Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria "Enzo Ferrari"

Corso di Laurea in Ingegneria Informatica

Progetto ed implementazione di un database per la gestione di metadati di campioni/esperimenti genomici

Relatore:
Chiar.mo Prof. Domenico Beneventano

Candidato:
Federico Bolelli

Anno Accademico 2013-2014

Parole chiave:

Espressione genica
Data Integration
Microarray
Database
MySQL
Talend
ETL
Tag

Ringraziamenti

Si ringrazia il Prof. Ing. Domenico Beneventano, docente presso il Dipartimento di Ingegneria "Enzo Ferrari", per la disponibilità, il tempo e l'impegno concesso. Si ringraziano, inoltre, il Prof. Ing. Silvio Bicciato, docente presso la Facoltà di Scienze della Vita dell'Università degli Studi di Modena e Reggio Emilia, i suo collaboratori presso il Centro Interdipartimentale di Ricerche Genomiche – CeIRG e il Prof. Mauro Leoncini, docente presso il Dipartimento di Scienze Fisiche, Informatiche e Matematiche della medesima università.

Indice

Lista delle Figure

Lista delle Tabelle

1 Introduzione

1.1 Motivazioni

Il recente sviluppo delle tecnologie del silicio, in concomitanza con le recenti scoperte in ambito biomedico e con lo sviluppo delle nanotecnologie, ha permesso la nascita di strumenti sempre più complessi per l'analisi di campioni di DNA e, più precisamente, di sequenze geniche e di sequenziamento del DNA. La quantità di dati descrittivi generati da esperimenti di questo tipo cresce esponenzialmente nel tempo e, per risultare utile in ambito medico, necessita di importanti strumenti di memorizzazione ed elaborazione. Il recupero, l'organizzazione e l'utilizzo delle informazioni descrittive dei campioni (meta-dati) è un passaggio estremamente critico che, se non condotto con molta cautela, può inficiare l'intera analisi. Infatti, le meta-informazioni determinano l'appaiamento corretto tra i file dei dati grezzi, generati dagli esperimenti, e gli identificativi dei campioni, e quindi, di fatti, il risultato della meta-analisi.

Vista l'importanza dei dati in questione e dei risvolti biomedici che da essi potrebbero scaturire, esistono banche dati internazionali (chiamate *microarray database*) che raccolgono le informazioni riguardanti i campioni ed i relativi *metadata*, consentono la ricerca e rendono disponibili i dati per analisi ed interpretazione. I più importanti *microarray database* sono Gene Expression Omnibus e Array Express.

Purtroppo, il recupero e l'organizzazione efficiente dei metadati sono spesso complicati da lacune di annotazione nelle banche dati, da relazioni non sufficientemente esplicite tra campioni biologici, caratteristiche fenotipiche e dati grezzi e, infine, da procedure di gestione informatica non ottimizzate (addirittura manuali) e quindi facilmente soggette alla generazione di errori.

Il presente lavoro di tesi, svolto in collaborazione con il Centro Interdipartimentale di Ricerche Genomiche – CeIRG, dell'Università degli Studi di Modena e Reggio Emilia, si pone come obiettivo quello di progettare ed implementare (nelle funzionalità di base) uno strumento atto a semplificare la gestione di metadati di espressione genica, all'interno del centro di ricerca. L'esigenza di un applicativo specifico deriva dalla necessità di disporre di strumenti *ad-hoc* che permettano la gestione flessibile ed integrata di metadati provenienti da *microarray database* e definiti localmente dal centro di ricerca, nonché dall'esigenza di trattare i dati secondo metodi ed algoritmi costruiti *ad-hoc* e differenti da quelli preesistenti.

1.2 Elementi fondamentali

Il progetto prevede quattro elementi fondamentali. Questi sono brevemente presentati nel presente paragrafo e saranno sviscerati nei successivi capitoli:

1. Modellazione e implementazione di un database che rispetti le specifiche dettate dal gruppo di ricerca, ma che al contempo permetta l'inserimento di (meta)dati provenienti da *microarray* database pubblici. Requisito fondamentale affinché quest'ultimo punto sia rispettato è che la struttura della nuova base di dati

rispecchi, almeno in parte, quella dei sistemi pubblici che si prenderanno in considerazione.

2. Creazione di uno strumento che permetta la connessione automatica ai server FTP contenenti i dati pubblici, e il conseguente download degli stessi in funzione delle richieste dell'utente.
3. Realizzazione, tramite strumenti ETL (Extract, Transform, Load), di un meccanismo per la manipolazione dei dati precedentemente scaricati e il caricamento degli stessi sul database di cui al punto uno. Lo strumento ETL scelto a tale scopo è Talend Open Studio.
4. Creazione di un'opportuna GUI per fornire un semplice interfacciamento con il database e con gli strumenti citati ai punti 2 e 3.

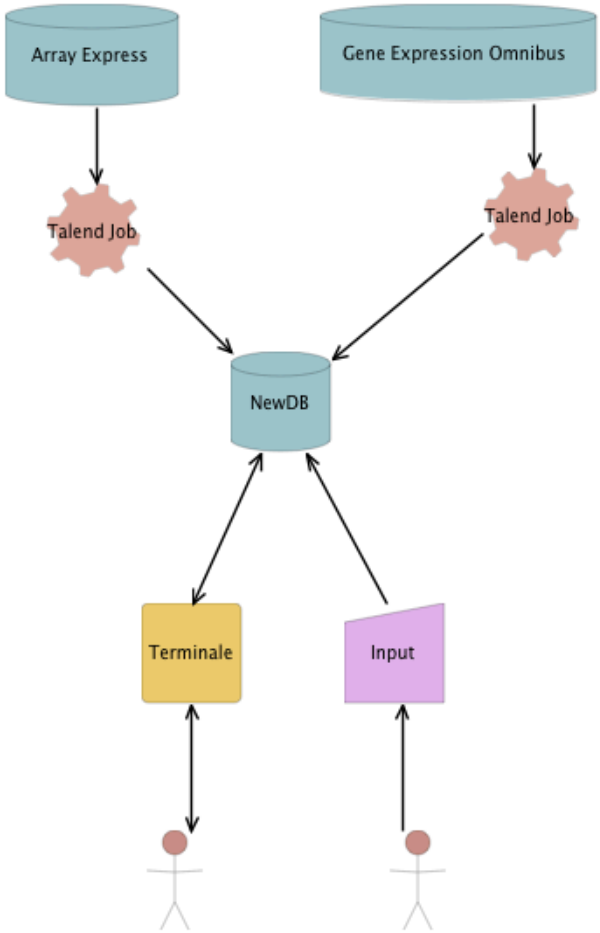

Figura 1: Business Model

La figura in alto mostra il modello di business e viene riportata con l'intento di inquadrare e riassumere, seppur in maniera semplificata, l'intero progetto. Il lavoro di tesi si concentrerà solamente sui primi tre elementi presentati.

1.3 Organizzazione della tesi

Il Capitolo 2 consiste di una breve introduzione biologica ai principi di funzionamento dei *microarray*: strumenti utilizzati per l'analisi trascrizionale. Il contenuto di questo capitolo è stato estratto dalle lezioni corso *"Laboratory of Bioinformatics"* tenuto dal Prof. Bicciato presso la facoltà di *Biotecnologie Industriali* dell'Università degli Studi di Modena e Reggio Emilia, nonché dai testi "*Immagini della biologia"* e *"Biologia"* (si consulti la bibliografia per maggiori dettagli).

Nel Capitolo 3 saranno analizzati nel dettaglio i più importanti *microarray database* e sistemi preesistenti di annotazione e manipolazione dei metadati in essi riportati. In particolare nel paragrafo 3.1 verranno presentati i protocolli di annotazione dei metadati, nei paragrafi 3.2 e 3.3 si mostrerà nel dettaglio lo schema concettuale alla base, rispettivamente, di Gene Expression Omnibus ed Array Express. Nei paragrafi 3.4 e 3.5 sarà analizzata un'applicazione web per il recupero, l'organizzazione e la meta-analisi di dati di espressione genica contenuti nella banca dati Gene Expression Omnibus (GEO) e saranno redatte le soluzioni implementative adottate.

Il Capitolo 4 si compone di una dettagliata analisi di progettazione del database, cuore dell'intero progetto. Tale progettazione terrà conto sia della struttura di base dei suddetti *microarray* database, che delle esigenze addizionali espresse dal centro di ricerca.

Nel Capitolo 5 verranno presentate le caratteristiche peculiari del protocollo ftp, i dettagli relativi ai server portatori dei metadati di interesse e le soluzioni adottate per il reperimento semiautomatico dei dati.

Nel Capitolo 6 si descriveranno i principali software di ETL (Extract, Transform, Load) e il loro funzionamento di base. In particolare, verranno presentate le caratteristiche peculiari di Talend Open Studio in relazione all'utilità che questo ha per il progetto di tesi.

Infine, nel Capitolo 7, si riportano le conclusioni e i futuri sviluppi del progetto.

2 I microarray

2.1 Introduzione

Nel 1975 Edward Southern ideò una tecnica di biologia molecolare (Southern Blot) per rivelare la presenza di specifiche sequenze di DNA in una miscela complessa. Il limite di tale tecnica era sostanzialmente legato al fatto che le sequenze rilevabili mediante un singolo esperimento erano poche, e il numero di sequenze geniche presenti in un organismo, seppur dipendente dal tipo di organismo (circa 6'000 per un lievito, 18'000/20'000 per un essere umano) è estremamente elevato. Negli anni, anche grazie allo sviluppo delle tecnologie di lavorazione del silicio, assistiamo alla nascita di nuove tecniche che permettono una rilevazione completa, più o meno precisa, delle sequenze di DNA/RNA. Nel 1996 nascono i Gene Chip (*microarray*): insieme di microscopiche sonde di DNA attaccate ad una superficie solida come vetro, plastica, o chip di silicio. I *microarray* permettono di esaminare simultaneamente la presenza di moltissimi geni all'interno di un campione di DNA o RNA (che spesso può rappresentare anche tutto il genoma di un organismo).

Un utilizzo tipico di queste tecniche, è quello di confrontare il profilo di espressione genica di un individuo malato con quello di uno sano per individuare quali geni sono coinvolti nella malattia. Tuttavia, lo strumento può essere impiegato in numerosi altri campi come, ad esempio, test sull'identità umana, sulle qualità del cibo o ancora in esami di diagnostica o di classificazione di bestiame.

Cercherò, nei paragrafi che seguono, di spiegare brevemente, e senza pretesa di esaustività, il funzionamento della tecnologia, in modo che il lettore possa cogliere da un lato l'utilità e i principi di base che ne regolano il funzionamento e, dall'altro, la necessità di fornirsi di un sistema informatico per il suo "utilizzo".

2.2 Struttura di DNA e RNA

L'acido desossiribonucleico o deossiribonucleico (DNA) è un acido nucleico che contiene le informazioni genetiche necessarie alla biosintesi di RNA e proteine, molecole indispensabili per lo sviluppo ed il corretto funzionamento della maggior parte degli organismi viventi.

Dal punto di vista chimico, il DNA è un polimero organico costituito da monomeri chiamati nucleotidi. Tutti i nucleotidi sono costituiti da tre componenti fondamentali: un gruppo fosfato, il deossiribosio (zucchero pentoso) e una base azotata. Le basi azotate, che possono essere utilizzate nella formazione dei nucleotidi da incorporare nella molecola di DNA, sono quattro: adenina, guanina, citosina e timina. Il DNA può essere più correttamente

Figura 2: Accoppiamento basi azotate nel DNA

definito come una doppia catena polinucleotidica (A,T,C,G), antiparallela, orientata, complementare, spiralizzata e informazionale. L'ordine della disposizione sequenziale dei nucleotidi costituisce l'informazione genetica.

L'RNA (acido ribonucleico) è chimicamente molto simile al DNA ma differisce da quest'ultimo per vari motivi:

- contiene lo zucchero ribosio anziché il deossiribosio (da qui il nome);
- una delle basi azotate, la timina (T), è sostituita dall'uracile (U);
- le molecole di RNA sono solitamente a singolo filamento.

Come si può chiaramente osservare in Figura 2 l'accoppiamento nucleotidico è forzato: la base azotata adenina, ad esempio, può interfacciarsi unicamente con la timina e via così. Un discorso analogo, con le differenze dovute alla struttura di base, vale per l'RNA. Questo è il principio chiave che regola il funzionamento di tutti, o quasi, i dispositivi di rilevazione genica.

2.3 Tecnologie di rilevazione genica

Le tecnologie di rilevazione genica sfruttano, come principio di base per il loro funzionamento, una caratteristica fondamentale degli acidi nucleici sopra descritti: quando un filamento di acidi nucleici incontra il suo complementare questi si appaiano, ovvero portano alla formazione di un acido nucleico a doppio filamento a partire da due molecole a singola elica. Prendiamo ad esempio la sequenza nucleotidica ACTG. Nel momento in cui questa incontra la sua complementare TGAC esse si legano. Supponiamo che la parola ACTG, intrappolata secondo una qualche tecnica su di un supporto solido, rappresenti il complementare dell'informazione genica cercata in un determinato campione. A questo punto poniamo in contatto il campione, precedentemente trattato con una sostanza marcante di tipo radioattivo fluorescente (che risponde ad una data frequenza), con il supporto solido. Al termine dell'esperimento possiamo eccitare con la giusta frequenza la superficie solida su cui era stata intrappolata la parola ACTG: se questa "risponde" con un impulso luminoso, possiamo affermare che l'informazione genica cerata (TGAC) era presente all'interno del campione in esame; non solo, a seconda dell'intensità luminosa con cui il supporto risponde è possibile, tramite specifici algoritmi, risalire alla densità di presenza della parola nel campione. Nel seguito sarà più chiaro il perché.

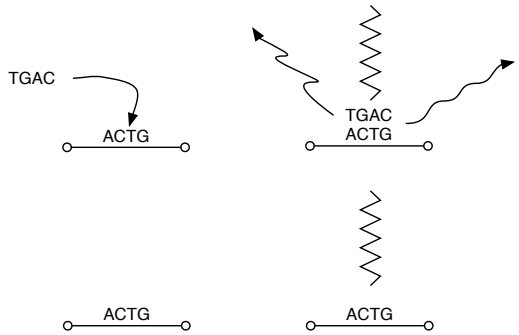

Figura 3: Principio generale di funzionamento delle tecnologie di rilevazione genica

2.3.1 Struttura e principio di funzionamento dei microarray

Come già accennato nei paragrafi precedenti e come suggerito dal nome, i *microarray* o *gene chip*, sono caratterizzati da una struttura solida suddivisa in celle, all'interno delle quali sono intrappolate determinate sequenze nucleotidiche. Ogni cella del supporto contiene più volte una determinata sequenza nucleotidica: questo permette di ridurre il rischio di errori dovuti all'incorretta immobilizzazione di una particolare sequenza sul supporto e, al contempo, permette di stimare, oltre che la presenza, anche la frequenza con cui una determinata parola si presenta all'interno del campione in esame. Attraverso appositi macchinari si esegue l'operazione di sovrapposizione del campione al *microarray* e, successivamente, tramite uno scanner, si procede all'eccitazione della superficie e alla memorizzazione dell'immagine di "risposta". Il risultato che si ottiene è quello rappresentato dal codice A in figura 5. Lavorare con informazioni di questo tipo risulta complesso, quindi, in seguito all'acquisizione dell'immagine, si procede al calcolo della "probabilità" di cella: ovvero, mediante un'analisi di tipo statistico si stabilisce il valore medio di intensità della cella (Figura 5 - B).

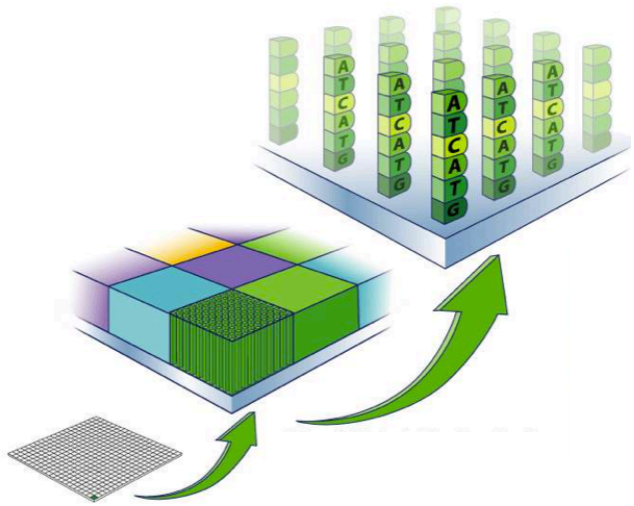

Figura 4: Particolare di una cella del microarray

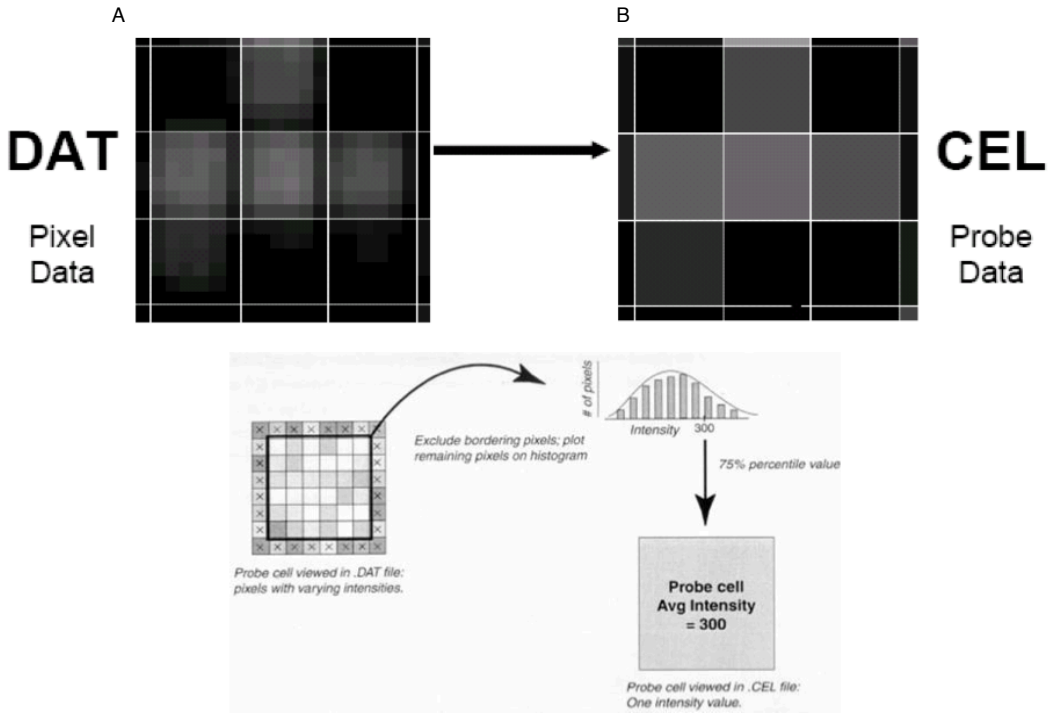

Figura 5: Particolare della conversione da immagine dello scanner a file grezzi.CEL

2.3.2 Tecniche di costruzione e di utilizzo

Le tecniche di costruzione dei microarray, così come quelle di quantificazione dei livelli di sequenza genica, sono molteplici.

Come già detto i *microarray* sono costituiti da un supporto solido che può essere di vario tipo: filtro di nylon, vetrino da microscopia o buffer di silicio. Su questi supporti vengono immobilizzate o costruite (sintetizzate) catene polinucleotidiche di lunghezza variabile dai 20 ai 100 nucleotidi (singolo filamenti di acidi nucleici).

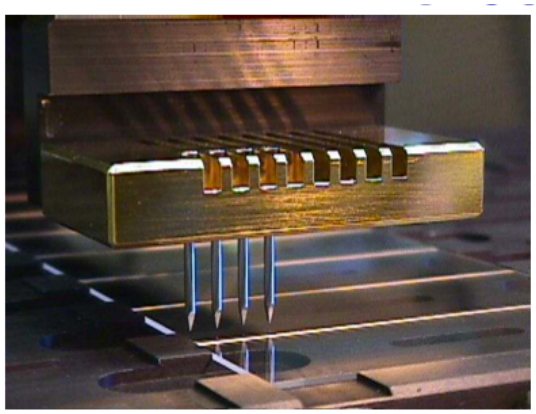

Figura 6: Particolare della costruzione di un microarray

Le tecniche di costruzione dei microarray possono essere suddivise in due categorie:

1. La prima tecnica consiste nel comprare una batteria prefabbricata di sequenze e nel depositare le stesse su di un sopporto solido che può essere di vario tipo;
2. La seconda, invece, consiste nel sintetizzate direttamente sul supporto solido le batterie di sequenze. Questo garantisce una maggiore precisione di realizzazione, a scapito di una maggiore complessità.

Lo stesso vale per le modalità di utilizzo, esse sono sostanzialmente due:

1. Intrappolare sulla matrice un'unica soluzione di RNA ottenuta combinando due popolazioni: la soluzione che vogliamo studiare e una sequenza di RNA di controllo. Sull'array avviene quindi la coibridazione di due soluzioni differente che dovranno essere precedentemente marcate con due fluorocromi differenti e poi miscelate. L'immagine che si ottiene è generata dalla compresenza del controllo e dell'esperimento (tecnica utilizzata in caso di supporto solito a vetrino);
2. In alternativa, è possibile ibridare sul supporto una sola sequenza di RNA, priva di controllo (tecnica utilizzata in caso di supporto solido a buffer di silicio).

Nel primo caso si ottengono due matrice di numeri, nel secondo, invece, una matrice unica.

2.3.3 Analisi dei dati

L'analisi dei dati viene eseguita su di un file di testo tabulato contenente, in numeri, le informazioni racchiuse nell'immagine .CEL (vedi Figura 5). Tale file contiene il livello di segnale di una data cella e le coordinate della stessa all'interno dell'array.

Analizzare i dati consiste nello stabilire, mediante opportuni algoritmi, quali geni si esprimevano in un determinato campione: le parole contenute all'interno delle celle, infatti, non sono rappresentative di un gene specifico ma di una sequenza che rappresenta solo una piccola parte dell'intera informazione genica. L'immagine seguente chiarisce il concetto appena spiegato.

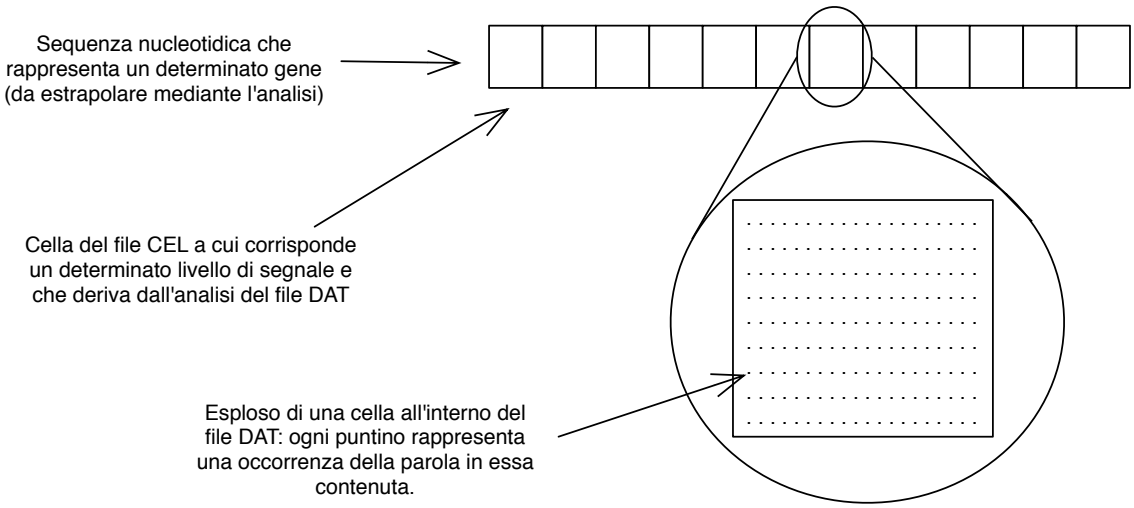

Figura 7: Particolare dei dati contenuti nei file .CEL e .DAT

3 Le sorgenti dei dati

L'enorme quantità di dati prodotti con la tecnologia dei *microarray* per l'analisi dell'espressione genica, ha imposto l'utilizzo di banche dati destinate a catalogarli e a renderli pubblici a beneficio della comunità scientifica e dello scambio fra gruppi di ricerca. Infatti, i profili di espressione genica, raccolti durante esperimenti progettati per studiare un determinato meccanismo biologico, contengono dati usufruibili per la verifica di ipotesi biologiche di vario tipo. Le banche dati a cui faremo riferimento per lo sviluppo del progetto, e su cui si basa il gruppo di ricerca, sono sostanzialmente due:

1. Gene Expression Omnibus (http://www.ncbi.nlm.nih.gov/geo/)
2. ArrayExpress (http://www.ebi.ac.uk/arrayexpress/).

Queste nascono con lo scopo di memorizzare sia i risultati di un esperimento (file .CEL) effettuato con le tecnologie precedentemente inquadrate, sia i metadati relativi all'esperimento stesso.

3.1 MIAME e MINSEQE

Con l'intento di facilitare lo sviluppo delle banche dati di *microarray* e di software di gestione degli stessi, nascono i protocolli MINSEQE[1] (http://www.mged.org/minseqe/) e MIAME[2] (http://www.mged.org/Workgroups/MIAME/miame.html). Il loro scopo è quello di descrivere le informazioni minime, relative ad esperimenti di *high-throughput sequencing nucleotidico*, che è necessario riportare per consentire un'interpretazione univoca dei dati, e facilitare la riproduzione dei risultati dell'esperimento.

MIAME e MINSEQE definiscono quindi le informazioni minime che devono accompagnare un esperimento effettuato con la tecnologia dei *microarray*. Tuttavia, essi non definiscono la struttura dei dati con cui tali informazioni devono essere riportate; non definiscono cioè il dizionario che deve essere utilizzato per esprimere i dati stessi. Questo si traduce in una sostanziale incongruenza nel formato dei dati che richiederebbe il ricorso ad algoritmi di data mining per l'estrazione di informazioni implicite.

[1] Minimum Information about a high-throughput SEQuencing Experiment.

[2] Minimum Information About a Microarray Experiment.

3.2 Gene Expression Omnibus

Gene Expression Omnibus (GEO), mantenuto da NCBI[3], è uno dei *microarray* database maggiormente utilizzati. GEO offre una piattaforma pre-costituita per immagazzinare i dati di espressione genica e, allo stato attuale, ospita un totale di 1.170.233 campioni organizzati in 48.729 esperimenti prodotti utilizzando 13.092 tipologie diverse di tecnologie *high-throughput*.
Il database è compatibile con il protocollo MIAME ed è caratterizzato dalla struttura riportata in Figura 8.

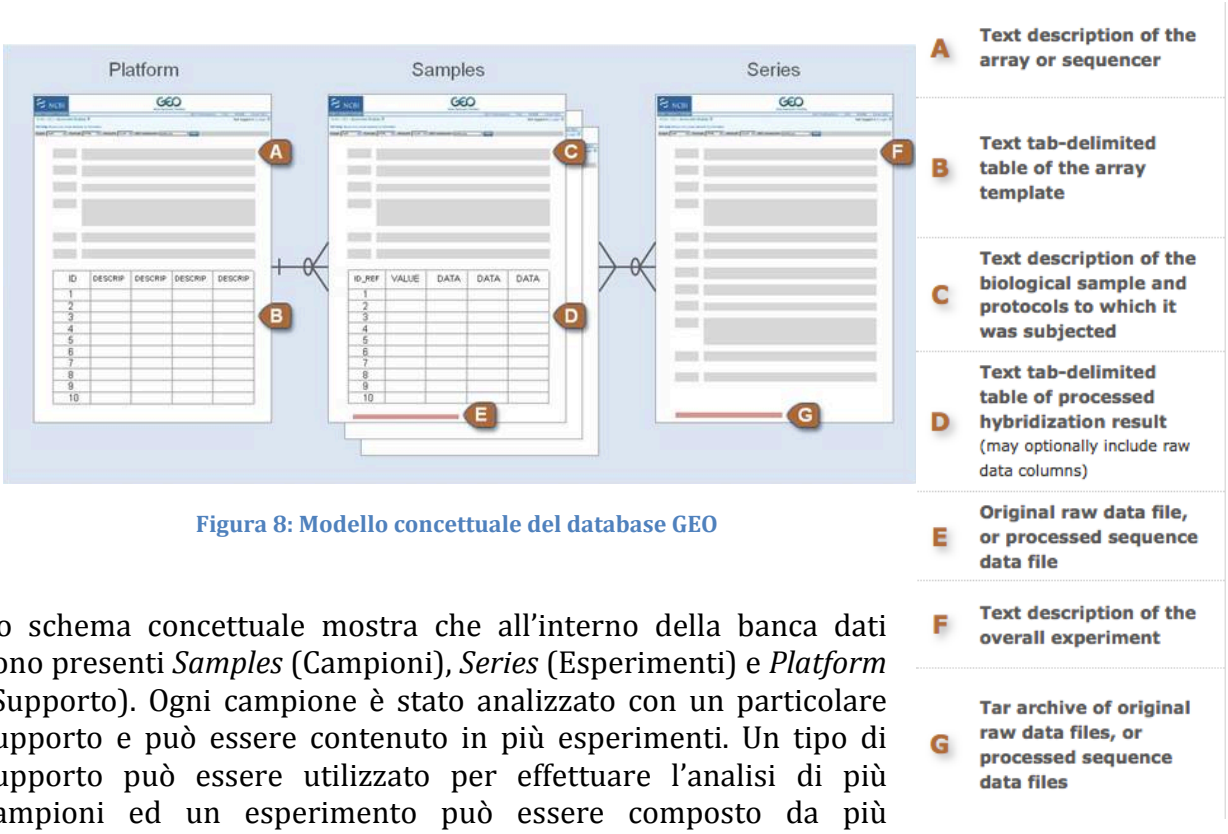

Figura 8: Modello concettuale del database GEO

Lo schema concettuale mostra che all'interno della banca dati sono presenti *Samples* (Campioni), *Series* (Esperimenti) e *Platform* (Supporto). Ogni campione è stato analizzato con un particolare supporto e può essere contenuto in più esperimenti. Un tipo di supporto può essere utilizzato per effettuare l'analisi di più campioni ed un esperimento può essere composto da più campioni.

3.2.1 Platform

Rappresenta il supporto fisico tramite cui è stata effettuata l'analisi di un determinato campione: i *microarray* non sono l'unico sistema di sequenziamento del DNA/RNA, inoltre, nel tempo, le strutture stesse dei *microarray* sono e possono cambiare, assumendo conformazioni completamente diverse; si rende quindi necessario tenere traccia dei dispositivi utilizzati per poter effettuare correttamente successive analisi.

[3] Il *National Center for Biotechnology Information* (NCBI), Centro Nazionale per le Informazioni Biotecnologiche, è una parte della National Library of Medicine (Biblioteca nazionale americana di medicina), che dipende a sua volta dall'Istituto per la salute americano. L'NCBI ha la sua sede a Bethesda (Maryland), ed è stato fondato nel 1988. Questo centro ospita e gestisce varie banche dati di genomica (GenBank), proteine e altre informazioni relative alle biotecnologie, nonché sviluppa strumenti e software per analizzare i dati del genoma. L'istituto rende anche disponibile un immenso database di citazioni di articoli scientifici, principalmente di carattere biomedico, con riferimenti agli articoli stessi, talvolta ad accesso libero (PubMed).

Platform si compone di una descrizione sintetica della matrice e, per le piattaforme basate su array, di una tabella di dati che definisce la matrice *template*: la geometria del supporto solido, ossia l'organizzazione spaziale delle sonde immobilizzate. Alle piattaforme viene assegnato un numero di accesso GEO univoco e stabile (GPLxxx). Una piattaforma può fare riferimento a molti campioni. Di seguito viene riportato un esempio di come l'applicazione web GEO potrebbe presentare una piattaforma.

Figura 9: Vista della piattaforma dall'applicativo web di GEO

3.2.2 Samples

Rappresenta il campione vero è proprio, descrive le condizioni in base alle quali esso è stato manipolato, le manipolazioni che esso ha subito, e i risultati ottenuti: quantificazione dei segnali raccolti. Ad ogni campione viene assegnato un numero GEO univoco e stabile

(GSMxxx). Un campione deve fare riferimento a una sola piattaforma e può essere incluso in più esperimenti. Di seguito è riportato un esempio di come l'applicazione web GEO può presentare un campione (Figura 10).

La descrizione completa di un campione si compone delle sue caratteristiche biologiche e dei protocolli sperimentali a esso applicati (meta-informazioni) e di una tabella di valori che rappresentano la quantificazione del segnale di ogni sonda della piattaforma di riferimento nel campione considerato. La tabella dei segnali può essere corredata da informazioni supplementari come, ad esempio, dai file che contengono i livelli di fluorescenza delle singole sonde prima della quantificazione (file .CEL).

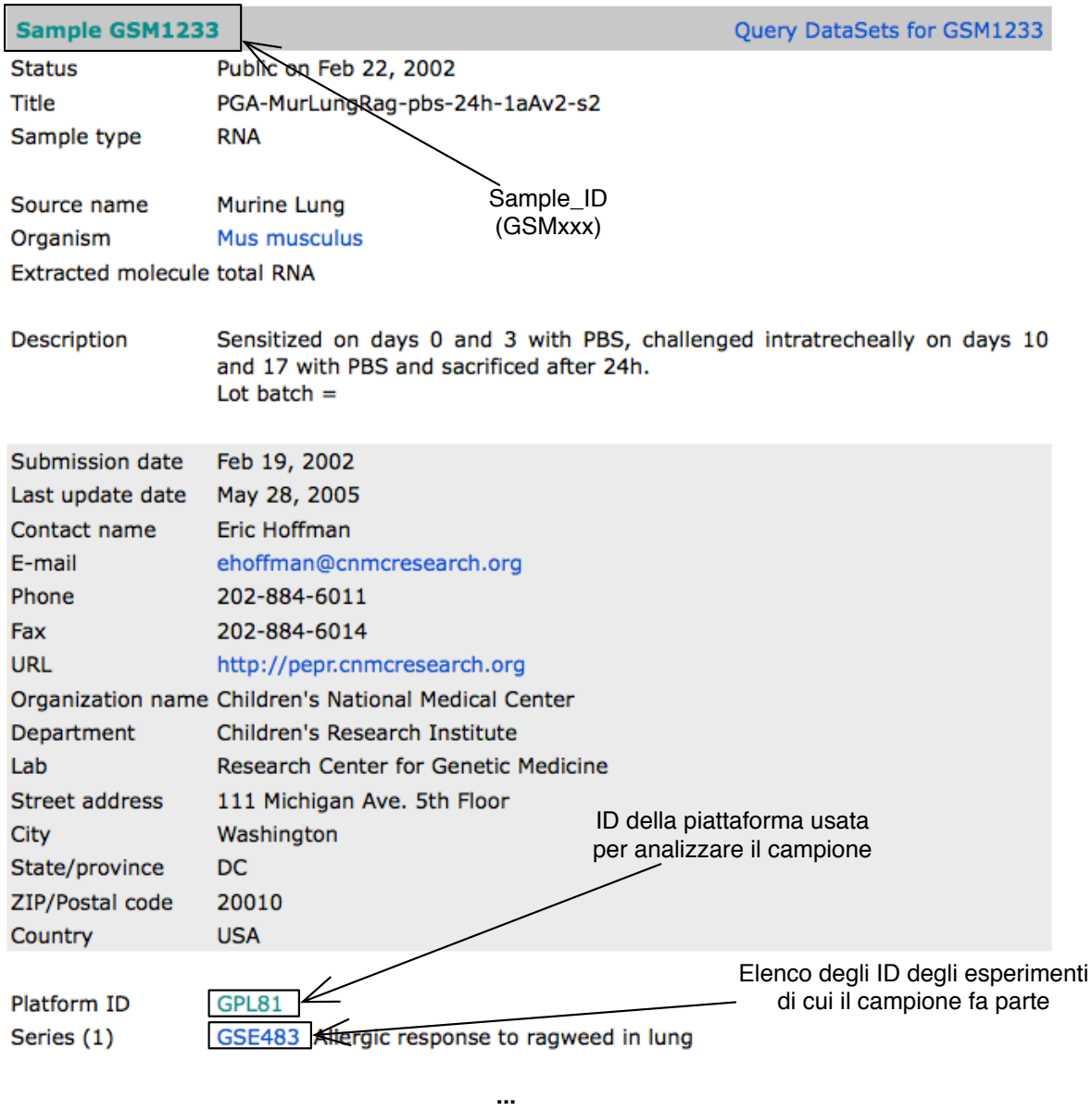

Figura 10: Vista del campione dall'applicativo web di GEO

3.2.3 Series

Rappresenta un esperimento, ne fornisce una descrizione complessiva ed è una collezione di campioni (*sample*). Ad ogni record della serie viene assegnato un numero di adesione GEO univoco e stabile (GSExxx). Di seguito un esempio (Figura 11).

Figura 11: Vista dell'esperimento dall'applicativo web di GEO

3.2.4 Dataset

Un GEO *DataSet* rappresenta un insieme di dati sperimentali elaborati, riassunti e classificati in base alle variabili sperimentali. A ogni GEO *DataSe*t è assegnato un identificativo univoco preceduto dal prefisso GDS. Grazie alla costruzione dei GEO *DataSet*, l'eterogeneità dei dati può essere notevolmente ridotta fornendo, contemporaneamente, all'utente un grafico che rappresenta il livello di espressione di ogni gene in ogni campione del set di dati e una prima analisi bioinformatica dell'intero esperimento.

3.2.5 Sottomissione e formato dei dati

Per la sottomissione dei dati a GEO, l'utente può avvalersi di moduli web, di fogli di calcolo, di un formato di testo semplice e di un formato XML. La scelta del formato dipende dal volume, dal tipo di dati da sottomettere e dal formato originale degli stessi. Il tipo di formato scelto per la sottomissione dei dati non è ovviamente importante, in quanto ciò che realmente conta è il rispetto degli standard imposti da MIAME. Una volta sottomessi, dati e metadati sono disponibili per il download. I formati disponibili sono sostanzialmente tre:

Download family	Format
SOFT formatted family file(s)	SOFT
MINiML formatted family file(s)	MINiML
Series Matrix File(s)	TXT

Figura 12: Formato dei dati

Il primo, SOFT (Simple Omnibus in Text Format), contiene dati e metadati in formato testuale ASCII.

Il secondo, MINiML (MIAME Notation in Markup Language) di maggior interesse per il progetto, è un linguaggio di markup strutturato secondo il formato XML e basato sul protocollo MIAME. La famiglia di file in formato MINiML riporta dati e metadati; in sostanza essi rappresentano un rendering XML del formato SOFT.

La definizione dello schema XML[4] MINiML è fornita dall'NCBI al seguente indirizzo http://www.ncbi.nlm.nih.gov/geo/info/MINiML.html.

Il terzo, ed ultimo, formato disponibile è un file di testo tabulato contenente la matrice dei valori ottenuti dai vari campioni dell'esperimento. Tutti i file presenti sul sito FTP di GEO sono compressi mediante gzip (estensioni .gz o .tgz).

[4] *XML (eXtensible Markup Language)* è un linguaggio di markup basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo. Il nome indica che si tratta di un linguaggio marcatore (markup language) estensibile (eXtensible) in quanto permette di creare tag personalizzati. Rispetto all'HTML, l'XML ha uno scopo ben diverso: mentre il primo definisce una grammatica per la descrizione e la formattazione di pagine web (layout) e, in generale, di ipertesti, il secondo è un metalinguaggio utilizzato per creare nuovi linguaggi, atti a descrivere documenti strutturati. Mentre l'HTML ha un insieme ben definito e ristretto di tag, con l'XML è invece possibile definirne di propri a seconda delle esigenze. L'XML è oggi molto utilizzato anche come mezzo per l'esportazione di dati tra diversi DBMS.

3.3 Array Express

ArrayExpress è un *microarray* database realizzato e mantenuto dall'EBI (*European Bioinformatics Institute*). In esso è possibile trovare dati relativi a esperimenti di espressione genica sottomessi dai gruppi di ricerca seguendo gli standard MIAME e MINSEQE ed ottenuti sia utilizzando piattaforme basate sui microarray sia da esperimenti di sequenziamento massivo. In questo momento, ArrayExpress contiene 1.466.864 campioni provenienti da 50.536 esperimenti diversi.

Così come GEO, anche in ArrayExpress, i dati sono organizzati in tre categorie principali: array, esperimenti e protocolli. Ad ognuna di esse corrisponde un identificativo specifico ed univoco. L'infrastruttura informatica di ArrayExpress è costituita dalla base di dati, dai moduli di sottomissione in formato MAGE-ML (MicroArray Gene Expression-Markup Language), da uno strumento per fare la sottomissione diretta senza passare dalla formattazione manuale in MAGE-ML e da un'interfaccia per l'interrogazione del database e il recupero dei dati (Figura 13). Per quanto riguarda il deposito e la conservazione delle informazioni, ArrayExpress si attiene strettamente alle procedure MIAME e MINSEQE in accordo con le disposizioni date dalla Microarray and Gene Expression Data Society (MGED). I dati sottomessi devono quindi essere assolutamente conformi alle disposizioni fornite da MIAME ed essere in formato MAGE-ML.

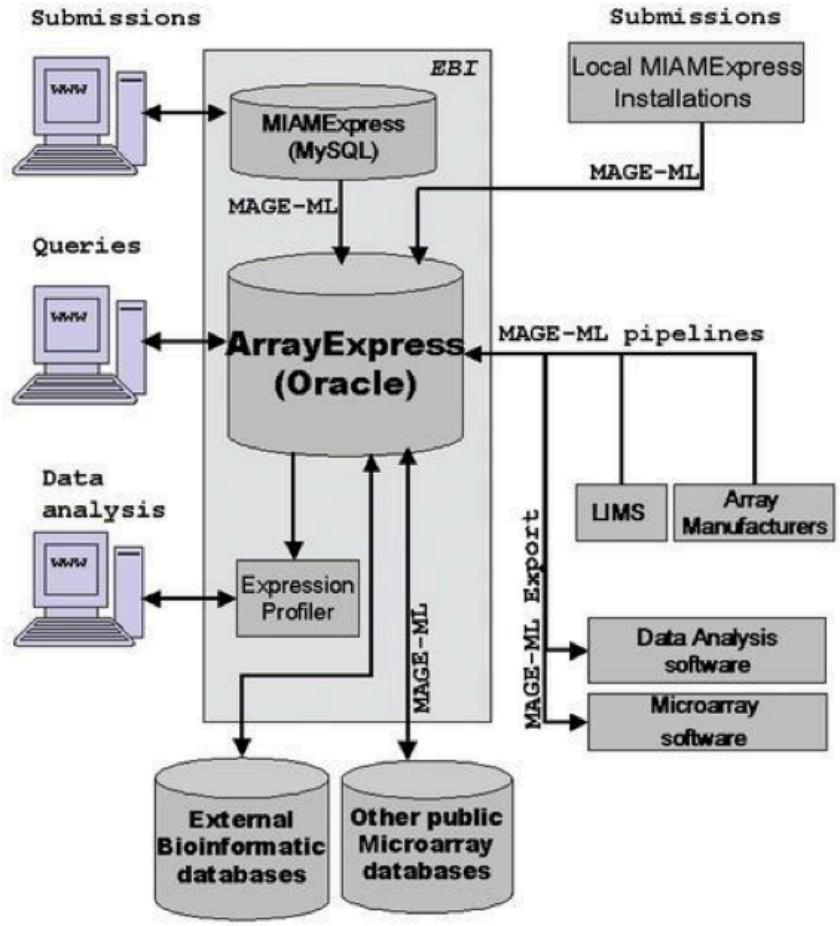

Figura 13: Array Express Business Model

Tuttavia, poiché la sottomissione via MAGE-ML è poco intuitiva, i curatori dell'EBI hanno predisposto un web-tool chiamato MIAMExpress che permette all'utente di annotare ciò che intende depositare nella banca dati direttamente nel formato richiesto dallo standard MIAME. MIAMExpress è facilmente utilizzabile ed è corredato da una chiara documentazione sulle modalità e sui file da sottomettere. La sottomissione avviene per passi e a ciascuno di essi corrisponde un differente protocollo sperimentale o di analisi dei dati che segue la scansione temporale dell'esperimento. A ogni protocollo viene associato un codice che potrà essere utilizzato per effettuare una ricerca selettiva. L'EBI rilascia un punteggio che viene associato al data-set e che è esplicativo del grado di compatibilità dei dati immessi con lo standard MIAME.

ArrayExpress si compone di due database, ossia un archivio nel quale sono conservate le annotazioni relative a ogni singolo esperimento (Experiments Archive) ed una galleria dei profili di espressione genica dei vari esperimenti dalla quale è possibile estrarre i profili di espressione genica a partire dal nome del gene, dalle proprietà e dalla similarità di profilo (Gene Expression Atlas). L'interrogazione sia dell'Experiments Archive sia del Gene Expression Atlas è semplice e intuitiva; infatti all'utente basta inserire il nome, l'identificativo o la proprietà di un gene o di diversi geni, per recuperare l'elenco degli esperimenti in cui il dato gene è stato studiato e visualizzare il suo profilo di espressione. É inoltre possibile filtrare gli esperimenti da recuperare in base alla specie, al tipo di microarray o al tipo di esperimento e, una volta identificato l'esperimento d'interesse, l'utente può ottenere ulteriori informazioni circa i campioni, il protocollo utilizzato o il disegno sperimentale ed esportare i dati (sia grezzi sia pre-processati, pre-analizzati) associati all'esperimento di suo interesse.

Ogni esperimento è corredato da una breve descrizione, da una lista dei fattori sperimentali e da un riassunto del disegno sperimentale. I dettagli sperimentali forniti in ArrayExpress sono suddivisi in diverse sezioni:

- *description*: una breve descrizione dell'esperimento fatta dallo sperimentatore che ha depositato i dati;
- *MIAME score*: quantifica la congruità dell'esperimento agli standard di MIAME. Lo score massimo equivale a 5 ed il punteggio viene costruito in base alla presenza di una descrizione completa del microarray utilizzato, del protocollo, dei fattori sperimentali, dei dati grezzi e/o di quelli processati;
- *contact:* il nome ed i contatti dallo sperimentatore che ha depositato i dati;
- *citations:* le informazioni circa le pubblicazioni relative all'esperimento selezionato, inclusi i link a Pubmed;
- *links:* i vari links tra i quali quelli al Gene Expression Atlas, al protocollo utilizzato e alla descrizione del tipo di microarray utilizzato;
- *experiments types:* il disegno sperimentale;
- *experiment factors:* una lista di nomi e valori riguardanti i fattori sperimentali utilizzati nell'esperimento;
- *sample attributes:* una lista di attributi utilizzati per descrivere i campioni.

Così come in GEO, i dati relativi ai segnali veri e propri possono essere presenti in una qualche forma pre-processata (ossia già normalizzati e/o analizzati) oppure in termini di dati grezzi (ad esempio le immagini dei .CEL file). Infine vale la pena notare che ArrayExpress ha recuperato da GEO e incluso nel proprio Experiments Archive tutte quelle serie presenti in GEO che rispettavano i criteri di deposito di MIAME. Di fatto questi esperimenti, identificati dall'acronimo GEOD, sono presenti in entrambe le banche dati.

3.4 A-MADMAN

A-MADMAN (annotation-based microarray data meta-analysis tool) è un'applicazione web (http://xlabserver4.biomed.unimo.it/amadman/) - realizzata dal centro di bioinformatica dell'Università degli Studi di Modena e Reggio Emilia con il quale ho collaborato - che consente il recupero, l'organizzazione e la meta-analisi di dati di espressione genica prodotti con la tecnologia Affymetrix e contenuti nella banca dati Gene Expression Omnibus (GEO). La procedura implementa una strategia di combinazione diretta dei dati e un metodo *ad-hoc* di integrazione delle diverse piattaforme e di rimozione degli effetti dovuti al singolo esperimento (meta-normalizzazione) a partire dai livelli di espressione generati per tipo di microarray. A-MADMAN quindi, prima quantifica il segnale di espressione per tutti i campioni ottenuti con un dato tipo di microarray, poi integra tipologie di microarray diverse e infine normalizza la matrice integrata per rimuovere tutti gli eventuali bias[5] sperimentali.

3.4.1 A-MADMAN: l'architettura del software

A-MADMAN è un'applicazione web utilizzabile per scaricare i dati di espressione depositati in GEO sotto forma di file .CEL, per riclassificarli, per organizzarli in un database locale e per analizzarli utilizzando approcci di meta-analisi. L'analisi dei dati produce un oggetto di R[6], l'ExpressionSet, che contiene:

- I segnali di espressione genica derivati dall'integrazione di esperimenti indipendenti;
- Tutte le caratteristiche e le meta-informazioni dei campioni che compongono il set di dati integrato.

Le meta-informazioni dell'ExpressionSet sono derivate sia dalle caratteristiche descritte in GEO sia da annotazioni ed etichettature proprietarie introdotte dall'utente. Da un punto di vista informatico, A-MADMAN è composto di una console, di un *job-server* e di un'applicazione web vera e propria (Figura 14).

La console è utilizzata per l'interrogazione di GEO, per l'importazione dei dati e per il popolamento dei file locali. In questa fase vengono utilizzati script di configurazione che contattano i server FTP di GEO, scaricano le serie specificate, i singoli campioni e le meta-informazioni ad essi collegati ed organizzano tutti i file in maniera gerarchica direttamente all'interno del computer in cui è installato A-MADMAN.

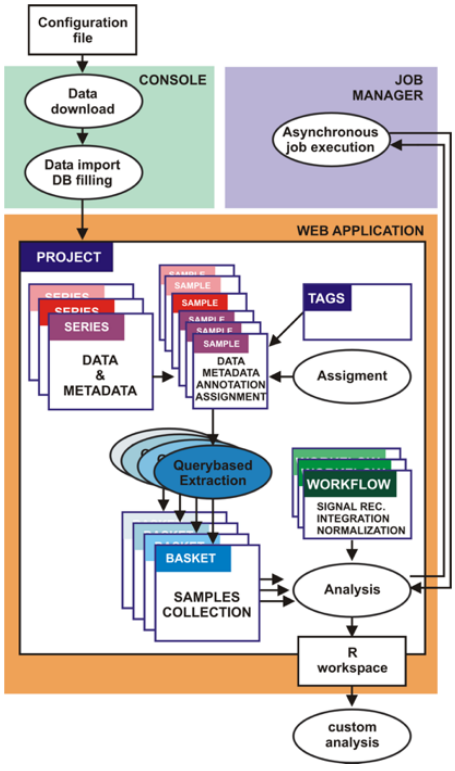

Figura 14: L'architettura di A-MADMAN

5 *Bias* termine che indica un errore sistematico.
6 *R* è un ambiente di sviluppo specifico per l'analisi statistica dei dati che utilizza un linguaggio di programmazione derivato e in larga parte compatibile con S.

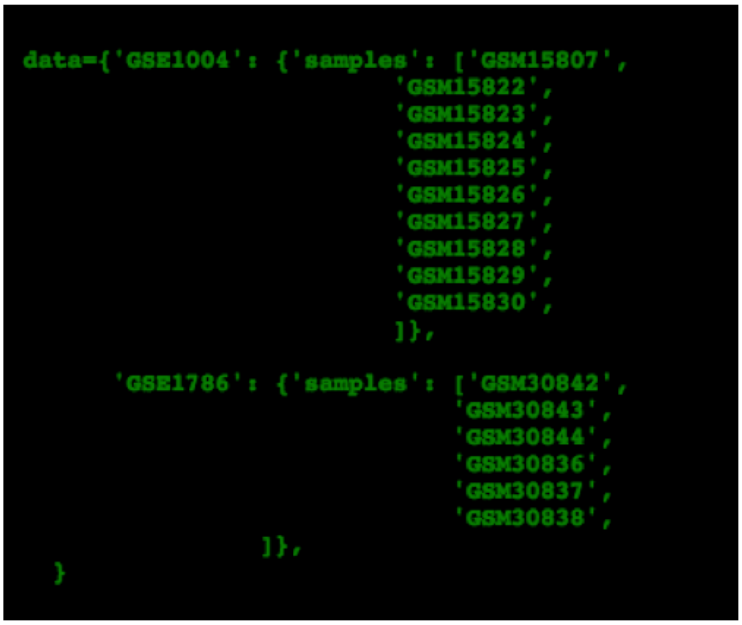

Figura 15: Esempio parametri di input per script di download dati di A-MADMAN

Il *job-server* è utilizzato per contattare il server FTP di GEO in maniera asincrona rispetto alle altre operazioni condotte dal programma. L'utilizzo di questa soluzione permette di velocizzare i processi di scaricamento dei dati che, considerate le notevoli dimensioni dei file, se condotti attraverso semplici cicli http di richiesta-risposta, potrebbero richiedere tempi di esecuzione non accettabili.

L'utilizzo pratico di A-MADMAN è, infine, garantito dall'applicazione web la cui interfaccia grafica user-friendly permette all'utente di organizzare, caratterizzare e analizzare i dati in modo semplice e intuitivo. Il software è scritto in python (http://www.python.org/), è basato sul sistema di gestione web chiamato Django (http://www.djangoproject.com/) e usa l'ambiente R (http://www.r-project.org/).

3.4.2 A-MADMAN: organizzazione e annotazione dei dati

L'utente accede all'applicazione web di A-MADMAN attraverso un account personale, ossia uno spazio di lavoro in cui i dati sono organizzati in progetti. Un progetto è una raccolta di campioni, esperimenti, annotazioni, raggruppamenti di campioni e analisi in cui tutti gli oggetti condividono un denominatore comune (ad esempio, una domanda biologica, uno o più tipi cellulari, una patologia, ecc.).

I campioni e le serie sono gli oggetti principali importati da GEO attraverso i comandi della console. Ogni campione è associato a un singolo file .CEL e alle meta-informazioni ricavate dal miniml file di GEO. Una serie è una raccolta di campioni e definisce, così come in GEO, un esperimento. In aggiunta a campioni e serie, l'utente può creare svariate entità addizionali che risultano poi estremamente utili nella successiva fase di riorganizzazione ed analisi dei dati.

In particolare, in A-MADMAN è possibile creare i seguenti oggetti:

- *tag*: è una etichetta descrittiva che può essere assegnata dall'utente a ogni campione e utilizzata per selezionare e riunire i campioni in gruppi omogenei non necessariamente previsti nello studio originario.
- *basket:* è una raccolta di campioni generata attraverso delle *query* logiche attuate sfruttando le *tag.*

L'annotazione attraverso le tag costituisce uno degli elementi di maggiore originalità di A-MADMAN in quanto consente non solo di completare, attraverso l'uso di un vocabolario controllato e curato direttamente dall'utente, la caratterizzazione dei campioni, ma soprattutto di raggruppare i campioni in nuove popolazioni omogenee espandendo quindi la varietà delle domande biologiche a cui l'analisi poteva rispondere. Oggi, le *tag*, per la modalità con cui sono strutturate, risultano essere uno dei principali limiti del sistema.

E' importante sottolineare che in A-MADMAN e nel presente progetto il termine *tag* viene utilizzato con un significato più generale rispetto a quello che normalmente gli viene attribuito. La definizione riportata in *Wikipedia* è la seguente: "Un *tag* è una parola chiave o un termine associato a un'informazione (un'immagine, una mappa geografica, un post, un video clip ...), che descrive l'oggetto rendendo possibile la classificazione e la ricerca di informazioni basata su parole chiave. I *tag* sono generalmente scelti in base a criteri informali e personalmente dagli autori/creatori dell'oggetto dell'indicizzazione."

In A-MADMAN e nel presente progetto il termine *tag* viene utilizzato per denotare un attributo di un campione/esperimento che a differenza degli altri (quali quelli provenienti da GEO) non è fisso nello schema del database ma viene aggiunto *localmente* dai ricercatori del laboratorio in modo *dinamico* e *variabile* tra esperimenti e campioni differenti. Quindi un *tag* è a tutti gli effetti un attributo e come tale avrà un valore associato[7]; pertanto nella definizione dei *basket* le *query* logiche sono costituite da predicati del tipo

Tag Operatore Valore

3.4.3 A-MADMAN: I limiti che hanno portato allo sviluppo di un nuovo sistema

Diversi sono i limiti di A-MADMAN riscontrati dal laboratorio di ricerca durante l'utilizzo del sistema. In primis occorre sottolineare come dati e metadati, una volta scaricati dalle banche dati pubbliche, vengano memorizzati in file senza l'ausilio di database. Questo comporta inefficienza di organizzazione dello spazio di memoria, ma soprattutto comporta enormi limiti dal punto di vista delle interrogazioni, nonché il rischio di inconsistenza dei risultati.

Altri importanti limiti risiedono, come detto, nell'organizzazione delle *tag*. Queste non sono suddivise in attributi dei campioni a seconda del significato. Questo significa che ogni campione presenta un numero indefinito di tag tutte riportate come unico attributo, seppur queste abbiano semantica completamente diversa. Ancora una volta gli aspetti negativi si registrano sui risultati delle *query*.

[7] Come verrà discusso nel seguito, non si considerano tag multi-valore in quanto riscritti come tag mono-valore.

Pur offrendo un'interfaccia grafica A-MADMAN richiede che molte delle operazioni siano eseguite da linea di comando.

Il limite più grosso è senza dubbio legato all'analisi integrata dei dati. A-MADMAN, infatti, contiene anche strumenti di analisi editabili tramite l'ambiente R. Questo comporta una serie di vincoli sulla creazione dei *basket*, vincoli legati alla struttura dei microarray tramite cui sono stati effettuati gli esperimenti. Il sistema sviluppato nell'ambito del progetto di tesi supererà tali limiti (questo è reso possibile grazie ai nuovi algoritmi di analisi dei dati).

Per tutti questi motivi, il centro di bioinformatica dell'Università degli Studi di Modena e Reggio Emilia, ha richiesto lo sviluppo di un sistema più versatile che risponda alle loro attuali esigenze.

3.5 Conclusioni

In seguito ad attenta analisi delle strutture e dei formati dei dati di cui si disponeva, si è giunti alla conclusione che, così come faceva A-MADMAN, anche il nuovo sistema adotterà come unica fonte dei dati Gene Expression Omnibus. I motivi sono sostanzialmente cinque:

1. La maggior parte dei dati disponibili su Array Express (circa il 95%) sono copie di dati in realtà già presenti in GEO, o non sono di interesse per il centro di ricerca;
2. La farraginosa procedura di caricamento dei dati fa si che la maggior parte delle pubblicazioni (quasi tutte) vengano effettuate su GEO e non su ArrayExpress.
3. Array Express fornisce lo schema XML adottato per il rispetto del protocollo MIAME: MAGE-ML, ma permette il download dei dati nel solo formato di testo tabulato. Questo comporterebbe un onero importante, e non giustificato date le considerazioni ai punti 1 e 2, del punto di vista della manipolazione dei dati.
4. Oltre a fornire i dati nel solo formato di testo tabulato, Array Express, è solito nominare in maniera differente, all'interno di esperimenti differenti, attributi semanticamente equivalenti. Ciò aggraverebbe ulteriormente il compito di trasformazione che precede il caricamento dei dati.
5. Per finire, occorre sottolineare che negli ultimi anni Array Express si è concentrato sull'elaborazione e lo storage di dati riguardanti il sequenziamento di DNA/RNA e abbia trascurato la branca riguardante i microarray.

4 Progettazione del Database

4.1 Progettazione concettuale

Lo sviluppo concettuale tramite il modello *Entity Relationship* sarà effettuato mediante una metodologia di progetto di tipo mixed, analizzando ad ogni passo sia le specifiche imposte dall'utente finale sia quelle dipendenti dai file esistenti: specifiche imposte dalla struttura delle basi di dati utilizzate dai sistemi GEO e ArrayExpress.

Il sistema che si vuole progettare deve essere in grado di trattare tutte le informazioni di interesse relative ad un esperimento (*series*) e ai campioni (*sample*) di cui esso si compone.

E' richiesta, inoltre, la possibilità di raggruppare gli esperimenti all'interno di progetti su cui verranno poi effettuate interrogazioni specifiche. Appare quindi evidente che una prima rappresentazione del problema secondo il modello concettuale E/R potrebbe essere la seguente:

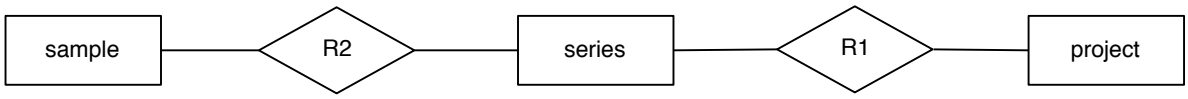

Figura 16: Modellazione Entity Relationship, primo passo.

Le entità *sample* e *series* sono caratterizzate da attributi sia semplici che multipli, che sono definiti a priori, e che dipendono dalle strutture delle basi di dati pubbliche. Essi verranno presentati nel dettaglio in seguito. Gli unici attributi di cui sono già stati presentati scopo e formato sono gli identificatori (vedi cap. 3: Le Sorgenti dei Dati). L'entità project, a differenza delle altre due, non riporta alcun attributo "derivato". Di seguito viene riportata una generica rappresentazione degli attributi (Figura 17).

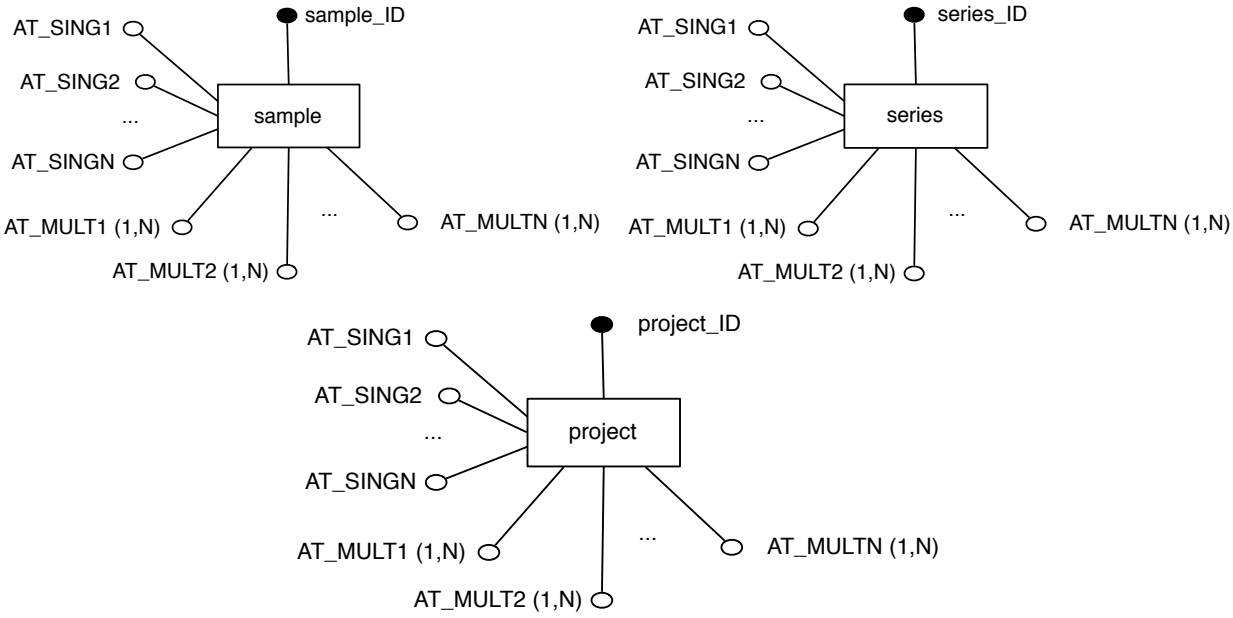

Figura 17: Rappresentazione generica degli attributi

Le relazioni e le cardinalità che legano *series* e sample rispecchiano quelle del sistema pubblico GEO, la novità è data dal progetto (*project*): un esperimento può appartenere a più progetti così come un campione può appartenere a più esperimenti. Un progetto non può presentare più volte lo stesso esperimento e un campione non può presentarsi più di una volta all'interno dello stesso esperimento. E' però possibile, all'interno di esperimenti differenti, trovare lo stesso campione ma solo se gli esperimenti appartengono a progetti diversi. Non deve quindi accadere, per evitare incongruenze nella successiva analisi dei dati (già discussa nei capitoli 2 e 3), che all'interno di uno stesso progetto compaia più volte lo stesso campione, anche se in esperimenti diversi (*vincolo (*), vedi pag 25*) . Una delle possibili rappresentazioni concettuali dei vincoli sopra espressi potrebbe essere la seguente, da notare come nello schema siano trascurati, per semplicità grafica, tutti gli attributi non strettamente legati ai vincoli che in questo momento si vogliono rappresentare.

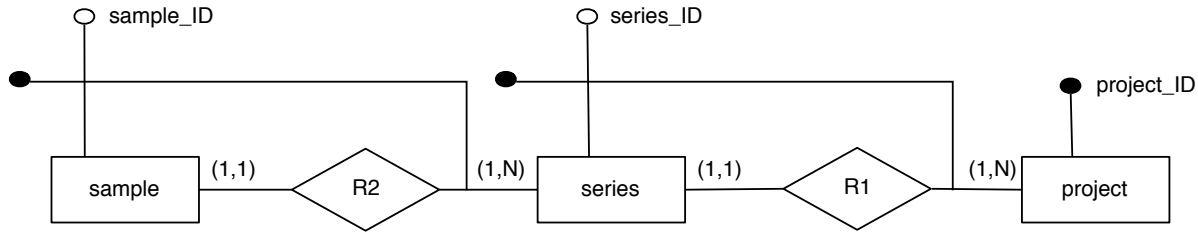

Figura 18: Modellazione Entity Relationship, secondo passo

Dal punto di vista concettuale, rispetto ai vincoli sopra espressi, lo schema presentato in Figura 18 è corretto. Occorre però tenere presente che la progettazione che si sta eseguendo non è solo dettata dalle specifiche imposte dall'utente finale (che in questo caso è il laboratorio di ricerca di bioinformatica), ma anche da file e database esistenti. Come già detto, infatti, il database che si vuole realizzare deve rispettare la struttura di base dei sistemi GEO e ArrayExpress che non identificano un campione con l'esperimento a cui esso appartiene. Inoltre, occorre tenere presente che l'adozione della soluzione presentata porterebbe alla violazione della seconda forma normale (2NF) e, quindi, a problemi di ridondanza nei dati e a possibili anomalie in fase di aggiornamento. Cerchiamo di chiarire il concetto con un esempio. La traduzione logica relazione dell'entità sample potrebbe essere, ad esempio, la seguente:

sample (sample_ID, series_ID,project_ID, title, ..)

FK: series_ID, project_ID REFERENCES series

Dove *title* rappresenta uno degli attributi dell'entità sample (gli attributi verranno analizzati nel dettaglio nei paragrafi successivi).

E' quindi evidente che la dipendenza funzionale *sample_ID -> title* dettata dalla struttura di base dei sistemi originari porta alla violazione della 2ND,

sample

sample_ID	series_ID	project_ID	title
GSM1233	GSE455	1	title1
...	...	...	...
GSM1233	GSE555	1	title1

Figura 19: Violazione della 2NF

infatti, l'attributo non primo *title* non dipende completamente dalla chiave sample_ID, series_ID,project_ID ma solo da una parte della stessa.

L'utilizzo dell'identificare composto misto non è quindi corretto. Lo schema proposto di seguito "risolve" i due problemi appena evidenziati. *Sample_ID* e *series_ID*, come già detto nei paragrafi precedenti, sono univoci e stabili in GEO.

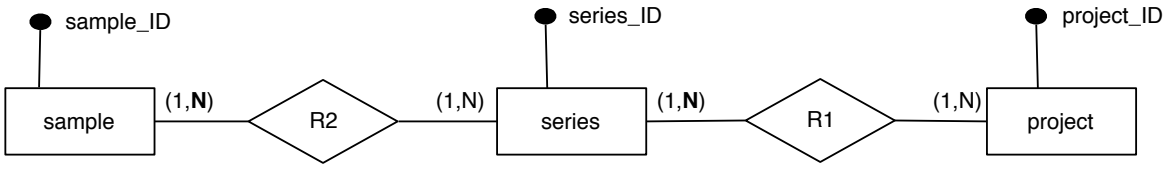

Figura 20: Modellazione Entity Relationship, terzo passo

Il vincolo (*) può essere soddisfatto mediante il modello concettuale solamente tenendo traccia, all'interno di ogni campione del/dei progetti a cui esso appartiene. Questo tipo di soluzione complicherebbe inutilmente lo schema e, soprattutto, denaturerebbe quella che è la struttura di base del sistema pubblico GEO. La situazione da evitare è chiarita nell'immagine successiva:

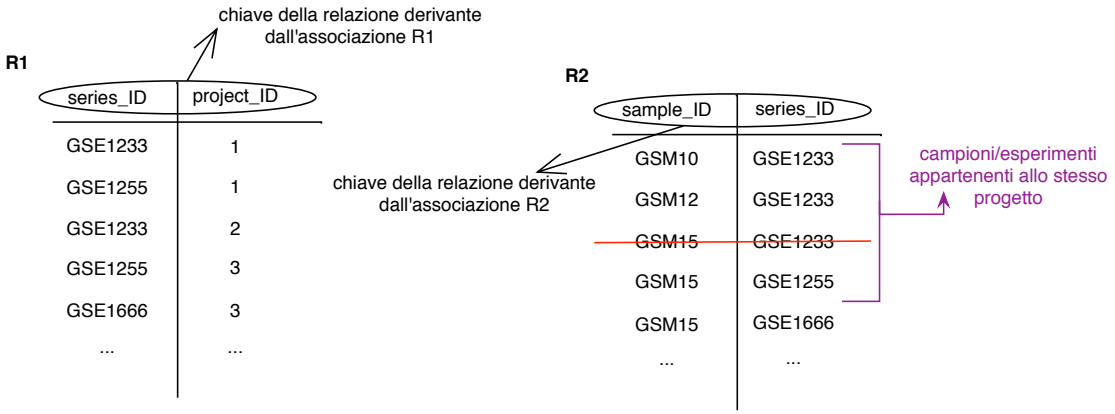

Figura 21: Vincolo sullo schema

Il campione GSM15 non può appartenere sia alla serie GSE1255 che alla serie GSE1233 perché queste appartengono entrambe al progetto 1: questo è il vincolo che si vorrebbe imporre ma che lo schema concettuale non ci consente di esprimere in maniera semplice ed efficace. Per evitare questa situazione è necessario ricorrere ad opportuni trigger. Un lettore attento avrà notato sicuramente un problema: se un eventuale trigger impedisce la presenza contemporanea del campione GSM15 negli esperimenti GSE1255 e GSE1233 il progetto 2 potrebbe essere sprovvisto del campione quando in realtà la sua presenza era richiesta; si potrebbe dunque pensare ad un trigger "intelligente" che scelga, dopo opportune analisi, quale sia la serie più opportuna da cui eliminare il campione. Una soluzione di questo tipo richiederebbe modifiche in cascata (non sempre possibili) di tutta la relazione. Cerchiamo di capire il motivo per cui il gruppo di bioinformatica del Centro Interdipartimentale di Ricerche Genomiche ci impone tale vincolo: quando si compie un'analisi, su di un set di campioni all'interno di esperimenti di un certo progetto, uno dei dati rilevanti è la concentrazione con cui una certa sequenza nucleotidica viene rilevata. Appare quindi evidente che se un campione viene analizzato più volte il valore della

"concentrazione" sarebbe falsato e privo di significato. La soluzione ovvia è che i campioni saranno registrati nel sistema senza eseguire alcun tipo di controllo: un campione potrà quindi appartenere a esperimenti diversi anche se questi si trovano nello stesso progetto. In fase di interrogazione dei dati i valori duplicati saranno eliminati. Da notare che questa soluzione è accettabile solo perché le interrogazioni saranno effettuate sempre a livello di progetto.

Come discusso nel capitolo precedente, una delle particolarità del database da progettare è la possibilità di gestire i *tag*, ovvero che l'utente possa aggiungere dinamicamente attributi ai campioni che "dipendono" dall'esperimento. L'esigenza di gestire i *tag* non ha consentito la costruzione di uno schema definitivo e comprensivo di tutti gli attributi. La soluzione adottata è illustrata in Figura 22: consiste nel definire uno schema di base, comprendente gli attributi comuni a tutti i campioni e a tutti gli esperimenti, derivanti, almeno in parte, dallo schema che caratterizza il *microarray* database GEO; e nel permettere l'inserimento di attributi aggiuntivi (i *tag* appunto) come istanze di opportune entità/associazioni dello schema E/R e quindi come tuple di opportune tabelle del corrispondente schema relazionale.

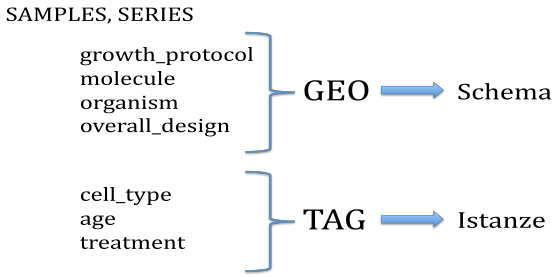

Figura 22: attributi fissi VS attributi variabili

In altre parole, la soluzione è paragonabile con quanto avviene nei cosiddetti *cataloghi di sistema* dei DBMS, ovvero una serie di tabelle di sistema che contengono informazioni sullo schema dei database, quali ad esempio il nome delle tabelle e degli attributi, la struttura delle *primary key* e *foreign key*.

Una prima modellazione potrebbe essere la seguente:

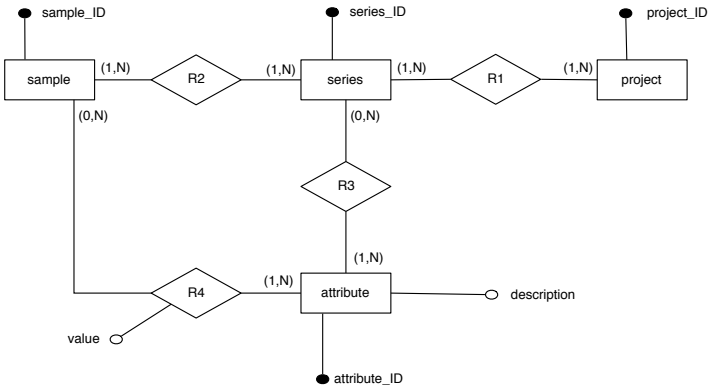

Figura 23: Modellazione Entity Relationship, quarto passo

Un esperimento può quindi avere uno o più attributi dinamici, un attributo fa riferimento ad uno o più esperimenti ed è associato ad uno o più campioni con un determinato valore. Si è deciso di non lasciare la possibilità all'utente di introdurre attributi dinamici multivalore; qualora si dovesse presentare la necessità si potranno inserire più attributi singoli differenti. Per chiarire quanto appena detto di seguito viene riportato un esempio: supponiamo che l'utente voglia inserire il nuovo attributo dinamico *contributor* e che questo sia un attributo multivalore, ovvero che ci sia la necessità di inserire due o più *contributor*, l'utente potrà inserire più attributi *contributor1*, *contributor2*, ecc.. ed associare ad ognuno di essi un determinato valore.

Si è deciso di adottare questo tipo di soluzione per semplicità e perché attributi dinamici di questo tipo non saranno frequenti e, quando presenti, non saranno caratterizzati da più di 3-4 valori.

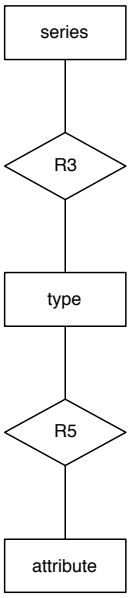

Figura 24: Una nuova entità per interrogazioni "intelligenti"

L'obiettivo finale sarà quello di interrogare dinamicamente il sistema per cercare i campioni che all'interno di un esperimento di un dato progetto possiedono le specifiche richieste. Le interrogazione saranno per lo più relative ad attributi inseriti dinamicamente dall'utente perché di maggior interesse. Per strutturare in maniera efficiente le interrogazioni dinamiche potrebbe essere utile introdurre una nuova entità, come mostrato in Figura 24. Lo scopo dell'entità *type* è quello di suddividere gli esperimenti in classi ognuna delle quali sarà caratterizzata da un certo numero di attributi. In questo modo sarà dunque possibile costruire interrogazioni dinamiche mirate. Cerchiamo di chiarire questo punto con un esempio. Supponiamo di considerare un esperimento di 100 campioni di cui 65 possiedono una struttura di "Tipo1" che prevede gli attributi "A1" e "A2", e 35 che possiedono una struttura di "Tipo2" che prevede solo l'attributo "A1". Suddividendo in tal modo gli esperimenti è possibile, in una futura interfaccia grafica, permettere la costruzione d'interrogazioni mirate che non permettano, ad esempio, l'inserimento di richieste sull'attributo A2 se si stanno eseguendo interrogazioni su esperimenti di "Tipo1", e così via. Ovviamente un esperimento sarà di uno e un solo tipo, un tipo farà riferimento ad uno o più esperimenti. Ogni tipo può presentare uno o più attributi e un attributo sarà presente in uno o più tipi di esperimenti. Di seguito viene fornita una delle possibili rappresentazioni grafiche dell'aggregazione binaria stabilita tra le due classi "Tipo" e "Attributo" nell'associazione "R5".

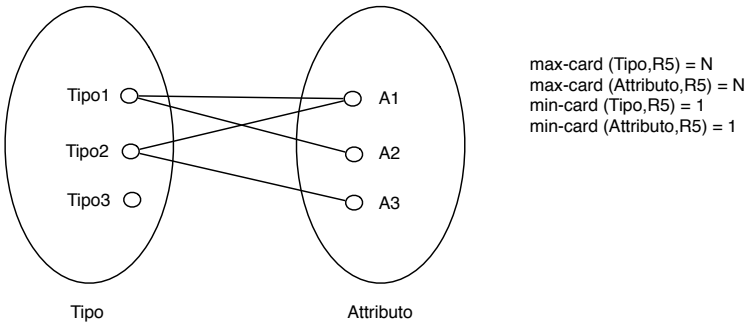

Figura 25: Aggregazione binaria tra le classi "Tipo" e "Attributo"

Lo schema concettuale complessivo, ancora privo della maggior parte degli attributi viene presentato in Figura 26.

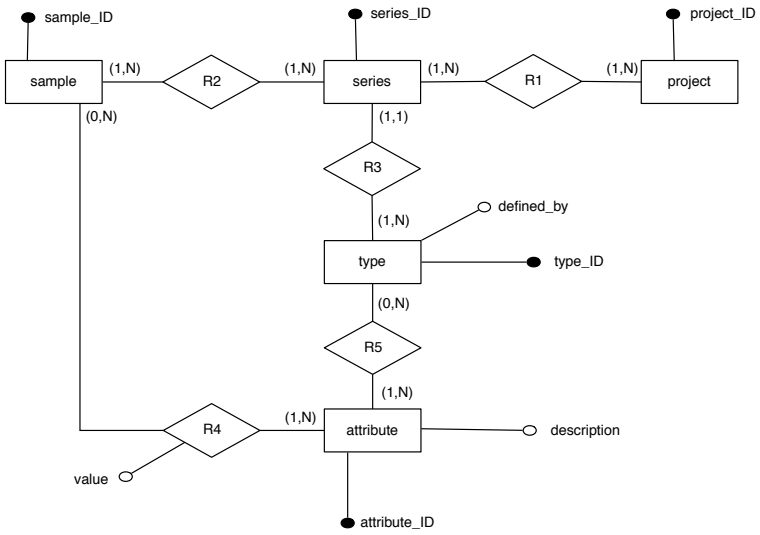

Figura 26: Schema concettuale privo di attributi non significativi

Rimane però un problema irrisolto: secondo lo schema presentato si ha, sia la possibilità di associare lo stesso campione ad esperimenti differenti, sia la possibilità di inserire attributi dinamici; quella che manca è la possibilità di associare ad un determinato campione attributi differenti a seconda dell'esperimento a cui esso appartiene (specifica imposta dall'utente finale). Tutto ciò può essere tradotto in E/R secondo due modalità presentate rispettivamente in Figura 27 la prima e in Figura 28 la seconda.

La prima soluzione non prevede alcuna modifica dello schema complessivo se non l'aggiunta di un attributo all'entità *sample*. *Sample_ID*, in questo caso, non rappresenta più l'identificatore univoco e stabile fornito da GEO ma è un identificatore progressivo interno.

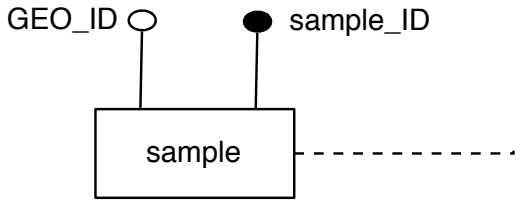

Figura 27: Soluzione 1

Occorre comunque memorizzare l'ID fornito da GEO (GEO_ID) ma nello schema esso non rappresenterà più una chiave. Questa prima soluzione genera ridondanze che non sono tanto causa di spreco di memoria, quanto di possibili anomalie in fase di caricamento e aggiornamento/modifica dei dati.

La seconda soluzione richiede, invece, una reificazione della relazione *R2.* Tale approccio eliminerebbe qualsiasi problema legato alla ridondanza dei dati ma, al contempo, complicherebbe lo schema. Da notare come in questo caso l'identificatore dell'entità campione (*sample*) corrisponderebbe all'identificatore univoco e stabile fornito da GEO (GSMXXX), non si renderebbe quindi necessaria l'aggiunta di un nuovo attributo identificatore interno.

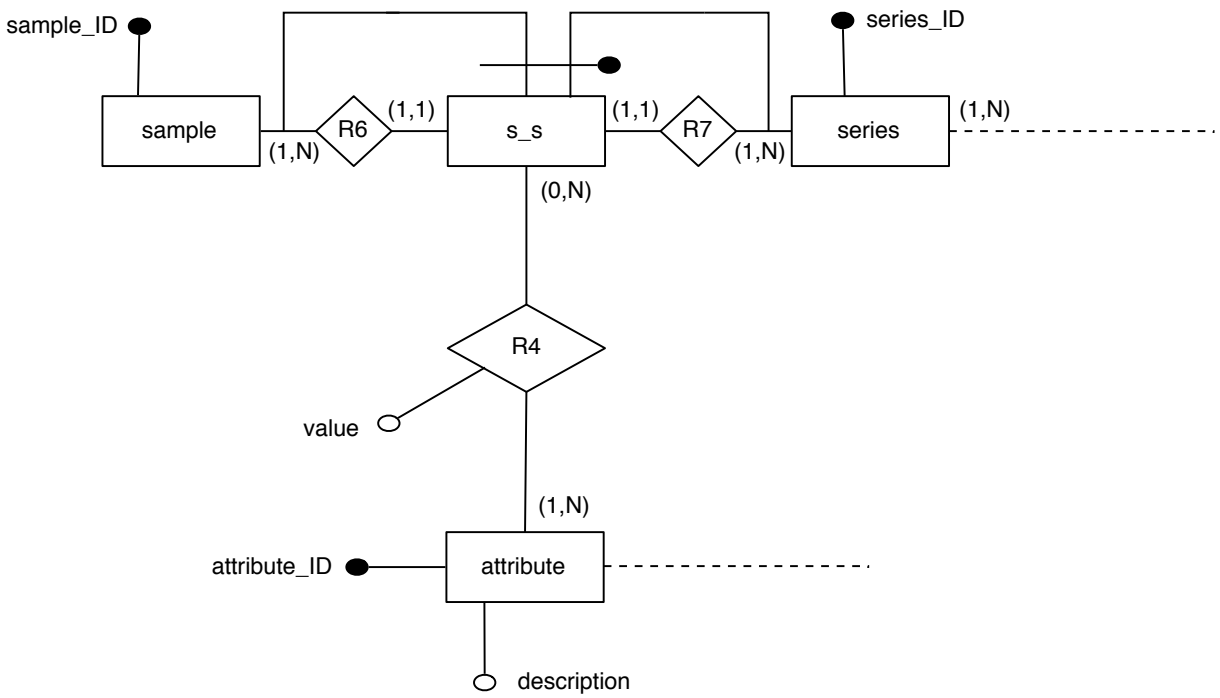

Figura 28: Soluzione 2

Tramite l'introduzione dell'entità *s_s* (*sample_series*) ottenuta dalla reificazione dell'associazione R2, sarà possibile inserire nella relazione corrispondente tuple come quelle mostrate in Figura 29, senza dover ricorrere alla distinzione di campioni che in realtà rappresentano lo stesso oggetto, caratterizzato da attributi differenti.

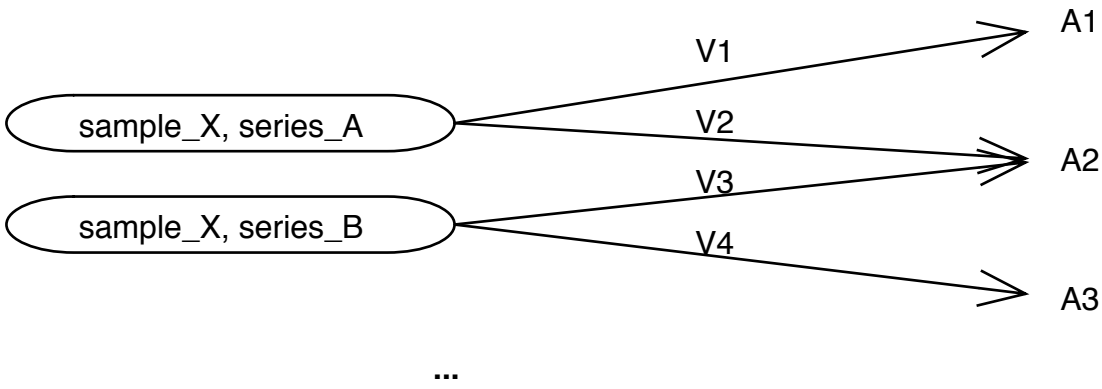

Figura 29: Tuple consentite dalla reificazione dell'associazione s_s

La scelta della soluzione da implementare è dettata dalla frequenza con cui questa operazione verrà effettuata, ovvero dalla frequenza con cui sarà richiesto di inserire nel sistema lo stesso campione (in esperimenti differenti) con attributi dinamici diversi. In seguito ad attenta analisi, si è giunti alla conclusione che la seconda soluzione è la più appropriata per le operazioni che in futuro si vorranno eseguire sulla base di dati.

4.2 Analisi degli attributi

4.2.1 Attributi del campione

Di seguito verranno analizzati tutti gli attributi dell'entità *sample* (campione) e verranno presentate le correlazioni presenti (*mapping*) con gli attributi derivanti dalla base di dati pubblica GEO.

Nuovo sistema	***Gene Expression Omnibus***
sample_ID: chiave identificativa della serie. L'attributo sample_ID sarà di tipo stringa e manterrà il precedente formato GSMxxx.	*!Sample_geo_accession*
Non di interesse	*!Sample_status*
Non di interesse	*!Sample_submission_date*
Non di interesse	*!Sample_last_update_date*
Non di interesse	*!Sample_type*
Non di interesse	*!Sample_channel_out*
title: di tipo stringa, a valore singolo e facoltativo esprime il nome attribuito all'esperimento.	*!Sample_title*
source_name: di tipo stringa, facoltativo e a valore singolo; esprime la sorgente del campione ovvero la modalità con cui il campione è stato ottenuto.	*!Sample_source_name_ch1*
organism: di tipo stringa, facoltativo e a valore singolo; riporta il nome dell'organismo da cui proviene il campione	*!Sample_organism_ch1*
characteristics: è un attributo di tipo stringa, facoltativo e a valore multiplo. Tale attributo riporta informazioni di vario tipo a proposito del campione; ad esempio tra le caratteristiche potrebbe essere espresso il genere (maschio/femmina)o l'età del campione ma anche molto altro	!*Sample_characteristics_ch1.*
treatment_protocol	*!Sample_treatment_protocol_ch1*
growth_protocol	*!Sample_growth_protocol_ch1*
molecule	*!Sample_molecule_ch1*,
extract_protocol	,*!Sample_extrat_protocol_ch1*
label	*!Sample_lable_ch1*
label_protocol	*!Sample_lable_protocol_ch1*

description	*!Sample_description*
Non di interesse	*!Sample_contact*
Non di interesse	*!Sample_scan_protocol*
Non di interesse	!Sample_hyb_protocol
platform_ID: rappresenta l'identificatore della piattaforma con cui un certo campione è stato analizzato. Nel sistema GEO esso era l'identificatore dell'entità piattaforma, in questo caso, dal momento che non interessa memorizzare nessuna informazione aggiuntiva riguardo alla piattaforma, esso comparirà come attributo stringa obbligatorio dell'entità campione. Il formato dell'attributo sarà coerente con il formato che esso presentava in GEO.	*platform_ID*

Tabella 1: Mapping degli attributi del campione

L'entità sample complessiva di tutti i suoi attributi sarà quindi rappresentata in E/R come mostrato in Figura 30.

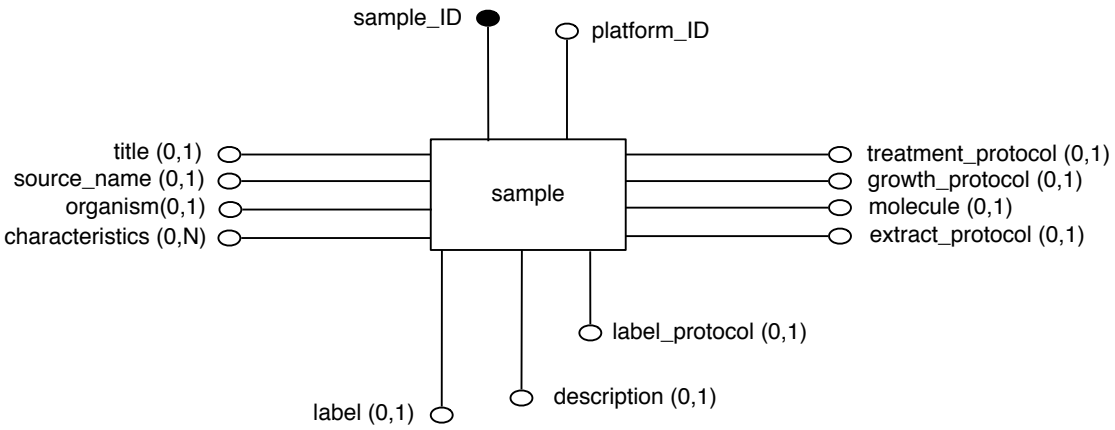

Figura 30: Entità "sample" con attributi

Per poter procedere alla successiva traduzione logica relazionale occorre eseguire una trasformazione dell'attributo multivalore *characteristics*; di seguito viene presentata una delle possibili alternative. La traduzione effettuata tiene conto del fatto che l'attributo è opzionale e che la sua cardinalità massima non è nota a priori (per semplicità grafica sono omessi tutti gli attributi del caso non significativi).

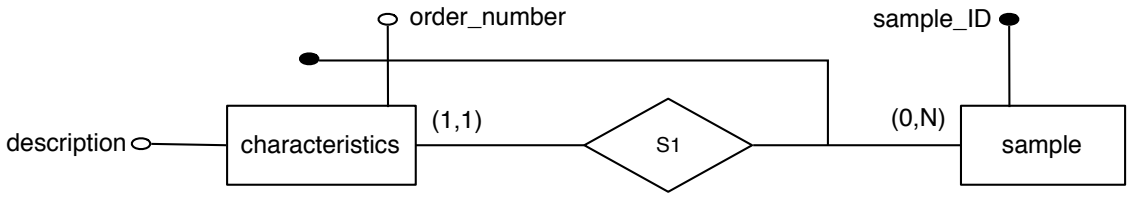

Figura 31: Traduzione dell'attributo multivalore "characteristics" per successiva traduzione logica

4.2.2 Attributi dell'esperimento

Di seguito verranno analizzati tutti gli attributi dell'entità *series* (esperimento) e verranno presentate le correlazioni presenti (*mapping*) con gli attributi derivanti dalla base di dati pubblica GEO.

Nuovo sistema	***Gene Expression Omnibus***
Contributor: attributo multivalore e facoltavio che riporta i nomi di coloro che hanno contribuito all'esperimento.	*!Series_contributor*.
Title: a valore singolo e facoltativo esprime il nome attribuito all'esperimento.	*!Series_title.*
Series_ID:	*!Series_geo_accession.*
Summary: attributo a valore multiplo, facoltativo e di tipo stringa	*!Series_summary*
Overall_design: descrizione generale dell'esperimento, a valore singolo, facoltativo e di tipo stringa.	*!Series_overall_design*
Non di interesse	*Pubmed_ID:* rappresenta l'identificativo (all'interno del database pubmed) dell'articolo corrispondente all'esperimento
Non di interesse	*!Series_type*
Non di interesse	*!Series_platform_taxid*
Non di interesse	Altri

Tabella 2: Mapping degli attributi dell'esperimento

L'entità *series* complessiva di tutti i suoi attributi sarà quindi rappresentata in E/R come mostrato in Figura 32.

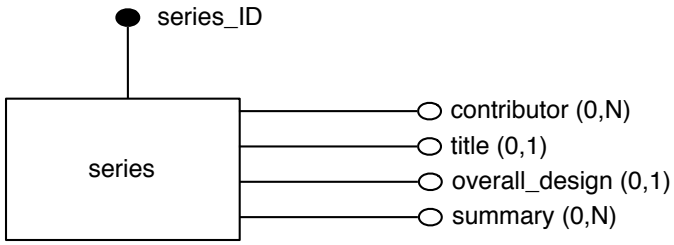

Figura 32: Entità "series" con attributi

Per poter procedere alla traduzione logica relazionale è necessario trasformare lo schema come mostrato in Figura 33. In entrambi i casi la traduzione tiene conto del fatto che gli attributi multivalore *contributor* e *summary* non sono obbligatori e che la cardinalità massima degli stessi non è mai nota a priori.

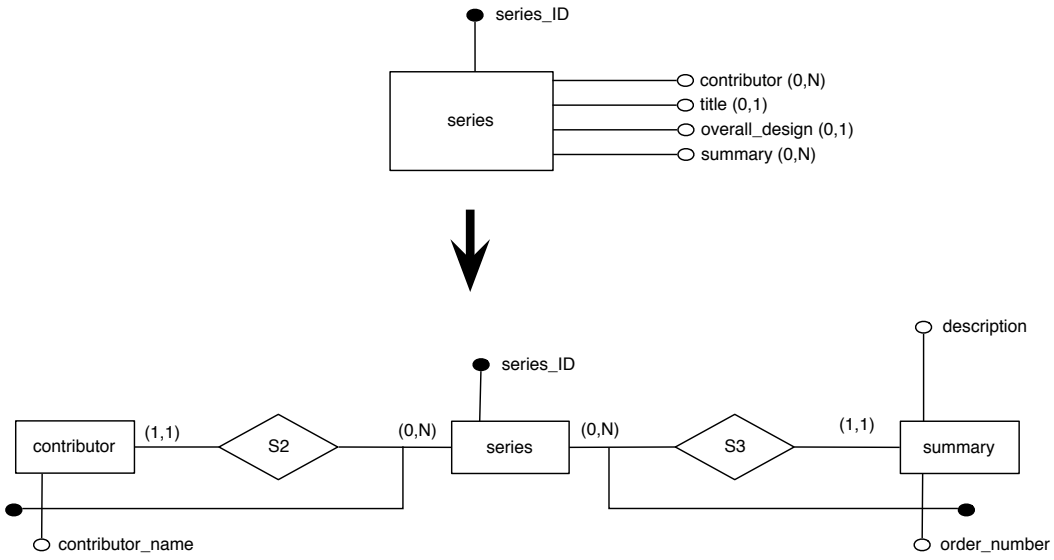

Figura 33: Traduzione degli attributi "contributor" e "summary" per successiva traduzione logica

4.2.3 Attributi del progetto

Come già detto l'entità progetto è propria di questo sistema e non trova duale nel database che sta alla base di Gene Expression Omnibus. Gli attributi richiesti per questa entità sono il nome (*name*), attributo di tipo stringa obbligatorio, e un attributo di tipo stringa opzionale per la descrizione del progetto. L'identificatore *project_ID* è un intero progressivo assegnato automaticamente dal sistema.

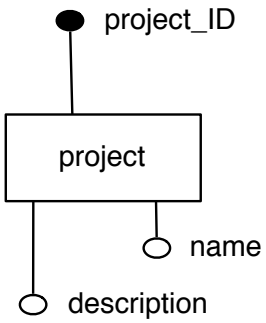

Figura 34: Entità "project" con attributi

4.3 Schema concettuale

Di seguito è riportato lo schema concettuale complessivo di tutte le associazioni e di tutti gli attributi.

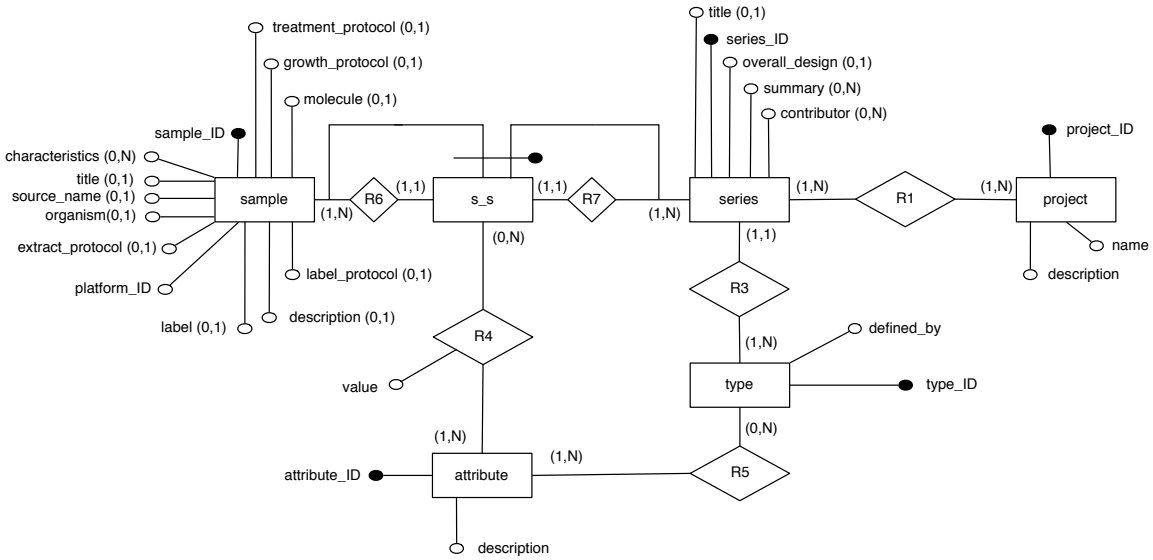

Figura 35: Schema concettuale, risultato finale

Lo schema sopra riportato può essere trasformato come mostrato nella figura seguente. Questa trasformazione è necessaria per poter procedere alla traduzione secondo il modello logico relazionale.

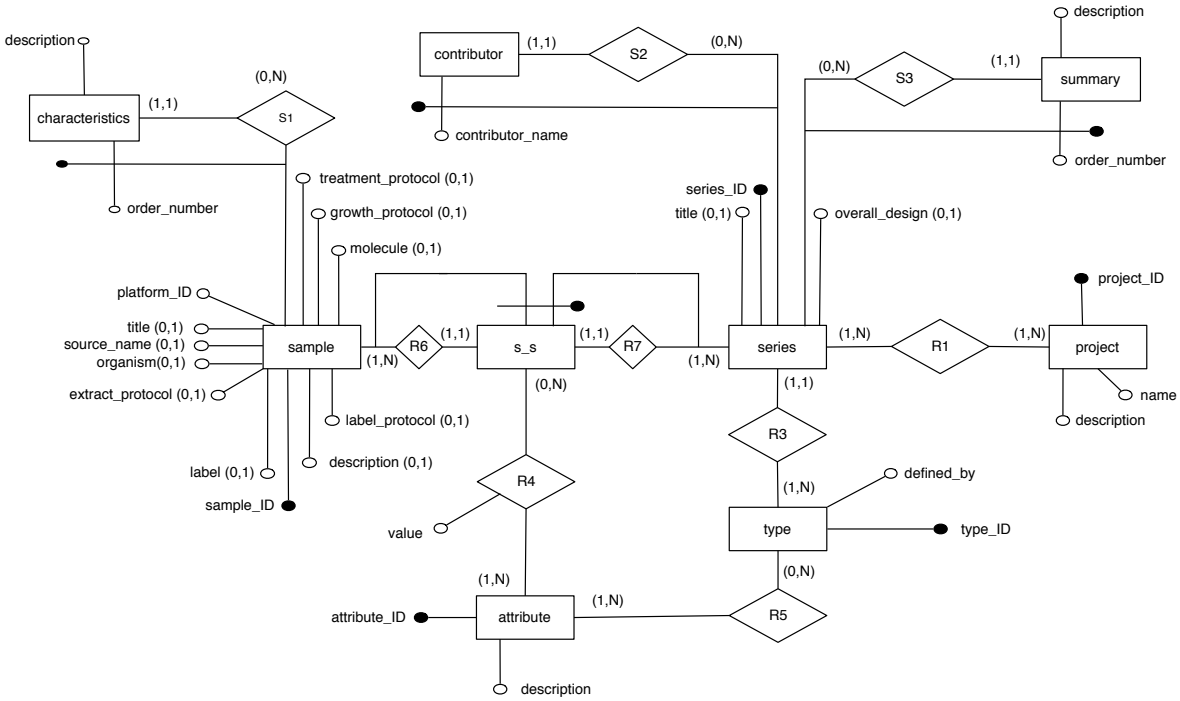

Figura 36: Schema concettuale, modifiche per successiva traduzione in modello logico relazionale

4.4 Progettazione logico relazionale

Nel presente paragrafo viene fornita una delle possibili traduzioni logiche relazionali dello schema concettuale, ottenuto secondo il modello *Entity Relationship* (E/R) tramite una metodologia mixed, presentato nel paragrafo precedente. La traduzione è stata effettuata utilizzando metodologie standard e non, dove possibile è stato specificato il nome della metodologia adottata.

project (project_ID, name, description)

type (type_ID, defined_by)

series (series_ID, title, overall_design, type_ID)

FK: type_ID REFERENCES type NOT NULL

project_series (project_ID, series_ID)

FK: project_ID REFERENCES project

FK: series_ID REFERENCES series

Project_series rappresenta la relazione indicata con R1 nello schema concettuale completo.

summary(order_number,series_ID, description)

FK: series_ID REFERENCES series

contributor(contributor_name, series_ID)

FK: series_ID REFERENCES series

attribute (attribute_ID)

made_of (attribute_ID, type_ID, description)

FK: attribute_ID REFERENCES attribute

FK: type_ID REFERENCES type

Made_of rappresenta la relazione indicata con R5 nello schema concettuale completo.

sample (sample_ID, title, source_name, organism, extract_protocol, label, label_protocol, description, molecule, growth_ptotocol, treatmente_protocol)

s_s(sample_ID, series_ID)

FK: sample_ID REFERENCES sample

FK: series_ID REFERENCES series

contains (sample_ID, series_ID, attribute_ID, value)

FK: sample_ID, series_ID REFERENCES s_s

FK: attribute_ID REFERENCES attribute

Contains rappresenta la relazione indicata con R4 nello schema concettuale completo.

characteristics(order_number, sample_ID, description)

FK: sample_ID REFERENCES sample

4.5 Codice creazione database

Di seguito viene riportato il codice per la creazione del DB secondo le specifiche precedentemente analizzate. Il codice è basato sulla sintassi standard SQL92, tuttavia presenta istruzioni specifiche del DBMS *SQLServer*. Dove necessaria è stata riportata una traduzione equivalente per l'implementazione in DBMS differenti come ad esempio *MySQL*.

```
CREATE SCHEMA [NuovoSistema]
GO
CREATE TABLE project(
        project_ID INTEGER PRIMARY KEY IDENTITY (0,1),
        name CHARACTER VARYING(1000),
        description CHARACTER VARYING(10000)
        /*
         * sintassi alternativa per la definizione della chiave primaria:
         * PRIMARY KEY(ID_project)
         */
);
/*      Nota importante: sebbene la sintassi di base sia comune (SQL92) alcune istruzioni specifiche, come ad esempio l'incremento automatico di un campo, richiedono comandi differenti a secondo dal DBMS che si vuole utilizzare. La sintassi utilizzata nell'istruzione che precede questo commento, "KEY IDENTITY(0,1)", permette di gestire l'auto increment del campo project_ID ma è valida solo se si utilizza SQLServer. Nel caso in cui si costruisca il DB in un DBMS differente occorre, probabilmente, utilizzare una differente sintassi. Per esempio, se il DBMS di riferimento fosse mySQL l'istruzione dovrebbe avere una sintassi del tipo:

                        "project_ID INTEGER PRIMARY KEY AUTO_INCREMENT"
*/
GO
CREATE TABLE type_(
        type_identifier INTEGER PRIMARY KEY IDENTITY (0,1),
        defined_by CHARACTER VARYING(1000)
);
GO
CREATE TABLE series(
        series_ID CHARACTER VARYING(20),
        title CHARACTER VARYING(1000),
        overall_design CHARACTER VARYING(1000),
        type_identifier INTEGER REFERENCES type_,
PRIMARY KEY (series_ID)
);
GO
CREATE TABLE project_series(
        project_ID INTEGER,
        series_ID CHARACTER VARYING(20),
PRIMARY KEY (project_ID, series_ID),
FOREIGN KEY (project_ID) REFERENCES project,
FOREIGN KEY (series_ID) REFERENCES series
);
/*      Nota: la sintassi utilizzata per la creazione dalla tabella project_series è valida quando l'istruzione viene eseguita in SQLServer. Nel caso in cui si voglia costruire la tabella in un DBMS come mySQL bisognerebbe prediligere la sintassi seguente:
CREATE TABLE project_series(
        project_ID INTEGER REFERENCES project,
        series_ID CHARACTER VARYING(20) REFERENCES series,
PRIMARY KEY (project_ID, series_ID)
);
oppure
CREATE TABLE project_series(
        project_ID INTEGER REFERENCES project,
        series_ID CHARACTER VARYING(20),
PRIMARY KEY (project_ID, series_ID),
FOREIGN KEY (series_ID) REFERENCES series(series_ID)
```

```
);
*/
GO
CREATE TABLE summary(
        order_number INTEGER IDENTITY (0,1),
        series_ID CHARACTER VARYING(20),
        description_ CHARACTER VARYING(1000) NOT NULL,
/* la clausola NOT NULL serve ad evitare la presenza di tuple prive di significato. La relazione summary, infatti, è presente solo per permettere l'inserimento di valori multipli dello stesso attributo, un' eventuale descrizione "vuota" non avrebbe senso di esistere. Al momento dell'importazione dei dati sarà necessario eseguire controlli su eventuali campi summary nulli*/
PRIMARY KEY (order_number,series_ID),
FOREIGN KEY (series_ID) REFERENCES series
);
/* Nota: valgono tutte le considerazioni riguardanti i codici di creazione delle tabelle precedenti che, d'ora in avanti, verranno omesse per semplificare la lettura. Il codice da utilizzare se si predilige mySQL a SQLServer dovrebbe essere del tipo:
CREATE TABLE summary(
        order_number INTEGER AUTO_INCREMENT,
        series_ID CHARACTER VARYING(20) REFERENCES series,
        description_ CHARACTER VARYING(1000) NOT NULL,
PRIMARY KEY (order_number,series_ID)
);
 */
GO
CREATE TABLE contributor(
        contributor_name CHARACTER VARYING(800),
        series_ID CHARACTER VARYING(20),
PRIMARY KEY (contributor_name, series_ID),
FOREIGN KEY (series_ID) REFERENCES series
);
/*Nota: 800 rappresenta il numero massimo di caratteri che una chiave primaria di tipo VARCHAR(X) può possedere se la tabella viene implementata in SQLServer. Nel caso in cui la tabelle venga implementata in mySQL il limite superiore scende a 767 caratteri*/
GO
CREATE TABLE attribute(
        attribute_ID INTEGER PRIMARY KEY IDENTITY (0,1),
        description CHARACTER VARYING(1000)
);
GO
CREATE TABLE made_of(
        attribute_ID INTEGER REFERENCES attribute,
        type_identifier INTEGER REFERENCES type_,
        description_ CHARACTER VARYING(1000),
PRIMARY KEY(attribute_ID,type_identifier)
);
GO
CREATE TABLE sample_(
        sample_ID CHARACTER VARYING(20),
        title CHARACTER VARYING(1000),
        source_name CHARACTER VARYING(1000),
        organism CHARACTER VARYING(1000),
        extract_protoco CHARACTER VARYING(1000),
        label CHARACTER VARYING(1000),
        label_protocol CHARACTER VARYING(1000),
        description_ CHARACTER VARYING(1000),
        molecule CHARACTER VARYING(1000),
        growth_protocol CHARACTER VARYING(1000),
        treatment_protocol CHARACTER VARYING(1000),
        platform_ID CHARACTER VARYING (20),
PRIMARY KEY (sample_ID)
);
GO
CREATE TABLE s_s(
        sample_ID CHARACTER VARYING(20),
        series_ID CHARACTER VARYING(20),
PRIMARY KEY(sample_ID, series_ID),
FOREIGN KEY(sample_ID) REFERENCES sample_,
FOREIGN KEY(series_ID) REFERENCES series
);
GO
CREATE TABLE contains_(
        sample_ID CHARACTER VARYING(20),
        series_ID CHARACTER VARYING(20),
        attribute_ID INTEGER,
        value CHARACTER VARYING(1000),
PRIMARY KEY(sample_ID, series_ID, attribute_ID),
FOREIGN KEY (sample_ID, series_ID) REFERENCES s_s,
FOREIGN KEY (attribute_ID) REFERENCES attribute
);
GO
CREATE TABLE characteristics(
        order_number INTEGER IDENTITY (0,1),
        sample_ID CHARACTER VARYING(20),
        description_ CHARACTER VARYING(1000),
PRIMARY KEY (order_number, sample_ID),
FOREIGN KEY (sample_ID) REFERENCES sample_
);
```

4.6 Schema delle relazioni

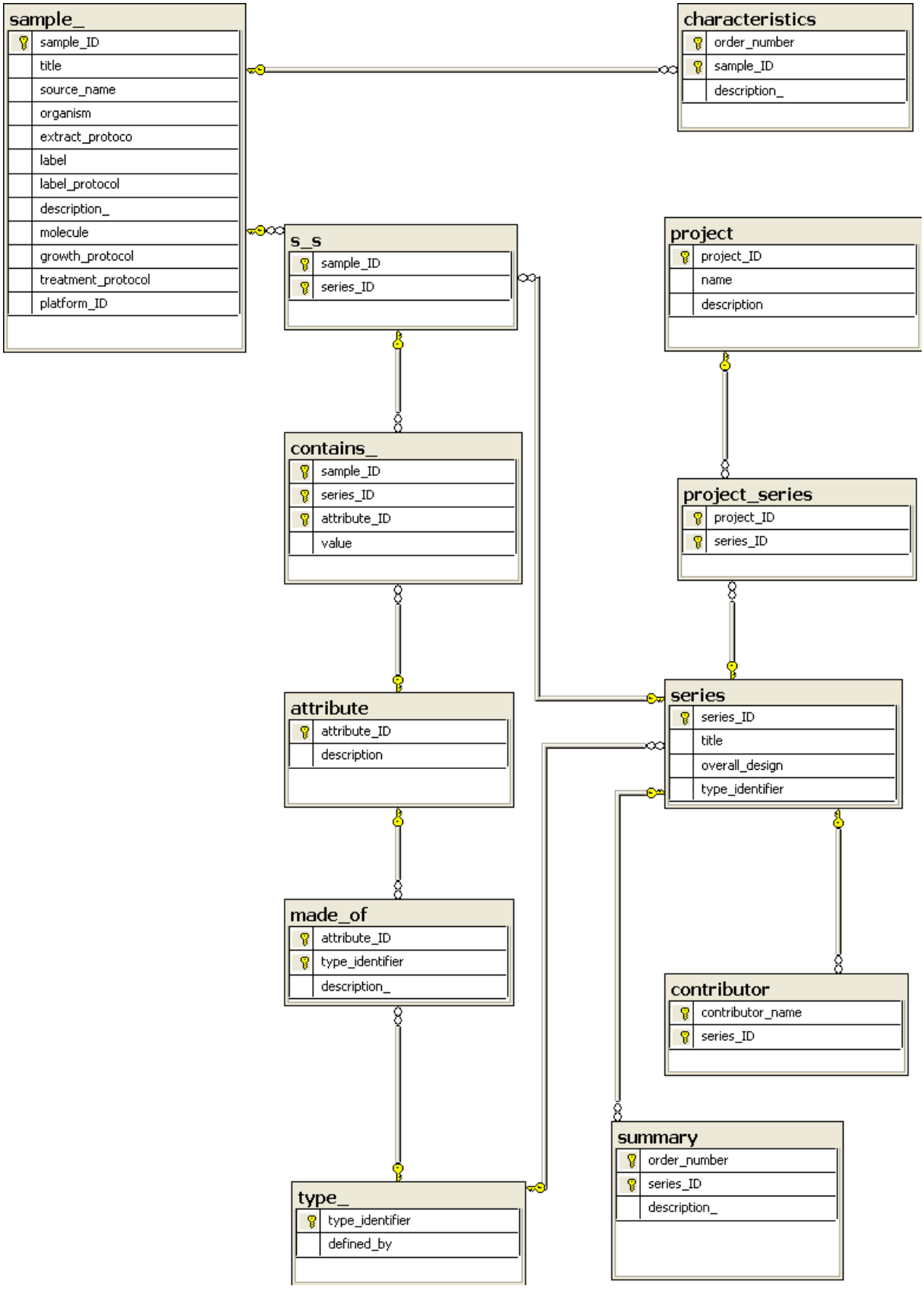

Figura 37: Schema delle relazioni

4.7 Query

Interrogazioni e creazione di *basket* (v. Capitolo 3) sono operazioni eseguite con molta frequenza dal gruppo di ricerca. L'obiettivo principale di quasi tutte le query richieste è quello di estrarre le informazioni primarie dei campioni, e degli esperimenti a cui essi sono associati, che rispettano determinati parametri. Nel 90% dei casi, le condizioni, che possono essere anche molto complesse, riguardano i cosiddetti *tag*, cioè attributi dinamici aggiungi *localmente* dai ricercatori del laboratorio e variabili tra esperimenti e campioni differenti (vedere Figura 38 e fare riferimento al Capitolo 3: Le Sorgenti dei Dati).

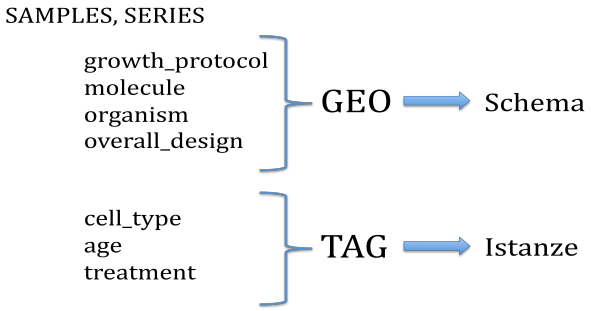

Figura 38: attributi fissi VS attributi variabili

Nel presente paragrafo si vogliono mostrate un paio di interrogazioni tipiche e "generali" (ricoprono e sono adattabili alla maggior parte dei casi) che coinvolgono i *tag*.

E' importante ribadire che un *tag* è un attributo e quindi come tale avrà associato un valore: questo avviene grazie alla tabella *contains_*.

I *tag* vengono inseriti dall'utente per descrivere un campione/esperimento e possono esse *riutilizzati* anche per altri campioni/esperimenti: in particolare si manifesta una certa regolarità per tipologia di esperimento. Allo scopo di facilitare l'utilizzo del sistema, si è deciso allora di introdurre, oltre alla tabella che contiene il nome del tag con una sua descrizione, anche una tabella con il tipo di esperimento e quindi un'associazione *molti-a-molti* rappresentata dalla tabella *made_of* che associa ogni tipo di esperimento ad un insieme di *tag*. In questo modo l'utente può *riutilizzare* i gruppi di *tag* per gli esperimenti dello stesso tipo.

Le peculiarità delle interrogazioni che coinvolgono i tag è legata a questa loro rappresentazione: un predicato su un *tag* del tipo

Tag Operatore Valore

richiede un accesso alla tabella *contains_* usando come chiave d'accesso il *tag*, la *series* e il *samples.* Questo per ogni *tag,* quindi, per poter estrarre informazioni derivanti da espressioni logiche sulle *tag* è necessario ricorrere all'utilizzo di *self join*.

Per prima cosa potrebbe essere opportuno costruire una vista contenente tutte le informazioni relative ad un determinato progetto e agli esperimenti/campioni che lo compongono.

```
CREATE VIEW info_project
        AS      SELECT DISTINCT sample.*, series.*, project.name
                FROM ((((project JOIN project_series ON (project.project_ID = project_series.project_ID))
                        JOIN series ON (series.series_ID = project_series.project_ID))
                        JOIN s_s ON (series.series_ID = s_s.series_ID))
                        JOIN samples_ ON (samples_.sample_ID = s_s.sample_ID))
                WHERE project.project_ID = progetto_richiesto
```

La clausola DISTINCT, seppur costosa dal punto di vista computazionale, è indispensabile per ovviare i problemi presentati nel paragrafo 4.1 relativamente a campioni duplicati all'interno dello stesso progetto. Per maggiori dettagli si vedano le pagine 24, 25 e 26. E' importante notare che la presenza della clausola DISTINCT rende la vista non aggiornabile, essa deve quindi essere ricreata in seguito ad aggiornamenti.

Prendiamo ora come riferimento la tabella *contains_* riportata in Figura 39, e supponiamo di voler estrarre le informazioni relative a tutti i campioni (di un determinato progetto) che rispettano la seguente espressione logica sulle *tag: tag1>10 AND tag2<100 AND tag3 = 'Sì.*

CONTAINS

samples	series	attribute_ID	value
1	1	TAG1	Sì
1	1	TAG2	15
1	1	TAG3	39
1	8	TAG1	No

Figura 39: Esempio tabella CONTAINS

```
SELECT info_project.*,  contains_.attribute_ID, contains_.value
FROM ((contains_ C1 JOIN info_project P ON (P.samples_ID = C1.samples_ID AND P.series_ID = C1.series_ID))
        JOIN contains_ C2 ON (C2.samples_ID = C1.samples_ID AND C2.series_ID = C1.series_ID))
        JOIN contains_ C3 ON (C3.samples_ID = C1.samples_ID AND C3.series_ID = C1.series_ID)
WHERE
        C1.attribute_ID = 'TAG1' AND C1.value>10
AND
        C2.attribute_ID = 'TAG2' AND C2.value<100
AND
        C3.attribute_ID = 'TAG3' AND C3.value = 'Sì'
```

Un *basket* significativo potrebbe essere quello che seleziona (nel progetto 2) le informazioni relative ai macrofagi che non hanno subito trattamenti e che hanno un età superiore ai 10 mesi. L'istruzione SQL sarebbe la seguente:

```
SELECT info_project.*,  contains_.attribute_ID, contains_.value
FROM ((contains_ C1 JOIN info_project P ON (P.samples_ID = C1.samples_ID AND P.series_ID = C1.series_ID))
        JOIN contains_ C2 ON (C2.samples_ID = C1.samples_ID AND C2.series_ID = C1.series_ID))
        JOIN contains_ C3 ON (C3.samples_ID = C1.samples_ID AND C3.series_ID = C1.series_ID)
WHERE  info_project.project_ID = 2
AND     C1.attribute_ID = 'cell_type' AND C1.value='macrophage'
AND     C2.attribute_ID = 'age' AND C2.value>10
AND     C3.attribute_ID = 'treatment' AND C3.value = 'No'
```

5 Il Download Semiautomatico dei Dati

5.1 Introduzione: File Transfer Protocol

Il server pubblico dal quale saranno estratti i dati è un server FTP. Nel presente paragrafo sono presentati i tratti caratteristici di server/client FTP e i fondamenti su cui il protocollo si basa. Nel seguito del capitolo verranno presentate le soluzioni implementante per il raccoglimento dei dati che precede la manipolazione e il caricamento degli stessi sul nuovo DB. Come si può facilmente intuire FTP è un protocollo per la trasmissione di dati tra host, esso si basa sul protocollo trasporto TCP (Transmission Control Protocol).

5.1.1 Cenni storici

FTP è uno dei primi protocolli di livello applicativo ed ha subito una lunga evoluzione negli anni. La prima specifica, sviluppata presso il MIT, risale al 1971 (RFC-114). L'attuale specifica fa riferimento all'RFC-959.

Gli obiettivi principali di FTP descritti nella sua RFC ufficiale furono:

- Promuovere la condivisione di file (programmi o dati)
- Incoraggiare l'uso indiretto o implicito di computer remoti.
- Risolvere in maniera trasparente incompatibilità tra differenti sistemi di memorizzazione di file tra host.
- Trasferire dati in maniera affidabile ed efficiente.

In realtà, l'ultima specifica non è rispettata. Il protocollo prevede un meccanismo di autenticazione, ma le informazioni di *login* viaggiano in chiaro. Questo meccanismo rende l'utilizzo di tale protocollo poco sicuro.

Il motivo per cui si è diffuso e ancora oggi risulta ampiamente utilizzato è legato alla possibilità di gestire informazioni pubbliche, che non richiedono autenticazione, instaurando connessioni anonime. Quando è richiesta la prerogativa di cifratura delle informazioni, occorre utilizzare nuove versioni del protocollo come ad esempio FTPS.

5.1.2 Il modello

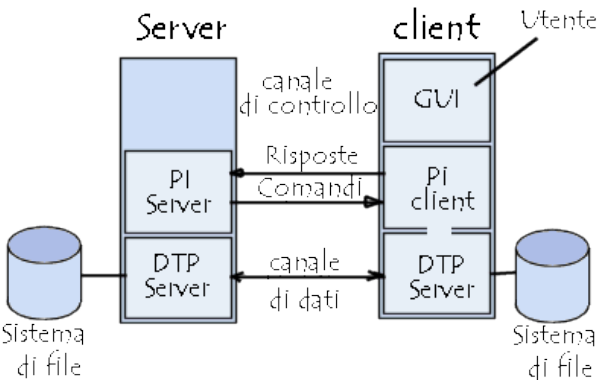

Figura 40: Modello FTP

- **PI** (*protocol interpreter*) è l'interprete del protocollo, utilizzato da client (User-PI) e server (Server-PI) per lo scambio di comandi e risposte. Solitamente ci si riferisce ad esso come "canale comandi".
- **DTP** (*data transfer process*) è il processo di trasferimento dati, utilizzato da client (User-DTP) e server (Server-DTP) per lo scambio di dati. Solitamente ci si riferisce ad esso come "canale dati".

5.1.3 Funzionamento generale

FTP, a differenza di altri protocolli come per esempio HTTP o SMTP, utilizza due connessioni separate per gestire comandi e dati. Un server FTP generalmente rimane in ascolto sulla porta 21 TCP a cui si connette il client. La connessione da parte del client determina l'inizializzazione del canale comandi attraverso il quale client e server si scambiano comandi e risposte. Lo scambio effettivo di dati (come per esempio un file) richiede l'apertura del canale dati, che può essere di due tipi:

In un canale dati di tipo *attivo* il client apre una porta solitamente casuale, tramite il canale comandi rende noto il numero di tale porta al server e attende che si connetta. Una volta che il server ha attivato la connessione dati al client FTP, quest'ultimo effettua il binding della porta sorgente alla porta 20 del server FTP.

In un canale dati di tipo *passivo* il server apre una porta solitamente casuale (superiore alla 1023), tramite il canale comandi rende noto il numero di tale porta al client e attende che si connetta.

Sia il canale comandi, sia il canale dati sono delle connessioni TCP; FTP crea un nuovo canale dati per ogni file trasferito all'interno della sessione utente, mentre il canale comandi rimane aperto per l'intera durata della sessione utente, in altre parole il canale comandi è persistente mentre il canale dati è non persistente.

Un server FTP offre svariate funzioni che permettono al client di interagire con il suo file system e i file che lo popolano, tra cui:

- Download/upload di file;
- Resume di trasferimenti interrotti;
- Rimozione e rinomina di file;
- Creazione di directory;
- Navigazione tra directory.

Ovviamente le funzioni di modifica e di upload non sono di interesse per il nostro progetto.

Come preannunciato nell'introduzione, FTP fornisce un sistema di autenticazione in chiaro degli accessi. Il client che si connette potrebbe dover fornire delle credenziali a seconda delle quali gli saranno assegnati determinati privilegi per poter operare sul file system. L'autenticazione cosiddetta *anonima* prevede che il client non specifichi nessuna password di accesso e che lo stesso abbia privilegi che sono generalmente di *sola lettura*.

5.1.4 Codici di risposta

Di seguito vengono riportati i principali codici di risposta del protocollo FTP.

- 1xx: Risposta positiva preliminare. L'azione richiesta è iniziata ma ci sarà un'altra risposta ad indicare che essa è effettivamente completata.
- 2xx: Risposta positiva definitiva. L'azione richiesta è completata. Il client può ora mandare altri comandi.
- 3xx: Risposta positiva intermedia. Il comando è stato accettato ma è necessario mandarne un secondo affinché la richiesta sia completata definitivamente.
- 4xx: Risposta negativa temporanea. Il comando non è andato a buon fine ma potrebbe funzionare in un secondo momento.
- 5xx: Risposta negativa definitiva. Il comando non è andato a buon fine e il client non dovrebbe più ripeterlo.
- x0x: Errore di sintassi.
- x1x: Risposta ad una richiesta informativa.
- x2x: Risposta relativa alla connessione.
- x3x: Risposta relativa all'account e/o ai permessi.
- x4x: Non meglio specificato.
- x5x: Risposta relativa al file-system.

5.1.5 Problemi relativi alla sicurezza

Come già detto nei paragrafi precedenti la specifica originale di FTP non prevede alcuna cifratura per i dati scambiati tra client e server. Questo comprende nomi utenti, password, comandi, codici di risposta e file trasferiti i quali possono essere "sniffati" o visionati da malintenzionati. Il problema è comune a diversi altri protocolli utilizzati prima della diffusione di SSL quali HTTP, TELNET e SMTP. Per ovviare al problema è stata definita una nuova specifica che aggiunge al protocollo FTP originale un layer di cifratura SSL/TLS più una nuova serie di comandi e codici di risposta. Il protocollo prende il nome di FTPS ed è definito nella RFC-4217.

5.2 Connessione FTP

Nel presente paragrafo verranno illustrati i principali comandi del programma ftp, disponibile nella configurazione standard della maggior parte delle distribuzioni Linux, e apparso per la prima volta nel BSD4.2. Gli esempi riportati faranno riferimento al server FTP NCBI ftp.ncbi.nlm.nih.gov all'interno del quale sono contenute tutte le informazione di interesse per questo progetto: primo fra tutti il database GEO.

5.2.1 Connessione tramite browser

E' possibile interrogare i server ftp tramite browser, essi offrono sicuramente un'interfaccia più vicina all'utente. Tuttavia, questa strada non è percorribile, senza inutili complicazioni, nel caso in cui sia necessario il download automatico di file.

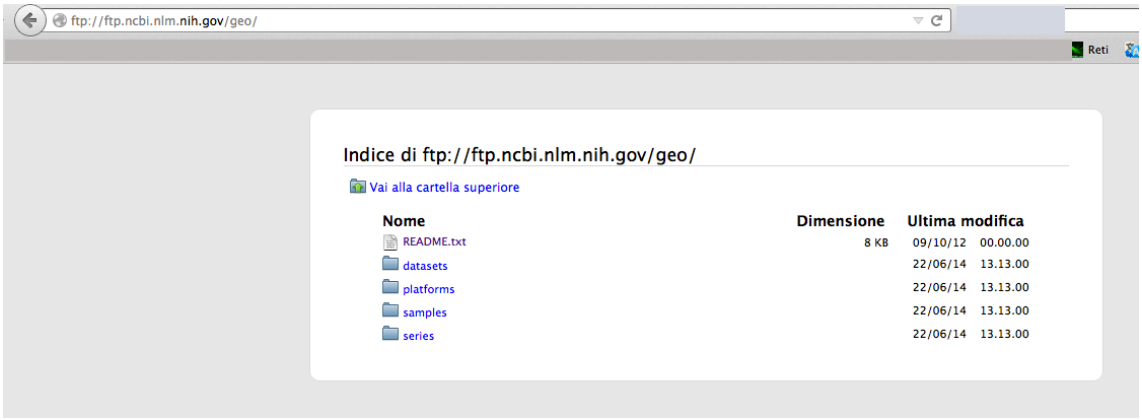

Figura 41: Connessione FTP tramite browser

5.2.2 Connessione da linea di comando (ambiente Unix)

Tramite il comando `$ ftp ftp.ncbi.nlm.nih.gov` (`$ ftp 130.14.250.7`) è possibile creare una connessione FTP al server ftp.ncbi.nlm.nih.gov. Il comando può accettare anche un indirizzo IP oltre che un nome simbolico[8]: questo permette l'utilizzo del programma ftp anche quando il sistema DNS non è operativo.

Il risultato dell'esecuzione del comando sopra specificato è rappresentato in Figura 42.

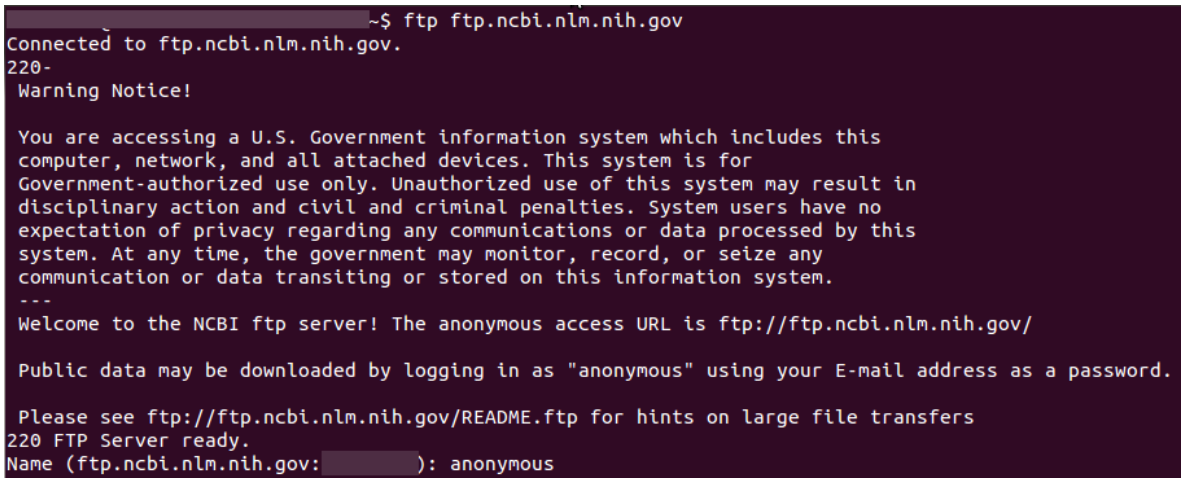

Figura 42: Connessione FTP da linea di comando

Il comando di richiesta della connessione viene raccolto dal server FTP che risponde al client richiedendo le credenziali di autenticazione. Nel caso specifico verrà utilizzata una connessione anonima. Questo tipo di connessione richiede come password un indirizzo e-mail valido. Al termine della sessione di autenticazione, nel caso in cui la connessione sia stabilita senza errori, vengono visualizzate informazioni riguardanti il SO che gira sul

[8] Pratica sconsigliata dal momento che l'indirizzo IP associato all'host name specificato potrebbe variare nel tempo. E' possibile conoscere l'indirizzo IP di un host di cui si conosce solamente il nome simbolico tramite l'utilizzo del semplice comando *ping*.

sistema remoto e la modalità di trasferimento dei file attiva, come mostrato in figura.

```
331 Anonymous login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Figura 43: Log In FTP da linea di comando

Se necessario è possibile modificare la modalità di trasferimento dei file tramite i comandi `ascii` o `binary`. La modalità binaria trasferisce i file, bit per bit, così come essi si trovano sul server FTP. La modalità ascii, invece, trasferisce direttamente il testo.

5.2.3 Navigare nel file system del server e trasferire dati

Il vantaggio principale del protocollo FTP è che esso usa molti dei comandi disponibili nei sistemi Unix. E' possibile, ad esempio, navigare nei direttori tramite il comando `cd` piuttosto che visualizzare il contenuto della directory corrente tramite il comando `ls`.

```
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
dr-xr-xr-x   4 ftp      anonymous     4096 Feb 26  2013 1000genomes
```

▪ ▪ ▪

```
dr-xr-xr-x   6 ftp      anonymous     4096 Feb 18 23:20 gene
dr-xr-xr-x 172 ftp      anonymous    16384 Jun 20 20:43 genomes
dr-xr-xr-x 1073741824 ftp      anonymous        0 Jun 22 04:02 geo
dr-xr-xr-x   4 ftp      anonymous     4096 Feb  6 00:36 giab
```

▪ ▪ ▪

```
dr-xr-xr-x   5 ftp      anonymous     4096 Apr 24  2009 tpa
dr-xr-xr-x   4 ftp      anonymous     4096 Sep 13  2012 variation
226 Transfer complete
ftp> cd geo
250 CWD command successful
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
-r--r--r--   1 ftp      anonymous     7197 Oct  9  2012 README.txt
dr-xr-xr-x 1073741824 ftp      anonymous        0 Jun 22 04:02 datasets
dr-xr-xr-x 1073741824 ftp      anonymous        0 Jun 22 04:02 platform
dr-xr-xr-x 1073741824 ftp      anonymous        0 Jun 22 04:02 samples
dr-xr-xr-x 1073741824 ftp      anonymous        0 Jun 22 04:02 series
226 Transfer complete
ftp>
```

Figura 44: Navigare nel file system del server

Una volta individuato il file da scaricare è possibile trasferirlo in locale tramite il comando `get`. Esso è seguito dal *path name* del file remoto e, opzionalmente, dal *path name* che questo dovrà avere in locale.

```
ftp> get README.ftp
local: README.ftp remote: README.ftp
200 PORT command successful
150 Opening BINARY mode data connection for README.ftp (1868 bytes)
226 Transfer complete
1868 bytes received in 0.01 secs (124.4 kB/s)
ftp>
```

Figura 45: Download da server FTP

Il protocollo FTP offre, inoltre, la possibilità di scaricare più di un file alla volta, tramite il comando `mget` (multiple get). Si possono specificare i file che mget deve scaricare come una lista di nomi di file separati da spazi, oppure tramite l'uso dei caratteri jolly. Per esempio:

```
ftp> mget gen*
```

5.3 File system del server Gene Expression Omnibus

La radice dei dati contenuti in GEO, come si può notare in Figura 41, è ftp://ftp.ncbi.nlm.nih.gov/geo/. Questo direttorio è caratterizzato da quattro sottodirettori, ognuno dei quali corrisponde ad una delle entità contenute in GEO.

- `datasets/`
- `platforms/`
- `samples/`
- `series/`

I dati non sono direttamente contenuti nelle cartelle sopra specificate ma sono suddivisi in range di ulteriori sottodirettori. Questo permette di risolvere problemi legati al timeout del browser. I range di subdirectories sono creati sostituendo gli ultimi tre caratteri con le lettere "nnn". Per esempio ftp://ftp.ncbi.nlm.nih.gov/geo/datasets/GDS1nnn/ contiene tutti i dati riportati nei sottodirettori GDS1001, GDS1002, ..., e GDS1995.

5.3.1 DataSets

Contiene un unico sottodirettorio `soft/` il quale riporta due tipi di file:

- `GDSxxx.soft.gz:` file soft in formato gzipped. Il contenuto di tale file non è altro che un estratto del valori dei campioni originari che costituiscono il dataset.
- `GDSxxx_full.soft.gz:` file soft in formato gzipped. In più, rispetto al precedente, contiene annotazioni di espressione genica riguardanti la piattaforma utilizzata per l'elaborazione degli esperimenti.

5.3.2 Series

I sottodirettori sono:

- `matrix/` che contiene i seguenti file:
  - `GSExxx_series_matrix.txt.gz:` riassunto testuale (in forma compressa) dell'esperimento. Esso comprende la matrice dei valori in formato tab delimitato generata dalle colonne di tutti i record *sample*. I dati generati da piattaforme diverse sono riportati in file differenti all'interno

dello stesso sottodirettorio.

- `miniml/` che contiene i seguenti file:
  - `GSExxx_family.xml.tgz:` file gzipped contenente i dati di un esperimento in formato MINiML. Il file contiene, inoltre, tutti i dati dei campioni e delle piattaforme associate all'esperimento. Esso costituisce l'elemento fondamentale per il recupero dei dati nell'ambito del presente progetto.
- `soft/`
  - `GSExxx_family.soft.gz:` contiene le stesse informazioni riportate nel file `GSExxx_family.xml,` l'unica differenza risiede nel formato che, in questo caso, come si può facilmente notare dall'estensione, è di tipo SOFT.
- `suppl/`
  - `GSExxx_RAW.tar:` contiene tutti i file supplementari corrispondenti ad un esperimento, ovvero tutti i file che coloro che effettuano la sottomissione vogliono mettere a disposizione (es. .CEL, .GPR o cDNA).

5.3.3 Platform

I sottodirettori sono:

- `annot/`
  - `GPLxxx.annot.gz:` file di *annotazione*[9] per le piattaforme che "partecipano" ai vari dataset.

- `miniml/`
  - `GPLxxx_family.xml.tgz:` file in formato MINiML contenente dati relativi a campioni processati tramite determinate piattaforme e appartenenti ad un gruppo di esperimenti.

- `soft/`
  - `GPLxxx_family.soft.gz:` contiene le stesse informazioni riportate nel file `GPLxxx_family.xml,` l'unica differenza risiede nel formato che, in questo caso, come si può facilmente notare dall'estensione, è di tipo SOFT.
- `suppl/`
  - `GPLxxx.xxx.gz:` file supplementari per una piattaforma. Essi contengono informazioni riguardanti l'array, non sono obbligatori al momento della sottomissione (es. .GAL, .XLS, .TXT).

5.3.4 Sample

I sottodirettori sono:

- `suppl/`
  - `GSMxxxxxx.xxx.gz:` file supplementari relativi ad un determinato campione. In genere contengono le informazioni di ibridazione derivanti dall'immagine originaria (es. .GPR, .CEL o .TIFF).

[9] Analisi della sequenza nucleotidica effettuata allo scopo di descrivere geni non noti e identificare geni noti nelle diverse localizzazioni cromosomiche.

5.4 Ottimizzazione del download ftp

Lo stesso NCBI dichiara che per ottenere le migliori prestazioni di download occorre che il buffer del client FTP abbia una dimensione di 32 MB. Questa modifica migliora sia download che upload, anche se quest'ultimi non sono per noi di interesse.

Per modificare il buffer nella *command line ncftp* è sufficiente utilizzare il comando *set so-bufsize*. Per modificare il buffer nella *command line* di *lukemftp* (client ftp di default per SUSE altri sistemi Unix), invece, occorre utilizzare i comandi *sndbuf* e *rcvbuf*, seguiti dalla dimensione che vogliamo assegnargli.

5.5 Scelte implementative

La scelta implementativa adottata per il download automatico dei dati di interesse prevede l'utilizzo di uno script PHP, interrogabile tramite browser. Lo script sarà installato su di un web server remoto, lo stesso che dovrà contenere i dati. L'utilizzo di uno script PHP permette di fornire all'utente una semplice interfaccia grafica, tramite il linguaggio HTML, per l'inserimento delle specifiche dei dati che si desidera scaricare. La versione beta dell'interfaccia sopra citata è riportata in Figura 46.

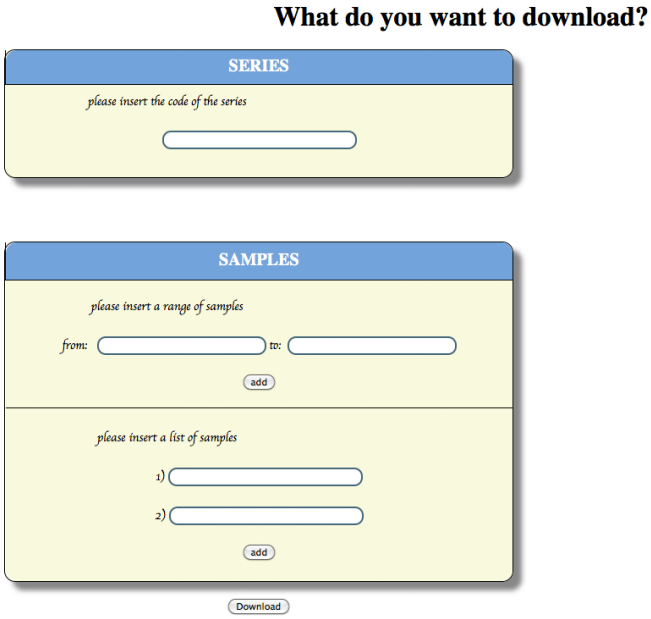

Figura 46: Interfaccia di download dei dati

Tramite il web server sarà possibile, inoltre, realizzare meccanismi di autenticazione e controllo di accesso basati su username e password. Questo tipo di soluzione permette di vietare l'accesso al server e il download a membri non autorizzati e di creare gruppi di lavoro per accessi personalizzati.

Ovviamente la connessione FTP al serve NCBI dovrà essere fatta in maniera anonima (si vedano paragrafi precedenti) e dovrà tenere conto dei parametri di ottimizzazione e della struttura del file system remoto esplicate nei paragrafi 5.3 e 5.4. Il linguaggio PHP offre una serie di comandi "pronti" per la gestione di tutti gli aspetti citati.

6 Sistemi ETL

I dati sono definiti come trasportatori di informazione e raramente hanno valore per l'utente. Un dato è ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione. In informatica: sono elementi di informazione costituiti da simboli che debbono essere elaborati. Le informazioni sono dati aggregati ad un livello in cui hanno senso per il supporto decisionale. L'informazione è una notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere. Perciò, per completare il progetto, è necessario utilizzare strumenti in grado di estrarre i dati dalle sorgenti individuate nel Capitolo 3 e produrre l'informazione necessaria per il popolamento del nuovo DB creato. In questo modo si viene a realizzare il secondo punto fondamentale del progetto: l'estrapolazione e il caricamento automatico/semiautomatico dei dati pubblici sulla base di dati appositamente strutturata.

Ad oggi esistono strumenti specializzati nell'estrazione dei dati e nella loro trasformazione: strumenti ETL. Essi svolgono in modo automatico (o parzialmente automatico) le funzioni di estrazione (*Extract*), trasformazione (*Transform*) e caricamento (*Load*) dei dati in un database ed in particolare, in un data warehouse[10]. Gli ETL sono un insieme di programmi semi automatizzati (perché raramente il processo di estrazione avviene in modo automatico) che consentono di individuare i dati da estrarre, permettono di estrarli, trasformarli e caricarli.

- *Estrazione*: i dati vengono estratti dalle fonti interne e esterne sulla base di un approccio che prevede una fase iniziale nella quale il database "vuoto" viene alimentato con i dati disponibili riferiti a periodi passati e fasi successive caratterizzate da estrazioni di natura incrementale.
- *Trasformazione*: è costituita da due fasi. Nella fase di pulitura, ci si propone di migliorare la qualità dei dati estratti dalle diverse fonti mediante la correzione di inconsistenze, inesattezze, carenze. Nella fase di trasformazione si procede ad ulteriori conversioni dei dati (che ne garantiscano l'omogeneità rispetto all'integrazione delle diverse fonti) e aggregazioni (per ottenere le sintesi necessarie a svolgere le analisi).
- *Caricamento*: i dati opportunamente estratti e trasformati vengono infine inseriti nelle strutture informative (tabelle) predisposte.

Analizzando il mercato degli strumenti specializzati nel settore dell'ETL si è deciso di utilizzare *Talend Open Studio* perché offre una grande quantità di connettori già pronti per interrogare tutti le tipologie di sorgenti.

Le funzionalità di ETL offerte da *Talend Open Studio* sono incluse nella piattaforma più generale di Integrazione Dati, per questo motivo il nome *specifico* spesso utilizzato è *Talend Open Studio* for *Data Integration*. Da un punto di vista generale, per Data Integration si intende

[10] Un *data warehouse* (o DW, o DWH) (termine inglese traducibile con magazzino di dati) è un archivio informatico contenente i dati di un'organizzazione, progettati per consentire di produrre facilmente analisi e relazioni utili a fini decisionali-aziendali. Componenti essenziali di un sistema data warehouse sono anche gli strumenti per localizzare, estrarre, trasformare e caricare i dati, come pure gli strumenti per gestire un dizionario dei dati.

l'estrazione, la trasformazione e l'integrazione di informazioni da più sorgenti dati, risolvendo di eventuali conflitti tra dati, in modo da costruire una *vista virtuale unificata* che possa essere interrogata dall'utente in modo trasparente rispetto alle sorgenti. In questo lavoro di tesi non ci occupiamo di integrazione dati, in quanto si considera una sola sorgente, ovvero GEO; l'utilizzo di tecniche di Integrazione Dati saranno importanti quando il sistema dovrà gestire dati provenienti da più *microarray* database, come verrà discusso nel capitolo degli sviluppi futuri.

6.1 Talend Open Studio

Talend Open Studio (TOS) è un software Open Source sviluppato da un azienda francese, l'omonima *Talend*. TOS è distribuito sotto licenza GPLv2 e ed è stato lanciato a ottobre del 2006. L'azienda che ha sviluppato il prodotto è privata, non è quotata in borsa. Inizialmente era finanziata con venture capital[11], e oggi, vanta più di 400 clienti in tutto il mondo, tra cui eBay, Yahoo, Sony e Virgin Mobile.
TOS, come detto, è open source, il suo guadagno principale deriva nel supporto alle aziende che integrano il loro prodotto, alla formazione e all'hosting. Oggi *Talend è* costituito da una famiglia di piattaforme che coprono le diverse esigenze richieste dal mercato. La famiglia dei prodotti è costituita da (citiamo solo le più importanti):

Prodotti per il **Data Management**:
- *Talend Open Studio for Data Integration*: soluzione open source che migliora il lavoro della DI[12] attraverso un ambiente grafico;
- *Talend Enterprise Data Integration*: soluzione con licenza a pagamento che estende le funzionalità del relativo prodotto open source e offre anche il supporto tecnico personalizzato;
- *Talend Open Studio for MDM & Talend Enterprise MDM[13]*: piattaforma per il data management, crea una vista unificata delle informazioni aziendali. Semplifica l'approccio per il *master data project* e combina funzionalità per la *data integration*, *data quality*, *data profiling*, *data mastering* e *data governace*;
- *Talend Integration Express*: è un servizio di host esteso da *Talend Open Studio* per la data integration con supporto tecnico all'azienda.

Prodotti per **Application Integration**:
- *Talend Open Studio for ESB & Talend Enterprise ESB (Enterprise Service Bus)*: è un'architettura software che attraverso il disegno permette di modellare la comunicazione/interazione tra applicativi in ambito *service-oriented architecture* (SOA);
- *Talend ESB Studio: tool* basato su Eclipse che permette la modellazione, configurazione e sviluppo di soluzioni di integrazione usando Talend ESB.

Prodotti per il **Cloud Based Integration**:
- *Talend Cloud*: è una piattaforma unificata per la *data integration*, *data quality*, *master data management* e *enterprise service bus*, che permette l'integrazione in sviluppo

[11] È il capitale finanziario fornito alle start-up in crescita, ad alto potenziale.
[12] *Data Integration*
[13] *Master Data Management*

cloud e sviluppi ibridi.

Nella famiglia *Talend*, non ci sono solo prodotti *open source*, ma anche prodotti *Enterprise* che estendono alcune funzionalità dei prodotti di base, gratuiti alle aziende. Per aver accesso a questi prodotti, bisogna comunicare direttamente con *Talend*, il quale in base alle esigenze dell'azienda, propone il costo, il tipo di servizio di assistenza e il tipo prodotto che più si adatta alle esigenze del cliente. Nel nostro progetto, visto che tratta di ETL e di DI, è stata analizzata la piattaforma *Talend Open Studio* per la *Data Integration*, più precisamente la versione 5.4.1, distribuita con licenza gratuita. Per avere un idea di cosa può offrire un prodotto *Enterprise* per la *Data Integration*, consultare la tabella che segue:

Features	Talend Open Studio for Data Integration	Talend Enterprise Data Integration
Job Designer	✓	✓
Business Modeler	✓	✓
450+ Connectors	✓	✓
Versioning	✓	✓
Shared Repository		✓
Wizards		✓
Management and Monitoring Tools		✓
Data Quality		
Clustering		
Indemnification/Warranty and Talend Support		✓
License	Open Source	Subscription

Figura 47: Confronto tra Talend Open Studio Data Integration open Source e Enterprise

Talend Open Studio per la *DI* offre una piattaforma unificata, che rende lo sviluppo e la gestione dell'integrazione dei dati molto semplice, fornendo un ambiente grafico in grado di gestire i processi aziendali e non solo. Attraverso l'uso di questa piattaforma è possibile raggiungere gli obiettivi d'integrazione dei dati, più velocemente, aumentando la produttività aziendale o, nel caso specifico, la produttività di ricerca, e, di conseguenza, i guadagni intesi non solo in senso monetario.

6.1.1 Descrizione

Talend Open Studio per DI, nel nostro progetto, è stato usato come strumento di ETL, cioè di estrazione di dati dalle sorgenti provenienti dal server FTP GEO, manipolazione e caricamento degli stessi. La piattaforma in questione è disponibile sia per sistemi Windows che per sistemi Unix; essa genera codice Java (o Perl in alternativa). I progetti ETL che

vengono creati con questo strumento, per i motivi sopra elencati, necessitano di una JVM (*Java Virtual Machine*) per poter essere eseguiti. Questa caratteristica fa sì che i progetti realizzati con *Talend* siano indipendenti dalla piattaforma: viene così facilitata l'integrazione con altre applicazioni. Come si nota dalla Figura 48, *Talend Open Studio* sfrutta l'ambiente di *Eclipse*[14], una nota piattaforma di sviluppo Java, ereditandone alcune utili funzionalità, tra le quali il *Debugger*.

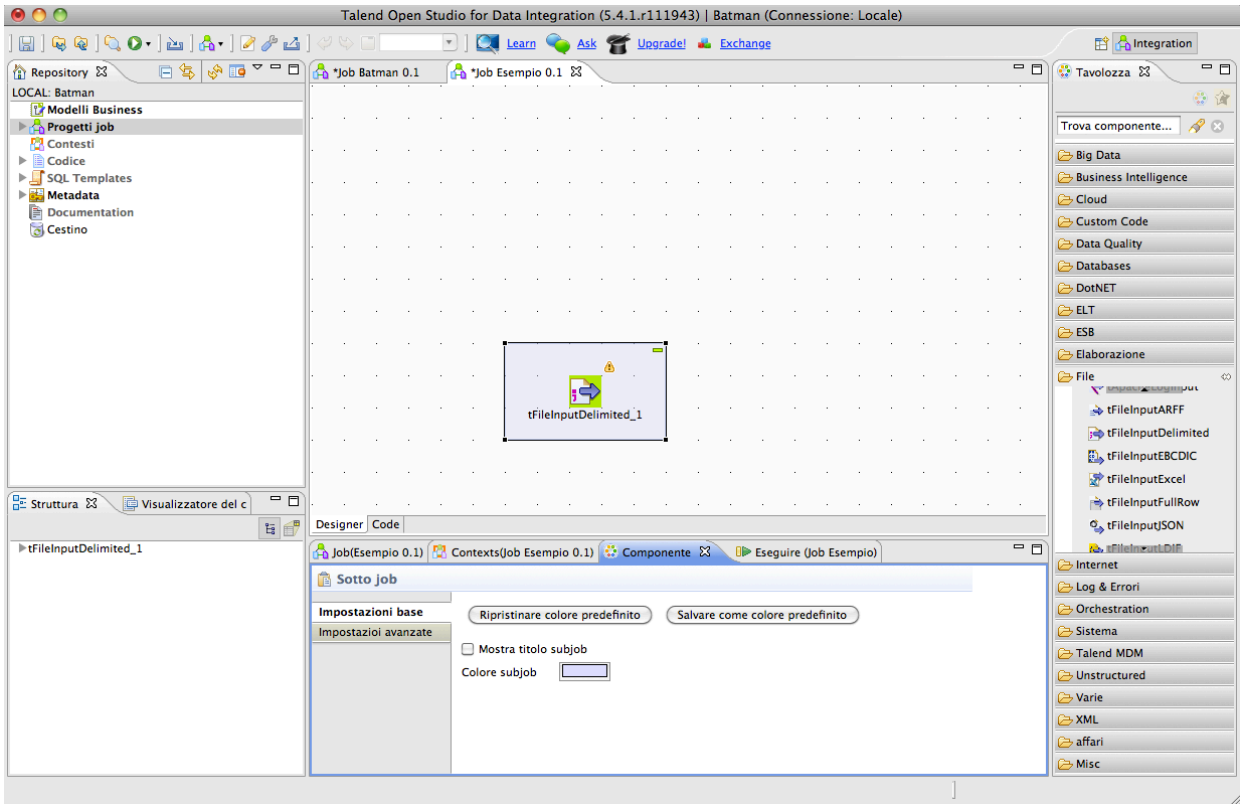

Figura 48: Interfaccia di TOS

La piattaforma di sviluppo, come si può notare, è semplice e intuitiva, perché, come accennato, richiama lo stile grafico di *Eclipse*. Per fornire un'idea di come l'ambiente di sviluppo è strutturato ne viene data una breve descrizione:

A sinistra è presente il pannello della *repository* del progetto, nel quale sono presenti:

- *Modelli Business*: permettono di disegnare graficamente i processi di ETL e la struttura dei dati;
- *Progetti Job*: costituisce la componente principale dell'IDE, tramite essa vengono creati i processi di lavorazione dei dati (job);
- *Metadata*: gestisce gli schemi dei dati e le connessioni ai database, consentendone la centralizzazione ed il riutilizzo;
- *Context*: le procedure di ETL possono essere personalizzate facendo uso di parametri. La *repository Context* consente la definizione di questi parametri;
- *Code*: l'utente può creare delle routine personalizzate, in codice Java o Perl a seconda

[14] *Eclipse* è un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma. Ideato da un consorzio di grandi società quali Ericsson, HP, IBM, Intel, MontaVista Software, QNX, SAP e Serena Software, chiamato Eclipse Foundation sullo stile dell'open source.

di come viene configurato il progetto. La *repository Code* raccoglie queste funzioni in librerie utilizzabili dagli strumenti di *Talend Open Studio*;
- *Documentazione*: file esterni possono essere integrati nel progetto per documentare le scelte adottate;
- *Cestino*: i documenti cancellati dai repository finiscono in un cestino, in attesa che l'utente ne svuoti il contenuto.

A destra sono presenti i *450+ connettori* offerti dalla suite, per potersi connettere alle sorgenti e creare le manipolazioni sui dati di cui si necessita.

Come si può notare in Figura 48, all'interno della sezione di destra è presente il componente "*tFileInpuntDelimeted*". Questo tipo di componente, giusto per fare un esempio, consente di collegarsi a file *CSV* o *TSV* di tipo delimitato. Quando lo si seleziona, nella parte bassa della GUI dell'IDE, viene proposta una serie di funzioni che il componente può offrire: è possibile, ad esempio, specificare la sorgente, il tipo di carattere di separazione, da quale riga incominciare a leggere, a quale riga fermarsi, ecc.

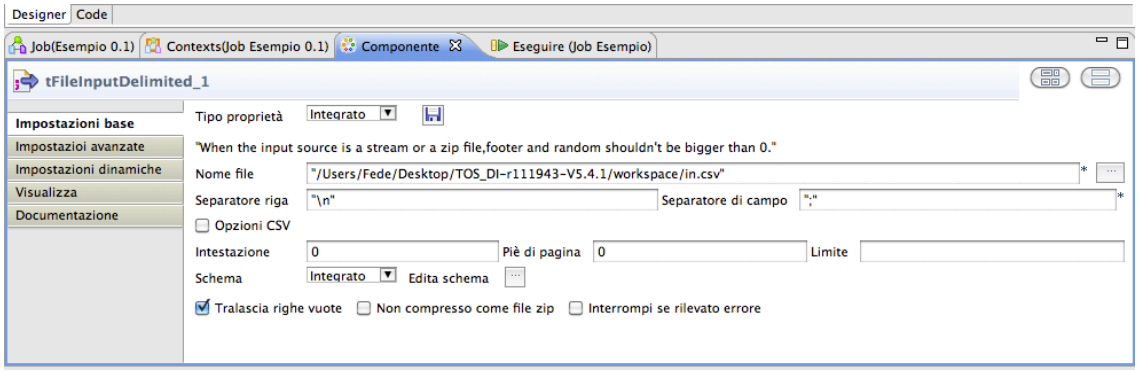

Figura 49: Impostazione dei connettori

Talend offre una moltitudine di connettori (450+), in grado di connettersi e manipolare qualsiasi sorgente; nel seguito vengono riportati alcuni esempi di sorgenti utilizzabili:

- PDF;
- CSV[15]
- TSV[16];
- Excel;
- Qualsiasi tipo di database relazionale e NoSQL[17];

15 *Comma Separated Values* (CSV) è un formato di file basato su file di testo utilizzato per l'importazione ed esportazione (ad esempio da fogli elettronici o database) di una tabella di dati. In questo formato, ogni riga della tabella (o record della base dati) è normalmente rappresentata da una linea di testo, che a sua volta è divisa in campi (le singole colonne) separati da un apposito carattere separatore, ciascuno dei quali rappresenta un valore.

16 Il formato TSV è una ulteriore variante del formato CSV, che utilizza il campo TAB (tabulazione) come separatore (*Tab Separated Values*). Questo formato viene utilizzato comunemente in diverse piattaforme (ad esempio su Macintosh è un formato molto comune).

17 NoSQL è un "movimento" che promuove sistemi software dove la persistenza dei dati è caratterizzata dal fatto di non utilizzare il modello relazionale, di solito usato dai database tradizionali (RDBMS). L'espressione NoSQL fa riferimento al linguaggio SQL, che è il più comune linguaggio di interrogazione dei dati nei database relazionali, qui preso a simbolo

- XML;
- Ftp;
- E tanti altri.

Una delle caratteristiche che ha reso *Talend* famoso e potente, è proprio la quantità di connettori che offre. Inoltre, se non esiste un connettore che soddisfi le proprie esigenze, è possibile creare un componente personalizzato con le caratteristiche desiderate. Per connettore si intendono quei componenti grafici che forniscono le funzioni per la connessione alle diverse sorgenti.

6.1.2 Caratteristiche di Talend

Nel presente paragrafo verranno descritte, analizzando un processo di estrazione dei metadati di espressione genica, le principali caratteristiche del prodotto *Talend Open Studio.* Tale processo di estrazioni consiste in:

1. Estrazione di dati da un file XML;
2. Trasformazione dei dati recuperati;
3. Inserimento dei dati trasformati in un database MySQL.

6.1.2.1 Prima Fase

La prima fase, quella di estrazione dei dati da un file XML, si compone come illustrato nel seguito:

- Definizione di un job: v. Figura 50:

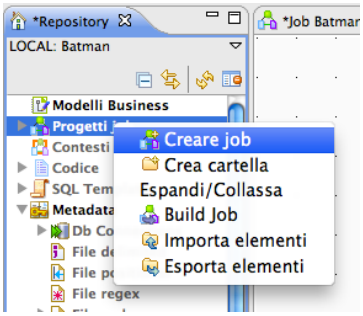

Figura 50: Creazione di un Job in talend

- Definizione della sorgente dalla quale si vogliono recuperare le informazioni: nella sezione *Metadata* è possibile selezionare la sorgente di interesse v. Figura 51:

dell'intero paradigma relazionale. Questi archivi di dati il più delle volte non richiedono uno schema fisso (schemaless), evitano spesso le operazioni di unione (join) e puntano a scalare in modo orizzontale;

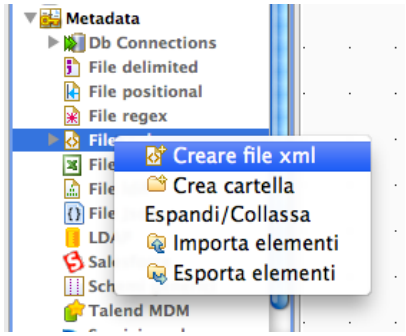

Figura 51: Gestione delle sorgenti e dei progetti

Siccome la nostra sorgente è un file xml, selezioniamo l'apposita voce. Quello che si nota, fin dai primi passaggi, è la semplicità e l'interfaccia *user friendly*, che offre questo strumento. I passaggi ovvi vengono saltati, procedendo nella creazione della sorgente xml da leggere, si arriva alla definizione dei "*loop path*", cioè il percorso dell'xml sui cui dobbiamo ciclare, per poter recuperare le informazioni. Nella Figura 52 è riportata la lettura di un file XML fornito da GEO. Il file in questione descrive in maniera formale un esperimento (series) caricato sulla banca dati, nell'esempio che segue vengono illustrati i passi per l'estrazione delle informazioni riguardanti i soggetti che hanno contribuito allo sviluppo dell'esperimento. Come si nota, con un semplice *drag&drop* si sceglie quello che sarà il percorso sul quale ciclare per recuperare le informazioni e gli elementi di interesse (v. Figura 52).

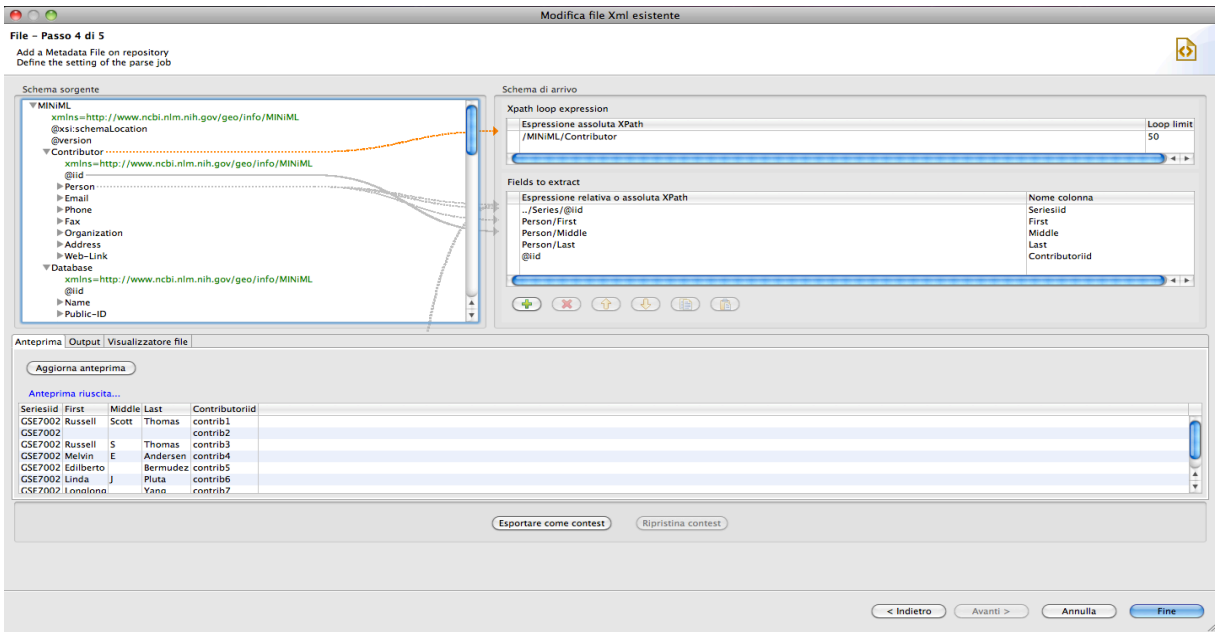

Figura 52: Editing dell'XML di un esperimento GEO

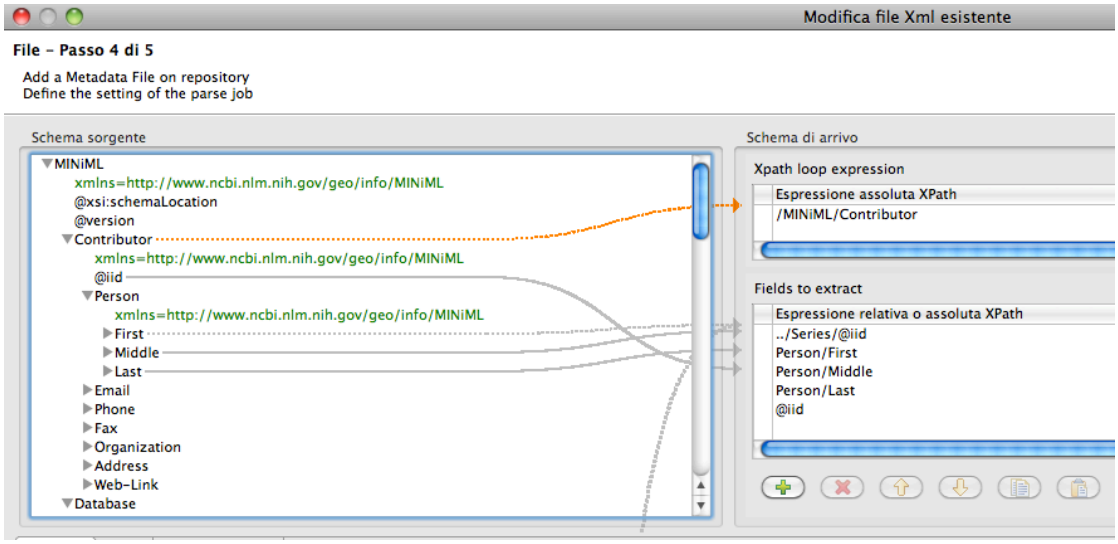

Figura 53: Definizione dei "loop path" e dei dati di interesse inerenti ai contributor

Con questo passaggio si conclude il primo punto, cioè abbiamo definito una sorgente XML dalla quale recuperiamo le informazioni. Il procedimento di definizione dei "*loop path*" e di estrazione dei dati dovrà essere ripetuto, nel nostro caso, una volta per ogni "gruppo" di dati che si necessità estratte dal file XML in questione. Questi file, infatti, contengono tutti i dati riguardanti un esperimento e i campioni che lo costituiscono: non sarebbe quindi possibile tramite la definizione di un unico "*loop path*", e quindi tramite la creazione di un unico file XML di import, gestire tutti i dati in esso contenuti. Per semplificare la lettura riporteremo di seguito solo la definizioni di estrazione XML dei dati relativi ai campioni.

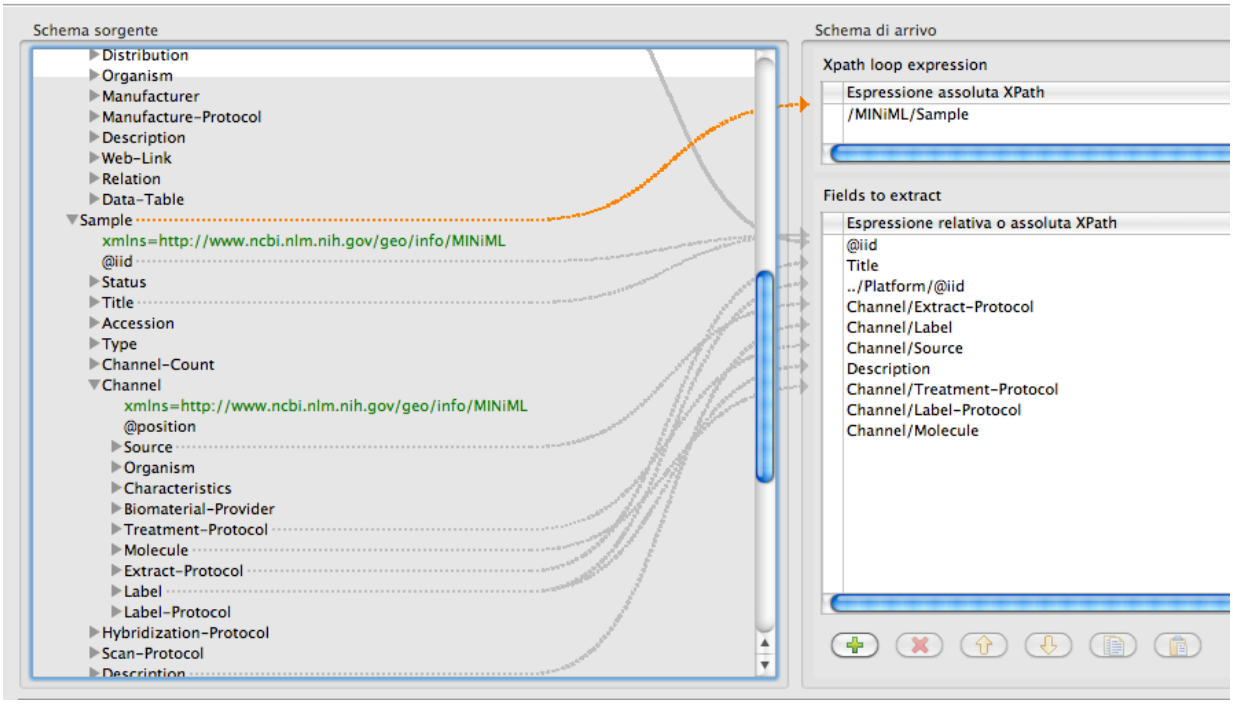

Figura 54: Definizione dei "lopp path" e dei dati di interesse inerenti al campione

Come si può notare dalla Figura 54 il sistema è estremamente potente e permette di generare "loop path" complessi per l'estrazione dei dati. E' possibile, inoltre, dopo aver generato i loop di estrazione e selezionato i dati di interesse, richiedere al sistema di generare un'anteprima di schema (v. Figura 52).

6.1.2.2 Seconda Fase

Definite le sorgenti, passiamo al secondo punto, cioè manipolare i dati che vengono estratti dalla sorgente. Ultimata la definizione degli XML questi compaiono nei *Repository*: con un semplice trascinamento, possiamo inserirli nel nostro progetto (v. Figura 55).

Quando trasciniamo i file XML, Talend ci chiede se vogliamo che quelle sorgenti:

1. Siano di input per altri componenti;
2. Si vuole solo estrarre i campi dagli XML definiti.

L'obiettivo è quello di manipolare le informazioni, per questo motivo scegliamo la prima opzione, cioè i file XML devono essere di input per altri componenti.

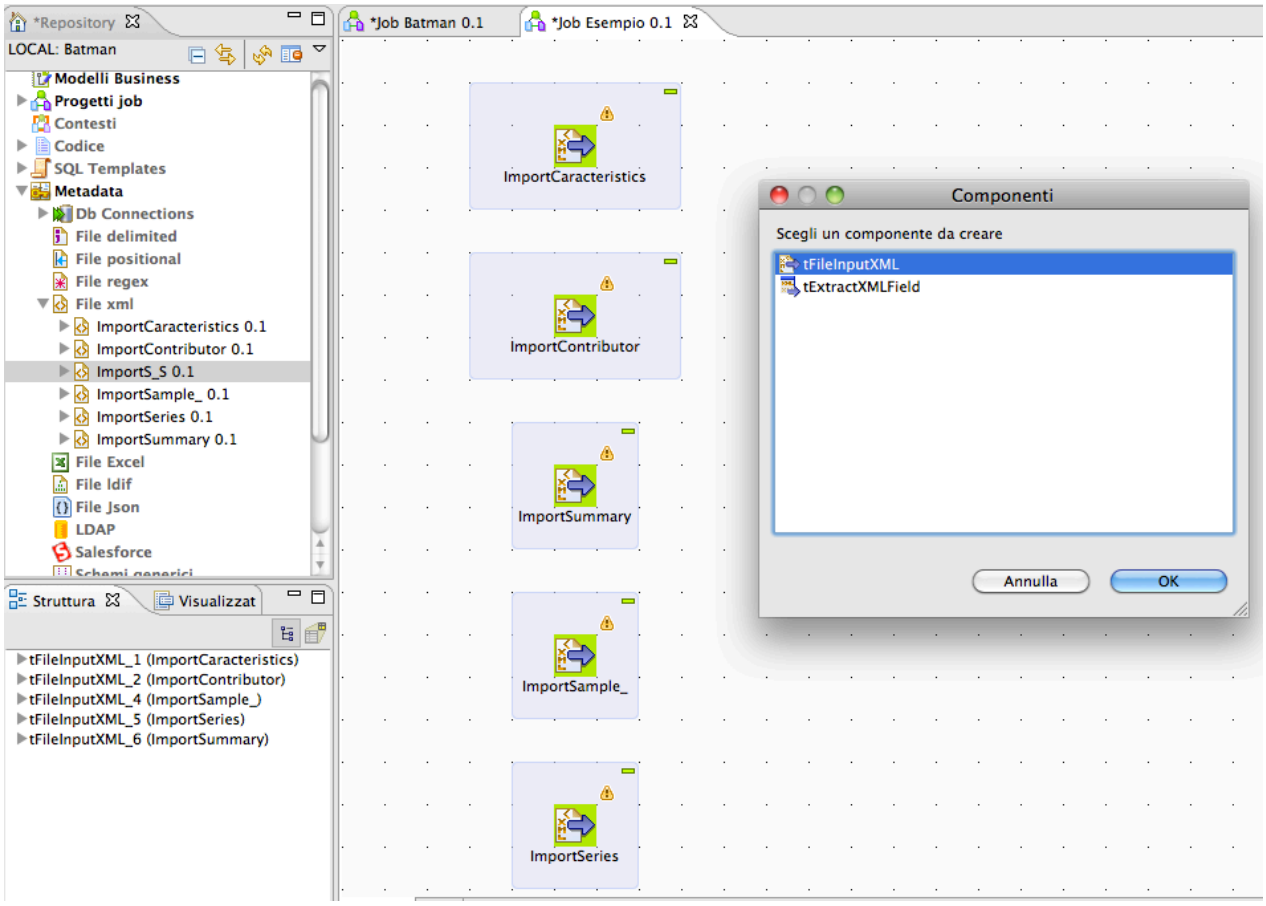

Figura 55: Aggiungere file di import al progetto

Per procedere alla manipolazione delle informazioni è necessario introdurre nuovi componenti al progetto. Nell'esempio che illustreremo saranno utilizzati due componenti fondamentali quali *tFilterRow* e *tMap*.

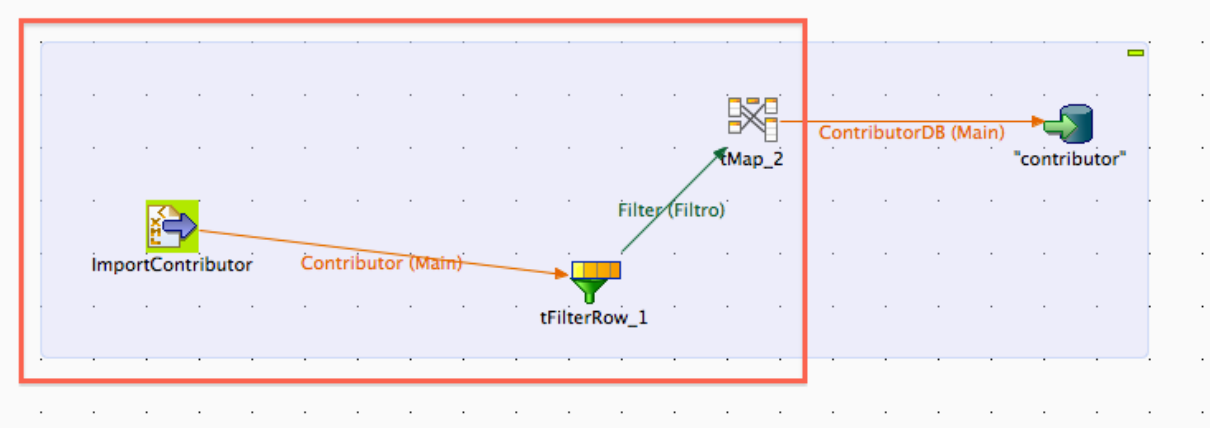

Figura 56: Filtraggio e manipolazione dei dati

Il primo, *tFilterRow*, ci permette di filtrare eventuali righe indesiderate. Facendo riferimento alla sezione *contributor* della struttura XML definita secondo il protocollo MIAME da GEO, giusto per fare un esempio, possiamo osservare come spesso siano presenti *contributor* definiti da soli valori nulli. E' evidente che tuple di questo tipo non vogliono essere riportate nello schema finale. Per questo motivo dovranno essere filtrate. Lo strumento di filtro riga ci permette di definire, attraverso operatori logici le condizioni di filtraggio.

Il secondo, *tMap*, ci consente di manipolare/trasformare/formattare le informazioni estratte. Aprendo il componente (doppio click sul componente), è possibile selezionare i campi che vogliamo memorizzare nel database (v. Figura 57).

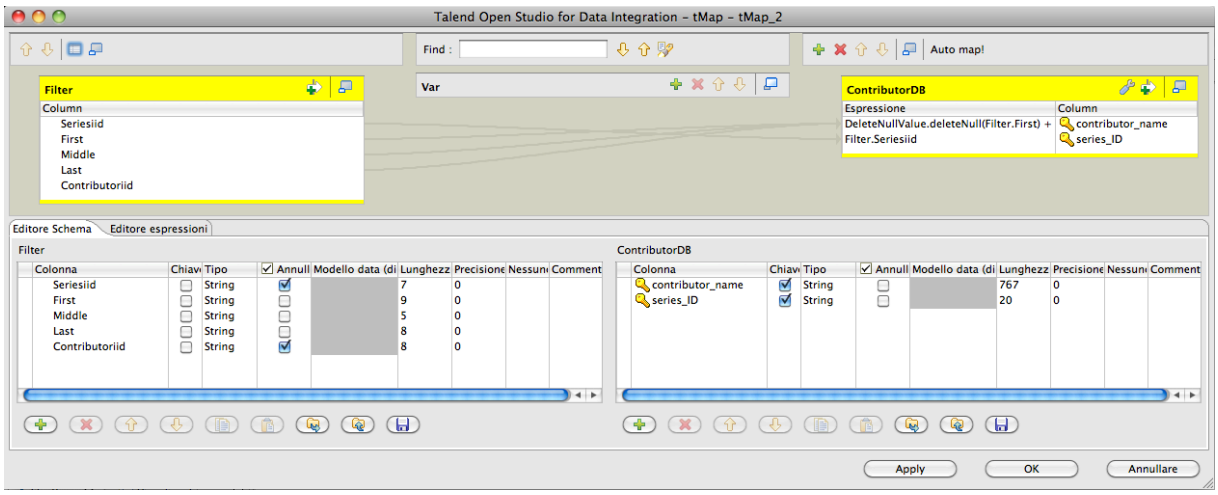

Figura 57: Manipolazione dei dati

Come si vede, attraverso il trascinamento, i campi in ingresso possono essere trasportati in uscita: in questo modo si selezionano gli attributi dell'xml di interesse. E' inoltre possibile, se necessario, definire i nomi delle colonne che deve avere la tabella di output e ridefinire gli attributi di tipo chiave ed eventuali possibili valori nulli.

Quanto fatto fino ad ora non si può definire vero ETL, ci siamo infatti limitati a selezionare

gli attributi che volevamo in uscita. Se si seleziona con doppio click uno dei campi di output, appare una finestra nella quale è possibile (v. Figura 58):

1. Scegliere il tipo di funzione di trasformazione da applicare sull'attributo selezionato;
2. Richiamare una funzione definita in una nostra classe;
3. Scrivere direttamente il codice java che definisce la trasformazione.

Nell'esempio riportato in figura l'operazione di manipolazione consiste nella concatenazione delle stringhe che costituiscono i nomi dei *contributor* procedendo all'eliminazione di eventuali componenti nulli. Per fare ciò si utilizza un metodo statico definito all'interno di una classe java costruita *ad hoc*. La definizione di nuovo codice java o perl può essere facilmente effettuata all'interno della sezione codice nella sezione *Repository*. Sarebbe opportuno accompagnare la stesura del codice da commenti costruiti secondo la specifica sintassi TOS, in modo tale da poter visualizzare correttamente la sezione aiuto del tab "*Generatore di espressione*" e gli eventuali *tag* di esempio.

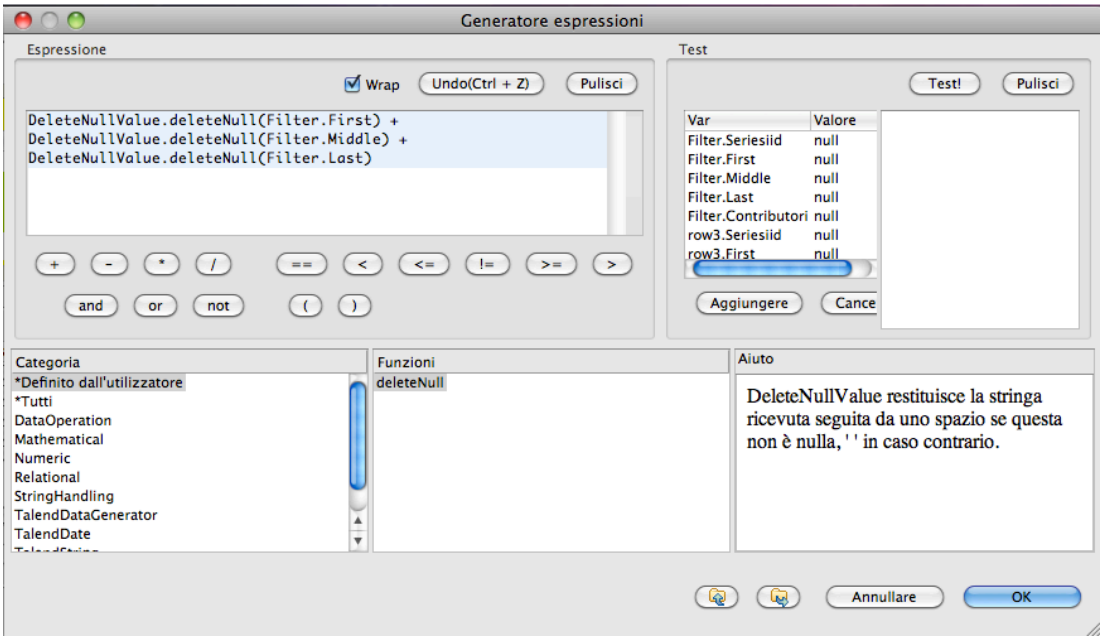

Figura 58: Trasformazione dei dati

L'operazione di trasformazione dello schema è stata eseguita per ogni relazione. Nella presente discussione sono stati riportati solo alcuni esempi, di facile lettura ma significativi.

Una volta definito lo schema di trasformazione, non ci resta che passare alla terza fase, cioè inserire i dati nel database MySQL strutturato nella seconda fase del progetto.

6.1.2.3 Terza fase

Per inserire i dati in una base di dati qualunque o in un file di qualsiasi estensione, basta selezionare dalla palette il tipo di sorgente di output o, se necessario, definire un nuovo tipo di sorgente di output nella sezione *Repository*. Nel caso in esame occorre anzi tutti creare la connessione con il database. Per farlo è sufficiente selezionare la voce crea

connessione nella sezione *Repository*. Una volta specificati i parametri di connessione (host, username, password, porta di ascolto della socket, ecc) e recuperato lo schema è possibile importare nel progetto, sempre tramite un semplice trascinamento, gli elementi di output e utilizzarli come contenitori per il risultato della manipolazione effettuata con gli strumenti *tMap*.

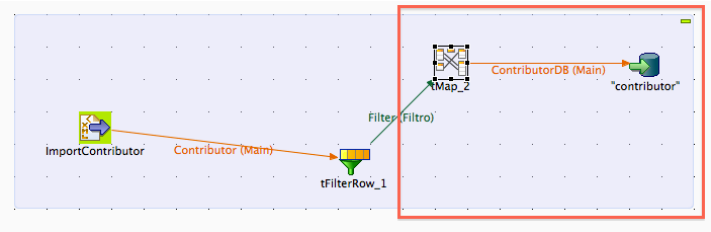

Figura 59: Connessione ad un database

Supponendo di volersi limitare all'import di *series* e *contributor* il progetto assumerebbe una fisionomia come quella del tipo riportato in figura. Come si può notare, l'esecuzione del progetto genera una serie di report che descrivono i passi di esecuzione. Se necessario è possibile eseguire il progetto in modalità *debug* impostando eventuali tracciamenti.

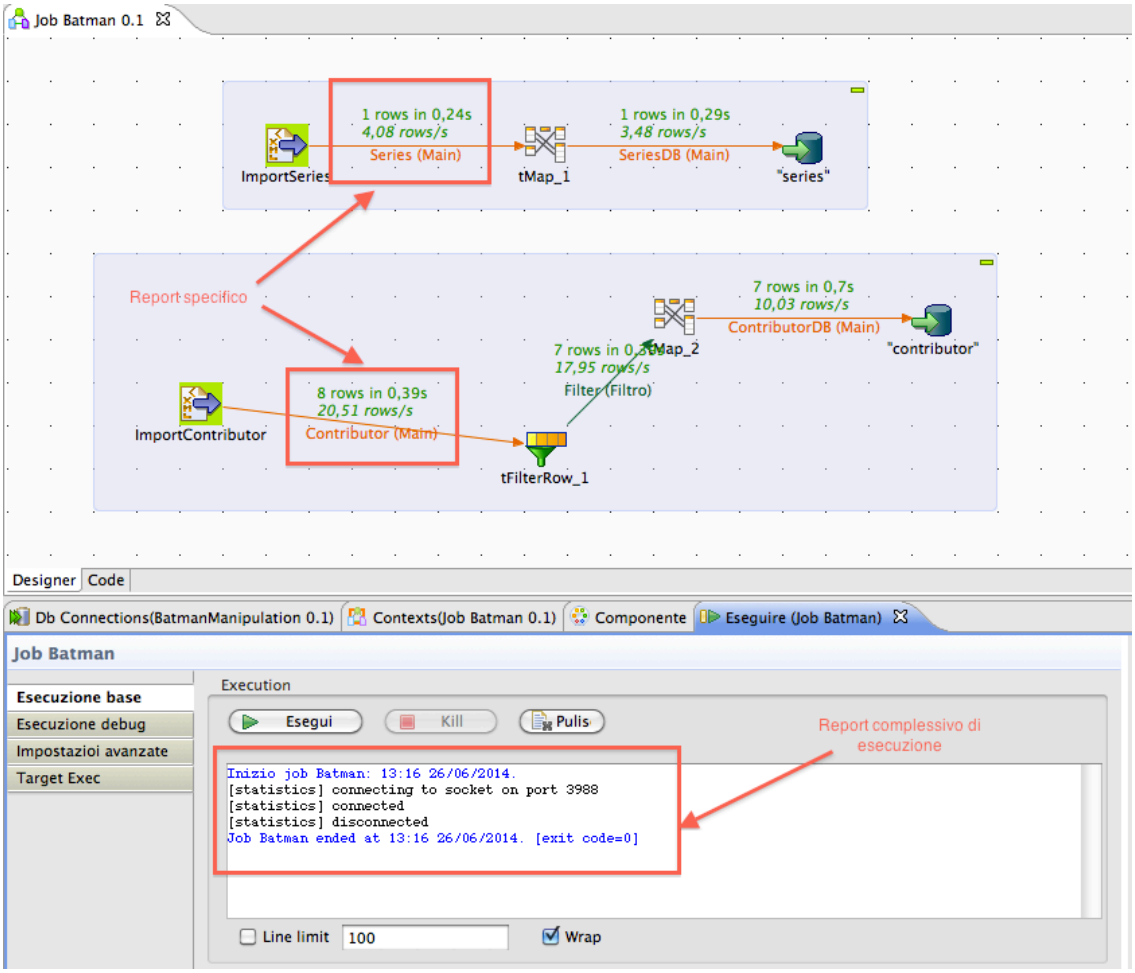

Figura 60: Report di esecuzione del Job

Il risultato dell'esecuzione del job produce un output lato DB come mostrato in Figura 61. Ovviamente quello riportato è il risultato relativo ad uno specifico file XML fornito in input

al sistema e limitato alle tabelle *contributor* e *series*.

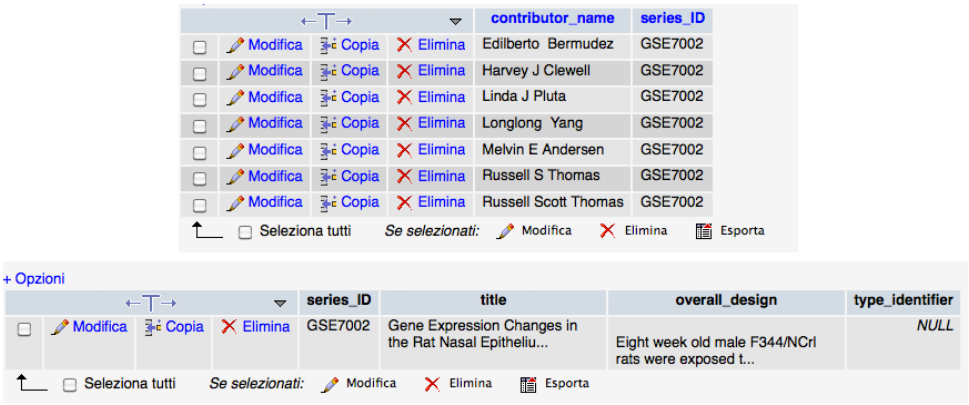

Figura 61: Risultato esecuzione del Job

6.2 Esportazione di un JOB di Talend

Quando si ultima un processo (JOB) di ETL, in *Talend*, si ha la possibilità di esportarlo in un JOB autonomo (v. Figura 62), cioè il processo viene trasformato in un eseguibile *jar* che ad ogni esecuzione, svolgerà la funzione per cui è stato creato.

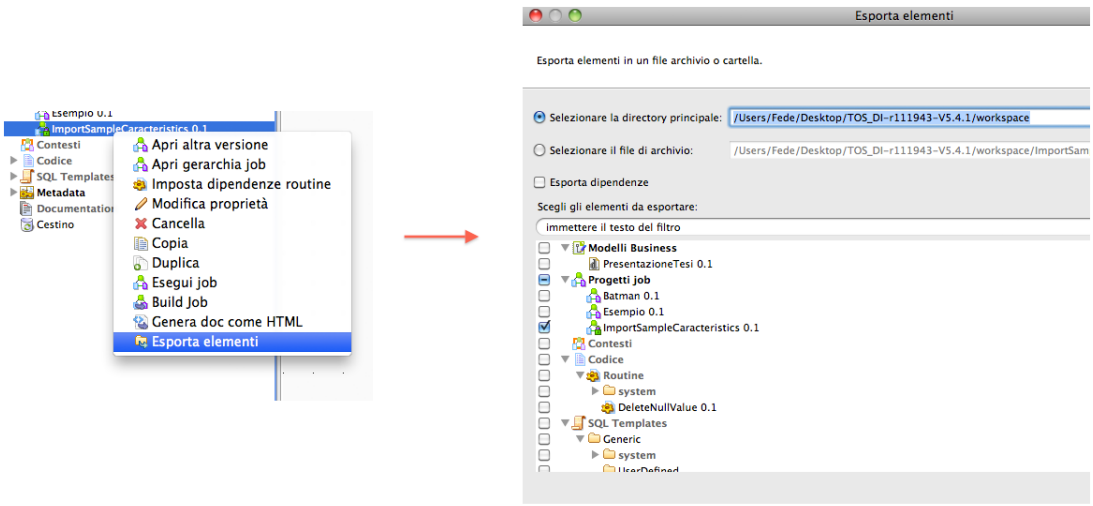

Figura 62: Esportazione di un JOB in Talend

Il risultato dell'esportazione è una cartella in cui sono presenti anche le librerie dei connettori utilizzati nel processo di estrazione. Se per esempio il nostro processo prevede l'estrazione di dati da una sorgente in formato Excel, nella cartella sono presenti le librerie che consentono la lettura del file Excel, oltre a quelle degli altri connettori utilizzati. Il JOB esportato contiene le seguenti cartelle (vengono analizzate solo quelle utili per l'integrazione: v. Figura 63):

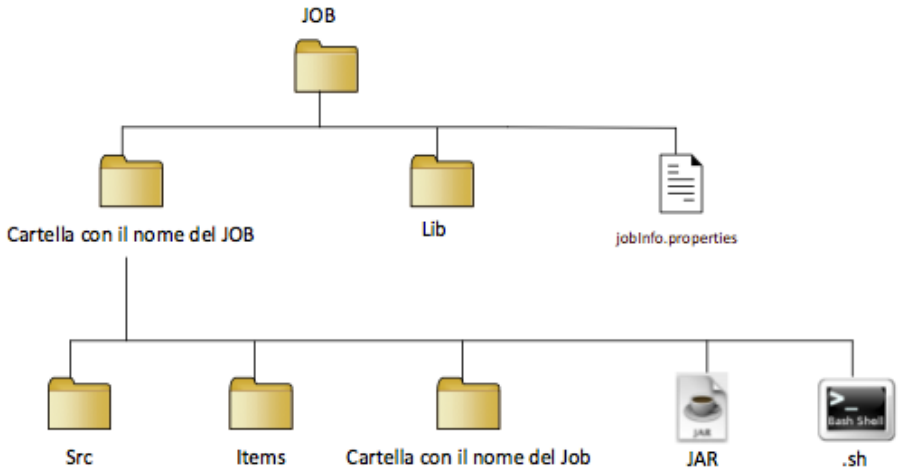

Figura 63: Contenuto di un JOB autonomo creato con Talend

- *Lib*: è una cartella che contiene le librerie usate nel processo di estrazione, come accennato prima, se abbiamo usato un componente per l'estrazione di dati da una sorgente Excel, la cartella contiene la libreria utile per leggere quella sorgente. Oltre alle librerie dei componenti, sono presenti anche le librerie personalizzate importate all'interno di *Talend*;
- *jobinfo.properties*: è un file che contiene informazioni utili ad identificare il JOB, es. quale versione di *Talend* è stata usata, la versione del job, ecc.;
- *Src*: contiene i sorgenti (classi .java) che sono stati creati da *Talend*. Ricordiamo che, anche se si lavora in maniera visuale, quello che viene prodotto è codice java. Contiene anche le routine/classi personalizzate che abbiamo usato all'interno del processo.
- *Items*: sono contenuti i file necessari a identificare i componenti usati all'interno del processo. Questo ci serve nel caso volessimo importare il JOB autonomo in un altro progetto, è possibile, insieme ai file "*src*", risalire a tutti i componenti che sono stati usati.
- *Cartella con il nome del job*: contiene un file "*Default.properties*", file di vitale importanza. Con questo file è possibile gestire le variabili globali (in Talend vengono chiamate di contesto) che usa il JOB. Nell'integrazione di un JOB in altre applicazioni java, l'unico mezzo di comunicazione tra i due sono le variabili di contesto.
- *JAR*: è l'eseguibile prodotto dall'esportazione, è quello che serve per eseguire il processo ETL creato. Però, di per se non vale nulla se non viene avviato con un opportuno comando, attraverso il quale oltre a lanciare il JAR, gli vengono passati anche le variabili di contesto. Le variabili di contesto possono essere utili se vogliamo parametrizzare una connessione a un database, locazione di un file, ecc.
- *.sh:* ha lo stesso nome del JOB, contiene il comando completo per eseguire il JAR insieme al contenuto delle variabili di contesto. Qui è stato analizzato il .sh (eseguibile per Unix) ma durante l'esportazione di un JOB è possibile scegliere anche l'eseguibile .bat, per Windows, o entrambi.

6.3 Integrazione di un JOB in un progetto java

Con il presente capitolo si vuole mostrare come sia possibile richiamare un progetto JOB creato in *Talend*, in un futuro progetto java. Un'operazione di questo tipo permetterebbe la creazione di un'interfaccia unifica per il download completamente automatizzato dei dati, il loro caricamento e le successive interrogazioni. Lo strumento in questione non sarà oggetto di questo elaborato. Come detto si vuole solo mostrate la possibilità di una futura implementazione.

Una volta prodotto il JOB autonomo, o meglio l'insieme di JOB autonomi necessari alla manipolazione dei dati per il successivo caricamento, non ci resta che includerlo in un progetto java. Per fare un esempio concreto, presupponiamo che il JOB autonomo consista nel:

1. Estrarre dati genici da un file XML proveniente da un server FTP (si veda Capitolo 5);
2. Manipolare/selezionare solo i dati di interesse;
3. Inserire i dati in un DBMS MySQL (si veda Capitolo 4);

Il processo ETL preso in considerazione riguarda manipolazione e import di dati inerenti ad un campione.

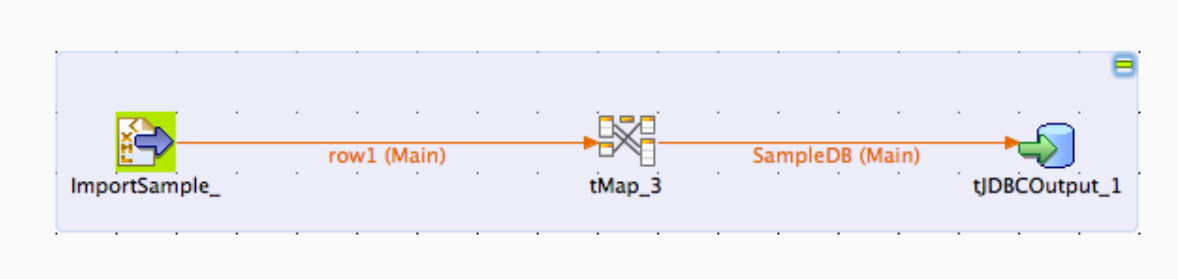

Figura 64: JOB di esempio

Nell'oggetto "*tJDBCOutput_1*" sono usate delle variabili di contesto, queste servono per comunicare con il JOB. In questo caso è stata parametrizzata la connessione jdbc del componente. In Figura 65 viene mostrato come richiamare le variabili di contesto definite nell'opportuna sezione di *Talend* (le variabili di contesto vengono definite della scheda *context*).

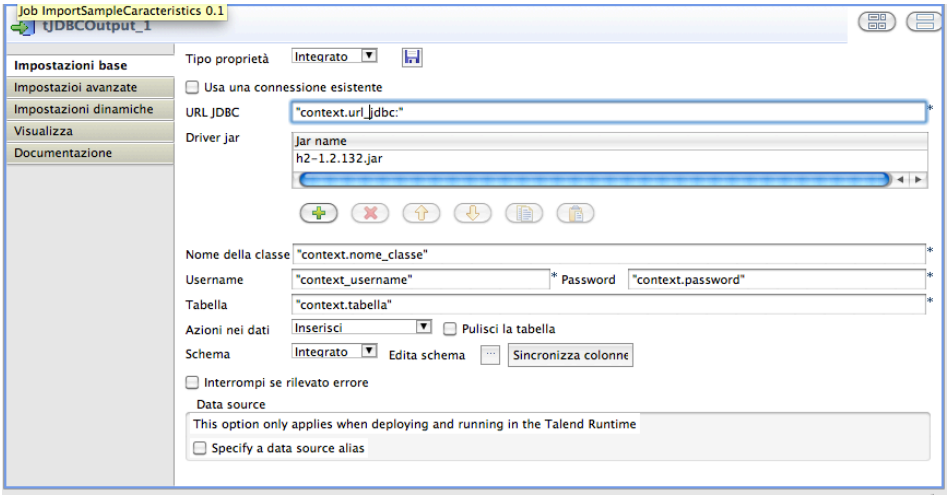

Figura 65: Variabili di contesto

Creiamo una classe all'interno della quale richiamiamo il JOB di *Talend* (v. Figura 66).

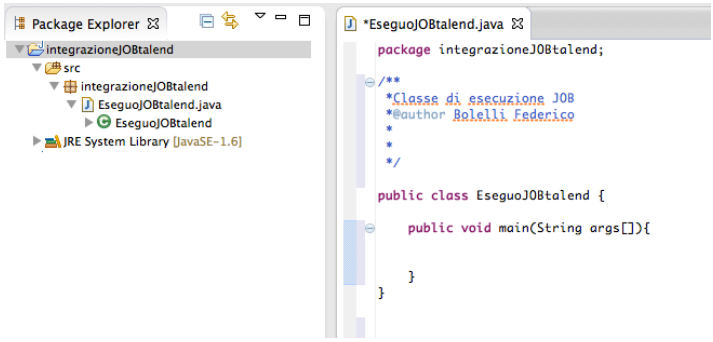

Figura 66: Classe per l'utilizzo di un JOB Talend

A questo punto occorre configurare il "*Build path*" della classe, per aggiungere le librerie dei componenti usati all'interno del JOB. Le librerie, in formato jar, da aggiungere (di solito tutte), sono presenti nella cartella *lib* del JOB esportato (fare riferimento allo schema in Figura 63), siccome abbiamo usato un connettore XML e un connettore jdbc, le librerie presenti in *lib* da importare sono:

- *H2-1.2.132.jar*: questa è stata importata all'interno del connettore JDBC; permette di avere i driver di connessione per le tabelle sql definite;
- *Dom4j-1.6.1.jar*: è una libreria open source che consente di lavorare con XML sulla piattaforma Java;
- *systemRoutines.jar*: sono le routine (classi) che usa *Talend* di default, es. le classi di conversione stringa o numerica (Math);
- *userRoutines.jar*: sono le routine (classi) personalizzate che definisce un utente;
- *importSample.jar*: è il JOB autonomo prodotto dalla procedura di esportazione da *Talend*; è la traduzione del codice java del processo visuale creato. All'interno del

nostro processo abbiamo utilizzato le variabili di contesto, quindi bisogna includere all'interno del progetto java la cartella che le contiene, semplicemente copiandola (per sapere dove si trovano fare riferimento sempre allo schema strutturale delle cartelle presente in Figura 63).

```
import test.ImportSample_0_1.ImportSample;
/*Importiamo la classe che gestisce il processo ETL
 *ImportSample_0_1 è il nome del jar prodotto dall'esportazione
 *ImportSample è il nome del progetto in Talend
 */

public class EseguoJOBtalend {

	public void main(String[] args){

		/*creo un istanza del JOB*/
		ImportSample talendJob = new ImportSample();
		/*lancio il JOB*/
		talendJob.runJob(new String[] {});
	}
}
```

Questo appena mostrato è il codice necessario a lanciare il processo ETL creato in *Talend*. È possibile lanciare il JOB con parametri di contesto personalizzati: al posto della stringa vuota, gli passiamo un array di String contenente i parametri (con gli stessi nomi definiti nel file "*Default.properties*"), es.

```
String [] context = {" --context_param host = " + host, --context_param host = " + username };
```

Il grado d'interazione con il JOB non è limitato al comando di lancio o di passaggio di parametri: è anche possibile recuperare i dati che processati dal JOB. In *Talend,* infatti, esiste un componente che si chiama *"tBufferOutput"*, che consente di rendere disponibili ad un altro JOB o web service i dati processati.

7 Conclusioni

Negli ultimi anni, le tecnologie *high-throughput* hanno rappresentato un grande passo avanti per l'analisi dell'espressione genica e centinaia di esperimenti basati sui microarray hanno generato una massa enorme di dati potenzialmente utili. La maggior parte di questi dati sono organizzati in banche dati pubbliche e la loro meta-analisi rappresenta un'enorme opportunità per studiare i meccanismi di regolazione dei processi biologici. Tuttavia, poiché i profili di espressione genica sono prodotti in laboratori diversi, utilizzando svariate tipologie di *microarray* e protocolli sperimentali, la loro meta-analisi rappresenta una sfida complessa per la bioinformatica. Infatti, l'analisi integrata di dati di *microarray* generati da gruppi di ricerca diversi su diverse piattaforme richiede metodi computazionali *ad hoc* per recuperare i dati grezzi e ogni relativa meta-informazione, per completare le annotazioni dei campioni se incomplete o inadeguate, per integrare più set di dati ottenuti utilizzando tecnologie diverse e per rimuovere, dal segnale finale integrato, le eventuali distorsioni indotte dallo specifico laboratorio o dalla particolare piattaforma. Gli applicativi presentati offrivano meccanismi all'avanguardia per risolvere problemi di annotazione. Nel tempo, con lo sviluppo degli algoritmi di analisi integrata dei dati, con l'aumentare del volume dei dati e con la nascita di nuove esigenze locali al Centro Interdipartimentale di Ricerche Genomiche – CeIRG, dell'Università degli Studi di Modena e Reggio Emilia si sono rivelati un fattore limitante.

Questo lavoro di tesi ha avuto come obiettivo primario lo studio di fattibilità, la progettazione e l'implementazione di un nuovo applicativo atto a migliorare, unificare e completare quelli preesistenti utilizzati dal centro CeIRG. Tutte le funzionalità di base di tale applicativo sono state implementate. I passi futuri per rendere operativo e completo l'applicativo consistono, sostanzialmente, nella creazione di un'interfaccia unificata per l'interazione semplificata con lo strumento e nell'implementazione dello stesso sulle macchine (terminali e server) del centro di ricerca. Una caratteristica fondamentale del sistema sviluppato è l'estrazione ed il caricamento automatico di dati proveniente dal *microarray* database. D'altra parte, in questo lavoro di tesi non ci siamo occupati dell'integrazione dati, in quanto è stata considerata la sola sorgente GEO; l'utilizzo di tecniche di Integrazione Dati saranno importanti quando il sistema dovrà gestire dati provenienti da più *microarray* database (es. Array Express oltre che GEO). Il primo passo in tale direzione sarà quello di estendere il processo ETL e di sviluppare features aggiuntive per le nuove sorgenti. Il passo successivo sarà quello di utilizzare *Sistemi di Data Integration*, come ad esempio il sistema MOMIS sviluppato dal DBGroup (www.dbgroup.unimore.it).

8 Bibliografia

Immagini della biologia (seconda edizione) Zanichelli editore – Neil A.Campbell, Jane B.Reece, Marthe R.Taylor, Eric J.simon.

Biologia – McGraw, Hill – Brooker, Widmaier, Graham, Stiling.

Progetto di Basi di Dati Relazionali – Pitagora Editrice Bologna – Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, Maurizio Vincini.

Programmazione a ogetti in Java: dai fondamenti a internet - Pitagora Editrice Bologna – Giacomo Cabri, Franco Zambonelli.

Lucidi delle lezioni di "Basi di Dati e Lab", corso tenuto dalla Professoressa Sonia Bergamaschi e dal Dott. Ing. Laura Po nell'anno 2013 presso l'Università degli studi di Modena e Reggio Emilia: laurea Triennale in Ingegneria Informatica.

Lucidi delle lezioni di "Reti di Calcolatori e Lab. ", corso tenuto dall'Ing. Riccardo Lancellotti nell'anno 2014 presso l'Università degli Studi di Modena e Reggio Emilia: laurea Triennale in Ingegneria Informatica.

Lucidi delle lezioni di "Laboratory of Bioinformatics" , corso tenuto dal Professor Bicciato nell'anno 2014 presso l'Università degli Studi di Modena e ReggioEmilia: laurea Magistrale in Biotecnologie Industriali.

9 Sitografia

http://www.ncbi.nlm.nih.gov/geo/

https://www.ebi.ac.uk/arrayexpress/

http://www.fged.org/projects/minseqe/

http://www.mged.org/Workgroups/MIAME/miame_1.1.html

http://compgen.bio.unipd.it/bioinfo/amadman/

http://www.talend.com/

http://www.xml.com/

http://it.wikipedia.org/wiki/XML

http://it.kioskea.net/contents/25-il-protocollo-ftp-file-transfer-protocol

http://it.wikipedia.org/wiki/File_Transfer_Protocol