ESAME_06_2021

Esercizi SQL

Dato il seguente schema logico relazionale

MEDICI (codice, cognome, nome)
PAZIENTI (codice_SNN, cognome, nome, data_nascita)
MEDICINALI (codice, nome, principio_attivo, marca)
VISITE (medico$_{fk}$, paziente$_{fk}$, data, diagnosi, medicinale*)

rispondere alle seguenti domande (implementando le interrogazioni in linguaggio SQL).

1. Elencare i medici che sono omonimi (stesso cognome e stesso nome) di almeno un paziente.

```sql
SELECT DISTINCT m.*
FROM medici.m, pazienti p
WHERE m.cognome = p.cognome
AND n.nome = p.nome;
```

2. Elencare i pazienti visitati almeno una volta e ai quali non sono mai stati prescritti medicinali.

```sql
SELECT p.*
FROM pazienti p
WHERE p.codice IN (SELECT paziente FROM visite)
   AND p.codice NOT IN (SELECT paziente FROM visite
                  WHERE medicinale IS NOT NULL);
```

3. Eliminare dal database le visite effettuate a pazienti nati prima dell'anno 1950.

```sql
DELETE FROM visiste
WHERE paziente IN (SELECT codice FROM visite
           WHERE data_nascita < '1950-01-01');
```

4. Elencare, in ordine alfabetico, i medici che nel 2020 hanno prescritto medicinali di una e una sola marca.

```sql
SELECT m.*
FROM medici m, visite v, medicinali md
WHERE m.codice = v.medico
   AND v.medicinale = md.codice
   AND v.data BETWEEN '2020-01-01' AND '2020-12-31'
 --AND v.data BETWEEN ⩾ '2020-01-01' AND v.data ⩽ '2020-12-31'
 --AND EXTRACT (year FROM v.data) = 2020
GROUP BY m.codice, m.cognome, m.nome
HAVING COUNT (DISTINCT md.marca) = 1
ORDER BY m.cognome, m.nome
```

5. Sapendo che le relazioni **MEDICI**, **PAZIENTI** e **VISITE** hanno cardinalita' $m, p, v \in \mathbb{N}$, indicare (motivando la risposta) le cardinalita' minime e massime delle seguenti interrogazioni:

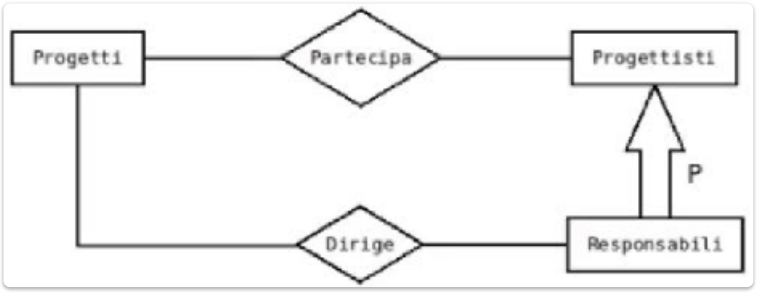
    1. `SELECT p.cognome, p.nome, v.data`
       `FROM pazienti p, visite v WHERE v.paziente = p.codice_SSN`

       min = v, max = v

2. `SELECT m.cognome`
   `FROM medici m, pazienti p WHERE m.nome = p.nome`

   min = 0, max = $m * p$

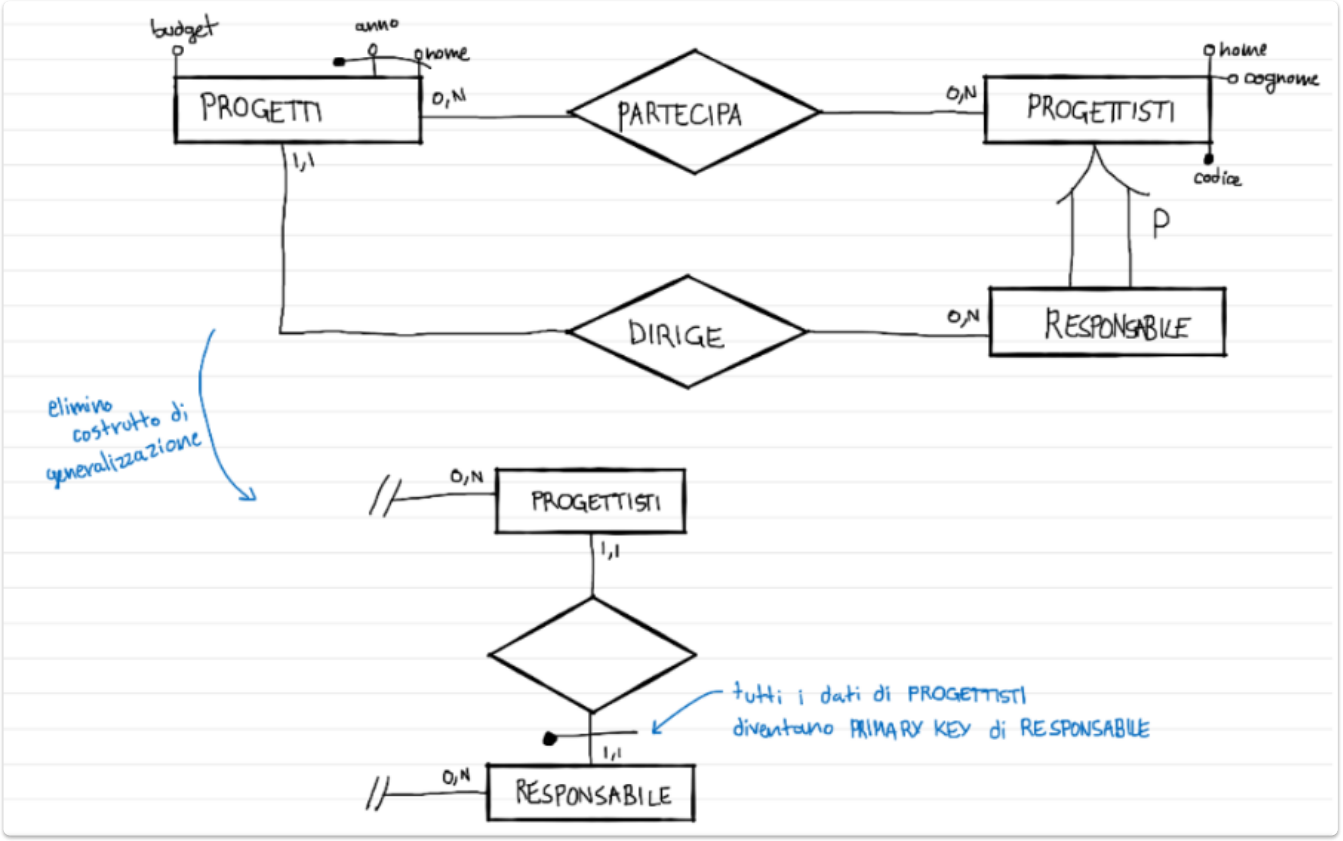
Esercizi ER



Completare la bozza di schema ER in figura (indicando gli attributi, gli identificatori e le cardinalita' mancanti) sapendo che:

- i progetti sono identificati da un anno di inizio e nome (piu' progetti iniziati in anni diversi possono avere lo stesso nome); per essi si tiene traccia del budget e sono diretti da un unico responsabile di progetto;
- per i progettisti, identificati da un codice, si conoscono nome e cognome; possono partecipare a piu' progetti e, se responsabili di progetto, dirigere piu' progetti;
- ad ogni progetto puo' partecipare un numero qualunque di progettisti.

Ristrutturare lo schema, eliminando il costrutto di generalizzazione in modo da mantenere distinte le entita' progettisti e responsabili di progetto.



Traduzione ER in logico relazionale

`PROGETTISTI` (codice, cognome, nome)
`RESP_PROJ` (progettisti_{fk})

`PROGETTI` (anno, nome, budget, resp_proj_{fk})
`PARTECIPA` ($[\text{anno, nome}]_{fk}$, progettista_{fk})

`NOTE` : in un'associazione *molti-a-molti*, deve essere codificata l'associazione

Esercizi teoria

1. Definire il concetto di transazione. Cosa si intende per atomicita' delle transazioni?

   Per transazione s'intende una sequenza di `read` e `write`, operazioni del `DBMS`, terminate da `commit` o `abort` (con roll-back). Per atomicita' s'intende la proprieta' delle sequenze `read` e `write` di essere un tutt'uno, se fallisce una, allora l'insieme fallisce.

2. Definire il concetto di conflitto tra due istruzioni in uno schedule. Uno schedule seriale puo' avere conflitti?

   Un conflitto crea, in un grafo, un ciclo non risolvibile `T1 → T2 → T3 → T1`. Il fatto che esista un conflitto non implica che lo schedule non sia serializzabile in base ai conflitti.

3. Spiegare la differenza tra fallimento di sistema e disastro. Spiegare l'uso dei record di check-point nei file di log nella fase di ripristino successiva a: (a) un fallimento di sistema; (b) un disastro.

   Fallimento di sistema → perdita della memoria volatile (RAM), riavvio DBMS.
   Disastro → perdita di memoria nel disco rigido, danno fisico.
   Nel caso del fallimento di sistema, fintanto che le transazioni hanno `commit` in fondo allo stack di esecuzione, allora queste saranno su disco fisso e non ci sara' problema.
   Per il disastro non ci sono molte soluzioni, il danno e' fatto (dump prevengono perdite di dati).