Relazione Progetto

I Struttura Concettuale

Abbiamo deciso di affrontare il progetto suddividendo il lavoro in elementi concettualmente separabili, in particolare abbiamo individuato tra questi elementi gli utenti, i post, il grafo per organizzare gli utenti e le loro relazioni, cercando quindi di rispecchiare il più possibile il paradigma della programmazione ad oggetti.

A Utenti

Come da consegna tutti gli utenti condividono alcune caratteristiche fondamentali e poi ne specificano altre differenziandosi in base alla tipologia.

Abbiamo dunque implementato una classe User e tre sottoclassi figlie (StandardUsr, BusinessUsr, GroupUsr) introducendo le varie caratteristiche aggiuntive. In questo modo tutti gli utenti sono User.

B Post

I post non si differenziano concettualmente tra di loro, e per questo sono implementati in un'unica classe, potenzialmente ampliabile a piacimento e con facilità.

C Grafo Template (TGraph)

Per organizzare i dati la scelta migliore si è rivelata quella di utilizzare una struttura dati a grafo, che permette di collegare tra loro nodi (utenti e post) tramite archi (relazioni) opportunamente scelti. Si è scelto di adottare l'approccio Template per questa struttura, in modo da renderla il più possibile generica e flessibile. Il grafo implementato è di tipo **direzionale**, per poter gestire relazioni unidirezionali come la *Conoscenza* di un individuo.

D Set Di Dati

I dati manipolati sono set di informazioni coerenti, come utenti, post e relazioni tra i vari elementi. Abbiamo quindi deciso di implementare una classe *DataSet* che permette di gestire ed analizzare un set di dati, secondo le specifiche del progetto.

Questa soluzione garantisce la possibilità di ampliare il programma facilmente permettendo di lavorare su set di dati diversi e separati, anche contemporaneamente ed eventualmente fare confronti ed analisi tra set di dati differenti. Questa parte non è stata implementata perché molto lontana dalle richieste del progetto.

E Avvio del programma

All’avvio del programma vengono automaticamente caricati eventuali files presenti all’interno della cartella “input”. Il loro nome è passato da linea di comando. In questo processo si verifica la correttezza in termini di numero di argomenti presenti e apertura riuscita dei files specificati. In caso affermativo, tutti i dati vengono caricati da opportuni metodi, che controllano la corretta formattazione, per poi aggiungere nodi o archi corrispondenti se considerati validi.

F Menù di controllo

L’elemento più vicino all’utente è l’interfaccia di controllo, che permette di eseguire tutte le operazioni di base (aggiungere, modificare, rimuovere dati) e funzioni avanzate: statistiche di vario genere, possibilità di esportare i dati su file.

L’interfaccia è organizzata a sezioni. Dal menù principale è possibile esportare i dati su file, stampare a video una panoramica del set di dati, visualizzare informazioni relative ad un utente tramite username (informazioni correlate e post pubblicati). È inoltre possibile accedere ai sottomenù di: modifica, statistica e ricerca.

1 Menù di modifica

In questo menù sono previste funzioni per aggiungere, rimuovere o modificare uno tra utenti, relazioni o post. Tutte queste operazioni sono sottoposte ai necessari controlli.

2 Menù di statistiche

In questo menù è possibile analizzare i dati nel loro complesso ricercando, come da consegna: utenti totali, amici e parenti per ogni utente, post con più like/dislike…

3 Menù di ricerca

Questo menù permette di accedere alle funzioni di analisi dati più elaborate, che sfruttano la struttura a grafo del set di dati. Esse sono: la stampa di alberi genealogici, la ricerca di “lupi solitari” e la ricerca per simpatia delle aziende.

II Organizzazione dei Dati

I dati, siano essi acquisiti da file o aggiunti tramite interfaccia, sono tutti organizzati in una struttura a grafo direzionale implementata nel *TGraph*. Gli utenti sono collegati tra di loro da *Relazioni* di varia natura (conoscenza, amicizia, parentela…).

Abbiamo inoltre deciso di utilizzare struttura analoga anche per i *Post*: anche questi ultimi sono salvati in una struttura a grafo della stessa natura di quella precedente. I post sono quindi collegati agli utenti di nuovo attraverso delle relazioni di vario tipo (proprietà, like, dislike).

L'utilizzo di un grafo per i *Post* offre prospettive interessanti come la possibilità di implementare relazioni tra gli stessi, come nel caso di un commento ad un post già esistente.

Adottando questa soluzione, alcuni problemi di analisi dati risultano semplificati, e non richiedono di analizzare per forza tutti i dati in nostro possesso per ottenere il risultato.

Essa inoltre offre anche degli iteratori che permettono di poter scandire tutti i dati al suo interno.

III Algoritmi e strutture utilizzati

A La mappa (map)

Questo contenitore template appartenente alla STL permette di associare un dato generico ad un'etichetta generica, a patto che sia presente un operatore < per questa etichetta.

All'interno della mappa, i dati sono organizzati secondo il modello dell'albero binario, permettendo una ricerca ottimizzata degli elementi.

La mappa ha inoltre a diposizione una serie di iteratori che permettono di scandire tutti gli elementi in essa contenuti.

B Il grafo (TGraph)

Il grafo template che abbiamo implementato si basa proprio sulle mappe. I nodi sono immagazzinati in una mappa (_nodes) che associa ad ogni nodo un'etichetta univoca. Le relazioni tra i nodi sono contenute in una matrice di tipo sparso realizzata per mezzo di una mappa di mappe (_adj).

TGraph espone un set di metodi pubblici per aggiungere nodi e relazioni, verificarne l'esistenza, nonché un metodo di ricerca basato sulla BFS.

C Altre classi implementate

1 User

Questa classe contiene le informazioni base di un utente che permettono di identificarlo univocamente e distinguerne il tipo. Inoltre, in questa classe sono contenuti dati comuni a tutti i tipi di utente, come la data e l'indirizzo.

Da questa classe sono derivate: StandardUser, BusinessUser, GroupUser.

2 Date

Oltre a immagazzinare una data, questa classe contiene metodi per leggere una stringa e estrarne una data in modo corretto, segnalando un eventuale errore. Sono stati inoltre implementati gli overload dell'operatore < per eseguire confronti e ordinamenti e dell'operatore << per poterla stampare con facilità.

3 Relation

Relation è una struttura le cui istanze vengono utilizzate come archi per collegare i nodi. In questo modo l'arco stesso è portatore di informazione sull'entità della stessa.

IV Controlli

Esistendo tre files di input differenti è stato necessario implementare controlli di varia natura.

In generale gli errori da gestire si presentano in due categorie; la prima è quella di formattazione non coerente con lo standard adottato, la seconda riguarda l'inserimento di dati che non soddisfano i requisiti.

1 Formattazione

Per ogni file vengono dapprima eseguite dai metodi di *load* alcune routines atte a verificare che il formato utilizzato nei files di input sia aderente alle specifiche. Viene ad esempio controllata la presenza e l'ordinamento dei delimitatori.

2 Inserimento

Prima di ogni inserimento dei dati viene verificato che i campi non siano vuoti né presentino incongruenze (es.: tipo non riconosciuto, data priva di senso…). In questa fase le funzioni di *add* verificano che non vengano inseriti dati duplicati o relazioni impossibili (es.: l'amicizia di un privato con un'azienda, anelli parentali non possibili…).

V Files di input a disposizione

Nella cartella *input* sono presenti due set di files.

Il set 1 contiene una coerente raccolta di dati privi di errori. Caricando questi files è possibile ottenere una dimostrazione dell'esecuzione del programma.

Il set 2 presenta un ampio insieme di casistiche di errore in ogni file di input. Caricando questi files si potrà, via via, osservare la reazione del programma a diversi tipi di fault.

È allegato un file che riporta per ogni riga (in ogni file) il tipo di errore.