Modificare ed eliminare tabelle in MySQL

Alessandro Bugatti (alessandro.bugatti@istruzione.it)

15 gennaio 2014

1 Modificare tabelle

Una volta create le tabelle, cioè dopo aver tradotto il nostro E-R in un modello relazionale ed averlo implementato tramite l'utilizzo dell'istruzione *CREATE TABLE*, il nostro compito come amministratori di database potrebbe sembrare finito. Questo, almeno dal punto di vista delle creazione dello schema del database, è vero nella maggior parte dei casi: possono però a volte sorgere delle esigenze (il cambiamento della realtà d'interesse piuttosto che la scoperta di errori nella progettazione concettuale) che ci impongono delle modifiche alle nostre tabelle e che dobbiamo essere in grado di fare senza per questo aver bisogno di creare le nostre tabelle una seconda volta. Le tabelle che intendiamo modificare potrebbero anche contenere dei dati (è anzi la situazione più comune) e quello che succederà dipenderà da quello che andremo a modificare (potrebbe crearsi ad esempio una nuova colonna vuota dove alcuni dati potrebbero non essere più validi).

L'istruzione SQL che permette di modificare una tabella ha la seguente sintassi:

```
ALTER TABLE nome_tabella azione,...;
```

Il significato di *nome_tabella* è chiaro, mentre *azione* è una fra le possibili azioni che possono essere compiute per alterare lo schema della tabella in questione. Noi ci limiteremo a vedere solo alcune delle possibilità più significative tramite degli esempi di utilizzo, rimandando al manuale chi fosse interessato a possibilità più complesse.

1.1 Aggiungere una nuova colonna

Supponiamo di aver creato una tabella con la seguente istruzione:

```
CREATE TABLE articolo
(
        ID INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
        nome CHAR(100) NOT NULL,
        descrizione TEXT NOT NULL ,
```

```
        data_di_produzione DATE NOT NULL
);
```

come già visto nella dispensa precedente. Dopo qualche tempo, quando la tabella contiene già una certa quantità di dati, ci accorgiamo che ogni articolo avrebbe bisogno anche di un campo prezzo di tipo float che all'inizio non avevamo pensato di inserire.

Per modificare la tabella in modo da inserire questo nuovo campo possiamo lanciare la seguente istruzione:

```
ALTER TABLE articolo

ADD prezzo FLOAT NOT NULL;
```

In questo modo abbiamo aggiunto una colonna di numeri di tipo float per contenere il prezzo di ogni singolo articolo (ovviamente la colonna non conterrà dati significativi, ma in questo caso il valore di default per un float, cioè zero).

1.2 Eliminare una colonna

Supponiamo adesso di voler cambiare la chiave primaria, perchè il campo ID che era un intero di tipo AUTO_INCREMENT non ci va più bene. Al suo posto vogliamo inserire un campo alfanumerico che rappresenti il codice dell'articolo secondo un sistema interno all'azienda. La prima cosa da fare è quella di eliminare la chiave primaria presente nella tabella utilizzando questa istruzione:

```
ALTER TABLE articolo

DROP ID;
```

Successivamente potremmo pensare di inserire la nostra nuova chiave primaria con il comando:

```
ALTER TABLE articolo

ADD codice CHAR(12) PRIMARY KEY;
```

Purtroppo questo comando, in presenza di articoli già inseriti, non funzionerà. Perchè? La causa è proprio la presenza di articoli nella tabella: difatti nel momento in cui venisse creata una nuova colonna questa sarebbe vuota, cosa non ammissibile per una chiave primaria, in quanto non sarebbe in grado di distinguere fra le diverse istanze (se la tabella fosse stata vuota questo problema non si sarebbe verificato). Possiamo allora limitarci ad aggiungere il nuovo campo senza indicarlo come chiave primaria:

```
ALTER TABLE articolo

ADD codice CHAR(12);
```

1.3 Aggiungere una chiave primaria

A questo punto potremo inserire i codici alfanumerici in corrispondenza di ogni articolo, avendo cura di non inserire valori duplicati, cosa che il database non

ci impedirebbe in quanto *codice* non è ancora indicato come chiave primaria. Solo dopo questa operazione, ottenuta attraverso l'utilizzo dei comandi DML[1], potremo settare il nostro campo *codice* come chiave primaria:

```
ALTER TABLE articolo

ADD PRIMARY KEY (codice);
```

Se all'interno del campo codice non si troveranno valori duplicati esso diventerà la nuova chiave primaria, altrimenti l'operazione non sarà permessa e riceveremo un messaggio d'errore che ci avviserà della presenza di valori duplicati.

1.4 Modificare il tipo di campo

Un'altra esigenza che può sorgere quando si utilizza un database è quella relativa alla modifica del tipo di un campo perchè ci si è accorti, ad esempio, che il tipo INT non è più sufficiente per registrare i dati che dobbiamo inserire, ma abbiamo bisogno di un tipo più capiente. La modifica di un campo (che può anche portare alla perdita di precisione o addirittura alla perdita di dati se fatta tra tipi incompatibili tra loro) viene espressa con la seguente sintassi:

```
ALTER TABLE nome_tabella MODIFY nome_colonna tipo_colonna;
```

Tornando al solito esempio della tabella articolo se volessimo trasformare la colonna prezzo da FLOAT a DOUBLE dicendo anche che deve per forza contenere un dato dovremmo eseguire il seguente comando:

```
ALTER TABLE articolo

MODIFY prezzo DOUBLE NOT NULL;
```

Può a volte succedere di voler cambiare il nome di una colonna, in questo caso la sintassi sarà:

```
ALTER TABLE nome_tabella CHANGE vecchio_nome nuovo_nome
    tipo_colonna;
```

1.5 Modificare una sequenza

L'ultimo esempio che vedremo riguarda la possibilità di modificare i dati contenuti all'interno di una colonna di tipo AUTO_INCREMENT, per eliminare i "buchi" che si sono creati a causa di inserimenti e successive cancellazioni di dati. Come sappiamo una colonna di tipo AUTO_INCREMENT incrementa un numero intero ogni volta che viene inserito un nuovo dato e se successivamente quel dato viene cancellato il suo numero progressivo rimane inutilizzato. Avendo la nostra tabella articolo all'interno della quale compare ancora il campo ID di tipo AUTO_INCREMENT e volendo ricostruire la sequenza in modo da utilizzare tutti i numeri ed eliminare i "buchi" si può procedere con le seguenti istruzioni:

[1] Data Manipulation Language, parte del linguaggio SQL che permette di inserire, modificare e cancellare i record di una tabella.

```
ALTER TABLE studenti

DROP ID;

ALTER TABLE studenti

ADD ID INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY FIRST,

AUTO_INCREMENT = 1;
```

Il comando DROP, come già visto, elimina la colonna *ID* e quindi tutti i valori in essa contenuti. Successivamente viene aggiunta di nuovo la colonna *ID* di tipo AUTO_INCREMENT dicendo anche che il numero da cui partire deve essere il numero 1 (con l'istruzione AUTO_INCREMENT = 1). Da notare la parola chiave FIRST che indica l'inserimento del nuovo campo come prima colonna, poichè nel caso non si indicasse nulla il campo verrebbe inserito come ultima colonna. Per posizionarlo in altri posti si può usare il comando AFTER *nome_colonna* che inserisce il campo dopo una colonna già presente di nome *nome_colonna*.

2 Eliminare una tabella

Se a questo punto volessimo eliminare la tabella articolo non dovremmo far altro che eseguire la seguente istruzione:

```
DROP TABLE articolo;
```

Attenzione che questa operazione è distruttiva e una volta lanciata elimina per sempre la tabella articolo e i dati in essa contenuti dal database. Se avessimo voluto eliminare contemporaneamente più tabelle esiste la possibilità di scrivere i nomi delle tabelle uno di seguito all'altro separati da una virgola, nel seguente modo:

```
DROP TABLE nome_tabella1, nome_tabella2, ;
```