ITIS-LS "Francesco Giordani" Caserta
prof. Ennio Ranucci
a.s. 2019-2020

Manuale SQL

Esercitazioni svolte in ambiente MySQL
(EasyPHP Devserver 17.0)

Accesso iniziale a mySql:

Prompt di DOS: "cmd";

- cd \; //change directory

Percorso

C:\Program Files (x86)\EasyPHP-Devserver-17\eds-binaries\dbserver\mysql5717x86\bin

Per accedere: "mysql -h -u -p "; //con username e password

Per accedere in localhost senza password: "mysql -u root".

Tipi di variabili:

Tipo	Descrizione	Esempi e range di variabilità
BIT	Singolo bit: Definisce il tipo booleano.	**0** (corrisponde a false o No) **1** (corrisponde a true o Si)
BIT(N)	Stringa di bit di lunghezza fissa pari a N.	**BIT(5)** Esempio di costante: "01101"
BIT VARYNG(N)	Stringa di bit di lunghezza variabile avente dimensione massima pari a N.	**BITVARYNG(5)** Esempio di costante:" 0011011"
CHARACTER o CHAR	Singolo carattere.	"C", "5", "%", "c"
CHARACTER(N) o CHAR(N)	Stringa di caratteri di lunghezza fissa pari a N caratteri. CHAR(1) corrisponde a CHARACTER o CHAR.	N varia da 1 a 15000. Esempio di costante: "rty56"
CHARACTER VARYNG(N) o VARCHAR(N)	Stringa di caratteri di lunghezza variabile con dimensione massima pari a N.	**VARCHAR(5)** Esempio di costante: "rty56"
DATE	Data nel formato "AAAA/MM/GG" oppure "AAAA-MM-GG".	"2006/12/25" "2006-12-25"

DECIMAL(P,D) o DEC(P,D)	Numero reale in	**DECIMAL(6,2)**
	fixedpoint con un numero massimo di cifre prima del punto decimale pari a P e un numero massimo di cifre dopo il punto decimale pari a D. Le dimensioni massime di P e D sono definite dall'implementazione	Esempio di costanti sono: 100.3, +0.7, -.3
DOUBLE PRECISIONE	Numero reale in floatingpoint con precisione per la mantissa doppia rispetto alla precisione di un FLOAT.	L'esponente generalmente varia da -127 a +128
FLOAT	Numero reale in floatingpoint con precisione definita dall'implementazione. Generalmente vale 15 per la mantissa.	Da 1E-28 a 1E+38. Stessi esempi del **REAL** per le costanti
FLOAT(P)	Numero reale in floatingpoint con precisione P per la mantissa.	La dimensione di P è definita dall'implemetazione. Generalmente P varia da 1 a 45
INTEGER o INT	Numero intero con precisione superiore a **SMALLINT**(generalmente occupano 4 byte, ma dipende dall'implementazione).	Generalmente da -2147483648 a +2147483647
INTEGER(N) o INT(N)	Numero intero con precisione N (numero massimo di cifre che il numero può contenere). Anche senon standard, è supportato dalla maggior parte delle implementazioni SQL.	**INT(5)**
SMALLINT	Numero intero con precisione inferiore a **INTEGER** (generalmente occupano 2 byte a memoria).	Generalmente da -32768 a +32767

REAL	Numero reale in floatingpoint con precisione definita dall'implementazione. Generalmente vale 7 per la mantissa.	Da 1E-38 a 1E+38. Esempi di costanti sono: 10E4 +24.7E-3 – 7.897E+12
TIME	Ora nel formato ora, minuti, secondi e millisecondi.	"10:34:25:42"
TIMESTAMP	Data e orario nel formato anno, mese, giorno, ora, minuti, secondi e millisecondi.	"2006/12/25 10:34:25:42"

DECIMAL (3,1) SIGNIFICA CHE IL TOTALE DELLE CIFRE E' 3 DI CUI 2 PARTE INTERA E 1 PARTE REALE

Comandi SQL DDL - Data Definition Language

(Agisce sulla struttura del Database) :

- **SHOW DATABASES;**

- **CREATE DATABASE <NomeDatabase> ;**

- **USE <NomeDatabase> ;**

- **SHOW TABLES;**

- **CREATE TABLE (...)**

Esempio: CREATE TABLE studenti (matStudente varchar (5) NOT NULL, cogStudente varchar (50) NOT NULL, nomStudente varchar (50) NOT NULL, cittaStudente varchar (50), PRIMARY KEY (matStudente));

- **DESCRIBE <nome tabella>;** //visualizza struttura tabella

~~- **ALTER TABLE ADD ( ) [BEFORE** <nomeAttributo>**];** //per aggiungere un campo~~

Esempio: ALTER TABLE studenti ADD (indirizzo varchar (50));

- **ALTER TABLE** studenti **ADD COLUMN** indirizzo char(50) **AFTER** CITTA**;** //per aggiungere un campo

- **ALTER TABLE CHANGE;** //modifica del nome di un campo

Esempio: ALTER TABLE studenti CHANGE indirizzo indStudenti varchar (50);

- **ALTER TABLE DROP;** //per eliminare un campo

Esempio: ALTER TABLE studenti DROP

indStudenti;

- **DROP TABLE <nome tabella> ;**//cancella una tabella

- DROP DATABASE <nome database>; //cancella un database

- CREATE UNIQUE INDEX ON (); //crea un indice su attributi chiave, se non specificato, su attributi non chiave

CREATE UNIQUE INDEX <nomeIndice> ON <nomeTabella>(<nomeAttibuto1>,<nomeAttributo2>,…<nomeAttributoN>); //crea indice

-DROP INDEX <nomeindice> **ON** <nomeTabella>; //elimina indice

- **ALTER TABLE** studenti **ENGINE**=InnoDB; //supporta le foreign key

- ALTER TABLE studenti ADD (siglaClasse varchar (5), FOREIGN KEY (siglaClasse) REFERENCES classi (siglaClasse)); // aggiunge la chiave esterna "siglaClassi" alla tabella "studenti";

Comandi SQL DML - Data Manipulation Language

(inserire, modificare e gestire dati memorizzati):

- INSERT INTO VALUES(…); //inserimento dei valori in una tabella

Esempio: INSERT INTO studenti (matStudente, cogStudente, nomStudente, cittaStudente) VALUES ("00001", "Ardivelo", "Nazareno", "Frignano");

- **UPDATE SET;** //aggiorna i dati di una

tabella Esempio: UPDATE<Tabella>

SET <nome campo> = <nuovo valore>

[WHERE<condizione>]

Esempio: UPDATE studenti SET nomstud=’caio’ WHERE matstud=’mat1’;

DEFAULT

-Create table voti (voto(2) not null, materia char(20)

default “inf”);

insert into voti(voto) values(10);

- **DELETE FROM WHERE;** //viene utilizzata per eliminare le righe di una tabella Esempio:

DELETE FROM <nome tabella> WHERE <condizione>;
DELETE FROM studenti WHERE voto=8;

Comandi SQL DCL - Data Control Language

(impostare politiche relative al controllo ed alla sicurezza dei dati):

- GRANT: Per concedere a gruppi di persone i diritti di accesso necessari ad interagire su un determinato insieme di dati.

Sintassi:

- GRANT <ElencoPrivilegi> ON [<NomeDatabase>.]<NomeTabella>

TO (<ElencoUtentiAutorizzati>) [PUBLIC]

[IDENTIFIED BY]

[WITH GRANT OPTION];

- **REVOKE:** Per revocare determinati permessi di accesso. Sintassi:

- REVOKE <ElencoPrivilegi> ON <NomeTabella>

FROM <ElencoUtentiAutorizzati> [PUBLIC];

Dove i componenti di <ElencoPrivilegi> possono essere:

ALL che consente l'accesso globale alla tabella specificata;

SELECT che consente l'accesso in lettura a tutti i campi della tabella specificata;

INSERT - UPDATE [(<Attributo>)] che consente l'inserimento di dati eventualmente solo sulla colonna specificata da <Attributo>;

DELETE che consente la cancellazione di righe della tabella specificata;

REFERENCES [(<Attributo>)] che consente il riferimento a tutti i campi della tabella o eventualmente solo a quello specificato da <Attributo> nei vincoli di integrità;

ALL PRIVILEGES rappresenta simultaneamente tutti i privilegi possibili riferiti a un oggetto;

Le viste

In SQL è possibile definire un'altra classe di tabelle, chiamate viste, che non sono fisicamente memorizzate nella base di dati (sono infatti costruite nella memoria RAM), ma che possono essere definite solo logicamente.

Sintassi:

- CREATE VIEW <NomeVista> AS

<Query>; dove:

<NomeVista> è il nome assegnato alla tabella fittizia della vista;

<Query> è una normale query formulata con il comando SELECT;

Per cancellare una vista:

- DROP VIEW <NomeVista>;

Query:

- Unione

L'unione di due tabelle è una nuova tabella che ha per schema lo stesso schema e per istanza l'unione delle tuple delle due tabelle (N.B. è necessario che le due tabelle abbiano gli attributi dello stesso tipo).

Esempio:

- (SELECT Cognome,Nome FROM registi) UNION (SELECT Cognome,Nome FROM attori);

- Intersezione

L'intersezione tra due tabelle è una nuova tabella che ha per schema lo stesso schema e per istanza l'intersezione delle tuple delle due tabelle.

Esempio:

- (SELECT Cognome,Nome FROM registi) INTERSECT (SELECT Cognome,Nome FROM attori);

MySQL non supporta INTERSECT per cui lo stesso obiettivo si può ottenere con la seguente:

- SELECT registi.cognome,registi.nome FROM registi INNER JOIN attori ON CodAttore = CodRegista;

- Differenza

La differenza tra due tabelle è una nuova tabella che ha per schema lo stesso schema e per istanza tutte le tuple della prima tabella che non sono presenti nella seconda tabella.

Esempio:

- (SELECT Cognome,Nome FROM registi) MINUS (SELECT Cognome,Nome FROM attori);

MySQL non supporta MINUS per cui lo stesso obiettivo si può ottenere con la seguente:

- SELECT * FROM clienti LEFT JOIN clienti09 ON clienti.CodCliente = clienti09.CodCliente WHERE clienti09.CodCliente IS NULL;

- Proiezione:

Data una tabella applicare l'operatore di proiezione significa creare una nuova tabella che ha per schema gli attributi indicati nella proiezione e per istanza la stessa istanza della tabella di partenza (senza ripetizioni).

Sintassi:

- SELECT * FROM <nome tabella> ; //mostra i valori contenuti in una tabella

- SELECT <elenco di colonne> FROM <nome tabella> //proiezione con ripetizione ;

- SELECT DISTINCT <elenco di colonne>FROM <nome tabella> ; //proiezione senza ripetizioni

- Selezione:

Data una tabella applicare l'operatore di selezione significa creare una nuova tabella che ha per schema lo stesso schema della tabella di partenza e per tuple (righe) quelle che verificano la condizione indicata.

Sintassi:

- SELECT * FROM <nome tabella> WHERE<criterio di selezione> ;

- Proieizione e selezione:

- SELECT <elenco di campi> FROM <nome tabella> WHERE <criterio di selezione> ;

- Congiunzione (JOIN):

Data due tabelle con almeno un attributo in comune, applicare l'operatore di congiunzione significa creare una nuova tabella che ha per schema l'unione dei due schemi riportando una sola volta gli attributi in comune e per istanza la concatenazione delle tuple che hanno lo stesso valore negli attributi in comune.

Sintassi:

SELECT * FROM <tabella 1> INNER JOIN <tabella 2> ON tabella1.attributo = tabella2.attributo;

Esempio:

SELECT * FROM studenti INNER JOIN classi ON studenti.siglaClasse = classi.siglaClasse;

Left Join:

Elenco dei clienti del 2010 che erano già clienti nel 2009 e dei clienti del 2009 che non sono rimasti clienti nel 2010:

- SELECT * FROM clienti09 LEFT JOIN clienti ON clienti09.CodCliente = clienti.CodCliente; Elenco

dei clienti del 2009 che non sono rimasti clienti nel 2010 (corrisponde alla differenza):

- SELECT * FROM clienti09 LEFT JOIN clienti ON clienti09.CodCliente = clienti.CodCliente WHERE clienti09.CodCliente IS NULL;

Right Join:

- SELECT * FROM clienti09 RIGHT JOIN clienti ON clienti09.CodCliente = clienti.CodCliente;

- SELECT * FROM clienti09 RIGHT JOIN clienti ON clienti09.CodCliente = clienti.CodCliente WHERE clienti09.CodCliente IS NULL;

Full Join:

Elenco dei clienti del 2009 e del 2010:
- SELECT * FROM clienti09 FULL JOIN clienti;

Il comando FULL JOIN non è supportato da MySQL quindi si ricorre all'unione dei due join esterni.

- (SELECT * FROM clienti09 LEFT JOIN clienti ON clienti09.CodCliente = clienti.CodCliente)

UNION

(SELECT * FROM clienti09 RIGHT JOIN clienti ON clienti09.CodCliente = clienti.CodCliente);

Cross Join (prodotto cartesiano):

Il CROSS JOIN è l'insieme di tutte le possibili concatenazioni.

- SELECT * FROM clienti09 CROSS JOIN clienti;

SubQuery

Per eseguire query complesse è possibile strutturare opportunamente più comandi select. Ciò consente di costruire un'interrogazione al cui interno sono presenti altre interrogazioni, dette sottointerrogazioni o subquery.

Tipi di Subquery:

Predicati ANY e ALL

- **ANY:** la condizione della clausola WHERE è vera se il valore dell'attributo <attributo> compare in almeno uno dei valori forniti dalla subquery<sottoquery>;

- **ALL:** la condizione della clausola WHERE è vera se il valore dell'attributo <attributo> compare in tutti quelli restituiti dalla subquery<sottoquery>;

Sintassi:

- SELECT <ListaAttributi>

FROM <ListaTabelle>

WHERE <Attributo><OperatoreRelazionale> ANY | ALL (<SottoQuery>);

Esempio:

"Nome e cognome dei dipendenti con stipendio netto maggiore rispetto a tutti gli stipendi medi d'Europa"

- SELECT Nome,Cognome

FROM Dipendenti

WHERE StipendioNetto> ALL (SELECT DISTINCT StipendioMedio FROM Stipendi WHERE Continente = "Europa")

Tipi di Subquery:

Predicati IN e NOT IN

- **IN:** la condizione della clausola WHERE è vera se il valore dell'attributo <attributo> appartiene all'insieme dei valori fornito dalla subquery<sottoquery>;

- **NOT IN:** la condizione della clausola WHERE è vera se il valore dell'attributo <attributo> non appartiene all'insieme dei valori fornito dalla subquery<sottoquery>;

Sintassi:

- SELECT <ListaAttributi>

FROM <ListaTabelle>

WHERE <Attributo><OperatoreRelazionale> IN | NOT IN

(<SottoQuery>); Esempio:

"Quanti sono gli animali dello zoo originari dell'Africa ?"

- SELECT COUNT(CodAnimale)

FROM zoo

WHERE CodRazza IN (SELECT CodRazzaFROM razza WHERE Continente="Africa");

Tipi di Subquery:

Predicati EXISTS e NOT EXISTS

- **EXISTS:** la condizione della clausola WHERE è vera se la subquery<sottoquery> produce una tabella non vuota;

- **NOT EXISTS:** la condizione della clausola WHERE è vera se il risultato della subquery<sottoquery> è una tabella vuota, senza alcuna riga;

Sintassi:

- SELECT <ListaAttributi>

FROM <ListaTabelle>

WHERE EXISTS | NOT EXISTS (<SottoQuery>);

Esempio: "Quali sono i clienti che hanno richiesto di viaggiare ?"

- SELECT * FROM CLIENTE C

WHERE EXISTS (SELECT * FROM Richiede H WHERE C.CodCli = H.CodCli);

Funzioni di aggregazione:

- COUNT

- SELECT COUNT() AS "Conteggio" FROM ;

- SELECT COUNT() FROM ; Esempio:

SELECT COUNT(*) AS "Conteggio" FROM classi;

SELECT COUNT(SiglaClasse) AS "Conteggio" FROM classi; SELECT COUNT(*) FROM

classi;

- ORDER BY

- SELECT * FROM ORDER BY DESC; //decrescente

- SELECT * FROM ORDER BY ASC; //crescente

Esempio:

SELECT * FROM classi ORDER BY numAula ASC;

- MIN E MAX

- SELECT MIN() FROM ;

- SELECT MAX() FROM ;

Esempio:

SELECT MIN(NumAula) FROM classi;

SELECT MAX(NumAula) FROM classi;

- GROUP BY

- SELECT COUNT() FROM GROUP BY ;

Esempio:

SELECT COUNT(*) FROM studenti GROUP BY citta;
SELECT citta,COUNT(*) FROM studenti GROUP BY citta;

- AVG

- SELECT AVG() FROM ;

Esempio:

SELECT AVG(eta) FROM studenti;

- HAVING E BETWEEN

- SELECT COUNT() FROM GROUP BY HAVING ;

- SELECT COUNT() FROM GROUP BY HAVING

BETWEEN; Esempio:

SELECT eta,citta,COUNT(*) AS "conteggio" FROM studenti GROUP BY eta HAVING eta>18;

SELECT eta,citta,COUNT(*) AS "conteggio" FROM studenti GROUP BY eta HAVING eta BETWEEN 10 AND 30;

Operatori di confronto:

- IN:

Richiede che il valore di <Attributo> sia tra quelli specificati da: <Valore1>, .. ,

<ValoreN> <Attributo> IN (<Valore1>, .. , <ValoreN>);

Esempio:

- SELECT * FROM clienti WHERE clienti.CodCliente IN(1,2);

- Between:

Richiede che il valore di <Attributo> sia compreso tra i valori <Min> e

<Max> <Attributo> BETWEEN <Min> AND <Max>;

- Not Between:

Richiede che il valore di <Attributo> non sia compreso tra i valori <Min> e

<Max> <Attributo> NOT BETWEEN <Min> AND <Max>;

- Like:

Richiede che il valore di <Attributo> assuma il formato specificato da <Espressione1>

<Attributo> LIKE <Espressione1>;

Esempio:

SELECT * FROM clienti WHERE clienti.Cognome LIKE("r%"); //tutti i cognomi dei clienti che iniziano per la lettera "r"

- Not Like:

Richiede che il valore di <Attributo> non assuma il formato specificato da <Espressione1>

<Attributo> NOT LIKE <Espressione1>;

Vincoli di dominio:

relativi al singolo attributo, generalmente utilizzati nella creazione della tabella (NOT NULL, DEFAULT, CHECK);

- Check: al suo interno è possibile usare gli operatori IN, BETWEEN, LIKE;

Il linguaggio SQL standard prevede la clausola CHECK nel comando CREATE TABLE proprio per controllare il valore dei dati immessi in una tabella.

MySQL ammette l'uso di questa clausola, ma solo per compatibilità con lo standard SQL, e tale opzione non ha alcun effetto concreto nel controllare i dati immessi nella tabella.

Trigger

Un trigger è un insieme di comandi che viene eseguito automaticamente a causa di eventi che modificano il database e può essere usato in sostituzione della clausola "Check".

Un trigger è un componente di un database abbinato a una tabella con la seguente sintassi:

- CREATE TRIGGER ON FOR EACH ROW

Esempio:

- CREATE TRIGGER InserimentoEditori BEFORE INSERT ON Editori

FOR EACH ROW BEGIN

IF NEW.nome IS NULL SET NEW.nome = concat('manca il nome!')

END IF;

END;

- **NOT NULL:** Il campo di una colonna non può avere valore "NULL";

- Default: consente di fornire un valore predefinito ad una colonna;

Vincoli di ennupla o tupla (riferiti a più attributi):

- **PRIMARY KEY;**

- **UNIQUE:** (crea un indice su attributi chiave, se non specificato, su attributi non chiave);

- **CHECK:** (riguarda più attributi);

Vincoli di integrità referenziale (riferiti alle associazioni, più tabelle): //FOREIGN KEY ... REFERENCES.

Esempio derivazione logica di un'associazione 1:N (Integrità Referenziale):

CREATE TABLE studenti (matricola varchar (5), siglaClassevarchar (5),nome varchar (50) NOT NULL, cognome varchar (50) NOT NULL, citta varchar(50), PRIMARY KEY (matricola), FOREIGN KEY (siglaClasse) REFERENCES classi);

Opzioni della FOREIGN KEY:

RESTRICT, NO ACTION oppure nessuna opzione (restituisce errore e non consente di proseguire)

- **SET DEFAULT** (non utilizzabile in MySQL);

- **SET NULL** (imposta gli attributi a null);

- **CASCADE** (consente l'operazione e la estende ai collegati);