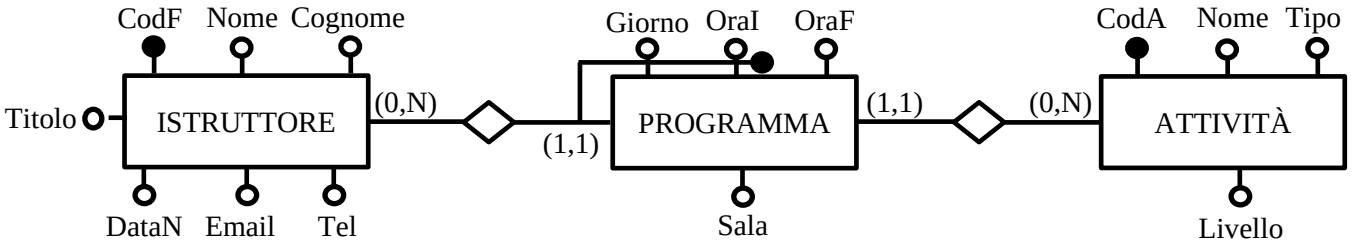
Basi di Dati

Creazione e popolamento di una base di dati - Esercitazione n. 8 - Soluzioni

1. Esercizi

Realizzare una base di dati per la gestione di alcune attività di una palestra.

Per ogni istruttore è noto il codice fiscale, il nome, il cognome, la data di nascita, il titolo di studio, 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.



1. Schema ER

   ISTRUTTORE (CodF, Nome, Cognome, DataN, Titolo, Email, Tel)
   ATTIVITÀ (CodA, Nome, Tipo, Livello)
   PROGRAMMA (*CodF*, Giorno, OraI, OraF, CodA, Sala)

2. Script SQL *creaDB.sql* 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 palestra;
use palestra;

-- 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(16) 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(50) 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(16) 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(50) 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
 );
```

3. Script SQL *popolaDB.sql* 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;
```

4. Testare gli script di creazione e popolamento sul DBMS MySQL.

5. 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 della tabella Attività, e CodFisc, identificatore univoco della tabella Istruttore.

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

6. 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).

7. Abilitare l'opzione di verifica automatica del vincolo di integrità referenziale (SET FOREIGN_KEY_CHECKS=1;) e verificare l'effetto su:

   a. 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 referenziate e poi quelle referenzianti. 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 inconsistenze.

b. 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.

8. 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.