Relazione al Progetto di Basi di Dati

Federico Amici

3 gennaio 2017

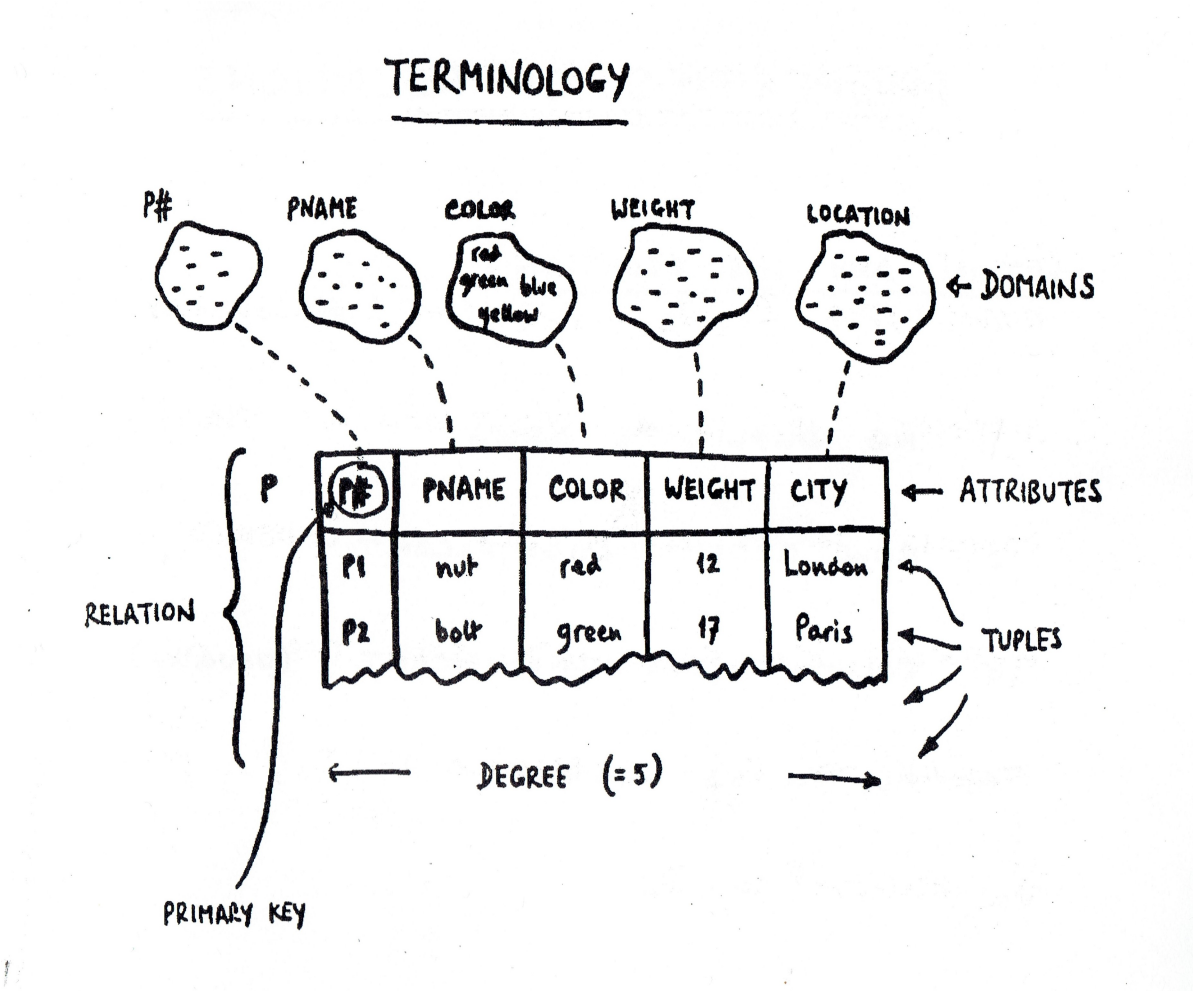


Figura 1: Lucido del seminario "An Introduction to Relational Data Base" tenuto nel 1976 da C. J. Date

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

La relazione qui presentata fornisce la documentazione del progetto relativo al corso di Basi di Dati nonchè delle digressioni sulle soluzioni adottate nella stesura della soluzione proposta. È stata aggiunta inoltre una presentazione utile per una eventuale esposizione orale del progetto stesso.
Una *base di dati* è una collezione di dati, tipicamente di dimensioni elevate, che descrive l'attivitá di un'organizzazione. Le dimensioni di questa collezione di dati sono di gran lunga maggiori di quelle relative alla memoria primaria, fatto che porta ad una gestione piú attenta dei meccanismi di accesso alle informazioni ivi contenute.
Come spesso accade in ambito ingegneristico, anche la progettazione di una base di dati viene realizzata mediante l'adozione del modello *"a strati"*.

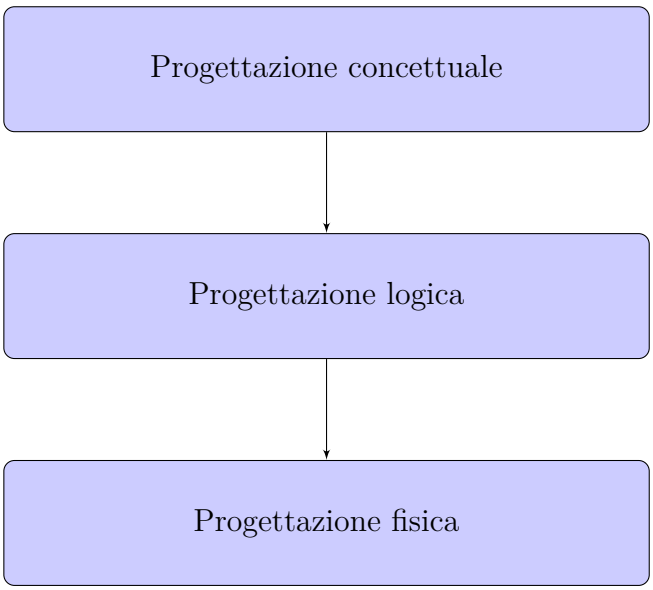
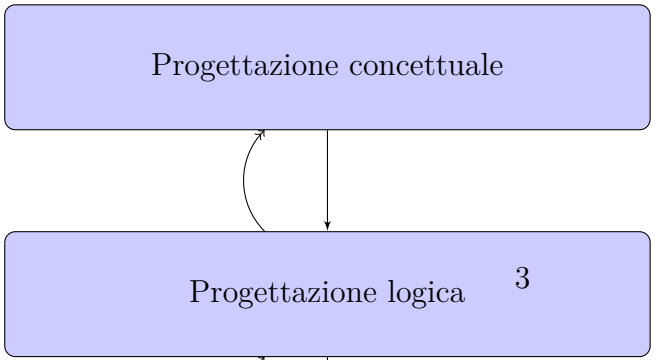


Figura 2: Primo approccio di sviluppo

La progettazione di una base di dati fa molto spesso parte dello sviluppo di un sistema software più complesso. Essendo tuttavia una parte importante nella realizzazione del progetto, nonchè dotata anch'essa di un certo grado di complessità viene spesso trattata come una componente disgiunta.
Come si evince dalla figura, cominceremo a progettare la base di dati ad alto livello mediante la fase di progettazione concettuale, durante la quale verrà redatto lo schema ER, un modello che ci permette di descrivere il sistema in termini di oggetti e delle relazioni che intercorrono fra gli stessi. Questa rappresentazione della base di dati é utile in quanto si possono esprimere le relazioni fra gli oggetti nella base di dati senza dover badare al modello logico adottato, né tanto meno al modello fisico.
La possibiliá di progettare la base di dati a strati viene offerta da una delle caratteristiche fondamentali di un DBMS, ovvero il fatto che realizza l'*indipendenza dei dati*. Infatti, a meno di correzioni di piccola entità, é possibile scegliere il modello dei dati che piú ci risulta consono ad ogni livello di progettazione, fatto che sulla carta rende questo processo molto flessibile. Tuttavia, con il maturare dell'esperienza in questo ambito, si sono consolidate alcune pratiche che sono risultate vincenti nel corso degli anni. Un esempio fra tutti é l'adozione in fase di progettazione concettuale del modello *Entity-Relationship*.
Altro approccio che è risultato vincente nel corso del tempo è quello dei processi di sviluppo di tipo iterativo che, a differenza dei genitori "a cascata", permettono di tornare ciclicamente all'inizio del processo di sviluppo per correggere il tiro a partire dalla fase di analisi dei requisiti in poi. Nel caso dello sviluppo di questo progetto non ci si avvale completamente delle tecniche dell'ingegneria del software, tuttavia per quanto possibile si adotterà il modello di sviluppo iterativo.

Lo schema di sviluppo deve dunque essere aggiornato a quello riportato a lato.



1.1 Il Modello Entitá-Relazione

Anno modello relazionale, anno modello er, perchè er?

1.2 Manuale d'uso

2 Progettazione concettuale

Una delle fasi piú importanti nella realizzazione di un progetto, che sia prettamente software o riguardante una base di dati, é costituita dalla *raccolta e analisi dei requisiti*. Ricordiamo che un *requisito* rappresenta un vincolo da rispettare sia nella fase del suo sviluppo sia nella fase di funzionamento del software. Come suggerisce [8], la prima causa del fallimento del progetto di un sistema software é l'insuccesso dell'ingegneria dei requisiti. In questo caso non ci si soffermerà con lo stesso dettaglio del processo UP su questa fase dello sviluppo software ma, essendo indubbiamente fondamentale, le verrà dedicato lo spazio che merita.

La *raccolta dei requisiti* nel caso di questo progetto è stata simulata dal documento di presentazione inviato dal Professor Galli in qualità di committente del lavoro. Il documento è scritto in linguaggio naturale, come accadrebbe nel caso di una stesura delle specifiche del sistema da realizzare a seguito di un incontro con il committente o con i futuri utenti dell'applicazione. Un aspetto importante della progettazione concettuale è l'elaborazione preliminare di tutti quei documenti utili sia agli sviluppatori del progetto che entrano in contatto con il mondo che vanno a modellare sia al committente per essere sicuro che la realizzazione del progetto sia in linea con le sue aspettative.

Generalmente quando si va a sviluppare un sistema software si entra a far parte di un mondo che non si conosce a fondo in quanto estraneo al proprio. Anche in questa occasione si è acquisita maggiore dimestichezza con gli argomenti relativi al mondo che si è modellato, attingendo a fonti esterne come astronomia.com. Il *glossario dei termini* raccoglie tutta quella terminologia, presente nella documentazione di presentazione del progetto che simula la *raccolta dei requisiti*, di cui non si era a conoscenza al momento della sua stesura.

Il documento è stato ristrutturato suddividendo le frasi presenti al suo interno secondo il concetto cui si legano, rimodellando talvolta la frase laddove non fosse più necessario collegarla al resto del documento.

2.1 Specifica fornita

Introduzione al progetto

L'Istituto Nazionale di Astrofisica insieme all'Università di Tor Vergata vogliono costruire una applicazione che permetta di importare in un database i dati raccolti sulle galassie da diversi satelliti (tra cui Herschel/PACS e Spitzer) al fine di poterli interrogare e gestire in modo più efficiente.

In particolare si sono voluti mettere assieme gli ultimi dati che sono stati catturati dal satellite Herschel/PACS che ha misurato il flusso degli spettri del lontano infrarosso, con quelli provenienti da altri satelliti precedenti come ad esempio il satellite Spitzer che venne utilizzato per misurare i flussi del vicino infrarosso.

I dati raccolti, una volta elaborati, sono utilizzati dagli scienzati per misurare le condizioni fisiche del gas all'interno delle galassie (es. temperatura, densità, metallicità, etc..).

In generale ogni galassia ha un proprio nome, una determinata posizione geografica spaziale e una distanza e/o redshift. Le galassie possono essere suddivise in sottogruppi a seconda della loro classificazione spettrale. Ogni galassia può avere un valore di luminosità espressa in

Watt, misurata relativamente ad uno specifico atomo ionizzato e di metallicità relativa a quella solare. Ad ogni galassia può essere associato il valore del flusso di una riga/linea spettrale. Ogni flusso è associato ad una riga della linea spettrale per un determinato atomo ionizzato ed è composto da una terna contenente il valore del flusso, il suo errore ed una flag che indica se è un upper-limit (ovvero se il valore misurato era minore della sensibilità dello strumento quindi al massimo è pari al minimo della sensibilità dello strumento).

Al momento le informazioni elaborate sono state raccolte in
5 file contenenti i dati in formato CSV. Ogni file è descritto in dettaglio nella sezione successiva.

File dei dati

File Header

Ogni file contiene un header testuale che descrive le colonne della tabella in formato CSV contenuta nel file. Inoltre può contenere i riferimenti bibliografici da cui sono stati estratti i dati. I riferimenti bibliografici possono non essere inclusi nel database dell'applicazione.

File 1) Catalogo delle galassie
Il file contiene alcune informazioni per individuare una specifica galassia. Ogni galassia è solitamente definita dalle sue coordinate angolari (ascensione retta e declinazione), il suo redshift e in ultimo la sua distanza. Per la distanza è definito anche il riferimento bibliografico da cui è stata tratta.
Ogni galassia è classificata rispetto alle sue proprietà ottiche (emissione spettrale). La classificazione spettrale prevede 6 macro-gruppi (S1, S1h, S2, LIN, Dwarf e H2). Inoltre nel file sono riportati anche i valori di luminosità proveniente dalle misure in X-Ray con il relativo riferimento bibliografico, i valori di metallicità (valore di 1 corrisponde alla metallicità del Sole) ed il relativo riferimento.

File 2) File dei flussi delle righe di Herschel/PACS
Il file contiene le informazioni provenienti dallo studio degli scienziati dell'INAF riguardo alla misura del flusso delle righe spettrali nel satellite Herschel/PACS. Per ogni galassia sono stati calcolati i flussi degli atomi ionizzati ossigeno (OIII, OI), azoto (NIII, NII) e carbonio (CII). Per ogni flusso, se è stato possibile calcolarlo, è riportato il suo valore ed il relativo valore di errore.
Nel caso il valore calcolato è inferiore alla sensibilità dello strumento, una flag è inserita e il relativo upper-limit è riportato.

File 3) File del flusso continuo di Herschel/PACS
Il file contiene i valori del flusso continuo, ovvero il flusso prodotto in generale dalle polveri interne a una galassia. Le polveri producono un rumore di fondo che nel diagramma flusso/frequenze generano una linea continua piatta. È solitamente utilizzato per misurare la temperatura e la massa della polvere. La struttura è simile alla tabella precedente, per tutte le 7 righe. L'apertura spaziale (numero di pixel utilizzati) è equivalente a quelli della tabella precedente.

File 4) File dei flussi delle righe di Spitzer
Il file è equivalente a quello relativo al flusso delle righe di Herschel/PACS, ma contiene i valori relativi alle righe misurate dal satellite Spitzer, ovvero per gli atomi ionizzati zolfo

(SIV, SIII), Neon (NeII, NeV, NeIII) e silicio (SiII).

File 5) File dei flussi delle righe di Herschel/PACS per tutti i valori di aperture size L'ultimo file contiene gli stessi campi del file 3, ma riporta i valori dei flussi per tutte e tre le differenti aperture size.

Tabella 1: Glossario dei termini

Termine	Descrizione	Sinonimi	Collegamenti
Galassia	Un grande insieme di stelle	-	
Linea spettrale	Effetto dell'interazione tra un sistema quantistico e singoli fotoni	-	
Satellite	Oggetto orbitante intorno ad un corpo celeste che ha dimensioni molto maggiori		
Redshift	Fenomeno secondo il quale la frequenza di luce osservata differisce da quella emessa		
Flusso spettrale			

Una volta esaminato a fondo il documento di presentazione, viene effettuato un lavoro di rimodellazione che porta ad uniformare il testo dei requisiti, rendendoli non ambigui dal punto di vista degli sviluppatori ed ancora corrispondenti alle esigenze del committente. Il documento di presentazione viene dunque ristrutturato, suddividendo il testo in gruppi di frasi che trattano lo stesso concetto e riscrivendo secondo il pattern *Per <dato> rappresentiamo <insieme di proprietà>*.
Il risultato della rimodellazione del documento é presentato di seguito:

Frasi di carattere generale
L'Istituto Nazionale di Astrofisica insieme all'Università di Tor Vergata vogliono costruire una applicazione che permetta di importare in un database i dati raccolti sulle galassie da diversi satelliti (tra cui Herschel/PACS e Spitzer) al fine di poterli interrogare e gestire in modo più efficiente.

In particolare si sono voluti mettere assieme gli ultimi dati che sono stati catturati dal satellite Herschel/PACS che ha misurato il flusso degli spettri del lontano infrarosso, con quelli provenienti da altri satelliti precedenti come ad esempio il satellite Spitzer che venne utilizzato per misurare i flussi del vicino infrarosso.

I dati raccolti, una volta elaborati, sono utilizzati dagli scienzati per misurare le condizioni fisiche del gas all'interno delle galassie (es. temperatura, densità, metallicità, etc..).

Ogni file contiene un header testuale che descrive le colonne della tabella in formato CSV contenuta nel file. Inoltre può contenere i riferimenti bibliografici da cui sono stati estratti

i dati. I riferimenti bibliografici possono non essere inclusi nel database dell'applicazione.

Ogni flusso è associato ad una riga della linea spettrale per un determinato atomo ionizzato

Ogni galassia è classificata rispetto alle sue proprietà ottiche (emissione spettrale).

Per ogni galassia sono stati calcolati i flussi degli atomi ionizzati ossigeno (OIII, OI), azoto (NIII, NII) e carbonio (CII).

Il flusso continuo è il flusso prodotto delle polveri interne a una galassia, le quali producono un rumore di fondo che nel diagramma flusso/frequenze generano una linea continua piatta.

Nel caso delle galassie, si è riscontrata una ambiguità nel documento, relativamente al fatto che il termine *distanza* non fosse ulteriormente approfondito, ad esempio specificando rispetto quale punto venga calcolata. Per questo si sono cercate ulteriori informazioni lungo il documento, avendone effettivamente riscontrate nel seguito.
Altra fonte di ambiguità è stata riscontrata nell'attributo luminosità: infatti nel documento di presentazione viene presentato come valore calcolato relativamente ad un atomo ionizzato, ma non c'è traccia nel seguito del testo di tale informazione. È da notare il fatto che nel riportare le frasi specifiche per ogni concetto non vengono evidenziati soltanto gli attributi relativi ad una entità ma anche eventuali relazioni.

Frasi relative alle galassie
Per ogni galassia rappresentiamo il nome, il nome alternativo, le coordinate angolari (ascensione retta e declinazione), il suo redshift, la classificazione spettrale (S1, S1h, S2, LIN, Dwarf e H2), la luminosità *rispetto a un atomo ionizzato*, la metallicità relativa al Sole e la distanza.

Il flusso è uno di quei concetti che più di altri esemplificano lo sforzo che deve essere messo in atto dagli sviluppatori per entrare in contatto con il mondo che stanno andando a modellare. Infatti lungo il documento troviamo una quantità di informazioni che, seppur ampia, risulta insufficiente per avere una chiara visione di come i concetti che sono spesso ignoti allo sviluppatore si leghino al resto del sistema. Il primo passo da fare è dunque raccogliere le informazioni presenti nel documento e trovare una linea comune e, se questo non fosse possibile, sorge l'esigenza di intervistare nuovamente il committente.

Frasi relative ai flussi
Per ogni flusso rappresentiamo il valore del flusso, il suo errore ed un upper-limit. Il flusso continuo è rappresentato analogamente a quello delle righe (precedente) I flussi delle righe Herschel/PACS e Spitzer differiscono per gli atomi rispetto ai quali sono misurati.

Frasi relative ai riferimenti bibliografici
Per i riferimenti bibliografici rappresentiamo i riferimenti bibliografici per la distanza, metallicità e luminosità di una galassia.

2.2 Operazioni sui dati

Operazione 1: Trova un utente registrato e controlla che la password inserita corrisponda a quella salvata nel database

Operazione 2: Inserisci di un utente all'interno del sistema

Operazione 3: Importa un nuovo file dei dati scientifici, aggiornando i valori precedenti con quelli nuovi

Operazione 4: Ricerca di una galassia per nome

Operazione 5: Ricerca di un insieme di galassie all'interno di un raggio data la posizione spaziale

Operazione 6: Ricerca di un insieme di galassie in base al parametro di redshift

Operazione 7: Ricerca di valori dei flussi ed il relativo errore di una o più righe spettrali di una specifica galassia

2.3 Regole di vincolo

Di seguito vengono presentate le regole di vincolo che sono state desunte dal modello ER, alcune frutto di fantasia.

RV1 Lo user-id *deve* avere un numero minimo di caratteri pari a 6
RV2 La password di un utente amministratore *deve* avere un numero minimo di caratteri pari a 9 e *deve* contenere almeno un numero
RV3 L'upper-limit viene settato se il valore misurato era minore della sensibilità dello strumento, quindi al massimo è pari al minimo della sensibilità dello strumento
RV4 Il file contenente dati sulle galassie dovrà essere salvato con il nome *MRTable3_Sample.csv*
RV5 Il file contenente dati sui flussi delle righe dovrà essere salvato con il nome *MRTable4_flux.csv*
RV6 Il file contenente dati sui flussi continui dovrà essere salvato con il nome *MRTable6_cont.csv*
RV7 Il file contenente dati sui flussi delle righe calcolati dal satellite Spitzer dovrà essere salvato con il nome *MRTable8_irs.csv*
RV8 Il file contenente dati sui flussi delle righe per tutti i valori di aperture dovrà essere salvato con il nome *MRTable11_C_3x3_5x5_flux.csv*
RV9 Lo *userid* di un utente del sistema non può coincidere con la *password*

2.4 Schemi ER

La stesura degli schemi ER può essere sia realizzata su carta, sia servendoci di strumenti software dedicati. I programmi appartenti a questa categoria vengono definiti *CASE software*, dove CASE sta per Computer-Aided Software Engineering. Come da aspettative, la scelta é molto ampia, sia tra i software commerciali sia tra quelli free. Dal momento che la complessità del progetto è di gran lunga inferiore a quella di una commissione reale, è parsa la scelta migliore quella di scegliere un software commerciale, ma che allo stesso tempo si comportasse come un freeware a patto di utilizzare un sottoinsieme delle funzionalità offerte.

Il software in questione è *ERWin* Community Edition, reperibile presso l'indirizzo `http://erwin.com/products/data-modeler/community-edition`.

Nonostante la velocità di stesura offerta dal programma CASE, si è preferito stilare lo schema di progettazione concettuale su carta in prima battuta e, solo in secondo tempo, riportato in un file, in modo da avere anche una copia di backup disponibile.

Tabella 2: Dizionario dei dati

Termine	Descrizione	Attributi	Identificatore
Galassia	Un grande insieme di stelle	Classificazione spettrale, Redshift, NomeAlternativo NomeGalassia	Nome Galassia
Coordinate angolari	Dati utili alla localizzazione della galassia	AscensioneRetta_ore, ascensioneretta_min, ascensioneretta_sec, declinazione_sign, declinazione_ore, declinazione_min, declinazione_sec	NomeGalassia *(esterno)*
Caratteristiche Galassia	Caratteristiche proprie della galassia	Distanza, luminosità, metallicità	NomeGalassia *(esterno)*
Errore	Insieme degli errori commessi sulle misure relative alla galassia	ErroreDistanza, ErroreMetallicità, FlagLuminosità	NomeGalassia *(esterno)*
Riferimento bibliografico	Insieme di riferimenti bibliografici delle misure effettuate su una galassia	Autore, titolo, anno	Codice
Flusso	Informazione sulle sostanze presenti nella galassia	Errore, valore, atomo, satellite	NomeGalassia *(esterno)*

Riga Flusso	Insieme di dati relativi alla misura di una riga spettrale	Errore, Valore, Riferimento	NomeGalassia (esterno), Atomo (esterno)

Di seguito viene riportato la prima stesura dello schema ER relativo al progetto. Come sottolineato in precedenza, il processo di sviluppo adottato è iterativo e quindi, ogni volta che si presenterà una miglioria possibile o ci si accorgerà di un errore si potrà tornare indietro e correggere.
Un esempio di correzione che è stata apportata allo schema ER è l'elliminazione delle generalizzazioni relative all'entità *Flusso* poichè nelle operazioni richieste dai requisiti non figura una distinzione tra le tipologie di flusso.

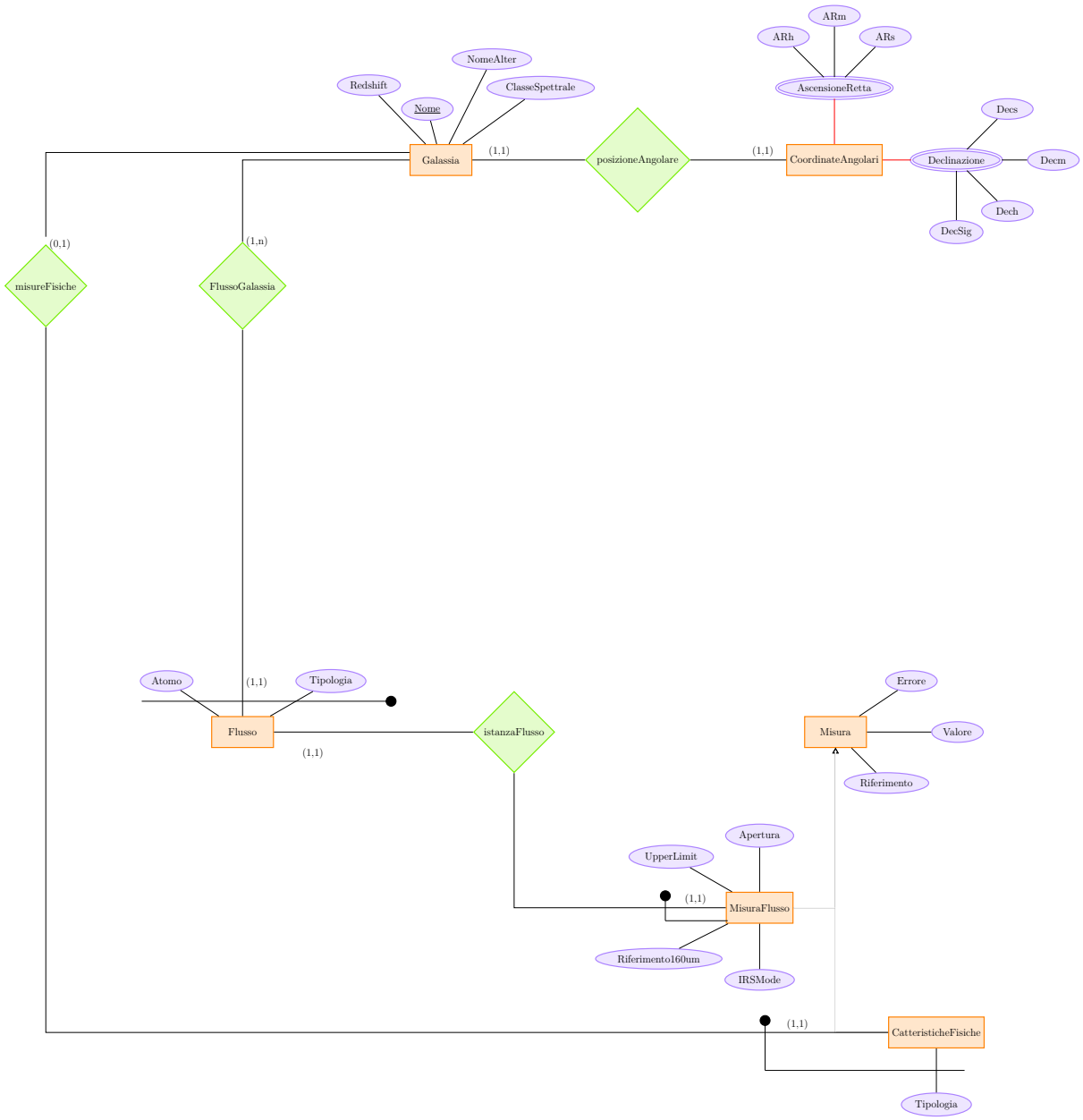


Figura 4: Prima stesura dello schema E-R

Nello schema E-R non figurano le entità *Utente Registrato* nè *Amministratore*. Questo deriva dal fatto che anche gli utenti dell'applicazione vengono gestiti tramite il sistema ma,

ciononostante, non hanno relazioni con le altre entità. Saranno certamente collegate in qualità di classi al resto del sistema tramite opportuni metodi, ma come entità rimangono distaccate dal resto dello schema.

3 Progettazione logica

3.1 Il modello relazionale

3.2 Definizione del carico di lavoro

3.2.1 Tavola dei volumi

La *tavola dei volumi* riporta il carico previsto a regime per ogni entità e per ogni relazione presenti nello schema ER. Nella *tavola delle operazioni* invece viene riportata per ogni operazione la frequenza di invocazione a regime, unitamente ad un simbolo che la caratterizza come interattiva o come batch, nel seguito indicate come di consueto con I e B. In questo caso tuttavia non vi sono operazioni che il sistema possa compiere in modalità *batch* poichè ognuna ha bisogno di essere portata a termine nel minor tempo possibile a partire dalla sua sottoposizione al sistema.
I volumi relativi alle entità e alle relazioni sono stati ricavati dall'esame dei file allegati alla traccia del progetto e, dal momento che tra i requisiti funzionali non vi è alcuna operazione di inserimento, i volumi a regime delle entità in relazione con Galassia sono tutti relativi al numero di galassie presenti nel relativo file (241, come riportato di seguito).
mentre la tavola delle operazioni è stata stesa attingendo alla personale fantasia.

Concetto	Tipo	Volume
Galassia	E	250
CoordinateAngolari	E	250
Flusso	E	$\sum flussi$
MisuraFlusso	E	$\sum flussi$
CaratteristicheFisiche	E	250
PosizioneAngolare	R	250
misureFisiche	R	250
FlussoGalassia	R	$\sum flussi$
istanzaFlusso	R	$\sum flussi$

Tabella 3: Tavola dei volumi

Concetto	Tipo	Volume
Operazione 1	I	1000 al giorno
Operazione 2	I	10 al giorno
Operazione 3	I	5 al giorno
Operazione 4	I	2000 al giorno
Operazione 5	I	500 al giorno
Operazione 6	I	500 al giorno
Operazione 7	I	2000 al giorno

Tabella 4: Tavola delle operazioni

3.3 Ristrutturazione dello schema ER

3.3.1 Analisi delle ridondanze

In questa fase bisogna trovare tutte le relazioni e gli attributi che possono essere ricavati a partire da altre relazioni o entità. Le ridondanze vanno mantenute solo nel caso in cui effettivamente apportano un vantaggio prestazionale al sistema. Nel caso dello schema E-R del sistema non vi sono ridondanze, il che ci permette di passare alla fase successiva.

3.3.2 Eliminazione delle generalizzazioni

Questa fase è necessaria in quanto i DBMS tradizionali non offrono la possibilità di rappresentare direttamente una generalizzazione. Bisogna dunque sostituire la gerarchia con uno schema che dia le stesse informazioni del precedente. Le alternative che si sono presentate sono due:

- Incorporare gli attributi di *Misura* nelle entità figlie

- Sostituire la specializzazione di *Misura* con due relazioni

Dal momento che la prima alternativa presenta un minor numero di accessi nelle operazioni relative ai flussi, è stata preferita quest'ultima. In seconda battuta dobbiamo tradurre i multiattributi poichè nemmeno questi vengono trattati dai DBMS tradizionali. Nel caso in esame sono state create due nuove entità, *Ascensione Retta* e *Declinazione*, identificate esternamente dall'entitità *Galassia*.

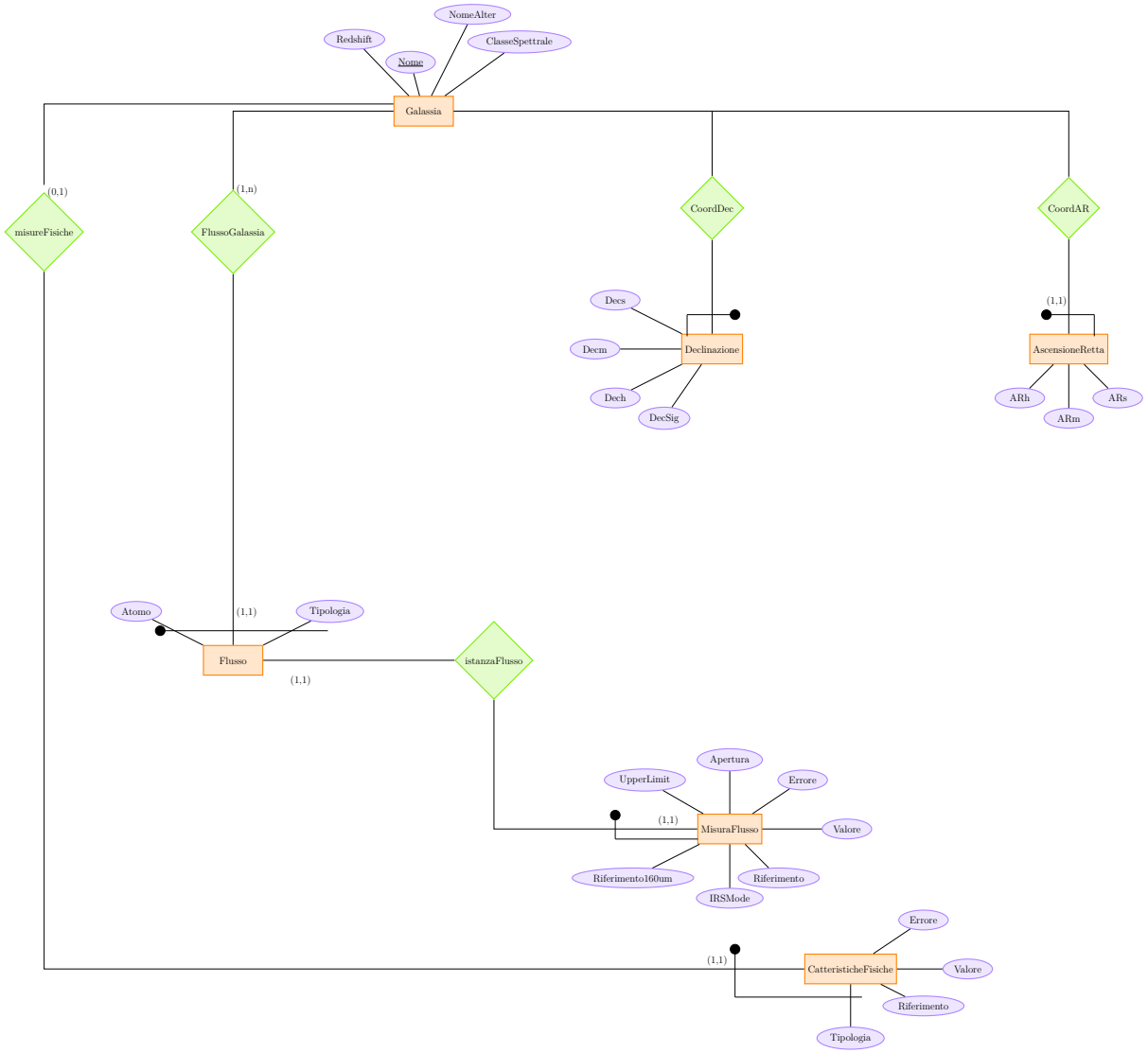


Figura 5: Schema E-R ristrutturato, eliminati i multiattributi e generalizzazioni

Una volta eliminati i multiattributi e le generalizzazioni, possiamo passare alla risoluzione degli identificatori esterni:

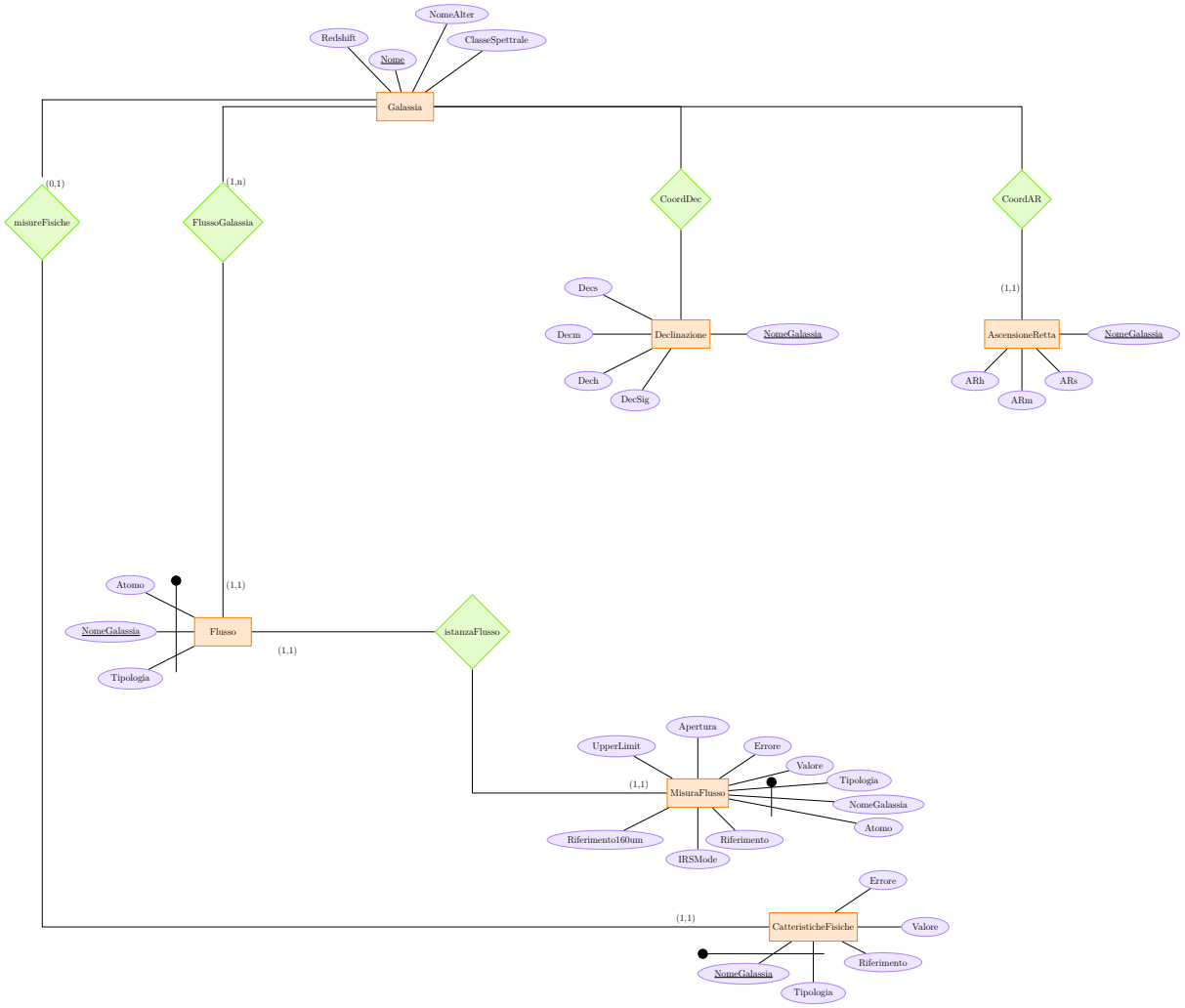


Figura 6: Schema E-R ristrutturato, eliminati gli identificatori esterni

Come previsto in un processo di sviluppo iterativo, in seguito alla fase di implementazione si è resa necessaria una modifica nello schema ER.
Infatti, implementando i requisiti, la modifica degli schemi ER iniziali era obbligatoria per continuare a rispettare le specifiche ed eliminare tutte quelle informazioni mai accedute nel database. Viene sintetizzato di seguito quanto è emerso:

- La distanza non è una caratteristica fisica delle galassie utilizzata da alcun requisito. Nel numero 6 la distanza considerata è calcolata fra due punti arbitrari. Di conseguenza può essere omessa dall'ER.
- Le caratteristiche fisiche metallicità e luminosità vengono sempre accedute insieme, di conseguenza risulta utile porle nella stessa entità in modo da migliorare le prestazioni di accesso al dato.
- Discorso analogo vale per le entità *Declinazione* ed *Ascensione Retta* che, essendo accedute sempre in coppia, tornano a formare un'entità unica.
- Non tutte le galassie hanno dati sulle caratteristiche fisiche, quindi la cardinalità Galassia-misureFisiche è (0,1).
- Oltre ad aver accorpato l'entità *Flusso* con le misure associate, è stato anche suddiviso in tante entità quante sono le tipologie. Questa scomposizione, oltre ad evitare la

presenza di valori nulli sugli attributi non presenti in tutte le tipologie di flusso, permette un accesso più rapido alle informazioni correlate al tipo di flusso ricercato.

- L'attributo di *tipologia* di *Flusso* viene sostituito dalle tre entità *FlussoContinuoHP*, *FlussoRigheHP* e *FlussoRigheSp* (i dati sul flusso continuo sono tutti provenienti dal satellite Herschel/PACS). Questa suddivisione aiuta nell'importazione dei file, evitando che vengano cancellati dati relativi ad altre tabelle.

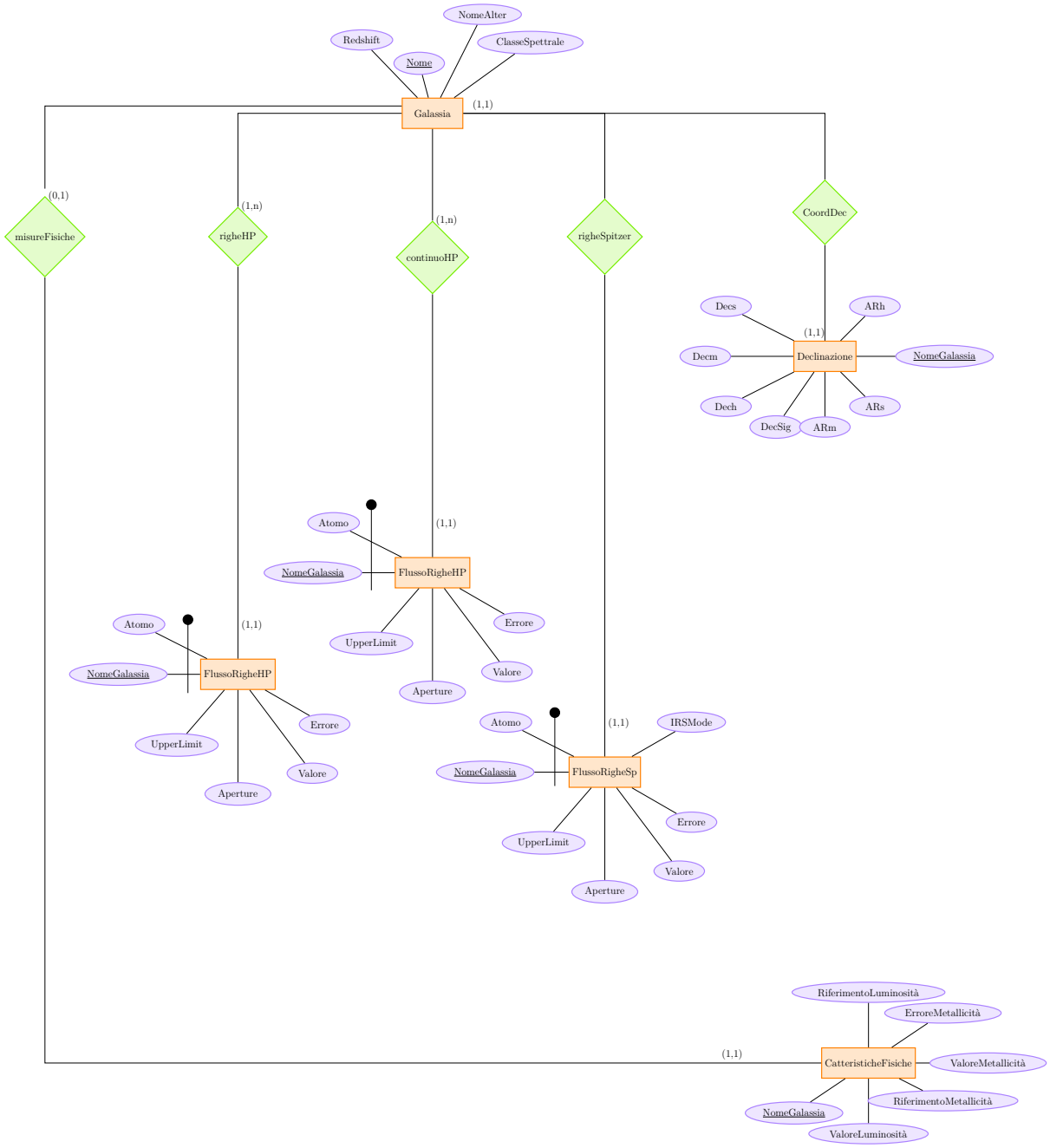


Figura 7: Schema E-R alla luce delle modifiche suggerite dalle fasi successive

3.4 Traduzione dello schema ER

La traduzione di uno schema ER ristrutturato è una fase della progettazione molto più algoritmica rispetto alla stesura dello schema ER stesso. Infatti vi sono un insieme di traduzioni standard a cui è possibile attenersi e che verranno utilizzate anche in questo contesto.

Galassia(Nome, Redshift, NomeAlternativo, ClasseSpettrale)
Declinazione(DecSig, Dech, Decm, Decs, NomeGalassia)
FK: NomeGalassia **REFERENCES** Galassia
AscensioneRetta(ARh, ARm, ARs, NomeGalassia)
FK: NomeGalassia **REFERENCES** Galassia
Flusso(Atomo, Tipologia, NomeGalassia)
FK: NomeGalassia **REFERENCES** Galassia
FK: NomeGalassia **REFERENCES** Galassia
MisuraFlusso(Valore, Errore, Riferimento, IRSMode, Riferimento160μm, UpperLimit)
FK: NomeGalassia **REFERENCES** Galassia
CaratteristicheFisiche(Valore, Errore, Riferimento, Tipologia, NomeGalassia)
FK: NomeGalassia **REFERENCES** Galassia

FlussoGalassia(NomeGalassia, Tipologia, Atomo)
FK: NomeGalassia **REFERENCES** Galassia
FK: Tipo, Atomo **REFERENCES** *Flusso* NOT NULL

4 Implementazione

4.1 Class diagram

Come ricordiamo dai principi di ingegneria del software, la fase di *analisi* definisce l'ambito di lavoro del sistema da realizzare, ovvero quali sono i compiti che deve effettivamente svolgere. La fase di *progettazione* per converso si occupa di definire in che modo questi compiti verranno portati a termine dal sistema.
Il fine più importante di questa fase è la stesura di un diagramma delle classi che rifletta ciò che il committente ha richiesto e che è stato inserito fra i requisiti funzionali del sistema. Di seguito viene riportato il diagramma delle classi per il sistema di cui si occupa la relazione, suddiviso sulla base dei requisiti funzionali in modo da risultare più chiaro. Nella stessa ottica, sono stati inseriti nelle classi solo attributi e metodi che vengono effettivamente richiamati nell'ambito del requisito.

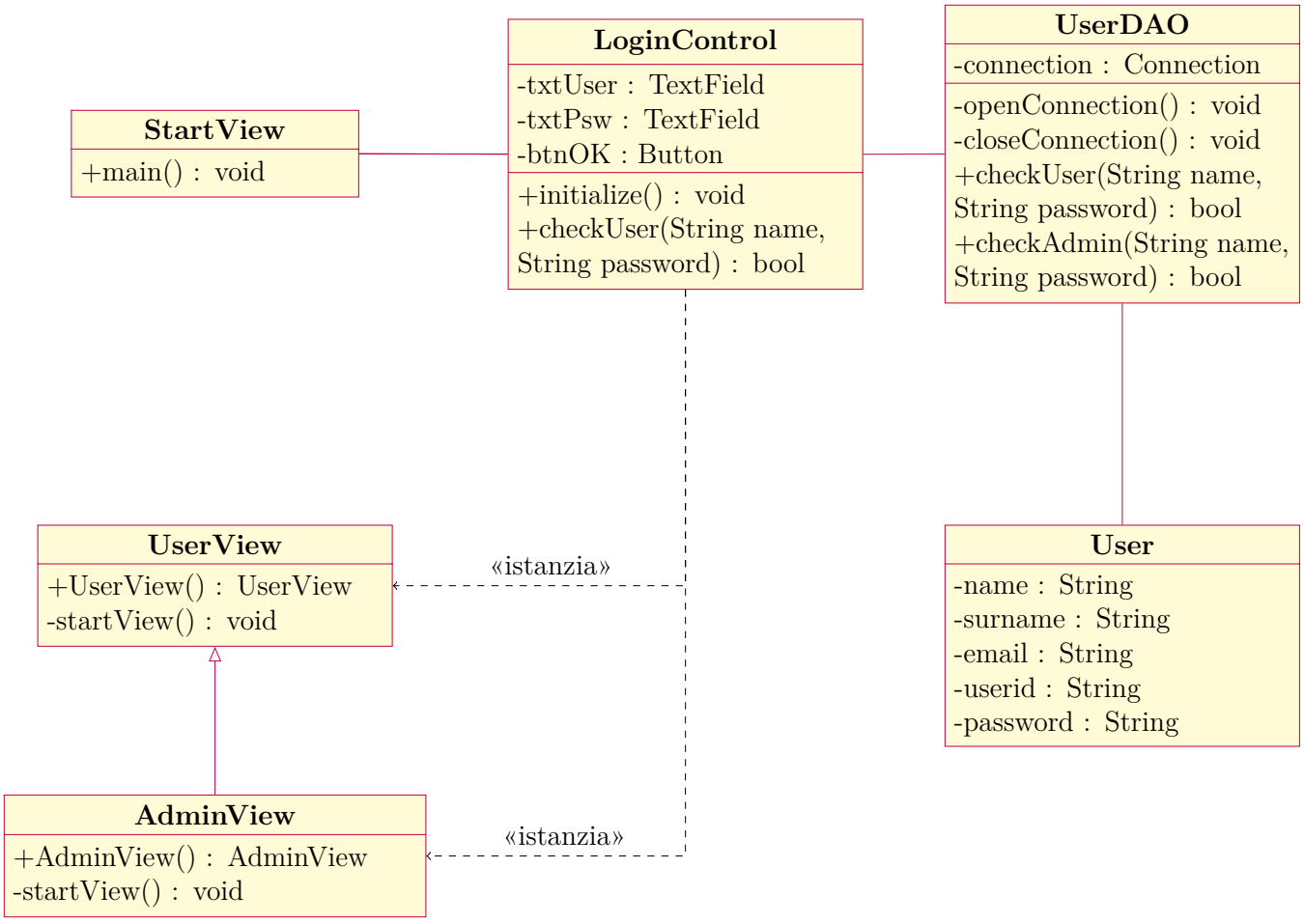


Figura 8: Vista requisito funzionale #1

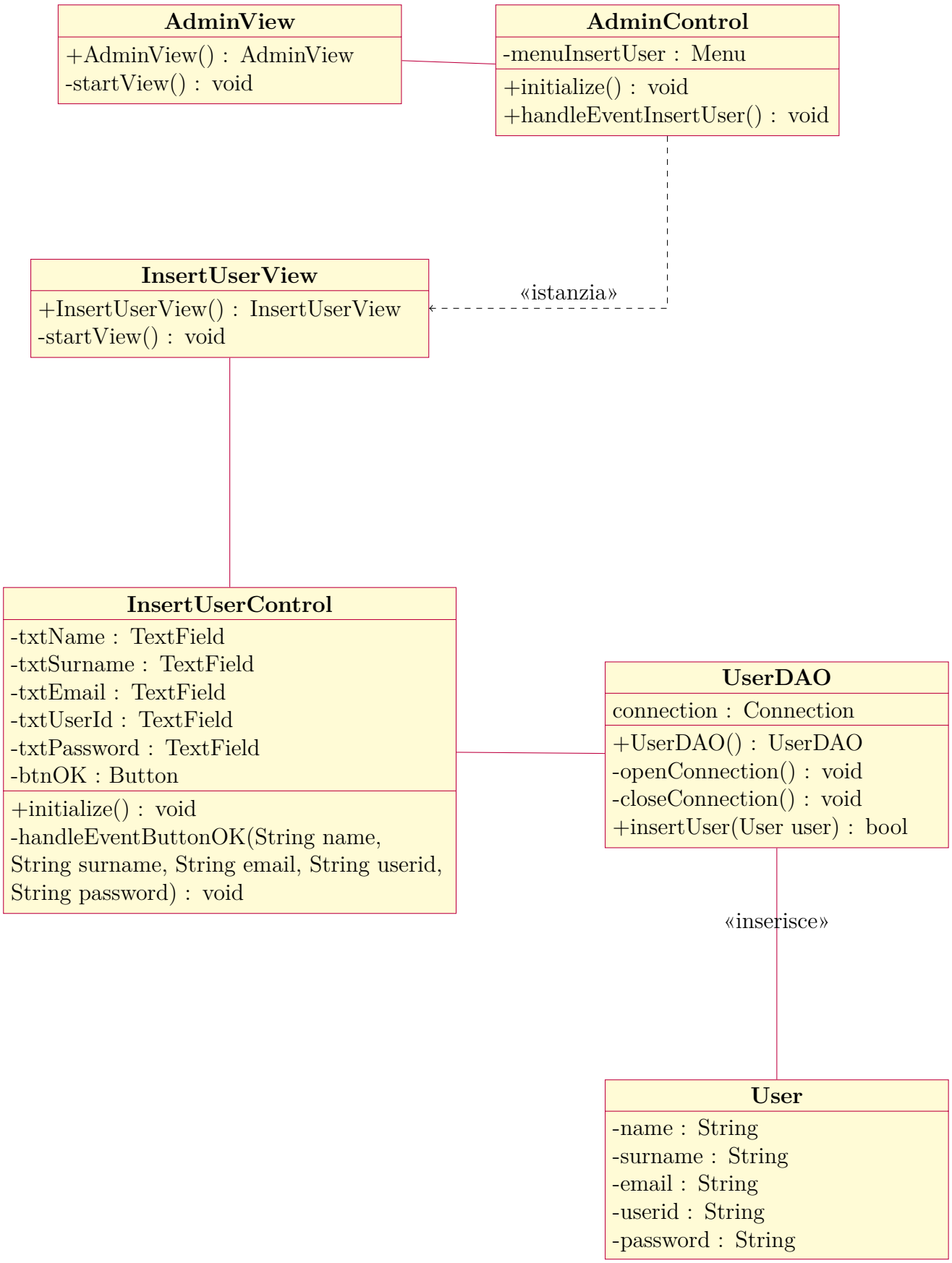


Figura 9: Vista requisito funzionale #3 - Inserimento di un utente

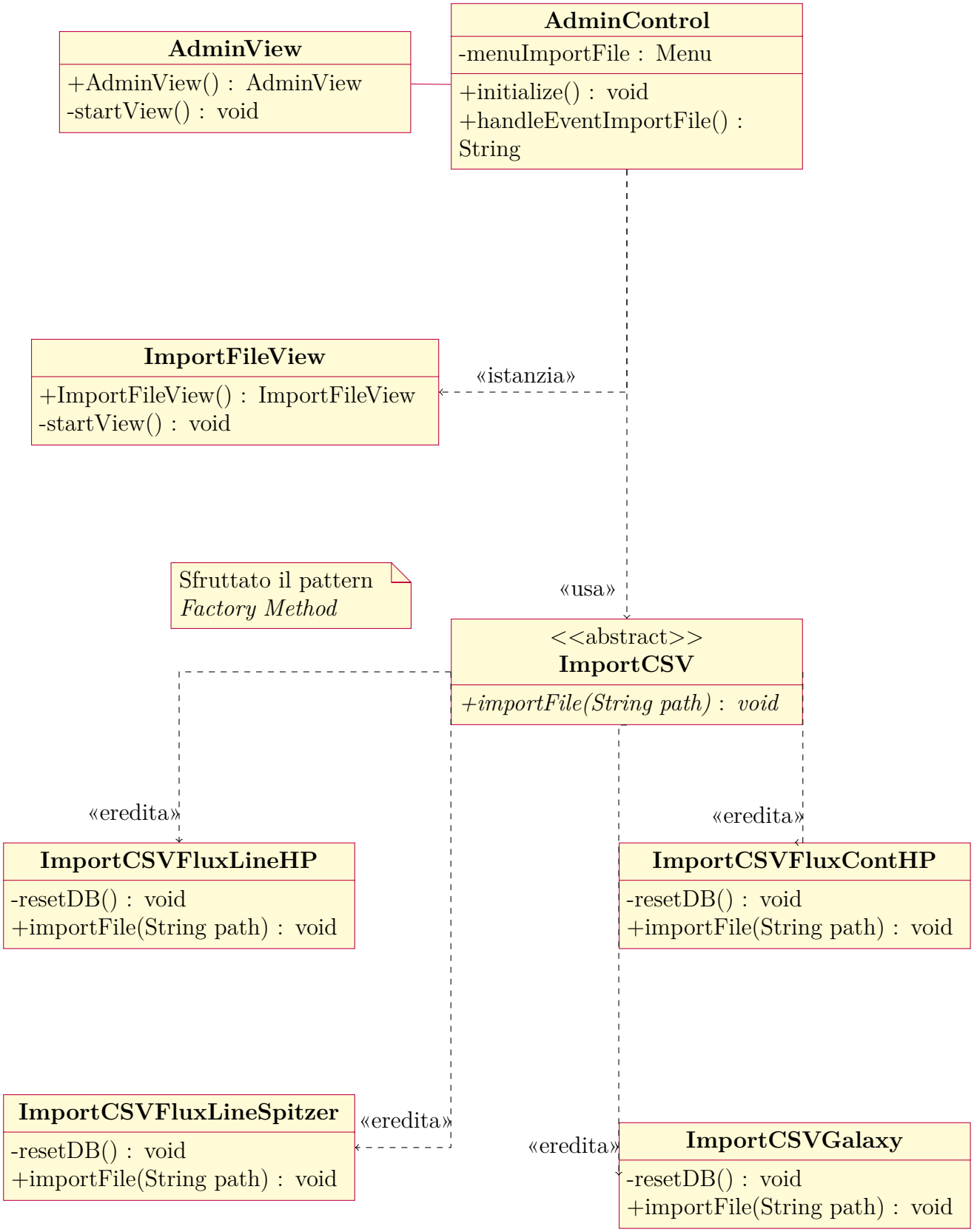


Figura 10: Vista requisito funzionale #4 - Import di un file

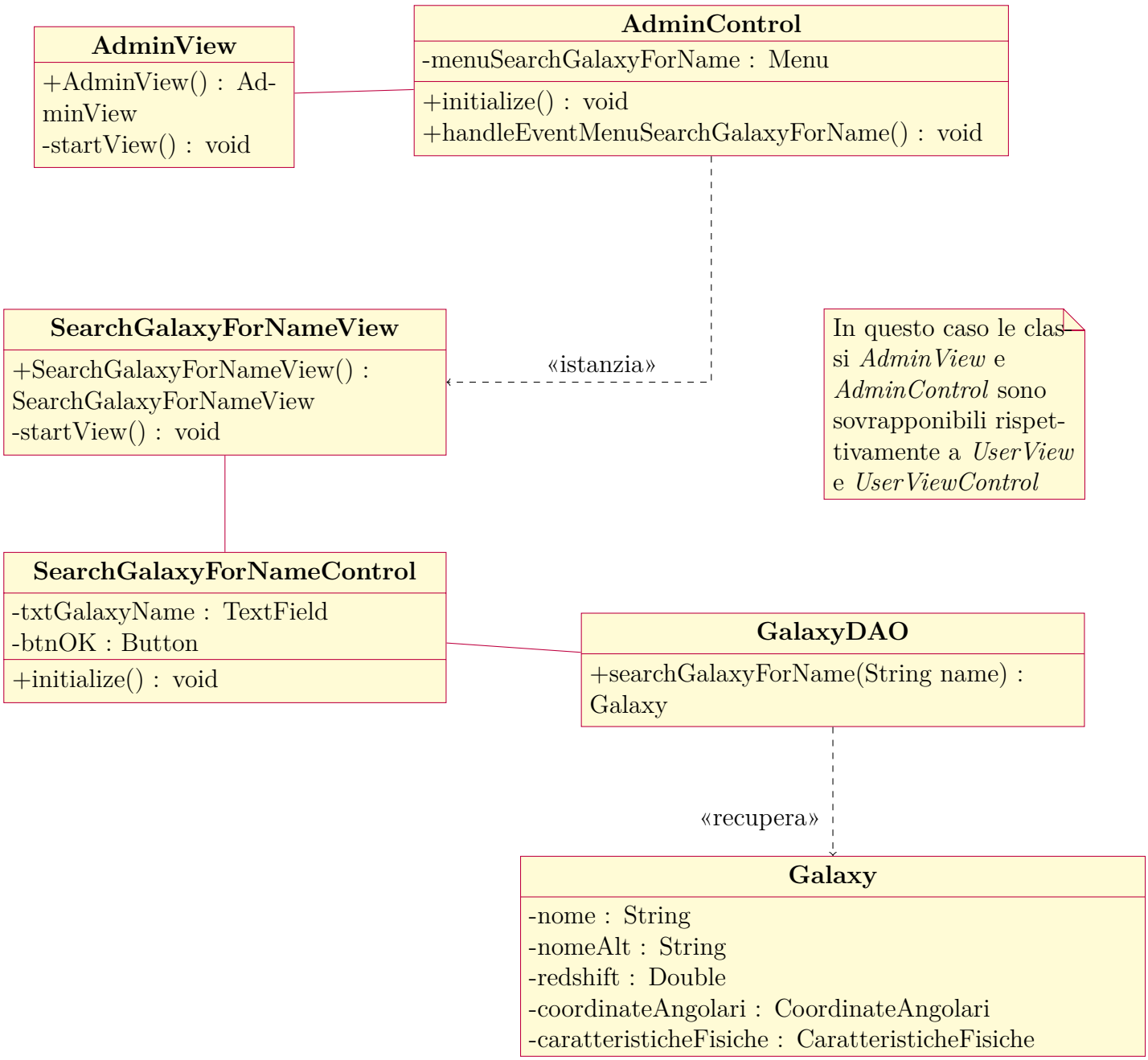


Figura 11: Vista requisito funzionale #5 - Ricerca di una galassia per nome

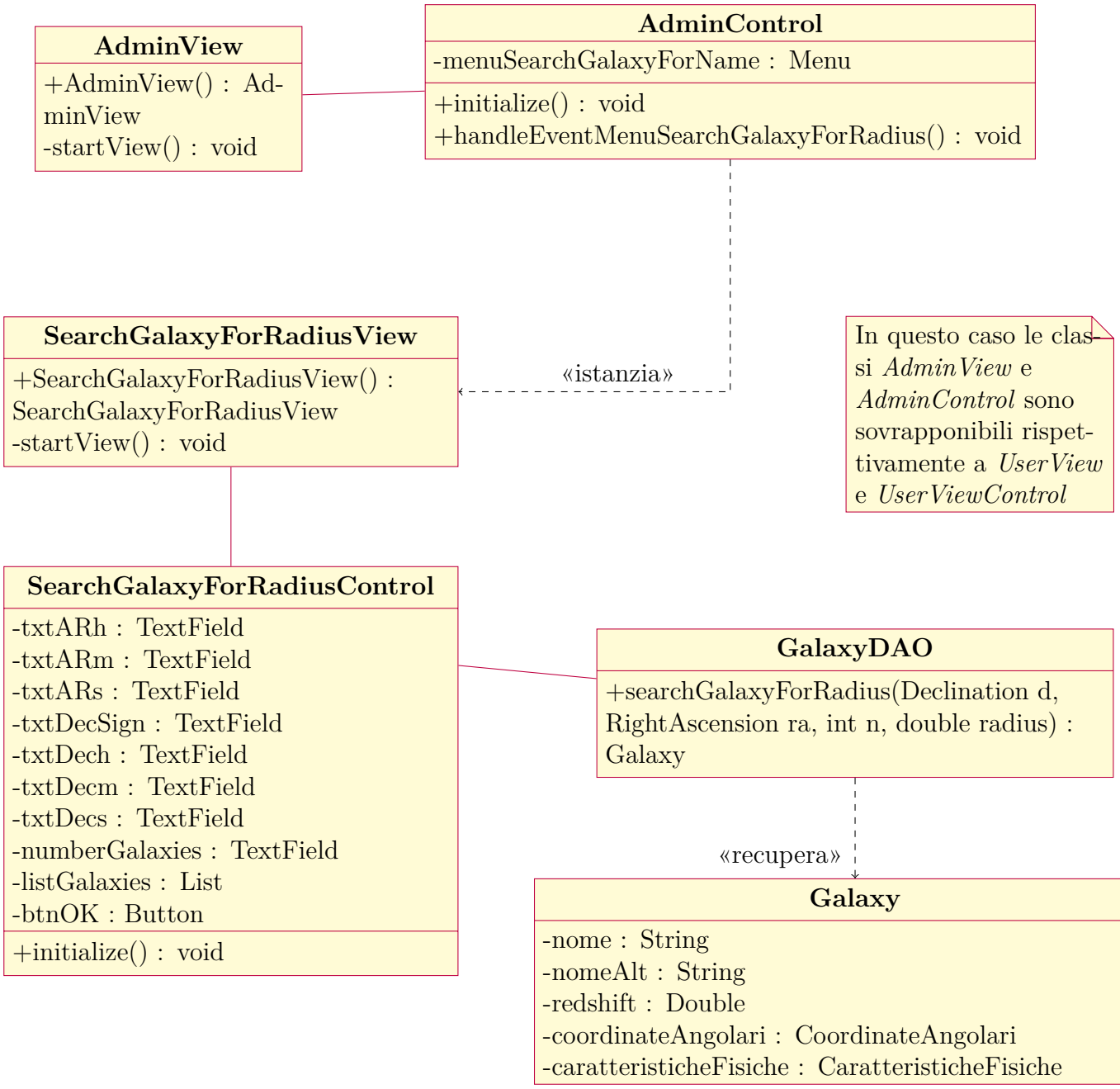


Figura 12: Vista requisito funzionale #6 - Ricerca le prime n galassie all'interno di un raggio

4.2 Scelta dell'ambiente

Dopo aver passato i tre livelli di progettazione del sistema informatico, possiamo passare alla fase di implementazione. La scelta del linguaggio di programmazione o dell'ambiente su cui sviluppare il progetto non é ovvia vista la moltitudine di alternative presenti.
Di certo c'è il linguaggio SQL non offre funzioni di interazione con i file né tanto meno la possibilità di c interfacce utente, quindi le funzioni SQL che comunicano con il database vanno necessariamente inserite in un linguaggio di programmazione che metta a disposizione tutte quelle funzioni che SQL non mette a disposizione.

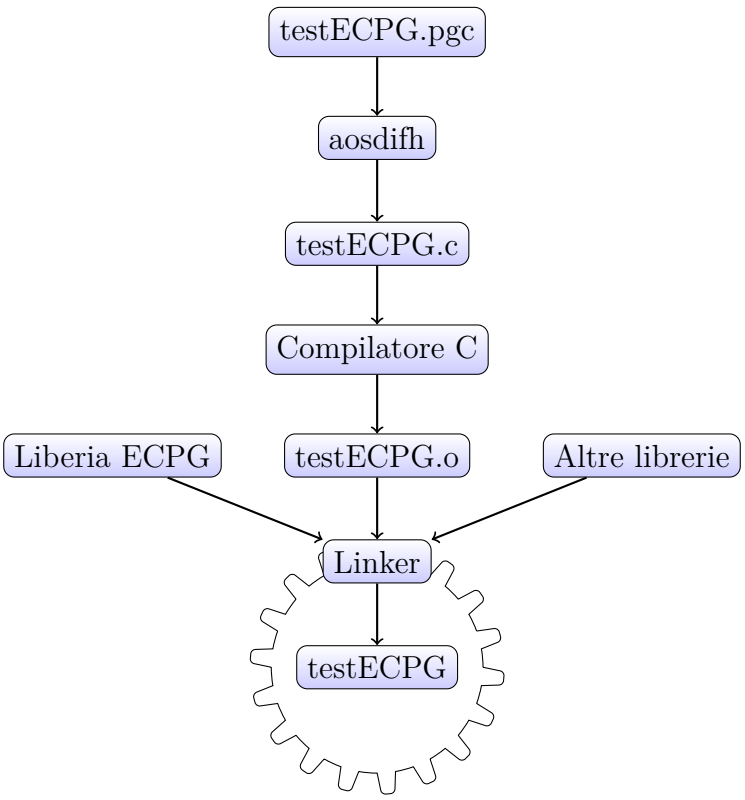
L'adozione di SQL all'interno di linguaggi di programmazione che fossero completi dal punto di vista operazionale ha attraversato varie fasi di complessità sintetizzabili in tre punti:

1. SQL integrato nei linguaggi di programmazione
2. Librerie sviluppate a partire dalle istruzioni SQL
3. Object Relational Mapping

Il problema principale che si é risolto al fine di integrare le query SQL in un altro linguaggio di programmazione é stato senz'altro lo sviluppo di software che fungesse da tramite tra i due linguaggi.

4.2.1 ECPG

A titolo di esempio viene riportato il caso di *ecpg*, un applicativo presente nell'installazione del DBMS Postgresql, che si occupa di tradurre un sorgente in linguaggio C il cui file ha estensione .pgc, fornendo in output un nuovo sorgente, stavolta con estensione .c, che si serve delle particolari funzioni di libreria del DBMS. Questo sorgente verrá poi compilato come un normale programma in linguaggio C. Questo processo viene riassunto nell'immagine qui di seguito:



Al fine di avere un'idea chiara di come si procede lavorando con questi strumenti, il processo é stato ripetuto in ambiente Ubuntu con dei sorgenti di test. Si parte dunque da un sorgente *testECPG.pgc*:

Listing 1: testECPG.pgc

```
#include <stdio.h>
#include <sys/types.h>
#include <libpq-fe.h>
#include <libpq/libpq-fs.h>

/* 1 - Viene inclusa la libreria relativa alla gestione degli errori tramite
   la variabile globale sqlca (SQL communication area) */
EXEC SQL INCLUDE sqlca;

int main (int argc, char **argv)
{
   /* 2 - Comando che realizza la connessione al database */
   EXEC SQL CONNECT TO testDB@localhost:5432 USER user/password;

   /*Come succede molto spesso nella gestione degli errori in linguaggio C, il
      valore ritornato da una funzione viene utilizzato anche come codice di
      errore, considerando lo 0 come esecuzione andata a buon fine */
   if (sqlca.sqlcode != 0) {
      printf("Errore di connessione al DB\n");
      printf("Errore %d\n", (int)sqlca.sqlcode);
   }

   /* 3 - Esegui un'operazione di test */
   EXEC SQL CREATE TABLE TableTest (number integer, ascii char(16));
   fprintf (stdout, "Created table TestTable\n");

   /* 4 - Esegui le operazioni */
   EXEC SQL COMMIT;

   /* 5 - Disconnessione dal database */
   EXEC SQL DISCONNECT ALL;

   return EXIT_SUCCESS;
}
```

che viene preprocessato dal comando *ecpg*:

username@host:./$ ecpg ecpg_sample.pgc

Il file risultante con estensione .c utilizza le particolari funzioni di libreria offerte dal DBMS PostgreSQL:

Listing 2: testECPG.c

```
/* Processed by ecpg (4.11.0) */
/* These include files are added by the preprocessor */
#include <ecpglib.h>
#include <ecpgerrno.h>
```

```
#include <sqlca.h>
/* End of automatic include section */

#line 1 "ecpg_sample.pgc"
#include <stdio.h>
#include <sys/types.h>
#include <libpq-fe.h>
#include <libpq/libpq-fs.h>

/* 1 - Viene inclusa la libreria relativa alla gestione degli errori tramite
    la variabile globale sqlca (SQL communication area) */

#line 1 "/usr/include/postgresql/sqlca.h"
#ifndef POSTGRES_SQLCA_H
#define POSTGRES_SQLCA_H

#ifndef PGDLLIMPORT
#if  defined(WIN32) || defined(__CYGWIN__)
#define PGDLLIMPORT __declspec (dllimport)
#else
#define PGDLLIMPORT
#endif  /* __CYGWIN__ */
#endif  /* PGDLLIMPORT */

#define SQLERRMC_LEN 150

#ifdef __cplusplus
extern     "C"
{
#endif

struct sqlca_t
{
   char    sqlcaid[8];
   long    sqlabc;
   long    sqlcode;
   struct
   {
      int        sqlerrml;
      char    sqlerrmc[SQLERRMC_LEN];
   }       sqlerrm;
   char    sqlerrp[8];
   long    sqlerrd[6];
   /* Element 0: empty             */
   /* 1: OID of processed tuple if applicable    */
   /* 2: number of rows processed        */
   /* after an INSERT, UPDATE or      */
   /* DELETE statement           */
   /* 3: empty             */
   /* 4: empty             */
   /* 5: empty             */
   char    sqlwarn[8];
   /* Element 0: set to 'W' if at least one other is 'W' */
```

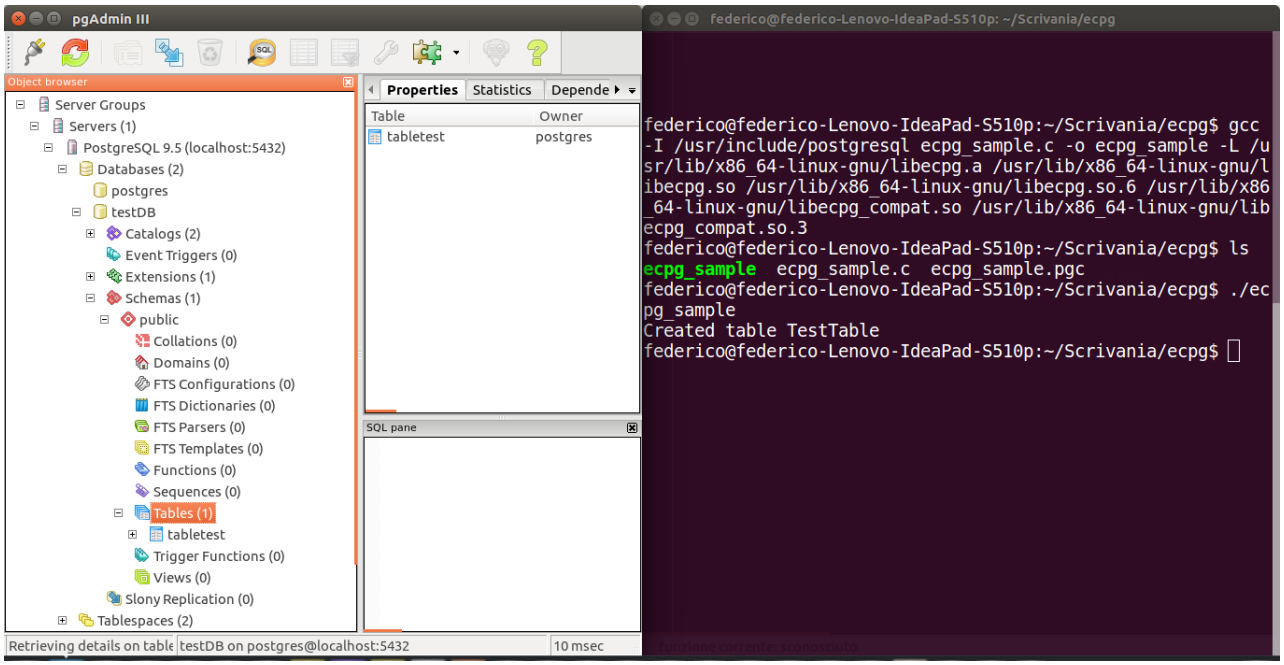
```
   /* 1: if 'W' at least one character string  */
   /* value was truncated when it was    */
   /* stored into a host variable.         */

   /*
    * 2: if 'W' a (hopefully) non-fatal notice occurred
    */   /* 3: empty */
   /* 4: empty             */
   /* 5: empty             */
   /* 6: empty             */
   /* 7: empty             */

   char    sqlstate[5];
};

struct sqlca_t *ECPGget_sqlca(void);

#ifndef POSTGRES_ECPG_INTERNAL
#define sqlca (*ECPGget_sqlca())
#endif

#ifdef __cplusplus
}
#endif

#endif

#line 7 "ecpg_sample.pgc"

int main (int argc, char **argv)
{
   /* 2 - Comando che realizza la connessione al database */
   { ECPGconnect(__LINE__, 0, "testDB@localhost:5432" , "postgres" ,
      "portento123" , NULL, 0); }
#line 12 "ecpg_sample.pgc"

   /*Come succede molto spesso nella gestione degli errori in linguaggio C, il
      valore ritornato da una funzione viene utilizzato anche come codice di
      errore, considerando lo 0 come esecuzione andata a buon fine */
   if (sqlca.sqlcode != 0) {
      printf("Errore di connessione al DB\n");
      printf("Errore %d\n", (int)sqlca.sqlcode);
   }

   /* 3 - Esegui un'operazione di test */
   { ECPGdo(__LINE__, 0, 1, NULL, 0, ECPGst_normal, "create table TableTest (
      number integer , ascii char ( 16 ) )", ECPGt_EOIT, ECPGt_EORT);}
#line 21 "ecpg_sample.pgc"

   fprintf (stdout, "Created table TestTable\n");

   /* 4 - Esegui le operazioni */
   { ECPGtrans(__LINE__, NULL, "commit");}
```

```
#line 25 "ecpg_sample.pgc"

   /* 5 - Disconnessione dal database */
   { ECPGdisconnect(__LINE__, "ALL");}
#line 28 "ecpg_sample.pgc"

   return EXIT_SUCCESS;
}
```

Questo file puó essere compilato tramite GCC, avendo l'accortezza di linkare le librerie di cui il programma ha bisogno. Il procedimento va eseguito manualmente nel caso in cui esse non siano presenti nella stessa directory del sorgente o nelle directory predefinite di GCC.

gcc -I /usr/include/postgresql ecpg_sample.c -o ecpg_sample -L /usr/lib/x86_64-linux-gnu/libecpg.a /usr/lib/x86_64-linux-gnu/libecpg.so /usr/lib/x86_64-linux-gnu/libecpg.so.6 /usr/lib/x86_64-linux-gnu/libecpg_compat.so /usr/lib/x86_64-linux-gnu/libecpg_compat.so.3

Il risultato che otterremo sará il seguente:



Nel procedimento riportato, l'utilizzo del linguaggio C come ospitante l'SQL non risulta efficiente in quanto, oltre alla complessitá del processo di compilazione, ad ogni modifica del codice si pone la necessitá di ripetere l'iter. Un makefile potrebbe tornare utile allo scopo, ma in ogni caso andrebbe mantenuto a seguito dell'aggiornamento del codice.

Non abbiamo significativi miglioramenti utilizzando la Call Level Interface offerta dal DBMS PostgreSQL relativamente al linguaggio C, ovvero una serie di librerie che rendono meno distaccato il codice relativo alle interrogazioni sul database rispetto al resto del codice.
Il codice risulta piú omogeneo, ma non abbiamo risolto il problema del constant recompiling:

Listing 3: testPGSQL.c

```
#include <stdio.h>
#include <stdlib.h>
#include <libpq-fe.h>

void do_exit(PGconn *conn) {

    PQfinish(conn);
    exit(1);
}

int main() {

    PGconn *conn = PQconnectdb("user=user password=password dbname=testDB");

    if (PQstatus(conn) == CONNECTION_BAD) {

        fprintf(stderr, "Connection to database failed: %s\n",
            PQerrorMessage(conn));
        do_exit(conn);
    }

    char *user = PQuser(conn);
    char *db_name = PQdb(conn);
    char *pswd = PQpass(conn);

    printf("User: %s\n", user);
    printf("Database name: %s\n", db_name);
    printf("Password: %s\n", pswd);

    PQfinish(conn);

    return 0;
}
```

Un linguaggio che si propone come soluzione sia al constant recompiling sia alla difficoltá della compilazione stessa é Java.

4.2.2 JDBC

La connessione alle basi di dati é una funzionalitá che Java ha messo a disposizione giá a partire dalla versione 1.2. Questo servizio viene offerto dalle API JDBC (Java Database Connectivity), le quali permettono la gestione dei dati indipendentemente dal DBMS utilizzato, purché SQL-based, mantenendo intatto il paradigma Java "Write once, Run anywhere".

Un programma Java che sia stato scritto con l'intento di accedere ad un DBMS SQL-based si servirá delle API JDBC per interfacciarsi con la base di dati, essendo ora provvisto di metodi per l'interrogazione e la modifica dei dati. Esse saranno indipendenti dal DBMS impiegato, mantenendo cosí l'indipendenza del modello logico.
Queste API dovranno interfacciarsi con un gestore dei driver, il quale si servirá anche delle API JDBC per comunicare con lo specifico DBMS. Il driver viene caricato tramite la chiamata *Class.forName*() che carica la classe se l'operazione non é stata svolta precedentemente. Tutti i driver JDBC hanno un blocco statico all'interno della definizione della classe, il quale é deputato alla registrazione del driver nel *DriverManager*. Questo blocco di codice, che potrebbe somigliare molto allo snippet riportato di seguito, verrá eseguito proprio in seguito alla chiamata del metodo *forName*() di *Class*:

Snippet da [4]

```
static {
    try {
            java.sql.DriverManager.registerDriver(new Driver());
    } catch (SQLException e) {
        throw new RuntimeException("Can't register driver!");
    }
}
```

La classe *DriverManager* dovrá quindi scorrere tutti i driver registrati e scegliere a runtime quello appropriato per svolgere la richiesta.

La Object-Relational Mapping é una tecnica di programmazione che viene adottata quando si vuole favorire l'integrazione di DBMS relazionali con linguaggi di programmazione orientati agli oggetti. Questo processo di fusione tra programma ad oggetti e database relazione viene messo in atto tramite metodi resi disponibili dal framework ORM. Alcuni dei piú diffusi sono:

- Hibernate
- Spring DAO
- Enterprise JavaBeans Entity Beans

4.2.3 Hibernate

Hibernate é un framework

4.3 Gestione delle versioni: Git & Github

In materia di controllo delle versioni del software, Git risulta essere uno tra i più diffusi al mondo. Linus Thorvalds cominciò lo sviluppo di questo applicativo per sostenere la crescita di contributi al suo primo grande progetto, Linux, come viene spiegato in un passo di questa intervista rilasciata presso il canale TED (link).
Git consente principalmente di sviluppare un codice non necessariamente monolitico, dando la possibilità di creare delle diramazioni (*branch*) al flusso principale. Una volta che le diramazioni sono abbastanza mature, c'è la possibilità di unire (*merge*) il codice sviluppato indipendemente dal *master branch*. Queste due caratteristiche sono particolarmente importanti in un progetto distribuito, sviluppato da più persone.

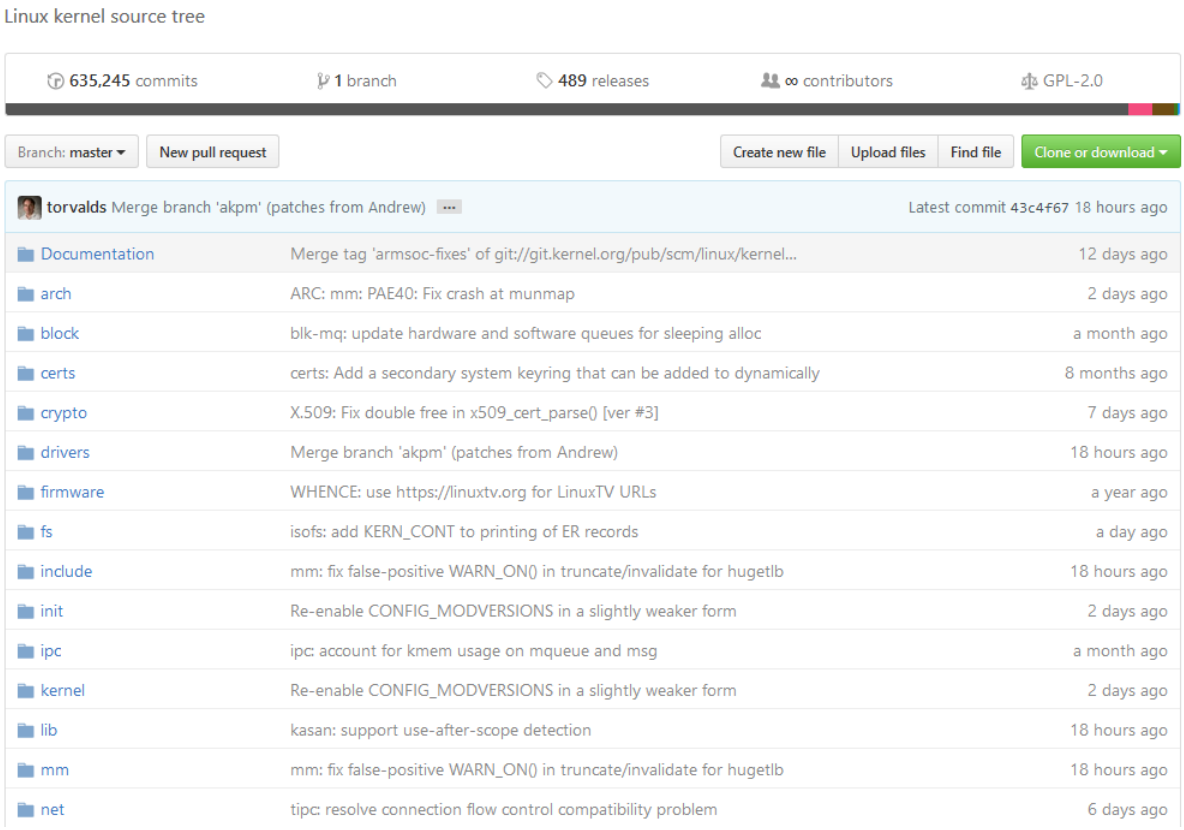


Figura 13: Pagina web su Github relativa al progetto Linux - Dicembre 2016

In questo caso il progetto risulta decisamente poco distribuito ma lo strumento si rivela ugualmente utile ogni volta che si ha intenzione di tornare ad una versione (funzionante) precedente a quella che si ha fra le mani.
Git è un sistema di controllo delle versioni distribuito ed in quanto tale necessita di spazio in rete per caricare il codice sviluppato, affinchè possa essere utilizzato da più persone o anche come backup personale nel caso di un progetto in solitaria. Se non si dispone di uno spazio in rete, ecco che entra in gioco la piattaforma Github, servizio di hosting di rete che si propone anche come intermediario tra l'utente e Git tramite i software standalone. Sul sito di Github è possibile inoltre visionare i repository pubblici e, se presente, leggere la wiki a corredo del codice.

L'indirizzo del repository relativo a questo progetto è repo

5 Cenni alla progettazione fisica

Listing 4: Cerca indici

```
SELECT i.relname as indname,
   i.relowner as indowner,
   idx.indrelid::regclass,
   am.amname as indam,
   idx.indkey,
      ARRAY(
         SELECT pg_get_indexdef(idx.indexrelid, k + 1, true)
         FROM generate_subscripts(idx.indkey, 1) as k
         ORDER BY k
      )  as indkey_names,
         idx.indexprs IS NOT NULL as indexprs,
         idx.indpred IS NOT NULL as indpred
            FROM  pg_index as idx
               JOIN  pg_class as i
               ON    i.oid = idx.indexrelid
                  JOIN  pg_am as am
                  ON    i.relam = am.oid;
```

Bibliografia

[1] Atzeni, Ceri, Fraternali, Paraboschi, Torlone "Basi di dati Modelli e linguaggi di interrogazione" Quarta edizione, McGraw-Hill, 2013

[2] Beneventano, Bergamaschi, Guerra, Vincini "Progetto di Basi di Dati Relazionali lezioni ed esercizi" Pitagora Editrice Bologna, 2007

[3] https://it.wikipedia.org/wiki/Object-relational_mapping;

[4] http://www.xyzws.com/javafaq/what-does-classforname-method-do/17

[5] https://www.postgresql.org/docs/9.3/static/client-interfaces.html

[6] http://www.astronomia.com

[7] http://docenti.unicam.it/tmp/2441.pdf

[8] J. Arlow, Neustadt, Uml e Unified Process