

Appunti di Basi di Dati

Andrea De Lorenzo, Giorgia Nadizar

March 23, 2022

1 SQL Data Definition Language

```
--Mostra i DB esistenti  
SHOW DATABASES;
```

```
--Crea un nuovo DB [se non esiste]  
CREATE DATABASE [IF NOT EXISTS] nomeDataBase;
```

```
--Elimina il DB [se esiste]  
DROP DATABASE [IF EXISTS] nomeDataBase;
```

```
--Tutti i comandi ora saranno riferiti a questo DB  
USE nomeDataBase;
```

```
--Creazione tabella  
CREATE TABLE [IF NOT EXISTS] nomeTabella(  
    nomeAttributo1 tipo,  
    attributo2 tipo,  
    ...  
    attributoN tipo  
)
```

1.1 Domini

- Numeri interi: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT
- Numeri razionali: float, double, numeric(i,n), decimal(i,n)
- Testo: CHAR, VARCHAR, BINARY, VARBINARY, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT
- Generico: TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB
- Tempo: YEAR, DATE, TIME, DATETIME, TIMESTAMP
- Spazio: GEOMETRY, POINT, LINESTRING, POLYGON
- Booleani: BOOL

1.2 Vincoli

- PRIMARY KEY: vicino all'attributo, o alla fine con l'attributo (gli attributi) chiave tra parentesi
- NOT NULL
- UNIQUE: vicino all'attributo o alla fine con gli attributi tra parentesi
- CHECK (supportato in MySQL sono da recente, non in MariaDB)
- FOREIGN KEY (attr1) REFERENCES Tabella(attr2): nella versione compatta si può specificare REFERENCES Tabella(attr2) vicino alla definizione di attr1. Per specificare il comportamento in caso di cancellazione si ha ON DELETE RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT. In caso di modifica, invece si ha: ON UPDATE RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT.
- CONSTRAINT [nome] ...: un qualsiasi tipo di vincolo può avere un nome. Si possono attivare con SET nome = 1 e disattivare con SET nome = 0
- ASSERTION: per specificare vincoli di integrità a livello di schema

1.3 Altro

- AUTO_INCREMENT
- nomeAttributo tipo DEFAULT valore
- nomeAttributo tipo COMMENT "commento"

1.4 Cancellazione

```
--Elimina la tabella [se esiste]
DROP TABLE [IF EXISTS] nomeTab1, nomeTab2, ...;
```

1.5 Cambiare lo schema

```
--Modifica la tabella
ALTER TABLE nomeTabella azione1 [, azione2, ...]
```

```
--Aggiunta colonna
ALTER TABLE nomeTabella
    ADD COLUMN nome tipo
    [ FIRST | AFTER nomeColonna ]
```

```
--Eliminazione colonna
ALTER TABLE nomeTabella
    DROP COLUMN nomeColonna
```

```
--Modifica colonna
ALTER TABLE nomeTabella
    CHANGE COLUMN nomeOriginale nomeNuovo tipo
```

```
--Ridenominazione tabella
ALTER TABLE nomeTabella
    RENAME TO nuovoNome
```

```
--Aggiunta di vincoli
ALTER TABLE nomeTabella
    ADD CONSTRAINT nome
    FOREIGN KEY (...)
    REFERENCES tabella(...)
```

```
--Rimozione di vincoli
ALTER TABLE nomeTabella
    DROP FOREIGN KEY nome
```

2 SQL Data Manipulation Language

```
--Selezione
SELECT attributo1 [, attributo2, ...]
  FROM tabella1 [, tabella2, ...]
  [WHERE condizione]
  [LIMIT x, n]
```

```
--Alias sugli attributi
SELECT attributo1 AS attr,
  FROM tabella1 [, tabella2, ...]
  [WHERE condizione]
```

```
--Alias e formule
SELECT attributo1/2 AS attrimezzi,
  FROM tabella1 [, tabella2, ...]
  [WHERE condizione]
```

```
SELECT t.attributo1,
  FROM tabella1 t
  [WHERE condizione]
```

```
--Ordinamento
SELECT ... FROM ... WHERE ...
  ORDER BY col1 [ASC|DESC], col2 [ASC|DESC], ...
```

```
--Rimozione duplicati
SELECT DISTINCT ... FROM ...
```

2.1 Condizioni

- Testo esatto: =
- Testo incompleto: [NOT] LIKE "%parteditesto" oppure [NOT] LIKE "_parteditesto"
- Più condizioni: AND oppure OR con le opportune parentesi
- Intervalli: col BETWEEN x AND y oppure col >= x AND col <= y
- Liste: col IN (val1, val2, ...)
- Null: col IS NULL

2.2 Funzioni

- Stringhe: `length()`, `reverse()`, `right()`, `trim()`, `concat()`
- Data e ora: `day()`, `year()`, `now()`, `month()`, `monthname()`
- Ordinamento: `FIELD(text, str1, str2, str3, ...)` ritorna la posizione della stringa `text` nella lista `str1, str2, str3, ...`

2.3 Decodificare le relazioni, unione ed intersezione

```
--Prodotto cartesiano
SELECT ... FROM tabella1, tabella2, ...
SELECT * FROM tabella1 CROSS JOIN tabella2;
```

```
--Inner join
SELECT ... FROM tabella1 t1
    INNER JOIN tabella2 t2
    ON t2.PK = t1.FK;
```

```
SELECT ... FROM tabella1
    INNER JOIN tabella2
    USING(attrComune);
```

```
--Senza specificare using (pericolose in caso di
    modifiche)
SELECT ... FROM tabella1
    NATURAL JOIN tabella2;
```

```
SELECT ... FROM tabella1
    LEFT [OUTER] JOIN tabella2
    ON PK = FK
```

```
SELECT ... FROM tabella1
    RIGHT [OUTER] JOIN tabella2
    ON PK = FK
```

```
SELECT ... FROM tabella1
    FULL [OUTER] JOIN tabella2
    ON PK = FK
```

```
SELECT ... FROM tabella1
    [INNER|LEFT|RIGHT|FULL] JOIN tabella2
    ON PK = FK
    [INNER|LEFT|RIGHT|FULL] JOIN tabella3
    ON PK = FK
```

```
SELECT ... FROM tabella1 t1
      [INNER|LEFT|RIGHT|FULL] JOIN tabella2 t2
      ON t1.a1 = t2.a2 AND t1.b1 <> t2.b2
```

```
--Unione (stessa struttura dei risultati richiesta)
SELECT ... FROM ...
      UNION [DISTINCT | ALL]
      SELECT ... FROM ...
      [UNION [DISTINCT | ALL]
      SELECT ... FROM ...]
      [ORDER BY criteri]
```

```
--Intersezione
SELECT ... FROM ...
      INTERSECT
      SELECT ... FROM ...
```

2.4 Raggruppare i dati

```
SELECT a1,a2,...,an
FROM tabella1 WHERE condizioni
GROUP BY a1,a2,...,an
```

```
SELECT a1,a2,...,an, aggregatore(ax)
FROM tabella1 WHERE condizioni
```

Esempi di aggregatori: COUNT, SUM, AVG, MAX, MIN

```
SELECT a1,a2,...,an, aggregatore(ax)
FROM tabella1 WHERE condizioni
GROUP BY a1,a2,...,an
```

```
SELECT a1,a2,...,an, aggregatore(ax)
      FROM tabella1 WHERE condizioni
      GROUP BY a1,a2,...,an
      HAVING condizioniAggregate
```

2.5 Subqueries

```
--Subquery indipendenti
SELECT a1,a2,...,an,(QUERY singolo val.)
      FROM (QUERY)
      WHERE a1 > (QUERY singolo val.)
      AND a2 IN (QUERY singolo attrib.)
```

```
--Subquery correlate
SELECT a1,a2,...,an
      FROM tab1 WHERE
      a1 > (SELECT c1
            FROM tab2
            WHERE tab2.c2 > tab1.a1)

SELECT a1,a2,...,an
      FROM tab
      WHERE EXISTS (QUERY singolo val.)
```

2.6 Aggiungere dati

```
--Inserimento a mano
INSERT INTO tabella(col1, col2, ...)
      VALUES (valore1, valore2, ...)
      [, (valore1, valore2, ...), ...]
```

```
--Inserimento e subquery
INSERT INTO tabella(col1, col2, ...)
      SELECT ...
```

2.7 Modificare dati

```
UPDATE tabella
SET col1 = valore1
[, col2 = val2...]
[WHERE condizione]
```

2.8 Eliminare dati

```
DELETE FROM tabella
[WHERE condizioni]
```