ESAME_01_2023

Table of contents

Esercizi SQL

Sia dato il seguente schema logico relazionale:

CorsiDiStudio(codice, nome, cognome)
Insegnamenti(codice, nome, cfu)
Studenti(matricola, cognome, nome, data_nascita)
Esami(studente_{fk}, cds_{fk}, insegnamento_{fk}, data, voto, lode)
Laureati(cdfs_{fk}, studente_{fk}, data, voto_finale)

Implementare le seguenti interrogazioni (in linguaggio SQL, se non altrimenti specificato):

1. Scrivere l'istruzione DDL per la creazione della tabella **Esami**: il voto deve essere compreso tra 18 e 30; la lode (un valore booleano) puo' essere attribuita solo se il voto e' 30.

```
CREATE TABLE esami (
   studente VARCHAR(10) NOT NULL,
   cds VARCHAR(10) NOT NULL REFERENCES corsi_di_studio(codice),
   insegnamento VARCHAR(10) NOT NULL REFERENCES insegnamenti(codice),
   data DATE NOT NULL,
   voto NUMERIC(2,0) NOT NULL,
   lode BOOLEAN NOT NULL,
   PRIMARY KEY(studente, cds, insegnamento),
   FOREIGN KEY (studente) REFERENCES studenti(matricola),
   CHECK(voto ⩾ 18 AND voto ⩽ 30)
)
```

2. Elencare i dati anagrafici degli studenti che si sono laureati senza avere mai registrato (negli esami per quel corso di laurea) un voto superiore a 27.

```
SELECT S.*, LAU.cds
FROM studenti S, laureati LAU
WHERE S.matricola = LAU.studente
   AND NOT EXISTS ( SELECT *
                    FROM esami E
                    WHERE E.studente = S.matricola
                            AND E.cds = LAU.cds
                            AND E.voto > 27
)
```

3. Eliminare gli esami svolti precedentemente all'anno 2000 da studenti che non si sono mai laureati.

```
DELETE FROM esami E
WHERE E.studente NOT IN ( SELECT studente FROM laureati )
```

```
    -- AND E.data < '2000-01-01' → alternativa
    AND EXTRACT( year FROM E.data ) < 2000
```

4. Fornire l'elenco degli studenti che si sono laureati per piu' di un corso di laurea.

```
SELECT studente
FROM laureati
GROUP BY studente
HAVING COUNT(*) > 1
```

```
SELECT L1.studente
FROM laureati L1, laureati L2
WHERE L1.studente - L2.studente
   AND L1.cds < L2.cds
```

5. Per ogni studente laureato del corso di laurea di codice '`LTINF`', calcolare la media ponderata dei voti ottenuti negli esami di profitto (senza considerare eventuali esami sostenuti per altri corsi di laurea).

```
SELECT E.studente, SUM(E.voto * I.cfu) / SUM(I.cfu) AS media_ponderata
FROM esami E, insegnamenti I, laureati LAU
WHERE E.studente = LAU.studente
   AND E.insegnamento I.codice
   AND LAU.cds = 'LTINF'
GROUP BY E.studente
```

Teoria

1. Cosa si intende per superchiave di una relazione $R(X)$ nel modello logico relazionale? Cosa si intende per chiave?

   Si dice *superchiave* di $R(X)$ un sottoinsieme degli attributi X, identificante univocamente una n-upla di R. Una *chiave* di $R(X)$ e' una superchiave che tuttavia non e' ridondante (minimale).
   e.g.: **Insegnamenti**(codice) → e' superchiave e chiave
   e.g.: **Insegnamenti**(codice, nome) → e' superchiave

2. Spiegare la differenza tra la clausola `WHERE` e la clausola `HAVING`.

   Entrambe le clausole implementano una selezione, scartando alcune righe senza farle vedere nel risultato: `HAVING` per le aggregazioni (`HAVING COUNT(), HAVING SUM()`, ...); `WHERE` per il resto

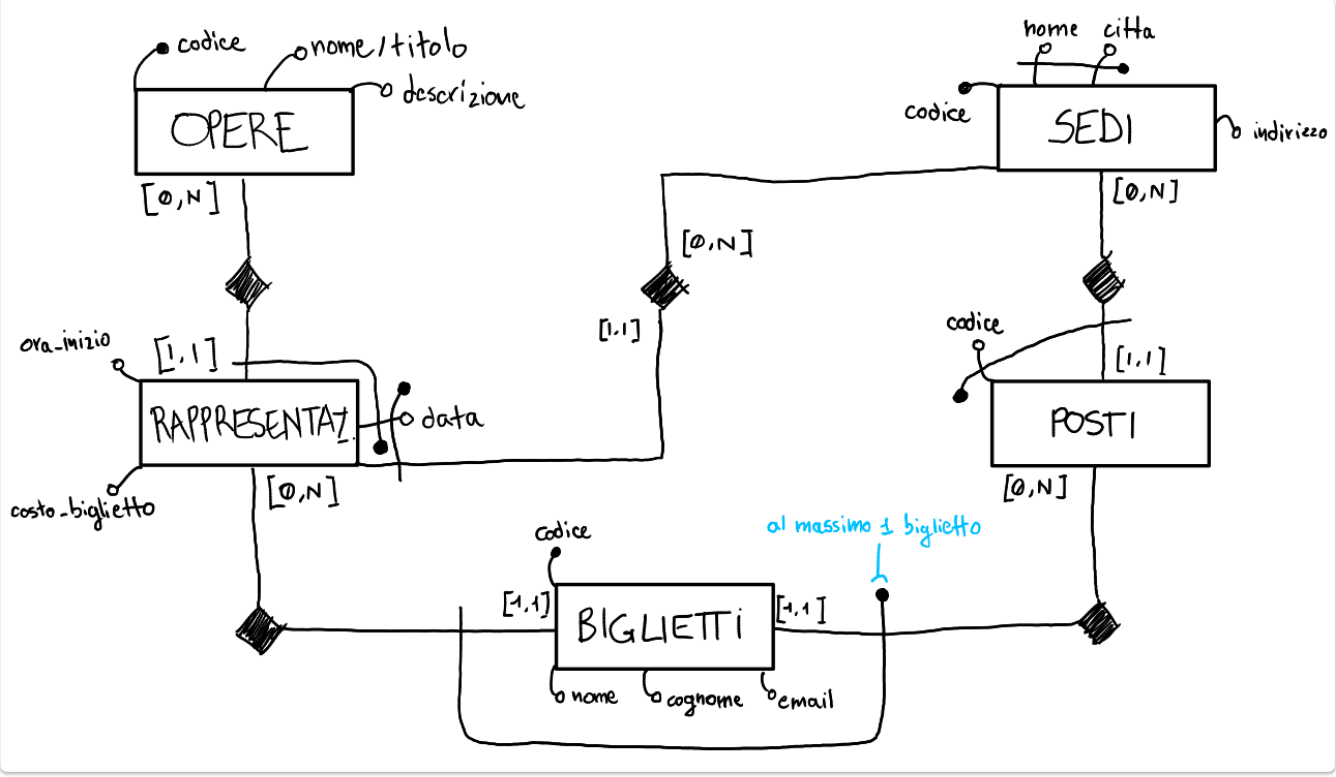
Esercizi ER

Mostrare lo schema concettuale ER per un database che codifica informazioni relative alla vendita di biglietti per opere artistiche (spettacoli teatrali, concerti, ecc.). Si richiede di modellare le informazioni seguenti.

1. Le *opere* artistiche sono identificate da un codice; per esse si tiene traccia del titolo (non necessariamente univoco) e di una descrizione testuale.
2. Le *sedi* di svolgimento delle rappresentazioni delle opere sono identificate da un codice; per esse si registrano il nome, la citta' e l'indirizzo; possono esistere sedi con lo stesso nome, ma solo se collocate in citta' distinte.
3. Ogni sede ha associato un certo numero di *posti* disponibili, identificati (all'interno di quella sede) da un codice.
4. Ogni opera artistica puo' essere rappresentata piu' volte. Ogni *rappresentazione* dell'opera viene identificata dalla sede e dalla data. Per essa si tiene traccia dell'opera, dell'ora di inizio e del costo del

biglietto (per semplicita', questo non dipende dal posto scelto). Non vi possono essere piu' rappresentazioni della stessa opera nella stessa data.

5. Per ogni rappresentazione si tiene traccia dei *bieglietti* venduti; ogni biglietto, identificato da un codice univoco, e' associato a uno e un solo posto numerato di quelli disponibili nella sede di svolgimento della rappresentazione; si tiene traccia di nome, cognome ed email dell'acquirente del biglietto.



Traduzione ER

Tradurre lo schema ER in uno schema logico relazionale, codificando oppurtamanete i vincoli dello schema.

`Opere`(codice, nome, descrizione)
`Sedi`(codice, [nome, citta]$_{UNIQUE}$, indirizzo)
`Posti`(codice, sede$_{fk}$)
`Rappresentazioni`(data, sede$_{fk}$, opera$_{fk}$, costo_biglietto, ora_inizio)

`Biglietti`
(codice, [data_rappr, sede_rappr]$_{fk(Rappresentazioni)}$, [codice_posto, sede_posto]$_{fk(Posti)}$, nome, cognome, email)

- CHECK(sede_rappr = sede_posto) --- un posto comprato a Milano non porta a Roma
- UNIQUE(data_rappr, sede, codice_posto) --- al massimo un biglietto

`Biglietti`
(codice, [data_rappr, sede]$_{fk(Rappresentazioni)}$, [codice_posto, sede]$_{fk(Posti)}$, nome, cognome, email)

- UNIQUE (data_rappr, sede, codice_posto) --- al massimo un biglietto

Esercizi teoria

Rispondere brevemente alle seguenti domande.

1. Quali fattori possono influenzare l'ordine di esecuzione di due trigger definiti per lo stesso evento sulla stessa tabella target?

Influenzanti il fattore di esecuzione sono: `BEFORE` o `AFTER` clausole del trigger, granularita' `FOR EACH ROW` o `FOR EACH STATEMENT` e stato `DEFERRED` o `NOT DEFERRED`. Inoltre da aggiungere che in PgSQL il nome per ordine alfabetico influenza l'esecuzione.

2. Spiegare cosa sono e come vengono calcolati l'undo-set e il redo-set durante la fase di rispristino da un fallimento di sistema.

   Per la regola di *atomicita'*, devono essere assicurate le `commit` delle transazioni; nel caso di fallimenti del sistema:

   - redo-set, inserite le transazioni che hanno fatto `commit` in tempo, prima del fallimento (e' presente un record di log dall'ultimo checkpoint, in avanti);
   - undo-set, inserite le transazioni che hanno scritto sul disco, che sono partite ma che non hanno `commit` nel file di log.