**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.

1. 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.

* 1. Trovare nome e cognome dei partecipanti agli eventi tenuti tra il 15 e il 20 dicembre 2013
  2. Trovare la persona che ha partecipato a tutti gli eventi di tipo “Festa Laurea”.
  3. Trovare le persone che hanno partecipato a tutte le categorie di eventi.
  4. Trovare le persone che non hanno mai organizzato un evento di tipo “Matrimonio”.

1. Scrivere in SQL le seguenti query:
   1. 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)

* 1. 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

)

* 1. 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;**

* 1. 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))

* 1. 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)

* 1. 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

1. Scrivere in algebra relazionale le seguenti query.
   1. Trovare nome e cognome dei partecipanti agli eventi che non si sono tenuti tenuti tra il 15 e il 20 ottobre 2009
   2. Trovare le persone che hanno partecipato a tutte categorie di eventi.

Vedi punto C compito A

* 1. Trovare le persone che non hanno mai organizzato un evento di tipo “Compleanno”.
  2. Trovare le persone che hanno partecipato a tutti gli eventi del 2013.

1. Scrivere in SQL le seguenti query:
   1. 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)

* 1. 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

)

* 1. 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))

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

Vedi compito A

* 1. 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

* 1. 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].



* Trovare i conti correnti che non hanno prodotti finanziari aperti [1 punto].



* Trovare le coppie di persone che hanno esattamente gli stessi cc [4 punti]



1. 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].