

**scritto da
Davide Cerbo**

MySQL

**X
TUTTI**

MYSQL PER TUTTI

di Davide Cerbo

*“Alcuni di voi per risolvere questo problema
ci metteranno mesi, altri tutta la loro vita”*

John Nash

INDICE

1. Le Basi
2. L'installazione
3. La configurazione
 - 3.1. Libreria necessaria per il richiamo in un linguaggio.
4. Il nostro primo database
 4. 1. Comandi SQL
5. Creare nuovi utenti in MySQL
6. I tipi disponibili in MySQL
7. Gli script: rendiamo il lavoro sequenziale
8. MySQL utilizzato in diversi linguaggi di programmazione: esempi pratici. (ASP, ASP.NET, PERL, PHP, Visual Basic, C/C++)
9. Conclusione e ringraziamenti

1. Le Basi

I database sono fondamentali nella programmazione di qualsiasi tipo di gestionali, questi permettono di organizzare in maniera ottimale i dati. Sui dati immagazzinati in un database si possono svolgere differenti operazioni, questi infatti possono essere modificati, aggiornati, aggiunti, eliminati o consultati, a seconda delle nostre necessità. Per svolgere queste operazioni si ricorre a un insieme di programmi detto *DBMS* (DataBase Management System) che in pratica è un software che fa da tramite tra l'utente e la base di dati, fornendo così una rappresentazione logica dei dati.

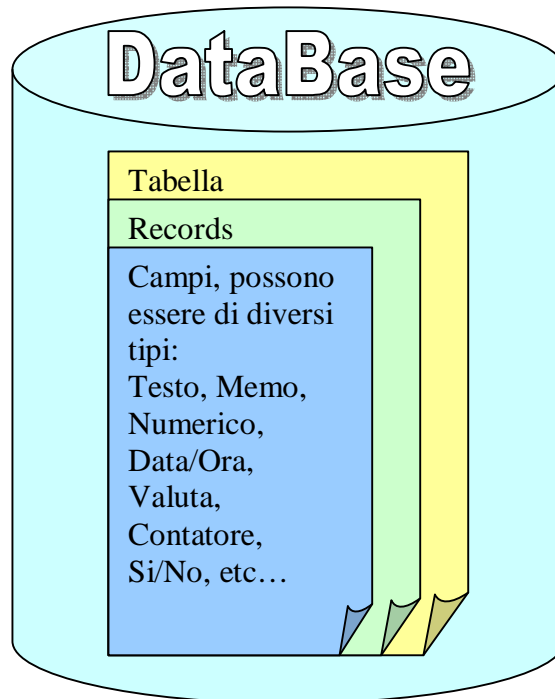
I DBMS forniscono diversi modi per rappresentare un database logicamente:

- Gerarchici.
- Ad oggetti.
- Relazionali.

Qui li ho elencati in ordine di apparizione sul mercato. I gerarchici hanno caratterizzato i primi DBMS sviluppati intorno alla metà degli anni 70. Questo sistema è basato sugli alberi. E' tutt'ora utilizzato anche se in pochi casi.

Il modello ad oggetti è dato da un insieme di classi, che definiscono le caratteristiche ed il comportamento degli oggetti del database. Questo permette di inserire nel database delle informazioni sulle operazioni da eseguire oltre ai dati. Ad oggi sfortunatamente manca un linguaggio standard per l'interrogazioni dei database basati sul modello ad oggetti.

Il modello relazionale nato da Codd nel 1970 è stato però iniziato ad utilizzare intorno al 1980. Questo modello è basato sul principio di insieme ed è il modello che noi useremo in questo tutorial. I dati sono organizzati in delle tabelle bidimensionali composte da un numero fisso di colonne dette attributi ed in genere sono etichettati e da un numero fisso di righe dette enuple. Le righe identificano un record, cioè un insieme di campi che corrispondono al valore della colonna in quella riga. Qui di seguito segue una rappresentazione grafica sommaria di un database; quella più in alto nella seguente lista è formata da quella seguente e così via:



Approfondiamo ora l'argomento DBMS. Elenchiamo ora i suoi componenti software basilari :

- **GESTORE DEGLI ACCESSI:** trasforma le operazioni di accesso ai dati in chiamate a procedure del gestore della memoria fisica.
- **PRECOMPILATORE DEL LINGUAGGIO OSPITE:** svolge un'analisi delle istruzioni del Database Definition Language (DDL) e ne ricava le specifiche chiamate alle procedure del DBMS e passa il codice trasformato al processore di interrogazioni.
- **PROCESSORE DI INTERROGAZIONI:** traduce le istruzioni dal linguaggio usato a quello del DBMS.
- **COMPILATORE DELLE DEFINIZIONI DEI DATI:** riceve le istruzioni dal linguaggio usato per definire la tipologia dei dati del database e genera il Dizionario dei dati.
- **GESTORE DELLA MEMORIA FISICA:** gestisce l'allocazione dei dati nella memoria di massa.
- **INTERFACCIA UTENTI:** permette all'utente di interagire col database.
- Il DBMS deve svolgere i seguenti compiti:
- Deve controllare in dati in modo da evitarne la ridondanza.
- Gestisce l'accesso ai dati attraverso uno schema concettuale, cioè deve fornire una organizzazione visiva del database indipendentemente dall'organizzazione fisica dei dati.
- Deve gestire l'accesso concorrente ai dati in modo da evitare blocchi e errori.

I database che prenderemo in considerazione in questa lezione sono quelli realizzati con MySQL è DBMS completamente gratuito ed estremamente veloce nato originariamente su UNIX, ma oggi presente anche su Linux e Windows. DBMS sta per DataBase Management System. Spieghiamo un po' cosa sono i DataBase:

Dopo questa piccola parte teorica elenchiamo gli strumenti che vi serviranno durante il tutorial:

- Il server MySQL (il nome del file in genere è: Mysql-shareware-3.xx.xx-win.zip (oppure la versione beta: Mysql-3.xx.xx-beta-win.zip. Per Linux sarà qualcosa tipo: mysql-x.xx.xx-pc-linux-gnu-xxxx.tar.gz).
- Un editor di testo (il notepad va benissimo).
- Tanta passione e un po' di tempo.

Per la prima voce troverete tutto su www.mysql.org per la seconda voce ormai tutti i sistemi operativi sono muniti di questo strumento (alcuni potrebbero preferire editor creati appositamente per MySQL, facilmente reperibili su molti siti web), per l'ultima voce sfortunatamente non so quali risorse consigliare.

2. L'installazione

Una volta effettuato il download passiamo finalmente all'installazione, qui di seguito presento l'installazione a seconda del diverso sistema operativo:

Windows 9x

Scompiattiamo il server SQL in una cartella di nome mysql (io terrò come a riferimento il drive c, quindi il percorso completo sarà c:\mysql). Apriamo il prompt di MS-DOS e scriviamo:

```
C:\> cd mysql
C:\mysql> cd bin
C:\mysql\bin>mysql -language=italian
(se usiamo la beta sostituiamo la riga di sopra con:
C:\mysql\bin>mysql -language=italian ).
```

Fatto questo siamo nella shell di MYSQL e possiamo iniziare a lavorare.

Windows 2000

Scompiattiamo il server SQL in una cartella di nome mysql (io terrò come a riferimento il drive c, quindi il percorso completo sarà c:\mysql). Apriamo il prompt di MS-DOS e scriviamo:

```
C:\> cd mysql
C:\mysql> cd bin
C:\mysql\bin> nysqld-nt – install
```

Tramite questo comando abbiamo creato un nuovo servizio sul server, ora non ci resta che avviarlo svolgendo le seguenti semplici operazioni:

1. Apriamo il pannello di controllo.
2. Selezioniamo la cartella “Strumenti di Amministrazione”.
3. Avviamo la voce “Servizi”.
4. Ci apparirà una lista, scorriamola fino a quando non troveremo la voce “MySQL” se non dovesse essere presente o abbiamo errato il processo di installazione o non è ancora aggiornata la finestra, in questo caso premiamo F5 per aggiornare.
5. Una volta selezionato il servizio dalla lista il servizio clicchiamo col tasto destro del mouse e selezioniamo la voce “Proprietà”.
6. Ci apparirà una finestra clicchiamo su “Avvia” per avviare il servizio. Se vogliamo evitare di avviarlo a mano ad ogni avvio di Windows possiamo scegliere da “Tipo di avvio” la voce “Automatico”.

Ora MySQL è perfettamente installato.

Linux

1. Prima di tutto scaricate l’apposito pacchetto dal solito www.mysql.org
2. Scompiattate il file appena scaricato in una cartella (sempre di nome mysql per comodità)
3. Assicuratevi di essere amministratori del sistema e fate partire lo script di configurazione automatica tramite il comando:

```
./configure
```

4. Dopo una breve attesa per la configurazione automatica potete finalmente passare all'installazione vera e propria. Digitate prima

```
Make
```

e subito dopo

```
make install
```

5. Una volta installato create le tabelle di sistema. Per fare ciò spostatevi nella directory in cui avete installato Mysql, accedete alla sottodirectory /bin ed eseguite:

```
./mysqk_install_db
```

6. Infine non vi resta che far partire il demone del mysql digitando:

```
./safe_mysqld &
```

Il carattere & serve fa eseguire il demone in background.

7. Se non ricevete errori l'installazione è avvenuta con successo, se no ripete l'installazione da capo, forse avete saltato alcuni passaggi.

Ora possiamo finalmente eseguire il nostro MySQL anche sotto Linux. ATTENZIONE!!! Questo tutorial è stato scritto con il sistema Windows come riferimento, non ci dovrebbero essere differenze, ma ci tengo comunque a precisarlo.

3. La configurazione

Una volta installato presenterà un user di default di nome "root" ed una password vuota. Per inserire la password scrivete:

```
C:\mysql\bin>mysqladmin -u root password nomepassword
```

Naturalmente sostituiremo nomepassword con la password che vogliamo inserire.

Per cambiare la password in qualsiasi momento scriveremo:

```
C:\mysql\bin\>mysqladmin -u root -p password nuovapassword
```

Dopo la pressione del tasto “Invio” vi sarà richiesta la vecchia password.

Per creare un nuovo utente scriveremo invece:

Ora possiamo finalmente accedere all’SQL vero e proprio per inviare i comandi presenti nella tabella presente tra poche righe. Per accedere scriveremo:

```
C:\mysql\bin\>mysql -uroot -ppassword
```

Naturalmente root e password dovranno essere sostituite rispettivamente dal nostro username e dalla nostra password.

Se avete seguito bene le istruzioni MySQL si eseguirà. Verrà aperto una specie di prompt di MS-DOS dove al posto di “C:\>” troveremo scritto “mysql>”.

Questa interfaccia è a riga di comando. Ogni comando deve essere terminato da ; e da un invio. Il solo tasto invio non basta!.

Per uscire basterà digitare:

```
mysql> quit;
```

Tramite il comando “use” selezioniamo il database sul quale vogliamo lavorare:

```
mysql> use nomepagina;
```

Ma come faccio a sapere quale database utilizzare se non sappiamo quali sono presenti sul mio computer? Per fare questo basta ricorrere semplicemente al comando “show”. Tramite la sintassi:

```
mysql> show databases;
```

Visualizziamo tutti i database presenti sulla nostra macchina, mentre se vogliamo visualizzare le tabelle da cui è formato basterà scrivere:

```
mysql> show tables;
```

Mentre se vogliamo vedere i campi che formano una tabella scriveremo:

```
mysql> show columns from nome_tabella;
```

Una volta che ci siamo resi conto dei database presenti nel nostro database possiamo anche selezionare quello che preferiamo tramite il già visto comando “use” per poi procedere a interrogarlo, ma prima di procedere analizziamo le voci principali dell’help che ci apparirà digitando \h o help o \?:

clear o \c cancella il comando;

exit o quit o \q esce.

\g o ; invia il comando SQL;

\. o source esegue uno script (vedremo in seguito come crearli);

\s o status visualizza le informazioni di stato del server;

\u è sinonimo dell’use già visto in precedenza.

3.1 Libreria necessaria per il richiamo in un linguaggio.

Spesso per utilizzare MySQL in un sistema operativo sono necessarie delle librerie dette MyODBC. Per installare la dll sopra citata basta eseguire un normale setup. Se si hanno i sorgenti è necessario prima compilare il tutto tramite queste tre righe da scrivere nel prompt di MS-DOS:

```
C:\> cd myodbc3-src  
C:\> nmake -f Win_Makefile  
C:\> nmake -f Win_Makefile install
```

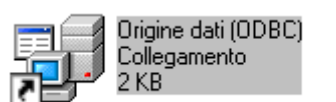
Perchè questa installazione vada a buon fine bisogna assicurarsi di aver installato MDAC, Microsoft Data Access SDK scaricabile da: www.microsoft.com/data/ . Se invece ci troviamo sotto Linux la storia si allunga di non poco, per questo vi rimando alla guida di questa libreria reperibile su http://www.mysql.org/products/myodbc/manual_toc.html . Qui troverete tutte le

informazioni che desiderate su questa libreria. Ho tralasciato Linux e non Windows perché questa guida è stata ideata pensando a Windows.

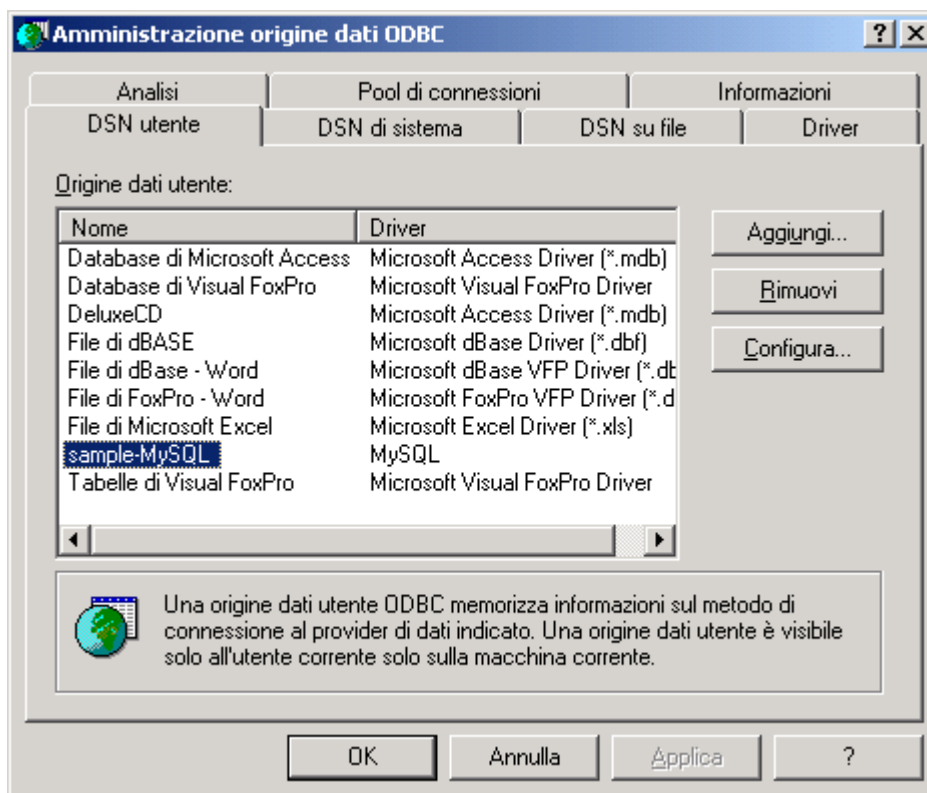
Se avete scaricato la versione già compilata basta scompattarla ed eseguire il setup.exe qui contenuto. Una volta terminata l'installazione aprite il Pannello di Controllo di Windows, cliccate su Strumenti di Amministrazione



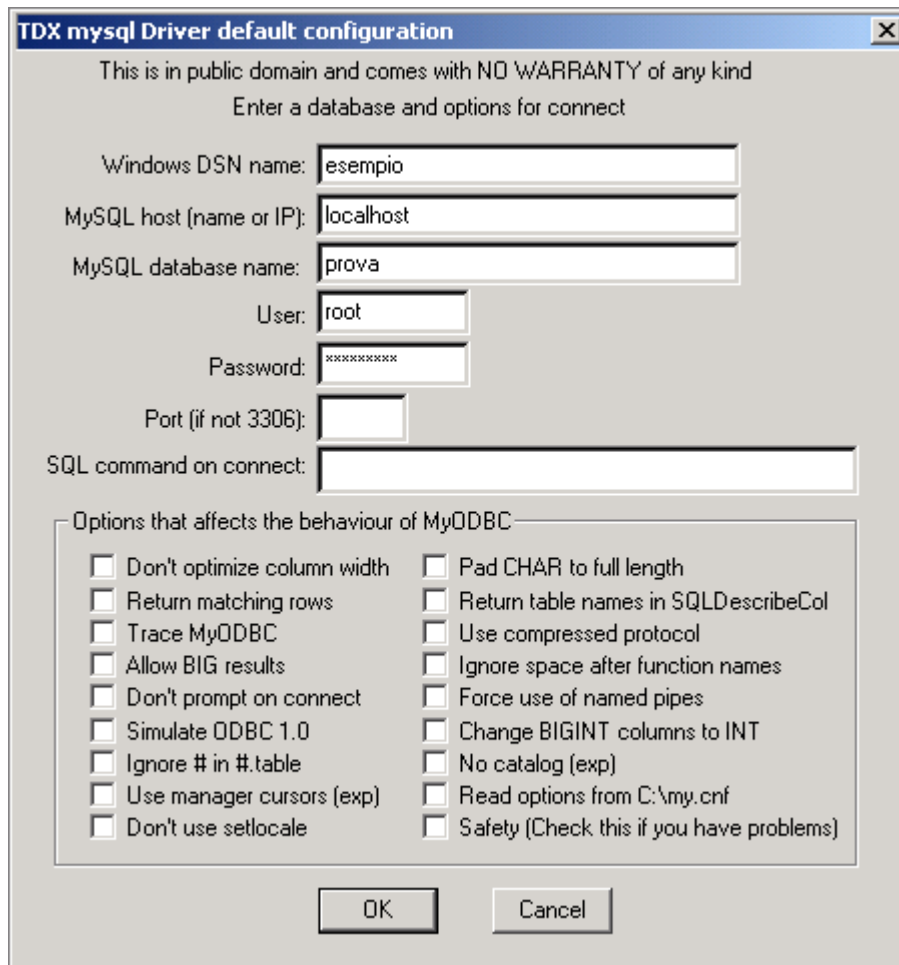
Poi cliccate su Origine Dati (ODBC)



Vi troverete davanti ad una finestra come la seguente:



Assicurateci dell'esistenza di questo DSN utente nel nostro sistema ne potremo creare uno nostro. Ci apparirà la seguente finestra:



Una volta completata potremo finalmente interrogare il nostro database MySQL tramite i linguaggi Microsoft TM compreso il nuovissimo ASP.NET. Le connessioni al database tramite i vari linguaggi le vedremo al punto sette di questo tutorial.

4. Il nostro primo database

Ora che abbiamo finalmente installato il nostro DBMS è giunto il momento di “giocarci” un po’. In questo paragrafo vedremo come creare un database, inserirci i dati e leggere i dati contenuti in esso. Per un riassunto dei principali comandi SQL vi rimando al paragrafo 4.1.

Per prima cosa creiamo il nostro database, per fare questo dobbiamo usare il comando CREATE DATABASE. Scriviamo quindi:

```
mysql> CREATE DATABASE Clienti;
```

Ora il nostro database esiste. Scriviamo un show databases per verificare che sia stato veramente creato. Selezioniamo poi il database appena creato scrivendo:

```
mysql> use Client;
```

Ora dobbiamo inserirci le tabelle. Visto che questo è un esempio creeremo una sola tabella di nome anagrafe in cui inseriremo solo tre campi: Nome, Telefono, Codice Fiscale. Inseriamo il codice fiscale perché essendo un identificativo univoco ci sarà molto utile nel caso noi volessimo fare ricerche nel nostro database. L'istruzione per creare il database è:

```
mysql> CREATE TABLE anagrafe(nome char(35) not null, telefono int not null, cod_fis char(16) not null);
```

Il not null indica che i campi non possono essere vuoti, omettendolo perdiamo questa clausola. Ora abbiamo creato la nostra tabella. Facciamo un show tables per vedere se tutto è andato per il giusto verso. Ora la tabella esiste ma è vuota. Inseriamoci dei dati. Per fare ciò ricorriamo all'istruzione INSERT INTO:

```
mysql> INSERT INTO anagrafe (nome, telefono, cod_fis) VALUES('Davide Cerbo', 0892419, 'CRBDVD...');
mysql> INSERT INTO anagrafe (nome, telefono, cod_fis) VALUES ('Laura Rispoli', 0892241, 'RSPLUR...');
mysql> INSERT INTO anagrafe (nome, telefono, cod_fis) VALUES ('Pinco Pallino', 0892324, 'PLLPNC...');
```

Notiamo alcune cose su questa istruzione. I valori dei campi dopo il VALUES vanno inseriti nell'ordine in cui sono citati i campi precedentemente. Le stringhe vanno inserite tra apici, i valori numerici no. Altra cosa che balza subito all'occhio è che queste istruzioni sono simili. Potrebbe risultare noioso scriverle tutte ogni volta. Per evitare questo basta usare i cursori della tastiera. Con le frecce su e giù si va alla riga precedente o successiva, mentre con le frecce destra e sinistra ci si sposta per la stringa, permettendoci quindi di modificare la stringa e rimandarla in esecuzione, senza doverla riscrivere da capo; più avanti vedremo un metodo ancora più veloce per ottimizzare i valori ripetitivi. Ora che i dati nel nostro database sono inseriti possiamo visualizzarli tramite l'istruzione SELECT. Scriviamo quindi...

```
mysql> SELECT * FROM anagrafe;
```

... e godiamoci lo spettacolo:).

Qui termina il nostro esempio guidato. Ora tocca alla vostra fantasia che potrete ampiamente sbizzarrire leggendo per bene il paragrafo 4.1 che è a dir poco fondamentale.

4.1. Comandi SQL

Nella tabella sottostante presento una panoramica sui principali comandi SQL. Il metodo migliore per capire i comandi naturalmente è fare pratica.

<i>PRINCIPALI COMANDI E FUNZIONI SQL</i>	
ALTER TABLE	Modifica la struttura di una tabella. Ha due clausole: ADD: aggiunge una colonna alla tabella, la sintassi è: “ALTER TABLE Tabella ADD NuovaColonna Char(1)” Char(1) è il tipo della colonna. Si può aggiungere anche INIT dopo il Char(1) per inizializzare la colonna. MODIFY: modifica le caratteristiche di una determinata colonna. La sintassi è: ALTER TABLE Tabella MODIFY NuovaColonna Char(1);
ASCII	Restituisce il valore ASCII di un carattere. Sintassi: ASCII('a');
ATAN	Calcola l'arcotangente del valore tra parentesi; Sintassi: SELECT ATAN(valore);
AVG	Da usare in combinazione con il SELECT restituisce il valore medio di una colonna specificata tra parentesi. Sintassi: SELECT AVG(Campo1) FROM tabella;
BETWEEN	Inserito in un SELECT vede se il valore di un campo è contenuto in un determinato range.

	<p>Esempio:</p> <pre>SELECT * FROM tabella WHERE campo BETWEEN A And Z;</pre>
CEILING	<p>Restituisce il valore integer minore che risulta maggiore o uguale all'espressione numerica specificata.</p> <p>Sintassi:</p> <pre>CEILING(valore);</pre> <p>Esempio:</p> <pre>SELECT CEILING(123.45), CEILING(\$-123.45);</pre> <p>Restituirà 124.00 e -123.00</p>
COMMIT	<p>Rende permanenti tutti gli eventuali aggiornamenti effettuati prima della scrittura di questo comando e naturalmente anche quelli successivi. La sintassi è molto semplice:</p> <pre>COMMIT;</pre>
CONCAT	<p>Concatena due stringhe. Sintassi:</p> <pre>CONCAT (stringa1, stringa2);</pre>
COS	<p>Funzione matematica che restituisce il coseno di un valore.</p> <p>Sintassi:</p> <pre>COS(valore)</pre>
COT	<p>Funzione matematica che restituisce la cotangente trigonometrica in radianti dell'angolo indicato nell'espressione float specificata.</p> <p>Sintassi:</p> <pre>COT(valore);</pre>
COUNT	<p>Conta i record della tabella.</p> <p>Esempio:</p> <pre>SELECT COUNT(*) FROM tabella;</pre> <p>Oppure</p> <pre>SELECT COUNT(*) FROM tabella WHERE <condizione>;</pre>
CREATE DATABASE	<p>Crea un database. La sintassi è:</p> <pre>CREATE DATABASE nome_database;</pre>
CREATE TABLE	<p>Crea una tabella.</p> <p>Esempio:</p> <pre>CREATE TABLE tabella(var1 char(15) not null, var2 char(20) not null);</pre> <p>Crea una tabella con due campi di nome var1 con una lunghezza di massimo 15 caratteri e var2 con una lunghezza massima di 20 caratteri; entrambi devono</p>

	essere non nulli.
DELETE	<p>Elimina uno o più record da una tabella.</p> <p>Sintassi:</p> <p>Delete * FROM tabella WHERE <condizione>;</p> <p>Esempio:</p> <p>Delete * FROM tabella WHERE id=1;</p>
DESCRIBE	<p>Fornisce la descrizione di una tabella, fornendo informazioni come i nomi dei campi ed altre notizie utili.</p> <p>Sintassi:</p> <p>DESCRIBE nomeTabella;</p>
DISTINCT	<p>Viene usato per selezionare i campi da un database escludendo i suoi duplicati.</p> <p>Esempio:</p> <p>SELECT DISTINCT campo FROM tabella;</p> <p>Oppure è usato in combinazione col Count per evitare di contare due volte i campi duplicati:</p> <p>Esempio:</p> <p>SELECT COUNT(DISTINCT campo) As varConta FROM tabella;</p>
DROP TABLE	<p>Cancella una tabella.</p> <p>Esempio:</p> <p>Drop Table tabella;</p>
FLOOR	<p>Restituisce il valore integer maggiore che risulta minore o uguale all'espressione numerica specificata.</p> <p>Sintassi:</p> <p>FLOOR(valore);</p> <p>Esempio:</p> <p>SELECT FLOOR(123.45), FLOOR(-123.45);</p> <p>Restituirà nell'ordine 123 e -124.</p>
GRANT	<p>Autorizza un utente ad accedere al nostro database.</p> <p>Sintassi:</p> <p>GRANT <tipo autorizzazione> ON <oggetto del database a cui si riferisce l'autorizzazione> TO <nome utente che si autorizza></p> <p>Esempio:</p> <p>GRANT SELECT ON tabella TO utente;</p>
GROUP BY	Da usare in combinazione con il SELECT raggruppa le voci in base a uno o più

	<p>campi da noi decisi. Sintassi:</p> <pre>SELECT * FROM tabella GROUP BY campo1...campoN;</pre> <p>In questo esempio invece lo combineremo con il COUNT, in modo da contare quanti record fanno parte del gruppo.</p> <pre>SELECT COUNT(*) FROM tabella GROUP BY campo1...campoN;</pre> <p>Ma nella voce di sopra noterete la mancanza di qualcosa, quindi sarebbe meglio fare:</p> <pre>SELECT campo1, campo2, campoN, COUNT(*) FROM tabella GROUP BY campo1...campoN;</pre> <p>In questo modo oltre a sapere quanti record ci sono con quelle caratteristiche, di fianco avremo anche il campo considerato.</p>
IN	<p>Seleziona dei record che hanno dei valori specificati in un campo.</p> <p>Esempio:</p> <pre>SELECT * FROM tabella WHERE campo IN ('valore1', 'valore2');</pre>
INSERT INTO	<p>Inserisce dei dati in una tabella. Lo inserisce in coda alla tabella.</p> <p>Esempio:</p> <p>Insert Into user (a, b, 11); <i>‘se si inseriscono tutti i campi.</i></p> <p><i>‘oppure</i></p> <pre>INSERT INTO tabella(campo1, campo2) VALUES (val_campo1, val_campo2);</pre> <p><i>‘inserisce solo nei campi specificati</i></p>
INTERSECT	<p>Se connettiamo due comandi SELECT con un INTERSECT otterremo gli elementi che sono presenti in un tutti i SELECT in questione.</p> <p>Esempio:</p> <pre>SELECT FROM tabella WHERE a = 3 INTERSECT SELECT FROM tabella2 WHERE b = 5;</pre>
LEFT	<p>Restituisce una parte della stringa partendo da sinistra. Vedere anche RIGHT e SUBSTRING.</p> <p>Sintassi:</p> <pre>LEFT(stringa,numero_caratteri);</pre> <p>Esempio:</p> <pre>SELECT LEFT('prova',3); restituirà 'pro'.</pre>
LENGTH	<p>Restituisce la lunghezza di una stringa.</p> <p>Sintassi:</p> <pre>LENGTH(stringa);</pre>

	<p>Esempio:</p> <pre>SELECT LENGHT('prova');</pre> <p>restituirà 5.</p> <p>Se per esempio vogliamo selezionare tutti i record che hanno Campo1 di lunghezza maggiore o uguale a 3 faremo:</p> <pre>SELECT (*) FROM tabella WHERE LENGHT(Campo1) >= 3;</pre>
LIKE	<p>E' una condizione del WHERE. Seleziona dei dati da un database dove il campo è simile ad uno da noi specificato. Il carattere jolly per indicare infiniti caratteri è il simbolo di percento (%), mentre il carattere jolly all'interno della stringa è l'underscore (_). Il primo esempio troverà ad esempio panettone, panettiere, appannato, mentre il secondo esempio troverà pala, para, paca, etc...</p> <p>Primo esempio:</p> <pre>SELECT * FROM tabella WHERE campo LIKE '%pan%';</pre> <p>Secondo esempio:</p> <pre>SELECT * FROM tabella WHERE campo LIKE 'pa_a';</pre>
LOG E LOG10	<p>Restituisce il logaritmo di un numero (LOG10 lo stesso, ma in base 10).</p> <p>Sintassi:</p> <pre>LOG(valore);</pre> <pre>LOG10(valore);</pre>
LOWER	<p>Riduce in minuscolo tutti i caratteri di una stringa.</p> <p>Sintassi:</p> <pre>LOWER(stringa);</pre> <p>Esempio:</p> <pre>SELECT LOWER('PROVA');</pre> <p>restituirà: prova.</p>
LTRIM	<p>Elimina gli spazi a sinistra di una stringa. Vedere anche RTRIM e TRIM.</p> <p>Sintassi:</p> <pre>LTRIM(stringa);</pre> <p>Esempio:</p> <pre>SELECT LTRIM(' stringa di prova ');</pre> <p>darà come risultato: 'stringa di prova '</p>
MIN E MAX	<p>Seleziona il minimo o il massimo da un database. Queste funzioni possono anche essere usate con stringhe e non solo con valori numerici come si potrebbe pensare.</p> <p>Esempio:</p> <pre>SELECT Min(campo) FROM tabella;</pre> <p>'il minimo</p>

	SELECT Max(campo) FROM tabella; 'il massimo
MINUS	<p>Se connettiamo due comandi SELECT con un MINUS otterremo gli elementi che sono presenti nel primo SELECT, ma non nel secondo.</p> <p>Esempio:</p> <pre>SELECT FROM tabella WHERE a = 3 MINUS SELECT FROM tabella2 WHERE b = 5;</pre>
MOD	<p>Restituisce il resto della divisione tra due valori.</p> <p>Sintassi:</p> <pre>MOD(5,2);</pre> <p>Esempio:</p> <pre>SELECT MOD(5,2);</pre> <p>darà come risultato 1.</p>
ORDER BY	<p>Ordina in ordine crescente (ASC) o decrescente (DESC) i dati contenuti in un Select.</p> <p>Esempio:</p> <pre>SELECT * FROM tabella Order By campo ASC; 'ordine crescente</pre> <pre>SELECT * FROM tabella Order By campo DESC; 'ordine decrescente</pre>
PI	<p>Restituisce il valore di pigreco con 6 decimali.</p> <p>Sintassi:</p> <pre>PI();</pre>
POWER	<p>Restituisce un valore dato in input elevato ad un determinata potenza.</p> <p>Sintassi</p> <pre>POWER(valore,potenza);</pre> <p>Esempio:</p> <pre>SELECT POWER(3,2);</pre> <p>Restituisce 9.</p>
RADIANS	<p>Converte un valore da gradi in radianti.</p> <p>Sintassi:</p> <pre>RADIANS(valore);</pre>
RAND	<p>Genera un numero casuale di tipo float in base al valore inserito tra parentesi compreso tra 0 e 1.</p>
REPLACE	<p>Sostituisce una serie di caratteri da una stringa con degli altri caratteri.</p> <p>Sintassi:</p> <pre>REPLACE(stringa,carattere_da_trovare,sostituto);</pre> <p>Esempio:</p>

	<p>SELECT ('prova','ov', 'a');</p> <p>Restituirà 'praa'.</p>
REVERSE	<p>Restituisce la stringa scritta al contrario.</p> <p>Sintasi:</p> <p>REPLACE(stringa);</p> <p>Esempio:</p> <p>SELECT REPLACE('stringa di prova');</p> <p>Restituirà: 'avorp id agnirts'</p>
REVOKE	<p>Revoca una autorizzazione concessa con GRANT.</p> <p>REVOKE <tipo autorizzazione> ON <oggetto del database a cui si riferisce l'autorizzazione> TO <nome utente che si autorizza></p> <p>Esempio:</p> <p>REVOKE SELECT ON tabella TO utente;</p>
RIGHT	<p>Restituisce una parte della stringa partendo da destra. Vedere anche LEFT e SUBSTRING.</p> <p>Sintassi:</p> <p>RIGHT(stringa,numero_caratteri);</p> <p>Esempio:</p> <p>SELECT RIGHT('prova',3); restituirà 'ova'.</p>
ROLLBACK	<p>Annulla tutte le modifiche fissate tramite il COMMIT.</p> <p>Esempio:</p> <p>REVOKE;</p>
ROUND	<p>Restituisce il valore arrotondato con un certo numero di decimali.</p> <p>Sintassi:</p> <p>ROUND(valore, numero_decimali);</p> <p>Esempio:</p> <p>SELECT ROUND(3.444245,2);</p> <p>Restituirà 3.44</p> <p>SELECT ROUND(3.5154654,2);</p> <p>Restituirà 3.52</p>
RTRIM	<p>Elimina gli spazi a destra di una stringa. Vedere anche LTRIM e TRIM.</p> <p>Sintassi:</p> <p>RTRIM(stringa):</p> <p>Esempio:</p>

	SELECT RTRIM(' stringa di prova '); darà come risultato: ' stringa di prova '
SELECT	<p>Il SELECT è uno dei comandi fondamentali dell' SQL. Tramite questo noi possiamo selezionare delle voci dal nostro database creandoci un insieme di record detto recordset. La sintassi è la seguente:</p> <p>SELECT (Campo1, Campo2, ... , Campo3) FROM tabella;</p> <p>Se vogliamo selezionare tutti i campi scriveremo:</p> <p>SELECT (*) FROM tabella;</p> <p>Il carattere asterisco [*] è detto wildcard è viene usato per indicare tutti i campi. Questo metodo risulta più lento perché sarà il computer a dover calcolare quali campi prendere e risulta inutile se ci serve un solo campo, ma nel caso in cui ci servano tutti i campi o quasi ci risulta molto rapido nella scrittura e quindi conveniente.</p> <p>Se vogliamo solo alcuni campi specificati dovremmo ricorrere a WHERE; questo ci permetterà di specificare un valore in base al quale selezioneremo i campi da prendere.</p> <p>Sintassi:</p> <p>SELECT (*) FROM tabella WHERE Campo1 = 'valorediprova';</p> <p>Per altre informazioni a riguardo vedete anche il LIKE su questo tutorial.</p>
SHOW	<p>Visualizza alcune informazioni sull' argomento specificato. I possibili argomenti sono:</p> <ul style="list-style-type: none"> • GRANTS, visualizza i permessi; • TABLE STATUS, visualizza alcune informazioni sullo stato delle tabelle; • STATUS, visualizza delle informazioni sullo stato del server; • VARIABLES, visualizza le i variabile e i valori del Server; • LOGS, visualizza i logs; • PROCESSLIST, visualizza i processi in esecuzione. Per terminare un processo bisogna scrivere KILL seguito dall' identificativo del processo (ID).
SIN	<p>Restituisce il seno di un valore espresso in radianti.</p> <p>Sintassi:</p> <p>SIN(valore);</p>
SQRT	<p>Esegue la radice quadrata di un valore.</p> <p>Sintassi:</p>

	SQRT(valore);
STD	Esegue la deviazione statistica standard su tutti i valori dell'espressione. Sintassi: SELECT STD(Campo1) FROM tabella;
SUBSTRING	Seleziona un determinato numero di caratteri a partire da qualsiasi punto della stringa . Vedere anche RIGHT e LEFT. Sintassi: SUBSTRING(stringa,numero_carattere_d'inizio,numero_caratteri); Esempio: SELECT SUBSTRING('prova',2,3); restituirà 'rov'.
SUM	Esegue la somma di tutti gli elementi del recordset. E' da usare in combinazione col SELECT. Sintassi: SELECT SUM(Campo1) FROM tabella;
TAN	Restituisce la tangente di una espressione. Sintassi TAN(valore);
TRIM	Elimina gli spazi a sinistra e a destra di una stringa. Vedere anche RTRIM e LTRIM. Sintassi: TRIM(stringa): Esempio: SELECT TRIM(' stringa di prova '); darà come risultato: 'stringa di prova'.
UNION	Se connettiamo due comandi SELECT con un UNION otterremo gli elementi che sono presenti nel primo SELECT seguiti da quelli presenti nel secondo SELECT. Esempio: SELECT FROM tabella WHERE a = 3 MINUS SELECT FROM tabella2 WHERE b = 5;
UPDATE	Modifica il valore di un campo già presente nel database. La sintassi è: UPDATE NomeTabella SET campo1 = 'nuovovalore1', campo2 = 'nuovovalore2' ... WHERE nomecampo = 'valorecampodiriferimento';
UPPER	Trasforma in maiuscolo tutti i caratteri di una stringa. Vedere anche LOWER Sintassi: LOWER(stringa);

```
Esempio:  
SELECT LOWER('prova');  
restituirà: PROVA.
```

5. Creare nuovi utenti in MySQL

Per creare utenti in MySQL si possono scegliere due strade. La prima consiste nel creare l'utente col GRANT, cioè mentre stiamo fornendo gli accessi ad un determinato utente. La sintassi del GRANT cambierà quindi in:

```
GRANT tipoprivilegio ON nomedatabase.nomeabella TO utente@host IDENTIFIED BY  
'nuovapassword' WITH GRANT OPTION;
```

Le parole in blu da sostituire credo siano abbastanza chiare, fatta eccezione per host. Ad host dovete sostituire l'indirizzo IP o il nome dell'host dal quale è consentita la connessione. Se al posto di host scriviamo % la connessione sarà accettata da qualsiasi host. Se il server gira in locale, l'host della macchina è localhost.

Altro metodo per creare un nuovo utente è quello di interagire direttamente sul database degli utenti. Per fare ciò dobbiamo andare a modificare la tabella user presente nel database mysql (creato di default durante l'installazione. Non cancellatelo per nessun motivo mi raccomando! Questo contiene informazioni basilari per il funzionamento di MySQL). Quindi apriamo il database in questione scrivendo:

```
mysql> use mysql;
```

Effettuiamo poi il seguente INSERT INTO:

```
INSERT INTO user VALUES('host','nomeutente',PASSWORD('password'), 'X', 'X', 'X', 'X', 'X',  
'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X');
```

oppure:

```
INSERT INTO user SET Host='host',User='nome utente' ,Password = PASSWORD(password),  
Select_priv= 'X', Insert_priv= 'X',Update_priv = 'X',Delete_priv= 'X',Create_priv
```

```
= 'X', Drop_priv= 'X', Reload_priv= 'X', Shutdown_priv= 'X', Process_priv= 'X', File_priv= 'X', Grant_priv= 'X', References_priv= 'X', Index_priv= 'X', Alter_priv= 'X';
```

Il numero degli argomenti può variare da versione a versione quindi è sempre meglio fare un semplice SELECT per visualizzare tutti i campi prima. Alla X va sostituito o Y o N, a seconda se vogliamo concedere oppure no i permessi a quell'utente. Se inseriamo tutte Y creiamo un SuperUser.

Una volta fatto questo INSERT INTO dobbiamo fare un FLUSH (questo comando aggiorna le variabili di sistema, in questo caso quelle riguardanti i privilegi) scrivendo:

```
mysql> FLUSH PRIVILEGES;
```

6. I tipi disponibili in MySQL

Quando creiamo il nostro database naturalmente dobbiamo definire i tipi dei campi. I tipi dei campi che usa MySQL sono i seguenti:

- TINYINT – Valore numerico compreso tra -128 e 127.
- TINYINT UNSIGNED – Valore numerico compreso tra 0 e 255.
- SMALLINT – Valore numerico compreso tra -32768 e 32767.
- SMALLINT UNSIGNED – Valore numerico compreso tra 0 e 65535.
- MEDIUM – Valore numerico compreso tra -8388608 e 8388607.
- MEDIUM UNSIGNED – Valore numerico compreso tra 0 e 16777215.
- INT o INTEGER – Valore numerico compreso tra -2147483648 e 2147483647.
- INT UNSIGNED – Valore numerico compreso tra 0 e 4294967295.
- BIGINT – Valore numerico compreso tra -9223372036854775808 e 9223372036854775807.
- BIGINT UNSIGNED – Valore numerico compreso tra 0 e 18446744073709551615.
- FLOAT (m,d) – Valore numerico con valori decimali. Compreso tra -3.402823466E+38 e -1.175494351E-38.
- DOUBLE(m,d) o REAL(m,d) – Valore numerico con valori decimali. Compreso tra -1.7976931348623157E+308 e -2.2250738585072014E-308.
- DECIMAL (m,d) o NUMERIC(m,d) – Valore numerico con decimali che sarà memorizzato come una serie di caratteri.

- DATE – Campo contenente una data in formato "aaaa-mm-gg".
- DATETIME – Campo contenente la data e ora, informato "aaaa-mm-gg hh:mm:ss".
- TIME – Campo contenente un'ora nel formato: "hh:mm:ss".
- CHAR – Campo contenente una stringa di caratteri che può avere la massima lunghezza di 255.
- TINYBLOB o TINYTEXT – Campo che può avere la massima lunghezza di 255 in formato binario nel primo caso o testo nel secondo.
- BLOB o TEXT - Campo che può avere la massima lunghezza di 65535 in formato binario nel primo caso o testo nel secondo.
- MEDIUMBLOB o MEDIUMTEXT - Campo che può avere la massima lunghezza di 16777215 in formato binario nel primo caso o testo nel secondo.
- LONGBLOB o LONGTEXT - Campo che può avere la massima lunghezza di 4294967295 in formato binario nel primo caso o testo nel secondo.
- ENUM ('value1','value2',...) – Campo che può contenere uno dei valori presente tra le parentesi.

7.Gli script: rendiamo il lavoro sequenziale

Spesso ci capita di dover fare lunghe sequenze di comandi SQL, ed è un po' scomodo scriverli nel prompt che ci offre MySQL di base, per questo possiamo ricorrere a diversi strumenti facilmente trovabili sul web, ma basta anche un semplice blocco note. Nel file di testo che andremo a creare dovremo inserire tutti i comandi che desideriamo creando un vero e proprio file batch. Fatto questo torniamo al nostro prompt di MySQL e digitiamo:

```
mysql> \. <percorso completo del file da eseguire>
```

Questa opzione è molto comoda nel caso in cui ci trovassimo a dover fare diversi comandi ripetitivi (copia e incolla docet) e ci permette anche di costruirci nel linguaggio che preferiamo il nostro piccolo editor di script.

8. MySQL utilizzato in diversi linguaggi di programmazione: esempi pratici.

Adesso che abbiamo creato il nostro database cosa ce ne facciamo se non possiamo interrogarlo tramite applicazioni esterne? Quindi ecco dei frammenti di codice che vi spiegheranno l'iterazione col database creati con MySQL in diversi linguaggi.

ASP

‘Questa connessione è DNS LESS per ulteriori informazioni consultate una guida relativa all’ASP. Per asp avete bisogno di una dll con un nome tipo Myodbc-2.xx.xx-win.zip) l’installazione di questa dll è già stata vista in precedenza.. Vi consiglio “ASP Guida di riferimento” di Weissinger edito da Apogeo.

```
strConn = "driver=MySQL;server=<indirizzo server>;uid=<nome utente>;pwd=<password>
;database=<nome database> Set Conn = Server.CreateObject("ADODB.Connection")
if err.Number = 0 then
Response.write "<P>Connessione riuscita!"
else
Response.write "<P>Errore: " & err.Description
end if
%>
```

ASP.NET

Per far funzionare questo esempio avete bisogno dei driverrace.it ODBC.NET non inclusi nel Framework del .NET ma scaricabili direttamente dal sito Microsoft.

```
<%@ CompilerOptions=/'
R:"C:\Programmi\Microsoft.NET\Odbc.Net\Microsoft.data.odbc.dll"' %>
<%@ Import Namespace = "Microsoft.Data.Odbc" %>
<%@ Import Namespace="System.Data" %>
<script language="C#" runat="server">
void Page_Load(Object sender, EventArgs e)
{
//Apro la connessione al database
OdbcConnection ob
ob = new
```

```

OdbcConnection("Driver={MySQL};Database=miodb;UID=user;PWD=password;");
//Interrogo il db e riempio il mio dataset con i valori appena ricavati
String sql = "SELECT * FROM TABELLA";
OdbcDataAdapter Com = new OdbcDataAdapter(sql, ob);
DataSet ds = new DataSet();
Com.Fill(ds);
Grid.DataSource = ds.Tables[0];
Grid.DataBind();
}</script> <html>
<head>
</head>
<body>
<p align=center><b>Esempio di connessione a MySQL con ASP.NET </b></p>
<ASP:DataGrid id="Grid" runat="server" Width="100%" align="center"border=1>
</ASP:DataGrid>
</body>
</html>

```

PERL

```

00 # prova.pl
01 use DBI;
02
03 $ddb = "DBI:mysql:database=<nomedatabase>";
04 $utente = "<nomeutente>";
05 $password = "<pass>";
06
07 # Apertura connessione
08 my $dbh = DBI->connect($ddb, $utente,
$password)
09 or die "NON MI RIESCO A CONNETTERE";
10
11 # Esempio query
12 my $query = $dbh->prepare("<SELECT * FROM nometabella>");
13 $query->execute();
14
15 # Visualizza a video i risultati
16 while (my $ref = $query->fetchrow_hashref()){
17 print "$ref->{'<CampoA>'} $ref->{'<CampoB>'}";
18 print "\n";
19 }

```

```

20
21 $query->finish();
22
23 # Chiusura connessione
24 $dbh->disconnect;
25
26 exit;

```

Naturalmente modificate le stringhe contenute tra < > e scritte in blu a seconda delle vostre necessità.

Prima di eseguire lo script dobbiamo installare tutti i driver necessari. Questi sono scaricabili dal solito www.mysql.org. I nomi dei file di cui abbiamo bisogno sono i seguenti:

- DBI-X.XX.tar.gz
- Data-ShowTable-X.X.tar.gz
- Msql-Mysql-modules-X.XXXX.tar.gz

Questi file sono per Linux, naturalmente esiste anche una versione per Windows, scaricabile sempre da www.mysql.org. Le X naturalmente stanno a sostituire i numeri della versione del file. Tramite questi file permetteremo al Perl DBI (DataBase Interface) di dialogare col DBD (DataBase Driver) che ha il contatto diretto con MySQL.

Per quanto riguarda l'installazione di questi componenti vi rimando ai file di aiuto (di nome readme o simile che troverete scompattando i file che avete scaricato

PHP

//Apertura database:

```
<?php
```

// Apro la connessione

```
$host = 'localhost';
```

```
$user = 'vostro_user';
```

```
$password = 'vostra_password';
```

```
mysql_connect($host,$user,$password) or die ("CONNESSIONE FALLITA!");
```

```
print "CONNESSIONE OK";
```

// Creo il database

```
mysql_create_db("nomedatabase")or die ("Non riesco a creare il database");
```

// provo a selezionare il database appena creato

```
mysql_select_db("nomedatabase ") or die ("SELEZIONE DATABASE FALLITA");
```

```
print "TUTTE LE OPERAZIONI SONO STATE ESEGUITE CON SUCCESSO";
```

```
?>
```

//Invio comandi SQL:

```
<?php
```

```

$dati = mysql_query("<COMANDO SQL>");
?>
<?php
$dati = mysql_query("select * from tabella");
//salvo i dati del select in un array
$array = mysql_fetch_array($dati);
?>

```

Per eseguire questo codice avete bisogno prima di aver installato il riferimento a MySQL. Per fare questo seguite questi semplici passi:

1. Andiamo nella directory dove si trovano i sorgenti del PHP.
2. ./configure --prefix-use-mysql=[directory in cui avete installato Mysql] [invio].
3. reinstalliamo PHP scrivendo:


```
make [invio]
```

 e


```
make install [invio].
```

Eseguiti questi semplici tre passaggi possiamo finalmente provare il nostro script.

Visual Basic

```
Private Sub myodbc_ado_Click()
```

‘questo esempio funziona tramite ado, quindi non dovrete aver bisogno di componenti aggiuntivi. Se volete altre informazioni sull’interrogazioni dei database in Visual Basic scrivetemi, vi invierò un tutorial apposito.

```
Dim conn As ADODB.Connection
```

```
Dim rs As ADODB.Recordset
```

```
Dim fld As ADODB.Field
```

```
Dim sql As String
```

```
‘connetti al server MySQL usando MySQL ODBC 3.51 Driver
```

```
Set conn = New ADODB.Connection
```

```
conn.ConnectionString = "DRIVER={MySQL ODBC 3.51 Driver};" _
    & "SERVER=localhost;" _
    & " DATABASE=test;" _
```

```
& "UID=venu;PWD=venu; OPTION=35"
```

```
conn.Open
```

```
'esempio di creazione di una tabella
```

```
conn.Execute "DROP TABLE IF EXISTS my_ado"
```

```
conn.Execute "CREATE TABLE my_ado(id int not null primary key, name varchar(20)," _  
            & "txt text, dt date, tm time, ts timestamp)"
```

```
'inserimento diretto
```

```
conn.Execute "INSERT INTO my_ado(id,name,txt) values(1,100,'venu')"
```

```
conn.Execute "INSERT INTO my_ado(id,name,txt) values(2,200,'MySQL')"
```

```
conn.Execute "INSERT INTO my_ado(id,name,txt) values(3,300,'Delete')"
```

```
Set rs = New ADODB.Recordset
```

```
rs.CursorLocation = adUseServer
```

```
'esempio di prelevamento dati
```

```
rs.Open "SELECT * FROM my_ado", conn
```

```
Debug.Print rs.RecordCount
```

```
rs.MoveFirst
```

```
Debug.Print String(50, "-") & "Initial my_ado Result Set " & String(50, "-")
```

```
For Each fld In rs.Fields
```

```
Debug.Print fld.Name,
```

```
Next
```

```
Debug.Print
```

```
Do Until rs.EOF
```

```
For Each fld In rs.Fields
```

```
Debug.Print fld.Value,
```

```
Next
```

```
rs.MoveNext
```

```
Debug.Print
```

```
Loop
```

```
rs.Close
```

```
‘esempio di inserimento tramite rs
```

```
rs.Open "select * from my_ado", conn, adOpenDynamic, adLockOptimistic
```

```
rs.AddNew
```

```
rs!Name = "Monty"
```

```
rs!txt = "Insert row"
```

```
rs.Update
```

```
rs.Close
```

```
‘esempio di aggiornamento tramite rs
```

```
rs.Open "SELECT * FROM my_ado"
```

```
rs!Name = "update"
```

```
rs!txt = "updated-row"
```

```
rs.Update
```

```
rs.Close
```

```
‘esempio di selezione tramite rs
```

```
rs.Open "SELECT * FROM my_ado"
```

```
rs.MoveNext
```

```
rs.MoveNext
```

```
rs.Delete
```

```
rs.Close
```

```
‘prelevare la tabella aggiornata dalla memoria
```

```
rs.Open "SELECT * FROM my_ado", conn
```

```
Debug.Print rs.RecordCount
```

```
rs.MoveFirst
```

```
Debug.Print String(50, "-") & "Updated my_ado Result Set " & String(50, "-")
```

```
For Each fld In rs.Fields
```

```
Debug.Print fld.Name,
```

```
Next
```

```
Debug.Print
```

```

Do Until rs.EOF
For Each fld In rs.Fields
Debug.Print fld.Value,
Next
rs.MoveNext
Debug.Print
Loop
rs.Close
conn.Close
End Sub

```



Per eseguire questo esempio assicuratevi di avere il file mysql.h . Si dovrebbe trovare in c:\mysql\include\.

```

#include <stdio.h>
#include "include/mysql.h"

main()
{
    MYSQL conn;
    MYSQL_RES *risultato;
    MYSQL_ROW row;
    unsigned int num_campi,i;
    my_ulonglong num_righe;

    mysql_init(&conn);
    mysql_options(&conn,MYSQL_OPT_COMPRESS,0);
    if (!mysql_real_connect(&conn,"NomeComputer","UserName","Password",
        "db_di_prova",0,NULL,0))
    {
        fprintf(stderr,"Connessione non riuscita, errore numero: %s\n",
            mysql_error(&conn));
    }

    if (mysql_query(&conn,"SELECT * FROM Tabella"))
        fprintf(stderr,"Query fallita!. Errore numero: %s\n",
            mysql_error(&conn));
    else
    {
        risultato=mysql_store_result(&conn);
        if (risultato) //sono stati trovati dei record
        {

```

```

//Legge il risultato della query
num_righe=mysql_num_rows(risultato);
printf("Il \"result set\" contiene %d righe; ",num_righe);

//Recupera il numero di campi per riga.
num_campi=mysql_num_fields(risultato);
printf("ogni riga e' composta da %d campi.\n",num_campi);
printf("-----\n");

//Itera tra le righe del risultato
while ((row=mysql_fetch_row(risultato)))
{
    unsigned long *lengths;
    lengths=mysql_fetch_lengths(risultato);
    for(i=0; i<num_campi; i++)
    {
        printf("[%.*s] ",(int)lengths[i],
            row[i] ? row[i] : "NULL");
    }
    printf("\n");
}
mysql_free_result(risultato);
}
}

return 0;
}

```

9. Conclusione e ringraziamenti

Detto questo metto momentaneamente la parola fine al mio tutorial, sperando di aggiornarlo presto con altri esempi dedicati alla programmazione, una serie di domande comuni ed una raccolta di “trucchi del mestiere”. Naturalmente ogni informazione aggiuntiva, segnalazione di errore, domanda è ben accetta. Distribuisco questo tutorial in un formato modificabile, proprio nella speranza di riceverne versioni ampliate e modificate nella mia casella di posta elettronica dagli stessi utenti che lo leggono.

Davide Cerbo
ccerb@tin.it
www.e-salerno.it

Salerno, lì 08/03/02.

<p>Risorse sul web: http://web.tiscali.it/informaker/ http://www.redangel.it</p>	<p> http://www.mysql.org http://www.html.it http://www.latoserver.it http://www.phpcenter.it </p>
--	---