Модел. и анализ информ. систем. Т. **17**, № **3** (2010) 144–161

УДК 519.854.2

Язык объектных запросов динамической информационной модели DIM

Рублев В.С.[1]

Ярославский государственный университет им. П.Г. Демидова

e-mail: roublev@mail.ru,

получена 21 июля 2010

Ключевые слова: объектная СУБД, язык запросов

Рассматривается задача разработки объектного языка манипулирования данными для новой объектной технологии СУБД DIM [1], который позволял бы просто описывать запросы сложной системы отношений данных этой модели.

1. Постановка задачи

Модель данных в СУБД Динамическая информационная модель DIM (базовые концепции см. в [1]) описывается совокупностями объектов $\mathbb{O}$, свойств объектов $\mathbb{A}$ и классов объектов $\mathbb{C}$, а также базовыми отношениями *наследования, включения, взаимодействия и истории*, вводимыми как для объектов, так и для их классов.

Каждый класс объектов DIM определяется набором *врожденных* свойств, среди которых атрибуты (*параметры* класса) и свойства включения объектов других классов, а также набором *наследуемых* свойств от других (родительских) классов. Каждый объект класса определяется множеством значений свойств класса, а сам класс рассматривается не только как шаблон, определяющий свойства объектов, но и как множество таких объектов.

Отношение наследования классов порождает отношение *наследования объектов*, при котором для любого объекта дочернего класса в каждом его родительском классе определяется точно один *родительский объект*. При этом объектом наследуются значения свойств родительского объекта (в частности, наследуются связи включения других объектов для родительского объекта, если они не определены для дочернего объекта). Родительский объект может быть определен в том же классе дочернего объекта (наследование связей включения других объектов является

[1]Работа проводилась при финансовой поддержке госконтракта 02.740.11.0207, ФЦП "Научные и научно-педагогические кадры инновационной России на 2009 – 2013 годы".

в этом случае единственной причиной такого отношения объектов), и тогда такое наследование называется *внутренним наследованием объектов*. Эта возможность задается отношением *внутреннего наследования для класса*, которое не обязательно для всех объектов класса.

Отношение *включения для классов* в общем случае связывает 3 класса объектов: *включающий* класс, *включаемый* класс и *класс включения*, который характеризует *качество включения* второго класса в первый. При этом любой объект включающего класса может содержать произвольное число объектов включаемого класса (в том числе и нулевое), и возникающее отношение объектов также называется *отношением включения*. В частном случае включающий и включаемый классы могут совпадать, и тогда такое отношение классов и соответствующих объектов называется отношением *внутреннего включения*. Если класс включения задан, то для включающего и включаемого объектов *объект включения* из этого класса определяется единственным образом. Поэтому такое отношение включения классов (и объектов) называется *функциональным* в отличие от *простого*, когда класс включения (и соответственно объект включения) не задается.

Отношение истории для объектов (используются идеи *темпоральных БД*) предполагает задание для каждого объекта интервала его "жизни". При удалении объекта сохраняется время его удаления, и он, возможно, становится объектом-предшественником для некоторого множества новых объектов-последователей с таким же временем создания. В общем случае множество объектов-предшественников может породить другое множество объектов-последователей. Помимо отношения истории объектов вводится отношение истории классов и свойств с аналогичными конструкциями.

Для описания поведения объектов вводится *отношение взаимодействия* классов (и соответственно их объектов). Здесь имеется некоторая аналогия с методами ООБД, где либо объекты являются экземплярами лишь одного класса, либо объекты других классов вводятся как параметры методов, и тогда они не являются методами этих других классов. Поэтому методы ООБД в отличие от *взаимодействий* являются несимметрично определенными. Для взаимодействующих объектов определены их *роли*. В общем случае отношения взаимодействия определяются для объектов 4-х классов с различными ролями их взаимодействия: *Откуда* (класс объектов, являющихся источником взаимодействия), *Куда* (класс объектов, являющихся целью взаимодействия), *Что* (класс объектов взаимодействия) и *Как* (собственно, класс объектов, характеризующих методы взаимодействия)(см. [2],[3]).

Тип объектов определяется свойствами самого класса (параметрами и связями включения), унаследованными свойствами и некоторым множеством взаимодействий, в каждом из которых объект может выступать в одной из ролей, определенных отношением взаимодействия классов.

Таким образом, понятие объекта является сложным, так как для выделения его свойств и их значений требуется работать не только со свойствами класса объекта, но и со свойствами, полученными по наследованию. Поэтому актуальной задачей является введение такого языка, который бы позволял пользователю просто задавать объекты одного класса (или нескольких классов с их связями), учитывая при этом не только параметры и свойства включения класса, но и все наследуемые свойства.

Язык запросов SQL для РСУБД является наглядным, но он не объектен. Группа ODMG, являющаяся создателем одной из технологий ООБД, разработала стандарт объектного язык запросов OQL (см. [4]). Но, во-первых, эта технология не преследует цель создания адаптивных БД, которые могут динамически изменять схему данных, а во-вторых, введенные в ней отношения классов не позволяют адекватно описывать любые дискретные детерминированные модели (см. [1]), что также является обоснованным там же свойством технологии DIM. Поэтому нужен язык объектных запросов, позволяющий манипулировать с данными в DIM. С одной стороны, этот язык при помощи своих конструкций должен точно определять то, что мы желаем выделить. Но с другой стороны, он должен быть прост в использовании с тем, чтобы можно было наглядно задать ту информацию, которую нужно выделить небольшим количеством понятных конструкций.

Построению языка объектно-динамических запросов ODQL (Object Dynamic Query Language), удовлетворяющего этим требованиям, и посвящена данная работа.

2. Два уровня языка запросов ODQL

Указанные требования, предъявленные к языку ODQL, противоречивы. По первому из них он должен содержать конструкции, отвечающие введенным отношениям DIM и, в частности, быть объектным, т. е.

- уметь выделять группу объектов из множества всех объектов того или иного класса по ограничениям на свойства этих объектов;
- уметь выделять свойства объекта независимо от того, являются ли они врожденными или наследуемыми;
- уметь выделять включенные объекты (как включенные в сам объект, так и включенные в его родительские объекты);
- уметь выделять свойства включенных объектов.

Но такие конструкции многочисленны и потому могут сделать запрос сложным и не наглядным, что противоречит второму требованию.

Второе требование весьма важно в одних случаях: пользователь должен уметь наглядно и просто получать доступ к данным и манипулировать ими, не зная в деталях отношения классов соответствующих объектов. В то же время в других случаях пользователь должен иметь возможность применять знание отношений объектов и классов для построения более сложных запросов.

Для разрешения этого противоречия мы вводим 2 уровня запросов ODQL:

- **нижний уровень языка**, позволяющий пользователю, используя знания об отношениях классов, точно описать запрос к данным, указывая при этом все отношения объектов и их классов;
- **верхний уровень языка**, при котором в описании запроса участвуют лишь конструкции, ограничивающие задание свойств группы объектов и класса (или

нескольких связанных классов) выделяемой группы объектов, а также список выделяемых свойств для этих объектов (всю остальную информацию можно получить из метауровня описания классов DIM).

2.1. Нижний уровень ODQL

Запрос в общем случае может содержать фразы **from, select, for, links, where**. Как и для запросов SQL фраза **select** определяет, что будет выбираться, фраза **from** – для каких сущностей следует выбор. Фразы **for, links** и **where** определяют условия выбора, т. е. соответствуют фразе **where** SQL.

2.1.1. Фраза select

Главной фразой запроса является **select**, которая выделяет данные необходимых сущностей:

– либо классы и свойства этих классов (параметры классов и связи классов);

– либо значения параметров объектов (и выражений от таких значений);

– либо объекты (в подзапросах).

В ней можно указать ключевые слова:

– **object**, что означает выделение либо объектов класса, указанного в этой фразе, либо множества "деревьев" связанных объектов классов, задаваемых фразой **from**, если класс фразой **select** не указан (выделение объектов происходит также, если фраза **select** содержит агрегатную функцию, выделяющую объекты);

– **class**, что означает выделение множества классов, определяемых фразами **from** и **links**;

– **property**, что означает выделение свойств классов (а не их значений), определяемых фразами **from** и **links**.

Эти ключевые слова используются для манипулирования данными тех или иных сущностей модели.

Запрос может содержать подзапросы (запросы, заключенные в круглые скобки) в любых фразах запроса, но тип подзапроса (тип выделяемых сущностей) должен соответствовать такой фразе. Подзапросы, выделяющие объекты одного класса, могут соединяться операциями над множествами объектов:

– **union** для объединения множеств объектов;

– **intersection** для пересечения множеств объектов;

– **complement** для дополнения множества объектов (множество объектов класса, не выделяемых подзапросом).

2.1.2. Фраза from

Фраза **from** содержит либо указание множества объектов одного класса, определяемого его именем или подзапросом выделения класса или объектов класса, либо перечисление через запятую имен классов объектов и (или) подзапросов выделения объектов или подзапросов выделения классов. Первый класс, указанный в этой фразе именем или подзапросом (первый класс подзапроса или класс, выделяемый подзапросом), называется *базовым*: выбор фразой **select** делается либо для объектов этого класса, либо для "деревьев" связанных объектов всех классов фразы **from**,

каждое из которых имеет "корень" в базовом классе. Фраза **from** может быть опущена, что будет означать выбор объектов класса, которому принадлежит первый указанный фразой **select** параметр. Чтобы указать класс выбираемого параметра, будем указывать имя этого класса перед именем параметра, разделяя их точкой (либо алиаса имени класса или подзапрпоса). Это делать не обязательно, так как класс каждого свойства однозначно определяется по его имени в силу ограничения однозначности (см. [1]).

2.1.3. Ограничения запроса и его выполнение

Ограничения на данные и связи, которые в SQL задаются фразой **where**, в ODQL разделяются на ограничения связи объектов, которые будем описывать фразой **links**, и ограничения значений параметров классов, которые будем определять фразами **for** (ограничения на значения параметров того или иного класса либо на совокупные значения выражений от параметров такого класса) или **where** (ограничения на совокупные значения параметров разных классов). Такое разделение ограничений может способствовать более эффективной реализации запроса:

– ограничения фразы **links** позволяют выделить классы, связанные с базовым классом, и определить порядок пошагового выполнения запроса при переходе от группы объектов одного класса к группе объектов связанного с ним класса;

– ограничения фразы **for** уменьшают число рассматриваемых объектов на каждом этапе выполнения запроса;

– ограничения фразы **where** позволяют завершить выделение рассматриваемых объектов.

Если запрос содержит подзапросы, то сначала выполняются все подзапросы, не содержащие других подзапросов, затем подзапросы, их содержащие, и т.д.

Выполнение запроса для классов состоит в определении группы классов, имеющих определенные типы связи, задаваемые фразой **links**.

Выполнение запроса для свойств классов состоит в определении свойств группы классов, имеющих определенные связи.

Выполнение запроса для объектов состоит в выделении объектов базового класса, для каждого из которых определяются связанные с ним объекты классов, указанных фразой **links** и удовлетворяющие ограничениям фраз **for** и **where** при их наличии.

Выполнение запроса для значений параметров и выражений от них состоит в выделении объектов базового класса и связанных с ними объектов других классов, определяемых фразой **links** и удовлетворяющих ограничениям фраз **for** и **where** при их наличии, и последующем вычислении выражений фразы **select** для каждого объекта базового класса и связанных с ним объектов.

2.1.4. Ограничения фразы for

Ограничения значения параметра фразы **for** представляют собой конъюнкцию предикатов, каждый из которых зависит от параметров только одного класса и определяет ограничение на выделение объектов этого класса. Операция конъюнкции обозначается запятой, дизъюнкции – знаком "|", отрицания – знаком "!". Вво-

дятся стандартные операции отношения ($>$, $<$, $>=$, $<=$, $=$, $<>$) и возможность указывать не только конечные значения, но и их диапазон (через знак ":"). Например, запрос

$$\textbf{for } p_1 = v_1|v_2|v_3 : v_4|v_5,\ p_2 <> v_6\ \ \textbf{select } p_3$$

должен выбрать все значения параметра p_3 для объектов класса с этим параметром, у которых значение параметра p_2 не равно v_6, а значение свойства p_1 либо принадлежит множеству $\{v_1, v_2, v_5\}$, либо интервалу $[v_3, v_4]$. Отметим, что здесь используется сокращение в записи дизъюнкции $p_1 = v_1|p_1 = v_2|p_1 = v_3 : v_4|p_1 = v_5$.

Следует заметить, что ограничения во фразе **for** могут использовать также и круглые скобки для изменения порядка проверки условия. Например, запрос

$$\textbf{for } p_1 <> v_1, (p_1 > v_6|p_2 < v_2)\ \textbf{select } p_3$$

выберет значение параметра p_3 для объектов, у которых значение p_1 не равно v_1, и либо значение p_1 больше v_6, либо значение p_2 меньше v_2.

Каждый предикат фразы **for** может содержать выражения с операциями над параметрами класса (в зависимости от типа параметра). Например, запрос

$$\textbf{for } p_1 * p_2 > v_1 - v_2\ \textbf{select } p_3$$

выберет значение параметра p_3 для объектов, у которых произведение значений параметров p_1 и p_2 больше разности значений v_1 и v_2.

Ограничения фразы **for** любого запроса распространяются на все его подзапросы за исключением тех, у которых указано ключевое слово **all**. В этом случае фраза **for** подзапроса снимает ограничения внешнего запроса и может содержать свои ограничения. Например, запрос

$$\textbf{for } c_1.p_1 = v_1,\ c_2.p_3 > v_2,\ c_2.p_4 < (\textbf{for all select max}(c_2.p_4))$$
$$\textbf{select min}(c_1.p_2 + c_2.p_3)\ \textbf{from } c_1, c_2\ \textbf{links } c_1\ \textbf{parent } c_2$$

выберет минимум суммы значений параметров p_2 и p_3 объектов класса c_1 и его дочернего класса c_2, для которых соответственно $p_1 = v_1$ и $p_3 > v_2$, а также значение p_4 меньше максимума для всех объектов класса c_2 (если бы ключевое слово **all** в подзапросе было снято, то максимум p_4 искался бы только по части объектов класса c_2, для которых $p_3 > v_2$).

2.1.5. Ограничения фразы links

Фраза **links** указывает список ограничений на отношения между объектами классов (ограничения связей). Если объекты класса c_p являются родительскими объектами класса c_d, то во фразе **links** указывается ограничение "c_p **parent** c_d". Если объекты класса c_l включены в объекты класса c_h через объекты класса c_{hie} (отношение функционального включения), то во фразе **links** указывается ограничение "c_h **contains**(c_{hie}) c_l". Если имеет место простое включение (без класса c_{hie}), то скобки остаются пустыми. Вместо имен классов в ограничении отношения можно указывать подзапрос, выделяющий объекты этого класса.

Замечание 1. Допускается только одно ограничение связи между двумя классами. Поэтому в случае, когда необходимо использовать 2 или более связи включения (например, фразой **for** накладывается ограничение на 2 разных класса связи отношения включения двух классов, или накладывается ограничение на один из классов отношения включения, а отношений включения несколько) необходимо фразой **from** для каждого отношения включения установить выделение объектов подзапросом и соединить эти подзапросы операциями над множествами объектов (**union, intersection, complement**) в зависимости от потребности. Также следует поступить, если между 2 классами имеется 2 или более цепочек отношений: для каждой цепочки отношений нужно написать во фразе **from** свой подзапрос выделения объектов и объединить подзапросы операциями над множествами объектов.

В запросах для классов выбор последних может определяться по фразе **links**. Например, запрос

$$\textbf{select class } p \textbf{ from } c \textbf{ links } p \textbf{ parent } c$$

определяет классы, которые являются непосредственно родительскими для базового класса c.

Если в ограничение связи в конце ключевого слова добавить символ '*', то это приведет к указанию транзитивного замыкания отношения для данного типа связи. Например, запрос

$$\textbf{select class } p.* \textbf{ from } c \textbf{ links } c \textbf{ parent* } p,\ c \textbf{ contains * } p$$

определяет все свойства всех классов, для которых базовый класс c будет родительским классом, но не обязательно непосредственным родителем, или содержит класс, но не обязательно непосредственно.

Если необходимо выделить все классы, связанные каким-либо отношением с базовым классом, вместо указания связей фраза **links** содержит конструкцию **all**[(n)], где необязательный параметр n указывает максимальное расстояние в графе связей классов от вершины базового класса (если он отсутствует, то по умолчанию равен 1). Так, например, запрос

$$\textbf{select class } * \textbf{ from } c \textbf{ links all}$$

выберет все классы, которые для базового класса c являются либо непосредственно родительскими, либо непосредственно дочерними, либо включенными, либо включающими, либо непосредственными предшественниками, либо непосредственными последователями, либо находящиеся в отношении взаимодействия.

При необходимости ограничить выборку объектов существованием определенной связи с объектом другого класса в ограничение связи перед указанием другого класса добавляется ключевое слово **exist**. Например, запрос

$$\textbf{select object from } c,\ i \textbf{ links } c \textbf{ contains exist } i$$

выбирает только те объекты базового класса c, для которых в классе i существует включенный объект.

Добавление в ограничение связи ключевого слова **hierarchy** перед типом связи приводит к иерархическому запросу, для которого стартовый объект базового класса определяется фразой **for** (например, указанием значений идентификационных параметров базового класса), и далее с него выделяется дерево объектов, находящихся в указанном связью классов отношении. Например, запрос

$$\textbf{for } p1 = v1,\ p2 = v2 \textbf{ select object from } c \textbf{ links } c \textbf{ hierarchy contains } c$$

выделяет иерархическое дерево объектов класса c, начиная с объекта с идентификационными свойствами v_1, v_2, находящихся в отношении внутреннего включения.

Для работы с историей свойств, объектов и классов используется ключевое слово **history**. Для выбора объектов, существовавших в определенный период времени, во фразе **for** можно использовать параметры объектов "Дата рождения" и "Дата смерти" (если таковые были введены).

Ограничение связи взаимодействия указывается конструкцией

<Откуда> (<Что>)**interaction**(<Как>) <Куда>.

Например, запрос

$$\textbf{select class } w,\ h \textbf{ from } f,\ t \textbf{ links } f\ (w)\textbf{interaction}(h)\ t$$

для классов f, t, имеющих роли взаимодействия *Откуда* и *Куда*, определит классы пары классов $(w,\ h)$, находящихся в ролях взаимодействия *Что* и *Как*.

Замечание 2. Следует отметить, что допускается только одно отношение (или цепочка отношений) между двумя классами. Поэтому, как и в замечании 1 (к ограничениям на отношение включения), в необходимых случаях (например 2 класса входят в отношение включения и в отношение взаимодействия) следует эту ситуацию определить при помощи подзапросов на множества объектов того или иного класса и операций над такими множествами. Назовем такой подход *ограничением однозначности отношений*.

2.1.6. Ограничения фразы where

Ограничения фразы **where** отличаются от ограничений фразы **for** тем, что предикат каждого из них может содержать выражение с параметрами разных классов. Предикаты ограничений связываются операциями конъюнкции, дизъюнкции и отрицания, нотация которых такая же, как и для фразы **for** (запятая, вертикальная черта и восклицательный знак). 2 подзапроса выделения объектов одного класса могут соединяться отношением включения **include**, которое истинно, если каждый объект второго подзапроса принадлежит также и первому подзапросу. Например, запрос

select $c.p1$ **from** c **where** (**select** c **from** j **links** c **contains exist** j)
include (**select** c **from** i **links** c **contains exist** i)

выбирает значение параметра p_1 класса c, для которого каждый объект, содержащий объект класса i, включает также объект класса j.

Во фразе **where** можно использовать конструкцию отношения **in** принадлежности объекта (класса a) множеству объектов:

$$a.obj \textbf{ in } (\textbf{select} \dots)$$

Если для вышеописанного примера написать запрос
select $c.p1$ **from** c, i **links** c **contains exist** i **where** $c.obj$ **in**
(**select** c **from** c, j **links** c **contains exist** j),
то будут выбраны значения параметра $c.p1$ только для объектов класса c, содержащих объекты как класса i, так и класса j. Но наиболее рационально в этом случае использовать соединение подзапросов в фразе **links**, исключив совсем фразу **where**:
select $c.p1$ **from** (**select object from** c, i **links** c **contains exist** i)
intersection (**select object from** c, j **links** c **contains exist** j)
по той причине, что при выполнении запроса сразу будут выделены необходимые объекты, а в предыдущем решении для большего множества выделенных объектов будет происходить еще проверка правильности выделения объекта.

2.1.7. Агрегатные функции

Общий вид конструкции с агрегатной функцией следующий:

$$f(< expr >)\ [\mathbf{on} < class.p_1 >, \ldots, < class.p_n >] \mid [\mathbf{on} < class_1 >, \ldots, < class_n >],$$

где f – имя агрегатной функции, <expr> – логическое выражение параметров класса или классов для функции **count** или арифметическое выражение – для остальных функций, **on** – условие группирования значений агрегатной функции, $< class.p_i >$, $< class_i >$ $(i \in \overline{1,n})$ – параметры одного класса объектов группирования или классы объектов группирования, каждый набор которых определяет группу, для которой вычисляется значение агрегатной функции. Таким образом, агрегатная функция вычисляется для объектов одного класса с условием группирования, определяемым порядком изменения параметров справа налево, либо – для объектов разных классов с условием группирования, определяемым порядком изменения объектов в классах справа налево.

При отсутствии условия группирования значение функции вычисляется

- для всех объектов базового класса, если она содержится фразой **for**;
- для всех объектов базового класса, удовлетворяющих ограничениям фраз **for** и **links**, если она содержится фразой **where**;
- для всех объектов базового класса, удовлетворяющих всем ограничениям фраз **for**, **links** и **where**, если она содержится фразой **select**.

Если есть условие группирования, то значение функции вычисляется для каждой группы с аналогичными ограничениями. Группы изменяются в порядке изменения объектов списка группирования справа налево.

Нижеперечисленные функции используются без условия группирования:

- **count** – вычисляет число объектов, для которых аргумент функции принимает значение **true**;
- **max** – вычисляет максимум значений аргумента функции;
- **min** – вычисляет минимум значений аргумента функции;

- **sum** – вычисляет сумму значений аргумента функции;
- **avrg** – вычисляет среднее значений аргумента функции;
- **std** – вычисляет стандартное отклонение значений аргумента функции.

Следующие функции вычисляются только при условии группирования:

- **maxcount** – вычисляет максимальное число объектов в группах;
- **maxsum** – вычисляет максимальную сумму значений в группах;
- **maxavrg** – вычисляет максимум средних значений в группах;
- **maxstd** – вычисляет максимум стандартных отклонений значений в группах;
- **maxmin** – вычисляет максимум минимальных значений в группах;
- **avrgcount** – вычисляет среднее число объектов в группах;
- **avrgsum** – вычисляет среднюю сумму значений в группах;
- **avrgavrg** – вычисляет среднее от средних значений в группах;
- **avrgstd** – вычисляет среднее от стандартных отклонений значений в группах;
- **stdcount** – вычисляет стандартное отклонение чисел объектов в группах;
- **stdsum** – вычисляет стандартное отклонение сумм значений в группах;
- **stdavrg** – вычисляет стандартное отклонение от средних значений в группах;
- **stdstd** – вычисляет стандартное отклонение от стандартных отклонений значений в группах.

Аналогичны функции, вычисляющие минимальное значение в группах: **mincount, minsum, minavrg, minstd, minmax**.

Нижеперечисленные функции без группирования выделяют объекты базового класса и используются только фразой **select** запроса (подзапроса):

- **objmax** – выделяет объекты с максимальным значением;
- **objmin** – выделяет объекты с минимальным значением.

Следующие функции выделения объектов связаны с одним классом группирования и аргументом, зависящим от параметров объектов других классов, связанных с классом группирования. Для каждого объекта класса группирования по выражению аргумента функции выделяется группа всех возможных наборов связанных с ним (с объектом класса группирования) объектов. Среди разных объектов класса группирования выделяются объекты, для групп которых выполняется условие по сравнению с другими объектами класса группирования. Например, функция

objmaxsum(<аргумент>) **on** d

выделит объекты класса d, для которых сумма значений аргумента в группе максимальна.

- **objmaxcount** – выделяет объекты класса группирования, для каждого из которых число наборов в группе, определяющих значение **true**, максимально;
- **objmaxsum** – выделяет объекты класса группирования, для каждого из которых сумма значений аргумента по всем наборам группы максимальна;
- **objmaxavrg** – выделяет объекты класса группирования, для каждого из которых среднее значение аргумента по всем наборам группы максимально;
- **objmaxstd** – выделяет объекты класса группирования, для каждого из которых стандартное отклонение значений аргумента по всем наборам группы максимально;
- **objmaxmin** – выделяет объекты класса группирования, для каждого из которых минимальное по всем наборам группы значение аргумента максимально;
- **objmincount** – выделяет объекты класса группирования, для каждого из которых число наборов в группе, определяющих значение **true**, минимально;
- **objminsum** – выделяет объекты класса группирования, для каждого из которых сумма значений аргумента по всем наборам группы минимальна;
- **objminavrg** – выделяет объекты класса группирования, для каждого из которых среднее значение аргумента по всем наборам группы минимально;
- **objminstd** – выделяет объекты класса группирования, для каждого из которых стандартное отклонение значений аргумента по всем наборам группы минимально;
- **objminmax** – выделяет объекты класса группирования, для каждого из которых максимальное по всем наборам группы значение аргумента минимально.

Описанный нижний уровень языка назовем так же, как и сам язык, – ODQL.

2.2. Верхний уровень ODQL

Верхний уровень отличается следующими особенностями:

1. Отсутствует фраза **links** (для поиска связи всех необходимых для выполнения запроса классов используется метауровень).
2. Фраза **from** содержит либо один класс (или вовсе не указывается, что означает выбор класса первого свойства), либо содержит имена нескольких связанных классов.
3. Если во фразах **select** или **for** указаны свойства, не являющиеся параметрами базового класса, фраза **from** неявно дополняется именами классов для таких свойств.

4. Имена свойств не предваряются именем класса (или его алиаса), которому они соответствуют, если во фразе **from** указан один класс либо эта фраза отсутствует.

Итак, на верхнем уровне ODQL мы лишь указываем значения свойств, которые хотим получить, ограничения на объекты и, возможно, класс объектов (или связанные классы).

Обозначим верхний уровень SODQL – Simplified ODQL.

Заметим, что каждому SODQL-запросу можно поставить в соответствие не менее одного ODQL-запроса, тогда как каждый ODQL-запрос соответствует только одному SODQL запросу. Рассмотрим пример: пусть класс c_h включает класс c_l через классы связи c_{hie1} и c_{hie2}, pr – параметр класса c_l. Тогда SODQL-запрос

$$\textbf{select } pr \textbf{ from } c_h$$

выделит значения параметра pr всех объектов класса c_l, которые включены в некоторый объект класса c_h *хотя бы через один* объект классов c_{hie1} и c_{hie2}. Ему соответствует ODQL-запрос

$$\textbf{select } c_2.pr \textbf{ from } c_h\ a_1, c_l\ a_2 \textbf{ where } a_1 \textbf{ contains } a_2$$

Но, рассмотрев ODQL-запрос

$$\textbf{select } a_2.pr \textbf{ from } c_h\ a_1, c_l\ a_2, c_{hie}\ a_3 \textbf{ where } a_1 \textbf{ contains } (a_3)\ a_2,$$

заметим, что он выделит множество объектов, отличное от множества, выделяемого предыдущим запросом.

Для манипулирования сущностями DIM введены следующие конструкции:

1. Запрос

   for <параметр> = <значение>,...

   create object

   [**from** <имя класса>]

   [**parent** (<запрос1>)]

   [**contains** ([<запрос2>]) (<запрос3>)]

   [(<запрос4>)**contains** ([<запрос5>])]

   создает объект класса <имя класса>, инициализируя его параметры значениями, указанными во фразе **for**. Если у объекта необходимо указать объект-родитель, используется модификатор **parent**, после чего следует в круглых скобках запрос, выделяющий этот объект по его идентификационным параметрам. Если родительских объектов несколько (множественное наследование), то модификатор **parent** используется соответствующее число раз. При необходимости можно сразу указать перечень объектов, включенных в данный объект. Если включение является функциональным, то первый запрос в круглых скобках после модификатора **contains** (в данном случае – <запрос2>)

выделяет объект связи (или создает его, используя вложенный подзапрос), а следующий за ним запрос выделит объекты, подлежащие включению. Аналогично предыдущей конструкции можно указать перечень объектов, в которых включен данный объект. При этом выделение объекта, включающего данный, идет с помощью <запроса4>, а указание объекта связи функционального включения делается с помощью <запроса5>. Вот пример такого запроса:

for Имя единицы = 'Тестовый объект 1', Коэффициент = 19243.12

create object

from Единицы измерения

2. Запрос

   create class <имя класса>

   parameters (<Имя 1>,...)

   [**parent** <имя класса1>]

   [**contains** [(<имя класса2>)] <имя класса3>]

   [<имя класса4>**contains** [(<имя класса5>)]]

   создаст класс с именем <имя класса>, параметрами, указанными в модификаторе **parameters**, родителями, определяемыми модификатором **parent** и включенными и включающими классами, определяемыми модификатором **contains**. Вместо данных имен классов родителей, включенных, включающих и классов связи включения могут использоваться запросы, определяющие эти данные.

3. Запрос

   create parameter <имя параметра> <тип параметра> <тип данных>

   создаст атрибут с именем <имя параметра> (если такового еще нет) с типом данных <тип данных> и типом параметра (временн*ой* или нет), указанном в данном <тип параметра>.

4. Запрос

   create link <тип связи>

   from <запрос1>

   [**through** <запрос2>]

   to <запрос3>

   создаст связь указанного *типа* (для связи наследования указывается тип связи **inheritance**, для связи включения – **inclusion**) от сущностей, определяемых фразой **from** к сущностям, определяемым фразой **to**. Для связи включения может использоваться фраза **through** для выделения сущности связи (класса или объекта), через которую необходимо создать связь включения.

Типы сущностей фраз (объект или класс) должны совпадать: нельзя устанавливать связь между объектами и классами. Для создания связи между классами вместо запросов могут использоваться имена классов.

Следует отметить, что лишь один запрос фраз **from** или **to** может выделять множество, состоящее из более чем одной сущности. Если результат запросов обеих фраз будет содержать 2 и более сущности, то не ясно, между какими именно сущностями устанавливается связь.

5. Запрос

   for <параметр> = <значение>, ...

   update object (<запрос>)

   изменяет значение свойств, указанных фразой **for** для объектов, выделенных запросом фразы **update object**. Следует заметить, что если свойство является идентификационным для класса объекта, то изменение значения этого свойства приведет к окончанию "жизни" текущего объекта, созданию объекта-последователя и созданию связи между ними в отношении истории.

   Для корректировки объектов вместо фразы **update** можно использовать фразу **correct**. В этом случае внесение изменений будет произведено без участия истории объектов.

6. Запрос

   update class (<запрос>)

   [**add parameters** (<Имя 1>,...)]

   [**remove parameters** (<Имя 1>,...)]

   Изменяет класс, выделяемый фразой **update class**, добавляя или удаляя параметры (связи модифицируются запросами **create link** и **delete link**). Вместо (<запроса>) можно явно указать имя класса. Добавляемые свойства не должны быть параметрами другого класса. Любое изменение класса влечет изменение всех объектов и создание объектов-предшественников.

   Для корректировки класса вместо фразы **update** можно использовать фразу **correct**, если нет необходимости хранить историю.

7. Запрос

   for datatype = <имя типа>, **name** = '<имя параметра>',
   type = <тип параметра>

   update parameter (<запрос>)

   изменяет *тип данных*, *имя* или *тип параметра* на указанные во фразе **for** для атрибутов, выделенных подзапросом <запрос> (этот подзапрос обязательно должен содержать фразу **select property**). Вместо подзапроса можно указать имя параметра (старое). Тип параметра может быть одного из следующих

видов: **additional** (дополнительный тип атрибута), **identic** (идентификационный тип атрибута), **nonidentic** (неидентификационный тип атрибута). Дополнительный тип атрибута означает, что он не является обязательным для всех объектов класса.

Например, запрос

for name = 'новое имя'

update parameter (**select property** <старое имя> **from** <класс объектов>)

выберет свойство 'старое имя' из класса 'класс объектов' и изменит его имя на 'новое имя', если это возможно.

Следует заметить причину использования для свойств различных ключевых слов: **parameter** для **update** и модификатор **property** для **select**. Это сделано намеренно для подчеркивания различия этих фраз: **parameter** – это *часть фразы* **update**, тогда как **property** – это *модификатор*, использующийся как опциональная часть фразы **select**.

Для корректировки параметра вместо фразы **update** можно использовать фразу **correct**, если нет необходимости сохранения истории параметра.

8. Запрос

   delete [hierarchy] object (<запрос>)

   удаляет, если это возможно, выбранное запросом <запрос> множество объектов. Если используется модификатор **hierarchy**, вместе с объектом удаляются и связанные с ним объекты, если они не связаны с другими объектами (неудаляемыми).

9. Запрос

   delete [hierarchy] class <имя класса>|(<запрос>)

   удаляет, если это возможно, выбранное запросом <запрос> множество классов. Вместо запроса можно указать имя класса. Если используется модификатор **hierarchy**, вместе с классом удаляются и связанные с ним классы, если они не связаны с другими классами (неудаляемыми).

10. Запрос

    delete parameter <имя свойства>|(<запрос>)

    удаляет, если это возможно, выбранное запросом <запрос> множество свойств. Вместо запроса можно указать имя свойства.

11. Запрос

    delete link <тип связи>

    from (<запрос>)

    [**through** (<запрос>)]

    to (<запрос>)

удаляет связь указанного *типа* от сущностей, определяемых фразой **from**, к сущностям, определяемым фразой **to**. Для связи включения может использоваться фраза **trough** для выделения сущности связи (класса или объекта). Сущности каждой фразы должны иметь один тип.

Отметим, что при реализации SODQL-запрос рационально предварительно транслировать в ODQL-запрос. При желании выделить в SODQL-запросе объекты более чем одного класса, достаточно во фразе **from** указать помимо базового класса остальные классы через запятую.

2.3. Пример

Приведем пример соответствия запросов (пример 1).

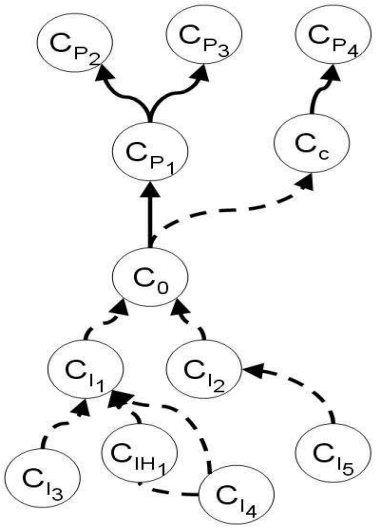

Рис. 1.

На рис. 1 показан класс c_0, для которого определены:

1) родительский класс c_{p_1}

2) включенные классы c_{i_1} и c_{i_2}

3) содержащий класс c_c (класс c_c включает в себя класс c_0).

4) для класса c_{p_1} определены два родительских класса c_{p_2} и c_{p_3}

5) для класса c_c определен родительский класс c_{p_4}

6) в класс c_{i_1} включены классы c_{i_3} и c_{i_4} простым включением. Дополнительно класс c_{i_4} включен функционально через класс связи c_{ih_1}.

7) в класс c_{i_2} включен класс c_{i_5}

Пусть каждый из указанных классов содержит соответствующие параметры a_0, a_{p_i} $(i = 1, \ldots, 4)$, a_{c_j} $(j = 1, \ldots, 5)$, a_c и a_{ih_1}. Приведем пример запроса на языке SODQL:

for $a_{p_2} = 2$, $a_{p_4} = 4$, $a_{i_5} = 5$, a_{ih_1} = 'строка'

select a_0, a_{i_4}, a_{p_3}, a_c

from c_0

Этот SODQL-запрос должен выделить значения свойств a_0, a_{p_3}, a_c объектов класса c_0, а также значения свойства a_{i_4} объектов, включенных в объекты класса c_{i_1}, которые в свою очередь включены в объекты класса c_0. Каждый из выбираемых объектов класса c_0 должен:

1) иметь значение 2 для свойства a_{p_2};

2) включаться в объект класса c_c, у которого значение свойства a_{p_4} равно 4;

3) включать объект класса c_{i_2}, который в свою очередь включает объект класса c_{i_5} со значением свойства a_{i_5}, равным 5;

4) включать объект класса c_{i_1}, который содержит объект класса c_{i_4} через объект класса связи c_{hi_1} со значением свойства a_{hi_1}, равным 'строка'.

Если проследить связи каждого участвующего класса, запрос можно уточнить: запрос должен выделить значения свойства a_0 объектов класса c_0 и свойства a_{p_3} родительского класса c_{p_3} через объекты родительского класса c_{p_1}. Также для таких объектов класса c_0 необходимо выделить содержащие их объекты класса c_c и из этих объектов получить значение свойства a_c.

Как отмечалось ранее (при введении SODQL), при указании свойств, не являющихся параметрами базового класса, фраза **from** дополняется именами классов таких свойств. Благодаря этому мы выделим объекты указанных классов и свяжем их с объектами базового класса. Теперь можно построить ODQL-запрос, соответствующий данному SODQL-запросу:

for $c_{p_2}.a_{p_2} = 2,\ c_{p_4}.a_{p_4} = 4,\ c_{i_5}.a_{i_5} = 5,\ c_{ih_1}.a_{ih_1} =$ 'строка'
select $c_0.a_0,\ c_{i_4}.a_{i_4},\ c_{p_3}.a_{p_3},\ c_c.a_c$
from $c_0,\ c_{p_1},\ c_{p_2},\ c_{p_3},\ c_c,\ c_{p_4},\ c_{i_1},\ c_{i_2},\ c_{i_4},\ c_{ih_1},\ c_{i_5}$
links c_{p_1} **parent** $c_0,\ \ c_{p_2}$ **parent** $c_{p_1},\ \ c_{p_3}$ **parent** $c_{p_1},\ \ c_c$ **contains** $c_0,$
c_{p_4} **parent** $c_c,\ \ c_0$ **contains** $c_{i_1},\ \ c_0$ **contains** $c_{i_2},\ \ c_{i_1}$ **contains**(c_{ih_1}) $c_{i_4},$
c_{i_2} **contains** c_{i_5}

Заметим, что так как ограничение **for** накладывает ограничение на класс связи c_{ih_1} и не накладывает ограничений на классы c_{i_1}, c_{i_4}, связанные двумя отношениями включения, то мы отношение простого включения не используем, что дает нам выполнение ограничения однозначности отношений.

3. Заключение

В заключение отметим, что поставленные задачи построения адекватного системе DIM объектного языка запросов, отвечающего требованиям простоты использования для манипулирования любыми данными, выполнены.

Список литературы

1. Писаренко Д.С., Рублев В.С. Объектная СУБД Динамическая информационная модель DIM и ее основные концепции // Моделирование и анализ информационных систем. 2009. Т. 16, № 1. С. 62–91.

2. Писаренко Д.С., Рублев В.С. Концепции взаимодействия Динамической информационной модели DIM // Актуальные проблемы математики и информатики. Ярославль: ЯрГУ, 2007. С. 75 – 80.

3. Писаренко Д.С., Рублев В.С. Синтез аппарата взаимодействий системы управления базами данных DIM // Материалы XVII международной школы-семинара «Синтез и сложность управляющих систем» имени академика О.Б. Лупанова. Новосибирск: ИМ СО РАН, 2008. С. 130 – 135.

4. Cattell R.G.G., Barry D.K. and other. The ObjectData standart: ODMG 3.0. San Francisco: Morgan Kaufman Publishers, 1999.

The Object Query Language of the Dynamic Information Model DIM

Roublev V.S.

Keywords: object DBMS, object query language

The article is devoted to the creation of a data manipulation language for a new object DBMS, which should allow us to describe a simple data query of this model.

Сведения об авторе:
Рублев Вадим Сергеевич,
Ярославский государственный университет им. П.Г. Демидова,
профессор кафедры теоретической информатики