

Manuale

SQL

❖ Istruzioni DDL

❖ Creazione di una tabella : CREATE TABLE

Il comando `CREATE TABLE` consente di definire una tabella del database specificandone le colonne, con il tipo di dati ad esse associate, e le regole di integrità.

La creazione di una tabella avviene attraverso un'istruzione che può assumere un'articolazione molto complessa, a seconda delle caratteristiche particolari che da questa tabella si vogliono ottenere. La sintassi più semplice è quella seguente:

```
CREATE TABLE <nome tabella>
    (
        <elemento tabella> [{, <elemento tabella>} . . . ]
    )
```

Le clausole <elemento tabella> sono l'elemento più importante dell'istruzione in quanto consentono di definire le colonne, i vincoli e gli altri elementi specifici della tabella che si sta creando. E' possibile definire una o più clausole <elemento tabella>. Se si definiscono più clausole, occorre utilizzare una virgola di separazione.

La sintassi utilizzata per definire una colonna, ovvero l' <elemento tabella>, è la seguente:

```
<nome colonna> <tipo di dati>
[<clausola di default>] [<vincolo della colonna>]
```

In questo modo, per definire una colonna è sufficiente fornire il nome della colonna, dichiarare il tipo di dati che in essa può essere contenuto ed , inoltre, si ha la possibilità di fornire un valore di default e specificare gli eventuali vincoli.

La seguente sintassi consente di creare una chiave esterna tramite un vincolo sulla tabella:

```
CREATE TABLE nome tabella
    (
        <elemento tabella>
        [...],

        [ CONSTRAINT <nome vincolo> ]
        FOREIGN KEY (<colonna referenziante> [ {, <col referenziante>} . . . ] )
        REFERENCES <tabella referenziata> [( <colonne referenziate> ) ]
    )
```

❖ Eliminazione di una tabella : DROP TABLE

L'eliminazione di una tabella, con tutto il suo contenuto, è un'operazione semplice che dovrebbe essere autorizzata solo all'utente che l'ha creata. La sintassi del comando è la seguente:

```
DROP TABLE <nome_tabella>
```

❖ Istruzioni DML

❖ Inserimento dati in una tabella : **INSERT**

L'istruzione *INSERT* consente di aggiungere dati alle tabelle del database secondo la seguente sintassi :

```
INSERT INTO <nome tabella>  
    [ ( <nome colonna> [ {, <nome colonna> } . . . ] ) ]  
VALUES  
    ( <valore> [ {, <valore>} . . . ] )
```

❖ Aggiornamento dati in una tabella : **UPDATE**

L'istruzione *UPDATE* consente di aggiornare i dati contenuti in un database. Col questa istruzione è possibile modificare i dati contenuti in una o più righe e una o più colonne. La sintassi dell'istruzione *UPDATE* è la seguente :

```
UPDATE <nome tabella> SET  
    <nome colonna> = <espressione>  
    [ {, <nome colonna> = <espressione>} . . . ]  
  
[WHERE <condizione di ricerca >]
```

❖ Cancellazione dati da una tabella : **DELETE**

Una volta che i record di una tabella sono stati inseriti e/o modificati, è possibile che si debba provvedere alla loro cancellazione. La eliminazione di record da una tabella avviene tramite l'istruzione *DELETE* secondo la seguente sintassi :

```
DELETE FROM <nome tabella>  
    [WHERE <condizione di ricerca>]
```

❖ Istruzione SELECT

```
SELECT [ DISTINCT | ALL] { * | <elenco colonne>}
FROM <nome tabella> [{, <nome tabella>} . . . ]
[ WHERE <condizione di ricerca>]
[ GROUP BY <specifica di raggruppamento>]
[ HAVING <condizione di ricerca> ]
[ ORDER BY <condizione di ordinamento> ]
```



Predicati

<p>BETWEEN</p>	<p>Il predicato <i>BETWEEN</i> ha un funzionamento molto simile all'operatore di maggiore o uguale e minore o uguale utilizzati insieme. Il predicato <i>BETWEEN</i> viene utilizzato insieme alla parola riservata <i>AND</i> per identificare un intervallo di valori che possono essere inclusi come condizione di ricerca nella clausola <i>WHERE</i>. Per poter essere utilizzati, i valori contenuti nella colonna identificata devono rientrare in questo intervallo. Quando si usa la clausola <i>BETWEEN</i>, occorre specificare la colonna cui viene applicata e le estremità inferiore e superiore dell'intervallo.</p> <p>Esempio :</p> <pre>SELECT Cognome, Nome FROM tbClienti WHERE (Cognome BETWEEN 'A' AND 'M') ORDER BY Cognome, Nome;</pre>
<p>NULL</p>	<p>Il predicato <i>IS NULL</i> confronta il valore in una colonna con il valore <i>Null</i>. L'uso di questo predicato è il solo modo per controllare la presenza del valore <i>Null</i> in una colonna. E' possibile inserire l'operatore di negazione <i>NOT</i> per valutare la condizione opposta, ovvero per controllare se un attributo non ha valore <i>Null</i>.</p> <p>Esempio :</p> <pre>SELECT Cognome, Nome, DataOrdine FROM tbClienti WHERE (DataOrdine IS NULL) ORDER BY Cognome;</pre>
<p>LIKE</p>	<p>Il predicato LIKE consente di fornire un argomento di ricerca più flessibile in cui si specificano valori <i>simili</i> ai valori contenuti nel database. Questo è particolarmente utile se si conosce solo una parte di un valore e si devono ricercare informazioni sulla base di queste informazioni parziali. Per esempio, si supponga di non conoscere esattamente l'Indirizzo di un cliente ma di conoscerne con certezza solo una parte. Utilizzando il predicato <i>LIKE</i> si possono richiedere valori simili a quelli conosciuti e pertanto capire se il database contiene le informazioni richieste.</p> <p>Il predicato <i>LIKE</i> confronta il valore di un attributo di tipo carattere con un modello di stringa che può contenere caratteri jolly.</p>

	<p>Esempio :</p> <pre>SELECT Cognome, Nome FROM tbClienti WHERE (Cognome Like 'M*') ORDER BY Cognome, Nome;</pre>
IN	<p>Il predicato IN consente di determinare se i valori contenuti nella colonna specificata di una tabella si trovano in un determinato elenco o in un'altra tabella (questo caso si tratterà nel paragrafo relativo alle subquery). Il predicato IN, in altre parole, controlla se un valore appartiene ad un insieme specificato di valori, cioè è possibile richiedere le righe di una tabella che hanno i valori di una colonna compresi in una lista di valori indicati dopo il predicato IN all'interno della condizione indicata nella clausola WHERE.</p> <p>Esempio :</p> <pre>SELECT Cognome, Nome FROM tbClienti WHERE (Nome IN ('Giuseppe', 'Mario', 'Carlo')) ORDER BY Cognome, Nome;</pre>



Funzioni di aggregazione

COUNT(<i>espr</i>), COUNT(*)	<p>Conteggio dei valori di una colonna, se per <i>espr</i> si specifica un nome di colonna, o conteggio di tutte le righe di una tabella o gruppo, se si specifica *. Con COUNT(<i>espr</i>) i valori NULL vengono ignorati, mentre con COUNT(*) vengono inclusi nel conteggio.</p> <p>Esempio :</p> <pre>SELECT COUNT(*) AS [Totale Clienti] FROM tbClienti;</pre>
SUM(<i>espr</i>)	<p>Somma dei valori di una colonna. La colonna deve contenere soltanto dati numerici. La funzione SUM considera i record contenenti campi di tipo <i>Null</i> come aventi valore 0.</p> <p>Esempio :</p> <pre>SELECT SUM(Fatturato) AS [Totale] FROM tbClienti;</pre>
MAX(<i>espr</i>)	<p>Valore massimo di una colonna o l'ultimo valore in ordine alfabetico per tipi di dati di testo. I valori NULL vengono ignorati.</p> <p>Esempio :</p> <pre>SELECT MAX(Fatturato) AS [FattMAX] FROM tbClienti;</pre>
MIN(<i>espr</i>)	<p>Valore minimo di una colonna o il primo valore in ordine alfabetico per tipi di dati di testo. I valori NULL vengono ignorati.</p> <p>Esempio :</p> <pre>SELECT MIN(Fatturato) AS [FattMIN] FROM tbClienti;</pre>
AVG(<i>espr</i>)	<p>Media dei valori di una colonna. La colonna deve contenere soltanto dati numerici. La media calcolata dalla funzione AVG equivale alla media aritmetica e non include nel calcolo i valori di tipo <i>Null</i> presenti nella colonna.</p> <p>Esempio :</p> <pre>SELECT AVG(Fatturato) AS [Mediafatt] FROM tbClienti</pre>