УДК 537.226; 538.915

© 2013 г. **А.Н. Родионов**, д-р техн. наук
(Вычислительный центр ДВО РАН, Хабаровск)

КАЧЕСТВО ДАТАЛОГИЧЕСКИХ СХЕМ. ПРИНЦИП МИНИМАЛЬНОЙ ИЗБЫТОЧНОСТИ И МИНИМАЛЬНО-ИЗБЫТОЧНЫЕ ДАТАЛОГИЧЕСКИЕ КОНСТРУКЦИИ. II

Рассмотрены и исследованы порожденные виды избыточных данных: синтетические типы и “вычисляемые” атрибуты. Выделены характерные признаки синтетических типов, обоснована необходимость включения их в состав моделей данных на правах системоформирующих типов. Сформулировано условие целесообразности присутствия “вычисляемых” атрибутов в составе структур даталогических конструкций.
Ключевые слова: сущность-связь, семантическое моделирование, модель данных, структурная избыточность, синтетические типы сущностей, “вычисляемые” атрибуты, справочники-типы, текущие состояния сущностей, предметная область.

Введение

Избыточность в моделях данных может быть не только преднамеренной, как, например, в случае с избыточностью свойств, но и являться следствием использования наперед заданных правил порождения структур баз данных. К одному из таких видов относится избыточность, возникающая вследствие включения в модели данных справочников, в которых хранятся не знаки – реальные сущности, а их обобщения – типы.

В работе [1] рассматриваются условия, которые приводят к появлению такого вида избыточности и с позиции увеличения быстродействия выполнения *Select*-инструкций обосновывается ее целесообразность.

Синтетические типы сущностей ни в какой предметной области обнаружить нельзя. Но в моделях данных они иногда встречаются. Происхождение таких типов, их отличительные признаки, виды взаимодействий, в которые они вступают между собой и с другими, “несинтетическими” типами, структуры их представляющие, – все эти вопросы составляют содержание второй части статьи.

Избыточность данных, порождаемая справочниками-типами

Модель данных признается полной, если посредством ее структур удается, в частности, отразить местоположение сущностей предметной области. Понятно, что сущности не пребывают в статичном состоянии, а время от времени или со-

вершают определенные перемещения в границах предметной области, или появляются, в ней или покидают ее. Способы, посредством которых задаются местоположения объектов, перечислены и описаны в [2]. Непосредственно к возникновению избыточности данных может приводить только один из них – способ "относительного позиционирования объектов", и только в тех случаях, когда он применяется к объектам-типам.

Объекты-типы – это класс объектов, полученный в результате применения абстракции обобщения к знакам (реально существующим объектам) и последующего отслеживания жизненных циклов уже не знаков, а только типов. Таким образом, обобщающий тип, – это именованная группа объектов, каждый экземпляр которой имеет одинаковые значения актуальных, отслеживаемых информационной системой свойств. Отличительная черта всех экземпляров объектов, отнесенных к такому типу, – способ, в соответствии с которым ведется их учет. Во всех операциях (действиях), связанных с этими объектами, отслеживаются не единичные (подобные) экземпляры, а одновременно несколько экземпляров, участвующих в конкретной операции (например, в операции списания или операции реализации). Из всего сказанного вытекает, что обязательным атрибутом структуры (таблицы), в которой хранятся названия обобщающих типов, является количественный атрибут, предназначенный для отражения числа подобных объектов, присутствующих в предметной области в определенный момент времени. Такую структуру еще часто называют "справочником", или "списком".

Требование "относительного позиционирования", которое предполагает "участие" других разновидностей объектов в задании позиционирования, приводит к необходимости исключения "количественного атрибута" из списка (справочника) и переноса его в структуру, посредством которой, собственно говоря, это позиционирование и осуществляется. Наглядно такая трансформация демонстрируется на рис.1. Структуру, которая содержит количественный атрибут, будем далее именовать "Склад", проводя своеобразную аналогию с функциями подсистемы управления запасами, где подобная структура используется наиболее часто.

Нетрудно заметить, что количество $Q_z = f(I_z, s_{1z}, s_{2z}, .., s_{jz})$, где I_z – значение, принимаемое атрибутом *Id_rf*; j – индекс справочника, участвующего в задании местоположения; s_{jz} – значение первичного ключа z-й записи j-го справочника. Понятно, что изменение значений s_{jz} – не что иное, как отражение факта изменения местоположения объекта.

Непосредственно к новым значениям, принимаемым s_{jz}, приводят действия, связанные с любыми перемещениями объектов. Такие действия, в свою очередь, находят отражение в других категориях структур, непосредственно фиксирующих сами факты перемещений. Это могут быть или документальные структуры, или "слабые сущности". И те и другие напрямую не связаны со складской структурой. Такие связи подразумеваемые и поддерживаются на прикладном программном уровне. Другими словами, как только в документальной структуре фиксируются изменения значений любой комбинации переменных Q_z, I_z или s_{jz} они, эти изменения, автоматически отражаются на содержании и мощности

складского отношения.

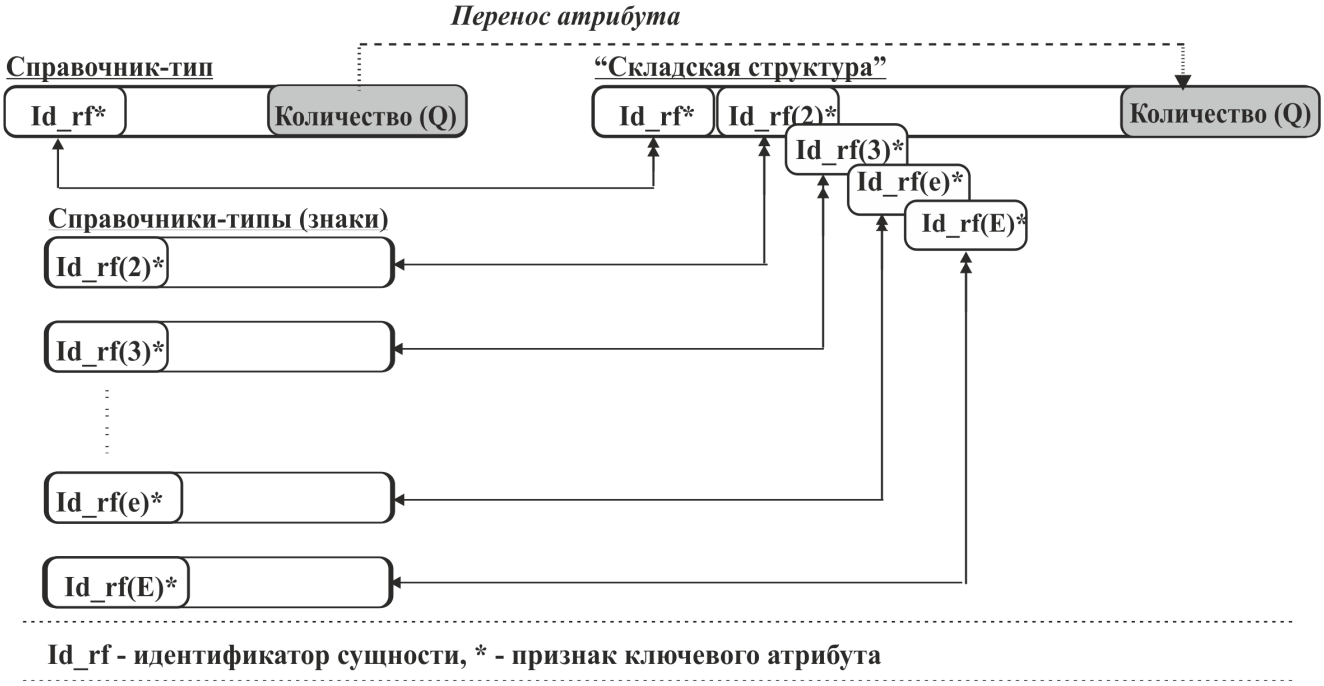

Рис. 1. Механизм порождения одной из “складских структур”, соответствующих справочнику-типу.

До сих пор никакой избыточности, обусловленной фиксацией местоположений экземпляров объектов-типов, не возникало, поскольку ни сама структура “склад”, ни ее атрибуты не являлись избыточными. Избыточность может возникнуть только в одном случае. Если количественных атрибутов в составе складской структуры по каким-либо причинам окажется несколько. И такие причины объективно существуют.

Для демонстрации сказанного остановимся на постановке одной из задач, которая практически всегда присутствует в функциональном поле любой фактографической информационной системы. Это задача фиксации начальных состояний сущностей. Применительно к рассматриваемому случаю необходимо определиться, на какой момент времени ***t*** будут фиксироваться значения Q_z. Предположим, что ***t*** – это текущее, астрономическое время t_c. Тогда для произвольно выбранного типа сущности в процессе решения задачи *“Фиксация или корректировка начальных остатков”* потребуется мгновенно (или за очень ограниченное время) ввести все $Q_z(t_c)$. Очевидно, что сделать это крайне затруднительно, если принять во внимание, что мощность подобных отношений бывает весьма значительной. Более того, значения $Q_z(t_c)$ ввиду изменения самого ***t*** и совершения разнообразных приходно-расходных операций за период $[t,\ t_c]$, $t \geq t_c$ постоянно корректируются.

Эта задача решается сегодня достаточно просто.

В одной из структур базы данных фиксируется некоторое начальное время t_n. Как правило, это ***00*** часов ***00*** минут первого дня текущего месяца. В складскую структуру для каждой z вводятся $Q_z(t_n)$. Соответственно все $Q_z(t)$ могут быть получены расчетным путем посредством выполнения простых арифметических действий:

$$Q_z(t) = \begin{cases} Q_z(t_n) + \sum_{p=1}^{P} Q_{zp}(t_n, t) - \sum_{r=1}^{R} Q_{zr}(t_n, t), t > t_n, \\ Q_z(t_n) - \sum_{p=1}^{P} Q_{zp}(t_n, t) + \sum_{r=1}^{R} Q_{zr}(t_n, t), t < t_n, \end{cases} \quad (1)$$

где P – количество "источников" поступления: приобретено "на стороне", поступило с другого склада, оприходовано вследствие ошибочного списания и т.п.; R – общее число "вариантов" убытия: списано на производство, потери от недостачи и порчи, реализовано, передано на другой склад и т.п.; $\sum_{p=1}^{P} Q_{zp}(t_n, t)$ – итоговый приход за период $[t_n, t]$; $\sum_{r=1}^{R} Q_{zr}(t_n, t)$ – итоговый расход за тот же период.

Подавляющее большинство задач управления транзакциями, а именно так принято называть задачи, которые фиксируют первичные данные, оперирует не $Q_z(t_n)$, а $Q_z(t_c)$. На практике оказывается, что крайне затратно каждый раз рассчитывать $Q_z(t_c)$ ввиду высоких накладных расходов на их получение. Во-первых, существенно возрастает нагрузка на вычислительные мощности (особенно в многопользовательских приложениях), во-вторых, чем больше ($t_c - t_n$), тем больше времени потребуется на получение $Q_z(t_c)$. Собственно говоря, именно этими соображениями обусловлено включение в состав складской структуры избыточного атрибута, хранящего значения $Q_z(t_c)$.

Как и в случае с избыточностью свойств [3], подтвердим экспериментальным путем, выполняя ***SQL-скрипт***, целесообразность хранения и постоянной актуализации избыточных по своей сути $Q_z(t_c)$. Выберем следующие значения входных характеристик: $P = 3$; $R = 3$; $Q_p(t) = 10^4$; $Q_r(t) = 10^4$; $Q_p(t)$ и $Q_r(t)$ здесь – средняя интенсивность соответственно приходных и расходных операций (количество записей, обрабатываемых в сутки); t принято равным ***суткам***. $Q_z(t_n) = 10^4$**.** Отношение числа сущностей, еще не представленных в "складской структуре", к тому количеству, которые там уже присутствуют, составляет 0,1. Тем самым признается, что информационная система преодолела начальный период эксплуатации и вышла на устойчивый режим работы.

Программа, отвечающая за актуализацию $Q_z(t_c)$**,** может выглядеть примерно так, как это показано в *SQL*-скрипте, созданном в среде *MS SQL (2008)*.

```
Use University;
SET STATISTICS TIME ON; Go
DECLARE @Dat_b datetime; Set @Dat_b = '  ';
DECLARE @Dat_e datetime; Set @Dat_e = '  '
IF OBJECT_ID('dbo.ITOG') is not null DROP table dbo.ITOG
/*Dat_b – первый день расчетного периода, Dat_n – последний день расчетного периода*/
/* Расчет итогового прихода по всем источникам за интервал [Dat_n, Dat_b] */
SELECT  Id_go, SUM(Kol) AS Kol Into dbo.ITOG From
 (SELECT  P_d.Id_doc, P_d.Id_go, P_d.Kol  FROM    Prihod1_head P_h
      INNER JOIN  Prihod1_details P_d ON P_h.Id_doc = P_d.Id_doc
         WHERE  P_h.Date_time >= @Dat_b AND P_h.Date_time <= @Dat_e
                UNION ALL
  SELECT    P_d.Id_doc, P_d.Id_go, P_d.Kol  FROM     Prihod2_head P_h
     INNER JOIN    Prihod2_details P_d ON P_h.Id_doc = P_d.Id_doc
         WHERE     P_h.Date_time >= @Dat_b AND P_h.Date_time <= @Dat_e
```

```
                UNION ALL
  SELECT    P_d.Id_doc, P_d.Id_go, P_d.Kol   FROM    Prihod3_head P_h
      INNER JOIN   Prihod3_details P_d ON P_h.Id_doc = P_d.Id_doc
         WHERE   P_h.Date_time >= @Dat_b AND P_h.Date_time <= @Dat_e )
           AS Itog_prihod (Id_doc,Id_go,Kol) Group by Id_go
/*Обновление Qz(tc) на основании данных по итоговому приходу*/
 UPDATE    Sklad SET    Qe = Sklad.Qb + ITOG.Kol
        FROM  Sklad INNER JOIN   ITOG ON Sklad.Id_go = ITOG.Id_go
DELETE FROM ITOG
        INNER JOIN  Sklad ON ITOG.Id_go = Sklad.Id_go
 INSERT SKLAD (Id_go,Id_prc,Qb,Qe) SELECT Id_go,Id_go,Kol,Kol  FROM ITOG
IF OBJECT_ID('dbo.ITOG') IS NOT NULL DROP table dbo.ITOG
/* Расчет итогового расхода по всем источникам за интервал [Dat_n, Dat_b] */
SELECT   Id_go, SUM(Kol) AS Kol Into dbo.ITOG FROM
      (SELECT  P_d.Id_doc, P_d.Id_go, P_d.Kol FROM  Rashod1_head P_h
         INNER JOIN  Rashod1_details P_d ON P_h.Id_doc = P_d.Id_doc
         WHERE    P_h.Date_time >= @Dat_b AND P_h.Date_time <= @Dat_e
                UNION ALL
  SELECT    P_d.Id_doc, P_d.Id_go, P_d.Kol FROM   Rashod2_head P_h
        INNER JOIN  Rashod2_details P_d ON P_h.Id_doc = P_d.Id_doc
         WHERE     P_h.Date_time >= @Dat_b AND P_h.Date_time <= @Dat_e
              UNION ALL
  SELECT    P_d.Id_doc, P_d.Id_go, P_d.Kol  FROM   Rashod3_head P_h
        INNER JOIN   Rashod3_details P_d ON P_h.Id_doc = P_d.Id_doc
        WHERE   P_h.Date_time >= @Dat_b AND P_h.Date_time <= @Dat_e )
       AS Itog_rashod (Id_doc,Id_go,Kol) GROUP BY Id_go
/* Обновление Qz(tc) на основании данных по итоговому расходу */
 UPDATE    Sklad SET    Qe = Sklad.Qe - ITOG.Kol
        FROM   Sklad INNER JOIN   ITOG ON Sklad.Id_go = ITOG.Id_go
SET STATISTICS TIME OFF
```

Полученная по результатам эксперимента зависимость, помеченная на рис. 2 символом ***"B"***, показывает, как растет время ожидания на получение $Q_z(t_c)$ в зависимости от числа записей, которые включаются в расчет. С каждым днем их количество увеличивается на $\boldsymbol{6 \cdot 10^4}$, что соответствует одному делению на оси абсцисс. (Согласно значениям входных параметров: $\boldsymbol{3 \cdot 10^4}$ – среднесуточная интенсивность приходных операция, $\boldsymbol{3 \cdot 10^4}$ – среднесуточная интенсивность расходных операций).

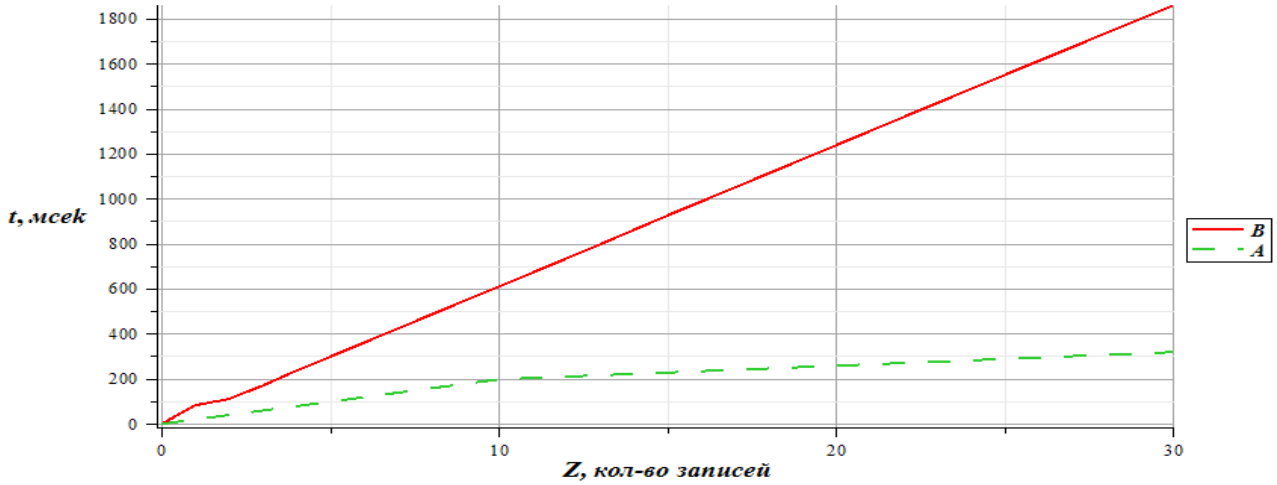

Рис. 2. Зависимости продолжительности выполнения *SQL*-запросов как функции от мощностей приходно-расходных и "складского" отношений.

Кривая ***"A"*** демонстрирует зависимость продолжительности выполнения ***Select-****инструкции* на выборку $Q_z(t_c)$ из "складской" структуры как функцию от

числа записей, содержащихся в последней.

Даже для значительных объемов информации, достигающих $9 \cdot 10^5$ за первые ***15*** дней расчетного периода, продолжительность составляет всего одну секунду, что на первый взгляд может показаться несущественным. Но в многопользовательских приложениях ожидание на получение $Q_z(t_c)$ будет возрастать многократно и достигать своих пиковых значений в моменты времени параллельного выполнения транзакций, запрашивающих $Q_z(t_c)$. Именно последнее соображение делает целесообразным хранение актуализированных значений $Q_z(t_c)$ в специализированных таблицах базы данных.

Сложные виды избыточности данных. Синтетические типы

До сих пор нами рассматривались простейшие виды избыточности, которые, как оказалось, целесообразно вводить в структурную часть баз данных для увеличения производительности выполнения *SELECT*-конструкций. Было показано, что избыточными могут быть как отдельные атрибуты, так и целые структуры. В свою очередь избыточные атрибуты могут содержать или “первичные” данные, или вычисляемые значения (“вторичные”, “производные” данные).

Кроме всего перечисленного, в модели данных могут быть включены структуры, в которых размещается информация, касающаяся объектов особого рода. Это, по аналогии с атрибутами, “вторичные” объекты. Характерный и, пожалуй, самый распространенный объект, принадлежащий к классу “вторичных”, – “бухгалтерские проводки”. Именно этот объект и будем далее использовать в качестве примера.

“Бухгалтерские проводки“ – тип сущности, которого в реальном мире не существует. По сути – это ***синтетический тип***, экземпляры которого формируются в соответствии с наперед заданным алгоритмом. Для начала выделим этот синтетический тип, проводя анализ на объектный состав одного из, пожалуй, самых распространенных управленческих документов – счета-фактуры. Но перед этим предварительно выполним одно действие – искусственно включим в счет-фактуру дополнительные атрибуты (“приписанные” атрибуты), в роли которых выступят идентификаторы бухгалтерских счетов. Именно так поступают (добавляя счета) бухгалтеры, формируя проводки. Подсхема модели данных, демонстрирующая соответствующие документальные и все необходимые справочные структуры данных, которые потребуются для проведения анализа, приведена на рис. 3.

В составе содержательной структуры присутствуют два атрибута: “Количество” и “Ставка НДС”, классифицируемые как “данные пересечения”. Полученное отношение находится в пятой нормальной форме, в чем при желании нетрудно убедиться, и формально является “слабой сущностью”.

Ситуация кардинально меняется после включения в схему соответствующего отношения атрибутов “***Id_ac(tmc)***” и “***Id_ac(nds)***”. Между парами атрибутов **“*Id_tmc*”**-“***Id_ac(tmc)***” и **“*Id_tmc*”**-“***Id_ac(nds)***” устанавливаются функциональные зависимости, которые переводят отношение во вторую нормальную форму. Набор функциональных зависимостей между атрибутами отношения будет теперь

включать: ***Id_doc, Id_tmc*, *Цена → Количество, Id_doc, Id_tmc*, *Цена → Ставка НДС*, *Id_tmc→Id_ac(tmc)*, *Id_tmc→Id_ac(nds).*** Если строго придерживаться теории нормальных форм, то структура (схема отношения) с таким набором атрибутов некорректна и не может присутствовать в модели данных. Тем не менее подобная конструкция оказывается вполне жизнеспособной, так как восполняет потребность в данных, необходимых для формирования проводок. Попутно заметим, что частичная нормализация отношений допустима при условии, что эти отношения участвуют в формировании отношений, принадлежащих к классу синтетических.

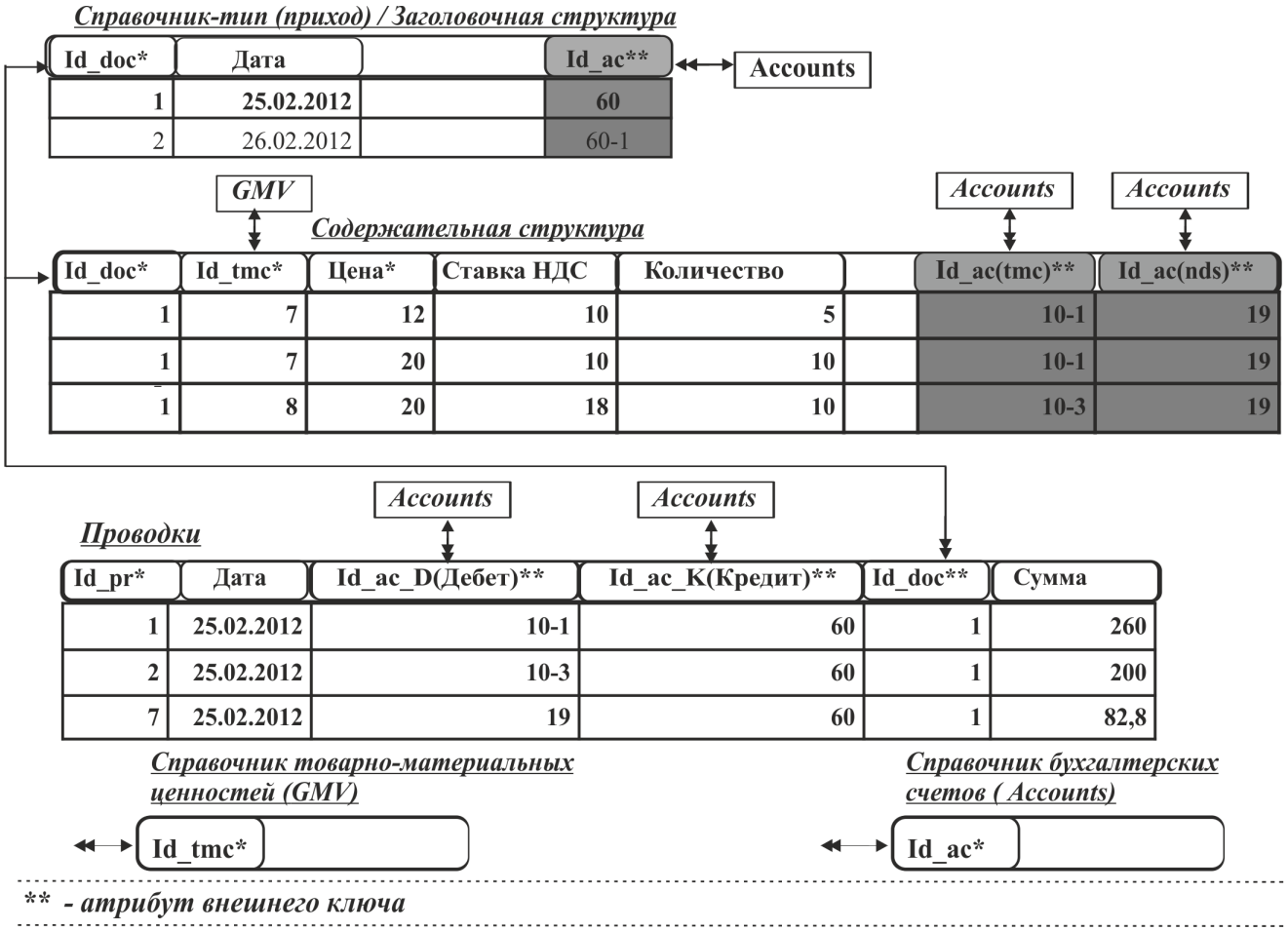

Рис. 3. Механизм формирования синтетического типа сущности и его экземпляров.

Определимся с отличительными признаки синтетического типа, которые в дальнейшем позволяли бы отличать его от “первичного”, оригинального типа. Для этого рассмотрим алгоритм, в соответствии с которым на основании содержащейся в таблицах информации (рис. 3), формируются проводки.

Бухгалтерский учет определяет “проводку” как “документальное оформление корреспонденции счетов при учете хозяйственных, финансовых операций с указанием дебетуемого и кредитуемого счета и суммы операции” [3].

Нетрудно догадаться, что атрибуты “*Id_ac*”, присутствующие в заголовочной и содержательной структурах, выполняют функцию указателей соответственно на дебет и кредит конкретной проводки. Алгоритм формирования проводок несложен и выглядит следующим образом:

запись заголовочной структуры “задает” кредит счета (значение, которое принимает “приписанный” атрибут – “*Id_doc”*);

счета дебетов определяются на основании значений “приписанных” атрибутов содержательной структуры. Например, данные первой записи дадут следующие две проводки: “$К_{60} – Д_{10\text{-}1}$” и “$К_{60} – Д_{19}$”;

значение каждой проводки $S_{K_{ac} - Д_{ac\_tmc}}$ и $S_{K_{ac} - Д_{ac\_nds}}$ легко найти, выполнив несложные арифметические действия:

$$S_{K_{ac} - Д_{ac\_tmc}} = \sum_{i:=\{t_i(Id\_ac(tmc))=Д_{ac\_tmc}\}} t_i(Цена) \cdot t_i(Количество), \quad (2)$$

$$S_{K_{ac} - Д_{ac\_nds}} = \sum_{i:=\{t_i(Id\_ac(nds))=Д_{ac\_nds}\}} (t_i(Цена) \cdot t_i(Количество))/100 \cdot t_i(Nds). \quad (3)$$

Здесь t_i (A_k) – значение, принимаемое атрибутом A_k, i-го кортежа t отношения (“содержательная структура”); $К_{ac}$, $Д_{ac}$ – значения счетов кредита и дебета соответственно.

Перечислим отличительные черты полученного синтетического типа.

Избыточность. Очевидно, что синтетический тип – это обобщение его отдельных экземпляров, которые, в свою очередь, образуются в результате преобразования экземпляров других типов, не обязательно первичных, в соответствии с наперед заданным алгоритмом.

Обладает системообразующим свойством “слабой сущности” – может существовать только при условии существования типов, принимающих участие в его синтезе, и включать атрибуты, интерпретируемые как “данные пересечения” (в нашем случае таким атрибутом является атрибут “сумма”).

В отличие от “слабых сущностей” способен вступать во взаимодействие с другими типами сущностей, обеспечивая их жизненные циклы и реализуя собственный жизненный цикл. Следствием сказанного будет использование структуры $S_2^M = (A^S, A^I, \hat{A})$ для его представления в модели данных, которая включает атрибут простого первичного ключа A^S, множество информационных атрибутов A^I, множество атрибутов внешних ключей $\hat{A}$.

Требует включения в состав структуры, его представляющей, обязательного атрибута, отражающего источник происхождения синтетического типа. В нашем случае таким атрибутом будет атрибут “*Id_doc”*.

В том, что этими свойствами обладают все без исключения синтетические типы, можно убедиться, если провести проверку на наличие этих свойств у любого из них. При этом, справедливости ради, мы не можем назвать ни одного подобного вторичного или третичного типа, хотя они, что весьма вероятно, существуют.

Для корректной идентификации синтетического типа можно ограничиться только двумя из четырех перечисленных свойств: “слабой сущности” и участия во взаимодействиях с другими типами. Приведем пример подобного взаимодействия. Этот пример, кроме того, понадобится в дальнейшем при изложении вопросов избыточности вычисляемых значений.

Во всех современных информационных системах в обязательном порядке

решается задача “синхронизации” финансовых и материальных потоков. Дело в том, что всякий материальный объект, кроме того, что может иметь одну или несколько единиц измерения (что требует отдельного рассмотрения), обладает еще и стоимостью. “Финансовая, денежная” сущность может существовать как в отрыве от своего материального объекта, так и находиться с ним в неразрывной связи. Собственно говоря, именно эти обстоятельства приводят к возникновению самой задачи синхронизации. Вот как примерно она может быть сформулирована: *“необходимо средствами структурной части модели данных показать, какие счета организации оплачены и на какую сумму, а какие нет”*. Модель данных, “отвечающая” на поставленный вопрос, демонстрируется на рис. 4.

Проводки

Id_pr*	Дата	Id_ac_D** (Дебет)	Id_ac_K** (Кредит)	Id_doc**	Сумма	“Незакрыто”
278	25.02.2012	60	51	-	1000	0
281	25.02.2012	60	51	-	3800	0
408	25.02.2012	60-1	51	-	780	680

“Связанные суммы”

Id_pr*	Id_doc*	“Связанная сумма”
278	1	1000
281	1	1500
281	2	2300
408	2	100

Заголовочная документальная структура

Id_doc*	Дата		Обороты по дебету	Обороты по кредиту
1	20.02.2012		2500	2500
2	21.02.2012		2400	2400

Рис. 4. Структуры подсхемы модели данных для хранения информации о синхронизации финансовых и материальных потоков.

В отношении “Связанные суммы” сосредоточены записи, показывающие, какая часть суммы конкретной “финансовой” проводки (проводки, у которой один из счетов – денежный счет) “пошла” на оплату конкретного счета. Так, например, сумма в 3800 денежных единиц (д.е.), соответствующая проводке *Id_pr* = 281, распределилась в следующей пропорции: 1500 д.е. пошли на оплату счета с *Id_doc* = 1, а 2300 д.е. – на оплату счета *Id_doc* = 2.

Подведем промежуточные итоги:

исследование сложной формы избыточности данных потребовало выделить в особую категорию типы сущностей, отличающихся от других и своим происхождением, и набором характерных свойств. Такие типы были названы “синтетическими”. Как и первичные типы, синтетические вступают в стандартные взаимодействия с другими типами, что в дальнейшем, в частности, позволяет моделировать их динамику точно таким же способом, каким моделируется динамика несинтетических, первичных типов;

обязательность присутствия в моделях данных таких типов объясняется не только целесообразностью быстродействия выполнения *Select*-запросов, но и необходимостью представления взаимодействий синтетических и несинтетических типов.

Избыточность вычисляемых значений

Теми же соображениями, о которых речь шла выше – снижением нагрузки на вычислительные мощности и минимальным ожиданием получения запрашиваемых данных, – вызвана необходимость в постоянном хранении актуальных (соответствующих определенному моменту времени, как правило, текущему) “вычисляемых” данных. Примеров, когда в схемы отношений на вполне обоснованных основаниях включаются “вычисляемые” атрибуты, можно привести множество. В счетах-фактурах, например, принято фиксировать обороты по дебету и кредиту документа, в отношениях, содержащих сведения о плановых показателях, – данные об их выполнении, в каждый кортеж, соответствующий основному средству, добавлять текущую остаточную стоимость или величину текущего износа и т.д.

Ввиду активного “использования” вычисляемых полей было бы желательно выработать некоторый набор соглашений, следование которым исключало бы появление в базах данных одних “вычисляемых” значений и допускало включение других. Кроме выигрыша в производительности, это также сделало бы возможным реализовать тотальный формализованный контроль достоверности “вычисляемых” данных посредством расширения набора стандартных каскадных процедур. Такие правила можно сформулировать.

Если задаться вопросом найти то общее, что присуще всем “вычисляемым” значениям (к которым, в частности, будет относиться и рассмотренный ранее “количественный” атрибут складской структуры), то вероятнее всего ответ будет следующим. Все “вычисляемые” атрибуты призваны отразить текущие *состояния* сущностей предметной области. Набор таких состояний конечен и ограничен и включает согласно [1]: местоположения экземпляров сущностей, действия, которые экземпляры выполняют и (или) которые совершаются над ними в местах их размещения (а для таких специфических действий как агрегирование/деагрегирование – текущие состояния экземпляров-полуфабрикатов), текущие значения совокупности показателей, отражающих различные аспекты стоимостных оценок экземпляров. Понятно, что для любого “вычисляемого” атрибута должен быть задан алгоритм получения его значений и иметься в наличии все исходные данные для проведения расчетов.

Некоторые “вычисляемые” атрибуты могут быть функционально связаны с другими “вычисляемыми” атрибутами и требовать задания последовательности вычислительных алгоритмов. В качестве примера можно привести такой часто встречающийся в базах данных “вычисляемый” атрибут как *задолженность*.

Задолженность (дебиторская или кредиторская) может появляться только у двух типов сущностей: личностей или организаций. Механизм ее формирования наглядно представлен на рис. 5. Для того, чтобы получить сальдо по организации

или личности, сначала понадобится произвести расчет сальдо по каждому приходно-расходному документу (этап II), предварительно указав, какие “денежные” суммы пошли на погашение задолженности по тому или иному документу (этап I), и только затем, используя полученные данные, получить значения искомых атрибутов (этап III).

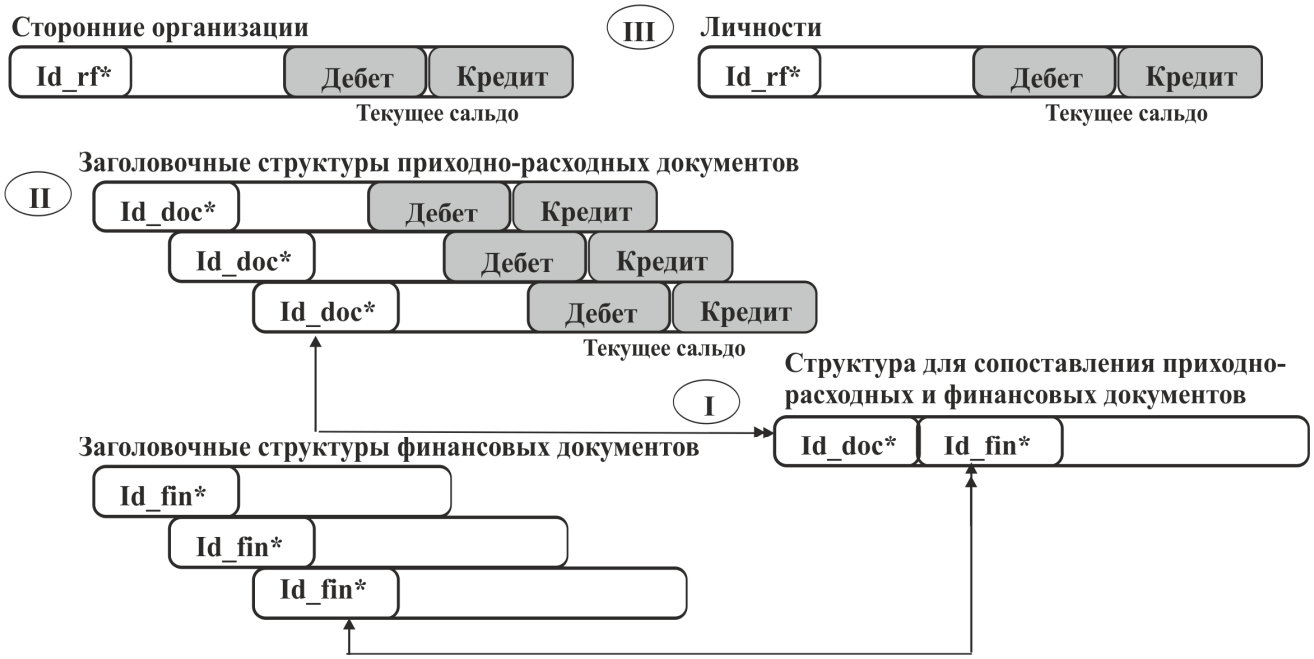

Рис. 5. Взаимосвязанные “вычисляемые” атрибуты в подсхеме модели данных.

Резюмируя сказанное, запишем условие целесообразности включения “вычисляемых” атрибутов в структуры баз данных:

$$E_{A_i^C} = \begin{cases} 1, M(D^C_{A_i^C}) >> M(D^D_{A_i^C}) \wedge A_i^C \in A^S, \\ 0, M(D^C_{A_i^C}) >> M(D^D_{A_i^C}) \wedge A_i^C \in A^S, \end{cases} \quad (4)$$

где $E_{A_i^C}$ – целесообразность, принимающая значения либо 0, либо 1; A_i^C – i-й “вычисляемый атрибут”; A^S – множество атрибутов, отражающих состояние экземпляров сущностей предметной области; $M(D^C_{A_i^C})$ – математическое ожидание продолжительности получения текущего значения вычисляемого атрибута; $M(D^D_{A_i^C})$ – требуемое значение математического ожидания.

Вне сомнения, что представленное соотношение дает весьма “грубую” оценку, так как не учитывает, что $M(D^C_{A_i^C})$ зависит от числа записей, которые подлежат обработке, и это число увеличивается со временем. Очевидно, что получение более точных оценок $E_{A_i^C}$ требует обязательного математического моделирования, что может составить предмет другого исследования. В настоящей работе мы ограничились только постановкой задачи и сделали это в самой общей форме.

Заключение

В статье системно изложены вопросы “разумной” избыточности данных, которая допустима в базах данных, когда нужно добиться минимальной продолжительности выполнения *Select*-конструкций. В некоторых случаях такая избыточность может приводить к возникновению семантической неопределенности – как, например, в случае с избыточностью кортежей, в некоторых – к денормализации отношений, когда формируются экземпляры синтетических типов. Главный результат проделанной работы – не сама избыточность, а *управление избыточностью,* которое должно строиться на четком понимании того, какие структурные организации, обеспечивающие поддержку избыточности, допустимы в моделях данных, а какие нет.

Полученные результаты, позволяют решить любые вопросы, связанные с управлением избыточностью, включая планирование (проектирование структур), проведение контроля достоверности избыточных данных (посредством разработки и реализации специализированных каскадных процедур баз данных), регулирование, которое сводится к “безболезненным” добавлениям или удалениям элементов структурной части баз данных.

Кроме этого, оказалось, что избыточность, присутствующая в “производных” документах (документах, содержащих продублированные или вторичные данные), может использоваться для управления последовательностями решения задач автоматизации. Другое приложение избыточности – участие в решении проблемы однозначного преобразования типов сущностей предметной области в структуры баз данных. Обозначенные вопросы могут и должны составить направления дальнейших исследований в этой области.

ЛИТЕРАТУРА

1. *Родионов А. Н.* Качество даталогических схем. Принцип минимальной избыточности и минимально-избыточные даталогические конструкции. I // Информатика и системы управления. – 2012. – №3 (33). – С. 3-13
2. *Райзберг Б.А., Лозовский Л.Ш., Стародубцева Е.Б.* Бухгалтерские проводки. – М.: ИНФРА-М, 2006.
3. *Родионов А. Н.* Критерии качества даталогических схем. Полнота моделей данных // Вестник ХГАЭП. – 2011. – №2 (53). – С. 28-51.

Статья представлена к публикации членом редколлегии А.Д. Плутенко.

E-mail:
Родионов Александр Николаевич. – ran@newmail.ru.