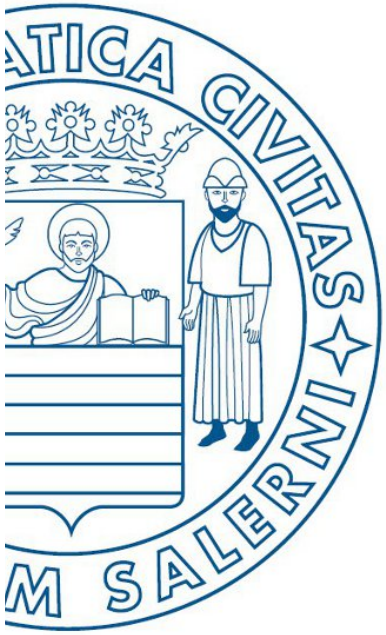


Università di Salerno
Dipartimento di
Ingegneria Industriale

UNIVERSITÀ DEGLI STUDI DI SALERNO

Fondamenti di Informatica

Introduzione al Linguaggio SQL

Prof. Christian Esposito

Corso di Laurea in Ingegneria Meccanica e Gestionale (Classe I)

A.A. 2017/18

OUTLINE

- Argomenti:
  - Introduzione al Linguaggio SQL;
  - Definizioni, Manipolazioni ed Interrogazioni in SQL;
  - Introduzione a MySQL.

Introduzione a SQL (1/3)

Structured Query Language (SQL) è un linguaggio per gestire basi di dati e contiene direttive sia del DDL che del DML per la definizione ed interrogazione dei dati. La diffusione dei SQL è dovuta alla intensa opera di standardizzazione dedicata a questo linguaggio svolta dagli organismi ANSI (American National Standards Institute) e ISO (International Standard Organization).

Nasce come strumento di lavoro dei laboratori IBM nel 1974.

- La prima definizione dello standard SQL è stata emanata nel 1986 dall'ANSI, e possedeva già gran parte delle primitive di formulazione di interrogazioni, mentre aveva un limitato supporto per la definizione e manipolazione degli schemi e delle istanze.

Introduzione a SQL (2/3)

- Lo standard è stato esteso in modo limitato nel 1989, con l'aggiunta più significativa dei vincoli di integrità referenziale. Questa versione prende il nome di SQL-89.
- Una seconda versione è stata pubblicata nel 1992 col nome ufficiale di SQL-92 o SQL-2, con l'aggiunta di nuove funzionalità. Successive versioni si sono avute nel 2003 e 2011, mirando all'arricchimento della semantica ed espressività del linguaggio.

Per quantificare in maniera puntuale, l'aderenza di un DBMS allo standard SQL sono stati definiti tre livelli di supporto dei costrutti del linguaggio:

- Entry SQL – abbastanza simile alla primordiale versione di SQL, da cui differisce per poche e lievi imprecisioni in SQL-89 corrette da SQL-2;

Introduzione a SQL (3/3)

- Intermediate SQL – contiene le caratteristiche ritenute più importanti per rispondere alle esigenze del mercato, e viene attualmente offerto da diverse versioni recenti dei prodotti relazioni di maggiore diffusione;
- Full SQL – rappresenta lo standard nella sua interezza, comprese molte funzioni avanzate.

Spessi alcune soluzione di DBMS offrono funzionalità che non sono standardizzate, utilizzando delle proprie sintassi ed interpretazioni semantiche differenti per le stesse funzionalità nello standard. I DBMS, pertanto, offrono delle lievi differenze implementative rispetto allo standard, costituendo un vero e proprio dialetto dell'SQL.

I Domini Elementari (1/4)

SQL mette a disposizione sei famiglie di domini elementari a partire dei quali si possono definire i domini associati agli attributi dello schema:

- character – permette di rappresentare singoli caratteri oppure stringhe. La lunghezza delle stringhe può essere fissa o variabile (dove si indica una lunghezza massima). Per ogni schema si può definire una famiglia di caratteri di default:

  character [varying] [(lunghezza)] [character set NomeFamigliaCaratteri]

  Per definire con questa sintassi un dominio “stringa di 20 caratteri” si può scrivere

  character(20),

I Domini Elementari (2/4)

Per “stringa di caratteri dell’alfabeto greco a lunghezza variabile, di lunghezza massima 1000” sarà impiegato

character varying (1000) character set Greek

Se la lunghezza è omessa, il sistema considera un carattere. SQL ammette anche una sintassi compatta char e varchar per character e character varying.

- bit – rappresenta una sequenza di valori binari per cui indicare la lunghezza (se omesso si intende un singolo valore). Il dominio viene solitamente impiegato per rappresentare attributi, detti flag, che specificano se l’oggetto rappresentato da una tupla possiede o meno una certa proprietà. La sintassi è:

  bit [varying] [(lunghezza)]

  Esiste la scrittura compatta varbit al posto di bit varying.

I Domini Elementari (3/4)

- Tipi numeri esatti sono i domini che permettono di rappresentare valori esatti, interi o con una parte decimale di lunghezza prefissata. Esistono in SQL quattro tipi numeri esatti:
  - numeric [(Presione [, Scala])]
  - decimal[(Presione [, Scala])]
  - integer
  - smallint

I Domini Elementari (3/4)

- Tipi numeri esatti sono i domini che permettono di rappresentare valori esatti, interi o con una parte decimale di lunghezza prefissata. Esistono in SQL quattro tipi numeri esatti:
  - numeric [(Presione [, Scala])]
  - decimal[(Presione [, Scala])]
  - integer
  - smallint

Numeri in base decimale, dove Precisione specifica il numero di cifre significative, e Scala è l'ordine di rappresentazione (quanti decimali).

I Domini Elementari (3/4)

- Tipi numeri esatti sono i domini che permettono di rappresentare valori esatti, interi o con una parte decimale di lunghezza prefissata. Esistono in SQL quattro tipi numeri esatti:
  - numeric [(Presione [, Scala])]
  - decimal[(Presione [, Scala])]
  - integer
  - smallint

Numeri in base decimale, dove Precisione specifica il numero di cifre significative, e Scala è l'ordine di rappresentazione (quanti decimali).

Per numeric la Precisione è un valore esatto, per decimal è un requisito minimo

I Domini Elementari (3/4)

- Tipi numeri esatti sono i domini che permettono di rappresentare valori esatti, interi o con una parte decimale di lunghezza prefissata. Esistono in SQL quattro tipi numeri esatti:
  - numeric [(Presione [, Scala])]
  - decimal[(Presione [, Scala])]
  - integer
  - smallint

  } Rappresentazione dei numeri interi, senza cifre decimali.

I Domini Elementari (3/4)

- Tipi numeri esatti sono i domini che permettono di rappresentare valori esatti, interi o con una parte decimale di lunghezza prefissata. Esistono in SQL quattro tipi numeri esatti.

- Tipi numerici approssimati sono impiegati per la rappresentazione di valori reali approssimati, con rappresentazione a virgola mobile dove ogni numero è dato dalla coppia mantissa ed esponente, e sono disponibili tre tipi:
  - float [(Precisione)]
  - real
  - double precision

I Domini Elementari (3/4)

- Tipi numeri esatti sono i domini che permettono di rappresentare valori esatti, interi o con una parte decimale di lunghezza prefissata. Esistono in SQL quattro tipi numeri esatti.

- Tipi numerici approssimati sono impiegati per la rappresentazione
  rappresentazione
  dato dalla coppi
  disponibili tre tipi:
  - float [(Precisione)]
  - real
  - double precision

La precisione indica il numero di cifre per la rappresentazione della mantissa, mentre la precisione dell'esponente dipende dall'implementazione. Double precision ha dimensione doppia di real.

I Domini Elementari (4/4)

- Data e ora sono stati introdotti per descrivere informazioni di natura temporale, importanti in numerosi contesti applicativi per rappresentare istanti di tempo:
  - date
  - time [(Precisione)] [with time zone]
  - timestamp [(Precisione)] [with time zone]

I Domini Elementari (4/4)

- Data e ora sono stati introdotti per descrivere informazioni di natura temporale, importanti in numerosi contesti applicativi per rappresentare istanti di tempo:
  - date
  - time [(Precisione)] [with time zone]
  - timestamp [(Precisione)] [with time zone]

Ciascuno di questi tipi è strutturato e decomponibile in un insieme di campi, ad es. date si compone di year, mounth, e day; time di hour, minute e second, mentre timestamp ha i campi da year a second.

I Domini Elementari (4/4)

- Data e ora sono stati introdotti per descrivere informazioni di natura temporale, importanti in numerosi contesti applicativi per rappresentare istanti di tempo:
  - date
  - time [(Precisione)] [with time zone]
  - timestamp [(Precisione)] [with time zone]

La precisione rappresenta il numero di cifre decimali che si devono utilizzare nella rappresentazione di frazioni di secondo, mentre l'opzione with time zone consente di accedere a due campi timezone_hour e timezone_minute che rappresentano la differenza di fuso tra l'ora locale e l'ora universale.

I Domini Elementari (4/4)

- Data e ora sono stati introdotti per descrivere informazioni di natura temporale, importanti in numerosi contesti applicativi per rappresentare istanti di tempo.
- Intervalli di tempo permette di rappresentare intervalli di tempo:

  interval PrimaunitàDiTempo [to UltimaUnitàDiTempo]

I Domini Elementari (4/4)

- Data e ora sono stati introdotti per descrivere informazioni di natura temporale, importanti in numerosi contesti applicativi per rappresentare istanti di tempo.
- Intervalli di tempo permette di rappresentare intervalli di tempo:

interval PrimaunitàDiTempo [to UltimaUnitàDiTempo]

Definiscono le unità di misura da usare, dalla più precisa alla meno precisa. Ad es. interval year to month così da indicare l'intervallo temporale come numero di anni e di mesi.

I Domini Elementari (4/4)

- Data e ora sono stati introdotti per descrivere informazioni di natura temporale, importanti in numerosi contesti applicativi per rappresentare istanti di tempo.
- Intervalli di tempo permette di rappresentare intervalli di tempo:

  interval PrimaunitàDiTempo [to UltimaUnitàDiTempo]
- boolean rappresenta i singoli valori booleani e rappresenta una restrizione del dominio bit.

Definizione di schema, tabella e domini (1/4)

SQL consente la definizione di uno schema di base di dati come collezione di oggetti (tabelle, domini, etc.) secondo la seguente sintassi:

create schema [NomeSchema] [[authorization] Autorizzazione] { DefElementoSchema}

Definizione di schema, tabella e domini (1/4)

SQL consente la definizione di uno schema di base di dati come collezione di oggetti (tabelle, domini, etc.) secondo la seguente sintassi:

```
create schema [NomeSchema] [ [authorization] Autorizzazione ] { DefElementoSchema}
```

Autorizzazione rappresenta il nome dell'utente proprietario dello schema, se omesso si intende l'utente che ha lanciato il comando.

Definizione di schema, tabella e domini (1/4)

SQL consente la definizione di uno schema di base di dati come collezione di oggetti (tabelle, domini, etc.) secondo la seguente sintassi:

```
create schema [NomeSchema] [ [authorization]
Autorizzazion  { DefElementoSchema}
```

Il nome dello schema può essere omesso, così da assumere il nome dell'utente proprietario dello schema.

Definizione di schema, tabella e domini (1/4)

SQL consente la definizione di uno schema di base di dati come collezione di oggetti (tabelle, domini, etc.) secondo la seguente sintassi:

create schema [NomeSchema] [[authorization] Autorizzazione] { DefElementoSchema}

Definizione dei suoi componenti, che non è detto avvenga contemporaneamente alla creazione dello schema, ma avvenire anche in fasi successive.

Definizione di schema, tabella e domini (1/4)

SQL consente la definizione di uno schema di base di dati come collezione di oggetti (tabelle, domini, etc.) secondo la seguente sintassi:

create schema [NomeSchema] [[authorization] Autorizzazione] { DefElementoSchema}

Una tabella SQL è costituita da una collezione ordinata di attributi e da un insieme di vincoli, con la seguente sintassi:

create table NomeTabella(NomeAttributo Dominio [ValoreDiDefault] [Vincoli] {, NomeAttributo Dominio [ValoreDiDefault] [Vincoli]} Altri Vincoli)

dove si associa un nome ed un elenco di attributi che compongono lo schema della tabella.

Definizione di schema, tabella e domini (1/4)

SQL consente la definizione di uno schema di base di dati come collezione di oggetti (tabelle, domini, etc.) secondo la seguente sintassi:

Per ogni attributo si definisce un nome, un dominio ed eventualmente un insieme di vincoli di valore sul dominio dell'attributo.

Una tabella è costituita da una collezione ordinata di attributi e da un insieme di vincoli, con la seguente sintassi:

```
create table NomeTabella( NomeAttributo Dominio
[ ValoreDiDefault ] [ Vincoli ] {, NomeAttributo Dominio
[ ValoreDiDefault ] [ Vincoli ]} Altri Vincoli )
```

dove si associa un nome ed un elenco di attributi che compongono lo schema della tabella.

Definizione di schema, tabella e domini (2/4)

La tabella è inizialmente vuota e il creatore ne possiede tutti i diritti di accesso e modifica degli elementi.

Nella definizione delle tabelle si può far riferimento ai domini predefiniti del linguaggio descritti precedentemente o a domini definiti dall'utente a partire dai domini predefiniti. Partendo dai domini predefiniti è possibile costruire nuovi domini con la seguente sintassi:

create domain NomeDominio as TipoDiDato
[ValoreDiDefault] [Vincolo]

Definizione di schema, tabella e domini (3/4)

Il dominio è caratterizzato da un nome, un dominio elementare, un eventuale valore di default, e un insieme di vincoli che rappresentano le condizioni rispettate dai valori del dominio. La dichiarazione di nuovi domini consente di associare un insieme di vincoli ad un nome di dominio, il che è utile per evitare di ripetere la stessa definizione di attributo nell'ambito di diverse tabelle.

Nella sintassi di definizione di domini e tabelle si può osservare la presenza di un valore di default in corrispondenza di ogni dominio ed attributo. Questo valore è impiegato quando si inserisce una nuova riga per cui il determinato attributo non ha valore. Se non specificato si ha il valore null.

Definizione di schema, tabella e domini (4/4)

La sintassi per la specifica di questi valori di default è la seguente:

default <GenericoValore | user | null>

dove GenericoValore rappresenta il valore compatibile con il dominio, mentre user impone come valore di default l'identificativo dell'utente che esegue il comando di aggiornamento della tabella.

Vincoli (1/7)

Sia nella definizione di domini che nella definizione di tabelle è possibile definire dei vincoli, ovvero delle proprietà che devono essere soddisfatte da ogni istanza della base di dati. Tali vincoli si dividono in intrarelazionali e interrelazionali.

I più semplici vincoli intrarelazioni sono i seguenti:

- not null – indica che il valore nullo non è ammesso come valore dell'attributo a cui è applicato;
- unique – viene applicato ad un attributo o un insieme di attributi di una tabella e impone che i valori dell'attributo siano una chiave, ovvero righe differenti della tabella non possano avere gli stessi valori; viene fatta un'eccezione per il valore nullo che può comparire su più righe senza violare il vincolo, in quanto si assume che i valori nulli siano tutti diversi tra loro.

Vincoli (2/7)

Ha una doppia sintassi:

- NomeAttributo NomeDominio unique – quando lo si applica ad un solo attributo;
- unique (Attributo {, Attributo}) – quando il vincolo deve essere applicato ad un insieme di attributi.

- primary key –specifica la chiave primaria in una tabella e può essere definito singolarmente per un attributo o un insieme di attributi. Gli attributi che compongono una chiave primaria non possono assumere il valore nullo, verificando implicitamente il vincolo not null, che può essere omesso.

Vincoli (3/7)

I vincoli interrelazionali più diffusi e significativi sono quelli di integrità referenziale, e l'SQL dispone di un apposito vincolo allo scopo, detto foreign key o chiave esterna.

Questo vincolo crea un legame tra i valori di un attributo della tabella corrente (detta interna) con i valori di un attributo di un'altra tabella (detta esterna). Il vincolo impone che per ogni riga della tabella interna il valore dell'attributo specificato, se diverso dal valore nullo, sia presente nelle righe della tabella esterna del corrispondente attributo, che deve essere ti tipo unique o solitamente una primary key.

Vincoli (4/7)

Il vincolo di chiave esterna è espresso in due modi diversi:

- Se c'è un solo attributo coinvolto, si può usare il costrutto sintattico references, con il quale si specificano la tabella esterna e il suo attributo a cui legare il vincolo;
- Quando il legame è rappresentato da un insieme di attributi, si fa uso del costrutto foreign key, posto al termine della definizione degli attributi, elencando l'insieme degli attributi nella tabella interna cui segue la definizione della corrispondenza con gli attributi della tabella esterna mediante l'uso di references.

  La corrispondenza tra attributi locali e quelli esterni avviene in base all'ordine di dichiarazione: al primo attributo di foreign key corrisponde il primo di references e via dicendo.

Vincoli (5/7)

Per gli altri vincoli, quando si rileva una violazione, l'operazione di inserimento viene rifiutata, nel caso dei vincoli di integrità referenziale è possibile decidere cosa fare quando si ha una violazione del vincolo.

Per le operazioni di modifica, è possibile reagire in uno dei seguenti modi:

- cascade – il nuovo valore dell'attributo nella tabella esterna viene riportato su tutte le corrispondenti righe di quella interna;
- set null – all'attributo referente viene assegnato il valore nullo al posto del valore modificato nella tabella esterna;
- set default – all'attributo referente viene assegnato il valore di default al posto del valore modificato nella tabella esterna;
- no action – l'azione di modifica non viene consentita.

Vincoli (6/7)

Per le operazioni di cancellazione di un elemento della tabella esterna si ha:

- cascade – si cancellano le corrispondenti righe nella tabella interna;
- set null – all'attributo referente viene assegnato il valore nullo al posto del valore cancellato nella tabella esterna;
- set default – all'attributo referente viene assegnato il valore di default al posto del valore cancellato nella tabella esterna;
- no action – l'azione di cancellazione non viene consentita.

Vincoli (7/7)

È possibile associare un comportamento diverso ad ogni tipologia di operazione indicando immediatamente dopo il vincolo di integrità referenziale:

on <delete | update> < cascade | set null | set default | no action >

Modifica degli schemi (1/2)

SQL fornisce primitive per la manipolazione degli schemi delle basi di dati che permettono di modificare le definizioni di tabelle:

- alter – permette di modificare domini e schemi di tabelle e può assumere varie forme sintattiche:
  - alter domain NomeDominio <set default ValoreDefault| drop default | add constraint DefVincolo | drop constraint NomeVincolo>
  - alter table NomeTabella <alter column NomeAttributo <set default ValoreDefault| drop default> | add constraint DefVincolo | drop constraint NomeVincolo add column DefAttributo| drop column NomeAttributo >

Modifica degli schemi (2/2)

Questi costrutti consentono di aggiungere/rimuovere entità e/o vincoli in schemi di tabelle e domini. Quando si aggiunge un nuovo vincolo, questo deve essere soddisfatto dai dati già presenti, si si ha una violazione, l'inserimento non viene consentito.

- drop – effettua delle rimozioni dei componenti, siano essi domini o schemi delle tabelle, e rispetta la seguente sintassi:

  drop < schema | domain | table > NomeElemento [restrict | cascade]

Modifica degli schemi (2/2)

Questi costrutti consentono di aggiungere/rimuovere entità e/o vincoli in schemi di tabelle e domini. Quando si aggiunge un nuovo vincolo, questo deve essere soddisfatto dai dati già presenti, si si ha una violazione, l'inserimento non viene consentito.

siano

guente

drop schema | domain | table > NomeElemento [restrict | cascade]

L'opzione restrict specifica che il comando non deve essere eseguito in presenza di oggetti non vuoti. Ad esempio, uno schema non è rimosso se contiene tabelle o altri oggetti. è l'opzione di default.

Modifica degli schemi (2/2)

Questi costrutti consentono di aggiungere/rimuovere entità e/o vincoli in schemi di tabelle e domini. Quando si aggiunge un nuovo vincolo, questo deve essere soddisfatto dai dati già presenti, si si ha una violazione, l'inserimento non viene consentito.

L'opzione cascade specifica che tutti gli oggetti specificati devono essere rimossi, e quando si rimuove una entità non vuota, anche gli oggetti che ne fanno parte sono rimossi, attivando una reazione a catena.

drop < sc[illegible]a | domain | table > NomeElemento
[restrict | cascade]

Manipolare un Database

SQL dispone di tre comandi per la manipolazione di una base di dati:

- insert – per l'inserimento di righe nella base di dati con due sintassi alternative:
  - insert into NomeTabella [ListaAttributi] <values (ListaDiValori) | SelectSQL >

Manipolare un Database

SQL dispone di tre comandi per la manipolazione di una base di dati:

- insert – per l'inserimento di righe nella base di dati con due sintassi alternative:
  - insert into NomeTabella [ListaAttributi] <values (ListaDiValori) | SelectSQL >

Nella prima forsma, la clausola values rappresenta esplicitamente i valori degli attributi della singola riga. Viene impiegata per inserire singole righe all'interno delle tabelle.

Manipolare un Database

SQL dispone di tre comandi per la manipolazione di una base di dati:

- insert – per l'inserimento di righe nella base di dati con due sintassi alternative:
  - insert into NomeTabella [ListaAttributi] <values (ListaDiValori) | SelectSQL >

Nella seconda forma, vengono inserite con un solo comando un insieme di tuple con valori estratti opportunamente interrogando la base di dati con una select.

Manipolare un Database

SQL dispone di tre comandi per la manipolazione di una base di dati:

- insert – per l'inserimento di righe nella base di dati con due sintassi alternative:
  - insert into NomeTabella [ListaAttributi] <values (ListaDiValori) | SelectSQL >
- delete – elimina righe dalle tabelle della base di dati:
  - delete from NomeTabella [where Condizione]

Manipolare un Database

SQL dispone di tre comandi per la manipolazione di una base di dati:

- insert – per l'inserimento di righe nella base di dati con due sintassi alternative:
  - insert into NomeTabella [ListaAttributi] <values (ListaDiValori) | SelectSQL >
- delete – elimina righe dalle tabelle della base di dati:
  - delete from NomeTabella [where Condizione]

Quando non viene espressa la condizione, vengono eliminate tutte le righe dalla base di dati, indicando la condizione verranno cancellate tutte le tuple che soddisfano la condizione.

Manipolare un Database

SQL dispone di tre comandi per la manipolazione di una base di dati:

- insert – per l'inserimento di righe nella base di dati con due sintassi alternative:
  - insert into NomeTabella [ListaAttributi] <values (ListaDiValori) | SelectSQL >
- delete – elimina righe dalle tabelle della base di dati:
  - delete from NomeTabella [where Condizione]
- update – permette di aggiornare uno o più attributi delle righe di una tabella che soddisfano una condizione:
  - update NomeTabella set Attributo = <Espressione | SelectSQL | null | default> {, Attributo = <Espressione | SelectSQL | null | default>} [where Condizione]

Manipolare un Database

SQL dispone di tre comandi per la manipolazione di una base di dati:

- insert – per l'inserimento di righe nella base di dati con due sintassi alternative:
  - insert into NomeTabella [ListaAttributi] <values (ListaDiValori) | SelectSQL >
- delete – elimina righe dalle tabelle della base di dati:
  - delete from NomeTabella [where Condizione]
- update – modifica i valori di alcune righe di una tabella:
  - update NomeTabella set Attributo = <Espressione | SelectSQL | null | default> {, Attributo = <Espressione | SelectSQL | null | default>} [where Condizione]

Se non è presente la condizione nel comando, allora si assume che la modifica deve essere applicata a tutte le tuple della tabella.

Interrogare un Database (1/10)

La parte di SQL dedicata alla formulazione di interrogazioni fa parte del DML, ed esprime le interrogazioni in maniera dichiarativa, specificando l'obbiettivo dell'interrogazione e non il modo attraverso cui ottenerlo. In ciò si contrappone all'algebra relazione che ha una natura imperativa, specificando i passi da compiere per estrarre le informazioni dalla base di dati.

L'interrogazione SQL per essere eseguita viene passata all'ottimizzatore delle interrogazioni (query optimizer), un componente del DBMS che analizza l'interrogazione e formula a partire da essa un'interrogazione equivalente nel linguaggio procedurale interno al DBMS e nascosto al suo utilizzatore.

Interrogare un Database (2/10)

Esistono in generale molti modi per esprimere una data interrogazione in SQL, il programmatore dovrà effettuare una scelta non sulla base dell'efficienza, ma su caratteristiche come la leggibilità e la modificabilità dell'interrogazione. SQL agevola così il lavoro del programmatore permettendogli di descrivere le interrogazioni in un modo astratto e di alto livello.

Le operazioni di interrogazione in SQL vengono specificate per mezzo dell'istruzione select, che presenta la seguente sintassi:

```
select ListaAttributi
from ListaTabelle
[where Condizione]
```

Interrogare un Database (2/10)

Esistono in generale molti modi per esprimere una data interrogazione in SQL, il programmatore dovrà effettuare una scelta non sulla base dell'efficienza, ma su caratteristiche come la leggibilità e la modificabilità dell'interrogazione. SQL agevola così il lavoro del programmatore permettendogli di descrivere le interrogazioni in un modo astratto e di alto livello.

Le operazioni di interrogazione in SQL vengono specificate per mezzo dell'istruzione select, che presenta la seguente sintassi:

```
select ListaAttributi
from ListaTabelle
[where Condizione]
```

Clausola select, o anche taget list – indica le colonne della tabella che verrà restituita come risultato dell'interrogazione.

Interrogare un Database (2/10)

Esistono in generale molti modi per esprimere una data interrogazione in SQL, il programmatore dovrà effettuare una scelta non sulla base dell'efficienza, ma su caratteristiche come la leggibilità e la modificabilità dell'interrogazione. SQL agevola così il lavoro del programmatore permettendogli di descrivere le interrogazioni in un modo astratto e di alto livello.

Le operazioni di interrogazione in SQL vengono specificate per mezzo dell'istruzione select, che presenta la seguente sintassi:

```
select ListaAttributi
from ListaTabelle
[where Condizione]
```

Clausola from – indica il nome di una o più tabelle che devono essere consultate nella fase di interrogazione operando un prodotto cartesiano tra di esse.

Interrogare un Database (2/10)

Esistono in generale molti modi per esprimere una data interrogazione in SQL, il programmatore dovrà effettuare una scelta non sulla base dell'efficienza, ma su caratteristiche come la leggibilità e la modificabilità dell'interrogazione. SQL agevola così il lavoro del programmatore permettendogli di descrivere le interrogazioni in un modo astratto e di alto livello.

Le operazioni di interrogazione in SQL vengono specificate per mezzo dell'istruzione select, che presenta la seguente sintassi:

```
select ListaAttributi
from ListaTabelle
[where Condizione]
```

Clausola opzionale where – indica la condizione che le tuple ritornate devono obbligatoriamente soddisfare.

Interrogare un Database (3/10)

È possibile definire degli alias per le colonne e tabelle indicate nella selezione:

```
select NomeAttributo [ [as] Alias ] {, NomeAttributo
[ [as] Alias ]}
from NomeTabella [ [as] Alias ] {, NomeTabella
[ [as] Alias ]}
[where Condizione]
```

Come argomento della clausola select è possibile indicare il carattere speciale *, che rappresenta tutti gli attributi delle tabelle nella clausola from.

Interrogare un Database (4/10)

È necessario specificare il nome della tabella con '.' prima del nome di un attributo, se in tabelle diverse si hanno attributi con lo stesso nome.

Nella clausola where possiamo indicare la condizione per mezzo di predicati semplici con gli operatori di confronto, aggregati con gli operatori boolean al fine di formare predicati più complessi. SQL mette a disposizione anche un operatore like per il confronto di stringhe, che supporta due caratteri speciali:

- '_' può rappresentare nel confronto un carattere arbitrario;
- '%' può rappresentare nel confronto un numero arbitrario.

Interrogare un Database (5/10)

È possibile verificare se un attributo dispone o meno in alcune righe dei valori nulli per mezzo dell'operatore is [not] null, che segue il nome di un attributo.

A differenza dell'algebra relazionale che vede le relazioni come insiemi omogenee dove non sono previste righe duplicate, nell'SQL le tabelle possono avere più righe uguali tra loro. Per emulare il comportamento dell'algebra relazionale, si ha la necessità di eliminare tali duplicati ogni qual volta si effettua una proiezione. Il programmatore specifica tale scelta con le seguenti parole chiavi dopo select:

- all – se si intende mantenere i duplicati nel risultato;
- distinct – se si vuole la rimozione dei duplicati.

Interrogare un Database (6/10)

Nella clausola from generalmente si inserisce una lista di nomi di tabelle separate dalla virgola, ciò indica il prodotto cartesiano tra le tabelle indicate. Si ha facoltà di scegliere un diverso tipo di join, indicandolo esplicitamente:

> from NomeTabella [[as] Alias] { [TipoJoin] join NomeTabella [[as] Alias] on CondizionediJoin}

Interrogare un Database (6/10)

Nella clausola from generalmente si inserisce una lista di nomi di tabelle separate dalla virgola, ciò indica il prodotto cartesiano tra le tabelle indicate. Si ha facoltà di scegliere un diverso tipo di join, indicandolo esplicitamente:

from NomeTabella [[as] Alias] { [TipoJoin] join NomeTabella [[as] Alias] on CondizionediJoin}

In questo modo la condizione di Join si sposta dalla clausola where a quella from, associata alle tabelle coinvolte nel join.

Interrogare un Database (6/10)

Nella clausola from generalmente si inserisce una lista di nomi di tabelle separate dalla virgola, ciò indica il prodotto cartesiano tra le tabelle indicate. Si ha facoltà di scegliere un diverso tipo di join, indicandolo esplicitamente:

from NomeTabella [[as] Alias] { [TipoJoin] join NomeTabella [[as] Alias] on CondizionediJoin}

Il parametro TipoJoin specifica qual è il tipo di join da usare, e ad esso si possono sostituire i seguenti termini:

- inner – interno altrimenti detto theta-join dell'algebra relazionale, ed è il valore di default che può essere omesso;
- right outer, left outer o full outer, con outer opzionale.

Interrogare un Database (7/10)

In aggiunta, ogni join può essere preceduto dalla parola chiave natural, così da consentire la specifica del join naturale dell'algebra relazionale che prevede di utilizzare nel join di due tabelle una condizione implicita di uguaglianza su tutti gli attributi caratterizzati dallo stesso nome.

Mentre una relazione è costituita da un insieme non ordinato di tuple, nell'uso reale delle basi di dati spesso si ha il bisogno di imporre un ordinamento alle tuple. SQL permette di specificare ciò con la clausola order by, con cui chiudere un'interrogazione:

order by AttriOrdinamento [asc | desc] {, AttriOrdinamento [asc | desc]}

Interrogare un Database (8/10)

SQL dispone di una serie di operatori aggregati, che effettuano operazioni su un insieme di tuple e non su ognuna di esse per volta. Esistono cinque operatori aggregati:

- count – opera un conteggio ed ha la seguente sintassi:

  count (<*|[distinct | all] ListaAttributi>)

  * restituisce il numero di righe totale, distinct restituisce il numero di diversi valori in una lista di attributi, mentre all restituisce il numero di valori diversi dal valore nullo; si assume all di default;
- sum – somma i valori restituiti da un'espressione;

Interrogare un Database (9/10)

- max e min – restituisce il massimo e minimo valore restituiti da un'espressione;
- avg – restituisce la media dei valori di un'espressione

  <sum | max | min | avg > ([distinct | all] AttrExpr)

  Distinct elimina i duplicati, all li mantiene, e il loro uso con max e min non ne altera il risultato.

SQL offre gli operatori insiemistici di union, intersect e except o minus per combinare le tabelle risultati da comandi di interrogazione.

Interrogare un Database (10/10)

Spesso si deve applicare un operatore aggregato separatamente a sottoinsiemi di righe. SQL mette a disposizione la clausola group by, che permette di specificare come dividere una data tabella in sottoinsiemi o gruppi, ed ammette come argomento un insieme di attributi. L'interrogazione raggrupperà le righe che possiedono gli stessi valori per questo insieme di attributi.

Una volta specificati i gruppi è possibile applicarvi dei predicati, che non sono valutati a livello di singole righe, ma considerando la totalità del membri del gruppo. In questo caso SQL introduce la clausola having seguita dal predicato che il gruppo deve soddisfare. Ogni sottoinsieme di righe costruito dalla group by farà parte del risultato dell'interrogazione solo se il predicato argomenti di having risulta soddisfatto.

Introduzione MySQL (1/10)

MySQL è un software di gestione di basi di dati open-source, che aiuta gli utenti a memorizzare, organizzare, e recuperare dati. è un programma molto potente con un alto grado di flessibilità

Una volta installato MySQL, è possibile accedervi alla shell digitando la seguente riga di comando nel prompt del terminale:

mysql -u root –p

Dopo aver digitato la password di root, è possibile iniziare a costruire il database. Due punti bisogna ricordare:

Tutti i comandi MySQL si concludono con il punto e virgola, se non lo si rispetta allora il comando non viene eseguito. I comandi sono solitamente scritti tutti in maiuscolo, mentre la base di dati, tabelle e testo in minuscolo così da renderli facilmente distinguibili. La linea di comando di MySQL non è case sensitive.

Introduzione MySQL (2/10)

MySQL organizza le sue informazioni in database, ognuno ospita tabelle con gli specifici dati, è possibile verificare quali database sono disponibili digitando:

SHOW DATABASES;

Sullo schermo dovrebbe apparire qualcosa del tipo:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.01 sec)
```

Introduzione MySQL (3/10)

Creare un database è molto facile: CREATE DATABASE database_name;

Digitiamo CREATE DATABASE events;

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| events             |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.01 sec)
```

Introduzione MySQL (4/10)

Il comando drop viene usato per la cancellazione di un database:

```
DROP DATABASE database_name;
```

Una volta che un database è stato creato, oppure abbiamo individuato quello con cui vogliamo lavorare, apriamo il database con:

```
USE events;
```

È possibile vedere le tabelle in questo database con il comando:

```
SHOW TABLES;
```

Introduzione MySQL (5/10)

Possiamo creare una nuova tabella, assegnandovi un nome ed uno schema:

```
CREATE TABLE potluck (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, name VARCHAR(20), food VARCHAR(30), confirmed CHAR(1), signup_date DATE);
```

Questo comando consente di effettuare le seguenti operazioni:

- crea una tabella potluck in events con 5 colonne - id, name, food, confirmed, and signup date.
- La colonna id ha un comando (INT NOT NULL PRIMARY KEY AUTO_INCREMENT) che automaticamente numera ogni riga.
- La colonna “name” è una sequenza di 20 caratteri, “food” di 30 caratteri, mentre “confirmed” ha un solo carattere, Y o N. La colonna signup_date è una data scritto come yyyy-mm-dd.

Introduzione MySQL (6/10)

Vediamo come la tabella appare quando si esegue il comando "SHOW TABLES;":

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_events |
+-----------------+
| potluck         |
+-----------------+
1 row in set (0.01 sec)
```

Possiamo richiamare lo schema della tabella con:

```
DESCRIBE potluck;
```

Bisogna ricordare che i nomi di tabelle e database sono case sensitive: potluck non è lo stesso di POTLUCK o Potluck.

Introduzione MySQL (7/10)

```
mysql>DESCRIBE potluck;
+------------+------------+-----+-----+---------+---------------+
| Field      | Type       | Null | Key | Default | Extra         |
+------------+------------+-----+-----+---------+---------------+
| id         | int(11)    | NO   | PRI | NULL    | auto_increment
|
| name       | varchar(20) | YES  |     | NULL    |               |
| food       | varchar(30) | YES  |     | NULL    |               |
| confirmed  | char(1)    | YES  |     | NULL    |               |
| signup_date | date       | YES  |     | NULL    |               |
+------------+------------+-----+-----+---------+---------------+
5 rows in set (0.01 sec)
```

Introduzione MySQL (7/10)

Per popolare la tabella possiamo inserire una riga con il seguente comando:

```
INSERT INTO potluck (id, name, food, confirmed, signup_date) VALUES (NULL, ‘John’, ‘Casserole’, ‘Y’, '2012-04-11');
```

Il ritorno di questo comando in caso di successo è Query OK, 1 row affected (0.00 sec).

Facciamo altri inserimenti e vediamo gli elementi della tabella:

```
mysql> SELECT * FROM potluck;
```

Introduzione MySQL (8/10)

```
mysql> SELECT * FROM potluck;
+----+-------+--------------+----------+------------+
| id | name  | food         | confirmed | signup_date |
+----+-------+--------------+----------+------------+
|  1 | John  | Casserole    | Y         | 2012-04-11  |
|  2 | Sandy | Key Lime Tarts | N       | 2012-04-14  |
|  3 | Tom   | BBQ          | Y         | 2012-04-18  |
|  4 | Tina  | Salad        | Y         | 2012-04-10  |
+----+-------+--------------+----------+------------+
4 rows in set (0.00 sec)
```

Introduzione MySQL (9/10)

Possiamo alterare i valori di righe inserire così da aggiornare la tabella, con il seguente comando:

```
UPDATE potluck
SET
confirmed = 'Y'
WHERE potluck.name ='Sandy';
```

Possiamo usare questo comando anche per aggiungere valori nelle celle che precedentemente erano state lasciate vuote. Possiamo anche alterare lo schema di una tabella aggiungendo una colonna:

```
ALTER TABLE potluck ADD email VARCHAR(40);
```

Possiamo anche indicare dopo quale colonna, e non alla fine, la nuova colonna va inserita, con la direttiva AFTER:

```
ALTER TABLE potluck ADD email VARCHAR(40) AFTER name;
```

Introduzione MySQL (10/10)

Possiamo rimuovere una colonna: ALTER TABLE potluck DROP email;
Possiamo anche rimuovere una tupla con il comando DELETE:

```
mysql> DELETE from potluck  where name='Sandy';
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM potluck;
+----+------+----------+----------+------------+
| id | name | food     | confirmed | signup_date |
+----+------+----------+----------+------------+
|  1 | John | Casserole | Y        | 2012-04-11  |
|  3 | Tom  | BBQ      | Y        | 2012-04-18  |
|  4 | Tina | Salad    | Y        | 2012-04-10  |
+----+------+----------+----------+------------+
3 rows in set (0.00 sec)
```

Notiamo che il progressivo associato ad ogni riga rimane invariato.

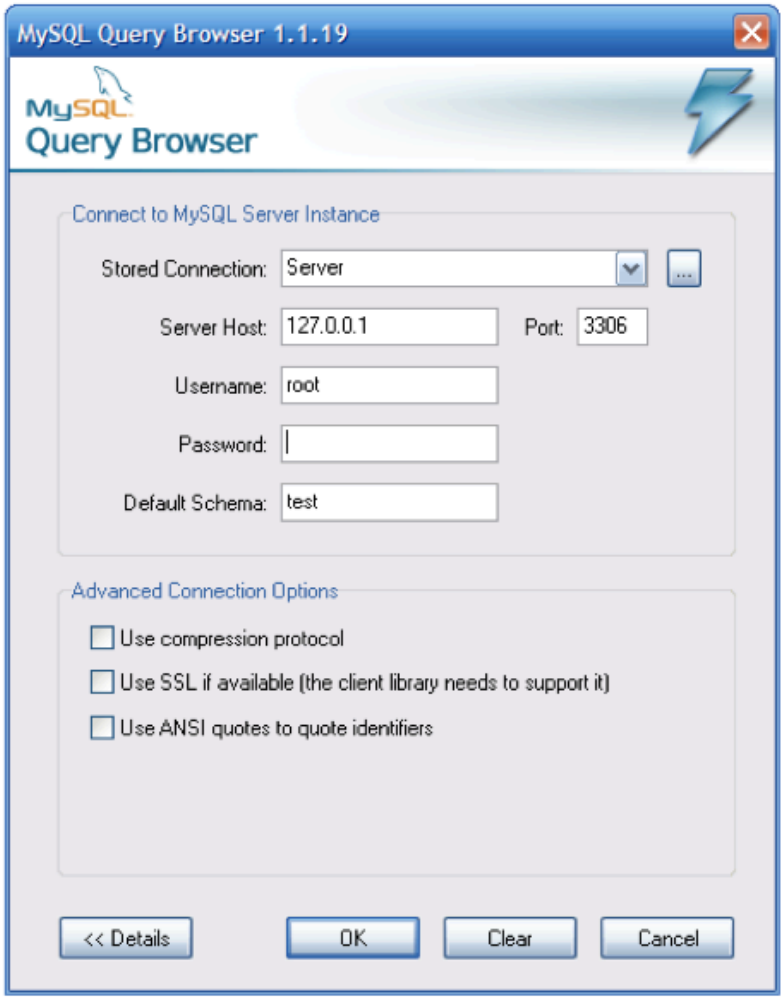
Query in MySQL (1/6)

MySQL Query Browser è uno strumento grafico fornito da MySQL per creare, eseguire ed ottimizzare query in ambiente grafico. MySQL Query Browser è progettato per effettuare query in maniera agevole ed analizzare i dati memorizzati nel vostro database MySQL.

Se da una parte tutte le query eseguite in MySQL Query Browser possono essere effettuate anche tramite l'utility da riga di comando mysql, MySQL Query Browser consente di eseguire query e modificare dati in maniera più intuitiva, tramite un'interfaccia grafica.

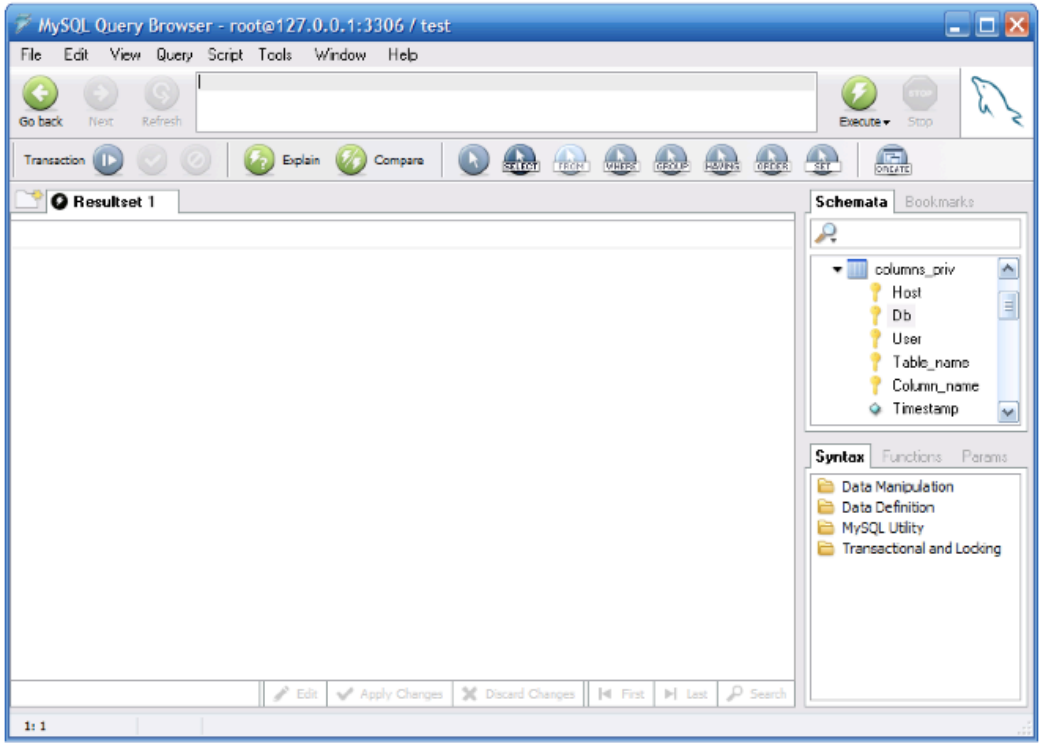
Dopo aver avviato MySQL Query Browser comparirà la finestra di dialogo di connessione. Occorre specificare il server MySQL al quale connettersi, le credenziali necessarie per l'autorizzazione sul suddetto server, l'indirizzo della macchina su cui il server gira (e su che porto è in ascolto), e il database di default (Schema) cui volete accedere. Potete inoltre specificare altre opzioni, se necessario.

Query in MySQL (2/6)



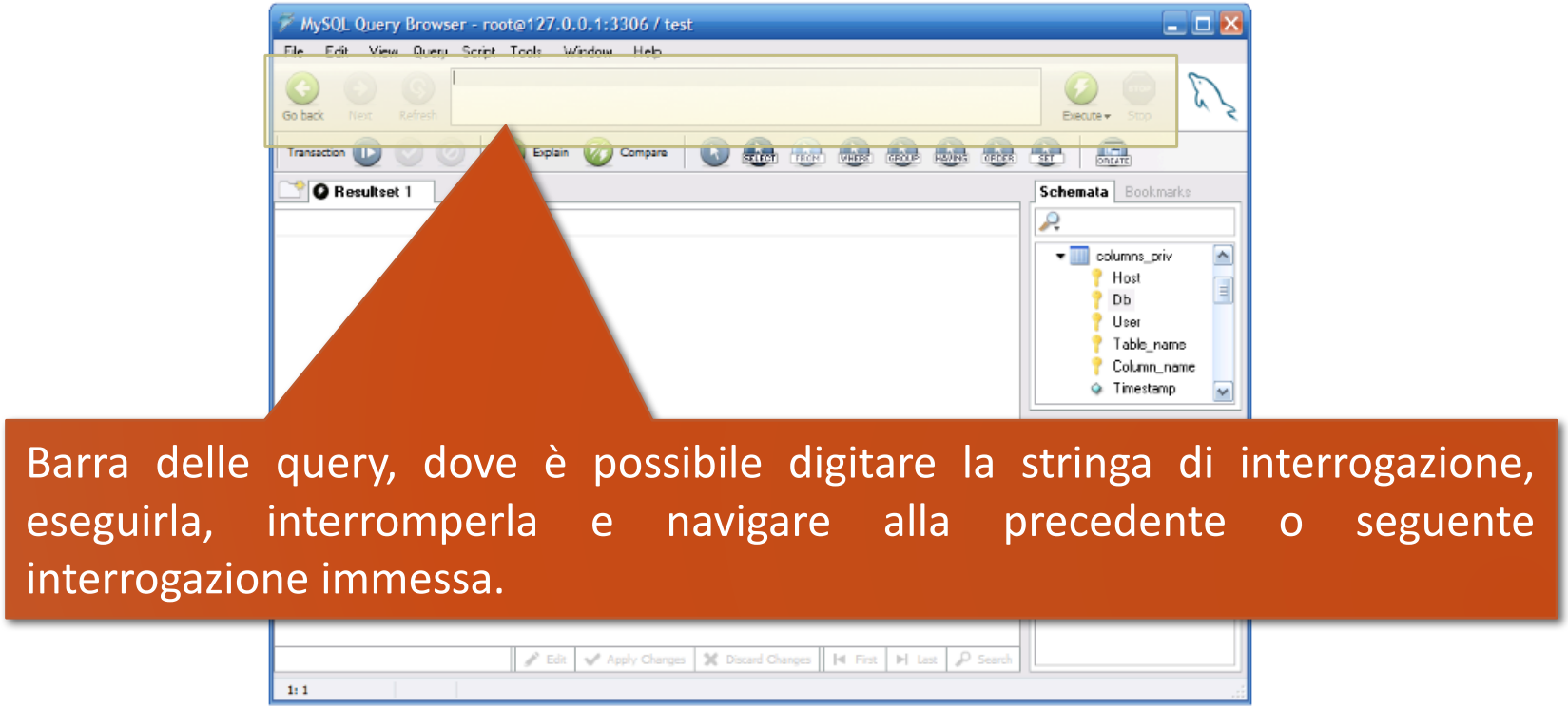
Query in MySQL (3/6)

Dopo aver stabilito correttamente una connessione al server MySQL, sarà presentata la finestra principale con tutte le funzionalità dell'applicazione.



Query in MySQL (3/6)

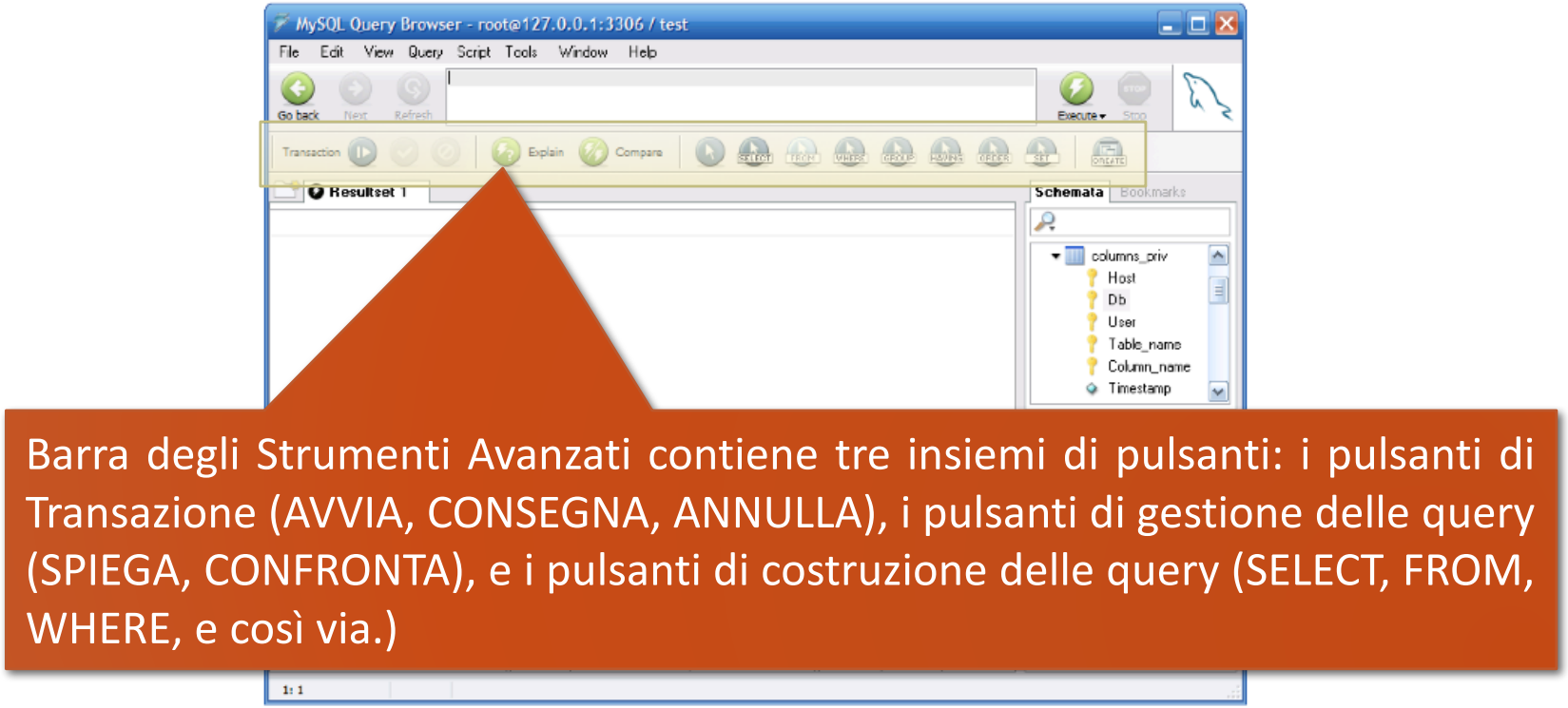
Dopo aver stabilito correttamente una connessione al server MySQL, sarà presentata la finestra principale con tutte le funzionalità dell'applicazione.



Barra delle query, dove è possibile digitare la stringa di interrogazione, eseguirla, interromperla e navigare alla precedente o seguente interrogazione immessa.

Query in MySQL (3/6)

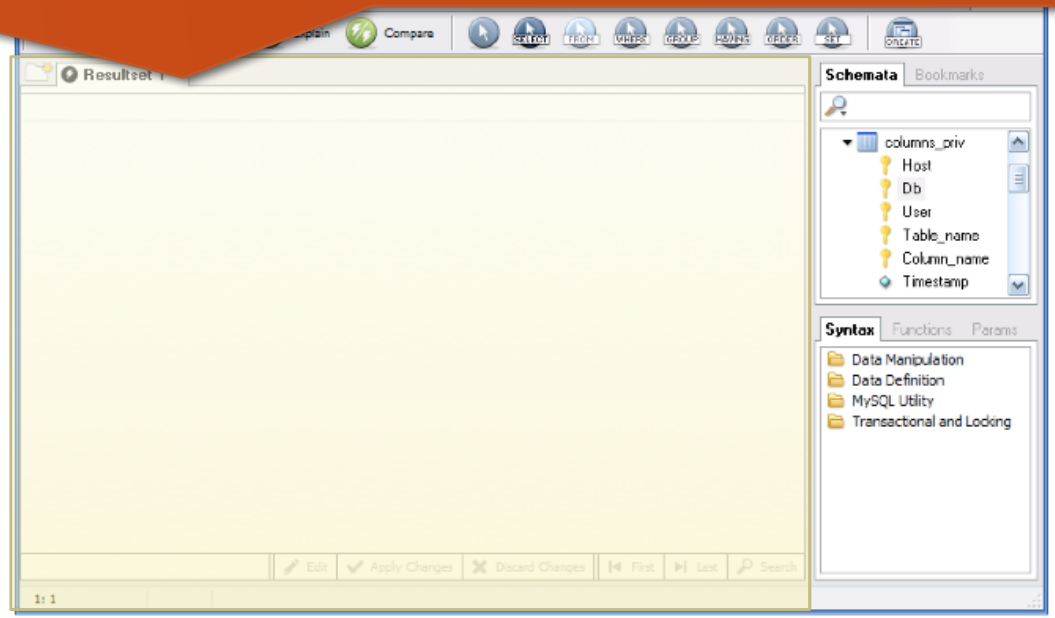
Dopo aver stabilito correttamente una connessione al server MySQL, sarà presentata la finestra principale con tutte le funzionalità dell'applicazione.



Query in MySQL (3/6)

Dopo aver stabilito correttamente una connessione al server

Area dei Risultati: Tutte le query sono visualizzate nell'area dei risultati, ed è possibile avere più linguette attive per volta, in modo da poter lavorare su query multiple. L'area dei risultati può essere divisa verticalmente e orizzontalmente per effettuare confronti, ed è possibile riunire query in diverse parti di un'area dei risultati separata, in modo da poter fare analisi master-detail.



Query in MySQL (4/6)

Esempi di stringhe di interrogazione:

- Visualizzare tutti gli attributi di tutte le tuple di una tabella:

  SELECT * FROM tabella;

- Visualizzare solo alcuni attributi di tutte le tuple di una tabella (proiezione):

  SELECT attrib_1,attrib_2 FROM tabella;

- Visualizzare tutti gli attributi solo delle tuple che rendono vera una certa condizione (selezione):

  SELECT * FROM tabella

  WHERE (condizione);

Query in MySQL (5/6)

- Visualizzare solo alcuni attributi solo delle tuple che rendono vera una certa condizione (proiezione + selezione):

```
SELECT attrib_1,attrib_2
FROM tabella
WHERE (condizione);
```

- Cercare se una data stringa è contenuta in un attributo di tipo stringa - Cerco la stringa "Paol" nell'attributo nome della tabella:

```
SELECT *
FROM tabella
WHERE nome LIKE "%paol%";
```

trova tutti i nomi come Paolo, Paola, Paolino, Giampaolo, ecc. LIKE non distingue fra maiuscole e minuscole.

Query in MySQL (6/6)

- Applicare una funzione aritmetica a sottoinsiemi aggregati di dati, dove la tabella è dotata di due attributi nome (tipo stringa) e importo (dato numerico); voglio fare le somme parziali - e visualizzare i risultati - degli importi di tutti i sottoinsiemi di tuple con lo stesso valore dell'attributo nome:

```
SELECT nome, SUM(importo)
FROM tabella
GROUP BY nome
ORDER BY SUM(importo) DESC;
```

  con l'ultima riga (facoltativa) comando di visualizzare le somme in ordine decrescente.

Esempio (1/6)

Si consideri la base di dati relazionale composta dalle seguenti relazioni:

impiegato

Matricola	Cognome	Stipendio	Dipartimento
101	Sili	60	NO
102	Rossi	40	NO
103	Neri	40	NO
201	Neri	40	SU
202	Verdi	50	SU
301	Bisi	70	IS

dipartimento

Codice	Nome	Sede	Direttore
NO	Nord	Milano	101
SU	Sud	Napoli	201
IS	Isole	Palermo	301

progetto

Sigla	Nome	Bilancio	Responsabile
Alpha	Vendite	30	202
Beta	Inventario	50	301
Gamma	Distribuzione	18	301

partecipazione

Impiegato	Progetto
101	Alpha
101	Beta
103	Alpha
103	Beta
201	Beta
202	Beta

Esempio (2/6)

Le varie relazioni hanno i seguenti vincoli di riferimento:

- tra l'attributo Dipartimento della relazione Impiegato e la relazione Dipartimento;
- tra l'attributo Direttore della relazione Dipartimento e la relazione Impiegato
- tra l'attributo Responsabile della relazione Progetto e la relazione Impiegato
- tra l'attributo Impiegato della relazione Partecipazione e la relazione Impiegato
- tra l'attributo Progetto della relazione Partecipazione e la relazione Progetto.

Scrivere gli statement SQL per creazione della base di dati.

Esempio (3/6)

impiegato

Matricola	Cognome	Stipendio	Dipartimento
101	Sili	60	NO
102	Rossi	40	NO
103	Neri	40	NO
201	Neri	40	SU
202	Verdi	50	SU
301	Bisi	70	IS

dipartimento

Codice	Nome	Sede	Direttore
NO	Nord	Milano	101
SU	Sud	Napoli	201
IS	Isole	Palermo	301

progetto

Sigla	Nome	Bilancio	Responsabile
Alpha	Vendite	30	202
Beta	Inventario	50	301
Gamma	Distribuzione	18	301

partecipazione

Impiegato	Progetto
101	Alpha
101	Beta
103	Alpha
103	Beta
201	Beta
202	Beta

Esempio (3/6)

impiegato

Matricola	Cognome	Stipendio	Dipartimento
101	Sili	60	NO
102	Rossi	40	NO
103	Neri	40	NO
201	Neri	40	SU
202	Verdi	50	SU
301	Bisi	70	IS

dipartimento

Codice	Nome	Sede	Direttore
NO	Nord	Milano	101
SU	Sud	Napoli	201
IS	Isole	Palermo	301

partecipazione

Impiegato	Progetto
	Alpha
	Beta
	Alpha
	Beta
	Beta
	Beta

progetto

```
CREATE TABLE impiegato(
  Matricola int primary key,
  Cognome varchar(255) not null,
  Stipendio int,
  Dipartimento varchar(2),
  foreign key (Dipartimento) references dipartimento(Codice)
);
```

Esempio (3/6)

impiegato

Matricola	Cognome	Stipendio	Dipartimento
101	Sili	60	NO
102	Rossi	40	NO
103	Neri	40	NO
201	Neri	40	SU
202	Verdi	50	SU
301	Bisi	70	IS

dipartimento

Codice	Nome	Sede	Direttore
NO	Nord	Milano	101
SU	Sud	Napoli	201
IS	Isole	Palermo	301

partecipazione

Impiegato	Progetto

progetto

Sigla	Nome
Alpha	Vendite
Beta	Inventario
Gamma	Distribuzion

```
CREATE TABLE dipartimento(
  Codice varchar(2) primary key,
  Nome varchar(255) not null,
  Sede varchar(255),
  Direttore int,
  foreign key (Direttore) references impiegato(Matricola)
);
```

Esempio (3/6)

impiegato

Matricola	Cognome	Stipendio	Dipartimento
101	Sili	60	NO
102	Rossi	40	NO
103	Neri	40	NO
201	Neri	40	SU
202	Verdi	50	SU
301	Bisi	70	IS

dipartimento

Codice	Nome	Sede	Direttore
NO	Nord	Milano	101
SU	Sud	Napoli	201
IS	Isole	Palermo	301

```
CREATE TABLE partecipazione(
  Impiegato int,
  Progetto varchar(255),
  primary key(Impiegato, Progetto),
  foreign key (Impiegato) references impiegato(Matricola),
  foreign key (Progetto) references progetto(Sigla)
);
```

partecipazione

Impiegato	Progetto
101	Alpha
101	Beta
103	Alpha
103	Beta
201	Beta
202	Beta

Esempio (3/6)

impiegato

Matricola	Cognome	Stipendi
101	Sili	60
102	Rossi	40
103	Neri	40
201	Neri	40
202	Verdi	5(
301	Bisi	

```
CREATE TABLE progetto(
  Sigla int primary key,
  Nome varchar(255) not null,
  Bilancio int,
  Responsabile int,
  foreign key (Responsabile) references
impiegato(Matricola)
);
```

progetto

Sigla	Nome	Bilancio	Responsabile
Alpha	Vendite	30	202
Beta	Inventario	50	301
Gamma	Distribuzione	18	301

101	Alpha
101	Beta
103	Alpha
103	Beta
201	Beta
202	Beta

Esempio (4/6)

Scrivere le seguenti interrogazioni:

1. Trovare matricola e cognome degli impiegati che guadagnano più di 50.
2. Trovare cognome e stipendio degli impiegati che lavorano a Roma.
3. Trovare cognome degli impiegati e nome del dipartimento in cui lavorano.
4. Trovare cognome degli impiegati che sono direttori di dipartimento.
5. Trovare i nomi dei progetti e i cognomi dei responsabili.
6. Trovare i nomi dei progetti con bilancio maggiore di 100 e i cognomi degli impiegati che lavorano su di essi.

Esempio (4/6)

Scrivere le seguenti interrogazioni:

1. Trovare matricola e cognome degli impiegati che guadagnano più di 50.
2. Trovare cognome e stipendio degli impiegati che
3. e degli impiegati e nome del dipartimento in cui lavorano.
4. Trovare cognome degli impiegati che sono direttori di dipartimento.
5. Trovare i nomi dei progetti e i cognomi dei responsabili.
6. Trovare i nomi dei progetti con bilancio maggiore di 100 e i cognomi degli impiegati che lavorano su di essi.

```
SELECT Matricola, Cognome
FROM IMPIEGATO
WHERE Stipendio > 50
```

Esempio (4/6)

Scrivere le seguenti interrogazioni:

1. Trovare matricola e cognome degli impiegati che guadagnano più di 50.
2. Trovare cognome e stipendio degli impiegati che lavorano a Roma.
3. T… cognome degli impiegati e nome del …no.
4. …mpiegati che sono direttori di …
5. Trovare i nomi dei progetti e i cognomi dei responsabili.
6. Trovare i nomi dei progetti con bilancio maggiore di 100 e i cognomi degli impiegati che lavorano su di essi.

```
SELECT DISTINCT Cognome, Stipendio
FROM IMPIEGATO, DIPARTIMENTO
WHERE Dipartimento=Codice
AND Sede='Roma'
```

Esempio (4/6)

Scrivere le seguenti interrogazioni:

1. Trovare matricola e cognome degli impiegati che guadagnano più di 50.
2. Trovare cognome e stipendio degli impiegati che lavorano a Roma.
3. Trovare cognome degli impiegati e nome del dipartimento in cui lavorano.
4. T… cognome degli impiegati che sono direttori di …

```
SELECT Cognome AS Impiegato, Nome
AS Dipartimento
FROM IMPIEGATO, DIPARTIMENTO
WHERE Dipartimento = Codice
```

… e i cognomi dei responsabili.

… con bilancio maggiore di 100 e i cognomi degli impiegati che lavorano su di essi.

Esempio (4/6)

Scrivere le seguenti interrogazioni:

1. Trovare matricola e cognome degli impiegati che guadagnano più di 50.
2. Trovare cognome e stipendio degli impiegati che

SELECT Cognome
FROM IMPIEGATO, DIPARTIMENTO
WHERE Matricola = Direttore

i impiegati e nome del
no.

4. Trovare cognome degli impiegati che sono direttori di dipartimento.
5. Trovare i nomi dei progetti e i cognomi dei responsabili.
6. Trovare i nomi dei progetti con bilancio maggiore di 100 e i cognomi degli impiegati che lavorano su di essi.

Esempio (4/6)

Scrivere le seguenti interrogazioni:

1. Trovare matricola e cognome degli impiegati che guadagnano più di 50.
2. Trovare cognome e stipendio degli impiegati che
4. T e cognome degli impiegati che sono direttori di d rtimento.
5. Trovare i nomi dei progetti e i cognomi dei responsabili.
6. Trovare i nomi dei progetti con bilancio maggiore di 100 e i cognomi degli impiegati che lavorano su di essi.

```
SELECT Nome AS Progetto, Cognome
AS Responsabile
FROM IMPIEGATO, PROGETTO
WHERE Matricola = Responsabile
```

Esempio (4/6)

Scrivere le seguenti interrogazioni:

1. Trovare matricola e cognome degli impiegati che guadagnano più di 50.
2. Trovare cognome e stipendio degli impiegati che

```
SELECT Nome, Cognome
FROM IMPIEGATO, PROGETTO, PARTECIPAZIONE
WHERE Sigla = Progetto AND Matricola = Impiegato
AND Bilancio > 100
ORDER BY Nome
```

ti e nome del

e sono direttori di dip nto.

5. Trov e i nomi dei progetti e i cognomi dei responsabili.
6. Trovare i nomi dei progetti con bilancio maggiore di 100 e i cognomi degli impiegati che lavorano su di essi.

Esempio (5/6)

7. Trovare cognome degli impiegati che guadagnano più del loro direttore di dipartimento.
8. Trovare cognome dei direttori di dipartimento e dei responsabili di progetto.
9. Trovare nomi dei dipartimenti in cui lavorano impiegati che guadagnano più di 60.
10. Trovare nomi dei dipartimenti in cui tutti gli impiegati guadagnano più di 60.
11. Trovare cognome degli impiegati di stipendio massimo.
12. Trovare matricola e cognome degli impiegati che non lavorano a nessun progetto.

Esempio (5/6)

7. Trovare cognome degli impiegati che guadagnano più del loro direttore di dipartimento.
8. Trova… cognome dei direttori di dipartimento e dei resp… progetto.

```
SELECT DISTINCT imp.Cognome
FROM IMPIEGATO imp, IMPIEGATO dir, DIPARTIMENTO
WHERE imp.DIPARTIMENTO = Codice AND dir.Matricola = Direttore
AND imp.Stipendio > dir.Stipendio
```

…mpiegati

…mpiegati guadagnano più di 60.

11. Trovare cognome degli impiegati di stipendio massimo.
12. Trovare matricola e cognome degli impiegati che non lavorano a nessun progetto.

Esempio (5/6)

7. Trovare cognome degli impiegati che guadagnano più del loro direttore di dipartimento.
8. Trovare cognome dei direttori di dipartimento e dei responsabili di progetto.
9. Trovare nomi dei dipartimenti in cui lavorano impiegati che guadagnano più di 60.

rtimenti in cui tutti gli impiegati

impiegati di stipendio massimo.

ognome degli impiegati che non getto.

```
SELECT Cognome
FROM IMPIEGATO, DIPARTIMENTO
WHERE Matricola = Direttore
UNION
SELECT Cognome
FROM IMPIEGATO, PROGETTO
WHERE Matricola = Responsabile
```

Esempio (5/6)

7. Trovare cognome degli impiegati che guadagnano più del loro direttore di dipartimento.
8. Trovare cognome dei direttori di dipartimento e dei responsabili di progetto.
9. Trovare nomi dei dipartimenti in cui lavorano impiegati che guadagnano più di 60.

rtimenti in cui tutti gli impiegati

impiegati di stipendio massimo.

ognome degli impiegati che non getto.

```
CREATE VIEW ID AS
SELECT *
FROM IMPIEGATO, DIPARTIMENTO
WHERE Dipartimento = Codice;

SELECT Nome
FROM ID
WHERE Stipendio > 60
```

Esempio (5/6)

7. Trovare cognome degli impiegati che guadagnano più del lo…nento.
8. Trova…ttori di dipartimento e dei respo…
9. Trova…nti in cui lavorano impiegati che g…

```
SELECT Nome
FROM ID
WHERE Codice NOT IN
   (SELECT Dipartimento
    FROM IMPIEGATO
    WHERE Stipendio <= 60)
```

10. Trovare nomi dei dipartimenti in cui tutti gli impiegati guadagnano più di 60.
11. Trovare cognome degli impiegati di stipendio massimo.
12. Trovare matricola e cognome degli impiegati che non lavorano a nessun progetto.

Esempio (5/6)

7. Trovare cognome degli impiegati che guadagnano più del loro direttore di dipartimento.
8. Trovare cognome dei direttori di dipartimento e dei resp
9. Trov ui lavorano impiegati che
10. Trov ui tutti gli impiegati guad
11. Trovare cognome degli impiegati di stipendio massimo.
12. Trovare matricola e cognome degli impiegati che non lavorano a nessun progetto.

```
SELECT Cognome
FROM IMPIEGATO i1
WHERE NOT EXISTS
   (SELECT *
    FROM IMPIEGATO i2
    WHERE i2.Stipendio > i1.Stipendio)
```

Esempio (5/6)

7. Trovare cognome degli impiegati che guadagnano più del loro direttore di dipartimento.
8. Trovare cognome dei direttori di dipartimento e dei responsabili di progetto.
9. Trova nti in cui lavorano impiegati che g
10. Trova nti in cui tutti gli impiegati guada
11. Trova egati di stipendio massimo.
12. Trovare matricola e cognome degli impiegati che non lavorano a nessun progetto.

```
SELECT Cognome
FROM IMPIEGATO
WHERE Matricola NOT IN
    (SELECT Impiegato
     FROM PARTECIPAZIONE)
```

Esempio (6/6)

13. Trovare matricola e cognome degli impiegati che lavorano a più di un progetto.
14. Trovare matricola e cognome degli impiegati che lavorano a un solo progetto.
15. Trovare per ciascun dipartimento lo stipendio medio degli impiegati che vi lavorano.
16. Trovare matricola e cognome degli impiegati che hanno lo stipendio superiore almeno del 10% rispetto allo stipendio medio del loro dipartimento.

Esempio (6/6)

13. Trovare matricola e cognome degli impiegati che lavorano a più di un progetto.
14. Trovare matricola e cognome degli impiegati che lavorano a un solo progetto.

```
SELECT DISTINCT Cognome
FROM IMPIEGATO
WHERE 1 < (SELECT COUNT(*)
    FROM PARTECIPAZIONE
    WHERE Impiegato = Matricola)
```

dipartimento lo stipendio medio avorano.

gnome degli impiegati che hanno e almeno del 10% rispetto allo stipendio medio del loro dipartimento.

Esempio (6/6)

13. Trovare matricola e cognome degli impiegati che lavorano a più di un progetto.
14. Trovare matricola e cognome degli impiegati che lavorano a un solo progetto.
15. ovare per ciascun dipartimento lo stipendio medio impiegati che vi lavorano.

gnome degli impiegati che hanno e almeno del 10% rispetto allo ro dipartimento.

```
SELECT Cognome
FROM IMPIEGATO
WHERE 1 = (SELECT COUNT(*)
    FROM PARTECIPAZIONE
    WHERE Impiegato = Matricola)
```

Esempio (6/6)

13. Trovare matricola e cognome degli impiegati che lavorano a più di un progetto.
14. Trovare matricola e cognome degli impiegati che lavorano a un solo progetto.
15. Trovare per ciascun dipartimento lo stipendio medio degli impiegati che vi lavorano.
16. [illegible]ovare matricola e cognome degli impiegati che hanno [illegible]ipendio superiore almeno del 10% rispetto allo [illegible]ro dipartimento.

```
SELECT Codice, Nome,
AVG(Stipendio)
FROM ID
GROUP BY Codice, Nome
```

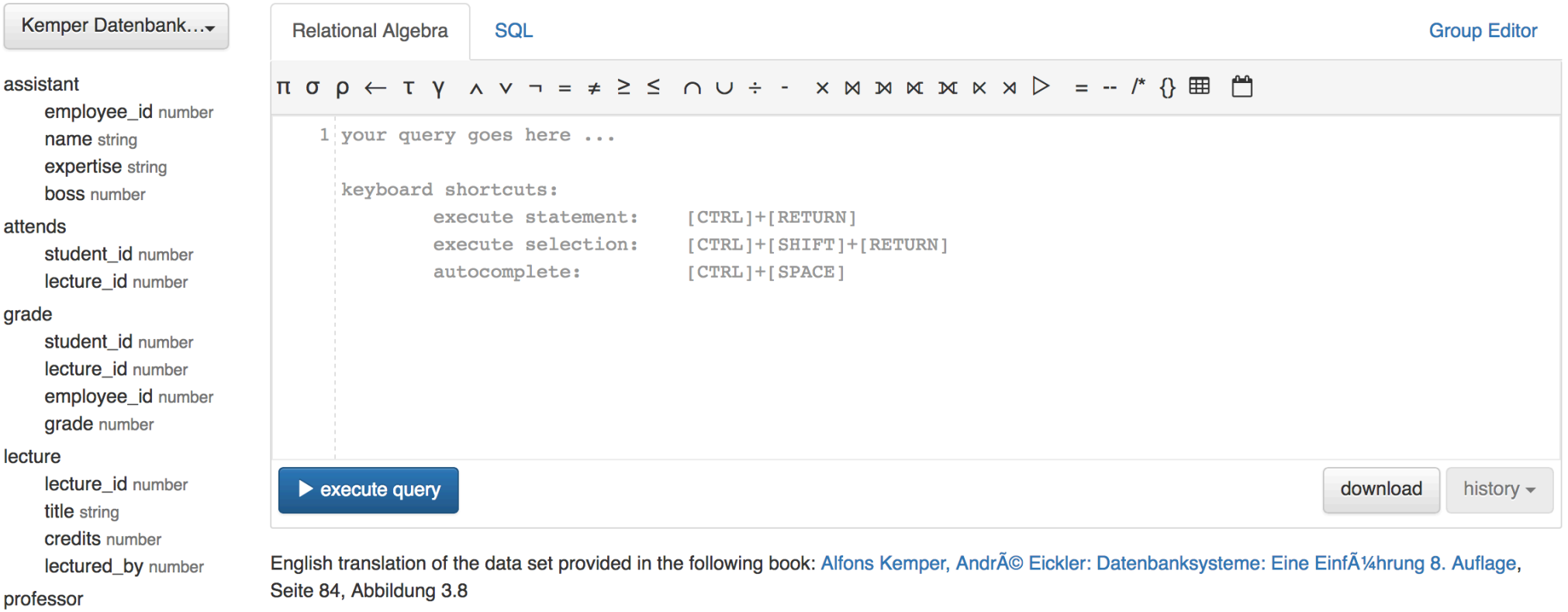
Esempio (6/6)

```
SELECT Matricola, Cognome, Stipendio
FROM IMPIEGATO i1
WHERE Stipendio >
(SELECT 1.1 * AVG(Stipendio)
FROM Impiegato i2
WHERE i1.Dipartimento = i2.Dipartimento)
```

1. degli impiegati che
1. degli impiegati che
15. lo stipendio medio degli impiegati che vi lavorano.
16. Trovare matricola e cognome degli impiegati che hanno lo stipendio superiore almeno del 10% rispetto allo stipendio medio del loro dipartimento.

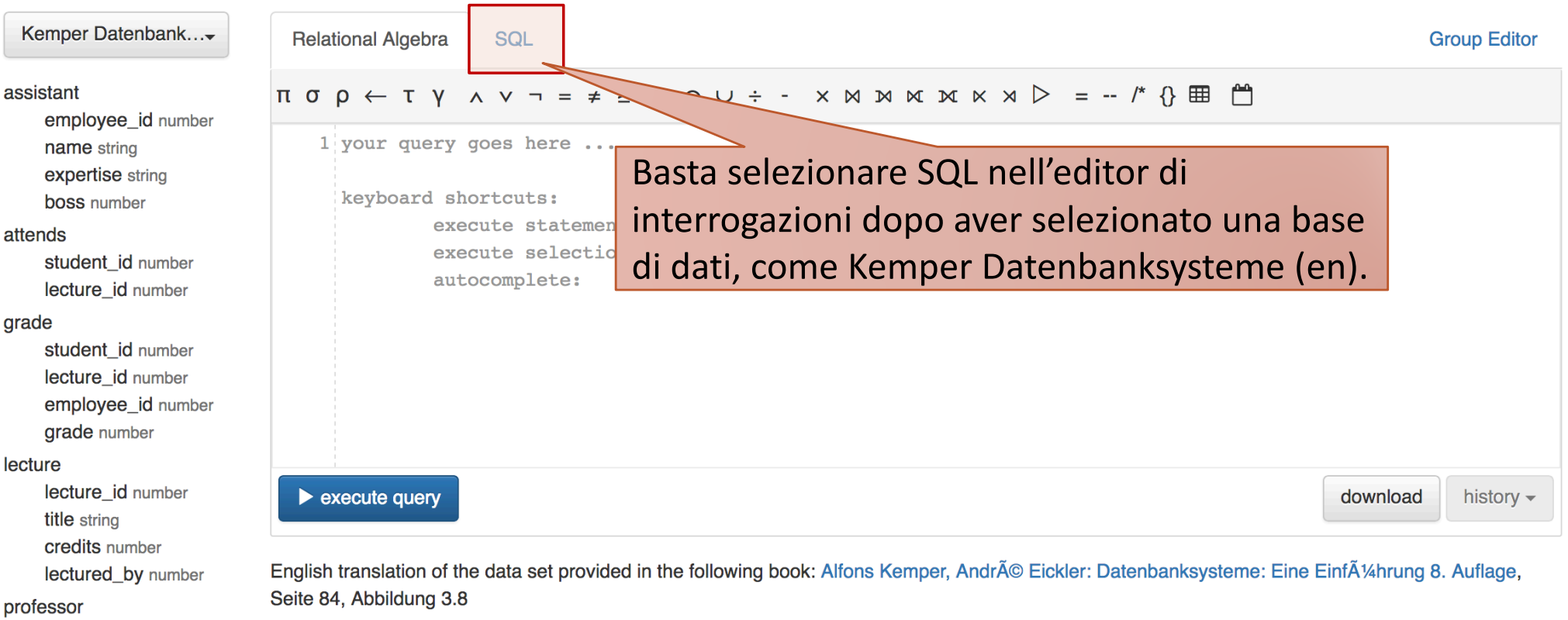
Esercitazione: RelaX (1/3)

RelaX dà la possibilità di interrogare una delle proprie basi di dati anche con SQL e non solo con l'algebra relazionale.

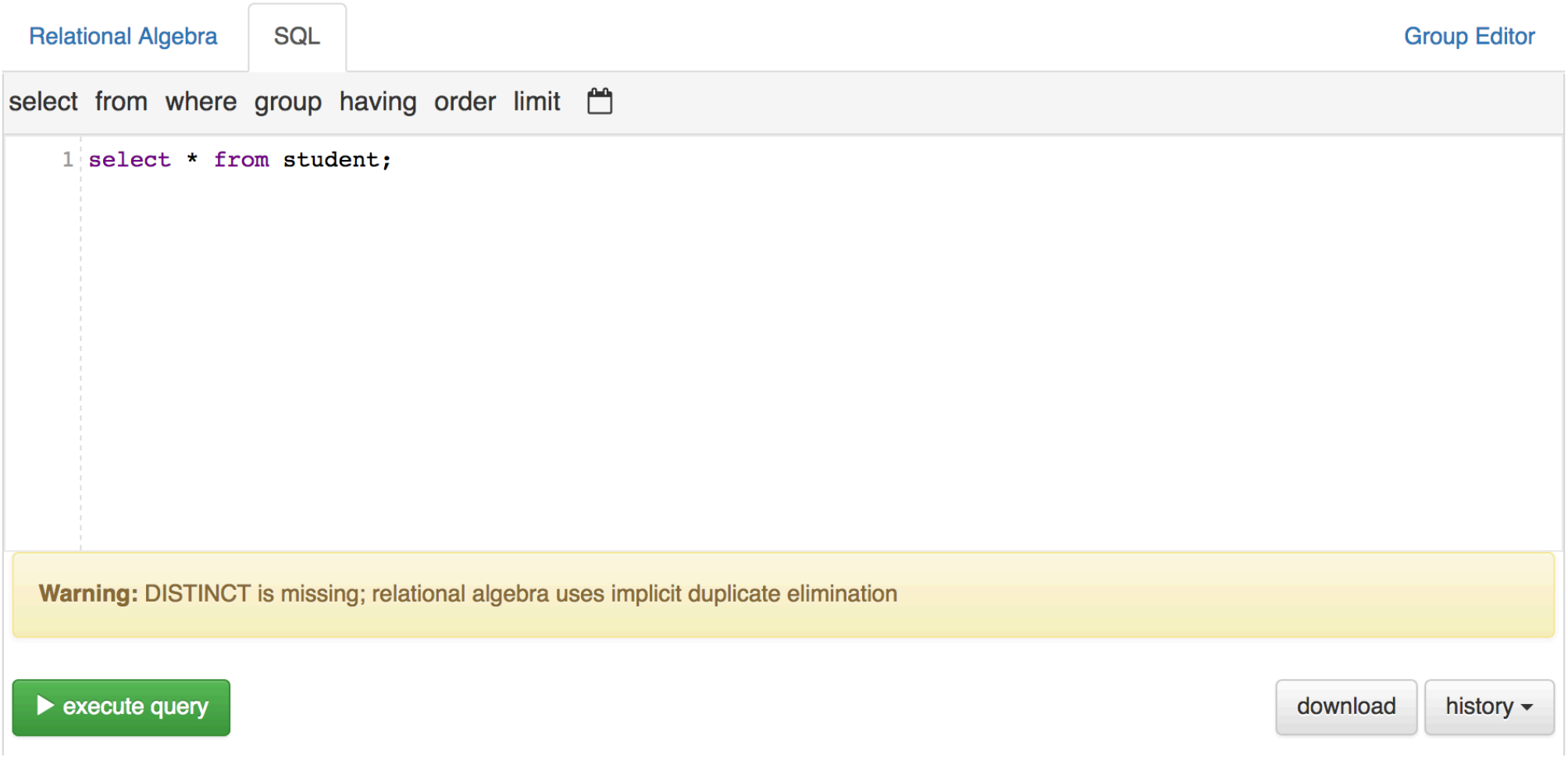


Esercitazione: RelaX (1/3)

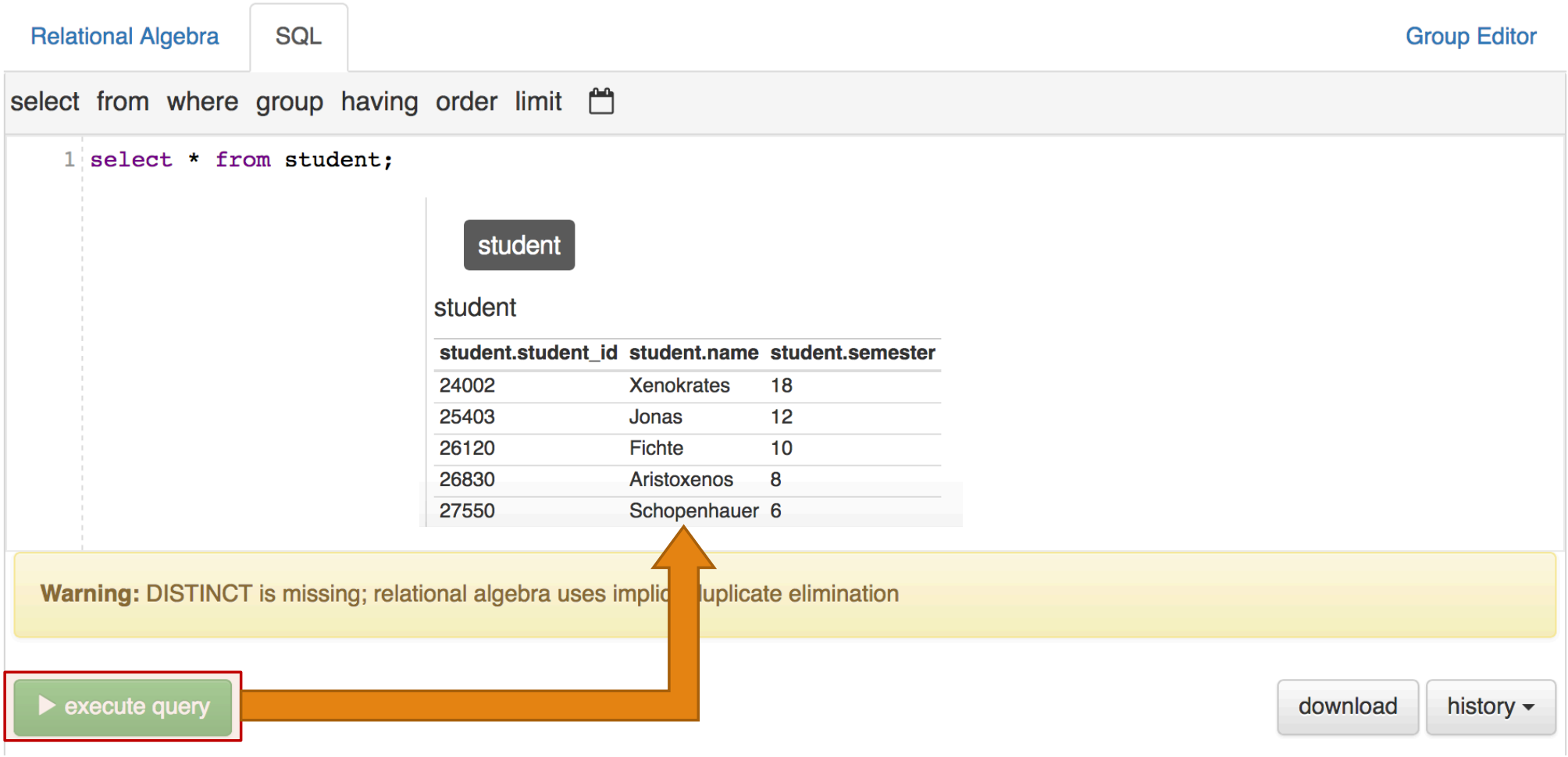
RelaX dà la possibilità di interrogare una delle proprie basi di dati anche con SQL e non solo con l'algebra relazionale.



Esercitazione: RelaX (2/3)



Esercitazione: RelaX (2/3)



Esercitazione: RelaX (3/3)

Utilizzare l'applicazione per formulare le seguenti interrogazioni mediante il linguaggio SQL:

- I nomi dei professori e dei relativi studenti che hanno ottenuto dei voti superiori a 2;
- Il titolo delle lezioni, e i nomi dei professori che le tengono, seguiti da studenti appartenenti a semestri con numero identificativo superiore o pari a 8;
- I crediti conseguiti dagli studenti con un voto maggiore o pari a 2 con il titolo del corso e il nome degli studenti;
- Il nome dei professori e i titoli dei corsi per cui alcuni studenti hanno ottenuto una votazione.

Basi di Dati e MATLAB (1/5)

MATLAB offre la possibilità di connettersi a un DBMS, per accedere ad una determinata base di dati, con la funzione database:

conn = database(datasource,username,password)

conn = database(datasource,username,password,driver,url)

Richiede il nome del database, un nome utente e la password. Nella seconda forma, richiede anche il nome del driver e l'url del BDMS.

Una volta stabilita questa connessione è possibile far eseguire delle query e poter ottenere i risultati in una variabile che MATLAB può successivamente elaborare. Questo è possibile per mezzo della funzione select, che richiede due input, il primo è la connessione verso un DBMS e il secondo è la stringa che contiene la query SQL.

Bisogna sempre chiudere la connessione, con la funzione close.

Basi di Dati e MATLAB (2/5)

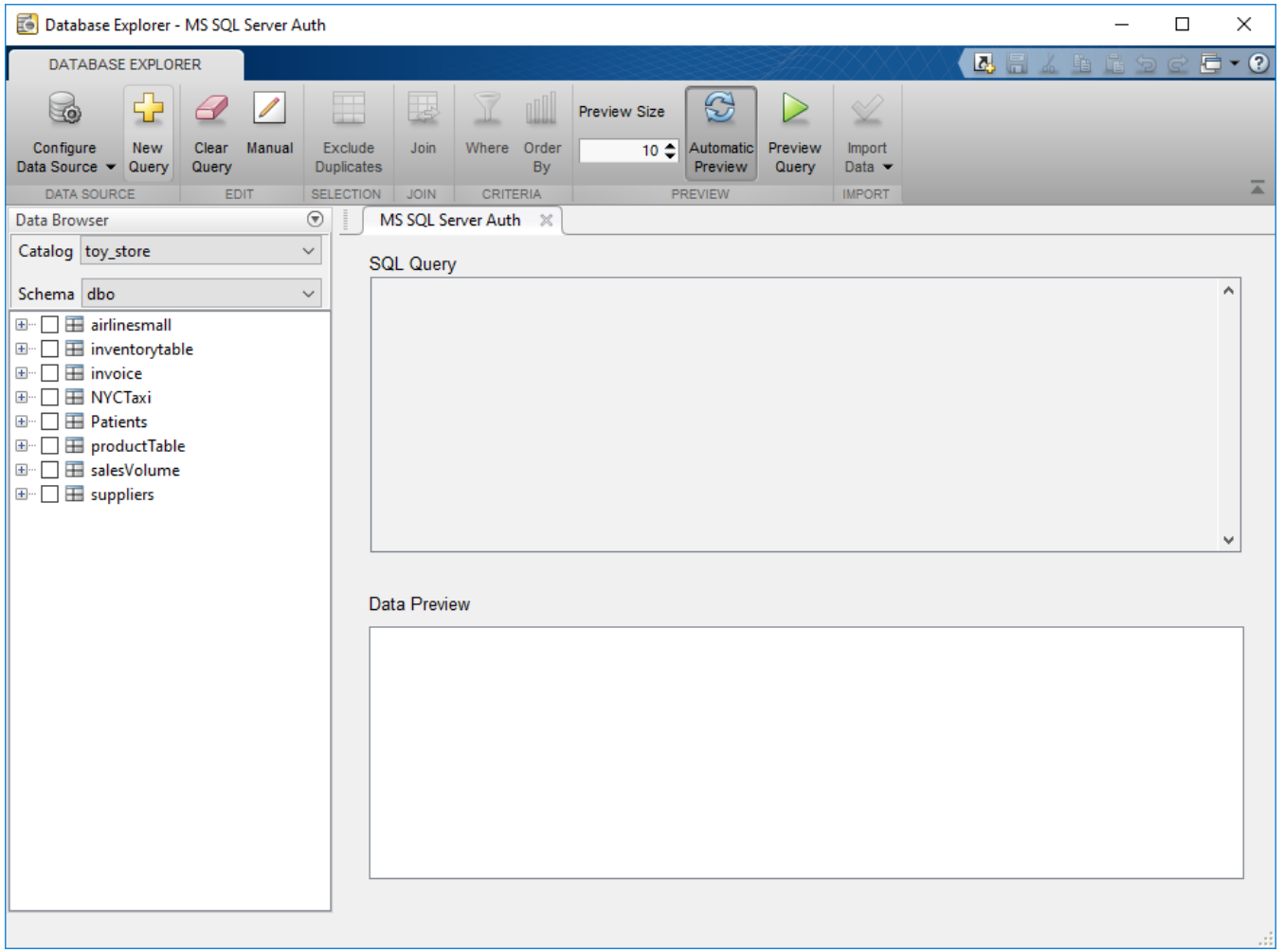
```
datasource = 'myDB';
driver = 'com.mysql.jdbc.DriverDatabase';
url = 'jdbc:mysql://devmetrics.mrkps.com/testing';
conn = database(datasource, 'usr', 'pwd', driver, url);
selectquery = 'SELECT * FROM inventoryTable';
data = select(conn,selectquery);
…
close(conn)
```

MATLAB dispone anche di un app chiamata Database Explorer che consente di connettersi a un database, esplorarne i dati, ed importare tuple nel workspace MATLAB in modo visuale.

Basi di Dati e MATLAB (3/5)

Questa app è invocabile:

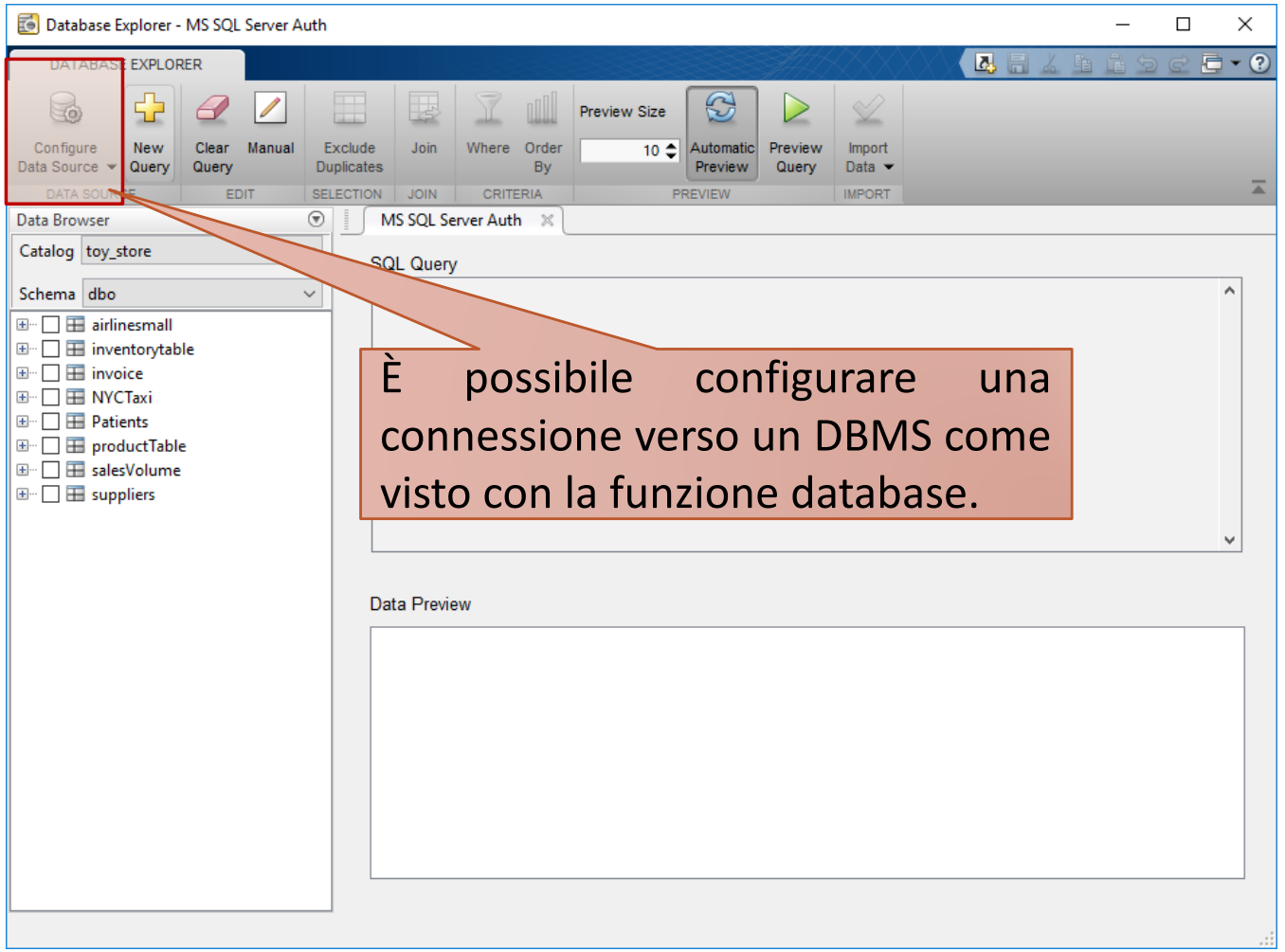
- digitando nella finestra dei comandi il comando databaseExplorer,
- selezionandola nella barra degli strumenti.



Basi di Dati e MATLAB (3/5)

Questa app è invocabile:

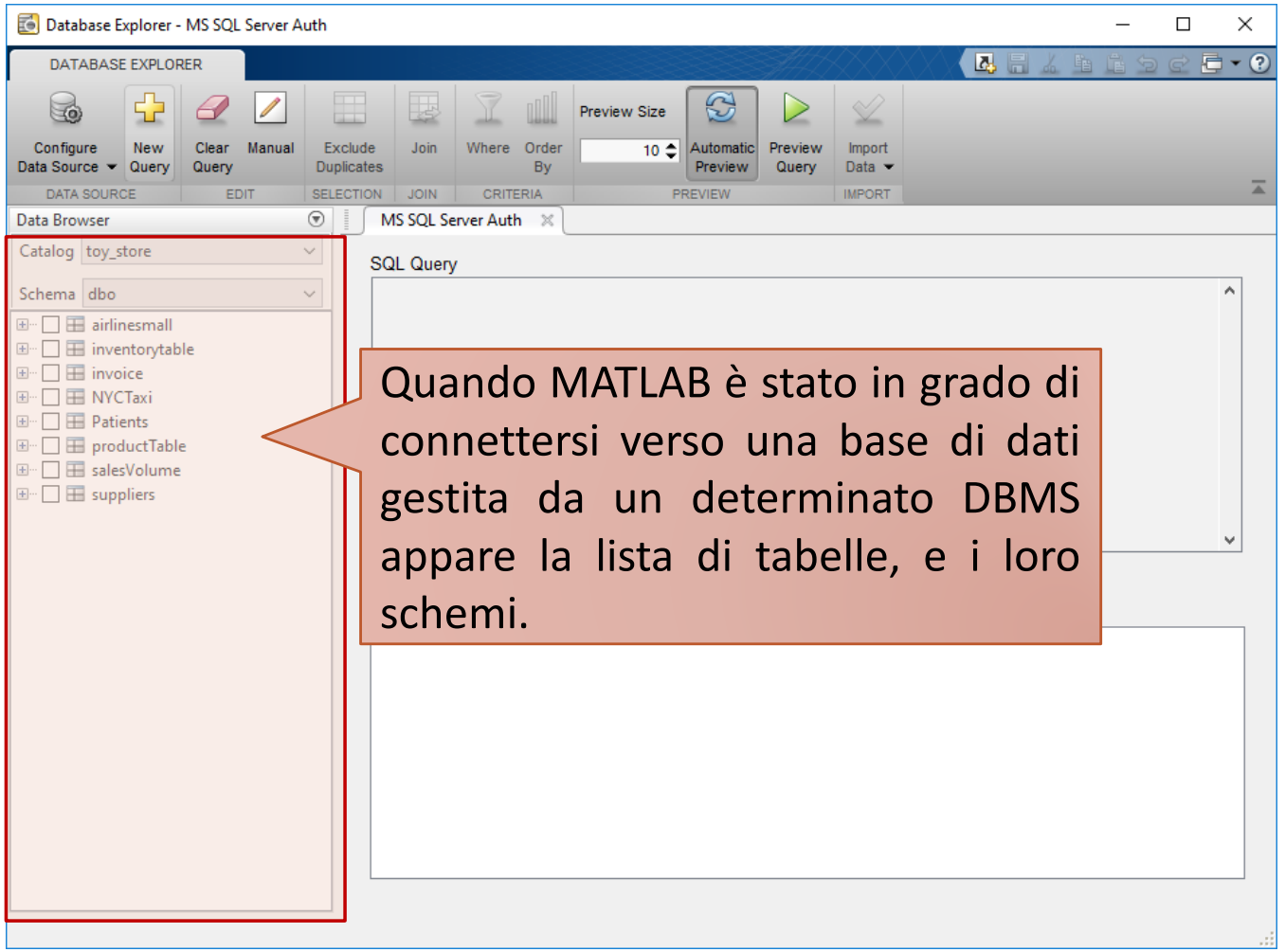
- digitando nella finestra dei comandi il comando databaseExplorer,
- selezionandola nella barra degli strumenti.



Basi di Dati e MATLAB (3/5)

Questa app è invocabile:

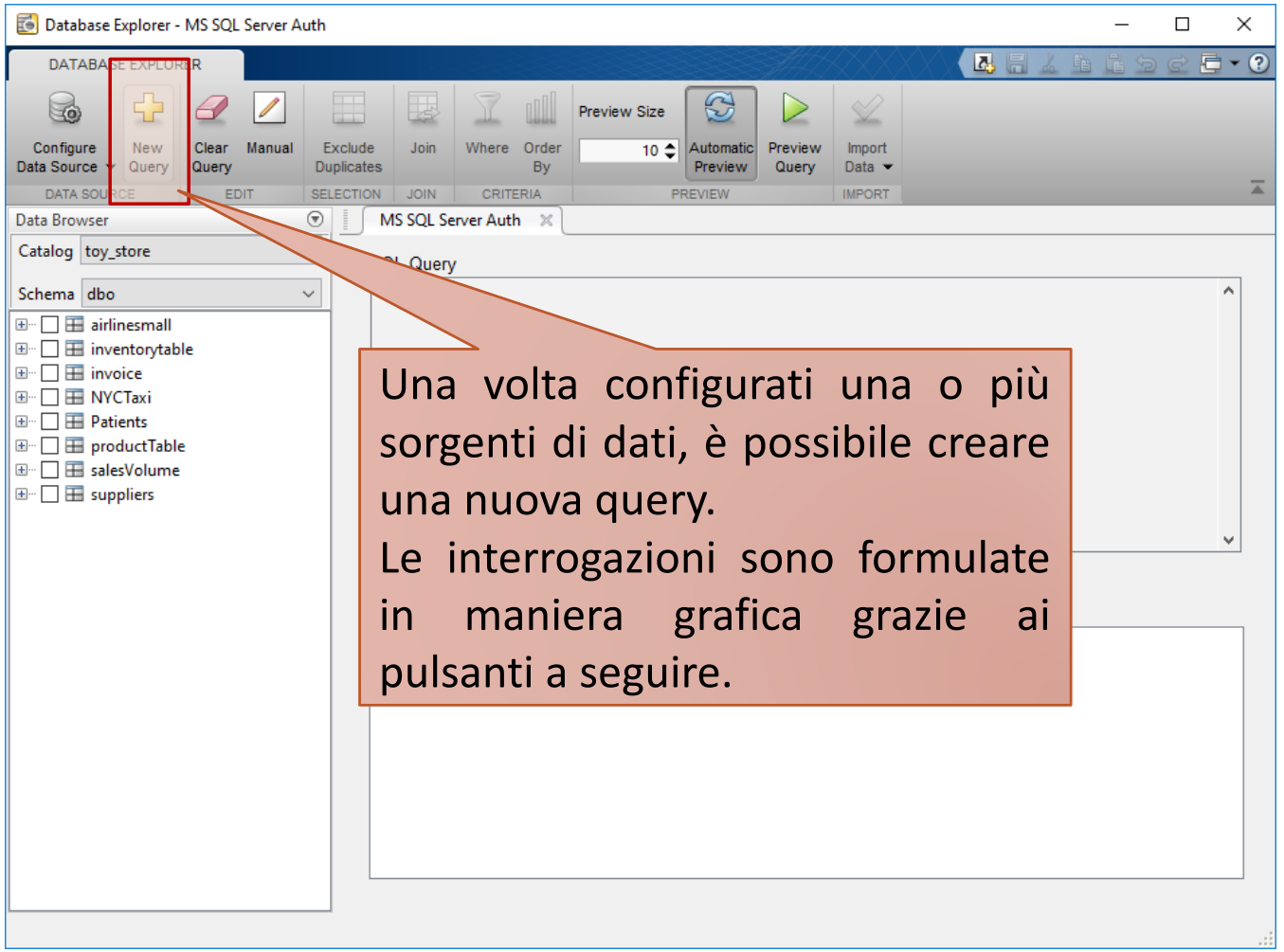
- digitando nella finestra dei comandi il comando databaseExplorer,
- selezionandola nella barra degli strumenti.



Basi di Dati e MATLAB (3/5)

Questa app è invocabile:

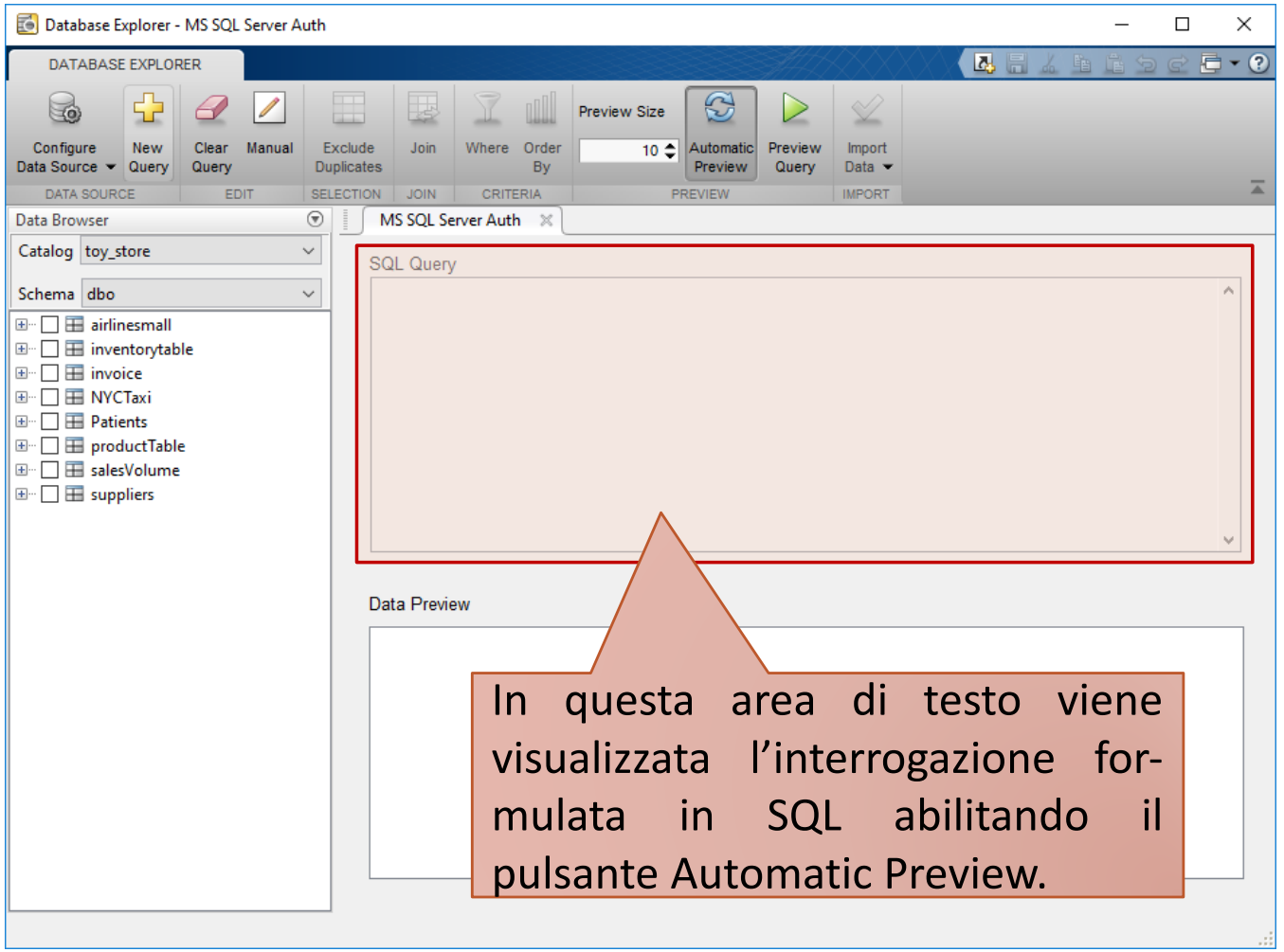
- digitando nella finestra dei comandi il comando databaseExplorer,
- selezionandola nella barra degli strumenti.



Basi di Dati e MATLAB (3/5)

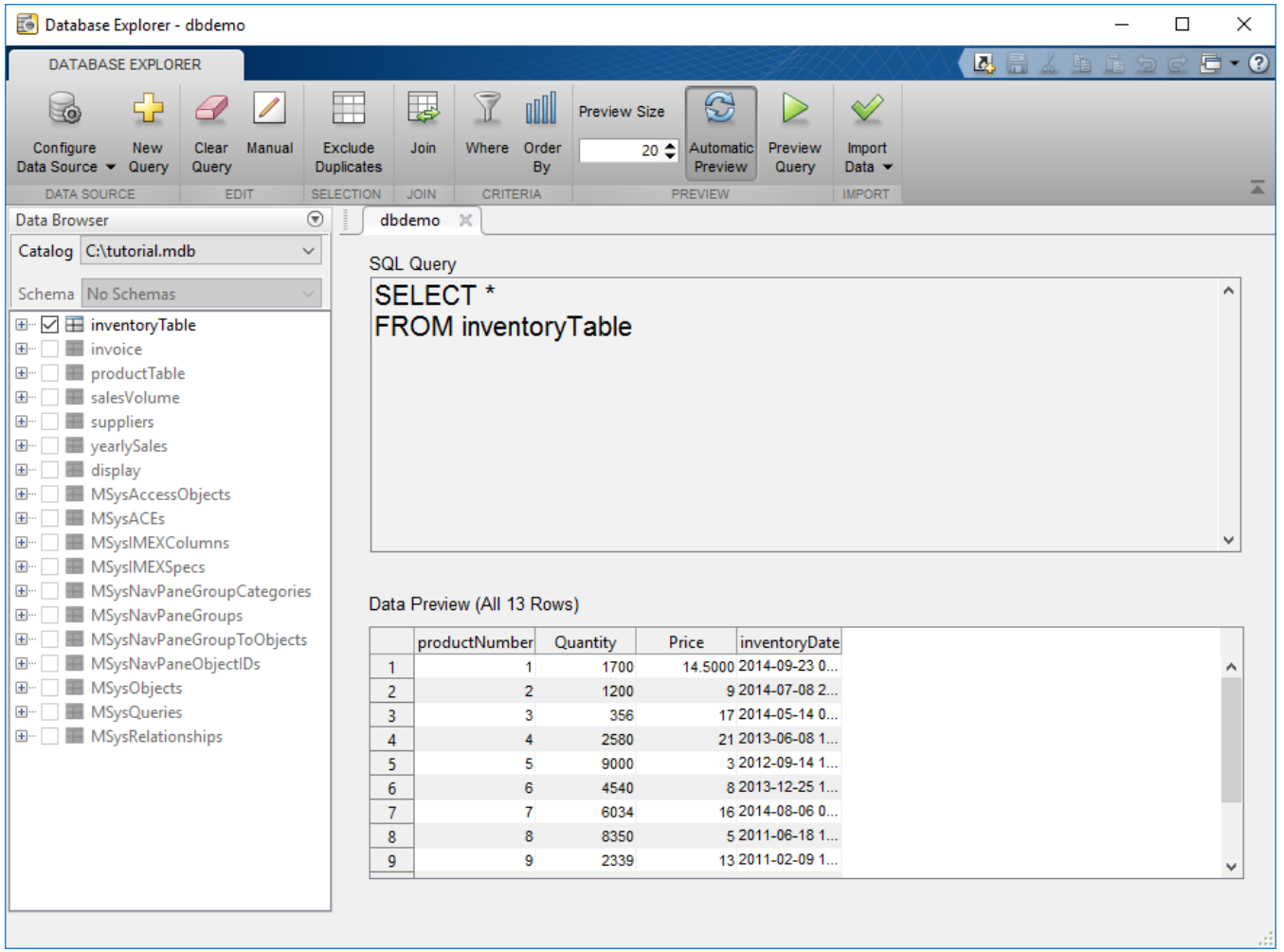
Questa app è invocabile:

- digitando nella finestra dei comandi il comando databaseExplorer,
- selezionandola nella barra degli strumenti.



Basi di Dati e MATLAB (4/5)

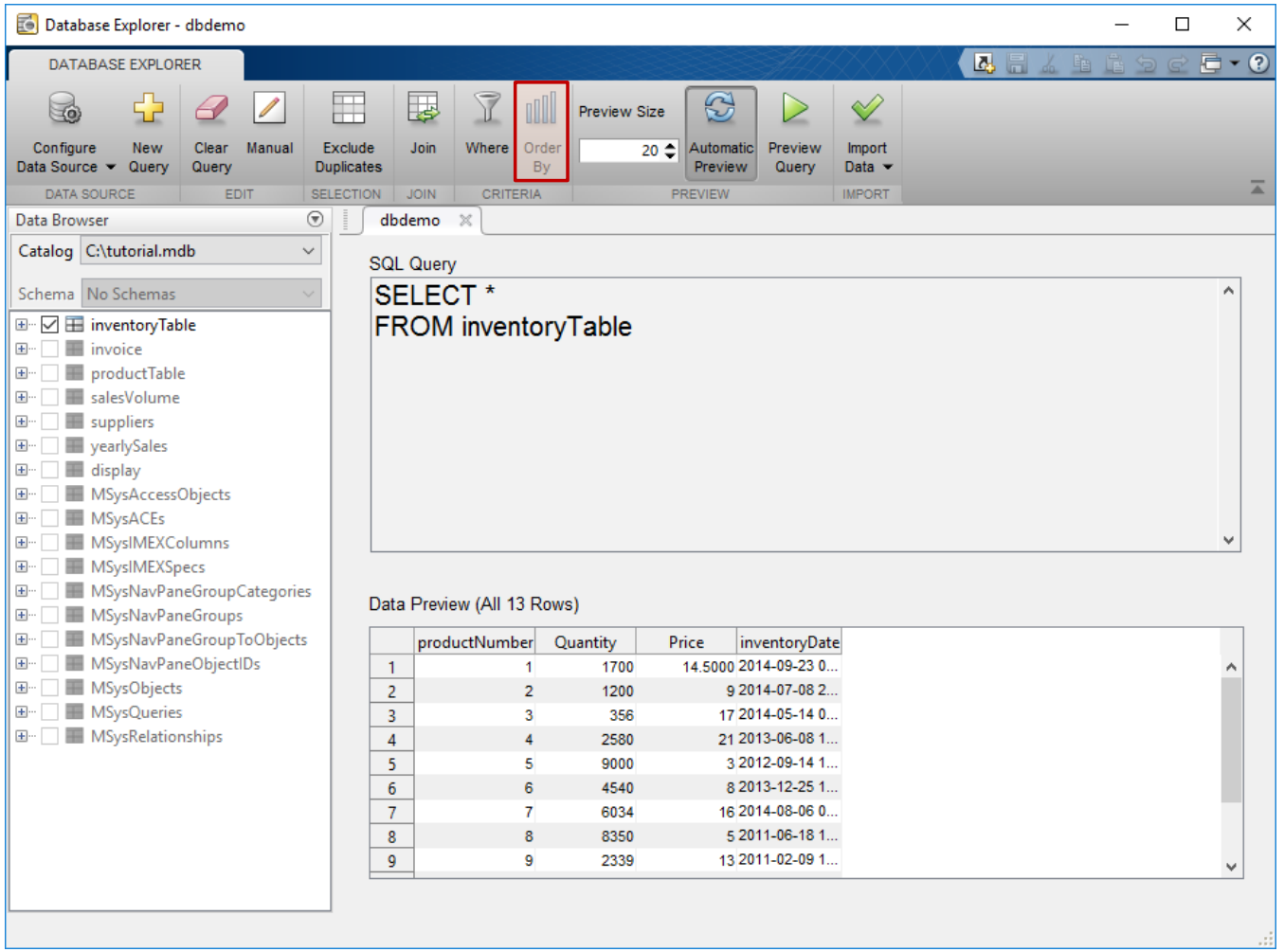
Ad esempio, selezionando una tabella otteniamo quanto segue.



Basi di Dati e MATLAB (4/5)

Ad esempio, selezionando una tabella otteniamo quanto segue.

Possiamo ordinare i risultati per mezzo di Order By.

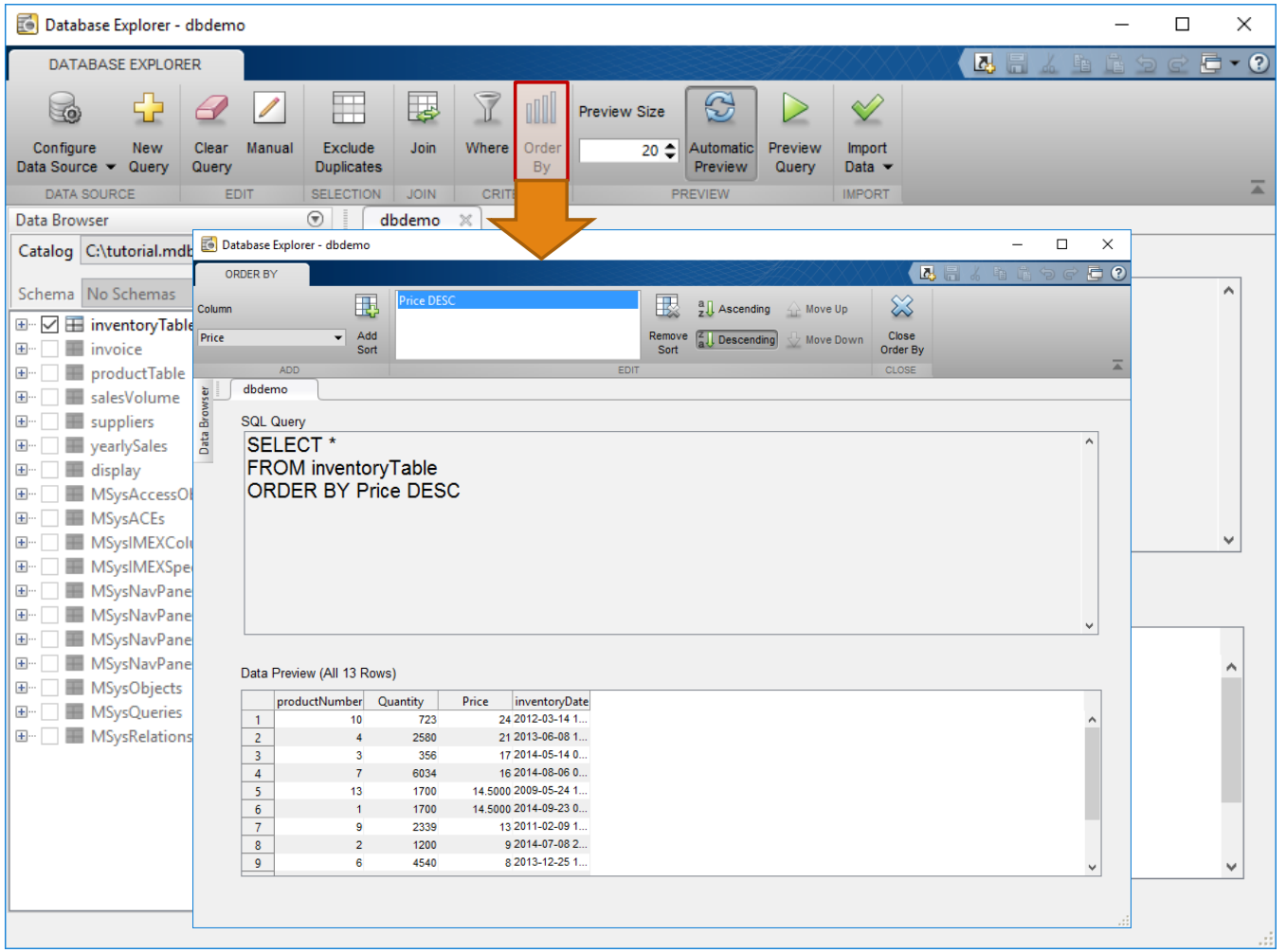


Basi di Dati e MATLAB (4/5)

Ad esempio, selezionando una tabella otteniamo quanto segue.

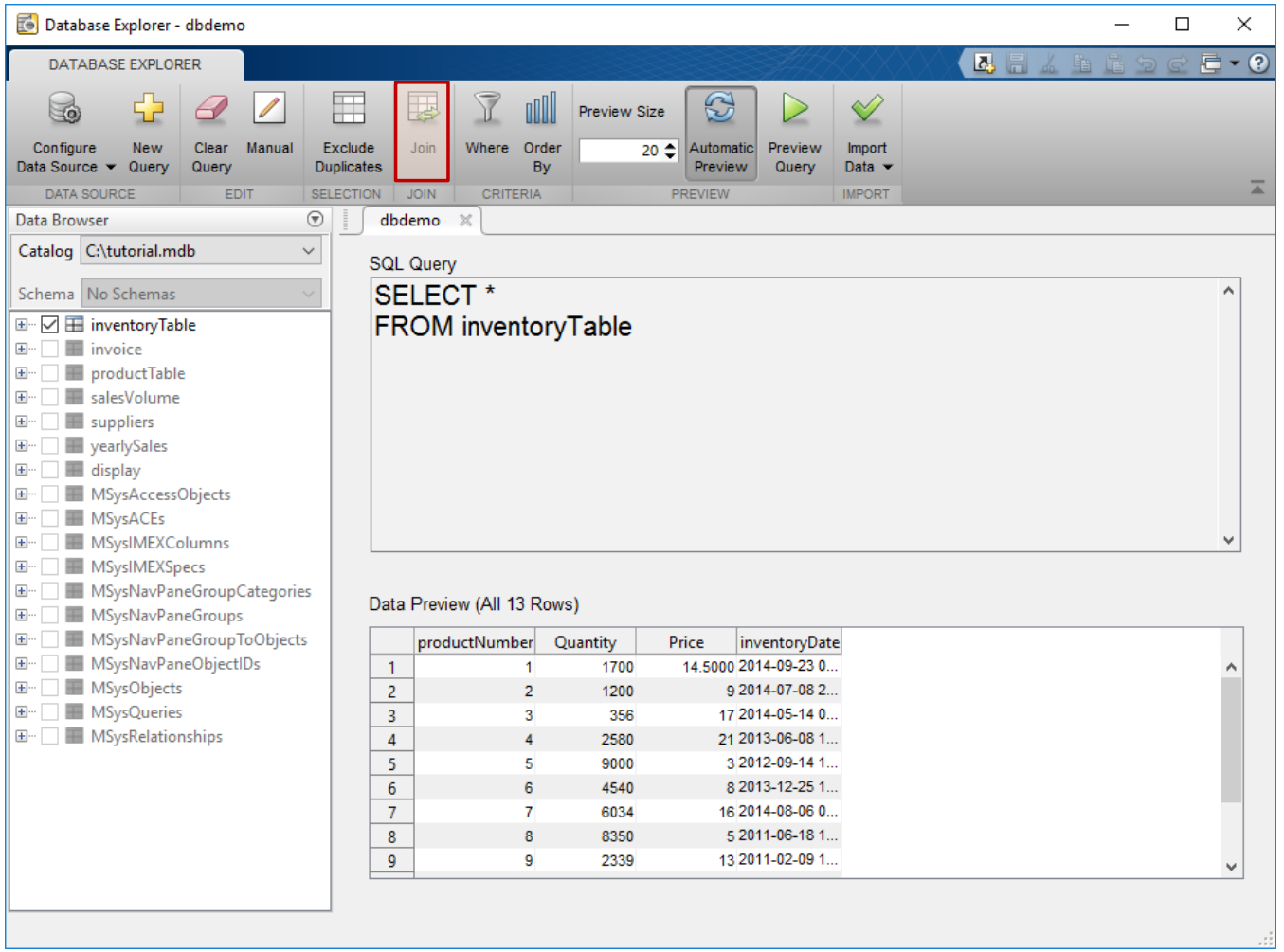
Possiamo ordinare i risultati per mezzo di Order By.

Possiamo selezionare la colonna e il tipo di ordinamento.



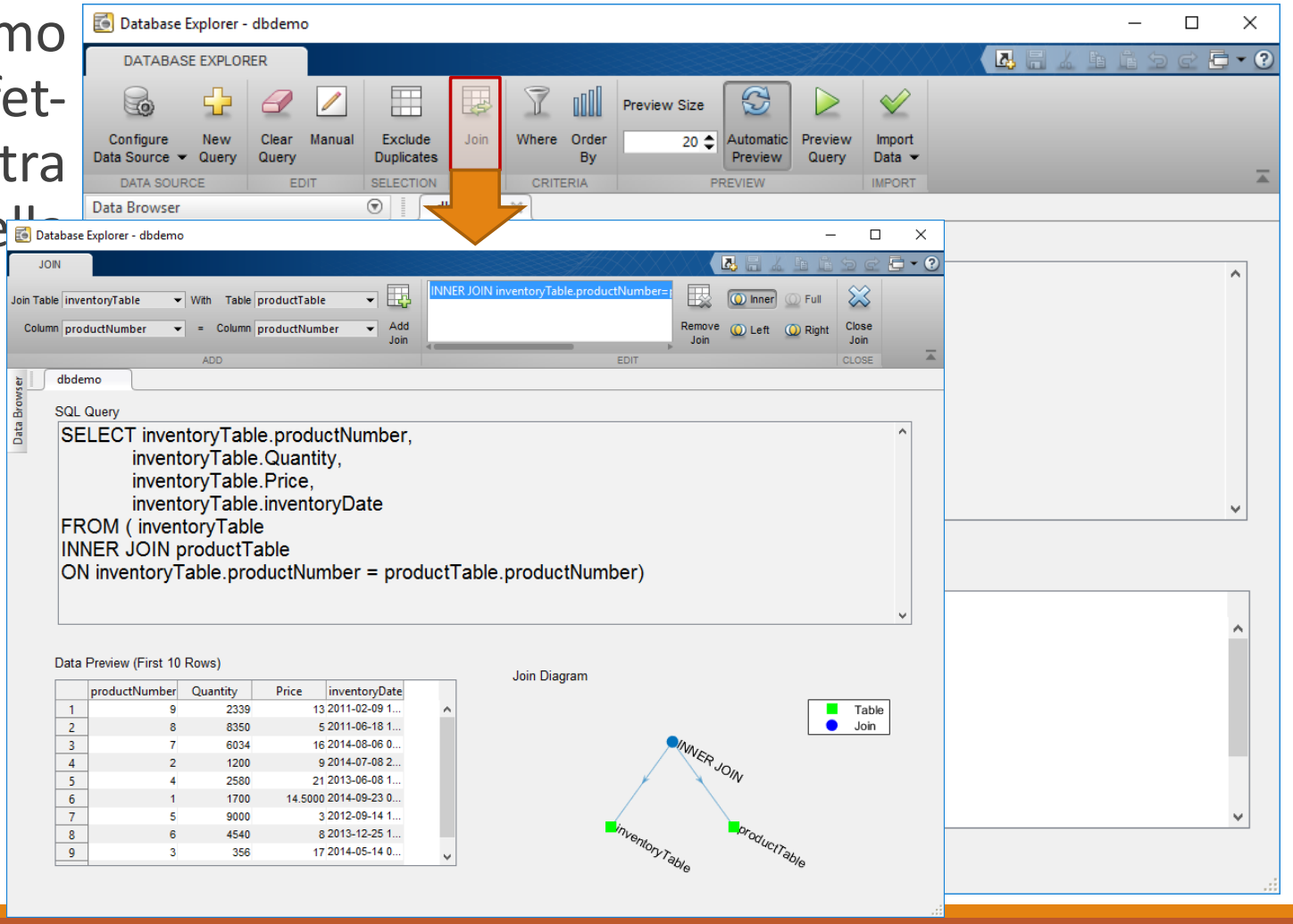
Basi di Dati e MATLAB (5/5)

Oppure possiamo scegliere di effettuare un Join tra delle tabelle della base di dati.



Basi di Dati e MATLAB (5/5)

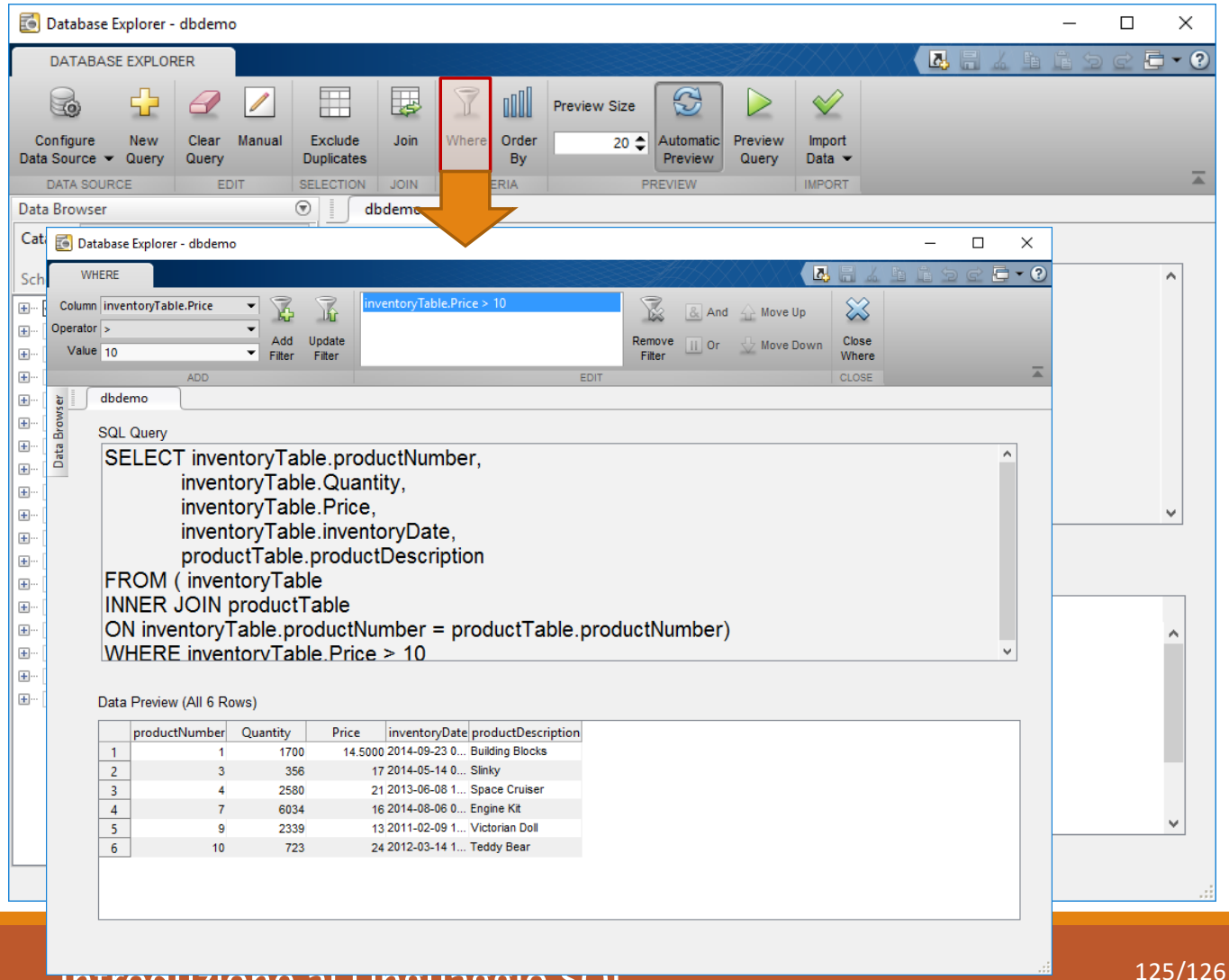
Oppure possiamo scegliere di effettuare un Join tra delle tabelle della base di dati.



Basi di Dati e MATLAB (5/5)

Oppure possiamo scegliere di effettuare un Join tra delle tabelle della base di dati.

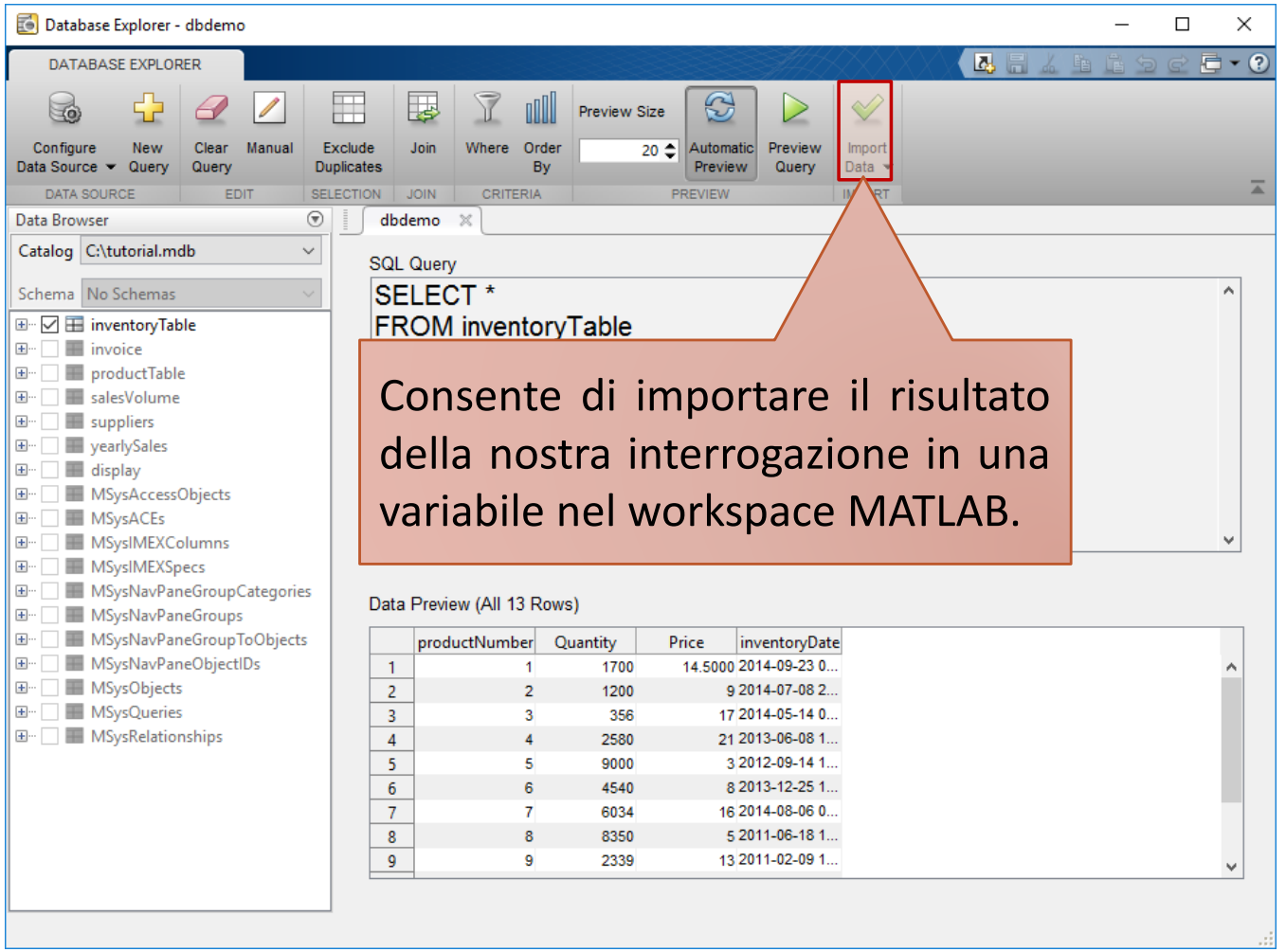
Successivamente possiamo chiudere con Close Join e pensare di inserire una condizione nella clausola Where.



Basi di Dati e MATLAB (5/5)

Oppure possiamo scegliere di effettuare un Join tra delle tabelle della base di dati.

Successivamente possiamo chiudere con Close Join e pensare di inserire una condizione nella clausola Where.



Riferimenti

- Capitolo 2
  - Paragrafo 3 [**Il Modello Relazione in SQL**].
- Capitolo 5
  - Paragrafi 4 [**Operazioni Relazionali**], 5 [**Ridenominazione ed Uso di Variabili**], 6 [**Funzioni di Aggregazione e Clausole di Raggruppamento**], 7 [**Query Insiemistiche e Query Nidificate in SQL**], 9 [**Sintassi delle Operazioni in SQL**].
- https://it.mathworks.com/products/database.html
- https://it.mathworks.com/help/database/ug/database.html