Transact-SQL & simple queries

UNIT 4. Part 1

Objetives

- To know SQL features and specially Transact-SQL used by SQL Server.
- Start with the SELECT sentence.

Basics

- SQL (Structured Query Language)
- Inicially used only to query a relational database-→ To define, control and manage data in relational/object-relational databases, even to write scripts.

Basics

- SQL is a universal language, used in all RDBMS.
  - DB2
  - SQL Server
  - Oracle
  - MySql
  - Access
- Has a standard definition, and each RDBMS has developed its own product.

Basics

- Includes programming sentences such as
  - Conditional structures
  - Loops
  - Output formatting
  - Object Oriented features
- But not GUIs Graphical User Interfaces.

Basics

- Executable:
  - From the graphical tool
  - In console mode - from the command line
  - Embedded in programs written in other languages

Basics

- SQL sentences are classified in:
  - DDL (Data Description Language)
  - DCL (Data Control Language)
  - DML (Data Manipulation Language)

Basics

- **DDL**, used to define the database **structure**:
  - Create databases
  - Create, modify o eliminate tables.
  - Create indexes, validation rules, etc.
- Usually used by the DB administrator BD.
- The closest to the physical level → the least standard.

Basics

- **DCL**, used to control:
  - Privileges granting (GRANT/REVOKE).
  - Transactional features (COMMIT/ROLLBACK).
- Used by the DBA and programmers.
- Varies very little.

Basics

- **DML**, used to manage data:
  - Insert rows in a table.
  - Update table rows.
  - Delete rows from a table.
  - Retrieve data from tables.
- Used by all users, programmers and DBA.
- Varies a little from a system to another.

Transact-SQL features

- The SQL version for SQL Server.
- Also includes:
  - Datatypes definition.
  - Variables Definition.
  - Conditional structures
  - Loop.
  - Exceptions control.
  - User Defined Functions.
- Not include:
  - GUIs.
  - Executable applications.

Transact-SQL features

- Powerful instructions.
- Closed to natural language (English).
- Follows a pattern:
  - The sentence begins with a verb which indicates what we want to be done.
  - Completed with the object on the action is done.
  - Followed by clauses (obligatory / optional) wich complete the sentence.
  - Finishes with semicolon ; (sometimes optional)

Transact-SQL features

- In DDL 3 verbs:
  - CREATE
  - DROP
  - ALTER
- On objects: TABLE, DATABASE, INDEX…
  - CREATE DATABASE mydb .....;
  - DROP TABLE mydb;

Transact-SQL features

- In DML 4 verbs:
  - INSERT
  - DELETE
  - UPDATE
  - SELECT

```
INSERT INTO mydb .....;

DELETE FROM mydb…;

SELECT id, name
  FROM Customers WHERE city='Valencia';
```

Identifiers format rules

- An identifier is used to reference an object.
- The rules for the format of regular identifiers depend on the database compatibility level.
- When the compatibility level is **90**, the following rules apply:

Identifiers format rules

- Must not be a Transact-SQL reserved word.
- Up to 128 characters with few exceptions.
- The first character must be one of the following:
  - A letter (caracteres latinos de ‘a’ a ‘z’ y de ‘A’ a ‘Z’…)
  - The underscore (_), at sign (@) or number sign (#).
- Do not use names that start with @@.
- Special characters and spaces are not allowed.
- Identifiers that do not comply with these rules must be delimited by brackets [].

Data types

- Numerics:
  - Int, bigint, smallint, tinyint
  - Bit
  - Decimal, numeric (default dec(18,0))
  - Money, smallmoney
  - Float, real
- Dates:
  - Datetime, smalldatetime

Data types

- Character Strings:
  - Char, varchar (default Char(1))
  - Nchar, nvarchar (default nchar(1))
  - Text, ntext (disused)
- Binary Strings :
  - Binary, varbinary (default binary(1))
  - Image (disused)
- Other
  - Cursor, uniqueidentifier, table …

Constants

- Represents a specific data value.
- Character string single quotation marks ´ ´
- Unicode strings N‘single quotation’
- Binary constants 0x prefix + hexadecimal
- Bit 0 or 1
- Datetime ‘single quotation marks’
- Numerics:
  - Not enclosed in quotation marks
  - Decimal point •
  - Float y Real by using scientific notation
  - Money € prefix (optional)

Expressions

- **Numeric operators**:
  - + (add), - (subtract), * (multiply), / (divide), % (modulo remainder of a division).
- **Bitwise (between integer values)**:
  - & (AND b a b), | (OR b a b), ^ (OR exclusive), ~ (NOT)
- **Comparison Operators**:
  - = , > , < , >= , <= , <> , != , !< , !>
- **Logical operators**:
  - ALL, AND, ANY, BETWEEN, EXISTS, IN, LIKE, NOT, OR, SOME.
- **String operator**:
  - + (string concatenation)

Functions

- Rowset Functions:
  - Return an object that can be used like table references in an SQL statement.
- Aggregate Functions:
  - Operate on a collection of values but return a single, summarizing value.
- Ranking Functions: (*de categoría*)
  - Return a ranking value for each row in a partition.
- Scalar functions:
  - Operate on a single value and then return a single value. Scalar functions can be used wherever an expression is valid.

Functions

- Depending on their operands datatype scalar functions are classified in:
  - Configuration Functions
  - **Conversion Functions**
  - **Date and Time Functions**
  - **Mathematical Functions**
  - Metada Functions
  - Security Functions
  - **String Functions**
  - System Functions
  - System Statistical Functions
  - Text and Image Functions

Other elements

- Variables
- IF … ELSE, CASE, WHILE…
- TRY… CATCH
- Messages
- Comments
- USE
- GO

DML. Simple queries

- SELECT is the most complex and powerful sentence:
  - Retrieves data from one o more tables,
  - Selects rows and columns
  - Obtains summaries from stored data
- Returns a logical table.
- We will study it over several units.

DML. Simple queries

- We will begin with simple queries based on one table and limit our study to this sintax:

SELECT
[ALL|DISTINCT]
[TOP *expression* [PERCENT] [WITH TIES]]
<select_list>
FROM <source>
[WHERE <search_condition>]
[ORDER BY {{column_*expression*|column_*position*}
[ASC|DESC]} [,...*n*]];

Source

FROM specifies where are the data to use:

<source>**::=**

table_na | *view_na* [[AS] *alias_na*]

- *table_na* represents a table name
- *nb_vista* represents a view name
- Alias_name is an alias that can be used either for convenience or to distinguish a table or view in a self-join or subquery. The table name cannot be used if an alias is defined.
- AS is optional and does not add any functionality.

Source

```
SELECT ...
FROM table1

SELECT ...
FROM table1 t1

SELECT ...
FROM table1 AS t1
```

Select list

Indicates the columns displayed in the result table.

<select_list> ::=
{ *
| {table_na|*view_na*|*alias_table*}.*
| { [{*table_na*|*view_na*|*alias_table*}.]
{*column_na*|$IDENTITY|$ROWGUID}
|<expression>
} [[AS] *alias_column*]
| *alias_column* = <expression>
} [,...*n*]

Select list

<expression> can be any valid expression:

- A simple expression:
  - A function reference
  - A local variable
  - A constant
  - A column **from** the query **source**
- A scalar subquery
- A complex expression generated from multiple expressions.
- A variable assigment like *@variable_local* = expression.

Select list

- To indicate a column, we use its name or we can use its qualified name.
- The qualified name is compulsory when the column name is duplicated in the query source (two tables have two columns with the same name, or the same table combines with itself).
- Table_name.column_name

```
SELECT midfac, productid, description, products.price
FROM products;
```

Select list

- **Alias_column**
  The header name of the column.
  Cannot be used in every clause (not in WHERE)
  Sintax:

  Column_name [AS] alias_column

  EX:

```
SELECT custid, name AS clientname
FROM customers;

SELECT custid, name AS [client name]
FROM customers;
```

Select list

- **Calculated columns**

  Its value is calculated from an expression.

  Ex:

```
SELECT city, zone, (sales-aim) AS superavit
FROM offices;

SELECT manid,productid,description,(stock*price) AS
  value FROM products;

SELECT name, MONTH(contract) AS [contract month],
  YEAR(contract) AS [contract year]
FROM  employees;

SELECT  ofnb, 'has sales… ' AS [ ], sales
FROM offices;
```

Select list

- *alias_column* **=** <expression>

Another way to indicate a calculated column and its alias name.

SELECT ofnb, superavit = sales-aim

SELECT ofnb, sales-aim AS superavit

Both produce the same result.

Select list

- Use of *

It is interpreted as *All the columns*

```
SELECT *
FROM offices;

SELECT *, (sales-aim) AS superavit
FROM offices;

SELECT offices.*
FROM offices;
```

Select list

- The keyword $IDENTITY is interpreted as "*the column which has the IDENTITY property*"

```
SELECT $IDENTITY, name, surname
FROM users;

SELECT userid, name, surname
FROM users;
```

Both produce the same result.

Select list

- The keyword $ROWGUID is interpreted as "*the column which has the* ROWGUID *property*"

Like $IDENTITY.