In questo modulo è affrontato lo studio di caso di un tipico esame di Stato informatica per la progettazione del modello ER con analisi dei requisiti funzionali e traduzione nel rispettivo modello logico. La trattazione è preceduta da un'esposizione teorica relativa alle basi di dati

Modello ER e modello logico: caso di studio

Prof. Michele Tarantino

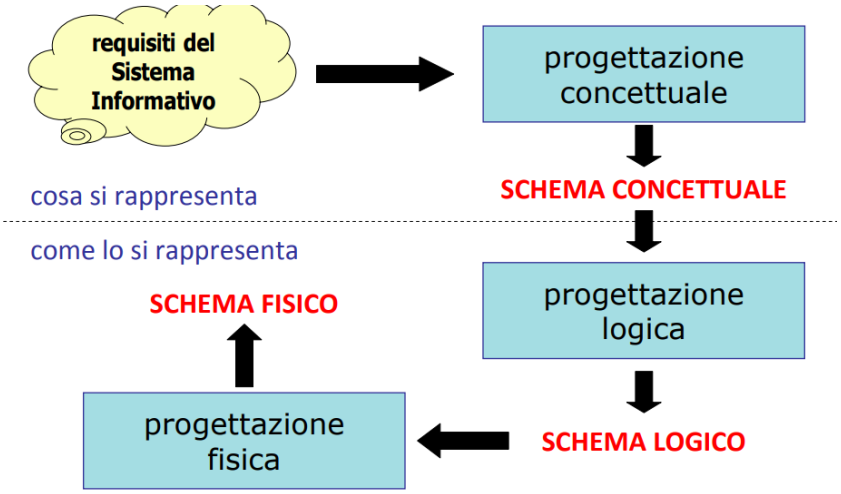
PARTE TEORICA

Per la progettazione e realizzazione di database è importante conoscere determinate definizioni e differenze, le varie fasi di sviluppo implementata ad una precisa e attenta analisi. Per dato si definisce un valore grezzo inserito in un contesto che varia a seconda del problema che si vuole risolvere, ma priva di significato. Da un dato è possibile ricavare un'informazione, o più dati consentono di ottenere più informazioni. Per informazione si intende il risultato ottenuto dall'elaborazione di uno o più dati. In informatica, l'insieme di più dati è conservato all'interno di file o di database. I database sono una collocazione integrata di dati gestita da un DBMS. Per DBMS (*Database Management System*) si intende un sistema software in grado di gestire grandi collezioni di dati integrate in modo efficace assicurando affidabilità e sicurezza. I dati che vengono salvati all'interno dei database prendono il nome di file o archivi dati, ossia un insieme di dati.

Un sistema informativo è un insieme di procedure e di risorse umane intenzionate alla finalizzazione e all'elaborazione di dati al fine di ottenere informazioni. Il sottoinsieme del sistema informativo è il sistema informatico, ossia un insieme di procedure e metodiche finalizzate al fine di gestire in modo automatizzato le informazioni in formato digitale. La progettazione di dati è una delle attività del processo di sviluppo dei sistemi informativi. La metodologia di questa attività prevede tre fasi di progettazione:

- progettazione concettuale: traduce le problematiche reali in uno schema concettuale facile da capire, senza occuparsi di come sarà costruito il database. Il modello Entità-Relazione (abbreviato con ER) è solitamente usato durante questo tipo di progettazione per definire gli aspetti statici del sistema, cioè i dati. Esso è un modello diagrammatico che descrive le entità da modellare, le relazioni che intercorrono tra di esse e le cardinalità delle relazioni.
- progettazione logica: il cui obiettivo è pervenire, a partire dallo schema concettuale, a uno schema logico che lo rappresenti in modo fedele, "efficiente" e indipendente dal particolare DBMS (Data Base Management System) adottato.
- progettazione fisica: ossia la vera e propria implementazione del database con il DBMS scelto. La traduzione dal modello logico a quello fisico consiste nella creazione delle tabelle contenenti i campi elencati tra le parentesi. In questa fase bisogna creare una tabella per ogni entità ed una colonna per ogni attributo.

La prima fase della progettazione è quella concettuale. È una modellizzazione della realtà che si vuole rappresentare. In questa fase va prodotto un diagramma ed è indipendente dallo strumento software che si intende utilizzare per la creazione del database. Serve a estrapolare i concetti alla base della realtà da informatizzare e metterli su carta.



Un modello dei dati è una collezione di concetti che vengono utilizzati per descrivere i dati, le loro associazioni, e i vincoli che questi devono rispettare. Prendono forma i modelli logici e concettuali, come prima citato il modello Entity-Relationship. Le regole di tutti i modelli devono essere assiomatiche.

Per l'interrogazione dei dati sulla base di dati si utilizza un particolare tipo di linguaggio definito SQL (*Structured Query Language*). SQL è il linguaggio standard *de facto* per DBMS relazionali, che riunisce in sé funzionalità di:

- definizione dei dati (DDL – *Data Definition Language*): definisce strutturalmente le tabelle, la definizione dei dati all'interno dele tabelle, eventuali vincoli e chiavi primari e chiave esterne;
- inserimento dei dati (DML – *Data Manipulation Language*): permette di inserire i dati sottoforma di record all'interno delle tabelle create con il DDL;
- controllo dei dati (DCL – *Data Control Language*): permette di controllare gli accessi e altri privilegi;
- interrogazione sui dati (QL – *Query Language*): è il sotto linguaggio per eccellenza per l'interrogazione dei dati presenti nelle tabelle. Ed è su questo sotto linguaggio che si andrà ad esporlo.

SQL è un linguaggio dichiarativo (non-procedurale), ovvero non specifica la sequenza di operazioni da compiere per ottenere il risultato. SQL è "relazionalmente completo", nel senso che ogni espressione dell'algebra relazionale può essere tradotta in SQL. Il modello dei dati di SQL è basato su tabelle anziché relazioni. Per l'impiego del linguaggio SQL sono stati predisposti vari costrutti:

SELECT → È l'istruzione che permette di eseguire interrogazioni (query) sul Database. La forma base è:

```
SELECT A1, A2
FROM R1, R2
WHERE <condizione>;
```

Il comando *select* determina che cosa si vuole ottenere come risultato. Si può intendere come un costrutto di proiezione. Se si vuole proiettare tutti gli attributi, è possibile scrivere con questa determinata forma:

```
SELECT *
FROM R1, R2
WHERE <condizione>;
```

Se si vuole, invece, visualizzare tutte le tuple di una tabella è possibile applicare la seguente struttura:

```
SELECT *
FROM R1
```

In questo caso la clausola *where* è omessa. Questa query restituirà tutta l'istanza della tabella presa in considerazione.

In ogni risultato di una query SQL si possono ottenere righe duplicate, ossia delle tuple (o record) che, proiettate grazie al *select*, si ripetono. Per eliminarle si usa l'opzione DISTINCT nella clausola SELECT:

```
SELECT DISTINCT A1, A2
FROM R1, R2
WHERE <condizione>
```

La clausola SELECT può contenere, inoltre, anche espressioni (definibile successivamente con un alias):

```
SELECT A1, E * 5 as PRODOTTO
FROM R1, R2
```

WHERE <condizione>

FROM → È l'istruzione che permette di determinare/definire le tabelle nella quale prendere gli attributi. Ovviamente si possono unire uno o più tabelle, o addirittura unirle tramite l'operazione di JOIN. Il JOIN è una clausola del linguaggio SQL che serve a combinare le tuple di due o più relazioni di una base di dati.

WHERE → È la clausola che permette di manipolare e di determinare le condizioni della query da cui poi trarre il risultato. In questa clausola si possono definire più condizioni che influiscono solamente ai rispettivi campi

GROUP BY → L'istruzione GROUP BY raggruppa le righe che hanno gli stessi valori in righe di riepilogo, in modo di fare un cluster degli attributi. L'istruzione GROUP BY viene spesso utilizzata con funzioni aggregate (COUNT, MAX, MIN, SUM, AVG) per raggruppare il set di risultati in base a una o più colonne. Il modello generale di una generica query SQL è la seguente:

SELECT A1, A2, FUNZIONE DI AGGREGAZIONE

FROM R1, R2

WHERE <condizione>

GROUP BY A1

HAVING <condizione su funzione di aggregazione>

ORDER BY A1 ASC (or DESC);

HAVING → Specifica una condizione di ricerca per un gruppo o una funzione di aggregazione. In genere HAVING viene inclusa in una clausola GROUP BY.

ORDER BY → Per ordinare il risultato di una query secondo i valori di una (o più colonne) si introduce la clausola ORDER BY, e per ogni colonna si specifica se l'ordinamento è per valori "ascendenti" (ASC, di default) o "discendenti" (DESC).

PARTE PRATICA

Il committente richiede la realizzazione di una base di dati per la memorizzazione di dati ed informazioni relative ad eventi espositivi per proporre alle diverse aziende del territorio software ad hoc per la gestione dei rischi ambientali. La parte che si andrà ad implementare è relativa esclusivamente alla memorizzazione dei diversi software, delle esposizioni, delle software house coinvolte e dei programmatori, indicando solo la descrizione del software proposto. I dettagli implementativi dei diversi software non è oggetto della richiesta del committente.

Dall'analisi richiesta si possono evincere le seguenti entità:

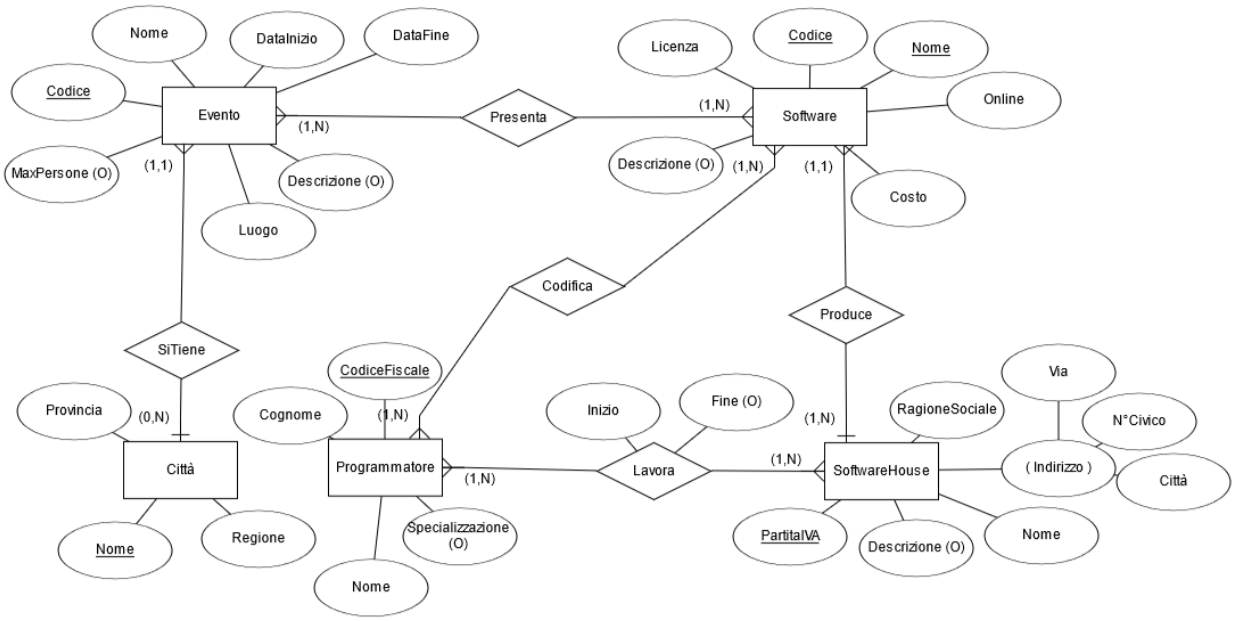
- Evento: rappresenta le informazioni relative agli eventi identificati in modo univoco tramite un codice (chiave primaria) che sono attivati per la presentazione dei software. Ogni evento è caratterizzato da un nome (non necessariamente unico – eventi diversi possono avere nomi uguali), da una data iniziale e finale dell'evento, dal luogo dove si svolge, da una descrizione, dal numero massimo di persone che possono accedere.
- Software: rappresenta le informazioni dei software presentati nei diversi eventi, dove ogni software è identificato in modo univoco da un codice, dal nome univoco del prodotto, da una licenza d'uso, da una descrizione, dalla disponibilità online del prodotto e il costo. Essendo una presentazione dei vari software, si presuppone che gli stessi non possono essere personalizzati.
- Software house: rappresenta le informazioni relative alle software house che presentato i loro prodotti nei diversi eventi. Ogni software house è rappresentata per mezzo di una partita iva (chiave primaria), da una ragione sociale (S.r.l., S.r.l.s., S.r.l.u., S.p.a.,…), dall'indirizzo della sede legale, da una descrizione.
- Programmatore: rappresenta le informazioni anagrafiche del programmatore che lavora presso la software house e che ha scritto il codice il software ed eventuali informazioni professionali.
- Città: tale entità rappresenta l'elenco delle città in cui è attivato almeno un evento espositivo, identificato con il nome della città (univoco), provincia e regione di appartenenza. Per ogni città si può memorizzare quanti eventi si sono tenuti nella città stessa (attributo derivato).

Dall'analisi funzionale e dalle entità sopra, si possono definire le seguenti relazioni tra entità:

- Evento / Software: sussiste una relazione molti a molti (M:M) in quanto l'evento propone più software e lo stesso software può essere proposto in più eventi.

- Software / Software house: sussiste una relazione uno a molti (1:M) in quanto un software è prodotto da un'unica software house ma una software house produce più software.
- Programmatore / Software: sussiste una relazione molti a molti (M:M) in quanto un programmatore può scrivere codice per diversi software e un software è scritto da diversi programmatori.
- Programmatore / Software house: sussiste una relazione molti a molti (M:M) in quanto un programmatore può lavorare per più software house in periodi diversi (attributi di relazione *inizio* e *fine*, indicando proprio il termine iniziale e quello finale di contratto per la software house) ed una software house possiede più programmatori.
- Evento / Città: sussiste una relazione uno a molti (1:M) in quanto un determinato evento essendo univoco si può svolgere in una sola città mentre una città come esplicitato può ospitare diversi eventi.

Dall'analisi delle entità e delle relazioni si può costruire il seguente modello concettuale Entità-Relazioni:



Dall'analisi funzionale sopra riportata e dal modello Entità-Relazioni esposto, si può tradurlo nel seguente modello logico nel quale le chiavi primarie sono identificate da un solo carattere di sottolineatura e le chiavi esterne con due caratteri di sottolineatura:

Eventi (Codice, Nome, Luogo, DataInizio, DataFine, MaxPersone, Città)

Città (Nome, Provincia, Regione)

Software (Codice, Nome, Licenza, Online, Costo, Descrizione, SoftwareHouse)

Presenta (Evento, Software)

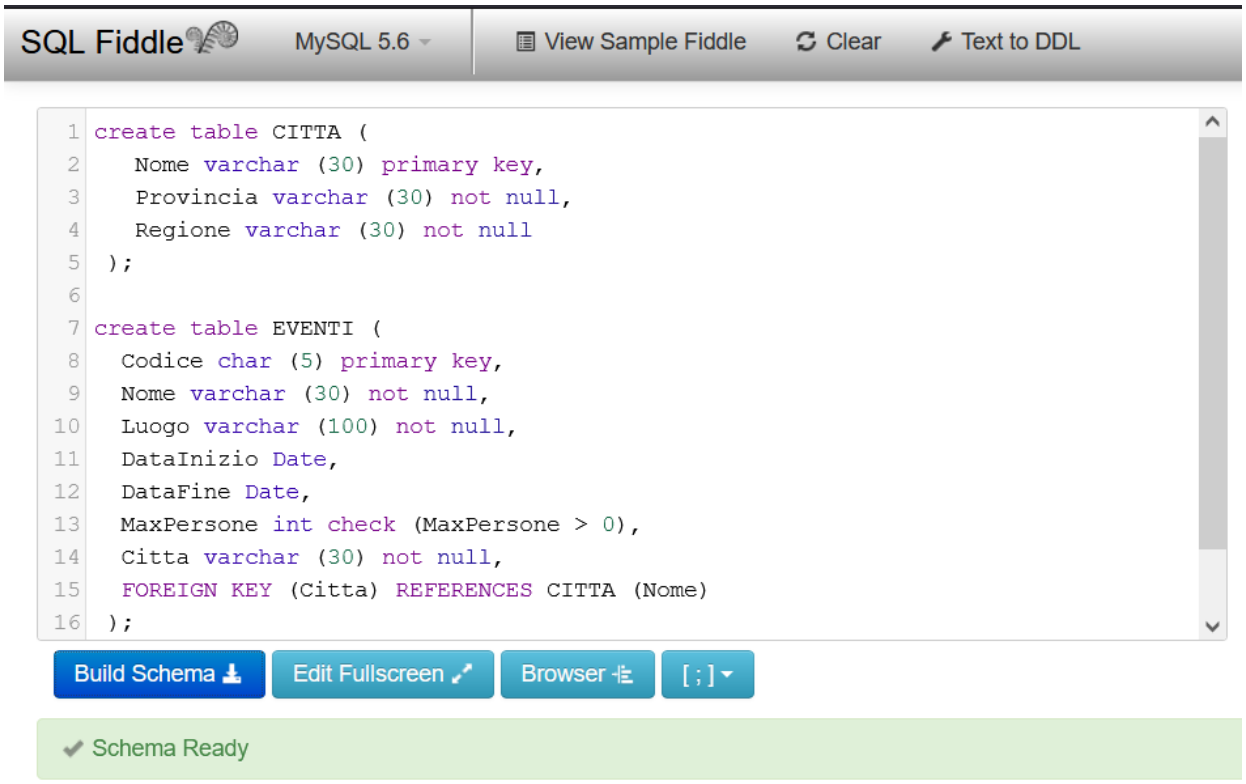
SoftwareHouse (PartitaIVA, RagioneSociale, Via, N°_Civico, Città, Descrizione)

Programmatori (CodiceFiscale, Cognome, Nome, Specializzazione)

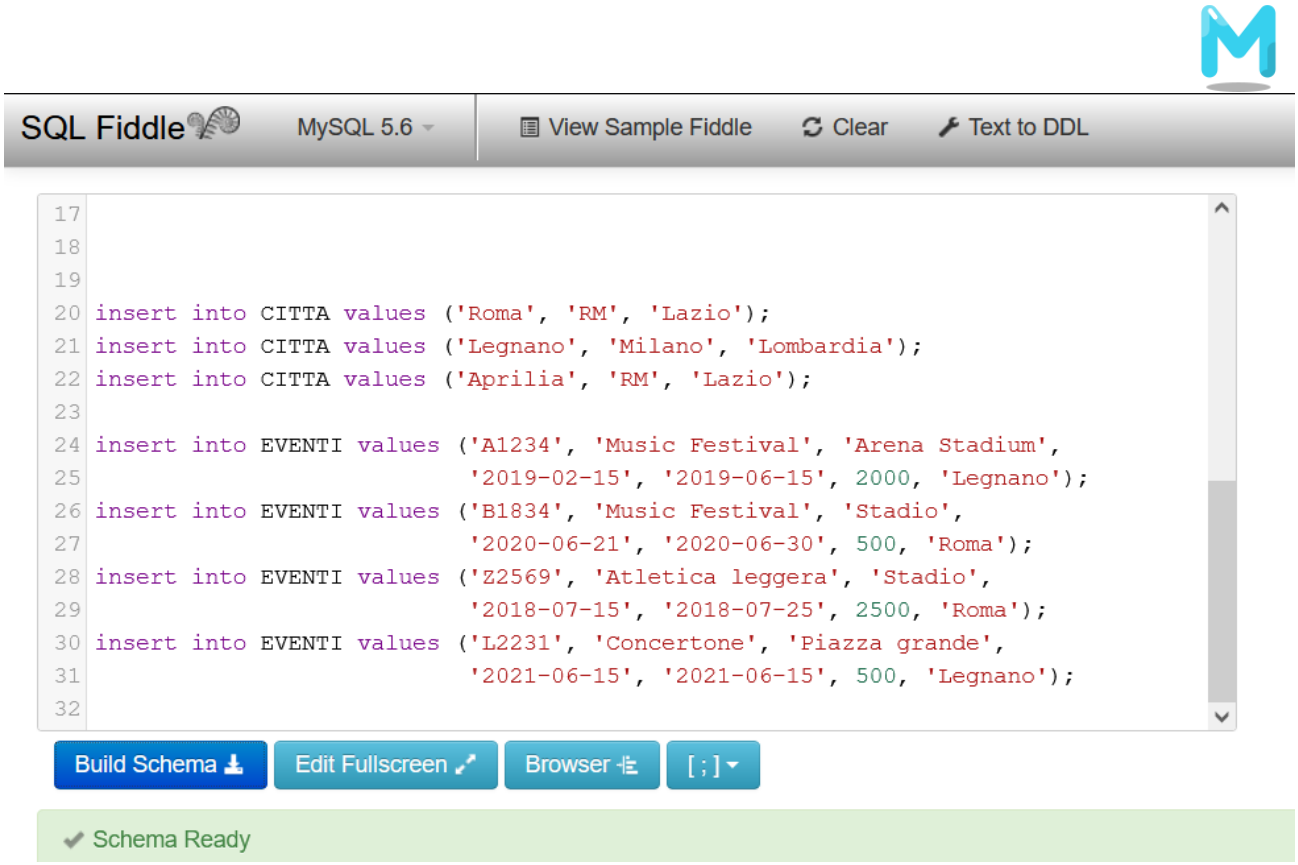
Lavora (Programmatore, SoftwareHouse, Inizio, Fine)

Codifica (Programmatore, Software)

Si implementano due tabelle significative implementate in linguaggio SQL/DDL con integrità referenziale:



Inserimento di alcune tuple significative all'interno delle due tabelle riportate sopra:



Query n°1 - Elenco di tutte le città che hanno eventi con un numero massimo di spettatori pari a 1000:

Il cui risultato è:

NOME
Legnano
Roma

Record Count: 2; Execution Time: 6ms View Execution Plan link

È necessario includere la clausola DISTINCT in quanto se sono presenti nel database città con oltre un evento queste andrebbero a replicarsi tante volte quanti sono il numero degli eventi collegati.

Query n°2 - per ogni città presente contare quanti eventi sono presenti:

```
1 select Citta, count(*) as NumeroEventi
2 from CITTA, EVENTI
3 where CITTA.Nome = EVENTI.Citta
4 group by Citta;
5
```

Run SQL Edit Fullscreen [;]

Il cui risultato è il seguente:

Citta	NumeroEventi
Legnano	2
Roma	2

Record Count: 2; Execution Time: 8ms View Execution Plan link

Query n°3 – Selezionare la città con evento avente la capienza massima:

```
select Citta, MAX(MaxPersone) as CapienzaMaggiore
from CITTA, EVENTI
where CITTA.Nome = EVENTI.Citta;
```

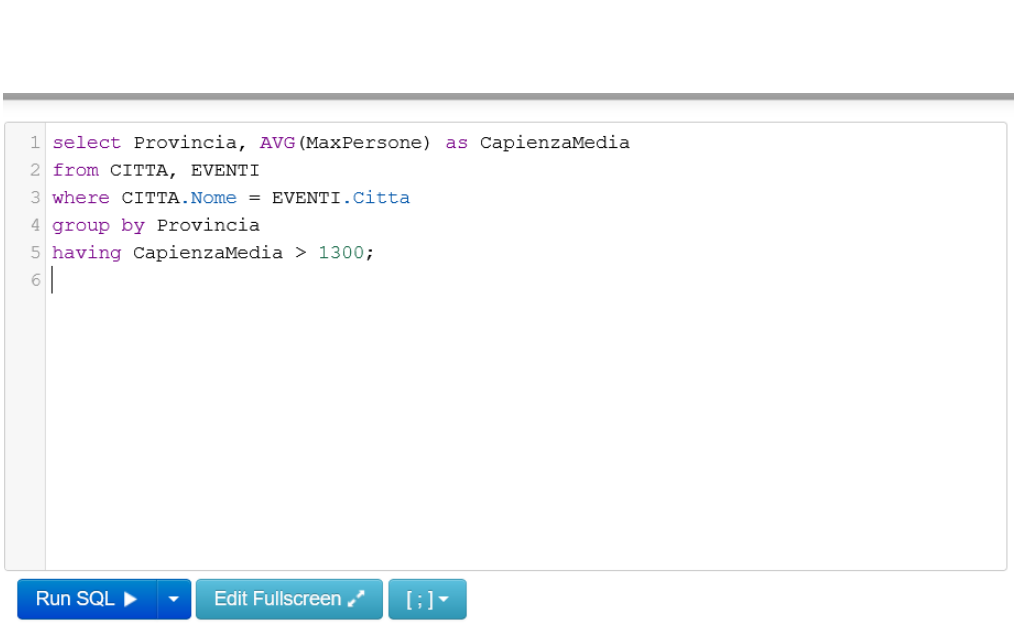
Run SQL ▶ | Edit Fullscreen | [;]

Il cui risultato è:

Citta	CapienzaMaggiore
Legnano	2500

✔ Record Count: 1; Execution Time: 10ms ✚ View Execution Plan ➦ link

Query n°4 – Selezionare le provincie le quali hanno una capienza media superiore a 1.300:



Il cui risultato è:

Provincia	CapienzaMedia
RM	1500

Record Count: 1; Execution Time: 12ms View Execution Plan link

Resta connesso e informato sui prossimi eventi, corsi e seminari:

Web
www.profmicheletarantino.com

Email
profmicheletarantino@gmail.com

Telefono
349 83 54 521

Facebook
@micheletarantinodocente

Instagram
@profmicheletarantino

Hai bisogno di un modulo personalizzato? Non esitare a contattarmi!