Funzionalità avanzate del sistema Biblatex/Biber

Ivan Valbusa

Sommario

In questo articolo si descrivono alcune funzioni poco note messe a disposizione dal pacchetto biblatex usato assieme al motore bibliografico Biber. È richiesta una conoscenza di base del funzionamento di questi programmi.

Abstract

This article describes some features available with the biblatex package when using Biber as the bibliographic engine. A basic knowledge of these programs is required.

1 Biber versus BibTEX

Nel 2010 Philipp Lehman ha rilasciato la prima versione stabile del pacchetto biblatex (Lehman, 2014) e da quel momento la composizione delle bibliografie con LaTeX ha intrapreso una nuova rotta.[1] Prima di allora non mancavano certo gli strumenti ma erano senza dubbio (troppo) poco elastici, in particolare per le esigenze degli utenti umanisti.

Il pacchetto di Lehman ha rivisto profondamente il modo di gestire la bibliografia e, in particolare, di modificare e creare stili bibliografici, attraverso un'interfaccia più intuitiva (usa direttamente le macro di LaTeX) rispetto alla complessa sintassi dei file di stile `.bst` (quelli, per intenderci, che si caricano con `\bibliographystyle`). Nell'arco di questi quattro anni sono nati parecchi stili bibliografici per biblatex e piano piano questo pacchetto è diventato il nuovo standard per la composizione delle bibliografie con LaTeX.

Ma biblatex da solo non è sufficiente per generare una bibliografia con le relative citazioni ed ha bisogno, esattamente come altri pacchetti (vedi il notissimo natbib), di un motore esterno che elabori gli archivi bibliografici e sovrintenda alla generazione delle etichette e all'ordinamento delle voci. Fino a poco tempo fa il motore bibliografico di riferimento era BibTEX, e con la sintassi prevista da questo motore si compongono ancora oggi molti archivi bibliografici. Con la diffusione di biblatex, soprattutto in ambito umanistico, sono però aumentate anche le esigenze e presto si è sentita la necessità di un nuovo motore bibliografico specificamente pensato per il potente pacchetto di Lehman. È nato così Biber (Kime e Charette, 2014), che è diventato ben presto il motore bibliografico predefinito di biblatex.[2] Un approccio alternativo è stato proposto già nel 2003 da Jean-Michel Hufflen con MlBibTEX, una reimplementazione di BibTEX con funzionalità multilingue, che è stato discusso in diversi contributi su ArsTEXnica e TUGboat. Si veda per esempio il recente Hufflen (2012).

A differenza del suo antesignano, Biber consente un controllo profondo dei dati contenuti all'interno degli archivi bibliografici: permette il *parsing* dei dati e la gestione avanzata dei riferimenti incrociati, di insiemi di voci (i cosiddetti *entry sets*) e di voci collegate (*related entries*). Inoltre permette di gestire in maniera elastica le liste di nomi (in particolare per i problemi di disambiguazione). Ha il supporto completo per la codifica Unicode. Prevede l'uso di opzioni *per-type* e molte altre funzioni. Nel seguito ne descriveremo alcune che riteniamo più significative e ancora poco note.

2 Il type `@xdata`

Alcune voci bibliografiche possono avere in comune uno o più dati. Per esempio, i volumi editi da una stessa casa editrice hanno i medesimi campi `publisher` e `location`. Con Biber si ha a disposizione lo speciale "tipo" di voce bibliografica `@xdata`, che funziona come un contenitore di dati che possono essere ereditati dalle altre voci attraverso l'apposito campo `xdata`.

Il sistema è molto utile, e la sua utilità aumenta con l'aumentare delle dimensioni dell'archivio bibliografico. In particolare permette di limitare al minimo i refusi, che sono sempre in agguato. Vediamo come funziona.

Innanzitutto creiamo una voce `@xdata` contenente solo due campi, nome dell'editore e luogo di edizione, e assegnamovi una chiave, esattamente come si fa con le altre voci standard:

```
@xdata{lat,
  Publisher = {Laterza},
  Location = {Roma-Bari}}
```

Dopodiché, nelle voci dello stesso editore, richiamiamo i dati attraverso il campo `xdata` che conterrà la chiave (in questo caso `lat`) assegnata alla voce `@xdata`:

1. Il pacchetto è ora mantenuto da Philip Kime, Audrey Boruvka e Joseph Wright.

2. Naturalmente, è ancora possibile usare BibTEX, passando al pacchetto l'opzione backend=bibtex.

```
@book{Cellucci:2008,
  Author = {Carlo Cellucci},
  Title = {Perché ancora la filosofia},
    xdata = {lat},
  Year = {2008}}

@book{casati:2012,
  Author = {Roberto Casati},
  Title = {Prima lezione di filosofia},
    xdata = {lat},
  Year = {2012}}
```

Il campo `xdata` può addirittura contenere una lista di chiavi che rimandano a più voci `@xdata`, come mostrato nella documentazione di biblatex:

```
@xdata{macmillan:name,
  publisher = {Macmillan},
}

@xdata{macmillan:place,
  location = {New York and London},
}

@xdata{macmillan,
  xdata = {macmillan:name,
                 macmillan:place},
}

@book{...,
  author = {...},
  title = {...},
  date = {...},
xdata = {macmillan},
}
```

Il funzionamento di questo campo è simile a quello dei campi `crossref` e `xref`, con la differenza che non stabilisce alcuna relazione del tipo *parent/child*, limitandosi semplicemente ad ereditare i dati. In questo senso assomiglia forse più alla funzione delle voci `@string`:

```
@string{jams = {J.~Amer. Math. Soc.}}

@article{bertram,
  author = {Bertram, Aaron and
    Wentworth, Richard},
  title = {Gromov invariants for
    holomorphic maps on Riemann surfaces},
  journaltitle = {jams},
  date = 1996,
  volume = 9,
  number = 2,
  pages = {529-571},
}
```

Un approccio consigliabile a tutti potrebbe essere di creare delle voci `@xdata` per ogni editore che si incontra, assegnandovi come chiave semplicemente il nome dell'editore, eventualmente abbreviato. Per esempio: `mulino` (il Mulino), `olms` (Georg Olms Verlag), `bollati` (Bollati Boringhieri), ecc. Come vedremo nella sezione 9, adottando questo approccio si potranno ottenere risultati ancor più performanti.

3 Riferimenti incrociati

Il campo `crossref` consente di gestire in maniera ottimale situazioni tipiche nel campo umanistico. Per esempio record relativi a parti di opere in uno o più volumi, come le voci `@inbook`/`@incollection`, `@bookinbook` e `@suppbook`/`@suppcollection`.

In situazioni del genere abbiamo una voce *parent*, poniamo un libro che raccoglie le opere di un autore (voce `@book`), che avrà un proprio campo `title`, e una voce *child*, poniamo un articolo (voce `@inbook`), che a sua volta avrà un proprio campo `title`. Nel momento in cui la voce *child* eredita i dati dalla voce *parent*, il titolo di quest'ultima dovrebbe diventare il *booktitle* della prima. BibTEX non è in grado di fare questa mappatura e di conseguenza nella voce *parent* è necessario duplicare il campo `title` nel campo `booktitle`.[3] Questa è solo una delle limitazioni di BibTEX e la conseguenza principale di tutto ciò è che in questo modo i record bibliografici diventano inutilmente "pesanti", contenendo di fatto numerosi e irrazionali doppioni, necessari solo per aggirare le limitazioni di BibTEX.

Con Biber queste limitazioni non esistono più, perché questo motore permette di definire delle regole molto flessibili che sovrintendono al sistema di *data inheritance*. Consideriamo il semplice caso seguente:

```
@Book{book,
  author = {Author},
  title = {Booktitle},
  publisher = {Publisher},
  location = {Location},
  date = {1995}}

@InBook{inbook,
  crossref = {book},
  title = {Title},
  pages = {5-25}}
```

Nella costruzione della voce `@inbook` Biber mappa automaticamente il campo `title` della voce `@book` nel campo `booktitle` della voce `@inbook` che vi si collega tramite `crossref`. Nel caso più semplice di una relazione `@book`/`@inbook` la regola (in forma semplificata) definita dal pacchetto è la seguente:

```
\DeclareDataInheritance{book}{inbook}{%
    \inherit{title}{booktitle}
    \inherit{subtitle}{booksubtitle}
  }
```

Ancora più importante è la possibilità di impedire che la voce *child* erediti un determinato campo. Consideriamo i seguenti record:

```
@collection{Facchinetti:2009,
  Booktitle = {Studies on English Modality},
  Date = {2009},
```

3. La cosa è da tempo nota, tanto che molti gestori di archivi bibliografici, tra cui BibDesk e JabRef, hanno opzioni che permettono di duplicare automaticamente i campi necessari.

```
  Editor = {Anastasios Tsangalidis
    and Roberta Facchinetti},
  Location = {Bern},
  Publisher = {Peter Lang},
  Title = {Studies on English Modality},
  Note = {The note of the parent entry}}

@incollection{Degani:2009,
  Author = {Marta Degani and Elisabetta
    Adami and Anna Belladelli},
  Crossref = {Facchinetti:2009},
  Pages = {13-54},
  Title = {The Use of Modal Verbs in
    Interpersonal Contexts}}
```

Se non forniamo al programma alcuna istruzione particolare, il campo `note` verrà ereditato dalla voce *child* (in questo caso `Degani:2009`) e il risultato sarà qualcosa di simile (quanto simile dipende dallo stile bibliografico scelto):

> Degani, Marta, Elisabetta Adami, and Anna Belladelli. «The Use of Modal Verbs in Interpersonal Contexts». In: *Studies on English Modality*. Ed. by Anastasios Tsangalidis and Roberta Facchinetti. **The note of the parent entry**. Bern: Peter Lang, 2009, pp. 13–54

Se al contrario introduciamo la seguente regola

```
\DeclareDataInheritance
  {collection}{incollection}{\noinherit{note}}
```

la voce `@incollection` non erediterà il campo `note` dalla voce `@collection`, generando questo risultato:

> Degani, Marta, Elisabetta Adami, and Anna Belladelli. «The Use of Modal Verbs in Interpersonal Contexts». In: *Studies on English Modality*. Ed. by Anastasios Tsangalidis and Roberta Facchinetti. Bern: Peter Lang, 2009, pp. 13–54

4 Il campo `ids`

Sarà capitato a molti di cambiare idea sull'etichetta assegnata a una voce bibliografica. Poniamo di aver usato la chiave `kant:1968` per indicare i *Kants Werke* e di accorgerci che, probabilmente, è più opportuno identificarli con la chiave `kant:werke`, anche per una maggiore leggibilità del sorgente. Certo, potremmo modificare la chiave direttamente nell'archivio, ma ciò comporterebbe che i file composti precedentemente, e che usano la prima chiave, non sarebbero più compilabili (a meno che non si modifichino tutte le citazioni manualmente).

Se si usa Biber esiste una soluzione razionale a questo inconveniente che, specialmente in ambito umanistico, è più frequente di quanto si possa immaginare. È infatti sufficiente inserire nel campo `ids` una lista di chiavi alternative, che funzionano come degli alias per la vera e propria *entrykey*, che deve in ogni caso essere presente:

```
@mvbook{kant:1968,
  ids = {kant:werke,kant:KW,kantswerke},
  Author = {Kant, Immanuel},
  Title = {Kants Werke. Akademie Textausgabe},
  Publisher = {Walter de Gruyter},
  Location = {Berlin},
  Year = {1968},
  Volumes = {9}}
```

Le chiavi alternative contenute nel campo `ids` permetteranno di richiamare la stessa voce indifferentemente con uno dei comandi

```
\cite{kant:1968}
\cite{kant:werke}
\cite{kant:KW}
\cite{kantswerke}
```

Questa funzionalità risulta particolarmente utile, per esempio, nel caso di lavori di gruppo, dove ciascun autore può verosimilmente aver usato chiavi diverse per le stesse fonti.

5 L'opzione `safeinputenc`

Come anticipato, Biber fornisce sopporto completo per la codifica Unicode e questa caratteristica lo rende lo strumento elettivo per l'uso con XeLaTeX o LuaLaTeX. Quando usiamo (pdf)LaTeX su file codificati Unicode, però, potremmo imbatterci in situazioni poco piacevoli. Come noto, il pacchetto `inputenc` (Jeffrey e Mittelbach, 2014), che in questo caso carichiamo con l'opzione `utf8`, non copre tutti i caratteri della tabella Unicode. Per rendersene conto è sufficiente compilare con (pdf)LaTeX questo semplice esempio minimo:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}

\begin{document}
  ș
\end{document}
```

Mentre il successivo esempio, se compilato con XeLaTeX, non restituisce alcun errore e mostra correttamente il carattere ș:

```
\documentclass{article}
\usepackage{fontspec}
  \setmainfont{Times New Roman}

\begin{document}
  ș
\end{document}
```

Quando compiliamo con (pdf)LaTeX ci si può quindi trovare nella situazione in cui alcuni caratteri inseriti nel file `.bib` non vengano riconosciuti, ottenendo tipicamente l'errore:

```
Package inputenc Error: Unicode char \u8:
  not set up for use with LaTeX
```

La soluzione drastica sarebbe di sostituire ogni occorrenza di un carattere non riconosciuto con

l'apposito comando LaTeX (e di fatto è quello che spesso si fa). Per esempio modificando tutti i caratteri ṣ in `\d{s}`. Così facendo, però, in un certo senso stiamo "depotenziando" il file `.bib` per venire incontro ai limiti di (pdf)LaTeX e se in futuro volessimo usare lo stesso file `.bib` con XeLaTeX, avremmo un archivio contenente complessi comandi al posto dei più leggibili caratteri Unicode.

In questo caso ci viene in aiuto l'opzione safeinputenc=true (o semplicemente safeinputenc), che forza l'opzione texencoding=ascii, ma solo nel caso sia caricato il pacchetto inputenc con l'opzione utf8.

```
\usepackage[utf8]{inputenc}
\usepackage[safeinputenc]{biblatex}
```

Grazie a quest'opzione, tutti i dati non-Ascii presenti nel file `.bib` verrano convertiti in formato Ascii. Per esempio, il carattere ṣ verrà convertito in `\d{s}`, permettendo quindi una corretta compilazione. Va però precisato che questo meccanismo funziona solo relativamente alle lettere latine con diacritici. Se infatti nel file `.bib` sono presenti lettere greche (ma anche cirilliche, ebraiche, arabe, giapponesi, ecc.) safeinputenc risulta inefficace.[4]

Si capisce quindi che questa soluzione è accettabile solo se nel file `.bib` ci sono pochi caratteri che causano il problema. Se invece si vuole avere un pieno supporto Unicode bisogna appunto rivolgersi a motori più evoluti come XeTeX o LuaTeX.

6 Formati non-BibTeX

L'utente LaTeX in genere compone i propri archivi bibliografici secondo la sintassi prevista da BibTeX, ma questa naturalmente non è l'unica sintassi attualmente disponibile. Vi sono infatti numerosi software di gestione della bibliografia che hanno un loro specifico formato e alcuni di questi sono anche molto diffusi. Naturalmente Biber riconosce la sintassi di BibTeX, ma con questo potente motore è possibile lavorare anche con archivi scritti in formati diversi.

Il comando `\addbibresource`, con il quale si carica un archivio bibliografico, accetta diverse opzioni, tra le quali l'opzione datatype, che permette di indicare il formato dell'archivio caricato. Per il momento i formati supportati, e ancora in fase sperimentale, sono i seguenti:

datatype=ris
: Formato RIS realizzato dalla Research Information Systems.[5]

datatype=zoterordfxml
: Formato Zotero RDF/XML. È la sintassi usata dall'estensione di Mozilla Firefox (notissimo *browser* multipiattaforma e *open source*).

datatype=endnotexml
: Formato Endnote XML. Si tratta di un software a pagamento molto diffuso in ambiente Windows.

Questa funzionalità si rivela utile nel caso in cui si debbano recuperare dati bibliografici da archivi *online* che non permettono di esportare in formato BibTeX ma, per esempio, solo in formato RIS, oppure nel caso di archivi condivisi tra più persone di un gruppo di ricerca. Un archivio in formato RIS, per esempio, si carica con

`\addbibresource[datatype=ris]{`⟨*filename*⟩`.ris}`

Per testare questa funzionalità si possono seguire questi due semplici passaggi:

1. Creiamo un semplice file in formato RIS,[6] che chiameremo `test.ris`, contenente un unico record bibliografico (lasciando una riga bianca sotto `ER -`!) identificato dalla chiave `test` (*tag* `ID`):

   ```
   TY  - BOOK
   ID  - test
   AU  - Lemmon, E.J.
   TI  - Beginning Logic
   PY  - 1965
   PB  - Thomas Nelson and Sons Limited
   ER  -
   (questa riga va lasciata vuota)
   ```

2. Compiliamo l'esempio minimale seguente

   ```
   % !TEX TS-program = pdflatex
   % !BIB TS-program = biber
   \documentclass{article}
   \usepackage[backend=biber]{biblatex}
   \addbibresource[datatype=ris]{test.ris}

   \begin{document}
   \cite{test}
     \printbibliography
   \end{document}
   ```

L'esempio è stato testato su una TeXLive 2014 aggiornata al 3 agosto 2014 e dovrebbe funzionare anche per le prossime versioni. A volte, tuttavia, possono sorgere situazioni apparentemente inspiegabili, spesso dovute a errori di sintassi nel file dell'archivio bibliografico. Per esempio, se non si lascia la riga vuota di cui abbiamo detto sopra, la voce bibliografica non verrà stampata.

7 Liste bibliografiche

Tra le funzionalità più utili di biblatex c'è senza dubbio la possibilità di creare facilmente una lista

4. Sono molto grato a Claudio Beccari, che mi ha fatto comprendere l'essenza di questo problema. Per un approfondimento sulla questione delle codifiche in LaTeX si rimanda a Beccari e Gordini (2014).

5. La gestione di questo formato in ConTeXt, attraverso il modulo publications, è stata recentemente trattata in Scarso (2014).

6. Per un'elenco completo dei *tag* RIS si rimanda alla pagina `http://en.wikipedia.org/wiki/RIS_(file_format)`.

delle sigle, grazie al comando `\printshorthands`. Questa lista viene generata recuperando il valore del campo `shorthand`, se presente, e componendovi a fianco l'intera voce bibliografica secondo il layout definito dall'ambiente shorthand impostato dal comando stesso (e modificabile con `\defbibenvironment`). Oltre alla possibilità di stampare la lista delle sigle, biblatex prevede un meccanismo generale che permette di creare una lista di qualsiasi campo presente nei record bibliografici (di fatto `\printshorthands` non è altro che un'istanza particolare di questo meccanismo più generale).

Immaginiamo di voler creare una lista delle abbreviazioni dei nomi delle riviste, un elemento paratestuale molto diffuso in testi di ambito giuridico, ma non solo. La *Rivista italiana di diritto e procedura penale*, per esempio, è generalmente siglata con RIDPP e un record di un articolo pubblicato su questa rivista dovrà avere almeno le seguenti righe:

```
@article{entrykey,
...
journaltitle = {Rivista italiana
  di diritto e procedura penale},
shortjournal = {ridpp},
...
}
```

Si noti che il campo `shortjournal` non ha alcuna formattazione, la quale viene gestita all'interno del file `.tex`, lasciando così l'archivio bibliografico "pulito" e disponibile anche per esigenze stilistiche differenti.

Per poter stampare la lista delle abbreviazioni delle riviste si deve innanzitutto creare un nuovo "tipo" di voce (tecnicamente un nuovo *driver*) che come unico compito abbia quello di stampare il campo `journaltitle`, ovvero il nome della rivista. Chiameremo questo driver `@shortjournal`:

```
\DeclareBibliographyDriver{shortjournal}{%
  \printfield{journaltitle}}
```

Fatto questo, per stampare la lista è sufficiente il comando

```
\printbiblist[title={Elenco abbreviazioni
    delle riviste citate}]{shortjournal}
```

Questo comando ha ha due argomenti. Il primo è opzionale e può contenere le stesse opzioni di `\printshorthands`; in particolare, in questo caso, il titolo da usare per la lista. Il secondo contiene il nome del driver appena definito, nome che viene usato anche per filtrare le voci che hanno il campo `shortjournal` ed, eventualmente, per formattare la lista delle abbreviazioni.

Il filtro è definito da `\DeclareBiblistFilter` e può essere modificato dall'utente in base alle proprie esigenze. La definizione di default che si ricava dal file `biblatex2.sty` corrisponde a

```
\DeclareBiblistFilter{shortjournal}{%
  \filter[type=field,filter=shortjournal]
  }
```

Sono possibili anche altri filtri più complessi, ma la documentazione di biblatex è ad oggi incoerente, visto che definisce il comando `\filter` in un modo diverso da come viene usato nel file di stile. In ogni caso, a titolo di esempio, possiamo creare un filtro per avere un elenco dei nomi delle sole riviste che appaiono in record che hanno il campo `shortjournal` ma non hanno il campo `series`:

```
\DeclareBiblistFilter{shortjournal}{%
\filter[type=field,filter=shortjournal]
\filter[type=notfield,filter=series]
}
```

Nell'esempio riportato nella figura 1 si noterà che l'unica voce contenente il campo `series` non compare nella lista delle abbreviazioni.

Il comando `\printbiblist`, analogamente a `\printshorthands`, racchiude la lista in un ambiente, che ha lo stesso nome assegnato al driver e al filtro. Il comando messo a disposizione dal pacchetto è il seguente:

```
\defbibenvironment{⟨name⟩}
  {⟨begin code⟩}
  {⟨end code⟩}
  {⟨item code⟩}
```

⟨*name*⟩ è il nome che assegnamo all'ambiente e allo stesso tempo la chiave da passare all'opzione `env` di `\printbiblist`; ⟨*begin code*⟩ e ⟨*end code*⟩ sono i comandi da far eseguire all'inizio e alla fine dell'ambiente (analogamente a quello che succede con il classico `\newenvironment`); ⟨*item code*⟩ contiene il codice da eseguire prima di ogni *entry* della lista. Per modificare l'ambiente di questa lista, ovvero per crearne uno nuovo, sono però necessarie anche altre macro. Di seguito vediamo un esempio molto semplice.

Immaginiamo di voler comporre le sigle in neretto e il titolo delle riviste in tondo. Siccome le sigle sono contenute nel campo `shortjournal`, sarà sufficiente usare

```
\DeclareFieldFormat{shortjournal}{%
  \bfseries#1}
```

Per avere i titoli in semplice tondo, ma solo nella lista delle abbreviazioni, servirà invece

```
\AtBeginBiblist{shortjournal}{%
\DeclareFieldFormat{journaltitle}{#1}
}
```

A questo punto definiamo un nuovo ambiente, che chiameremo shortjournal, con

```
\defbibenvironment{shortjournal}
  {\list{}{%
       \labelsep\biblabelsep
       \labelwidth=2cm
       \leftmargin\labelwidth
       \advance\leftmargin\labelsep
```

```
% !TEX TS-program = pdflatex
% !BIB TS-program = biber

\begin{filecontents}{printbiblist.bib}
@article{angenendt,
  author = {Angenendt, Arnold},
  title = {In Honore Salvatoris~--- Vom Sinn und Unsinn der Patrozinienkunde},
  journaltitle = {Revue d'Histoire Ecclésiastique},
  shortjournal = {rhe},
  date = 2002,
  volume = 97,
  pages = {431-456, 791-823}}
@article{gillies,
  Author = {Gillies, Alexander},
  date = 1933,
  Journaltitle = {Publications of the English Goethe Society},
  Shortjournal = {pegs},
  Series = {newseries},
  Title = {Herder and the Preparation of Goethe's Idea of World Literature},
  Volume = 9}
@article{sigfridsson,
  Author = {Sigfridsson, Emma and Ryde, Ulf},
  date = 1998,
  Journaltitle = {Journal of Computational Chemistry},
  Shortjournal = {jcc},
  Title = {Comparison of methods for deriving atomic charges
    from the electrostatic potential and moments},
  Volume = 19}
\end{filecontents}

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[english,italian]{babel}
\usepackage[style=philosophy-verbose]{biblatex}
  \addbibresource{printbiblist.bib}

\DeclareBibliographyDriver{shortjournal}{\printfield{journaltitle}}
\DeclareFieldFormat{shortjournal}{\scshape\bfseries#1}
\AtBeginBiblist{shortjournal}{\DeclareFieldFormat{journaltitle}{#1}}

\defbibenvironment{shortjournal}
  {\list{}{%
      \labelsep\biblabelsep
      \labelwidth=2cm
      \leftmargin\labelwidth
      \advance\leftmargin\labelsep
      \itemsep\bibitemsep
      \parsep\bibparsep
    \let\makelabel\printshortjournal}}
  {\endlist}
  {\item}
\newcommand*{\printshortjournal}{\printfield{shortjournal}}

\DeclareBiblistFilter{shortjournal}{%
  \filter[type=field,filter=shortjournal]
   \filter[type=notfield,filter=series]}

\begin{document}
\nocite{*}
\printbiblist[env=shortjournal,title={Elenco abbreviazioni delle riviste citate}]{shortjournal}
  \printbibliography
\end{document}
```

Codice 1: Esempio per la creazione di una lista delle abbreviazioni delle riviste citate.

Elenco abbreviazioni delle riviste citate

RHE Revue d'Histoire Ecclésiastique
JCC Journal of Computational Chemistry

Riferimenti bibliografici

Angenendt, Arnold, "In Honore Salvatoris – Vom Sinn und Unsinn der Patrozinienkunde", *Revue d'Histoire Ecclésiastique*, vol. 97 (2002), p. 431-456, 791-823.

Gillies, Alexander, "Herder and the Preparation of Goethe's Idea of World Literature", *Publications of the English Goethe Society*, nuova ser., vol. 9 (1933).

Sigfridsson, Emma e Ulf Ryde, "Comparison of methods for deriving atomic charges from the electrostatic potential and moments", *Journal of Computational Chemistry*, vol. 19 (1998).

FIGURA 1: Output del codice 1.

```
      \itemsep\bibitemsep
      \parsep\bibparsep
    \let\makelabel\printshortjournal}}
  {\endlist}
  {\item}
\newcommand*{\printshortjournal}{%
  \printfield{shortjournal}}
```

Abbiamo dovuto definire, come di consueto, lo stile dell'etichetta di ogni *item* (`\printshortjournal`), che in questo caso consiste semplicemente nel campo `shortjournal`. Come si può notare, nella definizione dell'etichetta non abbiamo usato nessuna macro di formattazione, che abbiamo invece passato precedentemente a `\DeclareFieldFormat`.

Questo approccio è consigliabile soprattutto se stiamo definendo un file di stile, in quanto consente all'utente finale di apportare facilmente, nel caso servisse, modiche stilistiche. Se per esempio si volessero le sigle in maiuscolo basterebbe

```
\DeclareFieldFormat{shortjournal}
  {\MakeUppercase{#1}}
```

A questo punto per stampare la lista nel formato appena definito sarà sufficiente aggiungere l'opzione env a `\printbiblist`:

```
\printbiblist[env=shortjournal,
  title={Elenco abbreviazioni delle riviste
    citate}]
```

Oltre allo stile delle sigle, questo nuovo ambiente aumenta a 2 cm la larghezza del box che contiene la sigla stessa. È possibile ovviamente personalizzare anche l'intestazione attraverso il comando `\definebibheading`, analogamente a quello che si fa con le classiche bibliografie.

Il listato 1 mostra un codice mimino compilabile contenente i comandi descritti in questa sezione, mentre nella figura 1 si può vedere il risultato finale.

8 Il campo `related`

Una voce bibliografica, nella sua veste più semplice, corrisponde a una fonte, sia essa un volume, un articolo o altro. Consideriamo il caso di un volume in lingua francese del quale è disponibile anche una traduzione italiana. A rigor di termini si tratta di due fonti distinte, ma è consuetudine comporle in un'unica voce. Una delle soluzioni più usate, almeno in Italia, è mostrata nell'esempio seguente ed è implementata negli stili del pacchetto biblatex-philosophy (VALBUSA, 2014):

> Jules-Henri Poincaré (1968), *La science et l'hypothèse*, Flammarion, Paris; trad. it *La scienza e l'ipotesi*, a cura di Corrado Sinigaglia, Bompiani, Milano 2003.

Con biblatex-philosophy abbiamo due meccanismi diversi per creare voci simili a quella mostrata in precedenza. Il primo prevede di creare un'unica voce che contenga tutti i dati necessari, usando campi particolari per i dati della traduzione:

```
@book{Poincare:1968-ORIG,
   Author = {Jules-Henri Poincaré},
   Title = {La science et l'hypothèse},
   Date = {1968},
   Location = {Paris},
```

```
  Publisher = {Flammarion},
  Transtitle = {La scienza e l'ipotesi},
  Transnote = {a cura di Corrado Sinigaglia},
  Transpublisher = {Bompiani},
  Translocation = {Milano},
  Transdate = {2003}}
```

Questa soluzione, sebbene possa andar bene per la maggior parte dei casi, ha dei limiti importanti. Innanzitutto i campi per la traduzione sono limitati, quindi se avessimo bisogno di inserire altre informazioni, le dovremmo mettere, con qualche stratagemma, in uno dei campi disponibili. Nell'esempio sopra si è usato il campo `transnote` perché non sono disponibili campi specifici per il curatore, il traduttore, il prefatore, ecc. della traduzione, e tutte queste figure vanno eventualmente inserite in questo campo. (Per inciso, i nuovi campi `trans*` sono stati definiti grazie a un comando che richiede Biber e che vedremo fra poco.)

A partire dalla versione 2.5, tuttavia, biblatex fornisce un meccanismo molto più potente e razionale se si usa Biber come motore bibliografico. Testo originale e traduzione vengono trattati come voci bibliografiche autonome (in effetti si tratta generalmente di due "oggetti" distinti) e il primo record richiama la voce collegata attraverso il campo `related`:

```
@book{Poincare:1968-ORIG,
  author = {Jules-Henri Poincaré},
  title = {La science et l'hypothèse},
  publisher = {Flammarion},
  location = {Paris},
  date = {1968},
  related = {Poincare:1968-ITA}}

@book{Poincare:1968-ITA,
  author = {Jules-Henri Poincaré},
  editor = {Corrado Sinigaglia},
  title = {La scienza e l'ipotesi},
  publisher = {Bompiani},
  location = {Milano}}
```

Poiché le voci sono indipendenti, con questo sistema abbiamo a disposizione tutti i campi previsti da biblatex, anche per la voce collegata. Ma con questo sistema si può fare molto di più.

Grazie al campo `relatedstring` possiamo cambiare la stringa di default ("trad. it", per i documenti in lingua italiana) che precede la voce collegata, in una stringa a piacere. Per esempio "prima ed. it.":

```
@book{Poincare:1968-ORIG,
   author = {Jules-Henri Poincaré},
   title = {La science et l'hypothèse},
   publisher = {Flammarion},
   location = {Paris},
   date = {1968},
   related = {Poincare:1968-ITA},
   relatedstring = {prima ed. it.}}
```

Inoltre, a partire dalla versione 1.6 di Biber le potenzialità sono ancora maggiori. Si possono per esempio creare voci "a cascata", anche molto complesse, in cui inserire i dati dell'edizione originale e di due diverse traduzioni:

> Popper, Karl R. 1934 *Logik der Forschung*, Springer, Wien; trad. ingl. *The Logic of Scientific Discovery*, 3ª ed., Hutchinson, London 1959; trad. it. *Logica della scoperta scientifica*, 3ª ed., Einaudi, Torino 1998.

I record concatenati che generano la voce appena citata sono i seguenti:

```
@book{popper-logik,
  Author = {Karl R. Popper},
  Date = {1934},
  Location = {Wien},
  Publisher = {Springer},
  Related = {popper-logik:ing},
  Relatedstring = {trad. ingl.},
  Title = {Logik der Forschung}}

@book{popper-logik:ing,
  Author = {Karl R. Popper},
  Date = {1959},
  Edition = {3},
  Location = {London},
  Publisher = {Hutchinson},
  Related = {popper-logik:ita},
  Title = {The Logic of Scientific Discovery}}

@book{popper-logik:ita,
  Author = {Karl R. Popper},
  Date = {1998},
  Edition = {3},
  Hyphenation = {italian},
  Location = {Torino},
  Publisher = {Einaudi},
  Title = {Logica della scoperta scientifica}}
```

Gli stili di biblatex-phylosophy permettono di comporre bibliografie commentate, attraverso il campo `annotation`. Nel caso queste voci avessero anche questo campo, verrebbe in ogni caso stampato solo quello della voce madre (ovvero la prima che contiene il campo `related`). Inoltre, di default le voci collegate non vengono incluse nella bibliografia a meno che non siano a loro volta citate.

Sono a disposizione anche altri campi per la personalizzazione delle voci "related". Per esempio, il campo `relatedtype` prevede di inserire una stringa di localizzazione definita nel file `.lbx` o attraverso `\DefineBibliographyStrings`. Il campo `relatedtype` negli stili biblatex-philosophy è impostato di default sulla stringa `translationas` (sic) definita come "trad. it.". Il campo `relatedstring` permette quindi di sovrascrivere la stringa specificata con `relatedtype`. Da questo punto di vista le due sintassi seguenti sono equivalenti:

```
Relatedstring = {trad. ingl.},
Relatedtype = {translationas},
```

Il vantaggio della seconda forma è che se si sta componendo un testo in inglese, la stringa usata

sarà automaticamente "trans.", sempre che la stringa sia stata definita nel file di localizzazione per la lingua inglese.

9 Modificare in modo dinamico i record bibliografici

Veniamo infine a quello che può essere considerato la vera bacchetta magica del sistema biblatex/Biber. Poniamo di avere un archivio molto corposo, nel quale in ciascuna voce di tipo `@book` abbiamo inserito nome dell'editore e luogo di edizione. Se ci viene richiesto di eliminare, per esempio, il nome dell'editore, dovremmo intervenire massicciamente nel file `.bib`, compromettendo in maniera seria l'archivio. Ma ci potrebbe venir richiesto di eliminare il nome dell'editore solo per un sottoinsieme specifico di voci in archivio.

Grazie a Biber è possibile agire sulle singole voci direttamente dal sorgente `.tex`, senza nemmeno aprire il file `.bib`. Si può decidere di eliminare il campo `publisher` per tutte le voci in archivio, solo per alcuni tipi, solo per alcuni archivi, ecc. Le possibilità sono pressoché infinite.

Visto che i comandi previsti per Biber accettano anche le espressioni regolari, si possono applicare filtri ancora più potenti. Questo codice, per esempio, permette di eliminare il campo `publisher` solo per le voci il cui campo `date` è un numero maggiore di 1900:

```
\DeclareSourcemap{
\maps[datatype=bibtex]{
\map{
\step[fieldsource=date,
match=\regexp{(1[^9]\d{2})},final]
\step[fieldset=publisher, null]
    }
  }
}
```

Il primo comando `\step` cerca in tutti gli archivi bibliografici caricati nel preambolo con `\addbibresource` le voci in cui il campo `date` corrisponde al valore di `match`, che è appunto un'espressione regolare. L'espressione regolare individuerà i numeri di 4 cifre che iniziano per 1 e che non hanno 9 come seconda cifra, ovvero tutti gli anni dal 1000 al 1899.[7] Il secondo `\step` elimina il campo `publisher` (attraverso la chiave `null`) delle voci individuate dal primo `\step`.

Con questo comando si possono anche creare nuovi campi, oppure aggiungere una *keyword* solo per quelle voci che rispondono a una determinata condizione. Poniamo di voler inserire le voci pubblicate fino al 1900 in una bibliografia indipendente. Le soluzioni consuete consistono o nell'usare delle categorie o nell'inserire una parola chiave all'interno dei record interessati. Entrambe queste strade comportano di individuare manualmente tutte le voci da includere e in alcuni casi ciò potrebbe essere frustrante, oltre che rischioso, potendo sempre dimenticare qualche voce.

Grazie a `\DeclareSourcemap` possiamo risolvere il tutto con questo "semplice" codice:

```
\DeclareSourcemap{
\maps[datatype=bibtex]{
\map{
\step[fieldsource=date,
match=\regexp{(1[^9]\d{2})},final]
\step[fieldset=keywords,
      fieldvalue={ottocento}]
    }
  }
}
```

e stampare quindi la bibliografia con

```
\printbibliography[
  title={Bibliografia (--1899)},
  keyword=ottocento]
```

Vediamo ora un caso di applicazione di `\DeclareSourcemap` ancora più interessante. Può succedere che in una voce bibliografica ci siano troppe informazioni, ovvero troppi campi, relativamente alla bibliografia che stiamo componendo. È da escludere, per i motivi detti sopra, di "emendare" la voce in archivio: se abbiamo inserito dei dati è un peccato elminarli perché in futuro ci potrebbero servire.

Fortunatamente con Biber si possono non solo eliminare (o aggiungere) campi specifici per insiemi di voci che rispondono a determinati criteri, ma anche per singole voci. Se ci si pensa bene, la chiave univoca assegnata a ogni voce, ovvero la *entrykey*, è di fatto uno speciale campo e non a caso la documentazione di biblatex, lo inserisce, per primo, nella sezione "Generic Fields". Ciò significa che se si può ricercare il contenuto di un determinato campo, come visto sopra, si può anche ricercare il contenuto del campo `entrykey`, ovvero il nome della chiave assegnata a una specifica voce. Vediamo innanzitutto il comando:

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \step[fieldsource=entrykey,
            match=\regexp{key1},
            fieldset=note,null]
    }
  }
}
```

La chiave `fieldsource=entrykey` predispone alla ricerca nella chiavi delle singole voci; `match=\regexp{key1}` individua la voce etichettata con `key1`. Infine, `fieldset=note,null` elimina da quella voce il campo `note`. Questo è forse lo strumento più potente di Biber, perché permette di agire su singole voci in maniera precisa, visto che le *entrykey* devono essere univoche.

7. Per una prima introduzione all'uso delle espressioni regolari si può fare riferimento alla pagina `http://it.wikipedia.org/wiki/Espressione_regolare`.

Vediamo un altro esempio che mostra l'utilità di questo sistema. Immaginiamo che ci venga richiesto di modificare il titolo breve (campo `shorttitle`) solo di alcune specifiche voci in archivio. Come sempre, vogliamo evitare di mettere mano all'archivio stesso. Servirà un codice simile al precedente, ma con un'aggiunta fondamentale:

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map[overwrite]{
      \step[fieldsource=entrykey,
            match=\regexp{key2}]
     \step[fieldset=shorttitle,
            fieldvalue={The new short title}]
    }
  }
}
```

Come si può notare, è stata passata al comando `\map` l'opzione overwrite (equivalente a overwrite=true). Questa opzione, infatti, è impostata di default su false per impedire che, se un campo contiene già dei dati, possa essere sovrascritto ovvero modificato (anche se può essere eliminato, come abbiamo visto precedentemente). Attivandola, quindi, viene permessa la sovrascrittura.

Un'altra funzionalità molto utile messa a disposizione da `\DeclareSourcemap` consiste nella possibilità di usare per i campi nomi nuovi definiti dall'utente stesso. Immaginiamo di dover stampare una determinata giurisprudenza, ovvero un elenco di sentenze. Le sentenze sono delle fonti analoghe alle altre e pertanto le si possono trattare come normali record bibliografici. Ecco un esempio di una voce di una giurisprudenza nazionale:

Cons. Stato, Sez. V, 28 set. 1980, n. 713

Qui l'autore è una precisa corte (in questo caso il Consiglio di Stato), inoltre v'è un'informazione specifica, ovvero l'indicazione della sezione. L'obiettivo finale è quello di poter scrivere un record con la seguente sintassi:

```
@jurisdiction{cstat:713/1980,
  Court = {Cons. Stato},
  Section = {5},
  Eventdate = {1980-09-28},
  Number = {713},
  Date = {1980},
  Volume = {1},
  Pages = {1499-}}
```

Innanzitutto dobbiamo creare il nuovo driver `@jurisdiction`, con eventuali macro di supporto, attraverso

```
\DeclareBibliographyDriver{jurisdiction}
{...}
```

Chi fosse interessato può leggere il codice nel file `philosophy-standard.bbx`, reperibile nella distribuzione LaTeX, all'interno della cartella `texmf-dist/tex/latex/biblatex-philosophy`.

I campi `court` e `section` non sono definiti nativamente da biblatex. Per poterli usare dobbiamo quindi mapparli su dei campi riconosciuti. Scegliamo quindi di mappare il primo nel campo `author` e il secondo nel campo `nameaddon`:

```
\DeclareStyleSourcemap{
\maps[datatype=bibtex]{
    \map{
     \step[fieldsource=court,
       fieldtarget=author]
     \step[fieldsource=section,
       fieldtarget=nameaddon]
      }
    }
  }
```

In questo modo il record sopra è già perfettamente utilizzabile. Si devono soltanto formattare i campi vecchi e nuovi, per ottenere il risultato finale atteso. Per il campo `nameaddon`, per esempio, servirà

```
\DeclareFieldFormat[jurisdiction]{nameaddon}
  {\ifinteger{#1}{%
    \bibcpsstring{section}~\RN{#1}}{#1}%
}
```

In questo modo, inserendo nel campo `nameaddon` un numero intero n, si avrà nell'output la stringa "Sez. n", con n in caratteri romani maiuscoli (`\RN`). Se invece si vuole indicare, per esempio, una "Adunanza plenale", si scriverà semplicemente

```
nameaddon = {Ad\adddotspace plen\adddot}}
```

Nella sezione 2 si è mostrata una modalità di creazione di record bibliografici attraverso il campo `xdata` molto promettente. Ora che abbiamo compreso, o almeno intuito, il funzionamento di `\DeclareSourcemap` possiamo mostrare una soluzione ancora più potente.

Immaginiamo di voler creare un archivio di editori, composto essenzialmente da voci `@xdata`. Sarebbe molto comodo assegnare a questo tipo di voci un nome più consono, per esempio `@mypublisher` e, analogamente, avere al posto del campo `xdata` un campo `mypublisher`. Riprendendo l'esempio che abbiamo mostrato precedentemente in 2, vorremmo pertanto poter definire uno scenario del genere:

```
@mypublisher{laterza,
  Publisher = {Laterza},
  Location = {Roma-Bari}}

@book{Cellucci:2008,
  Author = {Carlo Cellucci},
  Title = {Perché ancora la filosofia},
  mypublisher = {laterza},
  Year = {2008}}

@book{casati:2012,
  Author = {Roberto Casati},
  Title = {Prima lezione di filosofia},
  mypublisher = {laterza},
  Year = {2012}}
```

Perché la soluzione funzioni bisogna far conoscere a biblatex il nuovo driver e il nuovo campo. Per il primo è sufficiente:

```
\DeclareBibliographyAlias{mypublisher}{xdata}
```

Ma non è possibile far riconoscere il nuovo campo `mypublisher`, come ci si aspetterebbe, con

```
\DeclareFieldAlias{mypublisher}{xdata}
```

È necessario pertanto mappare il nuovo campo su `xdata` attraverso

```
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \step[fieldsource=mypublisher,
            fieldtarget=xdata]
    }
  }
}
```

10 Conclusione

In questo contributo abbiamo mostrato una piccolissima porzione delle potenzialità del sistema biblatex/Biber, ma si spera che almeno gli esempi riportati possano essere di spunto per nuove e più ingegnose soluzioni. In sede conclusiva mi permetto una riflessione sul senso "filosofico" di tutto questo.

Il sistema che abbiamo trattato si è sviluppato in modo così imponente principalmente per venire incontro alle esigenze degli umanisti. Il fatto che abbia un'infinità di funzioni e comandi è rassicurante, perché possiamo dire, quasi a priori, che probabilmente c'è una soluzione per ogni problema (bibliografico). Il vero limite, tuttavia, è che per scoprire *ogni* soluzione serve una conoscenza veramente profonda del programma stesso. E spesso, lo si deve ammettere, il gioco non vale la candela.

Come tutti i sistemi di gestione bibliografica, anche con biblatex si è adottato un approccio del tipo *bottom-up*. Invece di sforzarsi a rendere il più possibile razionali le pretese di certi utenti/autori/editori, eventualmente cassandole, le si sono assecondate, con la conseguenza che molti programmi (non solo biblatex) sono diventati troppo ricchi e troppo poco funzionali, rischio che sempre si corre quando si vuole accontentare tutti. Ma l'irrazionale, alla fine, sfugge sempre anche alla logica più potente e non è da escludere che vi siano esigenze umanistiche che non troveranno mai una soluzione con i meccanismi seppur vari previsti da biblatex.

È da augurasi che in futuro chi detiene il potere di stabilire le usanze, le consuetudini e le norme abbia la forza di fare un passo indietro, favorendo in questo modo il perfezionamento di sistemi razionali di gestione della bibliografia, che già esistono ma che rimangono confinati in una nicchia perché ognuno cerca di perseguire il meglio per sé piuttosto che il bene di tutti.

Riferimenti bibliografici

Beccari, C. e Gordini, T. (2014). *Codifiche in TEX e LATEX*. URL `http://www.guitex.org/home/images/doc/GuideGuIT/introcodifiche.pdf`.

Hufflen, J.-M. (2012). «Beyond BibTEX». *ArsTEXnica*, (14), pp. 7–14.

Jeffrey, A. e Mittelbach, F. (2014). «Il pacchetto inputenc». Versione 1.2b.

Kime, P. e Charette, F. (2014). «biber. a backend bibliography processor for biblatex». URL `http://biblatex-biber.sourceforge.net`. Versione 1.9.

Lehman, P. (2014). «The biblatex package». URL `http://www.ctan.org/tex-archive/macros/latex/exptl/biblatex/`. Versione 2.9a.

Scarso, L. (2014). «Un primo sguardo al modulo publications». *ArsTEXnica*, (17), pp. 29–36.

Valbusa, I. (2014). «Il pacchetto biblatex–philosophy». Disponibile su CTAN all'indirizzo: `http://www.ctan.org/tex-archive/macros/latex/exptl/biblatex-contrib/biblatex-philosophy/`.

▷ Ivan Valbusa
Dipartimento di
Filosofia, Pedagogia e Psicologia
Università degli Studi di Verona
`ivan dot valbusa at univr dot it`