



# No Database Limits

Index-based Distributed Architecture for Litigation Search & Review

By Nicholas Croce



### The Growing Challenge of Data Management

In the 20 years since the advent of desktop computing and email, the amount of information generated by corporations and individuals has grown exponentially. Corporations and firms are facing staggering amounts of data: an estimated 100 billion<sup>1</sup> emails are generated daily, and the average desktop can store up to 40 million pages of information and digital rich media (voicemail, video). The continued growth of data has forced corporations to expand efforts to manage information within the enterprise, including data storage, retention and control.

Beyond the sheer volume of data, corporations must also contend with the amended Federal Rules of Civil Procedure (FRCP) governing electronic discovery. The changes to the FRCP reference electronically stored information (ESI) specifically—including emails, instant messages, databases and all other forms of corporate data—as content that should be preserved for possible production. Because of the scope of these changes, not only is the corporate enterprise faced with new preservation requirements, it also must contend with the inevitability of producing all of its data for litigation or investigation in a timely manner.

Even with new technology, production is not getting any easier. Litigation databases are larger than ever and continue to grow. Where previously a large case constituted 50 to 100 GB of data, encompassing tens of thousands of records, cases within the same size range are now considered average, if not small. Because of data growth, discovery and review have become exceedingly challenging, rendering the established methods of linear review unsustainable. Simply, there is too much data and reviewers are reaching capacity even with contract and outsourced review resources.

### Examining the Standard: Relational Databases

The database standard for legal repository management has been the relational database, which has limits in scale and return. For large datasets, relational databases have been serviceable with intensive programming and support. This database structure, however, is beginning to reach capacity and may not be able to deliver the speed and other search tools required to accommodate the expanding size of datasets.

As ESI volume has exploded, attempts to use relational databases have encountered the “relational wall,” where this technology no longer provides the performance and functionality needed. This wall is encountered when systems need to extend information models to support data relationships, new data types, and extensible data types. Similarly, the wall appears when deployed in distributed environments with complex operations. Attempts to scale the wall with relational technology only leads to an explosion of tables, many joins, poor performance, poor scalability, and loss of integrity.<sup>2</sup>

---

<sup>1</sup> Peter Lyman & Hal R. Varian, *How Much Information?* (2003) <http://www.sims.berkeley.edu/how-much-info-2003>.

<sup>2</sup> Object Oriented Database vs. Relational Database by Objectivity Inc. Copyright 2000 – 2007.  
<http://www.objectivity.com/object-oriented-database-vs-relational-database.html>

To further examine this concept, the following table represents the basic structure of a relational database. Each row represents a file (or document), while each column represents a field of data. These systems can grow both vertically and horizontally as respective files or fields are added to a system. When a query is sent to the system, the search is essentially performed down each file and across all of the content.

File ID	File Content
001	Vincent saw John take the document from the server.
002	John saw the document on the server.
003	Vincent saw John.
004	John gave the document to Vincent.
005	Vincent saw the document.
006	The document was from the server.
007	John gave the server the document.
008	John saw Vincent take the document.
009	Take the document from the server.
010	John was on the network server.
011	Vincent was on the server.
012	John saw Vincent on the server.

When performing a basic search for the term "network," a relational database must examine each file and its associated content to find each occurrence. In this example, the search would have examined 12 files and all 69 words in the content to determine that only file 010 is responsive to the query.

### Next Generation Discovery and Scalability: Index Architecture

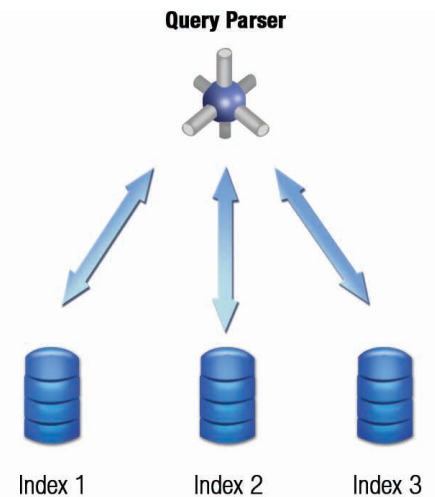
A new approach to discovery and review incorporates an Index architecture where review platforms are provided with a highly scalable and efficient solution. In an Index, information is coded directly into the engine, vastly improving the speed and responsiveness of operations. Index engines store the occurrences of each word and the respective document where those words appear. Demonstrated in the figure below, each individual word that is contained within the content of the files is sorted so that the system can immediately retrieve that word, and instantly returns the respective files to the user. This same operation in an Index requires the system to only look for the word "network" and can immediately return responsive file 010, essentially in one step.

Indexed Words	File ID
Document	001, 002, 004, 005, 006, 007, 008, 009
From	001, 006, 009
Gave	004, 007
John	001, 002, 003, 004, 007, 008, 010, 012
Network	010
On	010, 011, 012
Saw	001, 002, 003, 005, 008, 012
Server	001, 002, 006, 007, 009, 010, 011, 012
Take	001, 008, 009
The	001, 002, 005, 006, 007, 008, 009, 010, 011, 012
Vincent	001, 003, 004, 005, 008, 011, 012
Was	006, 010, 011

## No Database Limits

This illustration demonstrates how query performance is dramatically more efficient in an Index than in a relational database. Inference's architecture not only incorporates the Index engine into its technology, but also takes advantage of its scalability by using a Query Parser to allow for a distributed processing model as shown. When a query is entered into Inference, the Query Parser simultaneously distributes that command to each individual Index engine. The results are quickly returned from the engines and consolidated by the Parser before returning the information to the user. This provides a virtually unlimited scale of the system, without sacrificing performance, since each Index engine always operates at peak efficiency.

Inference was developed as next-generation electronic discovery software based on combining the Autonomy IDOL™ Index engine with the litigation workflow. Inference is the only solution available that leverages an Index engine versus a relational database, thus achieving the speed and scale necessary to manage the volume of discoverable information in litigation and investigation.



### Summary

For today's large litigation databases, a distributed Index architecture that provides unlimited scale and significant speed enhancements for electronic discovery applications and online review is now a requirement. Because of the architecture's ability to combine standard Boolean searching with enhanced analytics, users gain a streamlined method of review.

### About the Author

Nicholas Croce, President of Inference Data, leads the creation and development of Inference, the company's next generation analytics software for electronic discovery. Prior to joining Inference, Mr. Croce served as President of DOAR Litigation Consulting, a leader in electronic discovery and courtroom technologies.

### About Inference Data

Inference Data provides the leading electronic discovery analytics solution, Inference. The most scalable, user-intuitive and cost-effective data analysis and review platform, Inference utilizes Autonomy's state-of-the-art IDOL™ engine and provides collaborative tools that intelligently mine the results of electronic data, enabling the efficient management of the entire lifecycle of litigation review and regulatory investigation. For more information, visit [www.inferencedata.com](http://www.inferencedata.com).