Creating a Massive Master Index for HTML and Print

Dan Scott
IBM Canada, 8200 Warden Avenue
Markham, Ontario, Canada
(905) 413-3170
dan.scott@ca.ibm.com

Ronnie Seagren
IBM Canada, 8200 Warden Avenue
Markham, Ontario, Canada
(416) 313-1029
seagren@ca.ibm.com

ABSTRACT

An index connects readers with information. Creating an index for a single book is a time-honored craft. Creating an index for a massive library of HTML topics is a modern craft that has largely been discarded in favor of robust search engines. The authors show how they optimized a single-sourced index for collections of HTML topics, printed books, and PDF books. With examples from a recent index of 24,000 entries for 7,000 distinct HTML topics also published as 40 different PDF books, the authors discuss the connections between modern technology and traditional information retrieval methods that made the index possible, usable, and efficient to create and maintain.

Categories and Subject Descriptors

I.7.2 [**Document and Text Processing**]: Index Generation, Document Preparation--*Index generation*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing--*Indexing methods*; H3.7: Digital Libraries: User issues; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/ Hypermedia--*Navigation*

General Terms: Documentation, Design, Human factors

Keywords: Indexing, Search, Information retrieval methods, Online information, Navigation

Trademarks and Copyyright

1. THE PROBLEM

A project with a large library of existing documentation of about 40 books (several with multiple volumes) required a high-quality master index. The material was written using a proprietary SGML authoring tool and converted to both HTML for browser-based information and PDF. The library was being converted from a set of books into a task-oriented, topic-based set of documentation, so very little indexing time was available for the approximately 20 authors in two sites fully engaged in rewriting the documentation to conform to new guidelines for online topics, to incorporate new content, and to change content for the next release of their product. The four project editors responsible for the complete library were likewise busy assisting the structural conversion of the library. Customers were asking for more direct ways to find information. At a user conference, 52 percent of users said the former book indexes were their primary entry point into product information.

Given these imposing (but sadly typical) resource constraints, the information architect for the project and an editor with extensive indexing experience worked together to develop an approach that would use technology to maximize the available efforts of both authors and editors. This paper describes the approach from the perspectives of the authors, editors, and, most importantly, the users of this set of documentation. The approach is described in enough detail to enable other projects to adapt the approach to the constraints of their situation. The paper also indicates the future directions that the project will explore.

The challenges to producing a high-quality master index were not just posed by available resources or by available technology, but also by the writing culture of the project. The project had historically been heavily oriented towards writing and producing books—the HTML documentation set had been simply a collection of books converted to HTML. As a result, navigation through the HTML documentation was difficult; there were almost no links between books, and each book was organized under its own table of contents. Full-text search of the HTML books had been the only method of finding information across the complete library, if a user didn't know which book to consult directly.

The product market share was expanding, and new users had to learn the product quickly. However, cultural attitudes towards writing reinforced the problem of books as separate silos of information: authors were responsible for, and took justifiable pride in, producing their individual books, and while consistency in style and ease of access across the library was encouraged, it was much less important to the writers' satisfaction and explicit goals than completing their self-contained sets of information well.

Earlier, the product had offered a PostScript master index as a printed book and a file, in response to customer feedback about finding information across the growing library. There was never time to improve it, so it was eventually dropped, but at the same time, the need for better retrievability of information was increasing and search did not adequately meet that need. Users demanded both an interactive information finder and an easily scanned printable index.

The previous version of the library produced a master index in both PDF and PostScript formats to assist users on platforms on which search did not work. However, the PDF index was generated from the individual indexes of each PDF book after the content of the books had been frozen for translation, so there was no attempt or opportunity to enforce consistency in indexing style across books, even when editors were able to help writers improve individual book indexes. Any effort in directly editing the master index would have been lost on the library as a whole because the index entries were part of the source books. Even so, the process of creating the master index took days and required the cooperation of dozens of authors.

The PDF master index was only a limited replacement for search, since neither the PDF nor the PostScript version could provide direct links to the indexed information. The PDF master index forced users to find and open the corresponding PDF or printed book, then find the referenced page themselves.

As part of the effort to address the problems of navigating through the HTML documentation for the next release of the product, the information architect and the editors decided to produce a high-quality master index in HTML, even though the full-text search capability would be supported across all platforms for the next release.

2. WHY SEARCH IS NOT ENOUGH

A frequent question early in the project was "Why bother with an online index if you have a good search engine?" The problem with search is that typical search solutions are limited to the terms or keywords that are found within the content matter, making search results a happy coincidence between the user's terminology and the content's terminology.

If synonyms or preferred terms are addressed at all in typical search solutions, they are implemented as meta keywords. This turns an explicit advantage of the index (the capability of training the user in a product's vocabulary using *see* or *see also* references) into an implicit method to improve search results. The terms used in an index reflect the indexers' knowledge of the subject matter and the users of that set of documentation. While search solutions typically provide the user with the title, a ranking, and occasionally keywords in context, an index's primary, secondary, and tertiary entries give users more data and context with which to select the right piece of information to meet their needs.

3. PROBLEMS WITH THE PREVIOUS PDF MASTER INDEX

A previous version of the product shipped a PDF master index that was generated simply by collecting and sorting the author-created, book-specific indexes from each of the 40 books. An analysis of the resulting master index revealed that the indexing approach varied greatly across writers. There were many cases of inconsistent terminology. Singular versus plural nouns, preferences for verbs or nouns, and capitalization differences were the easiest problems to spot. Some authors indexed every occurrence of a given term, other authors indexed only the most significant usage of a given term, and others indexed almost nothing at all. Some authors indexed every possible synonym for a given term, and others indexed only the term itself. Some authors relied heavily on secondary and tertiary entries to lend structure to their index, while others relied almost exclusively on primary index entries.

Many writers clearly didn't understand the significance of such decisions on the customers who would ultimately use the indexes. The master index was over 350 pages, and seemingly small differences in primary entries sometimes resulted in users having to look through several pages to find a specific entry.

A complicating cultural problem was that, for many authors, indexing was an activity to be performed only when the content had been written and checked for technical accuracy. In many cases, this meant that the index for an individual book received very little attention, and, in most cases, the master index received no attention at all.

4. BARRIERS

It was clear that there were indexing problems throughout the library. However, without painstakingly analyzing each index in conjunction with the master index, it was impossible to determine what significant material might not have been indexed or which books had inconsistent capitalization and use of plurals. In the wake of the first master index in PDF, the editors initiated an attempt to address some of the most glaring inconsistencies by setting indexing guidelines, educating authors in the art of indexing, and encouraging authors to collaborate with other authors in the project to define standard indexing terminology for new or existing problem areas (for example, using specific labels for program entities so they made sense in the context of other entities from across the library, such as "SQL data type" versus "data type").

The effort met with some success, but the process of standardizing terminology across the library was unwieldy: index entries were maintained within the source files for each book, and the average book had between 50 and 200 source files. So, making a single terminology change that affected 10 percent of the source files in half of the books in the library would require opening, editing, and saving approximately 200 source files that are kept within a version control system. At five minutes per source file, that's almost 17 hours of work! In addition, the source files were often accessible only to the author, so there was a potential bottleneck and a conflict between demands on the author's time to write new content or revise existing index entries.

5. INDEX ENTRIES AS METADATA

Recognizing these problems along with others related to the shift to a topic-based architecture, we proposed a solution that required both technical and cultural change. The new approach to indexing was a shift from viewing index entries as content owned by each author and created solely during the writing process to treating index entries as metadata to be created and edited during a separate process.

The new topic-writing process involves adding all topics in the library to a relational database and storing index entries in a table related to the table that stores the topics. The database is affectionately known as Dobalina. The very nature of an index is a database, because they consist of records that are compiled into a readable, searchable format (Wright, 2001), or several for single-sourced information. The database maximized our ability to generate flexible outputs for different formats.

The initial indexing pass took advantage of the index entries in the legacy documentation: a Perl script stripped the index entries from the source files, replacing them with an SGML text entity unique to each file, and inserting the index entries into the database. Once the index entries were in the database, both authors and editors could then use a Web interface to the database to maintain the index entries. To build a PDF version of their books, the authors download a set of auxiliary SGML files from the database that define the SGML text entities as their corresponding index entries. The following example demonstrates how the index entries are dynamically generated from the database and incorporated into the SGML source.

```
<!-- Start of topic source file -->
&index1; <!-- Index text entity -->
<p>Some topic content.</p>
<!-- End of topic source file -->

<!-- Index entity definitions generated by
database -->
<!ENTITY index1 “<index>
<primary>SQL statements</primary>
<secondary>data definition</secondary>
<primary>data definition</primary>
<secondary>SQL statements</secondary>
</index>”>
```

5.1 Better living through automation

Storing the index entries in a database gives the team the ability to quickly generate the HTML master index, freeing authors from the requirement to painstakingly transform each of their books to create the individual indexes that composed the PDF master index. The process of creating the master index now takes approximately 15 minutes instead of days. Dynamic access to the entire set of index entries enables the team to easily isolate certain consistency problems, such as plural nouns or incorrect capitalization; to immediately identify topics without index entries; to help ensure library-wide consistency for new index terms by providing drop-down selection of terms; and to effect social change by eliminating the need to access source files to add, edit, or delete index entries.

Spell-checking the 24,000-odd master index entries takes about two hours. The changes to the index terms are made in the database, so the corrections are automatically reflected in the book indexes. The previous approach would have required communicating each correction to the writer, and the master index would still have been subject to the errors if a writer did not have time to integrate the changes throughout the affected source files.

5.2 Cultural and process changes

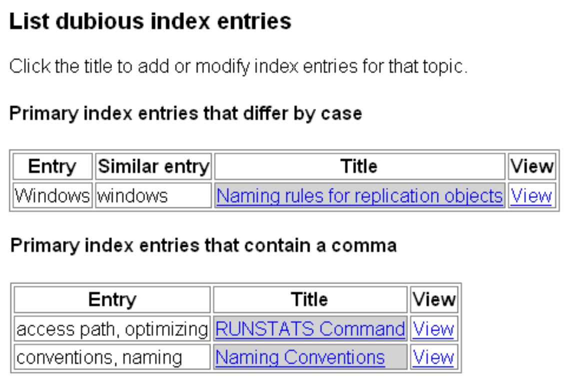
The role of editors changed significantly; rather than providing guidance to authors, the editors collectively edit the master index by directly manipulating index entries for the entire library through a Web interface. More extensive or complicated changes to index entries are accomplished through Perl scripts that manipulate the database records. Maintaining index entries in a database enables new methods of collaboration; for example, a writing team might designate their most skilled indexer to index the new content for a release.

6. THE AUTHOR EXPERIENCE

Indexing training provided detailed guidelines for the authors with a hands-on experience in group indexing. The detailed guidelines are included in the Appendix. They consist of two tables—one listing general indexing guidelines and a content-oriented table that lists specific consistency rules for the library. They also include some guidelines to help writers improve the quality of their PDF indexes.

After the index entries are in the database, authors can dynamically generate Web-based reports about the index entries for their book. One report simply lists the number of index entries per topic; as each topic should have at least one index entry, this is an easy way for writers to ensure that all topics are indexed and to see how extensively. This also helps writers meet the guideline that each PDF book should have no more than two locators for any index entry (primary-secondary-tertiary combination), to ensure that entries are at the best level of detail for users.

Another report, shown in Figure 1, flags specific index entries that might contravene the indexing guidelines. For example, the report checks for index entries that differ only by case and primary index entries that contain a comma, as well as other conditions. A primary index entry that contains a comma typically suggests that the entry should be split into primary and secondary entries so it will nest with other secondary entries.


List dubious index entries

Click the title to add or modify index entries for that topic.

Primary index entries that differ by case

Entry	Similar entry	Title	View
Windows	windows	Naming rules for replication objects	View

Primary index entries that contain a comma

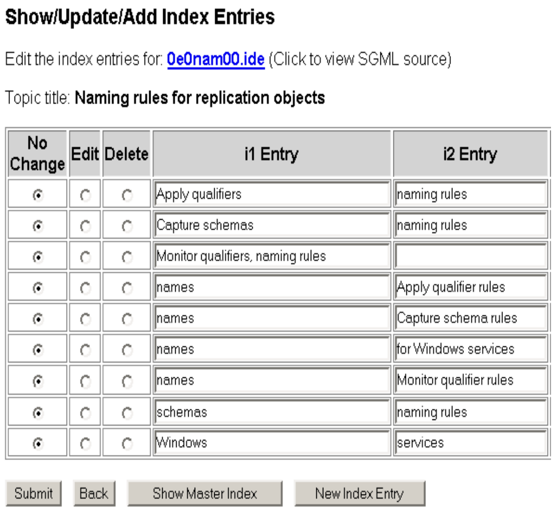
Entry	Title	View
access path, optimizing	RUNSTATS Command	View
conventions, naming	Naming Conventions	View

Figure 1. Report on inconsistent index entries

To ease the authors' job of verifying or addressing deficiencies noted in the reports, the reports provide links to both the content of the source file (**View** column) and the authors' indexing interface (**Title** column). The authors' screens are implemented as a Web-based form with two modes of work: updating existing index entries, and adding index entries.

6.1 Authors' update index screen

The update screen, shown in Figure 2, is a set of text fields that display the current primary, secondary, and tertiary entries (shown as *i1, i2,* and *i3,* respectively).



Show/Update/Add Index Entries

Edit the index entries for: 0e0nam00.ide (Click to view SGML source)

Topic title: **Naming rules for replication objects**

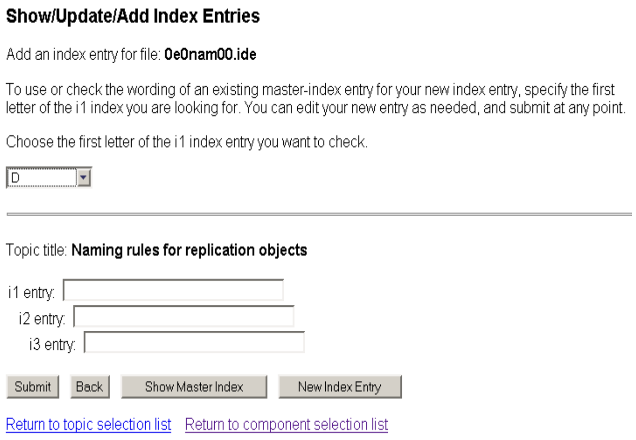
No Change	Edit	Delete	i1 Entry	i2 Entry
●	○	○	Apply qualifiers	naming rules
●	○	○	Capture schemas	naming rules
●	○	○	Monitor qualifiers, naming rules	
●	○	○	names	Apply qualifier rules
●	○	○	names	Capture schema rules
●	○	○	names	for Windows services
●	○	○	names	Monitor qualifier rules
●	○	○	schemas	naming rules
●	○	○	Windows	services

Figure 2. Authors' updating screen

When authors update an entry in a text field, the status of that entry automatically switches from **No Change** to **Edit**. When authors complete their changes, they click **Submit** to commit their changes for any flagged entries to the topic database. In Figure 2, for example, the writer can easily see that the third entry isn't done the same way as the first two, and that the third entry under *names* has an unnecessary *for*. Authors also have the option of deleting index entries by selecting the **Delete** radio button for any entries. After they submit their changes, they see a refreshed table with any deletions at the top followed by a list of changes.

6.2 Authors' add entry screen

The Add Index Entries screen, shown in Figure 3, enables authors to add index entries easily in a way that maintains consistency with the entries already in the database. To add a new index entry for a topic, the author begins by clicking the initial character for the new index entry from a drop-down box.



Show/Update/Add Index Entries

Add an index entry for file: **0e0nam00.ide**

To use or check the wording of an existing master-index entry for your new index entry, specify the first letter of the i1 index you are looking for. You can edit your new entry as needed, and submit at any point.

Choose the first letter of the i1 index entry you want to check.

D

Topic title: **Naming rules for replication objects**

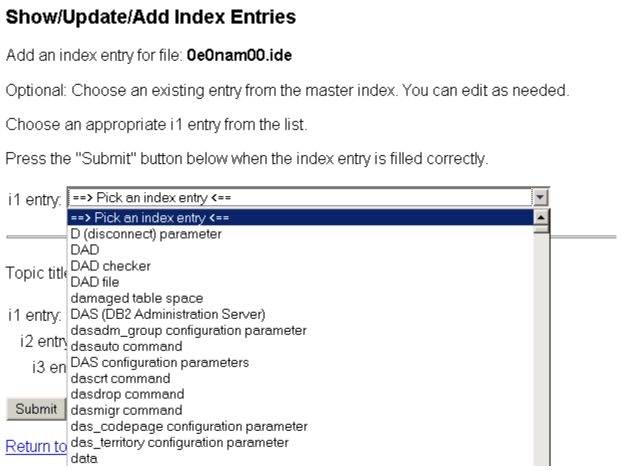
i1 entry:
i2 entry:
i3 entry:

Submit
Back
Show Master Index
New Index Entry
Return to topic selection list
Return to component selection list

Figure 3. Selecting the initial character of a new index entry

The author then selects the primary entry (the *i1 entry*) from the drop-down box that has been dynamically added to the form. This box lists all the primary entries that start with the initial character selected.

When an author picks a primary entry, as in Figure 4, it is automatically copied into the **i1 entry** text box. The author then drills down through the available secondary and tertiary entries, picking the relevant entries. At any time in the process of adding a new index entry, authors can manually enter or change the primary, secondary, or tertiary entries in the text boxes if the existing index entries in the database do not suit their needs. This approach provides a flexible way to limit, but not rigidly control, authors' indexing vocabulary. The next builds of the individual book or deliverable, the HTML master index, and the PDF master index reflect any new, deleted, or updated index entries.



Show/Update/Add Index Entries

Add an index entry for file: **0e0nam00.ide**

Optional: Choose an existing entry from the master index. You can edit as needed.

Choose an appropriate i1 entry from the list.

Press the "Submit" button below when the index entry is filled correctly.

i1 entry: ==> Pick an index entry <==

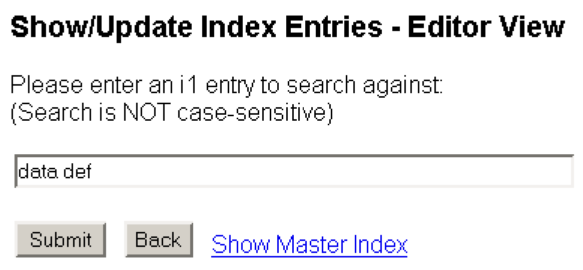
==> Pick an index entry <==
D (disconnect) parameter
DAD
DAD checker
DAD file
damaged table space
DAS (DB2 Administration Server)
dasadm_group configuration parameter
dasauto command
DAS configuration parameters
dascrt command
dasdrop command
dasmigr command
das_codepage configuration parameter
das_territory configuration parameter
data

Figure 4. Selecting the primary entry from the existing list

7. THE EDITOR EXPERIENCE

While the author interface focuses on a book-by-book (or deliverable-by-deliverable) view of the index, the role of the editors in the indexing effort is to improve the quality of the index across the entire library.

Their primary focus is on the master index in HTML. During the editing phase, a new master index in HTML was generated every two hours to enable the editors to quickly view the results of their work. The editors' indexing interface reflects their needs and approach to the task of editing the master index. Editors begin by finding a problem index entry in the master index using a separate browser window. Then, they use a string or substring of the primary index entry, as shown in Figure 5, to bring up a listing of all primary index entries beginning with that string. The search is not case-sensitive, so any variations in capitalization are included in the search results.

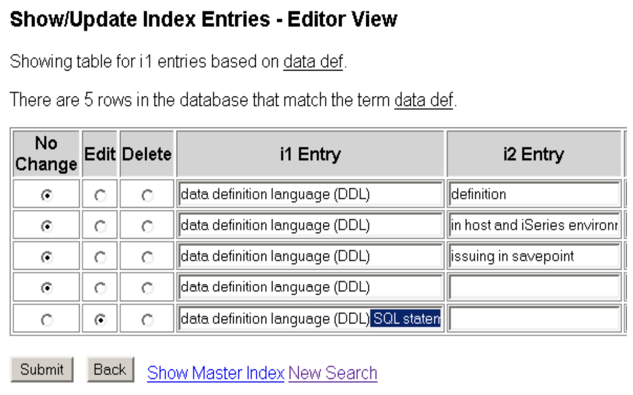

Show/Update Index Entries - Editor View

Please enter an i1 entry to search against:
(Search is NOT case-sensitive)

data def

Figure 5. Editors' screen for finding a primary index entry

The primary entry search yields matching results from across the entire library, so the editor can quickly address inconsistencies between two primary entries and rationalize secondary and tertiary entries. The list of search results appears in a screen similar to the authors' Show/Update/Add Index Entries screen, as shown in Figure 6.


Show/Update Index Entries - Editor View

Showing table for i1 entries based on data def.

There are 5 rows in the database that match the term data def.

No Change	Edit	Delete	i1 Entry	i2 Entry
●	○	○	data definition language (DDL)	definition
●	○	○	data definition language (DDL)	in host and iSeries environr
●	○	○	data definition language (DDL)	issuing in savepoint
●	○	○	data definition language (DDL)	
○	●	○	data definition language (DDL) SQL staten	

Figure 6. Editors' indexing screen for updating entries

When an editor submits a change, the editors' indexing interface refreshes with the results, making it easy to confirm that the expected change was made.

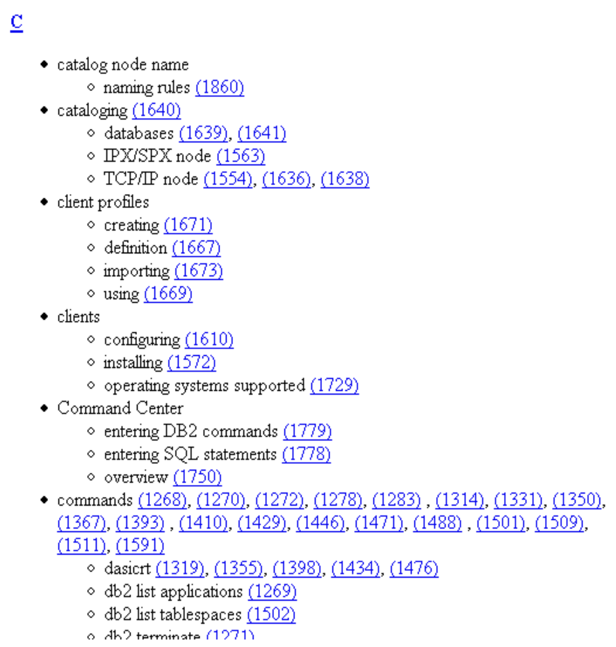
The next builds of the affected individual books or deliverables, the HTML master index, and the PDF master index reflect any new, deleted, or changed index entries.

8. USER EXPERIENCE: HTML

Several alternative presentation formats were considered for the HTML version of the master index. These included the format used by the individual book indexes in the previous version of the documentation, third-party formats, and an internally developed format.

8.1 Previous presentation format

Previous versions of the HTML indexes generated by the project's standard tools were created for individual books. These indexes were presented as nested, unordered lists of index terms. Links to the HTML pages were displayed as arbitrary four-digit integers, as shown in Figure 7. If an index entry pointed to more than one location in the HTML book, the links were displayed as arbitrary integers separated by commas.


C

- catalog node name
  - naming rules (1860)
- cataloging (1640)
  - databases (1639), (1641)
  - IPX/SPX node (1563)
  - TCP/IP node (1554), (1636), (1638)
- client profiles
  - creating (1671)
  - definition (1667)
  - importing (1673)
  - using (1669)
- clients
  - configuring (1610)
  - installing (1572)
  - operating systems supported (1729)
- Command Center
  - entering DB2 commands (1779)
  - entering SQL statements (1778)
  - overview (1750)
- commands (1268), (1270), (1272), (1278), (1283) , (1314), (1331), (1350), (1367), (1393) , (1410), (1429), (1446), (1471), (1488) , (1501), (1509), (1511), (1591)
  - dasicrt (1319), (1355), (1398), (1434), (1476)
  - db2 list applications (1269)
  - db2 list tablespaces (1502)
  - db2 terminate (1271)

Figure 7. Previous index presentation format

One of the disadvantages of this presentation format is that users have no criteria by which to choose among the links that might be presented for a single index entry. The arbitrary integers might even give users the impression of being some sort of relevance ranking. Another disadvantage is that, for a master index composed of approximately 40 books, the large number of primary, secondary, and tertiary index entries makes the index very difficult to scan in an online medium.

8.2 Third-party formats

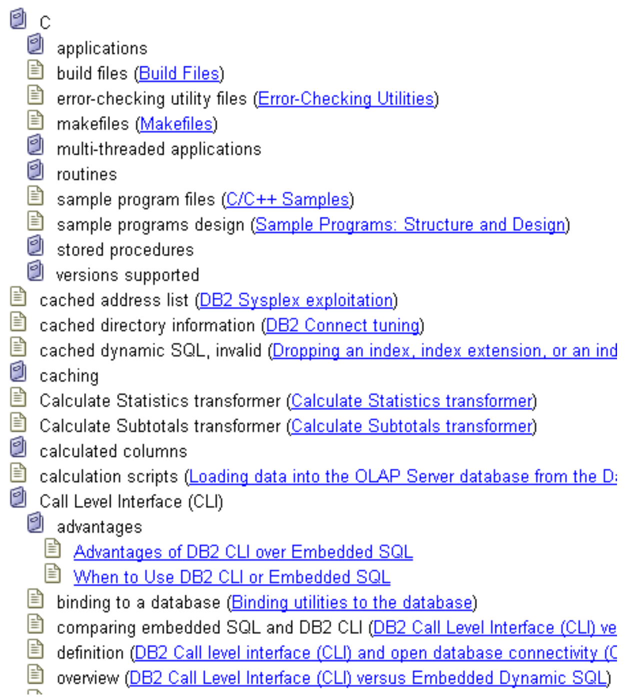
Some existing systems that support indexes, such as Sun JavaHelp™, limit index entries to a one-to-one mapping between each unique index entry and a corresponding locator. This was not an acceptable limitation for the project's large documentation set; each unique index entry needed to be able to map to multiple locators. Other systems, such as Microsoft® HTML Help, do support multiple locators for a single index entry through the use of a pop-up window but do not support the operating systems supported by the project.

Oracle Help for Java and Oracle Help for the Web (http://otn.oracle.com/tech/java/help/content.html) can display multiple topics per index entry using the topic titles, but they are currently limited to two levels of index entries. Oracle Help for the Web enables the user to type the first few characters of a primary entry to jump ahead to that section of the index, but requires the user to click on the index entry to display any links to topics.

We decided to display the index in the content pane of a help system partly to enable portability. If we later decide to deliver our information in a different help system – such as JavaHelp, the Eclipse help system, or Windows® Help – we can avoid any limitations in the help system's built-in support for indexes. Trying to drop 24,000 index entries into a navigation pane simply will not deliver acceptable performance. Assuming that we continue to deliver HTML-based documentation in future versions of the product, this approach will provide users of future versions of the product with a consistent means of accessing the master index.

8.3 Internally developed format

The solution for this project was to provide an expanding, collapsing master index, as shown in Figure 8. To assist in scanning, the HTML master index presents only the primary entries at first, but enables users to drill down to secondary and tertiary entries to find exactly the information they need. An index entry that maps to multiple topics displays the locators as a further nested list of links. To provide users with criteria by which they can judge which of several locators meets their needs, each locator is shown as the title of the corresponding topic, as illustrated below.



- C
  - applications
  - build files (Build Files)
  - error-checking utility files (Error-Checking Utilities)
  - makefiles (Makefiles)
  - multi-threaded applications
  - routines
  - sample program files (C/C++ Samples)
  - sample programs design (Sample Programs: Structure and Design)
  - stored procedures
  - versions supported
- cached address list (DB2 Sysplex exploitation)
- cached directory information (DB2 Connect tuning)
- cached dynamic SQL, invalid (Dropping an index, index extension, or an ind
- caching
- Calculate Statistics transformer (Calculate Statistics transformer)
- Calculate Subtotals transformer (Calculate Subtotals transformer)
- calculated columns
- calculation scripts (Loading data into the OLAP Server database from the D
- Call Level Interface (CLI)
  - advantages
    - Advantages of DB2 CLI over Embedded SQL
    - When to Use DB2 CLI or Embedded SQL
  - binding to a database (Binding utilities to the database)
  - comparing embedded SQL and DB2 CLI (DB2 Call Level Interface (CLI) ve
  - definition (DB2 Call level interface (CLI) and open database connectivity (C
  - overview (DB2 Call Level Interface (CLI) versus Embedded Dynamic SQL)

Figure 8. Current index presentation format

Nielsen's 1997 article "The Need for Speed" suggests that 34K is the maximum size of an optimal Web page for a ten-second response time for a dial-up connection to the Web. However, internal surveys of our users indicated that most users of our product access the documentation either from a local workstation or from a web server within their company's intranet. We wanted to give users the additional context provided by full topic titles, so we divided the content into multiple files. Given our typical user scenarios, we decided that we could allow somewhat larger files than recommended by Nielsen and divided the index by initial letter into 27 separate files, including one file for all index entries beginning with non-alphabetic characters.

The resulting average size of the set of index files is 100K, the median size of the set of index files is 60K, and the largest index file is 400K. The average index file loads and displays in less than one second from an intranet or local workstation, while the largest index file takes just over three seconds to load and display. These times are within Nielsen's maximum threshold for web usability. While the longest delays fall outside the threshold of optimal usability response times of less than one second, we feel that the slightly increased initial load time is balanced by the ease of scanning an index file that contains all of the entries for an initial character.

When we collected the complete set of 24,000 index entries in a single HTML file, the file size was over two megabytes and some browsers were incapable of displaying the file. The current index presentation format uses only two images, each smaller than one kilobyte, which are downloaded by the browser only once. Approximately five percent of the total size of the HTML index represents the JavaScript and CSS code that makes the index accessible by keyboard navigation. The majority of the content is due to the topic titles, which average 29 characters per title. Index entries average 13 characters per entry.

Initial usability sessions on the current index presentation format indicate that users understand how to work with the expanding and collapsing lists, prefer this format to our previous index format, and find the performance of the index from an intranet or from a locally workstation acceptable. However, as these sessions drew on the experiences of only four users, we recognize the need to do further research to confirm the validity of this presentation format.

9. USER EXPERIENCE: PDF

In PDF, the individual book indexes look like regular book indexes; that is, they display the index entries and one or more page numbers for those index entries. The decision to associate index entries with entire topics has the detrimental effect of forcing all index entries for a given topic to point to the beginning of that topic. For topics that are longer than a page, users might have to browse through subsequent pages to find the information represented by the index entry in which they are interested. However, length should not be an issue for most topics, as authors are encouraged to keep topics short to ensure good online display. The exception is some reference topics: a topic covering the syntax of a single command might be over 20 pages, and a user might legitimately be interested in just one of the optional arguments for that command. In future iterations of the topic database we hope to enable a more granular indexing strategy to address this shortcoming.

We decided to optimize our indexing effort for the master index (both in HTML and PDF), rather than for individual book indexes, so we had to relax some traditional indexing guidelines. One decision we made was to allow individual book indexes to contain primary entries with only a single secondary entry. Rather than concatenating such entries with a comma, we decided, for the sake of the master index, to keep these entries as primary and secondary entries. The following example demonstrates how an individual book index is normally edited to concatenate the primary and secondary entries:

```
SQL statements, overview – 453
```

However, to optimize such entries for the master index, the individual book index must contain distinct primary entries with a lone secondary entry, as in the following example:

```
SQL statements
      overview – 453
```

We felt that this decline in usability of the individual book indexes would be recouped by reducing the number of comma-spliced entries in the master index, as shown in the following example from the PDF version (the identifiers in front of the page numbers are short forms of the book names):

```
SQL statements, issuing – ADGv1: 238
SQL statements, overview – ADMINv1: 453
```

The master index instead presents the optimal arrangement of nested primary and secondary entries:

```
SQL statements
        issuing – ADGv1: 238
        overview – ADMINv1: 453
```

One indexing decision driven by the requirements of PDF that posed a potential compromise to the quality of the HTML master index instead improved the master index. It was necessary to artificially subdivide extremely long index entries into separate primary and secondary entries in PDF because the books display the index in a three-column format. A limitation of the PDF transform technology used by the authoring tool is that extremely long index entries that contain no spaces, such as the API keyword `SQL_DESC_DATETIME_INTERVAL_ PRECISION`, bleed over column boundaries rather than wrap cleanly. In those cases where many similar entries also start with the same prefix, the prefix was turned into a primary entry and the remainder was indexed as a secondary entry. This indexing decision reduces the number of primary entries in the master index, making it much easier for the user to scan the primary entries in the collapsed HTML master index. So the preceding example becomes one of many index entries with the primary entry "SQL_DESC":

```
SQL_DESC_
        BIND_OFFSET_PRT
        DATETIME_INTERVAL_PRECISION
        DISPLAY_SIZE
```

10. TECHNOLOGY

The HTML master index is implemented in HTML Version 4.0 using Cascading Style Sheets level 2 (CSS2 at http://www.w3.org/Style/CSS/), Document Object Model (DOM at http://www.w3.org/DOM/), and ECMAScript 262 (http://www.ecma.ch/ecma1/stand/ecma-262.htm) to enable the expanding and collapsing behavior. Microsoft Internet Explorer 5.0 and higher and Netscape 6 and higher support the expanding and collapsing behavior. Other browsers degrade gracefully to display nested unordered lists of the index terms and associated topics.

The topic database, known as "Dobalina," is implemented with a blend of open-source and proprietary products. The relational database is IBM® DB2® Version 7.2 (http://www.software.ibm.com/data/db2/udb), running on a reasonably powerful server, but it could fairly easily be replaced by an open-source database such as MySQL or PostgreSQL. The source format for documentation is SGML (produced with a tool built on ArborText using a proprietary IBMIDDOC DTD), which enables us to dynamically generate index entries as text entities. XML or some sort of manipulated HTML would also fit easily into this model.

The project uses the Apache Web server (http://httpd.apache.org) to display the Web scripting front end implemented in PHP (PHP: Hypertext Preprocessor, see http://php.apache.org). PHP also creates the auxiliary source files used to generate the PDF books.

The HTML master index is generated by a Perl script (http://www.perl.org), which connects to the database through the Perl database interface module (http://dbi.perl.org). Most of the initial processing of the documentation source files was also performed with Perl scripts.

11. FUTURE DIRECTIONS

The index improvement process for such a large information set is planned over several phases, as shown in Table 1. In this project, we are now planning Phase 3.

Table 1. Phases of the indexing effort

Phase 1: Recognize the problem and build internal support	Reassess master index quality problems Analyze information retrieval needs Experiment with scripts Build indexing requirements Develop indexing guidelines Collect user feedback
Phase 2: Create first version for writers and customers	Create the topic database Develop indexing interfaces Design the presentation of the master index and do initial user testing Establish flexible controlled vocabulary process and start adding *see* references Train authors in indexing and guidelines Edit master index (focus on primary entries) Conduct initial user testing of index presentation format
Phase 3: Continue refining the master index	Provide multiple entry points to the index in the product Implement *See also* references Review consistency of primary entries Edit master index (focus on subentries) Add more syntax and reference entries Implement and apply further reporting features Promote internal use and gather feedback on index content to refine user orientation of entries and to "fill holes" Do user testing Reuse user-centered design scenarios
Phase 4: Ensure completeness of contributing content areas	Continue to develop reporting features to improve overall consistency Ensure completeness of concept and task entries Edit PDF indexes to help ensure that content is adequately indexed Work with each small writing teams to improve index coverage Analyze problems with PDF and master indexes in other languages
Phase 5: Maintain and continue improving index	Establish iterative maintenance process Continue to improve presentation, technology, and content Build consensus to improve quality of localized indexes

This table shows our actual process and not necessarily the best possible task flow. For example, ideally, *See* and *See also* references would be fully implemented in phase 2.

One area for future development is the creation of additional reports and scripts. For example, a regular report could identify all the new entries that were not part of the last published index for special editing attention.

We are currently augmenting our lists of *see* references for synonyms, competitive terms, and obsolete terms. We plan to take full advantage of the capabilities of our relational database to create mappings between deprecated or less acceptable terms and acceptable terms, so that any PDF book with an index entry for a deprecated term automatically includes cross-references that lead to the acceptable term.

A new indexing screen for authors, with some of the function of the editors' indexing screen, will facilitate improvements of the PDF indexes.

We've put a lot of effort into making what's already in the index more consistent, usable, and predictable. But one of the biggest problems is to fill the remaining content holes, that is, what's not yet indexed. Customer, developer, and service feedback indicates the need to improve the granularity of indexing reference topics that document syntax.

We will also ensure that the index provides easy access to information needed to implement business and customer scenarios developed by the user-centered design and development teams, and continue to develop the usability of the index interfaces in the product. In Phase 4, we plan to work with each small writing team on specific ways to improve the retrievability of their information.

12. REFERENCES

[1] Nielsen, Jakob, "The Need for Speed," 1997; available at http://www.useit.com/alertbox/9703a.html

[2] Wright, Jan C., "Single-Source Indexing," *Proceedings of the 19th Annual International Conference on Systems Documentation*, October 2001; available at http://www.portal.acm.org