Informatica per le Scienze Umane - Corso di Laurea in Lettere
Appello del 17.06.2010 - Docente: Massimo Coppola
Modello Relazionale e SQL

Si consideri la seguente base di dati relativa ad un circolo di tennis, che nel calcolare le classifiche delle partite in singolo tiene traccia dell'ultima partita giocata da ogni coppia di aversari, e della loro suddivisione in squadre. Si noti che un giocatore può appartenere a più di una squadra.

```
TABLE Giocatori (
  codice: char(12) PRIMARY KEY,
  nome: char(40),
  cognome: char (40),
  sesso: bool,
  dataIscrizione: date,
  età: integer
)
```

```
TABLE Squadre (
  codice: char(10) PRIMARY KEY,
  nome: varchar(50),
  dataFondazione: date,
  categoria: char(20)
)
```

```
TABLE UltimoRisultato(
   Vincitore: char(12)
      REFERENCES Giocatori(codice) NOT NULL,
   Sconfitto: char(12)
      REFERENCES Giocatori(codice) NOT NULL,
   tipoTorneo: char (12),
   PRIMARY KEY(Vincitore, Sconfitto)
)
```

```
TABLE Affiliazioni (
   giocatore: char (12)
      REFERENCES Giocatori(codice),
   squadra: char(10)
      REFERENCES Squadre(codice),
   riserva: bool,
   PRIMARY KEY(giocatore, squadra)
)
```

Si formulino le seguenti interrogazioni tramite il linguaggio SQL:

1. Elencare i giocatori di cognome Rossi ed affiliati alla Squadra “Tramontana 87”. **(punti 5)**
2. Elencare i giocatori iscritti dopo il 1/05/2010 che non hanno ancora giocato una partita. **(punti 6)**
3. Elencare i giocatori che hanno giocato con avversario di sesso diverso, della stessa età, in un torneo di tipo Promozione. **(punti 6)**
4. Elencare i gruppi di tre giocatori A, B, C che hanno giocato tutte le partite A contro B, B contro C, C contro A. **(punti 6)**
5. Elencare le squadre che hanno almeno un giocatore che non appartiene ad alcun altra squadra. **(punti 2)**
6. Si modifichi la base di dati in modo da tenere traccia, per lo stesso insieme di giocatori e squadre, delle partite in doppio (due giocatori contro due) separatamente da quelle in singolo. **(punti 6)**

Informatica per le Scienze Umane - Corso di Laurea in Lettere
Appello del 17.06.2010 - Docente : Massimo Coppola
Parte II: Modello concettuale e XML

Per la gestione di una azienda che realizza sondaggi e campagna promozionali, gli impiegati lavorano in più uffici condivisi, e fanno parte di un insieme di team dedicati a distinti progetti e campagne.

- Ogni ***dipendente*** è caratterizzato da un proprio codice fiscale, il proprio nome, la data di assunzione, il ruolo (telefonista, analista, segretario, dirigente …), nonché dall'ufficio in cui lavora e dal team di cui fa parte.
- Ogni ***ufficio*** è caratterizzato da un codice identificativo alfanumerico, la sede (denominazione dell'edificio in cui è situato), l'ubicazione nella sede (alfanumerica), nonché dai dipendenti che vi lavorano e dai team che esso ospita.
- Ogni ***team*** è caratterizzato da un proprio codice, dal proprio nome, dal tipo di progetto su cui opera (sondaggio, promozione ecc.), nonché dalle persone che vi lavorano e dagli uffici (più di uno, in generale) in cui hanno delle risorse disponibili (archivi, personale ospitato o altro).

Non è richiesto introdurre nel modello in modo dettagliato l'insieme dei ruoli dei dipendenti, delle tipologie di progetti e di risorse.

1. Si rappresentino i fatti sopra descritti in uno schema concettuale UML **(9 punti)**

2. Si traduca lo schema concettuale in uno schema relazionale **(9 punti)**

3. Si fornisca un esempio di istanza dello schema contenente due team, due dipendenti e due uffici **(4 punti)**

4. Si rappresentino tutte le informazioni relative ad uno dei dipendenti rappresentati nell'istanza realizzata come un documento XML **(8 punti)**

5. Si fornisca il DTD per il documento XML di cui al punto precedente **(2 punti)**

Traccia di Soluzione della parte I

Risposta 1)
Qui si è scelta la risposta più completa, che fornisce codice, nome e cognome dei giocatori.

```
SELECT DISTINCT Giocatori(codice), Giocatori(nome), Giocatori(cognome)
FROM Giocatori JOIN Affiliazioni ON codice=giocatore
        JOIN Squadre ON squadra=Squadre.codice
WHERE cognome=”Rossi” AND Squadre.nome=”Tramontana 87”;
```

Risposta 2)
I giocatori che non hanno ancora giocato alcuna partita evidentemente non hanno alcun ultimo risultato nella base dati. Possiamo perciò prendere tutti i giocatori iscritti dalla data che ci interessa, ed escludere tutti quelli che abbiano giocato almeno un risultato nella base dati. Dobbiamo tenere conto però che ogni giocatore potrebbe risultare nella base dati sia come vincitore che come sconfitto. Notare che in questo caso il codice del giocatore è necessario onde evitare ambiguità nell'operazione insiemistica.

```
SELECT Giocatori(codice), Giocatori(nome), Giocatori(cognome)
FROM Giocatori
WHERE dataIscrizione > 01/05/2010
      EXCEPT
SELECT Giocatori(codice), Giocatori(nome), Giocatori(cognome)
FROM Giocatori JOIN UltimoRisultato ON codice = vincitore
      EXCEPT
SELECT Giocatori(codice), Giocatori(nome), Giocatori(cognome)
FROM Giocatori JOIN UltimoRisultato ON codice = sconfitto
```

Risposta 3)
Si assume ovviamente che il booleano della tabella Giocatori indichi con un valore il sesso maschile e con l'altro sesso femminile. Notare l'uso delle parentesi nella condizione, che benché tradotta quasi in modo letterale dalla definizione risulta complessa. È possibile semplificare la condizione utilizzando l'operazione di OR esclusivo (XOR) nella prima parte, per esprimere la condizione che il sesso dei due giocatori è diverso.

```
SELECT Giocatori(codice), Giocatori(nome), Giocatori(Cognome)
FROM Giocatori JOIN UltimoRisultato ON codice=vincitore
        JOIN Giocatori AS Giocatori2 ON sconfitto=Giocatori2(codice)
WHERE ( Giocatori(sesso)=TRUE AND Giocatori2(sesso)=FALSE OR
        Giocatori(sesso)=FALSE ANS Giocatori2(sesso)=TRUE ) AND
        Giocatori(età)=Giocatori2(età) AND UltimoRisultato(tipoTorneo)=”Promozione”;
```

Risposta 4)
È necessario effetuare la join a 3 della tabella Giocatori, che implica usare due volte la tabella UltimoRisultato come intermedia nella JOIN. Rinominare le tabelle in questo caso è necessario ad evitare ambiguità della query. Va esclusi esplicitamente i casi in cui i tre giocatori non sono distinti (è sufficiente verificare che il primo ed il terzo lo siano: perchè?). Anche in questo caso la risposta è possibile nei limiti dei dati contenuti nella nostra base dati, ovvero solo per quel che riguarda l'ultima partita giocata.

Questo incidentalmente può portare all'interessante conseguenza che nel caso

descritto, è possibile individuare gruppi di tre giocatori che hanno giocato assieme nelle tre possibili combinazioni solo se ognuno di loro ha vinto una partita e ne ha persa un'altra all'interno del gruppo. Tale deduzione dipende anche dall'interpretazione del testo del compito, e non era necessaria per risolvere l'esercizio.

```
SELECT Giocatori(codice), Giocatori2(codice), Giocatori3(codice)
FROM Giocatori JOIN UltimoRisultato ON codice = vincitore
        JOIN Giocatori AS Giocatori2 ON sconfitto=Giocatori2(codice)
        JOIN UltimoRisultato AS Risultato2 ON Giocatori2(codice)=Risultato2(vincitore)
        JOIN Giocatori AS Giocatori3 ON sconfitto=Giocatori3(codice)
WHERE Giocatore(codice)<>Giocatore3(codice);
```

Risposta 5)

Per semplicità ricaviamo solo il codice delle squadre richieste. La risposta può essere ottenuta (1) trovando tutti i giocatori che appartengono ad almeno due squadre, (2) escludendoli dalla tabella dei giocatori, e a questo punto individuando le squadre ad essi associate. Individuare i giocatori affiliati a piu' di una squadra è possibile facendo un Join della tabella Affiliazioni con se stessa, senza passare dai giocatori.

```
SELECT Giocatori(codice), Affiliazioni(squadra)
FROM Giocatori JOIN Affiliazioni
WHERE codice=giocatore
        EXCEPT
SELECT Affiliazioni(giocatore), Affiliazioni(squadra)
FROM Affiliazioni JOIN Affiliazioni AS Affiliazioni2
                        ON Affiliazioni(giocatore)=Affiliazioni2(giocatore)
WHERE Affiliazioni(squadra) <> Affiliazioni2(squadra)
```

Notiamo che (1) Nella prima select la JOIN con Affiliazioni serve unicamente a poter recuperare il codice della squadra da quello dei giocatori: altrimenti si potrebbe creare una tabella temporanea con i giocatori che ci interessano e poi usare quella per fare una seconda query sulla base dati. (2) La provenienza dei campi usati nelle due parti della except è diversa, ma essi corrispondono allo stesso tipo elementare, come richiesto dall'operazione insiemistica.

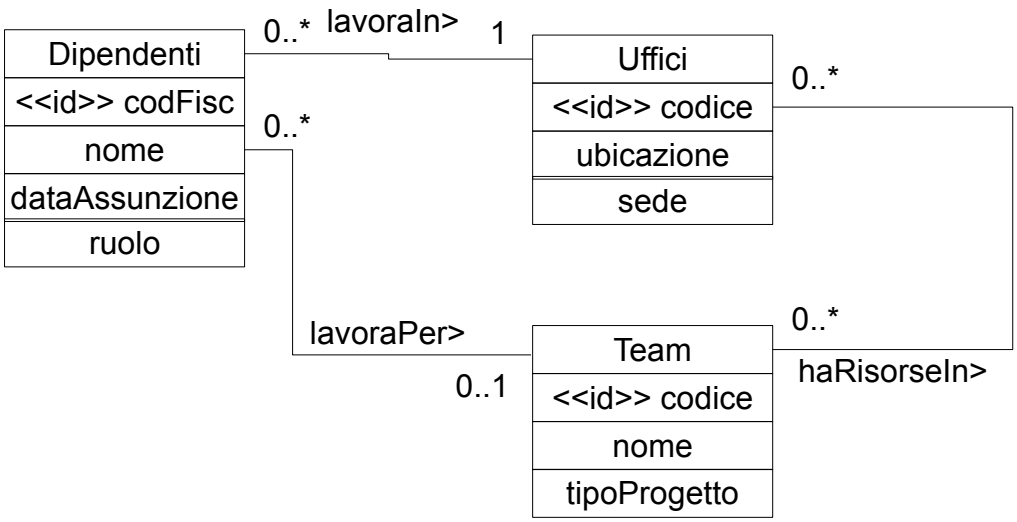
Risposta 6)

Definiamo una tabella aggiuntiva UltimoDoppio che i giocatori di una partita in doppio con le stesse regole della tabella UltimoRisultato già presente nel testo.

```
TABLE UltimoDoppio(
        Vincitore1: char(12) REFERENCES Giocatori(codice) NOT NULL,
        Vincitore2: char(12) REFERENCES Giocatori(codice) NOT NULL,
        Sconfitto1: char(12) REFERENCES Giocatori(codice) NOT NULL,
        Sconfitto2: char(12) REFERENCES Giocatori(codice) NOT NULL,
        tipoTorneo: char(12),
        PRIMARY KEY(Vincitore1,  Vincitore2,  Sconfitto1,  Sconfitto2)
)
```

Traccia di Soluzione della parte II
Risposta 1)

Schema concettuale UML



Risposta 2)
Abbiamo due collegamenti 1-m che si traducono come semplici riferimenti nella tabella Dipendenti. L'associazione tra uffici e team è più generale, e richiede perciò una tabella ausiliaria. Non introduciamo tabelle per i ruoli, i tipi di progetto, che saranno semplici campi carattere di lunghezza fissa. Inoltre *non* vi sono nel modello altre risorse (p.es. archivi, computer) oltre ai dipendenti. Dato che assumiamo di tenere traccia dei dipendenti associati agli uffici ed ai team, ma non delle altre risorse, vi potranno essere team associati ad uffici presso cui non hanno alcun dipendente ospitato.

```
TABLE Dipendenti{
  codFisc: char(16) PRIMARY KEY,
  nome: char (100) NOT NULL,
  dataAssunzione: date,
  ruolo: char(16),
  lavoraIn: char(4) REFERENCES Uffici(codice) NOT NULL,
  lavoraPer: char(8) REFERENCES Team(codice)
}

TABLE Uffici{
  codice: char(4) PRIMARY KEY,
  ubicazione: char (20),
  sede: char(50)
}

TABLE Team{
  codice: char(8) PRIMARY KEY,
  nome: char (20),
```

```
  tipoProgetto: char(32)
}

TABLE haRisorseIn{
  team: char(8) REFERENCES Team(codice),
  ufficio: char(4) REFERENCES Uffici(codice),
  PRIMARY KEY (team, ufficio)
}
```

Risposta 3)
Esempio di istanza della base dati.

Dipendenti

codFisc	nome	dataAssunz	ruolo	lavoraIn	lavoraPer
PTRPRK11E22Z456Z	Peter Parker	10/8/03	Contabile	D200	TEA00000
FLSTHM11D22X789I	Flash Thompson	11/04/01	Telefonista	A301	TEA00336

Uffici

codice	sede	ubicazione
D200	Pisa 1	D-200
A301	Firenze 3	3-01-nord

Team

codice	nome	tipoProgetto
TEA00000	Gestione Dipendenti	Amministrazione
TEA00336	Pontepi-TeamA	Sondaggio Telefonico

HaRisorseIn

team	ufficio
TEA00000	A301
TEA00000	D200
TEA00336	A301

Risposta 4)
In questa traccia si è scelto di memorizzare i codici degli elementi per mezzo di tag XML e non come attributi. Notare che in questa soluzione si sceglie di codificare la relazione molti a molti dall'azienda al quartiere, seguendo il verso della relazione in UML.

```
<?xml version="1.0"?>
<Dipendente>
    <codFisc>FLSTHM11D22X789I</codFisc>
    <nome>Flash Thompson</nome>
    <dataAssunz>11/04/01</dataAssunz>
    <ruolo>Telefonista</ruolo>
    <lavoraIn>
        <Ufficio>
```

```
            <codice>A301</codice>
            <sede>Firenze 3</sede>
            <ubicazione>3-01-nord</denominazione>
        </Ufficio>
    </risiedeIn>
    <lavoraPer>
        <Team>
            <codice>TEA00336</codice>
            <nome>Pontepi-TeamA</nome>
            <tipoProgetto>Sondaggio Telefonico</tipoProgetto>
            <haRisorseIn>
                <Ufficio>
                    <codice>A301</codice>
                    <sede>Firenze 3</sede>
                    <ubicazione>3-01-nord</denominazione>
                </Ufficio>
            </haRisorseIn>
        </Team>
    </lavoraPer>
</Dipendente>
```

Risposta 5)

Coerentemente con la rappresentazione XML, si codifica la relazione molti a molti dall'azienda al quartiere, seguendo il verso della relazione in UML.

```
<!DOCTYPE Dipendente [
        <!ELEMENT codFisc       (#PCDATA)>
        <!ELEMENT nome      (#PCDATA)>
        <!ELEMENT dataAssunz        (#PCDATA)>
        <!ELEMENT ruolo     (#PCDATA)>
        <!ELEMENT lavoraIn      (Ufficio)>
        <!ELEMENT lavoraPer       (Team)>
        <!ELEMENT Ufficio       (codice, sede, ubicazione)>
        <!ELEMENT codice       (#PCDATA)>
        <!ELEMENT sede      (#PCDATA)>
        <!ELEMENT ubicazione        (#PCDATA)>
        <!ELEMENT Team      (codice,nome,tipoProgetto,haRisorseIn+)>
        <!ELEMENT tipoProgetto       (#PCDATA)>
        <!ELEMENT haRisorseIn       (Ufficio)>
]>
```