Лингвистическую переменную в общем виде можно описать в виде пятёрки:

{ X, T, U, G, M}, где

X – имя (идентификатор) переменной,

T – множество атомарных термов,

U – множество значений базовой переменной,

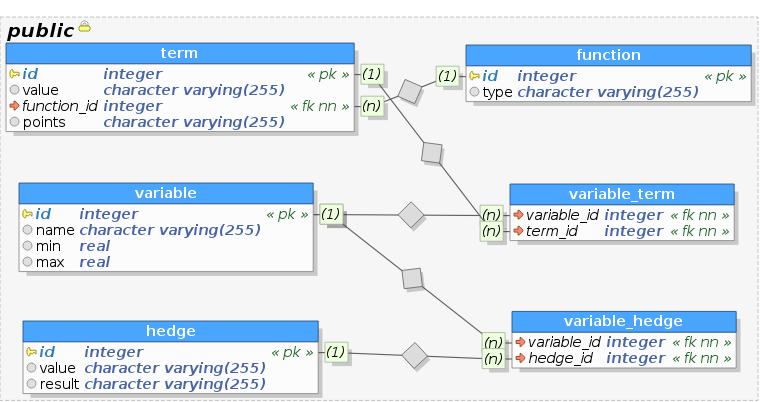
G – грамматика, порождающая новые (составные) термы,

M – семантические правила, которые определяют значение составных термов.

При формировании схемы данных, описывающей представление лингвистических переменных в реляционной СУБД, были рассмотрены следующие предпосылки.

1. Переменные могут быть как простыми, так и составными. К примеру, в выражении «холодная погода» из переменной «погода» можно выделить переменную «температура», к которой и будет относиться терм «холодная». В рамках нашей схемы мы будем рассматривать только простые переменные, а задачу декомпозиции сложных переменных мы вынесем за рамки данной работы.
2. Термы также могут быть простыми (атомарными) и составными. Любой составной терм, порождённый соответствующей грамматикой, может быть представлен в виде дерева разбора, состоящего из элементов грамматики (модификаторов) и атомарных термов. Таким образом, схема будет содержать атомарные термы и модификаторы, но не составные термы. Такой подход позволит вычислять значения сложных термов и синтезировать новые сложные термы.
3. Базовая переменная может иметь как количественную («возраст», «рост»), так и качественную природу («привлекательность»). Далее планируется рассмотрение вопроса о возможности представления качественных переменных в виде комплекса количественных, а пока мы будем рассматривать только количественные базовые переменные.
4. Принадлежность некоторого значения базовой переменной тому или иному терму можно выражать разными способами. В нашей схеме мы будем использовать описание в виде указания типа функции и набора узловых точек.
5. Различные атомарные термы одной лингвистической переменной могут иметь функции принадлежности разных типов.
6. Значения атомарных термов (типы функций и наборы точек) должны быть заданы экспертами до начала работы системы.
7. Модификаторы также должны быть определены заранее экспертами в аналитическом виде. Например, «не» эквивалентно «1 – x», «очень» эквивалентно «x^2».

Принимая во внимание вышесказанное, мы можем описать схему данных для хранения лингвистических переменных следующим образом:



Как было указано выше, составные термы могут быть представлены в виде деревьев разбора. Рассмотрим возможные типы узлов (в круглых скобках) и листьев (в квадратных скобках) таких деревьев.

1. Атомарный терм - [t].
2. Модификатор - [h].
3. Составной терм - (t):

(t) = [h] + [t];

(t) = [h] + (t);

(t) = [h] + (o).

1. Оператор («и», «или») для термов - (o):

(o) = [t] + [t];

(o) = [t] + (t);

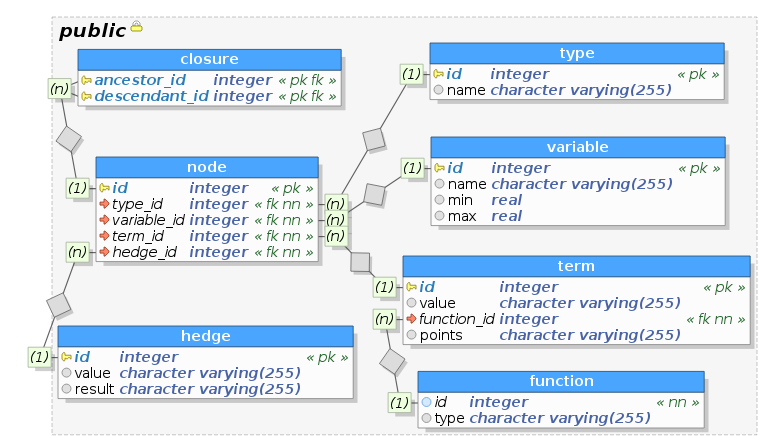
(o) = [t] + (o);

(o) = (t) + (t);

(o) = (t) + (o);

(o) = (o) + (o).

Существует несколько подходов к хранению иерархических структур в СУБД, наиболее перспективными в данном случае видятся adjacency list и closure table. В случае closure table схема данных будет иметь следующий вид:



Рассмотрим теперь вопрос представления нечётких правил в рамках нашей схемы.

1. Левая часть (антецедент) может состоять произвольного количества лингвистических переменных со значениями (термами), связанных логическими операторами.
2. Правая часть (консиквент) может состоять только из одной лингвистической переменной со значением.

Антецедент и консиквент также могут быть представлены деревьями разбора. В дополнение к уже рассмотренным типам узлов и листьев можно выделить следующие.

1. Лингвистическая переменная - [v].
2. Значение переменной - (v):

(v) = [v] + [t];

(v) = [v] + (t);

(v) = [v] + (o).

1. Логический оператор для переменных («и», «или», «не») - (l):

(l) = (v) + (v);

(l) = (v) + (l);

(l) = (l) + (l).

Таким образом, нечёткие правила могут быть представлены следующей схемой данных:

