2001

Hierarchical data and the derivational relationship between words

Andrew R. Hippisley
University of Kentucky, andrew.hippisley@uky.edu

Mariam Tariq

David Chang

Hierarchical data and the derivational relationship between words

Andrew Hippisley
Department of Computing
University of Surrey
Guildford, UK

Mariam Tariq
Department of Computing
University of Surrey,
Guildford, UK

David Cheng
Department of Computing
University of Surrey
Guildford, UK

Css1ah@surrey.ac.uk d.chneng@surrey.ac.uk m.tariq@surrey.ac.uk

Abstract

Databases that are currently being developed for repositories of digital linguistic information typically fall into the strictly relational category. We present work on an object-relational database which allows for the representation of hierarchical relationships. This has particular benefits for the storage and access of morphological data in the system of word formation. A model that makes use of inheritance hierarchies elegantly captures the relationship between a root, its derivative, and its derivative, and so on. We report on the experimental Node DataBlade, a software bundle that extends the functionality of the object-relational database system *Informix* (Brown 2001) by supporting the storage and manipulation of hierarchical data. We show how this functionality provides a way of capturing specifically inheritance relationships between members of a derivational family in Russian.

1 Introduction

A *derivational family* contains morphologically associated words and specifies the relationship between them. Dictionaries exist which list such families. An example for Russian is Tixonov's (1985) *Word-formation Dictionary of Russian*. In (1) we have the derivational family of *shkola* 'school'.

(1)

shkol(a)
shkol'**nik**
shkol'niche**sk**(ij)
shkol'nichesk**o**
shkol'**n**(yj)
shkol**i**(t')

Inflectional affixes are parenthesised since in a derivational family it is the relationship between stems that is of interest. For example the head-word *shkol(a)* has the stem *shkol-* where the formative /a/ is an inflectional marker. Affixes that are added at a given derivational layer appear in bold, as in *shkol'**nik*** 'pupil' which has the personal noun suffix *–nik*. Derivational relations are expressed by indentation: immediate derivatives of *shkol(a)* are *shkol'**nik*** 'pupil', *shkol'**n**(yj)* 'school (adjective)' and *shkol**i**(t')* 'to train'. In turn *shkol'**nik*** derives *shkol'niche**sk**(ij)* 'schoolboyish' which derives *shkol'nichesk**i*** 'in a schoolboyish manner'. [1] The derivational family can be interpreted as a hierarchy, shown in Figure 1.

[1] Note that stem final /k/ regularly alternates with /ch/ in the context of the suffix *–sk*.

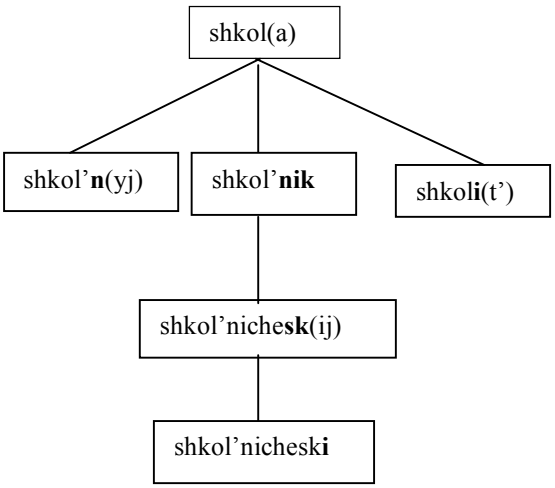

Figure 1. The *shkol(a)* hierarchy

The head word of a derivational family is represented as the root node. Derivatives are represented as daughter nodes, and co-derivatives as sister nodes.

In this paper we show how members of a derivational family can be stored in an object-relational database which represents hierarchically related data in the form of nodes. In section 2 we introduce the object-relational database system. There are two categories of query for eliciting knowledge about derivationally related words that the systems supports. The first is queries about relationships between members of a derivational family, such as what word is a given word derived from. Examples of such queries are outlined in section 3. In section 4 we discuss the second category of query, which elicits knowledge about the lexical productivity of a family and its members.

2 A database for hierarchical data

An object-relational database can be extended by modules that provide additional data types and their related functions. The Informix database system (Brown 2001) has one such module, the Node DataBlade (Hennum 2001), which introduces a data type, together with its related routines, that enables the storage and manipulations of hierarchical data. The data type is the node in a hierarchy of nodes which is identified using the Dewey Decimal Scheme. We can recast the hierarchy in Figure 2 as a list of nodes identified in this way, as in (2).

(2)

1.0 shkola
1.1 shkol'nik
1.1.1 shkol'nicheskij
1.1.1.1 shkol'nichesko
1.2 shkol'nyj (adj)
1.3 shkolit' 'train'
…

The root node is always identified as 1.0, where 1.1, 1.2. and 1.3 identify immediate daughters of the root. In turn 1.1.1 is an immediate daughter of the node 1.1, and so on. The associated routines allow the user to compare and sort nodes identified in this way and hence queries can be made about the relationship between the nodes in the list. In terms of derivational families the nodes represent members, as in Figure 1. We can therefore specify queries for co-derivatives, i.e. sisters of a node, immediate derivative, i.e. daughters of a node, and deriving words, i.e. ancestors of a node, etc.

Using the Node DataBlade derivational families are stored as separate tables, where members of the family are stored in rows with their features. The primary key for each row stores the node data type which identifies the member's status within the hierarchy. For each word member of a family, the features that are stored are illustrated in Table 1:

Features	*Examples*	
node ID	1.0	1.1
word	shkol(a)	shkol'nik
gloss	'school'	'pupil'
affix		-nik
derivational semantics		+person
syntactic category	Noun	Noun
morphological class	class 2	class 1

Table 1: Members and their features

As discussed above, the node ID feature identifies the member's status within the hierarchy. Most members will have a non-null value for the affix feature, which shows the affix used to derive the word. In our example, *shkol'nik* is derived with the affix *–nik*. The head word will usually be a simplex word, hence will have a null value for this feature. In cases of conversion, or zero affixation, a non-head word will lack a value for the affix feature. An example in our database is the adjective *zhilic(yj)* derived from the noun *zhilic(a)* 'female lodger' where affixation has not accompanied change in syntactic category. The derivational semantics feature marks the semantics associated with the derivation. For example, in *shkol'nik* the semantics associated with *–nik* affixation is +person.[2] Finally the syntactic category of each member is given, as well as the morphological class of that category. This is important for Russian where there are four productive classes for nouns, for example.

We populate a table with these features in order to build up a derivational family. Because we are encoding the hierarchical relations using the node data type it is trivial to add nodes to a tree, i.e. add extra members to a given family. For example we might decide to add the perfective verb *vyshkoli(t')* 'to train'. From Figure 1 we would clearly want to position it below the node representing *shkoli(t')*. The node id of *shkoli(t')* is 1.3, as seen from (1), hence using the Dewey Decimal System we specify a node id of 1.3.1 for our data entry, as shown in (3).

(3)

```
INSERT INTO shkola_morph
(node_id, word, gloss, affix,
der_sem, category, morph_class)
VALUES( '1.3.1','vyshkoli(t')',
'train, 'vy-, 'perfective','V',
'V_2');
```

3 Relationships between members of a derivational family

Given a particular complex word it would be useful to know what it is derived from, what its root is, and whether it derives any additional words. This is made possible by the Node data type and the special routines developed to compare the values of this data type. We can illustrate this category of queries using examples from the database.

3.1 Where a word is derived from

For a given complex word we might wish to know what word it is derived from. In terms of the hierarchy in figure 1, we are trying to determine the mother node of our query node. Imagine it is the word *shkol'nichsk(ij)* we are querying. We use the `GetParent` function provided by the Node DataBlade, as in (3) where it is highlighted. The result of the query is given in (5). Note form the SELECT statement that we did not query syntactic category or morphological class.

(4)

```
SELECT word, gloss, affix,
der_sem
   FROM shkola_morph
   WHERE node_id == GetParent(
'1.1.1' );
```

[2] For derivational semantics in Russian the author is referred to Townsend (1975) and Cubberley (1994) which is based on Shvedova (1980).

(5)

Word	Gloss	Affix	Der_sem
shkol'nik	pupil	-nik	person

We can expand the query to include not just the immediately deriving word but the entire derivational chain of our query word. We do this by searching up the branches of the hierarchy to the root node using the special function `isAncestor`. To illustrate consider the complex word *nevydelannost'* which can be roughly glossed as 'having a quality or character of something that has not been manufactured'. The word belongs to the *dela(t')* family which has been stored in the database. In (6) we give the query for the derivational chain of this item, which clearly shows the `isAncestor` function, and in (7) we give a table of results.

(6)

```
SELECT word, gloss, affix,
der_sem
   FROM delat_morph
 WHERE isAncestor( node_id,
'1.9.4.2.1' );
```

(7)

Word	Gloss	Aff	Der_sem
dela(t')	to do		
vydela(t')	manufacture	vy-	finish
vydelann(yj)	manufactured	-nn	participle adjective
nevydelann(yj)	unmanufactured	ne-	negative

The results are read from the bottom to the top. In this way the table can be interpreted as a hierarchy (8).

(8)

nevydelannost' (the query word)
< nevydelann(yj)
< vydelann(yj)
< vydela(t')
< dela(t')

3.2 What a word derives

In section 4 we look more closely at issues of lexical productivity, but for the moment we consider the question of what the actual words are, if any, that a given word in the family derives. In terms of a hierarchy we are searching for any daughter nodes of our query node. For the *shkol(a)* family represented in figure 1, the query would elicit *shkoli(t')*, *shkol'n(yj)* and *shkol'nik*. We illustrate again from the *dela(t')* family. The query is given in (9), and makes use of the `GetParent` function (4) in a special way. The search is for the nodes that are called when the parent is the node 1.0, in other words the daughters of the node 1.0.

(9)

```
SELECT word, gloss, affix,
der_sem
   FROM delat_morph
   WHERE GetParent( node_id )
== '1.0' ;
```

(10)

Word	Gloss	Aff	Der sem
delat's(ja)	to become	-sja	reflexive
delyva(t')	to do often	-iv	frequentative
delanij(e)	doing	-nij	nominalization
del(o)	thing		result noun
delatel'	doer	-tel'	person
izdelij(e)	make,brand	iz-,ij	?
vdela(t')	fit into	v-	in
vozdela(t')	to cultivate	voz-	up
vydela(t')	manufacture	vy-	finish
dodela(t')	finish doing	do-	finish
zadela(t')	block up	za-	close
nadela(t')	make a lot of	na-	quantity
naddela(t')	over do	nad-	over
nedodela(t')	under do	nedo-	under
obdela(t')	polish	ob-	expose to
otdela(t')	finish doing	ot-	finish
peredela(t')	re do	pere-	re
podela(t')	do a little of	po-	little

The results table shows that *dela(t')* is highly productive, forming the base of (at least) eighteen words in the language.

Moreover it is input to a variety of Word Formation Rules (WFRs). These include WFRs making use of suffixation, prefixation, in the case of *del(o)* 'thing' (row 5), zero affixation and even simultaneous prefixation and suffixation in the case of *izdelij(e)* 'brand' (row 7).

A related query is one eliciting the co-derivatives of a word, and we include this for completeness. Consider the complex word *vydelyva(t')*, the derived imperfective of *vydelat*' 'manufacture' , row 3 of (7) and row 10 of (10). It might be interesting to see a list of this word's co-derivatives. In terms of a hierarchy, we search for the sister nodes of the query node using again the function `GetParent`. In the first `GetParent` statement we are querying nodes which are daughters of the node that is the parent of 1.9.1, from the second GetParent statement. The results in (12) list the co-derivatives (sister nodes) of the query word.

(11)

```
SELECT word,gloss,affix,der_sem
  FROM delat_morph
  WHERE GetParent( node_id )==
GetParent( '1.9.1' )
   AND node_id !='1.9.1';
```

(12)

Word	Gloss	Aff	Der sem
vydela(t')sja	be manufactured	-sja	reflexive
vydelk(a)	manufacture	-k	result noun
vydelann(yj)	manufactured	-nn	participle adjective

In this section we have show how the node data type with the special functions `isParent`, `GetParent` and `isAncestor` allow for searches throughout a hierarchy of nodes to elicit information about the relationship between a query word and the members of its derivational family. In the next section we show how queries eliciting statistical information can yield information about the lexical productivity of a given derivational family.

4 Lexical productivity

One important characteristic of a derivational family is the lexical productivity of the family, i.e. the number of words a given member of the family derives. Statistical queries of this kind are possible by combining aggregate statements with the special functions of the Node DataBlade outlined above. In this section we look at the lexical productivity of members of a family. When comparing derivational families with one another, it is useful to elicit a more abstract measure of the productivity a family. We show how this can be done by querying the number of members situated at a given hierarchical level.

4.1 Productivity of family members

The productivity of a given word is based on the number of different words that it derives. In section 3.2 we were interested in what the words were; here we are concerned with how many there are. From figure 1 if we were querying the lexical productivity of *shkol'nichesk(yj)* the result would be one, but for *shkol'n(yj)* it would be zero, for example. Figure 1 is only a partial hierarchy of the *shkol(a)* family; in fact the word *shkol'n(yj)* has a number of derivatives. To find out how many, or how productive *shkol'n(yj)* is, we use the following query which has a mixture of in-built aggregate statements such as `COUNT`, and hierarchy-based definitions peculiar to the Node DateBlade, such as `root` and `branch`.

(13)

```
SELECT root.word,
   COUNT ( branch.node_id )
derivatives
 FROM shkola_morph root,
shkola_morph branch
  WHERE root.node_id <
branch.node_id
AND branch.node_id < Increment
( root.node_id )
AND root.node_id= '1.1'
AND length( branch.node_id ) ==
(length(root.node_id) + 1)
GROUP BY root.node_id,
root.word;
```

The results of the query are given in (14), showing that *shkol'nik* is the base of four words.

(14)

Word	Productivity
shkol'nik	4

To gather information about the productivity of all members of a derivational family, we can use the same query omitting the `AND root.node_id= '1.1'` statement. The results are given in (15) and show that the two most productive members of the family are the head word *shkol(a)* and *shkol'nik*.

(15)

Word	Productivity
shkol(a)	4
shkol'nik	4
shkol'nichesk(ij)	1
shkoljar	3
shkoli(t')	2

The largest family in our database is the *dela(t')* 'to do' family which has 137 members. Using the query above, (16) pinpoints which of these members is the most productive. There are two outliers, the head word *dela(t')* which derives eighteen words, and the noun *del(o)* 'thing' which derives eleven words. For the other members the range is between one and five derivatives.

(16)

Word	Productivity
dela(t')	18
delat's(ja)	2
delanij(e)	4
naddela(t')	4
del(o)	11
deljag(a)	2
del'n(yj)	3
delovit(yj)	2
bezdel'n(yj)	2
bezdelj(e)	1
bezdeljnik	4
bezdeljnicha(t')	2
delatel'	2
delann(yj)	2
vdela(t')	3
vdelyva(t')	2
vdelk(a)	1
vozdela(t')	3
vozdelyva(t')	3
vozdelann(yj)	1
nevozdelann(yj)	1
vydela(t')	4
vydelyva(t')	2
vydelann(yj)	2
nevydelann(yj)	1
dodela(t')	2
dodelyva(t')	2
dodelk(a)	1
zadela(t')	4
zadelyva(t')	2
zadel	2
zadel'shchik	1
nadela(t')	3
nedodela(t')	4
nedodelannyj	1
obdela(t')	5
obdelyva(t')	2
obdelk(a)	1
obdelann(yj)	1
neobdelann(yj)	1
otdela(t')	4
otdelyva(t')	2
otdelk(a)	3
otdelochnik	1
otdel'shchik	1
otdelann(yj)	2
peredela(t')	3
peredelyva(t')	3
peredelyvatel'	1

This lexical frequency may be useful in investigations into the relationship between the frequency of a word in a corpus and its lexical productivity index. The hypothesis is that lexemes with a high token frequency in a corpus of running text will also be lexemes with a high lexical productivity. The explanation may be that high token frequency is an index of psychological salience; only psychologically salient lexemes can be used as the base of a newly derived word. This is because the newly derived word must be semantically compositional, and this relies on (a) the use of a productive derivational operation (Baayen and Lieber 1991) but just as important (b) access to the semantics of the base. From (16) we would expect *dela(t')* and *del(o)* to have a significantly higher token frequency ranking in a corpus than the other members of the family. This would be easy to test given the 1 million word Uppsala corpus, and the frequency analysis carried out on it, reported in Corbett, Hippisley, Brown and Marriott (2001).

4.2 Comparing families for productivity

As well as the productivity of individual members it is interesting to see the overall productivity of one derivational family so that we can compare it with another. We do this by querying the number of nodes positioned at a given level of the hierarchy. Again returning to figure 1 we see that at the top level there is one node, as will be the case for all families since there is only one root node, and at the second level there are three nodes, whereas the third and fourth levels only have one node. We can see level 2 as the most productive level in this hierarchy, and compare this characteristic with other hierarchies that represent derivational families. In (17) we give the query to elicit the productivity of hierarchical levels for the *shkol(a)* family. (18) gives the results of the query.

(17)

```
SELECT length( node_id ) level,
count(*) no_of_words_at_level
   FROM shkola_morph
   GROUP BY 1
   ORDER BY 1;
```

(18)

Level	No. of words at level
	shkol(a)
Level 1	1
Level 2	4
Level 3	9
Level 4	1

From (18) we see that the most productive level is level 3 where over 50% of the entire family's members occur. Only one member is a level 4 derivative, and we might expect the deeper the level, the fewer the derivatives. It might be interesting to see what level derivatives typically occur at by comparing a number of families. (19) and (20) give the results of queries over a number of families.

(19)

level	shkol(a) 'school'	slov(o) 'word'	vremj(a) 'time'
1	1	1	1
2	4	**6**	12
3	**9**	5	**17**
4	1	1	5
5	0	0	1
6	0	0	0
7	0	0	0

(20)

level	zhi(t') 'live'	dela(t') 'do'	chelovek 'person'
1	1	1	1
2	11	18	**9**
3	**13**	**58**	5
4	10	46	2
5	3	8	0
6	0	4	0
7	0	2	0

For each family the most productive level is indicated by a figure in bold. The comparison yields an interesting result, namely that levels 2 and 3 are the most productive level, with *shkol(a)*, *vremj(a)* and *dela(t')* having the majority of their derivatives occur at level 3, and *chelovek* and *slov(o)* using level 2. Only one family, that headed by *dela(t')*, has a good proportion of its members as level 4 derivatives. For all families the lower levels yield the fewest derivatives, as might be expected.

4.3 Feature productivity

In this last sub-section we wish to say something about the features of the members of a derivational family, which were outlined in Table 1, section 2. It would be interesting to compare families according to the proportion of values of a given feature. For example, we might wish to know what proportion of a family is represented by words whose syntactic category is Noun. The query relies on functions normally associated with relational databases, which the DataBlade has access to since it is a module that extends an already existing object-relational database system. The query in (21) can be applied to a number of derivational families. The results are given in (22).

(21)

```
select (100 * (select
count(category) from
delat_morph
where category
='N')/count(category))
percentage from delat_morph;
```

(22)

Derivational Family	Proportion of nouns
Dela(t') 'to do'	42%
Vremja 'time'	39%
Shkola 'school'	53%
Chelovek 'person'	47%
Slovo 'word'	85%

(22) shows that for each family the greater proportion of members are nouns. With the derivational family headed by *slov(o)* as an outlier, the range is between roughly 40% and 50%. Similar queries can be made for the proportion of a particular morphological class, such as class 3 nouns, or of a particular affix, such as the nominalizer *–nij*, and so on.

5 Conclusions

We have shown how the Node data type, developed as an extension to the Informix object-relational database system, allows for the storage and manipulation of derivational word families and their members. This module has great potential as a research tool for investigations into morphological productivity on the one hand, as well as providing access to the hierarchical relations that obtain between derivationally associated words on the other. Further questions remain to be answered. Not least among them is how to develop an automated method for populating tables in the database. While derivational dictionaries such as Tixonov (1985) exist, we know of no such dictionaries in electronic form. A possible method would include scanning a consistently formatted text. Another

direction would be making use of computationally generative WFRs such as those proposed in Hippisley (2001).

References

Baayen, H. and Lieber, R. 1991. Productivity and English derivation: a corpus-based study. *Linguistics* 29. 801-43.

Brown, P. 2001. *Object-relational database development*. New Jersey: Prentice Hall and Informix Press.

Corbett, Greville; Hippisley, Andrew; Brown, Dunstan; and Marriott, Paul. (2001). Frequency, regularity and the paradigm: a perspective from Russian on a complex relation. In: Joan Bybee and Paul Hopper (eds) *Frequency and the Emergence of Linguistic Structure (TSL 45)*. Amsterdam: John Benjamins.201-226

Cubberley, P. 1994. *Handbook of Russian Affixes*. Columbus: Slavica.

Hennum, E. 2001. How to manage hierarchical data with the Node DataBlade module. Located: http://examples.informix.com/doc/case_studies/datablade/node/nodeAll.htm

Hippisley A. 2001. Word formation rules in a default inheritance framework. In: Booij, G and van Marle J (eds) *Yearbook of Morphology 1999*. Dordrecht: Kluwer. 221-261.

Shvedova, N. Ju. 1980. *Russkaja grammatika, tom 1* [= Russian grammar, volume 1]. Moscow: Academy of Sciences.

Tixonov, A. N. 1985. *Slovoobrazovatel´nyj slovar´ russkogo jazyka* [= Wordformation dictionary of Russian]. Moscow: Russkij jazyk.

Townsend, C. 1975. *Russian word-formation*. Columbus: Slavica.