Unidad 1 – Window Functions

Bases de Datos Aplicada

v1.0 – Marzo 2024

Universidad Nacional de La Matanza

DIIT
Departamento de Ingeniería e Investigaciones Tecnológicas

Conceptos en común

Las funciones que vamos a ver tienen una sintaxis similar. Veamos esos argumentos que aplican para todas por igual:

FUNCION () OVER (PARTITION BY … ORDER BY….) AS….

PARTITION BY → Divide el conjunto de resultados generado por la cláusula FROM en particiones a las que se aplica la función. Si no se especifica, la función trata todas las filas del conjunto de resultados de la consulta como un único grupo.

ORDER BY → Determina el orden de los datos antes de que se aplique la función. En la mayoría de las funciones es obligatoria

Conceptos en común

CON PARTITION BY

	ProductoID	Nombre	Categoria	Precio	OrdenPorCategoria
1	6	Reloj	Accesorio	40	1
2	7	Lentes	Accesorio	35	2
3	5	Gorra	Accesorio	20	3
4	3	Zapatos	Ropa	50	1
5	2	Pantalon	Ropa	30	2
6	1	Camisa	Ropa	25	3
7	4	Bufanda	Ropa	15	4

```
FUNC() OVER (PARTITION BY
Categoria ORDER BY Precio DESC)
```

Vemos como “agrupa” y particiona por *categoria* en este caso

SIN PARTITION BY

	ProductoID	Nombre	Categoria	Precio	OrdenPorCategoria
1	3	Zapatos	Ropa	50	1
2	6	Reloj	Accesorio	40	2
3	7	Lentes	Accesorio	35	3
4	2	Pantalon	Ropa	30	4
5	1	Camisa	Ropa	25	5
6	5	Gorra	Accesorio	20	6
7	4	Bufanda	Ropa	15	7

Funciones de Categoría

Las funciones de categoría devuelven un valor para cada fila de una partición. Según la función que se utilice algunas filas pueden recibir el mismo valor que otras. Las funciones de categoría son no deterministas.

RANK

DENSE_RANK

NTILE

ROW_NUMBER

Universidad Nacional de La Matanza
DIIT Departamento de Ingeniería e Investigaciones Tecnológicas

RANK()

Introducción

Devuelve el rango de cada fila en la partición de un conjunto de resultados. Asigna una clasificación única a cada fila distinta dentro de un conjunto de resultados en función de los valores de una o más columnas. Si dos (o más) filas tienen los mismos valores en las columnas especificadas, obtienen la misma clasificación.

RANK()

Sintaxis

RANK ( ) OVER ([partition_by_clause] order_by_clause)

Observaciones

Si dos o más filas se enlazan en un rango, cada fila enlazada recibe el mismo rango. Por ejemplo, si los dos mejores vendedores tienen el mismo valor de Ventas, los dos tienen el rango uno. El vendedor con el siguiente valor más alto de Ventas recibe el rango tres, porque ya hay dos filas con un rango superior. Por tanto, la función RANK no siempre devuelve enteros consecutivos.

RANK()

Ejemplo

```
RANK() OVER (PARTITION BY i.LocationID
ORDER BY i.Quantity DESC) AS Rank
```

ProductID	Name	LocationID	Quantity	Rank
494	Paint - Silver	3	49	1
495	Paint - Blue	3	49	1
493	Paint - Red	3	41	3
496	Paint - Yellow	3	30	4
492	Paint - Black	3	17	5
495	Paint - Blue	4	35	1
496	Paint - Yellow	4	25	2
493	Paint - Red	4	24	3
492	Paint - Black	4	14	4
494	Paint - Silver	4	12	5

(10 row(s) affected)

DENSE_RANK()

Introducción

Esta función devuelve el rango de cada fila dentro de una partición del conjunto de resultados, sin espacios en los valores de clasificación. El rango de una fila específica es uno más el número de valores de rango distintos anteriores a esa fila específica.

DENSE_RANK()

Sintaxis

DENSE_RANK ( ) OVER ([<partition_by_clause>] < order_by_clause >)

Observaciones

Si dos o más filas tienen el mismo valor de rango en la misma partición, cada una de esas filas recibirá el mismo rango. Por ejemplo, si los dos mejores vendedores tienen el mismo valor de Ventas, los dos tendrán un valor de rango de uno. El siguiente vendedor con mayor valor en Ventas tendrá un valor de rango de dos. Esto supera en uno el número de rangos distintos anteriores a la fila en cuestión. Por tanto, los números devueltos por la función DENSE_RANK no tienen espacios y siempre tienen valores de rango consecutivos.

DENSE_RANK()

Ejemplo

ProductID	Name	LocationID	Quantity	Rank
494	Paint - Silver	3	49	1
495	Paint - Blue	3	49	1
493	Paint - Red	3	41	2
496	Paint - Yellow	3	30	3
492	Paint - Black	3	17	4
495	Paint - Blue	4	35	1
496	Paint - Yellow	4	25	2
493	Paint - Red	4	24	3
492	Paint - Black	4	14	4
494	Paint - Silver	4	12	5

```
DENSE_RANK() OVER
(PARTITION BY i.LocationID
ORDER BY i.Quantity DESC)
AS Rank
```

NTILE()

Introducción

Distribuye las filas de una partición ordenada en un número especificado de grupos. Los grupos se numeran a partir del uno. Para cada fila NTILE devuelve el número del grupo al que pertenece la fila.

NTILE()

Sintaxis

NTILE (integer_expression) OVER ([<partition_by_clause>] < order_by_clause >)

Argumento nuevo

La cláusula INTEGER_EXPRESSION es una expresión de tipo entero positivo que especifica el número de grupos en que se debe dividir cada partición. Puede ser de tipo int o bigint.

NTILE()

Observaciones

- Si el número de filas de una partición no se puede dividir por *integer_expression*, producirá grupos de dos tamaños que difieren en un miembro.
- Los grupos de mayor tamaño preceden a los grupos de menor tamaño en el orden especificado por la cláusula OVER. Por ejemplo, si el número total de filas es 53 y el número de grupos es cinco, los tres primeros grupos tienen 11 filas y los otros dos grupos tienen 10 filas cada uno.
- Por otra parte, si el número total de filas es divisible por el número de grupos, las filas se distribuyen por igual entre los grupos. Por ejemplo, si el número total de filas es 50 y hay cinco grupos, cada grupo contiene 10 filas.

NTILE()

Ejemplo

FirstName	LastName	Quartile	SalesYTD	PostalCode
Linda	Mitchell	1	4,251,368.55	98027
Michael	Blythe	1	3,763,178.18	98027
Jillian	Carson	2	3,189,418.37	98027
Tsvi	Reiter	2	2,315,185.61	98027
Garrett	Vargas	3	1,453,719.47	98027
Pamela	Ansman-Wolfe	4	1,352,577.13	98027
Jae	Pak	1	4,116,871.23	98055
Ranjit	Varkey Chudukatil	1	3,121,616.32	98055
José	Saraiva	2	2,604,540.72	98055
Shu	Ito	2	2,458,535.62	98055
Rachel	Valdez	3	1,827,066.71	98055
Tete	Mensa-Annan	3	1,576,562.20	98055
David	Campbell	4	1,573,012.94	98055
Lynn	Tsoflias	4	1,421,810.92	98055

```
NTILE(4) OVER(
PARTITION BY PostalCode
ORDER BY SalesYTD DESC)
AS Quartile
```

ROW_NUMBER()

Introducción

Enumera los resultados de un conjunto de resultados. Concretamente devuelve el número secuencial de una fila dentro de una partición de un conjunto de resultados, empezando por 1 para la primera fila de cada partición.

Sintaxis

```
ROW_NUMBER ( )
   OVER ( [ PARTITION BY value_expression , ... [ n ] ] order_by_clause )
```

ROW_NUMBER()

Ejemplos

Sin PARTITION BY

Row#	name	recovery_model_desc
1	maestro	SIMPLE
2	model	FULL
3	msdb	SIMPLE
4	tempdb	SIMPLE

```
SELECT ROW_NUMBER()
OVER(ORDER BY name ASC) AS Row#, name,
recovery_model_desc
FROM sys.databases
WHERE database_id < 5;
```

Con PARTITION BY

Row#	name	recovery_model_desc
1	model	FULL
1	maestro	SIMPLE
2	msdb	SIMPLE
3	tempdb	SIMPLE

```
SELECT ROW_NUMBER()
OVER(PARTITION BY recovery_model_desc
ORDER BY name ASC) AS Row#, name,
recovery_model_desc
FROM sys.databases
WHERE database_id < 5;
```

FUNCIONES ANALITICAS

Las funciones analíticas calculan un valor agregado a partir de un grupo de filas. A diferencia de las funciones de agregado, estas funciones pueden devolver varias filas para cada grupo. Puede usar las funciones analíticas para calcular medias móviles, totales acumulados, porcentajes o resultados de N valores superiores dentro de un grupo.

LEAD
LAG
CUME_DIST
PERCENT_RANK
FIRST_VALUE
LAST_VALUE

Universidad Nacional de La Matanza
DIIT Departamento de Ingeniería e Investigaciones Tecnológicas

PERCENT_RANK()

Introducción

Calcula el rango relativo de una fila dentro de un grupo de filas de SQL Server. Se utiliza PERCENT_RANK para evaluar la situación relativa de un valor dentro de un conjunto de resultados de la consulta o de una partición. PERCENT_RANK es similar a la función CUME_DIST.

PERCENT_RANK()

Sintaxis

```
PERCENT_RANK( )
   OVER ( [ partition_by_clause ] order_by_clause )
```

Observaciones

El intervalo de valores devueltos por PERCENT_RANK es mayor que 0 y menor o igual que 1. La primera fila de cualquier conjunto tiene un PERCENT_RANK de 0. Se incluyen valores NULL de forma predeterminada y se tratan como los posibles valores más bajos.

PERCENT_RANK()

Ejemplo

Department	LastName	Rate	CumeDist	PctRank
Document Control	Arifin	17.7885	1	1
Document Control	Norred	16.8269	0.8	0.5
Document Control	Kharatishvili	16.8269	0.8	0.5
Document Control	Chai	10.25	0.4	0
Document Control	Berge	10.25	0.4	0
Information Services	Trenary	50.4808	1	1
Information Services	Conroy	39.6635	0.9	0.888888888888889
Information Services	Ajenstat	38.4615	0.8	0.666666666666667
Information Services	Wilson	38.4615	0.8	0.666666666666667
Information Services	Sharma	32.4519	0.6	0.444444444444444
Information Services	Connelly	32.4519	0.6	0.444444444444444
Information Services	Berg	27.4038	0.4	0
Information Services	Meyyappan	27.4038	0.4	0
Information Services	Bacon	27.4038	0.4	0
Information Services	Bueno	27.4038	0.4	0

(15 row(s) affected)

```
CUME_DIST () OVER (PARTITION BY Department ORDER BY Rate) AS
CumeDist, PERCENT_RANK() OVER (PARTITION BY Department ORDER BY Rate
) AS PctRank
```

CUME_DIST()

Introducción

Esta función calcula la distribución acumulativa de un valor en un grupo de valores. Es decir, CUME_DIST calcula la posición relativa de un valor especificado en un grupo de valores. Suponiendo un orden ascendente, el CUME_DIST de un valor en la fila r se define como el número de filas con valores menores o iguales que el valor de la fila r, dividido entre el número de filas evaluadas en la partición o el conjunto de resultados de la consulta. CUME_DIST es similar a la función PERCENT_RANK.

CUME_DIST()

Sintaxis

```
CUME_DIST( )
   OVER ( [ partition_by_clause ] order_by_clause )
```

Observaciones

CUME_DIST devuelve un intervalo de valores mayor que 0 y menor o igual que 1. Los valores equivalentes siempre se evalúan como el mismo valor de distribución acumulativa. CUME_DIST incluye valores NULL de forma predeterminada y los trata como los posibles valores más bajos.

CUME_DIST()

Ejemplo

Department	LastName	Rate	CumeDist	PctRank
Document Control	Arifin	17.7885	1	1
Document Control	Norred	16.8269	0.8	0.5
Document Control	Kharatishvili	16.8269	0.8	0.5
Document Control	Chai	10.25	0.4	0
Document Control	Berge	10.25	0.4	0
Information Services	Trenary	50.4808	1	1
Information Services	Conroy	39.6635	0.9	0.888888888888889
Information Services	Ajenstat	38.4615	0.8	0.666666666666667
Information Services	Wilson	38.4615	0.8	0.666666666666667
Information Services	Sharma	32.4519	0.6	0.444444444444444
Information Services	Connelly	32.4519	0.6	0.444444444444444
Information Services	Berg	27.4038	0.4	0
Information Services	Meyyappan	27.4038	0.4	0
Information Services	Bacon	27.4038	0.4	0
Information Services	Bueno	27.4038	0.4	0

(15 row(s) affected)

```
CUME_DIST () OVER (PARTITION BY Department ORDER BY Rate)
AS CumeDist, PERCENT_RANK() OVER (PARTITION BY Department
ORDER BY Rate ) AS PctRank
```

LEAD()

Introducción

LEAD proporciona acceso a una fila en un desplazamiento físico especificado que hay después de la fila actual. Se utiliza esta función analítica en una instrucción SELECT para comparar valores de la fila actual con valores de una fila posterior.

LEAD()

Sintaxis

LEAD (scalar_expression [, offset] , [default]) [IGNORE NULLS | RESPECT NULLS]
OVER ([partition_by_clause] order_by_clause)

Argumentos

- Scalar_expresión: Es el valor que se va a devolver en función del desplazamiento especificado.
- Offset: El número de filas hacia delante de la fila actual de la que se va a obtener un valor. **Si no se especifica, el valor predeterminado es 1.**
- Default: Valor que se devuelve cuando *offset* está fuera del ámbito de la partición. **Si no se especifica ningún valor predeterminado, se devuelve NULL.**
- IGNORE NULLS: se ignoran los valores NULL del conjunto de datos al calcular el primer valor en una partición.
- RESPECT NULLS: se respetan los valores NULL del conjunto de datos al calcular el primer valor en una partición. **RESPECT NULLS es el comportamiento predeterminado cuando no se especifica la opción NULLS.**

LEAD()

Ejemplo

BusinessEntityID	SalesYear	CurrentQuota	NextQuota
275	2005	367000.00	556000.00
275	2005	556000.00	502000.00
275	2006	502000.00	550000.00
275	2006	550000.00	1429000.00
275	2006	1429000.00	1324000.00
275	2006	1324000.00	0.00

```
LEAD(SalesQuota, 1,0) OVER (ORDER BY YEAR(QuotaDate)) AS
NextQuota
```

LAG

Introducción

LAG proporciona acceso a una fila en un desplazamiento físico especificado que hay antes de la fila actual. Use esta función analítica en una instrucción SELECT para comparar valores de la fila actual con valores de una fila anterior.

LAG

Sintaxis

LAG (scalar_expression [, offset] [, default]) [IGNORE NULLS | RESPECT NULLS]
OVER ([partition_by_clause] order_by_clause)

Argumentos

- Scalar_expresión: Es el valor que se va a devolver en función del desplazamiento especificado.
- Offset: El número de filas hacia atras de la fila actual de la que se va a obtener un valor. **Si no se especifica, el valor predeterminado es 1.**
- Default: Valor que se devuelve cuando *offset* está fuera del ámbito de la partición. **Si no se especifica ningún valor predeterminado, se devuelve NULL.**
- IGNORE NULLS: se ignoran los valores NULL del conjunto de datos al calcular el primer valor en una partición.
- RESPECT NULLS: se respetan los valores NULL del conjunto de datos al calcular el primer valor en una partición. **RESPECT NULLS es el comportamiento predeterminado cuando no se especifica la opción NULLS.**

LAG

Ejemplo

BusinessEntityID	SalesYear	CurrentQuota	PreviousQuota
275	2005	367000.00	0.00
275	2005	556000.00	367000.00
275	2006	502000.00	556000.00
275	2006	550000.00	502000.00
275	2006	1429000.00	550000.00
275	2006	1324000.00	1429000.00

```
LAG(SalesQuota, 1,0) OVER (ORDER BY YEAR(QuotaDate)) AS
PreviousQuota
```

FIRST_VALUE

Introducción

Devuelve el primer valor de un conjunto ordenado de valores.

Sintaxis

```
FIRST_VALUE ( [scalar_expression ] )  [ IGNORE NULLS | RESPECT NULLS ]
   OVER ( [ partition_by_clause ] order_by_clause [ rows_range_clause ] )
```

FIRST_VALUE

Argumentos

- Scalar_expresión: Es el valor que se va a devolver(escalar). *scalar_expression* no puede ser una función analítica
- IGNORE NULLS: se ignoran los valores NULL del conjunto de datos al calcular el primer valor en una partición.
- RESPECT NULLS: se respetan los valores NULL del conjunto de datos al calcular el primer valor en una partición. **RESPECT NULLS es el comportamiento predeterminado cuando no se especifica la opción NULLS.**

FIRST_VALUE

Ejemplo

JobTitle	LastName	VacationHours	FewestVacationHours
Accountant	Moreland	58	Moreland
Accountant	Seamans	59	Moreland
Accounts Manager	Liu	57	Liu
Accounts Payable Specialist	Tomic	63	Tomic
Accounts Payable Specialist	Sheperdigian	64	Tomic
Accounts Receivable Specialist	Poe	60	Poe
Accounts Receivable Specialist	Spoon	61	Poe
Accounts Receivable Specialist	Walton	62	Poe

```
FIRST_VALUE(LastName) OVER (PARTITION BY JobTitle ORDER BY
VacationHours ASC ROWS UNBOUNDED PRECEDING ) AS
FewestVacationHours
```

LAST_VALUE

Introducción

Devuelve el último valor de un conjunto ordenado de valores.

Sintaxis

LAST_VALUE ([scalar_expression]) [IGNORE NULLS | RESPECT NULLS] OVER ([partition_by_clause] order_by_clause [rows_range_clause])

LAST_VALUE

Argumentos

- Scalar_expresión: Es el valor que se va a devolver(escalar). *scalar_expression* no puede ser una función analítica
- IGNORE NULLS: se ignoran los valores NULL del conjunto de datos al calcular el primer valor en una partición.
- RESPECT NULLS: se respetan los valores NULL del conjunto de datos al calcular el primer valor en una partición. **RESPECT NULLS es el comportamiento predeterminado cuando no se especifica la opción NULLS.**

LAST_VALUE

Ejemplo

Department	LastName	Rate	HireDate	LastValue
Document Control	Chai	10.25	2003-02-23	2003-03-13
Document Control	Berge	10.25	2003-03-13	2003-03-13
Document Control	Norred	16.8269	2003-04-07	2003-01-17
Document Control	Kharatishvili	16.8269	2003-01-17	2003-01-17
Document Control	Arifin	17.7885	2003-02-05	2003-02-05
Information Services	Berg	27.4038	2003-03-20	2003-01-24
Information Services	Meyyappan	27.4038	2003-03-07	2003-01-24
Information Services	Bacon	27.4038	2003-02-12	2003-01-24
Information Services	Bueno	27.4038	2003-01-24	2003-01-24
Information Services	Sharma	32.4519	2003-01-05	2003-03-27
Information Services	Connelly	32.4519	2003-03-27	2003-03-27
Information Services	Ajenstat	38.4615	2003-02-18	2003-02-23
Information Services	Wilson	38.4615	2003-02-23	2003-02-23
Information Services	Conroy	39.6635	2003-03-08	2003-03-08
Information Services	Trenary	50.4808	2003-01-12	2003-01-12

```
LAST_VALUE(HireDate) OVER ( PARTITION BY Department ORDER
BY Rate ) AS LastValue
```

¿Dudas?

Universidad Nacional de La Matanza

DIIT
Departamento de Ingeniería e Investigaciones Tecnológicas