Chapter 73
VuFind:
Solr Power in the Library

Demian Katz
Villanova University, USA

Andrew Nagy
Serials Solutions, USA

ABSTRACT

Apache Solr, an open source Java-based search engine, forms the core of many Library 2.0 products. The use of an index in place of a relational database allows faster data retrieval along with key features like faceting and similarity analysis that are not practical in the previous generation of library software. The popular VuFind discovery tool was built to provide a library-friendly front-end for Solr's powerful searching capabilities, and its development provides an informative case study on the use of Solr in a library setting. VuFind is just one of many library packages using Solr, and examples like Blacklight, Summon, and the eXtensible Catalog project show other possible approaches to its use.

INTRODUCTION

Libraries have been using electronic systems to manage their holdings for decades. Often, this has been in the form of monolithic "Integrated Library Systems," or ILS's, which handle all of a library's needs, from acquisitions to cataloging to end-user searching, in a single package. This one-system-serves-all approach worked well for quite some time, but in recent years, the pace of technological change has increased, driven by the rapid growth of the online environment and the changing demands of the library community. Ubiquitous Web technologies, particularly search engines like Google, have changed the expectations of library patrons, who are now more likely to take a self-service-oriented approach to searching and to expect to be able to access resources 24/7 without librarian assistance.[1]

These needs have sparked a new market of products in the library space, known most commonly as "Library Discovery Tools." Products from the first generation of discovery tools are also known as Next Generation Catalogs, showing how they build on the traditional library catalog or OPAC (Online Public Access Catalog). These products have begun to take end-user search functionality away from the ILS and treat it as a separate ap-

DOI: 10.4018/978-1-4666-7230-7.ch073

plication. As a result, users have begun to benefit from faster, friendlier search functionalities while system administrators have gained a much greater level of control over the behavior of their public offerings.

Much of the discovery revolution can be credited to a single piece of software: Apache Solr, an open source search engine that provides a solid foundation for nearly any type of search application. This chapter takes a look at the functionality that makes Apache Solr so useful and examines some of the decisions that went into building VuFind, a popular Solr-powered discovery tool. Since VuFind is not the only discovery tool on the market, we will also examine several other products, which use Solr in slightly different ways.

INDEXES VS. DATABASES

At the core of your typical Integrated Library System, you are likely to find a relational database. Relational databases are extremely useful tools, which efficiently store and retrieve information by breaking data down into tables of granular data and keeping track of how these tables relate to one another. With such a database, it is quite simple to model many of the relationships that can be found in libraries. As a simplistic example, one database table might represent books, other authors, and other patrons. Additional tables could then track relationships–which authors wrote which titles, or which patrons are currently borrowing which books. The beauty of the relational model is that each key piece of data (book, author, patron) is stored only once. When you need to look up information, you join these pieces together using the known relationships in order to get the answers you seek. If you need to change a piece of data, you edit it in just one place, and thanks to all of the relationships, the update automatically affects every context in which the data might be accessed.

Although relational databases remain an important part of the library software landscape, they do have one significant weakness: they are not ideal for search applications, particularly the sort of search engines that users of the Web have become accustomed to–very fast and very accurate. While database systems usually have built-in indexing and can do certain types of well-defined searches very quickly, their performance suffers when it comes to complex, multi-field, or full-text searching. Their data model, with everything spread out across multiple tables, becomes a disadvantage when you want to search through everything quickly–data needs to be reconstituted before it can be searched, and the result is slower performance.

The answer to this problem is an index. An index turns the whole thing upside-down. Rather than concerning itself with efficient storage of unique values, an index instead focuses on fast retrieval of information stored in a heavily pre-processed format. Data is stored redundantly to ease fast lookup, and text from indexed records and user search queries may be analyzed in a variety of ways to increase the probability of finding matches. Obviously, an index is likely to use more memory and disk space than an equivalent relational database, but the benefits include faster, more flexible lookup and powerful results analysis (such as faceting) which are impractical using a relational database.

From the outside, a relational database and an index look very similar: both are tools for storing and retrieving data. However, each has its own strengths and weaknesses. A relational database remains a better tool for creating and managing information, but an index is unquestionably more powerful for finding things. Integrated Library Systems have traditionally been weak in search functionality because they have built all of their functionality on a single database-driven foundation. The new generation of Library Discovery Tools seeks to replace the weakest link in the ILS chain by layering index-based searching on top of the existing database infrastructure. None of these tools intend to completely replace the ILS,