



Accenni MySQL



Piccola premessa

Anche questa parte di lezioni, dedicata all'interazione tra PHP e MySQL, non ha pretese di essere una guida definitiva su tutto quello che è possibile fare con questo database, semplicemente spero che vi possa aiutare a comprendere gli articoli/libri che troverete ovunque.

Prima di tutto MySQL è il servizio che ci permette di gestire dei database, non l'unico ma il più diffuso (o almeno credo!) e utilizza un linguaggio standard chiamato SQL (system query language).

In queste poche e brevi lezioni, per iniziare a comprendere come utilizzarlo non ci vuole molto, vedremo come creare ed utilizzare un database e le tabelle usando appunto questo linguaggio e PHP.

Ora che le premesse le abbiamo fatte, non resta che iniziare a dare un'occhiata al lavoro che vi aspetta!



Lezione 1 - Rudimenti MySQL

Come ho scritto nella premessa MySQL esegue le istruzioni SQL, in questa lezione vedremo quali sono le funzioni che ci permetteranno poi con PHP di dire a MySQL di creare, cancellare, modificare e visualizzare database e tabelle.

Prima di tutto come creare un database, cosa semplicissima, ecco qua:

```
Create database nomedatabase;
```

Visto com'è semplice?

nomedatabase: è il nome del database che si vuole creare, ovviamente bisognerà ricordarselo per poi poterlo utilizzare!

Cancellare il database è altrettanto semplice:

```
Drop database nomedatabase;
```

Creare una tabella all'interno del nostro database è leggermente più complesso, in quanto bisognerà definire tutti i campi e rende la nostra query lunga, i tipi di dati dei campi li vedremo in una lezione specifica dato che sono molti:

```
Create table nometabella (definizione dei campi);
```

nometabella: è il nome della tabella che vogliamo creare all'interno del database;

definizione dei campi: è l'elenco dei campi (separati da ,) con l'indicazione del tipo di dati che conterranno.

Al contrario, l'eliminazione di una tabella è velocissima, come per l'eliminazione del database:

```
Drop table nometabella;
```

Io consiglio sempre di studiare a tavolino la tabella prima di crearla, anche se è comunque possibile modificarla utilizzando `Alter table`; è però indiscutibile che le tabelle studiate a tavolino sono le migliori!

Ora vediamo come operare con i dati inseriti nelle tabelle che abbiamo creato:

```
select nomecampi from nometabella where condizione;
```

```
insert into nometabella (nomecampo1, nomecampo2, ecc.) values (valore1, valore2, ecc);
```

```
update nometabella set nomecampo1=valore1, nomecampo2=valore2, ecc. where condizione;
```

```
delete from nometabella where condizione;
```

Calma, calma, ora le spieghiamo per bene; ricordate che queste sono query semplici (dico semplici perchè agiscono su di un'unica tabella).

La prima serve a selezionare una serie di record (o righe, si addice meglio a MySQL) che rispettano la condizione indicata; per selezionare tutte le righe di una tabella è sufficiente non mettere `where condizione`.

nomecampi è l'elenco dei campi separati da , su cui si vuole lavorare, per selezionare tutti i campi basta inserire * anziché scriverli tutti.

La terza query serve per inserire i dati nella nostra tabella, non mi sembra richieda molte spiegazioni, fate solo attenzione a mettere nello stesso ordine i



campi e i valori!

La terza istruzione serve per modificare i dati contenuti nelle righe che soddisfano la condizione, indicando i nuovi valori dei campi, se non si specifica la clausola where condizione, verranno modificate tutte le righe.

Infine l'ultima elimina le righe che corrispondono alla condizione imposta, anche in questo caso se si omette la condizione verranno eliminate tutte le righe.

Vediamo qualche esempio di utilizzo in modo da rendere il più possibile facile la comprensione, tenete presente che salto qualche passaggio, come farlo con PHP lo vedremo nelle prossime lezioni:

Creiamo il database:

```
create database test;
```

Creiamo una tabella con 2 campi di testo:

```
create table mio_test (nome varchar(30), cognome varchar(30));
```

Inseriamo 4 valore:

```
insert into mio_test (nome, cognome) values ('Marco', 'Rossi');
insert into mio_test (nome, cognome) values ('Luca', 'Bianchi');
insert into mio_test (nome, cognome) values ('Giovanni', 'Rossi');
insert into mio_test (nome, cognome) values ('Paolo', 'Verdi');
```

Ora vediamo come utilizzare la select:

```
select * from mio_test;
```

seleziona tutti i campi e tutte le righe

```
select nome from mio_test where cognome='Rossi';
```

seleziona il campo nome solo delle righe dove il cognome è rossi

Adesso usiamo update per cambiare i dati:

```
update mio_test set nome='Mario' where cognome='Bianchi';
```

cambia il nome di Luca Bianchi in Mario Bianchi

Ultimo esempio, cancelliamo tutti i Rossi:

```
delete from mio_test where cognome='Rossi';
```



Lezione 2 - Tipi di colonna

In questo articolo mi limito a riportare le indicazioni dal sito di MySQL, sperando di tradurle e riassumerle correttamente! (quello scritto in corsivo sono note personali)

MySQL supporta diversi tipi di dati per le colonne, che possono essere raggruppati in tre categorie: numerici, data e ora, stringhe.

I tipi di dati per le colonne sono elencati leggermente più sotto, le lettere seguenti sono la legenda utilizzata nelle descrizioni dei tipi:

M: indica la dimensione massima, 255.

D: si applica ai numeri in virgola mobile e indica il numero di decimali dopo il punto (nel sistema metrico italiano la virgola). Il valore massimo possibile è 30, ma non dovrebbe essere maggiore di M-2 (ovvero 253).

Le parentesi quadre indicano parti che sono opzionali.

Nota se specifichi ZEROFILL per una colonna, MySQL assegna automaticamente UNSIGNED alla stessa.

Attenzione: se fai una sottrazione tra due interi ed uno è UNSIGNED, anche il risultato sarà UNSIGNED.

Ecco ora i tipi di colonne:

TINYINT[(M)] [UNSIGNED] [ZEROFILL] Un intero molto piccolo, Il range con segno è da -128 a 127. Senza segno da 0 a 255.

BIT

BOOL sono sinonimi di TINYINT(1), *ovvero possono assumere valore solo 0 o 1*

SMALLINT[(M)] [UNSIGNED] [ZEROFILL] Un piccolo intero. Il range con segno è da -32768 a 32767. Senza segno da 0 a 65535.

MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL] Un intero di media grandezza. Il range con segno è da -8388608 a 8388607. Senza segno da 0 a 16777215.

INT

INTEGER [(M)] [UNSIGNED] [ZEROFILL] Intero. Il range con segno è da -2147483648 a 2147483647. Senza segno da 0 a 4294967295.

BIGINT[(M)] [UNSIGNED] [ZEROFILL] Un grande intero. Il range con segno è da -9223372036854775808 a 9223372036854775807. Senza segno da 0 a 18446744073709551615. Alcune note sulle colonne di tipo BIGINT: tutte le operazioni matematiche sono effettuate utilizzando colonne di tipo BIGINT con segno o DOUBLE, quindi non dovresti utilizzare colonne BIGINT senza segno con numeri superiori a 9223372036854775807 (63 bits) tranne che con le funzioni sui bit. Se lo utilizzi i decimali del risultato possono essere errati a causa di errori d'arrotondamento nella conversione da BIGINT a DOUBLE.

Di solito io uso questi tipi per creare un id automatico alle righe, ovviamente senza segno!

FLOAT(precision) [UNSIGNED] [ZEROFILL] Un numero a virgola mobile. La precisione può essere <=24 per quelli a singola precisione e tra 25 e 53 per quelli a doppia precisione.

FLOAT[(M,D)] [UNSIGNED] [ZEROFILL] Numero a virgola mobile singola precisione. Valori ammessi sono da -3.402823466E+38 a -1.175494351E-38, 0, e da 1.175494351E-38 a 3.402823466E+38. M è la dimensione visibile e D il numero di



decimali. **FLOAT** senza argomenti o **FLOAT(X)** con $X \leq 24$ indicano singola precisione.

DOUBLE(M,D) [**UNSIGNED**] [**ZEROFILL**] Numero virgola mobile doppia precisione. Valori ammessi sono da $-1.7976931348623157E+308$ a $-2.2250738585072014E-308$, 0, e da $2.2250738585072014E-308$ a $1.7976931348623157E+308$. M è la dimensione visibile e D il numero di decimali. **DOUBLE** senza argomenti o **DOUBLE(X)** con $25 \leq X \leq 53$ indicano un numero a virgola mobile a doppia precisione.

DOUBLE PRECISION(M,D) [**UNSIGNED**] [**ZEROFILL**]
REAL(M,D) [**UNSIGNED**] [**ZEROFILL**]
Sinonimi di **DOUBLE**.

DECIMAL(M,D) [**UNSIGNED**] [**ZEROFILL**] Numero a virgola mobile memorizzato come se fosse una stringa, usando un carattere per ogni numero. il punto decimale e, per i numeri negativi, il segno -, non sono contati in M (ma lo spazio è riservato). Se D è 0, i valori non avranno punto decimale. Il range massimo per i valori **DECIMAL** è lo stesso dei **DOUBLE**, ma attualmente può essere costretto usando M e D. Se D è omissso, default è 0. Se M è omissso, default è 10.

DEC(M,D) [**UNSIGNED**] [**ZEROFILL**]
NUMERIC(M,D) [**UNSIGNED**] [**ZEROFILL**]
Sinonimi di **DECIMAL**.

DATE Data. Il range è da '1000-01-01' a '9999-12-31'. MySQL mostra le date in formato 'YYYY-MM-DD' (in italiano *aaaa-mm-gg*), ma permette di assegnare i valori alle colonne di tipo **DATE** usando sia stringhe che numeri.

DATETIME Data e ora. Il range è da '1000-01-01 00:00:00' a '9999-12-31 23:59:59'. MySQL mostra **DATETIME** in formato 'YYYY-MM-DD HH:MM:SS', ma permette di assegnare i valori usando sia stringhe che numeri.

TIMESTAMP(M) Il range è da '1970-01-01 00:00:00' fino a qualche cosa nell'anno 2037 :D. MySQL Mostra questi valori in formato YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD, o YYMMDD, dipendentemente se M è 14 (o mancante), 12, 8, o 6, ma permette di assegnare i valori usando sia stringhe che numeri. Notare che **TIMESTAMP(M)** con M uguale a 8 o 14 sono considerate numeri mentre gli altri come stringhe.

TIME Ora. Il range è da '-838:59:59' a '838:59:59'. MySQL mostra questi valori in formato 'HH:MM:SS', ma permette di assegnare i valori usando sia stringhe che numeri.

YEAR(2|4) Anno a 2 o 4 cifre (default 4). I valori ammessi sono dal 1901 al 2155, 0000 nel formato a 4 cifre, e 1970-2069 se usi il formato a 2 cifre (70-69).

[NATIONAL] CHAR(M) [BINARY] Una stringa a lunghezza fissa riempita di spazi a destra fino alla lunghezza quando memorizzata. Gli spazi aggiunti sono rimossi quando si recupera il valore. I valori **CHAR** sono ordinati e confrontati senza tener conto di maiuscolo/minuscolo se l'attributo **BINARY** è dato. **NATIONAL CHAR** è il modo ANSI SQL per definire che una colonna deve utilizzare il set di caratteri standard.

CHAR Sinonimo di **CHAR(1)**.

[NATIONAL] VARCHAR(M) [BINARY] Una stringa a lunghezza variabile. Gli eventuali spazi sono rimossi quando la stringa viene memorizzata. Sono ordinati e confrontati senza tener conto di maiuscolo/minuscolo.



TINYBLOB

TINYTEXT Una colonna BLOB o TEXT con una lunghezza massima di 255 ($2^8 - 1$) caratteri.

BLOB

TEXT Una colonna BLOB o TEXT con una lunghezza massima di 65535 ($2^{16} - 1$) caratteri.

MEDIUMBLOB

MEDIUMTEXT Una colonna BLOB o TEXT con una lunghezza massima di 16777215 ($2^{24} - 1$) caratteri.

LOBLOB

LONGTEXT Una colonna BLOB o TEXT con una lunghezza massima di 4294967295 ($2^{32} - 1$) caratteri.

ENUM('value1','value2',...) Elenco. Un oggetto stringa che può avere solo un valore scelto nella lista dei valori 'value1', 'value2', ..., NULL o il carattere speciale "" sono errati. Può avere un massimo di 65535 valori distinti.

SET('value1','value2',...) Un set. Un oggetto stringa che può avere zero o più valori, ognuno dei quali deve essere scelto dalla lista dei valori 'value1', 'value2', ... Può avere un massimo di 64 membri (*64 valori contemporaneamente scelti dalla lista*).

E questo è tutto il riassunto che mi è possibile farvi, mi sembra che ci siano abbastanza tipi di campo per sbizzarrirvi come volete.



Lezione 3 - PHP e MySQL

Finalmente ci siamo arrivati, quindi non tergiversiamo ed iniziamo subito la lezioncina.

Per recuperare i dati da un database MySQL con PHP ci sono 5 passi, ovviamente il db e le tabelle devono già essere create:

- 1 - Connettersi a MySQL;
- 2 - Selezionare il database;
- 3 - Eseguire la query (e prendere i risultati);
- 4 - Liberare le risorse;
- 5 - Chiudere la connessione.

Vediamo subito come fare:

- 1 - Come connettersi a MySQL:

```
$db = mysql_connect("host", "utente", "password") //Connessione a mysql  
or die("Impossibile connettersi a MySQL!<br>".mysql_error());  
}
```

host è il nome dell'host su cui rieste MySQL, di solito localhost;
utente e *password* sono il nome utente e la password per collegarsi a MySQL.

La riga con `or die` serve per visualizzare un messaggio impostato da voi nel caso la connessione al database avvenga, come vedete ho anche aggiunto `mysql_error()` che mostrerà il tipo di errore di MySQL.

- 2 - Selezionare il database:

```
mysql_select_db("nomedatabase") or die("Impossibile selezionare il database  
".mysql_error());  
}
```

nomedatabase è il nome del db contenente le tabelle da cui vogliamo recuperare i dati; anche qui c'è il messaggio d'errore (come nel caso della connessione) personalizzato se la selezione del db non riesce.

- 3 - Eseguire la query (e prendere i risultati):

```
$risultato = mysql_query($query,$db) or die ("Impossibile eseguire query!  
". mysql_error());  
}
```

Con questa istruzione eseguiamo la query (contenuta nella variabile `$query`) e come al solito, se non funziona, abbiamo il messaggio d'errore per cercare di capire cosa non va. il risultato della query viene messo nella variabile `$risultato`.

A questo punto, per recuperare i dati, utilizziamo un ciclo `for`:

- 3bis - Eseguire la query (e prendere i risultati):

```
$righe = mysql_num_rows ($risultato);  
// così vediamo quante righe abbiamo ottenuto  
if ($righe == 0){echo "Nessuna riga soddisfa il criterio o la tabella è vuota!  
";}  
else {  
    for ($i = 1;$i<=$righe;$i++){  
        // questo è il ciclo che scorre tutte le righe  
        $valori = mysql_fetch_row ($risultato);  
        //prendiamo la riga e la mettiamo nel vettore $valori  
        echo $valori[0]."<br>"; //mostriamo la prima colonna  
    }  
}
```




```
        echo $valori[1]."<br>"; //mostriamo la seconda colonna
        echo $valori[2]."<br>"; //mostriamo la terza colonna e così via fino
all'ultima
        echo "<hr>"; //una riga orrizzontale per separare le varie righe!
    }
}
```

Bene, i valori sono tutti stampati a questo punto non rimane altro da fare che:

4 - Liberare le risorse:

```
mysql_free_result ($risultato);
}
```

E poi:

5 - Chiudere la connessione:

```
mysql_close($db);
}
```

Tutto fatto! Ovviamente in questo esempio i dati sono presentati in modo grezzo, tutti i campi uno sotto l'altro ed i record (pardon, righe) separati da una riga orizzontale, ma l'output potete impostarlo come preferite.

Una nota importante, se la vostra query non deve recuperare dei dati, come nel caso di una query che crea/inserisce/aggiorna una tabella, non avrete da eseguire i punti 3bis e 4, dato che non avete recuperato nulla!

Bene ora siete pronti per utilizzare il vostro database, se vi servono consigli tornate pure a trovarmi.