**УДК 004.89**

# ОБ ОДНОМ РАСШИРЕНИИ ЯЗЫКА МАНИПУЛИРОВАНИЯ ДАННЫМИ

**Попов С.В.**

ООО «Научно-внедренческая фирма БП+», Москва, *e-mail*:s-v-popov@yandex.ru

**Семантическое моделирования представляет собой эффективный инструмент при моделировании различных технологических процессов, создании цифровых двойников и решении задач искусственного интеллекта. Существует несколько языков, использующихся при создании семантических моделей, в той или иной степени эффективных. В этой статье автор предлагает конструкцию еще одного языка семантического моделирования, который представляет собой достаточно своеобразное расширение языка манипулирования данными *SQL*. Идея такого расширения возникла из опыта решения задач искусственного интеллекта, в том числе методами семантического моделирования. Достоинство языка в его относительно простой конструкции, в качестве базисных операций используются конструкции языка *SQL*, расширение происходит путем использования библиотеки *Pandas* языка *Python*. Основным вычислительным средством высокого уровня этого языка является механизм вычисления т.н. *неподвижных точек*, которые по сути и являются семантическими моделями или приближениями к ним. приближение здесь понимается как последовательность возрастающих неподвижных точек, в пределе имеющих окончательную семантическую модель . В результате удается добиться автоматизации построения неподвижных точек, не используя для этого внешних средств управления. Управление вычислением осуществляется самими данными. Этого удается добиться в результате фиксации частичного порядка, который определен на понятиях предметной области. Для повышения эффективности вычислений вводится дополнительное измерение, подобное времени. Это измерение позволяет избежать дублирований в порождении объектов-представителей понятий предметной области. Наконец, в соответствие с частичным порядком множества понятий предметной области конструируется вычислительная сеть, в которой вычисления определяются данными, а не какой-либо системой управления.**

Ключевые слова: семантическая модель, язык *SQL*, понятие, частичный порядок, предметная область, программа, цифровой двойник.

# ABOUT ONE EXTENSION OF THE DATA MANIPULATION LANGUAGE

**Popov S.V.**

LLC "Scientific and implementation firm BP+", Moscow, e-mail:s-v-popov@yandex.ru

**Semantic modeling is an effective tool for modeling various technological processes, creating digital doubles and solving artificial intelligence problems. There are several languages used to create semantic models that are more or less effective. In this article, the author proposes the construction of another semantic modeling language, which is a rather peculiar extension of the SQL data manipulation language. The idea of such an extension arose from the experience of solving artificial intelligence problems, including semantic modeling methods. The advantage of the language is its relatively simple construction, SQL constructs are used as basic operations, the extension occurs by using the Pandas library of Python. The main high-level computing tool of this language is the mechanism for calculating the so-called fixed points, which in fact are semantic models or approximations to them. the approximation here is understood as a sequence of increasing fixed points, in the limit having a final semantic model. As a result, it is possible to achieve automation of the construction of fixed points without using external controls for this. The calculation is controlled by the data itself. This can be achieved as a result of fixing the partial order, which is defined on the concepts of the subject area. To increase the efficiency of calculations, an additional dimension is introduced, similar to time. This dimension avoids duplications in the generation of objects-representatives of the concepts of the subject area. Finally, in accordance with the partial order of the set of concepts of the domain, a computer network is constructed in which calculations are determined by data, and not by any control system.**

Keywords: semantic model, SQL language, concept, partial order, subject area, program, digital twin.

Семантическое моделирование (или, по-другому, *моделирование в понятиях*) широко используется для моделирования технологических процессов и решения задач искусственного интеллекта [1, 2]. Особенно успешно семантическое моделирование используется для создания т.н. цифровых двойников технологического оборудования. Здесь каждому технологическому элементу соответствует свой фрагмент семантической сети. А все модули объединяются в некоторую информационную замкнутую конструкцию, отражающую семантику всей предметной области (здесь: технологического оборудования). Т.к. такая сеть подчиняется логическим закономерностям, то исключены противоречия, ведущие к неправильным моделям. Использование семантических сетей вполне возможно в различных областях: экономика, промышленность, медицина.

Осмысливая задачи, решаемые методами семантического моделирования, можно выделить основные приемы создания моделей, обобщив которые представляется возможным создание удобного языка построения моделей. Чтобы не быть голословным, поясним, что имеется в виду в данном случае. Построение семантической модели сводится к созданию системы понятий и заполнению объектами - их представителями. Для наших целей можно считать, что понятие представляется таблицей БД, записи которой суть представители этого понятия. Тогда вся БД представляет собой семантическую модель некоторой предметной области. Понятно, что при появлении новых первичных данных в БД, модель предметной области перестраивается.

В традиционном представлении расширение БД происходит за счет появления новых первичных данных. В общем случае, это требует пополнения ряда таблиц, непосредственно не связанных с первичными данными, а логически следующими из них. Тем самым, изменение таблиц-понятий модели происходит не только в результате пополнения первичными данными, но и за счет логического вывода. Понятно, что эти логические приемы не являются частью БД, они суть компоненты, наследуемые из предметной области. Так вкратце выглядят принципы реализации дедуктивных баз данных. Для простоты, мы не будем рассматривать случаи, когда возможно вычеркивание данных из таблиц в результате выполнения логического перехода. Т.е. будем находиться в парадигме логики без отрицания в заключении. В парадигме логики с отрицанием в заключении отрицание интерпретируется как вычеркивание данных. Это более общий случай и для первого изложения не годится.

Рассмотрим задачу по наполнению некоторого понятия объектами, которые строятся на основе других понятий. Т.к. понятие есть таблица, то новое понятие – это новая таблица, в общем случае, динамически пополняемая. Пополнение происходит за счет выборки из таблиц понятий-предков, преобразования этой выборки с помощью функций формирования новых

понятий, и вставки результата в таблицу понятия-потомка. Выбор и вставка легко реализуются в *SQL*, однако, преобразование кортежей с целью формирования новых объектов понятий-потомков в *SQL* имеет ограниченный характер. Хотя в общем случае формирование новых представителей понятий носит универсальный характер. Априорно эти процессы ограничить нельзя.

***Пример* 1**. Точка → Пара точек → Отрезок (Имя отрезка, длина отрезка). Если «Имя отрезка» связано с «Парой точек», то «Длина отрезка» с этими точками не связана. Она может задаваться произвольно, и ее значение никак не связано с текущим содержанием модели. Оно определяется из предметной области.

Выделение данных из таблиц в языке *SQL* происходит с помощью достаточно универсального оператора *SELECT*. Вставка данных в таблицу – оператора *INSERT* [3, 4]. Тем самым, присутствует база формализма пополнения понятий-таблиц на основе уже имеющихся данных в языке *SQL*. Вставка кортежей после выделения осуществляется с помощью конструкций *INSERT* … *SELECT* … . Не совсем точно, такие конструкции можно назвать *продукциями*, т.к. они сохраняют основной признак продукций: по имеющимся данным и некоторым условиям генерировать новые данные. Однако, если *SELECT* выделяет кортежи, то в последующем нужно помещать не их самих и не их части, а результат некоторого преобразования. Это может быть результат работы достаточно универсальных функций над частями кортежей, выделенных оператором *SELECT*. В том числе могут быть введены какие-то новые данные, например, как в случае задания длины отрезка на чертеже.

Следовательно, трудность решения проблемы создания нового представителя понятия-потомка состоит в формировании набора функций для преобразования кортежей, выделяемых конструкцией *SELECT* … из имеющихся понятий (таблиц). Т.е. расширении языка *SQL* за счет универсальных средств обработки кортежей. В данном случае может использоваться любой универсальный язык программирования, обеспечивающий удобный интерфейс с БД. Проиллюстрируем это применительно к языку *Python*, в котором есть удобные типы данных для этой цели. Например, фреймы.

В результате получаем расширение языка за счет введения конструкции *A* = *SELECT* …, где *A* переменная типа *фрейма*. В данном случае конструкция *SELECT* … языка *SQL* выделяет набор кортежей, которые размещаются во фрейме, в языке *Python* являющемся аналогом таблицы БД. Далее обработка фрейма происходит в самом языке *Python*. В результате формируется новый фрейм, который через интерфейс *Python* и БД обрабатывается оператором *INSERT*, и помещается в соответствующую таблицу БД. В результате понятие-

потомок расширяется. Это, в определенном смысле напоминает формирование *представлений* в языке *SQL*. Однако представления не выводят нас за рамки реляционной модели данных. А в данном случае желательно находиться в рамках более универсальной парадигмы вычислений, т.к. формирование признаков понятий требует универсальности, отсутствующей при преобразованиях информации в БД.

Предлагаемую конструкцию необходимо обобщить, позволив оператору *INSERT* осуществлять вставку кортежей сразу в несколько таблиц, что существенно ускорит выполнение продукций. В таком случае один сформированный с помощью конструкции *SELECT* фрейм может быть дублирован несколько раз и после обработки из него возникнут пополнения разных таблиц-понятий.

То, что для создания дедуктивной БД, предлагается расширить интерфейс *Python* и БД путем включения промежуточного звена в виде фреймов, не оригинально. Интересно, какие новые возможности в плане автоматизации обработки данных открываются в случае предлагаемого расширения языка *SQL*. Для этого следует уточнить, что является конечным результатом расширения модели при появлении новых первичных данных. Это, пожалуй, основной вопрос формирования модели, т.к. появление первичных данных в одной таблице, в силу логических зависимостей, влечет появление новых объектов–представителей других понятий. Т.е. влечет расширение других таблиц БД. Процесс расширения понятий, в общем случае, зацикливается, в результате возникает взаимосвязанное расширение нескольких таблиц. Этот процесс потенциально бесконечного расширения модели надо как-то измерять, т.к. в одних случаях он может сходиться, когда модель конечная, и достигается ее окончательная конструкция, в других – расходиться, когда модель бесконечная.

Подобное явление наводит на мысль о необходимости создания механизма управления порождением объектов-представителей понятий в логических моделях. Человек, в своих рассуждениях, управляет построением новых объектов, исходя из содержательных представлений о цели исходной задачи. Программа, решающая интеллектуальную задачу, не основывается на содержательных представлениях, поэтому в нее необходимо заложить знания, извлеченные из предметной области. Только тогда ее решения будут полным. В противном случае возможна потеря решений из-за отсутствия в логических построениях некоторых образов элементов предметной области.

Для того, чтобы оценивать сколь продуктивно происходит построение семантической модели, в [5] введено понятие *неподвижной точки*, которая есть признак того, что фрагменты модели или вся модель достигла полноты в отсутствии новых первичных данных. При

введении новых данных, неподвижная точка, естественно меняется. Но мы рассматриваем только операторы *INSERT* и *SELECT*, т.е. находимся в парадигме *позитивной* логики, в которой не происходит вычеркивание строк из таблицы, т.е. удаление объектов-представителей понятий. Поэтому всякая неподвижная точка после добавления новых первичных данных лишь расширяется.

В общем случае, если расширить базисные конструкции языка *SQL* еще и оператором *DELETE*, такая парадигма обработки данных потребует существенной теоретической проработки. Так как в таком случае возникает логическое отрицание, которое как-то должно интерпретироваться. Обычно введение отрицания приводит к существенным усложнениям формализма. Хотя большинство семантических моделей удается формулировать в терминах логики с позитивным заключением. Цель же настоящей статьи дать представление о возможных расширениях языка *SQL*, которые могут иметь существенное практическое значение при построении семантических моделей. Приведенные соображения возникли из практики создания семантических моделей для различных предметных областей.

Обратим внимание на следующее. Упомянутая ранее конструкция *INSERT … SELECT …* вместе с функциями обработки кортежей, определяет частичный порядок на множестве понятий. Действительно, все понятия-предки фигурируют в части *SELECT,* единственное понятие-потомок – в части *INSERT*. Тем самым, задается частичный порядок: предок-потомок. Этот частичный порядок может быть достаточно сложный. И, если использовать стандартные средства управления порождением понятий, то они будут иметь весьма сложную конструкцию.

Следует принять еще один содержательный постулат: *семантическая модель лишь тогда адекватная, когда она представляет собой неподвижную точку*. Действительно, если модель имеет тенденцию к расширению при отсутствии новых первичных данных, то это значит, что в текущем состоянии в ней не хватает некоторых объектов-представителей понятий, которые суть образы реальных понятий предметной области. Следовательно, основным приемом построения модели является нахождение неподвижной точки при очередном пополнении модели первичными данными. Однако, строить неподвижную точку вручную, по очереди запуская конструкции *INSERT … SELECT …* , весьма громоздко, да и возможности человека держать в поле сознания несколько правил пополнения, ограничены. Для решения этой проблемы, можно предложить второй уровень языка, который будет управлять построением объектов модели. Опыт показывает, что такой язык вполне реализуем, его можно назвать языком *описания стратегий* поиска решения. Но он будет эффективен, лишь в случае, когда число порождающих конструкций относительно невелико, а частичный

порядок на понятиях сравнительно прост. Следовательно, необходимо предложить стратегию автоматического порождения неподвижных точек, которая будет запускаться всякий раз, когда происходит пополнение модели новыми первичными данными.

Эта стратегия легко следует из описания частичного порядка понятий. Суть ее сводится к тому, что нужно запускать пополнение потомков, как только происходит обновление непосредственных предков. Стратегия сугубо переборная, и обладает одним существенным недостатком: при вторичном запуске конструкции *INSERT* … *SELECT* … оператором *SELECT* будут выделяться все записи таблиц, что приведет к дублированию некоторых объектов, помещаемых в таблицы оператором *INSERT*. Необходимо сделать так, чтобы конструкция *SELECT* выбирала для анализа только те записи таблиц, которые еще не рассматривались.

И тут мы приходим к необходимости ввести линейный порядок (аналог *времени*), на котором будем фиксировать момент появления нового объекта-представителя понятия, и точку последнего выполнения конструкции *INSERT* … *SELECT* … . Пусть конструкция *SELECT* имела последнее время $t$ выбора объектов $\boldsymbol{o}_1$, $\boldsymbol{o}_2$, …, $\boldsymbol{o}_q$, появившиеся в моменты соответственно $t_1$, $t_2$, …, $t_q$, Тогда новый объект-потомок может появиться лишь в случае, когда $t < \max(t_1, t_2, \ldots, t_q)$. Т.е. время появления, по меньшей мере, одного из объектов $\boldsymbol{o}_1$, $\boldsymbol{o}_2$, …, $\boldsymbol{o}_q$ позже последнего применения указанной операции *SELECT*.

Такой линейный порядок позволяет: во-первых, сохранить стратегию автоматического построения неподвижной точки, ориентируясь только на данные; во-вторых, осуществить управление пополнением понятий, избегнув не продуктивного повторного порождения объектов. Именно повторное порождение понятий приводит к информационному взрыву, когда решаются интеллектуальные задачи. Кроме того, это не потребует расширения языка за счет новой надстройки, управляющей порождением стратегий поиска. Тем самым, можно воспользоваться стандартными средствами СУБД.

Таким образом, предлагаемый язык порождения семантических моделей представляет собой расширение языка *SQL* путем внедрения некоторых конструкций языка *Python* (здесь было проиллюстрировано использование фреймов) и введения дополнительного измерения – времени для фиксирования моментов выполнения операций и генерации новых объектов. Все предлагаемые изменения выглядят вполне обозримыми и в экспериментальных работах себя оправдали.

Но существует еще один, не обычный с точки зрения традиционного программирования, шаг в развитие предлагаемого инструмента построения семантических моделей. Частичное упорядочение понятий, которые в семантической модели имеют своими

образами таблицы БД, позволяет реализовать его в виде вычислительной сети, когда понятия-предки определяют содержание понятий-потомков. При этом, пополнение потомков, как мы показали, может осуществляться автоматически, лишь только предки получают пополнение. Введение временной оси способствует такому автоматизму, т.к. всегда можно сравнить относительную новизну объекта относительно последнего выполнения продукции. Тем самым, получается вычислительная сеть, которая не предполагает участия человека или какой-либо управляющей надстройки для достижения окончательного решения в виде неподвижной точки, т.е. семантической модели предметной области. Эта сеть автоматически создается в соответствие с частичным порядком на множестве понятий предметной области. И ее вычисления определяются теми данными, которыми каждый момент времени пополняются понятия. Такую вычислительную сеть можно назвать самокорректирующейся. Остановка достигается, когда после введения первичных данных удается построить неподвижную точку.

Неким подобием такой самокорректирующейся сети является искусственная нейронная сеть, в которой входные данные при обучении определяют результат промежуточных вычислений и окончательной ошибки. Нейронную сеть легко представить в виде самокорректирующейся сети, если отделить этап вычисления ошибки от этапа коррекции весов. Тогда неподвижная точка достигается, исходя из условия, что ошибка на представительной выборке не меняется. В лучшем случае она становится нулевой.

Преимущество предлагаемого подхода по формированию семантических моделей состоит в следующем. Во-первых, используется реляционная БД для хранения данных, что во многом решает задачу представления и хранения данных. Во-вторых, используется отлаженная методика обработки данных, т.к. привлекаются известные и отлаженные средства как языка *SQL*, так и языка *Python*. Наконец, механизм автоматического формирования неподвижной точки позволит автоматизировать создание семантической модели, не вводя дополнительных конструкций по управлению этим процессом.

Управление вычислением (здесь: построение семантической модели путем наполнения понятий) от данных, в общем случае, может привести к информационному взрыву. В случае введения временного параметра информационный взрыв удается исключить, т.к. все понятия порождаются по мере необходимости. Т.е. при появлении нового содержания в понятиях-предках. Некое подобие этого аппарата было представлено в [5]. Здесь был сформулирован двухуровневый язык управления решением. Первый – продукционный уровень отвечал частичному порядку понятий. Каждое понятие также реализовывалось таблицей. Второй уровень – стратегический, задавал последовательность выполнения стратегий. Цель второго уровня была «экономическая». Сделать вычисление компактным и поэтому быстрым. Методы

этого языка позволяли вполне создавать интересные наработки, например, в области конструирования экспертных систем и решения интеллектуальных задач. Язык был полностью оригинальный, и лишь субъективные причины, приведшие к падению интереса к научным исследованиям в нашей стране, не дали развиться ему в общепризнанный язык для решения интеллектуальных задач.

Литература


1. Michael Kifer, Georg Lausen, James Wu. Logical Foundations of Object Oriented and Frame Based Languages. Journal of ACM 2015, vol. 42, p. 741-843
2. Tim Berners-Lee, The Semantic Web and Research Challenges, http://www.w3.org/2013/Talks/01-sweb-tbl/slide1- 0.html
3. Бен Форта SQL за 10 минут, М.: Вильямс, 2018, -280 с.
4. Прамодкуар Дж. Садаладж, Мартин Фаулер NoSQL: Новая методология разработки нереляционных баз данных. М.: Вильямс, 2017, -350 с.
5. Попов С.В. Логическое моделирование. М.: Тровант, 2016. – 255 с.