Corso di Laurea in

Informatica Umanistica Specialistica

Relazione

Verso un catalogo unico delle biblioteche di area pisana

Candidato: *Chiara Aiola*

Relatore: *Vittore Casarosa*

Correlatore: *Maria Simi*

Anno Accademico 2009-2010

Indice generale

Indice delle tabelle

Indice delle illustrazioni

Indice delle porzioni di codice

Introduzione

Il lavoro che viene presentato prende spunto da alcune riflessioni fatte sul funzionamento del Meta Opac Pisano, e nasce dalla proposta dell'autore del software del meta opac di provare a realizzare un repertorio unico delle opere presenti nei vari cataloghi delle biblioteche che fanno parte del Meta Opac Pisano.

Il Meta Opac Pisano è un motore di ricerca di documenti (libri, periodici) che interroga contemporaneamente i cataloghi delle biblioteche presenti nel sistema. La ricerca avviene per parole-chiave, e i risultati della ricerca vengono elencati per biblioteca di appartenenza.

Nonostante il Meta Opac Pisano svolga efficacemente le sue funzioni, sono stati rilevati due aspetti da migliorare: il primo riguarda la visualizzazione dei risultati della ricerca, il secondo l'interrogazione dei cataloghi.

Quando si esegue una ricerca, la pagina che viene presentata contiene l'elenco delle biblioteche del meta opac con accanto un numero che indica la quantità di record trovati in base ai criteri di ricerca specificati; per vedere l'elenco effettivo dei documenti bisogna quindi consultare uno per volta tutti i cataloghi per cui sono stati restituiti dei risultati. Questo potrebbe portare a una eccessiva confusione e frammentazione della ricerca di un'opera, in quanto manca una panoramica d'insieme dei risultati della ricerca. Il lavoro che si propone cerca di proporre una soluzione a questo difetto, cercando di dare una visione dei risultati della ricerca più completa e organica.

L'altro aspetto su cui ci si è soffermati è il sistema di ricerca del meta opac. Una volta impostati i filtri per la ricerca, il programma interroga uno per uno tutti i cataloghi delle biblioteche. Avendo un solo repertorio per tutte le opere invece, la ricerca si potrebbe effettuare su questo risparmiando risorse del sistema.

Il lavoro qui esposto quindi ha una doppio raggio di azione: nel backend, con un software che permette di ricavare, dall'unione dei singoli cataloghi, un repertorio unico delle opere presenti nelle biblioteche del Meta Opac Pisano, e nel frontend, intervenendo nell'interfaccia del sistema.

Inoltre il programma creato permette di avere dei prodotti accessori che possono essere utili per la gestione e condivisione delle informazioni, come il repertorio delle biblioteche in formato xml.

1. Uno studio sui sistemi di accesso ai cataloghi delle biblioteche

Da quando le biblioteche sono passate dalla tradizionale catalogazione a schede cartacee a un sistema informatizzato di gestione di schede elettroniche, si sono sviluppati programmi che permettono di accedere a questi archivi digitali attraverso apposite interfacce. Oggi quasi ogni biblioteca si è dotata di questi sistemi di consultazione, tra cui anche il sistema bibliotecario di area pisana, oggetto del lavoro qui presentato. Per questo motivo è stato trovato interessante uno studio, condotto da un gruppo di ricerca costituito dall'Università Ca' Foscari di Venezia, Università di Pavia e dall'Associazione Italiana Biblioteche, che tra il 2003 e il 2004 ha preso in esame i sistemi di accesso informatizzati ai cataloghi delle biblioteche; purtroppo il metaopac pisano, appena costituito ai tempi dell'inizio di questa ricerca, non è rientrato tra i sistemi valutati. Questo studio dal titolo “Opac Semantici” (Gnoli, Ridi, Visitin,2004, *“Di che parla questo catalogo? Un'indagine sugli accessi semantici negli opac italiani”*) è stato preso in considerazione durante diverse fasi della realizzazione del progetto qui presentato, per questo si è ritenuto utile presentare brevemente i punti dell'indagine che hanno suscitato maggiore attenzione.

1.1. Cos'è un OPAC

Uno dei concetti fondamentali alla base del progetto “Opac Semantici” è proprio quello di OPAC, acronimo di Online Pubblic Access Catalogue.

Un opac è un sistema di accesso ai cataloghi delle biblioteche, e segna il passaggio dalla catalogazione cartacea a quella elettronica. Infatti solo con l'informatizzazione delle biblioteche negli anni '80 è stato possibile creare dei sistemi di consultazione dei cataloghi elettronici, che avrebbero dovuto permettere agli utenti delle biblioteche di consultare autonomamente i cataloghi informatizzati per svolgere le proprie ricerche. L'obiettivo di un Opac infatti è proprio quello di rendere accessibili i cataloghi delle biblioteche a chi non ha competenze biblioteconomiche o a chi non ha ricevuto un apposito addestramento. Bisogna dire che la realizzazione piena di questo obiettivo deve ancora realizzarsi, in quanto non sempre i sistemi di accesso ai cataloghi messi a

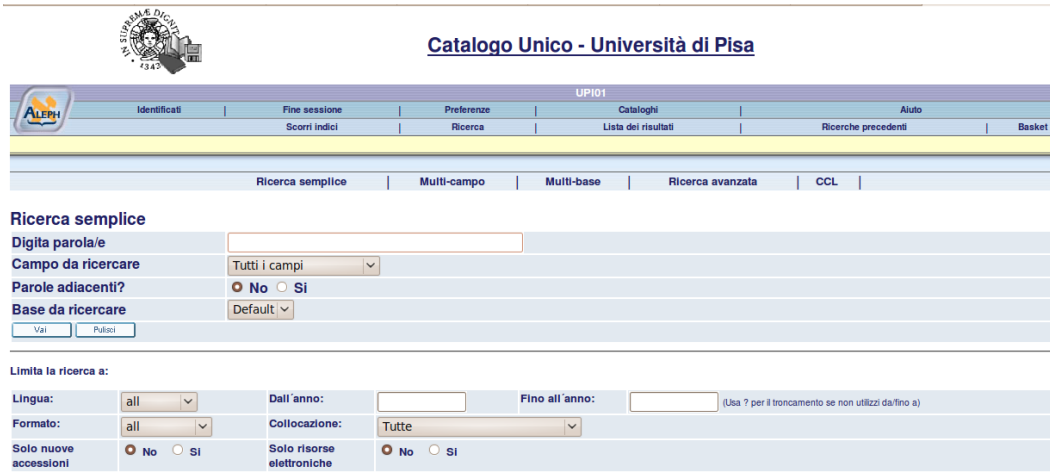


Immagine 1: Esempio di Opac, Il motore di ricerca del catalogo della biblioteca del Catalogo dell'Università di Pisa

disposizione dalle varie biblioteche risultano chiari all'utente e gli consentono di effettuare le ricerche in maniera efficace.

Un Opac è costituito da una interfaccia che mette a disposizione di chi vuole consultare il catalogo diversi modi per effettuare le ricerche.

I sistemi di ricerca che di solito si trovano negli Opac sono tre, *Scan, Search* e *Search* in ambiente testuale.

Lo *Scan* è lo scorrimento di liste, ad esempio di titoli o di autori (Imm. 2), il *Search* è la ricerca attraverso parole chiave da inserire in appositi campi (Imm. 3), mentre il *Search* in ambiente testuale è la ricerca a testo libero (Imm. 4).

Immagine 2: Esempio di Ricerca per scorrimento di liste. Biblioteca dell'Università di Venezia, http://iuavbc.iuav.it.

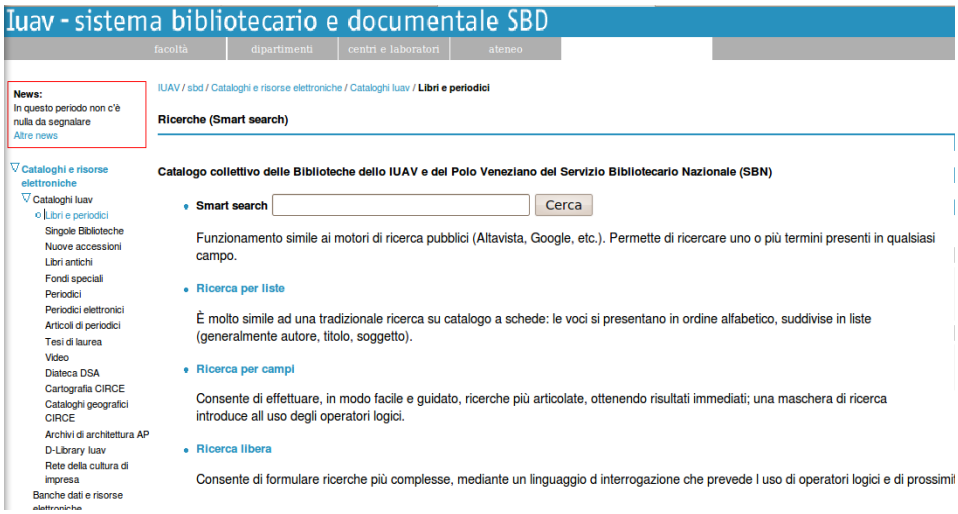


Immagine 3: Esempio di ricerca a testo libero. Biblioteca dell'Università di Venezia, http://iuavbc.iuav.it.

Immagine 4: Esempio di ricerca per compilazione di campi. Biblioteca dell'Università di Venezia, http://iuavbc.iuav.it.

Le ricerche di tipo *search* inoltre possono essere arricchite da operatori booleani che servono per organizzare la ricerca su più di un campo. Questi operatori sono AND, OR, NOT, e sono presenti in quasi tutti gli opac, anche se il loro significato, è opportuno

evidenziarlo, spesso rimane oscuro a un utente non esperto.

Accanto al *Search* e allo *Scan* esiste un altro tipo di ricerca, che si potrebbe chiamare *Surf*. Quest'ultimo sistema consiste nella navigazione del catalogo con link ipertestuali che, partendo da una singola scheda bibliografica, permettono di avviare altri tipi di ricerche, a loro volta di tipo *search*, se il termine cliccato lancia una nuova estrazione di dati, di tipo *list* se apre una lista di consultazione, o costituire un collegamento diretto a altre schede.

Sulla frequenza dei sistemi di ricerca presenti negli Opac Italiani si fa riferimento ad un articolo (Ridi, Gnoli, Visitin, 2004, "*Come vogliamo chiamarli? Operatori booleani e altre tecniche di information retrieval negli opac italiani*") che fornisce una serie di statistiche sulle tipologie di ricerca usate e sulla terminologia usata dagli opac italiani.I dati della ricerca presentati nell'articolo saranno tenuti in considerazione per la realizzazione dell'interfaccia di consultazione del catalogo unico delle biblioteche di area pisana.

1.2. Gli Opac semantici

Gli opac italiani censiti dall'Aib, associazione italiana biblioteche[1] al 2009, sono 1261. Negli ultimi anni la pratica di dotare le biblioteche di un opac è aumentata, e questo ha portato a un notevole incremento dei sistemi di accesso ai cataloghi online, quantità che dal 2004 al 2009 è raddoppiata. Proprio del 2004 è lo studio già citato di Aib e delle Università di Pavia e Ca' Foscari di Venezia sugli opac italiani. I risultati dell'indagine del gruppo di lavoro sono stati ritenuti molto utili per questo lavoro, per cui è sembrato interessante proporne una sintesi. Tutti i dati presentati in questo paragrafo sono tratti da: Gnoli, Ridi, Visitin,2004, *"Di che parla questo catalogo? Un'indagine sugli accessi semantici negli opac italiani"*.

I dati raccolti tra il 2003 e il 2004 sugli Opac italiani, hanno permesso di valutare quanto i sistemi di accesso ai cataloghi permettano agli utenti di effettuare ricerche basate sul contenuto delle opere. Per questo il loro lavoro è stato chiamato progetto "Opac semantici".

Il loro lavoro ha preso in considerazione un campione di 152 cataloghi su un totale di

1 http://www.aib.it/

oltre 600 opac censiti nel 2003 dall'Aib.

Ciò che la ricerca ha evidenziato è che, per quanto da sempre in biblioteconomia siano esistiti strumenti per la catalogazione semantica, queste tecniche non siano state totalmente sfruttate dai sistemi informatizzati, per cui gli opac che forniscono strumenti di ricerca basati sul contenuto delle opere spesso mostrano delle lacune.

Per quanto la ricerca per soggetto sia una pratica parecchio diffusa, presente tra il 75% e l'85% dei casi presi in esame, si osserva come spesso non siano fornite all'utente informazioni su come la classificazione per soggetto sia stata fatta e se e quali schemi preimpostati[2] siano stati seguiti. La mancanza di queste informazioni potrebbe non permettere all'utente di effettuare in modo ottimale le proprie ricerche. È importante sottolineare che si può considerare ricerca basata sul soggetto quella effettuata sugli effettivi campi “soggetto” delle schede bibliografiche, e non in generale su tutti i campi (in quel caso si parla di ricerca a testo libero, v. 2.1).

Nei casi in cui i cataloghi presentino una classificazione per argomento basata su schemi con delle classi di argomenti predefinite, la ricerca per quelle stesse classi invece è presente in una percentuale minore.

Emerge quindi che in generale il livello di “semanticità”, come lo hanno definito gli autori della ricerca qui presentata, è mediamente abbastanza basso, a causa della scarsa attenzione che si presta alla ricerca per argomento, che potrebbe al contrario essere molto significativa per gli utenti interessati non a un'opera nello specifico, rintracciabile semplicemente con una ricerca per titolo, ma a un tema in generale.

Alla luce di queste ricerche, il lavoro che qui si propone mira a indirizzare la costruzione del catalogo unico e della relativa interfaccia di consultazione verso un grado di semanticità migliore di quello attuale dell'opac del sistema bibliotecario pisano, per quanto è consentito dai dati di partenza.

2 Per la catalogazione semantica delle opere esistono degli schemi già definiti e adottati da più biblio-teche. L'argomento verrà approfondito a proposito dell'Opac del sistema bibliotecario pisano in 3.4

2. Il sistema bibliotecario d'area pisana

Il sistema bibliotecario pisano è nato nel 2003 dall'accordo dei Rettori delle tre università pisane, Università degli Studi di Pisa, Scuola Normale Superiore, Scuola Superiore Sant'Anna, e del presidente del Cnr, per sviluppare un servizio di accesso ai documenti di alta qualità e aperto a tutte le biblioteche di area pisana.

Il sistema di consultazione dei cataloghi, raggiungibile dall'indirizzo http://leonardo.isti.cnr.it, funziona come un motore di ricerca, che permette all'utente di inserire le parole chiave negli appositi campi e consultare i cataloghi delle biblioteche in cui sono contenute le opere trovate in base ai criteri di ricerca stabiliti.

2.1. I cataloghi

Le biblioteche che il sistema attualmente comprende sono divise in classi e sono:

Biblioteche Accademiche e di Ricerca

Biblioteca	N° record bibliografici	Classificazione Decimale Dewey	N° opere classificate
Università di Pisa	Monografie 779.654 Periodici 19.280	si	445.836
Scuola Normale Superiore	410.357	si	36.937
CNR; Area della Ricerca di Pisa	46.872	no	
Scuola Superiore Sant'Anna	39.610	si	
Biblioteca Universitaria: Polo SBN	118.641	si	694
Domus Galilæana	15.829	no	
Istituto Nazionale di Geofisica e Vulcanologia - Sezione di Pisa	142	si	142
Biblioteca della Soprintendenza per i Beni Architettonici e per il Paesaggio, per il Patrimonio storico, artistico ed etnoantropologico per le province di Pisa e Livorno	20.165	no	

Tabella 1: Biblioteche accademiche e di Ricerca

Biblioteche Ecclesiastiche

Biblioteca	N° record bibliografici	Classificazione Decimale Dewey	N° opere classificate
Biblioteca Arcivescovile "Cardinale Pietro Maffi" di Pisa	28.540	no	
Biblioteca Cathariniana di Pisa	20.960	si	143
Biblioteca del convento San Torpè	8.980	no	

Tabella 2: Biblioteche ecclesiastiche

Biblioteche locali

Biblioteca	N° record bibliografici	Classificazione Decimale Dewey	N° opere classificate
Biblioteca Comunale e Provinciale di Pisa	41.930	si	39.013
Biblioteche della Valdera, Basso Valdarno e Alta Valdicecina	317.397	si	tante
Biblioteca dei Ragazzi del Comune di Pisa	0	si	4560 (stimato)
OPAC della Biblioteca Comunale di San Miniato	54.761	si	50.000 (stimato)
Biblioteca del Circolo Agorà di Pisa	2.139	si	727 (stimato)
Biblioteca Babil - Rebeldìa di Pisa	511	si	511

Tabella 3: Biblioteche locali

Biblioteche scolastiche

Biblioteca	N° record bibliografici	Classificazione Decimale Dewey	N° opere classificate
Biblioteca del Liceo Classico "G. Galilei" di Pisa	554	no	
Biblioteca del Liceo Scientifico "U. Dini" di Pisa	3.642	si	3642
Biblioteca Istituto Comprensivo "L. Fibonacci" (Sede) - Pisa	3.331	si	2700 (stimato)
Biblioteca Istituto Tecnico Commerciale Pacinotti - Pisa	9.308	si	8903
Biblioteca IPSIA "A. Pacinotti" - Pontedera	1077	si	1070(stim)

Tabella 4: Biblioteche scolastiche

Biblioteche Specialistiche

Biblioteca	N° record bibliografici	Classificazione Decimale Dewey	N° opere classificate
Biblioteca Franco Serantini - Pisa	35.623	si	30.558
Biblioteca Domus Mazziniana - Pisa	14.460	no	
Biblioteca Istituzione Centro Nord-Sud della Provincia di Pisa	3.102	si	2.500 (stimato)
Biblioteca Associazione Casa della Donna - Pisa	3.606	No (collocazione in base DDC)	
L'Arsenale - Pisa: Cinema, Biblioteca e Centro di documentazione	33.669	no	
Biblioteca Istituto Lama Tzong Khapa di Pomaia - Pisa	3011	no	
Biblioteca Arcigay - Pisa	503	no	

Tabella 5: Biblioteche specialistiche

Ogni biblioteca gestisce e aggiorna in maniera autonoma il proprio catalogo e sceglie le informazioni che la scheda bibliografica deve contenere. All'interno di ogni catalogo emerge una compilazione non uniforme delle schede bibliografiche, in quanto non sempre queste contengono tutte le informazioni richieste (ad esempio la classificazione per argomento, come evidenziato dalle tabelle).

2.2. Il sistema di ricerca del Metaopac Pisano

Con il termine metaopac[3] pisano si intende il sistema di accesso ai cataloghi online delle biblioteche di area pisana.

L'interfaccia del metaopac presenta nella colonna di sinistra (Imm. 5) i campi per impostare i criteri di ricerca e le parole chiave da cercare. Nel metaopac pisano sono presenti tutti e tre i sistemi di ricerca visti in precedenza (v. 2.1): il *list,* il *search* e il *search* in ambiente testuale. Il primo metodo di ricerca, denominato “accesso agli indici da:”, è anche quello che all'interno dell'interfaccia del metaopac risulta meno visibile per la sua collocazione in basso a sinistra, sotto i campi da compilare per le ricerche di tipo *Search*; in un campo di testo si inserisce il termine chiave da cercare e il risultato

3 Metaopac deriva dall'unione dei termini meta e opac. Sul significato del secondo termine si veda 2.1.

sarà la lista dei titoli o degli autori, che contengono il termine ricercato, presenti in tutti i cataloghi del sistema. La ricerca *search* è sicuramente quella più utilizzata. Una volta compilati i campi con i termini da ricercare, combinandoli con gli appositi operatori booleani, i risultati della ricerca sono mostrati nella colonna di destra(Imm. 5); si osservi come venga riportato l'elenco di tutti i cataloghi con accanto il numero di record che corrisponde ai criteri di ricerca dell'utente.

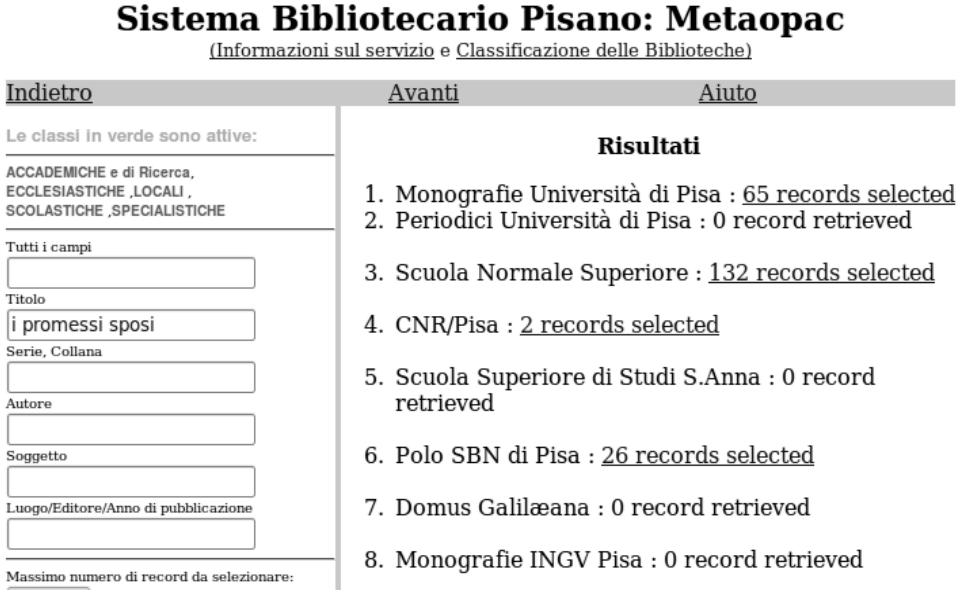


Immagine 5: Screenshot dei risultati di una ricerca fatta nel metaopac Pisano

Per vedere l'elenco delle opere bisogna consultare uno per uno i cataloghi per cui ci sono dei risultati (Imm. 5). Dopo aver scelto un catalogo, si apre una finestra con l'elenco dei record risultanti dalla ricerca (Imm.6). A questo punto si possono selezionare le opere e vedere le schede bibliografiche complete (Imm. 7). Questo sistema di consultazione è valido anche per le interfacce dei singoli cataloghi. Ogni catalogo infatti ha anche una propria pagina di interrogazione, che effettua le ricerche esclusivamente sul proprio catalogo, ma che presenta lo stesso tipo di campi e metodi di ricerca offerti dal metaopac.

Si noti come la scheda bibliografica offra la possibilità di continuare la ricerca con il metodo chiamato *surf*. Infatti il titolo dell'opera, l'autore, il Codice di Classificazione Dewey (DDC), sono dei link ad altre ricerche, e rispettivamente, se cliccati,restituiscono l'elenco delle schede con lo stesso titolo, l'elenco delle opere dello stesso autore e le opere con lo stesso DDC.

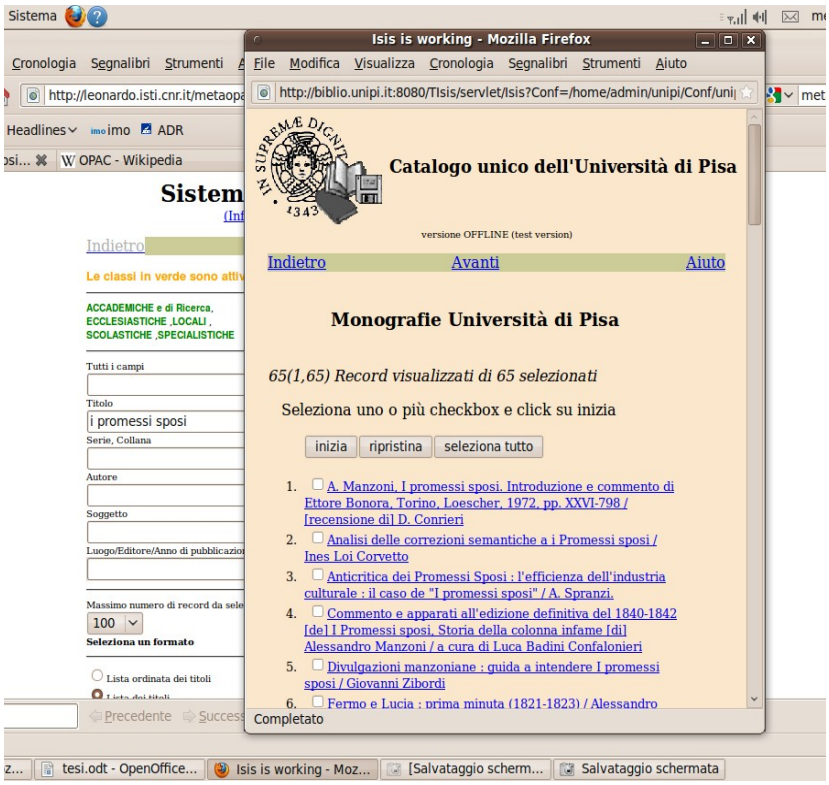


Immagine 6: Record selezionati per il catalogo dell'Università di Pisa

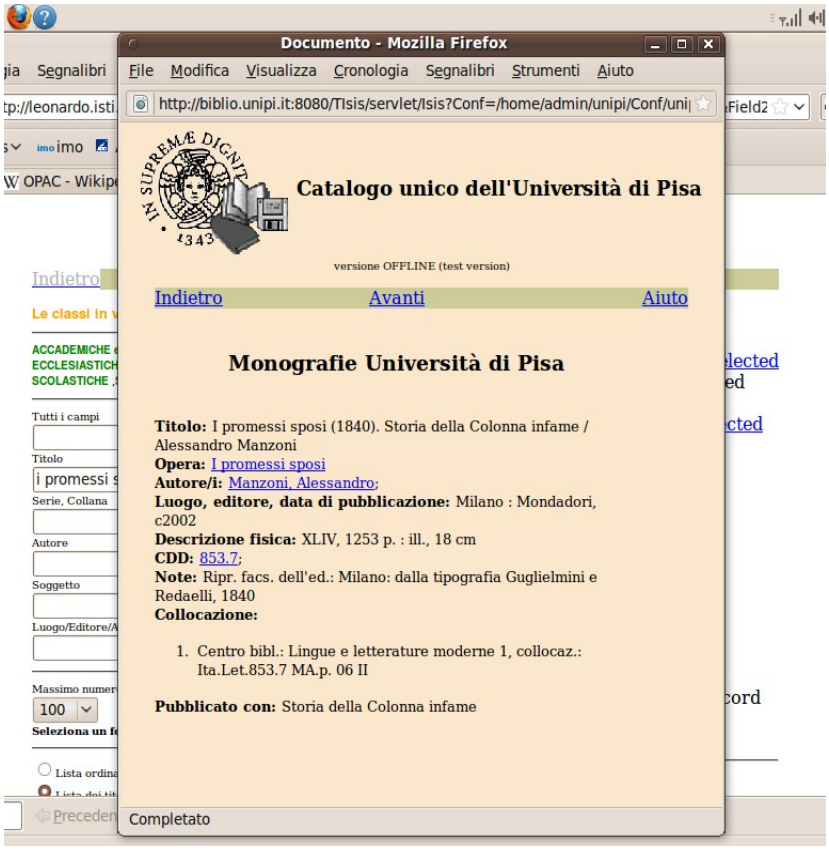


Immagine 7: Scheda bibliografica de "I Promessi Sposi" di Alessandro Manzoni

Per arrivare alla visualizzazione della scheda bibliografica dell'opera bisogna attraversare tre schermate, e questo rischia di rendere dispersiva la visualizzazione dei risultati, perché non si ha una visione d'insieme delle opere restituite dopo la ricerca; inoltre nella prima schermata (Imm. 6) nell'elenco dei cataloghi vengono inseriti anche quelli che non hanno record selezionati.

Dall'osservazione di questo difetto nella visualizzazione dei risultati della ricerca, si è pensato di proporre i record selezionati dal motore di ricerca in un modo più immediato, mostrando direttamente le opere e riportando per ognuna le biblioteche che la possiedono. Si è poi pensato di operare una modifica anche dal lato dei dati: anziché avere tanti cataloghi diversi su cui effettuare le ricerche e poi elaborare i risultati restituiti dai singoli cataloghi, non sarebbe meglio avere tutte le opere raccolte in un unico catalogo?

Da questa domanda ha preso inizio il progetto di realizzazione del catalogo unico[4], da interrogare con il motore di ricerca dell'interfaccia e che permetterà di visualizzare i risultati in una unica schermata.

2.3. Le schede bibliografiche: similarità e differenze

Le biblioteche del Meta Opac Pisano usano lo stesso tipo di scheda bibliografica, che prevede la compilazione dei campi presentati nella tab. 6.

La compilazione delle schede bibliografiche dovrebbe seguire delle convenzioni generali di catalogazione che permetterebbero di avere contenuti corretti e un catalogo coerente nel suo complesso. Tuttavia non sempre queste convenzioni vengono rispettate, e pur avendo tutte le biblioteche lo stesso tipo di scheda, i risultati della compilazione variano non solo da catalogo a catalogo, ma anche all'interno dello stesso da scheda a scheda, probabilmente in base al bibliotecario che ne ha curato la compilazione.

4 L'uso del termine “catalogo” potrebbe essere interpretato come improprio, in quanto si dovrebbe parlare di “repertorio”. Questa obiezione, per quanto corretta, sarà giustificata successivamente.

Titolo
Opera
Autore/i
Altri Autori
Luogo, editore, data di pubblicazione
Edizione
Descrizione fisica
Altri nomi
ISBN/ISSN
Serie
DDC
Soggetto/i
Collocazione
Tipo
Codice del Documento

Tabella 6: Esempio di scheda bibliografica

La cosa che emerge ad esempio è che non sempre le opere sono classificate in base all'argomento, anche se la scheda bibliografica prevede un apposito campo. Come si vede nelle tabb. 1-5, il numero di schede che riportano la classificazione decimale Dewey[5] (v. 2.4), che è lo standard di classificazione di documenti in base all'argomento tra i più diffusi e adottato da quasi tutte le biblioteche del Metaopac pisano, è spesso molto inferiore al numero delle opere del catalogo. A questo proposito si può osservare che spesso biblioteche più piccole come la Biblioteca dell'Istituto Tecnico Commerciale Pacinotti di Pisa hanno una maggiore attenzione alla classificazione rispetto a biblioteche più grandi, che forse hanno più difficoltà a controllare la catalogazione di un grande numero di opere.

L'analisi delle schede bibliografiche è importante in quanto il catalogo unico che verrà realizzato deriverà dall'unione di tutte le schede di incluse in tutti i cataloghi delle biblioteche del metaopac. L'operazione di unione delle schede comporta anche una

5 La *Dewey Decimal Classification,*in italiano Classificazione Decimale Dewey, è uno schema di classificazione numerica basata sull'argomento. Il primo livello di classificazione prevede 10 categorie che indicano argomenti generali, mentre dal secondo livello in poi la descrizione dell'argomento diventa più dettagliata. Per il funzionamento del sistema di classificazione Dewey si rimanda a http://it.wikipedia.org/wiki/Classificazione_decimale_Dewey

eliminazione di eventuali schede duplicate. Infatti potrà capitare che la stessa opera sia posseduta da più di una biblioteca, e in questo caso piuttosto che inserire più volte lo stesso tipo di informazione, si creerà per ogni opera un elenco di biblioteche che la possiedono. Quindi le schede bibliografiche del catalogo unico dovranno essere mantenute il più generiche possibile, dividendo le informazioni relative all'identità dell'opera, quindi titolo, autore/i, classificazione, luogo data e anno di pubblicazione, mentre per quelle che riguardano la collocazione, la descrizione fisica dell'opera e altre informazioni particolari si rimanda, con un collegamento (link), alle schede bibliografiche originali dei cataloghi di provenienza. Per fare questa operazione però è necessario confrontare fra di loro i dati delle schede bibliografiche per individuare quali opere siano uguali fra loro, e quindi debbano essere accorpate, e quali no. La descrizione finora fatta dei cataloghi delle biblioteche e delle relative schede può già fare intuire come questo sia stato un problema, perché il confronto di dati non riportati sempre secondo seguendo gli stessi criteri ha richiesto una serie di interventi correttivi che saranno descritti nei capitoli successivi.

Un altro problema rilevante è la mancanza di un campo che possa identificare in maniera univoca una determinata opera; il titolo da solo non può costituire un elemento identificativo, perché non sono rari i casi di libri con lo stesso titolo ma di contenuto diverso. Un altro aspetto non trascurabile riguarda la correttezza della compilazione dei vari campi della scheda bibliografica; errori di battitura, confusione nell'interpretare il significato dei vari campi, mancanza di regole comuni di scrittura fanno si che spesso anche opere uguali su schede bibliografiche diverse presentino delle differenze:

es: Catalogo Scuola Normale Superiore

Titolo: 1984 : a novel / by George Orwel **Autore/i:** Orwell, George; **Luogo, editore, data di pubblicazione:** New York : The New American Library, 1950 **Descrizione fisica:** 237 p., 18 cm	**Titolo:** 1984 / George Orwell **Autore/i:** Orwell, George; **Luogo, editore, data di pubblicazione:** New York : New American library, 1961 **Descrizione fisica:** 267 p., 19 cm

Tabella 7: Esempio di schede bibliografiche di "1986" di G. Orwell

Le due schede bibliografiche, appartenenti allo stesso catalogo, si riferiscono alla stessa opera, *1984* di George Orwell, ma presentano titoli parzialmente diversi: il primo titolo

è *1984: a novel*, il secondo *1984*; altre variazioni del titolo per questa opera, trovate su schede bibliografiche di altri cataloghi, sono state *nineteen eighty-four*, *nineteen eighty-four: a novel.* Per un utente che consulti il catalogo appare evidente che ci si sta riferendo allo stesso libro usando titoli scritti diversamente, ma per un software, che dovrebbe basare il confronto fra due titoli sull'uguaglianza delle lettere che li compongono, le opere non saranno uguali tra loro.

Questo aspetto è quindi da tenere in considerazione nel momento in cui si andranno a impostare i criteri di confronto fra le opere, e sarà trattato nel dettaglio nel capitolo 4.

2.4. La classificazione per argomento e il codice Dewey

Uno dei sistemi più utilizzati per la classificazione delle schede bibliografiche è quella per argomento. Molto spesso le stesse biblioteche organizzano la collocazione delle opere proprio in base a questo criterio. La classificazione per argomento però può non essere così intuitiva quanto si potrebbe inizialmente credere. Infatti potrebbe non essere semplice stabilire in maniera univoca e precisa l'argomento di un libro, inoltre ogni biblioteca potrebbe classificare gli argomenti in maniera diversa in base alla fantasia del catalogatore! Per questo generalmente le biblioteche si basano su sistemi di classificazione già redatti e ampiamente condivisi dalla comunità dei bibliotecari. Le biblioteche dell'area di Pisa non fanno eccezione, e quasi tutte si basano sulla classificazione decimale Dewey. Il campo “DDC” presente nelle schede bibliografiche indica proprio la Dewey decimal classification, in italiano classificazione decimale Dewey.

Questo sistema di classificazione assegna un codice numerico a ogni argomento, ha una struttura a livelli che partendo da argomenti molto generali permette di scendere sempre di più nel dettaglio. I primi 10 livelli della classificazione Dewey sono quelli illustrati nella tab. 8.

Per ognuno dei macro-argomenti (Codice Dewey da 100 a 900) esistono poi delle sottoclassi che individuano in maniera più dettagliata l'argomento, ad esempio per la classe 800, letteratura si avranno i sottocampi della tab. 9. Inoltre ogni sottoclasse ha dei sottoargomenti, per un totale di 10*10*10 = 1000 codici di classificazione a disposizione del bibliotecario.

000	Computer science, information & general works
100	Philosophy & psychology
200	Religion
300	Social sciences
400	Language
500	Science
600	Technology
700	Arts & recreation
800	Literature
900	History & geography

Tabella 8: Primo livello di Classificazione Decimale Dewey dalla OCLC Organization

800	Literature, rhetoric & criticism
810	American literature in English
820	English & Old English literatures
830	German & related literatures
840	French & related literatures
850	Italian, Romanian & related literatures
860	Spanish & Portuguese literatures
870	Latin & Italic literatures
880	Classical & modern Greek literatures
890	Other literatures

Tabella 9: Sottoargomenti per la classe 800 della classificazione Dewey

Il codice a tre cifre può avere estensioni per le correlazioni, luoghi, tempo, tipo di opera; con questi ulteriori codici, separati da un punto dalle prime tre cifre, la classificazione Dewey può raggiungere una dimensione di lunghezza indeterminata.

Si provi a classificare un'opera come “I Promessi Sposi”: questa rientrerà nella classe 800, Literature, sottoclasse 850 Italian literature, 853 Italian fiction, per cui il codice Dewey dei “Promessi Sposi” sarà 853.

Attualmente il codice Dewey è tradotto in 30 lingue, è usato in oltre 200.000 biblioteche in più di 135 paesi, e di questi oltre 60 lo usano per classificare la propria bibliografia nazionale[6]. Al 2003, su un campione di 152 cataloghi (corrispondente a circa il 20% degli opac esistenti in quel periodo) si contava che tra i sistemi che indicavano il sistema di classificazione adottato, 54 citavano la Dewey, 7 la CDU[7], 5 lo schema della Library of Congress e 6 schemi sviluppati localmente. Tra i 94 cataloghi che non indicano il sistema di classificazione adottato, molti presentano dati di classificazione, e spesso questi sono relativi alla Dewey; questo indica come spesso la scelta della classificazione Dewey sia data per scontata.

La gestione della classificazione nel metaopac pisano è quella illustrata dalle tabb. 1-5. Si vede come la percentuale di opere classificate si variabile, e come spesso biblioteche più piccole abbiano una maggiore percentuale di opere classificate. Dove è presente la classificazione, si specifica il sistema utilizzato, che in linea generale è la Dewey, anche se esistono pochi casi in cui sono seguiti altri modelli.

Un altro aspetto importante relativo all'uso della classificazione DDC nel metaopac pisano è che accanto ai codici numerici non è riportata una descrizione in linguaggio naturale, e quando presente è creata ad hoc dal bibliotecario, e non fa parte di una serie standardizzata di traduzioni dall'inglese all'italiano. Tutto ciò rende poco significativo per un utente che non abbia nozioni di classificazione Dewey il campo DDC, in quanto non potrebbe risalire da un numero alla descrizione dell'argomento.

La classificazione per argomento è un elemento molto importante per la correlazione delle opere sulla base del contenuto. Il fatto che non tutte le biblioteche adottino un sistema di classificazione è deleterio proprio per questo, perché si toglie la possibilità all'utente di effettuare ricerche semantiche.

6 Fonte: sito ufficiale della OCLC http://www.oclc.org.

7 Classificazione Decimale Universale, per approfondimenti vedi la relativa voce di Wikipedia all'indirizzo http://it.wikipedia.org/wiki/Classificazione_decimale_universale

3. Il catalogo unico delle biblioteche di area pisana

L'obiettivo di questo lavoro è la realizzazione di un catalogo unico delle opere delle biblioteche di area pisana. Partendo dalla situazione attuale, ovvero dall'analisi del metaopac pisano, della gestione delle ricerche e della visualizzazione dei risultati, si sono descritte le motivazioni che hanno fatto ritenere che avere tutti i dati raccolti in un unico catalogo potesse dare dei vantaggi dal punto di vista della consultazione da parte degli utenti e della ricerca sul lato dell'interfaccia.

Qui si descriveranno le caratteristiche che dovrà avere il catalogo unico.

3.1. Catalogo unico o repertorio?

Un catalogo bibliografico è l'elenco delle opere possedute da una biblioteca; ogni opera viene descritta da una serie di voci (titolo, autore, edizione ecc...) e vengono fornite le informazioni sulla collocazione dell'opera.

Il programma che verrà realizzato prenderà in input tutti i cataloghi presenti nel sistema bibliotecario di area pisana e restituirà un unico elenco, dove, come si è già detto, per ogni opera saranno indicati titolo, autore o autori, biblioteche di riferimento. Si è pensato che inserire anche le informazioni relative alla descrizione fisica dell'opera o la sua collocazione non servisse allo scopo del progetto, per due motivi: il primo che trattandosi di un catalogo unico relativo a più biblioteche, la collocazione e la descrizione fisica dell'opera variano da catalogo a catalogo, quindi si sarebbero dovute copiare quasi per intero le schede bibliografiche originali, compromettendo l'obiettivo iniziale di avere un catalogo compatto e omogeneo; il secondo perché il catalogo unico non deve sostituire quelli specifici di ogni biblioteca, ma deve costituirne una sintesi. Per questo il riferimento alle schede bibliografiche di origine rimane un punto fondamentale. Quando il progetto di realizzare questa opera unica ha iniziato a concretizzarsi, si preferiva chiamare l'oggetto nato dalla sintesi di tutti i cataloghi “repertorio”, termine che rende più l'idea di un elenco generale e che secondo alcuni sarebbe stato più appropriato. Il termine repertorio però fa perdere il collegamento con l'idea di biblioteca, ambiente che invece il termine catalogo evoca immediatamente. Quindi si è scelto di chiamare il risultato di questo lavoro di commistione “catalogo”,

magari sacrificando la correttezza a favore di una maggiore immediatezza del significato.

3.2. Compilare il catalogo unico: la gestione dei dati duplicati

I tipi di opere che si possono incontrare mentre si compila il repertorio possono essere:

1. Opere uniche, possedute da una sola biblioteca
2. Opere uguali, per titolo e autore e pubblicazione, possedute da più biblioteche
3. Opere simili, con stesso titolo e autore ma diversa pubblicazione, possedute da più biblioteche.

Nel caso delle opere uniche il lavoro è semplice, in quanto il catalogo conterrà un solo record bibliografico per quella determinata opera, ma come gestire le opere uguali(2) o simili(3)?

Il catalogo, come si è detto, dovrebbe riportare le opere uguali una sola volta, cercando quindi di evitare ripetizioni di opere già inserite nell'elenco. La disomogeneità dei dati di partenza però rende difficile l'effettiva individuazione delle opere ripetute, specialmente se si deve affidare questo compito a un software.

Inoltre, come si è già detto, si sente anche la mancanza di un identificatore univoco per ogni libro, in quanto il solo titolo è troppo soggetto ad errori e variazioni, altri campi non sempre sono presenti in tutti le schede e comunque spessosi riferisce alla copia fisica dell'opera, come ad esempio l'ISBN; un modo per potere identificare un opera in maniera quasi univoca può essere la coppia autore-titolo, anche qui considerando le possibili variazioni dovute a errori di inserimento.

Bisognerebbe intanto definire quando due opere possono dirsi uguali. Possono bastare solo lo stesso titolo e lo stesso autore o bisogna valutare anche altri elementi? Per affrontare questo problema si è partiti da delle osservazioni di tipo pratico: opere classiche, come i “Promessi Sposi”, o la “Divina Commedia”, possono essere edite da diverse case editrici. Tuttavia, il contenuto delle opere, e ciò vale tanto per il romanzo quanto per il poema, rimarrà uguale nelle diverse versioni (magari con qualche eccezione dovuta a diverse ricostruzioni filologiche). Potranno cambiare gli apparati critici, le illustrazioni, l'impaginazione e altri aspetti che si potrebbero definire

“marginali” rispetto al contenuto vero e proprio dell'opera. Per cui nel caso in cui su diverse schede bibliografiche compaia la stessa opera in edizione diverse, alla luce di quanto affermato prima, le due schede dovrebbero confluire in una sola. A questa soluzione si potrebbe però obiettare che spesso un utente va alla ricerca di una particolare edizione. Proprio tenendo conto di questa considerazione, si è scelto di mantenere le informazioni sulla pubblicazione e di fornirle all'utente nel momento in cui visualizza i risultati della sua ricerca. Per ottenere il massimo risultato di sintesi delle informazioni ripetute ma allo stesso tempo la minor perdita di precisione possibile, il confronto delle opere avverrà sui campi più “generici” titolo e autore, per verificare se le opere confrontate siano uguali o no, e successivamente in fase di creazione della scheda bibliografica verranno memorizzati i dati su cui si è fatto il confronto, e poi per ogni biblioteca che possiede l'opera le informazioni sulla pubblicazione, in modo che l'utente al momento della scelta abbia la possibilità di considerare anche questa informazione.

Altro problema da gestire è la possibilità di avere titoli che indicano la stessa opera ma con errori che dal punto di vista grafico li rendono diversi. All'occhio umano questi errori appaiono molto evidenti, e sarebbe facile per un operatore effettuare i confronti necessari, però il lavoro di creazione del catalogo unico deve essere svolto da un software, e questo complica le cose; bisognerà fornire al software una serie di istruzioni che gli permettano di valutare anche possibili variazioni di un titolo.

Ovviamente solo il confronto dei titoli non basta a verificare se due opere siano uguali o meno, bisogna trovare anche altri parametri su cui basare il confronto. Sicuramente andranno considerati gli autori, ma si è ritenuto utile restringere ancora la cerchia delle opere da confrontare effettuando dei raggruppamenti.

3.3. Raggruppamento delle opere

Ogni catalogo del metaopac Pisano raccoglie un grande numero di record. Tra i cataloghi più ricchi, quello dell'università di Pisa conta più di 800.000 opere, mentre la Scuola Normale Superiore ne contiene 370.318. Ma anche realtà più piccole come l'Istituto Tecnico Commerciale "A. Pacinotti", con le sue 8.636 voci, hanno un loro peso nel conto generale delle opere.

Lavorare direttamente sull'unione di tutti i cataloghi, comporterebbe la gestione di una

enorme mole di dati (almeno un paio di milioni di opere) disordinata e difficile da gestire. Per questo si è pensato di organizzare i dati dei cataloghi in gruppi basati su elementi comuni a tutte le opere. I sistemi per raggruppare i dati sarebbero potuti essere diversi, dall'ordine alfabetico per titolo o per autore all'anno di pubblicazione, ma questi criteri sono sembrati poco significativi; piuttosto si è cercato un sistema che riuscisse a ottimizzare il lavoro di selezione delle opere, creando dei gruppi che contenessero opere tra loro affini non solo dal punto di vista strettamente ortografico (abbiamo visto come a volte per una stessa opera possano esistere diversi titoli, anche dovuti a errori di digitazione) , ma anche contenutistico. Per potere effettuare questi raggruppamenti si è scelto di seguire il campo DDC della scheda bibliografica, che indica il codice di classificazione Dewey[8].

La classificazione Dewey permette di creare 10 gruppi di argomenti su cui distribuire le varie opere. Nell'esempio proposto (vedi tab. 7), su *1986* di Orwell, a fronte delle varianti del titolo le due schede bibliografiche avrebbero avuto lo stesso codice DDC: 820, che avrebbe permesso quantomeno di capire che, anche in presenza di titoli leggermente diversi, si tratta di opere collegate in quanto relative allo stesso argomento.

Raggruppare i titoli in base alla classificazione Dewey è un vantaggio per tutte quelle schede bibliografiche che riportano il codice DDC, ma ciò comporta l'esclusione dal raggruppamento di tutte le opere che non sono state classificate (che nel Metaopac Pisano costituiscono una buona percentuale).

Ciò nonostante si è deciso di seguire questa strada per l'unione dei cataloghi, per i vantaggi che offre: il codice DDC infatti permette di raggruppare opere realmente collegate tra loro dal punto di vista contenutistico, facendo si che solo all'interno di uno stesso gruppo possano esserci opere ripetute in quanto relative allo stesso argomento, mentre trovare la stessa opera in gruppi di argomenti diversi è altamente improbabile. Inoltre questo sistema di classificazione è regolato da precise regole controllate dalla OCLC[9] e quindi non è soggetto ad interpretazione da parte del bibliotecario e, infine, questo metodo di raggruppamento permette di creare una procedura di unione di

8 V. cap. 2.4 per un approfondimento della Classificazione Decimale Dewey

9 La OCLC, Online Computer Library Center (http://www.oclc.org) è una organizzazione mondiale al servizio delle biblioteche. La OCLC ha acquisito la proprietà ed i diritti associati al Dewey Decimal System nel 1988 e mantiene aggiornato il sistema di classificazione.

cataloghi estendibile ad altre biblioteche che si basino su questa classificazione, in quanto non sarebbe creato ad hoc per le specificità dei cataloghi del metaopac Pisano. Le schede bibliografiche che non hanno la classificazione Dewey potranno essere integrate in una fase successiva, quando con le opere classificate si sarà già popolato il repertorio su una base di partenza più solida. Inoltre alla luce degli studi presentati nel cap. 1, si è voluto impostare il lavoro per valorizzarlo dal punto di vista semantico. L'organizzazione dei dati basata proprio sulla classificazione per argomento permette di creare le premesse per una consultazione del catalogo di tipo semantico. Il codice DDC, opportunamente esplicitato con una descrizione in linguaggio naturale, permetterà agli utenti di accedere al catalogo navigando tra le classi della classificazione Dewey.

3.4. Conclusioni sui dati di partenza

I dati di partenza offerti dai cataloghi delle biblioteche dell'area di Pisa costituiscono una vera sfida per chi voglia trovare dei criteri comuni di compilazione delle schede bibliografiche. Le vie per cercare di accomunare le opere sono molteplici, e la scelta di una soluzione può volere dire sacrificare dei dati.

Riuscire a mantenere la maggiore quantità di informazioni possibile è uno dei punti fermi della progettazione del programma che si occuperà di realizzare il catalogo. Infatti lo scarto di quanto non conforme a determinate regole potrebbe significare dimezzare la quantità dei dati di partenza, e il catalogo unico perderebbe così gran parte del suo valore. D'altra parte cercare di salvare proprio tutto potrebbe significare portare il disordine dei singoli cataloghi dentro il catalogo unico. La via intermedia non è stata semplice da trovare, la scelta di raggruppare le opere in base alla classificazione Dewey già comporta l'esclusione in una prima fase di una buona quantità di opere. Parte di queste potrà essere inserita in un secondo momento,ma parte rimarrà esclusa dal catalogo. Però questa scelta di seguire la classificazione Dewey, se da un lato come si è detto comporterà l'esclusione di alcune informazioni, dall'altro conferirà un valore aggiunto al catalogo unico, dandogli quell'orientamento di tipo semantico tanto ricercato oggi negli opac italiani. L'utente che consulterà questo repertorio verrà avvisato dell'incompletezza dei dati, ma di contro gli verrà data la possibilità di costruire le proprie ricerche sulla base dei contenuti delle opere, essendo sicuro che la risposta alle sue richieste sarà completa ed esaustiva.

Trattandosi di un prodotto aggiuntivo al già esistente metaopac pisano,che comunque rimarrà attivo per le ricerche sui cataloghi delle singole biblioteche, si è ritenuto che il catalogo unico dovesse essere una sintesi dei cataloghi delle biblioteche basato sulla qualità dei dati, piuttosto che sulla quantità. Per la consultazione dei singoli cataloghi infatti, qualora i risultati della ricerca sul nuovo catalogo unico non fossero soddisfacenti, rimarrebbe sempre il metaopac, mentre il catalogo unico offrirà un accesso alle informazioni di tipo diverso e più selettivo, proponendo le schede bibliografiche più complete, e diventando un possibile strumento di condivisione delle informazioni bibliografiche del sistema bibliotecario di area pisana.

L'esclusione di alcuni record bibliografici potrà essere da incentivo per i bibliotecari a curare la correzione dei dati presenti nei propri cataloghi e a prestare attenzione all'inserimento di nuovi dati. Perché quanto non si potrà correggere in corso d'opera nella realizzazione del catalogo dovrà essere corretto nei dati di partenza. La correttezza dei dati di partenza, che si sarebbe dovuta dare per scontata, è un requisito importante quando si inizia un lavoro del genere, per cui è giusto che chi presenta i propri dati in maniera corretta possa beneficiare dei vantaggi che potrebbe portare il catalogo.

4. Verso il catalogo unico: la fase di progettazione

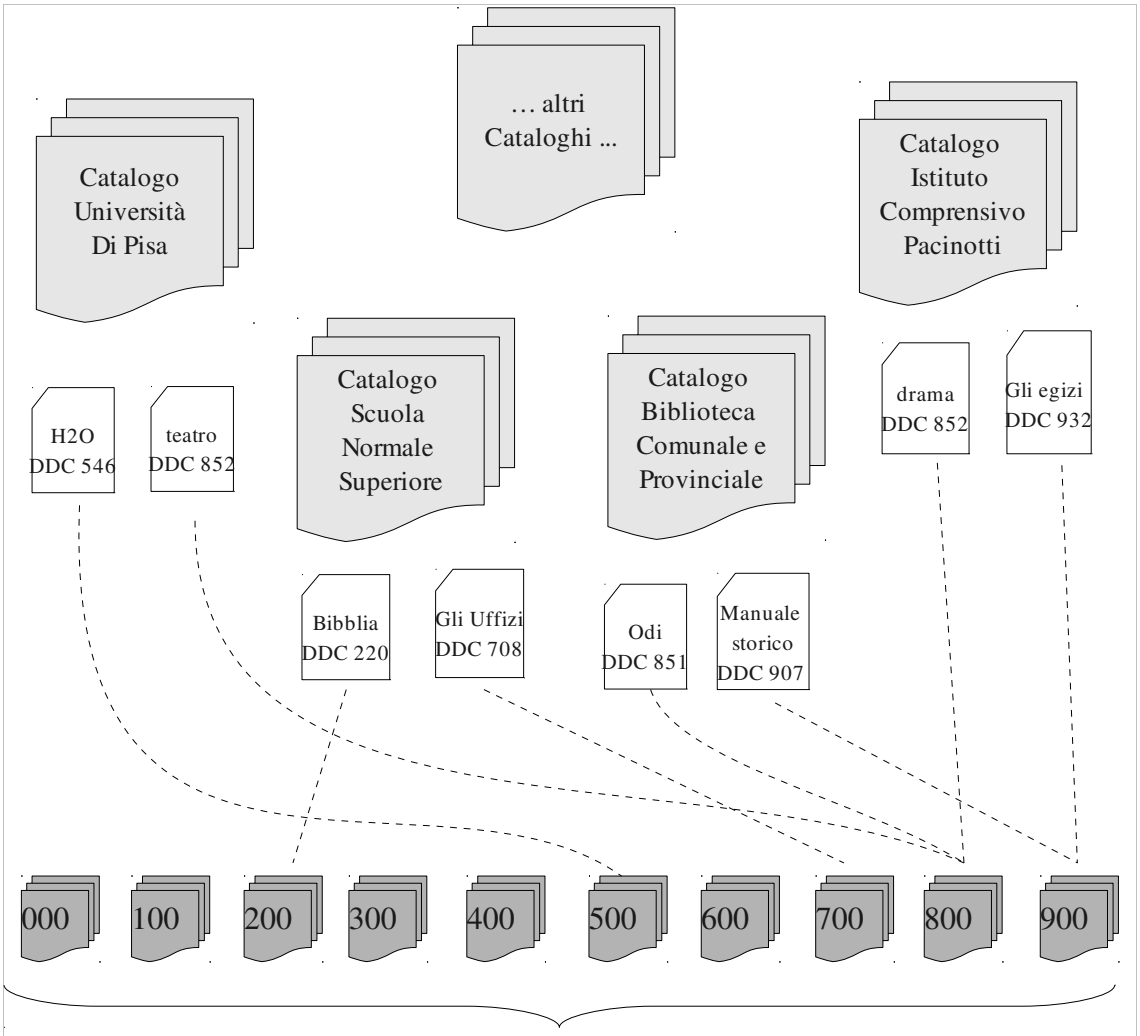
La realizzazione del catalogo unico avviene tramite una procedura automatizzata, gestita da un programma realizzato per questo scopo. Rendere eseguibile da un software delle operazioni che richiedono l'interpretazione umana non è semplice, e quindi occorre una precisa progettazione di ogni fase della realizzazione del programma dell'analisi dei dati di partenza ai risultati finali. La situazione dei dati di input è già stata delineata nei capitoli precedenti, ora si inizierà ad entrare nel merito della procedura che porterà da tanti cataloghi a uno solo.

4.1. La gestione dei dati di input

I dati di input sono costituiti da un catalogo per biblioteca. I cataloghi che vengono elaborati dal sistema sono in formato xml e contengono solo le informazioni che il catalogo unico deve contenere, quindi titolo, autore/i, casa editrice, luogo e data di pubblicazione, argomento (codice Dewey).

Il programma partendo da questi file xml dovrà produrre un unico catalogo da memorizzare in un database in modo che sia consultabile dall'interfaccia utente.

Come si è già detto in 3.3, per evitare di dovere gestire contemporaneamente i dati di tutte le schede bibliografiche, si è deciso di raggrupparle in base alla classificazione Dewey. Quindi il primo passo della procedura sarà quello di scorrere tutti i cataloghi e smistare le opere ognuna nel proprio gruppo in base al primo livello della classificazione Dewey (il primo numero del codice, v. Ill. 1).



Insiemi di schede bibliografiche divise per argomento in base al primo livello di classificazione Dewey

Illustrazione 1: Esempio di distribuzione delle schede bibliografiche in base al codice DDC

Come si vede dall'Ill. 1, ogni scheda bibliografica viene inviata al gruppo di appartenenza in base alla prima cifra del suo codice, che rappresenta la classificazione Dewey di primo livello.

Questa prima operazione di selezione permette di lavorare su un gruppo di schede bibliografiche alla volta.

All'interno di ogni gruppo verranno selezionate le opere da inserire nel catalogo unico. La selezione servirà solo a evitare la duplicazione delle schede che descrivono opere uguali. Gli elementi che possono distinguere o accomunare le opere sono due, titolo e autore (o autori). Tra i due elementi, si è osservato che il titolo è quello più soggetto a

errori, quindi il primo confronto fra le schede bibliografiche avverrà per autore. Nel caso in cui due opere abbiano lo stesso autore, si passerà al confronto fra i titoli, operazione che potrebbe richiedere degli interventi di revisione linguistica.

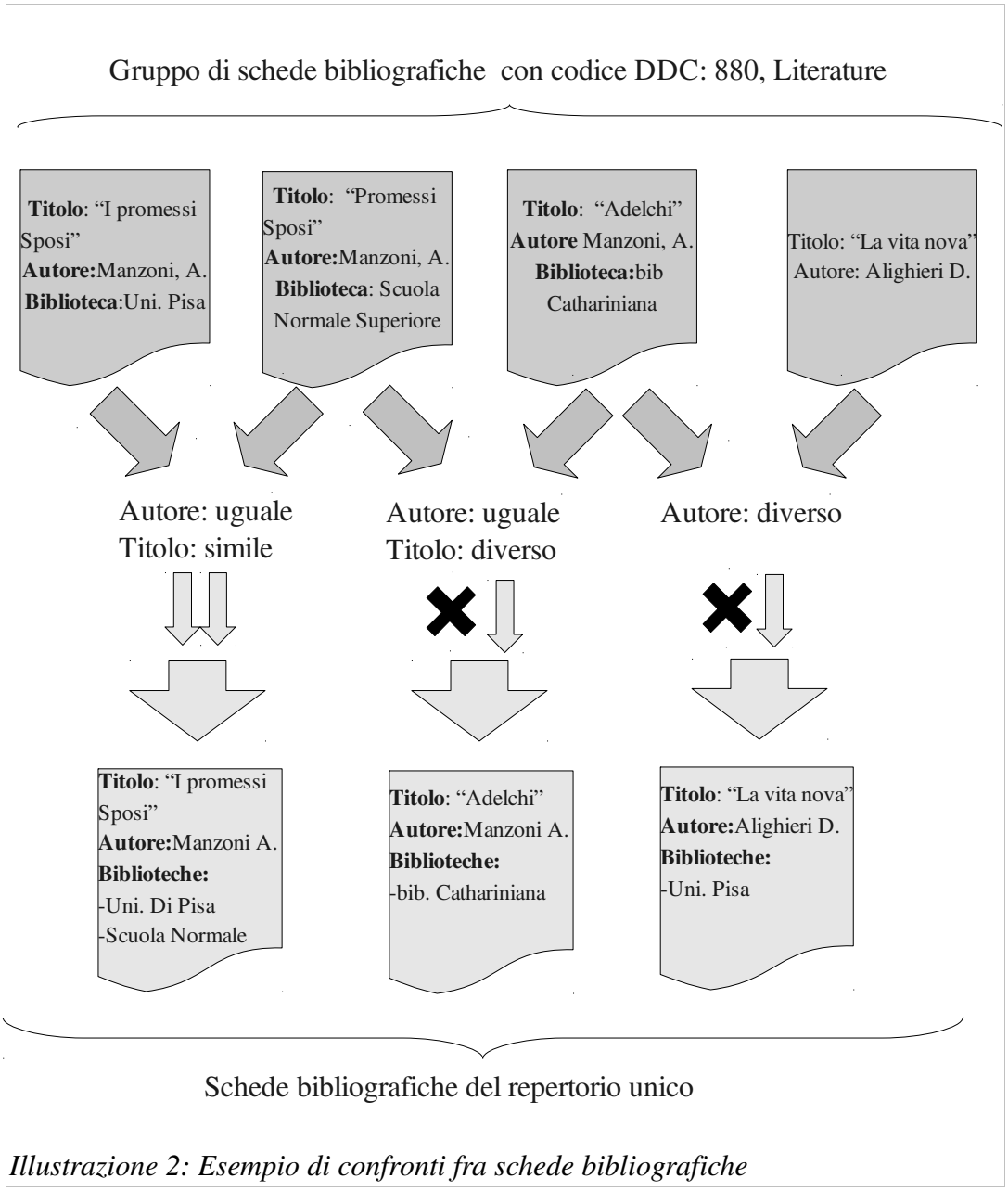


Illustrazione 2: Esempio di confronti fra schede bibliografiche

L'Ill. 2 mostra come avverrà il confronto fra le schede. Quando due opere hanno lo stesso autore (Ill. 2, 1° e 2° confronto da sinistra) viene controllato anche il titolo. Quest'ultimo potrà essere uguale, quando i titoli saranno scritti esattamente nello stesso modo, simili (1ª e 2ª scheda bibliografica) se avranno poche differenze tipografiche, o diversi se saranno scritti in maniera totalmente diversa.

I concetti di similarità e diversità dei titoli sono molto relativi in realtà; si potrebbe

definire il “grado” di similarità tra due titoli come il numero di cambiamenti da apportare a un titolo perché diventi uguale a un altro. A questo punto tra “Promessi sposi” e “I promessi sposi” ci sarebbero da fare due cambiamenti: l'aggiunta dell'articolo “I” e lo spazio tra l'articolo e la parola. Se si stabilisce una soglia al numero di cambiamenti possibili, si potrà dire che se servono più di un certo numero di modifiche i titoli saranno diversi, altrimenti saranno uguali e si ipotizzerà un errore nella digitazione di uno dei due. Questo sistema ha il vantaggio di poter essere applicato anche da un programma automatico. Il confronto fra i titoli comunque avverrà solo dopo che si sarà verificata l'uguaglianza degli autori (Illustrazione 2, 3° confronto da sinistra), così da evitare passaggi inutili. Nel caso in cui due opere siano riconosciute come uguali, avranno un'unica scheda bibliografica di riferimento nel catalogo unico, nella quale sarà riportato l'elenco di tutte le biblioteche che effettivamente possiedono l'opera. Una volta che il catalogo sarà terminato, i dati delle opere saranno organizzati come mostrato nell'Ill. 3.

Illustrazione 3: Esempio di scheda bibliografica del catalogo unico relativa a "I promessi sposi"

Come si vede dall'illustrazione 3, i dati di partenza andranno a confluire nelle schede bibliografiche del catalogo con questa struttura: il titolo e l'autore riportati una sola volta, il codice DDC dell'argomento e l'elenco delle biblioteche che possiedono l'opera con le informazioni sull'edizione che possiedono, in modo da dare all'utente tutte le

informazioni che potrebbero servirgli per scegliere presso quale biblioteca consultare l'opera.

4.2. L'output, come il catalogo viene consultato

Dopo che il catalogo sarà stato creato, occorrerà che ci sia un sistema in grado di accedere ai dati e di effettuare ricerche su questi ultimi.

L'analisi dell'interfaccia del metaopac pisano ha rivelato che il sistema ha bisogno di correzioni, inoltre, alla luce delle ricerche fatte sugli opac semantici (v. 1.2), si è scelto di dare un orientamento appunto semantico alle possibilità di ricerca offerte dalla nuova interfaccia di consultazione.

A proposito delle tecniche di interrogazione dei cataloghi online, l'Associazione Italiana Bibliotecari ha diffuso un carteggio avente per oggetto quali sistemi di ricerca all'interno di un catalogo siano preferibili[10]. Le osservazioni effettuate riguardano più che altro la scelta compiuta dall'utente quando si trova a dover scegliere tra diversi sistemi di ricerca. Quello che emerge dal carteggio fra diversi esperti del settore bibliotecario è che:

- Una volta che un utente ha iniziato a usare uno strumento di ricerca, sarà portato a continuare a usare lo stesso metodo per il futuro, per una sorta di abitudine acquisita.
- Spesso la scelta di un metodo di ricerca piuttosto che un altro non si basa su una effettiva riflessione o consapevolezza degli strumenti che si hanno a disposizione, ma da quale soluzione viene proposta per prima; la scelta è quasi casuale.
- Le preferenze degli utenti tra *search, search* testuale e *list,* dipendono dal tipo di utente, premesso comunque che egli abbia sperimentato tutti e tre i metodi di ricerca. Gli utenti più assidui sembrano preferire la navigazione per liste, mentre quelli occasionali la ricerca a testo libero stile *Google*.

Quindi sicuramente sarà utile presentare tutte e tre le modalità di ricerca, cercando di

10 Da AIB-WEB. Contributi, *Presentazione del catalogo e tecnica d'interrogazione*, Maurizio Di Girolamo e al.

dare a tutte la stessa visibilità per permettere all'utente di effettuare una scelta il meno casuale possibile.

Occorrerà poi impostare i campi della ricerca in modo da permettere quell'approccio di tipo semantico di cui si è già parlato, e a questo proposito si tornerà a fare riferimento a quello studio svolto dall'Università di Venezia, che ha preso in esame, valutandone pro e contro, diverse interfacce di interrogazione di cataloghi. Tra gli elementi messi in evidenza ci sono:

1. mancanza di informazioni sulla quantità di dati classificata per argomento e sistema di classificazione usato.
2. Mancanza degli equivalenti verbali ai codici numerici che indicano la classificazione per argomento.
3. La ricerca per soggetti spesso non avviene basandosi su un vocabolario controllato, ma prendendo le parole da qualsiasi campo.
4. È rara la ricerca per termine e stringa, ovvero dove alla ricerca per soggetto non viene restituita la lista delle schede bibliografiche che contengono il termine, ma un elenco di stringhe che restituiscono le frasi in cui il termine ricercato è contenuto.
5. La ricerca per classe è disponibile in meno casi rispetto a quella per soggetto, e comunque compare di solito nella ricerca avanzata, non essendo considerato un campo di ricerca base. Tuttavia la ricerca per classe, di solito inserita a uso dei bibliotecari che hanno una già una conoscenza dei sistemi di classificazione, non è considerata uno strumento di ricerca efficace, anche se, gestita correttamente, offrirebbe delle grandissime potenzialità.
6. Le schede di risposta a una ricerca contengono di solito solo i dati descrittivi, le informazioni di tipo semantico sono riservate a una visualizzazione dettagliata della scheda.

L'interfaccia di consultazione del catalogo unico di area pisana cercherà, per quanto concesso dai dati a disposizione, di soddisfare i requisiti prospettati dai precedenti punti, o quantomeno di giustificare i casi in cui non sarà possibile seguire queste linee guida e cercare delle soluzioni alternative.

Per quanto riguarda le informazioni sui dati classificati presenti nel metaopac (punto 1), già in questo scritto sono stati forniti i dati relativi alla quantità di opere classificate per ogni catalogo e lo schema di classificazione usato, quindi queste informazioni saranno riportate anche in una apposita area dell'interfaccia.

Il secondo punto (2) merita una riflessione a parte in quanto la questione dell'esplicitazione dei codici Dewey non è affatto banale. Il codice originale è in inglese e la traduzione italiana è disponibile in formato cartaceo o in pdf, quindi per potere essere usata occorre una trascrizione del codice in una struttura consultabile attraverso l'interfaccia del catalogo. L'OCLC mette a disposizione degli strumenti online molto utili per ricavare il significato del codice Dewey, ma purtroppo ancora l'Italiano non è tra le lingue disponibili. Quindi per quanto riguarda l'interfaccia del catalogo unico, verrà fatta una traduzione ad hoc della classificazione Dewey in modo da poter rendere disponibile un campo di ricerca in italiano. Però verranno sviluppati anche dei sistemi di comunicazione con i servizi web della Dewey inglese, primo per permettere agli utenti interessati di interagire con questo strumento, per le cui funzioni si rimanda al paragrafo 4.3, e poi per predisporre l'interfaccia all'interazione con il sistema per quando sarà disponibile anche in italiano.

La possibilità di effettuare ricerche per soggetto è una opzione già offerta dal metaopac. L'elenco dei soggetti unici del metaopac conta 440.577 voci, considerando che sommando le opere di tutti i cataloghi si arriva a oltre due milioni di opere, il numero di termini che compongono il soggettario è relativamente piccolo, anche perché alcuni soggetti accomunano più opere, altri solo una. Inoltre per poter strutturare in maniera coerente la ricerca per soggetto, bisognerebbe formare un vocabolario di riferimento per i soggetti partendo dai termini usati nelle varie schede bibliografiche, quindi eliminando termini ripetuti, raggruppando sotto un'unica voce i soggetti simili. Queste operazioni, non previste in questa fase della realizzazione dell'interfaccia del catalogo unico, possono diventare uno spunto per lo sviluppo futuro del programma e dell'interfaccia. Per cui i punti 3 e 4 saranno oggetto di sviluppi successivi.

Per come è strutturato il catalogo unico, si potrà invece puntare sulla ricerca per classe, fornendo due diversi accessi ai cataloghi: uno di tipo *list*, permettendo all'utente di navigare tra i diversi rami dell'albero della classificare Dewey, cercando le opere di

argomento più generico o specifico; il secondo accesso invece di tipo *search*, con la possibilità di cercare sia il codice numerico sia termini in linguaggio naturale.

Quindi il progetto dell'interfaccia cercherà di andare incontro a diversi tipi di esigenza di ricerca e a diversi tipi di utenti, fornendo tutti gli strumenti affinché l'esplorazione del catalogo vada a buon fine e offra risultati soddisfacenti.

4.3. Uno strumento online per l'esplicitazione dei codici di classificazione Dewey

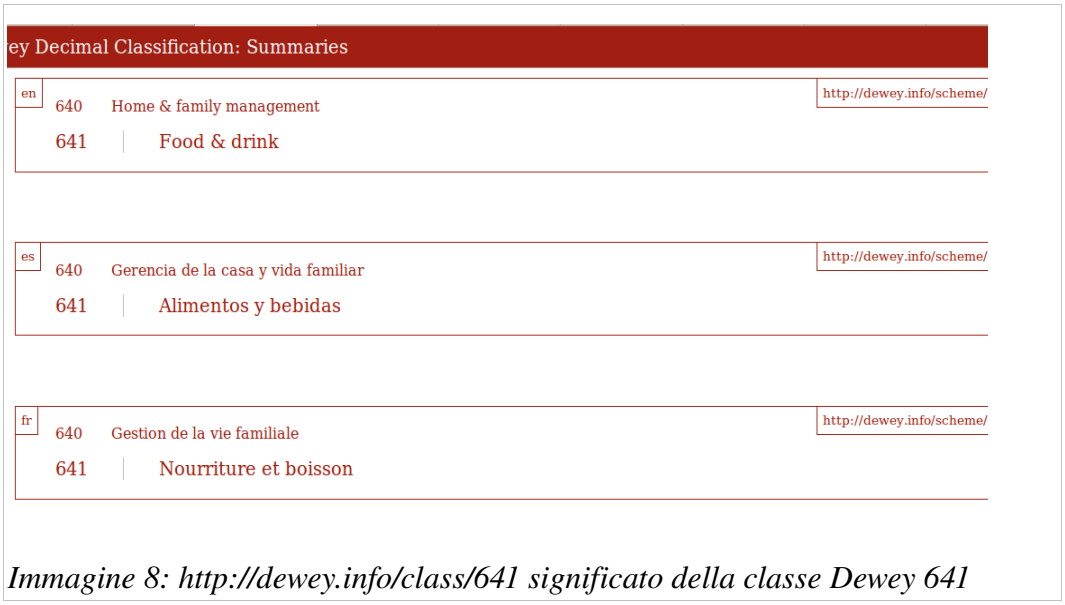


Immagine 8: http://dewey.info/class/641 significato della classe Dewey 641

I cataloghi del metaopac raccolti nel catalogo unico seguono tutti la classificazione Dewey. Gli equivalenti verbali dei codici non sono riportati dalle schede bibliografiche di partenza, se non in pochi casi nei quali non sono nemmeno rari gli errori; per questo si è preferito eliminare ogni forma di commento verbale e prendere dalle schede solo il codice numerico. L'OCLC (vedi nota 6) fornisce un sistema online per ottenere gli equivalenti verbali dei codici all'indirizzo http://dewey.info/. Il servizio messo a disposizione a questo indirizzo è tutto basato sulla costruzione di un url che permette di navigare tra i diversi livelli della classificazione Dewey. Ad esempio, se si volesse conoscere l'equivalente del codice 641 basterebbe aggiungere all'indirizzo http://dewey.info/class/ il numero di classe di cui si vuole cercare il significato, quindi nel nostro esempio 641. L'indirizzo http://dewey.info/class/641 restituisce una pagina con il significato della classe in diverse lingue (Imm. 8) e la classe superiore di derivazione.

L'indirizzo per accedere ai servizi sulla classificazione Dewey può essere ulteriormente specificato, ad esempio è possibile ottenere la descrizione della classe in formato SKOS riscrivendo l'indirizzo di prima con una l'ulteriore aggiunta di about.rdf http://dewey.info/class/641/about.rdf (Imm. 9)

Immagine 9: Descrizione in formato SKOS della classe Dewey 641

Il sistema permette anche di scegliere la lingua aggiungendo sempre nell'url l'apposita indicazione, ad esempio en per l'inglese, fr per il francese e così via. Le lingue disponibili sono inglese, francese, spagnolo, portoghese, russo, arabo, tedesco, africano, norvegese, svedese, cinese.

Questo strumento, specialmente nel suo formato SKOS, permetterebbe di cercare gli equivalenti verbali dei codici Dewey e di memorizzarli in una struttura dati locale, in modo da renderli accessibili anche se il sito di riferimento dovesse essere non disponibile.

Il riferimento diretto alla traduzione ufficiale della OCLC permetterebbe alle biblioteche di adottare uno standard nella trascrizione in linguaggio naturale dei codici, così da offrire all'utente sempre lo stesso tipo di informazione. Purtroppo la della versione italiana di questo strumento obbliga ad effettuare una traduzione in italiano valida solo per il metaopac pisano, tuttavia verranno forniti lo stesso dei rimandi alla versione inglese di *dewey.info* proprio per abituare l'utente a consultare questo strumento e per predisporre l'interfaccia nel caso in cui in futuro il servizio fosse disponibile anche in italiano.

5. La realizzazione del software

Il software che implementa la creazione del catalogo unico è stato interamente progettato e realizzato per questa applicazione, tuttavia si è cercato di organizzarlo nel modo più generico possibile, in modo da renderlo applicabile ad altri sistemi bibliotecari.

Il lavoro svolto dal software si può dividere in 3 fasi: organizzazione dell'input, elaborazione dei dati, restituzione dell'output.

Oltre a queste 3 fasi poi ci sono da considerare delle operazioni preliminari, che consistono in una procedura di pulizia dei dati di input e nella creazione delle strutture dati che fanno da supporto al programma.

Il software è stato realizzato in Java e si basa su un database mySql.

5.1. La costruzione del database

Il software memorizza tutti i dati su cui lavora, sia quelli temporanei che quelli definitivi, su un database. La scelta di usare delle tabelle di appoggio per memorizzare i dati ancora in fase di elaborazione è stata fatta considerando la quantità di informazioni che il programma deve gestire. Infatti tenere i dati temporanei in memoria sarebbe stato un peso non indifferente per il computer che esegue il programma, mentre invece averli su una struttura dati che può essere consultata abbastanza rapidamente è sembrato un buon guadagno in termini di efficienza. Il database su cui si appoggia il software è formato da 17 tabelle, 11 temporanee e 6 per il risultato finale dell'elaborazione. La struttura del database con un modello ER è rappresentata nelle illustrazioni 4 e 5. Nella Ill. 4 si vede la struttura di quella porzione di database che contiene le tabelle temporanee, mente in Ill. 5 la struttura del database definitivo che conterrà il catalogo unico. La tabella autori, presentata in entrambe le figure, è sempre la stessa tabella che viene usata in entrambi gli schemi.

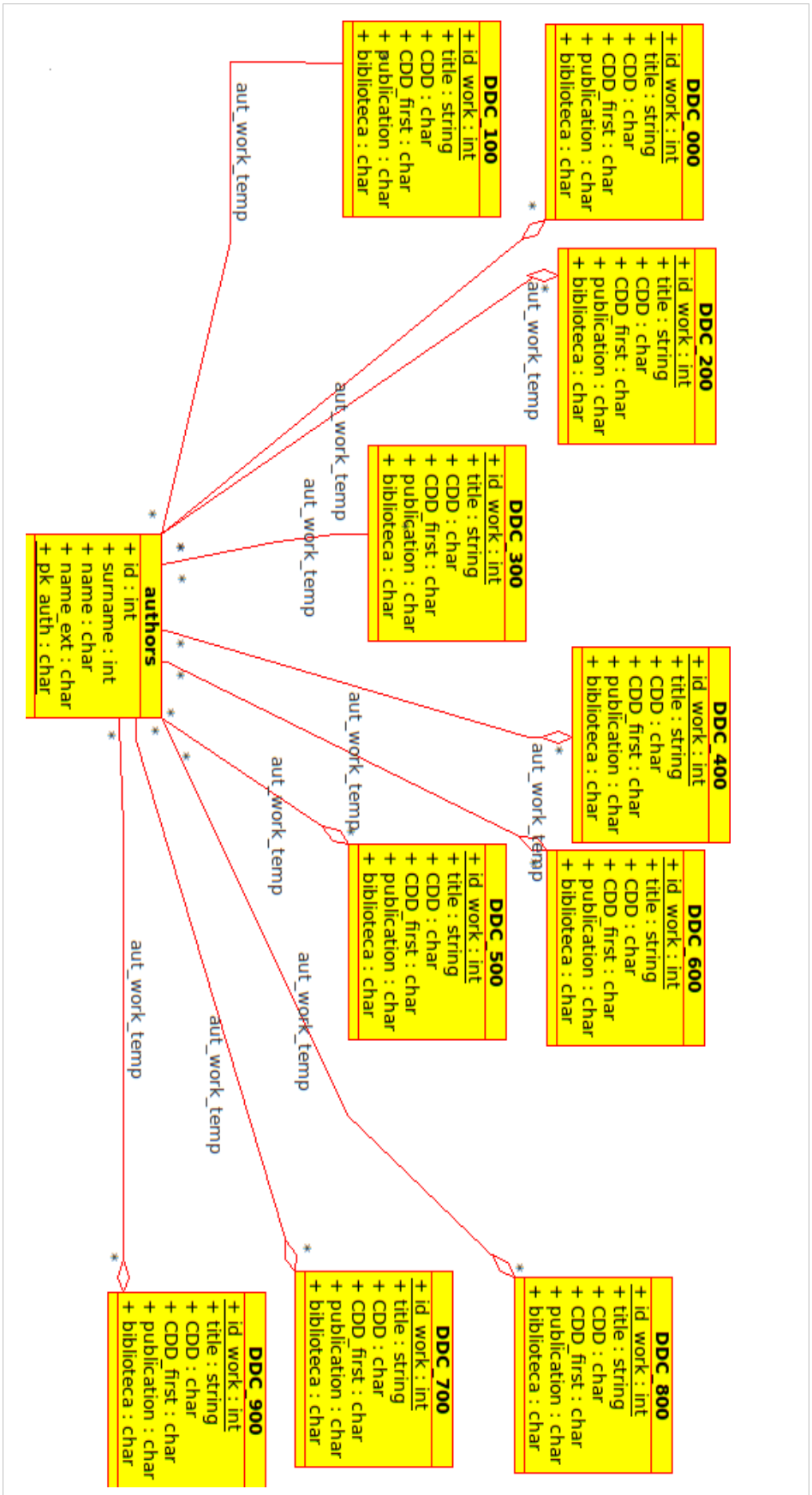


Illustrazione 4: Schema ER delle tabelle temporanee del database

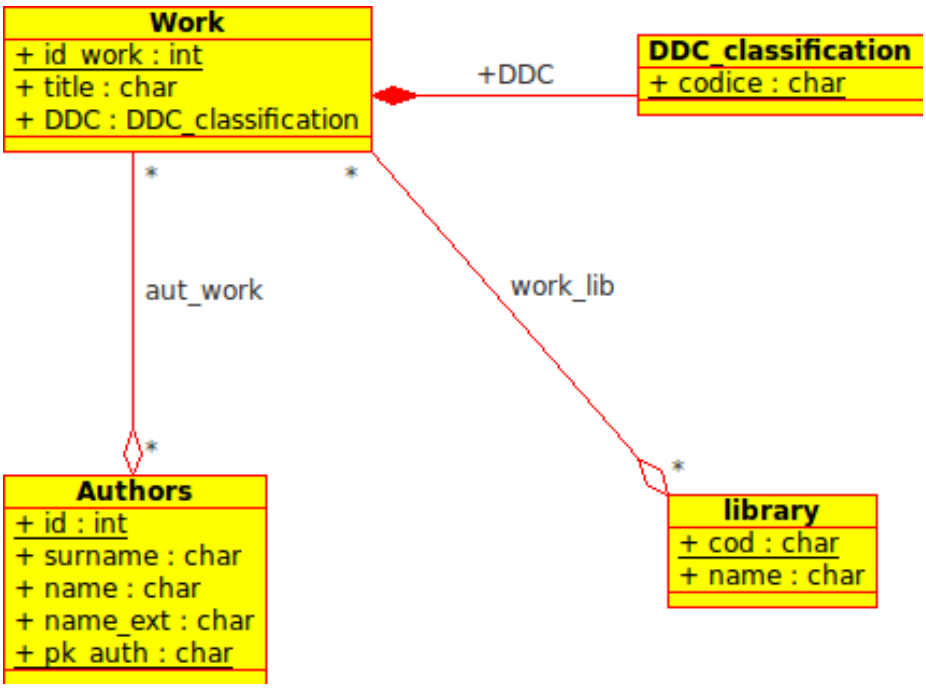


Illustrazione 5: Schema ER delle tabelle definitive del database

La parte di database che dovrà ospitare i dati temporanei è strutturata sul modello dell'Ill. 1, che rappresentava la divisione delle schede bibliografiche per argomento. Le tabella da DDC_000 a DDC_900 conterranno le opere che presentano una classificazione Dewey di primo livello corrispondente al numero della tabella. La tabella autori contiene l'elenco degli autori delle opere, ed è in relazione molti a molti con le tabelle DDC_*, per cui servirà una tabella di aggregazione (aut_work_temp) per collegare ogni opera ai suoi autori e viceversa.

L'Ill. 5 invece mostra la struttura dati del database che conterrà i dati del catalogo unico.

La tabella Work conterrà i dati identificativi dell'opera, il titolo, il codice DDC e sarà collegata alla tabelle authors e library da delle tabelle di aggregazione (aut_work e work_lib). Una ulteriore tabella DDC_classification contiene l'equivalente verbale dei codici Dewey.

5.2. Il formato dei dati di input

I dati di input, come si è detto più volte, sono i cataloghi delle biblioteche. Questi dati per potere essere elaborati dal programma devono avere un formato che il software possa gestire. Si è scelto di adottare il formato xml sia per l'input che, come si vedrà in seguito, per uno degli output del programma.

La struttura di un file xml è costituita da una serie di elementi, detti nodi, che creano una struttura ad albero, e quella dei cataloghi delle biblioteche è mostrata nell'Ill. 6.

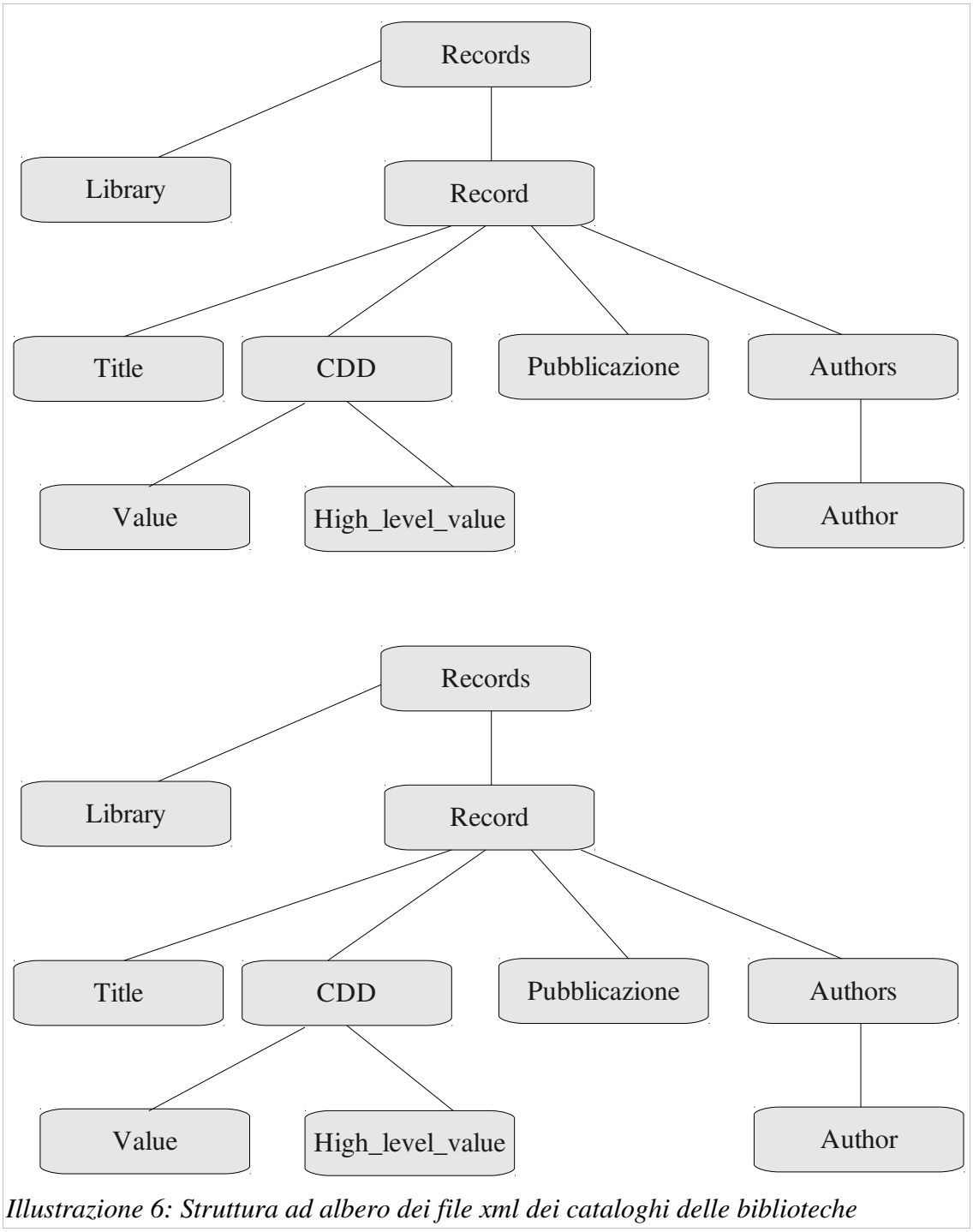


Illustrazione 6: Struttura ad albero dei file xml dei cataloghi delle biblioteche

Come si vede il nodo records contiene due tipi di nodi: library, che indica il nome della biblioteca di riferimento del catalogato, e record, che apre la descrizione di un record bibliografico. Un record contiene il title (titolo dell'opera), il DDC (il codice di classificazione Dewey), la pubblicazione dell'opera e authors, che puàò contenere uno o

più author (autore). Quella mostrata nell'Ill. 6 è la struttura ad albero generale, un esempio di catalogo è quello dell'Ill. 7

```
<records>
	<library>itcc</library>
	<record>
		<title><![CDATA[L'isola del tesoro]]></title>
		<authors>
			<author><![CDATA[STEVENSON, Robert
			Louis]]></author>
		</authors>
		<CDD>
			<value>808.8/r</value>
			<high_level_value>808</high_level_value>
		</CDD>
		<pubblicazione><![CDATA[Novara : De Agostini,
		c2006]]></pubblicazione>
	</record>
	<record>
		<title><![CDATA[A ciascuno il suo]]></title>
		<authors>
			<author><![CDATA[SCIASCIA,
			Leonardo]]></author>
		</authors>
		<CDD>
			<value>853.9</value>
			<high_level_value>853</high_level_value>
		</CDD>
		<pubblicazione><![CDATA[Torino : Einaudi,
		1966]]></pubblicazione>
	</record>
	<record>
	....
	</record>
</records>
```

Illustrazione 7: porzione di file xml del catalogo dell'Istituto tecnico commerciale "Pacinotti"

Come si vede dall'Ill.7 le informazioni che ogni record bibliografico porta sono titolo dell'opera, uno o più autori, la classificazione Dewey divisa in codice completo e high_level_value, che consiste nelle prime tre cifre del codice, e la pubblicazione.

La seconda riga del file xml riporta invece tra i tag <library> </library> il codice della biblioteca di appartenenza del catalogo.

Perché i cataloghi possano essere elaborati dal programma devono avere i campi qui illustrati.

5.3. Una operazione preliminare: la pulizia dell'input

I dati di input, prima di essere tradotti in formato xml per essere elaborati dal programma, sono contenuti nei database delle rispettive biblioteche. La codifica dell'xml deve essere in utf-8, perché il software può elaborare i testi correttamente solo se sono in questo formato.

Nel momento dell'inserimento dei dati nei cataloghi, capita che vengano introdotti dei caratteri che appartengono ad altri tipi di codifica. Quando il programma incontra uno di questi caratteri genera un errore e o ignora il carattere o si blocca. Né il primo né il secondo caso può essere ammesso, per cui occorre che i dati di input, nel momento in cui vengono passati al programma, siano privi di caratteri che possono generare errori. La soluzione per risolvere questo problema è che i file xml preliminarmente vengano sottoposti a una procedura che individui i caratteri con una codifica diversa e li ricodifichi in utf-8. E questo è quanto fa il programma prima di passare alla lettura dei dati dall'xml: esamina tutto il documento e sostituisce i caratteri errati. Questa procedura purtroppo richiede del tempo, ma è fondamentale per la corretta esecuzione del programma.

5.4. Organizzazione dell'input

I dati presenti nei file xml dei cataloghi delle biblioteche devono essere estratti, memorizzati e tenuti pronti per l'elaborazione. Java mette a disposizione due strumenti per estrapolare le informazioni da file xml: la libreria DOM e la libreria SAX. Entrambe queste librerie svolgono la funzione di individuare i nodi dell'albero xml e leggerne il contenuto, ma operano in modo diverso. La libreria DOM carica in memoria tutto

l'albero xml che gli viene passato in input e poi, con una serie di metodi, permette di navigare l'albero su tutti i livelli spostandosi di nodo in nodo. La libreria SAX invece si basa sugli eventi: scorre il file xml e ogni volta che incontra un nodo esegue le operazioni impostate per la sua gestione. Prima di decidere quale dei due sistemi sarebbe stato meglio usare, si è fatta una analisi delle dimensioni effettive dei file da gestire. Il catalogo dell'università di Pisa, che è quello contenente più record bibliografici, occupa 148 Mb di spazio, una quantità troppo grande da potere essere tenuta in memoria senza rischiare blocchi del sistema. Per cui si è scelto di optare per l'uso della libreria SAX. Il secondo problema è stato quello di memorizzare in qualche modo i dati ricavati dall'xml, che servono per le fasi successive del programma. Anche in questo caso non è conveniente memorizzare tutto su delle variabili, che occuperebbero tantissima memoria vista la quantità di informazioni. Quindi si è pensato di appoggiarsi su un database.

Grazie alla libreria SAX di Java il programma può scorrere il file xml, e man mano che incontra i record li inserisce nel database, tenendo così in memoria al massimo un record per volta.

Il parser di SAX è basato sugli eventi: un parser è un programma che analizza un input e ne determina la struttura grammaticale, in questo caso la struttura xml; SAX infatti individua le aperture e le chiusure dei tag, ed esegue dei comandi quando si verificano questi eventi (che nel parser sono indicati con startElement () e endElement()).

Il parser quindi inizia a scorrere il file xml. Quando incontra l'apertura di un tag controlla di che elemento si tratta, legge il contenuto dell'elemento e lo memorizza in una apposita classe creata per contenere le informazioni relative a una singola opera.

La procedura viene spiegata nell'Ill.8. Il parser prende in input il file xml e una classe Work che viene compilata con i dati presi dall'xml compresi tra i tag <record> e </record>, in particolare il al campo title viene dato il valore del tag <title>, a DDC viene assegnato il tag <value> e a DDC_first il tag <high_level_value> (questi ultimi nell'xml sono contenuti dentro <DDC></DDC>). Quando il parser trova il tag di chiusura </record>, e quindi ha finito di analizzare tutto il contenuto di un record bibliografico, invia l'oggetto Work creato alla tabella del database in base alla prima cifra del codice DDC. Così in questa fase viene effettuata anche la divisione delle opere

per argomento. Prima di passare alla fase successiva del programma, questa operazione di parsing dell'xml viene eseguita in sequenza per tutti i cataloghi che devono essere inclusi nel catalogo unico.

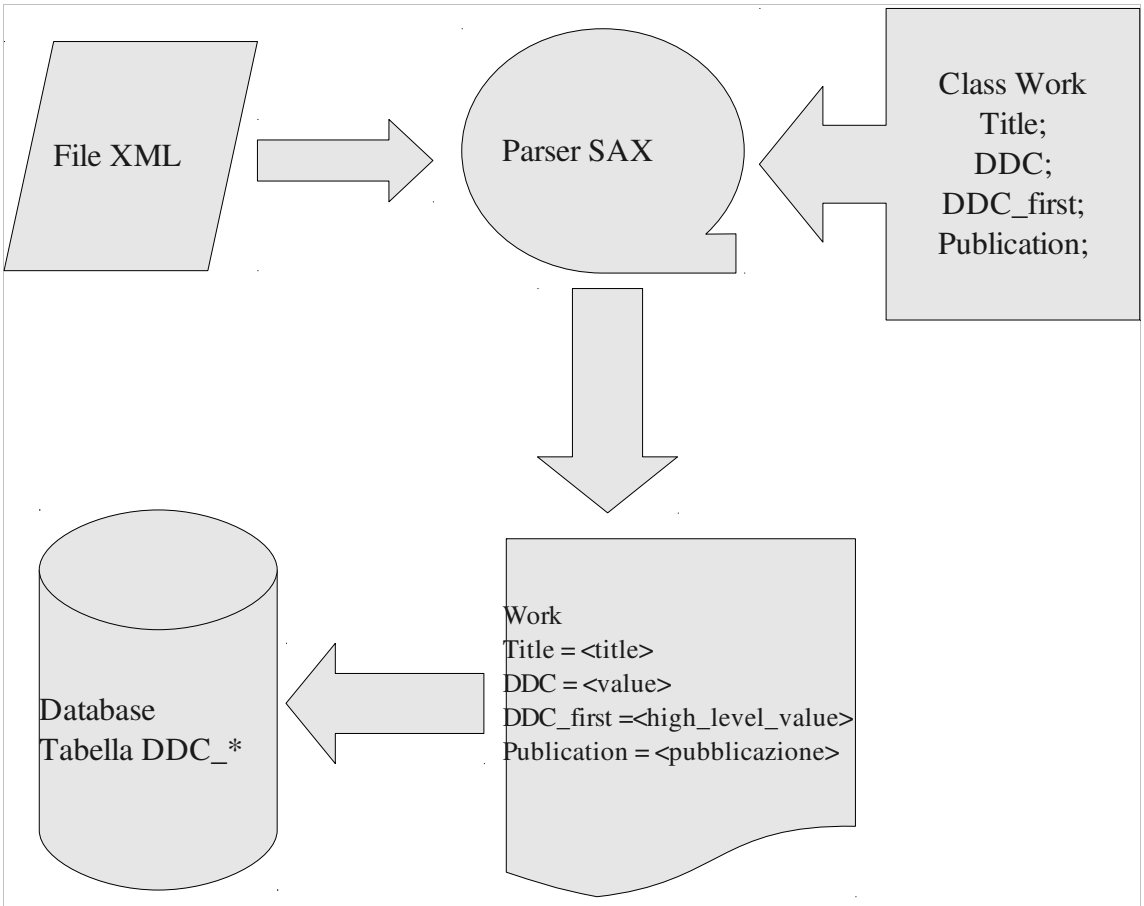


Illustrazione 8: Procedura di gestione dei dati di input

5.5. Elaborazione dei dati

Qui si farà una panoramica puramente descrittiva del programma realizzato, inserendo le porzioni di codice solo quando ritenute utili e interessanti, mentre si lascia al lettore la possibilità di approfondire gli aspetti tecnici nell'Appendice 1 che riporta per intero le porzioni di codice ritenute rilevanti per la comprensione del funzionamento del software.

La fase di elaborazione dei dati consiste nella selezione delle opere, eliminazione dei duplicati e inserimento dei record nelle tabelle del catalogo unico.

L'organizzazione del database con una tabella per argomento facilita notevolmente il lavoro sui dati, perché permette di lavorare su pochi record bibliografici per volta. Tuttavia anche le singole tabelle di argomenti contengono migliaia di record, quindi nel procedere con l'elaborazione bisogna fare attenzione a non sovraccaricare il sistema con troppi dati da tenere in memoria.

Le operazioni da svolgere in questa fase sono:

recupero delle opere dalle tabelle temporanee del database;

individuazione delle opere ripetute;

memorizzazione dei dati nelle tabelle definitive del catalogo unico.

In questa fase si ripropone il problema già visto quando si è parlato della gestione dei dati di input (v. 5.4), ovvero una gran quantità di dati da gestire, con in più la necessità di operare confronti fra di essi. Come si è già spiegato nell'Ill. 1, la divisione in sottogruppi per argomento è stata fondamentale per poter organizzare il materiale da controllare. Quella prima divisione è stata svolta basandosi solo sul primo numero della classificazione Dewey, e questa scelta è stata fatta per un motivo specifico: avere gruppi di argomenti generali permette di avere solo 10 tabelle nel database; se si fosse dovuto prendere in considerazione tutta la prima parte del codice (le prime 3 cifre) si sarebbe arrivati a 1000 tabelle, a e allora il problema non sarebbe più stata la quantità di dati per tabella, che sarebbe diventata irrisoria, ma la moltiplicazione delle tabelle da controllare. Quindi 10 tabelle per 10 macroargomenti è sembrato il giusto compromesso.

Arrivati in questa fase però anche il contenuto di una sola tabella risulta arduo da tenere in memoria per l'elaborazione, quindi si è deciso di sfruttare quella parte della classificazione Dewey che ancora non era stata utilizzata. Opere con titoli relativi allo stesso argomento avranno lo stresso codice completo, quindi selezionando di volta in volta le opere con uguale classificazione si lavorerà con campioni di opere non numerosissimi. La quantità di opere relative a un sottoargomento varia in base ai cataloghi di provenienza delle singole opere. Un catalogo di una istituzione religiosa ad esempio avrà la maggior parte delle sue opere nella sezione religione (200) o filosofia (100). Questo comporta un ulteriore vantaggio: analizzare i dati in base agli argomenti permette spesso di confrontare schede bibliografiche provenienti dallo stesso catalogo e

che quindi in teoria dovrebbero avere più elementi in comune, facilitando così le operazioni di confronto.

La ridistribuzione delle opere in base al sottoargomento stavolta non richiederà l'uso di un database di appoggio. Infatti avendo già disposto il materiale su un supporto di facile consultazione basterà recuperare selettivamente i dati di cui si ha bisogno. Così combinando diverse query al database si riesce a selezionare in modo efficace le opere di cui si vuole effettuare il controllo, tenere in memoria solo le informazioni necessarie, per non appesantire il programma, e inviare i dati elaborati alle tabelle definitive.

Recuperare le opere con uguale classificazione Dewey e confrontare tra loro prima gli autori e poi i titoli, questa è la sequenza delle operazioni da compiere. Come implementare la loro esecuzione è stato oggetto di diverse riflessioni, perché davanti a operazioni a prima vista banali esistono molteplici modi di esecuzione, ognuno con i suoi vantaggi/svantaggi in termini di complessità e benefici. Prima di scrivere il codice che permette di eseguire questi compiti, è stata condotta una operazione di analisi del problema.

La prima soluzione, che è anche la più ovvia, è stata quella del confronto “tutto con tutti”, ovvero un confronto sequenziale fra le opere (v. Ill. 9).

Questo sistema prevede la selezione dalla tabella di tutte le opere con uguale proprietà CDD_first (con una query del tipo “SELECT * FROM DDC_8 WHERE CDD_first = 811”) e confronto di tutte le opere estratte. Il confronto per autore richiede una ulteriore richiesta al database per recuperare gli autori delle opere contenuti nella tabella di aggregazione aut_work_temp. Una volta presenti sia le opere che gli autori, si può iniziare il confronto fra coppie di opere. Se un'opera risulta uguale a un'altra, allora si memorizzano le informazioni relative alla biblioteca di appartenenza della seconda e si verifica che la nuova opera non abbia altri autori diversi dalla prima.

Infatti una situazione che si potrebbe verificare è la seguente: date le opere T1 con gli autori collegati A1 A2, T2 con autori A2 A3, e T3 con autori A3, si devono effettuare i confronti tra le opere. Si sa che T1, T2 e T3 sono in realtà la stessa opera che ha per autori A1 A2 e A3, solo che, avendo molteplici autori, questi non sono stati inseriti nelle schede bibliografiche con gli stessi criteri, per cui tra le tre schede bibliografiche in esame non c'è una corrispondenza esatta. Il confronto tra i titoli delle schede

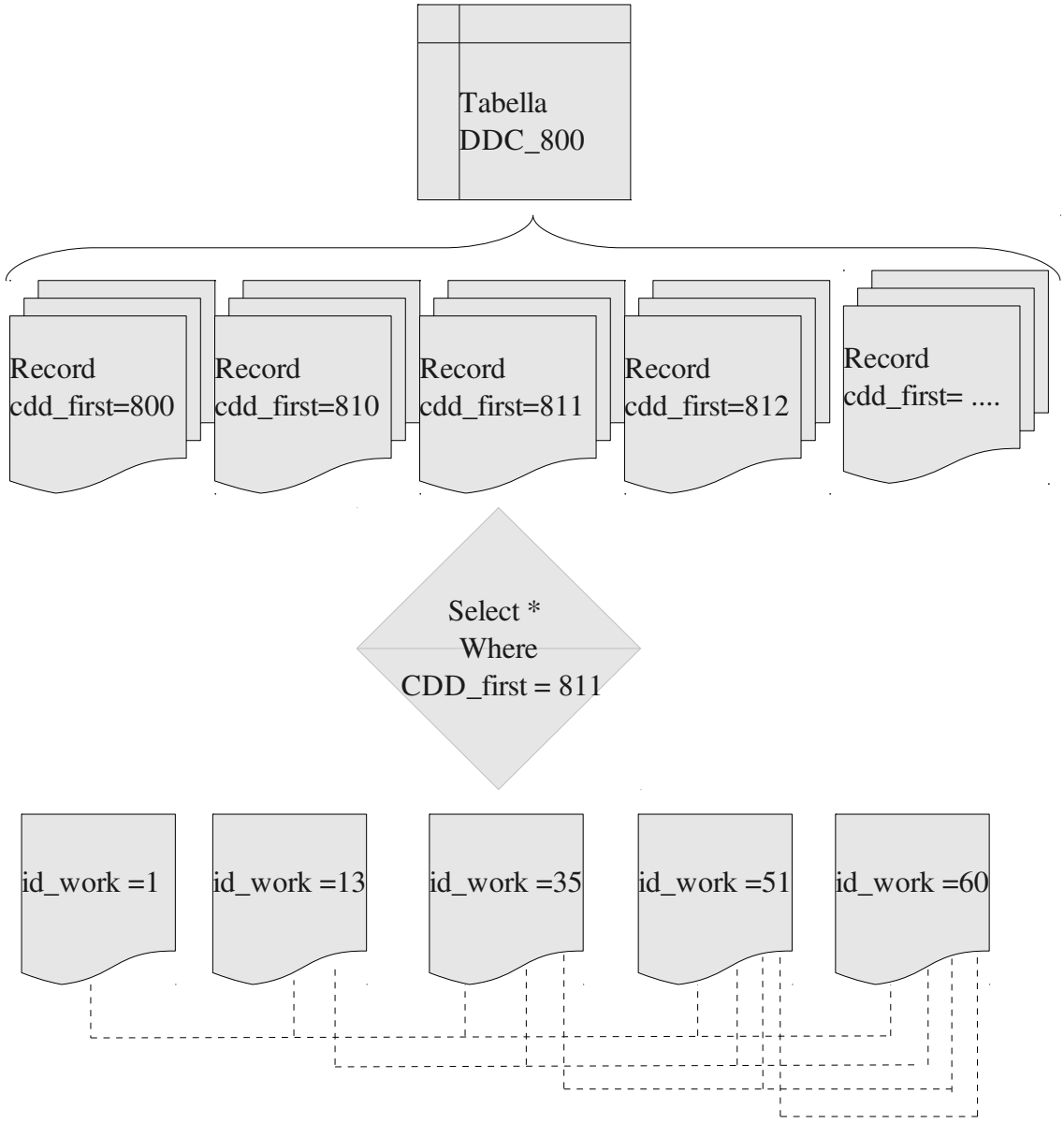


Illustrazione 9: Modello di confronto sequenziale

bibliografiche, tenendo conto della possibilità di avere situazioni di non corrispondenza del numero di autori, avviene in caso di uguaglianza di almeno un autore.

Nel caso presentato e seguendo il modello sequenziale di confronto prima illustrato, T1 troverà corrispondenza in T2 e le due opere verranno confrontate, T3 però verrà esclusa dal confronto non avendo autori in comune con T1. Il risultato del primo confronto sarà che T1=T2 ma T1 != T3, cosa che in realtà non è vera. La successiva serie di confronti, ovvero T2 con le schede rimanenti, verificherà che T2 e T3 hanno in comune l'autore A3, e il confronto dei titoli confermerà che T2 = T3, con evidente contraddizione della

proprietà transitiva! La possibilità che si verifichino tali situazioni obbliga a rivedere il meccanismo dei confronti o ad adottarne uno nuovo.

Il sistema di confronto sequenziale presentato in Ill. 9 ha una complessità di n!, dove n è il numero di schede da confrontare, e il numero di query da effettuare al database sono

1*CDD_first + n, ovvero 1 query per ogni sottoargomento per selezionare le opere con lo stesso ddc più una query per opera per selezionare gli autori. Se si aggiunge che per ogni confronto, nel caso in cui una scheda presenti autori aggiuntivi (esempio di T1 e T2), bisognerà eseguire una nuova ricerca per verificare l'esistenza di opere con gli stessi nuovi autori per cui il numero di confronti dell'algoritmo è destinato ad aumentare notevolmente. Inoltre gestire questa situazione dal punto di vista del codice di programmazione richiederebbe l'impiego di un gran numero di variabili di appoggio per i risultati temporanei, perché prima di inviare l'opera alla tabella del database bisogna verificare che tutti i controlli necessari siano stati effettuati.

Quindi si sono cercate altre soluzioni più efficienti per gestire il problema.

Innanzi tutto la selezione delle opere dalle tabelle. Come si è detto più volte le tabelle da DDC_000 a DDC_900 saranno consultate separatamente, per cui solo dopo che si saranno analizzate le opere della tabella precedente si iniziano i controlli su quella successiva. Poi per ogni tabella, dei 100 codici che la Dewey mette a disposizione per ogni classe generale, probabilmente non tutti saranno utilizzati; quindi cercare solo i codici per cui effettivamente esistono delle opere è il primo passo da compiere. A questo punto è iniziata la stesura dell'algoritmo che opera all'interno di ogni sottoargomento, e si è arrivati alla soluzione in Ill.10.

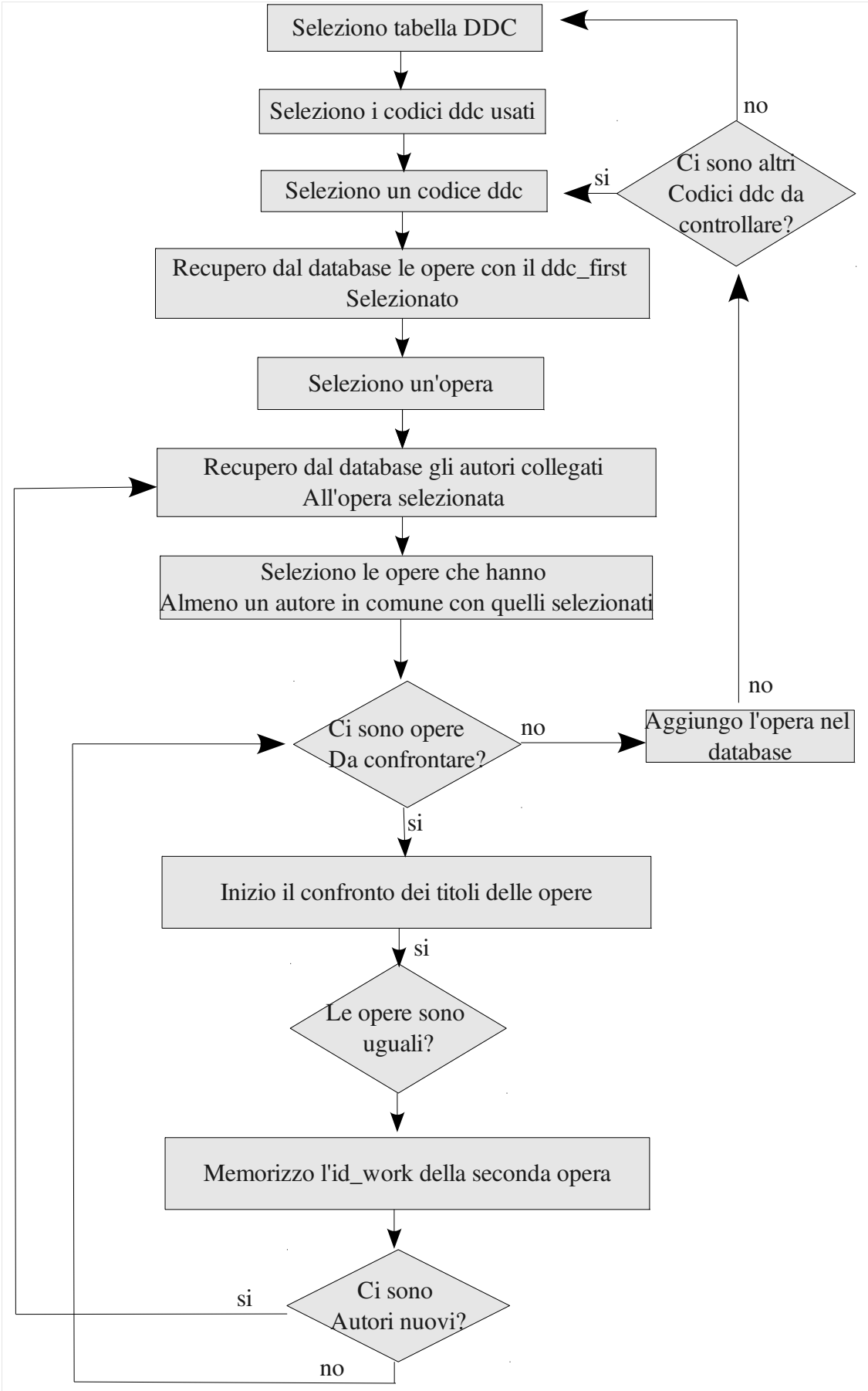


Illustrazione 10: Algoritmo di selezione e confronto di opere

Il primo passo è selezionare una delle 10 tabelle DDC_*. Per ognuna di queste tabelle poi si selezionano gli argomenti unici con una query al database(v. Cod. 1).

```
String query = “SELECT DISTINCT ddc_first FROM table*”;

* “table” è un parametro che assume per ogni iterazione
dell'algoritmo il valore di una tabella del gruppo DDC_.
```

Codice 1: Query di selezione dei ddc usati in ogni tabella

Una volta ottenuto l'elenco dei codici di classificazione usati per una specifica tabella, si compiono delle iterazioni sull'elenco dei DDC selezionando le opere che hanno lo stesso codice di classificazione (Cod. 2).

```
String query = “SELECT distinct title, id_work,CDD_first, CDD FROM
table* where CDD_first = cdd* ”;

* “table” è un parametro che assume per ogni iterazione dell'algoritmo il
valore di una tabella del gruppo DDC_.

* “cdd” è un parametro che varia per ogni iterazione sull'insieme dei codici
cdd unici.
```

Codice 2: Query di selezione delle opere che hanno un determinato codice ddc

Dal gruppo di opere che condividono lo stesso codice di classificazione bisogna ora individuare quelle che hanno gli stessi autori. Anche in questo caso si procede con una iterazione sul gruppo delle opere appena creato. Si comincia selezionando un'opera del gruppo e se ne cercano gli autori richiamandoli dal database (v. Cod. 3) e, successivamente, con un'altra query al database si cercano le opere che hanno almeno un autore in comune con l'opera selezionata precedentemente (v. Cod. 4).

```
String query = “SELECT distinct autwork.author, FROM aut_work_temp
as autwork where  autwork.work= id* AND autwork.ddc=ddc* ”;


* “id” è un parametro che indica l'id_work dell'opera di cui si cercano gli
autori.

* “ddc” è un parametro che varia per ogni iterazione sull'insieme dei codici
cdd unici.
```

Codice 3: Query di selezione degli autori di una determinata opera in base al suo id

```
String query = “SELECT work, author, FROM aut_work_temp as autwork
where  autwork.author in(authList*) AND autwork.work in
(remainingWorks*) and autwork.cdd = cdd* ”;

* “authList” è un parametro che contiene l'elenco degli autori di una data
opera

*”remainingWorks” è un parametro che contiene l'elenco delle opere tra cui
si cercano autori in comune con la lista authList.

* “ddc” è un parametro che varia per ogni iterazione sull'insieme dei codici
cdd unici.
```

Codice 4: Query di selezione delle opere che hanno autori comuni a una data opera

Se la query restituisce dei risultati, allora si procede al confronto della prima opera con quelle selezionate dal database altrimenti l'opera viene direttamente inviata alle tabelle del catalogo unico. Il confronto fra i titoli delle opere verrà approfondito nel paragrafo successivo, mentre qui si preferisce completare la descrizione dell'algoritmo.

Quindi a questo stato dell'algoritmo si ha un'opera (W) da confrontare con un elenco di opere (L) con cui ha almeno un autore in comune. Per ogni confronto fra (W) e un'opera dell'elenco (W^1) si possono verificare due casi: i titoli sono diversi e quindi si passa al confronto con l'opera successiva (se esiste), oppure i titoli sono uguali e quindi le opere vanno accorpate in un'unica scheda. In questo secondo caso verrà effettuata una ulteriore verifica sugli autori. Infatti si è detto che in caso di autori multipli, potrebbe

succedere che schede che descrivono opere uguali non riportino lo stesso numero di autori. Per questo bisogna verificare che l'opera da accorpare non porti degli autori nuovi. Quindi si estraggono dal database tutti gli autori dell'opera (W^1) e si verifica che non ci siano autori diversi rispetto a quelli di (W). Nel caso in cui (W^1) abbia autori in più, si ripetono ricorsivamente le operazioni fin qui descritte cercando opere che abbiano almeno un autore in comune con(W^1), in modo da trovare tutte le schede che potrebbero descrivere la stessa opera. Quando tutti i confronti fra (W) e le opere dell'elenco (L) sono esauriti e nessuna opera ha autori aggiuntivi da controllare, (W) viene inserito nel database arricchito con le informazioni relative ad autori, biblioteca e pubblicazione delle opere con cui è stata unita. Ogni volta che un'opera viene accorpata, viene eliminata dall'elenco di partenza che contiene tutte le opere con uguale ddc, in modo da non ripetere confronti inutili. In questo modo la complessità dell'algoritmo sarà di n! nel caso pessimo, che è quello mostrato in Ill. 11. Questo si verifica quando tutte le opere che hanno lo stesso codice ddc sono tutte diverse fra loro ma ognuna ha tutti gli autori dell'opera precedente più uno aggiuntivo. Questo fa si che ogni opera vada

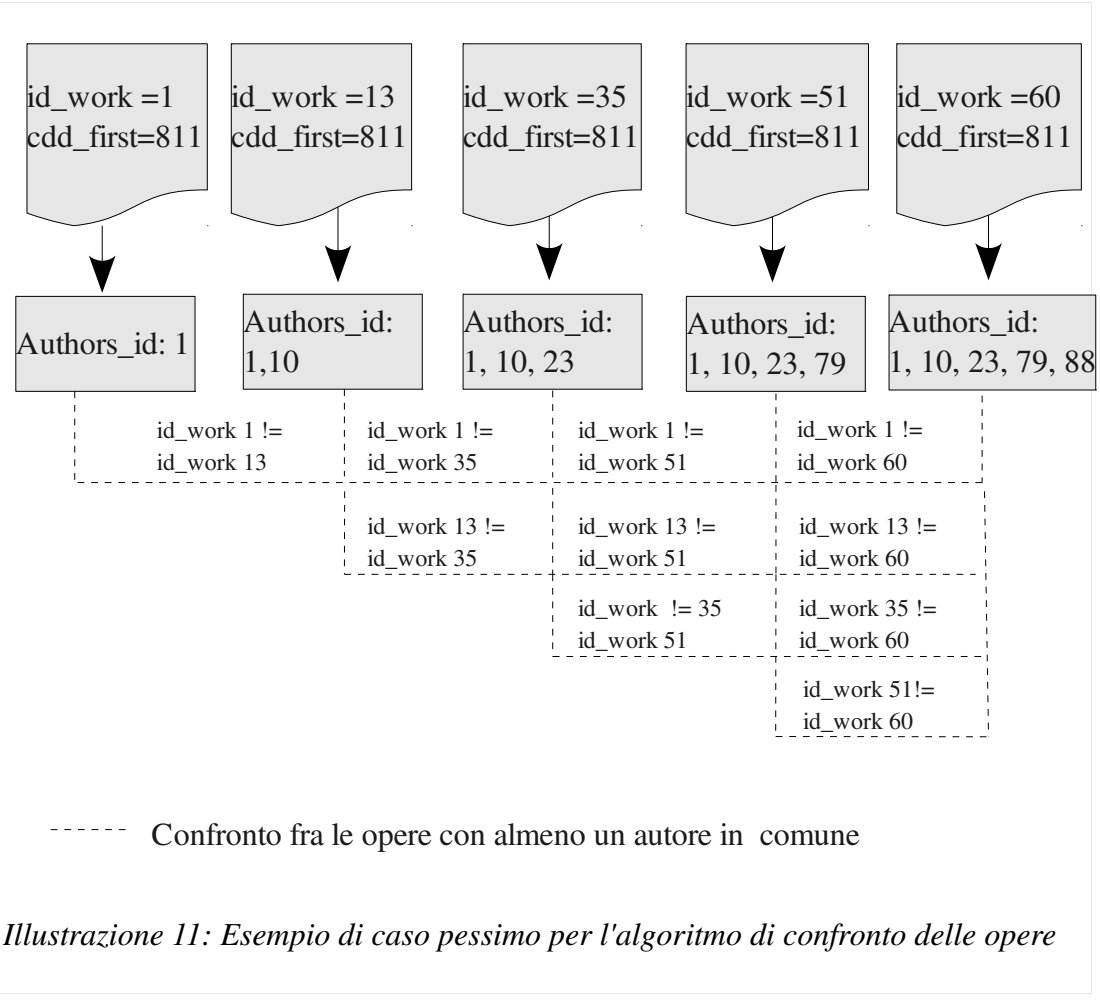


Illustrazione 11: Esempio di caso pessimo per l'algoritmo di confronto delle opere

confrontata con tutte le altre. Tuttavia è abbastanza raro che si possa verificare una simile circostanza.

Il caso migliore per questo algoritmo è che nessuna opera abbia autori in comune con le altre; in questo caso infatti non verrebbe operato nessun confronto. Altro caso abbastanza improbabile, ma che tuttavia è stato preso in considerazione per l'elaborazione di questo algoritmo, è il caso in cui tutte le opere siano uguali. Allora sia che tutte le opere abbiano lo stesso autore, quindi siano confrontate in un unico ciclo di confronti, sia che qualche opera venga recuperata in un secondo passaggio, la complessità è di n-1 confronti, in quanto solo la prima opera verrà confrontata con tutte le altre. A questo punto la complessità del caso medio sarà compresa tra 0 e n! ; nella pratica, un esempio di caso medio potrebbe essere quello dell'Ill. 12, in cui su un certo numero di opere ci sono un paio di coppie di opere ripetute e un caso di autore in comune tra opere diverse.

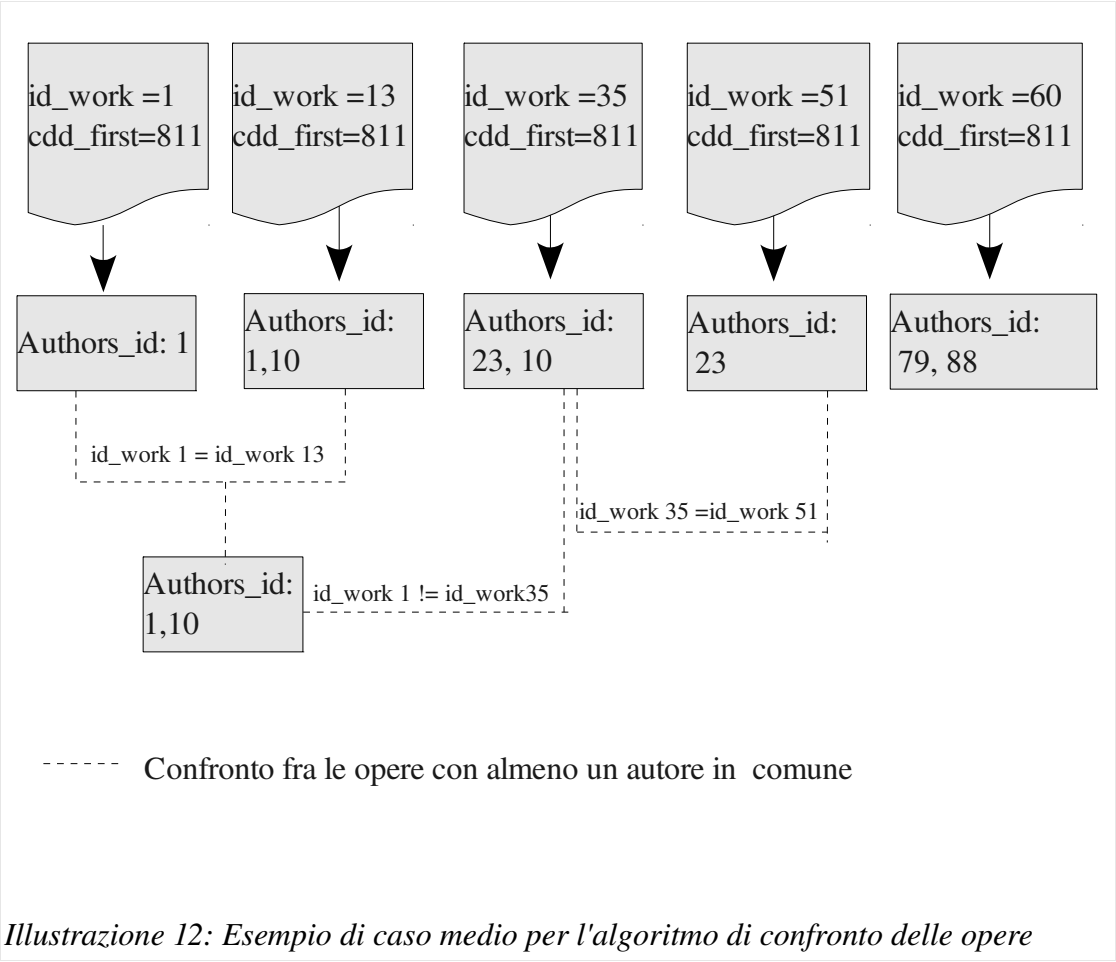


Illustrazione 12: Esempio di caso medio per l'algoritmo di confronto delle opere

Alla fine di tutte le iterazioni sugli argomenti e sulle tabelle DDC_*, la parte di database

destinata a contenere i dati sul catalogo unico sarà popolata di record. La tabella Work conterrà tutte le opere, la tabella work_lib che metterà in relazione le opere con le biblblioteche che le possiedono, e infine la tabella di aggregazione aut_work che collegherà le opere con i rispettivi autori contenuti nella tabella authors, che è l'unica tabella a rimanere invariata nel corso della elaborazione dei dati, in quanto eventuali ripetizioni dei nomi vengono gestite in fase di inserimento dei dati.

Il processo di elaborazione si conclude qui. Nel paragrafo 5.5.1. si riprende la fase di confronto dei titoli, che precedentemente era stata saltata per non perdere continuità nella spiegazione dell'algoritmo.

5.5.1. Il confronto dei titoli delle opere

Un aspetto interessante di questo lavoro è stato lo studio di procedure che individuassero automaticamente anomalie nella scrittura dei titoli delle opere. Infatti in fase di confronto tra due opere l'interpretazione del titolo diventa un aspetto cruciale. Stabilire se due titoli sono uguali, al di là della semplice corrispondenza delle lettere che li compongono, è una operazione che per un essere umano è abbastanza semplice: infatti un uomo riesce a individuare eventuali errori ortografici o di battitura solo attraverso la lettura dei testi, e il suo background culturale spesso gli consente di individuare tra due versioni di titoli quella corretta; infine, cosa non meno importante, può effettuare ricerche in caso di dubbi. Tutte queste operazioni non sono eseguibili con la stessa facilità da un programma: gli errori di scrittura possono essere individuati solo se rientrano in un insieme di casistiche che il software può gestire, le ricerche possono essere svolte solo su insiemi predefiniti di dati e soprattutto un programma non può fornire una interpretazione dei dati. Partendo da questi presupposti, si è capito che non tutte le anomalie presenti nei titoli possono essere gestite dal programma, se non con il rischio di perdere informazioni. Quindi si è cercato di individuare i casi più frequenti di differenziazione dei titoli e di fornire al software le istruzioni per individuarli e gestirli al meglio.

La procedura di confronto dei titoli prende avvio se si è già verificato che le opere da confrontare abbiano lo stesso codice di classificazione Dewey e almeno un autore in comune. Così si può ipotizzare che se i titoli presentano lievi differenze si può essere in

presenza di errori di trascrizione.

La prima questione da risolvere è stata come determinare quando due titoli sono differenti e come quantificare questa differenza. Come si è già detto in 4.1, il grado di similarità tra due titoli si potrebbe vedere come il numero di cambiamenti da fare per trasformare un titolo in un altro. Seguendo questo criterio allora la differenza fra due titoli può anche essere quantificata con un numero intero che indica quante modifiche sono servite per rendere due titoli uguali. Esiste un algoritmo che già implementa questa procedura, ed è l'algoritmo che calcola la Distanza di Levenshtein[11]. Per distanza di Levenshtein si intende proprio il numero che indica quante modifiche bisogna apportare a una stringa A affinchè diventi uguale a una stringa B. Grazie all'algoritmo che calcola questa misura, esiste una buona base di partenza per iniziare il confronto fra due titoli.

Infatti se la loro distanza di Levenstein è troppo grande allora saranno diversi, se è uguale a 0 saranno uguali mentre se è piccola esiste la probabilità che i due titoli siano uguali. Per determinare quando la distanza fra due titoli è “grande” o “piccola” si è deciso di impostare un tetto di tolleranza alla distanza misurata, che è 3. Infatti è ragionevole pensare che se uno dei titoli è stato soggetto a un errore di battitura, allora la differenza sarà di una o due lettere, al massimo tre. Se c'è una differenza dovuta a un articolo, congiunzione o comunque a parti del discorso che non portano significato anche in questo caso si troverebbe una differenza tra le stringhe minima. Invece se uno dei titoli presenta anche il sottotitolo o altre specificazioni allora la distanza tra le due stringhe crescerebbe così tanto da rendere molto difficile l'individuazione, da parte del programma, della causa delle differenze. Inoltre un altro aspetto tenuto in considerazione è la lunghezza del titolo, perchè nel caso di titoli particolarmente brevi un cambiamento di 3 elementi potrebbe stravolgerlo completamente. Per questo un altro limite imposto alla distanza di Levenshtein è che non sia superiore a metà della lunghezza della stringa.

La distanza di Levenstein viene calcolata prima sui titoli così come sono scritti nei

11 Nella teoria dell'informazione e nella teoria dei linguaggi, la distanza di Levenshtein, o distanza di edit, è una misura per la differenza fra due stringhe. Introdotta dallo scienziato russo Vladimir Levenshtein nel 1965, serve a determinare quanto due stringhe siano simili. Viene applicata per esempio per semplici algoritmi di controllo ortografico e per fare ricerca di similarità tra immagini, suoni, testi, etc. Da Wikipedia, voce “Distanza di Levenshtein” http://it.wikipedia.org/wiki/Distanza_di_Levenshtein.

cataloghi di partenza (vengono solo convertiti in minuscolo e confrontati senza spazi), poi su una loro versione normalizzata (senza caratteri speciali, apici, accentate); in base al risultato delle due misurazioni,se almeno una di esse rientra nei limiti fissati, verranno prese in considerazione per i controlli successivi i titoli che hanno riportato distanza di Levenshtein minore.

I titoli che hanno riportato una differenza minore o uguale a 3 verranno sottoposti ad altri controlli che hanno lo scopo di individuare in quale posizione si trova l'errore, perchè stabilire a priori che se i titoli che si differenziano per 3 elementi o meno sono uguali rischia di accomunare opere diverse e quindi di perdere informazioni. Invece per quanto possibile si cercherà di individuare la posizione dell'errore, e se si può ricondurre a una serie di casi gestiti dal programma, allora si interverrà di conseguenza, altrimenti i titoli verranno lasciati distinti.

I controlli sulle stringhe che vengono effettuati sono:

- presenza di numeri a inizio stringa;
- presenza di numeri finali;
- presenza di parole in più.

Il primo controllo deriva da una osservazione del catalogo. Si è notato che spesso la prima parte del catalogo (ordinato alfabeticamente) contiene titoli che iniziano per un numero che non farebbe parte del titolo stesso. Probabilmente quel numero indica un volume, o semplicemente è un refuso di qualche operazione, tuttavia potrebbe succedere che a causa di questi numeri iniziali due titoli non vengano riconosciuti come uguali. Quindi se la differenza fra due titoli è dovuta alla presenza di questi numeri (e quindi le stringhe private dei numeri risultato uguali Cod. 5) allora i titoli verranno considerati uguali e accorpati in una unica scheda. L'espressione regolare utilizzata per individuare gli eventuali numeri in posizione iniziale è `"^(\d)+\s(.*)"`, dove il simbolo `^` indica che l'espressione da trovare è a inizio stringa, `(\d)+` indica un carattere numerico ripetuto una o più volte, seguito da `\s(.*)`, ovvero uno spazio con dopo una qualsiasi sequenza di caratteri. Se il compilatore dell'espressione regolare trova una stringa così costruita, seleziona solo la parte di stringa dopo lo spazio (`return matcher.group(2)`), ovvero la parte di stringa che equivale alla parte di espressione

```
//Questo metodo prende in input una stringa e restituisce la stessa
//stringa senza numeri iniziali

private String DeleteNumber(String a) {

      String numReg = "^(\\d)+\\s(.*)";

      // Compile and get a reference to a Pattern object.
      Pattern pattern = Pattern.compile(numReg);

      Matcher matcher = pattern.matcher(a);

      // Find all the matches.
      while (matcher.find()) {
            return matcher.group(2);
      }
      return a;

      }

 “a” è un parametro che contiene la stringa con il titolo di un'opera

”numReg” è l'espressione regolare che individua la presenza di numeri
iniziali nella stringa.

Il doppio carattere \\ è necessario per non incorrere in errori nel
programma, perchè essendo il simbolo \ un carattere speciale occorre che gli
venga messo prima il carattere di escape \, quindi si avrà \\.
```

Codice 5: Metodo DeleteNumber per trovare e cancellare i numeri iniziali nei titoli

regolare inclusa nella seconda coppia di parentesi) .

Il secondo passo della serie di controlli è verificare la presenza di numeri a fine stringa. Questo passaggio serve per distinguere i libri che fanno parte di serie o collane numerate. In questo caso si controlla che la differenza fra due titoli non sia dovuta a un numero diverso in fondo al titolo. Se si verifica questa condizione i titoli sono considerati diversi (Cod. 6). L'espressione regolare che permette di effettuare questo controllo è `”\\s?(\\d+)+$”`, dove il simbolo `$` indica che l'espressione ricercata deve comparire a fine stringa, mentre `\s?(\d+)` indica che dopo uno spazio (il punto interrogativo vuol dire che lo spazio è opzionale) devono esserci uno o più numeri. Se questa condizione si verifica, i numeri finali dei titoli vengono confrontati fra loro e se almeno uno dei due titoli finisce per numero o se entrambi finiscono per un numero ma i loro valori sono diversi allora le opere sono diverse.

```
// questa funzione controlla se le due stringhe hanno numeri finali
//e se sono diversi
//  return true se sono titoli diversi


private boolean finalNumbers(String a, String b) {

      String finNum = "\\s?(\\d+)+$";
      String finA = "";
      String finB = "";

      // Compile and get a reference to a Pattern object.
      Pattern pattern = Pattern.compile(finNum);
      Matcher matcherA = pattern.matcher(a);

      while (matcherA.find()) {

            finA = matcherA.group(1);
      }

      Matcher matcherB = pattern.matcher(b);

      while (matcherB.find()) {


            finB = matcherB.group(1);
      }

      if (((finA != "") || (finB != "")) && (finA != finB)) {
            // almeno una delle due stringhe finisce per un numero
            //e i numeri finali sono diversi

            return true; // stringhe diverse
      }

      else {

            return false;
      }

      }
```

“a” e “b” sono parametri che contengono le stringhe con i titoli delle opere

”finNum” è l'espressione regolare che individua la presenza di numeri finali nella stringa.

Il doppio carattere \\ è necessario per non incorrere in errori nel programma, perchè essendo il simbolo \ un carattere speciale occorre che gli venga messo prima il carattere di escape \, quindi si avrà \\.

Codice 6: Metodo finalNumbers per trovare e confrontare i numeri presenti a fine titolo

Infine il terzo e ultimo passaggio del confronto fra i titoli è verificare se la differenza tra le due stringhe di testo è dovuta all'aggiunta di una parola(codd. 7-8). Può capitare infatti che un titolo venga scritto una volta con l'articolo e una volta senza, o con una congiunzione o preposizione in più. Per questo lo scopo di questo controllo è verificare se uno dei due titoli ha una parola in più rispetto all'altro, capire l'entità di questa parola e stabilire se i titoli possono essere considerati uguali o meno.

```
private boolean DeleteWords(String a, String b) {


String[] stopWords={"il","lo","la", "i", "gli","le", "un", "uno",
"una","di","a","da", "in", "con","su","per", "tra",
"fra","che","ma","non","the","a","it","of"};


int diff = NoBlankSpace(a).length() - NoBlankSpace(b).length();
//diff può assumere valori da 1 a 3


String numReg = "\\s([a-zA-Z]{"+ diff +"})\\s"


Pattern pattern = Pattern.compile(numReg);

a = " "+a; //mi serve mettere uno spazio all'inizio per fare
prendere anche i valori che compaiono a inizio stringa

Matcher matcher = pattern.matcher(a);


String prec;

String succ;

while (matcher.find()) {


      prec= a.substring(0, matcher.start());

      succ=a.substring(matcher.end());

      String delA = prec + succ;


      int newDist = checkDistance(delA.trim(), b);
```

Codice 7: Metodo deleteWords per controllare se la differenza fra i titoli è dovuta alla presenza di parole in più

```
	if (newDist == 0) {

	//cerco se la parola in più è compresa nelle stop Words
	for(int i=0;i<stopWords.length;i++){

		if(stopWords[i].equals(matcher.group(1))){

		newDist=-1;

		return true;

		}

	}
	/* se la differenza fra i due titoli è di una parola di al
	massimo due lettere allora le stringhe sono uguali*/

	if((diff<=2)){
		return true;

	}
	/*se la differenza fra i due titoli è di una parola di più
	di due lettere allora cerco di valutare il contesto*/

	else if ((diff>2) ){

	prec = prec.substring(prec.lastIndexOf(" "));

	succ = succ.substring(0,succ.indexOf(" "));

	/*se la parola si trova tra una parola di al massimo
	due lettere e una di più lettere potrebbe trattarsi di un
	aggettivo e quindi i titoli si considerano diversi*/

	if((prec.length()<=2)&&(succ.length())>=3){

			return false;
		}

		else if(prec.length()==0){

			return true;


		}
		//altri casi in cui la parola compare

		else return true;

		}

	}
	 // se la differenza tra i titoli non è di una parola precisa
	allora i titoli sono diversi

	return false;
}
```

Codice 8: Metodo deleteWords (parte 2) per controllare se la differenza fra i titoli è dovuta alla presenza di parole in più

Il metodo deleteWords() opera in più fasi: per prima cosa si definisce un insieme di stop Word[12] italiane e inglesi che serviranno in una fase successiva. Poi verifico quanta differenza c'è tra le due stringhe in termini di lunghezza, per stabilire quanto eventualmente dovrebbe essere lunga la parola in più; a questo punto si cercano nel titolo più lungo le parole che hanno lunghezza pari alla differenza del numero di caratteri della prima e della seconda stringa con l'espressione regolare `String numReg = "\\s([a-zA-Z]{"+ diff +"})\\s",` che cerca le parole formate da un certo numero di caratteri (il numero di caratteri è dato dalla variabile diff, calcolata facendo la differenza tra le lunghezze delle stringhe). Il valore di diff comunque non potrà mai essere superiore a 3, perché ricordiamo che si sta lavorando su stringhe che hanno già superato lo sbarramento iniziale della distanza di Levenstein inferiore a 3. Per ogni volta che il pattern trova una parola della lunghezza stabilita, questa è eliminata e viene di nuovo misurata la distanza di Levenstein sui due titoli. Se questi sono diventati uguali (la distanza misurata quindi è uguale a 0) allora vengono svolte una serie di operazioni che servono a capire se la parola in questione può essere un elemento di differenziazione tra due titoli. Per prima cosa si verifica se la parola estratta fa parte delle stop Word definite all'inizio del metodo, e in caso affermativo, i due titoli possono essere considerati uguali perché la parola non è portatrice di significato. Invece contrario invece, se la parola ricercata era di lunghezza 1, quindi se la differenza tra le due stringhe e di una parola di una sola lettera , allora anche in questo caso i due titoli vengono considerati uguali, altrimenti si cerca di verificare il contesto da cui la parola viene estratta. Se prima del termine ricercato si trova una parola corta, una o due lettere, e dopo una parola più lunga, allora si può ipotizzare che la parola trovata sia un aggettivo posto ad esempio tra il nome e l'articolo e allora i due titoli non verrebbero accorpati perché potrebbero essere differenti. Infine se nessuno dei casi precedenti si è verificato, comunque i titoli si considereranno uguali,

mentre se la differenza fra i due titoli non sta in una parola precisa ma in una serie di lettere aggiunte a diverse parole i titoli verranno considerati diversi, per evitare di fare accorpamenti casuali e impropri.

12 Le stop word sono parole che, data la loro elevata frequenza in una lingua, sono di solito ritenute poco significative in una ricerca bibliografica. Fra queste, ad esempio, si trovano articoli, preposizioni e congiunzioni. Di norma esse vengono eliminate dai termini di ricerca specificati dall'utente prima che la ricerca venga effettuata. Fonte: sito www.sbn.it sezione help online.

Finito il controllo sulle parole, le verifiche sui titoli sono concluse, in base al risultato restituito in ognuna delle fasi si deciderà quali opere debbano essere accorpate e quali no. Per alcuni esempi di applicazione dei controlli fin qui descritti si veda l'Appendice 2.

6. Gli output del programma: un catalogo, tanti formati

L'output del processo di elaborazione è il catalogo unico delle biblioteche di area pisana. Questo prodotto dell'elaborazione può essere memorizzato su diversi formati, e quelli scelti per questo lavoro sono due: il database e un file xml.

6.1. Il database del catalogo unico

La struttura finale delle tabelle del database destinate a contenere le opere del catalogo unico è già stata descritta nella Ill.5. La tabella Works contiene i titoli delle opere e il codice di classificazione corrispondente, la tabella aut_work è la tabella che collega le opere con i rispettivi autori, mentre quella lib_work aggrega le opere con le biblioteche di riferimento e contiene le informazioni sulla pubblicazione.

Per velocizzare le operazioni di ricerca attraverso l'interfaccia si sono creati degli indici sulle tabelle. I campi che maggiormente si consultano in fase di ricerca sono titolo e autore, quindi i campi indicizzati saranno dalla tabella Works la colonna title e dalla tabella authors le colonne name e surname. Le altre opzioni di ricerca che l'interfaccia mette a disposizione, ovvero l'edizione e l'argomento, non sono tra quelle più utilizzate, quindi creare un indice anche sulle colonne corrispondenti potrebbe comportare uno spreco di risorse.

6.2. Il catalogo in formato xml

Il secondo modo per salvare i dati del catalogo unico è quello di creare un file xml. Il database da solo sarebbe già stato sufficiente per contenere e rendere fruibili i dati del catalogo, però non è un supporto facilmente condivisibile. Quindi fin dalle prime bozze del progetto si è pensato realizzare una doppia versione catalogo, una per la consultazione e una per la condivisione. Per quest'ultima funzione si è scelto il formato xml. Le motivazioni di questa scelta si avvicinano molto a quelle che hanno portato a scegliere l'xml come formato dei dati di input: molti linguaggi di programmazione hanno sviluppato parser per analizzare e estrarre le informazioni dai tag dell'albero xml, i file possono essere generati e trasferiti senza la necessità di software specifici e i file xml possono essere gestiti da fogli di stile che permettono di visualizzare e formattare le

informazioni in essi contenuti. Il catalogo in formato xml potrà essere fatto circolare tra le biblioteche di area pisana e non solo; ogni biblioteca potrebbe sfruttarlo per un suo uso particolare, e, si potrebbe costituire un archivio delle opere presenti nel sistema bibliotecario d'area pisana nel corso del tempo. Però per fare si che il catalogo xml possa davvero diventare uno strumento di condivisione delle informazioni bisognerà che la sua struttura sia conforme a qualche standard, e a questo proposito si è deciso di basare la struttura del catalogo sullo standard TEI[13].

Scopo della TEI è fornire delle linee guida per la rappresentazione di contenuti, in modo da metterne in evidenza le caratteristiche salienti e soprattutto per far si che possano essere elaborate da un computer.

Le linee guida della TEI sono precise ma allo stesso tempo flessibili, in modo da permettere ad ognuno di creare lo schema di codifica più adatto alle proprie esigenze.

Per fare questo la TEI (qui si segue la versione P5) mette a disposizione una serie di elementi per codificare diverse caratteristiche di un testo. Questi elementi poi possono combinarsi tra loro in base alle regole stabilite dallo schema di codifica. Le TEI guidlines inoltre offrono una descrizione di ogni elemento e l'elenco di quali altri elementi può contenere.

Si prenda un esempio di testo codificato per capire il funzionamento dello schema di codifica (Ill. 13). Si vede come la struttura del file sia quella dell'xml, con elementi definiti da tag di apertura e chiusura e attributi.

13 La Text Encoding Initiative (TEI) è un consorzio di istituzioni internazionali, di ambito linguistico e letterario, che ha sviluppato uno standard per la rappresentazione dei testi in forma digitale. La missione della TEI è quella di sviluppare e mantenere una serie di linee guida di alta qualità per la codifica di testi umanistici e per sostenere il loro uso da parte di comunità di progetti, istituzioni e singoli individui. Fonte Wikipedia, voce Text Encoding Initiative all'indirizzo http://it.wikipedia.org/wiki/Text_Encoding_Initiative.

```
<text>

   <front>

     <head type="author">Giovanni Verga </head>

     <head type="title">Rosso Malpelo</head>

     <head type="subtittle">Novella tratta da "Vita dei campi"</head>

     <head type="editor">Codifica ed edizione digitale a cura di Aiola Chiara</head>

     <div type="intro"><p>Rosso Malpelo è una novella dell'opera di Giovanni
Verga, pubblicata nel 1880 e raccolta nella <emph>Vita dei campi</emph> , è uno
dei capolavori del verismo. In essa descrive la realtà di povertà e sfruttamento delle
classi disagiate in Sicilia alla fine del XIX secolo, realtà che egli conosceva ma che
emergeva altresì dalle inchieste del Regno d'Italia da poco formatosi (1861).

Oltre questo l'opera è anche un ritratto, umanissimo e di grande attualità, di un
adolescente (Rosso Malpelo), condannato dai pregiudizi e dalla violenza della gente
all'emarginazione e ad una tragica fine.</p></div>

    </front>
```

Illustrazione 13: Esempio di codifica di un testo secondo lo standard TEI

La porzione di testo codificato scelto per l'esempio inizia con un tag <text>; le guidelines della TEI per questo tag offrono questa descrizione:

> “<text> contiene un unico testo di qualsiasi tipo, sia esso unitario o composito, per esempio un testo in versi o teatrale, una raccolta di saggi, un romanzo, un dizionario, o una porzione di corpus.”

inoltre, sempre in base a quanto si trova nelle guidelines, questo tag può contenere:

analysis: interp interpGrp span spanGrp

certainty: certainty precision respons

core: cb gap index lb milestone note pb

figures: figure

iso-fs: fLib fs fvLib

linking: alt altGrp anchor join joinGrp link linkGrp timeline

spoken: incident kinesic pause shift vocal writing

textcrit: witDetail

textstructure: back body front group

transcr: addSpan damageSpan delSpan fw space

Analysis, certainty ecc. indicano i moduli di cui fanno parte gli elementi.

Sempre nel testo di esempio si vede che dopo <text> si apre il tag <front> (che fa parte del modulo textstructure), ma in base alle regole della Dewey si sarebbe potuto avere un'altro dei tag dell'elenco di sopra. Quali elementi usare per la codifica di un testo dipende da cosa si vuole mettere in evidenza.

Deciso il formato e lo standard dell'output del catalogo, bisogna ora definire lo schema di codifica.

Si parta dalle informazioni che si hanno a disposizione. Per ogni opera si conoscono titolo, autori, un elenco di biblioteche che la possiedono con la rispettiva pubblicazione. Quindi da questa base si possono iniziare a cercare gli elementi che meglio possono codificare questi dati.

Si parte sempre dal tag <text>, che è l'elemento di apertura della parte di documento relativa al contenuto vero e proprio (infatti il documento inizia con una intestazione su cui si tornerà successivamente). Dopo il text va aperto il <body>, che contiene il corpo del testo, dopodiché si possono inserire gli elementi che servono per descrivere il catalogo.

Dal modulo core si può includere l'elemento <biblFull>, "(citazione bibliografica strutturata) contiene una citazione bibliografica interamente strutturata nella quale sono presenti tutti i componenti di descrizione di un file TEI" (TEI guidelines); quindi dentro i tag <biblFull> si possono strutturare le informazioni relative a un record bibliografico. Gli elementi che questo tag può contenere e che servono per la codifica del catalogo si trovano elencati in tab. 10.

Il tag <biblFull> aprirà la descrizione di un record bibliografico. Al suo interno saranno strutturati gli elementi che descrivono l'opera: <titleStmt> conterrà i tag <title> e <author>, così le informazioni che identificano l'opera saranno ben separate dal resto , che invece sarà contenuto in <publicationStmt>; questo elemento sarà ripetuto per

ogni biblioteca che possiede l'opera. Al suo interno verranno collocati i tag <distributor>, con le informazioni sulla biblioteca e <ab> con quelle sulla pubblicazione. La TEI mette a disposizione elementi specifici per la pubblicazione, come <publisher> o <pubplace>, rispettivamente per la casa editrice e il luogo di pubblicazione, ma purtroppo le informazioni che sono nel catalogo non sono strutturate in maniera tale da poter dividere casa editrice da luogo e data di pubblicazione, quindi si ricorrerà ad un elemento generico.

<biblFull>	(citazione bibliografica strutturata) contiene una citazione bibliografica interamente strutturata nella quale sono presenti tutti i componenti di descrizione di un file TEI
<titleStmt>	(dichiarazione sul titolo) raggruppa le informazioni sul titolo di un'opera e sulle responsabilità del suo contenuto intellettuale.
<title>	contiene il titolo completo di una qualsiaisi opera;
<author>	in un riferimento bibliografico contiene il nome dell'autore (o degli autori), personale o collettivo, di un'opera; è la dichiarazione di responsabilità primaria di ciascuna unità bibliografica;
<publicationStmt>	(dichiarazione sulla pubblicazione) raggruppa le informazioni riguardo la pubblicazione o la distribuzione di un documento elettronico o di altra natura.
<distributor>	fornisce il nome di una persona o di un'organizzazione responsabile della distribuzione di un testo.
<ab>	(blocco anonimo) contiene una qualsiasi unità testuale a livello di componente che funge da contenitore anonimo di sintagmi o elementi interlivello simili al paragrafo ma senza il bagaglio semantico di quest'ultimo
<idno>	(numero identificatore) fornisce un identificatore, standard o meno, usato per identificare un'unità bibliografica;

Tabella 10: Elementi per la codifica del catalogo

Stabilito lo schema di codifica con gli elementi da utilizzare, attraverso lo strumento messo a disposizione dalla TEI, Roma[14] si genera la dtd di riferimento per il documento xml che verrà creato (per i dettagli sulla documentazione TEI relativamente agli elementi usati si veda l'Appendice 3).

Il documento così creato conterrà una intestazione, con informazioni relative alla sua produzione, e un tag <biblFull> per ogni opera del catalogo.

14 Il servizio di creazione online degli schemi di codifica è all'indirizzo http://www.tei-c.org/Roma/ .

7. L'interfaccia finale

L'interfaccia di consultazione è fondamentale per potere sfruttare le novità del nuovo catalogo. Infatti avere un insieme di dati ben ordinati e di buona qualità non basta se poi i sistemi per accedervi sono poco usabili o poco efficienti.

Per questo l'interfaccia del metaopac pisano è stata abbandonata in favore di una nuova. Questo sarà di sicuro un esperimento e ci vorranno diversi test per capire se la nuova interfaccia possa soddisfare le esigenze degli utenti del catalogo tanto da spingerli a preferire la nuova versione dell'interfaccia, ma è un tentativo che va fatto. Sicuramente il metaopac pisano va tenuto in considerazione per evitare di stravolgere così tanto l'ambiente di lavoro da creare difficoltà ai suoi utenti abituali. Quindi bisognerà trovare un compromesso tra vecchia e nuova interfaccia.

7.1. Recupero dell'interfaccia del metaopac pisano

Il metaopac pisano è già stato ampiamente descritto nel capitolo 2.2. in cui se ne sono analizzate le funzioni e i meccanismi di ricerca, e si sono individuati i problemi nella visualizzazione dei risultati delle interrogazioni. Qui si vuole capire cosa di questa interfaccia sia riutilizzabile.

Si parta dall'estetica del sito. L'utente del metaopac pisano è abituato ad una grafica minimale che mette in evidenza solo i campi per la ricerca e i risultati. Le altre informazioni sul servizio o sui cataloghi si trovano in altre sezioni del sito e sono accessibili dalla home page (http://leonardo.isti.cnr.it/metaopac/mop/mop.html). Quando si entra nella sezione “Accedi a MOP” ogni altra informazione di contorno scompare e si entra in un ambiente destinato solo alla ricerca. Questo tipo di impostazione grafica verrà sicuramente mantenuta nella nuova interfaccia, che sarà semplice e lineare, senza elementi che possano distrarre l'attenzione dall'obiettivo principale: la ricerca dei documenti. Anzi si cercherà di dare ancora più risalto agli strumenti di ricerca che l'interfaccia fornirà. Anche da questo punto di vista il riferimento al metaopac sarà chiaro: i sistemi di ricerca forniti dal sistema infatti (le *list,* il *search* e il *search in ambiente* testuale) sono validi e completi, per cui questi verranno mantenuti anche nella nuova interfaccia seppur con qualche piccola modifica.

Un'altra funzione utile del metaopac pisano è la possibilità di selezionare le biblblioteche su cui si vuole effettuare la ricerca, solo che capire come fare non è intuitivo. Quindi questa funzione verrà mantenuta ma verrà cambiato del tutto il sistema per fare scegliere all'utente i cataloghi. Non si interverrà invece sulle altre pagine del metaopac: la home page, la descrizione dei cataloghi, le informazioni varie per ora rimarranno le stesse.

7.2. La nuova interfaccia

La progettazione della nuova interfaccia del metaopac risente degli articoli letti a proposito degli opac e degli opac semantici (v. 1.1. e 1.2.), e una prima riflessione sulle caratteristiche della nuova interfaccia è già stata accennata in 4.2. . Quindi qui si illustrerà come nella pratica sono stati risolti i problemi riscontrati dopo la prima analisi sia del metaopac pisano sia degli opac in generale.

Innanzi tutto si è detto che un utente sceglie il sistema di ricerca da utilizzare poco consapevolmente, spesso guidato da ciò che per primo attira la sua attenzione. L'interfaccia del metaopac pisano potrebbe risentire molto di questa tendenza a causa della collocazione dei campi di ricerca. L'Imm.10 evidenzia alcune aree dell'interfaccia che presentano dei problemi.

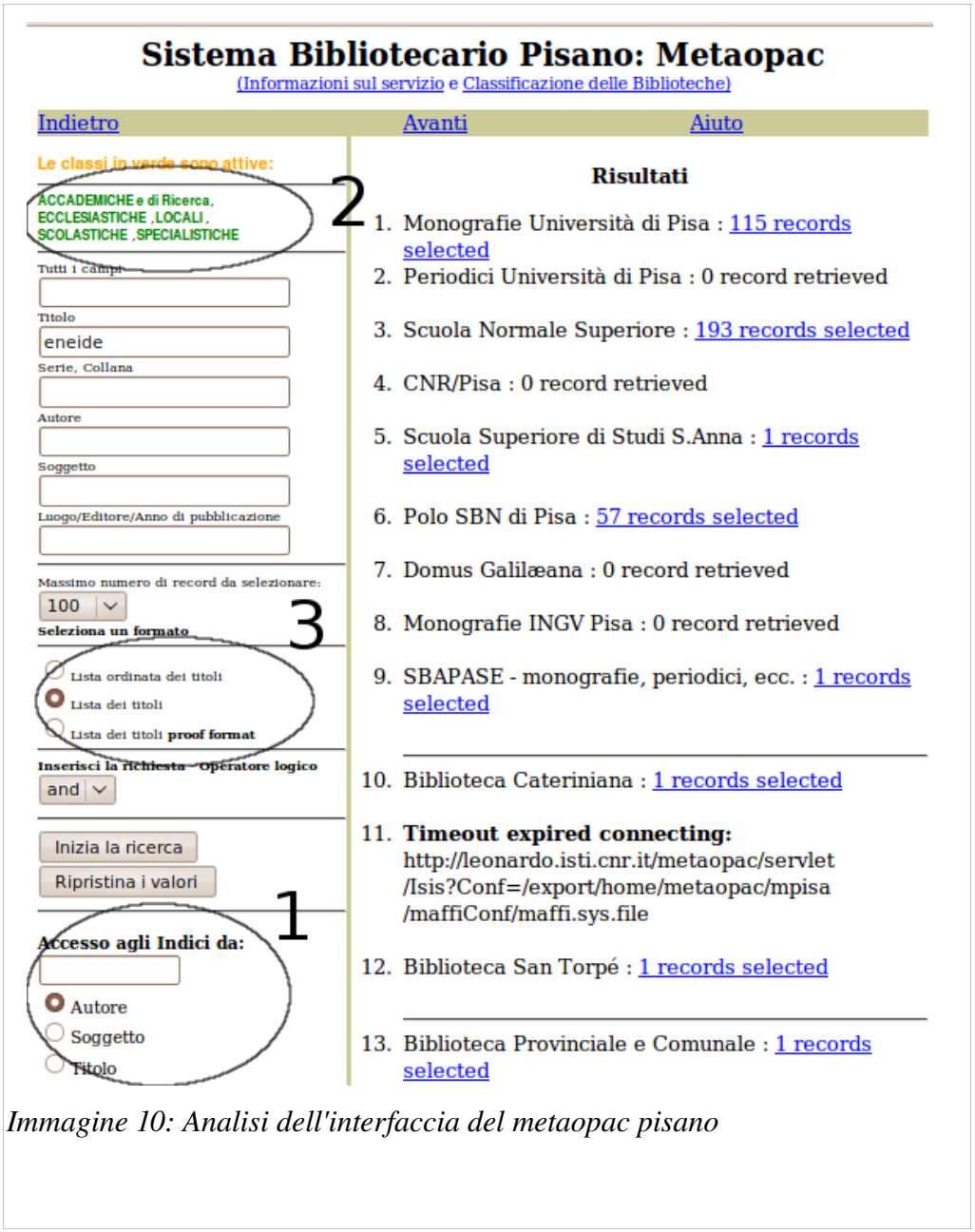


Immagine 10: Analisi dell'interfaccia del metaopac pisano

La zona indicata con il numero 1 contiene i campi di ricerca per ottenere delle liste di opere in base al titolo, autore o soggetto. Questo è il sistema di ricerca di tipo *list,* ed è poco visibile a causa della sua posizione nella parte inferiore della colonna; spesso quei campi di ricerca non sono nemmeno presenti da subito nella schermata, perché in base alla grandezza e alla risoluzione del monitor potrebbe essere necessario fare uno scroll in verticale della pagina per farli comparire. La sezione 2 dell'immagine mostra dove l'utente dovrebbe scegliere quali biblioteche consultare. Come si noterà, non esistono istruzioni che indichino all'utente che cliccando su una classe potrebbe scegliere su quali biblioteche effettuare la ricerca. Infine la terza sezione, mostra una serie di opzioni che

non sono di immediata comprensione, ovvero la scelta del formato di visualizzazione dei titoli. Questa opzione potrebbe spiazzare l'utente: che differenza c'è tra lista dei titoli e lista ordinata dei titoli? A prima vista i risultati della ricerca tra questi due formati sembrerebbe uguale, infatti entrambe le modalità restituiscono una visualizzazione come quella mostrata nell'Im. 11. Questa distinzione in realtà serve solo per quei cataloghi che non sono organizzati alfabeticamente per titolo, ma in base ad altri criteri di ordinamento specifici per il singolo catalogo. Le biblioteche che ordinano seguendo altri sistemi non sono molte e di solito sono biblioteche specialistiche, e questa funzione serve all'utente può scegliere se visualizzare le opere con l'ordine originale o con l'ordine alfabetico. Il formato proof format invece (im. 12) si distingue per il fatto che al posto dei campi titolo, autore ecc. mostra dei codici di identificazione.

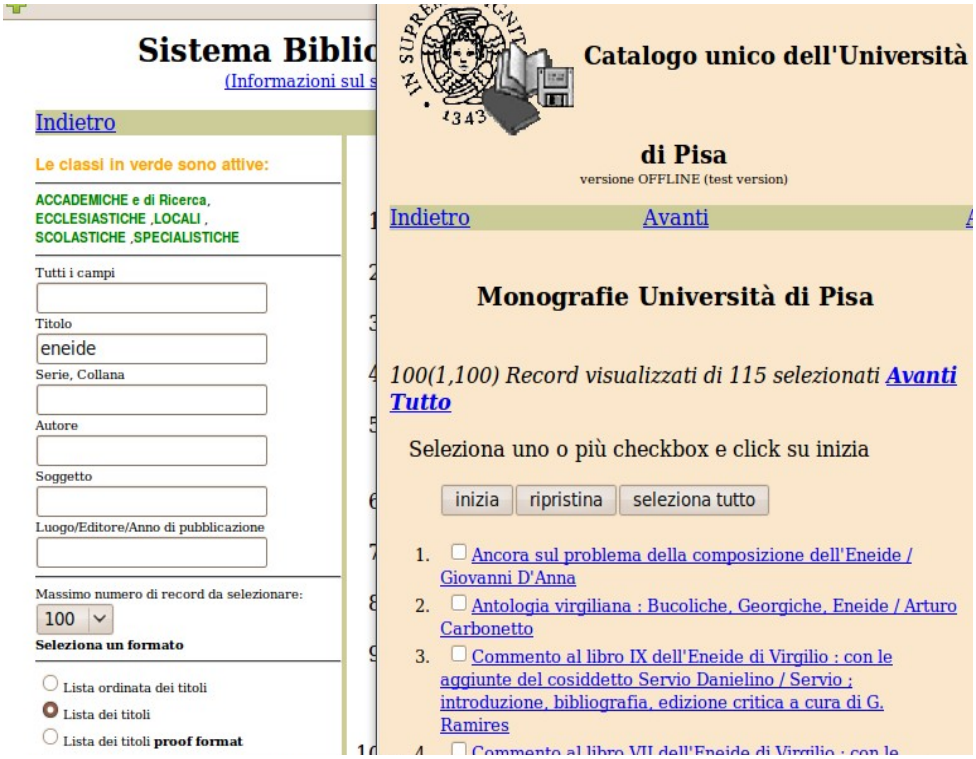


Immagine 11: Risultati della ricerca: Lista ordinata dei titoli

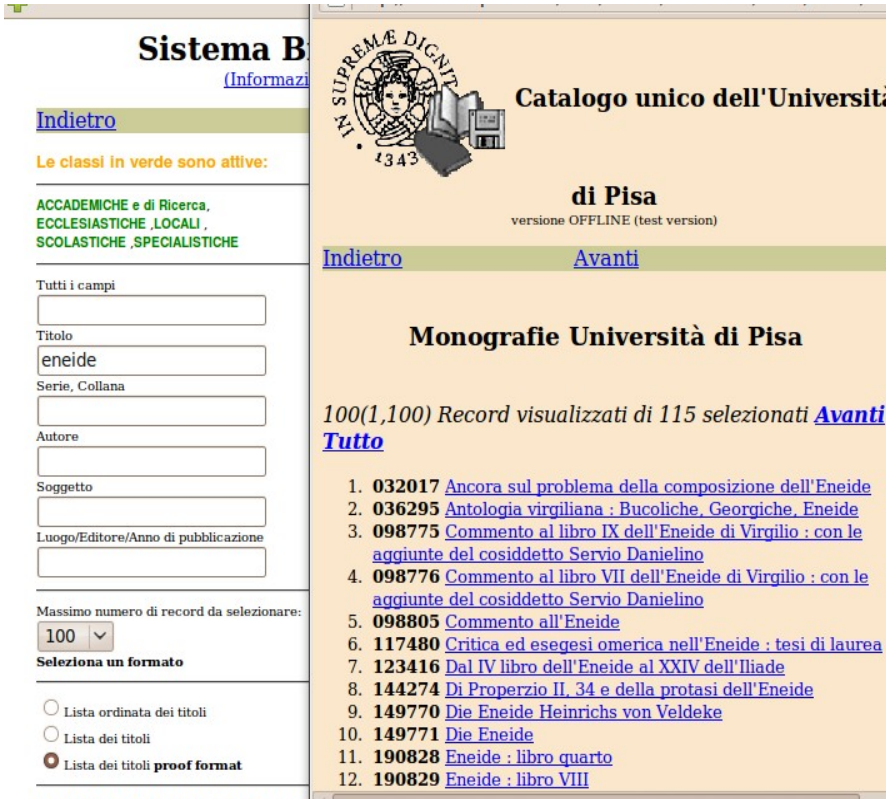


Immagine 12: Risultati della ricerca: Lista dei titoli proof format

L'opzione che permette di scegliere il formato del titolo, dato che non è chiarissima ed è specifica per i singoli cataloghi, nella nuova 'interfaccia non verrà proposta. Infatti il catalogo unico per la sua natura prescinde dalle specificità dei singoli cataloghi, per cui anche l'ordine di catalogazione originale viene sostituito dal più generico ordine alfabetico.

Alla luce delle considerazioni fatte, è sembrato opportuno gestire diversamente gli spazi destinati ai campi di ricerca per cercare di dare pari rilievo ai tre metodi esistenti, e per far si che l'utente, davanti alla possibilità di scegliere diverse vie per interrogare il catalogo, rifletta qualche attimo su quale sia la modalità migliore per il suo scopo.

I campi per le interrogazioni sono stati spostati dalla colonna laterale alla parte superiore della pagina, e sono divisi verticalmente in due blocchi distinti. Ciascun blocco ospita i campi necessari per il tipo di ricerca che propone. Dai campi a sinistra si può utilizzare il metodo *list*, per ottenere elenchi dei campi selezionati. Con un radio button l'utente può scegliere il campo da interrogare, e alla destra di questo si attiverà un campo di input in cui digitare il termine da ricercare. Infine un menù a tendina permetterà di scegliere le biblioteche su cui effettuare le ricerche.

La sezione di destra invece permette di svolgere ricerche di tipo search. I diversi campi di ricerca verranno combinati tra loro con l'operatore di default “and” e anche qui da un menù a tendina sarà possibile selezionare le biblioteche.

Per la ricerca a testo libero è stato ricavato un apposito spazio sotto i due blocchi del *search* e del *list.* Solo un campo di input nel quale inserire il testo da cercare su tutti i campi di ricerca e il pulsante di invio rendono l'interfaccia simile ai motori di ricerca sul web.

Con questa struttura, l'utente che accede a questa interfaccia subito ha davanti a sé, senza gerarchie, tutti i sistemi di ricerca che il sistema mette a sua disposizione.

Inoltre questo tipo di organizzazione dello spazio permette di avere il resto della pagina disponibile per la visualizzazione dei risultati della ricerca.

Un aspetto importante per quanto riguarda l'impostazione dell'interfaccia è la terminologia da utilizzare. Come evidenziano gli autori di una ricerca sugli opac italiani[15], di cui si è già parlato in 1.1., esiste una grande varietà nei termini utilizzati per indicare i diversi tipi di ricerca. Questo comporta una disomogeneità nel linguaggio e costringe l'utente a dover ogni volta interpretare il significato di determinati termini nel contesto del singolo opac. Questa molteplicità di diciture sembra essere dovuta alla difficoltà della traduzione dei termini inglesi *search* e *list* (o *scan*), parole che nella lingua d'origine hanno un senso ben preciso, ma che in italiano spesso non si sa bene come tradurre per cui talvolta si ricorre a complesse perifrasi.

Alcuni esempi tratti dall'articolo (v. nota 15) potrebbero chiarire questo problema.

L'Im. 13 mostra come, su 112 volte che la funzione di ricerca *Scan* (o *list*) viene denominata, questa abbia 16 diversi nomi (quelle poco frequenti o presenti una sola volta sono state accorpate sotto la voce “altro”), alcune abbastanza simili tra loro (lista, liste, ricerca liste, scorri liste, che derivano da una traduzione letterale dell'inglese *list*), altre invece completamente diverse, di difficile interpretazione o addirittura errate (sfoglia, browse, trova ecc...). Si comprende quindi le possibili difficoltà di un utente che in un ambiente effettua una “ricerca per liste” mentre in un altro opac per avere lo stesso risultato deve cercare gli “indici”.

15 Ridi, Gnoli, Visitin, 2004, “*Come vogliamo chiamarli? Operatori booleani e altre tecniche di information retrieval negli opac italiani*”.

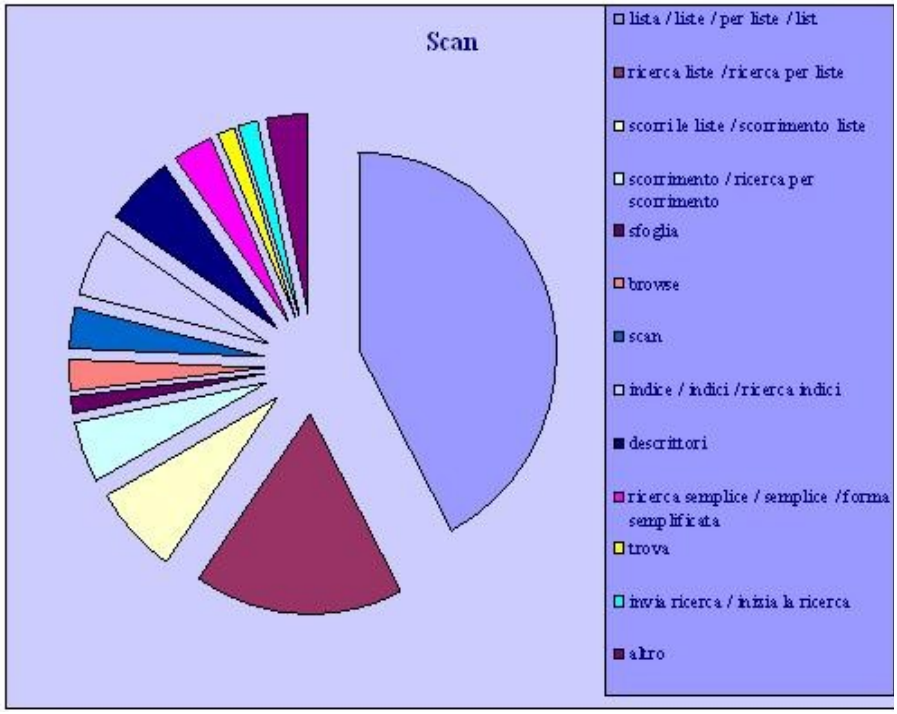


Immagine 13: Termini usati per denominare la ricerca di tipo Scan (o list). Fonte: Ridi, Gnoli, Visitin, 2004, "Come vogliamo chiamarli? Operatori booleani e altre tecniche di information retrieval negli opac italiani".

Anche la nuova interfaccia deve avere una denominazione per questo tipo di ricerca, e la scelta del termine da utilizzare è stata fatta cogliendo un suggerimento degli autori della ricerca sopra citata: "Da notare, infine, che nessuno abbia pensato di tradurre 'list' con il più elegante termine 'elenco', preferendo piuttosto adagiarsi sul pedissequo calco 'lista' o inerpicarsi verso degli improbabili 'indici' " (Ridi, Gnoli, Visitin, 2004). Da questo spunto la denominazione scelta è stata appunto "ricerca elenchi"

Stesso tipo di analisi è stata svolta per l'altro metodo di ricerca, il *Search*.

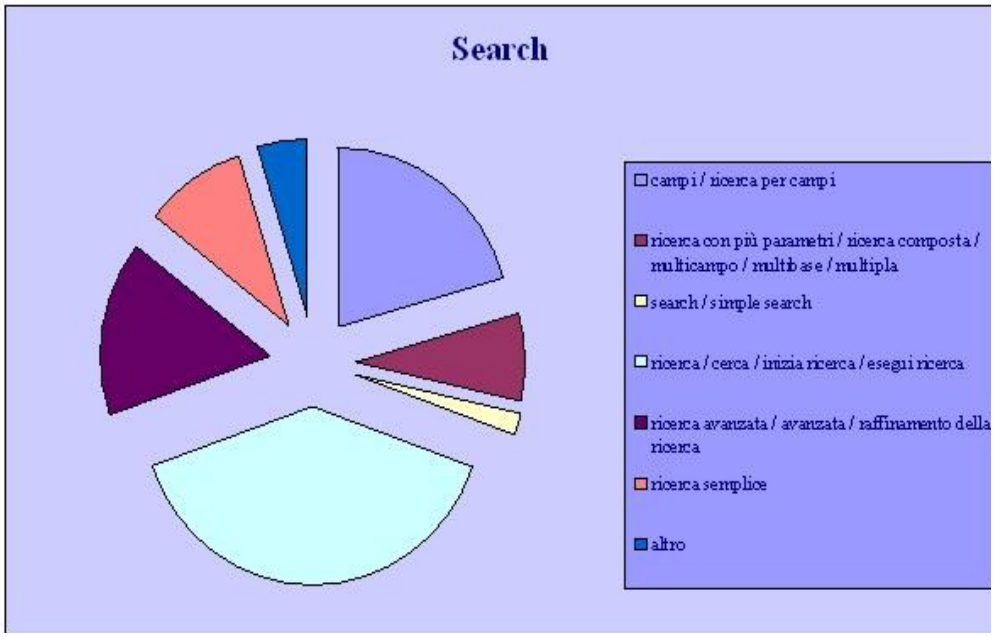


Immagine 14: Termini usati per denominare la ricerca di tipo Search. Fonte: Ridi, Gnoli, Visitin, 2004, "Come vogliamo chiamarli? Operatori booleani e altre tecniche di information retrieval negli opac italiani".

Anche in questo caso un grafico (Im. 14), evidenzia la distribuzione delle nomenclature per questo metodo di ricerca. Anche qui si è davanti a una varietà di nomi, alcuni chiari e corretti (campi, ricerca per campi, search) altri invece un po' meno (ricerca guidata, combinazione; nel grafico queste denominazioni sono incorporate nella voce altro). Anche qui la difficoltà sta nella traduzione di un concetto ben specifico in inglese e che non ha una parola corrispondente in italiano. Sempre in base alle osservazioni degli autori della ricerca, si è scelto di denominare la ricerca di tipo *search* con "Ricerca su più campi", evidenziando la differenza con la "ricerca elenchi" che può essere svolta su un unico campo.

Infine la ricerca in ambiente testuale. Anche in questo caso l'articolo fornisce delle statistiche sui diversi nomi che assume questo tipo di ricerca.

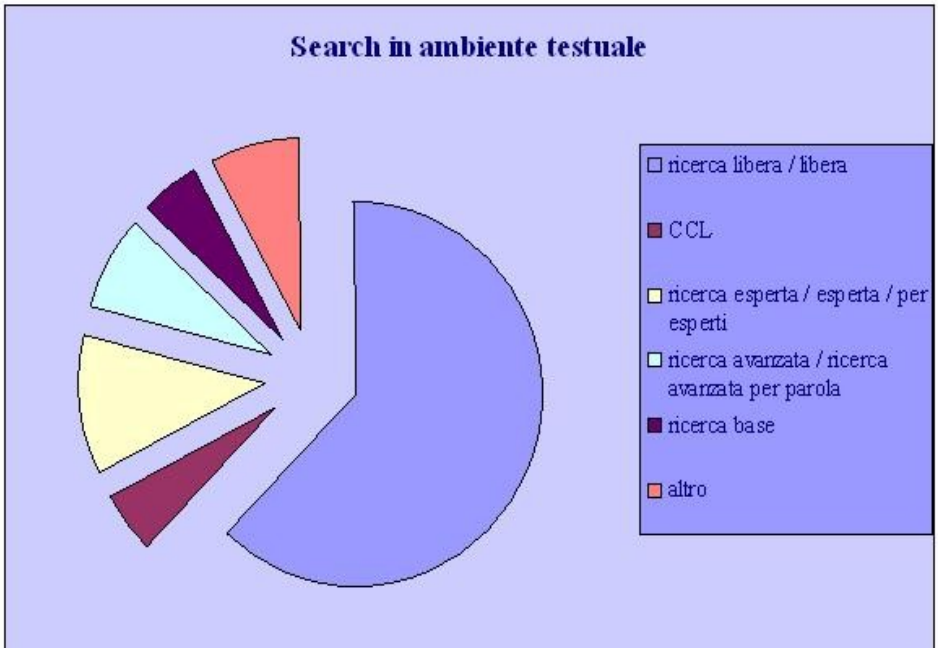


Immagine 15: Termini usati per denominare la ricerca in ambiente testuale. Fonte: Ridi, Gnoli, Visitin, 2004, "Come vogliamo chiamarli? Operatori booleani e altre tecniche di information retrieval negli opac italiani".

L'articolo suggerirebbe di usare, anziché una delle espressioni presentate nell'Im. 15, il termine "equazione" o "espressione". In questo caso però si è scelto di non seguire il consiglio degli autori, perché per quanto il senso di queste due parole sia corretto rispetto al tipo di ricerca che si deve effettuare, non è tuttavia ben comprensibile per gli utenti maggiormente abituati ad espressioni del tipo "ricerca libera". Il rischio di usare termini tipo "equazione" o "espressione" potrebbe essere quello di allontanare gli utenti non esperti da questo campo di ricerca. Quindi si è scelto di adottare l'espressione più comune "ricerca libera".

Terminata la progettazione delle sezioni dell'interfaccia destinate alla ricerca, si può passare alla visualizzazione dei risultati.

Come si è detto i risultati delle interrogazioni al catalogo saranno visualizzati in ordine alfabetico. Quali dati visualizzare è determinato dal tipo di ricerca effettuata, ma la tendenza generale è quella di ridurre al minimo i passaggi che l'utente deve fare per arrivare a consultare le informazioni sull'opera.

Il tipo di ricerca *list*, come dice già il nome, prevede la visualizzazione di liste. Quindi

quando la ricerca viene svolta con questo metodo i risultati saranno mostrati sotto forma di elenchi. Quando l'utente avrà scelto un elemento della lista, verranno aperte le schede bibliografiche correlate.

La ricerca per lista può essere effettuata in base al titolo, all'autore e alla classificazione: quella per titolo mostrerà direttamente l'elenco dei titoli con le informazioni correlate, quella per autore l'elenco degli autori e, dopo la scelta di un autore, la lista delle opere a lui collegate, e infine la classificazione, che visualizzerà le classi Dewey sia in formato numerico sia nell'equivalente verbale e permetterà di navigare tra i vari elementi fino alla visualizzazione dei titoli con il codice selezionato.

La lista delle classi Dewey ha un'altra funzione aggiuntiva: accanto al codice e alla descrizione infatti è presente un link alla pagina della documentazione ufficiale di cui si è parlato nel cap. 4.3., così che un utente non esperto della classificazione possa usare i servizi messi a disposizione dall'OCLC per trovare una determinata classe di argomenti.

Il metodo *search* e la ricerca a testo libero invece restituiscono l'elenco delle opere selezionate in base ai criteri specificati.

Per ogni opera che viene selezionata dal database vengono mostrati titolo, autori e una lista di biblioteche che la possiedono insieme alle informazioni sull'edizione posseduta.

Quando l'utente decide di avere maggiori informazioni su una determinata opera in possesso di una precisa biblioteca, attraverso un link può accedere direttamente alla scheda bibliografica del catalogo della biblioteca di appartenenza. Così il processo di ricerca di un'opera si conclude proprio dove l'opera è conservata, ma l'utente ha saltato, rispetto alla ricerca nel metaopac, tanti passaggi intermedi.

7.3. I puntatori alle schede bibliografiche di origine

La questione del link che dall'interfaccia del catalogo unico porta direttamente alla scheda bibliografica della singola biblioteca è stato oggetto di diverse riflessioni. Infatti le soluzioni ipotizzate per creare questo collegamento sono state varie. Inizialmente si è pensato di memorizzare l'id dell'opera nel catalogo di provenienza per poi costruire una query al database del catalogo per ottenere la scheda, ma questa soluzione non stata implementata perché si corre il rischio di perdere sincronia tra il catalogo unico e i

singoli cataloghi; infatti dopo l'aggiornamento del catalogo una singola biblioteca potrebbe passare del tempo (da un'ora a un paio di giorni) prima che anche il catalogo unico venga riaggiornato; quindi se la scheda bibliografica di un'opera dovesse cambiare id, non verrebbe più trovata fino al successivo aggiornamento. Per questo motivo si è cercato di studiare un meccanismo di creazione del puntatore al catalogo che fosse indipendente dall'id dell'opera e generico quanto basta per poter gestire interrogazioni a cataloghi strutturati diversamente.

Si è quindi pensato di sfruttare i sistemi di interrogazione dei cataloghi del metaopac. Questo elabora delle query ai singoli cataloghi, imposta dei parametri che specificano il tipo di ricerca e il modo di visualizzare i risultati direttamente nella url della pagina.

Dal catalogo unico si può riprodurre la url impostata dal metaopac per visualizzare la scheda bibliografica ricercata. Questa ovviamente non punterà al catalogo attraverso l'identificativo dell'opera, visto che nel catalogo unico non viene memorizzata, ma si costruirà una query di ricerca dell'opera nel catalogo della biblioteca passando come criteri di ricerca tutte le informazioni disponibili nel catalogo unico, quindi titolo, autore e pubblicazione. Così si sfruttano funzioni già esistenti e collaudate e si risparmia tempo e codice. Il modo in cui viene creata la url che punta alla scheda bibliografica del catalogo è semplice. Innanzi tutto su un file di configurazione esterno si impostano i link dei singoli cataloghi (cod. 9). Nei punti del link dove dovrebbero andare i vari parametri della ricerca, come titolo autore e pubblicazione, vengono messe delle variabili: %1$s per il titolo,%2$s per l'autore, %3$s per l'edizione. La funzione `sprintf($value,$tit,$auth,$pub)` sostituisce ai parametri della stringa $value, che assumerà di volta in volta il valore della stringa di configurazione del catalogo, le variabili $tit, $auth, $pub (rispettivamente titolo autore e pubblicazione). In questo modo si possono gestire i puntatori alle schede bibliografiche di cataloghi diversi semplicemente posizionando i parametri nei punti del link dove andrebbero titolo, autore e pubblicazione.

```
$array_bib =array();

$unipi ="http://biblio.unipi.it:8080/TIsis/servlet/Isis?
Conf=/home/admin/unipi/Conf/unipi.sys.file&Obj=@unipiSh.pft&Opt=
search&Field1=%1\$s&Field2=%2\$s&Field4=%3\$s&";


$array_bib['unipi']=$unipi;

foreach($array_bib as $k => $value){
      if($k == $cat){

      $val = sprintf($value,$tit,$auth,$pub);
      return $val;

      }

      }

$unipi è la variabile di configurazione del database
dell'università di Pisa.

$array_bib è l'array che contiene tutte le variabili di
configurazione.
```

Codice 9: File di configurazione dei link alle schede bibliografiche dei cataloghi.

Conclusioni

Il lavoro qui presentato non è la conclusione di un progetto, anzi, è l'inizio di un processo che mira a rinnovare il sistema di gestione e consultazione dei cataloghi delle biblioteche di area pisana.

Alla base del progetto infatti c'era l'idea di creare qualcosa di nuovo, che riuscisse a collegare tra loro in modo organico realtà che finora sono rimaste, nell'organizzazione, separate l'una dall'altra, e nella consultazione semplicemente affiancate, ma mai integrate fra loro. L'idea, anzi l'utopia, è che le biblioteche dell'area di Pisa inizino a pensare il proprio catalogo alla luce delle nuove possibilità offerte dall'esistenza di un catalogo unico che riunisca tutte le biblioteche dell'area, curando le proprie schede bibliografiche, classificando le opere ancora prive di codice DDC, prestando attenzione alla correttezza dei titoli, per far si che non solo all'interno del singolo catalogo, ma in tutta la rete delle biblioteche esista una coerenza nella gestione dei dati e una informazione il più completa possibile. Il catalogo unico potrebbe anche diventare uno strumento di supporto alla compilazione di una scheda bibliografica. Infatti un bibliotecario, cercando l'opera che deve catalogare nel catalogo unico, potrebbe scoprire che l'opera esiste già in un catalogo, ha un titolo e degli autori scritti con una certa ortografia e avere anche il codice di classificazione Dewey. Se ritiene corrette le informazioni presenti nel catalogo unico potrebbe uniformare la compilazione della propria scheda bibliografica a quanto già presente in catalogo, oppure contattare la biblioteca che possiede l'opera per concordare un modello di compilazione delle schede.

Le biblioteche, alla luce delle possibilità di sviluppo della ricerca in senso semantico, potrebbero arricchire le proprie schede con una descrizione dei soggetti dell'opera, oltre che estendere la classificazione Dewey a tutte le opere che non ce l'hanno e che per questo sono escluse dal catalogo unico.

In questo modo questo strumento potrebbe ancora crescere lungo la strada dell'opac semantico, mettendo a disposizione nuovi tipi di ricerca più orientati al contenuto del libro che ai metadati.

Dal lato del sofware invece progressi possono essere fatti affinando le procedure di confronto dei titoli. Con il supporto di chi si occupa quotidianamente della gestione dei

cataloghi, si potrebbero sviluppare delle procedure ancora più specifiche che potrebbero anche servire per correggere i dati di partenza qualora fossero presenti degli errori.

Le operazioni di accorpamento delle opere, registrate in appositi file di log, potrebbero essere inviate ai bibliotecari delle varie strutture per comunicare loro quali opere possiedono in comune con alte biblioteche e quali variazioni di titoli sono state rilevate, così da poter intervenire direttamente sui dati di partenza.

La condivisione delle informazioni infine è un altro punto di sviluppo fondamentale per il catalogo unico. Nella realtà di oggi non si può più pensare che le informazioni rimangano ristrette a un certo ambito, specialmente quando si tratta di web. Il metaopac pisano ad esempio potrebbe essere consultato da utenti provenienti da tutta Italia, e quindi bisogna essere sempre pronti a fornire, a chiunque acceda al sistema, informazioni complete e comprensibili. Allo stesso modo, chiunque potrebbe essere interessato ad avere una copia del catalogo unico, magari per aggregarlo ad altri cataloghi o semplicemente per renderlo consultabile anche da altre piattaforme.

La condivisione delle informazioni, così come la loro comprensione ,spesso si basa sulla condivisione di un codice di comunicazione. In ambito informatico questi codici sono definiti dagli standard. Cercare di avvicinare il catalogo unico ai più diffusi standard è un altro obiettivo di questo lavoro. Già ora si esporta in catalogo in formato xml-TEI, ma si può fare di più, magari creando sistemi di conversione da un sistema di rappresentazione della conoscenza a un altro, in modo che chiunque possa accedere ai dati del catalogo secondo le proprie esigenze.

Per concludere le riflessioni sulle possibilità di sviluppo del catalogo e sulla sua utilità si riportano alcuni brani di un articolo di Simona Turbanti, direttrice della biblioteca di chimica, che affronta questioni relative alla bonifica e alla qualità dei cataloghi (Simona Turbanti, 2007, “La bonifica del catalogo e il controllo di qualità: strumenti, tempi, strategie”)

> “(...) Strettamente connessa è la questione, meno rara in letteratura professionale, del controllo di qualità del catalogo stesso (o "monitoraggio"). Difficile capire come mai ci si sia soffermati poco diffusamente su tematiche che dovrebbero interessare molte biblioteche e sistemi bibliotecari. Forse la difficoltà di trovare procedure uniformi da adottare nel corso di queste

operazioni e il carattere pragmatico delle stesse ha indotto al quasi-silenzio, lasciando alla pratica di ciascuna biblioteca il compito di organizzarle e gestirle al meglio. In tal modo, però, viene meno la possibilità di condividere esperienze che potrebbero essere utili e spendibili in realtà diverse. È probabile altresì che si tratti di un intervento sul quale le strutture hanno sino ad ora investito relativamente poco, seguendo la logica di privilegiare la quantità di materiale bibliografico visibile sugli opac rispetto alla qualità delle schede catalografiche che quel materiale descrivono. (…)

L'esigenza di interventi correttivi può esistere però anche in presenza di un catalogo sorto non "dalle ceneri" di altre base dati, ma sviluppatosi nel tempo in modo poco organico a causa di massicce derivazioni o catture di notizie bibliografiche da risorse esterne e/o di conversioni retrospettive. Questo genere di approvvigionamento di dati comporta infatti una percentuale inevitabile di errori, quando non di duplicazioni di notizie, entrambi difficilmente evitabili *a priori* per la qualità molto variabile delle fonti da cui si importa nel primo caso e per gli inevitabili limiti dei programmi di schiacciamento automatico nel secondo.(...)

Se è innegabile, infatti, che registrazioni bibliografiche corrette *in toto* siano auspicabili in qualunque contesto, è pur vero che, a seconda del bacino di utenza istituzionalmente servito da una biblioteca e dal proprio catalogo, varino alcuni fattori, quali ad esempio la priorità di alcuni punti di accesso nei confronti di altri. D'altro canto, queste "priorità locali" sono vanificate dall'illimitatezza del Web che, rendendo visibile e accessibile l'opac a qualunque utente esterno e remoto imporrebbe ai cataloghi *on-line* la maggior correttezza e precisione possibili.(...)

Se lo scopo della bonifica è (...) quello di sanare *una tantum* una situazione di dati errati o incompleti, quello del monitoraggio consiste invece nell'evitare *a priori* che tale situazioni si verifichi. Più accurata sarà la "prevenzione" degli errori, meno tempo ed energie dovranno essere dedicati agli interventi correttivi *a posteriori* che, oltre ad essere costosi per le biblioteche, a seconda dei criteri scelti non sempre assicurano un risultato esente da lacune o difetti. (...)

La bonifica da una parte, il controllo di qualità dall'altra, uniti alla formazione continua e alla gestione consapevole del personale incaricato di operare a vario titolo e livello nel catalogo, costituiscono quindi una strada percorribile per

> salvaguardare quest'ultimo, almeno parzialmente, da un accrescimento incondizionato e disomogeneo e da quella che definirei "usura del tempo" a cui ogni strumento bibliografico è soggetto.”

I brani scelti da questo articolo fanno un riassunto del contesto nel quale il programma di creazione del catalogo unico ha operato.

La scelta di affidare alle parole di una bibliotecaria che opera quotidianamente nell'ambito del catalogo più ampio dell'area pisana la conclusione di questo lavoro vuole servire per dare da un lato la conferma che tutte le situazioni di difficoltà descritte nei vari capitoli sono oggettive, dall'altro per sottolineare quanto sia necessario operare una correzione dei cataloghi. Che questa operazione sia stata fatta non sui singoli cataloghi ma sulla loro unione fa poca differenza: lo scopo finale è lo stesso, avere dati di qualità, destinati non alla comunità “locale” ma al vasto mondo del web. Se il programma creato potrà essere di una qualche utilità per aiutare i bibliotecari con le operazioni di bonifica del catalogo sarà da vedere, ma le intenzioni sono queste e i tentativi di miglioramento andranno di sicuro in questa direzione.

Bibliografia

Gnoli Claudio, Ridi Riccardo, Visintin Giulia, 2004. *Di che parla questo catalogo?* . In: *Biblioteche Oggi*, anno 22, n. 8. p 23-29.

Gnoli Claudio, 2005. *Opac Semantici? Sprechi e potenzialità negli accessi per classe*. Atti dell'8° workshop "Teca del Mediterraneo".: Bari p 123-130.

McLaughling Brett, *Java e Xml,* Apogeo Editore, 2001.

Periodici elettronici

Barazia Caterina, 2004. *Opac semantici: indagine sugli accessi semantici nei cataloghi in rete italiani.* In *Aida Informazioni*, anno 22, n. 4 .

Ridi Riccardo , Gnoli Claudio, Visintin Giulia, 2004. *Come vogliamo chiamarli? Operatori booleani e altre tecniche di information retrieval negli opac italiani.* In *Bibliotime*, anno VII, n. 3.

Turbanti Simona, 2007. *La bonifica del catalogo e il controllo di qualità: strumenti, tempi, strategie.* In *Bollettino Aib*, 2007 n. 4 p. 451-458

Siti web

I cataloghi elettronici delle biblioteche. Tesi di Lucia Tronchin

indirizzo: http://www.burioni.it/forum/tronchin/introduzione.htm

Aib-Web. Contributi. Presentazione del catalogo e tecnica d'interrogazione

indirizzo: http://www.aib.it/aib/contr/digirolamo2.htm

OCLC web site

indirizzo: http://www.oclc.org/dewey/webservices/default.htm

Algoritmo per calcolare la distanza di Levenstein

indirizzo: http://mariano.altervista.org/wordpress/2008/09/11/forse-cercavi-distanza-di-levenshtein/

Sito ufficiale TEI

indirizzo: http://www.tei-c.org/

Appendice 1

Porzioni di codice del software per la creazione del catalogo unico

Codice relativo alla classe Work()

```
/* definisce un oggetto Work che contiene le informazioni relative
alle opere */

public class Work  {
      private String title;
      private String cdd;
      private String cdd_high;
      private String publication;
      private List<String> authors;
      private String id;
      private List<String> biblioteca;
      private List<String> idWorkAggregate = new ArrayList<String>();
```

Codice relativo alla classe SaxParserExample ().

```
//Questo  metodo  implementa il SAX parser di Java per il parsing dei
file Xml.
private void parseDocument() {
            //get a factory
            SAXParserFactory spf = SAXParserFactory.newInstance();

            try {
                  //get a new instance of parser
                  SAXParser sp = spf.newSAXParser();

//parse the file and also register this class for call backs
                  sp.parse(fileStre, this);

            }catch(SAXException se) {
                  se.printStackTrace();
```

```
            }catch(ParserConfigurationException pce) {
                  pce.printStackTrace();
            }catch (IOException ie) {
                  ie.printStackTrace();
            }

      }

//Event Handlers. Il sax parser si basa sugli eventi start element
//character ed end element

public void startElement(String uri, String localName, String qName,
Attributes attributes) throws SAXException {
            //reset
            tempVal = "";
            if(qName.equalsIgnoreCase("record")) {
                  //create a new instance of employee
                  tempWork = new Work();
                  authors = new ArrayList<String>();
                  tempWork.setBiblioteca(biblioteca);
            }
      }

public void characters(char[] ch, int start, int length) throws
SAXException {
            tempVal = new String(ch,start,length);
      }

/* in base all'elemento dell'xml che si sta analizzando memorizza i
dati in un oggetto Work()*/
public void endElement(String uri, String localName, String qName)
throws SAXException {

            if(qName.equalsIgnoreCase("record")) {

                  try {
                        if (Isnumeric(tempWork.getCdd_high())){
                        insertDDC();
                        }
```

```
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                //authors.removeAll(authors<String>);
        }else if (qName.equalsIgnoreCase("title")) {
                tempWork.setTitle(tempVal);
        }else if (qName.equalsIgnoreCase("pubblicazione")) {
                tempWork.setPublication(tempVal);
        }else if (qName.equalsIgnoreCase("value")) {
                tempWork.setCdd(tempVal);
        }else if (qName.equalsIgnoreCase("high_level_value")) {
                tempWork.setCdd_high(tempVal);
        }else if (qName.equalsIgnoreCase("author")) {
                authors.add(tempVal);
        }else if (qName.equalsIgnoreCase("authors")) {
                tempWork.setAuthors(authors);
        }else if (qName.equalsIgnoreCase("records")) {
                        System.out.print("");
        }else if (qName.equalsIgnoreCase("library")) {
                biblioteca = tempVal;
        }
}
```

```
/* Questo metodo inserisce nelle tabelle temporanee del database le
opere lette dall'xml /*

private void insertDDC() throws SQLException {
      //apro la transazione
      this.connection.beginTransaction(false);
      String numcat = "";

      String tit=StringMatching.togliApici(tempWork.getTitle());
      String pub="";
      if(tempWork.getPublication()!=null){
            pub= StringMatching.togliApici(tempWork.getPublication());
      }

      numcat = (tempWork.getCdd_high().charAt(0))+"";
      String query = "INSERT INTO DDC_"+ numcat+"00"  +
            " (title,publication,CDD_first,CDD,biblioteca) VALUES" +
            "('" +tit+"','" +pub+"','" +tempWork.getCdd_high()+"','"
            +tempWork.getCdd() +"','"+ biblioteca +"');";

      int id_work = this.connection.insertRecord(query,"DDC_"+ numcat
      +"00");
      if (id_work == -1){
            System.exit(0);
      }
      List<Integer> id_authors ;

      if(tempWork.getAuthors()!=null){
            Iterator itr = tempWork.getAuthors().iterator();
            while(itr.hasNext()) {
                  //***normalizzazione autore
                  String author= itr.next().toString();
                  String[] temp = author.split(",",2);
                  String name="";
                  String iniz="";
                  String surname="";
                  StringMatching check = new StringMatching();

                  if(temp[0].equals("")){
                        surname = StringMatching.togliApici(temp[1]);
```

```
                  temp[1]="";
            }
            else{
            surname = StringMatching.togliApici(temp[0]);
                  }
            if((temp.length>1)&&(!temp[1].equals("")) ){
                  name=StringMatching.togliApici(temp[1]);
                  name=name.replaceAll("\\."," ");
                  if(temp[1].trim().length()>1){
                        iniz=check.GetIniziale(name);
                  }
                  else{
                        iniz=temp[1].trim();
                        }
            }
//***normalizzazione autore
//inserimento  autori
int id_Aut;
String pk_auth = surname +"_"+ iniz + "_" + name;

String query_aut = "INSERT INTO authors(surname, name,
name_ext,pk_auth) VALUES
('"+surname+"','"+iniz+"','"+name+"','"+pk_auth+"')";

id_Aut = this.connection.insertRecord(query_aut,"Authors");

if (id_Aut == -99) {
      System.out.println("Autore Doppio");
      String ricerca_auth = "SELECT id from authors where
      pk_auth='"+ pk_auth +"'";

Vector<String[]> aut =
this.connection.eseguiQuery(ricerca_auth);

String[] riga = aut.elementAt(0);
id_Aut = Integer.parseInt(riga[0].toString());
}
if (id_Aut == -1) {
      System.exit(0);
}
```

```
//Inserisco in coda la riga in auth_work sempre in transazione
      query_aut = "INSERT INTO aut_work_temp(work, author,ddc) VALUES
      ("+ id_work +","+ id_Aut +",'" + numcat +"')";

      int ritorno=
      this.connection.insertRecord(query_aut,"Aut_work_temp");

      if (ritorno==-1){
            System.exit(0);
      }

 }
            }
            //scrivo effettivamente sul DB
            this.connection.doCommit();
      }
```

Codice relativo alla classe StringDistance ().

```
/* Questa classe calcola la distanza di Levenstein su due stringhe
public class StringDistance {

/** Questo metodo calcola il minimo tra tre numeri interi*/

      private int getMinimo(int a, int b, int c) {
            int min;
            min = a;

            if (b < min) {
                  min = b;
            }

            if (c < min) {
                  min = c;
            }
            return min;
            }
```

```
/** Funzione per il calcolo effettivo della distanza di Levenshtein*/

      int getDistanza(String s, String t) {
            int d[][];     // matrice
            int n;         // lunghezza di s
            int m;         // lunghezza di t
            int i;         // iterazioni su s
            int j;         // iterazioni su t
            char s_i;      // i-esimo carattere di s
            char t_j;      // j-esimo carattere di t
            int costo;

            n = s.length(); // n conterrà il num di caratteri di s
            m = t.length(); // m conterrà il num di caratteri di t
            // se la stringa sorgente è vuota
            if (n == 0) {
                  return m; // la distanza è il num di chr della dest.
            }
            // se la stringa destinazione è vuota
            if (m == 0) {
                  return n; // la distanza è il num di chr della sorg.
            }
            // creo la matrice (n+1)*(m+1)
            d = new int[n + 1][m + 1];

// la prima riga della matrice conterrà le distanze da 0 a n
// la distanza 1 è associata al primo chr della stringa, 0 al vuoto

            for (i = 0; i <= n; i++) {
            d[i][0] = i;
            }

// la prima colonna della mat. conterrà le distanze da 0 a m
// la distanza 1 è associato al primo chr della stringa, 0 al vuoto

            for (j = 0; j <= m; j++) {
            d[0][j] = j;
            }
```

```
        // esamino ogni carattere di s (i da 1 a n)
        for (i = 1; i <= n; i++) {
        s_i = s.charAt(i - 1);

        // Esamino ogni carattere di t (j da 1 a m)
        for (j = 1; j <= m; j++) {
        t_j = t.charAt(j - 1);

//Se l'i-esimo elemento di s è uguale al j-esimo elemento di t
        if (s_i == t_j) {// il costo è 0
        costo = 0;
        } else { // altrimenti il costo è 1
        costo = 1;
        }

// imposto la cella d[i][j] scegliendo il valore minimo tra:
//   - la cella immediatamente superiore + 1
//   - la cella immediatamente a sinistra + 1
//   - la cella diagonalmente in alto a sinistra più il costo
        d[i][j] = getMinimo(d[i - 1][j] + 1, d[i][j - 1] + 1, d[i
        - 1][j - 1] + costo);
        }
        }

// una volta completate tutte le iterazioni la cella
// d[n][m] contiene la distanza finale tra la stringa
// sorgente e quella di destinazione
        return d[n][m];
        }
}
```

Codice relativo alla classe StringMatching()

```
// metodo della classe StringMatching() che gestisce il confronto fra
i titoli
private boolean Compare() {
            int dist;
            String[]results = null;
            String title1 = this.input.getTitle();
            String title2 = this.selected.getTitle();

            //primo confronto delle stringhe basato solo sul calcolo
            //della distanza di Levenstein; dist è la distanza tra le
            //stringhe da confrontare

            dist = DirectControl();

            //controllo se DirectControl è andato a buon fine o se
            //bisogna continuare i controlli;

            if(dist==0){  // i titoli sono uguali
                  return true;
                  }
            else if(dist==-1){ // i titoli sono diversi
                  return false;
            }
            else {
                  if(dist>900){
// Per indicare che si deve lavorare sulle stringhe normalizzate viene
//aggiunto alla distanza calcolata 900.
                        dist=dist-900;
                        title1=Normalizza(title1);
                        title2=Normalizza(title2);
                  }

//si misura la lunghezza delle stringhe senza spazi per stabilire
//l'ordine con cui passare i parametri ai metodi successivi
                  String a = NoBlankSpace(title1);
                  String b = NoBlankSpace(title2);
```

```
            if (a.length() > b.length()) {
            // il metodo CheckSimilarity prende come primo parametro
            //la stringa più lunga

                  results = CheckSimilarity(title1, title2, dist);

            } else if (a.length() < b.length()) {
                  results = CheckSimilarity(title2, title1, dist);

            } else {
// Le stringhe hanno uguale lunghezza controllo direttamente i numeri
//finali dei titoli
                  if(finalNumbers(results[0], results[1])){
                        return false; //se final number restituisce
                                    //true le opere non sono uguali
                  }
                        else return false;
                  }

/*fine valutazione di DirectControl

CheckSimilarity verifica che non siano presenti numeri all'inizio dei
titoli. Se alla fine del confronto restituisce un Array vuoto i titoli
confrontati sono uguali altrimenti restituisce un Array contenente le
stringhe dei titoli eventualmente corrette */

            if((results!=null)&&(results.length==0)){
                  return true;
            }

            else if (finalNumbers(results[0], results[1])){
                  return true;
            }
//infine ultimo confronto se finalNumbers non ha dato risultati eseguo
DeleteWords
            else{
                  if(DeleteWords(results[0], results[1])) {
                        return true;
                   } else return false;
      }     }     }
```

Appendice 2

Questa appendice riporta alcuni esempi dei confronti fra i titoli

Appendice 3

Documentazione TEI relativamente agli elementi usati per la codifica del catalogo unico.

<ab> (anonymous block) contains any arbitrary component-level unit of text, acting as an anonymous container for phrase or inter level elements analogous to, but without the semantic baggage of, a paragraph.

Module linking

Used by «model.pLike»

May contain

core: «abbr» «add» «address» «bibl» «biblStruct» «binaryObject» «cb» «choice» «cit» «corr» «date» «del» «desc» «distinct» «email» «emph» «expan» «foreign» «gap» «gloss» «graphic» «hi» «index» «label» «lb» «list» «listBibl» «measure» «measureGrp» «mentioned» «milestone» «name» «note» «num» «orig» «pb» «ptr» «q» «quote» «ref» «reg» «rs» «said» «sic» «soCalled» «stage» «term» «time» «title» «unclear»

header: «biblFull» «idno»

linking: «alt» «altGrp» «anchor» «join» «joinGrp» «link» «linkGrp» «seg» «timeline»

Declaration

```
element ab
{
att.global.attributes,
att.typed.attributes,
att.declaring.attributes,
attribute part { "Y" | "N" | "I" | "M" | "F" }?,
macro.paraContent}
```

```
<div type="book" n="Genesis"
xmlns:ns186="http://www.tei-c.org/ns/1.0">
<div type="chapter" n="1">
<ab>In the beginning God created the heaven and the earth.</ab>
<ab>And the earth was without form, and void; and darkness was
upon the face of the deep. And the spirit of God moved upon the
face of the waters.</ab>
<ab>And God said, Let there be light: and there was light.</ab>
<!- ...->
</div>
</div>
```

Example

<author> in a bibliographic reference, contains the name(s) of the author(s), personal or corporate, of a work; for example in the same form as that provided by a recognized bibliographic name authority.

Module core

Used by «analytic» «monogr» «model.respLike»

May contain

core: «abbr» «add» «address» «binaryObject» «cb» «choice» «corr» «date» «del» «distinct» «email» «emph» «expan» «foreign» «gap» «gloss» «graphic» «hi» «index» «lb» «measure» «measureGrp» «mentioned» «milestone» «name» «note» «num» «orig» «pb» «ptr» «ref» «reg» «rs» «sic» «soCalled» «term» «time» «title» «unclear»

header: «idno»

linking: «alt» «altGrp» «anchor» «join» «joinGrp» «link» «linkGrp» «seg» «timeline»

Declaration

```
element author
{
att.global.attributes,
att.naming.attributes,
macro.phraseSeq}
```

```
<author xmlns:ns213="http://www.tei-c.org/ns/1.0">British
Broadcasting Corporation</author>
<author>La Fayette, Marie Madeleine Pioche de la Vergne, comtesse
de (1634-1693)</author>
<author>Anonymous</author>
<author>Bill and Melinda Gates</author>
<author>
<persName>Beaumont, Francis</persName> and
<persName>John Fletcher</persName>
</author>
<author>
<orgName key="BBC">British Broadcasting
Corporation</orgName>: Radio 3 Network
</author>
```

Example

Note Particularly where cataloguing is likely to be based on the content of the header, it is advisable to use a generally recognized name authority file to supply the content for this element. The attributes key or ref may also be used to reference canonical information about the author(s) intended from any appropriate authority, such as a library catalogue or online resource. In the case of a broadcast, use this element for the name of the company or

network responsible for making the broadcast.Where an author is unknown or unspecified, this element may contain text such as Unknown or Anonymous. When the appropriate TEI modules are in use, it may also contain detailed tagging of the names used for people, organizations or places, in particular where multiple names are given.

<biblFull> (fully-structured bibliographic citation) contains a fully-structured bibliographic citation, in which all components of the TEI file description are present.

Module header

Used by «model.biblLike»

May contain

header: «editionStmt» «extent» «notesStmt» «publicationStmt» «seriesStmt» «sourceDesc» «titleStmt»

Declaration

```
element biblFull
{
      att.global.attributes,
      att.declarable.attributes,
      (
            (
            titleStmt,
            editionStmt?,
            extent?,
            publicationStmt,
            seriesStmt?,
            notesStmt?
            ),
      sourceDesc*
      )
}
```

```
<biblFull xmlns:ns223="http://www.tei-c.org/ns/1.0">
<titleStmt>

<title>The Feminist Companion to Literature in English: women
writers from the middle ages
to the present</title>

<author>Blain, Virginia</author>

<author>Clements, Patricia</author>

<author>Grundy, Isobel</author>

</titleStmt>

<editionStmt>

<edition>UK edition</edition>

</editionStmt>

<extent>1231 pp</extent>

<publicationStmt>

<publisher>Yale University Press</publisher>

<pubPlace>New Haven and London</pubPlace>

<date>1990</date>

</publicationStmt>

<sourceDesc>

<p>No source: this is an original work</p>

</sourceDesc>

</biblFull>
```

Example

<distributor> supplies the name of a person or other agency responsible for the distribution of a text.

Module header

Used by «model.imprintPart» «model.publicationStmtPart»

May contain

core: «abbr» «add» «address» «binaryObject» «cb» «choice» «corr» «date» «del» «distinct» «email» «emph» «expan» «foreign» «gap» «gloss» «graphic» «hi» «index» «lb» «measure» «measureGrp» «mentioned» «milestone» «name» «note» «num» «orig» «pb» «ptr» «ref» «reg» «rs» «sic» «soCalled» «term» «time» «title» «unclear»

header: «idno»

linking: «alt» «altGrp» «anchor» «join» «joinGrp» «link» «linkGrp» «seg» «timeline»

Declaration

```
element distributor { att.global.attributes,
                      macro.phraseSeq }
```

```
<distributor xmlns:ns275="http://www.tei-c.org/ns/1.0">Oxford
Text Archive</distributor>
<distributor>Redwood and Burn Ltd</distributor>
```

Example

<title> contains a title for any kind of work.

Module core

Used by «analytic» «monogr» «series» «seriesStmt» «titleStmt»«model.emphLike» «model.msQuoteLike»

May contain

core: «abbr» «add» «address» «bibl» «biblStruct» «binaryObject» «cb» «choice» «cit» «corr» «date» «del» «desc» «distinct» «email» «emph» «expan» «foreign» «gap» «gloss» «graphic» «hi» «index» «label» «lb» «list» «listBibl» «measure» «measureGrp» «mentioned» «milestone» «name» «note» «num» «orig» «pb» «ptr»

«q» «quote» «ref» «reg» «rs» «said» «sic» «soCalled» «stage» «term» «time»

«title» «unclear»

header: «biblFull» «idno»

linking: «alt» «altGrp» «anchor» «join» «joinGrp» «link» «linkGrp» «seg» «timeline»

Declaration

```
element title
{
      att.global.attributes,
      att.canonical.attributes,
      attribute level { "a" | "m" | "j" | "s" | "u" }?,
      attribute type { xsd:Name }?,
      macro.paraContent}
```

```
<title xmlns:ns518="http://www.tei-c.org/ns/1.0">Information
Technology and the Research Process: Proceedings of
a conference held at Cranfield Institute of Technology, UK,
18-21 July 1989</title>
```

Example

```
<title xmlns:ns518="http://www.tei-c.org/ns/1.0">Hardy’s Tess of
the D’Urbervilles: a machine readable
edition</title>
```

Example

```
<title type="full" xmlns:ns518="http://www.tei-c.org/ns/1.0">

<title type="main">Synthèse</title>

<title type="subtitle">an international journal for

epistemology, methodology and history of

science</title>

</title>
```

Example

Note The attributes key and ref, inherited from the class att.canonical may be used to indicate the canonical form for the title; the former, by supplying (for example) the identifier of a record in some external library system; the latter by pointing to an XML element somewhere containing the canonical form of the title.

<titleStmt> (title statement) groups information about the title of a work and those responsible for its intellectual content.

Module header

Used by «biblFull» «fileDesc»

May contain

core: «author» «editor» «meeting» «respStmt» «title»

header: «funder» «principal» «sponsor»

Declaration

```
element titleStmt { att.global.attributes, ( title+,
                    model.respLike* ) }
```

```
<titleStmt xmlns:ns525="http://www.tei-c.org/ns/1.0">
<title>Capgrave’s Life of St. John Norbert: a machine-readable
transcription</title>
<respStmt>
<resp>compiled by</resp>
<name>P.J. Lucas</name>
</respStmt>
</titleStmt>
```

Example