Модел. и анализ информ. систем. Т. **19**, № **2** (2012) 97–108

УДК 519.854.2

Отношение истории и динамика схем баз данных СУБД DIM

Рублев В.С.

Ярославский государственный университет им. П.Г. Демидова

e-mail: roublev@mail.ru

получена 12 февраля 2012

Ключевые слова: база данных, СУБД DIM, схема базы данных, эволюция схемы, динамика типов

Рассматривается задача динамики типов для новой объектной СУБД DIM [1, 2] и связанная с ней динамика схем ее баз данных.

1. Постановка задачи

В основе введения новой объектной СУБД DIM [1, 2] лежит динамика информации, т. е. данных объектов, их свойств, типов и методов обработки данных. Поскольку в данной технологии темпоральный характер баз данных отражает *отношение истории* объектов, свойств, классов, то для каждой из перечисленных сущностей, историю изменения которых следует хранить во времени, вводятся два дополнительных параметра: *момент рождения* и *момент смерти*. В момент смерти объекта он перестает существовать на текущий момент, но на смену ему могут прийти новые объекты, которые мы называем *последователями* данного объекта, а сам данный объект и, возможно, другие объекты с таким же *моментом смерти*, мы называем *предшественниками*. Точнее говоря, *последователь* – это тот объект, который сменил данный на *момент* его *смерти* и, возможно, с другими объектами, а *предшественник* – это тот объект, который был сменен данным на *момент* его *рождения* (также, возможно, с другими объектами). На множестве всех объектов модели O введено антисимметричное транзитивное бинарное отношение, которое мы будем называть *отношением истории объектов*. Оно определяет объект-предшественник, объект-последователь и момент изменения. По этому отношению для каждого объекта может быть найдено *множество последователей* на *момент* его *смерти* и *множество предшественников* на *момент* его *рождения*.

Для описания динамики изменения классов вводятся аналогичные конструкции для классов: *момент рождения, момент смерти, отношение истории классов, классы-предшественники* и *классы-последователи*. Если производится изменение

класса, то все объекты класса претерпевают изменения и могут измениться также связи между классами и объектами. Для изменения классов (объектов) в модели вводятся специальные процедуры изменения классов (объектов). Если на смену некоторому классу c_{old} приходит класс c_{new}, то все объекты класса c_{old} прекращают свое существование, а на смену им приходят объекты класса c_{new}. При этом добавление/удаление параметров в классе и добавление/удаление связей с другими классами проходит следующим образом:

1) данные описания класса копируются в новый класс c_{new} с новым идентификатором класса (при этом класс c_{old} со старым идентификатором становится предшественником, а класс c_{new} с новым идентификатором становится последователем в отношении истории классов);

2) данные объектов старого класса c_{old} копируются в данные объектов нового класса, и каждый объект получает новый идентификатор (при этом каждый объект со старым идентификатором становится предшественником объекта с новым идентификатором в отношении истории объектов);

3) добавляются новые параметры нового класса c_{new} как дополнительные параметры;

4) для всех объектов нового класса определяются значения этих параметров;

5) группа каждого добавленного параметра нового класса c_{new} изменяется в соответствии с необходимостью;

6) удаляются ненужные параметры нового класса c_{new}, и проверяются ограничения целостности для объектов класса (уникальность идентификационных параметров каждого объекта);

7) добавляются связи нового класса c_{new} с другими классами (проверяются ограничения целостности, связанные с наличием для каждого объекта единственного родительского объекта для каждого родительского класса данного класса) и для объектов этих других классов определяются при необходимости связи с объектами нового класса (такие объекты получают новый идентификатор и связь с объектами старого идентификатора при помощи отношения истории объектов);

8) удаляются при необходимости ненужные связи нового класса c_{new} с другими классами и одновременно удаляются связи между объектами этих классов (для таких объектов других классов идет копирование в объекты с новым идентификатором, и введение связи со старыми объектами через отношение истории объектов); проверяются ограничения целостности для всех объектов, для которых старый класс c_{old} содержал родительские объекты.

Аналогично этим конструкциям вводится динамика свойств. Например, свойство может изменить тип данных или связь классов. Поэтому введены отношение истории параметров классов и свойств связей, которые позволяют определить значения параметров объектов и их связей в любой период их “жизни”.

Динамика схем баз данных DIM, включающая динамику классов и их связей, отражена во многих работах (см., например, [3] – [10]) как эволюция схем баз данных. Однако от эволюции динамику схем баз данных отличает дополнение ее отношением истории классов, связей, объектов, которая определяет динамику этих сущностей. Так как динамика схем существенным образом зависит от технологии баз данных, для DIM должен быть разработан свой подход. При этом помимо проверки целостности данных (объектов и их связей) выдвигается требование максимального использования преобразуемых данных, что трудно сделать вручную при большом изменении схемы.

Каждая схема классов DIM связана с конечным множеством O их объектов, с конечным множеством свойств A, связанных с параметрами этих классов, и с конечным множеством L связей объектов из O. Совокупность $\{O, A, L\}$ и функций динамики объектов и классов определяет OD-модель (см. [1, 2]) как формализованную дискретную детерминированную модель (там же).

Преобразование схемы данных DIM возможно разделить на 2 этапа, с которыми связаны 2 задачи:

1) не известна схема классов данных DIM, к которой надо преобразовать исходную схему классов DIM, но известна информация о свойствах объектов и их связях, к которым должны быть преобразованы исходные объекты и их связи; нужно получить новую (преобразованную) схему классов DIM;

2) известна схема классов данных DIM, к которой надо преобразовать исходную схему DIM; нужно получить алгоритм преобразования классов и их объектов.

2. Типы связи включения и их история при преобразовании схемы данных

В [1] рассмотрены 2 типа включения между включающим и включенным классом:

1) *простое включение*, когда на бинарном отношении этих классов (как множеств объектов) не вводится других отношений (отсутствует класс включения);

2) *функциональное включение*, когда на бинарном отношении этих классов вводится *функциональное* отношение, определяемое классом связи (для каждой пары объектов включающего и включенного единственным образом определяется объект класса связи).

В ряде случаев OD-модель определяет отношение k объектов, где $k > 3$. Его задание также возможно при помощи отношения включения, если расширить это отношение следующим образом:

1) выделяем 3 класса c_1, c_2, c_3 объектов, участвующих в этом отношении, где c_3 – класс связи включения для классов c_1, c_2;

2) остальные классы $c_4, \ldots, c_k$ объектов, участвующих в этом отношении, определяем как родительские для класса c_3;

3) определяем остальные параметры класса связи c_3.

При этом связь включения двух объектов классов c_1, c_2 может определяться несколькими объектами связи, которые имеют различные наборы родительских объектов. Поэтому такой тип включения назовем *мультивключением*. Пример мультивключения рассмотрен ниже в разделе 4.

Реализация включения (см. [2]) организуется в таблице включения классов, где задаются 3 поля для ссылок на включающий, включенный классы и класс связи (при простом типе включения последнее поле пустое), а также поля *Дата рождения* и *Дата смерти*, при помощи которых организуется история связей включения классов. Аналогично организуется связь включения объектов. При изменении типа включения классов замена класса включения обязательна, и отмечается историей этой связи в таблице включения классов, а также историей этой связи для всех пар объектов включающих и включенных.

3. Алгоритмы преобразования схемы данных

Рассмотрим задачи, поставленные в разделе 1.

В [1], [2] показано, что любая OD-модель, у которой связи определяются бинарными отношениями объектов, может быть реализована некоторой схемой классов DIM. При этом приведен алгоритм построения такой схемы (в доказательстве леммы 1 там же). Если же имеются связи более чем двух объектов, то с использованием мультивключения можно обобщить этот результат, скорректировав этот алгоритм. Поэтому по изменениям множества объектов O_{new}, свойств A_{new} и связей L_{new} можно по исходной OD-модели $\{O_{old}, A_{old}, L_{old}\}$ получить преобразованную OD-модель $\{O_{new}, A_{new}, L_{new}\}$, а по ней, использовав скорректированный алгоритм, построить преобразованную схему классов DIM. Этим и решается первая задача.

Перейдем к рассмотрению задачи 2. Назовем исходную схему классов *старой схемой*, а преобразованную – *новой схемой*. С целью решения второй задачи введем *орграф эволюции схемы* следующим образом:

1. В качестве вершин орграфа выбираются классы старой схемы, не вошедшие в новую схему (в дальнейшем будем их называть *старыми вершинами*), и классы новой схемы, отсутствующие или измененные по отношению к старой схеме (в дальнейшем будем их называть *новыми вершинами*), а также общие вершины старой и новой схем, для которых изменились связи с другими классами (в дальнейшем будем их называть *общими вершинами*).

2. Из *старой* вершины, соответствующей классу старой схемы, проводится дуга в отличную от нее *новую* вершину, соответствующую классу новой схемы, если некоторые параметры первого класса переносятся во второй. Такая дуга помечается именами переносимых параметров и в дальнейшем называется *дугой переноса*.

3. Вершины старой схемы соединяются связями классов старой схемы. То же делается для вершин новой схемы, но выделяются новые и изменившиеся по отношению к старой схеме связи, которые определяются как *дуги связи* для новых и общих вершин орграфа.

Из орграфа эволюции схемы выделяются компоненты связности, в каждую из которых включается компонента связности подграфа новых и общих вершин, связанная *дугами связи* и дополненная только теми старыми вершинами, которые связаны с ее новыми вершинами *дугами переноса*.

Предлагается следующий волновой алгоритм для каждой компоненты связности орграфа эволюции:

1. В качестве *начальной* выбирается любая *новая* вершина компоненты связности, которая не соответствует классу связи новой схемы и которая не имеет дуг связи с родительскими и включающими вершинами-классами для этой вершины-класса. Такая вершина должна существовать в силу ограничения определенности (см. [1]).

2. 2.1. Для выбранной вершины и каждой *дуги переноса*, ведущей в нее из старых вершин, осуществляется перенос данных в соответствии с именами переносимых параметров дуги. Для каждого из параметров класса этой вершины, являющихся новыми (отсутствуют в старой схеме) такой параметр вводится сначала в группу *дополнительных параметров*, затем определяются данные каждого объекта класса для этого параметра, после чего этот параметр перемещается в группу главных неидентификационных параметров или главных идентификационных параметров.

   2.2. Вводится *отношение истории* между классами начала дуг переноса и выбранным классом (для первых классов определяется текущая дата в качестве значения параметра *Дата смерти*, а для класса выбранной вершины это значение определяется для параметра *Дата рождения*).

   2.3. Если при переносе данных от старого класса к новому должен произойти перенос связи включения к классу общей вершины и тип включения не изменяется, то вводится история такой связи. При этом в таблице отношения включения классов для этой связи поле *Дата смерти* получает значение текущей даты, заносится новая связь включения и ее поле *Дата рождения* получает значение текущей даты.

   2.4. Для соответствующих объектов этих классов, связанных переносимым значением, вводится отношение истории объектов, при котором старый объект получает текущую дату в качестве значения *параметра* Дата смерти, а новый объект получает эту дату в качестве значения параметра *Дата рождения*.

   2.5. Если в п.2.3 связь включения получила историю, то в таблицу соответствующего отношения включения старых объектов для каждой записи старой связи определяется для поля *Дата смерти* значение текущей даты, а в таблицу, соответствующую отношению новых объектов, заносятся записи для нового отношения объектов со значением поля *Дата рождения*, равного текущей дате.

   2.6. Выбранная вершина помечается.

3. Если есть еще выбранные непомеченные вершины, то для каждой из них выполняется предыдущий шаг алгоритма; иначе выполняется следующий шаг.

4. Если есть еще невыбранные вершины компоненты, то выбираются те вершины, которые связаны с помеченными *дугами связи*, не отвечающими связям включения с изменением типа включения и для каждой из них выполняется шаг 2 алгоритма.

5. 5.1. Если среди помеченных вершин есть еще невыбранные вершины компоненты, которые являются классами связи включения и связаны с помеченными *дугами связи*, отвечающими связям включения с изменением типа включения, то они выбираются; иначе алгоритм заканчивает свою работу.

   5.2. Если связь включения изменяет тип, то вводится история такой связи: в таблице связи включения классов старая запись в поле *Дата смерти получает* значение текущей даты, и делается новая запись таблицы со ссылками на новые классы включающий, включенный, связи включения (если его нет, то пустая ссылка) и значением текущей даты для поля *Дата рождения*. При этом, если новая связь имеет тип *мультивключения*, то для класса связи этого включения в таблице наследования классов заводятся новые строки для этого класса как дочернего и всех родительских классов, определенных в новой схеме *дугами связи* наследования с родительскими классами и определяется текущая дата в качестве значения поля *Дата рождения*.

   5.3. Для нового класса связи отношения включения, если он есть, пополняются значениями переноса параметры переноса, если есть дуги переноса, ведущие к этому классу; также определяются значения параметров всех объектов этого класса связи. В таблице метауровня отношения включения классов связи все старые объекты этого отношения получают текущую дату в качестве *Даты смерти*. В таблице метауровня нового отношения включения классов все новые объекты получают текущую дату в качестве *Даты рождения*.

   5.4. Выбранная вершина связи включения помечается, и снова выполняется предыдущий шаг 5.

4. Пример преобразования схемы базы данных

В качестве примера рассмотрим ситуацию перевозки грузов кораблями. Имеются 3 основных сущности:

- *грузы*, определяемые именами;

- *порты*, каждый из которых нуждается в некоторых грузах в определенном *количестве*,

- *корабли*, каждый из которых перевозит некоторый список грузов с их *количествами* в некоторый список портов по определенному маршруту.

Предполагается, что в каждом из портов каждый корабль после разгрузки может быть догружен другими грузами, но мы будем рассматривать лишь часть общей задачи, которая не включает разгрузку и погрузку.

В схему БД введем 3 основных класса *Грузы, Порты* и *Корабли*. Введем отношение включения для классов *Порты* и *Грузы*, определяющее потребность тех или иных грузов в каждом порту. Для этого отношения определим класс связи *Грузы порта*. Для каждого корабля определим 2 сущности: *трюм* корабля, основной характеристикой которого является *грузовместимость*, и *навигатор* корабля (средство управления кораблем), основной характеристикой которого должен быть *маршрут* обхода портов, в которые корабль везет грузы. С этими сущностями свяжем дочерние для класса *Корабли* классы *Трюм* и *Навигатор*. Классы *Трюм* и *Грузы* свяжем отношением включения через класс связи *Грузы корабля*, определяющий количество того или иного груза, перевозимого в трюме корабля. Свяжем классы *Навигатор* и *Порты* отношением включения с классом связи *Маршрут*, который задает список портов маршрута корабля, определяя для каждого порта маршрута следующий порт. В самом классе *Навигатор* определим параметром *первый порт маршрута*. Полученная схема представлена на рис. 1.

Построенная БД по этой схеме DIM позволяет решать многие задачи, например, найти список кораблей, везущих максимальный груз и при этом в наименьшее число портов. Но в то же время анализ схемы показывает ее несовершенность. Например, схема не позволяет определить, сколько того или иного груза везет некоторый корабль в порт его маршрута. Поэтому схема нуждается в изменении.

Анализ класса связи *Грузы корабля показывает*, что для каждого груза, который везет корабль, необходимо определить не общее количество этого груза на корабле, а его количество для каждого порта, в который идет корабль. Задать отношение включения для 3 сущностей непосредственно в системе DIM нельзя, но можно для класса связи *Грузы корабля* определить родительский класс *Порты*, и тем самым количество груза корабля задавать для каждого порта в отдельности. При этом отношение включения класса *Грузы* в класс *Порты* может быть ликвидирована, так как количество груза, перевозимого в порт всеми кораблями, может быть определена суммированием количеств по всем кораблям, везущим такой груз в этот порт.

К тому же при преобразовании схемы можно дополнительно устранить классы *Трюм* и *Навигатор*, перенося их параметры в класс *Корабли*. В результате получается *новая схема*, изображенная на рис. 2.

Построим *орграф эволюции схемы*, определив в качестве *старых вершин* классы *Трюм, Навигатор* старой схемы, в качестве *общих вершин* новой схемы (новых вершин нет) классы *Корабли, Порты, Маршрут, Грузы, Грузы корабля*, в качестве *дуг переноса* дуги, связывающие старые вершины с классом *Корабли*, помеченные параметрами *Грузовместимость* и *Первый порт маршрута*, а в качестве новых *дуг связи* связь классов *Корабли* и *Грузы корабля*, связь классов *Корабли* и *Маршрут* и связь наследования классов *Порты* и *Грузы корабля*. На рис. 3 представлен этот орграф.

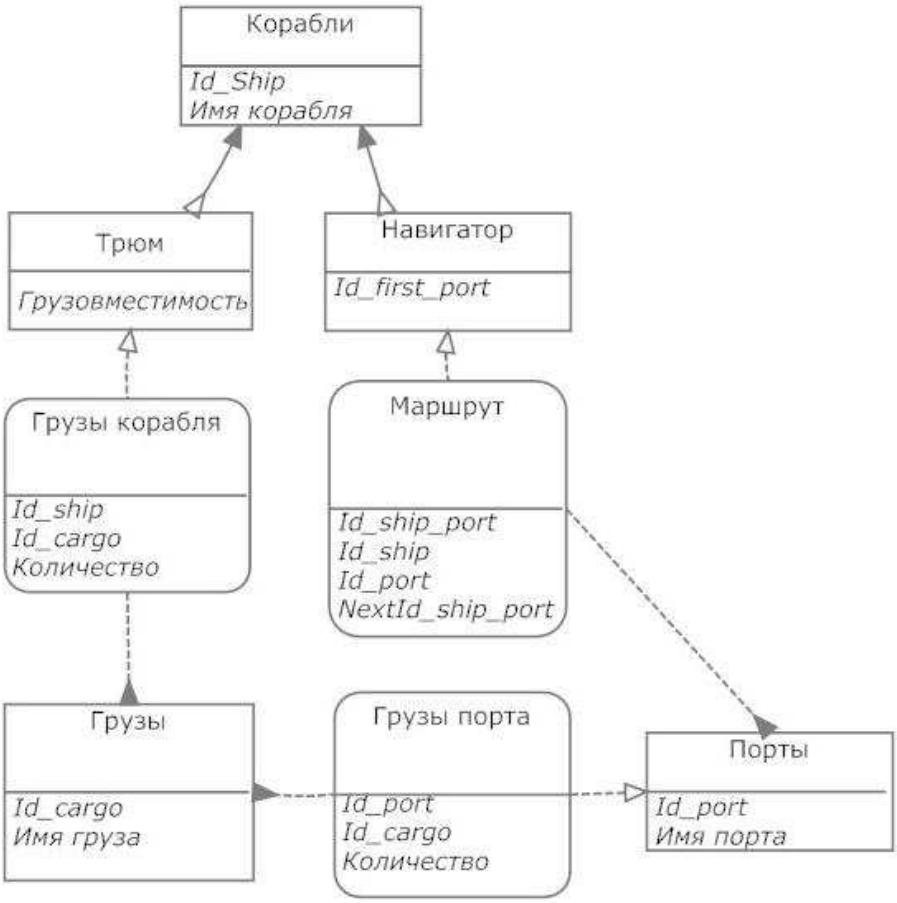

Рис. 1. Старая схема

Преобразование данных с дополнением отношениями истории осуществляется по алгоритму следующим образом:

1. В качестве начальной вершины выбирается класс *Корабли*. Образуется новый класс *Корабли* (новый идентификатор класса), в который переносятся параметры старого класса *Корабли*, а также дополняются параметрами *Грузовместимость, Первый порт маршрута (Id_first_port)*. Старый класс *Корабли* получает значение параметра *Дата смерти*, равное текущей дате, а новый класс получает это же значение для параметра *Дата рождения*. Идентификаторы обоих классов включаются в отношение истории классов.

2. В соответствии с дугами переноса классы *Трюм, Навигатор* получают значение параметра *Дата смерти*, равное текущей дате, и связываются отношением истории с новым классом *Корабли*. Каждый объект старого класса *Корабли* копируется в создаваемый объект нового такого же класса, а эти объекты связываются отношением истории объектов. При этом старый объект получают значение параметра *Дата смерти*, равное текущей дате, а новый объект получает это же значение для параметра *Дата рождения*. Используя это от-

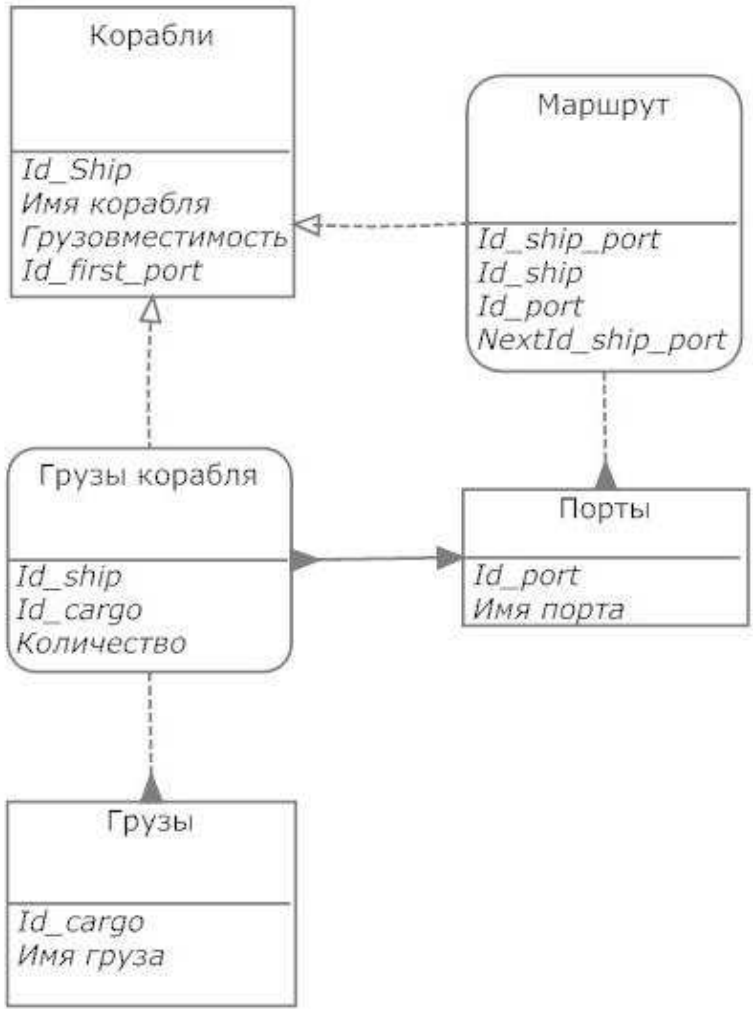

Рис. 2. Новая схема

ношение истории, связывающее идентификаторы объектов старого и нового классов *Корабли*, связываем отношением истории соответствующие первому идентификатору дочерние объекты классов *Трюм, Навигатор* и объект нового класса *Корабли*. При этом объекты классов *Трюм, Навигатор* получают значение параметра *Дата смерти*, равное текущей дате, а объект нового класса *Корабли* получает значения параметров *Грузовместимость, Первый порт маршрута (Id_first_port)* объектов классов *Трюм, Навигатор*. В этом и состоит перенос значений.

3. Выбирается класс *Маршрут*, связанный с новым классом *Корабли дугой связи*. Каждый объект нового класса *Корабли* связывается отношением включения с тем объектом связи класса *Маршрут*, с которым был связан объект класса *Навигатор*, связанный отношением истории с указанным объектом класса *Корабли*.

4. Выбирается класс *Грузы корабля*, связанный с новым классом *Корабли ду-*

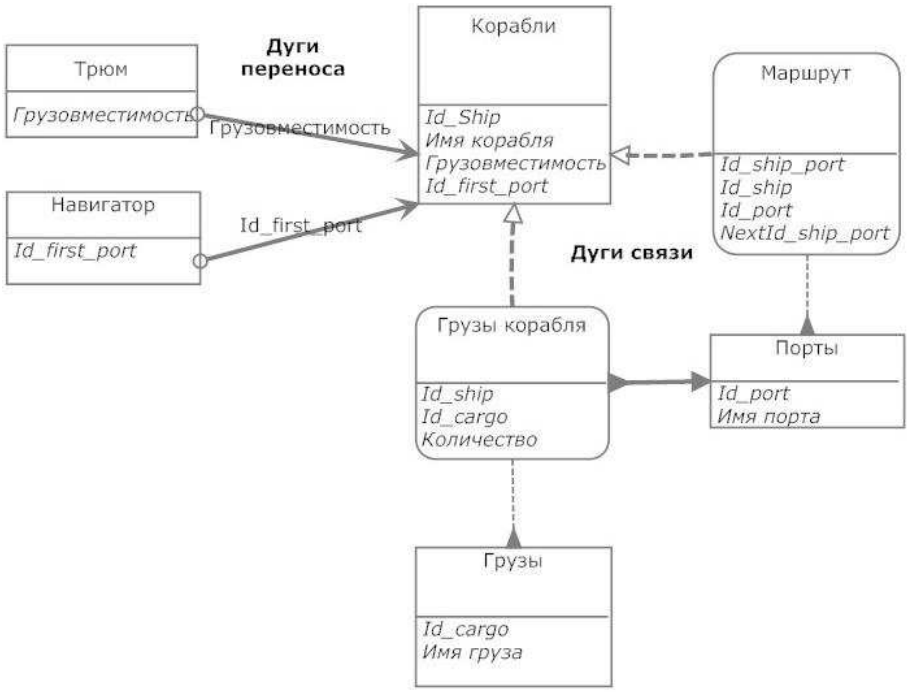

Рис. 3. Орграф эволюции схемы

гой связи. Каждый объект нового класса *Корабли* связывается отношением включение с тем объектом связи класса *Грузы корабля*, с которым был связан объект класса *Трюм*, связанный отношением истории с указанным объектом класса *Корабли*.

5. Выбирается класс *Грузы корабля*, связанный *дугой связи* с классом *Порты*. Каждый объект o класса *Грузы корабля*, определяющий количество некоторого груза для некоторого корабля, должен быть заменен на ряд объектов $o_1, \ldots, o_m$ по числу m портов, в которые этот корабль перевозит этот груз, а количество груза k разбито на количества $k_1, \ldots, k_m$ для каждого из портов, так что $\sum_1^m k_i = k$. Поэтому объект o получает значение параметра *Дата смерти*, равное текущей дате, а объекты $o_1, \ldots, o_m$ получают это же значение для параметра *Дата рождения*, после чего первый объект связывается с каждым из новых объектов (для корабля и груза) отношениями истории. После того, как все объекты класса *Грузы корабля* будут переопределены, вводится отношение наследования между классами *Порты* (родительским) и *Грузы корабля* (дочерним) и для каждого порта, в который корабль везет груз, определяется дочерний объект в классе *Грузы корабля*.

6. В результате не только схема переопределена, но и все данные перенесены, либо переопределены, а также введены отношения истории классов и объектов.

5. Заключение

Полученные алгоритмы позволяют не только преобразовать схему данных, но также связать отношениями истории данные схемы и объектов БД. Последнее очень важно для запросов, которые охватывают данные объектов, тип которых меняется во времени, за период, включающий изменения типа. Поэтому эти алгоритмы могут быть положены в основу построения *генератора взаимодействий эволюции схем.*

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. Динамическая информационная модель. Концепция новой объектной технологии баз данных. 2011. LAP LAMBERT Academic Publishing, Saarbrücken, Germany, 2011. 112 с.

3. Кукс С.В. Аксиоматизация эволюции схемы xml-баз данных // Программирование. 2003. Т. 29, № 3. С. 140 – 146.

4. Leonardi E., Bhowmick S. S. Detecting changes on unordered xml documents using relational databases: a schema-conscious approach // CIKM. 2005. P. 509 – 516.

5. Симановский А.А. Поддержка эволюции схем данных XML–реляционных баз данных. 2008. Программирование. Т.34, №1. С. 16 – 26.

6. Kim W., Chou H.-T. Versions of schema for object-oriented databases // VLDB. 1988. P. 148 – 159.

7. Osborn S. L. The role of polymorphism in schema evolution in an objectoriented database // IEEE Transactions on Knowledge and Data Engineering. 1989. Vol. 1, №3. P. 310 – 317.

8. Peters R. J., Ozsu M. T. An axiomatic model of dynamic schema evolution in objectbase systems // ACM Trans. Database Syst. 1997. Vol. 22, №1. P. 75 – 114.

9. Rashid A., Sawyer P. Object database evolution using separation of concerns // SIGMOD Record. 2000. Vol. 29, №4. P. 26 – 33.

10. Semantics and implementation of schema evolution in object-oriented databases / J. Banerjee, W. Kim, H.-J. Kim, H. F. Korth // SIGMOD Conference. 1987. P. 311 – 322.

Evolution of DBMS DIM Database Schemes

Roublev V.S.

Keywords: database, DBMS DIM, database scheme, scheme evolution, type dynamics

The article is devoted to a type dynamics for a new object DBMS DIM [1, 2] and the schemes evolution of its databases.

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