IL DATABASE MYSQL

Introduzione

Benvenute e benvenuti! In questa lezione faremo una breve introduzione pratica a SQL utilizzando il server MySQL e l'editor MySQL Workbench.

Nel dettaglio, vedremo:

- Che cos'è SQL
- Che cos'è MySQL Workbench
- Come creare una tabella

Bene, non ci resta che cominciare!

Che cos'è SQL

SQL, che sta per Structured Query Language (linguaggio di interrogazione strutturato), è essenzialmente un linguaggio progettato per creare, leggere, aggiornare ed eliminare dati dai database.

Praticamente qualsiasi sistema di gestione di database relazionali utilizzerà SQL come base per il modo in cui accede ai propri dati per la creazione, la lettura, l'aggiornamento e l'eliminazione.

Studiare il funzionamento di SQL ti permette di interagire con qualsiasi sistema di gestione di database relazionali.

Ogni sistema di gestione di database relazionali ha delle peculiarità, ma tutto ciò che è definito in SQL è universale tra tutti i diversi sistemi di gestione di database.

Un database è una raccolta di dati separati in diverse tabelle che rappresentano singoli modelli di dati. Quindi potresti avere una tabella utente, una tabella prodotti, una tabella ordini e tutte queste tabelle si collegheranno tra loro per creare connessioni tra i diversi dati.

Le righe sono i diversi record dei tuoi singoli modelli. Quindi, per esempio, un utente equivarrà a un record o riga all'interno della tua tabella e due utenti due record, o righe, separati in quella tabella.

Tutte le proprietà di quell'utente come il loro ID, nome, e-mail o password, saranno colonne all'interno del tuo database. È possibile collegare dati da tabelle diverse a dati di altre tabelle, creando appunto delle relazioni, così è possibile creare un complesso sistema di dati utilizzando i database.

È abbastanza semplice capire perché sia così importante: SQL si occupa di dati e i dati sono ovunque.

Quasi tutte le applicazioni che usiamo hanno una qualche forma di dati che deve essere salvata da qualche parte e i database, che utilizzano SQL, sono uno dei modi migliori e più semplici per archiviare i dati per qualsiasi applicazione, sia su piccola scala, sia in particolare per applicazioni su larga scala.

MySQL Workbench

Ora che abbiamo capito perché SQL è così importante e cos'è SQL, passiamo a MySQL Workbench per dimostrare la sintassi di SQL e poi approfondiremo i diversi comandi che possiamo usare con SQL per creare, leggere, aggiornare e cancellare i nostri dati.

Per seguire meglio la lezione dovete aver installato MySQL Server e MySQL Workbench dall'indirizzo: https://dev.mysql.com/downloads/workbench/

Supponendo che stiate utilizzando Windows e non abbiate già installato il server MySQL scaricate il MySQL Installer per installare il tutto.

Avviate MySQL Server e MySQL Workbench, che dovrebbe offrirvi una connessione predefinita: Local instance MySQL80

Apritela ed inserite la password che avete scelto all'installazione per giungere alla schermata principale di MySQL Workbench.

Per sapere cosa fa un'icona portate il cursore sopra di essa e aspettate che compaia il tooltip.

Bene, ora ho MySQL Workbench aperto e connesso al mio server MySQL locale.

Se non ho una scheda aperta per poter eseguire comandi SQL devo fare clic sull'icona nell'angolo in alto a sinistra che mi consente di creare una nuova scheda SQL per l'esecuzione di query.

Quello che scriverò nella scheda potrà essere salvato in un file .sql, che è un semplice file di testo.

La sintassi SQL

Approfondiamo la sintassi SQL, che fortunatamente è molto semplice da capire.

Esistono diverse parole chiave in SQL, come la parola chiave SELECT, WHERE, FROM e tutte verranno evidenziate in blu. Le parole chiave non fanno distinzione tra maiuscole e minuscole.

SQL utilizza una combinazione di parole chiave, nomi di tabelle e nomi di colonne per mettere insieme una query, o interrogazione. Ad esempio, potremmo usare la parola chiave SELECT, poi le colonne che ci interessano, poi FROM, quindi inserire il nome della tabella. Alla fine del comando SQL va inserito un punto e virgola.

Questo non è richiesto in ogni sistema di gestione del database, ma per scrivere due diverse query SQL in un file hai bisogno di un punto e virgola per separarle.

Inoltre, anche se le parole chiave possono essere scritte tutte maiuscole, tutte minuscole o qualsiasi altra combinazione di lettere maiuscole o minuscole, è buona pratica e standard scrivere tutte le parole chiave in maiuscolo per distinguerle da nomi di colonne e nomi di tabelle, che saranno in formato minuscolo.

Se hai bisogno di scrivere una stringa, che in informatica è un valore composto da una sequenza di caratteri, all'interno di SQL, racchiudila tra virgolette singole per distinguerla da una qualche parola chiave, tabella o colonna.

La creazione di un database

Iniziamo con la creazione effettiva di un database da utilizzare. In questo esempio, creerò un database per una casa discografica, che conterrà band, album e canzoni al suo interno.

Per iniziare, dobbiamo creare questo database. Al momento, infatti, non abbiamo alcun database nel nostro SQL Server se l'abbiamo appena creato.

Scriveremo il comando CREATE DATABASE.

Basta scrivere le parole CREATE, quindi DATABASE e il nome del database che si desidera creare.

Per adesso, creeremo solo un database chiamato test.

Quindi mettiamo test terminato con un punto e virgola.

All'interno di MySQL Workbench ci sono due modi diversi per eseguire un comando.

C'è l'icona a forma di fulmine più a sinistra che eseguirà qualsiasi cosa tu evidenzi, o tutto il file in assenza di una selezione.

Quindi, se vuoi eseguire solo alcuni comandi, puoi evidenziarli e fare clic su questo fulmine per eseguirli.

Oppure c'è la seconda opzione, che è il fulmine con il cursore, e questo eseguirà l'istruzione all'interno della quale si trova il cursore.

Quindi, se clicchiamo su una di queste icone dopo aver scritto l'istruzione "CREATE DATABASE test;", creeremo un database chiamato test.

Noterai che sembra non accada nulla, che il database non sia stato creato.

Questo perché MySQL Workbench non aggiorna subito l'interfaccia utente quando crei nuove cose. Se vai alla sezione Schemas trovi dove sono elencati i tuoi database e se fai clic sull'icona di aggiornamento apparirà il nostro database di test.

Se lo apriamo, troverai un database di test vuoto senza tabelle o altro al suo interno. Questo è quello che ci aspettiamo, dato che abbiamo appena usato CREATE DATABASE per creare quel database.

Ora rimuoviamo questo database.

Per farlo, useremo il comando DROP DATABASE, poi inseriamo di nuovo il nome del database che vogliamo eliminare, che nel nostro caso è test, e terminiamo con un punto e virgola.

Se eseguiamo questa istruzione, ci rendiamo conto che a sinistra il nostro database di test è stato rimosso. Abbiamo eliminato quel database, tutte le tabelle al suo interno e tutti i dati al suo interno. Quindi ora creiamo il database effettivo che vogliamo, che come ho detto prima, sarà per una casa discografica.

Quindi scrivi "CREATE DATABASE" e inserisci il nome del nostro database, che sarà casa_discografica. Se lo eseguiamo e aggiorniamo la sezione Schemas, vedi che abbiamo un database casa_discografica. Possiamo iniziare ad aggiungere tabelle a questo database e iniziare ad aggiungere dati in quelle tabelle.

Ora dobbiamo dire a SQL che useremo quel database e per farlo usiamo il comando USE.

Quindi digitiamo USE e il nome del database su cui vogliamo eseguire le nostre query.

Una volta eseguito "USE casa_discografica;" stiamo usando il database della casa discografica.

Vedrete che è in grassetto in MySQL Workbench, il che ci dice che ora stiamo usando quel database.

Quindi, da adesso in poi quando eseguiamo comandi come la creazione di tabelle o l'aggiunta di dati, li metterà nel database della casa discografica.

Ora possiamo lavorare sulla creazione della nostra prima tabella.

La creazione di una tabella

Useremo CREATE TABLE, e poi il nome della tabella che vogliamo creare.

Per iniziare, faremo una tabella di prova. Come abbiamo già visto prima, le tabelle hanno delle colonne al loro interno che rappresentano le diverse proprietà dell'oggetto che rappresentano.

Quando creiamo la nostra tabella dobbiamo dire con quali colonne la vogliamo creare.

Quindi inseriamo delle parentesi tonde all'interno delle quali inseriremo le diverse colonne che vogliamo per la nostra tabella.

Per esempio, in questa tabella aggiungeremo solo una colonna, chiamandola colonna_di_prova.

Dobbiamo dare a quella colonna un tipo perché il nostro database ha bisogno di sapere che tipo di dati sta memorizzando.

Nel nostro caso decidiamo che sia un numero intero, quindi useremo INT.

Poiché questa è una parola chiave, si usa tenerla tutta maiuscola per distinguerla dai nomi delle mie colonne e dai nomi delle tabelle.

Per rendere le query più leggibili possiamo scriverle su più righe e mettere il punto e virgola alla fine di quel comando.

Quindi ora posizioniamo il cursore all'interno del comando "CREATE TABLE prova (colonna_di_prova INT); e fai clic sull'icona per eseguirlo.

Se aggiorniamo il nostro schema a sinistra, ci accorgiamo che ora abbiamo un piccolo menu a tendina vicino alle nostre tabelle.

Abbiamo una tabella di prova e all'interno di quella tabella, nelle nostre colonne, abbiamo la colonna_di_prova, che è un numero intero.

Diciamo che vogliamo aggiungere un'altra colonna alla nostra tabella, ci siamo dimenticati di aggiungerla all'inizio e non vogliamo tornare indietro e ricreare questa tabella perché abbiamo già dei dati lì dentro.

Per questo abbiamo un comando chiamato ALTER TABLE che ci permetterà di cambiare le proprietà della nostra tabella dopo averla creata.

Quindi digitiamo semplicemente ALTER TABLE, e poi il nome della tabella che vogliamo modificare, che nel nostro esempio sarà la tabella di prova.

Poi gli diciamo cosa vogliamo fare. In questo caso gli diremo che vogliamo aggiungere una colonna con ADD.

Quindi metteremo il nome della colonna, che chiameremo semplicemente altra_colonna.

Poi scegliamo che tipo vogliamo che sia quella colonna, ad esempio una stringa.

In SQL ci sono molti modi diversi per determinare una stringa, ma il modo più semplice è usare un VARCHAR, che essenzialmente dice che si tratta di una sequenza di caratteri a lunghezza variabile.

Poi dobbiamo dire al VARCHAR la lunghezza massima che può essere, inserendo il numero tra parentesi tonde.

Nel nostro caso diremo che 255 è la lunghezza massima della nostra stringa.

Questo creerà una colonna stringa con una lunghezza massima di 255 caratteri, che verrà chiamata altra_colonna.

Quindi ora se eseguiamo "ALTER TABLE prova ADD altra_colonna VARCHAR(255);" e aggiorniamo i nostri Schemas vedrai che abbiamo un'altra colonna aggiunta alle colonne del nostro database di test.

Ora sappiamo come creare tabelle di database e come aggiungere colonne a quelle tabelle dopo averle create.

Dato che abbiamo questa tabella di test, e in realtà non la vogliamo, diamo un'occhiata all'eliminazione di quella tabella, che funziona analogamente all'eliminazione di un database.

Digitiamo DROP TABLE e il nome della tabella, seguita dal punto e virgola.

Se lo eseguiamo, vedrai che la nostra tabella di test si rimuove completamente dalla sezione delle tabelle nello schema della casa discografica.

Ora che abbiamo rimosso quella tabella lavoriamo sull'aggiunta di una tabella per una band che useremo all'interno del database della nostra casa discografica, poiché vogliamo rappresentare band diverse per la nostra casa discografica.

Quindi scriviamo CREATE TABLE.

E chiameremo la nostra tabella Bands, perché conterrà solo tutte le nostre band.

Ancora una volta, usiamo le nostre parentesi per dire quali saranno le nostre colonne.

L'unica cosa che ci interessa è il nome della band, quindi useremo "nome" come nome della nostra colonna.

Ancora una volta, vogliamo che questo sia un VARCHAR.

E diremo ancora 255 per la lunghezza.

Per assicurarci che la nostra band abbia sempre un nome, aggiungeremo "NOT NULL" dopo "VARCHAR(255)".

Questo dice che la nostra colonna non può contenere valori nulli al suo interno, il che significa che deve sempre avere un nome definito e genererà un errore se provi a inserire una band che non ha un nome.

Ma cosa succede se una band ha lo stesso nome di un'altra band? Come distinguiamo queste due band l'una dall'altra? Ed è qui che torna utile usare una colonna id.

In quasi tutte le tabelle che crei all'interno di un database vorrai aggiungere una colonna id per identificare in modo univoco quella riga in quella tabella.

Quindi aggiungiamo una colonna, che chiameremo id, che vogliamo sia un numero intero, perché è un modo semplice per distinguere cose diverse, e non vogliamo mai che sia NULL, cioè non definito, proprio come il nome.

Vogliamo che venga generato automaticamente ogni volta che aggiungiamo una nuova band alla nostra tabella, quindi gli daremo la proprietà di incremento automatico: AUTO_INCREMENT.

E questo dice che vogliamo incrementare automaticamente questo id ogni volta che aggiungiamo una nuova band.

Quindi la prima band sarà 1, la seconda band sarà 2 e così via.

E incrementerà automaticamente per noi senza che noi dobbiamo fare nulla.

Un'altra cosa che dobbiamo fare è aggiungere una virgola tra il nostro ID e la colonna del nome in modo che SQL sappia che si tratta di due colonne separate.

Infine, poiché questo id sarà l'identificatore della nostra tabella, è quella che viene chiamata chiave primaria.

Una chiave primaria è la colonna identificativa primaria per quella tabella.

Sarà ciò che identificherà un singolo record all'interno di una tabella.

Quindi useremo la parola chiave PRIMARY KEY per definire una chiave primaria e poi tra parentesi mettiamo quale colonna vogliamo che sia.

Nel nostro caso è la colonna id.

Ora, se eseguiamo questo comando e aggiorniamo il nostro schema, vedrai che abbiamo la tabella delle band.

All'interno di quella tabella delle band, abbiamo le colonne impostate.

Vedrai anche che abbiamo un indice per la nostra chiave primaria, che dice a SQL che questo campo è ciò che distingue ogni band dagli altri record di band nella nostra tabella e ci permetterà di fare query più veloci.

Se gli diamo un id, sarà molto più veloce che se interroghiamo la colonna del nome all'interno della nostra tabella delle band.

Ora creiamo la tabella degli album che conterrà i diversi album per le nostre diverse band all'interno del nostro database.

Usiamo di nuovo CREATE TABLE, chiameremo questa tabella "albums" poiché conterrà i nostri album.

Impostiamo le prime due colonne come quelle della tabella bands.

In più vogliamo sapere quando sono pubblicati, quindi aggiungeremo un anno_di_uscita, che è un numero intero, ai nostri album e non ci interessa se questo sia NULL perché forse non sappiamo quando l'album uscirà.

Poi dobbiamo essere in grado di collegare un album a una band.

Dobbiamo fare riferimento alla tabella della band dall'interno della nostra tabella dell'album.

Ed anche qui torna utile l'id che abbiamo creato, perché ora possiamo salvare quell'id nella tabella degli album.

Quindi aggiungiamo una colonna band_id, che conterrà l'id della band a cui è destinato questo album.

Questo sarà un numero intero, perché è uguale all'id nella tabella band.

E vogliamo assicurarci che non sia NULL, dal momento che ogni album deve essere composto da una band.

A questo punto dobbiamo definire la nostra chiave primaria proprio come abbiamo fatto prima.

Quindi inseriremo "PRIMARY KEY (id)" poiché l'id è la nostra chiave primaria anche per la tabella dell'album.

L'ultima cosa che dobbiamo fare è impostare che questo band_id faccia riferimento alla tabella delle band, definendo una chiave esterna, che è qualsiasi forma di chiave che fa riferimento a una tabella diversa da sé stessa.

Per fare questo, useremo la proprietà FOREIGN KEY e tra parentesi vogliamo mettere qual è la nostra chiave esterna, che è l'id della band.

È molto simile alla chiave primaria ma dobbiamo dire alla nostra chiave esterna a quale tabella fa riferimento.

Per farlo scriveremo "REFERENCES bands" per fare riferimento alla tabella delle band, poi dobbiamo esplicitare a quale colonna fare riferimento mettendola tra parentesi.

E ora abbiamo la nostra chiave esterna impostata tra i nostri album e le nostre band.

Quindi SQL non ci permetterà più di creare un album se gli diamo un band_id che non esiste già nella tabella delle band.

Inoltre, se proviamo a eliminare una band che ha album collegati a quella band, si genererà un errore che ci verrà detto che ci sono degli album che esistono per questa band.

Quindi non è possibile eliminare la band a meno che vengano rimossi anche gli album associati a quella band.

Ora, se eseguiamo questo codice, vedremo che se andiamo alle nostre tabelle e aggiorniamo, comparirà la tabella degli album.

E all'interno di questa tabella dell'album troveremo una chiave esterna che collega i nostri album alla tabella della nostra band.

Abbiamo anche tutte le colonne che abbiamo creato e l'indice per la nostra chiave primaria e band_id, che ci consente di effettuare ricerche rapide per queste diverse colonne all'interno del nostro database.

Ora che abbiamo finalmente creato tutte le tabelle di cui avevamo bisogno possiamo effettivamente iniziare a lavorare sull'aggiunta di dati alle nostre tabelle e interrogare le nostre tabelle, perché è proprio a questo che serve SQL, per aggiungere dati e leggerli.

Quindi useremo il comando INSERT INTO, poi definiremo il nome della tabella in cui inserire i dati, nel nostro caso Bands, poi tutte le diverse colonne da inserire all'interno delle parentesi.

Nel nostro caso abbiamo solo una colonna, che è la colonna del nome all'interno della nostra tabella delle band.

Come accennato in precedenza, la colonna id all'interno della nostra tabella delle band si genera automaticamente.

A questo punto dobbiamo inserire i diversi valori usando la parola chiave VALUES e, tra parentesi, metteremo il nome della band poiché corrisponde alla colonna del nome che abbiamo definito.

Quindi aggiungiamo gli Hot Sugar Band al nostro database, scrivendo il nome tra apici singoli, che vanno sempre utilizzati per specificare dei valori testuali.

Aggiungiamo il punto e virgola alla fine.

Se eseguiamo questo comando sembrerà che non succeda nulla ma in realtà abbiamo aggiunto quella band al nostro database. Aggiungiamo altre band nel nostro database.

Scriviamo lo stesso comando INSERT INTO e poi, se vogliamo inserire più di una voce all'interno della nostra tabella contemporaneamente, dopo VALUES mettiamo i valori di ogni record dentro le sue parentesi e li separiamo con la virgola.

Quindi diciamo che vogliamo inserire la band Deuce, poi per inserire un'altra band mettiamo una virgola e all'interno di altre parentesi inseriamo i valori delle colonne per la voce successiva.

Quindi inseriremo Avenged Sevenfold, infine inseriremo Postmodern Jukebox e metteremo il punto e virgola alla fine.

E ora se eseguiamo questa istruzione, aggiungeremo tre diverse band alla nostra tabella, e daremo a tutte un id univoco che verrà incrementato automaticamente.

Interrogare la tabella

Per interrogare i dati dalla nostra tabella, useremo il comando SELECT.

Scriveremo SELECT, poi le diverse colonne che vogliamo selezionare.

Se vogliamo selezionare tutte le colonne, mettiamo un asterisco.

Poi dobbiamo dirgli da quale tabella vogliamo interrogare, usando FROM.

Vogliamo interrogare la tabella delle band, quindi scriveremo "FROM bands".

Terminiamo con un punto e virgola ed eseguiamo il comando. In questo modo otteniamo i nostri diversi risultati: la nostra colonna id, la nostra colonna del nome, e poi i quattro nomi che abbiamo inserito, così come quattro id univoci che sono stati generati automaticamente dal database dal nostro incremento automatico.

Ma cosa succederebbe se, ad esempio, volessimo recuperare solo due band invece di tutte le diverse band? Per farlo, faremo semplicemente questa SELECT esattamente come prima, ma dobbiamo dargli un limite.

Quindi aggiungiamo "LIMIT 2" e questo ci restituirà solo le prime due band dalla nostra query.

Se lo eseguiamo, ora vedremo che otteniamo solo Hot Sugar Band e Deuce, che sono le prime due band restituite da questa query.

Conclusioni

Bene, siamo giunti alla fine di questa videolezione.

Ti ricordo che abbiamo approfondito la costruzione di un database. Nel dettaglio abbiamo visto:

- Che cos'è SQL
- Che cos'è MySQL Workbench
- Come creare e leggere un database

Grazie dell'attenzione!