Basi di Dati

CREAZIONE E POPOLAMENTO DI UNA BASE DI DATI

La finalità di questa esercitazione è quella di creare, date delle specifiche progettuale, appositi script di creazione e popolamento di una base di dati.

1. Istruzioni preliminari

1. Installare il software EASYPHP Dev Server 14.1 VC 11scaricabile dal sito Web http://www.easyphp.org/download.php

2. Lanciare EasyPHP dalla barra di Avvio di Windows

3. Cliccare sulla freccia in basso a destra sulla barra delle applicazioni, poi con il tasto destro sull'icona di EasyPHP

4. Aprire l'interfaccia *Administration* e poi cliccare su "Open" alla voce "MySQL Administration: PhpMyAdmin", sotto la scritta "Modules", per interfacciarsi con il DBMS MySQL

5. Per lanciare uno script SQL dall'interfaccia Web MySQL selezionare dal pannello *Importa* selezionare il selezionare il file e cliccare su *Esegui*

6. Per rilanciare più volte lo script di creazione/popolamento ricordarsi di cancellare eventuali istanze del database create in precedenza dal panello *Database*

2. Generazione degli script di creazione e popolamento del DB

1. Gli script sono semplici file di testo scritti con un qualsiasi editor (es., Notepad, Word, Wordpad)

2. Gli script vanno salvati con estensione *.sql*

3. Gli script contengono una sequenza di istruzioni ciascuna terminata con il simbolo ";"

4. Per interagire con il DBMS MySQL, sono necessarie le seguenti istruzioni preliminari (da scrivere all'inizio del file):
    a. SET storage_engine=InnoDB; (*attivazione dell'Engine InnoDB per la gestione delle basi di dati*)
    b. CREATE DATABASE IF NOT EXISTS NomeDababase; (*creazione del DB denominato NomeDatabase*)
    c. USE NomeDababase; (*impostazione del DB NomeDatabase appena creato come DB corrente*)
5. Per attivare la verifica automatica del vincolo di integrità referenziale è disponibile il comando:
    a. SET FOREIGN_KEY_CHECKS=0; (*disattivato*) oppure 1; (*attivato*)
6. Alle istruzioni preliminari seguono la sequenza di istruzioni in linguaggio SQL per la creazione e il popolamento del DB (CREATE TABLE e INSERT)

7. Ricordarsi si verificare la sintassi e i tipi di dato compatibili con quelli richiesti dal DBMS MySQL.
8. Se non indicato diversamente, MySQL esegue sempre il commit dopo *ogni* istruzione. Per la gestione delle transazioni sono disponibili i seguenti comandi:
    a. SET autocommit=0 (disattivato) / 1 (attivato); (*disattivazione/attivazione del commit automatico ad ogni istruzione*)
    b. START TRANSACTION; (*avvio della transazione*)
    c. COMMIT; (*commit di tutte le operazioni della transazione*)

3. Esercizi

Realizzare la base di dati corrispondente allo schema logico seguente relativo ad un alcune attività di una palestra (le chiavi primarie sono sottolineate e le chiavi esterne sono in corsivo).

Per ogni istruttore è noto il codice fiscale, il nome, il cognome, la data di nascita, l'indirizzo e-mail e il numero di telefono. Per ogni attività è noto il codice, il nome, il tipo (es. attività musicale) e il livello (un numero compreso tra 1 e 4). Il programma delle attività riporta il giorno (es. lunedì, martedì, ecc..) e l'ora di inizio e di fine in cui ogni istruttore svolge una determinata attività. Per ogni attività programmata è noto il numero della sala in cui si svolge.

ISTRUTTORE (CodF, Nome, Cognome, DataN, Email, Tel)

ATTIVITA' (CodA, Nome, Tipo, Livello)

PROGRAMMA (*CodF*, Giorno, OraI, OraF, *CodA*, Sala)

Svolgere le seguenti attività:

- Creare uno script SQL *creaDB.sql* con le istruzioni per la creazione della base di dati corrispondente allo schema logico riportato.

```
SET storage_engine=InnoDB;
SET FOREIGN_KEY_CHECKS=0;

CREATE DATABASE IF NOT EXISTS istruttori;
use istruttori;

-- drop tables list
 DROP TABLE IF EXISTS Programma;
 DROP TABLE IF EXISTS Istruttore;
 DROP TABLE IF EXISTS Attivita;

CREATE TABLE IF NOT EXISTS Istruttore (
   CodFisc VARCHAR(15) UNIQUE NOT NULL ,
   Nome VARCHAR(255) NOT NULL ,
   Cognome VARCHAR(255) NOT NULL ,
   DataN DATE NOT NULL ,
   Titolo VARCHAR(255) NOT NULL ,
   Email VARCHAR(255) NOT NULL ,
   Tel VARCHAR(255) NULL ,
   PRIMARY KEY (CodFisc)
```

```
);

CREATE TABLE IF NOT EXISTS Attivita (
  CodA VARCHAR(255) UNIQUE NOT NULL ,
  TipoA VARCHAR(255) NOT NULL ,
  Nome VARCHAR(255) NULL ,
  Livello ENUM(‘1’, ‘2’, ‘3’, ‘4’),
  PRIMARY KEY (CodA)
);

CREATE TABLE IF NOT EXISTS Programma (
  CodFisc VARCHAR(15) NOT NULL ,
  Giorno VARCHAR(15) NOT NULL ,
  OraI TIME NOT NULL ,
  OraF TIME NOT NULL ,
  Sala ENUM(‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ’6’, ’7’, ’8’, ’9’, ’10’),
  CodA VARCHAR(255) NOT NULL ,
  PRIMARY KEY (CodFisc,Giorno,OraI) ,
  FOREIGN KEY (CodFisc )
        REFERENCES Istruttore(CodFisc )
        ON DELETE CASCADE
         ON UPDATE CASCADE ,
  FOREIGN KEY (CodA )
         REFERENCES Attivita(CodA )
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

- Creare uno script SQL *popolaDB.sql* con le istruzioni per il popolamento della base di dati creata al punto precedente.

```
SET FOREIGN_KEY_CHECKS=1;

-- insert data records

SET autocommit=0;
START TRANSACTION;


INSERT INTO Istruttore (CodFisc,Nome,Cognome,DataN,Titolo,Email,Tel)
      VALUES            ('EEEEEE99E99E999E','Elena','Rossi','1979-09-05','Dottorato di
ricerca','nome.cognome@dominio.it',0110999999);

INSERT INTO Istruttore (CodFisc,Nome,Cognome,DataN,Titolo,Email,Tel)
      VALUES            ('LLLLLL99L99L999L','Luca','Bianchi','1984-10-
30','Laurea','nome.cognome@dominio.it',011099996);

INSERT INTO Istruttore (CodFisc,Nome,Cognome,DataN,Titolo,Email,Tel)
      VALUES            ('GGGGGG99G99G999G','Giulia','Neri','1968-10-04','Dottorato di
ricerca','nome.cognome@dominio.it',0110999998);
```

```
INSERT INTO Istruttore (CodFisc,Nome,Cognome,DataN,Titolo,Email,Tel)
      VALUES            ('99T99T999T','Tania','Bruni','1988-09-
01','Diploma','nome.cognome@dominio.it',0110999997);

INSERT INTO Attivita (CodA,TipoA)
      VALUES ('BBPWRL','Body building');

INSERT INTO Attivita (CodA,TipoA,Nome,Livello)
      VALUES ('MUSTUP','Musicale', 'Tone Up', '2');

INSERT INTO Attivita (CodA,TipoA,Nome,Livello)
      VALUES ('MUSGIN','Musicale', 'Ginnastica dolce', '1');

INSERT INTO Attivita (CodA,TipoA,Nome,Livello)
      VALUES            ('MUSGAG','Musicale', 'GAG', '1');

INSERT INTO Attivita (CodA,TipoA)
      VALUES ('MUSSPI','Cardio Fitness');

INSERT INTO Programma (CodFisc,Giorno,OraI,OraF,Sala,CodA)
      VALUES            ('99T99T999T','Lun','18:00:00','19:00:00','2','MUSGIN');

INSERT INTO Programma (CodFisc,Giorno,OraI,OraF,Sala,CodA)
      VALUES ('99T99T999T','Mar','19:00:00','20:30:00','1','MUSTUP');

INSERT INTO Programma (CodFisc,Giorno,OraI,OraF,Sala,CodA)
      VALUES ('EEEEEE99E99E999E','Mar','16:00:00','17:30:00','1','MUSTUP');

commit;
```

- Testare gli script di creazione e popolamento sul DBMS MySQL

In relazione alle attività precedenti, svolgere i seguenti passi:

- Specificare nello script di creazione del DB eventuali vincoli di dominio e/o di tupla appropriati e verificarne l'applicazione mediante l'interfaccia Web di MYSQL.

I vincoli presi in considerazione riguardano la presenza nella tabella Programma degli attributi CodA, identificatore univoco delle tabella AttivitàMusicale, e CodFisc, identificatore univoco delle tabella Istruttore.

```
FOREIGN KEY (CodFisc )
        REFERENCES Istruttore(CodFisc )
FOREIGN KEY (CodA )
         REFERENCES Attivita(CodA )
```

- Scegliere opportunamente le politiche di gestione dei vincoli più idonee al contesto analizzato.

Nel nostro caso come politica di gestione dei vincoli è stato scelto di usare l'opzione CASCADE, la quale propaga l'eventuale operazione di aggiornamento (ON UPDATE CASCADE) o cancellazione (ON DELETE CASCADE).

- Abilitare l'opzione di verifica automatica del vincolo di integrità referenziale (SET FOREIGN_KEY_CHECKS=1;) e verificare l'effetto su:
    - l'ordine delle istruzioni di creazione e popolamento delle tabelle

Usando il comando SET FOREIGN_KEY_CHECKS=1; la verifica del vincolo di integrità viene eseguita automaticamente ad ogni istruzione, per tale motivo nella scrittura di uno script è necessario creare e popolare prima le tabelle referenzianti e poi quelle referenziate. In caso contrario si genera un errore. Il problema viene risolto disabilitando l'opzione di verifica automatica, (SET FOREIGN_KEY_CHECKS=0;). In questo modo l'ordine di creazione e popolamento delle tabelle nello script non è più rilevante, anche se potrebbe causare l'insorgere di inconsistenza.

    - presenza di eventuali inconsistenze nei dati

Se si usa SET FOREIGN_KEY_CHECKS=0; l'ordine di creazione e popolamento delle tabelle non è più rilevante, ma potrebbero sorgere delle inconsistenze in quanto i controlli dei vincoli di integrità non vengono effettuati automaticamente ad ogni istruzione. Quindi istruzione come la INSERT nella tabella referenziante e DELETE nella tabella referenziata potrebbero generare inconsistenze.

Esempio: "INSERT INTO Programma (CodFisc,Giorno,OraI,OraF,Sala,CodA)..". La insert prevede l'inserimento di un valore nei campi CodFisc e CodA. Se le tabelle Istruttore e Attività non fossero state già create e popolate, i valori inseriti causerebbero problemi in quanto dati inconsistenti.

- Discutere eventuali criticità legati all’uso dell’autocommit nell’esecuzione dello script di popolamento della base di dati.

Utilizzando AUTOCOMMIT=1, ogni volta che una query viene eseguita, viene effettuata anche un'operazione di COMMIT,(ogni istruzione è una transazione distinta). Se invece AUTOCOMMIT=0 viene eseguito un commit unico alla fine (tutte le istruzioni in un unica transazione). Nell'esercitazione quest'ultima opzione è da preferire poiché eventuali problemi riscontrati durante l'esecuzione degli script per la creazione e per il popolamento della base di dati possono lasciare il database in uno stato inconsistente.

.