SQL

Master GIS 2019

Claudio Rocchini
claudio@rockini.name

Istituto Geografico Militare

gennaio 2019

Introduzione

SQL (Structured Query Language) è un linguaggio standard per l'uso di basi di dati.
É un linguaggio di programmazione funzionale, utilizzabile in modo univoco su tutti i software di basi di dati indipendentemente dal produttore.

SQL

I comandi SQL principali solo solo 7:

- interrogazione: SELECT
- definizione dei dati: CREATE, ALTER, DESTROY
- manipolazione dati: INSERT, UPDATE, DELETE

Pre Introduzione a SELECT

Struttura generale di una SELECT

```
SELECT {valori};
```

Calcolo della risposta fondamentale

```
SELECT 42;
```

Risposta fondamentale con nome

```
SELECT 42 AS valore;
```

Valore letterali: numeri

Espressioni aritmetiche

```
SELECT 45 + 21 * 2;
```

Chiamata di funzione

```
SELECT cos(3.1415926) + sqrt(49);
```

Richiesta di piú di un valore

```
SELECT sqrt(49), 3*5, cos(2);
```

Valore letterali: parole

Valori parola: fra apicetti singoli

```
SELECT 'Buongiorno';
```

La scritta *SELECT Buongiorno;* interroga l'attributo Buongiorno.

Parole che contengono apicetto

```
SELECT 'L''area dell''edificio';
```

Funzioni o operatori su parole

```
SELECT  LENGTH('casa'), 'pesce' || 'cane'
```

Attenzione alla differenza fra numeri e parole

Vero o falso

```
SELECT 99 < 100;   -- vero!
SELECT '99' < '100'; -- falso (ordine lex.)
```

... ma questo é vero!

```
SELECT TO_NUMBER('99') < TO_NUMBER('100');
        -- cast di tipo
SELECT '99'::INTEGER < '100'::INTEGER;
```

Tipi di dato

INTEGER : numero intero;

REAL, DOUBLE PRECISION : numero con la virgola in singola o doppia precisione;

CHARACTER(n) : stringhe di lunghezza n;

CHARACTER VARYING : stringhe di lunghezza variabile;

BOOLEAN : valori di verità (vero o falso)[1];

DATE : data e ora;

GEOMETRY : tipo geografico.

[1] curiosamente Oracle non ha questo tipo di dato.

Comandi di definizione

- CREATE TABLE : crea una tabella;
- DROP TABLE: distrugge una tabella;
- ALTER TABLE: modifica la struttura di una tabella.

Comandi di definizione

Struttura di creazione di una tabella

```
CREATE TABLE nome_tabella
(    nome_colonna1 TIPO1,
     nome_colonna2 TIPO2,
     ...
     nome_colonnan TIPOn
);
```

Comandi di definizione

Creazione di una tabella (semplice)

```
CREATE TABLE strada
(
    nome          CHARACTER VARYING,
    classifica    CHARACTER(2),
    larghezza     REAL
);
```

Ma andava bene anche scritto su una sola riga...

Distruzione di una tabella

```
-- attenzione: comando irreversibile
DROP TABLE strada;
```

Un inciso: commenti al codice

```
/* creazione della tabella
   contenente le strade */
CREATE TABLE strada
(
    nome CHARACTER VARYING, -- ... della strada
    classifica CHARACTER(2), -- strada, fiume, ...
    larghezza REAL  -- media in metri
);
```

Creazione avanzata di una tabella

```
CREATE TABLE strada
(
    nome          CHARACTER VARYING PRIMARY KEY,
    classifica    CHARACTER(2) NOT NULL,
    larghezza     REAL
);
```

Modifica della struttura di una tabella

Aggiunta di una colonna

```
ALTER TABLE strada
ADD num_corsie INTEGER;
```

Rendere obbligatorio

```
ALTER TABLE strada
ALTER COLUMN larghezza SET NOT NULL;
```

Comandi per Manipolazione dei dati

- INSERT : inserisce nuove righe in una tabella (quindi inserisce nuovi dati);
- DELETE: cancella righe da una tabella;
- UPDATE: modifica i dati esistenti di una tabella.

Struttura del comando INSERT

Struttura Generale

```
INSERT INTO nome_tabella
 (nome_col1, nome_col2, ... , nome_colN)
 VALUES (valore1, valore2, ... , valoreN);
```

Esempio reale

```
INSERT INTO strada
  (nome,classifica,larghezza,num_corsie)
VALUES
  ('A1','01',16,4);
```

Esempi di INSERT

Inserimento parziale di dati

```
INSERT INTO strada
  (nome,classifica,num_corsie)
VALUES
  ('Aurelia','02',4);
```

Inserimento di tutte le colonne

```
INSERT INTO strada
VALUES
  ('A23','01',12,4);
```

Un inciso: il valore NULL

Inserimento parziale di dati con NULL

```
INSERT INTO strada
VALUES
('Emilia','02',NULL,4);
```

Test dei vincoli

Fallisce: chiave priamia

```
INSERT INTO strada (nome,...)
  VALUES ('A1',...);
INSERT INTO strada (nome,...)
  VALUES ('A1',...);
```

Fallisce: classifica obbligatoria

```
INSERT INTO strada
  (nome,larghezza,num_corsie)
VALUES
  ('campestre',12,4);
```

Cancellazione di dati

Cancellazione totale di una tabella

```
DELETE FROM strada;
```

Es. Cancellazione parziale

```
DELETE   FROM strada
WHERE    larghezza>20
AND      classifica='02';
```

Altri esempi di filtri

Oppure e NON

```
...
WHERE nome='A1' OR NOT classifica='01'
```

Nomi che iniziano per Em

```
... WHERE nome LIKE 'Em%';
```

Campi vuoti

```
... WHERE classifica IS NULL;
```

Il comando UPDATE

```
UPDATE nome_tabella
SET      nome_colonna1 = valore1,
         nome_colonna2 = valore2,
         ...
WHERE {condizioni};
```

Esempi di Update

Aggiornamento Puntuale

```
UPDATE   strada
SET      num_corsie = 2
WHERE    nome='A23';
```

Aggiornamento a Tappeto

```
UPDATE   strada
SET      shape = ST_MakeValid(shape);
```

Una seconda tabella

```
CREATE TABLE clas_stradale
(
  codice          CHARACTER(2) PRIMARY KEY,
  descrizione     CHARACTER VARYING NOT NULL
);

INSERT INTO clas_stradale
        VALUES('01','autostrada');
INSERT INTO clas_stradale
        VALUES('02','extraurbana principale');
...
```

Dichiarazione di una relazione

```
ALTER TABLE strada
ADD CONSTRAINT   strada_classifica_fk
FOREIGN KEY      (classifica)
REFERENCES       clas_stradale(codice);
```

La relazione

Test sulle relazioni

Fallisce: classifica 99 non esiste

```
INSERT INTO strada
VALUES
('Canistracci','99',16,4);
```

Fallisce: classifica 01 in uso

```
DELETE
FROM    clas_stradale
WHERE   id='01'
```

Creazione di un indice

Indice alfanumerico

```
CREATE INDEX strada_larghezza_idx
ON strada (larghezza);
```

Indice Spaziale

```
CREATE INDEX strada_shape_idx
ON strada USING GIST (shape);
```

Struttura base di SELECT

Struttura di Base

```
SELECT   colonna1,colonna2,...,colonnaN
FROM     tabella
WHERE    {condizioni}
ORDER BY colonna1,colonna2;
```

Esempio

```
SELECT   nome, larghezza
FROM     strada
WHERE    classifica='01'
ORDER BY nome;
```

Struttura base di SELECT

Tutte le colonne

```
SELECT   *
FROM     strada;
```

Filtri complessi

```
SELECT   nome, classifica
FROM     strada
WHERE    larghezza BETWEEN 2 AND 12
AND      num_corsie IN (2,4);
```

Funzioni Aggreganti

Alcune funzione aggregano le righe di un tabella:

- MIN: minimo dei valori;
- MAX: massimo dei valori;
- AVG: media dei valori (*average* in inglese);
- SUM: somma dei valori;
- COUNT: numero di valori;
- ST_Extent (geografico): estensione totale;

Query Aggreganti

Struttura generale

```
SELECT   {funzioni_aggr.}(attributi)
FROM     tabella
WHERE    {condizione sulle righe}
GROUP BY {sub totali}
HAVING   (filtri sull'aggregato)
```

Query Aggreganti

Aggregazione totale

```
SELECT   MIN(larghezza), MAX(larghezza),
         AVG(larghezza)
FROM     strada;
```

Aggregazione Parziale

```
SELECT   classifica,
         MIN(larghezza), MAX(larghezza),
         AVG(larghezza)
FROM     strada
GROUP BY classifica;
```

La funzione COUNT

Si usa sempre con *

```
SELECT   COUNT(*)
FROM     strada;
```

Parziale e ordinata

```
SELECT   classifica, COUNT(*)
FROM     strada
GROUP BY classifica
ORDER BY COUNT(*) DESC;
```

Alcuni dettagli

Si possono interrogare piú tabelle

```
... FROM strada, clas_stradale ...
```

Nome e cognome di un attributo

```
SELECT   strada.nome, clas_stradale.nome
FROM     strada,
         clas_stradale;
```

Realizzare una Join

```
SELECT   strada.nome,
         clas_stradale.descrizione
FROM     strada,
         clas_stradale
        -- Condizione di Join
WHERE    strada.classifica =
         clas_stradale.codice;
```

Join Left Outer

```
SELECT   clas_stradale.descrizione,
         strada.nome
FROM     clas_stradale
         LEFT OUTER JOIN
         strada
         ON (strada.classifica =
                  clas_stradale.codice);
```

Join Spaziale

Normale

```
SELECT   strada.*, comu.*
FROM     strada, comu
WHERE    ST_Contains(comu.shape,strada.shape);
```

Con Alias

```
SELECT   s.*, c.*
FROM     strada AS s,
         comune AS c
WHERE    ST_Contains(c.shape, s.shape);
```

Auto Join

L'alias in questo caso é obbligatorio:

```
SELECT   c1.*, c2.*
FROM     comune AS c1,
         comune AS c2
WHERE    ST_Overlaps(c1.shape, c2.shape)
AND      c1.id < c2.id;
-- Come mai ci vuole < invece di <> ?
```

Creazione di una vista

Vista = query salvata con nome

```
CREATE  VIEW strade_w AS

SELECT  strada.nome,
        clas_strad.descrizione
FROM    strada, clas_strad
WHERE   strada.classifica=clas_strad.codice;
```

Si usa come un tabella

```
SELECT  * FROM strade_w
WHERE   classifica='01';
```

Creare una tabella da SELECT

```
CREATE  TABLE nuova_tabella AS

SELECT  strada.nome,
        clas_stradale.descrizione
FROM    strada,
        clas_stradale
WHERE  strada.classifica=clas_stradale.codice;
```

Inserire dati da SELECT

```
INSERT INTO vecchia_tabella

SELECT   strada.nome,
         clas_stradale.descrizione
FROM     strada,
         clas_stradale
WHERE   strada.classifica=clas_stradale.codice;
```

Gli schemi

Creazione

```
CREATE SCHEMA claudio;
```

Creazione

```
CREATE TABLE claudio.strada
(
    nome          CHARACTER(64) PRIMARY KEY,
    classifica    CHARACTER(2) NOT NULL,
    larghezza     REAL
);
```

Utilizzo degli schemi

Utilizzo

```
SELECT * FROM claudio.strada;
```

Se non specificato si intende PUBLIC

```
SELECT * FROM strada;
        -- stessa tabella di
SELECT * FROM public.strada;
```