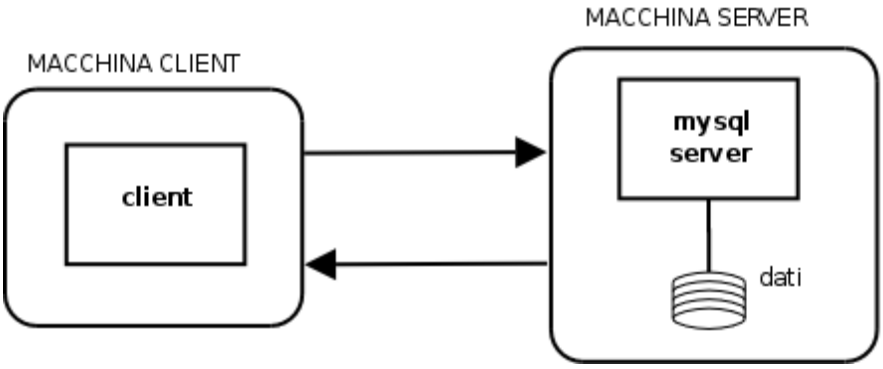
Laboratorio di Sistemi Informativi

Introduzione a MySQL
(modifica degli appunti di Gianluca Amato)

Il monitor di MySQL

MariaDB (MySQL) è un DBMS con architettura client/server.



Il vero e proprio DBMS è il programma server, che gestisce i database ed esegue i comandi SQL. Per collegarsi al server e impartire ad esso dei comandi, è necessario un programma client. Il più semplice di tutti è il **monitor** di MySQL (MariaDB). In Linux (in aula informatica) per far partire il monitor, entrare in una finestra terminale e lanciare il programma `mysql` (in Unix).

Noi useremo direttamente il programma MySQL Client(MariaDB.....).

In aula informatizzata bisogna dare il comando `mysql -h lamp -u <username> -p`, dove al posto di `<username>` va il nome utente che avete scelto per il vostro account sul server MySQL. Il parametro `-h <nomeserver>` consente di specificare su quale macchina risiede il server (lamp nel nostro caso). L'opzione `-u <username>` specifica con quale nome utente volete collegarvi e l'opzione `-p` serve a specificare che ci si vuole connettere utilizzano una password (che vi verrà richiesta dal sistema). Se l'account è privo di password, NON usare l'opzione `-p`.

Nel caso l'username che avete scelto per MySQL corrisponda al vostro username per l'accesso ai computer, allora basterà il comando `mysql -h lamp -p`

Il sistema risponde con qualcosa del tipo

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 11.3.0-MariaDB mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]>
```

e attende l'immissione dei comandi. I comandi devono esseri comandi SQL validi (riconosciuti da MariaDB - MySQL) e terminati da un punto e virgola. Anche se premete il tasto <Enter>, il sistema non esegue alcun comando finché non inserite anche un punto e virgola.

Vediamo alcuni comandi SQL di esempio:

- `SHOW SCHEMAS` (o SHOW DATABASES): mostra gli schemi (ovvero, le basi di dati o collezioni di tabelle) presenti sul server ai cui l'utente attuale ha diritto di accesso. Dopo l'installazione di MariaDB, ne sono visibili alcuni. Due in particolare world e sakila, saranno utili per effettuare esperimenti, (con accesso sia in lettura che in scrittura). Altri sono schemi di sistema. Ad es. l'`information_schema`, è uno schema informativo, che contiene le definizioni di tutti gli schemi e gli altri oggetti presenti nel sistema. Nella letteratura sulle basi di dati è chiamato spesso *dizionario dati* o *catalogo di sistema* (con accesso solo in lettura). Noi non lo useremo.

  In aula informatizzata vedrete invece tre o quattro schemi. Uno, con nome db_s<numeromatricola> (p.e. per la matricola 98500 il database si chiama db_s98500), è uno schema vuoto che potrete usare liberamente per svolgere le esercitazioni previste durante il corso. Infine, è presente lo schema `information_schema` di cui sopra. È possibile che vediate anche uno schema dal nome `airdb` che contiene dati utilizzati durante il corso nei precedenti anni accademici.

- `USE <nome_schema>` : seleziona uno schema come schema corrente. Tutte le operazioni SQL da quel momento agiranno all'interno dello schema selezionato.
- `SHOW TABLES`: mostra le tabelle che fanno parte dello schema corrente, impostato tramite il comando `USE`.
- `QUIT` (o il sinonimo `\q`) : esce dal monitor di MySQL.

Sono ovviamente supportati tutti (o quasi) i comandi SQL che studieremo nel corso di Basi di Dati: `SELECT`, `INSERT`, `DELETE`, `UPDATE`, `CREATE TABLE`.

LI VEDREMO IN SEGUITO

Ricordate che MariaDB> è il prompt di MariaDB! Non confondetelo con il prompt della shell! Dalla shell si possono invocare altri programmi e dare comandi al sistema operativo. Dal prompt di MariaDB, invece, si possono inserire dei comandi in SQL che vengono eseguiti dal server. Se provate un comando come `ls` il sistema risponde con

```
MariaDB> ls;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'ls' at line 1
```

Questo è un messaggio di errore standard di MariaDB. C'è un numero che identifica il tipo di errore seguito dalla sua descrizione. Il primo numero (subito dopo `ERROR`) è un codice di errore specifico di MariaDB, mentre il valore tra parentesi è un codice di errore standard definito dello standard SQL.

Attenzione. Se non si termina il comando con un punto e virgola, il monitor di MariaDB risponde con il prompt

```
->
```

e richiede di completare il comando. Questo perché, per MariaDB, un comando non è finito finché non appare il punto e virgola! In realtà, ci sono delle eccezioni. Ad esempio `USE` funziona anche senza punto e virgola finale. La differenza è che `USE` viene interpretato dal monitor di MariaDB, mentre gli altri comandi sono interpretati dal server. Onde evitare problemi, basta mettere sempre il punto e virgola alla fine.

Una sessione di lavoro col il monitor di MariaDB

Creiamo innanzitutto uno schema (o database) `test` con il comando

```
CREATE SCHEMA test; oppure  CREATE DATABASE test;
```

Uno schema è fondamentalmente un insieme di tabelle. All'inizio lo schema `test` è vuoto, non contiene neanche una tabella, il che non lo rende particolarmente utile... è il momento di iniziare a crearne qualcuna. Il comando SQL che fa a caso nostro è `CREATE TABLE`. Creiamo quindi una tabella per i dipendenti di un ufficio. Prima di tutto dobbiamo entrare nello schema, altrimenti il comando fallisce. Entriamo lì dentro con il comando `USE test;`

e successivamente:

```
CREATE TABLE dipendenti (
    codfiscale    char(16) primary key,
    nome          char(30),
    cognome       char(30),
    datanascita   date,
    anniserv      int default 0
);
```

In aula informatizzata utilizzare `USE` db_s<numerodimatricola> invece di `USE test`, in quanto lo schema `test` non è accessibile.

Il modo più comodo di inserire questi comandi è col "taglia e incolla". Il taglia è incolla in **Linux** si può usare in due modi distinti:

1. alla Windows: selezionare il testo da copiare sul browser, col tasto destro del mouse cliccare su *Copy*, poi spostarli sulla finestra e premere di nuovo il tasto destro del mouse: cliccare quindi su *Paste*.
2. in maniera abbreviata: selezionare col mouse il testo da copiare, spostarsi sulla finestra della shell e premere il tasto centrale del mouse.

Attenzione. I comandi di MariaDB possono essere digitati sia in maiuscolo che in minuscolo, e anche nel corso di queste lezioni verranno scritti ora in un modo ora nell'altro. Attenzione però che i nomi delle tabelle, degli schemi, dei campi nelle tabelle, etc.. potrebbero dover essere scritti esattamente con la giusta combinazione di maiuscole e minuscole con cui sono stati creati.

Ad esempio, per entrare nello schema test si può utilizzare `use test` o `USE test` ma non `use Test`.

È possibile inserire valori nella tabella con

```
INSERT INTO dipendenti VALUES ('codfisc1','mario','rossi','1960-12-
3',1);
```

Quando si inseriscono i valori in una tabella con il comando `insert`, dopo la parola chiave `values` vanno specificati tanti parametri quanti sono i campi nella tabella che si vuole modificare. Così, un comando del tipo

```
INSERT INTO dipendenti VALUES ('codfisc2','mario','bianchi');
```

termina con il messaggio di errore

```
ERROR 1136 (21S01): Column count doesn't match value count at row 1
```

Se non si vogliono specificare tutti i campi, ma solo alcuni di essi, è possibile usare un comando del tipo

```
INSERT INTO dipendenti(codfiscale,nome,cognome) VALUES
('codfisc2','mario','bianchi');
```

dove si indica, dopo il nome della tabella, i nomi dei campi che si vogliono specificare. Al solito, dopo `values` bisognerà specificare tanti elementi quanti sono i campi indicati dopo il nome della tabella. I campi che non sono indicati verranno riempiti con valori di default.

Possiamo vedere quello che abbiamo inserito con il comando `select`. Ad esempio:

```
mysql> SELECT * FROM dipendenti;


+------------+-------+---------+-------------+----------+
| codfiscale | nome  | cognome | datanascita | anniserv |
+------------+-------+---------+-------------+----------+
| codfisc1   | mario | rossi   | 1960-12-03  |        1 |
| codfisc2   | mario | bianchi | NULL        |        0 |
+------------+-------+---------+-------------+----------+
2 rows in set (0.00 sec)
```

Notare i valori di default per i campi `datanascita` e `anniserv`.

Attenzione. Un altro prompt che talvolta compare è

```
  '>
```

Questo vuol dire che nella riga precedente avete aperto degli apici, per inserire una stringa, senza poi chiuderli. Finché non inserite un altro apice, il sistema continuerà a riprovorvi questo prompt, e tutto quello che digitate (compresi i caratteri di "andata a capo") verrà a far parte della stringa. Nella maggior parte dei casi, quando vi trovate in questa situazione, avete sbagliato qualcosa. La soluzione migliore, allora, è premere prima un apice (per chiudere la stringa) e poi terminare il comando con `\c` invece che con il punto e virgola. Se un comando termina con \c esso non viene eseguito.

Talvolta è interessante vedere non il contenuto della tabella, ma la descrizione dei campi che la compongono. Si possono allora usare i seguenti comandi:

- `DESCRIBE <nometabella>` : visualizza la descrizione dei campi che compongono la tabella;
- `SHOW CREATE TABLE <nometabella>`: visualizza il comando `CREATE TABLE` che è necessario per ricostruire la tabella.

Ad esempio, `DESCRIBE dipendenti` restituisce il seguente risultato:

```
+-------------+------------+------+-----+---------+-------+
| Field       | Type       | Null | Key | Default | Extra |
+-------------+------------+------+-----+---------+-------+
| codfiscale  | char(16)   | NO   | PRI | NULL    |       |
| nome        | char(30)   | YES  |     | NULL    |       |
| cognome     | char(30)   | YES  |     | NULL    |       |
| datanascita | date       | YES  |     | NULL    |       |
| anniserv    | int(11)    | YES  |     | 0       |       |
+-------------+------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

I campi `field` e `type` sono autoesplicativi. Il campo `Null` indica se il corrispondente attributo può assumere valori nulli oppure no, il campo `Key` specifica qual è la chiave primaria della tabella, e il campo `Default` indica i valori di default dei vari attributi. Vedremo in futuro il significato del campo `Extra`.

Per quanto riguarda l'output del comando `SHOW CREATE TABLE dipendenti`, ci sono da notare due cose:

1. l'output è "sporcato" dalla presenza di vari caratteri che dovrebbero costituire l'intelaiatura di una tabella (come in `DESCRIBE <nometabella>`) ma che, a causa della dimensione eccessiva di quest'ultima, si trasformano solo in fonte di confusione; lo stesso inconveniente si verifica anche con il comando `SELECT` quando si vuole visualizzare un tabella dotata di troppe colonne;
2. il comando visualizzato non è lo stesso di quello che è stato dato per creare la tabella, perché vengono visualizzati anche i parametri opzionali (come `default`) che avevamo omesso.

Per quanto riguarda il primo problema, è possibile risolverlo sostituendo al punto e virgola finale la combinazione `\G`. Questa indica al monitor MySQL di visualizzare il risultato del comando in un altro formato, più adatto nel caso di tabelle particolarmente "larghe". Ad esempio, confrontare l'output normale di `SELECT * FROM dipendenti` con quello qui di seguito:

```
mysql> SELECT * FROM dipendenti\G
*************************** 1. row ***************************
 codfiscale: codfisc1
       nome: mario
    cognome: rossi
datanascita: 1960-12-03
   anniserv: 1
*************************** 2. row ***************************
 codfiscale: codfisc2
       nome: mario
    cognome: bianchi
datanascita: NULL
```

```
   anniserv: 0
2 rows in set (0.00 sec)
```

Errori e Warnings

Se a MySQL viene fornito un comando non valido, si verifica un errore che viene visualizzato sullo schermo. Ad esempio, questo è quello che succede se diamo un comando inesistente come `ls` o `prxy`. Talvolta si verifica nella esecuzione di un comando una condizione anomala ma non critica, per cui il comando viene eseguito lo stesso ma vengono generati dei *warning* (avvertimenti). Ad esempio, il comando

```
INSERT INTO dipendenti VALUES ('codfisc3','mario','rossi','1960-2-31',2);
```

genera un errore, causato dal fatto che non esiste il 31 febbraio 1960.

Due utili comandi per operare con errori e warning sono:

- `SHOW ERRORS`: visualizza l'elenco degli errori verificatisi nell'esecuzione dell'ultimo comando;
- `SHOW WARNINGS`: visualizza l'elenco dei warning e degli errori verificatisi nell'esecuzione dell'ultimo comando.

Ad esempio:

```
mysql> INSERT INTO dipendenti VALUES ('codfisc3','mario','rossi','1960-2-31',2);
```

ERROR 1292 (22007): Incorrect date value: '1960-2-31' for column 'datanascita' at row 1

In alternativa, il comando `WARNINGS` (o il sinonimo `\W`) dice al monitor di MySQL di visualizzare automaticamente i warnings quando si verificano. Il contrario è `NOWARNING` (o il sinonimo `\w`), che ripristina il comportamento di default.

Il comando HELP

Su MySQL è diponibile un comodo comando di aiuto: `HELP`. Esistono vari modi per utilizzarlo:

- `HELP`: visualizza i comandi che sono interpretati direttamente dal monitor di MySQL. Si può vedere, ad esempio, come i comandi `USE` e `QUIT` ricadano in questa categoria.

- `HELP CONTENTS`: visualizza un indice degli argomenti si quali è possibile chiedere un aiuto. Ad esempio, sulla mia macchina produce il seguente output:

  ```
  You asked for help about help category: "Contents"
  For more information, type 'help <item>', where <item> is one of the following
  categories:
     Account Management
     Administration
     Data Definition
  ```

```
Data Manipulation
Data Types
Functions
Functions and Modifiers for Use with GROUP BY
Geographic Features
Language Structure
Storage Engines
Stored Routines
Table Maintenance
Transactions
```

Si possono richiedere informazioni aggiuntive su uno degli argomenti col comando `HELP <argomento>`. Ad esempio, per informazioni sui tipi di dato supportati da MySQL, usare il comando `HELP Data Types`. I messaggi di aiuto per alcuni argomenti rimandano a loro volta a sotto-argomenti.

- `HELP <comando>`: visualizza informazioni sul comando specificato. Ad esempio, `HELP DROP TABLE` visualizza un messaggio di aiuto sul comando `DROP TABLE` (che serve ad eliminare una tabella). L'aiuto comprende sia una descrizione completa della sintassi del comando, che una spiegazione sommaria del suo funzionamento. È importante imparare a leggere la descrizione della sintassi, che utilizza alcune convenzioni particolare. Si consideri ad esempio l'output di `HELP DROP TABLE`:

  ```
  DROP [TEMPORARY] TABLE [IF EXISTS]
      tbl_name [, tbl_name] ...
      [RESTRICT | CASCADE]
  ```

  Notare che ci sono parole in maiuscolo e parole in minuscolo. Le parole in maiuscolo vanno scritte così come sono (ma, come abbiamo detto, si può usare una qualunque combinazione di maiuscole e minuscole). Le parole in minuscolo vanno invece sostituite con dei valori opportuni. In particolare `tbl_name` va sostituita con il nome della tabella che si vuole cancellare. Le parti tra parentesi quadre sono opzionali (si possono mettere oppure no). Le barre verticali separano delle alternative, tra le quali occorre scegliere. Infine, i tre puntini indicano che la parte precedente può essere ripetuta zero o più volte (in questo caso, stanno ad indicare che si possono specificare un qualunque numero di tabelle separate da virgole). Ad esempio, i seguenti comandi sono sintatticamente validi:
  - `DROP TABLE IF EXISTS dipendenti RESTRICT`
  - `dRoP TABLE dipendenti, prova`

  mentre quelli che seguono sono errati:
  - `DROP TABLE IF EXISTS dipendenti RESTRICT CASCADE`
  - `DROP TABLE RESTRICT`
  - `DROP dipendenti`

  Come ulteriore esempio, si consideri la sintassi del comando `DROP SCHEMA` (che serve a cancellare uno schema):

  ```
  DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
  ```

  Si noi l'uso della barra verticale tra parentesi graffe invece che quadre. Questo vuol dire che `DATABASE` e `SCHEMA` sono in alternativa l'uno con l'altra, ma che una delle due va messe obbligatoriamente.

Interfacce utente grafiche

Oltre al programma monitor, esistono altre interfacce utente per MySQL.
Queste interfacce, spesso di tipo grafico, facilitano il lavoro con MySQL, consentendo di

effettuare gran parte delle operazioni di amministrazione da un ambiente più confortevole che non la linea di comando.

Tra queste interfacce grafiche, quelle installate in aula informatizzata sono:

- *MySQL Administrator* (*Applications -> System Tools -> MySQL Administration*) è sviluppato per chi deve amministrare il server MySQL o uno dei suoi schemi. Consente facilmente di modificare, creare o distruggere schemi, tabelle, indici e utenti. Consente anche l'accesso a tutta una serie di parametri interni di MySQL che consentono di ottimizzarne le prestazioni.
- *MySQL Query Browser* (*Applications -> Programming -> MySQL Query Browser*) consente invece di operare facilmente con il contenuto delle tabelle, eseguendo interrogazioni e modificando i dati.

Ovviamente, come nel caso del monitor di MySQL, è possibile collegarsi a un server remoto: basta specificare il nome del computer a cui collegarsi, nome utente e password (se necessaria) da usare per il collegamento.

In realtà queste applicazioni sono state dismesse e sostituite con una applicazione molto più versatile denominata *MySQL Workbench*.

Infine, un'applicazione molto utilizzata per accedere ai server MySQL è *phpMyAdmin*. Si tratta di una applicazione web, da utilizzare con un browser, e quindi richiede la presenza sul computer di un software che faccia da server web. A causa di ciò l'installazione di *phpMyAdmin* potrebbe non essere tra le più semplici (ma io non lo uso, quindi non so darvi informazioni precise).