

## IXE: Ideare indeXing Engine

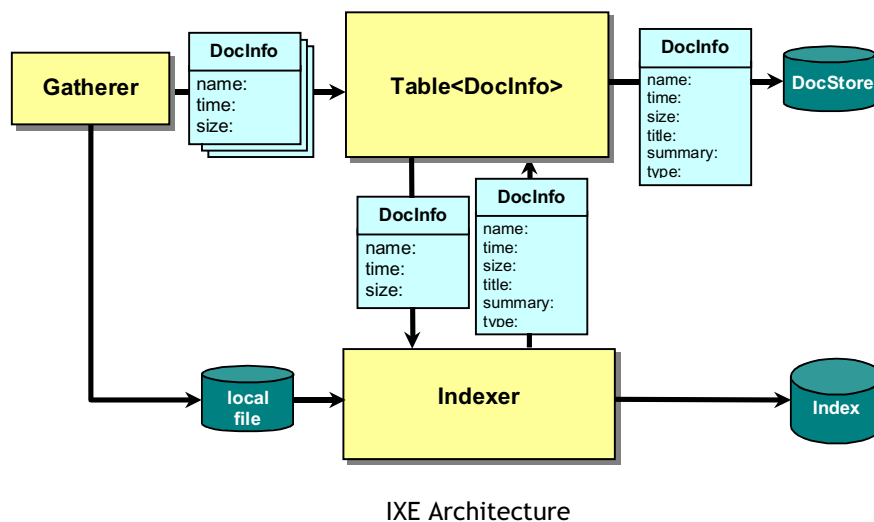
The advent of the written word brought with it the need to organize information so that it can easily be retrieved and processed. The most common means of organizing are *indexes* and *catalogues*. Indexes, for example analytical indexes of a book, a dictionary or a telephone directory, enable you to get hold of information quickly, if you know what you are looking for. On the other hand, if you don't know what you are looking for, thematic catalogues, which categorize information by subject, are useful. Examples of a catalogue are the yellow pages or library cards (a

prototype document file system). Now that information is filed electronically for the most part, the main challenge is to define algorithms for efficient and automated document indexing, and to build querying tools that are fast and accurate. This is the science of *Information Retrieval*.

Information Retrieval (IR) is centered on techniques for displaying, storing and accessing information. In IR the data searched for is known as *documents*, and documents are a collection of *terms*. A document consists of one or more pieces of text, while a term is a word or

phrase that helps describe the document and typically occurs in the document one or more times. For example, a document concerning vacations might contain terms such as 'trip', 'hotel', 'museum', 'beach', 'bikini' and so on.

Generally speaking, we can say that a document might be anything we want to look for and a term any of the characteristics that help define it. So a document could be a collection of fossils in a museum and the related terms the morphological characteristics of the fossils. Similarly documents could be songs and the terms the harmonies that form the music.



## What is IXE?

IXE is an indexing and search tool based on the most advanced technology currently available in state of the art Information Retrieval. IXE is a library of C++ classes for developing indexing and retrieval applications for huge volumes of documents:

- ❖ Web search engines
- ❖ document management systems
- ❖ indexing and searching company Intranets.

IXE is light and compact, making it suitable for indexing small amounts of information (such as document management systems for small companies, law firms or notaries) and for searching through a collection of multimedia documents. IXE is designed for maximum efficiency and scalability and can manage multiple collections of documents totaling several Terabytes in size.

### Highlights

- ❖ Creation of search services for the Web or Intranets
- ❖ High performance:
  - ❖ searches: ~50 msec on a 50 Mbyte index
  - ❖ indexing: 2GB an hour
- ❖ cutting edge technology
- ❖ flexibility provided by metaprogramming
- ❖ database architecture for C++ objects
- ❖ broad-based, scalable architecture (> 10 Terabytes of documents)
- ❖ highly competitive pricing
- ❖ Web Service version
- ❖ platforms: Linux, Unix, Windows

Indexing with IXE is very fast also thanks to an efficient merge function which supports scalable and distributed architectures. Individual spidering systems are placed alongside the different sources to be indexed (company department or Internet domain). Each spider creates a partial index which is then sent to a central location. Here the indexes are consolidated into one general index.

Parallel indexing saves time and cuts costs, giving these tangible benefits:

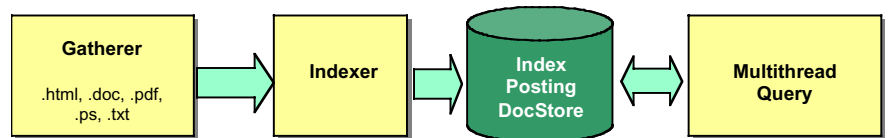
- ❖ savings in transmission bandwidth: transmitting indexes, which are highly compressed, requires far less band than transferring the original material;
- ❖ speed of indexing: new portions of indexes can be generated much more frequently than with traditional systems, ensuring a more up-to-date index and consequently more accurate replies;
- ❖ reduced hardware costs: IXE performance is excellent even on Linux servers, which have a considerably lower price-tag than the servers typically used to run this type of application.

The numerous technical innovations built into IXE have a significant impact not just on the performance, flexibility and ease-of-use of applications based on it, but equally importantly on the budget of an organization implementing an Information Retrieval system with IXE. Its performance is unbeatable on small, inexpensive Linux or Windows systems, as well as supporting higher end servers such as Solaris, HP-UX, Tru64 UNIX etc.

## Architecture

An IXE application consists of the following elements:

- ❖ **Gatherer:** gathers the documents to be indexed
- ❖ **Indexer:** creates an index for the documents collected
- ❖ **Document store:** filing of the documents
- ❖ **Query server:** runs queries against the indexes.



IXE comes with its own gatherer for collecting documents from the local file system or the Web, but users can easily interface this with other spidering systems, if they wish.

IXE library architecture (Figure 1) centers around a flexible object database.

## Document Store

The Document Store contains descriptions of the documents based on such attributes as *name*, *title*, *summary*, *date* and *size*, and if required the entire document can be stored. The IXE database can contain several terabytes of information.

Documents are described by *DocInfo* class objects (or one of their derivatives). The basic *DocInfo* class contains the essential attributes to identify documents in the archive: document ID, size and date modified. These elements ascertain whether two documents are in fact identical or one is a more recent version of the other,

in which case indexing will have to be repeated.

Applications can add and manage further information associated with each document simply by defining classes derived from *DocInfo*,

Document gathering, extraction, analysis, indexing and searching are performed by a C++ class library.

## Indexer

The indexer runs the index functions on the various fields or columns of a document. Each column has its own index which is created or updated each time a new document is added to the table. The indexer receives a *DocInfo* object for each document to be indexed and builds an index for each of the fields specified. For example, if we specify a *Field::fulltext* for a certain field or a *Field::external*, a *full text* index is built. In the first case the original document is indexed and stored (in DocStore), while in the second, IXE stores only the reference to the file that contains it.

The task of extracting terms from a document falls to the appropriate *reader* (from the *DocReader* class). These are selected according to the type of document examined. In addition to extracting terms from a document, the reader can extract other information to store in the related *DocInfo* fields. These in turn can be indexed and used during searches or when viewing results.

IXE can generate incremental indexes; new indexes are automatically added to the index and existing documents are substituted. Indexing functions can be managed either via command line, or via the *ixe.conf* configuration file.

## Document Reader

IXE's modular architecture means that readers developed to process specific document formats can be used. For example, with an *SQLReader* data can be imported from a relational database and searched with the access and ease of use that users have come to expect from Internet search engines.

The reader extracts not only terms but also other information associated with a document, e.g. case (upper or lower case...) and color (which is used to show where a term occurs in a document). In the case of the *HTMLReader* it can determine the tag or class of the element within which a particular word appears.

Each reader fills a special *DocInfo* structure with information taken from the document. This might be the document title or a summary. This information is stored in a database and extracted when queried for.

Some of the readers available:

**GenericReader** reads text documents and can extract text from Microsoft Office documents, discarding the encapsulated PostScript format.

**HTMLReader** analyses HTML pages, extracts titles and summaries and automatically assigns a color to each term based on the HTML tag in which it is found.

**SoifReader** interprets files written in Harvest SOIF format.

**MemReader** reads the text content of a string.

## Search Functions

IXE supports Boolean type search functions for phrases (such as: *body matches "network computing"*), on prefixes, on attributes (*site matches "it.unipi.\*"* and *body matches "network computing"*) and on colors (*body matches title = software AND linux*).

## Web Service

Web Service technology makes it possible to create services and applications that can be used via the Web. A Web Service is a self-describing module of programming that can be published, downloaded or invoked via the Web.

IXE has a Web Service which enables search capabilities to be made available on the Internet. For example a catalogue of products or services (flights, hotels, etc.) can be made available for real-time querying by other companies who in turn offer services to end-users (e.g. travel agencies).

Ideare's Web Service technology is one example of our commitment, often in collaboration with research and university institutions, to constant improvement and to ensuring IXE remains at the forefront of technical developments.

## Applications

IXE is a library of C++ classes supplying an Application Program Interface (API) for the development and integration of customized search applications.

IDEARE uses IXE in its family of SearchTone modules. The following is a list of these modules:

- ❖ **Web Search Engine**
- ❖ **Audio Search Engine**
- ❖ **Video Search Engine**
- ❖ **Image Search Engine**
- ❖ **Usenet Search Engine**
- ❖ **Wap Search Engine**
- ❖ **Automatic Categorization Engine**
- ❖ **Chat Engine**
- ❖ **Shopping Engine**

Clients can choose the ideal solution for their requirements from these modules or they can develop their own custom application.

## Advantages

IXE is a system that can be used to develop competitive Information Retrieval solutions. The costs involved undoubtedly cannot be beaten no matter what criteria are used:

- ❖ Software license costs
- ❖ Application development

- ❖ Hardware needed
- ❖ Transmission bandwidth used
- ❖ Index generation times

Independent performance tests carried out by clients in real-life situations have shown that IXE has the best indexing and query times. IDEARE is happy to carry out further tests with IXE in conjunction with clients' applications.

IXE offers a range of characteristics hard to find in other IR products:

- ❖ Ease of programming: C++ class library with metaprogramming
- ❖ View of document extracts
- ❖ Proximity Rank
- ❖ Link Popularity Ranking
- ❖ Search by color
- ❖ Search over multiple columns
- ❖ Multithread and distributed search servers
- ❖ No limit to document size

## System requirements

IXE runs on Linux, MS Windows, Solaris, Tru64 UNIX. Minimum requirements vary according to the number of documents to be processed.

## Ideare SpA

Lungarno Mediceo, 56  
56126 Pisa

Tel: (+39) 050 575300

Fax: (+39) 050 575583

<http://www.ideare.com>

---

### IXE Competitive Edge

---

<b>Indexing</b>	2GigaBytes/hour	2-3 times faster than competitors
<b>Query</b>	2.500 queries/sec.	10 times faster than competitors
<b>hardware requirements</b>	Inexpensive PC Linux Servers	Less than half the price