УДК 519.85

В.С. Рублев

Запросная полнота языка ODQL динамической информационной модели DIM

Обосновывается запросная полнота объектного языка ODQL [1] манипулирования данными для новой объектной технологии СУБД DIM [2]–[4]. Строится алгоритм преобразования верхнего уровня SODQL [1] в нижний уровень ODQL, что делает возможным автоматическое преобразование простого описания запроса SODQL в полное его описание на ODQL.

Ключевые слова: объектная СУБД, язык запросов, запросная полнота.

V.S. Rublev

Completeness of the ODQL Query Language of the DIM Dynamic Information Model

The paper is devoted to a justification of completeness for ODQL query language [1] for new object-oriented DBMS technology DIM [2]–[4]. Is constructed the algorithm of transforming a top level SODQL [1] into a lower level ODQL that makes possible an automatic transformation of SODQL query into its full description within ODQL.

Key words: object DBMS, a query language, query language completeness.

1. Постановка задачи

Описанная в [1]–[3] СУБД динамическая информационная модель DIM основывается на совокупностях объектов O, свойств объектов A и классов объектов C, а также базовых отношениях *наследования, включения, взаимодействия и истории*, введенных как для объектов, так и для их классов. Эта система является объектной СУБД, но отличается от известных технологий (объектно-реляционной [2] и объектно-ориентированной [5]) направленностью на адекватное описание любых дискретных детерминированных моделей (см. в [2]), которое адаптируется к изменениям во времени не только объектов, но и их типов.

Для манипулирования данными (объектами, классами, свойствами) в [4] введен язык объектных запросов ODQL, нижний уровень которого позволяет полно отразить всю сложность отношений объектов и классов в DIM, а верхний уровень SODQL этого языка позволяет достаточно просто описывать сложные запросы к данным этой системы.

Нижний уровень языка ODQL содержит подобно языку SQL для реляционных СУБД фразы **select** и **from**, описывающие список выбора и классы объектов для выбора. Но выбираться могут не только свойства объектов (атрибуты данных), но и другие сущности системы (объекты или классы объектов). Фраза **where** ограничений выбора в SQL разделена на 3 фразы:

1) **for**, описывающая ограничения на объекты отдельно для каждого класса;
2) **links**, описывающая ограничения на связи (отношения) классов объектов;
3) **where**, описывающая ограничения на связанные между собой объекты разных классов.

Такая конструкция запроса может описать сложные отношения объектов и классов DIM.

Запросы верхнего уровня SODQL языка могут содержать только фразу **select** без указания других фраз; они не содержат подзапросов, указания классов для выбираемых свойств, а также фразы **links**. Вводятся конструкции **create**, **update**, **delete** для создания сущностей (классов, объектов, параметров, связей классов и объектов) системы, их изменения и удаления.

Вместе с тем, в [4] не затронуты следующие задачи:

1) является ли язык ODQL запросно-полным в том смысле, что для любой совокупности связанных между собой данных системы может быть построен запрос манипуляции с этими данными;
2) как построить преобразование запроса на верхнем уровне SODQL в нижний уровень ODQL.

Решению этих задач посвящена данная работа.

2. Запросная полнота языка ODQL

Запросную полноту языка ODQL выражают следующие 3 утверждения:

Теорема 1. *Для любого класса DIM, связанного с ним множества классов, связанного множества объектов этих классов и их свойств существует запрос на языке ODQL, который выделяет заданное множество классов, или множество объектов, или множество свойств.*

Теорема 2. *Для любых множеств классов DIM, связных объектов этих классов, идентификационных свойств этих объектов и любого допустимого изменения этих свойств существует последовательность запросов на языке ODQL, которая определяет такое изменение свойств, порождая новые объекты-последователи и связывая их со старыми объектами-предшественниками отношениями истории объектов.*

Теорема 3. *Для любого изменения классов DIM и связанных с ними допустимых изменений объектов существует последовательность запросов ODQL, приводящая к этим изменениям.*

Заметим, что доказательство теоремы 1 достаточно провести для случая выделения объектов. Для выделения свойств достаточно будет выделить объекты, обладающие нужными свойствами (под выделением свойств подразумевается выделение сущностей, а не получение значений свойств объектов), а для них получить сами свойства модификатором **property**. Для выделения классов можно действовать аналогичным образом с модификатором **class**.

Прежде чем приводить доказательство теорем, введем необходимые обозначения и построения.

Пусть C – множество всех классов *объектной* модели DIM, O – множество всех *объектов* этой модели, A – множество свойств всех ее объектов.

Строчными буквами мы будем обозначать объекты или классы (например, o – объект, c – класс), а прописными буквами – множества объектов или классов (например, O – множество объектов, C – множество классов).

Рассмотрим гипероргграф $\vec{G}(\mathrm{O}, \vec{A})$, вершинами которого являются объекты, а дугами (гипердугами) – отношения либо наследования (ориентированные от родительского объекта к дочернему), либо включения (ориентированные от включенного к включающему при простом включении или дополнительно через объект класса связи при его наличии), либо истории (ориентированные от предшественника к последователю), либо взаимодействия, ориентированные от объектов *Откуда, Куда, Что* к объекту *Как*. Каждую вершину $o \in \mathrm{O}$ пометим идентификатором соответствующего объекта.

Каждую дугу, соответствующую отношению наследования, пометим символом P. Каждую дугу, соответствующую отношению включения, пометим символом C и множеством идентификаторов объектов связей включения. При этом если одна из связей отвечает простому включению (без класса связи), то соответствующий идентификатор полагается равным 0 (для отличия этой ситуации от случая, когда отношение с классом связи имеет место). Каждую дугу, соответствующую отношению истории, пометим символом H. Каждую гипердугу, соответствующую отношению взаимодействия, пометим символом I.

Заметим, что гипероргграф $\vec{G}(\mathrm{O}, \vec{A})$ не содержит циклов в силу ограничения определенности системы DIM (см. [2]).

Пусть граф G(O ,A) – неориентированный гиперграф, полученный из гипероргграфа $\vec{G}(\mathrm{O}, \vec{A})$ потерей ориентации всех дуг. Назовем его гиперграфом связи объектов.

Теперь введем аналогичные построения для классов.

Рассмотрим гипероргграф $\vec{F}(\mathrm{C}, \vec{B})$, вершинами которого являются классы, а дугами (гипердугами) – отношения либо наследования (ориентированные от родительского класса к дочернему), либо включения (ориентированные от включенного к включающему при простом включении или дополнительно через класс связи при его наличии), либо взаимодействия, ориентированные от классов *Откуда, Куда, Что* к классу *Как*. Каждую вершину $c \in \mathrm{C}$ пометим идентификатором соответствующего класса.

Каждую дугу, соответствующую отношению наследования, пометим символом P. Каждую дугу, соответствующую отношению включения, пометим символом H и множеством идентификаторов

классов связей включения. При этом если одна из связей отвечает простому отношению включения (*простое включение*), то соответствующий идентификатор полагается равным 0.

Заметим, что орграф $\vec{F}(\mathrm{C},\vec{B})$ не содержит гиперорциклов в силу ограничения определенности системы DIM (см. [2]), не считая дуг вида $b=\{c,c\}$ (то есть дуг, входящих в один и тот же класс и выходящих из него, которые соответствуют отношениям *внутреннего включения* и *внутреннего наследования*).

Пусть гиперграф $F(\mathrm{C},B)$ – неориентированный гиперграф, полученный из гиперорграфа $\vec{F}(\mathrm{C},\vec{B})$ потерей ориентации всех дуг. Назовем его гиперграфом связи классов.

Для целей доказательства теорем нам не потребуются отношения взаимодействия и истории. Поэтому опустим их в гиперграфах связи объектов и классов. В результате получим графы связи объектов и классов.

Теперь введем понятия *опорного класса*, *множества накладываемых ограничений*, *множества выбираемых свойств* и *слоя*.

Нас далее будет интересовать не произвольный набор объектов, а объекты *класса*, определенного произвольным образом (а также связанные с ними объекты). Назовем такой класс *опорным классом* и будем обозначать c_0. Чаще всего этот класс определен пользователем. Если же такой класс не задан, выберем его произвольно. Например, выберем в качестве опорного класса c_0 класс первого выбираемого свойства.

С объектами базового класса могут быть связаны объекты других классов. Если нам, кроме объектов базового класса, необходимо выделить некоторые объекты, связанные с объектами базового, то множество таких классов мы обозначим C_{frm}.

На каждый из объектов системы DIM мы можем наложить *ограничения* – перечень условий, которым должен удовлетворять каждый выбираемый объект. Это могут быть ограничения на свойства объекта, на связи некоторых классов объектов, на наличие связи с некоторыми другими объектами или совокупность этих ограничений (сущность объекта определяется этими понятиями). Выражение, которым мы можем задать ограничения, может быть достаточно сложным. Однако, поскольку ограничение представляет собой некоторую логическую функцию, мы всегда можем построить для нее ДНФ (дизъюнктивно-нормальную форму)

$$con_1 OR \ldots OR con_\delta,$$

где каждое логическое слагаемое con_q $(q\in\overline{1,\delta})$ дизъюнкции представляет собой конъюнкцию ограничений

$$rs_1\ AND\ \ldots\ AND\ rs_\alpha\ AND\ lnk_1\ AND\ \ldots\ AND\ lnk_\beta\ AND\ cn_1\ AND\ \ldots AND\ cn_\gamma$$

Здесь

- rs_i $(i\in\overline{1,\alpha})$ – ограничения на свойства объектов;
- lnk_j $(j\in\overline{1,\beta})$ – ограничения на связь с множеством объектов;
- cn_k $(k\in\overline{1,\gamma})$ – ограничения на связь классов (например, указания выбора объектов через определенный класс включения).

Каждое свойство ограничения rs_i может быть свойством как самого класса c_0, так и связанных с ним классов.

Далее каждому объекту $o_i\in\mathrm{O}$ сопоставим множество его параметров $A_i\subseteq\mathrm{A}$ (поскольку параметрами обладают классы, а не объекты, необходимо отдельно указать сопоставление через классы). Элементы множества A_i – это параметры класса объекта, соответствующего вершине o_i в графах $\vec{G}$ и G.

Поскольку у выбираемых объектов нас могут интересовать не все свойства, а лишь некоторые, то введем понятие *выбираемых свойств объекта*: множество $A'_i = \{a'^1_i, ..., a'^{m_i}_i\} \subseteq A_i$ – множество всех выбираемых свойств каждого объекта $o_i \in \mathrm{O}$ (естественно, для объектов одного класса эти множества совпадают). Множество $A' = \bigcup_i A'_i$ назовем *множеством выбираемых свойств*. Если $A' = \varnothing$, то ставится задача выделения множества объектов, в противном случае – задача получения значений свойств выделяемых объектов.

Теперь введем понятие слоя.

1. *Слоем 0-го уровня* (*нулевым слоем*) назовем множество классов *X*, состоящее из одного элемента – класса c_0.

2. Теперь для множества *X* рассмотрим все соседние (смежные) вершины в графе $\vec{F}(\mathrm{C}, \vec{B})$. В нашем случае это будут классы, являющиеся непосредственными родителями, наследниками, включенными и содержащими класс c_0 (классы связи, через которые происходит включение, также принадлежат этому слою). Такое множество классов мы будем называть *слоем первого уровня* (*первым слоем*).

3. Каждый следующий *слой i-го уровня* состоит из классов, не вошедших в предыдущие слои и являющихся соседними для классов (i-1) слоя в графе $\vec{F}(\mathrm{C}, \vec{B})$.

Заметим, что слоев конечное число (так как число классов конечно). Общий вид первых двух слоев изображен на рис. 1.

На рисунке указан класс c_0 (в центре). Он является единственным элементом слоя нулевого уровня. Элементами первого слоя являются следующие классы:

1) родительские классы $c_{p^1_l}$ ($l=\overline{1, n_1}$);

2) дочерние классы – наследники $c_{d^1_j}$ ($j=\overline{1, m_1}$);

3) содержащие класс c_0 простым включением классы $c_{c^1_k}$ ($k=\overline{1, r_1}$);

4) включенные в класс c_0 простым включением классы $c_{c^1_i}$ ($i=\overline{1, q_1}$);

5) содержащие класс c_0 функциональным включением классы $c_{ch^1_g}$ ($g=\overline{1, t_1}$) через соответствующие классы связи $c^0_{ch^1_g}$;

6) включенные в класс c_0 функциональным включением классы $c_{ih^1_w}$ ($w=\overline{1, u_1}$) через соответствующие классы связи $c^0_{ih^1_w}$.

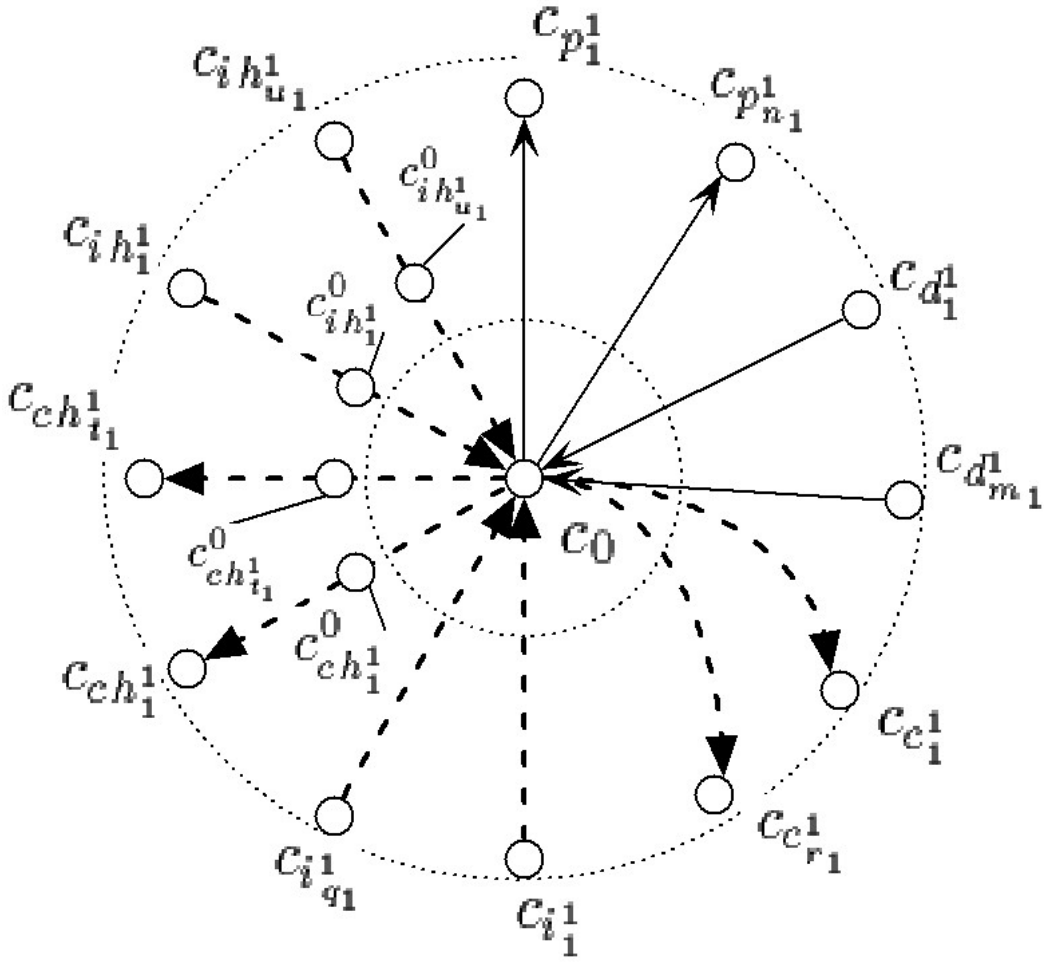

Рис. 1.

Теперь по имеющимся опорному классу c_0, выбираемым свойствам A' и множествам ограничений $con_1,...,con_\delta$ необходимо построить ODQL-запрос, выделяющий заданные свойства (или объекты, если A' пусто).

Доказательство теоремы 1.

Приведем алгоритм построения запроса для произвольного элемента ограничений con_q.

Будем по очереди рассматривать слои от первого до максимального уровня. Обозначим через I номер текущего слоя и через Add – множество классов, участвующих в ограничении на связь классов, то есть $Add = \{c_{ch_1^I},...,c_{ch_{i_I}^I},c_{ih_1^I},...,c_{ih_{u_I}^I}\}$.

1. Для каждого класса слоя I ищем ограничения rs_i, которые содержат только параметры этого класса (в первую очередь рассматриваем ограничения на параметры классов связей, а затем – обычных классов, не являющихся классами связи (этот порядок оптимизирует вычисления). Если такие ограничения найдены, то:

1) если ограничение не содержит агрегатную функцию, добавляем такое ограничение с именем класса перед каждым параметром во фразу **for** через операцию конъюнкции (запятую);

2) если ограничение содержит агрегатную функцию для всех объектов класса, добавляем такое ограничение с именем класса перед каждым параметром в фразу **for** через операцию конъюнкции (запятую);

3) удаляем эти ограничения из множества ограничений на свойства rs_i;

4) класс добавляем в фразу **from**, если его в ней еще нет;

5) проходим от слоя I до слоя 0 по дугам графа $F(\mathrm{C},\vec{B})$ от вершины, являющейся рассматриваемой.

мым классом, до вершины класса c_0; при этом добавляем новые классы во фразу **from** и, возможно, новые ограничения на связь классов во фразу **links**.

2. Оставшиеся ограничения rs_i, если они есть, содержат параметры разных классов. Добавляем каждое такое ограничение во фразу **where**, а также классы во фразу **from**, если они там отсутствуют, и связи добавленных классов во фразу **links**.

3. Подобные шагам 1 и 2 операции проводим для выбираемых свойств множества A', добавляя каждое выражение во фразу **select** и корректируя при необходимости фразы **from** и **links**.

4. Если $Add \cap X \neq \varnothing$ (в текущем слое есть классы, участвующие в ограничении на связь классов), то

1) для каждого класса, принадлежащего пересечению, находим ограничения cn_k, в которых он участвует;

2) для каждого ограничения добавляем его классы во фразу **from**, если они еще там не присутствуют, и добавляем текущее ограничение во фразу **where**;

3) от найденного класса идем по слоям от I до 0, используя дуги графа F, вставляя встреченные классы во фразу **from**, если их там еще нет, и добавляя соответствующие дуге ограничения во фразу **where**, если это ограничение еще не указано;

4) найденный класс удаляем из множества Add.

Заметим, что каждый класс принадлежит лишь одному слою. Поэтому каждый класс множества Add будет принадлежать не более чем одному слою.

5. Если $C_{frm} \cap X \neq \varnothing$ (в текущем слое есть классы, с объектами которых необходимо связать объекты базового класса), то

1) от каждого класса, принадлежащего пересечению, идем по слоям от I до 0, используя дуги графа $\vec{F}$, вставляя встреченные классы во фразу **from**, если их там еще нет, и добавляя соответствующие дуге ограничения во фразу **where**, если это ограничение еще не указано;

2) найденный класс удаляем из множества C_{frm}.

6. Если множество ограничений rs_i, или множество выбираемых свойств A', или множество классов, участвующих в ограничении на связь, или множество классов C_{frm}, объекты которых необходимо связать с объектами базового класса, **не пусто**, то увеличиваем I на 1, множество X заменяем на вершины следующего слоя и переходим к пункту 1.

7. Если множество выделяемых свойств пусто, то добавляем во фразу **select** модификатор **object**.

В результате мы получим ODQL запрос, выделяющий заданные объекты и свойства для некоторого ограничения con_q. Применим этот алгоритм для каждого ограничения $con_q \in \{con_1, ..., con_\delta\}$ и соединим результирующие ODQL-запросы операцией **union**, удалив из результата дублирующиеся конструкции.

Теперь покажем, что приведенный алгоритм построения ODQL-запроса действительно строит запрос, выделяющий нужные сущности.

Цель ODQL запроса – выделить желаемое множество объектов. Объект как сущность DIM мы задаем через совокупность значений свойств и через связи с другими объектами. Поэтому и ограничения на выделяемое множество объектов мы можем задать в виде совокупности ограничений на свойства объектов и связность объектов с некоторым, ранее уже выделенным множеством других объектов. Все ограничения этих типов перечислены во фразе **for** ODQL-запроса.

Далее во фразе **from** первым указан класс объектов, объекты которого мы выделяем (базовый класс). Если какие-либо свойства, на которые наложены ограничения, не являются параметрами базового класса, то класс объектов этих свойств также указан во фразе **from**, а во фразе **links** указан путь получения объектов класса с таким свойством от объектов базового класса (для получения объектов используется орграф $\vec{G}$): для указания связи двух объектов через отношение *наследования* используется оператор **parent**, а для указания связи двух объектов через отношение *включения* исполь-

зуется оператор **contains**. Поэтому для любого объекта базового класса мы можем получить объекты, связанные с ним.

Теорема 1 доказана.

Замечание. Мы не только доказали теорему 1, но и привели алгоритм построения указанного в утверждении теоремы запроса. Этот алгоритм можно использовать при компиляции SODQL запроса в ODQL; отличие состоит в том, что отсутствуют ограничения на связь классов (так как в SODQL нет фразы **links**).

Естественно, в результате использования данного алгоритма получившийся ODQL-запрос будет больш*и*м за счет нескольких запросов, соединенных через **union**. Этого можно избежать, если ограничения на свойства в SODQL-запросе уже представляют собой конъюнкцию.

В качестве примера выполнения алгоритма рекомендуем рассмотреть трансляцию SODQL-запроса, приведенного в разделе 2.3.

Доказательство теоремы 2 основано на использовании для каждого заданного объекта запроса **update object**. Но важным является порядок выбора этих объектов. Поэтому для заданного множества связных объектов в графе связи их классов найдем все классы, каждый из которых не включен ни в один другой класс множества классов этих объектов и не является родителем другого класса для того же множества классов (такой класс должен существовать в силу указанного ранее ограничения определенности). Сделаем его базовым и определим схему слоев от него. Используя волновой алгоритм, подобный описанному для схемы классов на рис. 1, будем проходить по слоям классов от нулевого слоя и обновлять те объекты, которые меняют свои идентификационные свойства при помощи запроса **update object**.

Доказательство теоремы 3 основано на использовании для каждого заданного класса запроса **update class** и каждого его объекта запроса **update object**. Но опять-таки важным является порядок выбора классов и порядок выбора объектов каждого класса. Аналогично доказательству предыдущей теоремы построим такую же схему слоев классов. Используя волновой алгоритм для этой схемы классов, будем изменять классы в порядке их слоев с использованием запроса **update class** и после каждого изменения класса изменять его объекты с использованием запроса **update object**, если это необходимо.

3. Заключение

В заключение отметим, что поставленные задачи обоснования запросной полноты языка запросов ODQL и разработки алгоритма преобразования SODQL-запроса в ODQL-запрос выполнены. Но для того, чтобы объектный язык запросов был качественным, необходимо, чтобы он допускал реализацию оптимальных по времени выполнения запросов. Этой проблеме будет посвящена следующая статья.

Библиографический список

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

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

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

4. Рублев, В.С. Язык объектных запросов динамической информационной модели DIM [Текст] / В.С. Рублев // Моделирование и анализ информационных систем. – Т.17, № 3. – 2010. – С. 144–161.

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