

Lab Lesson 09

Giacomo Bergami

March 20, 2018

Mini Database

Si vuole creare un programma che legga da file due tipi di file: un file di **dati** ed un file di **interrogazioni**.

Il file di **dati** fornisce una rappresentazione tabulare, dove ogni colonna è separata da una virgola (tranne l'ultima colonna), ed ogni riga è delimitata dalla usuale andata a capo. La prima riga contiene sempre un'*intestazione*, ovvero definisce il nome di ciascuna colonna. Tutte le righe rimanenti contengono dei valori numerici, sempre separati da virgole. Ad esempio, il file in Figura 1 dovrà essere caricato in memoria come la tabella in Figura 2.

Il file di interrogazione fornisce invece un modo per manipolare i **dati** caricati in memoria, e per generare nuove tabelle. Il database è inizializzato con la tabella vuota senza intestazione. Ogni comando di tale linguaggio di interrogazione è scritto in una linea differente. La lista completa dei comandi è fornita qui sotto.

- **LOAD filename**

Carica una tabella **dati** da un file salvato nella posizione **filename**. Se il file non esiste, rimane in memoria la tabella precedentemente caricata. Una tabella senza intestazione (e quindi, senza righe) è una stringa di lunghezza zero.

A,B,C	A	B	C
1,2,3	1	2	3
1,2,4	1	2	4
4,5,6	4	5	6
4,5,7	4	5	7
5,5,7	5	5	7
7,8,7	7	8	7
7,11,23	7	11	23

Figure 1: File **dati**

Figure 2: Rappresentazione tabulare

A	B
1	2
1	2
4	5
4	5
5	5
7	8
7	11

Figure 3: **project A,B**

A	B	C
7	8	7
7	11	23

Figure 4: FiLteR B > 5

A	K
1	4
4	10
5	5
7	19

Figure 5: GROUP B by A wiTH + as K

- PRINT

Stampa con `System.out.println` la rappresentazione della tabella in memoria come in Figura 1.

- DUMP filename

Salva la stessa stringa che viene stampata dal metodo precedente in un file denominato filename.

- COUNT

Stampa a terminale una nuova linea contenente il numero di righe della tabella in memoria.

- PROJECT a_1, \dots, a_n

Aggiorna la tabella caricata in memoria selezionando quali colonne devono essere restituite. Tali colonne vengono selezionate tramite il loro nome. Osserva, uno stesso nome di colonna non deve essere ripetuto; se così fosse, non si applica la proiezione e la tabella corrente rimane quella precedentemente caricata (Figura 3). Se esiste un a_i non presente nell'intestazione, a_i viene semplicemente non restituito nella tabella finale: se non esiste nessuna colonna, si può restituire una tabella vuota.

- FILTER a_i op j

La tabella correntemente caricata in memoria mantiene solamente quelle righe le cui celle corrispondenti alla colonna di nome a_i hanno un valore k , tale per cui la valutazione di " k op j " è vera. In particolare, si vogliono interpretare solo gli operatori op di confronto $<$, $>$, $<=$, $>=$, $=$ e $!=$. Se la colonna a_i non è presente nella tabella, si restituisce la tabella vuota con l'intestazione precedente.

- GROUP a_i BY a_j WITH op AS a_k

Se a_i e a_j esistono nella tabella corrente e se $a_j \neq a_k$, restituisce una nuova tabella \tilde{t} con intestazione $\{a_j, a_k\}$; altrimenti, si restituisce la tabella vuota senza intestazione.

Per ottenere \tilde{t} , nella tabella già caricata si "raggruppano" tutte le righe che hanno lo stesso valore numerico k per a_j e, su tutti i valori di a_j , si effettua l'operazione

`op`, ottenendo v . `op` può essere un'operazione di somma su array o di prodotto su array. Per ogni valore k , si restituisce la riga $\{k, v\}$.

Ciascuna della parte dei comandi scritti in maiuscolo deve essere valutata in modo case insensitive.

Si vuole inoltre definire una classe **Client** che interpreti i comandi scritti da terminale come un *driver*, ed una **Interpreter** che esegua questi comandi quando scritti su file. Si pensi ad attuare una politica di riuso del codice, che consenta di non replicare il codice due volte. Si utilizzi quindi una classe madre/padre **MiniQueryParser**, che venga estesa da entrambe le classi. . .

Come autovalutazione, questa esercitazione vi fornisce tre cartelle ed una classe:

- La cartella **table** contenente tre semplici tabelle
- La cartella **query** contenente otto esempi di interrogazioni.
- La cartella **expected** contenente il risultato atteso stampato da terminale dal vostro codice.
- La classe **TestCorrectTable**, che effettua l'autovalutazione.

Si fornisce inoltre un Makefile per snellire la procedura di compilazione (**make**), pulitura dei file class inutili (**make clean**) e dell'esecuzione del codice (**make run**).