9. APPLICAZIONE SPERIMENTALE DI CDSWARE RELATIVA AL MODULO BIBCONVERT

9.1 INTRODUZIONE

Il presente capitolo rappresenta la parte sperimentale vera e propria dell'intero elaborato di tesi.

Tale parte sperimentale deriva da un incontro con i membri dello SBA (Servizio Bibliotecario d'Ateneo) dell'Università di Messina, per i quali si è realizzata un'applicazione del software CDSware relativa al problema della migrazione dei record bibliografici immagazzinati nel sistema di automazione di biblioteca Aleph 500 al sistema documentario CDSware da loro adottato.

E' importante osservare come il compito assegnato dal dott. Feminò dello SBA comporti la realizzazione di un sistema di gestione dei record che prende spunto da quello già realizzato presso il CERN di Ginevra, descritto al paragrafo 8.3.

All'inizio del lavoro si è utilizzata l'ultima release disponibile online, cioè la versione 0.1.2, che in seguito è stata abbandonata, poiché instabile e mancante di alcuni dei moduli presenti nell'unica (al momento in cui si scrive il presente elaborato) versione stabile 0.0.9, successivamente adottata per lo sviluppo della tesi.

La soluzione a tale compito, dopo una prima analisi di tipo generale dell'applicazione, ha coinvolto l'utilizzo di due moduli interni all'applicazione CDSware: BibConvert e BibFormat.

BibFormat avente il compito sostanziale di creare un formato generale di

rappresentazione dell'output dei record trasferiti da Aleph a CDSware. Tale parte nell'ambito del progetto dello SBA è stata affrontata e trattata nella tesi di Amelotti Ercole [B1].
BibConvert, invece, ha il compito di effettuare la conversione dei record importati da Aleph in CDSware, rispettivamente dal formato interno UNIMARC al formato MARC 21.
La descrizione e l'analisi di tale modulo, che trae spunto dalle informazioni presenti nella documentazione interna all'istallazione di CDSware, rappresenta la prima parte del presente capitolo, dove vengono presentati: la struttura del file di configurazione, le funzioni di formattazione e le funzioni di generazione automatica di valori, necessari a BibConvert per effettuare il proprio compito.
Infine, la seconda parte presenta la soluzione realizzata per risolvere il compito richiesto dallo SBA.

9.2 BIBCONVERT

BibConvert è il modulo funzionale di CDSware progettato e sviluppato per effettuare la conversione di record di metadati espressi in un formato qualunque, in un altro formato supportato dal database locale.
Il suo compito è quello di elaborare i record di metadati precedentemente raccolti e convertirli nel formato XML MARC 21 affinché possano essere successivamente caricati nel database.

Gli utilizzi più comuni per cui tale modulo viene solitamente impiegato riguardano la conversione di record ricevuti da diverse sorgenti di dati, o attraverso acquisizioni automatiche di dati non OAI compatibili e di dati non ben strutturati, o ancora tramite migrazione di dati da un sistema ad un altro, ecc..

BibConvert fornisce un metodo flessibile per la conversione dei metadati, in quanto accetta un qualsiasi formato in input, anche non standard, e, in generale, lo converte in un altro formato XML qualunque in uscita.
Affinché BibConvert operi in tal modo in maniera corretta, deve essere opportunamente configurato in accordo alle necessità dell'utente. Tale configurazione viene effettuata attraverso la creazione di un file di conversione che specifica all'applicazione in che modo il file di ingresso, contenente un certo insieme di record di metadati espressi in uno specifico formato, deve essere convertito nel file di output in formato XML supportato dal database locale (MARC 21). E' dunque necessario creare un file di conversione differente per ciascun formato di metadati.
Tale file è un file di testo ed ha la seguente struttura:

```
### Esempio di file di configurazione
### di bibconvert per la conversione di
### record in formato XML MARC 21


=== Sezione 1: Definizione del record sorgente ===

#   Esempio di due linee di estrazione:

Autori---%AU---MAX---;---
Titolo---%TI---EOL------


=== Sezione 2: Definizione dei campi sorgente ===

#   Esempio di due linee di definizione:

Autori---<:COGNOME:>-<:NOME:>
Titolo---<:TITOLO:>
```

```
=== Sezione 3: Definizione del record di output ===

#   Esempio di due linee di definizione:

Autori::CONF(Autori,,0)---
<datafield id="700" i1="" i2="">
  <subfield code="a">
    <:Autori*::COGNOME::CAP():>, <Autori*::NOME::CAP()::ABR():>
  </subfield>
</datafield>

Titolo::CONF(TI,,0)---
<datafield tag="245" i1="" i2="">
  <subfield code="a">
     <:TI::TI::SUP(SPACE, ):>
  </subfield>
</datafield>


### Fine del file di configurazione
```

Come è possibile notare dall'esempio sopra riportato, nel testo del file di configurazione di BibConvert è possibile utilizzare dei caratteri che hanno dei significati particolari:

- "#" - è il carattere di commento su singola linea, analogo al "//" del C++. Tutto ciò che si trova sulla stessa linea, a destra di tale simbolo, viene considerato dall'applicazione come un commento
- "= = =" - i tre segni di uguaglianza consecutivi dichiarano all'applicazione, l'inizio di una delle tre sezioni di cui il file di configurazione si compone. I caratteri successivi a tale sequenza di simboli vengono ignorati, come fossero un commento
- "**---**" - i tre simboli trattino consecutivi vengono utilizzati per separare i vari valori

Il file di conversione è composto, come accennato sopra, dalle seguenti tre sezioni:

- *Definizione del record sorgente*
- *Definizione dei campi sorgente*
- *Definizione del record di output*

Esse verranno descritte in maniera più dettagliata nei paragrafi successivi.

9.2.1 Definizione del record sorgente

E' la prima sezione del file che permette l'estrazione dei campi dal record di metadati d'ingresso e una mappatura tra i vari tag del record in esame, espressi da una stringa particolare univocamente individuabile, e i corrispondenti tag di comodo utilizzati per riferirsi ad essi nelle sezioni successive.

La sintassi di una linea generica di estrazione è la seguente:

nome---parola chiave---stringa di terminazione---elemento ripetibile ---

dove:

- *parola chiave* - è la stringa di caratteri, univocamente individuabile, che indica un tag del file di record di metadati passato in input. Per esempio, il campo del titolo viene individuato dalla stringa %TI.
- *nome* - è il nome di comodo assegnato per un determinato tag di input, affinché ci si possa riferire ad esso nelle sezioni successive in luogo della parola chiave.

- *stringa di terminazione* - indica quali caratteri estrarre dal file di ingresso, a partire dalla fine della parola chiave indicata, che facciano parte del campo del record in esame. E' possibile utilizzare stringhe di terminazione differenti che danno all'applicazione indicazioni diverse su come comportarsi per l'estrazione:
    - *MAX* - indica di estrarre il massimo numero di elementi disponibile nel record di metadati a partire dalla parola chiave
    - *MIN* - indica di estrarre il minimo numero di elementi disponibile a partire dalla parola chiave
    - *EOL* - indica di estrarre tutti gli elementi fino a fine riga, a partire dalla parola chiave
- *Elemento ripetibile* - indica all'applicazione, quando presente, che è possibile la presenza contemporanea di più tag del tipo indicato dalla parola chiave, cioè che tale tag è ripetibile. Spesso come *elemento ripetibile* si utilizza un simbolo punto e virgola: ";"

Come detto in precedenza, i tre trattini consecutivi, "---", rappresentano un separatore obbligatorio per i valori sulla stessa riga. Esso deve essere indicato anche nel caso di valori di lunghezza zero, cioè non presenti.
Per esempio, considerando il caso di un tag non ripetibile, l'*elemento ripetibile* non sarà presente. Ciò, comporta l'inserimento in tale riga di due separatori consecutivi, cioè sei trattini: "------".

Un esempio di definizione dei campi autore (ripetitivo) e titolo (non ripetitivo) è la seguente:

```
Autore---%AU---MAX---;---
Titolo---%TI---EOL------
```

dove:

%AU e **%TI** sono le parole chiave che indicano rispettivamente, nel record originario, il campo autore e il campo titolo;

Autore è il nome di comodo utilizzato nel resto del file in luogo di %AU;

Titolo è il nome di comodo utilizzato nel resto del file in luogo di %TI;

MAX è la stringa di terminazione che indica di estrarre il massimo numero di elementi disponibile nel record di metadati a partire da %TI;

EOL è la stringa di terminazione che indica di estrarre tutti gli elementi nel record di metadati a partire da %AU, fino a fine riga;

"**;**" è l'elemento ripetibile che indica la possibile presenza di più tag %AU, cioè che esso è ripetibile;

"**---**" è il separatore obbligatorio tra valori sulla stessa riga. Nel caso di titolo sono presenti due di tali separatori consecutivi, in quanto l'elemento ripetibile non è presente.

9.2.2 Definizione dei campi sorgente

In questa sezione viene definita la struttura interna che può avere ciascun campo estratto dal record di metadati di input in accordo alla definizione espressa nella sezione precedente.

La sintassi di una generica linea di definizione è la seguente:

nome---<:SOTTOCAMPO:>[-<:SOTTOCAMPO:>, …]

dove:

- *nome* - è il nome di comodo definito nella sezione precedente e che verrà

utilizzato in luogo del campo che rappresenta

- *<:SOTTOCAMPO:>* - è il sottocampo appartenente al campo indicato da nome. E' possibile, come si nota dalla sintassi, specificare più sottocampi per un determinato campo. Il nome del sottocampo può essere lo stesso di quello del campo
- "-" – viene utilizzato come costante di separazione tra i vari sottocampi, nel caso ne vengano specificati più di uno facenti parte della struttura interna del campo in esame

Come già detto, i tre trattini consecutivi "---", rappresentano un separatore obbligatorio tra valori.
Le parentesi quadre indicano degli argomenti opzionali, mentre ", …" indica che tali argomenti possono essere più di uno.

Un esempio di definizione dei campi autore (ripetitivo) e titolo (non ripetitivo) è la seguente:

```
Autore---<:COGNOME:>-<:NOME:>
Titolo---<:TITOLO:>
```

dove:
Autore e **Titolo** sono i nomi di comodo utilizzati in luogo rispettivamente di %AU e %TI della sezione precedente;
<:COGNOME:> e **<:NOME:>** sono due sottocampi appartenenti alla struttura interna del campo autore;
<:TITOLO:> è il sottocampo appartenente alla struttura interna del capo titolo;
"**-**" è la costante di separazione tra i due sottocampi cognome e nome del campo autore. Non è presente nella definizione riguardante il titolo poiché contiene un unico sottocampo.

9.2.3 Definizione del record di output

Questa sezione descrive il layout del record di output che deve essere creato dalla conversione, insieme alle corrispondenze dei campi sorgente definiti nella sezione precedente.

La sintassi di una generica linea di definizione per un record di output in formato XML MARC 21è la seguente:

[CODICE]---COSTANTE[<:VARIABILE:>, …][COSTANTE, …]

dove:

- *CODICE* – è una stringa opzionale usata per favorire la leggibilità
- *COSTANTE* – è una stringa che viene inserita nel record di output così come è indicata. E' possibile avere ulteriori costanti opzionali
- *VARIABILE* – indica all'applicazione cosa deve essere inserito nel record di output per un determinato tag di input, tale valore dipende o dal contenuto del campo di metadati del record di ingresso specificato e dalle eventuali funzioni di formattazione applicate (vedi par. 9.3), o da una determinata funzione precisata tra quelle a disposizione, che generano valori automaticamente durante la conversione (vedi par 9.4)

Anche in questo caso i tre trattini consecutivi, "---" indicano un separatore obbligatorio, tra un eventuale *CODICE* e *COSTANTE*.
Le parentesi quadre indicano degli argomenti opzionali, mentre ", …" indica che tali argomenti possono essere più di uno.

La sintassi specificata sopra è molto generica, in particolare:

1. per l'elemento CODICE è anche possibile specificare qualche funzione di formattazione, adoperando la seguente sintassi:

   CODICE::FUNZIONE()

   dove:

   - *FUNZIONE()* rappresenta una delle funzioni di formattazione fornite dall'applicazione (par. 9.3). Questo viene fatto, per esempio, per evitare di inserire nel record di output informazioni relative ad un campo non presente nel record di input.

2. per l'elemento VARIABILE le generiche sintassi possibili sono le seguenti:

   a) **<:nome[*]::SOTTOCAMPO[::FUNZIONE(), …]:>**
   dove:

      - *nome* - è il nome di comodo definito nella prima sezione e che viene utilizzato per rintracciare nel record di input il campo che rappresenta.
      - *SOTTOCAMPO* - è il sottocampo appartenente al campo indicato da nome. Indica all'applicazione di inserire al suo posto il contenuto del sottocampo.
      - *FUNZIONE()* – indica all'applicazione l'eventuale funzione di formattazione da applicare al contenuto del campo, prima di inserirlo nel record di output.
      - "*" - dice all'applicazione che il campo indicato da nome è ripetibile, quindi dovranno essere prese in considerazione tutte le occorrenze di tale campo nel record di input.

- "::-" è un separatore obbligatorio tra *nome* e *SOTTOCAMPO* ed eventualmente, tra *SOTTOCAMPO* e *FUNZIONE()* e tra *FUNZIONE()* e *FUNZIONE()*.

b) <:VALORE_GENERATO:>

- *VALORE_GENERATO* – indica all'applicazione quale funzione applicare tra quelle che generano valori automaticamente durante la conversione, onde inserire nel record di output tale valore.

Vengono di seguito presentati alcuni esempi.

Esempio 1:

L'esempio più semplice è il seguente:

```
HEAD---<record>
```

dove:

HEAD è il CODICE. In questo caso indica che ci si riferisce all'inizio del record;
<record> è la COSTANTE che viene inserita nel record così come è.
Tale linea di esempio è solitamente utilizzata all'inizio della sezione di definizione del record di output per aggiungere in esso il tag <record> che ne indica l'inizio.

Esempio 2:

```
LINGUA---<datafield tag="041" ind1="0" ind2="">
              <subfield code="a">
                 ita
```

```
            </subfield>
        </datafield>
```

LINGUA è il CODICE. In questo caso indica che ci si riferisce alla lingua del testo.

Le parti successive:

<datafield tag="041" ind1="0" ind2="">

<subfield code="a">

ita

</subfield>

</datafield>

sono stringhe costanti che vengono inserite nel record così come sono, nello stesso ordine.

Esempio 3:

```
TI::CONF(TI,,0)---<datafield tag="245" i1="" i2="">
                      <subfield code="a">
                         <:TI::TI::SUP(SPACE, ):>
                      </subfield>
                  </datafield>
```

TI è il CODICE. Ci si riferisce al campo titolo.

CONF(TI,,0), **SUP(SPACE,)** sono le funzioni di formattazione applicate.

Le parti:

<datafield tag="245" i1="" i2="">

<subfield code="a">

</subfield>

</datafield>

sono stringhe costanti che vengono inserite nel record così come sono, nello

stesso ordine.

<:TI::TI::SUP(SPACE,):> è la VARIABILE, che specifica all'applicazione di inserire nel record di output, tra le stringhe <subfield code="a"> e </subfield>, il contenuto del campo titolo sopprimendo gli spazi presenti.

"**::**" è il separatore obbligatorio.

Esempio 4:

```
SW---<datafield tag="909" ind1="C" ind2="S">
         <subfield code="w">
            <:DATE(%Y%W,6):>
         </subfield>
      </datafield>
```

SW è il CODICE.

Le parti:

<datafield tag="909" ind1="C" ind2="S">

<subfield code="w">

</subfield>

</datafield>

sono stringhe costanti che vengono inserite nel record così come sono, nello stesso ordine.

<:DATE(%Y%W,6):> è la VARIABILE. Richiama la funzione DATE che genera un valore durante la conversione in funzione dei parametri passati, in particolare l'applicazione inserirà nel record di output tra le stringhe <subfield code="w"> e </subfield> la data corrente nel formato: anno (4 caratteri) e numero della settimana corrente (2 caratteri).

Una volta realizzato il file di configurazione adatto ai propri scopi, BibConvert sarà in grado di convertire il file di record di metadati di input, in accordo alle

corrispondenze presenti, in maniera automatica.

Attualmente BibConvert può essere configurato ed utilizzato solo attraverso linea di comando, dato che non è stata ancora realizzata la sua interfaccia grafica.

Nei paragrafi immediatamente seguenti verranno analizzate le funzioni di formattazione e quelle per la generazione di valori fornite dall'applicazione, alcune delle quali sono state accennate precedentemente.

9.3 FUNZIONI DI FORMATTAZIONE

BibConvert permette di elaborare ogni campo di un record di metadati in input attraverso una varietà di funzioni che ne modificano parzialmente o interamente il valore originale.
Ve ne sono disponibili di tre tipi: quelle che prendono singoli caratteri, parole o l'intero valore del campo processato.

Ogni funzione richiede un certo numero di parametri che devono essere inseriti tra le parentesi che seguono il nome, nel giusto ordine. Se viene specificato un numero insufficiente di parametri, la funzione utilizza quelli di default.
Le funzioni devono essere richiamate rispettando le lettere maiuscole e minuscole dei loro nomi.
Quelle fornite dall'applicazione sono le seguenti:

ADD(prefisso,suffisso) – aggiunge prefisso/suffisso
KB(kb_file,[0-2]) – controlla in kb_file e sostituisce il valore
ABR(x,suffisso) – abbrevia ad "x" caratteri e aggiunge suffisso
ABRW(x,suffisso) - abbrevia e aggiunge suffisso

ABRX(x,suffisso) - abbrevia solo parole più lunghe di “x” e aggiunge suffisso

CUT(prefisso,suffisso) – rimuove una stringa all’inizio o alla fine

REP(x,y) – sostituisce una stringa con un’altra

SUP(tipo,stringa) – sopprime i caratteri di un certo tipo o li sostituisce con una stringa

LIM(n,L/R) - elimina i caratteri in eccesso all’inizio o alla fine

LIMW(stringa,L/R) – rimuove a sinistra o a destra di una data stringa

WORDS(n,L/R) – limita a “n” parole da sinistra o destra

MINL(n) - elimina tutte le parole più brevi del limite specificato

MAXL(n) - elimina tutte le parole più lunghe del limite specificato

MINLW(n) – elimina il valore di input se non contiene almeno un certo numero di caratteri

EXP(stringa,1/0) – elimina le parole che contengono o meno una certa stringa

EXPW(tipo) – elimina le stringhe del tipo specificato

IF(valore,valoreTRUE,valoreFALSE) – effettua un confronto e rimpiazza con uno dei due valori specificati

UP – converte in maiuscolo

DOWN – converte in minuscolo

CAP – converte il primo carattere in maiuscolo

SHAPE – sopprime gli spazi non ammessi

NUM – converte in numero

SPLIT(n,h,stringa,da) – suddivide in più linee

SPLITW(sep,h,stringa,da) – suddivide in più linee

CONF(campo,valore,1/0) – conferma la validità di un campo

CONFL(valore,1/0) - conferma la validità di un campo

RANGE(da,a) - conferma le entry nell’intervallo specificato

Tali funzioni verranno trattate nei paragrafi successivi.

9.3.1 ADD(prefisso,suffisso)

Aggiunge il valore specificato tra parentesi, come prefisso e/o suffisso al valore del campo del record di metadati a cui viene applicata, in funzione della posizione, se prima o dopo la virgola rispettivamente, in cui tale valore viene specificato.

Se la funzione viene richiamata senza alcun parametro, nel seguente modo:

```
ADD(,)
```

per default non viene effettuata alcuna aggiunta.

ADD può essere utilizzata, ad esempio, per aggiungere il nome del campo come prefisso al valore stesso.

Esempio:

```
ADD(AUTORE = ,)
```

È il prefisso per il campo contenente il nome dell'autore principale.

9.3.2 KB(kb_file,[0-2])

Confronta il valore di input con quelli (input_value) presenti all'interno del file specificato, kb_file, e eventualmente lo sostituisce con un altro valore (output_value).

Il file kb_file, indicato nel parametro, è un file di testo che rappresenta una tabella contenente una corrispondenza tra possibili valori di input e i valori di output:

{**input_value---output_value**}

Nel caso in cui non viene trovata la corrispondenza tra il valore di input e quelli presenti nel file, il valore di input stesso, per default, viene mantenuto senza alcuna modifica.
Tale comportamento può comunque essere cambiato modificando l'entry del valore di default: **_DEFAULT_---default value** in kb_file.

E' possibile specificare, come secondo parametro, un numero compreso tra 0 e 2, col seguente significato:
KB(file,0) ricerca il codice KB dentro il valore passato
KB(file,1) ricerca l'esatto valore passato
KB(file,2) come 0 ma non case sensitive

Gli spazi ai margini non vengono considerati. Il valore di output non viene ulteriormente formattato.

9.3.3 ABR (x,suffisso)

Abbrevia le parole presenti nel campo del record in input in funzione dei parametri specificati. In particolare, ogni parola viene abbreviata al numero "x" di caratteri indicati nel primo parametro se più lunga e, se specificato, viene aggiunto un suffisso indipendentemente dal fatto che la parola sia stata abbreviata o meno.

Se non viene specificato alcun parametro, per default, ogni parola viene abbreviata al primo carattere e a questo viene aggiunto, come suffisso, un punto, come se la funzione venisse chiamata nel seguente modo:

```
ABR(1,.)
```

La tabella seguente mostra un esempio in cui la funzione viene chiamata più volte con parametri differenti per lo stesso input, ottenendo risultati di output diversi:

funzione	**input**	**output**
ABR()	cognome_nome	c._n.
ABR(1,)	cognome_nome	c_n
ABR(9,COMMA)	cognome_nome	cognome,_nome,

Gli spazi sono indicati con un simbolo underscore "_".

Nel terzo esempio il valore del secondo parametro, COMMA, indica la virgola.

9.3.4 ABRW(x,suffisso)

E' analoga alla funzione ABR descritta nel paragrafo precedente, con la differenza che essa considera l'intero valore del campo del record passatole in input come un unica parola.

Nel caso non venga indicato alcun parametro, per default, la funzione si comporta come se fosse invocata in tal modo:

```
ABRW(1,.)
```

9.3.5 ABRX(x,suffisso)

Abbrevia esclusivamente le parole più lunghe del numero di caratteri "x" specificato nel primo parametro, solo in tal caso verrà aggiunto in coda, se presente, il suffisso.

Per default, ogni parola viene abbreviata al primo carattere e a questo viene aggiunto, come suffisso, un punto; come se la funzione venisse chiamata nel seguente modo:

```
ABRX(1,.)
```

9.3.6 CUT(prefisso,suffisso)

Rimuove, se presente, la stringa specificata come prefisso o suffisso, rispettivamente all'inizio o alla fine del valore del campo del record di metadati a cui viene applicata.

E' opposta alla funzione ADD discussa al paragrafo 9.3.1.
Se la funzione viene richiamata senza alcun parametro, per default, non viene rimossa alcuna stringa.

9.3.7 REP(x,y)

Ricerca, all'interno del valore del campo del record di metadati cui viene applicata, la stringa specificata nel primo parametro con cui essa viene invocata. Ognuna di queste stringhe, eventualmente presenti, viene sostituita con la stringa indicata nel secondo parametro.

Se la funzione viene richiamata senza alcun parametro, non viene effettuata alcuna sostituzione.

9.3.8 SUP(tipo,stringa)

Tutti i gruppi di caratteri appartenenti al tipo specificato nel primo parametro vengono soppressi o sostituiti con una stringa indicata nel secondo parametro.

I tipi ammessi che possono essere specificati nel parametro "tipo" ed i loro significati sono i seguenti:

SPACE – caratteri invisibili incluso NEWLINE
ALPHA – caratteri alfabetici
NALPHA – caratteri non alfabetici
NUM – caratteri numerici
NNUM – caratteri non numerici
ALNUM – caratteri alfanumerici
NALNUM – caratteri non alfanumerici
LOWER – lettere minuscole
UPPER – lettere maiuscole
NPUNCT – caratteri non di punteggiatura

Nella seguente tabella vengono visualizzati tre esempi di SUP richiamata con parametri differenti, applicata allo stesso valore di input, ed i corrispondenti valori di output.

Funzione	input	output
SUP(SPACE,-)	giu_2004	giu-2004
SUP(NNUM)	giu_2004	2004
SUP(NUM)	giu_1999	giu_

Gli spazi sono indicati con un simbolo underscore "_".

9.3.9 LIM(n,L/R)

Limita il valore del campo del record di metadati cui viene applicata al numero "n" di caratteri indicati nel primo parametro, eliminando i caratteri in eccesso a destra o a sinistra, a seconda che il lato specificato nel secondo parametro sia rispettivamente "R" o "L".

Nella tabella seguente vengono mostrati due esempi del funzionamento della funzione LIM() per lo stesso valore di input:

Funzione	input	output
LIM(4,L)	giu_2004	2004
LIM(4,R)	giu_2004	giu_

9.3.10 LIMW(stringa,L/R)

Ricerca all'interno del valore di input la stringa specificata nel primo parametro

e, nel caso in cui tale ricerca ha esito positivo, rimuove tutto ciò che si trova alla sua sinistra o alla sua destra, in funzione del valore indicato nel secondo parametro: "L" o "R" rispettivamente.

funzione	input	output
LIMW(_,R)	giu_2004	giu_

Nota: Si è notato, nel corso dell'esperienza effettuata durante la realizzazione del file di conversione per la parte sperimentale di tale elaborato di tesi, che la funzione LIMW si comporta in modo anomalo quando si specifica come secondo parametro "L". In particolare, se la stringa specificata nel primo parametro non è presente all'interno del valore di input cui LIMW viene applicata, essa genera un errore che comporta la terminazione forzata dell'esecuzione dell'applicazione bibconvert che la utilizza. Tale comportamento non è stato riscontrato specificando come secondo parametro "R".

9.3.11 WORDS(n,L/R)

Limita il numero di parole contenute nel campo del record di metadati in input al numero "n" specificato nel primo parametro. Vengono eliminate tutte le parole in eccesso dal lato opposto a quello specificato nel secondo parametro.

I valori assegnati alla funzione *WORDS* come parametri di default, sono i seguenti:
Primo parametro = 0
Secondo parametro = R

Dunque se non viene precisato alcun parametro la funzione viene chiamata come se fosse:

```
WORDS(0,R)
```

La tabella seguente mostra alcuni esempi di come lavora tale funzione:

funzione	input	output
WORDS(1)	giu_2004	2004
WORDS(1,L)	giu_2004	giu_

9.3.12 MINL(n)

Elimina dal contenuto di input tutte le parole più brevi del limite "n" specificato nel parametro.
Le parole di lunghezza esattamente uguale al limite indicato vengono mantenute.
Il valore del parametro assegnato per default è 1.

La tabella seguente mostra alcuni esempi:

funzione	input	output
MINL(4)	Fisica delle particelle	Fisica delle particelle
MINL(5)	Fisica delle particelle	Fisica particelle

9.3.13 MAXL(n)

Elimina dal contenuto di input tutte le parole più lunghe del limite "n" specificato nel parametro.
Le parole di lunghezza esattamente uguale al limite indicato vengono mantenute.

Il valore del parametro assegnato per default è 0.

La tabella seguente mostra alcuni esempi:

funzione	input	output
MAXL(4)	Fisica delle particelle	delle
MAXL(5)	Fisica delle particelle	delle

9.3.14 MINLW(n)

Cancella l'intero valore di input se è esso non contiene almeno un numero di caratteri pari a quello specificato nel parametro "n".

Essa viene spesso usata per la validazione dei record creati, in quanto se ad esempio si hanno 20 caratteri nell'intestazione, chiamando la funzione: MINLW(21), il record non sarà considerato valido se non contiene almeno 21 caratteri compresa l'intestazione.
Per default il valore del parametro assegnato alla funzione è 1, che non comporta

dunque alcun cambiamento.

9.3.15 EXP(stringa,1/0)

Ricerca all'interno del campo del record di input le parole contenenti la stringa specificata nel primo parametro e, se il valore precisato per il secondo parametro è "0", elimina le parole che non contengono tale stringa, viceversa, se tale valore è "1", elimina le parole che contengono la stringa.

Per default, se non vengono specificati i parametri, la funzione viene chiamata come se fosse:

```
EXP(,0)
```

lasciando il valore in input inalterato.

La tabella seguente mostra qualche esempio del modo di operare della funzione:

funzione	**Input**	**output**
EXP(@,0)	mail to: libdesk@cern.ch	libdesk@cern.ch
EXP(:,1)	mail to: libdesk@cern.ch	mail libdesk@cern.ch
EXP(@)	mail to: libdesk@cern.ch	libdesk@cern.ch

9.3.16 EXPW(tipo)

Elimina dal contenuto del campo del record in input tutte le stringhe del tipo specificato come parametro.

I tipi supportati in tale funzione sono i seguenti:
ALPHA – alfabetico
NALPHA – non alfabetico
NUM – numerico
NNUM – non numerico
ALNUM – alfanumerico
NALNUM – non alfanumerico
LOWER – lettere minuscole
UPPER – lettere maiuscole
PUNCT – punteggiatura
NPUNCT – non punteggiatura

SPACE non è ammesso come parola chiave, poiché tutti i caratteri di spaziatura sono considerati come separatori di parole.

La seguente tabella mostra qualche esempio:

funzione	input	output
EXPW(NNUM)	giu_2004	2004
EXPW(NUM)	giu_2004	giu

Gli spazi sono indicati con un simbolo underscore "_".

9.3.17 IF(valore,valoreTRUE,valoreFALSE)

Confronta il valore di input con quello del primo parametro. Se il risultato di tale confronto è TRUE, sostituisce il valore di input con il secondo parametro (valoreTRUE), viceversa con il terzo parametro (valoreFALSE).

Nel caso in cui il valore di input deve essere mantenuto, si può usare come valore per il secondo o terzo parametro la parola chiave "ORIG".

Per default si ha:

```
IF(,,)
```

La tabella seguente mostra qualche esempio:

Funzione	input	output
IF(giu_2004,giu)	giu_2004	giu
IF(lug_2004,lug)	giu_2004	
IF(lug_2004,lug,ORIG)	giu_2004	giu_2004

9.3.18 UP

Converte tutti i caratteri del valore in input in maiuscolo.

9.3.19 DOWN

Converte tutti i caratteri del valore in input in minuscolo.

9.3.20 CAP

Converte il carattere iniziale di ogni parola in maiuscolo e i caratteri rimanenti in minuscolo.

9.3.21 SHAPE

Sopprime tutti gli spazi non ammessi.

9.3.22 NUM

Se il valore di input contiene almeno un carattere numerico, la funzione lo converte in un numero sopprimendo tutti gli altri caratteri.
Gli zeri iniziali vengono cancellati.

9.3.23 SPLIT(n,h,stringa,da)

Suddivide il valore del campo del record in input su più linee in funzione degli argomenti che le vengono passati. Ogni linea contiene al più "n+h+lunghezza di stringa" caratteri.

Dove:
Il parametro "h" rappresenta il numero di caratteri presi come intestazione (header).

Il parametro “n” indica il numero di caratteri che seguono l’header.
Il parametro “stringa” è una stringa addizionale che può essere inserita come separatore tra l’header e il valore seguente.

L’header viene ripetuto all’inizio di ogni linea.

E’ possibile inoltre limitare la presenza della stringa di separazione così che essa appaia a partire da una linea successiva alla prima specificando come quarto parametro, “da”, il numero della linea desiderato.

9.3.24 SPLITW(sep,h,stringa,da)

Suddivide il valore del campo del record di metadati in input in più linee, sostituendo, dove presente, il separatore di linea precisato nel primo parametro “sep” con CR/LF.

Così come avviene per la funzione SPLIT trattata nel paragrafo precedente, anche in questo caso i primi “h” caratteri sono presi come intestazione e ripetuti all’inizio di ogni linea.
Una stringa addizionale può essere inserita come separatore tra l’header ed i caratteri che lo seguono, essa viene specificata nel terzo parametro, “stringa”.
Anche in questo caso è possibile limitare la presenza della stringa di separazione così che essa appaia a partire da una linea successiva alla prima, specificando come quarto parametro, “da”, il numero della linea desiderato.

9.3.25 CONF(campo,valore,1/0)

Conferma o meno la validità di un campo.

Il valore di input viene preso così com'è o rifiutato in funzione di un altro campo del record, che viene specificato nel primo parametro.
Se tale campo contiene la stringa specificata nel secondo parametro "valore", allora il valore di input viene mantenuto o rifiutato rispettivamente se il valore specificato nel terzo campo è "1" o "0".

9.3.26 CONFL(stringa,1/0)

Il valore di input viene confermato se esso contiene (secondo parametro 1) o non contiene (secondo parametro 0) la stringa specificata nel primo parametro.

9.3.27 RANGE(da,a)

Serve a selezionare le entry desiderate da un campo ripetitivo.

Essa conferma le entry il cui output cade nell'intervallo specificato dai due parametri, "da", "a", compresi tali valori limite.
Come limite superiore è possibile utilizzare la parola chiave MAX.
Il range specificato può solo essere continuo.

Tale funzione è utile per esempio per il campo autore, dove la prima entry ha una definizione differente dalle altre.:

```
AU::RANGE(1,1)---Autore principale: <:AU::COGNOME:>, <:AU::NOME:>
```

Prende il primo nome dal campo AU definito.

AU::RANGE(2,MAX)---Altri autori: <:AU::COGNOME:>, <:AU::NOME:>
Prende il resto dei nomi dal campo AU.

9.4 VALORI GENERATI

Durante la conversione di un record, i valori possono essere presi sia dal record sorgente che generati durante il processo stesso.
Le funzioni trattate nei paragrafi seguenti si occupano di generare tali valori.

9.4.1 DATE(formato,n)

Genera la data corrente nella forma stabilita nel primo parametro, mentre il secondo specifica il numero di cifre richieste.

Il formato della data deve essere specificato in accordo con la notazione ANSI C, esso può essere composto dai seguenti componenti:

%a	nome del giorno della settimana abbreviato
%A	nome del giorno della settimana completo
%b	nome del mese abbreviato
%B	nome del mese completo
%c	rappresentazione data e ora
%d	giorno del mese (01-31)
%H	ora (00-23)(formato 24 ore)
%I	ora (01-12)(formato 12 ore)
%j	giorno dell'anno (001-366)

%m mese (01-12)

%M minuto (00-59)

%p equivalente locale di a.m. o p.m.

%S secondo (00-59)

%U numero della settimana nell'anno, iniziando da domenica (00-53)

%V numero della settimana nell'anno

%w giorno della settimana, iniziando da domenica. (0-6)

%Wnumero della settimana nell'anno, iniziando da lunedì.(00-53)

%x rappresentazione data locale

%X rappresentazione ora locale

%y anno (senza il prefisso del secolo)

%Y anno (col prefisso del secolo)

%Z nome del fuso orario

%% %

Per default si ha: DATE(,10)

9.4.2 WEEK(diff)

Inserisce due cifre rappresentanti il numero della settimana corrente %V (vedi elenco paragrafo precedente), incrementato, eventualmente, del valore specificato nel parametro "diff".

Se il numero risultante è negativo, il valore restituito è zero (00).
I valori sono mantenuti fino a 99, i valori di tre cifre vengono abbreviati eliminando quella più a sinistra.

Alcuni esempi sono i seguenti:

WEEK(-4) restituisce 48, se la settimana corrente è la 52-esima.

WEEK restituisce il numero della settimana corrente

9.4.3 SYSNO

Lavora in modo analogo alla funzione DATE (vedi paragrafo 9.4.1), ma il formato della data risultante è fissato, in modo da attenersi con i requisiti di ulteriori gestioni di record.

Tale formato è: "whhmmss", dove:

w giorno della settimana corrente (da 0 a 6)

hh ora corrente

mm minuto corrente

ss secondo corrente

Il numero di sistema, se generato in questo modo, contiene un valore variabile che cambia ogni secondo.

Il numero di sistema è un identificatore del record, quindi è necessario assicurare che esso sia unico per l'intera elaborazione del record.

Diversamente dalla funzione DATE, che genera semplicemente il valore nel formato specificato, SYSNO mantiene il valore persistente per tutto l'intero record ed esclude collisioni con altri record che sono generati nel periodo di una settimana con granularità YYYY-MM-DDThh:mm:ssZ (vedi tesi par 7.5).

Viceversa, non è possibile utilizzare DATE per generare un numero di sistema.

Il numero di sistema è unico solo nell'arco di una settimana, in accordo con la definizione corrente.

9.4.4 OAI

Inserisce un identificatore OAI incrementato di uno per ogni record.

Il valore iniziale che viene utilizzato nel primo record di un gruppo può essere specificato da linea di commando usando l'opzione:
–o<valore_iniziale>.

9.5 REALIZZAZIONE DELL'IMPORTAZIONE DEI DATI BIBLIOGRAFICI

Di seguito verrà descritta e analizzata la soluzione elaborata per la risolvere il compito affrontato, riguardante l'importazione di record di metadati nel formato interno di CDSware, MARC 21, dal formato UNIMARC di Aleph 500. Questa, rappresenta il cuore della parte sperimentale dell'elaborato di tesi presente, svolta in collaborazione e per conto del Servizio Bibliotecario d'Ateneo (SBA) di Messina.

Come accennato precedentemente, tale importazione è resa possibile attraverso l'utilizzo dei due moduli, BibConvert e BibFormat, interni all'applicazione CDSware.

BibFormat si occupa fondamentalmente della definizione del formato di rappresentazione in output dei record bibliografici all'utente finale. Tale modulo è oggetto dell'elaborato di tesi di Amelotti Ercole [B1]; si faccia perciò riferimento a tale documento per qualsiasi considerazione fatta in merito a tale applicazione di CDSware.
BibConvert, invece, permette la trasformazione vera e propria dei record di

metadati dal formato interno di Aleph al formato XML MARC 21 di CDSware.

Nei paragrafi successivi, dopo aver accennato al formato UNIMARC e alla mappatura UNIMARC-MARC 21 per i campi utilizzati e aver mostrato la struttura dei record di Aleph, verranno trattate le fasi della procedura di conversione per i formati dei record bibliografici adottati allo SBA di Messina.

9.5.1 Breve descrizione del formato dei record UNIMARC

Il formato UNIMARC, le cui origini sono comuni a quelle del formato MARC 21, ampiamente trattato al cap. 4 di tale elaborato, possiede una struttura dei record analoga a quella del "cugino" americano.
Difatti, i componenti dei record, che hanno la funzione di identificare e distinguere gli elementi dei dati, sono gli stessi: *tag*, *indicatori* e *codici di sottocampo*. Essi adottano anche le stesse regole sintattiche.
Onde evitare di ricadere in ripetizioni superflue, si rimanda chiunque fosse interessato, ai paragrafi della tesi 4.4, 4.5 e ai relativi sottoparagrafi.

La fondamentale differenza riguarda invece l'organizzazione dei dati, che nel formato UNIMARC è stata preparata in maniera differente.
I blocchi UNIMARC infatti, sono così organizzati:

- 0XX Blocco di identificazione
- 1XX Blocco delle informazioni codificate
- 2XX Blocco delle informazioni descrittive
- 3XX Blocco delle note
- 4XX Blocco dei legami

- 5XX Blocco dei titoli in relazione
- 6XX Blocco dell'analisi semantica
- 7XX Blocco della responsabilità intellettuale
- 8XX Blocco dei dati internazionale
- 9XX Blocco di uso locale

"XX" rappresentano due numeri qualunque compresi tra 0 e 9.

9.5.2 Mappatura dei campi UNIMAR-MARC 21 utilizzati

Mostriamo di seguito la mappatura dei campi UNIMARC nei relativi MARC 21. Tale mappatura comprende solo quei campi che lo SBA ha ritenuto utile convertire, anche in funzione dei record bibliografici da essi gestiti.
Per una descrizione dettagliata delle specifiche di conversione dei record bibliografici dal formato UNIMARC a quello MARC 21, si consiglia di prendere in considerazione la documentazione preparata dal "Network Development and MARC Standards Office" della Library of Congress [S18].

Campi UNIMARC	Sottocampi UNIMARC	Campi MARC 21	Sottocampi MARC 21
001		001	
005		005	
010 ##	$a	020	$a
011 ##	$a	022	$a
101 0#	$a	041 0#	$a
101 1#	$a	041 1#	$a
102 ##	$a	044 ##	$a

200 1#	$a $e $f	245 1#	$a $b $c
205 ##	$a $f	250 ##	$a $b
207 #1	$a	362 1#	$a
210 ##	$a $c $d	260 ##	$a Sb $c
215 ##	$a $d	300 ##	$a $c
225 0#	$a $v	490 1#	$a $c
225 2#	$a $v	490 1#	$a $v
300 ##	$a	500 ##	$a
410 #1	$v	760 0#	$g
512 1#	$a	246 14	$a
610 ##	$a	653 0#	$a
610 0#	$a	653 0#	$a
700 #1	$a	100 1#	$a
702x1	$a	700 1#	$a
801 #0	$a e $b $g	040 ##	$a $e

La tabella esposta sopra mostra i tag UNIMARC con indicatori e sottocampi e i relativi MARC 21.

I campi MARC 21 presenti sono stati trattati al paragrafo 4.8 del presente elaborato di tesi.

9.5.3 Struttura dei record UNIMARC di Aleph 500

I record prelevati da Aleph 500 presso lo SBA di Messina possiedono una struttura particolare, propria del sistema di automazione di biblioteca in esame.
La struttura dei campi presenti in tali record è la seguente:

numero_di_registrazione parola_chiave contenuto_del_campo

dove:

- *numero_di_registrazione* – è un numero di nove cifre, che rappresenta il numero di registrazione del record assegnato da Aleph. Ad esempio: 000001189
- *parola_chiave* – è una stringa, univocamente individuabile all'interno del file, che rappresenta un determinato campo. In particolare, un esempio di tale stringa adottata da Aleph per il campo 700 con il primo indicatore non presente ed il secondo posto ad 1 è la seguente: 700-1-L-. Da notare che i trattini sono stati qui aggiunti, uno in luogo di ogni spazio bianco presente, per questioni di leggibilità.
- *contenuto_del_campo* – è la stringa che include le informazioni presenti in un determinato campo, contenente, se presenti, anche i codici di sottocampo. Aleph utilizza per i loro delimitatori "$$".

Un esempio di record fornito dal sistema, che deve essere convertito nel formato XML MARC 21 è il seguente:

```
000001189 FMT   L BK
000001189 LDR   L -----nam--22--------450-
000001189 001   L 000001189
000001189 005   L 20020320094625.0
```

```
000001189 010   L $$a88-15-06306-4
000001189 100   L $$a20010911h1998----km-y0itay0103----ba
000001189 1010  L $$aita
000001189 102   L $$aIT
000001189 2001  L $$a<<Le >>trappole del welfare$$fMaurizio Ferrera
000001189 210   L $$aBologna$$cIl Mulino$$dc1998
000001189 215   L $$a168 p.$$d21 cm
000001189 2250  L $$aContemporanea$$v99
000001189 300   L $$aSegue: Appendice
000001189 410 1 L $$12001$$aContemporanea$$v99
000001189 5121  L $$a<<Uno >>stato sociale sostenibile per l'Europa
del 21. secolo
000001189 6100  L $$aEuropa$$aPolitica sociale
000001189 6100  L $$aWelfare state
000001189 700 1 L $$aFerrera,$$bMaurizio
000001189 CAT   L $$aFALDUTO$$b00$$c20010911$$lMES01$$h1252
000001189 CAT   L $$c20020211$$lMES01$$h1301
000001189 CAT   L $$aSF04$$b01$$c20020320$$lMES01$$h0932
000001189 CAT   L $$aSF04$$b01$$c20020320$$lMES01$$h0946
000001189 801 0 L $$aIT$$bSBA Messina$$gRICA
```

9.5.4 Creazione del file di configurazione

Come già visto al par. 2 del presente capitolo, affinché BibConvert effettui la conversione di un file di record di metadati espressi in un determinato formato in un altro di output, è necessario creare il file di configurazione adatto alle proprie necessità.

Nel nostro caso, il file di configurazione è stato creato in modo tale da specificare all'applicazione che i record di input, presi da Aleph, devono essere convertiti in formato XML MARC 21, supportato dal database di CDSware.

Nei tre paragrafi seguenti viene mostrato il codice del file di configurazione che è

stato realizzato per la risoluzione del compito, in funzione delle tre sezioni di cui il file si compone e che sono:

- *Sezione 1: Definizione del record sorgente*
- *Sezione 2: Definizione dei campi sorgente*
- *Sezione 3:Definizione del record di output*

9.5.4.1 Sezione 1

Tale sezione permette la mappatura tra i vari campi dei record di input e i corrispondenti tag di comodo utilizzati per riferirsi ad essi nelle sezioni successive del file di configurazione.

Essa incomincia con tre segni di uguaglianza consecutivi: "= = =" che ne dichiarano all'applicazione l'inizio, seguiti dal nome della sezione, che viene considerato come un commento. Tale linea, per la nostra applicazione è la seguente:

```
=== Definizione del record sorgente ===
```

Ricordiamo che una generica linea di codice per tale sezione ha la seguente sintassi:

nome---parola chiave---stringa di terminazione---elemento ripetibile---

La forma generica della *parola chiave* fornita da Aleph, come visto al par. 9.5.3, sappiamo essere : "aaabb-L-"
Dove con "aaa" si indica il tag UNIMARC, mentre con "bb" i due indicatori ad

esso successivi, eventualmente uno spazio per ogni indicatore assente, i trattini rappresentano degli spazi bianchi.

Per l'elemento *nome* si è deciso di utilizzare la seguente regola per nominare i tag di comodo cui riferirsi nelle sezioni successive:

- Se il tag del record di metadati di input, prelevato da Aleph500 in formato UNIMARC non possiede indicatori, allora il nome di comodo ad esso assegnato sarà lo stesso del tag in esame.

  Per esempio se il tag UNIMARC considerato è 205 il nome del tag di comodo sarà 205

- Se il tag possiede entrambi gli indicatori, il nome corrispondentemente assegnato sarà formato dai tre caratteri del tag più i due degli indicatori, nell'ordine in cui essi compaiono. Per esempio se il tag considerato è 601 e i suoi indicatori sono entrambi presenti: 1 e 0 rispettivamente, allora il tag di comodo sarà chiamato 60110.

- Se il tag possiede solo un indicatore, il nome corrispondente sarà formato dai tre caratteri del tag, da una x nella posizione dell'indicatore mancante e dall'indicatore presente, nella sua posizione. Per esempio se il tag UNIMARC considerato è 200 ed esso ha come primo indicatore 1 ed il secondo non specificato, allora il tag di comodo avrà come nome: 2001x.

Inoltre, si è scelto di utilizzare come *stringa di terminazione* in tutte le linee della sezione, la stringa EOL che indica di estrarre tutti gli elementi fino a fine riga, a partire dalla parola chiave.
Mentre come *elemento ripetibile* viene utilizzato, in tutti i casi in cui è possibile la presenza multipla dello stesso tag, il simbolo ";".

Come esempio, vengono qui presentate tre delle mappature create, relativamente

all'applicazione allo SBA, che riassumono vari casi possibili:

```
001---001   L ---EOL------
2001x---2001  L ---EOL------
700x1---700 1 L ---EOL------
```

9.5.4.2 Sezione 2

Ogni campo estratto dal record di metadati di input può avere, in accordo alla definizione espressa nella sezione precedente, una struttura interna, che viene descritta in questa sezione.

Si è definito ogni campo del record di input, contrassegnato in tale sezione dal nome di comodo definito in quella precedente, come formato da un unico sottocampo, chiamato con lo stesso nome del campo di partenza, senza specificare la struttura interna dei vari campi.
Si è deciso di intraprendere tale scelta, perché, in base a quanto appreso tramite i file di esempio forniti e a quanto successivamente verificato, il funzionamento dell'applicazione non varia indicando dei sottocampi.
La fase di estrazione dei sottocampi e del loro contenuto viene effettuata nella terza sezione.

Relativamente all'applicazione realizzata presso lo SBA, essa incomincia con la seguente linea:

```
=== Definizione dei campi sorgente ===
```

mentre, la struttura dei campi è stata impostata come per quelli seguenti:

```
001---<:001:>
2001x---<:2001x:>
700x1---<:700x1:>
```

9.5.4.3 Sezione 3

In quest'ultima sezione viene specificato a BibConvert il layout che devono avere i record di output che devono essere creati dalla conversione. Viene dunque realizzata la corrispondenza tra i tag di comodo e gli elementi XML dei tag MARC 21 e dei relativi sottocampi. Tali record verranno memorizzati all'interno di un file XML di output.

Per quanto riguarda l'applicazione in questione, tale sezione incomincia con la seguente linea:

```
=== Definizione del record di output ===
```

che segnala all'applicazione l'inizio della terza sezione.

Per ogni possibile tag UNIMARC del record di input che si desidera mappare nel tag XML MARC 21 corrispondente, deve essere scritta una parte di codice nel file di configurazione che dica a BibConvert come formattare il contenuto del tag in esame, se presente, e cosa andare ad inserire nel file XML di output.
Tale codice è compreso tra:
HEAD---<record> e **FOOT---</record>** che dicono all'applicazione di inserire nel file di output i tag XML <record> e </record> rispettivamente, cioè il tag radice e il corrispettivo di chiusura (vedi paragrafo 6.4.4), che racchiuderanno dunque l'intero record.

Di seguito viene mostrato il codice scritto per un determinato tag di input, cercando in tal modo di far comprendere, in generale, l'intero codice redatto in questa sezione.

```
010::CONF(010,,0)---
<datafield tag="020" ind1="" ind2="">
  <:010*::010::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield code="a">):>
  <:010*::010::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::
    REP($$,):>
  <:010*::010::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
```

In particolare, il codice presentato sopra indica all'applicazione BibConvert la corrispondenza tra il tag UNIMARC 010 e il tag MARC 21 020.
Da notare che tale codice è stato spezzato in più righe e indentato per questioni di leggibilità. Nel file di configurazione ogni pezzo di codice per un determinato tag deve essere scritto su un'unica linea, viceversa l'applicazione non funzionerà correttamente.

Tenendo in considerazione quanto detto al paragrafo 9.2.3, circa la sintassi di una generica linea di definizione del layout, si può osservare innanzitutto l'indicazione del codice **010** ad inizio riga, per specificare il tag UNIMARC preso in considerazione.

Immediatamente dopo, si applica, come per ogni altro tag del record, ad esclusione dell'ultimo, il 980 (che contiene una stringa costante che va sempre specificata e che indica che il record è stato creato dallo SBA di Messina), la funzione **CONF**, per evitare di inserire nel record XML di output informazioni relative ad un campo non presente in quello di input.
La funzione infatti, confronta se il contenuto del campo specificato come primo parametro, in questo caso "010", è uguale a quello presente nel secondo, "". Il

terzo parametro posto a zero, indica a BibConvert di non inserire nel file di output alcun dato nel caso in cui la condizione è verificata, cioè se "010" non è presente.
In tal modo, solo se la condizione non si verifica, verrà inserita da BibConvert, nel file di output, la stringa costante **<datafield tag="020" ind1="" ind2="">**, seguita poi dal contenuto del tag opportunamente elaborato in base alle funzioni di formattazione applicate.

Da notare che per i campi di controllo, cioè quelli che vanno da 001 a 009, si utilizza un'altra stringa costante: **<controlfield tag="001">** per esempio, dove "controlfield" specifica appunto che si tratta di un campo di controllo e non di un campo dati. Per tali campi, successivamente a tale stringa, viene inserito il contenuto del tag originale corrispondente, senza specificare alcuna formattazione. Come si può vedere, non vengono inseriti nel file XML, né indicatori, né codici di sottocampo, in quanto tali campi sono gli unici che non ne contengono.

Per quanto concerne le operazioni successive da effettuare sul contenuto del tag UNIMARC di partenza, analizziamo il codice in tre passi successivi:

Passo 1:
Riguarda il codice di sottocampo da inserire, che seguirà il tag XML datafield precedentemente creato da BibConvert.
La relativa porzione di codice che è stata scritta per fare ciò è:

```
<:010*::010::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield code="a">):>
```

Essa dice all'applicazione di prelevare il contenuto del tag 010, ripetibile ("*"), ed aggiungervi in coda, tramite la funzione ADD, il suffisso "$$a".

Successivamente viene applicata la funzione LIMW che elimina tutto ciò che si trova a sinistra (L) del primo "$$a" incontrato. Infine la funzione IF controlla che tale contenuto in fase di elaborazione sia uguale a ciò che è specificato nel primo parametro, in questo caso "$$a". Se il confronto ha esito positivo sostituisce il contenuto con il secondo parametro, viceversa con il terzo.
BibConvert immagazzina poi tale valore nel file di output.

Per chiarire meglio tale modo di procedere, consideriamo un esempio analizzando i vari passi fin qui eseguiti.
Supponiamo che il contenuto del tag 010 di input sia: **$$a88-15-06306-4**
Dopo l'applicazione della funzione ADD(,$$a) diventa:
$$a88-15-06306-4$$a
In seguito alla funzione LIMW($$a,L): **$$a88-15-06306-4$$a**
Poi IF($$a,,<subfield code="a">):> trova che il contenuto è diverso da $$a è si avrà: **<subfield code="a">**, che verrà inserito nel file di output.
Viceversa, se il contenuto del tag 010 fosse: **$$d€10,00** si avrebbe:
prima: **$$d€10,00$$a**
poi: **$$a**
infine: "", in quanto il confronto dà esito positivo.

In tal modo, si è cercato di mantenere il comportamento del file di configurazione il più elastico possibile. Come si vede, infatti, questa strategia permette di inserire un determinato codice di sottocampo nel record di output, solo se veramente esiste il suo corrispondente nel record di input, piuttosto che mettere direttamente delle stringhe costanti; in quanto per un certo tag non vengono sempre specificati tutti i sottocampi.

Passo 2:
Fatto ciò bisognerà formattare il contenuto da inserire nel record di output per

quel determinato codice di sottocampo.

In funzione del codice seguente:

```
<:010*::010::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::REP($$,):>
```

BibConvert preleverà nuovamente il contenuto originale del tag 010, rielaborando le informazione contenute.
Prima viene aggiunto il suffisso "$$a" tramite la funzione ADD, in seguito LIMW cancellerà tutto ciò che si trova alla sinistra del primo "$$a" incontrato. Attraverso la funzione REP viene poi eliminata ogni occorrenza di "$$a", e conseguentemente LIMW col secondo parametro impostato ad "R", cancella tutto ciò che si trova a destra di un eventuale "$$" e una nuova chiamata di REP cancella il "$$" rimanente. In tal modo rimane solo il contenuto compreso tra il tag 010 $a ed un eventuale sottocampo successivo.

Si è cercato di realizzare un codice il più flessibile possibile, che permetta di gestire i vari casi che possono presentarsi, come ad esempio: l'assenza del sottocampo in esame (nell'esempio $a) all'interno del contenuto del campo, la presenza di più sottocampi, il sottocampo in esame è preceduto da altri sottocampi e dai rispettivi contenuti, ecc..

Ad una prima lettura potrebbe sembrare superfluo chiamare la funzione ADD(,$$a).
In realtà ciò è stato fatto poiché la funzione LIMW chiamata specificando come valore per il secondo parametro "L", piuttosto che "R", presenta un comportamento anomalo. Difatti, se applicata ad una stringa che non contiene almeno un'occorrenza del valore indicato come primo parametro, la funzione genera un errore che interrompe l'esecuzione di BibConvert; differentemente da

quanto avviene se il secondo parametro è “R” il cui funzionamento risulta essere corretto.
Si considera opportuno dunque che il codice di BibConvert, per quanto riguarda tale funzione, venga corretto dagli sviluppatori del CERN, eliminando tale bug.

Passo 3:
Infine, agendo in modo analogo a quanto fatto al passo 1, il codice seguente:

```
<:010*::010::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
```

dice a BibConvert di inserire, solo nel caso fosse presente il sottocampo $$a nel contenuto del campo 010, il relativo tag XML di chiusura subfield.

Eseguiti tali passi, viene infine aggiunto il tag </datafield> (o </controlfield> nel caso di campo di controllo) di chiusura a quello inizialmente aperto.

I passi 1, 2 e 3, qui analizzati vengono effettuati, in modo analogo, per tutti i campi di dati contenti uno o più sottocampi. In particolare, per questi ultimi, tali passi vengono ripetuti per ogni possibile sottocampo.
Le eventuali differenze riguardano il passo 2, in quanto, in funzione del possibile contenuto di un certo sottocampo si possono applicare ulteriori funzioni di formattazione.

9.5.5 File di configurazione

Di seguito viene mostrato l’intero file di configurazione realizzato per la risoluzione del compito assegnato dallo SBA di Messina.
Per motivi di leggibilità, le linee di codice della terza sezione, riguardanti la

conversione dei vari tag UNIMARC di input nei rispettivi tag MARC 21, verranno suddivise in più linee indentate.

```
### File di Configurazione di BibConvert
### per convertire record dal formato UNIMARC
### di Aleph500 al formato XML MARC 21


=== Definizione del record sorgente ===

001---001   L ---EOL------
005---005   L ---EOL------
010---010   L ---EOL---;---
011---011   L ---EOL------
1010x---1010  L ---EOL------
1011x---1011  L ---EOL------
102---102   L ---EOL------
2001x---2001  L ---EOL------
205---205   L ---EOL------
207x1---207 1 L ---EOL---;---
210---210   L ---EOL------
215---215   L ---EOL---;---
2250x---2250  L ---EOL---;---
2252x---2252  L ---EOL---;---
300---300   L ---EOL---;---
410x1---410 1 L ---EOL------
5121x---5121  L ---EOL---;---
610---610   L ---EOL---;---
6100x---6100  L ---EOL---;---
700x1---700 1 L ---EOL------
702x1---702 1 L ---EOL---;---
801x0---801 0 L ---EOL------
```

```
=== Definizione dei campi sorgente ===

001---<:001:>
005---<:005:>
010---<:010:>
011---<:011:>
1010x---<:101:>
1011x---<:1011x:>
102---<:102:>
2001x---<:2001x:>
205---<:205:>
207x1---<:207x1:>
210---<:210:>
215---<:215:>
2250x---<:2250x:>
2252x---<:2252x:>
300---<:300:>
410x1---<:410x1:>
5121x---<:5121x:>
610---<:610:>
6100x---<:6100x:>
700x1---<:700x1:>
702x1---<:702x1:>
801x0---<:801x0:>


=== Definizione del record di output ===

HEAD---<record>
001::CONF(001,,0)---
<controlfield tag="001"><:001::001:></controlfield>
005::CONF(005,,0)---
<controlfield tag="005"><:005::005:></controlfield>
010::CONF(010,,0)---
<datafield tag="020" ind1="" ind2="">
  <:010*::010::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield code="a">):>
  <:010*::010::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::
    REP($$,):>
```

```
  <:010*::010::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
011::CONF(011,,0)---
<datafield tag="022" ind1="" ind2="">
  <:011::011::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield code="a">):>
  <:011::011::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::
    REP($$,):>
  <:011::011::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
801x0::CONF(801x0,,0)---
<datafield tag="040" ind1="" ind2="">
  <:801x0::801x0::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:801x0::801x0::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::
    REP($$b, )::LIMW($$,R)::REP($$,)::REP(&,%26):>
  <:801x0::801x0::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
  <:801x0::801x0::ADD(,$$g)::LIMW($$g,L)::IF($$g,,<subfield
    code="e">):>
  <:801x0::801x0::ADD(,$$g)::LIMW($$g,L)::REP($$g,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:801x0::801x0::ADD(,$$g)::LIMW($$g,L)::IF($$g,,</subfield>):>
</datafield>
1010x::CONF(1010x,,0)---
<datafield tag="041" ind1="0" ind2="">
  <:1010x::1010x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:1010x::1010x::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:1010x::1010x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
1011x::CONF(1011x,,0)---
<datafield tag="041" ind1="1" ind2="">
  <:1011x::1011x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:1011x::1011x::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:1011x::1011x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
```

```
102::CONF(102,,0)---
<datafield tag="044" ind1="" ind2="">
  <:102::102::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield code="a">):>
  <:102::102::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:102::102::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
700x1::CONF(700x1,,0)---
<datafield tag="100" ind1="1" ind2="">
  <:700x1::700x1::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:700x1::700x1::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::
    REP($$b, )::LIMW($$,R)::REP($$,)::REP(&,%26):>
  <:700x1::700x1::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
2001x::CONF(2001x,,0)---
<datafield tag="245" ind1="1" ind2="">
  <:2001x::2001x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:2001x::2001x::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::REP(<<,)::
    REP(>>,)::LIMW($$,R)::REP($$,)::REP(&,%26):>
  <:2001x::2001x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
  <:2001x::2001x::ADD(,$$e)::LIMW($$e,L)::IF($$e,,<subfield
    code="b">):>
  <:2001x::2001x::ADD(,$$e)::LIMW($$e,L)::REP($$e,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:2001x::2001x::ADD(,$$e)::LIMW($$e,L)::IF($$e,,</subfield>):>
  <:2001x::2001x::ADD(,$$f)::LIMW($$f,L)::IF($$f,,<subfield
    code="c">):>
  <:2001x::2001x::ADD(,$$f)::LIMW($$f,L)::REP($$f,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:2001x::2001x::ADD(,$$f)::LIMW($$f,L)::IF($$f,,</subfield>):>
</datafield>
5121x::CONF(5121x,,0)---
<datafield tag="246" ind1="1" ind2="4">
  <:5121x*::5121x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:5121x*::5121x::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::REP(<<,)::
```

```
    REP(>>,)::LIMW($$,R)::REP($$,)::REP(&,%26):>
  <:5121x*::5121x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
205::CONF(205,,0)---
<datafield tag="250" ind1="" ind2="">
  <:205::205::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield code="a">):>
  <:205::205::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:205::205::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
  <:205::205::ADD(,$$f)::LIMW($$f,L)::IF($$f,,<subfield code="b">):>
  <:205::205::ADD(,$$f)::LIMW($$f,L)::REP($$f,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:205::205::ADD(,$$f)::LIMW($$f,L)::IF($$f,,</subfield>):>
</datafield>
210::CONF(210,,0)---
<datafield tag="260" ind1="" ind2="">
  <:210::210::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield code="a">):>
  <:210::210::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::REP(<<,)::
    REP(>>,)::LIMW($$,R)::REP($$,)::REP(&,%26):>
  <:210::210::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
  <:210::210::ADD(,$$c)::LIMW($$c,L)::IF($$c,,<subfield code="b">):>
  <:210::210::ADD(,$$c)::LIMW($$c,L)::REP($$c,)::REP(<<,)::
    REP(>>,)::LIMW($$,R)::REP($$,)::REP(&,%26):>
  <:210::210::ADD(,$$c)::LIMW($$c,L)::IF($$c,,</subfield>):>
  <:210::210::ADD(,$$d)::LIMW($$d,L)::IF($$d,,<subfield code="c">):>
  <:210::210::ADD(,$$d)::LIMW($$d,L)::REP($$d,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:210::210::ADD(,$$d)::LIMW($$d,L)::IF($$d,,</subfield>):>
</datafield>
215::CONF(215,,0)---
<datafield tag="300" ind1="" ind2="">
  <:215*::215::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield code="a">):>
  <:215*::215::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:215*::215::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
  <:215*::215::ADD(,$$d)::LIMW($$d,L)::IF($$d,,<subfield code="c">):>
  <:215*::215::ADD(,$$d)::LIMW($$d,L)::REP($$d,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
```

```
  <:215*::215::ADD(,$$d)::LIMW($$d,L)::IF($$d,,</subfield>):>
</datafield>
207x1::CONF(207x1,,0)---
<datafield tag="362" ind1="1" ind2="">
  <:207x1*::207x1::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:207x1*::207x1::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::REP(
    $$,)::REP(&,%26):>
  <:207x1*::207x1::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
2250x::CONF(2250x,,0)---
<datafield tag="490" ind1="1" ind2="">
  <:2250x*::2250x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:2250x*::2250x::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::REP(<<,)::
    REP(>>,)::LIMW($$,R)::REP($$,)::REP(&,%26):>
  <:2250x*::2250x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
  <:2250x*::2250x::ADD(,$$v)::LIMW($$v,L)::IF($$v,,<subfield
    code="c">):>
  <:2250x*::2250x::ADD(,$$v)::LIMW($$v,L)::REP($$v,)::LIMW($$,R)::REP(
    $$,)::REP(&,%26):>
  <:2250x*::2250x::ADD(,$$v)::LIMW($$v,L)::IF($$v,,</subfield>):>
</datafield>
2252x::CONF(2252x,,0)---
<datafield tag="490" ind1="1" ind2="">
  <:2252x*::2252x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:2252x*::2252x::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::REP(<<,)::
    REP(>>,)::LIMW($$,R)::REP($$,)::REP(&,%26):>
  <:2252x*::2252x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
  <:2252x*::2252x::ADD(,$$v)::LIMW($$v,L)::IF($$v,,<subfield
    code="v">):>
  <:2252x*::2252x::ADD(,$$v)::LIMW($$v,L)::REP($$v,)::LIMW($$,R)::REP(
    $$,)::REP(&,%26):>
  <:2252x*::2252x::ADD(,$$v)::LIMW($$v,L)::IF($$v,,</subfield>):>
</datafield>
300::CONF(300,,0)---
<datafield tag="500" ind1="" ind2="">
```

```
  <:300*::300::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield code="a">):>
  <:300*::300::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:300*::300::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
610::CONF(610,,0)---
<datafield tag="653" ind1="0" ind2="">
  <:610*::610::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield code="a">):>
  <:610*::610::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:610*::610::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
6100x::CONF(6100x,,0)---
<datafield tag="653" ind1="0" ind2="">
  <:6100x*::6100x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:6100x*::6100x::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::LIMW($$,R)::REP(
    $$,)::REP(&,%26):>
  <:6100x*::6100x::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
702x1::CONF(702x1,,0)---
<datafield tag="700" ind1="1" ind2="">
  <:702x1*::702x1::ADD(,$$a)::LIMW($$a,L)::IF($$a,,<subfield
    code="a">):>
  <:702x1*::702x1::ADD(,$$a)::LIMW($$a,L)::REP($$a,)::
    REP($$b, )::LIMW($$,R)::REP($$,)::REP(&,%26):>
  <:702x1*::702x1::ADD(,$$a)::LIMW($$a,L)::IF($$a,,</subfield>):>
</datafield>
410x1::CONF(410x1,,0)---
<datafield tag="760" ind1="0" ind2="">
  <:410x1::410x1::ADD(,$$v)::LIMW($$v,L)::IF($$v,,<subfield
    code="g">):>
  <:410x1::410x1::ADD(,$$v)::LIMW($$v,L)::REP($$v,)::LIMW($$,R)::
    REP($$,)::REP(&,%26):>
  <:410x1::410x1::ADD(,$$v)::LIMW($$v,L)::IF($$v,,</subfield>):>
</datafield>
980---
<datafield tag="980" ind1="" ind2="">
```

```
  <subfield code="a">RECORD SBA MESSINA</subfield>
</datafield>

FOOT---</record>

### La configurazione termina qui
```

Un esempio di record ottenuto in output da BibConvert utilizzando il file di configurazione mostrato sopra è il seguente:

```
<record>
  <controlfield tag="001">000001189</controlfield>
  <controlfield tag="005">20020320094625.0</controlfield>
  <datafield tag="020" ind1="" ind2="">
    <subfield code="a">88-15-06306-4</subfield>
  </datafield>
  <datafield tag="040" ind1="" ind2="">
    <subfield code="a">IT SBA Messina</subfield>
    <subfield code="e">RICA</subfield>
  </datafield>
  <datafield tag="041" ind1="0" ind2="">
    <subfield code="a">ita</subfield>
  </datafield>
  <datafield tag="044" ind1="" ind2="">
    <subfield code="a">IT</subfield>
  </datafield>
  <datafield tag="100" ind1="1" ind2="">
    <subfield code="a">Ferrera, Maurizio</subfield>
  </datafield>
  <datafield tag="245" ind1="1" ind2="">
    <subfield code="a">Le trappole del welfare</subfield>
    <subfield code="c">Maurizio Ferrera</subfield>
  </datafield>
  <datafield tag="246" ind1="1" ind2="4">
    <subfield code="a">
      Uno stato sociale sostenibile per l'Europa del 21. secolo
    </subfield>
```

```
  </datafield>
  <datafield tag="260" ind1="" ind2="">
    <subfield code="a">Bologna</subfield>
    <subfield code="b">Il Mulino</subfield>
    <subfield code="c">c1998</subfield>
  </datafield>
  <datafield tag="300" ind1="" ind2="">
    <subfield code="a">168 p.</subfield>
    <subfield code="c">21 cm</subfield>
  </datafield>
  <datafield tag="490" ind1="1" ind2="">
    <subfield code="a">Contemporanea</subfield>
    <subfield code="c">99</subfield>
  </datafield>
  <datafield tag="500" ind1="" ind2="">
    <subfield code="a">Segue: Appendice</subfield>
  </datafield>
  <datafield tag="653" ind1="0" ind2="">
    <subfield code="a">Europa</subfield>
  </datafield>
  <datafield tag="653" ind1="0" ind2="">
    <subfield code="a">Welfare state</subfield>
  </datafield>
  <datafield tag="760" ind1="0" ind2="">
    <subfield code="g">99</subfield>
  </datafield>
  <datafield tag="980" ind1="" ind2="">
    <subfield code="a">RECORD SBA MESSINA</subfield>
  </datafield>
</record>
```

Il corrispettivo record di input è quello mostrato al par. 9.5.3

9.5.6 Conversione e successivo caricamento dei record

Una volta realizzato il file di configurazione per la conversione dei record di input dal formato UNIMARC di Aleph 500 a quello XML MARC 21 di output, come descritto nei paragrafi precedenti, è necessario effettuare la procedura di conversione vera e propria eseguendo BibConvert e la successiva procedura per il caricamento dei record convertiti nel database locale di CDSware.

Attualmente BibConvert può essere configurato ed utilizzato solo attraverso linea di comando, dato che non è stata ancora realizzata dagli sviluppatori del CERN la sua interfaccia grafica.
Dunque, partendo dal presupposto che i record bibliografici prelevati da Aleph siano memorizzati in un file, sarà possibile effettuare la conversione di tali record, digitando in modalità superuser, la seguente linea di comando:

```
#bibconvert -cconvertSBA.cfg < sampleSBA.txt> outSBA.xml
```

dove:
bibconvert – é lo script dell'applicazione BibConvert che permette la creazione di un file che conterrà i record di metadati in formato XML secondo le specifiche definite nel file di conversione convertSBA.cfg;
c – rappresenta uno dei possibili parametri da passare a bibconvert ed indica che di seguito verrà specificato il file di configurazione;
convertSBA.cfg – é il file di configurazione creato, e descritto nei paragrafi precedenti, che indica a bibconvert la corrispondenza tra i vari tag presenti nel file di input e i relativi MARC 21 e la formattazione da applicare al loro contenuto;
sampleSBA.txt - é il file di input contenente i record prelevati da Aleph. Tale file viene presentato a bibconvert con un semplice reindirizzamento dell'input ("<");
outSBA.xml - é il file che, alla fine dell'elaborazione del modulo bibconvert, conterrà i record bibliografici nel formato XML MARC 21 ottenuti a partire da

quelli del file sampleSBA.txt. Tale file viene presentato a bibconvert con un semplice reindirizzamento dell'output (">").

A questa fase di trasformazione, segue la procedura di caricamento dei record convertiti nel database locale, che viene effettuata attraverso l'esecuzione dello script bibupload per il caricamento dei record bibliografici, attraverso:

```
#bibupload outSBA.xml
```

dove:
bibupload - é lo script che consente il caricamento di record bibliografici;
outSBA.xml - é il file ottenuto dall'elaborazione vista sopra, tramite l'esecuzione dello script bibconvert, che bibupload caricherà nel database di CDSware.

E' possibile inoltre, impostare la formattazione della rappresentazione in output dei record bibliografici caricati precedentemente attraverso il modulo BibFormat trattato nella tesi di Amelotti Ercole [B1], che può essere consultata per qualunque approfondimento in merito.

Una volta effettuata la procedura di caricamento nel sistema attraverso lo script bibupload, sarà possibile indicizzare i campi dei record bibliografici al fine di consentire a CDSware ricerche di record basate sugli indici creati.
Tale procedura può essere effettuata da linea di comando nel seguente modo:

```
#bibwords add 1189-14607 author,keyword
```

dove:
bibwords - é lo script che permette l'aggiornamento degli indici delle tabelle "wordsindex" e "wordsindex_field" del database di CDSware, in modo da consentire la ricerca per campi dei record bibliografici;

add 1189-14607 - tale opzione indica all'applicazione bibwords di aggiungere gli indici nelle tabelle "wordsindex" e "wordsindex_field" per i record bibliografici che vanno dal numero 1189 al 14607. Tale numero di record é espresso nel tag di controllo "001" dei record bibliografici;
author, keyword - rappresentano gli indici (separati da una virgola), cioè i campi dei record indicati sopra che dovranno essere indicizzati. Ovviamente alle parole author e keyword sono associati i corrispondenti tag MARC21 dei record di input, rispettivamente i tag "100 $a" e "6530 $a".

Alla fine di questa procedura sarà possibile ricercare i record bibliografici esportati attraverso l'interfaccia WebSearch di CDSware.

BIBLIOGRAFIA

[B1] Amelotti Ercole, *Protocollo OAI-PMH negli Open Archive e applicazione CDSware per la rappresentazione dei relativi dati bibliografici*, tesi di laurea in informatica, Università degli Studi di Messina, A.A. 2003-2004 (relatore Puccio L., correlatore De Robbio A.).

SITOGRAFIA

[S18] http://www.loc.gov/marc/unimarctomarc21.html