

Approfondimenti



SUBQUERY o Query annidate

Le *subquery* sono QUERY all'interno di un'altra QUERY più esterna.

Attenzione: ci sono casi in cui la subquery deve necessariamente restituire un unico valore e casi in cui può restituire un elenco di valori. Tutto dipende da dove si usa!

Esempi:

```
SELECT column_name(s)  
FROM table_name  
WHERE column_name IN (SELECT...);
```



La SubQuery può restituire un
elenco di valori

```
SELECT column_name(s)  
FROM table_name  
WHERE column_name >= (SELECT...);
```



La SubQuery deve restituire un
unico valore.

SUBQUERY o Query annidate

Una query SELECT può essere annidata in un'altra query SELECT all'interno della clausola **FROM**

In questo caso la subquery può restituire una tabella con più righe e colonne, eventualmente battezzata con un nuovo nome (tramite as)

```
SELECT *
```

```
FROM (SELECT colonna1, colonna2  
      FROM tabella  
      WHERE condizione) as NomeNuovaTabella
```

```
WHERE condizione;
```

UNION

L'operatore **UNION** viene utilizzato per unire i risultati di 2 o più query.

E' necessario, però, che il risultato della query composta sia **omogeneo**, ovvero è necessario che tutte le query da unire restituiscano lo stesso tipo di risultato (cioè una tabella con stesso nome, stesso numero di attributi, e attributi con lo stesso nome e dello stesso tipo). In altri termini:

Ogni SELECT deve necessariamente:

- avere lo stesso numero di colonne,
- le colonne devono avere tipi di dati simili
- le colonne devono essere nello stesso ordine

```
SELECT column_name FROM table1
      UNION
SELECT column_name FROM table2;
```

Intersezione e Differenza con query annidate

IN e **NOT IN** possono essere utilizzati, sotto opportune condizioni, per costruire le operazioni insiemistiche di **intersezione** e di **differenza di tabelle**.

Intersezione:

```
SELECT *  
FROM Impiegati  
WHERE ID IN (SELECT ID FROM NuoviDipendenti);
```

Differenza:

```
SELECT *  
FROM NuoviDipendenti  
WHERE ID NOT IN (SELECT ID FROM Impiegati);
```

NULL Values?

NON è possibile verificare se un campo è nullo utilizzando gli operatori standard di comparazione (=, <, or <>).

È necessario invece usare ***IS NULL*** e/o ***IS NOT NULL***.

```
SELECT column_names  
FROM table_name  
WHERE column_name IS NULL;
```

```
SELECT column_names  
FROM table_name  
WHERE column_name IS NOT NULL;
```

ANY e ALL

ANY viene utilizzato in una clausola Where in espressioni del tipo: $x > \text{ANY Elenco}$.

Il predicato ANY è vero se il confronto è vero **per almeno uno** dei valori dell'elenco. La condizione di ricerca è falsa se la sottoquery restituisce un insieme vuoto oppure se il confronto è falso per ciascuno dei valori restituiti dalla sottoquery.

ALL viene utilizzato in una clausola Where in espressioni del tipo: $x \leq \text{ALL Elenco}$.

Il predicato ALL restituisce vero se il confronto è vero **per ciascuno** dei valori in Elenco. La condizione di ricerca è falsa se il confronto è falso per almeno uno tra i valori dell'elenco restituito dalla sottoquery.

Valgono le seguenti equivalenze:

Attributo **IN (SELECT...)**



Attributo = **ANY (SELECT...)**

Attributo **NOT IN (SELECT...)**



Attributo **<> ALL (SELECT...)**

EXISTS

L'operatore **EXISTS** viene utilizzato per verificare l'esistenza di qualsiasi record in una sottoquery.

Assume il valore **true** se la sottoquery restituisce uno o più record, altrimenti **false**.

```
SELECT column1, column2, ...  
      FROM table1  
      WHERE EXISTS  
      (SELECT column_name FROM table2 WHERE condition);
```


EXISTS

Il predicato EXISTS controlla se vengono restituite righe dall'esecuzione della sottoquery:

- la condizione di ricerca è vera se la Select nidificata produce una o più righe come risultato,
- è falsa se la subquery restituisce un insieme vuoto.

NOTA: Il predicato Exists può essere negato nella costruzione della condizione di ricerca inserendo la parola **NOT** prima di Exists.

Il predicato Exists è il solo che non confronta un valore con uno o più altri valori.

Le colonne utilizzate nella sottoquery di una clausola Exists sono irrilevanti:

quindi, per brevità, comunemente si utilizza la forma Select * nella sottoquery.

CASE

L'istruzione **CASE** restituisce il valore corrispondente alla prima delle condizioni specificate che risulta vera (come nel costrutto SWITCH() di C#).

```
SELECT CASE  
    WHEN condition1 THEN result1  
    WHEN condition2 THEN result2  
    WHEN conditionN THEN resultN  
    ELSE result  
END AS column_name, ...  
FROM table1;
```

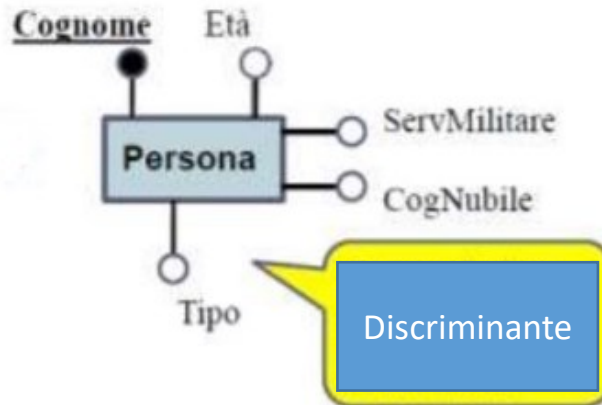
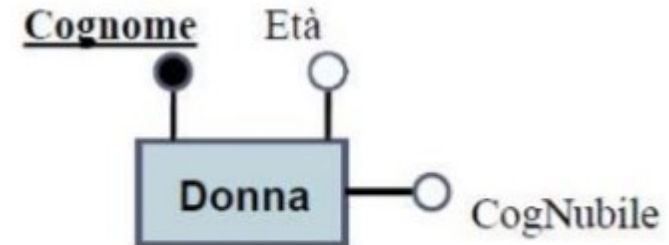
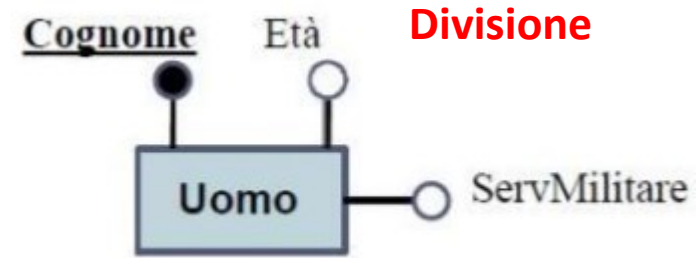
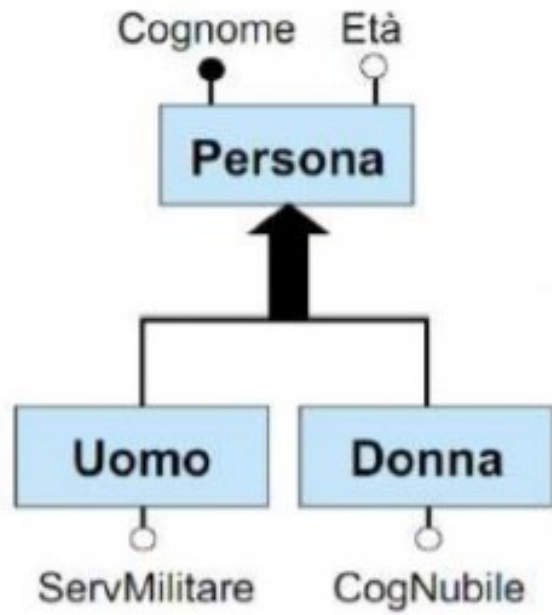
Quindi, una volta che una condizione è vera, interrompe la lettura e restituirà il risultato. Se nessuna condizione è vera, restituisce il valore nella clausola ELSE.

Demo

Union
Case



Generalizzazione e tabelle DB



Esportare Database (struttura e/o dati)

Viene generato un file .sql con gli scripts che SQL Server Management Studio crea automaticamente.

The image shows the SQL Server Enterprise Manager interface. On the left, a tree view lists databases: MasterCourse, PhoneServiceDB, ProductDatabase, ProductsDB, and a partially visible 'Pro...'. The 'Tasks' menu is open, showing options like 'New Database...', 'New Query', 'Script Database as...', 'Tasks', 'Policies', 'Facets', 'Start PowerShell', 'Azure Data Studio', 'Reports', 'Rename', 'Delete', 'Refresh', and 'Properties'. The 'Generate Scripts...' option is highlighted. To the right, the 'Set Scripting Options' dialog box is open, showing the 'Introduction' tab. The 'Specify how scripts should be saved' section has 'Save as script file' selected. The 'File name' is 'C:\Users\RenataCarriero\Documents\script.sql'. The 'Save as' section has 'Unicode text' selected. The 'Advanced' button is visible. In the background, the 'Advanced Scripting Options' dialog box is also open, showing the 'Options' tab. The 'Types of data to script' dropdown is set to 'Schema and data'. The 'Table/View Options' section is expanded, showing 'Script Change Tracking' (True), 'Script Check Constraints' (True), 'Script Data Compression Options' (False), and 'Script Foreign Keys' (True).

MasterCourse
PhoneServiceDB
ProductDatabase
ProductsDB
Pro...
Tasks
Policies
Facets
Start PowerShell
Azure Data Studio
Reports
Rename
Delete
Refresh
Properties

Generate Scripts...

Extract Data-tier Application...
Deploy Database to Microsoft Azure SQL Database...
Export Data-tier Application...
Upgrade Data-tier Application...
Import Flat File...
Import Data...
Export Data...

Set Scripting Options

Introduction
Choose Objects
Set Scripting Options
Summary
Save Scripts

Specify how scripts should be saved.

Save as notebook
File name: C:\Users\RenataCarriero\Documents\script.sql

Save as script file
Files to generate: Single script file
File name: C:\Users\RenataCarriero\Documents\script.sql
Overwrite existing file
Save as: Unicode text

Advanced

Advanced Scripting Options

Options

Script DROP and CREATE	Script CREATE
Script Extended Properties	True
Script for Server Version	SQL Server 2016
Script for the database engine edition	Microsoft SQL Server Express Edition
Script for the database engine type	Stand-alone instance
Script Logins	False
Script Object-Level Permissions	False
Script Owner	False
Script Statistics	Do not script statistics
Script USE DATABASE	True
Types of data to script	Schema and data
Table/View Options	Data only
Script Change Tracking	Schema and data
Script Check Constraints	Schema only
Script Data Compression Options	False
Script Foreign Keys	True

Types of data to script
Generates script that contains schema only or schema and data.

OK Cancel

Domande?



Ricordate il feedback!



© 2020 iCubed Srl



La diffusione di questo materiale per scopi differenti da quelli per cui se ne è venuti in possesso è vietata.

iCubed s.r.l.

Piazza Duca D'Aosta, 12 20124 MILANO

Phone: +39 02 57501057

P.IVA 07284390965

