Объектные модели ODMG и SQL десять лет спустя: нет противоречий

С. Д. Кузнецов <kuzloc@ispras.ru>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, дом 25

Аннотация. В 2005 г. я написал статью, в которой приводил наиболее существенные черты стандартов ODMG 3.0 (объектная модель ODMG) и SQL:2003 (модель данных SQL) и убедительно (как мне тогда казалось) доказывал, что сходство между объектной моделью и объектными расширениями SQL является чисто внешним, что за близкими на вид синтаксическими конструкциями скрываются глубинные различия модельного уровня. Примерами таких различий являются фоннеймановское разыменование объектных идентификаторов в модели ODMG по сравнению с ассоциативным разыменованием ссылочных значений в модели SQL, раздельное и независимое хранение объектов одного объектного типа в модели ODMG по сравнению с хранением всех строк типизированной таблицы в одной этой таблице в модели SQL, хранение объектных идентификаторов в экстенте в модели ODMG и хранение в аналоге экстента самих объектов в модели SQL и т.д. С тех пор прошло много лет, за которые я понял многие вещи, неправильно или недостаточно правильно понимавшиеся мной тогда, и постепенно пришел к выводам, что:

1. различия, которые мне казались глубинными, таковыми не являются, да и вообще не являются различиями уровня модели;
2. объектные расширения SQL обеспечивают не меньшие (а скорее большие) возможности, чем объектная модель ODMG;
3. при разумном (с позиций сообщества баз данных) использовании СУБД, основанной на модели ODMG, будут создаваться базы данных и средства манипулирования ими, близкие к тем, которые предписывает модель данных SQL.

Ключевые слова: модели данных, объектная модель, ODMG 3.0, SQL, разыменование объектных ссылок, размещение объектов.

DOI: 10.15514/ISPRAS-2015-27(1)-9

Для цитирования: Кузнецов С.Д. Объектные модели ODMG и SQL десять лет спустя: нет противоречий. Труды ИСП РАН, том 27, вып. 1, 2015 г., стр. 173-192. DOI: 10.15514/ISPRAS-2015-27(1)-9.

1. Введение

В 2005 г. я написал статью [1], в которой приводил наиболее существенные черты стандартов ODMG 3.0 [2] (объектная модель ODMG) и SQL:2003 [3-4] (модель данных SQL) и убедительно (как мне тогда казалось) доказывал, что сходство между объектной моделью и объектными расширениями SQL является чисто внешним, что за близкими на вид синтаксическими конструкциями скрываются глубинные различия модельного уровня. Вот цитата из [1], которая достаточно точно характеризует мои тогдашние взгляды: «Единственное, что объединяет модели данных SQL и ODMG, – развитая система типов с возможностью определения новых типов с произвольно сложной структурой значений. “Объектные” же расширения SQL, по сути, представляют собой дальнейшую синтаксическую маскировку операций естественного соединения.»

С тех пор прошло 10 лет, за эти годы я понял многие вещи, неправильно или недостаточно правильно понимавшиеся мной тогда, и постепенно пришел к выводам, что:

1. различия, которые мне казались глубинными, таковыми не являются, да и вообще не являются различиями уровня модели;
2. объектные расширения SQL обеспечивают не меньшие (а скорее большие) возможности, чем объектная модель ODMG;
3. при разумном (с позиций сообщества баз данных) использовании СУБД, основанной на модели ODMG, будут создаваться базы данных и использоваться средства манипулирования ими, близкие к тем, которые предписывает модель данных SQL.

Данная статья имеет следующую структуру. Во втором разделе кратко (и критически) обсуждаются основные черты модели ODMG и «объектной части» модели SQL, требуемые для понимания сути статьи. В третьем разделе рассматриваются два различия моделей ODMG и SQL, которые полагались фундаментальными в [1], и с новых позиций обосновывается реальная незначительность этих различий. В четвертом разделе демонстрируется, что «объектная часть» модели данных SQL содержит не меньше (а реально больше) «объектных» возможностей, чем модель ODMG, а при разумном использовании СУБД, основанных на модели ODMG, будут применяться методы и средства, близкие к тем, которые предписываются моделью SQL. В последнем разделе статьи приводятся заключительные замечания.

2. Основные черты объектной модели ODMG 3.0 и «объектной части» модели SQL

Много раз и по разным поводам мне приходилось писать о моделях данных ODMG и SQL. Наиболее кратко об этом говорилось в [1], наиболее полно – в [4], умеренно подробно – в [5]. В принципе, здесь можно было бы ограничиться ссылками на эти прошлые публикации, но мне кажется, что без

краткого критического повторения основных черт этих объектных моделей было бы трудно уловить основной смысл статьи. Естественно, далее предполагается, что читатели имеют общее представление о стандартах ODMG 3.0 и SQL:2003.

Общей характеристикой моделей данных ODMG и SQL является то, что обе они являются полнотиповыми. Говоря неформально, это означает, что в обеих моделях поддерживаются практически неограниченные возможности определения новых типов данных, и при определении любого нового типа данных можно использовать любой ранее определенный тип.

2.1 ODMG 3.0

В объектной модели ODMG имеются две категории типов: литеральные и объектные типы. Понятие литерального типа данных совпадает с традиционным понятием типа: тройка {множество значений, набор операций, множество литералов}. В состав литеральных типов входят атомарные и конструируемые типы. Атомарные литеральные типы включают традиционный набор типов целых чисел и чисел с плавающей точкой, типы символьных строк и булевский тип. Булевский тип в модели ODMG обычный, двузначный, поскольку в этой модели в явном виде неопределенные значения отсутствуют (неявно в разных местах стандарта они упоминаются, и это можно отнести к числу небрежностей, которых в стандарте предостаточно).

К конструируемым литеральным относятся структурные литеральные типы и типы коллекций. Структурные литеральные типы – это прямые аналоги типов записи в языках C/C++ (именованное множество именованных типизированных атрибутов). При определении структурных литеральных типов невозможно использовать наследование и определять операции. Несколько сбивает с толку то, что типом атрибута структурного литерального типа может являться не только любой ранее определенный литеральный тип, но и любой ранее определенный объектный тип. В последнем случае (имея в виду, что элементами объектного типа являются объекты) можно представить, что «значениями» такого атрибута являются объекты, что невозможно, поскольку объекты являются не значениями, а контейнерами, содержащими значения. Конечно, такое представление противоречит здравому смыслу и недопустимо. Но с этой (с моей точки зрения) небрежностью стандарта мы разберемся немного позже.

Наконец, конструируемые литеральные типы коллекций включают типы множеств, мультимножеств, списков и словарей (dictionary – множества пар <ключ, значение>, причем все ключи в этих парах должны быть различными). Типом элемента коллекции может быть любой ранее определенный литеральный или объектный тип. После многолетнего общения с моделью SQL типы мультимножеств кажутся мне подозрительными и не слишком нужными, упорядоченных типов данных я опасаюсь на основе многолетнего общения с XQuery [7] в контексте XML-ориентированных СУБД; наконец,

мне неизвестно, каким образом можно здраво использовать тип справочников в среде полноценной базы данных. Поэтому разумным (и действительно нужным) литеральным типом коллекций в модели ODMG считаю тип множества.

Объектные типы данных делятся на атомарные типы (определяемые пользователями объектные типы, user-defined types, UDT) и типы коллекций. Наиболее существенными в контексте объектно-ориентированных баз данных являются UDT, хотя, по моему мнению, их невозможно здравым образом использовать, не используя также и объектные типы коллекций.

Объектный тип служат для создания объектов с внутренней структурой и набором операций, специфицированными в определении данного объектного типа. В отличие от традиционных типов данных у любого объектного типа имеется операция создания и инициализации объекта, выделяющая память для нового объекта, присваивающая его атрибутам начальные значения и возвращающая уникальный идентификатор объекта (Object Identifier, OID), который в дальнейшем можно разыменовывать для доступа к атрибутам объекта и вызова его операций. Очевидно, что в соответствии со своими свойствами OID является *ссылкой на объект* (или абстрактным *адресом* объекта).

Мне кажется столь же очевидным, что понятие *объекта* в модели ODMG (да и в большинстве объектно-ориентированных языков программирования) практически (с точностью до различия в терминологии) не отличается от понятия *переменной* в более традиционных моделях данных и языках программирования. Действительно, переменная является типизированным контейнером, который может содержать значения соответствующего типа данных. Динамические переменные создаются операцией new, которая выделяет память для новой переменной и возвращает типизированную ссылку на соответствующий контейнер. Типом этой ссылки является *ссылочный тип*, парный типу значений, которые может содержать переменная. Так, например, в языках C/C++ парным ссылочным типом для типа integer является тип *integer.

Видимо, основным источником недоразумений, связанных с использованием терминов *объект, атрибут объектного типа* и т.д. является отсутствие в модели ODMG понятия ссылочного типа. В определении объектного типа под одним именем склеены определения типа значений объектов и ссылочного типа для соответствующих объектов (можно сказать, типа OID'ов объектов данного объектного типа). Везде, где объектный тип используется для типизации атрибутов, элементов коллекций и т.д., в действительности имеется в виду ссылочный тип, значениями которого являются OID.

Основными компонентами определения UDT являются определения атрибутов, связей и операций. В этой статье несущественны особенности средств определения операций в обсуждаемых моделях данных, поэтому этот аспект обсуждаться не будет.

Раздел определения атрибутов не отличается от соответствующего раздела определения структурного литерального типа: атрибуты именуются и типизируются, причем типом атрибута может являться любой ранее определенный литеральный или объектный тип. Разыменовывая значение атрибута объектного типа (OID), можно «перейти» к соответствующему объекту, т.е. получить доступ к его атрибутам и операциям. Если определить атрибут литерального типа множества с элементами некоторого объектного типа (типа множества OID'ов заданного объектного типа), то, перебирая элементы этого множества, можно последовательно «получить доступ» ко всем соответствующим объектам. Однако такие «ассоциации» являются однонаправленными: в объекте, на который указывают подобные ссылки, отсутствует какая-либо информация об их наличии.

Для организации двунаправленных «ассоциаций» используется механизм связей. Бинарные связи (а других в модели ODMG и нет) определяются сразу для двух объектных типов: в каждом из них задается имя связи, объектный тип, к которому она ведет, и имя парной связи в этом объектном типе. Такой способ определения двунаправленных бинарных ассоциаций в модели ODMG приходится применять по той причине, что OID представляет собой аналог абстрактного «фоннеймановского» адреса, при разыменовании OID происходит простой «переход по адресу». В этом состоит отличие от реляционной модели данных и модели данных SQL, где для организации связей видов 1:1 и 1:n используется механизм внешних ключей, для разыменования «ссылки» используется коммутативная операция соединения, и «связи» автоматически получаются двунаправленными. Это различие между моделями ODMG и SQL в [1] ошибочно казалось мне фундаментальным, и в третьем разделе статьи я постараюсь развеять это заблуждение.

Чтобы завершить обсуждение особенностей атомарного объектного типа, требуется сначала немного поговорить об объектных типах коллекций. Такие типы, как и литеральные типы коллекций, определяются с использованием конструктора типа, которому сообщается вид коллекции (множество, мультимножество, список или справочник) и тип элементов коллекции (любой ранее определенный литеральный или объектный тип). Объекты типа коллекции, как и объекты атомарных объектных типов, имеют OID и представляют собой контейнеры, хранящие соответствующие литеральные коллекции. Набор операций объектных типов коллекций предопределен и содержит, в частности, операцию, позволяющую перебирать элементы соответствующей коллекции.

По моему мнению, как и для литеральных типов коллекций, практическую ценность в контексте объектных баз данных имеют только объектные типы множеств. Более того, как я показываю в четвертом разделе статьи, наибольшей значимостью обладает одно «самоприменение» объектных типов множества внутри модели ODMG. Речь идет о понятии *экстента* атомарного объектного типа.

При определении атомарного объектного типа, помимо других свойств, несущественных для целей этой статьи, можно указать, что для этого объектного типа требуется автоматически поддерживать экстент – объект типа коллекции (по умолчанию, множества), типом элементов которого является определяемый атомарный объектный тип. Сам этот тип коллекции определяется неявным образом.

Как говорится в стандарте ODMG 3.0, экстент атомарного объектного типа в любой момент времени содержит все существующие объекты этого типа (как говорилось выше, имеются в виду OID-ы объектов). Наличие такого регулярного хранилища однотипных объектов позволяет использовать ненавигационные средства языка запросов OQL, определенного в модели данных ODMG.

При определении атомарных объектных типов можно использовать механизм наследования. В этой статье для нас несущественны особенности этого механизма, главное, что он поддерживает традиционную *семантику включения*: объекты, относящиеся к любому подтипу некоторого супертипа, являются и объектами этого супертипа. Семантика включения естественным образом поддерживается и для экстентов: значение экстента любого подтипа некоторого супертипа является подмножеством значения экстента этого супертипа.

Экстент атомарного объектного типа получает имя, совпадающее с именем этого типа. Экстенты являются единственным видом объектов, которые получают имена автоматически, без потребности дополнительных взаимодействий с системой.

Конечно, наличие в модели ODMG возможности явного конструирования объектных типов коллекций позволяет образовывать произвольное число регулярных хранилищ объектов (OID-ов) для одного и того же атомарного объектного типа. Но у экстента имеется ряд преимуществ: простота определения, наличие неявно присвоенного имени, автоматическое отслеживание содержимого (добавление OID при создании новых объектов и удаление OID при уничтожении объектов).

Можно заметить, что экстент атомарного объектного типа в модели ODMG напоминает таблицу в модели SQL. В [1] утверждалось, что при внешнем сходстве у этих двух понятий имеется принципиальное различие: в экстенте размещаются не сами объекты, а их OID-ы, а таблица является в некотором смысле контейнером, хранящем именно сами строки таблицы. В третьем разделе я покажу, что и это различие принципиальным не является.

2.2 Объектная часть модели SQL

В [1] при обсуждении модели данных SQL я ссылался на стандарты SQL:1999 и SQL:2003, хотя следовало сослаться еще и на стандарт SQL:1992 [8], в котором были заложены основы традиционной части модели SQL. За прошедшие годы были выпущены два новых стандарта: SQL:2008 и SQL:2011

[9]. В этих стандартах имеется ряд новшеств, влияющих на представление модели данных SQL в целом (в частности, темпоральные возможности [10]). Но на объектную часть модели SQL эти новшества не влияют, и я продолжаю опираться и в данной статье на стандарт SQL:1999 и его расширения в стандарте SQL:2003.

Как и в модели ODMG, фундаментом объектной части модели SQL является система типов. В модели SQL используется только традиционное понятие типа данных, никаких аналогов объектных типов модели ODMG в модели SQL нет. Имеются следующие основные категории типов:

1. точные числовые типы;
2. приближенные числовые типы;
3. типы символьных строк;
4. типы битовых строк;
5. типы даты и времени;
6. типы временных интервалов;
7. булевский тип;
8. типы коллекций;
9. анонимные строчные типы;
10. типы, определяемые пользователем (User Defined Type, UDT);
11. ссылочные типы.

Особенности первых шести категорий типов для этой статьи несущественны. Булевский тип в SQL трехзначный, включающий значения *true*, *false* и *unknown*, причем в контексте булевского типа значение *unknown* считается тем же самым, что и неопределенное «значение» NULL (это одно из самых неудачных мест стандарта SQL, но и оно для данной статьи несущественно). Коллекции в модели SQL могут быть массивами, множествами и мультимножествами, причем типом элемента коллекции может быть любой ранее определенный тип данных. Анонимные строчные типы (безымянные типы записи), главным образом, нужны для обеспечения возможности определять вложенные таблицы (таблицы, у которых хотя бы один столбец содержит значения-таблицы). Действительно существенными для объектной части модели данных SQL являются UDT – типы данных, определяемые пользователями (да и то не все) и ссылочные типы.

В категории UDT для объектной части модели SQL не существенны индивидуальные (distinct) типы, которые позволяют контролируемым образом учитывать в базах данных SQL семантику данных, иным образом представляемых с использованием базовых типов данных. Зато структурные UDT играют в объектной части модели решающую роль.

Внешне определение структурного UDT в модели SQL напоминает определение атомарного объектного типа в модели ODMG: набор типизированных атрибутов (типом атрибута может быть любой ранее определенный тип данных) и набор операций. При определении структурного UDT можно использовать достаточно простой механизм одиночного

наследования. Но, в отличие от атомарного объектного типа, структурный UDT в SQL является типом данных в традиционном смысле (т.е. множеством структурных значений) и поэтому ничего похожего на спецификацию связей в определении структурного UDT не содержится.

После определения структурного UDT его можно использовать (как и любой определенный тип данных) для определения новых типов данных (типов коллекций, анонимных строчных типов или UDT) или типизации столбцов таблиц. В последнем качестве структурные UDT объектную часть модели SQL не затрагивают. Важнейшими компонентами, вместе с которыми структурные UDT образуют объектную модель, являются типизированные таблицы и ссылочные типы.

Про ссылочные типы удобнее говорить после введения понятия типизированной таблицы. Формально типизированная таблица – это таблица, создаваемая с использованием конструкции CREATE table_name OF type_name, где type_name – имя ранее определенного структурного UDT. После выполнения такой операции в базе данных создается базовая таблица с именем table_name, которая при традиционной (не объектной) трактовке содержит n+1 столбец, имена и типы данных последних n которых совпадают с именами и типами атрибутов структурного UDT type_name. Если типизированная таблица используется исключительно в традиционной трактовке, содержимое первого столбца не существенно.

При объектной трактовке типизированной таблицы она представляется как контейнер объектов (отдельному объекту соответствует строка таблицы). Первый столбец типизированной таблицы в этом случае содержит так называемые ссылочные значения, уникальные типизированные значения, типом которых является ссылочный тип, парный структурному UDT, на котором определена типизированная таблица.

В модели SQL ссылочный тип может быть определен (вернее, объявлен, поскольку практически никакие специальные определения не требуются) для любого структурного UDT. При объявлении ссылочного типа требуется указать один из трех возможных способов генерации ссылочных значений: генерируется системой, задается пользователем или является комбинацией значений других атрибутов структурного UDT. Поскольку от ссылочного значения требуется уникальность (вообще говоря, полная уникальность; т.е. любое ссылочное значение одного ссылочного типа должно отличаться от любого другого ссылочного значения), реально работает только первый вариант – ссылочные значения, генерируемые системой.

Заметим, что с позиций реляционной модели данных ссылочные значения, генерируемые системой, больше всего похожи на суррогатные возможные ключи, которые, по всей видимости, впервые были предложены в [12], а более внятно были описаны в [11]. Интересные рассуждения относительно логических отличий суррогатных ключей в реляционной модели от объектных идентификаторов в объектных моделях приводятся в [13]. У теоретиков

реляционной модели отношение к суррогатным ключам варьируется от умеренно положительного до резко отрицательного (более правильным считается использование возможных ключей, отражающих семантику предметной области). Но если суррогатные ключи в таблицах базы данных используются, то основное их назначение состоит в обеспечении связок «возможный_ключ-внешний_ключ» для выполнения операций естественного соединения.

Ссылочное значение в объектной части модели SQL является аналогом OID в модели ODMG. В предыдущем подразделе уже отмечалось, что отсутствие понятия ссылочного типа является явным недостатком модели ODMG, приводящим к путанице при описании модели и затруднениям при ее понимании. В модели SQL все гораздо понятнее. Ссылочное значение (ссылка на объект, или уникальный идентификатор объекта) образуется при создании нового объекта (вставке новой строки в типизированную таблицу), сохраняется в нулевом столбце соответствующей строки (в стандарте SQL этот столбец называется «самоссылающимся» – self-referencing) и может сохраняться, в частности, в любом столбце любой таблицы (в частности, типизированной), если этот столбец определен на том же ссылочном типе.

Впоследствии при выполнении, например, запроса к базе данных может быть выполнена операция разыменования ссылочного значения и обеспечен доступ к атрибутам и операциям объекта. Как говорилось в предыдущем подразделе, в [1] меня сильно смущала природа этой операции «разыменования», которая из-за реляционного происхождения модели SQL является не фоннеймановской, а ассоциативной, основанной на использовании операции естественного соединения. В третьем разделе я поясню, почему считаю заблуждением свою тогдашнюю точку зрения, что специфика разыменования в модели SQL противоречит принципам объектно-ориентированной модели данных.

Наконец, поговорим еще немного о типизированных таблицах. В модели SQL можно представлять, что любая типизированная таблица является контейнером объектов, поскольку с точки зрения пользователя новый объект создается (и инициализируется) путем вставки в типизированную таблицу новой строки. На основе одного структурного UDT можно определить несколько типизированных таблиц, и все они будут контейнерами, сохраняющими однотипные объекты.

Если типизированная таблица A определена на структурном UDT T, и структурный UDT T_1 является непосредственным подтипом типа T, то на типе T_1 можно определить типизированную таблицу A_1, являющуюся подтаблицей таблицы A. Для супертаблиц и подтаблиц поддерживается семантика включения: любой объект, созданный в таблице A_1, считается входящим и в таблицу A.

Заметим, что понятие типизированной таблицы в объектной части модели SQL напоминает понятие экстента в модели ODMG. При поверхностном

рассмотрении и типизированная таблица, и экстент являются контейнерами однотипных объектов, к которым можно адресовать декларативные запросы на выборку объектов. Однако при более внимательном взгляде между типизированными таблицами и экстентами заметно различие, которое в [1] мне также казалось фундаментальным.

Основным различием казалось то, что в модели ODMG экстент объектного типа сохраняет не сами объекты, а их OID'ы. Операция создания объекта (т.е. в основном выделения памяти под объект) – это операция объектного типа, а не экстента. Природа экстента является такой же, как у любой другой «коллекции объектов», вручную поддерживаемой пользователем объектной базы данных: все такие коллекции содержат OID'ы объектов, а сами объекты располагаются независимо один от другого где-то в «куче» базы данных.

В модели SQL операция создания объекта является операцией типизированной таблицы (правильнее сказать, над типизированной таблицей). Объект создается путем выполнения операции вставки в таблицу новой строки, и естественно представлять себе, что эта строка является частью таблицы. Кстати, здесь полезен некоторый дуализм в понимании того, что такое таблица в модели данных SQL. В теоретически правильном понимании (следуя, в частности, представлениям Дейта и Дарвена в Третьем манифесте, см., например, [6]) таблица, это переменная базы данных, тип которой определяется заголовком таблицы, а значениями являются мультимножества строк, соответствующих этому заголовку. При таком понимании, конечно, некорректно считать, что у отдельной строки таблицы имеется какое-то собственное пространство хранения.

Однако возможно и дуальное понимание таблицы как типизированного контейнера (вообще говоря, изменяемого размера), в котором распределяется память для хранения однотипных строк. Можно утверждать, что эти два понимания не противоречат одно другому, и при втором понимании у каждой строки, входящей в таблицу имеется свое собственное пространство хранения, прямой адрес которого пользователям не сообщается (в следующем разделе мы увидим, что и это не всегда так).

Как и в модели ODMG, в модели SQL можно сохранять в базе данных дополнительные коллекции ссылочных значений существующих однотипных объектов, однако такие коллекции принципиально (как мне казалось в [1]) отличаются от коллекции объектов, сохраняемой в типизированной таблице: типизированная таблица хранит сами объекты плюс их ссылочные значения, а упомянутые дополнительные коллекции объектов – только ссылочные значения.

Перейдем теперь к критике взглядов автора десятилетней давности.

3. Действительно ли существенны различия моделей ODMG и SQL?

После публикации статьи [1] прошло много лет. За эти годы я много раз рассказывал студентам об особенностях моделей ODMG и SQL, много размышлял об этом и постепенно изменял свое мнение о существенности различий между этими моделями.

3.1 Разные способы адресации объектов

В модели ODMG объекты адресуются своими OID'ами. OID – это абстрактный аналог однонаправленного фоннеймановского адреса. OID хранится только на стороне ссылающейся сущности. При выполнении операции создания нового объекта его OID определяется системой, является обратным параметром операции создания и, вообще говоря, не сохраняется внутри создаваемого объекта. Из-за однонаправленности объектной ссылки в модели ODMG требуется дополнительное понятие связи, представляемой контролируемой парой однонаправленных ссылок.

В объектной части модели SQL объекты адресуются ссылочными значениями, которые с позиций реляционной модели данных являются суррогатными ключами. Ссылочное значение объекта, вообще говоря, хранится внутри объекта и на стороне любой ссылающейся на него сущности. Операция разыменования ссылочного значения (доступа по ссылке) производится на основе коммутативной реляционной операции естественного соединения, которая по своей сути является ассоциативной. В частности, по этой причине в модели SQL не требуется дополнительное понятие связи (все ссылки являются двунаправленными).

Действительно ли эти различия являются фундаментальными различиями моделей ODMG и SQL? Первое и основное соображение, которое заставило меня в этом усомниться, основано на следующем наблюдении.

Принято считать, что подавляющее большинство программируемых компьютеров, начиная от времени их появления в середине 1940-х гг., и до наших дней, основано на архитектуре фон Неймана. В соответствии с принципами фон Неймана, программа и обрабатываемые ей данные хранятся в однородной прямоадресуемой памяти. Программы выполняются последовательно команда за командой, пока не встретится команда перехода, т.е. насильственного изменения счетчика команд. Команды могут вырабатывать прямые адреса памяти, из которых выбираются и в которые записываются данные.

Однако при этом никого не смущает, что в этих компьютерах с 1970-х гг. между аппаратурой управления виртуальной памятью и аппаратурой управления основной памятью присутствует поддерживаемый аппаратурой дополнительный уровень сверхбыстродействующей ассоциативной памяти – кэш. Если не стремиться к полной строгости, можно считать, что кэш – это

буферная память, состоящая из блоков одного и того же небольшого размера (часто 64 байта). При каждой выборке команд или данных из основной памяти соответствующий блок памяти буферизуется в одном из блоков кэша, причем в этом же блоке запоминается виртуальный адрес, обращение по которому привело к этой выборке.

При выработке некоторой командой виртуального адреса для чтения команд или данных аппаратура управления виртуальной памятью до преобразования виртуального адреса в физический обращается к кэшу, в котором происходит параллельный ассоциативный поиск этого виртуального адреса во всех блоках. При удачном завершении поиска нужные команды или данные передаются в процессор, а при неудачном завершении продолжается работа аппаратуры управления виртуальной памятью.

Запись по виртуальному адресу работает аналогично, но после размещения блока данных в кэше нужные данные обязательно проталкиваются и в основную память. Естественно, аппаратура управления кэш-памятью производит в случае надобности замещение блоков кэша.

Так вот, операция «разыменования» виртуального адреса с использованием кэша больше всего напоминает операцию естественного соединения из реляционной модели данных: коммутативную операцию, выполняемую в параллельной и ассоциативной манере. Никого не смущает, что фоннеймановский виртуальный адрес сначала «разыменовывается» в компьютерах совершенно не фоннеймановским образом. По всей видимости, различие к технике разыменования ссылок на объекты не должно быть серьезным модельным различием и при сравнении моделей ODMG и SQL.

Стоит сделать еще одно замечание, показывающее, насколько не обязательно особенности моделей данных прямо влияют на реализации. Десять лет тому назад, когда писалась статья [1], мне казалось очевидным, что единственным разумным способом выполнения операции разыменования ссылочных значений в объектной части модели SQL является применение естественного соединения. Действительно, в первом столбце типизированной таблицы хранятся суррогатные значения возможного ключа, а в столбцах таблиц, ссылающихся на эту типизированную таблицу, сохраняются соответствующие значения внешнего ключа. Чтобы перейти от строки ссылающейся таблицы к объекту типизированной таблицы нужно выполнить, строго говоря, естественное полусоединение этих таблиц.

Всем известно, что любая операция соединения является сравнительно дорогостоящей. Поэтому я был удивлен, когда вскоре после написания [1] в одной из статей, посвященной некоторым деталям реализации объектных расширений в Oracle, прочитал, что в этой системе запросы, написанные на «объектном диалекте» SQL с применением синтаксических конструкций навигации по объектным ссылкам, выполняются эффективнее, чем эквивалентные запросы с соединениями. Я тогда действительно считал, что этот «объектный диалект» SQL является тем, что принято называть

«синтаксическим сахаром»: подслащиванием SQL в угоду любителям объектно-ориентированных конструкций.

Ан был неправ, что отметил уже в 2007 г. в [14]. Всем известно, что, начиная с System R [15], практически во всех SQL-ориентированных СУБД с построчным хранением таблиц для каждой строки таблицы на физическом (обычно скрытом от пользователя) уровне поддерживается уникальный идентификатор (в System R он назывался tid – tuple identifier), который обеспечивает «почти прямой» доступ к строке во внешней памяти. В Oracle аналог такого идентификатора называется rowid и, хорошо ли это или плохо, он открыт для пользователей баз данных. Можно объявить любую базовую (реально хранимую в базе данных) таблицу как содержащую столбец rowid (реально этот столбец в базе данных не хранится, поскольку система и так знает значение rowid для любой строки любой таблицы). Эти значения можно читать, сохранять (явно) в столбцах других таблиц и использовать в запросах для прямого (совсем не реляционного) доступа к строкам.

Тот же фокус использовала Oracle при организации доступа к объектам, сохраняемым в «объектных таблицах» (аналог типизированной таблицы в модели SQL). В Oracle ссылочные значения, генерируемые системой, в действительности являются rowid. Поэтому самоссылающийся столбец явным образом в объектной таблице не хранится, но его значения известны системе и доступны пользователям в качестве ссылочных значений объектов.

С одной стороны, Oracle не нарушает предписаний объектной части модели SQL. Если не присматриваться к деталям представления ссылочных значений, то можно представлять, что операция их разыменования основана на естественном соединении. Но в действительности ссылка трактуется как прямой однонаправленный адрес.

С моей точки зрения, приведенные замечания и наблюдения убеждают в том, что различия в способе разыменования ссылок на объекты не являются принципиальными различиями моделей ODMG и SQL.

3.2 Разные подходы к размещению объектов в памяти базы данных

В модели ODMG операция создания и инициализации объекта является операцией соответствующего атомарного объектного типа. Если при определении объектного типа не специфицировано требование наличия экстента этого типа, то любой заново созданный объект существует изолированно от других объектов данного типа, если не будет выполнена явная операция помещения OID созданного объекта в какой-либо контейнер (объект типа коллекции, скорее всего, множества). Если экстент в определении атомарного объектного типа специфицирован, то при создании любого нового объекта данного типа его OID автоматически помещается в этот экстент (объект типа коллекции, скорее всего, множества).

С моей точки зрения, база данных, в которой поддерживаются изолированные объекты, не слишком осмысленна, поскольку, в частности, к ней невозможно адресовать декларативные запросы, поддержка коллекций OID вручную обременительна и не очень полезна, и поэтому наиболее естественным образом объектная база данных в модели ODMG представляется в виде набора экстентов, именованных объектов типа множества, элементами которого являются OID'ы атомарного объектного типа (более подробно и конкретно остановимся на этом вопросе в четвертом разделе статьи).

В объектной части модели SQL операция создания и инициализации нового объекта является операцией типизированной таблицы (над типизированной таблицей) – вставка новой строки. Поскольку для типизированных таблиц в модели SQL допускаются как объектное представление (типизированная таблица как контейнер объектов), так и традиционное (унаследованное от реляционной модели данных) представление (типизированная таблица как переменная со значениями-множествами строк), на уровне модели данных приходится считать, что объекты, созданные в данной типизированной таблице, в ней и располагаются.

Изолированных объектов в модели SQL не бывает, любой существующий объект входит к какую-то типизированную таблицу. Как и в модели ODMG, никто не мешает создавать дополнительные объекты, сохраняющие коллекции ссылочных значений заданного типа, вручную контролировать их содержимое и производить в них поиск (как и в случае ODMG, я не уверен, что эта потенциальная возможность полезна). Тем самым, в модели SQL объектная база наиболее естественным образом представляется в виде набора типизированных таблиц, именованных объектов типа множества, элементами которого являются объекты, структура и поведение которых соответствует структурному UDT типизированной таблицы.

Так что в этом отношении модели ODMG и SQL различаются только тем, что в экстентах хранятся OID'ы объектов, а в типизированных таблицах – сами объекты. В [1] я считал и это различие фундаментальным. Теперь мне так не кажется.

Во-первых, в стандарте ODMG недаром почти всегда говорится «объект» в тех случаях, когда имеется в виду OID: объекты как значения атрибутов объектных типов, объекты как элементы коллекций и т.д. С одной стороны, как отмечалось выше, это вносит путаницу, поскольку понятно, что один объект не может быть значением, например, нескольких атрибутов. Но с другой стороны, это делается умышленно для простоты и удобства пользователей объектных баз данных, которым совершенно ни к чему думать о вспомогательных операциях разыменования.

Во-вторых, хотя в модели SQL действительно разный статус имеют типизированная таблица (реальный контейнер объектов) и любая коллекция ссылочных значений (контейнер ссылок на объект), на практике более

естественно использовать именно типизированные таблицы, и в этом случае отсутствие операции разыменования для пользователей может пройти так же незаметно, как наличие этой операции при использовании экстентов в модели ODMG.

Наконец, таблица в модели SQL – это понятие более высокого логического уровня, чем экстент в модели ODMG. На физическом уровне строки таблиц хранятся в блоках внешней памяти независимо одна от другой. В таблицу они связываются благодаря наличию идентификатора таблицы во всех ее хранимых строках или за счет использования индексов. Так что в действительности на уровне хранения данных типизированная таблица вовсе не является контейнером объектов.

В целом, материал этого раздела демонстрирует, что фундаментальных различий между моделью ODMG и объектной частью модели SQL нет.

4. «Объектная мощь» SQL и «реляционная натура» ODMG

Сравнивать модели данных, описанные совершенно в разных стилях и с использованием разной (и не очень строгой) терминологии, дело трудное и неблагодарное. Однако, как мне кажется, в целом мне удалось сопоставить объектную модель ODMG и объектную часть модели SQL. Это сопоставление позволило мне прийти к трем основным выводам, первый из которых содержится в конце предыдущего раздела.

Второй вывод состоит в том, что **«объектные» возможности модели SQL не меньше тех, которые предоставляются моделью ODMG**. Действительно, системы типов моделей ODMG и SQL сопоставимы. В SQL система типов богаче и строже (более адекватный булевский тип, имеются явные ссылочные типы, отсутствуют смущающие ум объектные типы и т.д.), но по сути обеспечиваются те же возможности, что в ODMG. Наследование и семантика включения поддерживаются почти на равных в обеих моделях. Инкапсуляция объектов в обеих моделях фактически отсутствует (для сокрытия отсутствия инкапсуляции состояния объектов используются старые фокусы с неявной поддержкой автоматически определяемых методов *observer* и *mutator* для всех атрибутов объектов [12]). Язык запросов OQL из модели данных ODMG немногим отличается (не только синтаксически, но и по своей семантике) от «объектного диалекта» языка запросов SQL.

Проблема SQL (и в том числе объектной части этой модели данных) состоит не в том, что в ней чего-то не хватает, а в громадном переизбытке возможностей. В частности, объектные средства SQL во многом трудно оценить по той причине, что их можно использовать практически в произвольной комбинации с бесчисленными «традиционными» возможностями. Например, с использованием одного и того же структурного UDT с *n* атрибутами можно определить «объектную» типизированную таблицу с *n+1* атрибутом, но допускается и определение «традиционной»

таблицы с одним столбцом, типом которого является данный структурный UDT.

Наконец, третий, заключительный вывод формулируется следующим образом: **при разумном использовании любой СУБД, основанной на модели ODMG, будут использоваться базы данных и средства манипулирования ими, близкие к тем, которые предписываются моделью данных SQL**.

На самом деле, язык OQL, описанный в [2] содержит набор средств, которые обеспечивают возможности явной навигации по графам объектов, связанных объектными ссылками, а также возможности, близкие к SQL, для формирования декларативных запросов над коллекциями OID. С точностью до терминологии и синтаксических обозначений можно считать, что OQL входит в современный SQL (начиная с SQL:2003 [4]).

Чтобы иметь возможность разумно и эффективно пользоваться этими возможностями, наиболее естественно представлять базу данных в виде набора экстентов объектных типов, определенных в схеме базы данных. Такое представление базы данных является естественным, поскольку средства автоматической поддержки экстентов предопределены в модели ODMG, и экстенты являются единственными сущностями объектной базы данных, которые автоматически получают имена (совпадающие с именами соответствующих атомарных объектных типов). Поддержка дополнительных коллекций OID'ов кажется мне обременительной, ненадежной и избыточной (помимо необходимости явно отслеживать состав таких коллекций, им нужно еще и нетривиальным образом присваивать имена).

Легко видеть, что в этом случае схема объектной базы данных становится очень похожей на схему SQL-ориентированной базы данных, построенной в объектном стиле. Атомарные объектные типы (если не обращать внимания на их странности) напоминают структурные UDT SQL, а экстент – это почти то же, что и типизированная таблица (в конце концов, что представляет собой внутренняя структура объекта в модели ODMG как не строку в смысле SQL). Эти соображения в целом обосновывают приведенный выше третий вывод.

Но можно сказать больше. В SQL-ориентированных базах данных никто не запрещает обращаться с типизированными таблицами как с таблицами традиционными: выполнять проецирование, соединения таких таблиц и т.д. Конечно, результирующие таблицы типизированными уже не будут, но в мире SQL это не должно никого смущать, поскольку в одной базе данных могут содержаться и типизированные, и традиционные таблицы. Поскольку экстент базы данных ODMG является аналогом типизированной таблицы SQL, то в принципе было бы достаточно логичным допустить над ними операции, доступные для традиционных таблиц SQL.

И здесь мы интересным образом отбрасываемся на двадцать лет в прошлое, когда я пытался придумать аналог реляционной алгебры [16]. Не буду здесь воспроизводить суть этой старой статьи, которая неожиданно снова стала актуальной. Но хочу заметить следующее. Модель данных SQL эклектична и

из-за этого трудно постижима, но зато она не ограничивает пользователей по чисто идеологическим причинам, для которых отсутствуют технические подтверждения. Кто знает, не стоит ли (или не стоило ли?) временами жертвовать идеологическими принципами?

5. Заключение

Основная цель этой статьи состоит в том, чтобы исправить ошибки, сделанные автором в публикации 10-летней давности. Думаю, что здесь речь идет не только об установлении более справедливого отношения к моделям ODMG и SQL. Объектно-ориентированные СУБД снова начинают пользоваться спросом, единственной общепринятой объектной моделью является ODMG 3.0, и эта статья показывает, что (а) эта модель не так уж плоха, как может показаться при чтении стандарта [2], и (б) объектная часть модели SQL фактически ничем не отличается от ODMG. Возможно, эти заключения принесут пользу разработчикам и пользователям баз данных.

Десять лет тому назад я выступил на семинаре Московской секции ACM SIGMOD с докладом [17], по мотивам которого была написана статья [1]. Доклад был с пониманием встречен аудиторией, которая в целом согласилась с моими тогдашними доводами. В конце 2014 г. я выступил на том же семинаре с докладом [18], по мотивам которого написана эта статья. И этот доклад тоже был встречен аудиторией вполне лояльно, и с моими теперешними доводами никто всерьез не спорил. Поэтому в конце доклада я пообещал собравшимся, что если в мире моделей данных не произойдет что-либо непредвиденное, то следующий раз я выступлю с докладом на близкую тему не раньше, чем через 10 лет. Думаю, что то же следует сказать и насчет статей по этой тематике.

Список литературы

[1]. Сергей Кузнецов. "Объектны" ли объектные расширения языка SQL?, 2005. http://citforum.ru/database/articles/sql_odmg/

[2]. The Object Data Standard: ODMG 3.0. Edited by R.G.G. Cattel, Douglas K. Barry. Morgan Kauffmann Publishers, 2000.

[3]. Jim Melton. "Advanced SQL:1999. Understanding Object-Relational and Other Advanced Features". Morgan Kaufmann Publishers, 2003.

[4]. Сергей Кузнецов. Наиболее интересные новшества в стандарте SQL:2003, 2004. http://citforum.ru/database/sql/sql2003/

[5]. С.Д. Кузнецов. Базы данных: языки и модели. Москва, Бином, 2008.

[6]. С.Д. Кузнецов. Три манифеста баз данных: ретроспектива и перспективы. Базы данных и информационные технологии XXI века. Материалы международной научной конференции. Москва, 29-30 сентября 2003 г. Москва, РГГУ, 2004, стр. 52-229. http://citforum.ru/database/articles/manifests/

[7]. Дон Чемберлин. XQuery: язык запросов XML. Открытые системы, № 1, 2003, стр. 61-72. Оригинал: D. Chamberlin. XQuery: An XML query language. IBM SYSTEMS JOURNAL, VOL 41, NO 4, 2002, pp. 597-615.

[8]. C.J. Date. A Guide to SQL Standard (4th Edition). Addison-Wesley Professional, 1996)
[9]. Fred Zemke. What's new in SQL:2011. SIGMOD Record, Volume 41, Number 1, March 2012, pp. 67-73.
[10]. Krishna Kulkarni and Jan-Eike Michels. Temporal Features in SQL:2011. SIGMOD Record, Volume 41, Number 3, September 2012, pp. 34-43.
[11]. Э. Ф. Кодд. Расширение реляционной модели для лучшего отражения семантики. Системы управления базами данных № 5, 1996, издательский дом «Открытые системы», новая редакция: 2009 г., http://citforum.ru/database/classics/codd_2/. Оригинал: E.F. Codd. Extending the Database Relational Model to Capture More Meaning. ACM Transactions on Database Systems, Vol. 4, № 4, December 1979.
[12]. Hall, P., Owlett, J., and Todd, S. Relations and Entities. In Modelling in Data Base Management Systems, G.M.Nijssen, Ed., North-Holland Pub. Co., Amsterdam, 1976.
[13]. Date, C. J. Relational Database Writings 1994–1997. Chapter 12. Object Identifiers vs. Relational Keys. Addison-Wesley, 1998.
[14]. С.Д. Кузнецов. Объектно-реляционные базы данных: прошедший этап или недооцененные возможности? Труды Института системного программирования, т. 13, часть 2, М., ИСП РАН, 2007, стр. 115-140.
http://ispras.ru/ru/proceedings/docs/2007/13/2/isp_2007_13_2_115.pdf (pdf)
http://citforum.ru/database/articles/ordbms10/ (html)
[15]. С.Д. Кузнецов. Развитие идей и приложений реляционной СУБД System R. Москва, ВИНИТИ, 1989, Тем. изд. "Итоги науки и техники. Вычислительные науки". Т.1. Стр. 3-75. http://www.citforum.ru/database/articles/art_27.shtml
[16]. S. Kuznetsov. OODMBS's Query and Programming Languages: What Do They Provide and What Do We Need. Extended Information Systems Technology. Proceedings of the International East/West Database Workshop, Klagenfurt, Austria, 25-28 Sept. 1994, J.Eder, L.A.Kalinichenko (Eds), Springer-Verlag, 1995, 138-147. DOI: 10.1007/978-1-4471-3577-7_10.
[17]. С.Д. Кузнецов. Объектно-ориентированные базы данных и объектные расширения языка SQL. Семинар Московской секции ACM SIGMOD, 23 декабря 2004 г. http://synthesis.ipi.ac.ru/sigmod/seminar/s20041223
[18]. С.Д. Кузнецов. Объектные расширения SQL «объектны»! Семинар Московской секции ACM SIGMOD, 25 декабря 2014 г.
http://synthesis.ipi.ac.ru/sigmod/seminar/s20141225

ODMG and SQL object models ten years later: there are no contradictions

S.D. Kuznetsov <kuzloc@ispras.ru>
Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, Russia, 109004.

Abstract. In 2005, I wrote an article in which I discussed the most important features of the standards ODMG 3.0 (ODMG object model) and the SQL:2003 (SQL data model) and convincingly (as it seemed to me) argued that the similarity between the object model and object extensions to SQL is purely external, that close in mind syntactic constructions hide

deep differences at the model level. Examples of such differences include von Neumann-style dereference of ObjectIDs in the ODMG model vs join-style dereference of reference values in the SQL model, separate and independent store of objects of one and the same object type in the ODMG model vs store of all raws of a typed table (SQL analogy of object) within this table, store of ObjectISs within extents in the ODMG model vs store within analogy of extent of objects their self in the SQL model, etc. Since then, it took many years for which I understood many things that were wrongly or insufficient correctly understood by me then, and gradually came to the conclusions that:

1. differences that seemed to me deep, are not such, and indeed are not differences at the model level;
2. the object extensions of SQL provide no less (and rather large) capabilities than the ODMG object model;
3. reasonably (from the standpoint of the database community) used DBMSes based on the ODMG data model, one will create databases and tools to manipulate them similar to those prescribed by the SQL data model.

Keywords: data model, object model, ODMG 3.0, SQL, dereferencing, object relocation.

DOI: 10.15514/ISPRAS-2015-27(1)-9

For citation: Kuznetsov S.D. ODMG and SQL object models ten years later: there are no contradictions. *Trudy ISP RAN/Proc. ISP RAS*, vol. 27, issue 1, 2015, pp. 173-192 (in Russian). DOI: 10.15514/ISPRAS-2015-27(1)-9

References.

[1]. Sergey Kuznetsov. "Objektny" li objektnye rasshirenija jazyka SQL? [Whether are "object" the object extensions of SQL?], 2005 (in Russian). http://citforum.ru/database/articles/sql_odmg/

[2]. The Object Data Standard: ODMG 3.0. Edited by R.G.G. Cattel, Douglas K. Barry. Morgan Kauffmann Publishers, 2000.

[3]. Jim Melton. "Advanced SQL:1999. Understanding Object-Relational and Other Advanced Features". Morgan Kaufmann Publishers, 2003.

[4]. Sergey Kuznetsov. Naibolee interesnye novshestva v standarte SQL:2003 [The most interesting innovations in the SQL:2003 standard], 2004 (in Russian). http://citforum.ru/database/sql/sql2003/

[5]. S.D. Kuznetsov. Bazy dannyx: jazyki i modeli [Databases: Languages and Models], Moscow, Binom, 2008 (in Russian).

[6]. S.D. Kuznetsov. Tri manifesta baz dannyx: retrospektiva i perspektivy [Three manifests of databases: Retrospect and Prospects]. In Databases and Information Technology in XXI Century. Proceedings of the International Conference. Moscow, RSGU, 2004, pp. 52-229 (in Russian). http://citforum.ru/database/articles/manifests/

[7]. D. Chamberlin. XQuery: An XML query language. IBM SYSTEMS JOURNAL, VOL 41, NO 4, 2002, pp. 597-615.

[8]. C.J. Date. A Guide to SQL Standard (4th Edition). Addison-Wesley Professional, 1996)

[9]. Fred Zemke. What's new in SQL:2011. SIGMOD Record, Volume 41, Number 1, March 2012, pp. 67-73.

[10]. Krishna Kulkarni and Jan-Eike Michels. Temporal Features in SQL:2011. SIGMOD Record, Volume 41, Number 3, September 2012, pp. 34-43.

[11]. E.F. Codd. Extending the Database Relational Model to Capture More Meaning. ACM Transactions on Database Systems, Vol. 4, № 4, December 1979.

[12]. Hall, P., Owlett, J., and Todd, S. Relations and Entities. In Modelling in Data Base Management Systems, G.M.Nijssen, Ed., North-Holland Pub. Co., Amsterdam, 1976.

[13]. Date, C. J. Relational Database Writings 1994–1997. Chapter 12. Object Identifiers vs. Relational Keys. Addison-Wesley, 1998.

[14]. S.D. Kuznetsov. Objektno-reljacionnye bazy dannyx: proshedshij ehtap ili nedoocenennye vozmozhnosti? [Object-relational databases: the last stage or undervalued opportunities?]. Trudy ISP RAN [Proceedings of ISP RAS], v. 13, part 2, M., ISP RAS, 2007, pp. 115-140. (in Russian)
http://ispras.ru/ru/proceedings/docs/2007/13/2/isp_2007_13_2_115.pdf (pdf)
http://citforum.ru/database/articles/ordbms10/ (html)

[15]. S.D. Kuznetsov. Razvitie idej i prilozhenij reljacionnoj SUBD System R [Evolution of ideas and applications of the relational DBMS System R]. Moscow, VINITI, 1989, " Itogi nauki i texniki. Vychislitelqnye nauki [The results of science and technology. Computer Science]". V.1. Pp. 3-75. http://www.citforum.ru/database/articles/art_27.shtml

[16]. S. Kuznetsov. OODMBS's Query and Programming Languages: What Do They Provide and What Do We Need. Extended Information Systems Technology. Proceedings of the International East/West Database Workshop, Klagenfurt, Austria, 25-28 Sept. 1994, J.Eder, L.A.Kalinichenko (Eds), Springer-Verlag, 1995, 138-147. DOI: 10.1007/978-1-4471-3577-7_10

[17]. S.D. Kuznetsov. Objektno-orientirovannye bazy dannyx i objektnye rasshirenija jazyka SQL [Object-oriented databases and object extensions of the SQL query language]. Seminar Moskovskoj sekcii ACM SIGMOD, 23 dekabrja 2004 g. [Seminar of the Moscow ACM SIGMOD chapter, December 24th, 2004] (in Russian),
http://synthesis.ipi.ac.ru/sigmod/seminar/s20041223

[18]. S.D. Kuznetsov. Objektnye rasshirenija SQL «objektny»! [Object extensions of SQL are «object»!]. [Seminar of the Moscow ACM SIGMOD chapter, December 25th, 2014] (in Russian), http://synthesis.ipi.ac.ru/sigmod/seminar/s20141225.