

ESERCIZI IN PREPARAZIONE ALL'ESAME

ESERCIZIO 1

Si consideri il seguente schema relazionale relativo ad una porzione di un ipotetico sistema di gestione di una biblioteca:

- *scrittore*(nome, sesso, nazione)
 - *libro*(ISBN, titolo, autore, genere), dove l'attributo *autore* (resp. *genere*) e' chiave esterna sulla relazione *scrittore* (resp. *generi*)
 - *socio*(id_socio, nome, sesso, eta)
 - *ha_letto*(ISBN, *socio*), dove l'attributo *socio* (resp. ISBN) e' chiave esterna su *socio* (resp. *libro*).
 - *generi*(nome, sala)
- (1) Si definisca uno script SQL per la generazione di uno schema *biblioteca* che implementa lo schema relazionale proposto. Tale script dovra' essere composto da 2 parti principali: La prima, per cancellare le tabelle omonime eventualmente presenti nella base di dati, la seconda per generare lo schema definendo vincoli opportuni ed in particolare:
- Non si possono cancellare le informazioni su uno scrittore, se e' presente un libro dello stesso.
 - Se si cancella/aggiorna un libro, le consultazioni ad esso relative vanno cancellate/aggiornate in cascata
 - Se un socio della sala di lettura viene aggiornato/cancellato, nelle consultazioni ad esso relative l'attributo *socio* viene aggiornato/cancellato in cascata

```
DROP TABLE IF EXISTS scrittore CASCADE;
DROP TABLE IF EXISTS libro CASCADE;
DROP TABLE IF EXISTS socio CASCADE;
DROP TABLE IF EXISTS ha_letto CASCADE;
DROP TABLE IF EXISTS generi CASCADE;
```

```
CREATE TABLE generi (
nome VARCHAR PRIMARY KEY,
sala CHAR);
```

```
CREATE TABLE scrittore (
nome VARCHAR PRIMARY KEY,
sesso CHAR CHECK(sesso='F' OR sesso='M' OR sesso IS NULL),
nazionalita VARCHAR);
```

```
CREATE TABLE libro ( ISBN VARCHAR PRIMARY KEY,
titolo VARCHAR NOT NULL,
```

```
autore VARCHAR REFERENCES scrittore ON DELETE RESTRICT,
genere VARCHAR REFERENCES generi ON DELETE CASCADE);
```

```
CREATE TABLE socio (
id_socio VARCHAR PRIMARY KEY,
nome VARCHAR,
sesso CHAR CHECK(sesso='F' OR sesso='M' OR sesso IS NULL),
eta INTEGER);
```

```
CREATE TABLE ha_letto (
socio VARCHAR REFERENCES socio ON DELETE CASCADE ON UPDATE CASCADE,
ISBN VARCHAR REFERENCES libro ON DELETE CASCADE ON UPDATE CASCADE,
PRIMARY KEY (socio,ISBN));
```

- (2) Si popoli opportunamente lo schema generato utilizzando i dati nei file di testo allegati all'esercizio.

- (3) Si definiscano in SQL le seguenti interrogazioni:

- (a) Ottenere i nomi dei soci di sesso femminile che hanno letto qualche libro

```
SELECT DISTINCT nome
FROM biblioteca.socio JOIN biblioteca.ha_letto ON id_socio=socio
WHERE sesso='F';
```

- (b) Determinare i titoli dei libri nella sala A

```
SELECT titolo
FROM biblioteca.libro JOIN biblioteca.genere ON genere=nome
WHERE sala='A';
```

- (c) Ottenere i titoli dei libri e la sala in cui sono collocati

```
SELECT L.titolo, G.sala
FROM biblioteca.libro L JOIN biblioteca.genere G ON L.genere = G.nome;
```

- (d) Ottenere i titoli dei libri e la sala in cui sono collocati, includendo le opere di cui non e' possibile reperire la collocazione

```
SELECT L.titolo, G.sala
FROM biblioteca.libro L LEFT JOIN biblioteca.genere G ON L.genere = G.nome;
```

- (e) Identificare i soci (i.e. elencarne gli *id*) della biblioteca che hanno letto almeno un libro nella sala A

```
SELECT DISTINCT H.socio
FROM (biblioteca.ha_letto H NATURAL JOIN biblioteca.libro L ) JOIN
biblioteca.genere G ON L.genere = G.nome
WHERE G.sala = 'A';
```

- (f) Determinare i nomi delle coppie di soci della biblioteca che hanno letto uno stesso libro

```
SELECT S1.nome,S2.nome FROM biblioteca.socio S1, biblioteca.socio S2
WHERE (S1.id_socio,S2.id_socio) IN (SELECT H1.socio,H2.socio
FROM biblioteca.ha_letto H1
JOIN biblioteca.ha_letto H2 ON H1.ISBN=H2.ISN
WHERE H1.socio<H2.socio);
```

- (g) Elencare i nomi dei soci della biblioteca che hanno letto almeno un libro nella sala A
- ```
SELECT S.nome FROM biblioteca.socio S
WHERE S.id_socio IN (SELECT H.id_socio FROM biblioteca.ha_letto H
 WHERE H.ISBN IN (SELECT L.ISBN
 FROM biblioteca.libro L, biblioteca.genere G
 WHERE L.genere=G.nome AND G.sala='A'));
```
- (h) Determinare gli autori dei libri letti da almeno una donna
- ```
SELECT L.autore FROM biblioteca.libro L
WHERE L.ISBN IN (SELECT H.ISBN FROM biblioteca.ha_letto H
                 JOIN biblioteca.socio S ON S.id_socio=H.socio
                 WHERE S.sesso='F');
```
- (i) Determinare i soci della biblioteca che non hanno mai letto un libro situato nella sala B
- ```
SELECT S.id_socio FROM biblioteca.socio S
WHERE S.id_socio IN (SELECT H.socio FROM biblioteca.ha_letto H
 WHERE H.ISBN IN (SELECT L.ISBN
 FROM biblioteca.libro L, biblioteca.genere G
 WHERE G.sala=' B' AND L.genere=G.nome));
```
- (j) Determinare i soci della biblioteca che non hanno mai letto un libro situato nella sala B, ma hanno letto qualche libro
- ```
SELECT S.id_socio FROM biblioteca.socio S
WHERE S.id_socio IN (SELECT H.socio FROM biblioteca.ha_letto H
                     WHERE H.ISBN IN (SELECT L.ISBN
                                       FROM biblioteca.libro L, biblioteca.genere G
                                       WHERE G.sala=' B' AND L.genere=G.nome))

INTERSECT
SELECT H.socio FROM biblioteca.ha_letto H;
```
- (k) Determinare i titoli dei libri che si trovano nella stessa sala del libro intitolato *Ossi di Seppia*
- ```
SELECT titolo
FROM biblioteca.libro L1 JOIN biblioteca.genere G1 ON L1.genere =
G1.nome
WHERE G1.sala IN (SELECT sala
 FROM biblioteca.genere G2
 WHERE G2.nome IN (SELECT L2.genere
 FROM biblioteca.libro L2
 WHERE L2.titolo = ' Ossi di Seppia'));
```
- (l) Ottenere gli autori dei libri il cui titolo precede, nell'ordinamento alfabetico, il titolo del libro con ISBN 88-55-55555-5
- ```
SELECT L.autore FROM biblioteca.libro L
WHERE L.titolo < ( SELECT titolo FROM biblioteca.libro
                  WHERE ISBN='88-55-55555-5');
```
- (m) Modificare l'interrogazione precedente in modo da ottenere solo gli autori di libri letti da qualche socio
- ```
SELECT L.autore FROM biblioteca.libro L
WHERE L.titolo < (SELECT titolo FROM biblioteca.libro
 WHERE ISBN='88-55-55555-5')
```

- ```

INTERSECT
SELECT L.autore FROM biblioteca.libro L
WHERE L.ISBN IN (SELECT ISBN FROM biblioteca.ha_letto);

```
- (n) Identificare i soci che hanno letto libri il cui ISBN e' inferiore all'ISBN di qualcuno dei libri nella sala B
- ```

SELECT S.nome FROM biblioteca.socio S
WHERE S.id_socio= ANY (SELECT H.socio FROM biblioteca.ha_letto H
 WHERE H.ISBN < ANY (SELECT L.ISBN
 FROM biblioteca.libro L
 JOIN biblioteca.genere G
 ON L.genere=G.nome
 WHERE G.sala='B'));

```
- (o) Ottenere i titoli e gli autori dei libri il cui ISBN sia superiore a qualcuno degli ISBN dei libri letti da Clotilde Bianchi
- ```

SELECT L.titolo, L.autore FROM biblioteca.libro L
WHERE L.ISBN > ANY (SELECT H.ISBN FROM biblioteca.ha_letto H
                    WHERE H.socio=(SELECT id_socio FROM biblioteca.socio
                                   WHERE nome='Clotilde Bianchi'));

```
- (p) Ottenere le coppie di soci che hanno letto gli stessi libri
- ```

SELECT S1.nome,S2.nome FROM biblioteca.socio S1, biblioteca.socio
S2
WHERE S1.id_socio < S2.id_socio /* Elimina ridondanze */
/* Non esiste alcun libro letto dal primo che non si stato letto dal sec-
ondo */
AND NOT EXISTS (SELECT H.ISBN
 FROM biblioteca.ha_letto H
 WHERE(H.socio=S1.id_socio
 AND H.ISBN NOT IN (SELECT H1.ISBN
 FROM biblioteca.ha_letto H1
 WHERE H1.socio=S2.id_socio))
 OR(H.socio=S2.id_socio
 AND H.ISBN NOT IN (SELECT H1.ISBN
 FROM biblioteca.ha_letto H1
 WHERE H1.socio=S1.id_socio)));

```
- (q) Ottenere i nomi dei soci che hanno letto tutti i libri di poesia
- ```

SELECT S.nome FROM biblioteca.socio S
WHERE NOT EXISTS (SELECT * FROM biblioteca.libro L
                  WHERE L.genere='poesia'
                  AND NOT EXISTS (SELECT * FROM biblioteca.ha_letto H
                                  WHERE L.ISBN=H.ISBN AND S.id_socio=H.socio));

```
- (r) Ottenere i titoli dei libri letti da piu' di un lettore
- ```

SELECT L.titolo FROM biblioteca.libro L
WHERE 1 < (SELECT COUNT(*) FROM biblioteca.ha_letto H
 WHERE H.ISBN=L.ISBN);

```
- (s) Ottenere, per ciascun socio che abbia letto qualche libro, il numero di libri letti
- ```

SELECT socio, COUNT(*)

```

- FROM biblioteca.ha_letto
GROUP BY socio;
- (t) Ottenere, per ciascun socio, il numero di libri letti da tale socio (incluso i soci che non hanno letto alcun libro)
SELECT S.id_socio, COUNT(DISTINCT H.ISBN)
FROM biblioteca.socio S LEFT OUTER JOIN biblioteca.ha_letto H ON S.id_socio=H.socio
GROUP BY S.id_socio;
- (u) Identificare gli autori di cui e' presente il maggior numero di libri
SELECT autore FROM biblioteca.libro
GROUP BY autore
HAVING COUNT(*) >= ALL (SELECT COUNT(*) FROM biblioteca.libro
GROUP BY autore);
- (v) Ottenere i nomi dei soci ed il numero dei libri letti dai soci che hanno letto meno libri di Clotilde Bianchi
SELECT S.nome, COUNT(*)
FROM biblioteca.socio S JOIN biblioteca.ha_letto H ON S.id_socio=H.socio
GROUP BY S.nome
HAVING COUNT(*) < ANY (SELECT COUNT(*)
FROM biblioteca.ha_letto H1
JOIN biblioteca.socio S1 ON S1.id_socio=H1.socio
WHERE S1.nome='Clotilde Bianchi');
- (w) Determinare l'autore preferito dai soci di sesso femminile (i.e. i cui libri sono stati maggiormente letti)
CREATE VIEW preferenzeF AS
SELECT L.ISBN,COUNT(*) AS n
FROM libro L, ha_letto H, socio S
WHERE L.ISBN=H.ISBN AND H.socio=S.id_socio AND S.sesso='F'
GROUP BY L.ISBN;
- SELECT autore
FROM libro,preferenzeF
WHERE preferenzeF.ISBN=libro.ISBN
GROUP BY autore
HAVING SUM(n) >= ALL (SELECT SUM(n)
FROM libro, preferenzeF
WHERE preferenzeF.ISBN=libro.ISBN
GROUP BY autore);
- (4) Si supponga di voler mantenere un'archivio storico dei soci della biblioteca, e di aver creato a tale proposito un'opportuna tabella nella base di dati con il seguente schema:

ex_socio(nome, data_disdetta)

Si defisca un trigger *archivia_socio*, per memorizzare nella tabella *ex_socio* i clienti della biblioteca che disdettano il loro abbonamento (Si noti che la data di disdetta puo' essere ottenuta utilizzando la funzione di sistema *current_date*, che restituisce la data corrente).

Soluzione:

```
CREATE FUNCTION archivia_socio() RETURNS TRIGGER AS $BODY$  
BEGIN  
INSERT INTO biblioteca.ex_socio(nome, data_disdetta) VALUES(old.nome, current_date);  
RAISE NOTICE "Socio archiviato";  
RETURN NULL;  
END;  
$BODY$  
LANGUAGE PLPGSQL;
```

```
CREATE TRIGGER cancella_socio AFTER DELETE ON biblioteca.socio  
FOR EACH ROW EXECUTE PROCEDURE archivia_socio();
```

- (5) Si scriva un'applicazione Java che, caricando l'opportuno driver JDBC, si connette al DB di riferimento e:
- Crea all'interno dello schema *biblioteca_giovani* la tabella seguente (imponendo i vincoli opportuni):
 - *libri_teen*(ISBN, *titolo*, *autore*)
 - Popola la tabella creata sulla base dei dati presenti nello schema *biblioteca*: In particolare, *libro_teen* deve contenere i libri letti da almeno 2 teenagers (soci di età compresa tra 13 e 19 anni).
 - Stampa sul file *libri_teen.txt* il contenuto della tabella *libri_teen*