22 giugno 2009

Linguaggi di Programmazione

Nome e Cognome	
Corso di laurea	

1. Specificare la grammatica BNF di un linguaggio SQL-like per l'interrogazione di database relazionali, come nel seguente esempio di programma:

```
use anagrafe;

SELECT *
FROM Persone
WHERE citta = 'firenze';

SELECT nome, cognome, professione
FROM Persone;

use statistiche;

SELECT a+b, c*(d+e-4), f/g
FROM Parametri, Distribuzioni
WHERE a = b+1 AND b > 10 OR (a > 0 AND f < 100+c-d);

SELECT a, b, c
FROM (SELECT c, d FROM Sondaggi WHERE d > 0) AS Sond, Medie
WHERE c > 0 AND d = h;
```

L'istruzione **use** apre un database al quale fanno riferimento le interrogazioni successive. L'apertura di un nuovo database rimuove l'accesso al database corrente ed abilita l'accesso a quello nuovo. Ogni interrogazione è composta dal pattern **SELECT-FROM-WHERE**, in cui la clausola **WHERE** è opzionale. La clausola **SELECT** specifica una lista di espressioni aritmetiche su nomi di attributi e costanti intere, oppure il metacarattere '*****' per indicare tutti gli attributi in gioco. La clausola **FROM** specifica una lista di tabelle, in cui ogni tabella è identificata da un nome eventualmente preceduto da una interrogazione tra parentesi e dalla keyword **AS**. La clausola **WHERE** specifica una espressione booleana che coinvolge gli operatori logici **AND** e **OR** applicati a operazioni di confronto (mediante gli operatori **=**, **<**, **>**) tra espressioni aritmetiche.

2. Specificare la semantica operazionale della seguente espressione relazionale di estensione di una tabella:

extend *T*
by *a* **=** *E*
where *p*

(*esempio di frase*)

```
extend Prodotti
by netto = lordo - tara
where costo > 100
```

Note:

- Il nome del nuovo attributo non può coincidere con il nome di un altro attributo;
- Il nuovo attributo viene inserito in coda allo schema della tabella;
- Le tuple che non soddisfano il predicato della clausola **where** vengono estese con il valore **NULL**;
- L'espressione di estensione non altera la tabella operando, ma genera una nuova tabella.

3. Definire nel linguaggio *Scheme* la funzione **`(revatoms L)`** che, ricevendo in ingresso una generica lista **`L`**, computa la lista speculare degli atomi di **`L`**, come nei seguenti esempi:

`L`	**`(revatoms L)`**
`()`	`()`
`(a b)`	`(b a)`
`(x () y (1 2 (c d)) z)`	`(z y x)`
`((a b c))`	`()`

4. Definire nel linguaggio *Haskell* mediante pattern-matching la funzione **`medie`** (protocollo incluso), avente in ingresso una lista **`F`** di funzioni, una lista **`G`** di funzioni ed una lista **`R`** di numeri reali. Le tre liste hanno lo stesso numero $n \geq 0$ di elementi. La funzione **`medie`** genera una lista di n numeri reali m_i così definita:

$$\forall i \in [1..n] \;\; (m_i = (\mathbf{F}[i](\mathbf{R}[i]) + \mathbf{G}[i](\mathbf{R}[i])) / 2)$$

5. Specificare nel linguaggio *Prolog* il predicato **`positivi(N,P)`**, che risulta vero qualora **`P`** sia la sottolista dei numeri positivi (maggiori di zero) della lista di numeri **`N`**.

6. Illustrare il meccanismo di interpretazione di un programma *Prolog*.