número de pedidos	Importe máximo
101	2113	1	225,00
102	2106	2	21,30
102	2120	1	37,50
103	2111	2	21,00
105	2103	4	275,00
105	2111	1	37,45
106	2101	1	14,58
107	2109	1	313,50
107	2124	2	24,30
108	2112	1	29,25
108	2114	1	71,00
108	2118	3	14,20

GROUP BY ALL

ALL includes all groups and result sets, even those that do not have any rows that meet the search condition specified in the WHERE clause.

When ALL is specified, null values are returned for the summary columns of groups that do not meet the search condition.

You cannot specify ALL with the CUBE or ROLLUP operators.

This feature will be removed in a future version of Microsoft SQL Server. Avoid using it.

GROUP BY ALL

```
SELECT rep, clie, count(numpedido) AS [Número de pedidos],
MAX(importe) AS [Importe máximo]
FROM pedidos
WHERE YEAR(fechapedido) = 1997
GROUP BY  ALL rep, clie
ORDER BY rep, clie;
```

Result

Rep	clie	Número de pedidos	Importe máximo
101	2102	0	NULL
101	2108	0	NULL
101	2113	1	225,00
102	2106	2	21,30
102	2120	1	37,50
103	2111	2	21,00
105	2103	4	275,00
105	2111	1	37,45
106	2101	1	14,58
106	2117	0	NULL
107	2109	1	313,50
107	2124	2	24,30
108	2112	1	29,25
108	2114	1	71,00
108	2118	3	14,20

GROUP BY – ROLLUP / CUBE

GROUP BY [ALL] expression_*agrupacion* [,...*n*]
[WITH { CUBE | ROLLUP }]

ROLLUP generates the simple GROUP BY aggregate rows, plus subtotals and also a grand total row.

With GROUP BY a,b,c WITH ROLLUP, one row with a subtotal is generated for each unique combination of values of (a,b,c), (a, b), (a) and a grand total row is also calculated.

CUBE outputs a grouping for all permutations of expressions in the <composite element list>.

For example GROUP BY a,b,c WITH CUBE

One row with a subtotal is generated for each unique combination of (a,b,c), (a, b), (a, c), (b, c), (a), (b) and (c). And a grand total row is also calculated.

GROUP BY – ROLLUP

SELECT rep, clie, count(numpedido) AS [Número de pedidos], MAX(importe) AS [Importe máximo] FROM pedidos
WHERE YEAR(fechapedido) = 1997 **GROUP BY rep, clie WITH ROLLUP**;

Rep	clie	Nºpedidos	Importe máximo
101	2113	1	225,00
101	NULL	1	225,00
102	2106	2	21,30
102	2120	1	37,50
102	NULL	3	37,50
103	2111	2	21,00
103	NULL	2	21,00
105	2103	4	275,00
105	2111	1	37,45
105	NULL	5	275,00
106	2101	1	14,58
106	NULL	1	14,58
107	2109	1	313,50
107	2124	2	24,30
107	NULL	3	313,50
108	2112	1	29,25
108	2114	1	71,00
108	2118	3	14,20
108	NULL	5	71,00
-	-	-	-
NULL	NULL	23	450,00

GROUP BY – CUBE

SELECT rep, clie, count(numpedido) AS [Número de pedidos], MAX(importe) AS [Importe máximo] FROM pedidos
WHERE YEAR(fechapedido) = 1997 GROUP BY rep, clie WITH CUBE;

You must add these rows to the previous result:

Rep	clie	Nºpedidos	Importe máximo
NULL	2101	1	14,58
NULL	2103	4	275,00
NULL	2106	2	21,30
NULL	2107	1	6,32
NULL	2108	1	56,25
NULL	2109	1	313,50
NULL	2111	3	37,45
NULL	2112	2	450,00
NULL	2113	1	225,00
NULL	2114	1	71,00
NULL	2118	3	14,20
NULL	2120	1	37,50
NULL	2124	2	24,30

GROUPING()

```
SELECT rep, clie, count(numpedido) AS [Nºpedidos], MAX(importe) AS [Importe máximo], GROUPING(clie) AS [Fila resumen] , GROUPING(rep) AS [Total final]
FROM pedidos
WHERE YEAR(fechapedido) = 1997
GROUP BY  rep, clie WITH ROLLUP;
```

Rep	clie	Nºpedidos	Importe máximo	Fila Resumen	Total Final
101	2113	1	225,00	0	0
101	NULL	1	225,00	1	0
102	2106	2	21,30	0	0
102	2120	1	37,50	0	0
102	NULL	3	37,50	1	0
103	2111	2	21,00	0	0
…	…	…	…	…	…
NULL	NULL	23	450,00	1	1

HAVING

HAVING works like the WHERE clause, but instead of selecting source rows, it selects result rows from a summary query.

It selects summary rows so that in the condition you won't be able to include any kind of column, only what you can express in the select list.

IN THE HAVING CLAUSE THERE CANNOT BE A SIMPLE COLUMN (COLUMN WITHOUT AGGREGATE FUNCTION) IF IT IS NOT A GROUPING COLUMN.

```
SELECT oficina,
count(numemp) AS [Número
de empleados]
FROM empleados
GROUP BY oficina
HAVING COUNT(numemp)<2;
```

```
SELECT oficina,
count(numemp) AS [Número de
empleados]
FROM empleados
GROUP BY oficina, objetivo
HAVING SUM(ventas)<objetivo;
```

Rules to remenber

IN THE SELECT LIST THERE CANNOT BE A SIMPLE COLUMN (COLUMN WITHOUT AGGREGATE FUNCTION) IF IT IS NOT A GROUPING COLUMN.

IN THE HAVING CLAUSE THERE CANNOT BE A SIMPLE COLUMN (COLUMN WITHOUT AGGREGATE FUNCTION) IF IT IS NOT A GROUPING COLUMN.

Use of derived tables

- We have seen that aggregate functions cannot be nested. In those cases the use of a derived table would solve the problem.
- Ex. We want to know the maximum number of employees per office.
- So we have to calculate the maximum number of employees per office. It would be MAX(COUNT(numemp)) but we cannot nest aggregate functions.
- We can solve this problem using a derived table:

```
SELECT MAX(cuantosempleados) AS media
FROM (SELECT COUNT(numemp) AS cuantosempleados
        FROM Empleados GROUP BY oficina) AS derivada;
```

Remember

- Knowing in which order the clauses are performed can be useful to understand what happens when we run a query and what we can or cannot do.
    - FROM
    - WHERE
    - GROUP BY
    - HAVING
    - Select list
    - DISTINCT (in select list)
    - ORDER BY
    - TOP