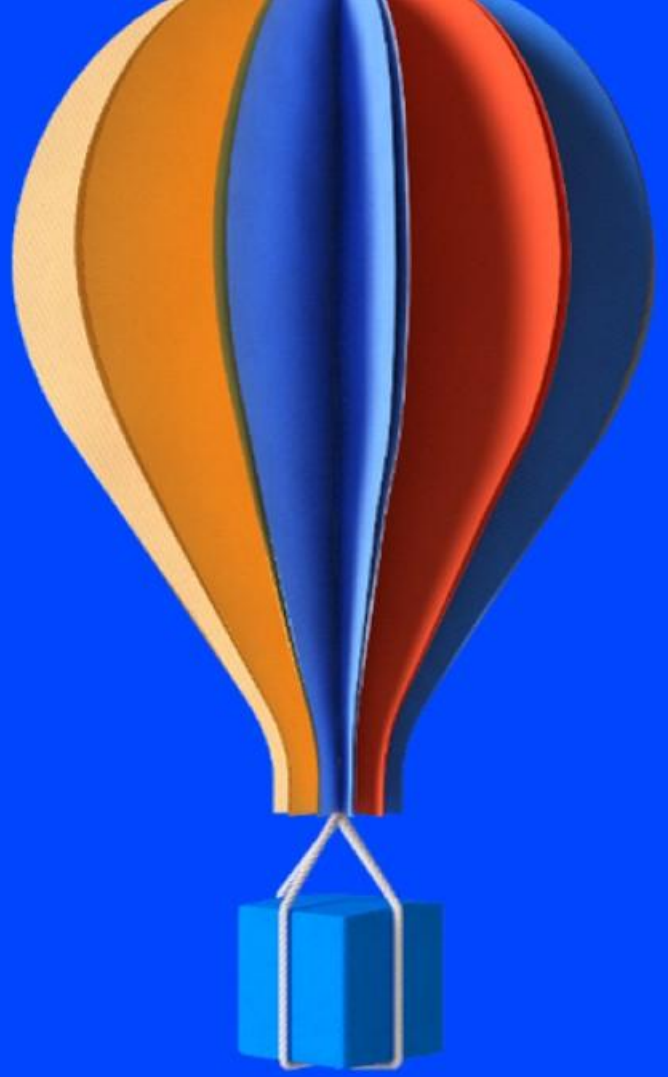
Data Analyst

Bases de Dados SQL



cegid Academy

Index

04 Selecting Data

Chapters

Data Types

A data type is an attribute that specifies the type of data that the object can hold

- In SQL Server, each column, local variable, expression, and parameter has a related data type
- Data types categories:
  - Exact numerics
  - Approximate numerics
  - Date and time
  - Character strings
  - Unicode character strings
  - Binary strings
  - Other data types

See resume table

Data Conversion

- Need for data types conversion
  - When data from one object is moved to, compared with, or combined with data from another object, the data may have to be converted from the data type of one object to the data type of the other
  - When data from a Transact-SQL result column, return code, or output parameter is moved into a program variable, the data must be converted from the SQL Server system data type to the data type of the variable
- Types of conversions
  - Implicit conversions are not visible to the user, since SQL Server automatically converts the data from one data type to another
  - Explicit conversions use the CAST or CONVERT functions

SELECT

- Statement used to extract data from tables or views
- A simple SELECT uses one table but is possible to combine several tables using JOINs and INTERSECTs
- To build a correct SELECT you must provide:
  - Tables to retrieve data from (FROM)
  - Columns to show
  - Conditions that data must comply (optional WHERE)
  - Output order (optional ORDER BY)

Logical processing order

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

```
5  SELECT select_list [INTO new_table]
1  [FROM table_source]
2  [WHERE search_condition]
3  [GROUP BY group_by_expression]
4  [HAVING search_condition]
6  [ORDER BY order_expression [ASC|DESC]]
```

Operators

- Arithmetic Operators
- Logical Operators
- Comparison Operators

Arithmetic Operators

Operator	Description	Example
+	Addition	5 + 2 = 7
-	Subtraction	5 - 2 = 3
*	Multiplication	5 * 2 = 10
/	Division	5 / 2 = 2.5
%	Modulo (remainder)	5 % 2 = 1

Logical Operators

Operator	Description	Example
AND	Returns TRUE if both conditions are TRUE	WHERE age > 18 AND city = 'NY'
OR	Returns TRUE if at least one condition is TRUE	WHERE age < 18 OR city = 'LA'
NOT	Reverses the result of a condition	WHERE NOT (age > 18)
ALL	TRUE if all subquery values meet the condition	WHERE salary > ALL (SELECT salary FROM employees)
ANY	TRUE if any subquery value meets the condition	WHERE salary > ANY (SELECT salary FROM employees)
EXISTS	TRUE if a subquery returns any rows	WHERE EXISTS (SELECT * FROM orders WHERE customer_id = c.id)

Comparison Operators

Operator	Description	Example
=	Equal to	a = b
<>	Not equal to	a <> b
!=	Not equal to (alternative)	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b
BETWEEN	Within a range	a BETWEEN 1 AND 10
LIKE	Pattern matching	name LIKE 'A%'
IN	Match any value in a list	a IN (1,2,3)
IS NULL	Is NULL	a IS NULL
IS NOT NULL	Is not NULL	a IS NOT NULL

SELECT summary

Clause / Command	Purpose
SELECT	Retrieves data from one or more tables or views
SELECT INTO	Creates a new table and inserts the result of a SELECT query into it
WHERE	Filters rows based on a specified condition before grouping or selecting
GROUP BY	Groups rows that have the same values in specified columns
HAVING	Filters groups created by GROUP BY based on a condition
ORDER BY	Sorts the result set by one or more columns (ascending or descending)

SELECT

SELECT select_list
FROM table_source

Syntax

```
1  -- SELECT simples
2  SELECT [DepartmentID], [Name], [GroupName]
3    FROM [AdventureWorks2012].[HumanResources].[Department];
```

100 %

Results | Messages

	DepartmentID	Name	GroupName
1	1	Engineering	Research and Development
2	2	Tool Design	Research and Development
3	3	Sales	Sales and Marketing
4	4	Marketing	Sales and Marketing
5	5	Purchasing	Inventory Management
6	6	Research and Development	Research and Development
7	7	Production	Manufacturing
8	8	Production Control	Manufacturing
9	9	Human Resources	Executive General and Administration
10	10	Finance	Executive General and Administration
11	11	Information Services	Executive General and Administration
12	12	Document Control	Quality Assurance
13	13	Quality Assurance	Quality Assurance
14	14	Facilities and Maintenance	Executive General and Administration
15	15	Shipping and Receiving	Inventory Management
16	16	Executive	Executive General and Administration

SELECT

```
SELECT select_list
INTO new_table
FROM table_source
```

```
5   -- SELECT + INTO
6   SELECT [DepartmentID], [Name], [GroupName] INTO [AdventureWorks2012].[HumanResources].[Department_backup]
7   FROM [AdventureWorks2012].[HumanResources].[Department];
8
```

100 %

Messages

```
(16 row(s) affected)
```

SELECT

```
SELECT select_list
FROM table_source
WHERE search_condition
```

```
 9  -- SELECT ... WHERE 1
10 SELECT [DepartmentID], [Name], [GroupName]
11  FROM [AdventureWorks2012].[HumanResources].[Department]
12  WHERE Name = 'Production';
13
14  -- SELECT ... WHERE 2
15 SELECT [DepartmentID], [Name], [GroupName]
16  FROM [AdventureWorks2012].[HumanResources].[Department]
17  WHERE Name LIKE 'Production%';
```

100 %

Results | Messages

	Departmen...	Name	GroupName
1	7	Production	Manufacturing

	Departmen...	Name	GroupName
1	7	Production	Manufacturing
2	8	Production Control	Manufacturing

Predicate LIKE

Find strings similar or not exactly equal to
Uses wildcards: % and _

Wildcards

%	more than 1 character
_	1 character in that position

SELECT

SELECT select_list
FROM table_source
GROUP BY group_by_expression

```
19  -- SELECT ... GROUP BY 1 --> ERRO PORQUE QUANDO SE AGRUPA TEM DE SE AGREGAR VALORES
20  SELECT [DepartmentID], [Name], [GroupName]
21  FROM [AdventureWorks2012].[HumanResources].[Department]
22  GROUP BY [GroupName];
23
```

100 %

Messages

```
Msg 8120, Level 16, State 1, Line 2
Column 'AdventureWorks2012.HumanResources.Department.DepartmentID' is invalid in the select list becaus
```

```
24  -- SELECT ... GROUP BY 2
25  SELECT [GroupName], [Name]
26  FROM [AdventureWorks2012].[HumanResources].[Department]
27  GROUP BY [GroupName], [Name];
28
```

100 %

Results | Messages

	GroupName	Name
1	Research and Development	Engineering
2	Research and Development	Tool Design
3	Sales and Marketing	Sales
4	Sales and Marketing	Marketing
5	Inventory Management	Purchasing
6	Research and Development	Research and Development

```
29  -- SELECT ... GROUP BY 3
30  SELECT [GroupName], COUNT([GroupName]) AS Contagem
31  FROM [AdventureWorks2012].[HumanResources].[Department]
32  GROUP BY [GroupName];
33
```

100 %

Results | Messages

	GroupName	Contagem
1	Executive General and Administration	5
2	Inventory Management	2
3	Manufacturing	2
4	Quality Assurance	2
5	Research and Development	3
6	Sales and Marketing	2

SELECT

```
SELECT select_list
FROM table_source
GROUP BY group_by_expression
HAVING search_condition
```

```
34  -- SELECT ... GROUP BY ... HAVING
35  SELECT [GroupName], COUNT([GroupName]) AS Contagem
36  FROM [AdventureWorks2012].[HumanResources].[Department]
37  GROUP BY [GroupName]
38  HAVING COUNT([GroupName]) > 2;
```

100 %

Results | Messages

	GroupName	Contagem
1	Executive General and Administration	5
2	Research and Development	3

SELECT

```
SELECT select_list
FROM table_source
ORDER BY order_expression [ASC|DESC]
```

```
40  -- SELECT ... ORDER BY v1
41  SELECT [DepartmentID], [Name], [GroupName]
42  FROM [AdventureWorks2012].[HumanResources].[Depart
43  ORDER BY [Name];
44
45  -- SELECT ... ORDER BY v2
46  SELECT [DepartmentID], [Name], [GroupName]
47  FROM [AdventureWorks2012].[HumanResources].[Depart
48  ORDER BY [Name] DESC;
```

100 %

Results | Messages

	DepartmentID	Name	GroupName
1	12	Document Control	Quality Assurance
2	1	Engineering	Research and Developmen
3	16	Executive	Executive General and Adm

	DepartmentID	Name	GroupName
1	2	Tool Design	Research and Developmen
2	15	Shipping and Receiving	Inventory Management
3	3	Sales	Sales and Marketing

```
50  -- SELECT ... ORDER BY v3
51  SELECT [GroupName], COUNT([GroupName]) AS Contagem
52  FROM [AdventureWorks2012].[HumanResources].[Departm
53  GROUP BY [GroupName]
54  ORDER BY COUNT([GroupName]) DESC;
55
56  -- SELECT ... ORDER BY v4
57  SELECT [GroupName], COUNT([GroupName]) AS Contagem
58  FROM [AdventureWorks2012].[HumanResources].[Departm
59  GROUP BY [GroupName]
60  ORDER BY COUNT([GroupName]) DESC, [GroupName] ASC;
```

100 %

Results | Messages

	GroupName	Contagem
1	Executive General and Administration	5
2	Research and Development	3
3	Sales and Marketing	2
4	Inventory Management	2
5	Manufacturing	2
6	Quality Assurance	2

	GroupName	Contagem
1	Executive General and Administration	5
2	Research and Development	3
3	Inventory Management	2
4	Manufacturing	2
5	Quality Assurance	2
6	Sales and Marketing	2

05

Working with Multiple Tables

Chapters

1. Join Types
2. CROSS JOIN
3. INNER JOIN
4. OUTER JOIN
5. UNION
6. EXCEPT
7. INTERSECT

The JOIN clause allows you to combine related data from multiple table sources

JOIN type	Description	Includes rows from	Matched rows	Unmatched rows
CROSS JOIN	Returns the Cartesian product of the two tables (all combinations)	Every combination of Table A × Table B	No matching	All combinations
INNER JOIN	Returns rows when there is a match in both tables	Both Table A and Table B	Yes	No
LEFT [OUTER] JOIN	Returns all rows from the left table, and matched rows from the right table	All from Table A, matched from Table B	Yes	Left table only
RIGHT [OUTER] JOIN	Returns all rows from the right table, and matched rows from the left table	All from Table B, matched from Table A	Yes	Right table only
FULL [OUTER] JOIN	Returns rows when there is a match in one of the tables. Includes all unmatched rows from both	All from Table A and Table B	Yes	Both sides
SELF JOIN	A join of a table to itself. Often used with aliases	Same table (used twice with alias)	Condition	If using outer self join

Combine all possible combinations of two sets

- Cartesian Product

```
SELECT c.CategoryName, p.ProductName
FROM dbo.Categories AS c
CROSS JOIN dbo.Products AS p
```

Name
Davis
Funk
King

Product
Alice Mutton
Crab Meat
Ipoh Coffee

Name	Product
Davis	Alice Mutton
Davis	Crab Meat
Davis	Ipoh Coffee
Funk	Alice Mutton
Funk	Crab Meat
Funk	Ipoh Coffee
King	Alice Mutton
King	Crab Meat
King	Ipoh Coffee

List tables in FROM Clause separated by JOIN operator

- Table aliases preferred
- Table order does not matter

```
SELECT c.CategoryName, p.ProductName
FROM dbo.Categories AS c
INNER JOIN dbo.Products AS p
ON c.CategoryID = p.CategoryID


-- ON tabela1.pk = tabela2.fk
```

LEFT OUTER JOIN

- Return all rows from first table and only matches from second

```
SELECT c.ContactName, o.OrderID
FROM dbo.Customers AS c
LEFT OUTER JOIN dbo.Orders AS o
ON c.CustomerID = o.CustomerID
```

	ContactName	OrderID
498	Henriette Pfalzheim	10766
499	Henriette Pfalzheim	10833
500	Henriette Pfalzheim	10999
501	Henriette Pfalzheim	11020
502	Marie Bertrand	NULL

RIGHT OUTER JOIN

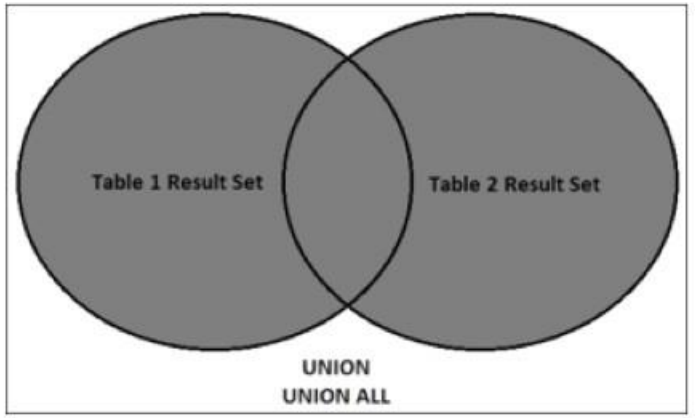
- Return all rows from second table and only matches from first

```
SELECT c.ContactName, o.OrderID
FROM dbo.Customers AS c
RIGHT OUTER JOIN dbo.Orders AS o
ON c.CustomerID = o.CustomerID
```

Used to combine sets from the same table or different tables

- All selected columns need to be of the same data type

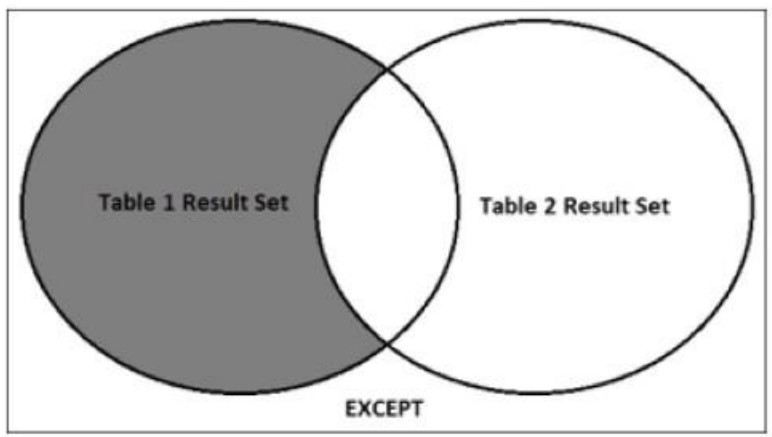
```
-- List the countries where we have customers OR suppliers

-- Without union
SELECT Country FROM dbo.Suppliers
SELECT Country FROM dbo.Customers

-- With union
SELECT Country FROM dbo.Suppliers
UNION
SELECT Country FROM dbo.Customers
```



EXCEPT

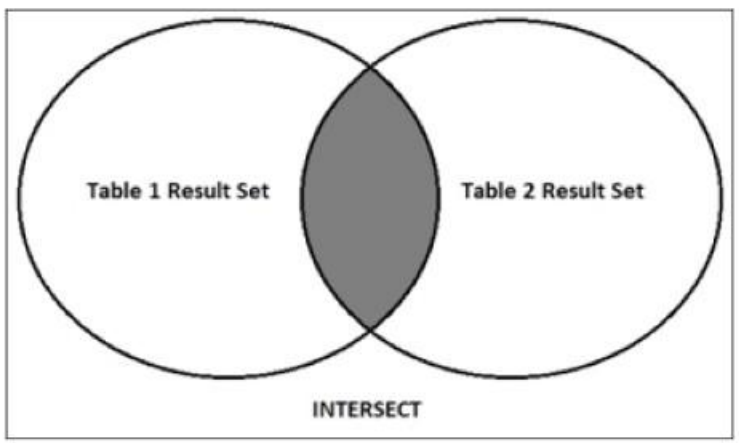
Used to find what records exists in one set and not in the other set

```
-- Return a list of customer's countries where there are no suppliers
SELECT DISTINCT Country FROM dbo.Customers
EXCEPT
SELECT DISTINCT Country FROM dbo.Suppliers
```



Used to find common values between sets from different tables

```
-- Create a list with countries from both our customers AND suppliers
SELECT DISTINCT Country FROM dbo.Suppliers
INTERSECT
SELECT DISTINCT Country FROM dbo.Customers
```



06 Manipulating Data

Chapters

1. INSERT
2. UPDATE
3. DELETE

This statement inserts a single row by default

```
INSERT [NomeTabela]
(
  [coluna1], [coluna2], …
)
VALUES
(
  value1,
  value2, …
)
```

```
-- Omitting the PK
INSERT dbo.Categories
(
  CategoryName,
  [Description]
)
VALUES
(
  'New Category',
  'With nothing...'
);
```

```
INSERT dbo.Categories
(
  CategoryID,
  CategoryName,
  [Description]
)
VALUES
(
  , -- ERRO! OMITIR A COLUNA
  'New Category',
  'With nothing...'
);
```

```
-- Insert value into PK
INSERT dbo.Region
(
  RegionID,       -- não é identity, mas é int
  RegionDescription
)
VALUES
(
  (SELECT MAX(RegionID) + 1 FROM dbo.Region),
  'New Region'
);
```

UPDATE statement is used to modify data

- Updates rows in a table or view filtered with a WHERE clause
- Only columns specified in the SET clause are modified

```
UPDATE dbo.Region
SET RegionDescription = 'New Region'
WHERE RegionDescription = 'Old Region';
```

```
UPDATE dbo.Products
SET UnitPrice = UnitPrice * 1.10, Discontinued = 0
WHERE ProductName =  'Guaraná Fantástica';
```

```
-- Suggestion: add a table Notes field and set it's value to 'changed' to rollback
UPDATE dbo.Products
SET Discontinued = 0
FROM dbo.Categories AS c
INNER JOIN dbo.Products AS p
ON c.CategoryID = p.CategoryID
WHERE p.Discontinued = 1
```

DELETE without a WHERE clause deletes all rows

- In this case is better to use TRUNCATE TABLE
- Minimally logged
- It will fail if the table is referenced by a foreign key constraint in another table

Use a WHERE clause to delete specific rows

```
DELETE FROM dbo.Region; -- Apaga toda a tabela!
```

```
DELETE FROM dbo.Region
WHERE RegionDescription = 'Old Region';
```

```
DELETE dbo.Products
FROM dbo.Categories AS c
INNER JOIN dbo.Products AS p
ON c.CategoryID = p.CategoryID
WHERE c.CategoryName = 'Beverages';
```