ESEMPI MYSQL

INGRESSO IN MYSQL

1) mysql -h localhost -u root -p

2) digitare la password (i caratteri sono invisibili e il prompt non si muove)

USCITA DA MYSQL

exit

VISUALIZZARE L'ELENCO DI TUTTI I DATABASE CREATI

SHOW DATABASES;

POSIZIONARSI IN UN DATABASE GIA' CREATO

USE nome_database;

VISUALIZZARE L'ELENCO DI TUTTE LE TABELLE DI UN DATABASE

SHOW TABLES;

VISUALIZZARE LE ISTRUZIONI USATE PER CREARE UNA TABELLA

SHOW CREATE TABLE tabella;

ISTRUZIONI PER DEFINIRE I DATI

CREARE UN NUOVO DATABASE

CREATE DATABASE nome_database;

CREARE UNA TABELLA CON DIVERSI TIPI DI ATTRIBUTI

```
CREATE TABLE tabella
(

stringa CHAR(20),          stringa di lunghezza massima 20 caratteri
numero_int INT,            numero intero
numero_dec
DECIMAL(5,2),              numero con 3 cifre intere (*) e 2 decimali
data DATE,                 data in formato AAAA-MM-GG
testo TEXT                 testo lungo e di lunghezza variabile (fino a 65.000
                           caratteri)


);
```

(*) il primo numero della coppia (5) indica il numero massimo di cifre intere + 2 byte per il segno e per il punto (che funge da virgola).

CREARE UNA TABELLA CON CHIAVE PRIMARIA E CHIAVE ESTERNA

Nella tabella_1 vengono definite una chiave primaria (primaria_1)
e una chiave esterna (esterna_1) che sara' confrontata con la 聽
chiave primaria (primaria_2) di tabella_2.

```
CREATE TABLE tabella_1
(
primaria_1 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
dato CHAR(100),
esterna_1 INT NOT NULL,
FOREIGN KEY (esterna_1) REFERENCES tabella_2 (primaria_2)
);
```

la chiave primaria puo' anche essere dichiarata cosi':

```
...
primaria_1 INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY (primaria_1),
...
```

CREARE UNA TABELLA CHE ABBIA LA STESSA STRUTTURA DI UNA TABELLA GIA' ESISTENTE

CREATE TABLE nuova_tab
LIKE vecchia_tab;

i dati di vecchia_tab non vengono copiati in nuova_tab.

VISUALIZZARE LA STRUTTURA DI UNA TABELLA

DESC tabella;

CANCELLARE UNA TABELLA

Cancella tutti i dati memorizzati, nonche' le informazioni che descrivono la tabella:

DROP TABLE tabella;

e' un comando pericolosissimo; prima di azionarlo, conviene fare una copia di sicurezza della tabella.

CAMBIARE IL NOME DI UNA TABELLA

ALTER TABLE nome_vecchio
RENAME AS nome_nuovo;

AGGIUNGERE UN ATTRIBUTO A UNA TABELLA

La prima istruzione aggiunge il nuovo attributo alla fine di quelli gia' presenti; la seconda istruzione colloca il nuovo attributo dopo un altro attributo gia' presente in tabella; la terza colloca il nuovo attributo al primo posto.

ALTER TABLE tabella
ADD nuovo_attributo tipo_dato;

ALTER TABLE tabella
ADD nuovo_attributo tipo_dato AFTER altro_attributo;

ALTER TABLE tabella
ADD nuovo_attributo tipo_dato FIRST;

CAMBIARE LA POSIZIONE DI UN ATTRIBUTO NELLA TABELLA

ALTER TABLE tabella
MODIFY COLUMN attributo_da_spostare tipo_dato AFTER altro_attributo;

ALTER TABLE tabella
MODIFY COLUMN attributo_da_spostare tipo_dato FIRST;

CAMBIARE IL NOME DI UN ATTRIBUTO DI UNA TABELLA (EVENTUALMENTE ANCHE IL TIPO DI DATO)

ALTER TABLE tabella
CHANGE nome_vecchio nome_nuovo tipo_dato

tipo_dato puo' essere uguale a quello precedente (e in questo caso cambia solo il nome dell'attributo), oppure puo' essere diverso (in questo caso cambia anche il tipo di dato).

MODIFICARE IL TIPO DI DATO DI UN ATTRIBUTO DI UNA TABELLA

ALTER TABLE tabella
MODIFY nome_attributo nuovo_tipo;

MODIFICARE LO STATUS DI UN ATTRIBUTO DI UNA TABELLA GIA' CREATA, TRASFORMANDOLO IN CHIAVE PRIMARIA

attributo_1 diventa chiave primaria.

ALTER TABLE tabella
ADD PRIMARY KEY (attributo_1);

TOGLIERE LO STATUS DI CHIAVE PRIMARIA ALL'ATTRIBUTO DI UNA TABELLA GIA' CREATA

ALTER TABLE tabella
DROP PRIMARY KEY;

dopo questo comando, l'attributo relativo continua a esistere, ma non e' piu' chiave primaria, ed ha Null=NO e Default=0.

**MODIFICARE LO STATUS DI UN ATTRIBUTO DI UNA TABELLA GIA' CREATA,
TRASFORMANDOLO IN CHIAVE ESTERNA**

attributo_1 della tabella_1 diventa una chiave esterna che sara' confrontata con attributo_2, chiave primaria della tabella_2.

```
ALTER TABLE tabella1
ADD FOREIGN KEY (attributo_1)
REFERENCES tabella2 (attributo_2);
```

CANCELLARE UN ATTRIBUTO DI UNA TABELLA

Cancella dalla struttura della tabella l'attributo e tutti i dati relativi:

```
ALTER TABLE tabella
DROP attributo;
```

e' un comando pericolosissimo.

ISTRUZIONI PER MODIFICARE I DATI

INSERIRE UNA NUOVA TUPLA IN UNA TABELLA

```
INSERT INTO tabella
SET
attributo1=dato1,
attributo2=dato2,
attributo3=dato3;
```

Se la tabella ha una chiave primaria che si incrementa automaticamente ad ogni inserimento e sbaglio nell'inserimento con INSERT INTO (ad es., perche' mi sono dimenticato di aggiungere il valore di una chiave esterna) l'inserimento non viene effettuato, ma il valore della chiave primaria viene incrementato lo stesso.

Si puo' fare anche cosi':

INSERT INTO tabella
VALUES(dato1,dato2,dato3);

se si vogliono inserire valori solo in un sottoinsieme degli attributi della nuova tupla:

INSERT INTO tabella (attributo2,attributo4)
VALUES (dato2,dato4);

MODIFICARE IL VALORE DELL'ATTRIBUTO DI UNA O PIU' TUPLE

UPDATE tabella
SET nome_attributo=nuovo_valore
WHERE (condizione che identifica la/le tupla/e da modificare);

e' un comando pericolosissimo, perche' se ci si dimentica di delimitare la/le tupla/e da modificare con il comando WHERE, tutti i valori di quell'attributo in tabella assumeranno il nuovo valore!

CANCELLARE UNA O PIU' TUPLE

Cancella tutte le tuple per cui e' vera la condizione:

DELETE FROM tabella
WHERE (condizione);

anche questo e' un comando pericolosissimo.

COPIARE TUTTI I DATI DI UNA TABELLA IN UN'ALTRA TABELLA

INSERT INTO tabella_2
SELECT *
FROM tabella_1;

copia tutti i dati di tabella_1 in tabella_2 (accodandoli, se tabella_2 aveva gia' dei dati); le due tabelle devono avere struttura identica.

ISTRUZIONI PER CERCARE E VISUALIZZARE DATI

VISUALIZZARE TUTTI GLI ATTRIBUTI DI TUTTE LE TUPLE DI UNA TABELLA

SELECT *
FROM tabella;

VISUALIZZARE SOLO ALCUNI ATTRIBUTI DI TUTTE LE TUPLE DI UNA TABELLA (PROIEZIONE)

SELECT attrib_1,attrib_2
FROM tabella;

VISUALIZZARE TUTTI GLI ATTRIBUTI SOLO DELLE TUPLE CHE RENDONO VERA UNA CERTA CONDIZIONE (SELEZIONE)

SELECT *
FROM tabella
WHERE (condizione);

VISUALIZZARE SOLO ALCUNI ATTRIBUTI SOLO DELLE TUPLE CHE RENDONO VERA UNA CERTA CONDIZIONE (PROIEZIONE + SELEZIONE)

SELECT attrib_1,attrib_2
FROM tabella
WHERE (condizione);

CERCARE SE UNA STRINGA DATA E' CONTENUTA IN UN ATTRIBUTO DI TIPO STRINGA

Cerco la stringa "Paol" nell'attributo nome della tabella:

SELECT *
FROM tabella
WHERE nome LIKE "%paol%";

cosi' trovo tutti i nomi come Paolo, Paola, Paolino, Giampaolo, ecc.;
LIKE non distingue fra maiuscole e minuscole.

VISUALIZZARE UN CAMPO CALCOLATO (DOPPIO)

```
SELECT importo,(importo*2) AS doppio
FROM tabella;
```

VISUALIZZARE LE TUPLE MOLTO LUNGHE IN RIGHE DIVERSE

```
SELECT *
FROM tabella
\G
```

Se si mette ; alla fine, funziona lo stesso, ma esce il messaggio: ERROR: No query specified.

APPLICARE UNA FUNZIONE ARITMETICA A SOTTOINSIEMI AGGREGATI DI DATI

tabella e' dotata di due attributi nome (tipo stringa) e importo (dato numerico); voglio fare le somme parziali - e visualizzare i risultati - degli importi di tutti i sottoinsiemi di tuple con lo stesso valore dell'attributo nome:

```
SELECT nome,SUM(importo)
FROM tabella
GROUP BY nome
ORDER BY SUM(importo) DESC;
```

con l'ultima riga (facoltativa) comando di visualizzare le somme in ordine decrescente.

CREARE UNA VISTA

La vista e' una tabella che contiene i dati ottenuti con l'interrogazione posta dopo AS:

```
CREATE VIEW nome_vista
AS
SELECT ...
FROM ...
WHERE ...;
```

sulla tabella vista e' possibile formulare ulteriori interrogazioni:

```
SELECT ...
FROM nome_vista
WHERE ...
```

il comando CREATE VIEW salva su file le istruzioni per ricreare la vista, ma non salva i dati della tabella vista; gli aggiornamenti apportati alle tabelle usate per creare la vista sono applicati automaticamente anche alla vista stessa (ma questo non vale per le modifiche della struttura delle tabelle: aggiunta, eliminazione o modifica di nome/tipo attributi);

il comando SHOW TABLES mostra anche le viste create; per cancellare una vista, bisogna digitare: DROP VIEW nome_vista;

VISUALIZZARE INSIEME I DATI DI DUE TABELLE CON UNION

Prima tabella: Europa

Nazione	*Capitale*
Francia	Parigi
Gran Bretagna	Londra
Italia	Roma

Seconda tabella: Africa

Nazione	*Capitale*
Algeria	Algeri
Marocco	Rabat
Senegal	Dakar

```
SELECT *
FROM Europa
UNION
SELECT *
FROM Africa;
```

si ottiene:

Nazione	*Capitale*
Francia	Parigi
Gran Bretagna	Londra
Italia	Roma
Algeria	Algeri
Marocco	Rabat
Senegal	Dakar

VISUALIZZARE DIVERSAMENTE I DATI DI UNA TABELLA CON UNION

Tabella: partite

Squadra_1	*Gol_1*	*Squadra_2*	*Gol_2*
Inter	2	Milan	1
Juventus	3	Lazio	3

```
SELECT Squadra_1 AS Squadra, Gol_1 AS Gol
FROM partite
UNION ALL
SELECT Squadra_2 AS Squadra, Gol_2 AS Gol
FROM partite;
```

si ottiene:

Squadra	*Gol*
Inter	2
Juventus	3

```
Milan     1

Lazio     3
```

UNION non ripete - nel risultato - tuple identiche; se si vuole evitare questo effetto (ad es., perche' possono esserci nella tabella due partite con le stesse squadre e lo stesso numero di gol) bisogna usare UNION ALL.

ALTRE ISTRUZIONI UTILI

IMPORTARE DATI DA UN FILE DI TESTO A UNA TABELLA SQL

Nel file di testo (in formato CSV) i valori degli attributi della stessa tupla devono trovarsi nella stessa riga e devono essere separati dalla virgola. Ad ogni tupla corrisponde una riga.

Al prompt di linux digitare:

mysql --local-infile -u root -p nome_database

LOAD DATA LOCAL INFILE '/percorso/nome_file.txt'
INTO TABLE tabella
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
(attributo_1,attributo_2, ...);

Il nome del file di dati da importare deve avere (senza considerare l'estensione) lo stesso nome della tabella in cui importare i dati.

SALVARE IN UN UNICO FILE (DI TESTO) TUTTE LE TABELLE DI UN DATABASE

Questo vale quando si usa il motore InnoDB.

1) bisogna operare dalla shell di linux (non da quella di mysql);

2) posizionarsi nella directory dove si vuole collocare il file coi dati del db;

3) digitare:

mysqldump -u root -p nome_database>nome_file

4) digitare password di mysql.

SCRIVERE E UTILIZZARE UNO SCRIPT SQL

1) in un file di testo (con estensione sql) scrivere le varie istruzioni SQL; il file puo' essere salvato in qualsiasi directory accessibile all'utente;

2) entrare in mysql e posizionarsi con use nel database contenente le tabelle da manipolare;

3) digitare:

SOURCE /percorso/nome_file_script.sql;

UTILIZZARE VARIABILI IN UNO SCRIPT SQL

In uno script possono essere usate una o piu' variabili contenenti valori di riferimento da usare per confronti e altre operazioni; il valore viene assegnato dall'utente alla variabile prima della chiamata dello script; il nome di una variabile deve essere preceduto da @ (ad es., @chiave_ricerca).

Testo dello script, salvato in un file con estensione sql:

```
SELECT *
FROM tabella
WHERE attributo=@chiave_ricerca;
```

assegnazione di un valore a una variabile (dal prompt di Mysql):

SET @chiave_ricerca=valore; (numero, data, stringa, ecc.)

il contenuto di una variabile inserita nel modo anzidetto puo' essere visualizzato con:

SELECT @chiave_ricerca;

poi si aziona lo script:

SOURCE /percorso/nome_file_script.sql;