SQL Server

rortegag

Navegación

Mostrar bases de datos:
```
SELECT name FROM sys.databases
GO
sp_databases
GO
```
Mostrar tablas:
```
SELECT name FROM sys.tables
GO
sp_tables
GO
```
Matar proceso:
```
KILL process_id
GO
```
Mostrar las estructura de una tabla:
```
SELECT * FROM information_schema.columns
WHERE table_name = 'nombre_tabla'
GO
exec sp_columns NombreTabla
GO
```
Mostrar lista de procesos:
```
SELECT * FROM master..sysprocesses
GO
sp_who
GO
```

Ediciones de SQL Server

SQL Server Developer
para su uso en el desarrollo y prueba de bases de datos.

SQL Server Express
para bases de datos pequeñas con un tamaño de hasta 10 GB de capacidad de almacenamiento en disco.

SQL Server Standard
para capacidades de programación ricas, innovaciones de seguridad y un rendimiento rápido para las aplicaciones de nivel medio y los mercados de datos.

SQL Server Enterprise
para capacidades de misión crítica para lograr una escala, seguridad, alta disponibilidad y un rendimiento líder sin precedentes para su base de datos de nivel 1, inteligencia empresarial y cargas de trabajo de análisis avanzado.

Servicios y herramienta de SQL Server

Para la gestión de datos:
SQL Server Integration Services (SSIS)
SQL Server Data Quality Services
SQL Server Master Data Services

Para el análisis de datos:
SQL Server Analysis Services (SSAS)
SQL Server Reporting Services (SSRS)

Select

Consultas básicas con SELECT
```
SELECT * FROM db.tabla
SELECT campo1 FROM db.tabla
SELECT campo1, campo2, ... FROM db.tabla
```

Select Distinct

DISTINCT una columna
```
SELECT DISTINCT campo FROM bd.tabla
GO
```
DISTINCT varias columnas
```
SELECT DISTINCT campo1, campo2
FROM bd.tabla
GO
```

Select Top

Usar TOP con un valor constante
```
SELECT TOP 10 campo2 FROM bd.tabla
ORDER BY campo2 DESC
GO
```
Usar TOP para un porcentaje de filas
```
SELECT TOP 1 PERCENT campo2
FROM bd.tabla ORDER BY campo2 DESC
GO
```
Uso de TOP WITH TIES para incluir las filas que coinciden con los valores de la última fila
```
SELECT TOP 3 WITH TIES campo1, campo2
FROM bd.tabla ORDER BY campo2 DESC
GO
```

Offset Fetch

Saltar los 10 primeros resultados y devolver el resto
```
SELECT campo1, campo2 FROM bd.tabla ORDER BY campo1, campo2
OFFSET 10 ROWS
GO
```
Saltar los 10 primeros resultados y seleccionar los 10 siguientes
```
SELECT campo1, campo2 FROM bd.tabla ORDER BY campo1, campo2
OFFSET 10 ROWS FETCH NEXT 10 ROWS ONLY
GO
```
Obtener los 10 resultados con más valor
```
SELECT campo1, campo2 FROM bd.tabla
ORDER BY campo2 DESC, campo1 OFFSET 0 ROWS
FETCH FIRST 10 ROWS ONLY
GO
```

Order By

La única manera de garantizar que las filas del conjunto de resultados estén ordenadas es utilizar la cláusula ORDER BY
```
SELECT * FROM db.tabla ORDER BY campo1 | expresión [ASC | DESC ]
GO
```

Where

Para obtener las filas de la tabla que cumplen una o varias condiciones, se utiliza la cláusula WHERE
```
SELECT * FROM db.tabla WHERE condición
GO

SELECT * FROM db.tabla1 WHERE campo1 IN
(SELECT * FROM db.tabla2 WHERE campo2 = “CA”)
GO
```

Para más información: https://learn.microsoft.com/es-es/sql/sql-server/?view=sql-server-ver15

SQL SERVER

rortegag

Null

Encontrar resultados en el campo4 que su valor es nulo

```
SELECT campo1, campo2, campo3, campo4
FROM bd.tabla WHERE campo4 = NULL
GO
```

Para comprobar si un valor es nulo o no, siempre se utiliza el operador IS NULL

```
SELECT campo1, campo2, campo3, campo4
FROM bd.tabla WHERE campo4 IS NULL
GO
```

Para comprobar si un valor no es nulo, utilizar el operador IS NOT NULL.

```
SELECT campo1, campo2, campo3, campo4
FROM bd.tabla WHERE campo4 IS NOT NULL
ORDER BY campo1, campo2
GO
```

And

Uso del operador AND

```
SELECT * FROM db.tabla WHERE campo1 = 1
AND campo2 > 400 ORDER BY campo2 DESC
GO
```

Or

Uso del operador OR

```
SELECT campo1, campo2 FROM bd.tabla
WHERE campo2 < 200 OR campo2 > 6000
ORDER BY campo2
GO
```

In

Uso de IN con una lista de valores

```
SELECT campo1, campo2 FROM bd.tabla
WHERE campo2 IN (89.99, 109.99, 159.99)
ORDER BY campo2
GO
```

Between

Uso de BETWEEN con números

```
SELECT campo1, campo2 FROM bd.tabla
WHERE campo3 BETWEEN 49.99 AND 99.99
ORDER BY campo2
GO
```

Uso de NOT BETWEEN con fechas

```
SELECT campo1, campo2 FROM bd.tabla
WHERE campo2 NOT BETWEEN '20170115'
AND '20170117' ORDER BY campo2
GO
```

Group By

La cláusula GROUP BY permite organizar las filas de una consulta en grupos.

```
SELECT * FROM db.tabla GROUP BY campo1
GO
```

Like

% (porcentaje) representa que contenga cualquier carácter

```
SELECT campo1, campo2, campo3 FROM bd.tabla WHERE campo3
LIKE 'z%' ORDER BY campo2
GO
```

_ (guión bajo) representa un solo carácter

```
SELECT campo2 FROM bd.tabla WHERE campo3 LIKE '_u%'
ORDER BY campo2
GO
```

[lista de caracteres] representa un único carácter que debe ser uno de los especificados en la lista

```
SELECT campo1, campo2, campo3 FROM bd.tabla WHERE campo3
LIKE '[YZ]%' ORDER BY campo2
GO
```

[carácter-carácter] representan un único carácter que debe estar dentro de un rango especificado

```
SELECT campo1, campo2, campo3 FROM bd.tabla WHERE campo3
LIKE '[A-C]%' ORDER BY campo2
GO
```

[^lista de caracteres o rango] representan un solo carácter que no está en el rango o la lista de caracteres especificados

```
SELECT campo1, campo2, campo3 FROM db.tabla WHERE campo3
LIKE '[^A-X]%' ORDER BY campo2
GO
```

El operador NOT LIKE

```
SELECT * FROM db.tabla WHERE campo2 NOT LIKE 'A%'
GO
```

LIKE con ESCAPE especifica que el carácter ! es el carácter de escape.

```
SELECT campo1, campo2 FROM db.tabla WHERE campo1
LIKE '%30!%%' ESCAPE '!'
GO
```

Alias

Alias de columnas

```
SELECT campo1 + ' ' + campo2 FROM bd.tabla ORDER BY campo1
GO
```

Uso de AS

```
SELECT campo1 + ' ' + campo2 AS 'Nombre completo' FROM bd.tabla
ORDER BY campo1
GO
```

Uso de un alias en una cláusula ORDER BY

```
SELECT campo1 'Nombre' FROM bd.tabla ORDER BY 'Nombre'
GO
```

Alias para una tabla

```
SELECT t1.campo1, campo2, campo3, campo4 FROM bd.tabla1 t1
INNER JOIN bd.tabla2 t2 ON t2.campo4 = t1.campo1
GO
```

Having

La cláusula HAVING se utiliza a menudo con la cláusula GROUP BY para filtrar grupos basados en una lista de condiciones especificada.

```
SELECT * FROM db.tabla GROUP BY campo1 HAVING condición
GO
```

Para más información: https://learn.microsoft.com/es-es/sql/sql-server/?view=sql-server-ver15

SQL SERVER

rortegag

Joins

INNER JOIN

```
SELECT t1.campo1, t1.campo2,
t2.campo1, t2.campo2 FROM bd.tabla1 t1
INNER JOIN bd.tabla2 t2 ON t2.campo2 = t1.campo1
GO
```

LEFT JOIN

```
SELECT t1.campo1, t1.campo2,
t2.campo1, t2.campo2 FROM bd.tabla1 t1
LEFT JOIN bd.tabla2 t2 ON t2.campo2 = t1.campo1
GO
```

RIGHT JOIN

```
SELECT t1.campo1, t1.campo2,
t2.campo1, t2.campo2 FROM bd.tabla1 t1
RIGHT JOIN bd.tabla2 t2 ON t2.campo2 = t1.campo1
GO
```

FULL JOIN

```
SELECT t1.campo1, t1.campo2,
t2.campo1, t2.campo2 FROM bd.tabla1 t1
FULL JOIN bd.tabla2 t2 ON t2.campo2 = t1.campo1
GO
```

CROSS JOIN

```
SELECT campo1, campo2, campo3, 0 AS cantidad
FROM bd1.tabla1 CROSS JOIN bd2.tabla1
ORDER BY campo2, campo3
GO
```

Subconsultas

Una subconsulta es una consulta anidada dentro de otra sentencia como SELECT, INSERT, UPDATE o DELETE.

```
SELECT campo1, campo2, campo3 FROM db.tabla1
WHERE campo1 IN (SELECT campo1 FROM db.tabla2
WHERE campo2 = 'España') ORDER BY campo3 DESC
GO
```

Subconsulta correlacionada

Una subconsulta correlacionada es una subconsulta que utiliza los valores de la consulta externa. En otras palabras, la subconsulta correlacionada depende de la consulta externa para sus valores.

```
SELECT campo2, campo3, campo5 FROM db.tabla s1
WHERE campo3 IN (SELECT MAX (s2.campo3)
FROM db.tabla s2 WHERE s2.campo1 = s1.campo1
GROUP BY s2.campo1) ORDER BY campo1, campo2
GO
```

Exists

El operador EXISTS es un operador lógico que permite comprobar si una subconsulta devuelve alguna fila. El operador EXISTS devuelve TRUE si la subconsulta devuelve una o más filas

```
SELECT * FROM db.tabla WHERE
EXISTS (SELECT NULL) ORDER BY campo1
GO
```

Grouping Sets

Los conjuntos de agrupación definen varios conjuntos de agrupación en la misma consulta

```
SELECT campo1, campo2, COUNT(campo3) FROM db.tabla
GROUP BY GROUPING SETS ((campo1, campo2), (campo1),
(campo2), ())
GO
```

La función GROUPING indica si una columna especificada en una cláusula GROUP BY está agregada o no. Devuelve 1 si está agregada o 0 si no está agregada en el conjunto de resultados

```
SELECT GROUPING(campo1), GROUPING(campo2), campo1,
campo2 FROM bd.tabla GROUP BY
GROUPING SETS ((campo1, categor2), (campo1), (campo2),())
ORDER BY campo1, campo2
GO
```

Cube

El CUBE es una subcláusula de la cláusula GROUP BY que permite generar múltiples conjuntos de agrupación

```
SELECT d1, d2, d3, funcion(c4) FROM db.tabla GROUP BY
CUBE(d1, d2, d3)
GO
```

Rollup

ROLLUP es una subcláusula de la cláusula GROUP BY que proporciona una forma abreviada de definir múltiples conjuntos de agrupación. A diferencia de la subcláusula CUBE, ROLLUP no crea todos los posibles conjuntos de agrupación basados en las columnas de dimensión; el CUBE hace un subconjunto de ellos

```
SELECT d1, d2, d3, funcion(c4) FROM db.tabla GROUP BY
ROLLUP (d1, d2, d3)
GO
```

Any

El operador ANY es un operador lógico que compara un valor escalar con un conjunto de valores de una sola columna devueltos por una subconsulta

```
SELECT * FROM db1.tabla1 WHERE campo1 = ANY (
SELECT campo4 FROM db2.tabla1 WHERE campo3 >= 2)
ORDER BY campo2
GO
```

All

El operador ALL es un operador lógico que compara un valor escalar con una lista de valores de una sola columna devuelta por una subconsulta

```
SELECT * FROM db.tabla WHERE campo3 [< >] ALL (SELECT
AVG (campo3) avg FROM db.tabla GROUP BY campo5)
ORDER BY campo3 DESC
GO
```

Para más información: https://learn.microsoft.com/es-es/sql/sql-server/?view=sql-server-ver15

SQL SERVER

rortegag

Union

UNION es una de las operaciones de conjunto que permiten combinar los resultados de dos sentencias SELECT en un único conjunto de resultados que incluye todas las filas que pertenecen a las sentencias SELECT de la unión

```
SELECT campo1, campo2 FROM db.tabla1 UNION
SELECT campo1, campo2 FROM db.tabla2
GO

SELECT campo1, campo2 FROM db.tabla1 UNION ALL
SELECT campo1, campo2 FROM db.tabla2
GO
```

Insert

Uso de INSERT básico

```
INSERT INTO db.tabla (campo1, campo2, campo3, campo4)
VALUES (valor1, valor2, valor3, valor4)
GO
```

Uso de múltiples VALUES

```
INSERT INTO db.tabla (campo1, campo2, campo3, campo4)
VALUES (valor1, valor2. valor3, valor4),
(valor1, valor2. valor3, valor4),
(valor1, valor2. valor3, valor4),
GO
```

Insert Into Select

Para insertar datos de otras tablas en una tabla, se utiliza la sentencia INSERT INTO SELECT

```
INSERT INTO db.tabla1 (campo1, campo2, campo3, campo4)
SELECT campo1, campo2, campo3, campo4 FROM db.tabla2
ORDER BY campo10, campo11
GO
```

Update Join

Uso de UPDATE JOIN básico

```
UPDATE db.tabla1
SET db.tabla1.campo1 = t1.campo2 * t2.campo3
FROM db.tabla1 t1 INNER JOIN db.tabla2 t2
ON t1.campo5 = t2.campo5
GO
```

Delete

Uso de DELETE básico

```
DELETE [ TOP ( expresion ) [ PERCENT ] ]  FROM db.tabla
[WHERE condicion]
GO
```

Tipos de datos de cadenas de caracteres

CHAR (0 carácteres - 8000 carácteres)
VARCHAR (0 carácteres - 8000 carácteres)
VARCHAR (max) (0 carácteres - 2^31 carácteres)
TEXT (0 carácteres - 2,147,483,647 carácteres)

Tipos de datos numéricos aproximados

FLOAT(n) (−1.79E+308 - 1.79E+308)
REAL (−3.40E+38 - 3.40E+38)

MÁS ↓

Intersect

INTERSECT combina conjuntos de resultados de dos o más consultas y devuelve filas distintas que son producidas por ambas consultas

```
SELECT campo1 FROM db.tabla1 INTERSECT
SELECT campo1 FROM db.tabla2 ORDER BY campo1
```

Except

EXCEPT compara los conjuntos de resultados de dos consultas y devuelve las filas distintas de la primera consulta que no salen en la segunda.
En otras palabras, el EXCEPT resta el conjunto de resultados de una consulta de otra

```
SELECT campo1 FROM db1.tabla1 EXCEPT
SELECT campo4 FROM db2.tabla1
```

Merge

La sentencia MERGE que permite realizar tres acciones al mismo tiempo

```
MERGE tabla_objetivo USING tabla_origen
ON condicion
WHEN MATCHED THEN declaración_update
WHEN NOT MATCHED THEN declaración_insert
WHEN NOT MATCHED BY SOURCE THEN DELETE
GO
```

Pivot

El operador PIVOT gira una expresión de valores de la tabla. Convierte los valores únicos de una columna en múltiples columnas en la salida y realiza agregaciones en cualquier valor de columna restante

```
SELECT * FROM (SELECT campo1, campo2, campo3
FROM db.tabla1 t1 INNER JOIN db.tabla2 t2
ON t1.campo1 = t2.campo1) p
PIVOT(COUNT(campo1) FOR campo2 IN (
[Grupo1], [Grupo2], [Grupo3], ...)) AS tabla_pivot
GO
```

Tipo de datos

Tipos de datos numéricos exactos

BIGINT (−2^63 - 2^63−1)
INT (−2^31 - 2^31−1)
SMALLINT (−2^15 - 2^15)
TINYINT (0 - 255)
BIT (0 - 1)
DECIMAL (−10^38+1 - 10^381−1)
NUMERIC (−10^38+1 - 10^381−1)
MONEY (−922,337, 203, 685,477.5808 - +922,337, 203, 685,477.5807)
SMALLMONEY (−214,478.3648 - +214,478.3647)

Tipos de datos de cadenas de caracteres Unicode

NCHAR (0 carácteres - 4000 carácteres)
NVARCHAR (0 carácteres - 4000 carácteres)
NTEXT (0 carácteres - 1,073,741,823 carácteres)

Para más información: https://learn.microsoft.com/es-es/sql/sql-server/?view=sql-server-ver15

SQL SERVER

rortegag

Tipo de datos

Tipos de datos de fecha y hora

DATETIME (1753-01-01 - 9999-12-31)
SMALLDATETIME (1900-01-01 - 2079-06-06)
DATE (0001-01-01 - 9999-12-31)
TIME (00:00:00.0000000 - 23:59:59.9999999)
DATETIMEOFFSET (0001-01-01 - 9999-12-31)
DATETIME2 (0001-01-01 - 9999-12-31)

Tipos de datos binarios

BINARY (0 bytes - 8000 bytes)
VARBINARY (0 bytes - 8000 bytes)
IMAGE (0 bytes - 2,147,483,647 bytes)

Otros tipos de datos

CURSOR - para las variables o el parámetro OUTPUT del procedimiento almacenado que contiene una referencia a un cursor.
ROWVERSION - exponer números binarios únicos generados automáticamente dentro de una base de datos.
HIERARCHYID - representan una posición de árbol en una jerarquía de árboles.
UNIQUEIDENTIFIER - 16-byte GUID.
SQL_VARIANT - almacenar valores de otros tipos de datos.
XML - almacenar datos XML en una columna, o una variable de tipo XML.
Tipo Spatial Geometry - representar los datos en un sistema de coordenadas plano.
Tipo Spatial Geography - almacenar datos elipsoidales (tierra redonda), como las coordenadas de latitud y longitud del GPS.
TABLE - almacenar temporalmente un conjunto de resultados para procesarlos posteriormente.

Columnas computadas

```
ALTER TABLE nombre_tabla ADD campo_com
AS (campo1 + ' ' + campo2)
GO
```

Renombrar tabla

```
EXEC sp_rename 'vieja_nombre_tabla',
'nuevo_nombre_tabla'
GO
```

Eliminar tabla

```
DROP TABLE bd.es.nombre_tabla
GO
```

Eliminar tabla si existe

```
DROP TABLE IF EXISTS nombre_tabla
GO
```

Crear tabla temporal

```
CREATE TABLE ##nombre_tabla
(campo1 VARCHAR(MAX), campo2 DEC(10,2))
GO
```

Crear sinónimo

```
CREATE SYNONYM [nombre_esquema.]
nombre_sinónimo FOR object
GO
```

Definición de datos

Crear base de datos

```
CREATE DATABASE TestBD
GO
```

Borrar base de datos

```
DROP DATABASE TestBD
GO
```

Borrar base de datos si existe

```
DROP DATABASE IF EXISTS TestBD
GO
```

Crear esquema

```
CREATE SCHEMA nombre_esquema
[AUTHORIZATION nombre_propietario]
GO
```

Modificar esquema

```
ALTER SCHEMA nombre_esquema
TRANSFER [tipo_entidad ::] nombre_seguro
GO
```

Eliminar esquema

```
DROP SCHEMA nombre_esquema
GO
```

Eliminar esquema si existe

```
DROP SCHEMA IF EXISTS nombre_esquema
GO
```

Crear tabla

```
CREATE TABLE [nombre_basededatos.][nombre_esquema.]
nombre_tabla (campo_pk tipo_dato PRIMARY KEY,
campo1 tipo_date NOT NULL, campo2 tipo_dato, ..,
restricciones_tabla)
GO
```

Columna IDENTITY

```
CREATE TABLE bd.tabla (campo_id INT IDENTITY(1,1)
PRIMARY KEY, campo1 VARCHAR(50) NOT NULL)
GO
```

Crear secuencia

```
CREATE SEQUENCE [nombre_esquema.] nombre_secuencia
[AS tipo_entero] [START WITH valor_inicial]
[INCREMENT BY valor_incrementado]
[{MINVALUE [valor_mínimo]} | {NO MINVALUE} ]
[{MAXVALUE [valor_máximo]} | {NO MAXVALUE} ]
[CYCLE | {NO CYCLE} ]  [{CACHE [tamaño_caché]} |
{NO CACHE}]
GO
```

Añadir columna

```
ALTER TABLE nombre_tabla ADD nombre_columna
tipo_dato restricción_columna
GO
```

Modificar columna

```
ALTER TABLE nombre_tabla
ALTER COLUMN nombre_columna nuevo_tipo_dato(tamaño)
GO
```

Eliminar columna

```
ALTER TABLE nombre_tabla DROP COLUMN nombre_columna
GO
```

Truncar tabla

```
TRUNCATE TABLE nombre_tabla
GO
```

Para más información: https://learn.microsoft.com/es-es/sql/sql-server/?view=sql-server-ver15