Esercizi

Un database per la memorizzazione dei dati del cosiddetto *registro elettronico* utilizzato da un istituto scolastico è caratterizzato dalla seguente porzione di schema logico relazionale:[1]

MATERIE(<u>codice</u>, nome)
DOCENTI(<u>matricola</u>, cognome, nome)
CLASSI(<u>nome</u>, anno_corso, docente_tutor$^{\star}_{fk}$)
STUDENTI(<u>matricola</u>, cognome, nome, classe$_{fk}$, data_ultima_connessione)
INSEGNAMENTI(<u>materia$_{fk}$</u>, <u>classe$_{fk}$</u>, docente$_{fk}$, ore_settimanali)
VOTI(studente$_{fk}$, materia$_{fk}$, tipo, voto, data_voto)
COMUNICAZIONI(<u>codice</u>, classe$^{\star}_{fk}$, studente$^{\star}_{fk}$, data, testo)

Con riferimento allo schema relazionale suddetto, esprimere le seguenti interrogazioni in linguaggio SQL (ove non altrimenti specificato).

1. Scrivere l'istruzione DDL per la creazione della relazione VOTI. Codificare i vincoli che impongono:
   - che l'attributo "tipo" possa avere come valore solo "scritto" oppure "orale";
   - che l'attributo "voto" sia compreso tra 0.0 e 10.0, con precisione al decimo di intero;
   - che uno studente non possa avere attribuiti più voti nella stessa materia e nella stessa data.
2. Esprimere il vincolo di integrità che impone che ogni docente non possa insegnare più di una materia per classe.
3. Estrarre l'elenco delle materie insegnate dal docente Zaffanella Enea. Esprimere l'interrogazione sia in SQL che in algebra relazionale.
4. Eliminare dal database tutti i voti relativi alle materie di nome "Matematica" e "Informatica".
5. Estrarre l'elenco degli studenti che non hanno ancora ottenuto un voto orale in almeno uno degli insegnamenti previsti dalla loro classe.
6. Creare una vista relazionale che calcoli, per ogni classe dell'istituto, la corrispondente numerosità (ovvero, il numero di studenti della classe).
7. Utilizzando la vista definita al punto precedente, elencare le classi dell'istituto in ordine decrescente di numerosità, indicando anche (quando presente) cognome e nome del tutor.
8. Creare la vista relazionale COMUNICAZIONI_RILEVANTI che associa ad ogni matricola di studente il testo e la data delle comunicazioni a lui (direttamente o indirettamente) indirizzate: ovvero, oltre alle comunicazioni personali che lo riguardano, anche quelle indirizzate alla sua classe e quelle indirizzate a tutti gli studenti dell'istituto (cioè, quelle che non sono specifiche per uno studente o per una classe).
9. Utilizzando la vista precedente, creare la vista relazionale NEWS che associa ad ogni studente il testo e la data delle comunicazioni per lui rilevanti aventi una data maggiore o uguale a quella dell'ultima connessione effettuata dallo studente; elencare i risultati in ordine decrescente di data.
10. Estrarre per ogni materia l'elenco degli studenti che hanno un voto medio insufficiente (cioè inferiore a 6.0), indicando il nome della materia, la matricola dello studente e il voto medio calcolato.
11. Fornire l'elenco dei docenti (cognome e nome in ordine alfabetico) che hanno un impegno didattico settimanale (numero totale di ore di insegnamento) maggiore di zero ma inferiore a 18.
12. Esprimere come espressione dell'algebra relazionale il vincolo che impedisce ad uno studente di conseguire un voto in una materia se la classe dello studente non ha effettivamente associato un insegnamento per quella materia.

[1] Un attributo annullabile è indicato con la notazione A*. I vincoli di chiave esterna sono indicati con la notazione A$_{fk}$; la relazione a cui fa riferimento il vincolo è ricavabile facilmente dal nome dell'attributo.

```
/* Soluzione prova itinere del 05/05/2014 */


 -- 1)

create table voti (
       studente integer  not null references studenti(matricola),
       materia varchar not null references materie(codice),
       tipo varchar(7) not null,
       voto numeric(3,1) not null,
       data_voto date not null,
       check (tipo = 'scritto' or tipo = 'orale'),
       -- check (tipo in ('scritto', 'orale')),
       -- check (tipo like 'scritto' or tipo like 'orale'), PEGGIO DEGLI
ALTRI
       check (voto between 0.0 and 10.0),
       primary key(studente, materia, data_voto) -- PUÃ' ESSERE ANCHE
UNIQUE
);


 -- 2)

 alter table insegnamenti
       add constraint max_docente_per_classe
          /*    constraint max_docente_per_classe   era opzionale */
       unique (classe, docente);

 -- 3)

 /* potevi includere o il nome delle materie, o il codice della materia,
 ma solo se era varchar perchÃ¨ probabilmente il codice sarebbe stato
 significativo */

 select distinct m.codice, m.nome
 from insegnamenti i, docenti d, materie m
 where i.materia = m.codice
       and i.docente = d.matricola
       and d.cognome = 'Zaffanella' and d.nome = 'Enea';
 order by m.nome;

 /* algebra relazionale

 PROJ_{codice, nome_materia}
   ((insegnamenti JOIN_{insegnamenti.docenti = docenti.matricola}
      SEL_{cognome='Zaffanella' and nome = 'Enea'} (docenti))
      JOIN_{insegnamenti.materia = materie.codice}
         REN_{nome_materia <- nome} (materie))

 */

 -- 4)

 delete from voti
   /*puoi usare una sola tabella, e delete va seguito subito da from */
 where materia in (
       select codice
       from materie
```

```
      where nome in ('Matematica', 'Informatica')
);

-- 5)

select distinct s.matricola, s.cognome, s.nome
from studenti
wheres.classe = i.classe
  and not exists (
      select *
      from voti v
      where v.studente = s.matricola
        and v.materia = i.materia
      and v.tipo = 'orale'
   )

-- 6)

create view numerosita_classi (classe, numero_studenti) as
       select classe, count(*)
       from studenti
       group by classe;

-- 7)

select nc.classe, nc.numero_studenti, d.cognome, d.nome
from numerosita_classi nc,
          classi c left outer join docenti dt on c.docente_tutor =
dt.matricola
where nc.classe = c.nome
order by nc.numero_studenti desc

-- 8)

create view comunicazioni_rilevanti (studente, data, testo) as
select s.matricola, c.data, c.testo
from comunicazioni c, studenti s
where c.classe is null and c.studente is null

union

select s.matricola, c.data, c.testo
from comunicazioni c, studenti s
where c.studente = s.matricola

union

select s.matricola, c.data, c.testo
from comunicazione c, studenti s
where c.classe = s.classe

-- ALTRIMENTI USAVI L'OR SULLE CONDIZIONI AL POSTO DI FARE LE UNION

/* create view comunicazioni_rilevanti (studente, data, testo) as
select s.matricola, c.data, c.testo
from comunicazioni c, studenti s
where (c.classe is null and c.studente is null)
      or  c.studente = s.matricola
```

```
      or c.classe = s.classe

*/

-- 9)

create view news (studente, data, testo) as
select cr.*
from comunicazioni_rilevanti cr, studenti s
where s.matricola = cr.studente
      and cr.data >= s.data_ultima_connessione
order by cr.data desc;

--10)

select s.matricola, s.cognome, s.nome, m.nome as materia, avg(v.voto) as
media
from  voto v, materia m, studente s
--potevi anche tenere solo la matricola
where v.materia = m.codice
  and v.studente =s.matricola
group by s.matricola, s.cognome, s.nome, materia
having avg(v.voto) < 6.0;
order by s.cognome, s.nome, m.nome
-- media lo accettano solo pochi sistemi


--11)

select d.cognome, d.nome
from insegnamenti i, docenti d
where i.docente = d.matricola
group by d.matricola, d.cognome, d.nome
having sum(i.ore_settimanali) between 1 and 17
order by d.cognome, d.nome

-- 12)

/*
algebra relazionale

PROJ_{materia, studente} (voti)
\
REN studente <- matricola
     ( PROJ_{insegnamenti.materia, studenti.matricola}
             (insegnamenti
                 JOIN_{insegnamenti.classe = studenti.classe}
             studenti
             )
      )
= 0

*/
```