BASI DI DATI - PROFF. CAMPI, CERI, POZZI

I PROVA IN ITINERE - 22 NOVEMBRE 2004

A. SQL (P. 8)

Si considerino le tabelle che rappresentano i dati di un gioco a quiz dove un concorrente deve rispondere alle domande proposte scegliendo la risposta tra un insieme di risposte prefedinito per quella domanda.

Concorrente (CodiceFiscale, Nome, Cognome, Città, Vincita)
Domanda (Id, Testo, Premio, IdRispostaCorretta)
Risposta (IdDomanda, IdRisposta, Testo)
Risposta_Concorrente (CodiceFiscale, IdDomanda, IdRisposta)

Si noti che per ogni domanda esistono più risposte, di cui solo una è corretta; si assuma inoltre che tutte le domande siano poste a tutti i concorrenti.

1. Estrarre il testo delle domande alle quali nessun concorrente ha *mai* risposto in modo *errato* (P.3).

```
select Id, Testo
from Domanda D
where not exists  ( select *
                    from RispostaConcorrente
                    where IdDomanda = D.Id
                       and IdRisposta <> D.IdRispostaCorretta  )
oppure

select Id, Testo
from Domanda D
where IdRispostaCorretta = ALL (  select IdRisposta
                                  from RispostaConcorrente
                                  where D.Id = IdDomanda  )
oppure ancora

( select Id, Testo
  from Domanda )
     except
( select Id, Testo
  from Domanda join Risposta_Concorrente on Id = IdDomanda
  where IdRisposta <> IdRispostaCoretta )
```

Assolutamente *NON* select Id, Testo /* Errore grave: chiede che nessuno abbi amai indovinato nulla */
from Domanda
where not exists (select *
from Domanda join Risposta_Concorrente on Id=IdDomanda
where IdRisposta <> IdRispostaCoretta)

2. Estrarre le domande che tutti i vincitori di più di 1000 euro hanno sbagliato (P.3)

*Siccome tutte le domande sono poste a tutti i concorrenti, posso raggruppare **per domanda** le sole risposte sbagliate da vincitori di più di 1000 euro, e trattenere solo i gruppi con tante risposte quanti vincitori:*

```
select D.Id
from Domanda D, Risposta_Concorrente RC, Concorrente C
where RC.CodFis = C.CodFis and D.Id = RC.IdDomanda and RC.IdRisp <> D.RispCorr
   and C.Vincita > 1000
group by D.Id
having count (*) =  (   select count(*)
                        from Concorrente
                        where Vincita > 1000 )
```

*Oppure si può definire una vista per individuare – ad esempio – tutte le risposte **giuste** dei concorrenti "ricchi", cercando poi le domande che non vi compaiono (perché nessun "ricco" ha risposto correttamente)*

```
 Crete view RispOkRicchi (IdDom, IdRisp) as
 select Id, IdRispostaCorretta
 from (Domanda join Risposta_Concorrente RC on Id = IdDomanda) join Concorrente C
        on RC.Codice Fiscale = C.CodiceFiscale
 where C.Vincita > 1000 and RC.IdRisposta = IdRispostaCorretta

  select Id
  from Domanda
  where Id not in   (    select IdDom
                         from RispOkRicchi   )
```

Oppure ancora si può applicare direttamente la definizione: scelgo una domanda se la sua risposta corretta non compare tra quelle date dai ricchi (per quella domanda)

```
select ID
from Domanda
where (Id, IdRispostaCorretta) <> ALL                  /* equivalentemente "not in" */
        ( select IdDomanda, IdRisposta
          from Concorrente C join Risposta_Concorrente RC on C.CodFisc = RC.CodFisc
          where Vincita > 1000 )
```

3. Scrivere un comando di update che assegna al campo Vincita di ogni Concorrente il valore calcolato in base alle risposte che ha dato (P.2).

```
update Concorrente C
   set Vincita =  (  select sum(Premio)
                     from Domanda join Risposta_Concorrente R
                              on ( Id = IdDomanda and IdRisposta = IdRispostaCoretta )
                     where R.CodiceFiscale = C.CodiceFiscale)
```

B. LINGUAGGI FORMALI DI INTERROGAZIONE (P. 6)

1. Esprimere in Algebra ottimizzata, Calcolo **e** Datalog la prima interrogazione dell'esercizio precedente.

Algebra (non ottimizzata – manca il push delle proiezioni)

$$\Pi_{Id,Testo} (DOM - (DOM \triangleright\!\!<_{Id=IdDomanda \wedge IdRispostaCorretta <> IdRisposta} RIS\_CON))$$

TRC:

$\{ t \mid \exists t_D \in DOM$ *(Esiste la domanda)*
$\quad (t[Id, Testo] = t_D[Id, Testo] \wedge$
$\quad \neg (\exists t_R \in RIS\_CON$ *(e non c'è alcuna risposta di alcun concorrente)*
$\qquad (t_R[IdDomanda] = t_{\mathbf{D}}[Id] \wedge$ *(alla **stessa** domanda)*
$\qquad\ t_R[IdRisposta] <> t_{\mathbf{D}}[IdRispostaCorretta])$ *(con una risposta sbagliata)*
$\quad)$
$\quad)$
$\}$

Datalog:

```
RispostaSbagliata ( IdDom ) :-  DOM ( IdDom, _, _, IdCorretta ),
                                RIS_CON ( _, IdDom, IdErrore ),
                                IdCorretta <> IdErrore

DomandaMaiSbagliata ( Id, Testo ) :-  DOM ( Id, Testo, _, _ ),
                                      ¬ RispostaSbagliata ( Id )

? – DomandaMaiSbagliata ( X, Y )
```

2. Esprimere in **due** linguaggi a scelta tra Algebra, Calcolo e Datalog l'interrogazione che estrae il concorrente di Milano che ha vinto più soldi. (P.3)

Algebra (non ottimizzata):

$$\Pi_{CodFisc,Nome,Cognome}(\sigma_{Città='Milano'} CON - (\sigma_{Città='Milano'} CON \triangleright\!\!<_{Vincita<Vincita} \sigma_{Città='Milano'} CON))$$

TRC:

$\{ t \mid \exists t_C \in CON$ *(Esiste il concorrente)*
$\quad (t[CodFisc, Nome, Cognome] = t_C[CodFisc, Nome, Cognome] \wedge$
$\quad t_C[Città] = 'Milano' \wedge$ *(di Milano)*
$\quad \neg (\exists t_{C2} \in CON$ *(e non c'è alcun concorrente)*
$\qquad (t_{C2}[Città] = 'Milano' \wedge$ *(di Milano con)*
$\qquad\ t_{C2}[Vincita] > t_C[Vincita])$ *(una vincita superiore a **quella**)*
$\qquad)$
$\quad)$
$\}$

Datalog:

```
VincitaSuperata(CF) :-  CON ( CF, _, _, _, VincBassa ),
                        CON ( _, _, _, _, VincAlta ),
                        VincAlta > VincBassa

VincitaMassima ( CodFisc ) :-  CON ( CodFisc, _, _, _, _ ),
                               ¬ VincitaSuperata (CodFisc )

? – VincitaMassima ( Z )
```

C. DDL (P. 2)

Scrivere i comandi SQL per

1. Creare le tabella Risposta_Concorrente, effettuando opportune e ragionevoli ipotesi su domini e vincoli e reazioni ai cambiamenti.

```
CREATE TABLE Risposta_Concorrente (
        CodiceFiscale char[16] references Concorrenti(CodFisc)        /* Non varchar!! */
                on delete no action
                on update cascade,
        IdDomanda integer,
        IdRisposta char,                                    /* può bastare - ad esempio: a, b, c… ! */
        primary key (CodiceFiscale, IdDomanda),
        foreign key (IdDomanda, IdRisposta)
                references Risposta (IdDomanda, IdRisposta) on delete cascade
                                                            on update cascade)
```

*È fondamentale che l'integrità referenziale sia applicata alla **coppia** di valori in Risposta. Ogni altra chiave esterna o è sintatticamente scorretta o rischia di non vincolare la risposta ad essere una risposta lecita (ad esempio vincolare solo l'IdDomanda ad appartenere a Domanda ma non l'IdRisposta permetterebbe di rappresentare una risposta 'd' a una domanda che prevede solo 3 opzioni)*

2. Aggiungere alla tabella CONCORRENTE l'attributo Indirizzo.

```
ALTER TABLE Concorrente
        ADD COLUMN Indirizzo varchar(255)
```