Multilingual Semantic Matching with OrdPath in Relational Systems

A Kumaran **Peter Carlin**
Microsoft Corporation
{*kumarana,peterca*}*@microsoft.com*

Abstract

The volume of information in natural languages in electronic format is increasing exponentially. The demographics of users of information management systems are becoming increasingly multilingual. Together these trends create a requirement for information management systems to support processing of information in multiple natural languages *seamlessly. Database systems, the backbones of information management, should support this requirement effectively and efficiently. Earlier research in this area had proposed multilingual operators [7, 8] for relational database systems, and discussed their implementation using existing database features.*

In this paper, we specifically focus on the SemEQUAL *operator [8], implementing a multilingual semantic matching predicate using WordNet [12]. We explore the implementation of* SemEQUAL *using OrdPath [10], a positional representation for nodes of a hierarchy that is used successfully for supporting XML documents in relational systems. We propose the use of OrdPath to represent position within the Wordnet hierarchy, leveraging its ability to compute transitive closures efficiently. We show theoretically that an implementation using OrdPath will outperform those implementations proposed previously. Our initial experimental results confirm this analysis, and show that the OrdPath implementation performs significantly better. Further, since our technique is not specifically rooted to linguistic hierarchies, the same approach may benefit other applications that utilize alternative hierarchical ontologies.*

1 Introduction

The volume of information in *natural languages* in electronic format is increasing exponentially [9] and the demographics of users of information management systems are becoming increasingly multilingual [2, 11]. Together these trends create a requirement for information management systems to support processing of information in multiple *natural languages* seamlessly. Database systems, the backbones of information management, should support this requirement effectively and efficiently. The minimal requirement is that the underlying database engines (typically relational), provide similar functionality and efficiency for multilingual data as that associated with processing unilingual data, for which they are well-known. Earlier research in this area had proposed multilingual functions [7, 8] for relational database systems, and discussed ways of implementing them using existing features provided by the database systems. We specifically focus on the SemEQUAL function,

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

as defined in [8], implementing a boolean predicate *SemEQUAL*($word_1$,$word_2$), which is true if $word_1$, or any of its synonyms, when translated into the language of $word_2$, are in the semantic transitive closure of $word_2$ or its synonyms. In order to implement this operator, the WordNet [12] ontological hierarchy is used, along with the semantic relationships between its component multilingual word forms.

In this paper, we explore the implementation of SemEQUAL using OrdPath [10], a representation of position in a hierarchy that is used successfully for supporting XML documents in relational systems. We propose the use of OrdPath to represent position within the Wordnet hierarchy, leveraging its ability to compute transitive closures efficiently. We show theoretically that an implementation using OrdPath will outperform those implementations proposed previously. Our initial experimental results confirm this analysis, and show that the OrdPath implementation performs significantly better. Further, since our technique is not specifically rooted to linguistic hierarchies, the same approach may benefit other applications that utilize alternative hierarchical ontologies.

1.1 Organization of the Paper

The paper is organized as follows: Section 2 outlines the semantic matching problem and provides a brief overview of WordNet lexical resources and OrdPath. Section 3 outlines the implementation approaches considered. Sections 4 and 5 theoretically and experimentally compare the various approaches and Section 6 concludes the paper, outlining our future research directions.

2 Multilingual Semantic Matching Problem

In this section, we state the problem of multilingual semantic matching of word-forms in different languages. We then provide a brief introduction to the resources used: a standard linguistic resource – the WordNet [12, 4], an ontological operator as defined in [8], and OrdPath, a hierarchy-numbering scheme [10]. Finally we outline simplifications to the problem for purposes of analysis and implementation.

2.1 Problem Definition

Consider a hypothetical multilingual portal, *Books.com*, with a sample product catalog [10] as shown in Figure 1, where the *Category* attribute stores the classification of the book in the original language of publication.

Table 1: **Sample *Books.com* Catalog**

Author	**Author_ FN**	**Title**	**Price**	**Category**	**Language**
Durant	Will	History of Civilization	$ 149.95	History	English
Descartes	Renè	Les Méditations Metaphysiques	€49,00	Philosophie	French
Franklin	Benjamin	Ein Amerikanischer Autobiography	€19,95	Autobiography	German
Gilderhus	Mark	History & Historians	$ 19.95	Historiography	English
Nero	Bicci	Il Coronation del Virgin	€99,00	Arti Fini	Italian
Nehru	Jawaharlal	Letters to My Daughter	£15.00	Journal	English
$\Sigma\alpha\rho\rho\eta$	$\kappa\alpha\tau\epsilon\rho\upsilon\alpha$	$\Pi\alpha\chi\upsilon\delta\alpha\sigma\tau\sigma\Pi\alpha\upsilon o$	€12,00	$M o\upsilon\sigma\kappa\eta$	Greek
Lebrun	François	L'Histoire De La France	€75,00	Histoire	French
Franklin	Benjamin	Un Américain Autobiographie	€19,95	Autobiographie	French

In today's database systems, a query with a selection condition of (Category = 'History'), would return *only* those books that have *Category* as `History` in English, although the catalog also contains history books in French, Greek and German. A multilingual user may be better served, however, if all the history books in all languages (or in a specified set of languages) are returned. A query using the SemEQUAL function of [8] as given below,

```
SELECT Author,Title,Category FROM Books
WHERE Category SemEQUAL 'History'
InLanguages {English, French}
```

and a result set, as given in Table 2, would therefore be desirable.

Table 2: **Multilingual Semantic Selection**

Author	**Title**	**Category**
Durant	History of Civilization	History
Lebrun	L'Histoire De La France	Histoire
Franklin	Un Américain Autobiographie	Autobiographie
Gilderhus	History & Historians	Historiography

It should be noted that the SemEQUAL function shown here is generalized to return not just the tuples that are equivalent in meaning, but also with respect to *specializations*, as in the last two tuples that are reported in the output. `Historiography` (*the science of history making*) and `Autobiography` are specialized branches of `History`. To determine semantic equivalence of word-forms across languages and to characterize the SemEQUAL functionality, we take recourse to WordNet [12], a standard linguistic resource that is available in multiple languages and, very importantly from our perspective, features *interlingual* semantic linkages.

2.2 A Brief Introduction to WordNet

In this section, we provide a brief introduction to WordNet [12, 4]. WordNet arranges the concepts of a language using psycho-linguistic principles, using word-forms as a canonical representation.

2.2.1 Word Form and Word Sense

A word may be thought of as a lexicalized concept; simply, it is the written form of a mental concept that may be an object, action, description, relationship, etc. Formally, it is referred to as a *Word-form*. The concept that it stands for is referred to as *Word-sense*, or in WordNet parlance, *Synset*. The defining philosophy in the design of WordNet is that a synset is sufficient to identify a concept for the user. For example, the word-form `bird` corresponds to several different synsets, two of which are {*a vertebrate animal that can typically fly*} and {*an aircraft*}; each of these two synsets is denoted differently with subscripts in Figure 1. Two words are said to be synonymous, *or semantically the same*, if they have the same synset and hence map to the same mental concept. The synsets are divided into five distinct categories and we explore below only the *Nouns* category. WordNet contains a *lexical matrix* that converts a *word form* (lexicographic representation) to a *word sense* (the semantic atom of the language, namely, the Synset).

2.2.2 Noun Taxonomical Hierarchy

WordNet organizes all relationships between the concepts of a language as a semantic network between synsets. In particular, the nouns in English WordNet are grouped under approximately twenty-five distinct *Semantic Primes* [4], covering distinct conceptual domains, such as *Animal*, *Artifact*, etc. Under each of the semantic primes, the nouns are organized in a taxonomic hierarchy, as shown in Figure 1, with *Hyponyms* links signifying the `is-a` relationships (shown in solid arrows). Efforts similar to the English WordNet are underway [4] in several languages, including Indian, Chinese and European languages. A common feature among such efforts is that they all strive for a taxonomic hierarchy in a respective language that has synsets that may be mapped to a set of English synsets. Further, inter-linking of semantically equivalent synsets between WordNets of different languages are being designed in some languages. Figure 1 shows a simplified interlinked hierarchy (shown as dotted arrows) in English and German.

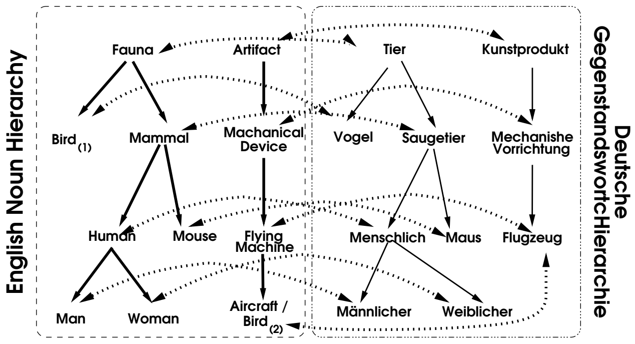

Figure 1: **Sample Interlinked *WordNet* Noun Hierarchy**

2.3 Implementing SemEQUAL using WordNet

WordNet enables mapping word forms to synsets in other languages and comparing them. Denoting an interlinked taxonomic hierarchy of the multilingual strings by $\mathcal{H}_{\mathcal{ML}}$, the SemEQUAL operator is formally defined [8] as follows:

Definition: Given two multilingual noun strings w_i and w_j, and the interlinked multilingual taxonomical hierarchy $\mathcal{H}_{\mathcal{ML}}$, ($w_i$SemEQUAL$w_j$) $\iff$ ($w_i \cap \mathcal{T}_{\mathcal{H}_{\mathcal{ML}}}(w_j) \neq \phi$), where $\mathcal{T}_{\mathcal{H}_{\mathcal{ML}}}(x)$ computes the transitive closure of x in $\mathcal{H}_{\mathcal{ML}}$.

The basic skeleton of the algorithm to semantically match a pair of multilingual strings is outlined in Figure 2. Here, the SemEQUAL function takes two multilingual strings w_1 and w_2 as input. It returns true if the string w_2 is a member of the transitive closure of w_1 in the multilingual taxonomic hierarchy $\mathcal{H}_{\mathcal{ML}}$. Note that the w_2 could be the values from the column Category in the Catalog table, and w_1 could be the user specified category, say `History`.

```
SemEQUAL ($w_1$, $w_2$)
Input: Multilingual Strings $w_1$, $w_2$, Taxonomic Hierarchy $\mathcal{H}_{\mathcal{ML}}$ ( as a resource)
Output: true or false
1.  $\mathcal{TC}_Q \leftarrow$ TransitiveClosure($w_2$, $\mathcal{H}_{\mathcal{ML}}$);
2.  if ($\{w_1\} \cap \mathcal{TC}_Q$) return true else return false;
```

Figure 2: **The SemEQUAL Operator Algorithm**

SemEQUAL as defined in Figure 2 may be implemented with the following 3 steps:

1. All synonyms of $word_1$ in the language of $word_2$, yielding a set of words, $\mathcal{W}_1$.

2. Find the semantic transitive closure of $word_2$ in the taxonomic hierarchy, $\mathcal{H}_{\mathcal{ML}}$, yielding $\mathcal{W}_2$.

3. Output true if $\mathcal{W}_1 \cap \mathcal{W}_2 \neq \phi$.

Though WordNet hierarchies are directed acyclic graphs, we simplified them into trees, by duplicating all shared nodes and their descendants, with multiple parents. Since the number of such shared nodes is not high[3], the additional storage overhead is not significant.

2.4 A Brief Introduction to Ordpath

OrdPath is a tree numbering schema designed for storage and query of XML data [10]. It is one of a number of novel tree numbering schemes devised in recent years [6]. Ordpaths are formed by concatenating bitstrings, each bitstring representing the position of that Ordpath at one level of the tree. Each bitstring is Huffman encoded so small values (and thus tree nodes with small fanout) take less space than large values. For example, using integers to represent the bitstring for each level and delimiting them by periods, an OrdPath $1.5.3$ represents the 3^{rd} child of the 5^{th} child of the root node. An interested reader is referred to [10], for details on Ordpath. Ordpath encoding of hierarchies has the following three key properties:

1. **Encoding of position in the hierarchy** If a and b are two OrdPaths, and the binary comparison $a < b$ is true, then a precedes b in a depth-first traversal in the tree represented by the OrdPaths. This property allows us to test easily whether one OrdPath is in the transitive closure of another: namely, if $a = prefix(b)$, then b exists in the closure of a. This can be easily encapsulated in a function *IsDescendant(path1,path2)* which is true if *path2* is in the closure of *path1*. *IsDescendant(path1,path2)* can be expressed as the conjunctive predicate (*path2* $>=$ *path1 and path2* $<$ *DescendantLimit(path1)*), where *DescendantLimit(path1)* is the largest possible value a descendant of *path1* can have. *DescendantLimit(path1)* can be computed easily.

2. **Small size** Each level of each node in the hierarchy is represented by a variable-length bit string. This results in OrdPaths being quite compact. For instance, WordNet has about $75,000$ nodes, with a maximum depth of 19, a maximum fanout of 398, an average depth of 9.2 and an average fanout of 1. The average Ordpath value representing a node in such a tree will take 30 bits.

3. **Support insert and delete** OrdPaths numbering allow additions and deletions of nodes in the hierarchy, while maintaining the other properties. While we do not anticipate frequent updates to the noun hierarchy, this property allows any domain-specific local updates to the hierarchy to be done efficiently.

3 Implementation Approaches Considered

In this section, we discuss alternative approaches for implementing SemEQUAL in relational database systems. Some of these approaches have been tried in the research literature [8], but are presented here for comparison with the proposed method. Specifically, the following four ways of implementing SemEqual are analyzed. They primarily differ in the representation of the WordNet hierarchy in the relational system, and how the transitive closure is computed.

Parent-child In this representation, the hierarchy relationships are represented as Parent-Child relations. Hence, each row in the table represents a link in the hierarchy. For a given node n, there may be n_{pc} rows in the hierarchy table, each representing a child relationship with n as the parent.

Pre-computed All the transitive ancestor-descendent relationships associated with a node are represented as a set of rows explicitly. Hence, every node n will have a set of n_{ad} relationships, with each row representing one ancestor-descendent relationship between n and a node in its transitive closure.

Inlined pre-computed All the ancestor-descendent relationships associated with a node are represented as one row in the inlined table (assuming that the number of descendents for the node n does not exceed a specified large limit). The inlined descendents are stored in a variable array, associated with the root of the transitive closure.

WordPath The position of a node in the hierarchy is represented by an OrdPath. This representation stores the WordNet hierarchy implicitly.

3.1 Schema Representation for Analysis

In this paper, for the analysis and comparison of different transitive closure computing methodologies, we use a standard Information Retrieval (IR) scenario, where a collection of documents is searched for occurrences of a user-specified specific search term. While a standard search considers an inverted index (or equivalently a standard B+ index structure), SemEQUAL searches the specified search term, and also all elements in its transitive closure with respect to WordNet noun hierarchy. In addition, by appropriate use of multilingual WordNet hierarchies, a query term may be searched across languages, as outlined in [8]. Note that the analysis outlined in this paper applies to other scenarios as well; for instance, SemEQUAL may be used as a join condition between two document collections. However, for clarity of explanation, we restrict our discussion to the simple scenario of searching in a single document collection.

We define the following four tables in this scenario, as explained in detail below. The primary key of a table is indicated with underlining of the appropriate key columns, and the order in which they are defined.

- **Words(<u>Language</u> int, <u>Word</u> string, <u>WordId</u> int)** This table maps words to integers, providing a reference key to be used in other tables, thus reducing storage space. *WordId* is unique across all languages. A secondary index on *WordId* enables translating from *WordId* back to string representation efficiently.

- **Corpus(<u>WordId</u> int, DocumentIdList binary)** This table represents the document collection (referred to as corpus) being searched. Corpus is stored as an inverted index; with a *WordId* as the primary key, accompanied by a compact list of documents that it occurs in. Here we assume that the list of *DocId*s is represented as a list of deltas from one docid to the next, as described in [1].

- **Synset(<u>SourceLanguage</u> int, <u>SourceWordId</u> int, <u>TargetLanguage</u> int, <u>TargetWordId</u> int)** This table stores the set of synonyms of a word, irrespective of the languages, thereby extending the traditional definition of synsets in WordNet, to include the interlingual links between synsets in the languages. This table contains all inter- and intra-language synonyms of words, where intra-language synonyms have the same source and target languages, and inter-language synonyms have different ones.

- **TcWordNet(<u>WordId</u> int, <u>ClosureId</u> int)** This view stores the transitive closure of all of WordNet. Each row of this table contains a *WordId* and one node its closure, represented by *ClosureId*. This is a view because the storage of the transitive closure varies between the approaches.

3.2 Common Query

We use a common IR query of searching a corpus for a given word, for our analysis. The common search query will use SemEQUAL, thus including multilingual semantic search based on the taxonomic hierarchy defined by the interlinked WordNets. It has been shown in [8] that the complexity of *SemEQUAL*(w_1, w_2) is linearly proportional to the number of languages used in the hierarchy. It is reasonable to assume that the set of languages in which the documents in the Corpus table occur is known, or the languages of interest are specified by the user. Therefore, we add a third argument to SemEQUAL, namely *SearchLanguageList*, that specifies the list of languages in which to consider semantic matches in. Such an assumption greatly reduces the effort to generate the transitive closure of the query terms in all languages. The query examined is thus:

```
Select C.DocumentIdList from Corpus C
where SemEqual(C.WordId, @SearchWord, @SearchLanguageList)
```

The parameter `@SearchWord` indicates the search term and `@SearchLanguageList` provides the list of languages in which to look for semantic matches. This query has an algebraic, non-cost-based, rewrite against the above schema, to:

```
Select C.DocumentIdList from Words W where W.Word=@SearchWord
Inner join Synset S
   where S.SourceLanguage = W.Language
   and S.SourceWordId = W.WordId
   and TargetLanguage IN @SearchLanguageList
Inner join TcWordNet T where S.TargetWordId = T.WordId
Inner join Corpus C where C.WordId = T.ClosureId
```

The generated query is used in subsequent analysis, to compare the efficiency of each of the proposed methods to compute the transitive closure.

3.3 Detailed Overview of Approaches

3.3.1 Parent-Child: WordNet(<u>WordId</u> int, ParentWordId int)

In this approach WordNet is stored as a row per parent-child relationship. The transitive closure is computed via a standard SQL:1999 recursive query. Table 3 shows a sample portion of the table.

Table 3: Parent-Child Hierarchy Table

WordId	ParentWordId
Mammal	Animal
Aquatic Mammal	Animal
Dog	Mammal
Tiger	Mammal
Dolphin	Aquatic Mammal

3.3.2 Pre-Computed Closure (PreComputed): WordNetPC (<u>WordId</u> int, <u>ClosureWordId</u> int)

In this approach WordNet is stored with the pre-computed transitive closure in the hierarchy. Each element in a word's transitive closure requires one row. Table 4 shows a sample portion of the table.

Table 4: Pre-Computed Closure

WordId	ClosureWordId
Animal	Mammal
Animal	Aquatic Mammal
Animal	Dog
Animal	Tiger
Animal	Dolphin

3.3.3 Inlined Pre-Computed Closure (Inlined): WordNetIL (WordId int,ClosureIdList array(int))

Assuming an efficient implementation of variable arrays is available in the database system, the duplication of WordId in the pre-computed closure can be eliminated. Alternatively, in database systems that support prefix compression on keys, this methodology may be viewed as the pre-computed closure with prefix compression on WordId. Either way, ordering the closure list on *WordId* maximizes the efficiency of subsequent processing, in particular searching the closure for a particular word. Table 5 shows a sample portion of the table.

Table 5: Inlined Closure

WordId	**ClosureIdList**
Animal	{Mammal, Aquatic Mammal, Dog, Tiger, Dolphin, ...}
Mammal	{Dog, Tiger, ...}
Aquatic Mammal	{Dolphin, ...}

With variable array the in-lined table is unrolled to compute the transitive closure, as follows:

```
Create view TcWordNet as
   Select * from WordNetIL W cross apply unnest(W.ClosureIdList)
```

3.3.4 WordPath

Under this approach, the position of a word in the WordNet hierarchy is encoded by an OrdPath, called a *WordPath* subsequently in this paper. The *WordId* used in other approaches is replaced by *WordPath*. This does not cause an increase in storage size. The key advantage is the elimination of any table representing the WordNet hierarchy. This is possible because the only operation the WordNet table is used for is to enable determining, for any pair of words, whether the first is in the transitive closure of the second. The information necessary to determine this is encoded in the *WordPath* itself. Another advantage is that with the Corpus table having a primary key of *WordPath*, all words in the transitive closure of a given word are co-located. Assuming a type *path* that encapsulates OrdPath functionality, the resulting schema is as follows:

- Words (Language int, Word string, WordPath path)
- Corpus (WordPath path, DocumentIdList binary)
- Synset (SourceLanguage int, SourceWordPath path, TargetLanguage int, TargetWordPath path)

Using the above schema, the standard IR query is rewritten as:

```
Select C.DocumentIdList from Words W where W.Word=@SearchWord
Inner join Synset S
   on S.SourceLanguage=W.Language
   and S.SourceWordPath=W.WordPath
   and TargetLanguage IN @SearchLanguageList
Inner join Corpus C on IsDescendant(C.WordPath, S.TargetWordPath)
```

Comparing this to the query in Section 3.2, the WordPath query eliminates the join with TcWordNet. In addition, the final join between Synset and Corpus is on the *IsDescendant(column,path2)* predicate; as outlined in Section 2.4 this is a conjunctive predicate and the join condition becomes a range seek.

4 Theoretical Analysis

In this section, we present a theoretical analysis of the performance of the *WordPath* approach in comparison to the other approaches outlined in Section 3. Specifically, we compare key parameters affecting relational query performance: the storage size, logical IO, and CPU usage.

4.1 Definitions and Assumptions

First, we define the following symbols and terms that are used in subsequent analysis:

- W - Number of words in WordNet noun hierarchy. For English WordNet, this is 75,000, and it is assumed to be of similar order in other WordNets [4].
- A - Average number of words in the transitive closure for a given word. For the noun hierarchy of English WordNet, this is 9.2.
- S - Average number of synsets for a given word. For English WordNet, s is 1.23, and it is expected to be similar in other WordNets [4].
- M - Average number of corpus documents with a given WordId.
- L - Number of languages for which WordNets are stored and used for processing. Currently, approximately 30 WordNets exist at various stages of completion. [5]
- l - Number of languages involved in a SemEQUAL query. Here assumed to be 3, the average number of languages a user may be interested in (per query).
- C - Average length of a word. For English Wordnet this is 11 characters.
- $PageSize$ - Number of bytes per database page. This number varies from 4K to 64K in various commercial DBMS systems. We assumed 8K for our analysis.

Next, the following simplifying assumptions are made, to make the analysis easier. The first two assumptions affect all approaches proportionately, and hence will not affect the comparisons. The last two assumptions affect WordPath negatively compared to the other approaches, and hence the results provide a lower bound on the relative benefit of WordPath.

- **Non-leaf B-tree page costs are assumed to be 0**. In most schemas the non-leaf tree pages are a small percentage of the database size. Navigation of these pages usually takes a small percentage of the IO and CPU for a given query.
- **Per-row overhead costs are assumed to be 0**. In most database systems the per-row storage overheads are relatively low, typically, 1-2 bytes per row.
- **It is assumed that all IO - Sequential and Random IO - has the same cost**. Clearly, for logical IO this is true, while for physical IO, this assumption is not. However, sequential IO is only possible in the WordPath case or when the storage required for the *DocId*s for a given word exceed a page - in other words, for extremely large Corpus sizes.
- **The average Wordpath size is assumed to be 4 bytes**, the same as that for WordId. Note that the actual average WordPath size for WordNet is less than this.

Given the above definitions, *M*, the number of documents with a given word, determines the scope of a scenario. In the following table, we present some scenarios to indicate approximately the scope of the databases involved. We consider in our analysis document collections up to that corresponding to $M = 100,000$.

Table 6: Values of M and associated scenarios

M	Scenario	Notes
10	Company Product Catalog	10^5 products, 10 words/title
100	Large County Library	10^6 book titles, 10 words/title
1,000	U.S. Library of Congress	10^7 book titles
10,000	PubMed abstract index	10^7 abstracts, 1000 words/abstract
100,000	PubMed article index	10^7 article texts, 10000 words/article

4.2 Storage Size Comparison

Based on the tables described in Sections 3.1 and 3.3, the total storage size of each table, is as shown in Table 7.

Table 7: Table Storage Sizes

Table	Size	Other Structures
Words	$(8 + C) * W * 2$	2 indices
Synsets	$16 * W * S * l$	
WordNet, Parent-Child	$8 * W$	
WordNet, Pre-computed	$8 * A * W$	
WordNet, Inlined	$(4 * A + 4) * W$	
Corpus	$W * M$	

Using the above formulae, the total schema storage size for each of the approaches, and for different scenarios, are computed and shown in Table 8.

Table 8: Storage Sizes (MB)

M	ParentChild	PreComputed	Inlined	WordPath
10	45.4	54.1	49.7	44.8
100	51.8	60.5	56.2	51.2
$1,000$	116.2	124.9	120.5	115.6
$10,000$	759.9	768.6	764.3	759.3
$100,000$	$7,197.2$	$7,205.9$	$7,201.6$	$7,196.6$

The corpus size eventually becomes the most dominant factor. Since in all approaches the corpus table has the same number rows (though not in the same order in each approach), the storage size for all approaches eventually converges. For smaller collections, the Parent-Child and WordPath approaches have approximately equal storage, and have a slight advantage, space-wise.

4.3 Logical IO

For cold queries, defined as those that are run when pages needed for answering the queries are not in-memory, all unique logical IOs become physical IOs and dominate response time. For warm queries, logical IO is still, in general, the leading factor in response time of a query. However, how much logical IO an approach takes depends on the query plan followed. In all scenarios considered here, the best plan is to first lookup relevant synsets, compute their respective closures and then lookup each closure member in the corpus. In the WordPath approach, the best plan is to compute the synsets and lookup each synset in the corpus. Table 9 outlines formulas for the IO required at each stage for every approach.

Table 9: Logical IO for each Step of each Approach

Approach	**GetId**	**Get Synsets**	**Get Closures**	**Corpus Lookup**
ParentChild	1	$16 * SL/PageSize$	$ASl + \lceil 4 * ASl/PageSize \rceil$	$ASl * \lceil M/PageSize \rceil$
PreComputed	1	$16 * SL/PageSize$	$Sl * \lceil 8A/PS \rceil$	$ASl * \lceil M/PageSize \rceil$
Inlined	1	$16 * SL/PageSize$	$Sl * \lceil (4A + 4)/PS \rceil$	$ASl * \lceil M/PageSize \rceil$
WordPath	1	$16 * SL/PageSize$	0	$Sl * \lceil AM/PageSize \rceil$

Below, we present a detailed examination of each of the stages presented in Table 9:

- **GetId:** In all alternatives, this is a single IO to translate from string to WordId or WordPath, as appropriate in an approach.
- **Get Synsets:** In all alternatives, this is a lookup of the query language and word Id/Path, followed by a scan of the synsets in each relevant target language. We assume that the l/L ratio (languages of interest to the query vs total languages represented in the system) can vary and all L languages are fixed and co-located. Further, it is assumed that the l languages are distributed such that all pages for a given path are touched. Note that since $S \doteqdot 1$ and $L < 30$, this is usually a single IO.
- **Get Closures: Basic Parent-Child** Assuming no co-location due to lack of correlation between WordId and position in the hierarchy, each word in the transitive closure will be a separate logical IO. There are $S * l$ closures to compute and A words in the average closure, resulting in $A * S * l$ co-located rows, each 4 bytes long. In addition, there is IO associated with temporary storage of intermediate/final closures.
- **Get Closures: Pre-Computed** These are all co-located for a given synset, but not across synsets or languages. Each closure takes $8 * A$ bytes of space.
- **Get Closures: Inlined** The analysis is same as that of Pre-Computed approach, other than each closure takes $4A + 4$ bytes of space.
- **Get Closures: WordPath** No closure computation is required.
- **Corpus Lookup: (Basic, PreComputed and Inlined)** With no co-location, each of the $A * S * l$ closure elements can be assumed to require separate IO. Each of these lookups requires $M/PageSize$ IOs.
- **Corpus Lookup: WordPath** Synsets and languages are not necessarily co-located, but the A elements of each of these closures are. Hence, each closure takes $A * M/PageSize$ bytes of space.

Using the above formulae, Table 10 shows the total logical IOs required by each approach for different scenarios, and for different implementation methodologies.

Table 10: Logical IO for each Approach in each Scenario

M	**ParentChild**	**PreComputed**	**Inlined**	**WordPath**
10	27	15	15	4
100	27	15	15	4
$1,000$	27	15	15	5
$10,000$	38	26	26	20
$100,000$	159	151	151	143

It may be observed that the WordPath approach shows a significant IO advantage in most of the modeled scenarios for the given set of parameters and assuming frequency of words in the corpus follows a continuous distribution.

4.4 CPU Complexity

Beyond logical IO, CPU complexity is most strongly tied to rows processed and relational operators selected in the execution tree. In Table 11, we compare these factors for each approach.

Table 11: Rows Processed and Relational Operators for each Approach

Approach	**Rows Processed(formula)**	**Rows Processed(number)**	**Operators**
ParentChild	$1 + 2 * Sl + 2 * ASl$	73	$3 + A$
PreComputed	$1 + 2 * Sl + 2 * ASl$	73	4
Inlined	$1 + 2 * Sl + 2 * ASl$	73	4
WordPath	$1 + Sl + ASl$	39	3

We note that the WordPath approach processes only about 50% of the rows, compared with that of other approaches. This is because rows processed is dominated by the $A*S*l$ term, for which WordPath has a leading coefficient of 1 while the others have a 2. We also note that rows processed is independent of Corpus size. Other than Parent-Child which requires several scans of the hierarchy table, the differences in plan complexity are not significant among the other approaches.

5 Experimental Results

We implemented the ParentChild, PreComputed and WordPath approaches as outlined in this paper in SQL Server 2005 using the noun hierarchy of Wordnet Version 1.5. Two simplifications were made to ease implementation: First, the ordpath type in SQL Server is under development, so a static depth-first numbering scheme was used as a substitute. Second, it was assumed that S=1 rather than 1.23. Neither simplification is expected to affect results significantly. Tables 12 and 13 show the results for small corpus sizes M=10 and M=100. In both, search terms were picked so that transitive closure size was equal to the Wordnet average of 9.

Table 12: Performance Comparison (M=10)

Approach	**CPU time**	**Logical IO**	**Physical IO**	**Elapsed time(cold)**
Parent-Child	107413	2442	16	291050
Pre-Computed	1764	26	11	140625
WordPath	1209	8	7	88320

Table 13: Performance Comparison (M=100)

Approach	**CPU time**	**Logical IO**	**Physical IO**	**Elapsed time(cold)**
Parent-Child	108643	2451	19	331926
Pre-Computed	2088	35	14	189983
WordPath	1397	10	8	139406

The above figures indicate that the actual performance is in line with that expected from the theoretical analysis, confirming our claim that OrdPath methodology significantly outperforms earlier implementation methodologies. OrdPath's efficient encoding scheme eliminates the need for explicit computation of transitive closures, and thus exhibits superior performance for SemEQUAL query. We are currently experimenting with larger database sizes (M values up to 100,000) and with a variety of hierarchies. We hope to report a full study in due course.

6 Conclusion & Future Directions

In this paper, we explored the implementation of a multilingual semantic matching function, SemEQUAL, using OrdPath [10] to represent position in the WordNet [12] hierarchy. OrdPath enables efficient determination of membership in transitive closures, a key step in implementing SemEQUAL. We analyzed theoretically the performance of existing implementations, and with proposed OrdPath methodology, and showed that our implementation would incur significantly less IO and CPU costs, resulting in more efficient processing of SemEQUAL. We implemented the various approaches in SQL Server, and the performance figures for small database sizes confirm that the performance of OrdPath is in line with that predicted by our analysis, and significantly better than the exisitng implementations. We are currently undertaking a through study of OrdPath for different sizes of hierarchies, query types and database sizes, and we hope to report the results in due course. Further, since our technique is not specifically rooted to linguistic hierarchies, the same approach may benefit other applications that utilize alternative hierarchical ontologies.

References

[1] Brewer, E. Combining Systems and Databases: A Search Engine Retrospective. *Readings in Database Systems: Fourth Edition*. Joseph M. Hellerstein and Michael Stonebreaker (*eds.*, MIT Press, Cambridge, MA. 2005). *http://www.cs.berkeley.edu/ brewer/papers/SearchDB.pdf*.

[2] The Computer Scope Ltd. *http://www.NUA.ie/Surveys*.

[3] Devitt A. and Vogel, C. The Topology of WordNet: Some Metrics. *Proceedings of the Second Global WordNet Conference*, 2004. *http://www.fi.muni.cz/gwc2004/proc/119.pdf*.

[4] Fellbaum, C. and Miller, G. A. WordNet: An electronic lexical database (language, speech and communication). *MIT Press*, Cambridge, MA, 1998.

[5] Global WordNets in the World. *http://www.globalwordnet.org/gwa/wordnet_table.htm*.

[6] Koch, C., Processing queries on tree-structured data efficiently. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, Chicago, IL. *http://www-dbs.cs.uni-sb.de/ koch/download/pods2006.pdf*.

[7] Kumaran, A., and Haritsa, J. R. LexEQUAL: Multilexical Matching Operator in SQL. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004. *http://portal.acm.org/citation.cfm?id=1007715*.

[8] Kumaran, A., and Haritsa, J. R. SemEQUAL: Multilingual Semantic Matching in Relational Systems. *Proceedings of the* 10^{th} *International Conference on Database Systems for Advanced Applications*, 2005. *http://www.springerlink.com/index/YKKAWFYBB22CGBPF.pdf*.

[9] Lyman, P., and Varian, H. R. How Much Information. *http://www.sims.berkeley.edu/research/projects/how-much-info/*.

[10] O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., and Westbury, N. ORDPATHs: insert-friendly XML node labels. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004. *http://portal.acm.org/citation.cfm?id=1007686#references*.

[11] The WebFountain Project. *http://www.almaden.ibm.com/WebFountain*.

[12] The WordNet. *http://www.cogsci.princeton.edu/w̃n*.