

MySQL e MySQL Workbench

Prof. Giovanni Micale

Cosa è MySQL

- Il più popolare Database Management System relazionale open source basato su SQL (SQL:2003 o SQL3)
- Supportato da Oracle Corporation
- In grado di lavorare in sistemi client/server o embedded
- Possiede interfacce per diversi linguaggi, compreso un driver ODBC, due driver Java, un driver per Mono e .NET ed una libreria per Python.
- Di base è composto da un client con interfaccia a riga di comando e un server, entrambi disponibili sia per sistemi Unix o Unix-like come GNU/Linux che per Windows, anche se prevale un suo utilizzo in ambito Unix.
- Oggi l'ultima versione disponibile è la 9.5.0

Storia di MySQL

- Creato dalla società MySQLAB sin dal 1979 soltanto dal 1996 supporta anche SQL.
- Sun Microsystem nel 2008 rileva la società per 1 miliardo di dollari



- Nel 2010 Oracle acquista Sun per 7,5 miliardi di dollari possedendo così anche MySQL.

ORACLE

Engine

- MySQL mette a disposizione diversi **tipi di tabella (“storage engine”)** per la memorizzazione dei dati.
- Ognuno presenta proprietà e caratteristiche differenti.
- Esiste una **API** che si può utilizzare per creare nuovi tipi di tabella che si possono installare senza necessità di riavviare il server.
- Due sono i sistemi principali:
 - **Transazionali**: sono più sicuri, permettono di recuperare i dati anche in caso di crash, e consentono di effettuare modifiche tutte insieme;
 - **Non transazionali**: sono più veloci, occupano meno spazio su disco e minor richiesta di memoria.

Engine: Mylsam

- **MyISAM** era lo storage engine di default dal MySQL 3.23 fino al MySQL 5.4.
- **MyISAM** utilizza la struttura **ISAM** e deriva da un tipo più vecchio, oggi non più utilizzato, che si chiamava appunto **ISAM**.
- È un motore di immagazzinamento dei dati estremamente veloce e richiede poche risorse, sia in termini di memoria RAM, sia in termini di spazio su disco.
- Il suo limite principale rispetto ad alcuni altri SE consiste nel mancato supporto delle transazioni e alle foreign key.
- Ogni tabella MyISAM è memorizzata all'interno del disco con tre file:
 - un file **.frm** che contiene la definizione della tabella,
 - un file **.MYD** per i dati
 - un file **.MYI** per gli indici

Engine: InnoDB

- **InnoDB** è un motore per il salvataggio di dati per MySQL, fornito in tutte le sue distribuzioni (Default dalla versione 5.5).
- La sua caratteristica principale è quella di supportare le transazioni di tipo **ACID**.

Engine: InnoDB

- **InnoDB** mette a disposizione le seguenti funzionalità:
 - transazioni SQL con savepoint e transazioni XA;
 - lock a livello di record;
 - foreign key;
 - integrità referenziale;
 - colonne AUTOINCREMENT;
 - tablespace.
- InnoDB offre delle ottime performance in termini di prestazioni e utilizzo della CPU specialmente quando si ha a che fare con una grande quantità di dati.
- InnoDB può interagire tranquillamente con tutti gli altri tipi di tabelle in MySQL.

Engine: InnoDB

- Le tabelle InnoDB sono soggette alle seguenti limitazioni:
 - Non è possibile creare più di **1000 colonne** per tabella;
 - Su alcuni sistemi operativi le **dimensioni del tablespace** non possono superare i **2 Gb**;
 - La grandezza di tutti i **file di log** di InnoDB deve essere inferiore ai **4 Gb**;
 - La grandezza minima di un tablespace è di **10 MB**;
 - Non possono essere creati indici di tipo **FULLTEXT** con **MySQL 5.5 o precedente**;
 - Le **SELECT COUNT(*)** su tabelle di grandi dimensioni possono essere molto lente.

Download di MySQL

- Tutti i software e i tool di MySQL sono disponibili all'indirizzo <http://dev.mysql.com/downloads>
- Per Windows, fino alla versione 8.0.44, era presente anche un installer unico per tutti i tool della suite MySQL

④ MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL NDB Cluster
- MySQL Router
- MySQL Shell
- MySQL Operator
- MySQL NDB Operator
- MySQL Workbench
- MySQL Installer for Windows
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/.NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP
- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

**MySQL Enterprise Edition
for Developers**

Free for learning, developing,
and prototyping.



Download Now »

Download alternativo di MySQL

- Alternativamente si può installare un pacchetto che include web server e DBMS :
 - **EasyPhp**
 - **XAMPP**

Download di MySQL (Windows)

MySQL Community Downloads

- [MySQL Yum Repository](#)
- [MySQL APT Repository](#)
- [MySQL SUSE Repository](#)
- [MySQL Community Server](#)
- [MySQL NDB Cluster](#)
- [MySQL Router](#)
- [MySQL Shell](#)
- [MySQL Operator](#)
- [MySQL NDB Operator](#)
- [MySQL Workbench](#)
- [MySQL Installer for Windows](#)
- [C API \(libmysqlclient\)](#)
- [Connector/C++](#)
- [Connector/J](#)
- [Connector/.NET](#)
- [Connector/Node.js](#)
- [Connector/ODBC](#)
- [Connector/Python](#)
- [MySQL Native Driver for PHP](#)
- [MySQL Benchmark Tool](#)
- [Time zone description tables](#)
- [Download Archives](#)

**MySQL Enterprise Edition
for Developers**

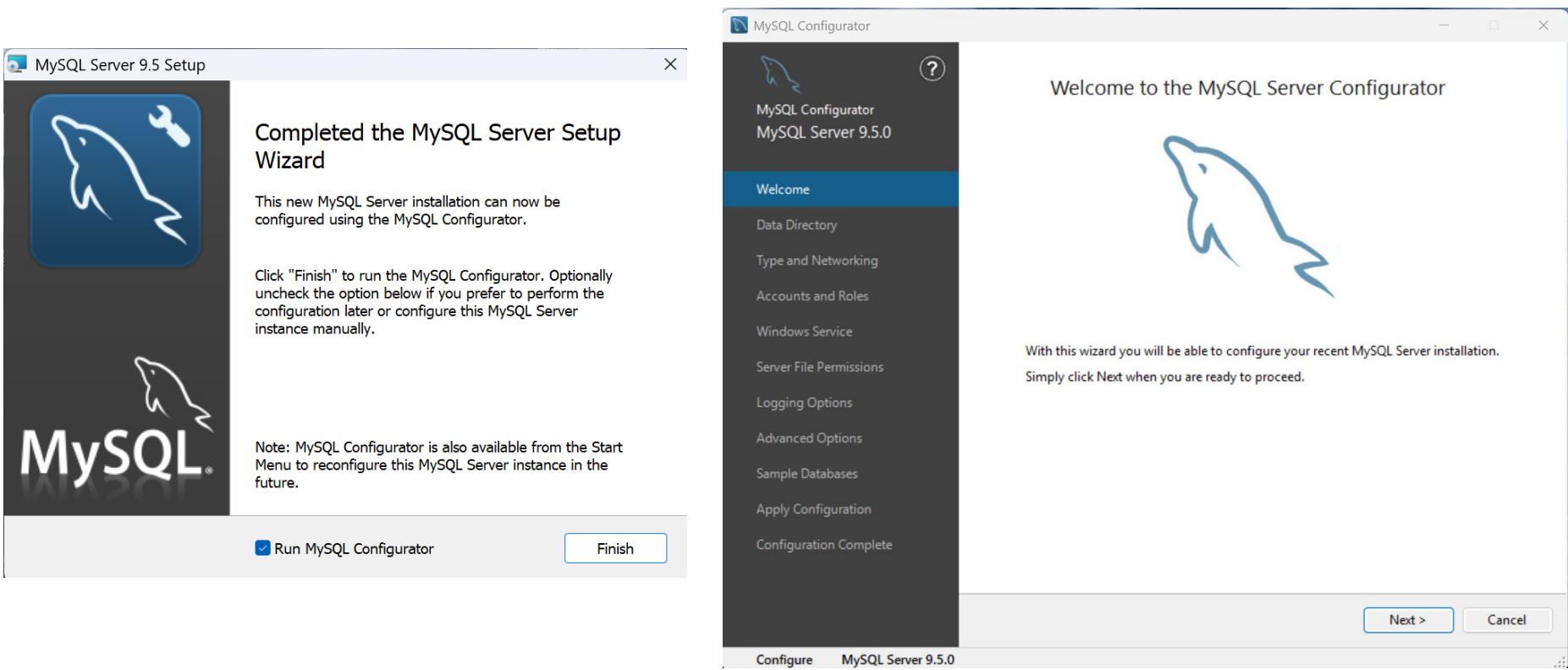
Free for learning, developing,
and prototyping.



Download Now »

Installazione di MySQL (Windows)

- Una volta scaricato l'installer, lo si esegue e si procede con l'installazione.
- Al termine è possibile lanciare MySQL Configurator, che consente la configurazione iniziale di MySQL e la creazione dell'utente root con le sue credenziali.



Impostazioni di rete

The screenshot shows the MySQL Configurator interface for MySQL Server 9.5.0. The left sidebar lists various configuration categories: Welcome, Data Directory, Type and Networking (which is selected and highlighted in blue), Accounts and Roles, Windows Service, Server File Permissions, Sample Databases, Apply Configuration, and Configuration Complete.

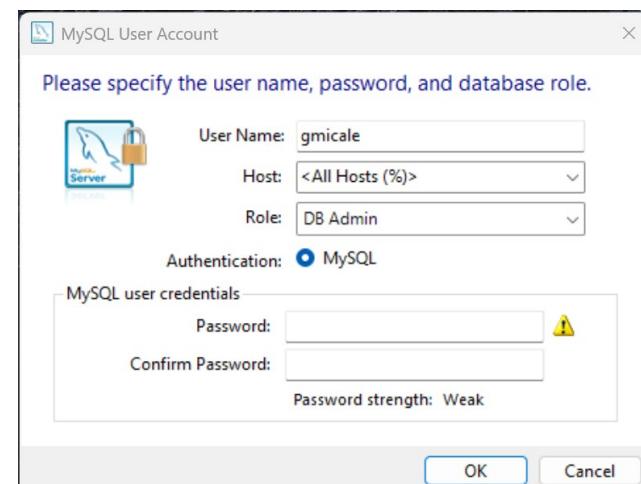
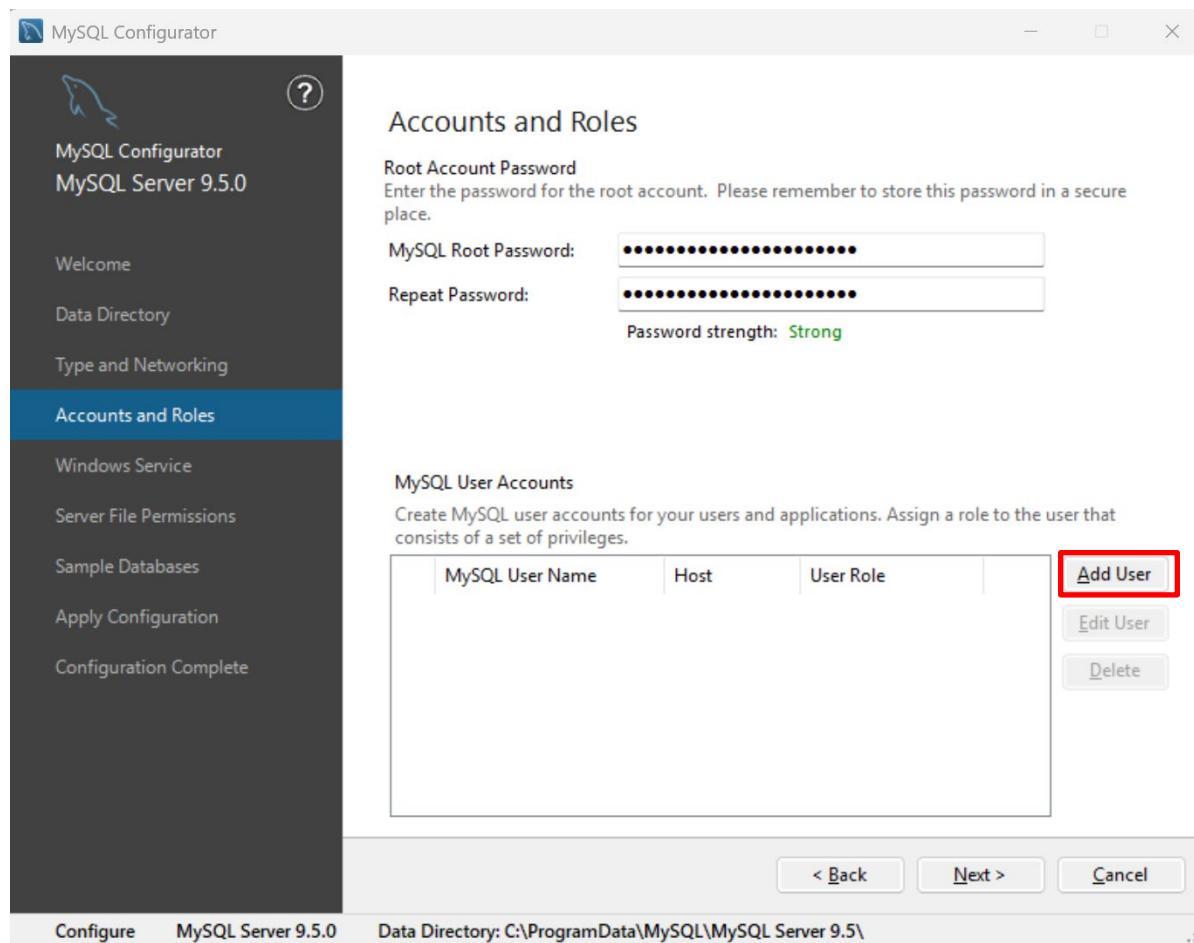
The main content area is titled "Type and Networking". Under "Server Configuration Type", it says: "Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance." A dropdown menu shows "Config Type: Development Computer".

The "Connectivity" section includes fields for "Port:" (set to 3306) and "X Protocol Port:" (set to 33060). It also contains checkboxes for "TCP/IP" (checked), "Open Windows Firewall ports for network access" (checked), "Named Pipe" (unchecked), "Pipe Name:" (set to MYSQL), "Shared Memory" (unchecked), and "Memory Name:" (set to MYSQL).

The "Advanced Configuration" section contains a checkbox for "Show Advanced and Logging Options" (unchecked).

At the bottom, there are navigation buttons: "< Back", "Next >" (highlighted in blue), and "Cancel". The status bar at the bottom displays "Configure MySQL Server 9.5.0" and "Data Directory: C:\ProgramData\MySQL\MySQL Server 9.5\".

Creazione account root e utenti



Definizione del servizio di sistema

MySQL Configurator

MySQL Configurator
MySQL Server 9.5.0

Welcome
Data Directory
Type and Networking
Accounts and Roles
Windows Service
Server File Permissions
Sample Databases
Apply Configuration
Configuration Complete

Windows Service

Configure MySQL Server as a Windows Service

Windows Service Details
Please specify a Windows Service name to be used for this MySQL Server instance.
A unique name is required for each instance.

Windows Service Name:

Start the MySQL Server at System Startup

Run Windows Service as ...
The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

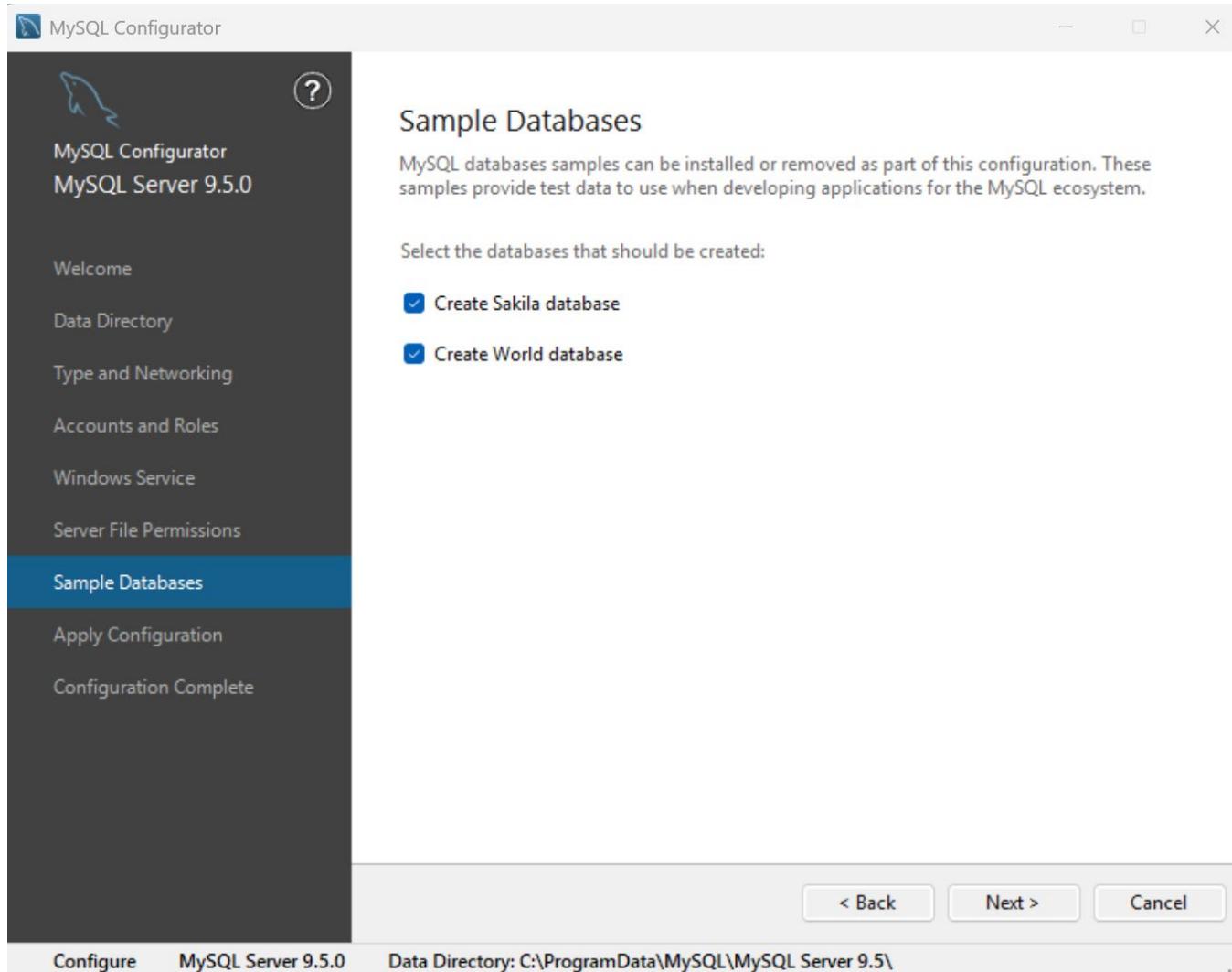
Standard System Account
Recommended for most scenarios.

Custom User
An existing user account can be selected for advanced scenarios.

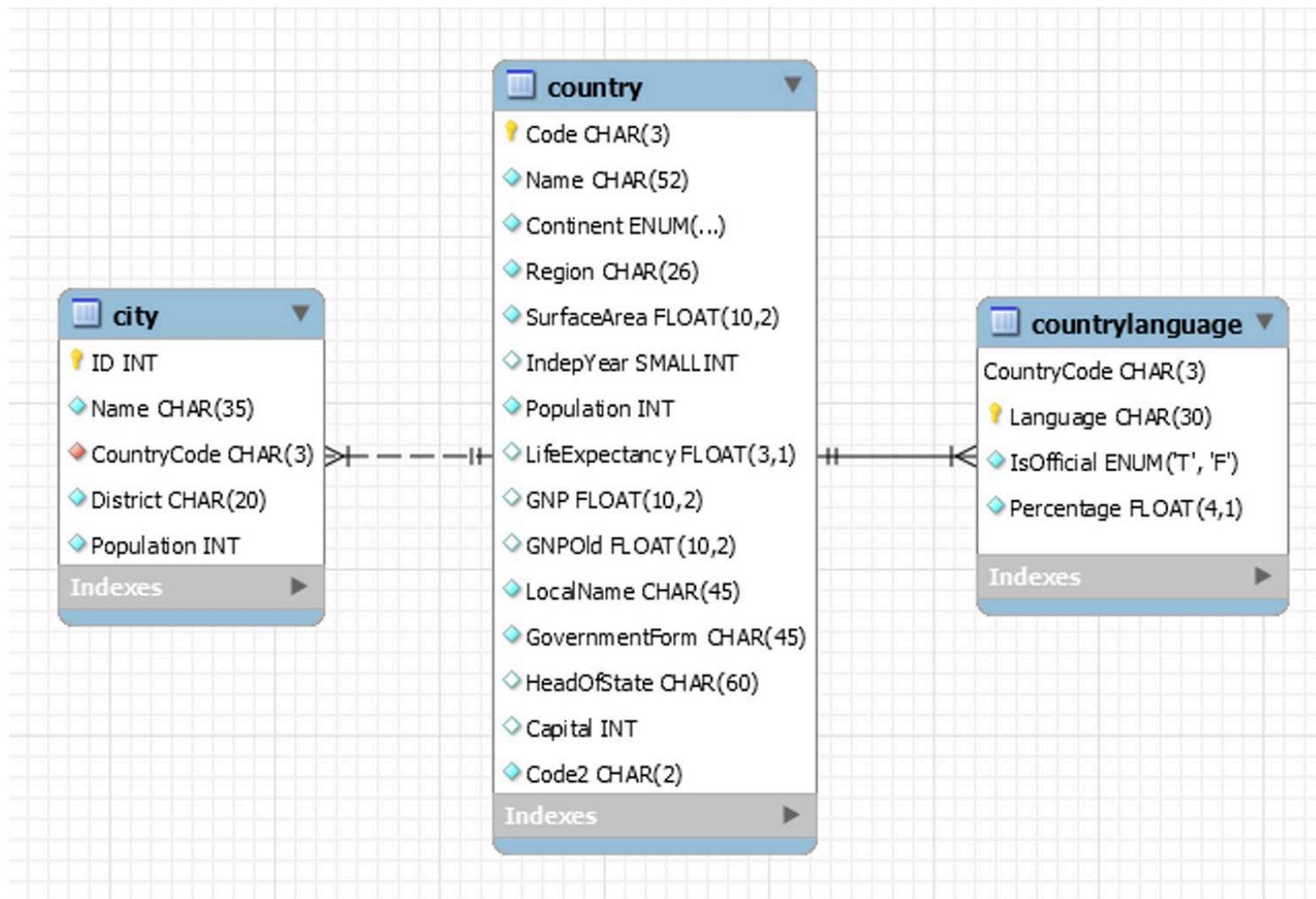
< Back Next > Cancel

Configure MySQL Server 9.5.0 Data Directory: C:\ProgramData\MySQL\MySQL Server 9.5\

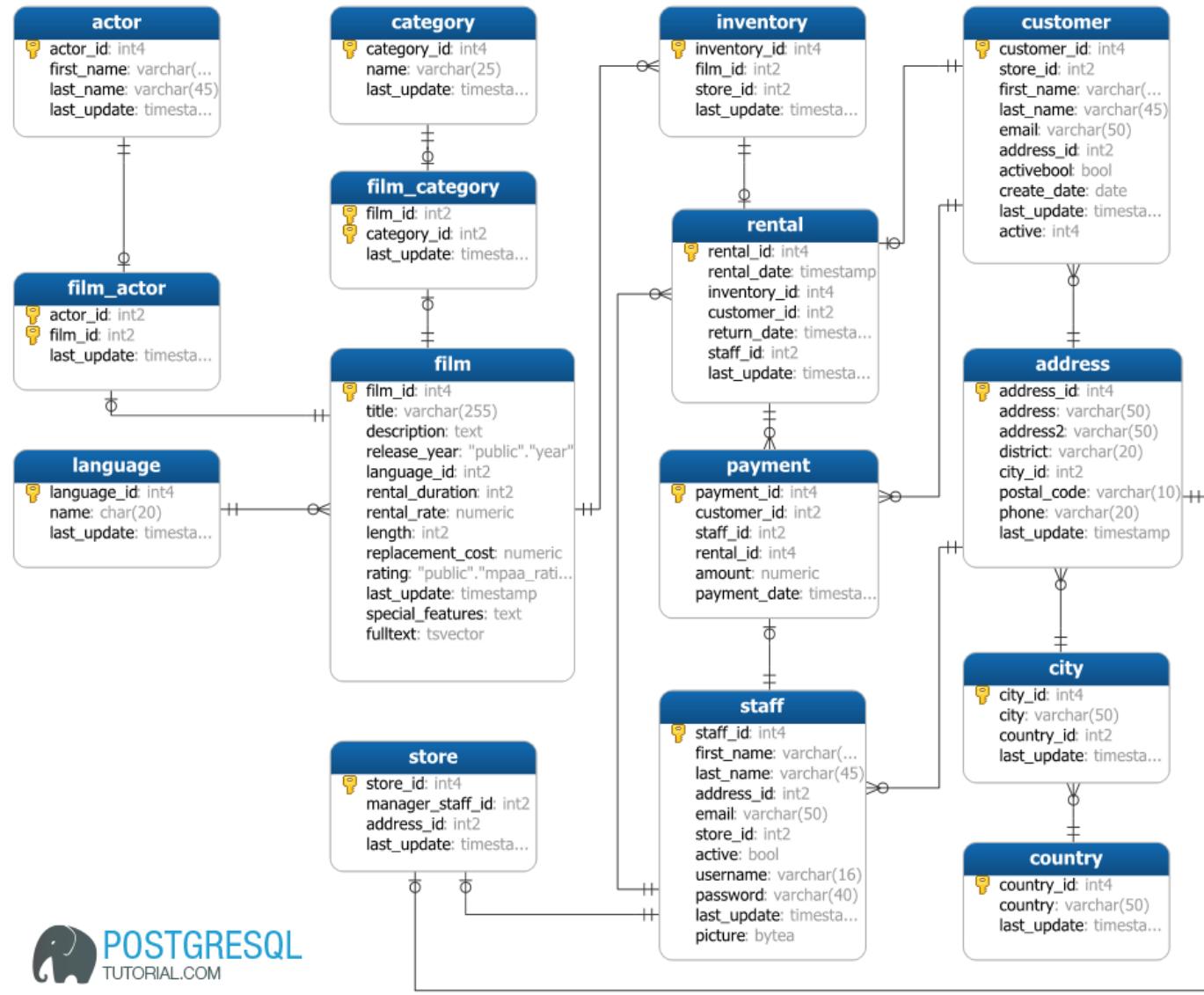
Installazione (su richiesta) di database di esempio



World Database



Sakila Database (DVD Rental Stores)



Aggiunta di mysql all'environment di sistema

- Per eseguire MySQL da riga di comando nel prompt dei comandi di Windows occorre aggiungere la cartella contenente i comandi client MySQL all'environment di sistema (variabile PATH)
- Tipicamente, sotto Windows, la cartella è del tipo «[C:/Program Files/MySQL/MySQL Server <versione>/bin](#)»

MySQL: Comandi base

Principali client MySQL

- **mysql**: il client ufficiale per interagire con i database.
- **mysqladmin**, per lo svolgimento di ogni genere di operazione di configurazione del server;
- **mysqlcheck**, che si occupa della manutenzione delle tabelle;
- **mysqldump**, indispensabile per il backup.
- **mysqlimport**, che permette di importare tabelle nei database;
- **mysqlshow**, che fornisce informazioni su database, tabelle, indici e molto altro.

Connessione/disconnessione al server MySQL

- Per connettersi al server è necessario fornire login e password

```
shell> mysql -h host -u user -p  
Enter password: *****
```

- *host* and *user* rappresentano:
 - l'hostname dove risiede MySQL;
 - lo username di un utente che possiede un account sul server;
- *-p* specifica al server la richiesta della password all'utente.
- Se l'host è il localhost non è necessario specificare il parametro *-h*

Esempio

```
C:\>mysql -u apulvirenti -p
```

```
Enter password: *****
```

```
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 46
```

```
Server version: 5.7.17 MySQL Community Server (GPL)
```

```
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql>
```

Account e Privilegi

- Come creare un utente:

CREATE USER ‘name’@‘host’ **IDENTIFIED BY** ‘PASSWORD’;

- Come assegnare privilegi ad un utente:

GRANT privilege,... **ON** *| ‘db’. *| ‘db’. ‘table’ **TO** ‘username’@‘host’, ...;

- ‘privilege’ può essere uno dei seguenti valori:
 - **ALL**
 - **USAGE**
 - **SELECT, INSERT, UPDATE, DELETE**
 - **CREATE, ALTER, INDEX, DROP, CREATE VIEW, TRIGGER**

Esempio

```
C:\>mysql -u root -p  
Enter password: *****  
  
mysql> CREATE USER 'user'@'localhost' IDENTIFIED BY 'password';  
mysql> GRANT ALL ON nomeDB.* to 'user'@'localhost';
```

Account e Privilegi

- Rimuovere un utente:

DROP USER ‘*name*’@‘*host*’,...;

- Rimuovere i privilegi di un utente:

REVOKE *privilege*,... **ON** *| ‘*db*’.*| ‘*db*’.’*table*’ **FROM** ‘*username*’@‘*host*’,...;

- ‘**privilege**’ può essere uno dei seguenti valori:
 - **ALL PRIVILEGES**
 - **USAGE**
 - **SELECT, INSERT, UPDATE, DELETE**
 - **CREATE, ALTER, INDEX, DROP, CREATE VIEW, TRIGGER**

Comando SHOW

- SHOW ha diverse opzioni e da informazioni riguardo ai database alle tabella, collone, indici, ecc. Da anche informazioni riguardo il server.

```
SHOW [FULL] COLUMNS FROM tbl_name [FROM db_name] [LIKE 'pattern']  
SHOW CREATE DATABASE db_name  
SHOW CREATE TABLE tbl_name  
SHOW DATABASES [LIKE 'pattern']  
SHOW [STORAGE] ENGINES  
SHOW ERRORS [LIMIT [offset,] row_count]  
SHOW GRANTS FOR user  
SHOW INDEX FROM tbl_name [FROM db_name]  
SHOW INNODB STATUS  
SHOW [BDB] LOGS  
SHOW PRIVILEGES  
SHOW [FULL] PROCESSLIST  
SHOW STATUS [LIKE 'pattern']  
SHOW TABLE STATUS [FROM db_name] [LIKE 'pattern']  
SHOW [OPEN] TABLES [FROM db_name] [LIKE 'pattern']  
SHOW [GLOBAL | SESSION] VARIABLES [LIKE 'pattern']  
SHOW WARNINGS [LIMIT [offset,] row_count]
```

Lista dei DB presenti

```
mysql> SHOW databases;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sakila         |
| sys            |
| world          |
+-----+
6 rows in set (0.012 sec)
```

Lista delle tabelle presenti in un DB

```
mysql> SHOW TABLES FROM sakila;
+-----+
| Tables_in_sakila      |
+-----+
| actor                  |
| actor_info              |
| address                |
| category               |
| city                   |
| country                |
| customer               |
| customer_list          |
| film                   |
| film_actor              |
| film_category           |
| film_list               |
| film_text               |
| inventory              |
| language                |
| nicer_but_slower_film_list |
| payment                |
| rental                 |
| sales_by_film_category |
| sales_by_store           |
| staff                  |
| staff_list              |
| store                  |
+-----+
23 rows in set (0.080 sec)
```

Lista delle colonne di una tabella

```
mysql> SHOW COLUMNS FROM sakila.actor;
+-----+-----+-----+-----+-----+
| Field | Type           | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| actor_id | smallint unsigned | NO   | PRI | NULL    | auto_increment |
| first_name | varchar(45) | NO   |     | NULL    | |
| last_name | varchar(45) | NO   | MUL | NULL    | |
| last_update | timestamp      | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_TIMESTAMP |
+-----+-----+-----+-----+-----+
4 rows in set (0.046 sec)
```

Alternativa: DESCRIBE *nomeTabella*

Lista degli indici di una tabella

```
mysql> SHOW INDEX FROM sakila.actor;
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
-----+-----+
| Table | Non_unique | Key_name           | Seq_in_index | Column_name | Collation
| Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
Visible | Expression |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
-----+-----+
| actor |      0 | PRIMARY          |           |       1 | actor_id    | A
200 |    NULL |    NULL |        | BTREE     |       | YES        |
NULL   |           |
| actor |      1 | idx_actor_last_name |           |       1 | last_name   | A
121 |    NULL |    NULL |        | BTREE     |       | YES        |
NULL   |           |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
-----+-----+
2 rows in set (0.074 sec)
```

Comando USE

- Consente di referenziare direttamente una tabella di un DB senza dover scrivere «*nomedb.nometabella*»
- Sintassi: USE *nomedb*

```
mysql> USE sakila;
Database changed
mysql> SHOW columns FROM customer;
+-----+-----+-----+-----+
| Field      | Type           | Null | Key | Default       | Extra
+-----+-----+-----+-----+
| customer_id | smallint unsigned | NO   | PRI | NULL          | auto_increment
| store_id    | tinyint unsigned  | NO   | MUL | NULL          |
| first_name  | varchar(45)      | NO   |      | NULL          |
| last_name   | varchar(45)      | NO   | MUL | NULL          |
| email       | varchar(50)       | YES  |      | NULL          |
| address_id  | smallint unsigned | NO   | MUL | NULL          |
| active      | tinyint(1)        | NO   |      | 1             |
| create_date | datetime         | NO   |      | NULL          |
| last_update | timestamp        | YES  |      | CURRENT_TIMESTAMP | DEFAULT_GENERATED on
update CURRENT_TIMESTAMP |
+-----+-----+-----+-----+
9 rows in set (0.013 sec)

mysql> SELECT count(*) FROM actor;
+-----+
| count(*) |
+-----+
|     200 |
+-----+
1 row in set (0.010 sec)
```

MySQL: Tipi di dato

Tipi di Dati: Numerici

Tipo	Byte	Min. Value (Signed/Unsigned)	Max. Value (Signed/Unsigned)
TINYINT	1	-128/0	127/255
SMALLINT	2	-32.768/0	32.767/65535
MEDIUMINT	3	-8.388.608/0	8.388.607/16.777.215
INT	4	-2.147.483.648/0	2.147.483.647/4.294.967.295
BIGINT	8	-9.223.372.036.854.775.808/0	9.223.372.036.854.775.807 /18.446.744.073.709.551.615
FLOAT	4	+/-1.175494351E-38	+/-3.402823466E+38
DOUBLE	8	+/-2.225073858507201E-308	+/-1.7976931348623157E+308

Tipi di Dati: Numerici

Tipo	Byte	Min. Value (Signed/Unsigned)	Max. Value (Signed/Unsigned)
INTEGER			Equivale ad INT
DOUBLE PRECISION			Equivale a DOUBLE
REAL			Equivale a DOUBLE
DECIMAL(M[,D])	M+2		Tutti i numeri di M cifre di cui D decimali.
NUMERIC(M[,D])			Equivale a DECIMAL
BIT(M)	M		una sequenza di M bit (MySQL 5.5)

Tipi di Dati: Date e Tempi

Tipo	Byte	Range
DATE	3	dal '01/01/1000' al '31/12/9999'
DATETIME	8	dal '01/01/1000 00:00:00' al '31/12/9999 23:59:59'
TIMESTAMP[(M)]	4	dal '01/01/1970' al '31/12/2037'
TIME	3	da '-838:59:59' a '838:59:59'
YEAR[(M)]	1	per YEAR(4) dal '1901' al '2144'

Tipi di Dati: Testo e Altro

Tipo	Byte	Max Length
CHAR[(M)]/BINARY[(M)]	M	M
VARCHAR(M)/VARBINARY[(M)]	L+1	M
TINYBLOB/ TINYTEXT	L+1	255
BLOB/TEXT	L+2	65.535
MEDIUMBLOB/MEDIUMTEXT	L+3	16.777.215
LONGBLOB/LONGTEXT	L+4	4.294.967.295
ENUM('value1','value2',...)	1 o 2 byte	65535 elementi
SET ('value1','value2',...)	1,2,3,4 o 8 byte	64 elementi
JSON	---	---

L è la lunghezza effettiva del testo memorizzato.

MySQL da riga di comando: Funzioni e operatori

Operatori

- Aritmetici:
 - "+" (**addizione**);
 - "-" (**sottrazione**);
 - "*" (**moltiplicazione**);
 - "/" (**divisione**);
 - "%" (**modulo**);
- Matematici:
 - **ABS(X)**
 - **FLOOR(X)**
 - **CEILING(X)**
 - **SIN(X)**
 - **COS(X)**
 - **LN(X)**
 - **LOG(X)**
 - **LOG(B,X)**
- Logici:
 - **NOT(!);**
 - **AND(&&);**
 - **OR(||)**
 - **XOR;**
- Confronto (Risultato 1 o 0):
 - **=, <=>** (NULL-safe);
 - **<>, !=;**
 - **<=, <, >= ;**
 - **>;**
 - **IS NULL;**
 - **IS NOT NULL.**

Operatori

- Per controllare se un numero è all'interno di un range di valori si può usare una delle seguenti espressioni:
 - *expr BETWEEN min AND max*
 - *expr NOT BETWEEN min AND max*
- Per confrontare rispetto ad una lista fissata di valori:
 - *expr IN (value, ...)*
 - *expr NOT IN (value, ...)*
- **COALESCE(*val, ...*)**: restituisce il primo elemento non-NUL di una lista;
- **INTERVAL(*N,N1,N2,N3,...*)**:
Ritorna:
 - 0 se *N < N1*;
 - 1 se *N < N2*;
 - ecc...
 - -1 se *N* è NULL.

Funzioni su stringhe

- Funzione conversioni case:
 - **LOWER(str)** , **LCASE(str)**
 - **UPPER(str)** , **UCASE(str)**
- Funzioni manipolazione stringhe:
 - **ASCII(str)**: ritorna il valore numerico del carattere più a sinistra di str;
 - **BIN(N)**: Ritorna una stringa che rappresenta il valore binario di N;
 - **CONCAT_WS(separator,str1,str2,...)**: il primo argomento è il separatore il resto gli argomenti;
 - **CONV(*N,from_base,to_base*)**: converte i numeri tra differenti basi;
 - **BIT_LENGTH(str)**: Ritorna la lunghezza della stringa str in bit;
 - **CHAR(*N,...*)**: interpreta ogni argomento N come intero e ritorna una stringa consistente dei caratteri dati dal codice numerico degli interi;
 - **CHAR_LENGTH(str)**: ritorna la lunghezza della stringa misurata in caratteri;
 - **CONCAT(*str1,str2,...*)**: ritorna la stringa che si ottiene concatenando gli argomenti. Ritorna NULL se un argomento è NULL.

Funzioni su stringhe

- Funzioni manipolazione stringhe:
 - **ELT(*N,str1,str2,str3,...*)**: Ritorna str1 se N=1, str2 se N=2 ect...;
 - **FIELD(*str,str1,str2,str3,...*)**: Ritorna la posizione di str in str1,str2,ect...;
 - **FIND_IN_SET(*str,strlist*)**: Ritorna un valore nel range tra 1 a N se la stringa str è nella lista delle stringhe strlist (ovvero N sottostringhe, separate da ",");
 - **HEX(*N_o_S*)**: ritorna il valore esadecimale della stringa;
 - **LEFT(*str,len*)**: Ritorna i len caratteri più a sinistra di str;
 - **INSTR(*str, substr*)**: Ritorna la posizione della prima occorrenza della substr in str;
 - **LENGTH(*str*)**: Ritorna la lunghezza della stringa str in bytes;
 - **LOCATE(*substr,str*) / LOCATE(*substr,str,pos*)**: La prima sintassi ritorna la posizione delle prima occorrenza di substr in str, la seconda inizia la ricerca dalla posizione pos;
 - **LTRIM(*str*)**: Ritorna str con gli spazi iniziali rimossi.

Funzioni su stringhe

- **REPEAT**(*str,count*): Ritorna una stringa formata da *str* ripetuta *count* volte;
- **REPLACE**(*str,from,to*): Rimpiazza tutte le occorrenze di *from* in *to* da *str*;
- **REVERSE**(*str*): Ritorna la stringa invertita;
- **RIGHT**(*str,len*): Ritorna i *len* caratteri più a destra di *str*;
- **RTRIM**(*str*): Ritorna la stringa *str* con gli spazi finali rimossi;
- **STRCMP**(*expr1,expr2*): Ritorna **0** (**zero**) se le due stringhe sono uguali, **-1** se il primo argomento è più piccolo del secondo, **1** altrimenti.

Funzioni su stringhe

```
SELECT C.customer_id,
       CONCAT(C.first_name, ' ', C.last_name) AS name,
       CONCAT_WS(' ', A.address, IF(STRCMP(A.address2,'')=0, NULL, A.address2)) AS address,
       A.postal_code, A.phone, CI.city, CO.country
  FROM
customer AS C
  JOIN address AS A ON A.address_id = C.address_id
  JOIN city AS CI ON CI.city_id = A.city_id
  JOIN country AS CO ON CO.country_id = CI.country_id
 WHERE active <> 0;
```

Ricerca Full-Text

- **MATCH** (*col1,col2,...*) **AGAINST** (*expr* [**IN BOOLEAN MODE** | **WITH QUERY EXPANSION**]):
 - **MATCH ... AGAINST** è utilizzata per ricerche full text:
 - ritorna la rilevanza tra il testo che si trova nelle colonne (*col1,col2,...*) e la query *expr*.
 - La similarità è un valore positivo in virgola mobile.
- La funzione **match** esegue una ricerca in linguaggio naturale per una stringa contro un testo che è rappresentato da una o più colonne incluse in un indice FULL TEXT.
- La ricerca di default è case-insensitive.

Ricerca Full-Text

```
SET @WORD = 'circus';
SELECT film_id, title,
       MATCH(title, description) AGAINST (@WORD) AS score
  FROM film_text
 WHERE
       MATCH(title, description) AGAINST (@WORD);
```

Ricerca Full-Text

- È possibile eseguire una ricerca full-text in boolean mode, i segni + e – indicano le parole che devono essere presenti o assenti, rispettivamente per un match che occorre.
- A volte la stringa di ricerca è troppo corta e potrebbe tralasciare dei risultati significativi, è possibile utilizzare **query expansion** per ovviare a questo problema.

Ricerca Full-Text

```
SET @WORD = 'circus';
SELECT film_id, title,
       MATCH(title, description) AGAINST (@WORD WITH QUERY EXPANSION) AS score,
       MATCH(title, description) AGAINST (@WORD) AS score2
  FROM film_text
 WHERE
       MATCH(title, description) AGAINST (@WORD WITH QUERY EXPANSION);
```

Controllo del flusso

CASE **value**

```
    WHEN compare_value THEN result
    [WHEN compare_value THEN result ...]
    [ELSE result]
```

END

CASE

```
    WHEN condition THEN result
    [WHEN condition THEN result ...]
    [ELSE result]
```

END

IF (**expr1**, **expr2**, **expr3**)

IFNULL (**expr1**, **expr2**)

NULLIF (**expr1**, **expr2**)

Controllo del flusso

```
*****  
SELECT film_id, title,  
CASE rating  
    WHEN 'G' THEN 'General Audiences'  
    WHEN 'PG' THEN 'Parental Guidance Suggested'  
    WHEN 'PG-13' THEN 'Parents Strongly Cautioned'  
    WHEN 'R' THEN 'Restricted'  
    WHEN 'NC-17' THEN 'No one 17 and under admitted'  
    ELSE 'There is nothing else'  
END AS ExplainedRating,  
IF(STRCMP(rating, 'NC-17')=0, 'YES', 'no') AS RequireID FROM film;
```

MySQL: Stored objects

Cosa sono gli stored objects

- Porzioni di codice SQL che vengono memorizzate sul server e possono essere richiamate successivamente
- Tipi di stored objects:
 - Stored procedures (CREATE PROCEDURE)
 - Stored functions (CREATE FUNCTION)
 - Trigger (CREATE TRIGGER)
 - Eventi (CREATE EVENT)
 - Viste (CREATE VIEW)

Stored procedures

- Una routine creata con CREATE PROCEDURE e invocabile tramite comando CALL
- Non ha un valore di ritorno, ma può modificare i parametri con cui viene invocata
- Tali parametri possono essere poi ispezionati all'esterno da colui che invoca la procedura
- Può generare dei result sets da restituire al chiamante
- I DBMS che supportano le stored procedure effettuano la loro compilazione una sola volta (al loro inserimento).

Esempio

```
mysql> CREATE PROCEDURE proc_ciao () SELECT 'Ciao Mondo';

mysql> CALL proc_ciao ();
+-----+
| Ciao Mondo      |
+-----+
| Ciao Mondo      |
+-----+
```

Stored procedure

Per la creazione, alcuni database managers impiegano linguaggi procedurali:

- [PL/pgSQL](#) di PostgreSQL
- [PL/SQL](#) di Oracle
- [SQL PL](#) di DB2
- [Transact-SQL](#) di SQL Server
- [MySql Stored Procedure](#) di MySQL,

Stored procedure

Essendo SQL è un linguaggio dichiarativo, le Stored Procedures rappresentano una sua estensione procedurale grazie ai suoi costrutti:

BEGIN, END, DECLARE, FOR, WHILE, LOOP, IF, etc

Se scritte in altri linguaggi standard (C/C++), esse sono compilate come oggetti esterni ed integrati in MySQL.

Sintassi di una stored procedure

CREATE

[DEFINER = { user | CURRENT_USER }]

PROCEDURE nome_procedura ([parametri[,...]])

[caratteristiche ...] corpo_della_routine

Parola chiave DEFINER

DEFINER non è un parametro obbligatorio; serve per assegnare un proprietario alla routine.

Nel caso in cui questo non venga specificato verrà considerato come owner predefinito l'utente corrente.

Corpo della procedura

Il corpo della procedura può essere una semplice SELECT oppure un blocco formato da più istruzioni (SELECT e non).

Il blocco di istruzioni va racchiuso tra le parole chiave BEGIN e END

Le istruzioni sono separate mediante «;»

Se il blocco è definito da linea di comando, è necessario prima ridefinire il delimitatore di comandi di MySQL tramite il comando DELIMITER

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE nome_procedura (p1 INT)
-> BEGIN
-> blocco istruzioni
-> END
-> //
mysql> DELIMITER ;
```

Dichiarare variabili locali per una procedura

```
DECLARE variable_name datatype(size) DEFAULT default_value;
```

I tipi delle variabili sono quelli primitivi supportati da MySQL:
INT, VARCHAR, DATETIME,

Quando viene dichiarata una variabile, il suo valore iniziale è NULL. E' possibile assegnare un valore utilizzando il comando DEFAULT.

```
DECLARE total_sale INT DEFAULT 0
```

```
DECLARE x, y INT DEFAULT 0
```

Assegnare valori alle variabili locali

```
DECLARE total_count INT DEFAULT 0  
      SET total_count = 10;
```

Un altro modo per assegnare un valore ad una variabile è tramite l'istruzione SQL:

```
DECLARE total_products INT DEFAULT 0  
      SELECT COUNT(*) INTO total_products  
      FROM products
```

Blocchi annidati e scope delle variabili

- . . .
- DECLARE x INT (3); ————— **Scope of x**
- BEGIN
- . . .
- DECLARE
- y INT; ————— **Scope of y**
- BEGIN
- . . .
- END;
- . . .
- END;

Dichiarare variabili di sessione

Una variabile dichiarata con DECLARE è visibile solo all'interno della procedura in cui si trova

Usando il comando SET è possibile anche dichiarare variabili di sessione (indicate con @var_name) la cui visibilità è estesa all'intera sessione corrente

```
SET @var_name = expr [, @var_name = expr] ...
```

```
SET @x = 0
```

A differenza di DECLARE, il comando SET può essere usato sia all'interno che all'esterno di una procedura

Tipi di parametri di una Stored Procedure

IN: è un parametro a cui viene assegnato un valore v dall'esterno tramite l'invocazione della procedura mediante CALL (CALL v).

- **OUT (non è standard):** è un parametro che riceve un valore all'interno della procedura tramite SET o INTO; al termine della procedura il valore del parametro viene assegnato ad una variabile x definita nell'invocazione della procedura mediante CALL (CALL @x).
- **INOUT (non è standard):** rappresenta una combinazione tra i due parametri precedenti.

Esempi con parametri IN

```
mysql> CREATE PROCEDURE proc_in (IN p INT(3)) SET @x = p//
```

```
mysql> CALL proc_in (14)//
```

```
mysql> SELECT @x//  
+-----+  
| @x   |  
+-----+  
| 14   |  
+-----+
```

Esempi con parametri IN

```
DELIMITER //
CREATE PROCEDURE GetOfficeByCountry
    (IN countryName VARCHAR(255))
BEGIN
    SELECT city, phone
    FROM offices
    WHERE country = countryName;
END //
DELIMITER ;
```

```
mysql> CALL GetOfficeByCountry('USA')
```

Esempio con parametro OUT

```
mysql> CREATE PROCEDURE proc_out (OUT p INT)
-> SET p = -2 //  
  
mysql> CALL proc_out (@y)//
mysql> SELECT @y//  
+-----+  
| @y   |  
+-----+  
| -2   |  
+-----+
```

Esempio con parametri IN e OUT

```
DELIMITER $$  
CREATE PROCEDURE CountOrderByStatus  
( IN orderStatus VARCHAR(25) , OUT total INT)  
BEGIN  
    SELECT count(orderNumber) INTO total  
    FROM orders  
    WHERE status = orderStatus;  
END$$  
DELIMITER ;
```

```
mysql> CALL CountOrderByStatus('Shipped',@total);  
mysql> SELECT @total AS total_shipped;
```

Esempio con parametro INOUT

```
DELIMITER $$  
CREATE PROCEDURE 'Capitalize' (INOUT str VARCHAR(1024))  
BEGIN  
    DECLARE i INT DEFAULT 1;  
    DECLARE myc, pc CHAR(1);  
    DECLARE outstr VARCHAR(1000) DEFAULT str;  
    WHILE i <= CHAR_LENGTH(str) DO  
        SET myc = SUBSTRING(str, i, 1);  
        SET pc = CASE WHEN i = 1 THEN ' '  
        ELSE SUBSTRING(str, i - 1, 1) END;  
        IF pc IN (' ', '&', '''', '_', '?', ';',  
        ':', '!', ',', '-', '/', '(', '.') THEN  
            SET outstr = INSERT(outstr, i, 1, UPPER(myc));  
        END IF;  
        SET i = i + 1;  
    END WHILE;  
    SET str = outstr;  
END$$  
DELIMITER ;
```

```
mysql> SET @str = 'mysql stored procedure tutorial';  
mysql> CALL Capitalize(@str);  
mysql> SELECT @str;
```

Controlli Condizionali

```
IF expression THEN commands  
END IF;
```

```
IF expression THEN commands  
  ELSEIF expression THEN commands  
  ELSE commands  
END IF;
```

```
CASE  
  WHEN expression THEN commands  
    ...  
  WHEN expression THEN commands  
  ELSE commands  
END CASE;
```

Loop

```
WHILE espressione DO  
    Istruzione  
END WHILE
```

```
REPEAT  
    Istruzione;  
    UNTIL Espressione  
END REPEAT
```

Esempio con WHILE

```
DELIMITER $$  
DROP PROCEDURE IF EXISTS WhileLoopProc$$  
CREATE PROCEDURE WhileLoopProc()  
BEGIN  
    DECLARE x INT;  
    DECLARE str VARCHAR(255);  
    SET x = 1;  
    SET str = '';  
    WHILE x <= 5 DO  
        SET str = CONCAT(str,x,',');  
        SET x = x + 1;  
    END WHILE;  
    SELECT str;  
END$$  
DELIMITER ;
```

Esempio con REPEAT

```
DELIMITER $$  
DROP PROCEDURE IF EXISTS WhileLoopProc$$  
CREATE PROCEDURE WhileLoopProc()  
BEGIN  
    DECLARE x INT;  
    DECLARE str VARCHAR(255);  
    SET x = 1;  
    SET str = '';  
    REPEAT  
        SET str = CONCAT(str,x,',');  
        SET x = x + 1;  
    UNTIL x > 5  
    END REPEAT;  
    SELECT str;  
END$$  
DELIMITER ;
```

Cursori

All'interno delle stored procedures è possibile definire dei cursori che vengono usati per setacciare un insieme di righe restituite da una query e permettono di controllarle individualmente.

I cursori vengono definiti subito dopo le variabili all'interno della procedura

Cursori

A partire da MySQL 5.x, i cursori hanno le seguenti proprietà:

- **Read only:** non è possibile modificare il contenuto di un cursore.
- **Non-scrollable:** è possibile leggere i dati nel cursore solo in avanti e senza salti in maniera sequenziale.
- **Asensitive:** evitare di aggiornare le tabelle sulle quali sono stati aperti dei cursori: in questi casi è possibile ottenere dalla lettura dei cursori risultati sbagliati.

Cursori

```
DECLARE cursor_name CURSOR FOR SELECT_statement;
```

Prima di usarlo per leggere righe, il cursore va aperto con il comando OPEN.

```
OPEN cursor_name;
```

Per leggere righe della tabella con il cursore si usa il comando FETCH: il contenuto può essere salvato in una lista di variabili.

```
FETCH cursor_name INTO variable list;
```

Per chiudere e rilasciare la memoria occupata, si usa il comando CLOSE:

```
CLOSE cursor_name;
```

Cursori

- Per evitare che, in mancanza di dati disponibili da leggere tramite FETCH, SQL sollevi un errore fatale, subito dopo il cursore occorre dichiarare anche un HANDLER
- Nel caso dei cursori la condizione scatenata è NOT FOUND
- EXIT fa terminare immediatamente la procedura, CONTINUE la fa proseguire fino alla fine

```
DECLARE handler_action HANDLER
FOR condition_value [, condition_value] ...
statement

handler_action: {
    CONTINUE
    | EXIT
    | UNDO
}

condition_value: {
    mysql_error_code
    | SQLSTATE [VALUE] sqlstate_value
    | condition_name
    | SQLWARNING
    | NOT FOUND
    | SQLEXCEPTION
}
```

Esempio con cursore

```
DELIMITER $$  
DROP PROCEDURE IF EXISTS CursorProc$$  
CREATE PROCEDURE CursorProc()  
BEGIN  
    DECLARE no_more_products, quantity_in_stock INT  
        DEFAULT 0;  
    DECLARE prd_code VARCHAR(255);  
    DECLARE cur_product CURSOR FOR  
        SELECT productCode FROM products;  
    DECLARE CONTINUE HANDLER FOR  
        NOT FOUND SET no_more_products = 1;  
  
    ...
```

Esempio con cursore

```
...
OPEN cur_product;
REPEAT
    FETCH cur_product INTO prd_code;
    SELECT quantityInStock INTO quantity_in_stock
    FROM products
    WHERE productCode = prd_code;
UNTIL no_more_products = 1
END REPEAT;
CLOSE cur_product;
...
```

Vantaggi e svantaggi delle Stored Procedures

- + Le SP sono riusabili e trasparenti a qualsiasi applicazione dato che girano sul server.
- + Migliorano l'astrazione (chi invoca la procedura può ignorarne i dettagli implementativi)
- + Accesso controllato alle tabelle in quanto solo gli utenti a cui è concesso l'uso della procedura potranno effettuare letture o modifiche alla tabella.
- + Controllo centralizzato su certi vincoli d'integrità non esprimibili nelle tabelle.

- I DBMS utilizzano linguaggi dedicati e differenti per la stesura di procedure.
- La logica dell'applicazione è spostata sul server che dovrà essere in grado di sostenere il carico di lavoro.
- MySQL permette di definire soltanto procedure scritte in linguaggio SQL. Quindi non si ha libertà di scegliere il linguaggio da utilizzare.

Stored functions

- Una routine creata con CREATE FUNCTION e invocabile direttamente all'interno di una SELECT
- Ha un valore di ritorno il cui tipo può essere specificato al momento della definizione della funzione
- Nel corpo della funzione è possibile definire una SELECT o un blocco di istruzioni delimitato da BEGIN e END, con le stesse regole e gli stessi costrutti visti per le stored procedures

Sintassi di una Stored Function

CREATE [DEFINER = { user | CURRENT_USER }]

FUNCTION nome_procedura ([parametri[,...]])

RETURNS type

[caratteristiche ...] corpo_della_routine

Nel corpo della funzione deve esserci una istruzione del tipo:

RETURN *value*

Esempio

```
mysql> CREATE FUNCTION hello (s CHAR(20))
      ->     RETURNS CHAR(50) DETERMINISTIC
      ->     RETURN CONCAT('Hello, ', s, '!');

Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT hello('world');

+-----+
| hello('world') |
+-----+
| Hello, world! |
+-----+
1 row in set (0.00 sec)
```

Eventi

- CREATE EVENT permette di creare un evento schedulato, da eseguire una volta sola o a intervalli regolari

```
CREATE [DEFINER = user] EVENT [IF NOT EXISTS] event_name ON SCHEDULE schedule [ON COMPLETION [NOT] PRESERVE] [ENABLE | DISABLE | DISABLE ON {REPLICA | SLAVE}] [COMMENT 'string'] DO event_body;  
  
schedule: { AT timestamp [+ INTERVAL interval] ... | EVERY interval [STARTS timestamp [+ INTERVAL interval] ...] [ENDS timestamp [+ INTERVAL interval] ...] }  
  
interval: quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE | WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE | DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
```

```
CREATE EVENT e_hourly  
ON SCHEDULE  
EVERY 1 HOUR  
COMMENT 'Clears out sessions table each hour.'  
DO  
DELETE FROM site_activity.sessions;
```

Trigger e viste

```
CREATE
    [DEFINER = user]
    TRIGGER [IF NOT EXISTS] trigger_name
trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    [trigger_order]
    trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }

trigger_order: { FOLLOWS | PRECEDES } other_trigger_name
```

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = user]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

MySQL: Window function

Window Function

Una window function permette di eseguire operazioni simili alle funzioni di aggregazione su un insieme di righe.

A differenza delle funzioni di aggregazione che raggruppano le righe in una sola in base ad un criterio:

- Una window function produce un risultato per ogni riga della query (chiamata riga corrente)
- Il risultato prodotto per la riga corrente X è determinato a partire dal contenuto di X e di altre righe (chiamate righe query) individuate sulla base di alcuni criteri
- La riga corrente + le righe query formano la cosiddetta «window» su cui opera la funzione

Definizione di una window function tramite clausola OVER

In questo caso la funzione di aggregazione è seguita dalla clausola OVER per indicare che si tratta di una window function.

All'interno della clausola OVER si specifica quali righe formano la window:

- OVER() indica che la window su cui opera la funzione è l'intero insieme di righe
- OVER(PARTITION BY x,y ...) indica che le righe devono essere partizionate sulla base dei valori delle colonne x,y,... e la window su cui opera la funzione è la partizione cui appartiene la riga corrente

Definizione di una window function tramite clausola OVER

Il partizionamento delle righe avviene dopo l'esecuzione di FROM, WHERE, GROUP BY, e HAVING e prima di ORDER BY, LIMIT e DISTINCT.

La window function può essere una classica funzione di aggregazione oppure una funzione speciale che può essere usata solo in questo contesto

Esempio

Con raggruppamento

```
mysql> SELECT SUM(profit) AS total_profit
      FROM sales;
+-----+
| total_profit |
+-----+
|      7535 |
+-----+
mysql> SELECT country, SUM(profit) AS country_profit
      FROM sales
      GROUP BY country
      ORDER BY country;
+-----+-----+
| country | country_profit |
+-----+-----+
| Finland |        1610 |
| India   |        1350 |
| USA     |       4575 |
+-----+-----+
```

Esempio

Con window function

```
mysql> SELECT
    year, country, product, profit,
    SUM(profit) OVER() AS total_profit,
    SUM(profit) OVER(PARTITION BY country) AS country_profit
  FROM sales
 ORDER BY country, year, product, profit;
```

year	country	product	profit	total_profit	country_profit
2000	Finland	Computer	1500	7535	1610
2000	Finland	Phone	100	7535	1610
2001	Finland	Phone	10	7535	1610
2000	India	Calculator	75	7535	1350
2000	India	Calculator	75	7535	1350
2000	India	Computer	1200	7535	1350
2000	USA	Calculator	75	7535	4575
2000	USA	Computer	1500	7535	4575
2001	USA	Calculator	50	7535	4575
2001	USA	Computer	1200	7535	4575
2001	USA	Computer	1500	7535	4575
2001	USA	TV	100	7535	4575
2001	USA	TV	150	7535	4575

Funzioni di aggregazione utilizzabili

Name	Description
<u>AVG ()</u>	Return the average value of the argument
<u>BIT_AND ()</u>	Return bitwise AND
<u>BIT_OR ()</u>	Return bitwise OR
<u>BIT_XOR ()</u>	Return bitwise XOR
<u>COUNT ()</u>	Return a count of the number of rows returned
<u>COUNT(DISTINCT)</u>	Return the count of a number of different values
<u>GROUP_CONCAT ()</u>	Return a concatenated string
<u>JSON_ARRAYAGG ()</u>	Return result set as a single JSON array
<u>JSON_OBJECTAGG ()</u>	Return result set as a single JSON object
<u>MAX ()</u>	Return the maximum value
<u>MIN ()</u>	Return the minimum value
<u>STD ()</u>	Return the population standard deviation
<u>STDDEV ()</u>	Return the population standard deviation
<u>STDDEV_POP ()</u>	Return the population standard deviation
<u>STDDEV_SAMP ()</u>	Return the sample standard deviation
<u>SUM ()</u>	Return the sum
<u>VAR_POP ()</u>	Return the population standard variance
<u>VAR_SAMP ()</u>	Return the sample variance
<u>VARIANCE ()</u>	Return the population standard variance

Funzioni window specifiche

Name	Description
<u>CUME_DIST()</u>	Cumulative distribution value
<u>DENSE_RANK()</u>	Rank of current row within its partition, without gaps
<u>FIRST_VALUE()</u>	Value of argument from first row of window frame
<u>LAG()</u>	Value of argument from row lagging current row within partition
<u>LAST_VALUE()</u>	Value of argument from last row of window frame
<u>LEAD()</u>	Value of argument from row leading current row within partition
<u>NTH_VALUE()</u>	Value of argument from N-th row of window frame
<u>NTILE()</u>	Bucket number of current row within its partition.
<u>PERCENT_RANK()</u>	Percentage rank value
<u>RANK()</u>	Rank of current row within its partition, with gaps
<u>ROW_NUMBER()</u>	Number of current row within its partition

Definizione di una window function tramite clausola WINDOW

E' possibile definire la window function in una clausola separata WINDOW e richiamarla nella clausola OVER

```
SELECT ..., FUNZIONE() OVER nome_finestra, ...
FROM ...
WINDOW nome1 AS (...)[, nome2 AS (...), ...]
```

Esempio

```
mysql> SELECT
    val,
    ROW_NUMBER() OVER w AS 'row_number',
    RANK()      OVER w AS 'rank',
    DENSE_RANK() OVER w AS 'dense_rank'
  FROM numbers
 WINDOW w AS (ORDER BY val);
+-----+-----+-----+-----+
| val | row_number | rank | dense_rank |
+-----+-----+-----+-----+
|   1 |          1 |     1 |         1 |
|   1 |          2 |     1 |         1 |
|   2 |          3 |     3 |         2 |
|   3 |          4 |     4 |         3 |
|   3 |          5 |     4 |         3 |
|   3 |          6 |     4 |         3 |
|   4 |          7 |     7 |         4 |
|   4 |          8 |     7 |         4 |
|   5 |          9 |     9 |         5 |
+-----+-----+-----+-----+
```

Utilizzo dei frame

E' possibile specificare come «window» anche un sottoinsieme della partizione (chiamato «frame»), che è un i cui confini sono definiti secondo la sintassi:

PARTITION BY field [ORDER BY field] [ROWS {*frame_start* | BETWEEN *frame_start* AND *frame_end*}]

frame_start può essere:

- CURRENT_ROW: la riga corrente
- UNBOUNDED PRECEDING: la prima riga della partizione
- **N** PRECEDING: la N-esima riga precedente alla riga corrente

frame_end può essere:

- CURRENT_ROW: la riga corrente
- UNBOUNDED FOLLOWING: l'ultima riga della partizione contenente la riga corrente
- **N** FOLLOWING: la N-esima riga che segue la riga corrente

Se si usa ROWS *frame_start*, allora *frame_end* è la riga corrente

Esempi di frame

1) Considera tutte le righe della partizione a cui appartiene la riga corrente dalla prima riga della partizione a quella corrente

PARTITION BY subject ORDER BY time
ROWS UNBOUNDED PRECEDING

2) Considera la riga corrente, la riga precedente della stessa partizione (se esiste) e la riga successiva della stessa partizione (se esiste)

PARTITION BY subject ORDER BY time
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING

Esempio

```
mysql> SELECT
    time, subject, val,
    FIRST_VALUE(val) OVER w AS 'first',
    LAST_VALUE(val) OVER w AS 'last',
    NTH_VALUE(val, 2) OVER w AS 'second',
    NTH_VALUE(val, 4) OVER w AS 'fourth'
  FROM observations
WINDOW w AS (PARTITION BY subject ORDER BY time
              ROWS UNBOUNDED PRECEDING);
```

time	subject	val	first	last	second	fourth
07:00:00	st113	10	10	10	NULL	NULL
07:15:00	st113	9	10	9	9	NULL
07:30:00	st113	25	10	25	9	NULL
07:45:00	st113	20	10	20	9	20
07:00:00	xh458	0	0	0	NULL	NULL
07:15:00	xh458	10	0	10	10	NULL
07:30:00	xh458	5	0	5	10	NULL
07:45:00	xh458	30	0	30	10	30
08:00:00	xh458	25	0	25	10	30

MySQL: Import/Export

Import di script SQL (file .sql)

aerei.sql

```
drop table if exists aerei;
create table aerei (
    id int primary key auto_increment,
    produttore char(20) not null,
    modello char(20) not null,
    dataimm date,
    numposti int,
    commento text
);

insert into aerei values (null,'boeing','747','2000-10-10',350,''),
    (null,'MDD','Super80',null,null,'prova');
```

Import di script SQL (file .sql)

- Caricamento dal prompt dei comandi

```
mysql db_name < text_file
```

- Caricamento dal prompt dei comandi, nell'ipotesi in cui il file .sql inizi con l'istruzione «USE *db_name*»

```
mysql < text_file
```

- Caricamento dalla shell mysql (dopo aver eseguito «USE *db_name*»)

```
mysql> source file_name
```

```
mysql> \. file_name
```

Import di dati in un DB da file di testo

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name.txt'  
[REPLACE | IGNORE]  
INTO TABLE tbl_name  
[FIELDS  
    [TERMINATED BY '\t']  
    [[OPTIONALLY] ENCLOSED BY ''']  
    [ESCAPED BY '\\\']  
]  
[LINES  
    [STARTING BY ''']  
    [TERMINATED BY '\n']  
]  
[IGNORE number LINES]  
[(col_name,...)]
```

- Il comando LOAD DATA INFILE consente il caricamento di una tabella leggendo le righe da un file di testo.

Opzione LOCAL

- Se l'opzione LOCAL non è specificata, allora il file di testo è caricato dal server su cui si trova il DB nel path indicato
- Se l'opzione LOCAL è specificata, allora il file viene letto dal client e inviato al server, che ne crea una copia e poi lo carica nel DB

Esempi

countries.csv

```
id,country,population,capital
1,USA,"329,500,000",Washington D.C.
2,Canada,"38,010,000",Ottawa
3,UK,"67,220,000",London
4,Spain,"47,350,000",Madrid
5,Greenland,"56,367",Nuuk
```

```
CREATE SCHEMA data;
USE data;
CREATE TABLE `countries` (
`id` int DEFAULT NULL,
`country` text,
`population` text,
`capital` text);

LOAD DATA INFILE 'countries.csv'
INTO TABLE countries
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

- L'opzione IGNORE qui è necessaria perché il file CSV ha una riga di intestazione

Esempi

```
mysql> LOAD DATA INFILE '/tmp/test.txt'  
-> INTO TABLE test LINES STARTING BY  
"yyy";
```

- Quindi un file contenente

yyy"Row", 1

blablabla yyy"Row", 2

- Può essere letto e verrà caricato come ("row",1), ("row",2)

Alternativa: mysqlimport

- Client mysqlimport richiamabile direttamente dal prompt dei comandi

```
mysqlimport [options] db_name textfile1 [textfile2 ...]
```

```
$> mysqlimport --local test imptest.txt
test.imptest: Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
$> mysql -e 'SELECT * FROM imptest' test
+-----+
| id   | n           |
+-----+
| 100  | Max Sydow    |
| 101  | Count Dracula|
+-----+
```

Backup

- Backup di un database

```
mysqldump db_name > backup-file.sql
```

- Backup di più database

```
mysqldump --databases db_name1 [db_name2 ...] > my_databases.sql
```

- Backup di tutti i database

```
mysqldump --all-databases > all_databases.sql
```

MySQL: Monitoraggio delle performance

EXPLAIN

- Consente di capire le performance di una query e mostra quali indici effettivamente la query sta usando.

EXPLAIN SELECT *select_options*

- Se richiamato su una tabella, equivale a DESCRIBE *tbl_name* o SHOW COLUMNS FROM *tbl_name*

EXPLAIN *tbl_name*

Le colonne indicizzate devono essere stand alone

```
mysql> explain select * from event where year(event.date) < '2003';
+-----+-----+-----+-----+-----+-----+
| table | type  | possible_keys | key   | key_len | ref   | rows  | Extra       |
+-----+-----+-----+-----+-----+-----+
| event | ALL   | NULL          | NULL  |      4  | NULL  |     6 | Using where |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> explain select * from event where event.date < '2003-01-01';
+-----+-----+-----+-----+-----+-----+
| table | type  | possible_keys | key   | key_len | ref   | rows  | Extra       |
+-----+-----+-----+-----+-----+-----+
| event | ALL   | Index_2       | NULL  |      4  | NULL  |     6 | Using where |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

MySQL Workbench

Cosa è MySQL Workbench

- Un tool grafico con un'interfaccia utente per operare con il server MySQL e i database presenti.

Funzionalità presenti:

- Creazione e gestione delle connessioni al server
- Creazione di modelli grafici dello schema del database
- Editor per la creazione/modifica di tabelle, viste, colonne, indici, trigger, e funzioni
- Amministrazione del server e degli utenti
- Monitoraggio delle performance del server MySQL
- Migrazione dei dati verso altri DBMS
- Supporto ai prodotti MySQL Enterprise (solo nella versione Enterprise)

Download di MySQL Workbench

MySQL Community Downloads

- [MySQL Yum Repository](#)
 - [MySQL APT Repository](#)
 - [MySQL SUSE Repository](#)
 - [MySQL Community Server](#)
 - [MySQL NDB Cluster](#)
 - [MySQL Router](#)
 - [MySQL Shell](#)
 - [MySQL Operator](#)
 - [MySQL NDB Operator](#)
 - [MySQL Workbench](#)
 - [MySQL Installer for Windows](#)
- [C API \(libmysqlclient\)](#)
 - [Connector/C++](#)
 - [Connector/J](#)
 - [Connector/.NET](#)
 - [Connector/Node.js](#)
 - [Connector/ODBC](#)
 - [Connector/Python](#)
 - [MySQL Native Driver for PHP](#)
 - [MySQL Benchmark Tool](#)
 - [Time zone description tables](#)
 - [Download Archives](#)

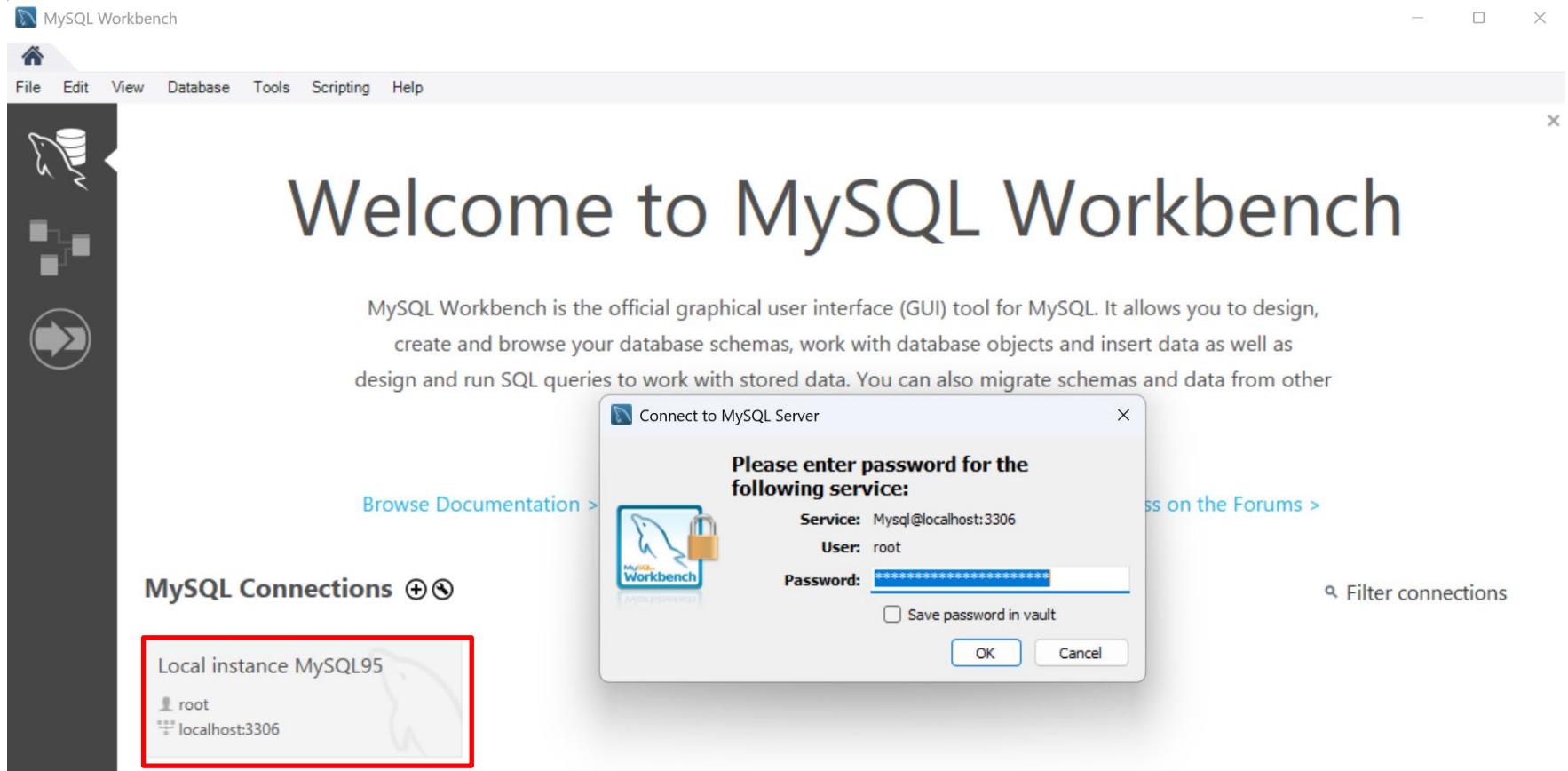
**MySQL Enterprise Edition
for Developers**

Free for learning, developing,
and prototyping.



Download Now »

Connessione al DB



The screenshot shows the MySQL Workbench interface. On the left, there's a sidebar with icons for Home, Database, Tools, and Help. The main area has a large "Welcome to MySQL Workbench" title. Below it, a text block describes the tool's features. A "MySQL Connections" section lists a connection named "Local instance MySQL95". This connection entry is highlighted with a red rectangular box. To the right of the connection list, a "Connect to MySQL Server" dialog box is open, prompting for a password. The dialog includes fields for Service (Mysql@localhost:3306), User (root), Password (redacted), and a Save password in vault checkbox. At the bottom of the dialog are OK and Cancel buttons.

MySQL Workbench

File Edit View Database Tools Scripting Help

Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other

Browse Documentation >

MySQL Connections ⊕ ✖

Local instance MySQL95

root

localhost:3306

Connect to MySQL Server

Please enter password for the following service:

Service: Mysql@localhost:3306

User: root

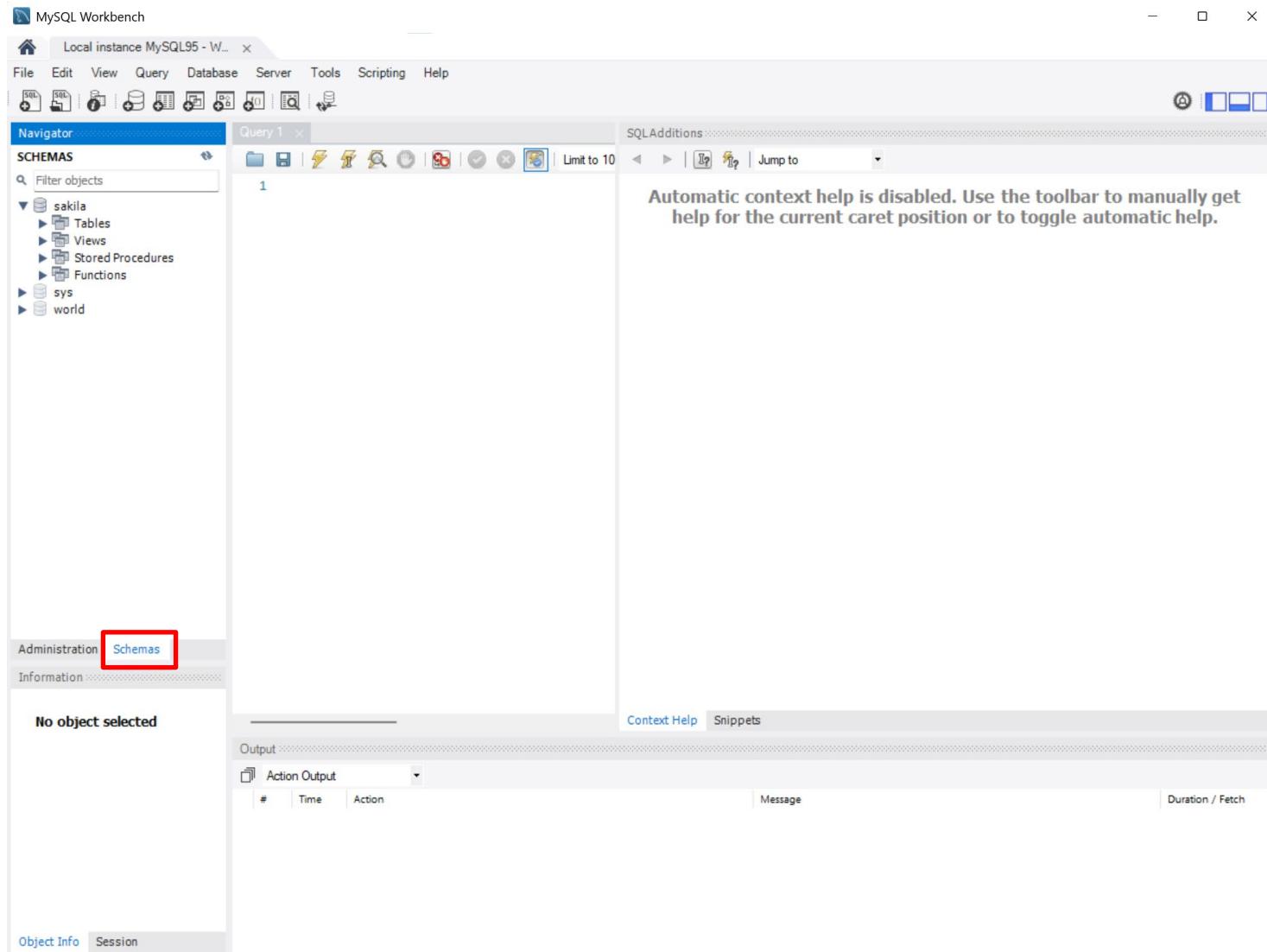
Password: *****

Save password in vault

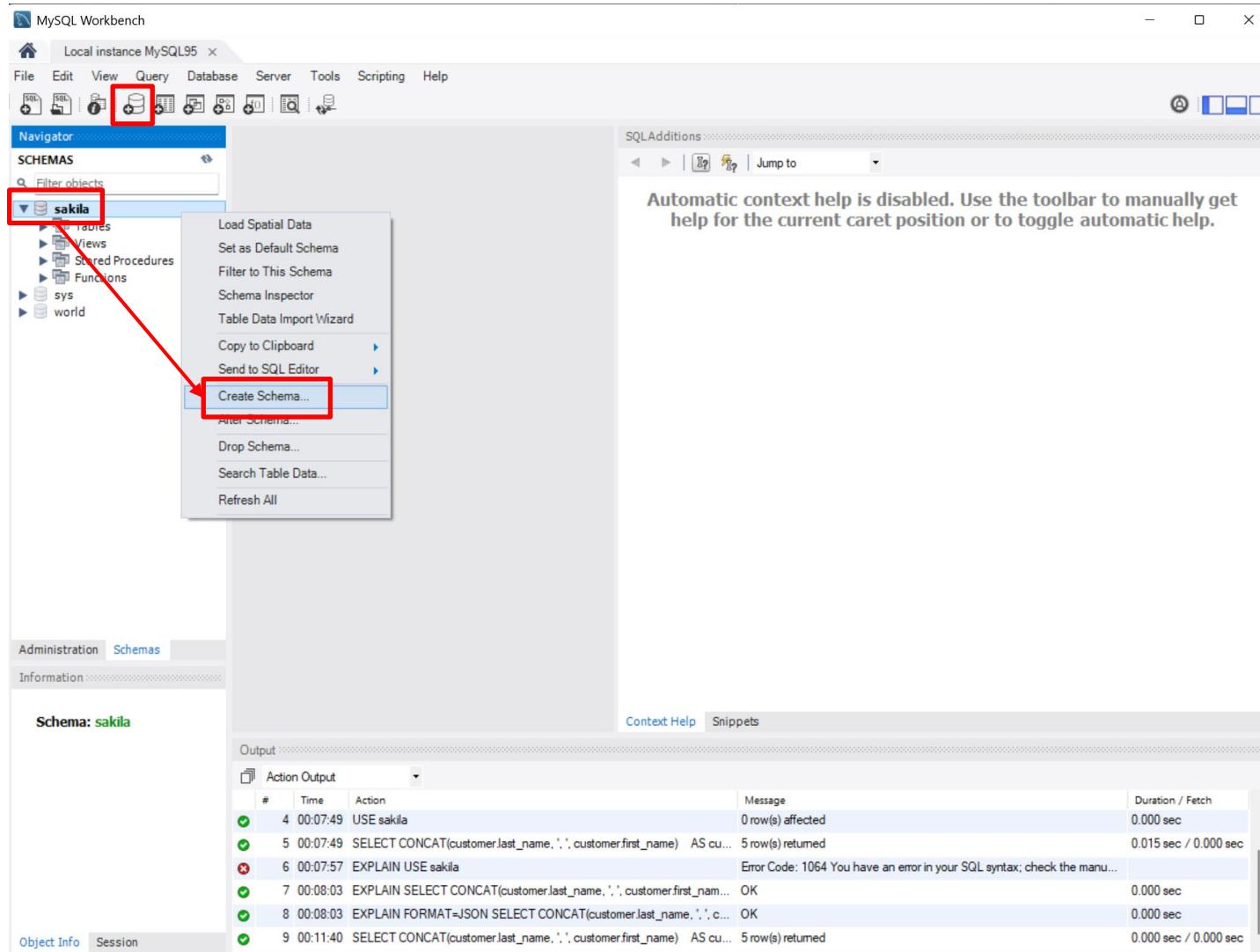
OK Cancel

Filter connections

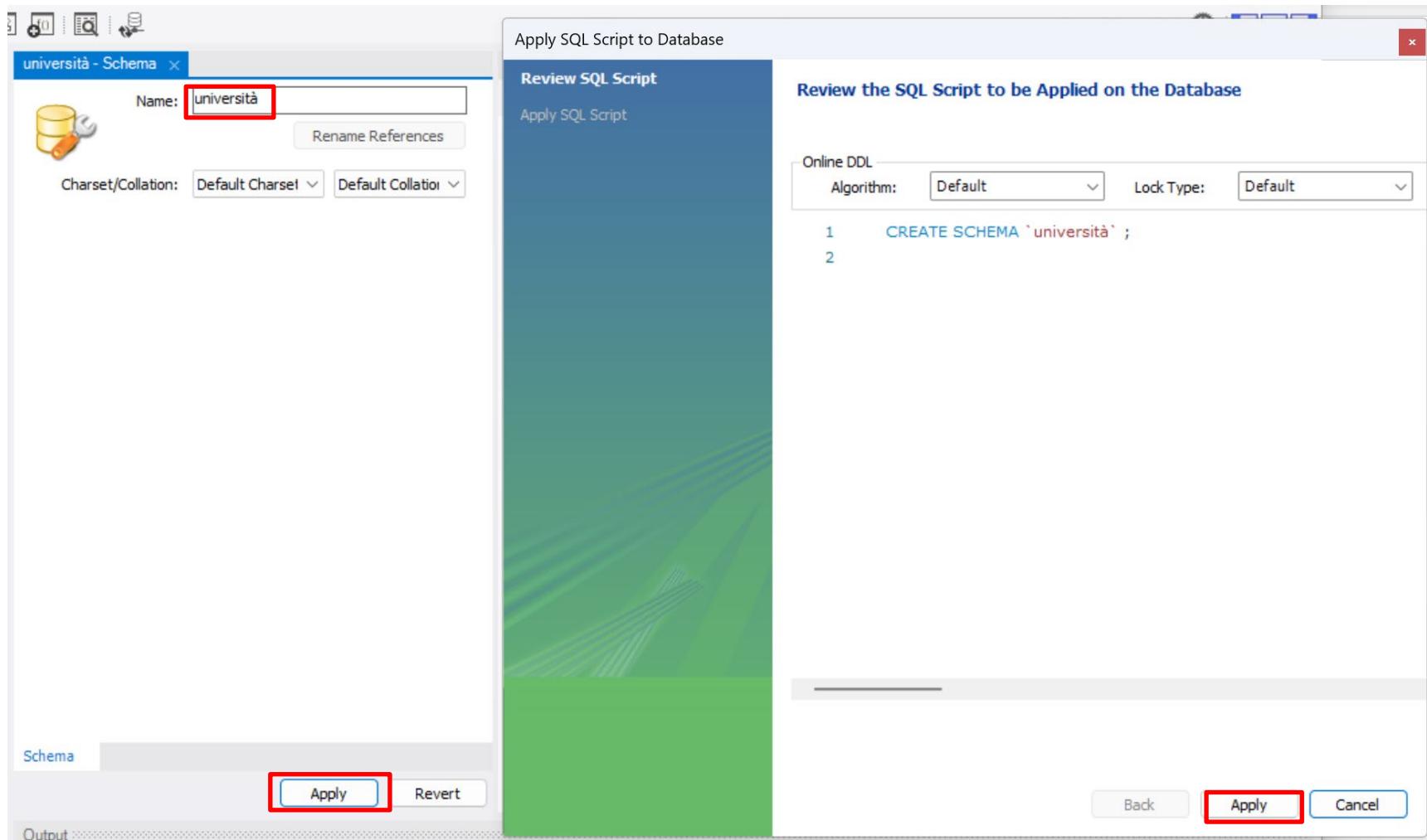
Pannello «Schemas»



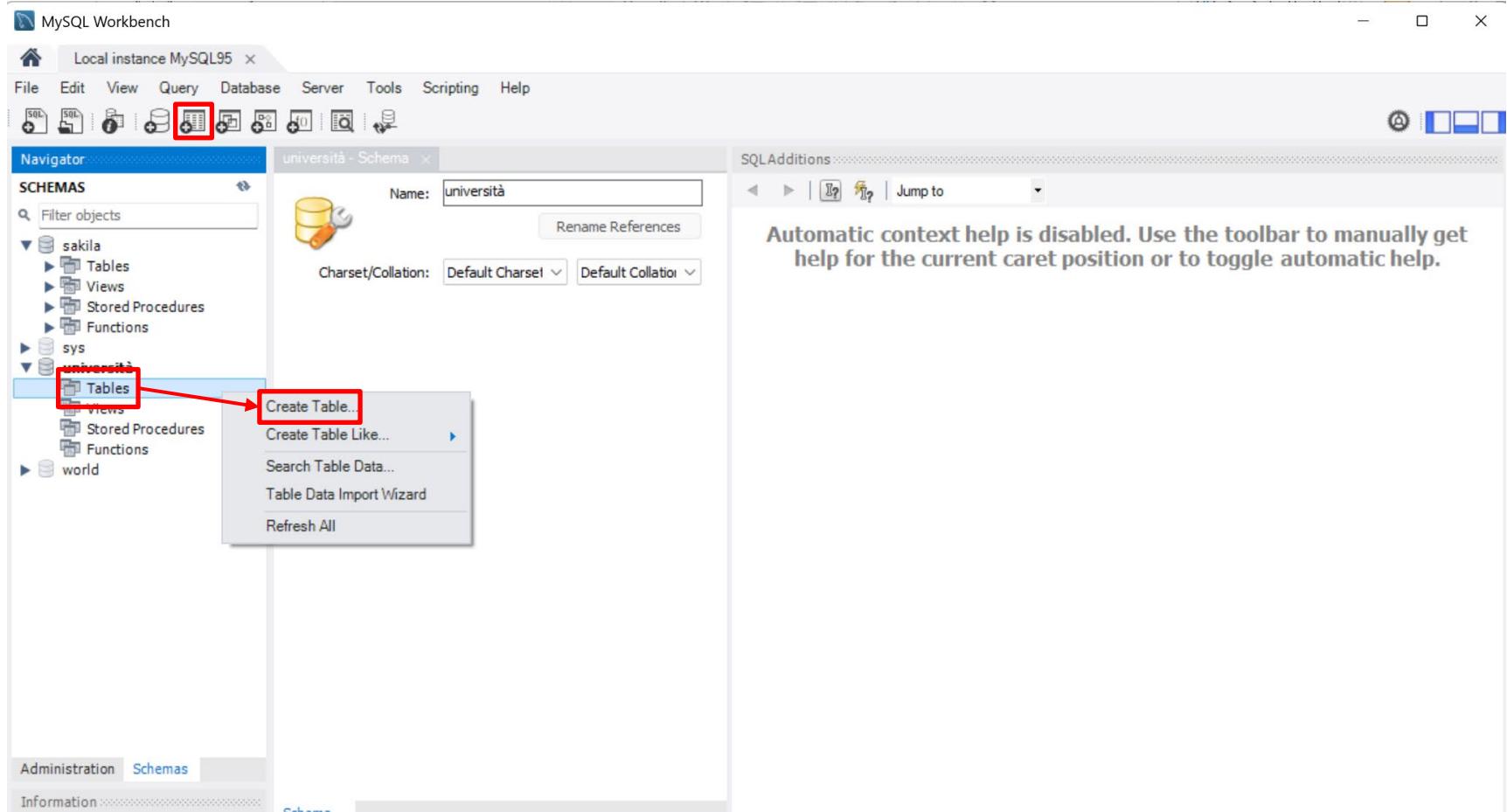
Creazione di un nuovo database



Creazione di un nuovo database



Creazione di una nuova tabella



Creazione di una nuova tabella

università - Schema Studenti - Table

Table Name: **Studenti** Schema: **università**

Charset/Collation: Default Charset Default Collation Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Matricola	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>						
Nome	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: **Studenticolo** Data Type: **VARCHAR(45)**

Charset/Collation: Default Charset Default Collation

Comments:

Storage: Virtual Stored
 Primary Key Not Null Unique
 Binary Unsigned Zero Fill
 Auto Increment Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

Creazione di una nuova tabella

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

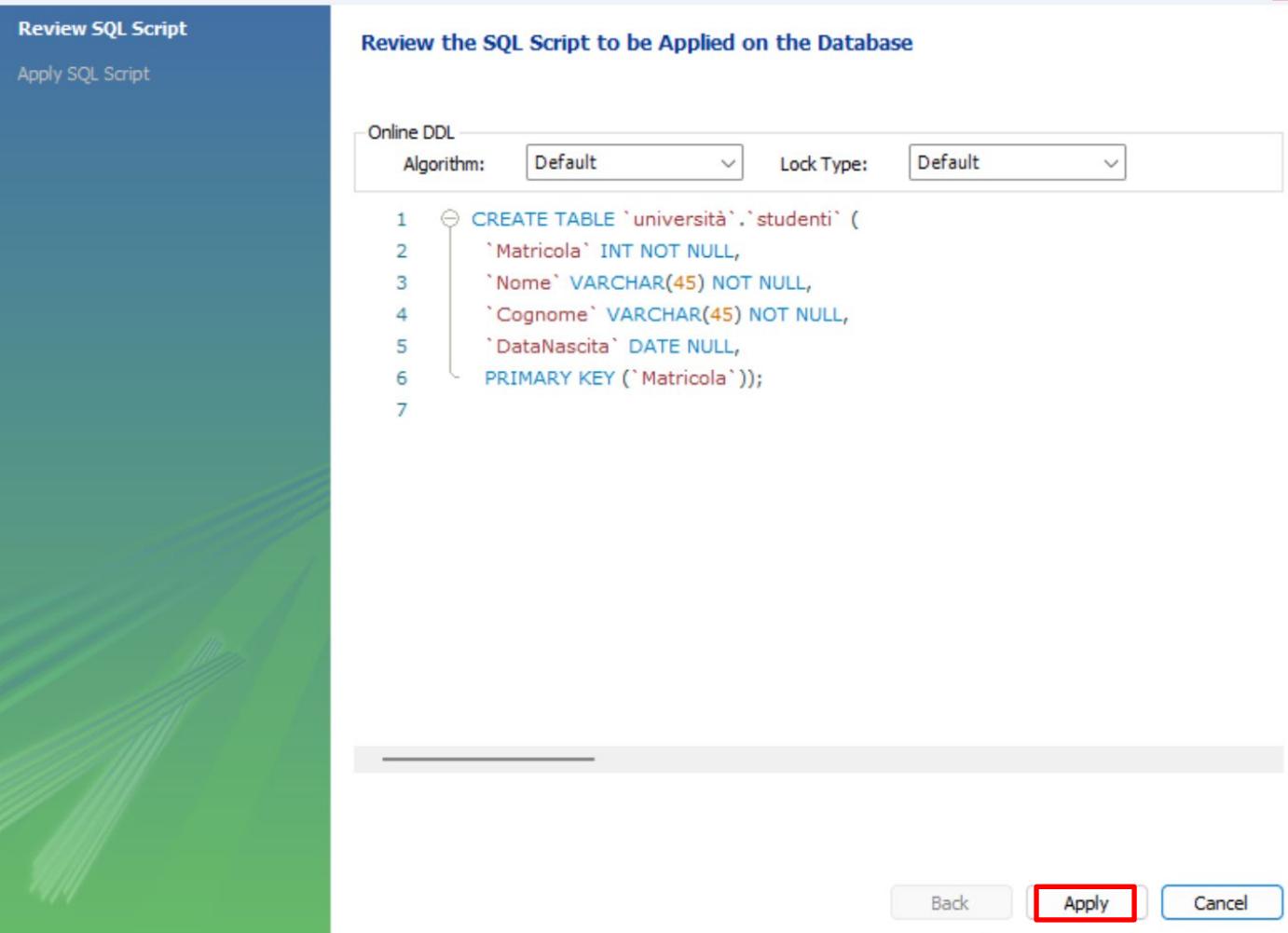
Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Default Lock Type: Default

```
1 CREATE TABLE `università`.`studenti` (
2     `Matricola` INT NOT NULL,
3     `Nome` VARCHAR(45) NOT NULL,
4     `Cognome` VARCHAR(45) NOT NULL,
5     `DataNascita` DATE NULL,
6     PRIMARY KEY (`Matricola`));
7
```

Back Apply Cancel



Popolamento di una tabella

MySQL Workbench

Local instance MySQL95

File Edit View Query Database Server Tools Scripting Help

Navigator: università - Schema studenti - Table studenti - Table studenti - Table

SCHEMAS

Filter objects

sakila

- Tables
- Views
- Stored Procedures
- Functions

sys

università

- Tables
- studenti

Columns

- Matricola
- Nome
- Cognome
- DataNascita

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

world

1 • SELECT * FROM università.studenti;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	Matricola	Nome	Cognome	DataNascita
	124567	Mario	Rossi	13/08/2001
	124558	Alberto	Bianchil	NULL
*	NULL	NULL	NULL	NULL

Popolamento di una tabella

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

	Matricola	Nome	Cognome	DataNascita
	124506	Marco	Neri	1998-11-02
	124558	Alberto	Bianchi	2002-07-02
	124567	Mario	Rossi	2001-08-13
	124578	Franco	Rossi	1999-09-12
▶	124588	Massimo	Guidi	2002-03-28
*	NULL	NULL	NULL	NULL

Result Grid
Form Editor
Field Types
Query Stats

studenti 1 × Apply Revert

Popolamento di una tabella

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

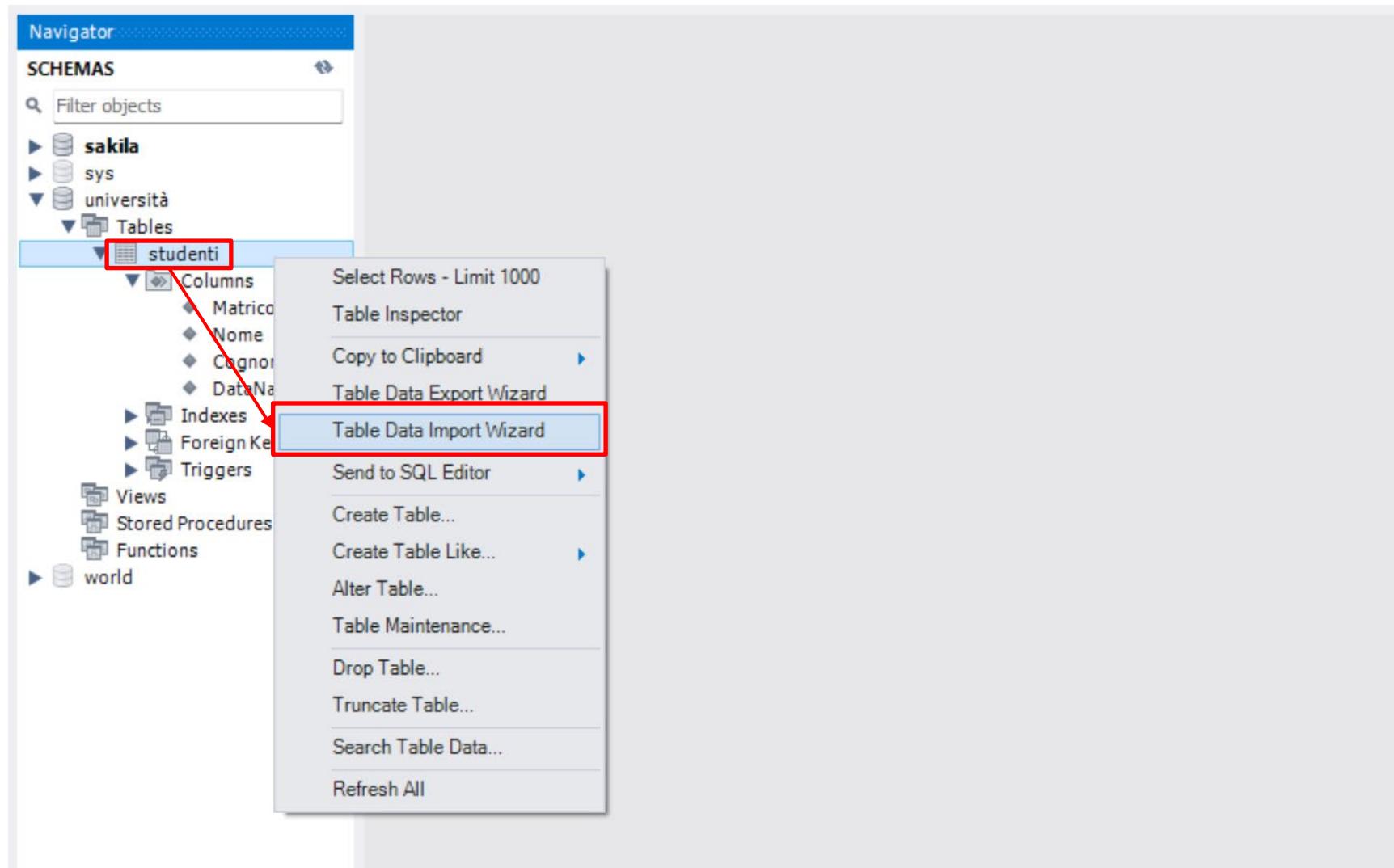
```
1  INSERT INTO `università`.`studenti` (`Matricola`, `Nome`, `Cognome`, `DataNascita`) VALUES ('124567', 'Mario', 'Rossi', '13/08/2001');
2  INSERT INTO `università`.`studenti` (`Matricola`, `Nome`, `Cognome`, `DataNascita`) VALUES ('124558', 'Alberto', 'Bianchi', '15/07/2002');
3  INSERT INTO `università`.`studenti` (`Matricola`, `Nome`, `Cognome`, `DataNascita`) VALUES ('124578', 'Franco', 'Rossi', '12/09/1999');
4  INSERT INTO `università`.`studenti` (`Matricola`, `Nome`, `Cognome`, `DataNascita`) VALUES ('124506', 'Marco', 'Neri', '02/11/1998');
5  INSERT INTO `università`.`studenti` (`Matricola`, `Nome`, `Cognome`, `DataNascita`) VALUES ('124588', 'Massimo', 'Guidi', '28/03/2002');
6
```

Back

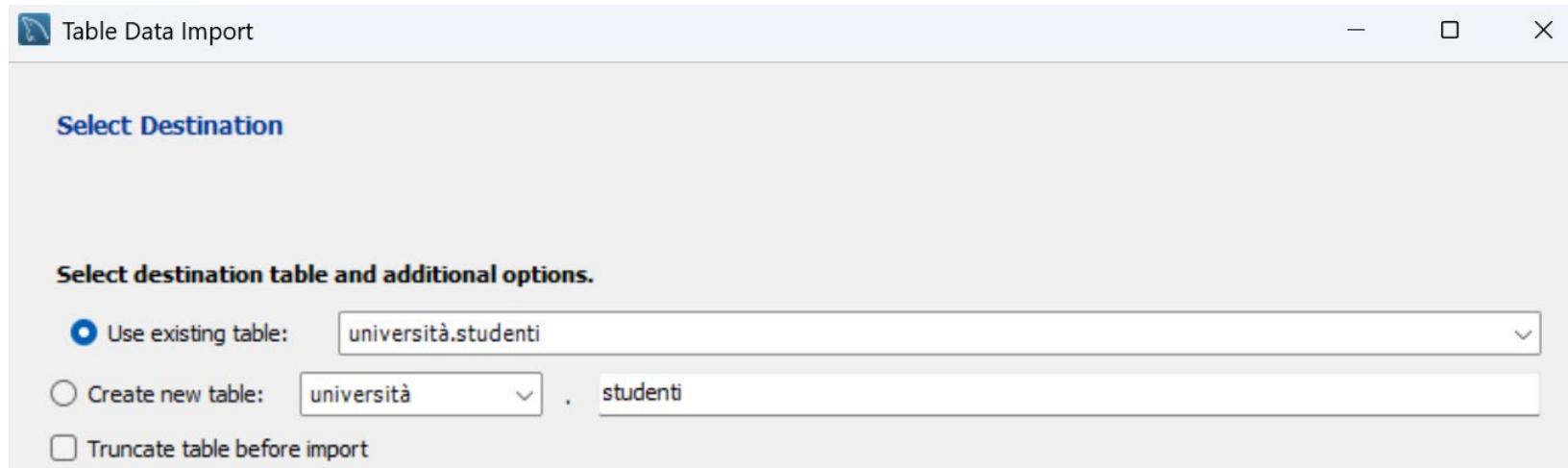
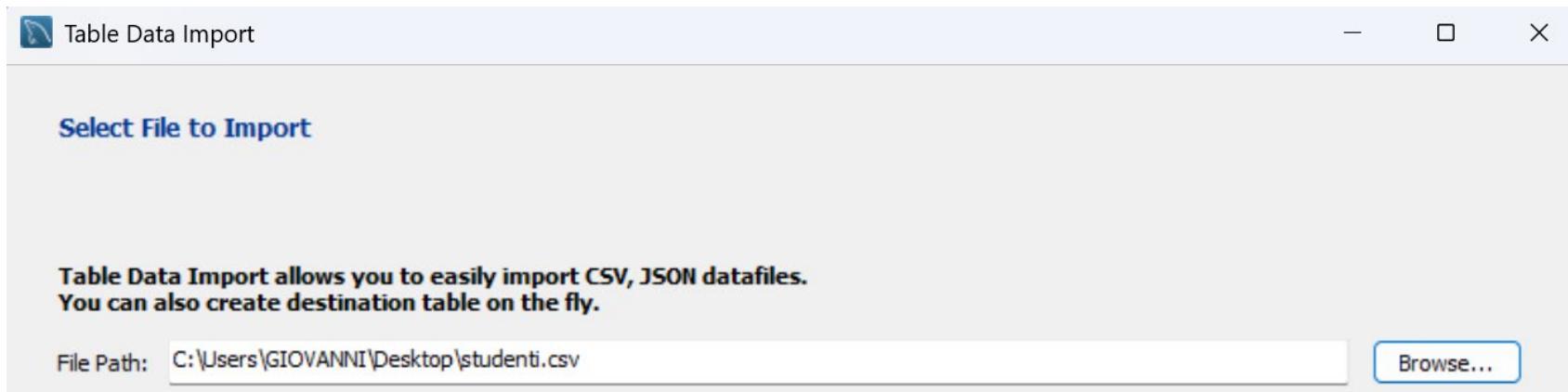
Apply

Cancel

Popolamento di una tabella da file di dati



Popolamento di una tabella da file di dati



Popolamento di una tabella da file di dati

Table Data Import

Configure Import Settings

Detected file format: csv 

Encoding: utf-8

Columns:

Source Column	Dest Column
<input checked="" type="checkbox"/> Matricola	Matricola
<input checked="" type="checkbox"/> Nome	Nome
<input checked="" type="checkbox"/> Cognome	Cognome
<input checked="" type="checkbox"/> DataNascita	DataNascit

Matricola	Nome	Cognome	DataNascita
124784	Francesco	Russo	2002-02-23
121357	Alberto	Pappalardo	1999-03-01
124644	Giada	Sapienza	1998-09-12

Popolamento di una tabella da file di dati

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the database schema. The 'università' schema is expanded, showing the 'studenti' table. The 'Columns' section lists 'Matricola', 'Nome', 'Cognome', and 'DataNascita'. The 'Tables' section also lists 'Indexes', 'Foreign Keys', and 'Triggers'. Below these are 'Views', 'Stored Procedures', and 'Functions'. The main workspace shows a 'Result Grid' with a red border around the data area. A SQL query window above the grid displays the command: 'SELECT * FROM università.studenti;'. The result grid contains the following data:

	Matricola	Nome	Cognome	DataNascita
▶	121357	Alberto	Pappalardo	1999-03-01
	124506	Marco	Neri	1998-11-02
	124558	Alberto	Bianchi	2002-07-02
	124567	Mario	Rossi	2001-08-13
	124578	Franco	Rossi	1999-09-12
	124588	Massimo	Guidi	2002-03-28
	124644	Giada	Sapienza	1998-09-12
	124784	Francesco	Russo	2002-02-23
*	NULL	NULL	NULL	NULL

Interrogazioni SQL

MySQL Workbench

Local instance MySQL95 ×

File Edit View Query Database Server Tools Scripting Help

Schemas

università - Schema studenti - Table studenti studenti - Table studenti studenti SQL File 3*

Create a new SQL tab for executing queries

Filter objects

sakila

- Tables
- Views
- Stored Procedures
- Functions

sys

università

- Tables
 - studenti
 - Columns
 - Matricola
 - Nome
 - Cognome
 - DataNascita
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions

world

Administration Schemas

SQL Editor

1 • **SELECT Matricola**
2 **FROM studenti**
3 **WHERE Cognome = "Rossi"**

The screenshot shows the MySQL Workbench interface. The title bar indicates a local instance of MySQL95. The main window displays a tree view of database schemas: sakila, sys, università, and world. The università schema is expanded, showing its tables, columns, indexes, foreign keys, triggers, views, stored procedures, and functions. The studenti table is selected. In the center, there is a SQL editor pane containing three lines of SQL code: 'SELECT Matricola', 'FROM studenti', and 'WHERE Cognome = "Rossi"'. The third line is highlighted with a red box. A toolbar above the editor contains various icons for database management tasks.

Risultato dell'interrogazione

università - Schema studenti - Table studenti studenti - Table studenti SQL File 3*

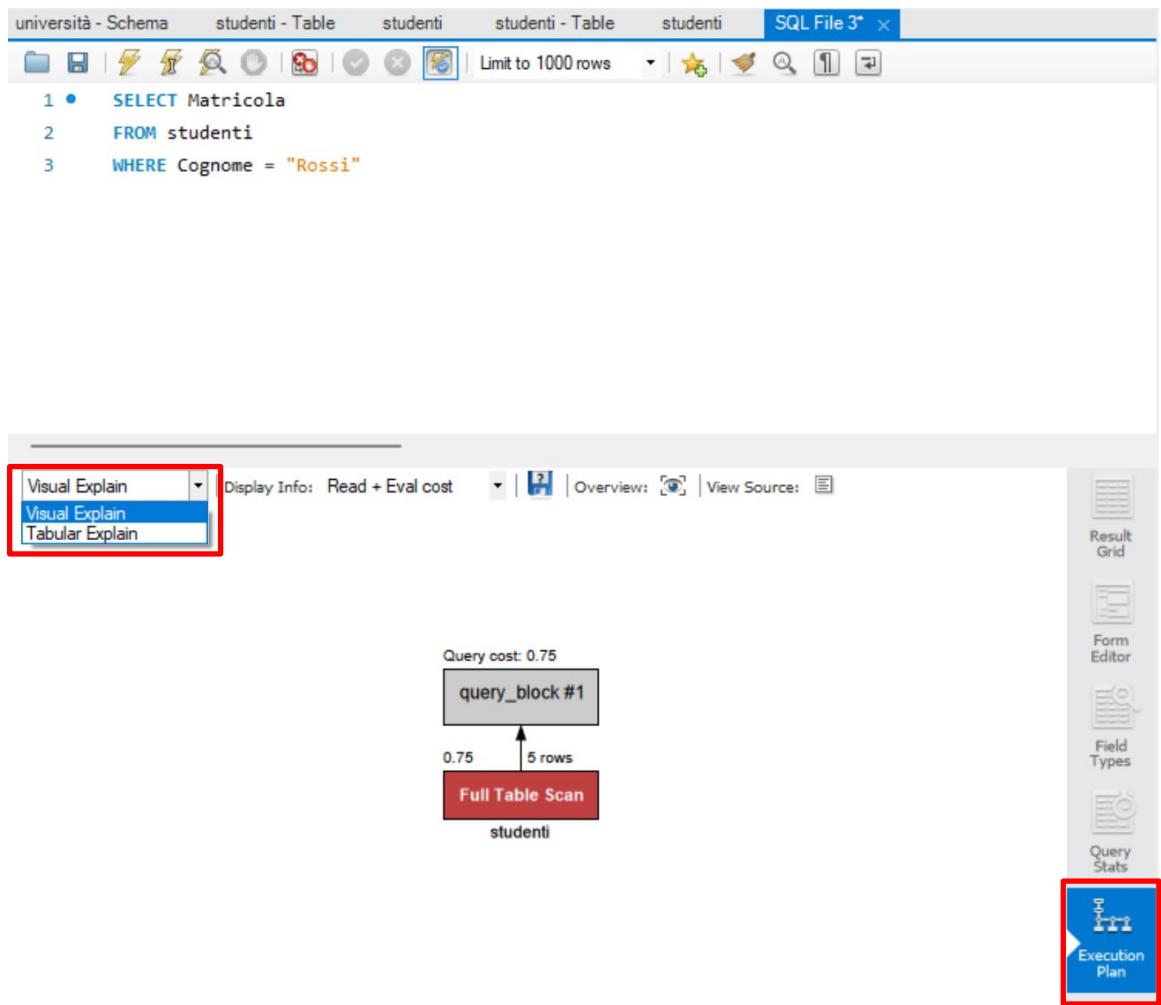
1 • `SELECT Matricola
FROM studenti
WHERE Cognome = "Rossi"`

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Matricola
124567
124578
NULL

Execution plan della query

- In che modo è stata eseguita la query?
- Sono state effettuate full scan sulle tabelle o sono state usate chiavi e/o indici?
- Quante righe sono state generate ad ogni passo?
- Modalità grafica o testuale/tabulare (equivalente alla EXPLAIN)
- Strumento utile per capire se e dove ottimizzare le query



Un esempio più complesso

università - Schema studenti - Table studenti studenti - Table studenti SQL File 3*

```
1 •   SELECT CONCAT(customer.last_name, ', ', customer.first_name)
2       AS customer, address.phone, film.title FROM rental
3   INNER JOIN customer ON rental.customer_id = customer.customer_id
4   INNER JOIN address ON customer.address_id = address.address_id
5   INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
6   INNER JOIN film ON inventory.film_id = film.film_id
7   WHERE rental.return_date IS NULL
8   AND rental_date + INTERVAL film.rental_duration DAY < CURRENT_DATE()
9   LIMIT 5;
```

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content: | Fetch rows: |

	customer	phone	title
▶	KNIGHT, GAIL	904253967161	HYDE DOCTOR
	MAULDIN, GREGORY	80303246192	HUNGER ROOF
	JENKINS, LOUISE	800716535041	FRISCO FORREST
	HOWELL, WILLIE	991802825778	TITANS JERK
	DIAZ, EMILY	333339908719	CONNECTION MICRO COSMOS

Result Grid
Form Editor
Field Types
Query Stats
Execution Plan

Un esempio più complesso

università - Schema studenti - Table studenti studenti - Table studenti studenti SQL File 3*

1 • SELECT CONCAT(customer.last_name, ', ', customer.first_name)
2 AS customer, address.phone, film.title FROM rental
3 INNER JOIN customer ON rental.customer_id = customer.customer_id
4 INNER JOIN address ON customer.address_id = address.address_id
5 INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
6 INNER JOIN film ON inventory.film_id = film.film_id
7 WHERE rental.return_date IS NULL
8 AND rental_date + INTERVAL film.rental_duration DAY < CURRENT_DATE()
9 LIMIT 5;

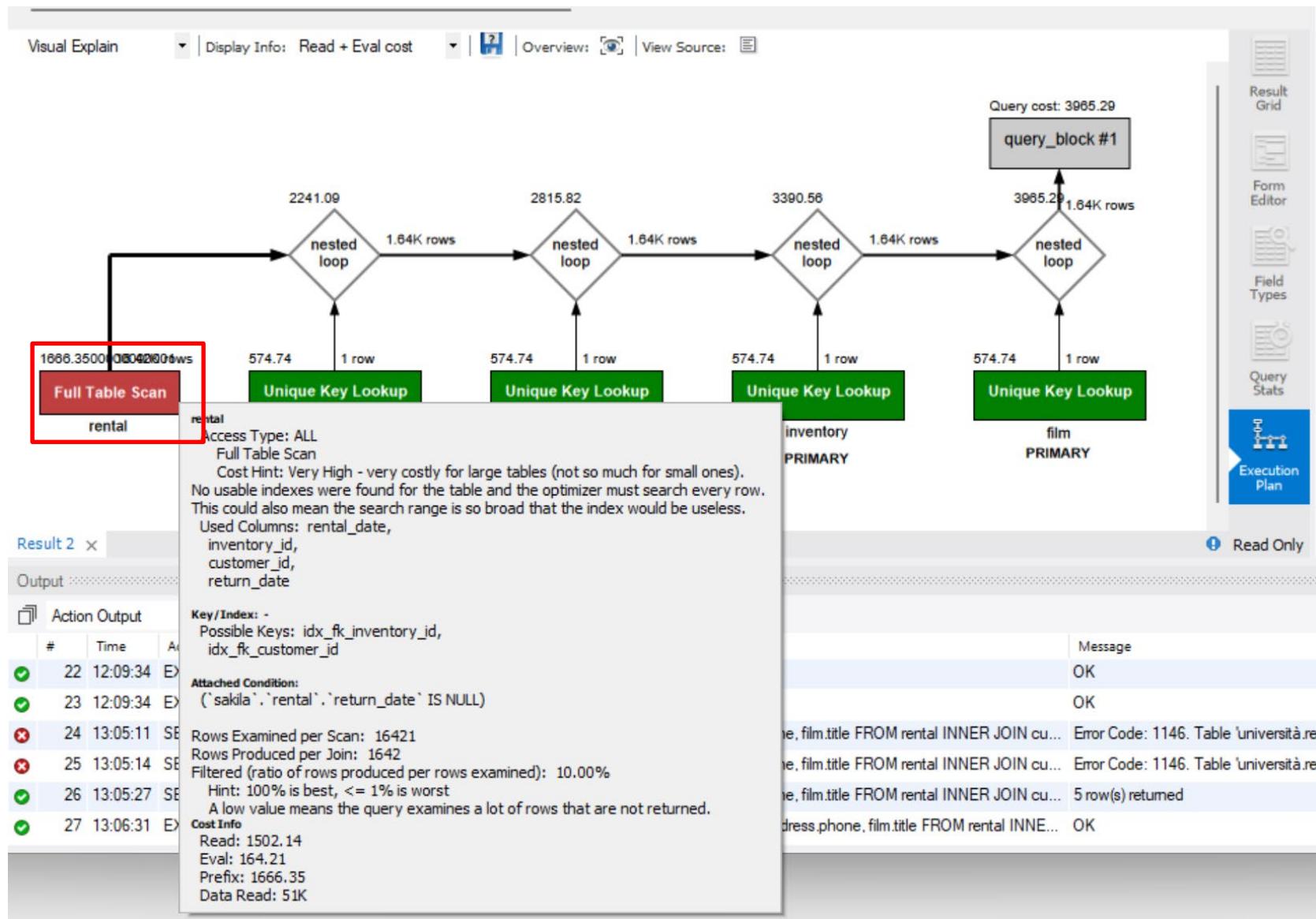
Visual Explain | Display Info: Read + Eval cost | Overview: | View Source: |

Query cost: 3965.29
query_block #1

```
graph LR; A[Full Table Scan  
rental] --> B((nested loop)); B -- "2241.09  
1.64K rows" --> C((nested loop)); C -- "2815.82  
1.64K rows" --> D((nested loop)); D -- "3390.56  
1.64K rows" --> E((nested loop)); E -- "3965.29  
1.64K rows" --> F[query_block #1]; B -- "1 row" --> B_UK[Unique Key Lookup  
customer  
PRIMARY]; C -- "1 row" --> C_UK[Unique Key Lookup  
address  
PRIMARY]; D -- "1 row" --> D_UK[Unique Key Lookup  
inventory  
PRIMARY]; E -- "1 row" --> E_UK[Unique Key Lookup  
film  
PRIMARY]
```

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Un esempio più complesso



Stored procedures

The screenshot shows the MySQL Workbench interface with the 'università - Schema' selected. In the Navigator pane, under the 'sakila' schema, the 'Stored Procedures' node is expanded, showing three procedures: 'film_in_stock', 'film_not_in_stock', and 'rewards_report'. A red square highlights the 'film_in_stock' icon. The main pane displays the SQL DDL code for the 'film_in_stock' procedure.

Name: film_in_stock

DDL:

```
1 • CREATE DEFINER=`root`@`localhost` PROCEDURE `film_in_stock`(IN p_film_id INT, IN p_store_id INT, OUT p_film_count INT)
2     READS SQL DATA
3     BEGIN
4         SELECT inventory_id
5             FROM inventory
6             WHERE film_id = p_film_id
7             AND store_id = p_store_id
8             AND inventory_in_stock(inventory_id);
9
10        SELECT COUNT(*)
11            FROM inventory
12            WHERE film_id = p_film_id
13            AND store_id = p_store_id
14            AND inventory_in_stock(inventory_id)
15            INTO p_film_count;
16    END
```

Stored procedures

Navigator: università - Schema studenti - Table studenti studenti - Table studenti SQL File 3* film_in_stock - Routine ×

The name of the routine. The DDL statement.

SCHEMAS

Filter objects

sakila

- Tables
- Views
- Stored Procedures
 - film_in_stock
 - film_not_in_stock
 - rewards_report
- Functions

sys

università

world

Name: film_in_stock

DDL:

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `film_in_stock`(IN p_film_id INT, IN p_store_id INT, OUT p_film_count INT)
2     READS SQL DATA
3     BEGIN
4         SELECT inventory_id
5             FROM inventory
6             WHERE film_id = p_film_id
7             AND store_id = p_store_id
8             AND inventory_in_stock(inventory_id);
9
10        SELECT COUNT(*)
11            FROM inventory
12            WHERE film_id = p_film_id
13            AND store_id = p_store_id
14            AND inventory_in_stock(inventory_id)
15            INTO p_film_count;
16    END
```

Call stored procedure sakila.film_in_stock

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

p_film_id	1	[IN]	INT
p_store_id	1	[IN]	INT
p_film_count	count	[OUT]	INT

Execute Cancel

Stored procedures

The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Includes icons for file operations, search, and database navigation.
- Tab Bar:** Shows tabs for "università - Schema", "studenti - Table", "studenti", "studenti - Table", "studenti", "SQL File 3*", "film_in_stock - Routine", and "film_in_stock".
- Query Editor:** Displays the following SQL code:

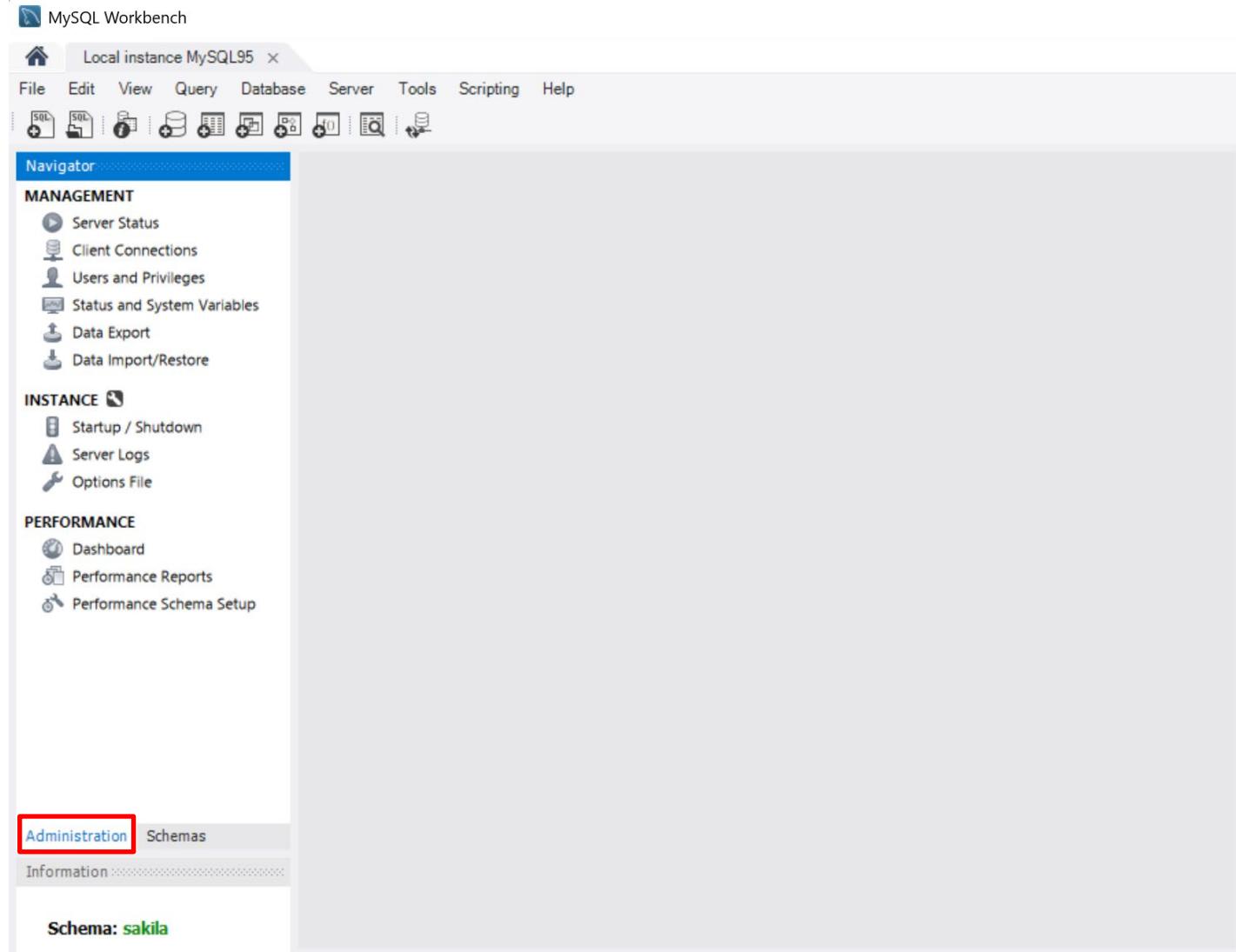
```
1 • set @p_film_count = 0;
2 • call sakila.film_in_stock(1, 1, @p_film_count);
3 • select @p_film_count;
4
```

The lines from 2 to 3 are highlighted with a red box.
- Result Grid:** Shows the output of the query:

Result Grid
@p_film_count
4

The entire grid is highlighted with a red box.

Pannello «Administration»



Dashboard

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard**
- Performance Reports
- Performance Schema Setup

Administration - Dashboard

Network Status

Statistics for network traffic sent and received by the MySQL Server over client connections.

Incoming Network Traffic (Bytes/Second)

receiving 8.00 B/s

Outgoing Network Traffic (Bytes/Second)

sending 5.32 KB/s

Client Connections (Total)

limit 151

MySQL Status

Primary MySQL Server activity and performance statistics.

Table Open Cache

Efficiency
97%

InnoDB Status

Overview of the InnoDB Buffer Pool and disk activity generated by the InnoDB storage engine.

InnoDB Buffer Pool

Usage
19%

read reqs.
0 pages/s

write reqs.
0 pages/s

disk reads
0 #/s

Redo Log

- data written 0 B/s
- writes 0 #/s

InnoDB Disk Writes

writing 0.00 B/s

Doublewrite Buffer

- writes 0 /s

InnoDB Disk Reads

reading 0.00 B/s

Administration Schemas

Information

Schema: sakila

Client connessioni

Navigator Administration - Client Connect... x

MANAGEMENT

- Server Status
- Client Connections**
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Local instance MySQL95

Client Connections

Threads Connected: 4 Threads Running: 2 Threads Created: 4 Threads Cached: 0 Rejected (over limit): 0

Total Connections: 25 Connection Limit: 151 Aborted Clients: 1 Aborted Connections: 1 Errors: 0 ⓘ

ID	User	Host	DB	Command	Time	State	Thread ID	Type	Name	Parent Process	Instrumented	Info
5	event_scheduler	localhost	None	Daemon	134399	Waiting on event	44	FOREGROUND	thread/sql/event	1	YES	NULL
7	None	None	None	Daemon	134399	Suspending	47	FOREGROUND	thread/sql/conn	1	YES	NULL
20	root	localhost	sakila	Sleep	423	None	61	FOREGROUND	thread/sql/others	0	YES	NULL
21	root	localhost	sakila	Sleep	473	None	62	FOREGROUND	thread/sql/others	48	YES	NULL
24	root	localhost	None	Query	0	executing	65	FOREGROUND	thread/sql/others	0	YES	SELECT t.PROCESSLIST_ID,IF
25	root	localhost	None	Sleep	1	None	66	FOREGROUND	thread/sql/others	0	YES	NULL

Administration Schemas

Information

Utenti e privilegi

Screenshot of the MySQL Workbench Administration - Users and Privileges interface.

The left sidebar shows the Navigator with sections: MANAGEMENT (Server Status, Client Connections, **Users and Privileges**, Status and System Variables, Data Export, Data Import/Restore), INSTANCE (Startup / Shutdown, Server Logs, Options File), and PERFORMANCE (Dashboard, Performance Reports, Performance Schema Setup). The Administration and Schemas tabs are selected in the bottom-left.

The main window title is "Administration - Users and Privileges". It displays "Local instance MySQL95" and "Users and Privileges".

The "User Accounts" table lists:

User	From Host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost

The "Select an account to edit or click [Add Account] to create a new one" section contains tabs: Login, Account Limits, Administrative Roles, Schema Privileges. The "Login" tab is active.

Form fields in the "Login" tab:

- Login Name:
- Authentication Type: (Note: For standard password and host-based authentication, select 'Standard').
- Limit to Hosts Matching: (% and _ wildcards may be used)
- Password: (Type a password to reset it. Note: Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.)
- Confirm Password: (Enter password again to confirm.)
- Expire Password:
- Authentication String: (Authentication plugin specific parameters.)

Buttons at the bottom: Add Account, Delete, Refresh, Revert, Apply.

Export di database

Screenshot of the MySQL Workbench Administration - Data Export interface.

Left Sidebar:

- MANAGEMENT**
 - Server Status
 - Client Connections
 - Users and Privileges
 - Status and System Variables
 - Data Export** (highlighted with a red box)
 - Data Import/Restore
- INSTANCE**
 - Startup / Shutdown
 - Server Logs
 - Options File
- PERFORMANCE**
 - Dashboard
 - Performance Reports
 - Performance Schema Setup

Administration - Data Export

Object Selection tab selected.

Tables to Export section:
Schema: sakila, sys, università, world
The "sakila" schema is highlighted with a red box.

Export Options section:

- Export to Dump Project Folder**: C:\Users\GIOVANNI\Documents\dumps\Dump20251108
- Export to Self-Contained File** (selected): C:\Users\GIOVANNI\Documents\dumps\Dump20251108.sql

The "Export to Self-Contained File" option is highlighted with a red box.

Export Type dropdown menu:

- Dump Structure and Data (selected)
- Dump Structure and Data
- Dump Data Only
- Dump Structure Only
- Dump Triggers

The "Dump Structure and Data" option is highlighted with a red box.

Buttons:

- Refresh
- Select Views
- Select Tables
- Unselect All
- Start Export

Bottom status message: Press [Start Export] to start...

Import di database

The screenshot shows the MySQL Workbench interface for managing a MySQL 9.5 instance. The left sidebar contains navigation links for Management, Instance, and Performance. The main window is titled "Administration - Data Import/Res..." and shows the "Data Import" tab for "Local instance MySQL95".

Import Options: A red box highlights the "Import Options" section. It contains two radio button options: "Import from Dump Project Folder" (disabled) and "Import from Self-Contained File" (selected). The selected file is "C:\Users\GIOVANNI\Documents\exports\export.sql". Below this, a note states: "Select the SQL/dump file to import. Please note that the whole file will be imported."

Default Schema to be Imported To: This section includes a "Default Target Schema:" dropdown and a "New..." button. A note to the right explains: "The default schema to import the dump into. NOTE: this is only used if the dump file doesn't contain its schema, otherwise it is ignored."

Select Database Objects to Import: This section allows selecting objects from a project folder. It has two tabs: "Imp..." and "Schema". The "Schema Objects" tab is currently active.

Action Buttons: At the bottom right, there is a dropdown menu with options: "Dump Structure and Data" (selected), "Dump Structure and Data", "Dump Data Only", and "Dump Structure Only". To the right of the dropdown are buttons for "Select Views", "Select Tables", "Unselect All", and a large red-bordered "Start Import" button.

Table Information: On the far left, under "Administration Schemas", there is information for the "studenti" table. It lists columns: "Matricula" (int PK), "Nome" (varchar(45)), and "Cognome" (varchar(45)). Below this, a message says "Press [Start Import] to start...".