Soluzione Esercitazione 6

Soluzioni Esercitazione 6

Esercizio 1

```
CREATE TABLE Sala (NomeSala VARCHAR(15) PRIMARY KEY,
                   Piano INTEGER NOT NULL,
                   Capienza INTEGER NOT NULL,
                   TelefonoSala VARCHAR(14) UNIQUE,
                   Videoproiettore BOOL DEFAULT FALSE NOT NULL)
```

I numeri di telefono sono tipicamente univoci all'interno dello stesso edificio. Il Videoproiettore si può ipotizzare che sia un elemento aggiuntivo e accessorio delle sale, per cui possiamo pensare di metterlo falso di default. Con il not null garantiamo che non venga forzato l'inserimento di NULL nel videoproiettore.

```
CREATE TABLE Evento (IDEvento INTEGER PRIMARY KEY,
                     NomeEvento VARCHAR(30) NOT NULL,
                     Descrizione VARCHAR(255),
                     DataInizio DATE NOT NULL,
                     DataFine DATE NOT NULL
                     CHECK(DataFine >= DataInizio),
                     NomeSala VARCHAR(15) NOT NULL REFERENCES
                        Sala.NomeSala
                        ON UPDATE CASCADE
                        ON DELETE NO ACTION);

CREATE TABLE Organizza (IDEvento INTEGER REFERENCES
                         Evento.IDEvento
                         ON UPDATE CASCADE
                         ON DELETE CASCADE,
                        CFOrganizzatore CHAR(16) REFERENCES
                        Organizzatore.CFOrganizzatore
                         ON UPDATE CASCADE ON DELETE NO ACTION,
                        PRIMARY KEY (IDEvento, CFOrganizzatore));
```

Esercizio 2

Specificare in SQL la creazione delle tabelle Corso e Frequenza, definendo i vincoli di tupla e di dominio ritenuti opportuni ed esprimendo eventuali vincoli di integrità referenziale relativi a tutte le tabelle dello schema.

```
CREATE TABLE Corso(CodCorso VARCHAR(10) PRIMARY KEY,
                   Descrizione VARCHAR(255) NOT NULL,
                   Tipo ENUM("Monoset.", "Biset.") NOT NULL
                        DEFAULT "Monoset.",
                   DataInizio DATE NOT NULL,
                   DataFine DATE NOT NULL)
```

```
CREATE TABLE Frequenza(CodiceCorso CHAR(5),
                       Vasca VARCHAR(5),
                       Giorno DATE,
                       OraInizio TIME CHECK(OraInizio
                                BETWEEN 8:00 AND 20:00),
                       Cliente CHAR(16) REFERENCES
                         PERSONA(CF) ON UPDATE CASCADE
                                     ON DELETE CASCADE,
                       DataDiIscrizione DATE NOT NULL,
                       PRIMARY KEY (CodiceCorso, Vasca,
                         Giorno, OraInizio, Cliente),
                       FOREIGN KEY (CodiceCorso, Vasca,
                                    Giorno, OraInizio)
                       REFERENCES
                       CALENDARIO(CodiceCorso, Vasca,
                                  Giorno, OraInizio)
                       ON UPDATE CASCADE ON DELETE NO ACTION)
```

Esercizio 3

Specificare in SQL il vincolo che controlla che la data di iscrizione non sia posteriore alla data di fine del corso.

```
CREATE ASSERTION VerificaDate CHECK (
    NOT EXISTS (SELECT *
                FROM FREQUENZA AS F, CORSO AS C
                WHERE F.CodiceCorso = C.CodCorso
                AND F.DataIscrizione > C.DataFine))
```

Esercizio 4

Specificare in SQL il vincolo che controlla che in uno stesso giorno non ci siano più di 50 persone che frequentano lezioni.

```
CREATE ASSERTION VerificaNumeroPresenti CHECK (
    50>=ALL (SELECT COUNT(DISTINCT Cliente)
             FROM FREQUENZA
             GROUP BY Giorno))
```

Si potrebbe anche esprimere con il non esiste:

```
CREATE ASSERTION VerificaNumeroPresenti CHECK (
    NOT EXISTS (SELECT Giorno
             FROM FREQUENZA
             GROUP BY Giorno
             HAVING COUNT(DISTINCT Cliente) > 50))
```

Esercizio 5

Specificare in SQL i comandi di creazione delle tabelle FARMACIA ed APERTURA, definendo i vincoli di tupla e di dominio secondo le indicazioni specificate più sotto ed esprimendo gli eventuali vincoli di integrità referenziale verso le altre tabelle.

```
CREATE TABLE Farmacia(NumeroFarmacia INTEGER,
                      Citta VARCHAR(32),
                      Nome VARCHAR(255) NOT NULL,
                      Indirizzo VARCHAR(64) NOT NULL,
                      Telefono CHAR(10) NOT NULL,
                      Titolare CHAR(6) NOT NULL REFERENCES
                            TITOLARE(Codice) ON UPDATE CASCADE
                                             ON DELETE NO ACTION,
                      DataFondazione DATE NOT NULL,
                      PRIMARY KEY(NumeroFarmacia, Citta))

CREATE TABLE APERTURA(NumeroFarmacia INTEGER,
                      Citta VARCHAR(32),
                      Data DATE,
                      PRIMARY KEY(NumeroFarmacia, Citta, Data),
                      FOREIGN KEY(NumeroFarmacia, Citta) REFERENCES
                      FARMACIA(NumeroFarmacia, Citta)
                        ON UPDATE CASCADE
                        ON DELETE NO ACTION)
```

Esercizio 6

Specificare in SQL il vincolo che controlla che a Roma ci sia stata almeno una farmacia aperta in ogni giorno dell'anno 2018.

```
CREATE ASSERTION ApertureRoma CHECK(
    365 = (SELECT COUNT(DISTINCT Data)
           FROM APERTURA
           WHERE Citta = "Roma"
           AND Data BETWEEN "1/1/2018" AND "31/12/2018"))
```

Esercizio 7

Esprimere il vincolo che verifica che non ci siano contratti di affitto relativi a palazzi non ancora agibili

```
CREATE ASSERTION CheckAgibile CHECK(
    NOT EXISTS(
        SELECT *
        FROM Affitto
        NATURAL JOIN (Palazzo NATURAL JOIN Appartamento)
        WHERE DataInizio < DataAgibilita)
    )
```

Esercizio 8

Si esprima il vincolo che verifica che non ci siano affitti con sovrapposizioni.

```
CREATE ASSERTION Controllo_Sovrapposizioni CHECK(
    NOT EXISTS (
        SELECT *
        FROM AFFITTO AS A1
```

```
        JOIN AFFITTO AS A2 ON A1.CodAppar=A2.CodAppar
        WHERE (A1.DataInizio
            BETWEEN A2.DataInizio AND A2.DataFine)
            AND (A1.DataInizio <> A2.DataInizio
            OR A1.CodiceCliente <> A2.CodiceCliente)))
```

Le ultime condizione nell'OR servono per considerare l'esclusione della tupla rappresentante lo stesso affitto.

Esercizio 9

Specificare in SQL la creazione delle tabelle TWEET e SEGUE, definendo i vincoli di tupla e di dominio ritenuti opportuni ed esprimendo eventuali vincoli di integrità referenziale relativi a tutte le tabelle dello schema.

```
CREATE TABLE Tweet(
    TweetId INTEGER PRIMARY KEY,
    Testo VARCHAR(255) NOT NULL,
    Utente INTEGER NOT NULL
        REFERENCES Utente(IdUtente)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    DataPubblicazione DATE NOT NULL,
    NazionePubblicazione VARCHAR(50) NOT NULL,
    Retweet INTEGER DEFAULT NULL
    REFERENCES Tweet(TweetId)
        ON UPDATE CASCADE
        ON DELETE SET NULL)

CREATE TABLE Segue(
    Utente INTEGER NOT NULL
        REFERENCES Utente(IdUtente)
        ON UPDATE CASCADE ON DELETE CASCADE,
    UtenteSeguito NOT NULL
        REFERENCES Utente(IdUtente)
        ON UPDATE CASCADE ON DELETE CASCADE,
    DataInizio DATE NOT NULL,
    PRIMARY KEY (Utente, UtenteSeguito))
```

Esercizio 10

Specificare in SQL il vincolo che controlla che nessun tweet contenuto nella base di dati abbia data di pubblicazione precedente alla data di iscrizione dell'utente che lo ha pubblicato.

```
CREATE ASSERTION VerificaDate CHECK(
    NOT EXISTS(
        SELECT *
        FROM Tweet AS T, Utente AS U
        WHERE T.Utente=U.IdUtente
        AND T.DataPubblicazione<U.DataIscrizione))
```

Esercizio 11

Considerando lo schema di base di dati, specificare in SQL il vincolo che controlla che la data della riparazione sia compresa in un intervallo in cui il meccanico che la ha eseguita era effettivamente assunto dall'autofficina.

```
CREATE ASSERTION VerificaDate CHECK (
    NOT EXISTS (
        SELECT *
        FROM Riparazione AS R
        WHERE NOT EXISTS (
            SELECT *
            FROM Assunzioni AS A
            WHERE R.Meccanico=A.CodiceMeccanico
            AND R.Data>=A.DataAssunzione
            AND (R.DataLicenziamento IS NULL
                OR R.Data<=A.DataLicenziamento))))
```