УДК 004.4

© 2012 г. **А.Н. Родионов**, д-р техн. наук
(Вычислительный центр ДВО РАН, Хабаровск)

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

В первой части работы вводятся минимально-избыточные структуры баз данных, предназначенные для хранения актуальных значений однозначных и многозначных переменных свойств сущностей. Представлены результаты выполнения стандартных *Select*-конструкций, доказывающие целесообразность дублирования данных, принадлежащих справочным объектам. Исследован ряд структурных типов, являющихся источниками избыточных кортежей. Предложены решения по реструктуризации таких структур, исключающие появление “лишних” кортежей в базах данных.

***Ключевые слова:*** модель данных, избыточность структур данных, кортеж, свойство, ограничение целостности, однозначные и многозначные свойства, нормализация и нормальные формы отношений.

## Введение

Один из основополагающих принципов баз данных утверждает, что данные не должны дублировать друг друга и быть производными (вытекать друг из друга). Но часто этот принцип нарушается [1 – 3]. В каких случаях эти нарушения являются обоснованными, а в каких ими можно пренебречь, в каких случаях они жизненно необходимы, а в каких противопоказаны и выполняют разрушительную роль? Все эти вопросы ставятся и решаются в статье.

В настоящей работе избыточность не трактуется как избыточность, порождаемая ненормализованными отношениями [2, 10]. Предполагается, что нормализация проведена и все схемы отношения соответствуют пятой нормальной форме. Избыточность, о которой далее пойдет речь, это избыточность и отдельных атрибутов, и отдельных кортежей, и целых отношений. Но избыточность, обусловленная другими обстоятельствами. Одно из них – производительность баз данных, другое – однозначность семантической интерпретации данных.

Виды и “конфигурации” избыточности напрямую зависят от совокупности принципов, выполняющих роль своеобразных ограничений, в соответствии с которыми ведется проектирование концептуальных схем данных. Все эти ограничения перечисляются в первой части статьи.

Во второй части приводится перечень отдельных категорий избыточных

данных, дается их краткая характеристика и классификация. Указывается основная проблема избыточности – опасность рассогласованности данных и перечисляются меры, направленные на решение упомянутой проблемы.

В третьей части работы рассматриваются структурные решения по обеспечению избыточности однозначных и многозначных переменных свойств сущностей. Приводятся данные экспериментов, доказывающие эффективность предлагаемых избыточных структур.

Четвертая часть посвящена исследованию такого явления как "*избыточность кортежей*". "*Лишние кортежи*", которые могут появляться в структурах, представляющих свойства, не позволяют провести однозначную семантическую интерпретацию соответствующих данных. Приводятся модифицированные схемы отношений, исключающие возникновение неопределенности подобного рода в базах данных.

## Концепция организации баз данных как системы последовательных двухуровневых отображений сущностей предметной области

Основное назначение фактографических баз данных информационных систем – фиксация состояний объектов предметной области, из которых, собственно говоря, эта предметная область и состоит. О том, что это за состояния и каким образом они представляются в табличных структурах моделей данных, подробно сказано в [4 – 9]. Обозначим наиболее важные и значимые для настоящей работы моменты, вытекающие из полученных ранее результатов:

1) каждому типу сущности предметной области соответствует от одного до нескольких даталогических объектов, принадлежащих либо к категории "справочники", либо к категории "группировки";

2) объекты в концептуальной схеме представляются посредством ограниченного набора предварительно типизированных структур, отличающихся друг от друга атрибутным составом;

3) объекты одного типа сущности образуют кластер, отличающейся собственными, принадлежащими только ему, моделями жизненных циклов (МЖЦ);

4) для синтеза объектов используется *DPSI-метод* (*Domain-Pattern-Sample-Instance method*), основу которого составляют алгоритм доменно-объектной идентификации/классификации и абстракции обобщения двух видов: "знаки→тип" и "типы→тип";

5) свойства объектов рассматриваются как некоторый результат, полученный в ходе взаимодействия двух или более объектов, принадлежащих разным кластерным группам;

6) экземпляры объектов фиксируются в справочных структурах двух разновидностей: справочниках-знаках и справочниках-типах;

7) непосредственно свойства объектов, представленные обособленными атрибутами, могут находить отражение в составе: справочных структур; структур, именуемых "слабыми сущностями"; структур, представляющих документы предметной области;

8) документальные структуры и "слабые сущности" являются взаимозаме-

няемыми структурами. При этом в цепочке отображений сущностей предметной области *сущность→объекты→структуры* документальная структура считается первичной по отношению к “слабой сущности”;

9) обязательными характеристиками сущностей следует считать их состояния и местоположение, которое они занимают в предметной области;

10) включение в состав модели данных справочников-типов может приводить к появлению “сцепленных” с ними структур особого рода – “складских структур”, основное назначение которых – хранение данных о количестве и местоположении объектов кластерного типа на заранее определенную дату. Такая дата представляет собой одну из обязательных констант информационной системы. К ней “привязываются” все состояния объектов (сущностей) предметной области;

11) абстрактная МЖЦ типа сущности предполагает указание всех потенциальных местоположений объектов, допустимых перемещений объектов между этими местоположениями и наборов действий, которые могут совершать объекты и совершаться над объектами как в этих местоположениях, так и в процессе перемещения между местоположениями.

Использование перечисленных положений, фактически играющих роль принципов, в соответствии с которым ведется проектирование базы данных, формирует комплекс условий для возникновения избыточности данных и типов (видов) этой избыточности.

Еще раз обратим внимание на то, что использование других принципов будет способствовать порождению отличного набора избыточных данных.

### Источник и виды избыточных данных

По сути, база данных могла бы обойтись без избыточных данных, поскольку все они либо дублируют первичные данные, либо являются производными от них. Но естественное стремление снизить время на поиск нужных данных заставляет ставить и каким-то образом решать вопрос относительно разумной избыточности данных.

Следование перечисленным выше принципам проектирования структур баз данных позволяет выделить несколько разновидностей избыточных данных и доказать их целесообразность. Часть видов можно отнести к простым видам, часть – к сложным. Вторые от первых отличает то, что они представляют собой сущности с явно выраженными собственными жизненными циклами (наличие как минимум одного жизненного цикла – отличительный признак сущности).

К простым видам, если руководствоваться указанным признаком, можно отнести: избыточность *свойств*, включая их атрибутную и структурную избыточности, избыточность *кортежей*, избыточность “вычисляемых” *значений* и избыточность, порождаемую *справочниками-типами*.

Целесообразность структурных решений, обеспечивающих поддержку перечисленных видов избыточности данных, подкрепим далее статистическими зависимостями, отражающими взаимосвязи между продолжительностью выполнения запросов на выборку данных и мощностью соответствующих “избыточных”

отношений.

Переходя далее к изложению структурных решений, нельзя не сказать и об обратной стороне избыточности.

Избыточность данных – это потенциальный источник их рассогласованности, нарушения целостности. Согласовать данные (сделать так, чтобы они удовлетворяли всем ограничениям, которые на них накладываются) в случае “согласия” на их избыточность можно только разрабатывая и используя в последующем специализированные программные модули. Такие модули должны входить в состав прикладного программного обеспечения, поскольку стандартные, штатные средства СУБД по вполне очевидным причинам (вытекающим из перечисленных выше признаков) не сориентированы на решение подобного рода задач. Понятно, что система прикладного программного обеспечения должна быть также снабжена соответствующими модулями мониторинга и восстановления данных в случаях, если последние окажутся рассогласованными. И это, пожалуй, – основной недостаток, порождаемый “избыточностью”.

Кроме того, несколько увеличивается длина программного кода, обрабатывающего первичные данные, поскольку в него требуется включать обязательные блоки, формирующие и записывающие вторичные данные в соответствующие табличные структуры базы данных.

## Избыточность свойств

Избыточность свойств – простейший вид избыточности данных. В работе [9] были представлены актуальные для целей даталогического моделирования системы классификации свойств. Согласно предложенным системам классификации следует проводить различие между однозначными и многозначными свойствами (классификационный признак – число значений, одновременно принимаемых свойством), а также постоянными, условно-постоянными и переменными свойствами (классификационный признак – изменяемость свойств во времени). Для хранения значений свойств в той же работе предлагается задействовать специализированные структуры, атрибутный состав которых задается в зависимости от класса свойства, определенного на основании комбинации упомянутых классификационных признаков. На эти структуры и будем далее ориентироваться, полагая их эталонными – неизбыточными и достаточными для отражения динамики свойств.

Наиболее “труднодоступными” свойствами (по отношению к времени, затрачиваемому на извлечение принимаемых ими значений из базы данных) оказываются свойства, отнесенные к парам: “многозначные-переменные” (*МП*), “однозначные-переменные” (*ОП*), поскольку все они “привязаны” к датам, на которые фиксируются их текущие значения. Очевидно, что актуальными (востребованными) значениями окажутся те, которые будут иметь свойства в текущий момент времени. Ретроспективные значения, что естественно, востребованы в гораздо меньшей степени.

Покажем, за счет каких структурных перестроек, приводящих к избыточности данных, может быть снижено время доступа к актуальным, действующим на

текущий момент времени значениям таких свойств.

Сравнительно просто решается вопрос с представлением текущего значения “однозначного, переменного” свойства (рис. 1). Для этого достаточно включить в состав справочной структуры (вне зависимости от того, будет ли эта справочная структура представлять собой справочник-знак или справочник-тип) атрибут, содержащий значение, которое свойство принимает в текущий момент времени. При этом должно выполняться следующее явное ограничение целостности.

Атрибут “Значение свойства” структуры “Справочник–тип (знак)” содержит либо “null-значение”, либо равен $p_{ij}(t)$, где $i$ – индекс свойства $j$-й сущности; $t$ – момент времени, на которое зафиксировано значение свойства; $t > t_k$, где $t_k$ – моменты времени, на которые ранее регистрировались значение свойства $p_{ij}$.

Такую избыточность, обеспечиваемую посредством дополнительного атрибута, далее будем именовать “атрибутной избыточностью”.

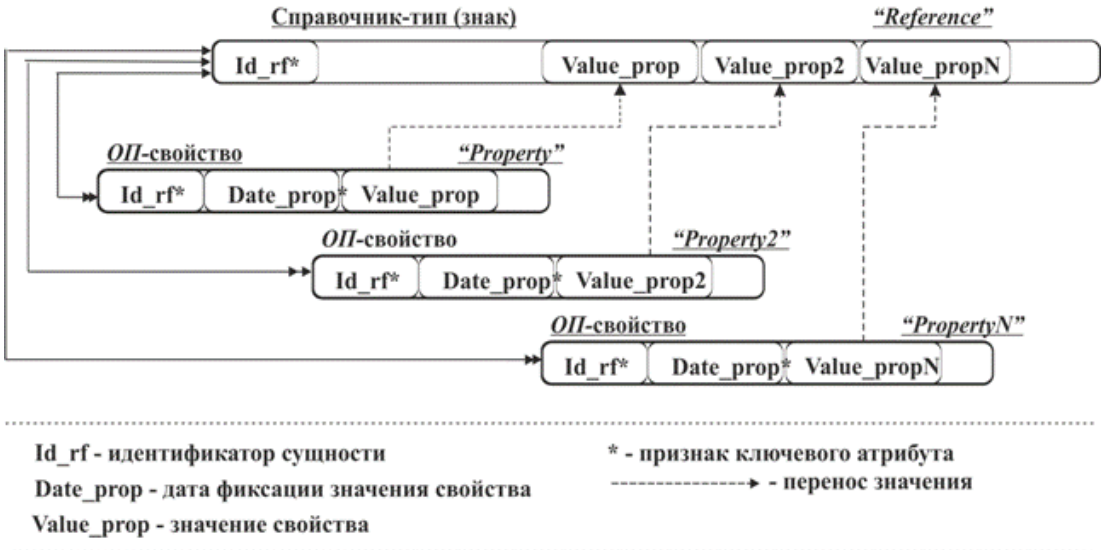



*Рис. 1.* Избыточность атрибута в представлении “однозначного-переменного” свойства.

Насколько эффективна такая конструкция (рис. 1), можно судить по графикам, представленным на рис. 2. Одно деление на оси абсцисс соответствует $10^5$ записей, содержащихся в ***Reference***-структуре, и по $10^6$ записей в каждой ***Properties***-структуре.

Зависимости, помеченные символом *“A”*, отражают продолжительность выполнения ***Select***-конструкций, осуществляющих поиск текущих значений свойств в ***Properties***-структурах как функцию от суммарной мощности отношений: ***“Reference”*** и одной, двух или трех ***“Properties”***.

Формулировка запросов на поиск соответственно одного, двух и трех значений свойств в базе данных, созданной в среде MS SQL (2008), выглядит следующим образом (листинги 1, 2 и 3 соответственно).

***Листинг 1***

```
SELECT  R.Id_rf, P.Value_prop
FROM    Property P INNER JOIN
        (Select P.Id_rf,MAX(Date_prop) As Dat From Property P Group By P.Id_rf) As Nvp (Id_rf,Dat)
      ON P.Id_rf = Nvp.Id_rf AND P.Date_prop = Nvp.Dat
       RIGHT OUTER JOIN
   Refrence R ON Nvp.Id_rf = R.Id_rf
```

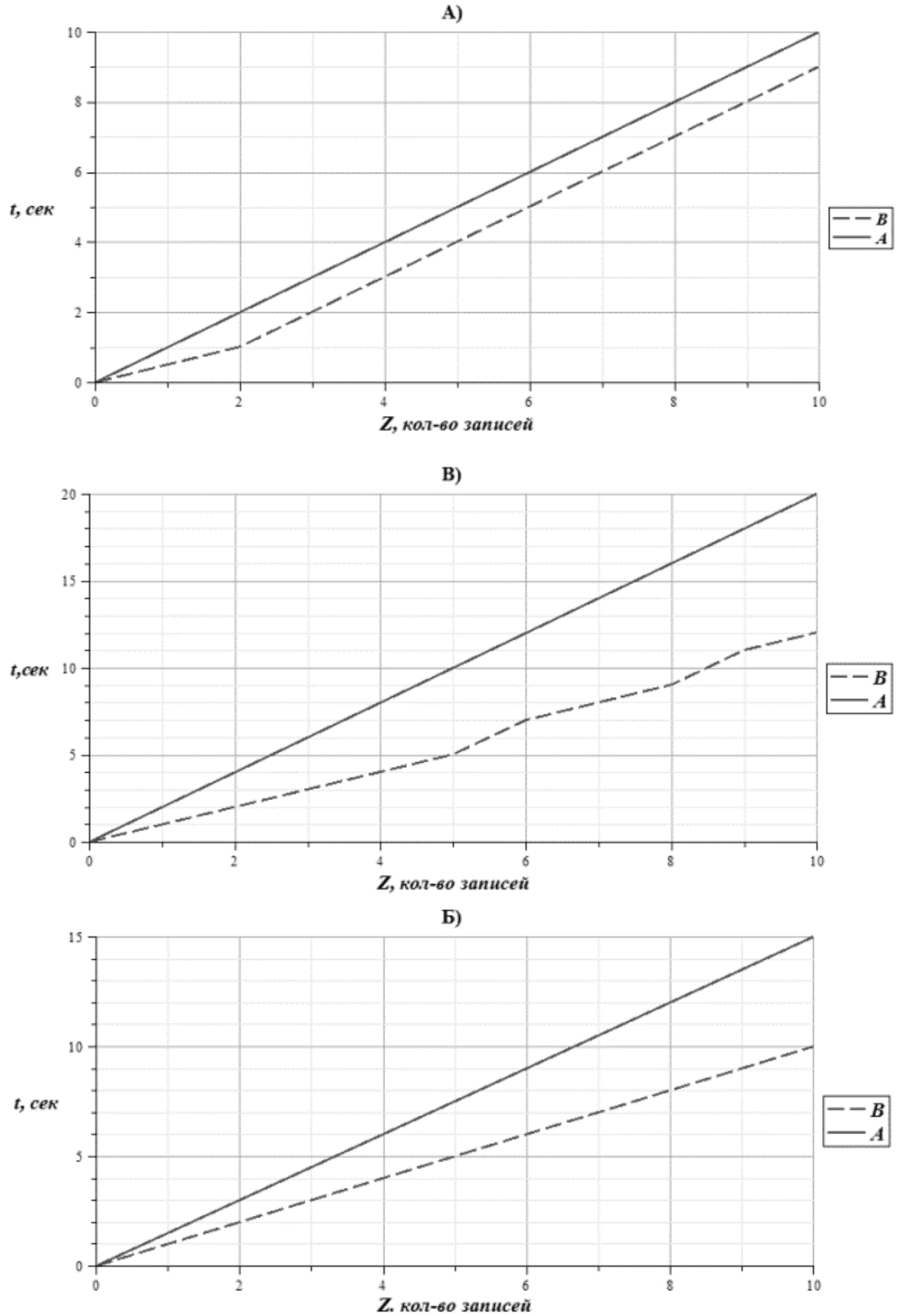



*Рис. 2.* Зависимости продолжительности выполнения *SQL*-запросов от суммарной мощности *Reference*-отношения и мощностей: а) одной; б) двух; в) трех *Property*-структур.

***Листинг 2***

```
SELECT     R.Id_rf, P2.Value_prop As Value_prop2, P.Value_prop AS Value_prop
FROM       (Select P.Id_rf,MAX(Date_prop) As Dat      From Property P          Group By P.Id_rf) As Nvp
(Id_rf,Dat)
                  INNER JOIN
               Property P ON Nvp.Id_rf = P.Id_rf AND Nvp.Dat = P.Date_prop
                  RIGHT OUTER JOIN
               Refrence R LEFT OUTER JOIN
            (Select P2.Id_rf,MAX(Date_prop) As Dat From Property2 P2 Group By P2.Id_rf) As Nvp2
(Id_rf,Dat)
                  INNER JOIN
             Property2 P2 ON Nvp2.Id_rf = P2.Id_rf AND Nvp2.Dat = P2.Date_prop ON R.Id_rf = Nvp2.Id_rf
ON
                Nvp.Id_rf = R.Id_rf
```

***Листинг 3***

```
SELECT     R.Id_rf, P.Value_prop, P2.Value_prop AS Value_prop2, P3.Value_prop AS Value_prop3
FROM       (Select P.Id_rf,MAX(Date_prop) As Dat      From Property P          Group By P.Id_rf) As Nvp
(Id_rf,Dat)
          INNER JOIN  Property P ON Nvp.Id_rf = P.Id_rf AND Nvp.Dat = P.Date_prop
          RIGHT OUTER JOIN  Property3 P3
          INNER JOIN
           (Select P3.Id_rf,MAX(Date_prop) As Dat From Property3 P3 Group By P3.Id_rf) As Nvp3
(Id_rf,Dat)
          ON P3.Id_rf = Nvp3.Id_rf AND P3.Date_prop = Nvp3.Dat
         RIGHT OUTER JOIN  Refrence R ON Nvp3.Id_rf = R.Id_rf
          LEFT OUTER JOIN  (Select P2.Id_rf,MAX(Date_prop) As Dat From Property2 P2 Group By
P2.Id_rf) As              Nvp2(Id_rf,Dat)
         INNER JOIN  Property2 P2
         ON Nvp2.Id_rf = P2.Id_rf AND Nvp2.Dat = P2.Date_prop ON R.Id_rf = Nvp2.Id_rf ON  Nvp.Id_rf =
R.Id_rf
```

Зависимости "*B*" получены в предположении того, что текущие значения свойств дублируются в ***Reference***-структурах посредством атрибутов-свойств. (соответствующие *SQL*-запросы в виду их простоты и очевидности не приводятся).

Обратим внимание на то, что структурная сложность запросов увеличивается по мере роста числа свойств, что создает определенные трудности в конструировании и отладке таких запросов.

Все зависимости, показанные на графиках (см. рис. 2), получены в результате выполнения представленных запросов на компьютере со стандартными характеристиками: процессор Intel Celeron 2.5 GHz, ОЗУ 4ГБ, жесткий диск емкостью 250 ГБ.

Как нетрудно заметить, продолжительность выполнения запросов не является сколько-нибудь значительной для справочного отношения с одним или несколькими динамическими свойствами, мощность которого превышает даже $10^5$ записей. Но в многопользовательских приложениях она увеличится как минимум в разы и как максимум – на порядки. Заметим, что именно для таких приложений атрибутная избыточность – одно из эффективных решений повышения их производительности.

То же самое справедливо и в отношении справочных структур, содержащих большее число *ОП*-свойств. Закономерно, что увеличение количества последних увеличивает и время ожидания на получение текущих значений свойств. Из представленных графиков видно, что при этом возрастает значение $|t_i - t_r|_M^N$, где $t_i$ и $t_r$ продолжительности выполнения запросов в неизбыточных и избыточных подсхемах, соответственно $M$ – мощность справочного отношения и $N$ – количество *ОП*-свойств. Например, $|t_i - t_r|_{3\cdot10^5}^2 = 1.5$, а $|t_i - t_r|_{3\cdot10^5}^3 = 3$.

При включении дополнительных атрибутов в справочные структуры видоизменяется формальная схема справочного кластера *e-й* сущности. Справочная структура $S_1^M = (\bar{a}, A^I)$, где $\bar{a}$ – атрибут первичного суррогатного ключа; $A^I$ – множество информационных атрибутов [5], приводимых к виду: $S_1^M = (\bar{a}, A^I, V)$. Здесь $V = (v_1, v_2, v_k, v_K)$ – множество значений *ОП*-свойств *e-й* сущности. Соответственно семейство $S_6^E = (A^C, \tilde{A})$, используемых для хранения динамики свойств, конкретизируется как: $A^C = \{\bar{a}, \tilde{a}^d\}$, где $\tilde{a}^d$ – атрибут, принимающий значение из домена “Даты”; $\tilde{A} = \{a^c\}$ – атрибут, значение которого – “данные пересечения” (в нашем случае значение *k-го ОП*-свойства сущности).

При этом между элементами множеств $V$ и $\{S_6^E\}$, принадлежащих *e-й* сущности, устанавливается взаимно однозначное соответствие.

В отличие от *ОП*-свойства несколько сложнее решается задача фиксации текущего значения *МП*-свойства. Здесь не обойтись без включения в схему организации данных дополнительной структуры, в которой, как и в предыдущем случае, будут содержаться текущие значения свойства (рис. 3).

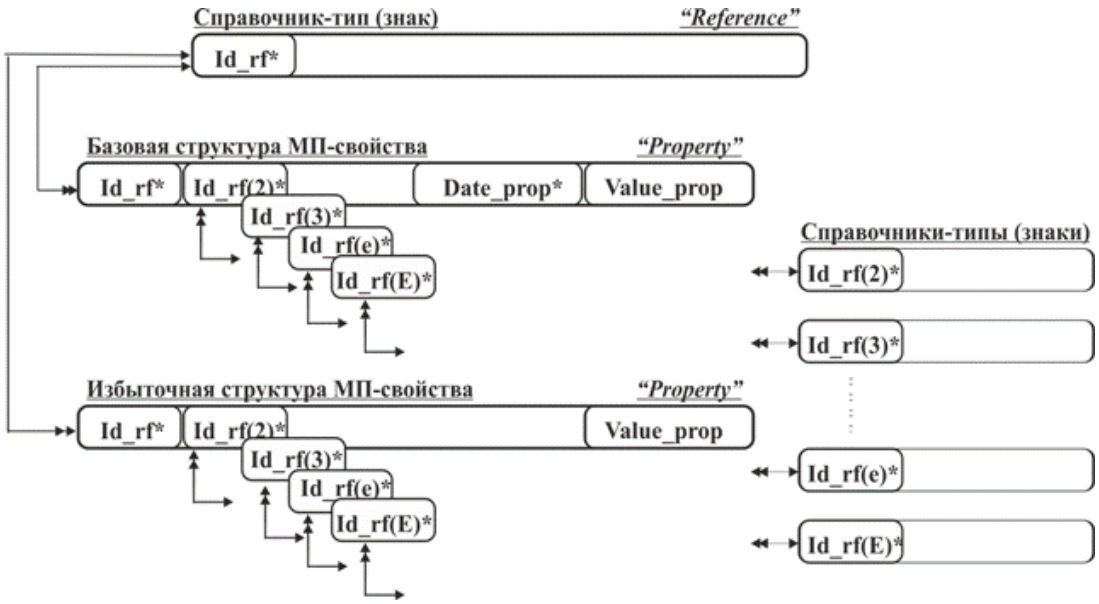



*Рис. 3.* Структурная избыточность в представлении *МП*-свойства.

Нетрудно заметить, что в этом случае избыточность данных сопровождается и избыточностью структур, в которых эти данные хранятся. Как и в случае с “однозначным-переменным” свойством, подтвердим эффективность избыточно-

сти данных, построив и выполнив *Select*-запросы на получение текущих значений свойств.

На графике (рис. 4) демонстрируется зависимость продолжительности выполнения запросов на выборку данных как функция от числа записей, хранящихся в справочной таблице (*References*) и одной таблице-свойстве (*Property*).

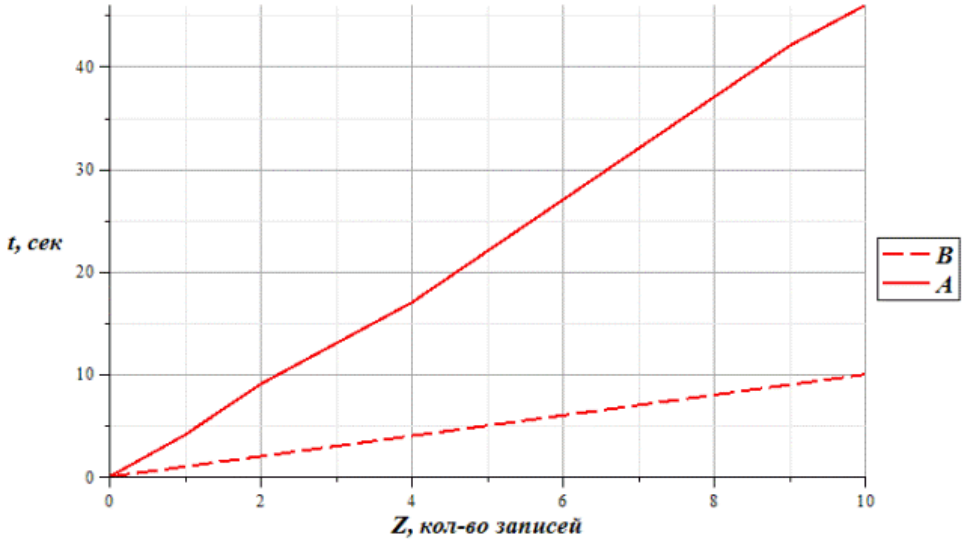



*Рис. 4.* Зависимость продолжительности выполнения *SQL*-запросов от суммарной мощности *Reference* и *Property* – отношений для одного *МП*-свойства.

Мощности соответствующих отношений, в которых производился поиск, те же, что и мощности *Reference* и *Property*-отношений для одного *ОП*-свойства.

Нет особой необходимости конструировать и выполнять запросы, манипулирующие данными двух и более *Property*-таблиц, поскольку эффективность использования избыточной конструкции очевидна. Так, по результатам эксперимента выигрыш во времени выполнения запроса на получение значений *МП*-свойства $10^5$ экземпляров справочных объектов при использовании только одной избыточной структуры составил 3 секунды, $2*10^5$ – 7 секунд и т.д., по возрастающей, что хорошо видно из графика, изображенного на рис. 5.

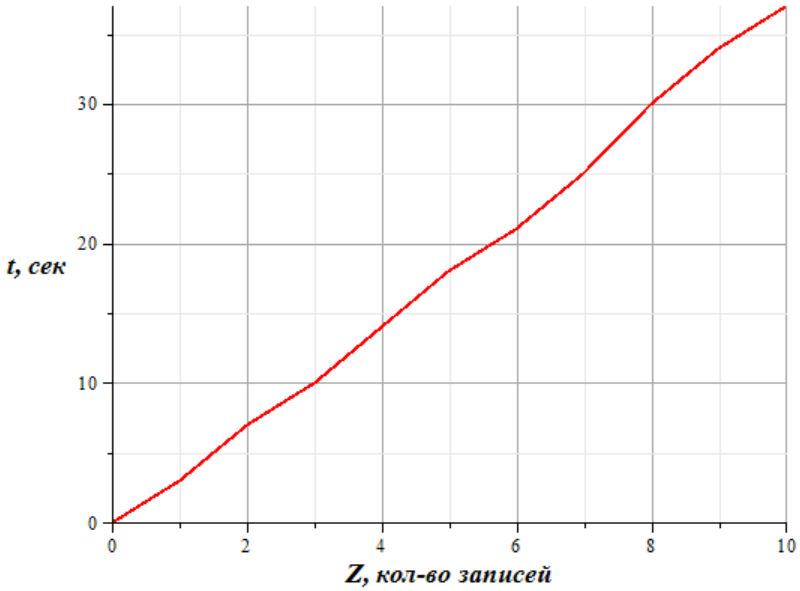



*Рис. 5.* Выигрыш в продолжительности поиска значений *МП*-свойства при использовании одной избыточной *Property*-структуры.

Формулировка запроса, использовавшегося для обращения к избыточной структуре *МП*-свойства, приводится в листинге 4.

***Листинг 4***

```
SELECT     P.Value_prop, R.Id_rf
FROM       Property_multy P INNER JOIN
      (SELECT   Id_rf, Id_rf2, Max(Date_prop) as Dat    FROM   Property_multy
         Group by Id_rf,Id_rf2) as Nvp (Id_rf,Id_rf2,Dat)
       ON P.Id_rf = Nvp.Id_rf AND P.Id_rf2 = Nvp.Id_rf2 AND   P.Date_prop = Nvp.Dat
 RIGHT OUTER JOIN
          Refrence R ON P.Id_rf = R.Id_rf
```

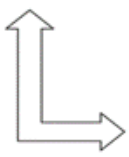
Нетрудно заметить, что формальная схема базовой структуры *МП*-свойства $S_6 = (\bar{a}\ , \overline{A}_I, \tilde{a}^d, \bar{a}^c)$ [5], где $\overline{A}_I$ множество наследуемых атрибутов первичного ключа, участвующих во взаимодействии, для представления избыточных значений *МП*-свойства трансформируется к виду: $S_6 = (\bar{a}\ , \overline{A}_I, \bar{a}^c)$.

## Избыточность кортежей

Использование базовой структуры $S_6$ для хранения значений “многозначного переменного” свойства может приводить к побочным, нежелательным эффектам – появлению “*избыточных кортежей*”. Рассмотрим следующую ситуацию.

Имеется подсхема (рис.6), предназначенная для отражения хобби (увлечений) личностей (справочники *личностей* и *увлечений* на схеме не показаны). Требуется найти текущие хобби личности с $Id\_prs = 1$. Вероятно, это будут бокс, шахматы и горные лыжи. А дайвинг? В отношении последнего ничего определенного сказать нельзя. Если дайвинг – одно из текущих увлечений личности, то кортеж с индексом “*I*” окажется лишним (избыточным). Если справедливо обратное, то избыточным будет кортеж с индексом “*B*”.

| | Id_rf* | Хобби | Дата |
|---|---|---|---|
| *A* | 1 | 1 (Дайвинг) | 01.01.2012 |
| *B* | 1 | 2 (Горные лыжи) | 01.01.2012 |
| *C* | 1 | 3 (Бокс) | 02.02.2012 |
| *D* | 1 | 4 (Шахматы) | 02.02.2012 |
| *I* | 1 | 2 (Горные лыжи) | 02.02.2012 |



| | Id_rf* | Хобби | Дата начала | Дата окончания |
|---|---|---|---|---|
| *A* | 1 | 1 (Дайвинг) | 01.01.2012 | |
| *B* | 1 | 2 (Горные лыжи) | 01.01.2012 | 15.01.2012 |
| *C* | 1 | 3 (Бокс) | 02.02.2012 | |
| *D* | 1 | 4 (Шахматы) | 02.02.2012 | |
| *I* | 1 | 2 (Горные лыжи) | 02.02.2012 | |

*Рис. 6.* Пример демонстрации избыточных кортежей.

Устранить возникшую неопределенность не составляет большого труда. Достаточно включить в состав соответствующей $S_6$-структуры атрибут, содержащий или дату, на которую прекращается действие определенного увлечения, или бинарный признак, выполняющий аналогичную функцию. Тогда вместо $S_6 = (\bar{a}\ , \overline{A}_I, \tilde{a}^d, \bar{a}^c)$ следует использовать $S_6 = (\bar{a}\ , \overline{A}_I, \tilde{a}_b^d, \tilde{a}_e^d, \bar{a}^c)$, где $\tilde{a}_b^d$ и $\tilde{a}_e^d$ – соответственно атрибуты даты начала и окончания действия свойства.

Теперь ни кортеж *"B"*, ни кортеж *"I"* не является избыточным. Кортеж *"B"* несет информацию о том, что в период с 01.01.2012 по 15.01.2012 личность с *Id_prs* = 1 увлекалась горными лыжами, а *"I"* утверждает, что с 02.02.2012 хоббинабор вновь пополнился горными лыжами. При этом, правда, возникает явное ограничение целостности в отношении, представляющем МП-свойство, не может появиться кортеж, если в том же отношении уже присутствует кортеж с теми же значениями $\bar{a}$ и $\overline{A}_I$, но с $\tilde{a}_e^d \notin D$.

Формальная схема избыточной *МП*-структуры в данном случае не меняется.

В отношении *ОП*-свойства также необходимо отразить факт завершения его действия. Для этого следует "снабдить" соответствующую $S_6$-структуру атрибутом $\tilde{a}_e^d$, посредством которого будет фиксироваться момент окончания действия свойства.

Добавление этого атрибута несколько усложняет представленные в виде листингов 1, 2 и 3 *SQL*-выражения и увеличивает продолжительность их выполнения за счет дополнительных сравнений, но никак не отражается на сделанном ранее выводе об эффективности избыточных структур, представляющих динамические свойства.

## Заключение

В первой части статьи рассмотрены простейшие виды избыточности данных, обеспечивающие повышение быстродействия выполнения *Select*-конструкций в структурированных и связанных массивах информации – базах данных. В известной степени избыточность – альтернатива нормализации, так как обе преследуют одну и ту же цель – сократить время доступа к данным.

В отличие от денормализации, приводящей к аномалиям вставки, удаления и обновления, "плата" за избыточность намного ниже и сводится к легкореализуемым дополнительным требованиям к двум классам задач.

Задачи оперативной обработки транзакций должны фиксировать избыточность в заранее предопределенных наборах данных. Задачи администрирования, в свою очередь, – снабжены функциями проверки актуальности и правильности таких данных.

Избыточность может возникнуть не только в результате включения в схемы организации данных избыточных атрибутов (как в случае с *ОП*-свойствами) или избыточных структур (как в случае с *МП*-свойствами), но и вследствие игнорирования требования обязательной фиксации моментов времени начала и окончания действия свойств. В этом случае в соответствующих отношениях вероятно появление "лишних кортежей". Сделанные в работе уточнения формальных схем

структур, представляющих *МП*-свойства, и рекомендации по использованию *null*-значений в избыточных атрибутах *ОП*-свойств исключают появление как самих избыточных кортежей, так и непосредственно связанных с последними семантических неопределенностей.

ЛИТЕРАТУРА


1. *Вьейра Р.* SQL SERVER 2000. Программирование: 2ч. – М.: БИНОМ, 2004.
2. *Кузнецов С. Д.* Основы баз данных. – М.: Интернет ун-т информ. технологий, 2005.
3. *Новиков Б. А., Домбровская Г. Р.* Настройка приложений баз данных. – СПб.: БХВ-Петербург, 2006.
4. *Родионов А.Н.* Моделирование данных: от сущностей к структурам данных. Сущности и объекты // Вестник ХГАЭП. – 2010. – №1 (46). – С. 43-61.
5. *Родионов А.Н.* Моделирование данных: от сущностей к структурам данных. Структуры концептуальной модели // Вестник ХГАЭП. – 2010. – №6 (51). – С. 49-68.
6. *Родионов А. Н.* “Мигрирующие” объекты моделей данных: ”слабые сущности” и документы // Вестник ХГАЭП. – 2010. – №1 (52). – С. 40-65.
7. *Родионов А. Н.* Критерии качества даталогических схем. Полнота моделей данных // Вестник ХГАЭП. – 2010. – №2 (53). – С. 28-51.
8. *Родионов А. Н.* Качество даталогических схем. Принцип компактности моделей данных и его приложения. I // Информатика и системы управления. – 2012. – №1 (31). – С. 16-27.
9. *Родионов А. Н.* Качество даталогических схем. Принцип компактности моделей данных и его приложения. II // Информатика и системы управления. – 2012. – №2 (32). – С. 3-13.
10. *Цикритзис Д., Лоховски Ф.* Модели данных. – М.: Финансы и статистика, 1985.



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



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



***VI Международная конференция***
***«Параллельные вычисления и задачи управления» (PACO’2012)***
проводится *федеральным государственным бюджетным учреждением науки Институтом проблем управления им. В.А.Трапезникова Российской академии наук (ИПУ РАН)*
***с 24 по 26 октября 2012 г. в Москве***

Цель конференции – представление и обсуждение новых направлений и достижений в области параллельных и распределенных вычислительных технологий, а также их приложения к задачам управления. Основные направления конференции:
математические модели и вычислительные методы распараллеливания вычислений;
технологии программирования параллельных и распределенных систем;
параллельные и распределенные вычисления в задачах моделирования, идентификации, анализа, управления и оптимизации;
проблемы интеграции данных, программ, процессов и систем в глобальной компьютерной среде;
мультиагентная парадигма организации параллельных вычислений и распределенной обработки информации;
надежные вычисления и защита информации в распределенных компьютерных средах;
проекты вычислительных и управляющих систем с параллельной и/или распределенной обработкой информации;
архитектуры распределенных вычислительных и управляющих систем в компьютерных сетях.

Сайт конференции: http://paco2012.ipu.ru//