Home Page
Novità
Aiuto

Il linguaggio SQL
L'istruzione INSERT INTO

http://www.vbsimple.net/database/db_08_04.htm

Difficoltà: 2 / 5

La prima istruzione utilizzata dalle query di inserimento è INSERT INTO, che presuppone che la tabella di destinazione esista e presenta due sintassi differenti:

```
INSERT INTO <tabella> [ (campi) ] <istruzione SELECT>
INSERT INTO <tabella> [ (campi) ] VALUES( <valori> )
```

Ciascuno dei due casi può essere indicato con due possibilità. Vediamole tutte e quattro:

```
INSERT INTO <tabella> <istruzione SELECT>
```

Inserisce all'interno della tabella indicata i valori recuperati dall'istruzione SELECT (che può presentare qualsiasi operatore, come una normale query di interrogazione). L'ordine con cui i valori sono immessi nella tabella di destinazione è quello dal primo all'ultimo campo in ordine. Prendiamo due tabelle come esempio: la prima è la solita **dipendenti** con i campi id, nome, cognome e ruolo. La seconda tabella la chiameremo **dirigenti** e conterrà i campi nome, cognome e ID (notare l'assenza del campo *RUOLO*).

TABELLA dipendenti					**TABELLA dirigenti**		
ID	**NOME**	**COGNOME**	**RUOLO**		**NOME**	**COGNOME**	**ID**
1	Filippo	Cambrini	Direttore				
2	Lucia	Totti	Impiegata				
3	Mauro	Festina	Direttore				
4	Paolo	Cotroneo	Direttore				

Con una semplice istruzione SELECT recupereremo tutti i direttori dalla tabella dipendenti come segue:

```
SELECT id, nome, cognome FROM dipendenti WHERE ruolo='Direttore'
```

Naturalmente non sarà possibile utilizzare l'asterisco al posto dei nomi dei campi perché il numero di colonne delle due tabelle è differente; l'istruzione INSERT INTO richiede che il numero di campi da inserire sia uguale a quello specificato nella tabella di destinazione.

Recuperate le tre righe sarà necessario inserirle all'interno della tabella dirigenti. Prima di inserire quest'istruzione SELECT in coda all'istruzione INSERT INTO dobbiamo sistemare l'incongruenza! Infatti i campi verrebbero inseriti in ordine dal primo all'ultimo, cioè da sinistra verso destra; per cui il campo ID finirebbe su Nome, il campo Nome su Cognome e Cognome su ID. Vediamo come risolvere questa problematica:


```
INSERT INTO dirigenti
SELECT nome, cognome, id FROM dipendenti WHERE ruolo='Direttore'
```

La soluzione più naturale è quella di riordinare i campi nell'istruzione SELECT. È importante ricordare l'ordine dei campi i cui dati verrebbero inseriti nella tabella: i nomi dei campi infatti sono ignorati e la sequenza è quella da sinistra verso destra. Se invece avessimo utilizzato quest'altra istruzione SELECT avremmo ottenuto un risultato inaspettato e non voluto:

```
INSERT INTO dirigenti
SELECT cognome, nome, id FROM dipendenti WHERE ruolo='Direttore'
```

I dati del campo cognome sarebbero finiti nel campo nome e viceversa per l'altro. Il campo ID invece sarebbe andato al suo posto, in coda a tutti gli altri.
Un'altra soluzione invece consiste nel non disordinare i campi nell'istruzione SELECT ma piuttosto specificare l'ordine con cui i dati dovrebbero essere inseriti nella tabella di destinazione con una sintassi del genere:

```
INSERT INTO <tabella> ( <campi> ) <istruzione SELECT>
```

Utilizzabile come nell'esempio seguente:

```
INSERT INTO dirigenti(id, nome, cognome)
SELECT id, nome, cognome FROM dipendenti WHERE ruolo='Direttore'
```

Quest'istruzione forza l'inserimento dei dati nei rispettivi campi (ID, Nome e Cognome) della tabella di destinazione, rendendo l'inserimento corretto.

È tuttavia possibile inserire singoli valori all'interno di una tabella, senza utilizzare un'istruzione SELECT, ad esempio per includere dei valori arbitrari; la sintassi generale è questa:

```
INSERT INTO <tabella> VALUES( <valori> )
```

Ciascun valore specificato a seguito dell'operatore VALUES e seguito da parentesi, deve essere separato da una virgola. Ad esempio inseriremo nella tabella dirigenti i seguenti valori

- Nome: Mimmo
- Cognome: Corinzi
- ID=10

```
INSERT INTO dirigenti VALUES('Mimmo', 'Corinzi', 10)
```

Naturalmente 10 essendo un numero non sarà circondato dal carattere di apostrofo che contraddistingue soltanto le strighe.
In maniera analoga ai casi precedenti possiamo forzare l'ordine d'inserimento dei dati nei campi:

```
INSERT INTO <tabella> ( <campi> ) VALUES( <valori> )
```


Da trascrivere quindi con:

```
INSERT INTO dirigenti(id, cognome, nome) VALUES(10, 'Corinzi', 'Mimmo')
```

Esiste inoltre una seconda istruzione per l'inserimento di dati che non tutti i database supportano. Questa, a differenza della precedente, richiede che la tabella di destinazione non esista; infatti sarà l'istruzione stessa a crearla, con i dati specificati. Se la tabella di destinazione dovesse esistere sarà generato un errore:

```
SELECT <campi> INTO <tabella> FROM ...
```

Si tratta di una normale istruzione SELECT, che può includere qualsiasi operatore o altra clausola, che contiene la specifica della tabella di destinazione da creare ed in cui inserire i dati estratti dall'espressione seguente.

```
SELECT * INTO dipendenti2 FROM dipendenti
```

Questa istruzione semplicemente copia l'intera tabella dipendenti dentro una nuova tabella di nome dipendenti2. Saranno creati i campi con i nomi ed i tipi corrispondenti della tabella di origine (dipendenti). Naturalmente l'espressione avrebbe potuto essere anche molto più complessa, con la completa libertà e flessibilità offerte dall'istruzione SELECT.

Tuttavia questa copia non sarà assolutamente identica; saranno infatti creati i campi dello stesso tipo ma non saranno stabiliti i vincoli, le chiavi o le caratteristiche avanzate presenti nella tabella di origine.

```
SELECT nome, cognome, id INTO dirigenti FROM dipendenti WHERE
ruolo='Direttore'
```

Riprodurrà gli effetti ottenuti con gli esempi precedenti ma in più si occuperà di creare la nuova tabella dirigenti:

Fibia FBI
14 Marzo 2004


Torna all'indice della sezione Database
