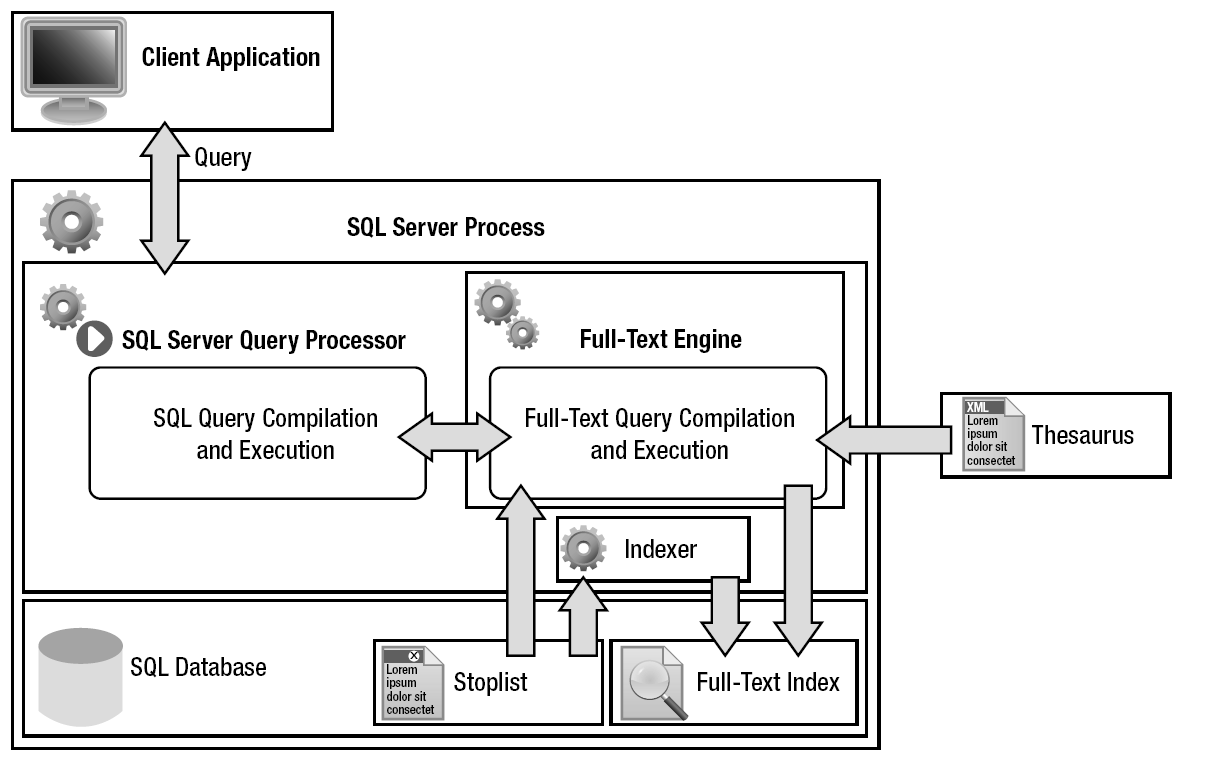
**ECM Library Demo**

1. Purpose of the meeting: what are the outcomes we would like from this meeting or possible actions moving forward?
   1. Market feedback – advice/feedback/input on our product /gut reactions
   2. Potential partnership
2. What we using at this meeting
   1. 1.5 million documents
   2. A few hundred emails
   3. Several thousand attachments
   4. Servers in Phoenix
   5. Wireless connection
3. Purpose
   1. Archive
   2. Backup
   3. Search
   4. Retrieval
   5. eDiscovery
4. Our Advantage
   1. This is the ultimate Search Engine
   2. Costs
   3. Implementation
   4. Administrative effort
   5. Setup time
   6. Maintainability – distribution network
   7. Expansion - Unlimited storage – Clustering - Ultimate Speed
   8. Reliability – Backup – Replication - Mirroring
   9. The team -
   10. Updates and Fixes
   11. Batch archive – immediate archive – immediate search across the enterprise
   12. Concurrent users
   13. Hardware requirements
   14. Business Intelligence – number of searches – types of searches – quality of searches
5. Our disadvantage
   1. New to the field
   2. Unknown quantity
6. Our Search
   1. Simple as Google™
   2. Much more concise
   3. Uses full reduction search capabilities
   4. Unbelievably Fast due to inverted tree search
   5. Ability to search for details that were formally unattainable
   6. Legal adherence
   7. Repeatable
      1. HR harassment
      2. Company security
      3. Competition
7. Libraries
8. Groups
9. Archive
   1. Selectable
   2. Sustainable
   3. Adjustable
   4. Expandable
      1. Other Repositories
      2. File servers
      3. Laptops
      4. Desktops
      5. Structured data stores (databases)
      6. Mainframe data
      7. SAP Data (archive and search)
10. Search
    1. Full word
    2. Partial word
    3. Wild cards
    4. Taxonomy
       1. Classonomy
       2. Inclusionary
       3. Exclusionary
       4. Substitution
    5. Inflection
    6. Thesaurus – multiple simultaneously
    7. Full phrases
    8. Metadata
    9. OCR
    10. Data quality of unstructured data
    11. Time domain
    12. Business meaning and weighting
    13. Exact search and weights
    14. TheProcess

# Some Confidential Stuff

DO NOT LEAVE THIS BEHIND – It is for informational purposes only

So how do we pull this together and how do we get the speed and precision we do. It took many years of development and research and some real computer savants – the geeks that geeks are ashamed to be seen with. Our small team started working on Athenaeum in 1998 and in 2008 and 2009 combined BM25, Chaos math and fractals to develop an archival and search process that is difficult to conceive and harder to implement. But once implemented, it gives content management and searching an undisputed “super advantage”. Not only do we find what you are looking for, we tell you with a great deal of accuracy, just how relevant it is.

## Statistics for Ranking

When an index is built, statistics are collected for use in ranking. To minimize the size of the index and computational complexity, the statistics are often rounded.

When a catalog is being built, the algorithm creates small indexes as data is indexed, then merges the indexes into a large index. This process is repeated many times. A final merging of indexes into one large master index is called a "master merge". Some statistics are taken from individual indexes that contain query results, and some from the master index; others are computed only when a master merge takes place. As a result, the ranking statistics can vary greatly in accuracy and timeliness. This also explains why the same query can return different rank results over time as indexes are merged. Further, as full-text indexed data is added, modified, and deleted, those changes also will impact statistics and rank computation.

The list below includes some commonly used statistics that are important in ***calculating rank***:

* **Property:** An attribute of a document. This corresponds to a column in SQL Server.
* **Document:** The entity that is returned in queries. In SQL Server this corresponds to a row. A document can have multiple properties, just as a row can have multiple full-text indexed columns.
* **Index:** A single inverted index of one or more documents. This may be entirely in memory or on disk. Many query statistics are relative to the individual index where the match occurred.
* **Catalog:** A collection of indexes treated as one entity for queries. Catalogs are the unit of organization visible to the SQL Server administrator.
* **Word:** The unit of matching in the full-text engine. Streams of text from documents are tokenized into words by language-specific word breakers.
* **Occurrence:** The word offset in a document property as determined by the word breaker. The first word is at occurrence 1, the next at 2, and so on. In order to avoid false positives in phrase and proximity queries, end-of-sentence skips 8 occurrences. End-of-paragraph skips 128 occurrences.
* **Key:** Combination of a property and a word.
* **HitCount:** The number of times the key occurs in the result.
* **Log2:** The highest-order bit set in a 4-byte value. Log base 2 was chosen over any other type of logarithmic computation for performance reasons. It is much faster than computing log base 10 or log base *e*.

unsigned Log2 ( unsigned long s )

{ for ( unsigned iLog2 = 0; s != 0; iLog2++ )

s >>= 1;

return iLog2;

}

* **IndexDocumentCount:** Total number of documents in the index.
* **KeyDocumentCount:** Number of documents in the index containing the key.
* **MaxOccurrence:** The largest occurrence stored in an index for a given property in a document.
* **MaxQueryRank:** The maximum rank returned by the engine (1000).

### Ranking of CONTAINSTABLE

Statistical Weight = Log2 (( 2 + IndexDocumentCount ) / KeyDocumentCount )

Rank = min( MaxQueryRank, HitCount \* 16 \* StatisticalWeight / MaxOccurrence )

Phrase matches are ranked just like individual keys except that KeyDocumentCount (the number of documents containing the phrase) is assumed to be 1. This can be wrong in many cases, and leads to phrases having relatively higher weights than individual keys.

### Ranking of ISABOUT

ISABOUT is a vector-space query in traditional information retrieval terminology. The default ranking algorithm used is ***Jaccard***, a widely known formula. The ranking is computed for each term in the query and then combined as described below.

**ContainsRank** = same formula used for CONTAINSTABLE ranking of a single term (above).

**Weight** = the weight specified in the query for each term.  MaxQueryRank is the default weight.

**WeightedSum** = Σ[key=1 to n] ContainsRankKey \* WeightKey

**Rank** =  ( MaxQueryRank \* WeightedSum ) / ( ( Σ[key=1 to n] ContainsRankKey2) + (Σ[key=1 to n] WeightKey2) - (WeightedSum))

The sums are computed using unsigned 4-byte integers. For this reason, no more than 4294 keys can be in any given vector query because the integer may overflow (this condition is checked and such queries are failed). The other math is done in 8-byte integers.

# The Magic

### Ranking of FREETEXT

Freetext ranking is based on the ***OKAPI BM25*** ranking formula. This is part of “informational theory” and allows a group of retrieved words, phrases or terms to be systematically ranked based on a set of complex formuli. The below was our basis, but the full set of equations that forms the foundation for the FREETEXT or inflectional search is much more complex.

### The ranking function:

BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document (e.g., their relative proximity). It is not a single function, but actually a whole family of scoring functions, with slightly different components and parameters. One of the most prominent instantiations of the function is as follows.

Given a query *Q*, containing keywords *q*1,...,*qn*, the BM25 score of a document *D* is:

 \text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})},

where *f*(*qi*,*D*) is *qi*'s in the document *D*, | *D* | is the length of the document *D* in words, and *avgdl* is the average document length in the text collection from which documents are drawn. *k*1 and *b* are free parameters, usually chosen as *k*1 = 2.0 and *b* = 0.75. IDF(*qi*) is the IDF weight of the query term *qi*. It is usually computed as:

\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5},

where *N* is the total number of documents in the collection, and *n*(*qi*) is the number of documents containing *qi*.

There are several interpretations for IDF and slight variations on its formula. In the original BM25 derivation, the IDF component is derived from the Binary Independence Model.

In using the above approach, each term in the query is ranked, and the values are summed. ***Freetext queries will add words to the query via inflectional generation*** (stemmed forms of the original query terms); these words are treated as separate terms with no special weighting or relationship with the words from which they were generated. Synonyms generated from the Thesaurus feature are treated as separate, equally weighted terms.

Rank = Σ[Terms in Query] w ( ( ( k1 + 1 ) tf ) / ( K + tf ) ) \* ( ( k3 + 1 ) qtf / ( k3 + qtf ) ) )

Where:

* w is the Robertson-Sparck Jones weight.
* Originally, w is defined as:

w = log10 ( ( ( r + 0.5 ) \* ( N – n – R + r + 0.5 ) ) / ( ( R – r + 0.5 ) \* ( n – r + 0.5 ) ) )

***This was simplified to:***

w = log10 ( ( ( r + 0.5 ) \* ( N – R + r + 0.5 ) ) / ( ( R – r + 0.5 ) \* ( n – r + 0.5 ) ) )

* R is the number of documents marked relevant by a user. This is not implemented in SQL Server 2005 full-text search, and thus is ignored.
* r is the number of documents marked relevant by a user containing the term. This is not implemented.
* N is the number of documents with values for the property in the query.
* n is the number of documents containing the term.
* K is ( k1 \* ( ( 1 – b ) + ( b \* dl / avdl ) ) ).
* dl is the document length, in word occurrences.
* avdl is the average document length of the property over which the query spans, in word occurrences.
* k1, b, and k3 are the constants 1.2, 0.75, and 8.0, respectively.
* tf is the frequency of the term in a specific document.
* qtf is the frequency of the term in the query.

## Combine the above methods with Chaos Mathematics and magic heppens.

## The name "chaos theory" comes from the fact that the systems that the theory describes are apparently disordered, but chaos theory is really about finding the underlying order in apparently random data.

One mathematician, Helge von Koch, captured this idea in a mathematical construction called the Koch curve. To create a Koch curve, imagine an equilateral triangle. To the middle third of each side, add another equilateral triangle. Keep on adding new triangles to the middle part of each side, and the result is a Koch curve. (See figure 4) A magnification of the Koch curve looks exactly the same as the original. It is another self-similar figure.

The Koch curve brings up an interesting paradox. Each time new triangles are added to the figure, the length of the line gets longer. However, the inner area of the Koch curve remains less than the area of a circle drawn around the original triangle. Essentially, it is a line of infinite length surrounding a finite area. Well certainly, you like I are sitting on the edge of your seat by now in anticipation of understanding how this applies to data. Our team found that this theory, when combined with fractal theory, can give a near perfect representation of the occurrence of business data within the real world. Please, read on.

To get around this Koch imposed difficulty, mathematicians invented fractal dimensions. Fractal comes from the word fractional. The fractal dimension of the Koch curve is somewhere around 1.26. A fractional dimension is impossible to conceive, but it does make sense. The Koch curve is rougher than a smooth curve or line, which has one dimension. Since it is rougher and more crinkly, it is better at taking up space. However, it's not as good at filling up space as a square with two dimensions is, since it doesn't really have any area. So it makes sense that the dimension of the Koch curve is somewhere in between the two.

## Combine the above methods with Fractals and the magic reaches epic proportions.

Fractal has come to mean any image that displays the attribute of self-similarity. The bifurcation diagram of the population equation is fractal. The Lorenz Attractor is fractal. The Koch curve is fractal; data when viewed in mathematical terms is fractal.

A scientist by the name of Feigenbaum, later looked at the bifurcation diagram. He was interested in how fast the bifurcations come. He discovered that they reoccur at a constant rate. The rate was calculated to be 4.669. In other words, he discovered the exact scale of the self-descriptor. Make a diagram 4.669 times smaller, and it looks like the next region of bifurcations. He decided to look at other equations to see if it was possible to determine a scaling factor for them as well. Much to his surprise, the scaling factor was exactly the same. Not only was this complicated equation displaying regularity, the regularity was exactly the same as a much simpler equation. He tried many other functions, and they all produced the same scaling factor, 4.669.

This was a revolutionary discovery. He had found that a whole class of mathematical functions behaved in the same, predictable way. This universality would help other scientists easily analyze chaotic equations. Universality gave scientists the first tools to analyze a chaotic system. Now they could use a simple equation to predict the outcome of a more complex equation.

So, was our a team ground breaker or set of master minds, not a chance? But we did make the rash allegation and assumption that data is part of the natural environment, occurring in a manner that can be represented using several “outside the norm” methods of calculations to determine the order of rank, occurrence, probability of existence, and repetition (self-description).

And that being said, “We are the ECM Library development team and very proud to present to you Agile Content Management.”