Основы моделирования объектно-ориентированных баз данных

К.-Д. Шеве

В статье приводятся результаты о формальном обосновании объектно-ориентированной модели данных; целью работы является внесение вклада в развитие единой и всеобъемлющей математической теории объектно-ориентированных баз данных.

Четкое разграничение объектов и значений в объектно-ориентированной модели данных оказывается крайне существенным. Для структуризации значений и объектов используются, соответственно, типы и классы. Такая структуризация может быть основана на любой системе типов. Описываются различные подходы к системам типов и их семантике и утверждается, что теория объектно-ориентированных моделей данных на основе произвольной системы типов приводит к теории типов с семантикой, определенной в терминах теории топосов.

Как следствие, становится возможным обобщить решения проблем уникальной идентификации объектов и общности. Оказывается, что для этого классы должны быть полностью представимы значениями. Такие классы называются представимыми значениями. При этом идентификаторы объектов превращаются всего лишь в вопрос имплементации. Следовательно, можно рассуждать в терминах, не зависящих от идентификаторов.

Схематика и реализации объектно-ориентированной модели данных формулируются в терминах общей теории категорий. Таким образом, модель оказывается представленной теорией в интуиционистской логике высшего порядка и ассоции-

рованной с топосом, определенным через систему типов. Более того, в случае представимости значениями можно показать, что что идентификаторы объектов могут быть опущены на логическом уровне. Это позволяет строить запросы как логически, так и алгебраически, и устанавливает начальную точку для вывода в объектно-ориентированной модели данных.

1. Введение

Недостатки реляционного подхода к базам данных стимулировали многочисленные исследования, направленные на создание более адекватных моделей данных. Утверждалось, что ключевой технологией систем и языков баз данных будущего станет *объектно-ориентированный подход* [8]. В результате исследований появилось несколько новых систем [5, 6, 7, 9, 19, 20, 21, 22, 24, 27, 38, 40, 41, 70]. Однако, в отличие от исследований в области реляционных баз данных, пока не удалось достичь формального соглашения о том, что именно составляет объектно-ориентированную базу данных [11, 12, 14].

Основной вопрос «Что есть объект?» кажется тривиальным, но уже здесь возникает широкий спектр возможных ответов. В объектно-ориентированном программировании понятие объекта предназначалось для обобщения понятия абстрактного типа данных с дополнительной возможностью *наследования*. В этом смысле объектная ориентированность включает изоляцию данных в полунезависимых модулях для обеспечения высокой производительности разработки программного обеспечения. В объектно-ориентированных базах данных объект также рассматривается как основная единица данных – подход, на выбор которого значительное влияние оказали существующие семантические модели данных [2, 30, 31, 43, 44, 63]. Таким образом, объектно-ориентированные базы данных состоят из независимых объектов, но в то же время они должны предоставлять средства поддержания непротиворечивости на межобъектном уровне; последнее требование в некотором смысле противоречит основам объектной ориентированности.

Теоретические исследования в области объектно-ориентированных баз данных предпринимаются нечасто. Немногочисленные результаты в этой области можно разбить на три группы. К первой группе относятся исследования выразительности и сложности языков запросов, в которых реализованы создание объектов и удаление дублирующихся элементов. Основой здесь являются идеи, заложенные в язык IQL [3]. Результаты второй группы относятся к свойствам объектно-ориентированных моделей данных и их семантическому обоснованию [12, 14, 15, 16, 54, 55]. В исследованиях третьей группы базы данных рассматриваются как теории, определенные логическими программами [4, 37].

Многие исследователи объектно-ориентированных баз данных считают, что объекты – это абстракции, имеющие в своей основе реально существующие вещи, и они должны обладать идентичностью [8]. Таким образом появляется различие между *значениями* и *объектами* [11, 12]. Значение идентифицирует само себя, в то время как идентификация объекта не зависит от хранимых в нем значений. Идентичность объекта как правило определяется кодом из идентификаторов [1, 3, 36]. Абстрагируясь от чисто физического уровня, можно сказать, что идентификатор объекта не изменяется на протяжении жизненного цикла этого объекта. Идентификаторы упрощают разделение и модификацию данных. Однако такие абстрактные идентификаторы не освобождают от задачи обеспечения механизма уникальной идентификации объектов. В объектно-ориентированном программировании достаточно использовать имена объектов, но извлечение большого количества данных по именам теряет всякий смысл.

В большинстве подходов к построению объектно-ориентированных баз данных объекты рассматриваются в паре со значением некоторой фиксированной структуры. С нашей точки зрения это противоречит самой цели использования объектов – абстрагированию от реальности. В реальных ситуациях есть несколько аспектов, изменяющихся с течением времени, которые следовало бы отразить в модели объектов. Как следствие, мы выбрали модель, в которой объект состоит из уникального идентификатора id, множества пар

(тип, значение) (T_i, v_i), множества пар (ссылка, объект) (ref_j, o_j) и множества операций m_k.

Типы используются для структуризации значений. Таким образом, возникает первая задача – выбор семантики системы типов, то есть набор типов, которые могут быть определены и использованы при задании схем. Мы рассмотрим три подхода: простую систему типов с семантикой теории множеств, типизированное λ-исчисление и слегка расширенный вариант *полиморфизма Жирара-Рейнольдса* [17, 42, 48]. Как известно, для третьего подхода не существует теоретико-множественной модели. Однако можно найти подходящую модель в терминах *эффективного топоса* [34, 32, 50] или даже в терминах топосов Гротендика [47]. Более того, если оценивать качество модели с точки зрения вычислительных аспектов, оказывается, что выбор топосо-теоретической модели позволяет реализовать эффективные вычисления [49].

Классы являются структурирующими примитивами для объектов, обладающих одинаковой структурой и поведением. Очевидно, что при рассмотрении объекта с разных позиций может оказаться, что объект принадлежит более чем одному классу, или меняет классовую принадлежность. В объектно-ориентированной модели данных структура класса включает в себя значения объектов и ссылки на объекты. Число классов изменяется с течением времени, в то время как типы остаются неизменными. Отношения между классами задаются ссылками и ссылочными ограничениями на идентификаторы объектов. Более того, каждому классу соответствует набор операций. Схема задается набором определений классов и явно заданными ограничениями целостности. Будет показано, что семантика объектно-ориентированной модели данных может быть задана неким универсальным способом, не зависящим от системы типов, лежащей в основе.

Важный класс проблем в объектно-ориентированных базах данных связан с уникальной идентификацией объектов и существованием общих операций обновления информации [55]. В соответствии с [1, 13], неизменяемая идентификация объектов может быть осуществлена в терминах абстрактных идентификаторов. Основное преимущество этого подхода заключается в том, что в нем легко реализуются разделение данных, изменяемость значений и циклические структуры [46]. С другой стороны, идентификаторы объектов оказываются бессмысленными для пользователя и потому должны быть спрятаны. В случае теоретико-множественной семантики уникальная идентификация основывается на *представимости значениями*. Это понятие может быть расширено на общий случай. То же самое относится и к проблеме обновления.

Далее мы покажем, как классы, схемы и реализации объектов могут быть выражены в терминах теории категорий. Воспользовавшись внутренней логикой топоса, мы сможем определить схемы и реализации объектов посредством теорий; более того, с помощью предикатов существования, идентичности и описания интуиционистской логики мы сможем избавиться от идентификаторов объектов. После задания базовых понятий можно определять алгебраические и логические запросы. Однако последний шаг зависит от представимости значениями, необходимого для общности [55], в то время как уникальная идентификация объектов требует только *слабую идентифицируемость значениями* [16, 55]. Тем не менее можно сделать небольшое обобщение, опущенное в данной работе, которое позволит охватить и этот случай.

Предполагается, что читатель знаком с теорией категорий [10], теорией топосов [35, 39] и их отношением к интуиционистской логике высокого порядка [28, 39, 60].

2. Системы типов

Начнем с краткого рассмотрения трех различных систем типов и их семантики. Первая система представляет собой простую систему типов с теоретико-множественной семантикой, вторая – типизированное лямбда-исчисление с семантикой в декартовых замкнутых категориях, третья – полиморфное типизированное λ-исчисление, или λ-исчисление второго порядка.

Во всех трех системах типы задаются базовыми типами и кон-

структорами. Последние представляют собой типы со свободными (типовыми) переменными. Типы без свободных переменных будем называть *собственными*. Будем считать, что в число базовых типов входит тип абстрактных идентификаторов ID. Тип T, в который не входит идентификатор ID, будем называть *типом значений*.

Простая система типов. Если воспользоваться теоретико-множественной моделью, тип можно представить как неизменяемое подмножество некоторого универсального множества. Для установления связи между значениями различных типов используется отношение подтипизации. Мы будем рассматривать систему, состоящую из *базовых типов*, таких как $BOOL, NAT, INT, STRING$ и т.д., и *конструкторов типов* для записей, конечных множеств, списков и т.д. *Типы* могут быть определены вложением. Более того, мы будем рассматривать и *рекурсивные типы*, семантика которых определяется рациональными деревьями. Дадим более формальное определение системы типов:

$$t := b \mid x \mid (a_1 : t_1, \ldots, a_n : t_n) \mid \{t\} \mid [t] \mid \mu x.t.$$

Семантика типов, представленных множествами значений, определяется обычным способом. Далее систему типов можно расширить за счет *отношения подтипизации* $t' \leqslant t$ [42]. Это отношение задает функции подтипизации $t' \to t$. Подробности мы опустим.

Пусть t' – собственный тип, входящий в тип t. Определим *отношение вхождения*

$$o : t \times t' \to \Omega,$$

где Ω – истинностный объект из универсального множества, например, Ω=BOOL.

Типизированное λ-исчисление. В типизированном λ-исчислении основную роль играют функциональные типы, то есть система типов может быть определена как

$$t := b \mid x \mid (a_1 : t_1, \ldots, a_n : t_n) \mid t_1 \to t_2.$$

Семантика типизированного λ-исчисления может быть описана в терминах декартовых замкнутых категорий.

Полиморфизм. В качестве третьего подхода мы выбрали слегка обогащенную версию *полиморфизма Жирара-Рейнольдса* (Girard-Reynolds Polymorphism, GRP). В этом случае типы задаются языком

$$t := b \mid x \mid t_1 \times \ldots \times t_n \mid t_1 \to t_2 \mid \Pi x.t,$$

где b – некоторый набор базовых типов, включающий тип ID идентификаторов объектов, x – переменная-тип, $\times$ представляет собой произведение типов, $\to$ задает функциональные типы, а Π – непредикативную полиморфную абстракцию с x, пробегающим по всем типам [42, 48].

Прежде всего напомним определение понятия топоса. *Топос E* – это конечно-полная декартово-замкнутая категория с подобъектным классификатором, то есть существует объект Ω и глобальный элемент *истина* : $\mathbb{1} \to \Omega$, такие что для каждого мономорфизма $f : A \to B$ существует единственный классифицирующий морфизм $cl(f) : B \to \Omega$ такой, что f и $triv$ определяют коуниверсальные квадраты $cl(f)$ и *истины*. Здесь $\mathbb{1}$ означает конечный объект.

Теперь определим, что означает модель теории типов в топосе E. GRP-*модель* в топосе E состоит из небольшой внутренней категории $\mathbb{E}$, замкнутой по отношению к конечным произведениям, взятию экспоненты и $Ob(\mathbb{E})$-индексированным произведениям, а также вложение в E, сохраняющее эти свойства. В качестве примера такой модели приведем категорию Per частичных отношений эквивалентности в эффективном топосе Eff.

Более подробная информация о построении подобных моделей содержится в [34, 47].

3. Схематика объектно-ориентированной модели данных

В этом разделе мы опишем слегка модифицированную версию объектно-ориентированной модели данных из [52, 54, 58]. Все объекты реального мира обладают идентичностью. Как следствие, для

описания идентичности объектов вводятся абстрактные (предоставленные системой) идентификаторы. Тем не менее, ни объекты реального мира, ставшие основой абстракции, ни абстрактные идентификаторы не могут быть использованы для идентификации объекта.

В противовес существующим объектно-ориентированным моделям данных [1, 3, 5, 6, 7, 8, 9, 20, 21, 27, 38, 40, 46, 61] объекту не сопоставляется единственным образом тип. Напротив, в реальном мире многие аспекты объектов могут изменяться с течением времени. Поэтому мы решили разрешить ассоциировать объект более чем с одним типом и даже изменять ассоциированные типы на протяжении времени жизни объекта. То же касается и ссылок на другие объекты.

Понятие класса. Понятие класса позволяет группировать объекты, имеющие одинаковую структуру; оно позволяет единым образом охватывать как значения, так и ссылки. Более того, общие операции над объектами, такие как создание и удаление объектов или обновление ссылок и значений, могут быть ассоциированы с классом при условии, что эти операции могут быть определены однозначно. Объекты могут принадлежать к разным классам, что гарантирует, что каждый объект нашей абстрактной модели окажется охваченным хотя бы одним классом. Так же, как значения могут быть определены только через типы, объекты могут быть определены только через классы.

Каждый объект в классе состоит из идентификатора, набора значений и набора ссылок на объекты из других классов. Идентификаторы могут быть представлены с помощью типа уникальных идентификаторов ID. Значения и ссылки могут быть объединены в тип-представление, в котором каждое вхождение идентификатора означает ссылку на объект другого типа. Таким образом, мы можем определить структуру класса, используя типы со свободными переменными.

Что касается динамики, мы будем разделять видимые и скрытые операции, чтобы подчеркнуть различие между операциями, доступными пользователю, и остальными. Все операции на классе, включая скрытые, могут быть вызваны другими операциями, но только скрытые операции могут манипулировать идентификаторами.

Обозначим через N некоторый (довольно большой) набор имен.

(i) Пусть t – тип-значение со свободными переменными $\alpha_1, \ldots, \alpha_n$. Назовем структурным выражением выражение, полученное из t заменой α_i на $r_i : C_i$, где r_i – попарно различные ссылочные имена, а C_i – имена классов $(i = 1, \ldots, n)$.

(ii) Класс состоит из имени класса $C \in N$, структурного выражения S, множества имен надклассов $\{D_1, \ldots, D_m\} \subseteq N$, и набора операторов $\{m_1, \ldots, m_k\}$. r_i называется *ссылкой* из класса C на класс C_i. Тип, полученный из S заменой каждой ссылки $r_i : C_i$ на идентификатор типа ID, называется *представляющим типом* класса C, а тип $U_C = (ident : ID, value :: T_C)$ – типом-классом C.

(iii) *Сигнатура операции* состоит из имени операции $M \in N$, набора пар входной параметр/входной тип $l_i :: T_i$ $(l_i \in N)$ и множества пар выходной параметр/выходной тип $o_j :: T'_j$ $(o_j \in N)$. Будем записывать это как

$$o_1 :: T'_1, \ldots, o_m :: T'_m \leftarrow M(l_1 :: T_1, \ldots, l_n :: T_n).$$

(iv) *Операция* M на классе C состоит из сигнатуры операции с именем M и телом, рекурсивно построенным из следующих конструкций:

(a) присваивания $x := E$, где x является либо переменной-классом x_C, либо локальной переменной из S, а E – терм того же типа, что и x,

(b) операторов *skip* (пропустить), *fail* (отказ), *loop* (цикл),

(c) суперпозиции $S_1; S_2$, выбора $S_1 \square S_2$, проекции $x :: T \mid S$, защиты $\mathcal{P} \to S$, ограниченного выбора $S_1 \boxtimes S_2$, где $\mathcal{P}$ – правильная формула, а x – переменная типа T,

(d) подстановок $x'_1, \ldots, x'_i \leftarrow C' : S'(E'_1, \ldots, E'_j)$, где S' – операция на классе C' с входными параметрами $l'_1, \ldots, l'_j$ и выходными параметрами $o'_1, \ldots, o'_i$, такими, что переменные o'_f, x'_f имеют один и тот же тип и терм E'_g имеет тот же тип, что и переменные l'_g.

(v) Операция M на классе C с сигнатурой $o_1 :: T'_1, \ldots, o_m :: T'_m \leftarrow M(l_1 :: T_1, \ldots, l_n :: T_n)$ называется операцией с *определенным значением* тогда и только тогда, когда все T_i $(i = 1, \ldots, n)$ и T'_j $(j = 1, \ldots, m)$ являются собственными типами-значениями.

(vi) Схема $\mathcal{S}$ – это конечный набор классов $C_1, \ldots, C_n$, замкнутый относительно ссылок, взятия надклассов и вхождений имен классов в операции.

Семантика. Сначала предположим, что лежащая в основе модели система типов имеет теоретико-множественную семантику. Определим реализации схематики объектно-ориентированной модели данных.

Реализация $\mathcal{D}$ схемы $\mathcal{S}$ – это приписывание каждому классу C значения $\mathcal{D}(C)$ типа U_C, при котором выполняются следующие условия:

уникальность идентификаторов: Для каждого класса C справедливо

$$\forall i :: ID. \forall v, w :: T_C. (i, v) \in \mathcal{D}(C) \Rightarrow u = w. \tag{1}$$

целостность включений: Для каждого подкласса C' класса C имеем

$$\forall i :: ID. i \in dom(\mathcal{D}(C)) \Rightarrow i \in dom(\mathcal{D}(C')). \tag{2}$$

Более того, если T_C – подтип типа T'_C с функцией подтипизации $f : T_C \to T'_C$, то:

$$\forall i :: ID. \forall v :: T_C. (i, v) \in \mathcal{D}(C) \Rightarrow (i, f(v)) \in \mathcal{D}(C'). \tag{3}$$

целостность ссылок: Для каждой ссылки из C на C' с соответствующим отношением вхождения o_r справедливо

$$\forall i, j :: ID. \forall v :: T_C. (i, v) \in \mathcal{D}(C) \wedge o_r(v, j) \Rightarrow j \in dom(\mathcal{D}(C')). \tag{4}$$

Опираясь на теорию топосов, мы можем перефразировать определение реализации базы данных. Вместо того, чтобы рассматривать множество $D(C)$, мы можем рассматривать подобъект $DC \overset{J}{\hookrightarrow} ID \times T_C$, то есть мономорфизм в $\mathbb{E}$. Пусть $\pi : ID \times T_C \to ID$ – каноническая проекция. Тогда уникальность идентификаторов означает, что $\pi \circ j$ инъективно. Рассмотрим $j_C : im(DC) \to ID$. $j_C \circ i$ задает факторизацию образа $\pi \circ J$, являющуюся дальнейшей факторизацией фактора, заданного j_D, в том случае, если C является подклассом D. Далее, пусть $\overline{DC}$ – подобъект $DC \times ID$, классифицированный

$$DC \times ID \overset{j \times id}{\hookrightarrow} ID \times T_C \times ID \overset{\pi \times id}{\longrightarrow} T_C \times ID \overset{o_r}{\to} \Omega,$$

где o_r соответствует ссылке из C на D. Факторизация, задаваемая $j_r : im(\overline{DC}) \hookrightarrow ID$, должна быть дальнейшей факторизацией фактора, заданного j_D.

Семантика операций может быть определена через предикатные преобразователи, как показано в [26, 45] для классического случая и в [57] – для семантики, основанной на теории топосов.

Пример. Рассмотрим простой пример университетской базы данных, основанной на простой системе типов с теоретико-множественной семантикой. Сначала вводятся *типы* и *классы*, затем показываются примеры *реализации*.

```
Type ПОЛНОЕ_ИМЯ = (Имя:STRING, Фамилия:STRING,
    Звания:STRING)
Type ЧЕЛОВЕК = (ИдентификационныйНомер:NAT,
    Имя:ПОЛНОЕ_ИМЯ)
Type ЖЕНАТЫЙ_ЧЕЛОВЕК = (ИдентификационныйНомер:NAT,
    Супруг:α)
    Пусть схема состоит из следующих классов:
Class ЧеловекС
    Structure Человек
End ЧеловекС
Class ЖенатыйЧеловекС
    IsA ЧеловекС
    Structure (ИдентификационныйНомер:NAT,
        Супруг:ЖенатыйЧеловекС)
End ЖенатыйЧеловекС
Class СтудентС
```

```
    IsA ЧеловекC
    Structure (НомерСтудента:NAT, Руководитель:ПрофессорC,
        Специальность:ФакультетC)
End СтудентC
Class ПрофессорC
    IsA ЧеловекC
    Structure (ИдентификационныйНомер:NAT, Возраст:NAT,
        Оклад:NAT, Факультет:ФакультетC)
End ПрофессорC
Class ФакультетC
    Structure (НазваниеФакультета:STRING)
End ФакультетC
```

Обозначим через $\mathcal{D}$ имя реализации.

$$\begin{aligned}
\mathcal{D}(\text{ЧеловекC}) = \{&(i_1, (123, (\text{«}John\text{»}, \text{«}Denver\text{»}, \{\text{«}Prof\text{»}, \text{«}Dr\text{»}\}))),\\
&(i_2, (124, (\text{«}Mary\text{»}, \text{«}Stuart\text{»}, \{\text{«}Dr\text{»}\}))),\\
&(i_3, (456, (\text{«}John\text{»}, \text{«}Stuart\text{»}, \{\}))),\\
&(i_4, (567, (\text{«}Laura\text{»}, \text{«}James\text{»}, \{\}))),\\
&(i_5, (987, (\text{«}Dave\text{»}, \text{«}Ford\text{»}, \{\})))\}\\
\mathcal{D}(\text{ЖенатыйЧеловекC}) = \{&(i_1, (123, i_2)),\\
&(i_2, (124, i_1))\}\\
\mathcal{D}(\text{ПрофессорC}) = \{&(i_1, (123, 48, 8000, i_6))\}\\
\mathcal{D}(\text{СтудентC}) = \{&(i_3, (456, 1023, (\text{«}John\text{»}, \text{«}Stuart\text{»}, \{\}), i_1, i_6, i_7)),\\
&(i_4, (567, 2134, (\text{«}Laura\text{»}, \text{«}James\text{»}, \{\}), i_1, i_6, i_7))\}\\
\mathcal{D}(\text{ФакультетC}) = \{&(i_6, (\text{«}ComputerScience\text{»})),\\
&(i_7, (\text{«}Philosophy\text{»})),\\
&(i_8, (\text{«}Music\text{»}))\}
\end{aligned}$$

4. Представимость значениями

С точки зрения объектно-ориентированного подхода база данных представляет собой огромный набор объектов произвольно сложной структуры. Как следствие, возникает проблема уникальной *идентификации* и извлечения объектов из этого набора.

Каждый объект в базе данных является абстракцией объекта реального мира, обладающего идентичностью. Для выражения этой идентичности в объектно-ориентированной модели данных используются абстрактные уникальные идентификаторы, составляющие тип ID. Такие идентификаторы можно считать неизменяемыми. Однако с точки зрения системы перестановка или утеря идентификаторов сама по себе не должна повлиять на поведение базы данных в целом.

Для пользователя абстрактные идентификаторы объектов не несут никакой смысловой нагрузки. Мы покажем, что уникальная идентификация объектов в классе приводит к понятию *идентифицируемости значениями*. Для корректного определения общих операций обновления требуется вводить более сильное понятие – *представимости значениями*. Теоретико-множественный случай представимости значениями рассмотрен в [54, 55].

(i) Класс C называется *идентифицируемым значениями*, если и только если существует собственный класс значений I_C, называемый типом значений-идентификаторов, такой, что для всех реализаций $\mathcal{D}$ схемы $\mathcal{S}$ существует морфизм $c : T_C \to I_C$ такой, что суперпозиция

$$DC \hookrightarrow ID \times T_C \xrightarrow{\pi_2} T_C \xrightarrow{c} I_C$$

инъективна.

(ii) Класс C называется *представимым значениями*, если и только если существует тип значений-представителей V_C такой, что для всех реализаций $\mathcal{D}$ схемы $\mathcal{S}$ существует морфизм $c : T_C \to V_C$ такой, что для любого типа значений-идентификаторов I_C и любой факторизации образа $T_C \xrightarrow{c_V} DVC \hookrightarrow V_C$ существует морфизм $c' : DVC \to I_C$ с $c_I = c' \circ c_V$.

Легко видеть, что каждый класс, представимый значениями, идентифицируем значениями. Более того, *класс значений-представителей* V_C единственен с точностью до изоморфизма.

Хотелось бы построить алгоритм для вычисления типов V_C и I_C, являющихся собственными типами значений, удовлетворяющих некоторым условиям. Для этого мы естественным образом расширим отношение подтипизации на структурные выражения, позаботившись о IsA-отношениях. Далее, каждое структурное надвыражение S' и каждая реализация задают морфизм $IS' : DC \to DC' \hookrightarrow ID \times T_{S'}$, используя представляющий тип $T_{S'}$ S'.

Алгоритм. Положим $F(C_i) = T_i$, если существует структурное надвыражение на C_i, заданное $c_i : T_{C_i} \to T_i$; в противном случае $F(C_i)$ будет неопределенным. Если ID входит в одно из $F(C_i)$, соответствующих $r_j : C_j$ $(j \neq i)$, мы будем писать ID_j.

Далее, пока возможно, следует применять следующие правила:

(i) Если $F(C_j)$ - собственный тип значений и ID_j входит в один из $F(C_i)(j \neq i)$, заменим соответствующий ID_j в $F(C_i)$ на $F(C_j)$.

(ii) Если ID_i входит в одно из $F(C_i)$, рекурсивно переопределим $F(C_i)$ как $F(C_i) == S_i$, где S_i получается в результате замены ID_i в $F(C_i)$ на имя типа $F(C_i)$.

Число итераций конечно, так как существует лишь конечное число классов. Если ни одно из правил неприменимо, заменим все оставшиеся вхождения ID_j в $F(C_i)$ именами типов $F(C_j)$, если $F(C_j)$ определено.

Заметим, что алгоритм вычисляет (взаимно) рекурсивные типы.

Граф ссылок класса C в схеме $\mathcal{S}$ – это минимальный нагруженный граф $G_{ref} = (V, E, l)$, удовлетворяющий следующим условиям:

(i) Существует вершина $v_C \in V$ с $l(v_C) = \{t, C\}$, где t – тип верхнего уровня в структурном выражении S класса C.

(ii) Для каждого собственного вхождения типа $t \neq ID$ в T_C существует единственная вершина $v_t \in V$ с $l(v_t) = \{t\}$.

(iii) Для каждой ссылки $r_i : C_i$ в структурном выражении S класса C граф ссылок G^i_{ref} является подграфом G_{ref}.

(iv) Для каждой вершины v_t или v_C, соответствующей $t(x_1, \ldots, x_n)$ в S, существует единственное ребро, соединяющее v_t или v_C соответственно с v_{t_i}, если x_i – класс t_i, или с v_{t_i}, если x_i – ссылка $r_i : C_i$. В первом случае $l(e^{(i)}_t = \{S_i\})$, где S_i – имя соответствующего селектора; во втором случае ребро помечается $\{S_i, r_i\}$.

Пусть $\mathcal{S} = \{C_1, \ldots, C_n\}$ – схема. Пусть $\mathcal{S}' = \{C'_1, \ldots, C'_n\}$ – другая схема, причем для каждого i существует структурное надвыражение на C_i, определенное некоторым $c_i : T_{C_i} \to T_{C'_i}$. Тогда *граф идентификации* G_{id} класса C_i получается из графа ссылок C'_i заменой каждой метки C'_j на C_j.

Используя введенные определения и обозначения, легко увидеть, что для класса C, для которого существуют структурные надвыражения для всех классов C_i, входящие в качестве меток в некоторый граф идентификации G_{id} класса C, тип I_C, построенный нашим алгоритмом по структурным надвыражениям, использованным в определении G_{id}, является собственным типом значений.

Теорема 1. *(i) Пусть C – класс в схеме S, для которого существуют структурные надвыражения для всех классов C_i, входящие в качестве меток в некоторый граф ссылок G_{ref} класса C. Пусть V_C – тип $G(C)$, построенный алгоритмом по отношению к тривиальным структурным надвыражениям. Пусть $I(C)$ – значение $F(C)$, вычисленное алгоритмом по отношению к произвольной системе структурных надвыражений. Тогда C представим значениями; V_C – представляющий тип, и каждый из I_C – тип идентифицирующих значений.*

(ii) Пусть C – класс в схеме $\mathcal{S}$, для которого существует общий метод обновления. Тогда C представим значениями. Более того, все подклассы и надклассы C представимы значениями.

(iii) Пусть C – представимый значениями класс в схеме $\mathcal{S}$, все надклассы и подклассы которого также представимы значениями. Тогда существует единственная общая операция обновления на C.

Доказательство повторяет доказательство для теоретико-множественного случая в [55].

5. Логическая реконструкция

Мы показали, насколько важную роль играет семантика типов в объектно-ориентированной модели данных. Имея топос типов, мы можем определить реализации схемы, основанной на нем. Единственное допущение, которое необходимо сделать, заключается в наличии типа ID идентификаторов объектов. Более того, из [28, 35, 39] известно, что топосы непосредственно связаны с интуиционистской логикой высшего порядка.

В принципе существует два эквивалентных подхода к логике топоса. Первый задается языком Митчела-Бенабу и семантикой Крипке-Джояла [39]. Второй подход основан на языках Формана-Скотта [28] и более общей логике, определяющей синтаксис и интерпретацию (произвольного) топоса.

Мы будем придерживаться второго подхода, так как в нем сразу же появляются равенство, существование и описание [60]. Напомним, что язык Формана-Скотта $\mathcal{L}$ состоит из

- двух множеств $Sort$ и $Const$ классов и констант;
- отображения подмножеств классов $[\cdot] : U_{n\in\mathbb{N}} Sort^n \to Sort$, записываемого как $(A_1, \ldots, A_n) \mapsto [A_1, \ldots, A_n]$;
- семейства счетных множеств $\{Var_s\}_{s\in Sort}$, индексированного классами;
- отображения $\sharp : Const \to Sort$, приписывающего каждой константе ее класс.

Мы также будем использовать $Var = \bigcup_{s\in Sort} Var_s$ для обозначения множества переменных. Если $x \in Var$ – переменная, через $\sharp x$ обозначим ее класс. Через $\mho = []$ обозначим пустое подмножество классов, состоящее из истинностных значений.

Термы $\mathcal{T}_s(\mathcal{L})$ класса $s \in Sort$ языка $\mathcal{L}$ строятся из $\mathcal{L}$ как наименьшие множества, включающие все переменные x класса s, все константы с $\sharp c = s$, и $\mathbf{I}x.\varphi$ для каждой переменной x с $\sharp x = s$ и каждой формулы φ из $\mathcal{T}_s(\mathcal{L})$.

Формулы в $\mathcal{L}$ – это элементы наименьшего множества $\mathcal{F}(\mathcal{L})$, включающего следующие элементы:

- $\mathbf{E}\tau$ для каждого терма $\tau \in \mathcal{T}_s(\mathcal{L})$;
- $\tau \equiv \sigma$ для термов τ, σ одного и того же класса s;
- $\tau(\sigma_1, \ldots, \sigma_n)$ для термов $\sigma_i \in \mathcal{T}_{s_i}(\mathcal{L})$ и $\tau \in \mathcal{T}$;
- $\varphi \wedge \psi$ для формул φ и ψ;
- $\varphi \Rightarrow \psi$ для формул φ и ψ;
- $\forall x.\varphi$ для переменных $x \in Var$ и формул φ.

Теперь можно ввести остальные связки $\neg, \vee, \Leftrightarrow$, предикат $=$ и квантор $\exists$.

Введение символа *описания* $\mathbf{I}$ требует некоторых объяснений. Неформально $\mathbf{I}x.\varphi$ означает *единственное* x, *удовлетворяющее* φ. Однако такое x может и не существовать.

В логике данная проблема решается введением предиката формального *существования* $\mathbf{E}$; $\mathbf{E}\tau$ означает, что τ существует. Это формализуется путем выделения множеств $\tilde{A}$ возможных элементов и разрешения $\mathbf{E}$ выбирать подмножества актуальных элементов. После этого связанным переменным разрешается принимать значения только из области актуальных элементов. При интерпретации логики на топосе данная конструкция отображается в частичный морфизм-классификатор.

Введение предиката существования также модифицирует предикат равенства $=$, который рассматривается как свойство актуальных элементов. Для сравнения возможных элементов вводится предикат *эквивалентности* $\equiv$. Несуществующие элементы считаются эквивалентными. В силу того, что равенство можно определить в терминах предикатов существования и эквивалентности, только $\equiv$ рассматривается как примитивный предикат.

Выше мы уже отмечали, что $\mho$ рассматривается как класс, состоящий из истинностных значений. Следовательно, формула $\tau()$ с $\tau \in \mathcal{T}_{\mho}(\mathcal{L})$ является формулой, утверждающей τ.

Мы опустим описание аксиом и правил, описывающих оператор вывода $\vdash$, а также описание интерпретации $\mathcal{L}$ на произвольном топосе. Заметим только, что каждая теория T в $\mathcal{L}$ канонически определяет топос $\mathbb{E}(T)$, называемый топосом определяемых типов и определяемых тотальных функций, и каждый топос E может быть записан в такой форме. В частности, существует каноническая интерпретация $\mathcal{L}$ в $\mathbb{E}(T)$, являющаяся полной и непротиворечивой.

Для того, чтобы определить $\mathbb{E}(T)$, нам понадобятся определения типов и отношений через специальные синтактические формы. Такие типы отражают множество возможных подмножеств, ассоциированное с множеством подмножеств классов.

Тип A – это терм, имеющий вид $\mathbf{I}y :: [s].\forall x :: s.(\varphi \Leftrightarrow y(x))$. *Отношение* f из s в t – это терм вида $\mathbf{I}z :: [s,t].\forall x :: s, y :: t.(\varphi \Leftrightarrow z(x,y))$. Тип A или отношение f называется *определяемым*, если и только если определяющая формула замкнута.

Воспользуемся более удобным обозначением типа A, определенного формулой φ: $A = \{x :: s|\varphi\}$. Для терма τ класса s получаем формулу $\tau \in A$. Для переменной x с $\sharp x = s$ можно пользоваться кванторами $\forall x \in A$ и $\exists x \in A$.

Отношение f можно обозначить как $f^\sharp(\tau)$ для $\mathbf{I}y :: t.f(\tau, y)$ для $\tau \in \mathcal{T}_s(\mathcal{L})$, даже если мы не знаем, является f графом или функцией. Более того, мы будем пользоваться *функциональной абстракцией*, записывая $\lambda x :: s.\sigma$ вместо $\mathbf{I}z :: [s,t].\forall x :: s, y :: t.(y = \sigma \Leftrightarrow z(x,y))$.

Наконец, два отношения f и g из типа A в тип B назовем *эквивалентными* по отношению к T, если и только если выполнено $T \vdash \forall x \in A.(f^\sharp(x) \equiv g^\sharp(x))$.

Пусть T – теория над $\mathcal{L}$. Топос $\mathbb{E}(T)$ *определяемых типов и определяемых тотальных функций* в качестве объектов содержит определяемые типы $\mathcal{L}$, а в качестве морфизмов из A в B – классы эквивалентности определяемых отношений из A в B, для которых истинно $T \vdash \forall x \in A.f^\sharp(x) \in B$. Для $f \in Hom(A,B)$ и $g \in Hom(B,C)$ суперпозиция $g \circ f \in Hom(A,C)$ определяется $\lambda x \in A.(g^\sharp(f^\sharp(x)))$.

Схематика и реализации как теории. Пусть задан топос E. Сформулируем определение реализации не в терминах категорий, а через ассоциированную логику. Напомним, что классы в логике – объекты в E, константы класса A – морфизмы $c : \mathbb{1} \to \tilde{A}$, где $\eta_A : A \to \tilde{A}$ – *частичный морфизм-классификатор* A [35, 39, 53], и отображение подмножеств классов переводит $A_1, \ldots, A_n$ в $\Omega^{A_1 \times \ldots \times A_n}$.

Рассмотрим мономорфизм $j : DC \hookrightarrow ID \times T_C$ и каноническую проекцию $\pi_1 : Id \times T_C \to ID$. Как и выше, $j_C : im(DC) \hookrightarrow ID$ порождается из факторизации образа $\pi_1 \circ j$. Так как по нашему предположению $\pi_1 \circ j$ инъективно, общее свойство образов порождает единственный мономорфизм $i : im(DC) \to DC$.

Так как мы предположили представимость значениями, $\pi_2 \circ j$ также инъективно. Следовательно, $\pi_2 \circ j$ задает мономорфизм из $im(DC)$, подобъекта ID, в T_C. Значит, общее свойство частичного морфизма-классификатора η_{T_C} порождает единственный мономорфизм $I(C) : ID \to \tilde{T}_C$.

Рассмотрим морфизм $o_r : T_C \times ID \to \Omega$, соответствующий ссылке r из класса C на класс D. Так как $\eta_{T_C} \times I(D) : T_C \times ID \to \tilde{T}_C \times \tilde{T}_D$ определяет мономорфизм, можно рассмотреть частичный морфизм-классификатор для Ω, $\eta_\Omega = id_\Omega$. Он определяет единственный морфизм $\tilde{o}_r : \tilde{T}_C \times \tilde{T}_D \to \Omega$. Положим $I(r) = \hat{\tilde{o}}_r = \tilde{T}_C \to \Omega^{\tilde{T}_D}$ равной сопряжению его экспоненты.

Морфизмы $I(C)$ для всех классов $C \in \mathcal{S}$ и $I(r)$ для всех ссылок в $\mathcal{S}$ (предположим, что каждая ссылка обладает уникальным именем) достаточны для описания объектов. Фактически эти морфизмы можно рассматривать как семантически ассоциированные с реализацией. Синтаксически вместо них можно использовать имена классов и ссылок.

Таким образом, формулы из $\mathbf{EI}o.\varphi$ превращаются в некоторые «основные факты», описывающие некоторую реализацию. Более того, следующие формулы задают аксиоматику схемы $\mathcal{C}$:

$$\forall o.\mathbf{E}C(o) \Rightarrow \mathbf{E}D(o) \text{ если } C \text{ – подкласс } D \tag{5}$$

$$\forall o.\mathbf{E}C(o) \Rightarrow \forall o'.(r(C(o))(D(o')) \Rightarrow \mathbf{E}D(o') \text{ для ссылок } r \text{ из } C \text{ на } D \tag{6}$$

$$\forall o, o'.C(o) = C(o') \Rightarrow o = o' \quad (7)$$

Рассмотрим множество $Ax(\mathcal{S})$ формул (5), (6) и (7), определенных для схемы $\mathcal{S}$. Они соответствуют теории $T_0 = \{\varphi | Ax(\mathcal{S}) \vdash \varphi\}$. Если включить в рассмотрение еще и множество формул $Ax(I)$, заданное некоторой реализацией, быть может, только «основные факты», получится теория $T_0 = \{\varphi | Ax(\mathcal{S}) \cup Ax(I) \vdash\}$.

Заметим, что каждая модель нашей теории в топосе $\mathbb{E}(T)$ определяет логический морфизм $\mathbb{E}(T') \to \mathbb{E}(T)$ [28].

Наконец, заметим, что $I(C)$ может быть построено, даже если не предполагать представимость значениями, но в этом случае не получится мономорфизма. В общем случае факты, такие как $\mathbf{I}o.C(o) = t$, могут не существовать, то есть $\mathbf{EI}o.C(o) = t$ при факторизации может и не отобразиться в истину. Как следствие, единственной моделью станет несовместный топос. Тем не менее, обобщить наши рассуждения на случай слабой идентифицируемости значениями можно.

Отбрасывание идентификаторов. В работе [16] идентификаторы объектов были названы понятием, имеющем значение только для имплементации. Это приводит к требованию слабой идентифицируемости значениями. В наших построениях, в которых предполагалось выполнение более сильного условия представимости значениями, ненужность идентификаторов в определении схемы определяется тем фактом, что $I(C)$ является морфизмом в $\tilde{T}_C$, а не в $ID \times T_C$, как при первоначальном построении.

Тем не менее, тип $\tilde{T}_C$ все еще включает идентификаторы, соответствующие ссылкам, но, как показано в [53, 54, 55], типы значений, которые могут использоваться для идентификации объектов, могут быть эффективно вычислены. Приведем соответствующие построения для E.

Итак, рассмотрим коуниверсальные квадраты $I(r) : \tilde{T}_C \to \Omega^{\tilde{T}_D}$ и $id_{\Omega^{\tilde{T}_D}}$. Таким образом мы определяем объект $T_{C,r,D}$ и морфизмы $exp'(r) : T_{C,r,D} \to \Omega^{\tilde{T}_D}$ и $exp(r) : T_{C,r,D} \to \tilde{T}_C$; последний морфизм инъективен. Так как $I(r) \circ I(C) = id_{\Omega^{\tilde{T}_D}} \circ (I(r) \circ I(C))$, общее свойство коуниверсальных квадратов задаёт единственный мономорфизм $I(C, r, D) : Id \to T_{C,r,D}$ с $exp(r) \circ I(C, r, D) = I(C)$.

Данную конструкцию можно воспроизвести по отношению ко всем мономорфизмам, соответствующим ссылкам, включая построенный выше $exp'(r)$. Это определяет диаграмму $D : \Gamma \to E$. Пусть $\mathcal{O}$ обозначает предел D. Существует единственный мономорфизм $I : ID \to \mathcal{O}$ такой, что все морфизмы $I(C)$ задаются I и D.

Заметим, что можно предположить, что все объекты из $D(\Gamma)$ ограничены, то есть существует мономорфизм в некоторый фиксированный объект $\mathcal{R}$. В этом случае $\mathcal{O}$ оказывается подобъектом $\mathcal{R}$.

Построение $\mathcal{O}$ склеивает типы и ссылки, но все еще не вводит объекты без идентификаторов. Чтобы избавиться от идентификаторов, введем отображение $\pi_C : T_C \to {T^0}_C$, уничтожающее идентификаторы. Формально π_C представляет собой универсальный квадрат $U \to \mathbb{1}$ и $U \to T_C$, где мономорфизмы задают коуниверсальный квадрат сопряжения экспонент $\hat{o}_r$ и $истина \circ triv_{ID}$.

Пусть f, g определяют универсальный квадрат π_C и $I(r)$. Тогда с помощью коуниверсального квадрата f и g можно получить объект ${T^0}_{C,r,D}$ и морфизм $\pi_{C,r,D} : T_{C,r,D} \to {T^0}_{C,r,D}$. Пусть D' – диаграмма, являющаяся расширением D за счет описанных выше морфизмов. Таким образом мы получаем искомую систему типов без типа ID, которой можно пополнить логику.

Запросы. В рамках реляционной модели существует два подхода к запросам – подход, основанный на реляционной алгебре, и подход, основанный на реляционном исчислении. Введем аналогичные конструкции в объектно-ориентированной модели данных.

С точки зрения алгебры мы можем пользоваться всеми операциями, предоставляемыми системой типов. Синтактически это означает, что в качестве запросов могут выступать все замкнутые термы. Семантика определяется морфизмами $t : \mathbb{1} \to \tilde{T}$. Кроме того, каждый класс C определяет запрос с семантикой, заданной $I(C) : ID \to \tilde{T}_C$ в реализации I. Сочетание этих базовых запросов с помощью любых операторов системы типов задает простой язык запросов [55]. Заметим, что в реляционном подслучае мы получим операторы реляционной алгебры за исключением объединения.

Далее, для комбинирования запросов необходимы полиморфные операторы. Для запросов, определенных морфизмами $I_1 \to A$ и

$I_2 \to B$ и функциями $A \to C$ и $B \to C$ можно рассмотреть «внутренний» коуниверсальный квадрат $A \times_C B \to B$, $A \times_C B \to B$, а также «внешний» коуниверсальный квадрат $I = I_1 \times_C I_2$. В силу общности, получим единственный морфизм $I \to A \times_C B$, определяющий семантику запроса-коуниверсального квадрата. В реляционной алгебре таким коуниверсальным квадратам соответствует оператор объединения.

Для классов, содержащих ссылки, можно также рассматривать запросы $\{r/D\}.C$, определенные *подстановкой* класса D вместо ссылки $r : D$. Семантически это соответствует коуниверсальному квадрату $\tilde{T}_C \times_r \Omega^{\tilde{T}_D}$ над $id : \Omega^{\tilde{T}_D} \to \Omega^{\tilde{T}_D}$ и $I(r) : \tilde{T}_C \to \Omega^{\tilde{T}_D}$. В этом случае морфизмы $I(C) : ID \to \tilde{T}_C$ и $I(r) \circ I(C) : ID \to \Omega^{\tilde{T}_D}$ задают единственный мономорфизм $ID \hookrightarrow \tilde{T}_C \times_r \Omega^{\tilde{T}_D}$, определяющий семантику запросов-подстановок в ссылки.

В случае исчисления все гораздо проще, так как мы можем воспользоваться ассоциированной логикой. В силу того, что классы и ссылки включены в логику, запрос просто задается термом $\mathbf{I}x.\varphi$ с определяющей формулой φ. Это обобщает реляционный подход.

6. Заключение

В работе мы описали фундаментальные идеи и логическую семантику, лежащие в основе объектно-ориентированных баз данных. Мы начали с построения отдельных блоков схематики объектно-ориентированной модели данных, то есть с типов и классов. Сразу же мы убедились в решающем значении семантики типов. Объекты рассматривались как абстракции элементов реального мира, так как они обладают неизменяемой идентичностью. На первом этапе эта идентичность выражается посредством абстрактных идентификаторов из некоторого типа ID. С объектом ассоциируется не только одно значение некоторого типа. Объект может содержать несколько значений разных типов, и этот набор может меняться с течением времени. Для структуризации объектов используются классы. В каждый момент времени класс соответствует набору объектов со

значениями одних и тех же типов и ссылками на объекты из фиксированного набора классов.

Разумно выбрать семантику, основанную на теории топосов. В этом случае сделанные выше рассуждения могут быть обобщены в терминах теории категорий. В данной семантике получено общее решение проблем идентификации и общности. Уникальная идентификация объектов и существование общих операций обновления в классе требуют представимости класса значениями.

Так как теория топосов тесно связана с интуиционистской логикой высшего порядка, мы смогли сформулировать понятия из области объектно-ориентированных баз данных в терминах теории категорий, а затем перевели все в логику. Это позволило определить алгебру и исчисление запросов. Наложив условие представимости значениями, мы показали, как избавиться от идентификаторов объектов, которые, как было показано ранее, являются исключительно вопросом имплементации.

Полученные результаты представляются разумным логическим фундаментом объектно-ориентированных баз данных. Более того, наши результаты позволяют связать базы данных с последними исследованиями в области основ информатики в рамках теории типов и эффективных вычислений.

Тем не менее, наши исследования – это только первый шаг в направлении исследования возможностей вывода в объектно-ориентированных базах данных. В качестве дальнейших направлений исследований выделим геометрические теории (высшего порядка) [39, 68, 69].

Что касается динамики объектно-ориентированных баз данных и формализации семантики операций, мы планируем попробовать аксиоматическую семантику в смысле предикатных преобразователей Дийкстры [23, 26, 45]. Основная проблема с использованием этой теории заключается в том, что она зависит от используемой подходящей логики, в которой гарантируется существование предикатных преобразователей с требуемой семантикой. В классической теории используется бесконечная логика первого порядка $\mathcal{L}^{\omega}{}_{\omega\infty}$; обобщение на логику топосов описано в [53, 57].

Наконец, типы можно сделать более гибкими, если расширить алгебраические спецификации функциональными и истинностными типами высшего порядка и определить топосы – модели таких теорий. Этот подход описан в [53, 56].

Открытой остается проблема связи подобных теорий с теорией синтетических областей (грубо говоря, теорией областей на топосе) [29, 33, 51, 64]. Основное допущение этой теории состоит в том, что «области» являются объектами в топосе, причем все морфизмы между ними непрерывны и все конструкторы базируются исключительно на категориальных свойствах и не ссылаются на свойства, зависящие от порядка теории. И в этом случае эффективный топос становится хорошим источником примеров.

Список литературы

[1] S. Abiteboul: *Towards a deductive object-oriented database language*, Data & Knowledge Engineering, vol. 5, 1990, pp. 263 – 287

[2] S. Abiteboul, R. Hull: *IFO: A Formal Semantic Database Model*, ACM ToDS, vol. 12 (4), December 1987, pp. 525 – 565

[3] S. Abiteboul, P. Kanellakis: *Object Identity as a Query Language Primitive*, in Proc. SIGMOD, Portland Oregon, 1989, pp. 159 – 173

[4] H. Aït-Kaci: *An Overview of LIFE*, in J. W. Schmidt, A. A. Stognij (Eds.): Proc. Next Generation Information Systems Technology , Springer LNCS, vol. 504, 1991, pp. 42 – 58

[5] A. Albano, G. Ghelli, R. Orsini: *Types for Databases: The Galileo Experience*, in Type Systems and Database Programming Languages, University of St. Andrews, Dept. of Mathematical and Computational Sciences, Research Report CS/90/3, 27 – 37

[6] A. Albano, G. Ghelli, R. Orsini: *Objects and Classes for a Database Programming Language*, FIDE technical report 91/16, 1991

[7] A. Albano, G. Ghelli, R. Orsini: *A Relationship Mechanism for a Strongly Typed Object-Oriented Database Programming Language*, in A. Sernadas (Ed.): *Proc. VLDB 91*, Barcelona 1991

[8] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, S. Zdonik: *The Object-Oriented Database System Manifesto*, Proc. 1st DOOD, Kyoto 1989

[9] F. Bancilhon, G. Barbedette, V. Benzaken, C. Delobel, S. Gamerman, C. Lécluse, P. Pfeffer, P. Richard, F. Velez: *The Design and Implementation of O_2, an Object-Oriented Database System*, Proc. of the ooDBS II workshop, Bad Münster, FRG, September 1988

[10] M. Barr, C. Wells: *Category Theory for Computing Science*, Prentice-Hall 1990

[11] C. Beeri: *Formal Models for Object-Oriented Databases*, Proc. 1st DOOD 1989, pp. 370 – 395

[12] C. Beeri: *A formal approach to object-oriented databases*, Data and Knowledge Engineering, vol. 5 (4), 1990, pp. 353 – 382

[13] C. Beeri, Y. Kornatzky: *Algebraic Optimization of Object-Oriented Query Languages*, in S. Abiteboul, P. C. Kanellakis (Eds.): Proc. ICDT '90, Springer LNCS 470, pp. 72 – 88

[14] C. Beeri: *New Data Models and Languages - the Challange* in Proc. PODS '92

[15] C. Beeri, T. Milo: *Subtyping in OODBs*, in Proc. PODS'91

[16] C. Beeri, B. Thalheim: *Can I see your Identification, please?*, Proc. of the Workshop on Database Semantics, Řež, January 1995 (to appear)

[17] K. B. Bruce, A. R. Meyer: *The Semantics of Second Order Polymorphic Lambda Calculus*, in G. Kahn, D. B. MacQueen, G. Plotkin (Eds.): *Semantics of Data Types*, Springer LNCS 173, 1984, 131-144

[18] L. Cardelli, P. Wegner: *On Understanding Types, Data Abstraction and Polymorphism*, ACM Computing Suerveys 17,4, pp 471 – 522

[19] L. Cardelli: *Typeful Programming*, Digital Systems Research Center Reports 45, DEC SRC Palo Alto, May 1989

[20] M. Carey, D. DeWitt, S. Vandenberg: *A Data Model and Query Language for EXODUS*, Proc. ACM SIGMOD 88

[21] M. Caruso, E. Sciore: *The VISION Object-Oriented Database Management System*, Proc. of the Workshop on Database Programming Languages, Roscoff, France, September 1987

[22] R.G.G. Cattell: *Object Data Management: Object Oriented and Extended Relational Database Systems*, Addison-Wesley, 1991

[23] P. Cousot: *Methods and Logics for Proving Programs*, in J. van Leeuwen (Ed.): *The Handbook of Theoretical Computer Science*, vol B: "Formal Models and Semantics", Elsevier, 1990, 841-993

[24] A. Dearle, R. Connor, F. Brown, R. Morrison: *Napier88 - A Database Programming Language?*, in Type Systems and Database Programming Languages, University of St. Andrews, Dept. of Mathematical and Computational Sciences, Research Report CS/90/3, 10 – 26

[25] K. Denninghoff, V. Vianu: *Database Method Schemas and Object Creation*, in Proc. PODS '93, 265-275

[26] E. W. Dijkstra, C. S. Scholten: *Predicate Calculus and Program Semantics*, Springer-Verlag, 1989

[27] D. Fishman, D. Beech, H. Cate, E. Chow et al.: *IRIS: An Object-Oriented Database Management System*, ACM ToIS, vol. 5(1), January 1987

[28] M. P. Fourman: *The Logic of Topoi*, in J. Barwise (Ed.): *Handbook of Mathematical Logic*, North-Holland Studies in Logic, vol. 90, 1977, 1053-1090

[29] P. Freyd: *Recursive Types reduced to Inductive Types*, in J. Mitchell (Ed.): *5th Symposium on Logic in Computer Science*, Philadelphia, 1990

[30] M. Hammer, D. McLeod: *Database Description with SDM: A Semantic Database Model*, J. ACM, vol. 31 (3), 1984, pp. 351 – 386

[31] R. Hull, R. King: *Semantic Database Modeling: Survey, Applications and Research Issues*, ACM Computing Surveys, vol. 19(3), September 1987

[32] J. Hyland: *The Effective Topos*, in A. Troelstra, D. van Dalen (Eds.): *The L.E.J. Brouwer Centenary Symposium*, North Holland, 1982, 165-216

[33] J. Hyland: *First Steps in Synthetic Domain Theory*, in A. Carboni, M. Pedicchio, G. Rosolini (Eds.): *Category Theory '90*, Springer LNM, vol. *1488*, 1992

[34] J. Hyland, E. Robinson, G. Rosolini: *The Discrete Objects in the Effective Topos*, Proc. LMS *60* (1990), 1-60

[35] P. Johnstone: *Topos Theory*, LMS Monographs vol. *10*, Academic Press, 1977

[36] S. Khoshafian, G. Copeland: *Object Identity*, Proc. 1st Int. Conf. on OOPSLA, Portland, Oregon, 1986

[37] M. Kifer, G. Lausen. *F-Logic: A Higher-order Language for Reasoning about Objects, Inheritance and Schema*, in Proc. SIGMOD 1989, 134-146

[38] W. Kim, N. Ballou, J. Banerjee, H. T. Chou, J. Garza, D. Woelk: *Integrating an Object-Oriented Programming System with a Database System*, in Proc. OOPSLA 1988

[39] S. Mac Lane, I. Moerdijk: *Sheaves in Geometry and Logic – A First Introduction to Topos Theory*, Springer Universitext, 1992

[40] D. Maier, J. Stein, A. Ottis, A. Purdy: *Development of an Object-Oriented DBMS*, OOPSLA, September 1986

[41] F. Matthes, J. W. Schmidt: Bulk Types – Add-On or Built-In?, in Proc. DBPL III, Nafplion 1991

[42] J. C. Mitchell: *Type Systems for Programming Languages*, in J. van Leeuwen (Ed.): *The Handbook of Theoretical Computer Science*, vol B: "Formal Models and Semantics", Elsevier, 1990, 365-458

[43] J. Mylopoulos, P. A. Bernstein, H. K. T. Wong: *A Language Facility for Designing Interactive Database-Intensive Applications*, ACM ToDS, vol. 5 (2), April 1980, pp. 185 – 207

[44] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis: *Telos: Representing Knowledge About Information Systems*, ACM ToIS, vol. 8 (4), October 1990 pp. 325 – 362

[45] G. Nelson: *A Generalization of Dijkstra's Calculus*, ACM TOPLAS, vol. 11 (4), October 1989, pp. 517 – 561

[46] A. Ohori: *Representing Object Identity in a Pure Functional Language*, Proc. ICDT 90, Springer LNCS, pp. 41 – 55

[47] A. M. Pitts: *Polymorphism is Set Theoretic, Constructively*, in D.H. Pitt, A. Poigné, D.E. Rydeheard (Eds.): *Category Theory and Computer Science*, Springer LNCS *283*, 12-39

[48] J. C. Reynolds: *Polymorphism is not Set-Theoretic*, in G. Kahn, D. B. MacQueen, G. Plotkin (Eds.): *Semantics of Data Types*, Springer LNCS 173, 1984, 145-156

[49] G. Rosolini: *Categories and Effective Computations*, in D.H. Pitt, A. Poigné, D.E. Rydeheard (Eds.): *Category Theory and Computer Science*, Springer LNCS *283*, 1-11

[50] G. Rosolini, E. Robinson: *Colimit Completions and the Effective Topos*, *Journal of Symbolic Logic* 55 (1990), 678-699

[51] G. Rosolini: *Notes on Synthetic Domain Theory*, University of Genova, February 1995

[52] K.-D. Schewe, B. Thalheim, I. Wetzel, J. W. Schmidt: *Extensible Safe Object-Oriented Design of Database Applications*, University of Rostock, Preprint CS-09-91, September 1991

[53] K.-D. Schewe: *Specification of Data-Intensive Application Systems*, Habilitation Thesis, TU Cottbus, 1994

[54] K.-D. Schewe, J. W. Schmidt, I. Wetzel: *Identification, Genericity and Consistency in Object-Oriented Databases*, in J. Biskup, R. Hull (Eds.): Proc. ICDT '92, Springer LNCS 646, 341-356

[55] K.-D. Schewe, B. Thalheim: *Fundamental Concepts of Object Oriented Databases*, Acta Cybernetica, vol. 11 (4), 1993, 49-84

[56] K.-D. Schewe: *A Semantics for Type Specifications Based on Topos Theory*, TU Cottbus, Technical Report I-5 / 1994

[57] K.-D. Schewe: *A Non-Classical Generalization of Dijkstra's Calculus – Axiomatic Semantics for Typed Program Specifications*, TU Cottbus, Technical Report I-6 / 1994

[58] K.-D. Schewe, B. Thalheim, I. Wetzel: *Foundations of Object Oriented Database Concepts*, University of Hamburg, Report FBI-HH-B-157/92, October 1992

[59] K.-D. Schewe, J. W. Schmidt, D. Stemple, B. Thalheim, I. Wetzel: *A Reflective Approach to Method Generation in Object Oriented Databases*, University of Rostock, Rostocker Informatik Berichte, no. 14, 1992

[60] D. S. Scott: *Identity and Existence in Intuitionistic Logic*, in M. P. Fourman, C. J. Mulvey, D. S. Scott (Eds.): *Applications of Sheaves*, Springer LNM *753*, 660-696

[61] M. H. Scholl, H.-J. Schek: *A Relational Object Model*, in Proc. ICDT 90, Springer LNCS, pp. 89 – 105

[62] D. Stemple, T. Sheard, L. Fegaras: *Reflection: A Bridge from Programming to Database Languages*, in Proc. HICSS '92

[63] S. Y. W. Su: *SAM*: A Semantic Association Model for Corporate and Scientific-Statistical Databases*, Inf. Sci., vol. 29, 1983, pp. 151 – 199

[64] P. Taylor: *The Fixed Point Property in Synthetic Domain Theory*, in G. Kahn: *6th Symposium on Logic in Computer Science*, Amsterdam 1991, 152-160

[65] J. Van den Bussche, Dirk Van Gucht: *A Hierarchy of Faithful Set Creation in Pure OODBs*, in J. Biskup, R. Hull (Eds.): Proc. ICDT '92, Springer LNCS 646, 326-340

[66] J. Van den Bussche, Dirk Van Gucht: *Semi-determinism*, in Proc. PODS '92, ACM Press, 191-201

[67] J. Van den Bussche: *Formal Aspects of Object Identity in Database Manipulation*, Ph.D. Thesis, University of Antwerp, 1993

[68] S. Vickers: *Geometric Theories and Databases*, in M.P. Fourman, P.T. Johnstone, A.M. Pitts (Eds.): Applications of Category Theory in Computer Science, London Mathematical Society Lecture Notes Series, Cambridge University Press, 1992, 288-314

[69] S. Vickers: *Geometric Logic in Computer Science*, in G.L. Burn, S.J. Gray, M.D. Ryan (Eds.): Theory and Formal Methods 1993, Springer WiCS, 1993, 37-54

[70] S.B. Zdonik, D. Maier: *Readings in Object Oriented Database Systems*, Morgan Kaufmann Publishers, 1990

Часть 3.
Математические модели