Basi di Dati
Prof. Alfredo Pulvirenti
A.A. 2014-2015
Prova in itinere 18 dicembre 2014
(A)

Dato lo schema:

EVENTO(id, titolo, data, categoria, costo_partecipazione, idcatering)
ORGANIZZATORE(id,idevento)
PERSONA(id, nome, cognome, provincia_residenza)
PARTECIPANTE(idevento,idpersona)
CATERING(idcompagnia,nome,descrizione)
CATEGORIAEVENTO(id,descrizione)

1. Identificare le chiavi primarie ed esterne dello schema.
   Le chiavi primarie sono sottolineate in modo continuo quelle esterne sono sottolineate in modo tratteggiato.
   Si noti che se si suppone che un evento ha un unico organizzatore allora la relazione ORGANIZZATORE ha come chiave primaria solo idevento.

2. Scrivere in algebra relazionale le seguenti query
   Per risolvere le query useremo le relazioni con un nome abbreviato.
   E ← EVENTO
   O ← ORGANIZZATORE
   P ← PERSONA
   PA ←PARTECIPANTE
   C ← CATERING
   CE ← CATEGORIAEVENTO

   Useremo la notazione puntata per distinguere due campi con lo stesso nome in relazioni differenti.

   a. Trovare nome e cognome dei partecipanti agli eventi tenuti tra il 15 e il 20 dicembre 2013

   $$\pi_{nome,cognome}\left(\mathrm{P} \triangleright \triangleleft_{id=\mathrm{p}.idpersona} PA \triangleright \triangleleft_{idevento=id} \sigma_{data \geq 15/12/2013 \wedge data \leq 20/12/2014}(\mathrm{E})\right)$$

   b. Trovare la persona che ha partecipato a tutti gli eventi di tipo “Festa Laurea”.

   $$\delta_{id \leftarrow idevento}(PA) \div \pi_{\mathrm{E}.id}\left(\sigma_{descrizione='FestaLaurea'}(\mathrm{E} \triangleright \triangleleft_{\mathrm{E}.categoria=CE.id} CE)\right)$$

   c. Trovare le persone che hanno partecipato a tutte le categorie di eventi.

   $$\delta_{id \leftarrow categoria}\left(\pi_{idpersona,categoria}(PA \triangleright \triangleleft_{idevento=id} \mathrm{E})\right) \div \pi_{id}(CE)$$

d. Trovare le persone che non hanno mai organizzato un evento di tipo “Matrimonio”.

$$\pi_{id}(\mathrm{O}) - \pi_{\mathrm{O}.id}(\mathrm{O} \triangleright \triangleleft_{\mathrm{O}.idevento=\mathrm{E}.id} \mathrm{E} \triangleright \triangleleft_{\mathrm{E}.categoria=CE.id} \sigma_{descrizione='MATRIMONIO'}(CE))$$

3. Scrivere in SQL le seguenti query:
   a. Trovare gli eventi che hanno avuto il maggiore incasso.

```
SELECT sum(costo_partecipazione) incasso, EVENTO.id
FROM EVENTO, PARTECIPANTE
WHERE EVENTO.id = PARTECIPANTE.idevento
group by  EVENTO.ID
HAVING incasso = (SELECT max(incasso)
                  FROM (SELECT sum(costo_partecipazione) incasso, EVENTO.id
                        FROM EVENTO, PARTECIPANTE
                        WHERE EVENTO.id = PARTECIPANTE.idevento
                    GROUP BY  EVENTO.ID)) e)
```

   b. Trovare gli eventi che hanno avuto il minor numero di partecipanti nel 2013.

```
SELECT count(*) num_partecipanti, EVENTO.id
FROM EVENTO,PARTECIPANTE
WHERE EVENTO.id=PARTECIPANTE.idevento
AND year(EVENTO.data) = 2013
GROUP BY EVENTO.id
HAVING count(*) = (SELECT min(num_partecipanti)
                   FROM (SELECT count(*) num_partecipanti, EVENTO.id
                         FROM EVENTO,PARTECIPANTE
                         WHERE EVENTO.id=PARTECIPANTE.idevento
                           AND year(EVENTO.data) = 2013
                           GROUP BY EVENTO.id) t
                     )
```

   c. Scrivere un trigger usando la sintassi Oracle che per ogni partecipate iscritto ad un evento incrementi il numero di partecipanti di una tabella Partecipanti(idevento,totale_partecipanti).

```
CREATE TRIGGER T1
AFTER INSERT ON PARTECIPANTI
FOR EACH ROW
DECLARE
  X number;
  select count(*) into X from partecipanti
  where idevento = new.idevento;
    IF X = 0 THEN
            insert into partecipanti values(new.idevento,1);
    ELSE
            select totale_partecipanti into X from partecipanti
            where idevento = new.idevento;

            update partecipanti set totale_partecipanti = X+1
            where idevento = new.idevento;
    ENDIF;
END;
```

   d. Trovare le persone che hanno partecipato a tutti gli eventi organizzati da un organizzatore di Catania.

```
   SELECT *
FROM persone p
WHERE NOT EXISTS (SELECT *
                  FROM organizzatore o, persona
```

```
o.id = persona.id AND
persona.provincia_residenza = 'CT' AND
NOT EXISTS (SELECT *
                FROM partecipante pa
                WHERE pa.idpersona=p.id AND
                    pa.idevento=o.idevento))
```

e. Supponendo che le coppie di persone che hanno partecipato allo stesso evento siano amici. Trovare le coppie di persone che non sono amiche ma sono connesse da una catena di amici comuni.

```
CREATE VIEW amici AS
SELECT p1.idpersona as amico1,p2.idpersona as amico2
FROM partecipante p1, partecipante p2
WHERE  p1.idpersona > p2.idpersona
AND p1.idevento = p2.idevento

with recursive nonAmici(amicoX,amicoY) AS
(           select amico1, amico2
            from amici
    union all
            select A.amicoX, amico2
            from nonAmici A, amici
            where D.amicoY = amico1)

select *
from nonAmici na
WHERE NOT EXISTS (SELECT *
                    FROM amici
                    WHERE na.amicoX=amico1 AND na.amicoY=amico2)
```

f. Definire un vincolo di integrità che consenta di non inserire come partecipante l'organizzatore di una festa.

```
ALTER TABLE partecipante
ADD CONSTRAINT myCheckConstraint
CHECK(NOT EXISTS
            (SELECT *
            FROM organizzatore
            WHERE id = idpartecipante));
```

Si noti che la soluzione non consente di inserire come partecipante una persona che risulta organizzatore di un evento anche DIVERSO da quello per cui si sta effettuando l'inserimento. Per superare a tale limite si può inserire un'altra condizione.

```
ALTER TABLE partecipante
ADD CONSTRAINT myCheckConstraint
CHECK(NOT EXISTS
            (SELECT *
            FROM organizzatore o
            WHERE id = idpartecipante
            AND o.idevento=idevento));
```

Basi di Dati
Prof. Alfredo Pulvirenti
A.A. 2014-2015
Prova in itinere 18 dicembre 2014
(B)

Dato lo schema:

PERSONA(id, nome, cognome, provincia_residenza,stato_civile)
ORGANIZZATORE(id,idevento)
PARTECIPANTE(idevento,idpersona)
CATERING(idcompagnia,nome,descrizione,sede)
CATEGORIAEVENTO(id,descrizione)
EVENTO(id, titolo, data, categoria, costo_partecipazione, idcatering)

1. Identificare le chiavi primarie ed esterne dello schema

   Vedi compito A per le chiavi primarie ed esterne (la presenza dell'attributo stato_civile in questo schema non ha effetto sulle chiavi). Vedi compito a per le relazioni abbreviate negli esercizi di Algebra Relazionale

2. Scrivere in algebra relazionale le seguenti query.

   a. Trovare nome e cognome dei partecipanti agli eventi che non si sono tenuti tenuti tra il 15 e il 20 ottobre 2009

   $$\pi_{nome,cognome}(\mathrm{P} \triangleright \triangleleft_{id=\mathrm{p}.idpersona} PA \triangleright \triangleleft_{idevento=id} \sigma_{data \leq 15/10/2009 \vee data \geq 20/10/2013}(\mathrm{E}))$$

   b. Trovare le persone che hanno partecipato a tutte categorie di eventi.

      Vedi punto C compito A

   c. Trovare le persone che non hanno mai organizzato un evento di tipo "Compleanno".

   $$\pi_{id}(\mathrm{O}) - \pi_{\mathrm{O}.id}(\mathrm{O} \triangleright \triangleleft_{\mathrm{O}.idevento=\mathrm{E}.id} \mathrm{E} \triangleright \triangleleft_{\mathrm{E}.categoria=CE.id} \sigma_{descrizione='COMPLEANNO'}(CE))$$

   d. Trovare le persone che hanno partecipato a tutti gli eventi del 2013.

   $$\delta_{id \leftarrow idevento}(PA) \div \pi_{id}(\sigma_{data \geq 1/1/2013 \wedge data \leq 31/12/2013}(\mathrm{E}))$$

3. Scrivere in SQL le seguenti query:
    a. Trovare gli eventi che hanno avuto il minor incasso.

```
SELECT sum(costo_partecipazione) incasso, EVENTO.id
FROM EVENTO, PARTECIPANTE
WHERE EVENTO.id = PARTECIPANTE.idevento
group by  EVENTO.ID
HAVING incasso = (SELECT min(incasso)
                  FROM (SELECT sum(costo_partecipazione) incasso, EVENTO.id,
                        FROM EVENTO, PARTECIPANTE
                        WHERE EVENTO.id = PARTECIPANTE.idevento
                    GROUP BY  EVENTO.ID)) e)
```

    b. Trovare gli eventi “Laurea” che hanno avuto il maggior numero di partecipanti “sposati”.

```
SELECT count(*) num_partecipanti, Evento.id
FROM EVENTO,PARTECIPANTE
WHERE EVENTO.id=PARTECIPANTE.idevento
AND stato_civile=”coniugato”
GROUP BY EVENTO.id
HAVING count(*) = (SELECT max(num_partecipanti)
                   FROM (SELECT count(*) num_partecipanti, EVENTO.id
                         FROMx EVENTO,PARTECIPANTE
                         WHERE EVENTO.id=PARTECIPANTE.idevento
                            AND stato_civile=”coniugato”
                            GROUP BY EVENTO.id) t
                     )
```

    c. Trovare le persone che hanno partecipato a tutti gli eventi organizzati da un organizzatore di Roma.

```
  SELECT *
FROM persone p
WHERE NOT EXISTS (SELECT *
                  FROM organizzatore o, persona
                  o.id = persona.id AND
                  persona.provincia_residenza = ‘RM’ AND
                  NOT EXISTS (SELECT *
                              FROM partecipante pa
                              WHERE pa.idpersona=p.id AND
                                   pa.idevento=o.idevento))
```

    d. Definire un vincolo di integrità che consenta di non inserire come partecipante l’organizzatore di una festa.

        Vedi compito A

    e. Creare una vista che contenga le coppie distinte di persone che hanno partecipato allo stesso evento di tipo compleanno. Supponendo che tali coppie di persone siano amici. Trovare le coppie di persone che non sono amiche ma sono connesse da una catena di amici comuni.

        Vedi compito A

    f. Scrivere un trigger usando la sintassi Oracle che per ogni partecipate iscritto ad un evento incrementi il numero di

partecipanti di una tabella Partecipanti(idevento,totale_partecipanti).

Vedi compito A

BASI DI DATI
Prof. Alfredo Pulvirenti
17 dicembre 2014

a) Dato lo schema

CONTOCORRENTE(id_conto,saldo,data_apertura)
PRODOTTOFINANZIARIO(id_conto, id_prodotto,data_stipula,numero_rate,id_contraente)
PERSONA(id_persona,nome,cognome,data_nascita)
TITOLARECC(Id_persona,ID_conto)
TRANSAZIONE(id_contoIN,id_contoOUT,data,causale,dare_avere,importo)
TRANSAZIONEPRODOTTOFINANZIARIO(id_conto,data,causale ,importo,id_prodotto)

1. Identificare le chiavi primarie ed esterne dello schema
2. Scrivere le seguenti query in algebra relazionale.
   - Trovare il conto corrente (cc) aperto in data 11/11/2012 con due intestatari e con saldo minimo [2 punti].
     $R2 = TITOLARECC$
     $R3 = TITOLARECC$
     $R4 = \pi_{id\_conto}(R2 \rhd\lhd_{R2.id\_conto=R3.id\_conto \wedge R2.id\_persona>R3.id\_persona} R3)$
     $R1 = \pi_{id\_conto,saldo}(\sigma_{data\_apertura=11/11/12}(CONTOCORRENTE \rhd\lhd R4))$
     $R5 = R1$
     $R1 - \pi_{R1.id\_conto,R1.saldo}(R1 \rhd\lhd_{R1.saldo<R5.saldo} R5)$
   - Trovare i conti correnti che non hanno prodotti finanziari aperti [1 punto].
     $R1 = \pi_{id\_conto}(CONTOCORRENTE)$
     $R2 = \pi_{id\_conto}(PRODOTTOFINANZIARIO)$
   - $R1 - R2$
   - Trovare le coppie di persone che hanno esattamente gli stessi cc [4 punti]
     $R1 = TITOLARECC$
     $R2 = TITOLARECC$
     $R3 = \pi_{R1.id\_persona,R2.id\_persona}(R1 \rhd\lhd_{\substack{R1.id\_conto=R2.id\_conto \\ R1.id\_persona>R2.id\_persona}} R2)$
     $R4 = \pi_{R1.id\_persona,R2.id\_persona}(R1 \rhd\lhd_{\substack{R1.id\_conto>R2.id\_conto \\ R1.id\_persona>R2.id\_persona}} R2)$
   - $R3 - R4$
3. Scrivere le seguenti query in SQL
   - Trovare il cc con il saldo più alto [1 punto]
     ```
     SELECT id_conto
     FROM CONTOCORRENTE CC
     WHERE NOT EXISTS (SELECT *
                       FROM CONTOCORRENTE CC1
                       WHERE CC1.saldo > CC.saldo)
     ```
   - Trovare la persona che ha il saldo totale più elevato [2 punti]
     ```
     SELECT id_persona,sum(saldo)
     FROM CONTOCORRENTE NATURAL JOIN TITOLARE CC
     GROUP BY id_PERSONA
     ```

```
HAVING sum(saldo) >=
                (SELECT saldo_tot
                   FROM (SELECT id_persona,sum(saldo) saldo_tot
                 FROM CONTOCORRENTE NATURAL JOIN TITOLARE CC
                 GROUP BY id_PERSONA)
```

- Trovare il prodotto finanziario che ha più rate [2 punti]

```
- SELECT id_prodotto
- FROM PRODOTTOFINANZIARIO PF
- WHERE NOT EXISTS (SELECT *
-                          FROM PRODOTTOFINANZIARIO PF1
-                          WHERE PF1.numero_rate > PF.numero_rate)
```

- Implementare un trigger che per ogni Transazione (anche transazione su prodotto finanziario) aggiorna automaticamente tutti i cc coinvolti [3 punti].

```
CREATE TRIGGER syncCC
after insert on TRANSAZIONE
for each row
    referencing new as N
Begin
    UPDATE CONTOCORRENTE SET saldo=saldo+importo WHERE
    id_conto=N.id_contoIN;
    UPDATE CONTOCORRENTE SET saldo=saldo-importo WHERE
    id_conto=N.id_contoOUT;
End;

CREATE TRIGGER syncCCPF
after insert on TRANSAZIONEPRODOTTOFINANZIARIO
for each row
    referencing new as N
Begin
    UPDATE CONTOCORRENTE SET saldo=saldo-importo WHERE
    id_conto=N.id_conto;
End;
```

b) Progettazione [11 punti]

Si deve automatizzare un’agenzia di viaggi. Ogni viaggio viene svolto in un certo intervallo di date, ha un nome e un costo e la categoria. Per ogni viaggio bisogna memorizzare in quali alberghi vengono effettuati i pernottamenti. Per ogni albergo si devono mantenere una serie di informazioni riguardanti il numero di stelle e i vari servizi offerti (piscina, ristorante, garage, ecc.). Gli alberghi sono catalogati per città e per nazione. Per ogni viaggio, inoltre, si devono memorizzare le informazioni circa i partecipanti, per potergli proporre via e-mail nuove mete simili a quelle già effettuate. La similarità è stabilità dal costo e dalla categoria del viaggio.

Per ogni viaggio vengono memorizzati i commenti dei partecipanti insieme con una valutazione da 1 a 5. Per ciascun commento bisogna tener traccia dell'età del partecipante e il suo status (viaggiatore solitario, viaggiatore in coppia, viaggiatore con famiglia, viaggiatore in gruppo). Si mantiene una newsletter con la data in cui è stata inviata la pubblicità e con l'indicazione del viaggio pubblicizzato e della data di partenza.

1. Effettuare la progettazione concettuale descrivendo lo schema ER ai vari livelli di raffinamento.
2. Effettuare la progettazione logica dando come output lo schema relazionale, una tavola dei volumi ed un elenco di operazioni con le rispettive frequenze.
3. Discutere la presenza di eventuali anomalie nello schema e motivarle.

c) Descrivere quali sono i modi di implementare le join a livello fisico [4 punti].