



IL LINGUAGGIO SQL

Il linguaggio SQL (Structured Query Languages) è un linguaggio non procedurale che è diventato uno standard tra i linguaggi per la gestione dei database relazionali.

Il linguaggio procedurale semplifica il lavoro del programmatore poichè non deve scrivere il programma ma solo poche righe di codice e sarà poi il motore del linguaggio che, passo dopo passo, scrive in automatico tutte le istruzioni. Oggi il linguaggio SQL viene usato in tutti i prodotti DBMS come linguaggio di comandi per l'utente della base di dati. Il linguaggio SQL consente all'utente di :

1. definire la struttura delle relazioni del database (funzioni di DDL);
2. modificare i dati contenuti nel database , con le operazioni di inserimento , variazione e cancellazioni (funzioni di DML);
3. gestire il controllo degli accessi e i permessi per gli utenti (funzioni di DCL);
4. porre interrogazioni al database (funzioni di Query Language);

IDENTIFICATORI E TIPI DI DATI

Questo linguaggio utilizza i caratteri alfabetici , le cifre decimali, gli operatori aritmetici e di confronto . Gli identificatori (nomi delle tabelle e degli attributi) sono costituiti da sequenze di caratteri con lunghezza massima uguale a 18 caratteri : devono iniziare con una lettera e possono anche contenere un carattere _.

Es:

NomeTabella.NomeAttributo

COMANDI E ISTRUZIONI

-CREATE TABLE (crea una nuova tabella)

Es:

CREATE TABLE Personale

(Matricola char(5) ,
Cognome char(30) ,
Nome char(30) ,

Gli operatori relazionali ricevono come argomento una relazione o un insieme di relazioni e restituiscono una singola relazione come risultato.

Vediamo brevemente questi otto operatori:

RESTRICT: restituisce una relazione contenente un sottoinsieme delle tuple della relazione a cui viene applicato. Gli attributi rimangono gli stessi.

PROJECT: restituisce una relazione con un sottoinsieme degli attributi della relazione a cui viene applicato. Le tuple della relazione risultato vengono composte dalle tuple della relazione originale in modo che continuino ad essere un insieme in senso matematico.

TIME: viene applicato a due relazioni ed effettua il prodotto cartesiano delle tuple. Ogni tupla della prima relazione viene concatenata con ogni tupla della seconda.

JOIN: vengono concatenate le tuple di due relazioni in base al valore di un insieme dei loro attributi.

UNION: applicando questo operatore a due relazioni compatibili, se ne ottiene una contenente le tuple di entrambe le relazioni. Due relazioni sono compatibili se hanno lo stesso numero di attributi e gli attributi corrispondenti nelle due relazioni hanno lo stesso dominio.

MINUS: applicato a due relazioni compatibili, ne restituisce una terza contenente le tuple che si trovano solo nella prima relazione.

INTERSECT: applicato a due relazioni compatibili, restituisce una relazione contenente le tuple che esistono in entrambe le relazioni.

DIVIDE: applicato a due relazioni che abbiano degli attributi comuni, ne restituisce una terza contenente tutte le tuple della prima relazione che possono essere fatte corrispondere a tutti i valori della seconda relazione.

Nelle seguenti tabelle, a titolo di esempio, sono raffigurati i risultati dell'applicazione di alcuni operatori relazionali alle relazioni Persone e Figli. Come nomi per le relazioni risultato si sono utilizzate le espressioni che le producono.

Persone

numero_persona	nome	cognome	data_nascita	sesso	stato_civile
2	Mario	Rossi	29/03/1965	M	Coniugato
1	Giuseppe	Russo	15/11/1972	M	Celibe
3	Alessandra	Mondella	13/06/1970	F	Nubile

Figli

numero_persona	nome_cognome	eta	sesso
2	Maria Rossi	3	F
2	Gianni Rossi	5	M

RESTRICT (Persone)

sesso='M'

numero_persona	nome	cognome	data_nascita	sesso	stato_civile
2	Mario	Rossi	29/03/1965	M	Coniugato
1	Giuseppe	Russo	15/11/1972	M	Celibe

PROJECT sesso (Persone)

sesso

M

F

RESTRICT (Persone)

sesto='M'

n.	nome	cognome	nascita	sesto	stato_civile	nome	eta'	sesto
Mario	Rossi	cognome	29/03/1965	M	Coniugato	Maria Rossi	3	F
Mario	Rossi	cognome	29/03/1965	M	Coniugato	Gianni Rossi	5	M

CREATE DATABASE nome_database

Con PostgreSQL e' anche disponibile un comando invocabile dalla shell Unix (o dalla shell del sistema utilizzato) che esegue la stessa operazione:

createdb nome_database

Per creare il nostro database bibliografico utilizzeremo quindi il comando:

createdb biblio

Una volta creato il database e' possibile creare le tabelle che lo compgono. L'istruzione SQL preposta a questo scopo e':

```
CREATE TABLE nome_tabella (  
nome_colonna tipo_colonna [ clausola_default ] [ vincoli_di_colonna ]  
[ , nome_colonna tipo_colonna [ clausola_default ] [ vincoli_di_colonna ] ... ]  
[ , [ vincolo_di_tabella ] ... ] )
```

nome_colonna: e' il nome della colonna che compone la tabella. Sarebbe meglio non esagerare con la lunghezza degli identificatori di colonna, dal momento che l'SQL Entry Level prevede nomi non piu' lunghi di 18 caratteri. Si consulti comunque la documentazione dello specifico database. I nomi devono iniziare con un carattere alfabetico.

tipo_colonna: e' l'indicazione del tipo di dato che la colonna potra' contenere. I principali tipi previsti dallo standard SQL sono:

- **CHARACTER(n)**
Una stringa a lunghezza fissa di esattamente n caratteri. CHARACTER puo' essere abbreviato con CHAR
- **CHARACTER VARYING(n)**
Una stringa a lunghezza variabile di al massimo n caratteri. CHARACTER VARYING puo' essere abbreviato con VARCHAR o CHAR VARYING.
- **INTEGER**
Un numero intero con segno. Puo' essere abbreviato con INT. La precisione, cioe' la grandezza del numero intero che puo' essere memorizzato in una colonna di questo tipo, dipende dall'implementazione del particolare DBMS.
- **SMALLINT**
Un numero intero con segno con precisione non superiore a INTEGER.
- **FLOAT(p)**
Un numero a virgola mobile, con precisione p. Il valore massimo di p dipende dall'implementazione del DBMS. E' possibile usare FLOAT senza indicazione della precisione, utilizzando quindi la precisione di default, anch'essa dipendente dall'implementazione. REAL e DOUBLE PRECISION sono dei sinonimi per un FLOAT con una particolare precisione. Anche in questo caso le precisioni dipendono dall'implementazione, con il vincolo che la precisione del primo non sia superiore a quella del secondo.

- *DECIMAL(p,q)*
Un numero a virgola fissa di almeno p cifre e segno, con q cifre dopo la virgola. DEC e' un'abbreviazione per DECIMAL. DECIMAL(p) e' un'abbreviazione per DECIMAL(p,0). Il valore massimo di p dipende dall'implementazione.
- *INTERVAL*
Un periodo di tempo (anni, mesi, giorni, ore, minuti, secondi e frazioni di secondo).
- *DATE, TIME e TIMESTAMP*
Un preciso istante temporale. DATE permette di indicare l'anno, il mese e il giorno. Con TIME si possono specificare l'ora, i minuti e i secondi. TIMESTAMP e' la combinazione dei due precedenti. I secondi sono un numero con la virgola, permettendo cosi' di specificare anche frazioni di secondo.

clausola_default: indica il valore di default che assumerà la colonna se non gliene viene assegnato uno esplicitamente nel momento della creazione della riga. La sintassi da utilizzare e' la seguente:

DEFAULT { valore | NULL }

dove, valore e' un valore valido per il tipo con cui la colonna e' stata definita.

vincoli_di_colonna: sono vincoli di integrita' che vengono applicati al singolo attributo. Sono:

- NOT NULL, che indica che la colonna non puo' assumere il valore NULL.
- PRIMARY KEY, che indica che la colonna e' la chiave primaria della tabella.
- una definizione di riferimento, con cui si indica che la colonna e' una chiave esterna verso la tabella e i campi indicati nella definizione. La sintassi e' la seguente:

*REFERENCES nome_tabella [(colonna1 [, colonna2 ...])]
[ON DELETE { CASCADE | SET DEFAULT | SET NULL }]
[ON UPDATE { CASCADE | SET DEFAULT | SET NULL }]*

Le clausole ON DELETE e ON UPDATE indicano quale azione deve essere compiuta nel caso in cui una tupla nella tabella referenziata venga eliminata o aggiornata. Infatti in tali casi nella colonna referenziante (che e' quella che si sta definendo) potrebbero esserci dei valori inconsistenti. Le azioni possono essere:

- CASCADE: eliminare la tupla contenente la colonna referenziante (nel caso di ON DELETE) o aggiornare anche la colonna referenziante (nel caso di ON UPDATE).
- SET DEFAULT: assegnare alla colonna referenziante il suo valore di default.
- SET NULL: assegnare alla colonna referenziante il valore NULL.
- un controllo di valore, con il quale si permette o meno l'assegnazione di un valore alla colonna, in base al risultato di un'espressione. La sintassi da usare e':

CHECK (espressione_condizionale)

dove espressione_condizionale e' un'espressione che restituisce vero o falso.

Ad esempio, se stiamo definendo la colonna COLONNA1, definendo il seguente controllo:

CHECK (COLONNA1 < 1000)

in tale colonna potranno essere inseriti solo valori inferiori a 1000.

vincolo_di_tabella: sono vincoli di integrita' che possono riferirsi a piu' colonne della tabella. Sono:

- la definizione della chiave primaria:

PRIMARY KEY (colonna1 [, colonna2 ...]) Si noti che in questo caso, a differenza della definizione della chiave primaria come vincolo di colonna, essa puo' essere formata da piu' di un attributo.

- le definizioni delle chiavi esterne:

FOREIGN KEY (colonna1 [, colonna2 ...]) definizione_di_riferimento

La definizione_di_riferimento ha la stessa sintassi e significato di quella che puo' comparire come vincolo di colonna.

- un controllo di valore, con la stessa sintassi e significato di quello che puo' essere usato come vincolo di colonna.

L'istruzione SQL che effettua l'inserimento di una nuova riga in una tabella e' INSERT. La sintassi con cui essa viene usata piu' comunemente e':

```
INSERT INTO nome_tabella [ ( elenco_campi ) ]  
VALUES ( elenco_valori )
```

nome_tabella e' il nome della tabella in cui deve essere inserita la nuova riga.

elenco_campi e' l'elenco dei nomi dei campi a cui deve essere assegnato un valore, separati fra loro da una virgola. I campi non compresi nell'elenco assumeranno il loro valore di default o NULL se non hanno un valore di default. E' un errore non inserire nell'elenco un campo che non abbia un valore di default e non possa assumere il valore NULL. Nel caso in cui l'elenco non venga specificato dovranno essere specificati i valori di tutti i campi della tabella.

elenco_valori e' l'elenco dei valori che verranno assegnati ai campi della tabella, nell'ordine e numero specificati dall'elenco_campi o in quello della definizione della tabella (se elenco_campi non viene specificato). I valori possono essere un'espressione scalare del tipo appropriato per il campo o le keyword DEFAULT o NULL, se il campo prevede un valore di default o ammette il valore NULL.

Il precedente esempio di inserimento viene quindi eseguito tramite le seguenti istruzioni SQL:

```
INSERT INTO Person VALUES ( 1, 'Agosti', 'Maristella' );  
INSERT INTO Person VALUES ( 2, 'Benfante', 'Lucio' );
```