Esercitazione n.4

La Bramea Viaggi s.p.a. ha richiesto la progettazione di una base di dati per la gestione delle gite turistiche in vista della bella stagione.

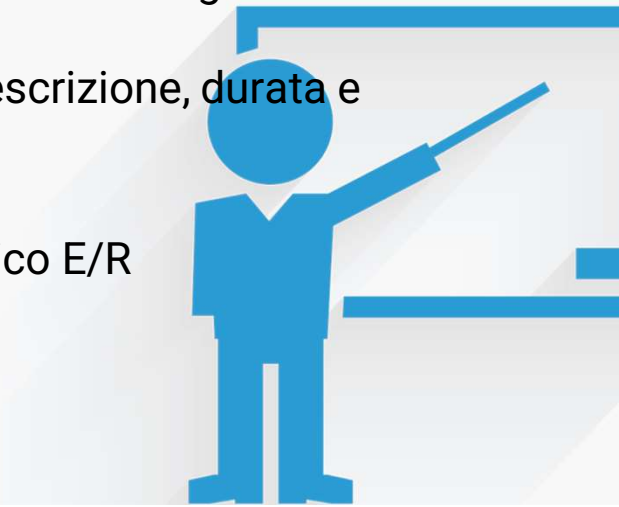
Si richiede la possibilità di inserire le informazioni relative alle gite organizzate.

Ogni gita è caratterizzata da una data di partenza, un responsabile, un itinerario.

Deve essere possibile recuperare le informazioni relative al responsabile della gita ovvero: nome, cognome, contatto telefonico ed email.

I partecipanti alla gita sono caratterizzati dall'avere nome, cognome, data e luogo di nascita ed email.

Per gli itinerari bisogna tenere traccia delle seguenti informazioni: descrizione, durata e prezzo.



- Descrivere il modello concettuale e logico mediante schema grafico E/R
- Creare le opportune tabelle del database 'Bramea Viaggi'

icubed

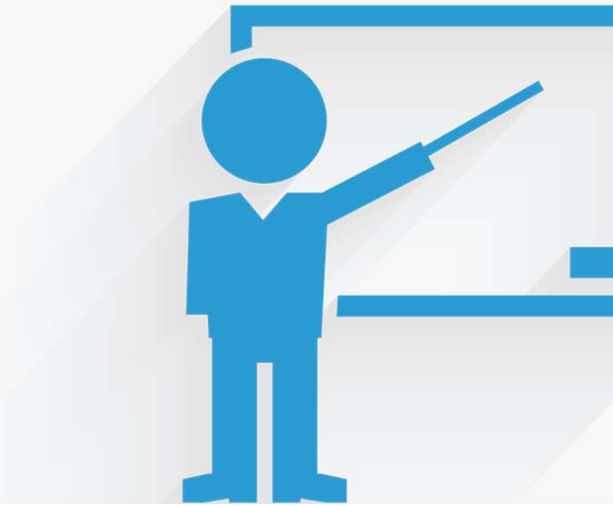
Esercitazione n.4

Dopo aver inserito alcuni dati eseguire le sequenti query:

1. Mostrare tutti i dati dei partecipanti di Roma
2. Mostrare i dati degli itinerari con prezzo superiore ai 500 euro o durata superiore ai 7 giorni
3. Selezionare la data di partenza delle gite il cui itinerario ha un prezzo superiore ai 100 euro
4. Mostrare nome, cognome e numero di telefono dei responsabili delle gite in partenza il 3 Aprile 2022
5. Mostrare i dati degli itinerari ordinati per prezzo e per durata
6. Mostrare gli itinerari con durata massima e minima
7. Mostrare le gite in partenza tra il 1 Gennaio 2022 ed il 31 Dicembre 2022

icubed

Approfondimenti



icubed

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.**

icubed

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;
```

icubed

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;
```

icubed

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);
```

icubed

NULL Values?

NON è possible 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;
```

icubed

ANY e ALL

ANY viene utilizzato in una clausola Where in espressioni del tipo: x > 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 <= 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...)**

icubed

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);
```

icubed

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`.

icubed

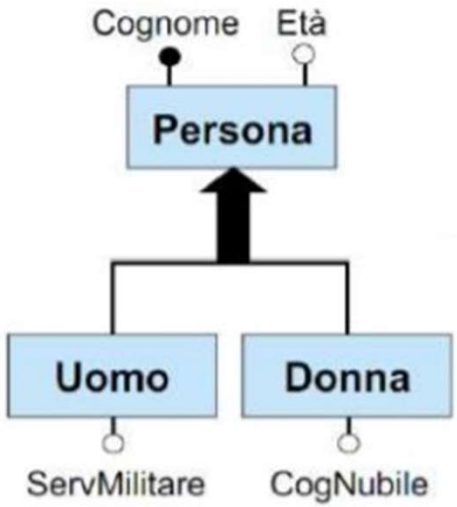
Demo

Union

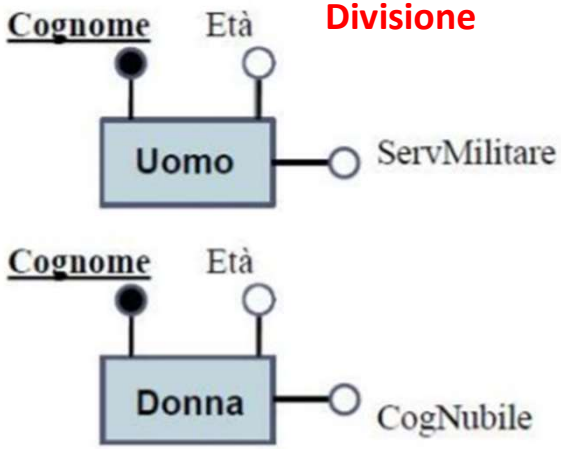
Case

icubed

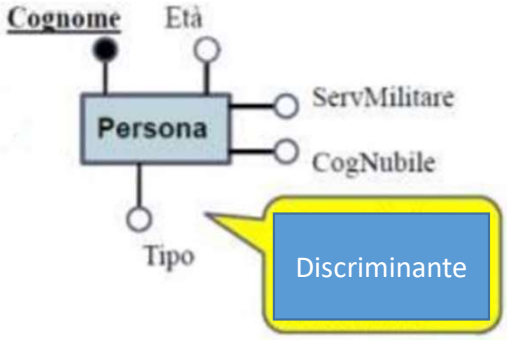
Generalizzazione e tabelle DB



Divisione



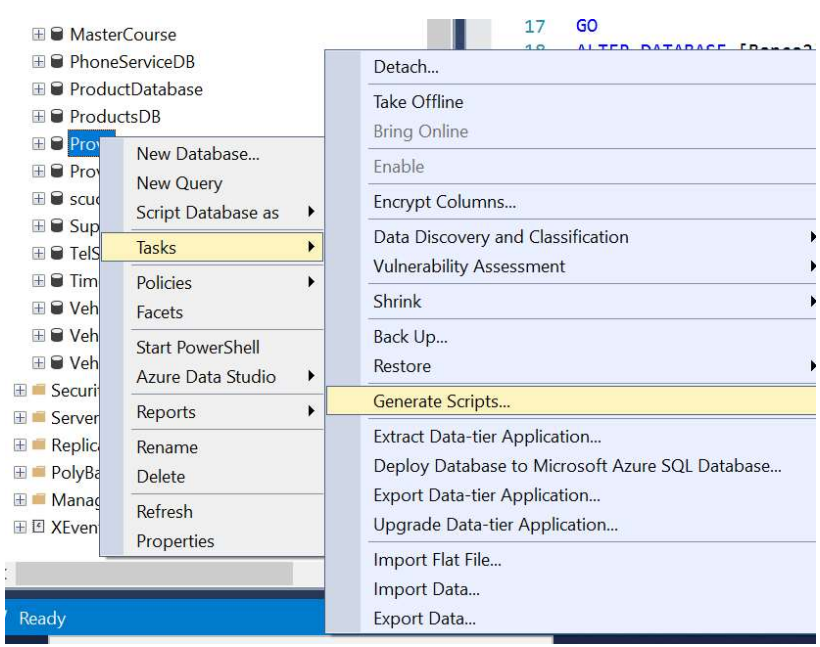
Fusione

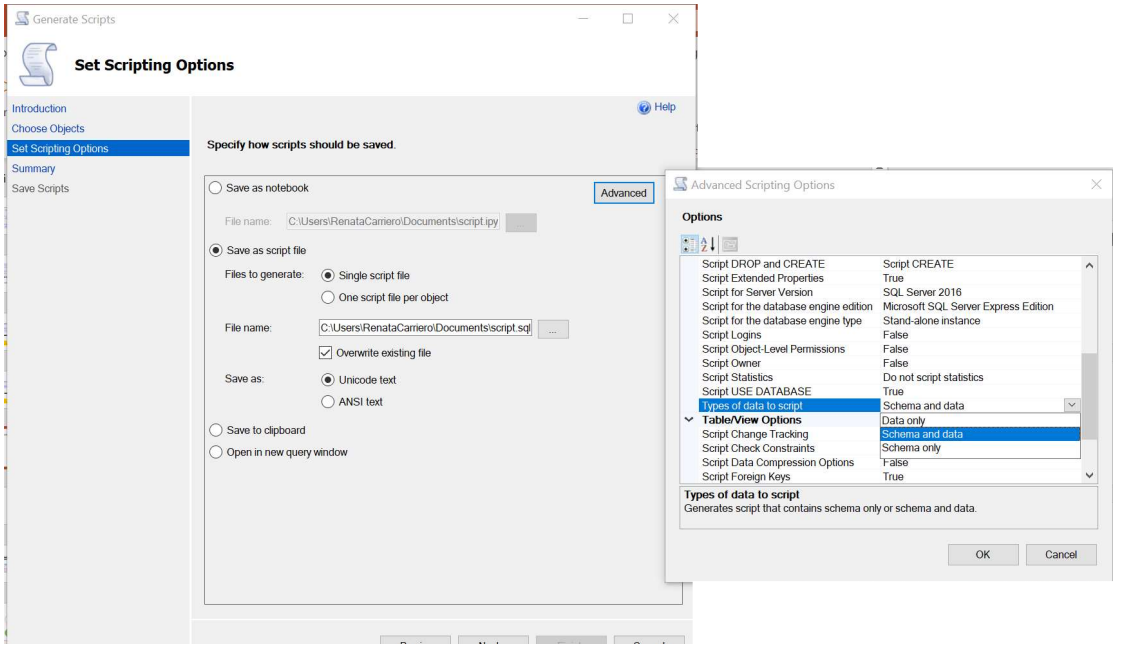


icubed

Esportare Database (struttura e/o dati)

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





icubed

Domande?

Ricordate il feedback!

© 2021 iCubed Srl

iCubed s.r.l.

Piazza Duca D'Aosta, 12 20124 MILANO

Phone: +39 02 57501057

P.IVA 07284390965