# Стратегии объектно-реляционного отображения: систематизация и анализ на основе паттернов[1]

*В.А. Семенов, С.В. Морозов, С.А. Порох*

**Аннотация.** Данная статья посвящена методам отображения прикладных объектно-ориентированных данных в реляционную модель. В ней проводится систематизация данных методов, а также их анализ на основе введенной системы паттернов. Задача функционально полного отображения моделей данных рассматривается на примере EXPRESS нотации, получившей распространение в качестве стандартного средства информационного моделирования научных и промышленных данных. В статье также описывается разрабатываемый в среде Oracle программно-инструментальный комплекс, реализующий несколько альтернативных стратегий отображения на основе рассмотренных паттернов и предоставляющий базовую функциональность объектно-реляционного посредника.

## *1. Введение*

В условиях бурного развития информационных технологий и стремительной эволюции систем управления информацией одной из актуальных задач, перманентно возникающей перед разработчиками приложений, является выбор системы управления базой данных (СУБД), в наилучшей степени соответствующей особенностям решаемых прикладных задач хранения данных и эффективного управления ими в ходе пользовательских сессий. С одной стороны, СУБД должна поддерживать необходимую степень абстракции данных, а с другой стороны, она должна быть ориентирована на структурные особенности их организации и характер использования.

Выбор ключевой парадигмы и базовой метамодели системы управления: реляционной, объектно-ориентированной, гибридной объектно-реляционной, XML-ориентированной — является одним из важнейших аспектов поиска СУБД наряду с анализом функциональной полноты языка запросов, поддерживаемых моделей транзакций, возможностей работы с версиями, средств конфигурирования приложений. Часто политика организации в отношении приобретения и использования новых программных продуктов, а также устойчивое положение производителя на рынке оказываются не менее

[1] Работа поддержана грантами РФФИ (04-01-00527) и Фонда содействия отечественной науке (http://www.science-support.ru).



существенными факторами, влияющими на выбор СУБД.

В условиях широкого распространения объектно-ориентированной методологии разработки прикладных систем и одновременного доминирующего положения на рынке реляционных СУБД привлекательным решением оказывается использование промежуточного слоя программного обеспечения, предоставляющего необходимые объектно-ориентированные интерфейсы к данным, хранимым под управлением реляционной СУБД [1, 2]. В этом случае удачно сочетаются известные преимущества реляционных СУБД:

- возможность поддержки наследуемых (legacy) систем, использующих традиционные решения;
- простота использования технологии, основанной на понятной табличной модели и математически строгой теории реляционной алгебры;
- широкое распространение и тщательная многолетняя апробация предлагаемых на рынке продуктов,

с достоинствами объектно-ориентированных СУБД в отношении:

- сложности реализуемых прикладных моделей;
- эффективности реализации объектных запросов, выражаемых как в навигационном, так и в предикативном стилях;
- предоставления естественных для приложений объектно-ориентированных интерфейсов на популярных языках реализации.

При этом вполне достижимыми оказываются цели прозрачного манипулирования хранимыми данными (orthogonally transparent persistence), сформулированные в Манифесте объектно-ориентированных баз данных [3]. К ним относятся поддержка сложных объектов (complex objects), объектной идентификации (object identity), инкапсуляции (encapsulation), типизации и классификации (types & classes), переопределения, перегрузки и позднего связывания (overriding, overloading, late binding), полноты манипулирования (computational completeness), расширяемости (extensibility), сохранения (persistence), использования вторичных хранилищ (secondary storage management), а также обеспечение совместного параллельного доступа (concurrency), возможностей восстановления (recovery), реализации динамических запросов (ad hoc query facility).

Существенно, что значительная часть функций по реализации указанных свойств ложится при этом на промежуточный слой, а его создание сопряжено с комплексом проектных решений, затрагивающих сопровождаемость (maintainability), производительность (performance), простоту применения (simplicity) и обеспечение интероперабельности (interoperability) между клиентскими приложениями и сервером.

Одним из принципиальных вопросов, возникающих при создании промежуточного объектно-реляционного слоя, оказывается решение проблемы гармонизации данных, представленных существенно разными метамоделями (Metadata Impedance Mismatching), и поиск методов объектно-реляционного

отображения, адекватных решаемым прикладным задачам. Данные методы активно развиваются с конца восьмидесятых годов и отражены в многочисленных публикациях [4–8].

Тем не менее, в последнее время методы объектно-реляционного отображения вызывают особый интерес в связи со становлением функционально развитых стандартов объектно-ориентированного моделирования и хранения данных в рамках международной программы стандартизации промышленных информационных моделей ISO-10303 STEP [9], а также деятельности индустриальных групп OMG, ODMG. Актуальными при этом представляются задачи:

- функционально полного отображения моделей данных, специфицированных на языках EXPRESS [10], UML [11], ODL [12];
- систематизации методов отображения, в том числе, на основе паттернов [13–16];
- комплексного анализа методов для типовых моделей данных и контекстов их использования, включая анализ производительности на стандартных наборах тестов для объектно-ориентированных СУБД.

В настоящей работе задача отображения рассматривается для моделей данных, описанных на языке информационного моделирования EXPRESS. Мы не видим принципиальных различий между EXPRESS и современными версиями стандартов языков UML и ODL в отношении полноты конструкций моделирования данных, включая задание ограничений для них. Однако, с учетом двадцатилетнего опыта практического применения EXPRESS для описания сложных моделей инженерных данных и огромного ресурса разработанных междисциплинарных стандартов информационных моделей для ключевых отраслей науки и промышленности, его использование в методическом и практическом контексте исследования методов объектно-реляционного отображения кажется вполне оправданным и естественным.

Раздел 2 представляет собой краткое введение в язык моделирования данных EXPRESS, предлагающее пример спецификации прикладных данных, иллюстрирующий использование различных конструкций языка. В разделе 3 проводится систематизация методов объектно-реляционного отображения, основанная на классификации модельно-зависимых и модельно-независимых стратегий и паттернов реализации принципов объектно-ориентированного моделирования. В разделе 4 описываются разработанные CASE инструменты и пакеты программ на PL/SQL, реализующие альтернативные стратегии объектно-ориентированного отображения для СУБД Oracle. В заключении коротко суммируются результаты и даются рекомендации по использованию рассмотренных стратегий отображения и разработанных программных решений.

## 2. Объектно-ориентированное моделирование на EXPRESS

Кратко рассмотрим подмножество языка EXPRESS, непосредственно относящееся к спецификации объектно-ориентированных данных. EXPRESS



включает в себя также довольно развитую императивную часть, предназначенную для определения поведенческих свойств объектов и задания ограничений на них. Однако в силу предмета статьи эти подробности будут опущены, а их описание может быть найдено в оригинале стандарта языка [10].

Язык EXPRESS поддерживает набор стандартных, встроенных в него элементарных типов данных INTEGER, REAL, NUMBER, LOGICAL, BOOLEAN, BINARY и STRING для представления, соответственно, целых, вещественных и произвольных числовых данных, логических и булевых значений, последовательностей двоичных данных и строк. Для перечислимых типов предусмотрена специальная конструкция ENUMERATION. Агрегатные типы ARRAY, SET, BAG и LIST предоставляют возможность определения различного рода контейнеров, таких как массивы, множества, мультимножества и списки. Опционально могут быть заданы их размеры, способы индексации элементов, условие множественности эквивалентных элементов для массивов и списков, а также допустимость разреженности элементов в массивах. Селективные типы, вводимые оператором SELECT, позволяют использовать переменные и константы, принимающие значения одного из альтернативных типов, объявленных в списке оператора. Новые производные типы данных создаются на основе стандартных и предопределенных типов с помощью конструкции TYPE. Допускается произвольная вложенность определений пользовательских типов, которая, в частности, обеспечивает создание многомерных массивов, вложенных селективных и агрегатных конструкций.

Типы GENERIC, AGGREGATE, а также ARRAY, SET, BAG и LIST OF GENERIC обеспечивают обобщенную реализацию функций и процедур с использованием абстракций простых и агрегатных данных.

Для объектных типов используется конструкция ENTITY, предусматривающая разнообразные модели простого и множественного наследования с помощью квалификаторов AND, ANDOR, ONEOF. При специфицировании объектного типа задаются атрибуты и ассоциации различной кратности (EXPLICIT), обратные ассоциации (INVERSE), а также производные вычисляемые свойства объектов (DERIVED). Последние определяются типами и выражениями, которые могут включать в себя значения явных атрибутов, константы, исполняемые операторы, включая вызов функций и процедур, как стандартных, так и пользовательских.

Ограничения целостности данных задаются непосредственно при определении объектного типа с помощью конструкции WHERE, определяющей логические условия в виде выражений логического типа, а также с помощью квалификатора UNIQUE, приписывающего условие уникальности атрибутам, ассоциациям и производным свойствам в популяциях родственных объектов. Для задания глобальных ограничений над разнородными объектами предусмотрена конструкция RULE, позволяющая описать ограничение в виде формальной спецификации функции логического типа.

Определения глобальных констант, простых и объектных типов данных, глобальных ограничений объединяются в разделе информационной схемы модели (SCHEMA). Посредством конструкций импорта USE и REFERENCE достигается возможность использования в одной схеме определений из других схем, что обеспечивает разработку сложных информационных моделей путем иерархической композиции отдельных схем. Таким образом, охватываются разнообразные практически содержательные случаи объектно-ориентированного моделирования прикладных данных.

Ниже (см. рис. 1) представлен пример информационной модели на языке EXPRESS — схема `ActorResource`, специфицирующая информацию о персонах и организациях, участвующих в совместном проекте, их ролях в нем и отношениях между ними.

```
SCHEMA ActorResource;

  TYPE ActorSelect = SELECT (Organization, Person);
  END_TYPE;

  TYPE AddressTypeEnum = ENUMERATION OF (OFFICE, HOME, USERDEFINED);
  END_TYPE;

  TYPE Label = STRING(255);
  END_TYPE;

  TYPE ActorRole = Label;
  END_TYPE;

  ENTITY Address
    ABSTRACT SUPERTYPE OF (ONEOF(PostalAddress, TelecomAddress));
      Purpose           : AddressTypeEnum;
      UserDefinedPurpose : OPTIONAL STRING;
    INVERSE
      OfPerson          : SET OF Person FOR Addresses;
      OfOrganization    : SET OF Organization FOR Addresses;
    WHERE
      WR1 : (Purpose <> AddressTypeEnum.USERDEFINED) OR
            ((Purpose = AddressTypeEnum.USERDEFINED) AND
              EXISTS(UserDefinedPurpose));
  END_ENTITY;

  ENTITY PostalAddress
    SUBTYPE OF(Address);
      AddressLines     : LIST [1:?] OF Label;
  END_ENTITY;

  ENTITY TelecomAddress
    SUBTYPE OF(Address);
      TelephoneNumbers        : OPTIONAL LIST [1:?] OF Label;
      FacsimileNumbers        : OPTIONAL LIST [1:?] OF Label;
      ElectronicMailAddresses : OPTIONAL LIST [1:?] OF Label;
      WWWUrls                 : OPTIONAL LIST [1:?] OF Label;
    WHERE
      WR1 : EXISTS (TelephoneNumbers) OR EXISTS (FacsimileNumbers) OR
            EXISTS (ElectronicMailAddresses) OR EXISTS (WWWUrls);
```



```
  END_ENTITY;

  ENTITY Organization;
      Id          : INTEGER;
      Name        : Label;
      Description : OPTIONAL STRING;
      Roles       : LIST [0:?] OF UNIQUE ActorRole;
      Addresses   : LIST [1:?] OF UNIQUE Address;
    INVERSE
      IsRelatedBy : SET OF OrganizationRelationship FOR
RelatedOrganizations;
      Relates     : SET OF OrganizationRelationship FOR
RelatingOrganization;
      Engages     : SET OF Person FOR EngagedIn;
    UNIQUE
      UR1 : Id;
  END_ENTITY;

  ENTITY OrganizationRelationship;
      Name                 : Label;
      Description          : OPTIONAL STRING;
      RelatingOrganization : Organization;
      RelatedOrganizations : SET [1:?] OF Organization;
  END_ENTITY;

  ENTITY Person;
      Id           : INTEGER;
      FamilyName   : OPTIONAL Label;
      GivenName    : OPTIONAL Label;
      MiddleNames  : OPTIONAL LIST [1:?] OF Label;
      PrefixTitles : OPTIONAL LIST [1:?] OF Label;
      SuffixTitles : OPTIONAL LIST [1:?] OF Label;
      Roles        : LIST [0:?] OF UNIQUE ActorRole;
      Addresses    : OPTIONAL LIST [1:?] OF UNIQUE Address;
      EngagedIn    : SET OF Organization;
    UNIQUE
      UR1 : Id;
    WHERE
      WR1 : EXISTS(FamilyName) OR EXISTS(GivenName);
  END_ENTITY;

END_SCHEMA;
```

*Рис. 1. Спецификация схемы* `ActorResource` *на языке EXPRESS.*

Представление схемы `ActorResource` в графической нотации EXPRESS-G изображено на Рис. 2.

Приведенный пример достаточно иллюстративен и не нуждается в дополнительных комментариях. Текстовая нотация спецификаций модели довольно близка упоминаемому выше языку описания объектных данных ODL, а графическая — аналогична языку моделирования UML.

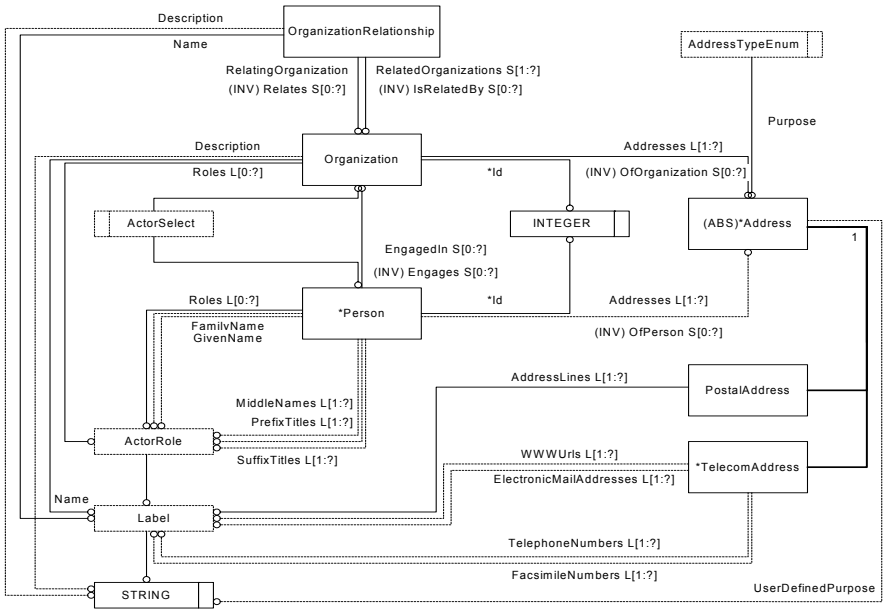


*Рис. 2. Представление схемы* `ActorResource` *в нотации EXPRESS-G.*

К настоящему времени в рамках международных программ по стандартизации прикладных информационных моделей и интероперабельности программных приложений накоплен значительный ресурс многопрофильных междис-циплинарных моделей. Ресурс охватывает такие научные и промышленные области, как машиностроение, авиационную и космическую промышленность, судостроение, нефтегазовый комплекс, архитектуру и строительство, электронную промышленность, фармацевтику, геоинформатику. Значительная часть разработанных на языке EXPRESS спецификаций принята в качестве документов ISO-10303. Другая часть разрабатывается непосредственно промышленными альянсами для последующего представления в международную организацию по стандартам.

К существенным особенностям прикладных информационных моделей следует отнести:

- сложность и масштабность моделей, выражающиеся в большом количестве типов, определяемых в рамках одной информационной схемы, в применении альтернативных механизмов множественного наследования и полиморфного переопределения свойств объектных типов, а также в использовании вложенных агрегатных и селективных конструкций и двунаправленных ассоциаций;
- необходимость поддержки запросов к данным в декларативном предикативном и навигационном стилях, а также эффективную



реализацию базовых операций манипулирования ими;
- широкий контекст использования моделей в приложениях, оперирующих как с данными одной многопрофильной информационной схемы, так и с данными нескольких независимых схем.

Данные особенности обуславливают поиск эффективных решений для объектно-реляционного отображения и их систематизацию для комплексного использования в конкретных прикладных контекстах.

## *3. Общая систематизация подходов*

### 3.1. Классификация паттернов отображения

Независимо от особенностей применяемых подходов нам видится ряд связанных между собой аспектов отображения прикладных данных из объектно-ориентированной модели в реляционную. Прежде всего, это технические вопросы семантического отображения в реляционную метамодель базовых конструкций языка EXPRESS, а именно:

- элементарных базовых типов;
- перечислимых типов;
- ассоциативных связей между объектами;
- селективных типов;
- агрегатных типов;
- вложенных структур данных, основанных на базовых, перечислимых, ассоциативных, селективных и агрегатных типах данных;
- простых и сложных объектных типов в рамках модели множественного наследования;
- информационных схем.

Не менее существенными для практического применения являются часто противоречащие друг другу проблемы:

- выбора стратегии отображения в соответствии с контекстом использования семантики информационной модели;
- поддержки метаданных в реляционном представлении и их конструктивного применения в ходе пользовательских сессий;
- эффективности реализации объектных запросов и операций манипулирования объектами (создание, модификация, удаление);
- оптимизации используемых ресурсов, включая затраты памяти;
- сопровождаемости решений и их гибкости по отношению к возможной эволюции используемых прикладных моделей.

На Рис. 3 представлена общая классификация методов объектно-реляционного отображения, в основе которой лежит принцип выделения перечисленных выше аспектов и соответствующих им альтернативных решений (паттернов).

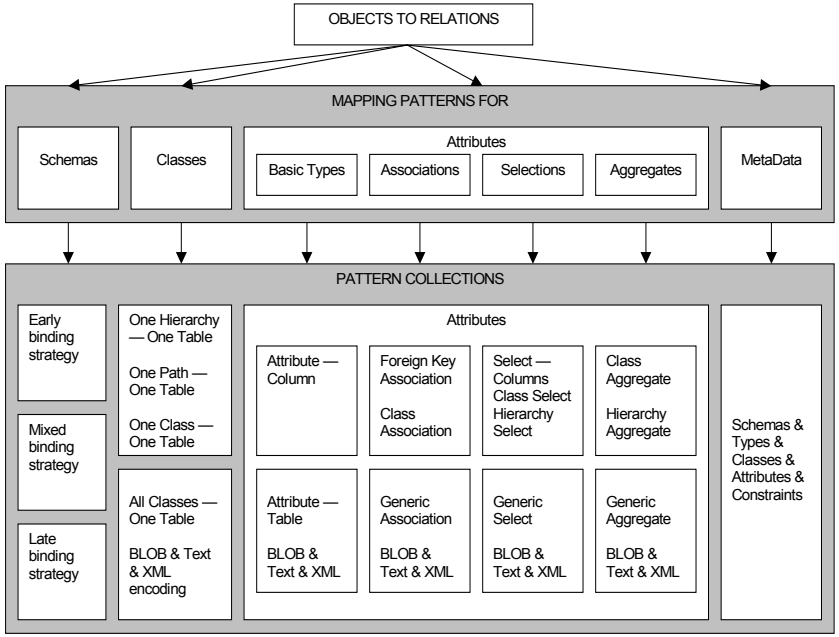


*Рис. 3. Классификация паттернов объектно-реляционного отображения.*

## 3.2. Отображение информационных схем

На наш взгляд вопрос о выборе стратегии отображения в рамках схемо-зависимого (СЗ) или схемо-независимого (СН) подходов является наиболее принципиальным для систематизации методов объектно-реляционного отображения и их адекватного применения в приложениях.

### *3.2.1. Схемо-независимая стратегия*

СН-стратегия ориентирована на использование единой реляционной схемы для представления произвольных прикладных данных, модели которых специфицированы на EXPRESS. Для приложений, оперирующих одновременно с несколькими перманентно изменяемыми прикладными моделями, применение СН-стратегии является наиболее предпочтительным. Сопровождение и администрирование реляционной БД с фиксированной системой таблиц, состав и структура которой не меняется, достаточно просты.

К издержкам стратегии следует отнести необходимость поддержки и использования словарей метаданных, без которых реализация промежуточного объектно-реляционного слоя невозможна. Сами словари также могут быть представлены в виде таблиц, хранящих информацию об исходных прикладных моделях и включенных в состав единой реляционной системы (см. Рис. 4).



Другим недостатком является относительно низкая эффективность выполнения базовых операций манипулирования объектами, поскольку их реализация сопряжена с необходимым дополнительным анализом сопутствующих метаданных.

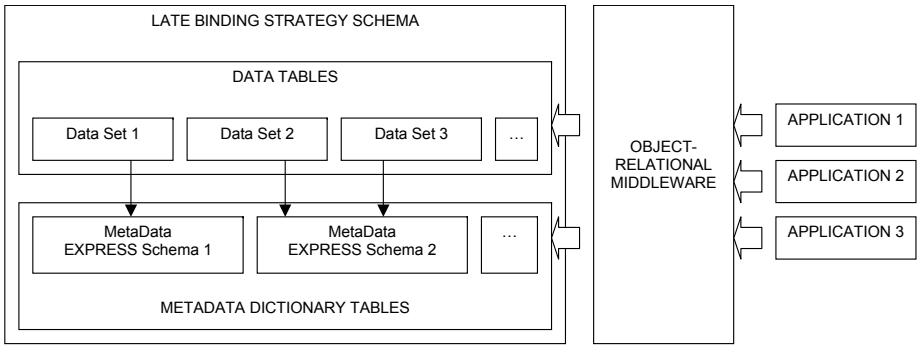


*Рис. 4. Организация реляционной системы для СН-стратегии.*

### *3.2.2. Схемо-зависимая стратегия*

СЗ-стратегия в большей степени ориентирована на приложения, оперирующие с единственной моделью данных, не изменяемой на протяжении всего жизненного цикла проекта. Организация реляционной системы в этом случае может отражать и учитывать особенности конкретной прикладной модели. СЗ-стратегией не исключается и одновременная поддержка нескольких моделей (см. Рис. 5). Однако в силу сложности сопровождения и администрирования реляционных БД с большим количеством таблиц ее использование в этом случае может оказаться крайне обременительным.

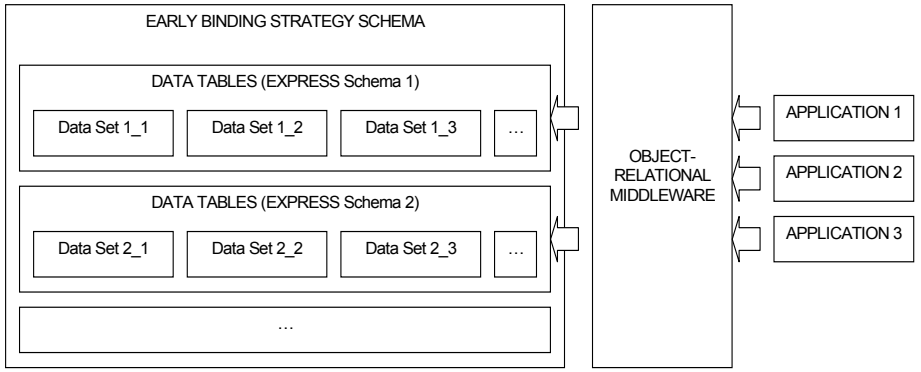


*Рис. 5. Организация реляционной системы для СЗ-стратегии.*

Достоинством СЗ-стратегии является возможность более эффективной реализации объектных запросов и операций манипулирования объектами за счет

непосредственной адресации к таблицам с хранимыми данными, в отличие от СН-стратегии, где такая адресация осуществляется косвенно через таблицы метаданных.

Как разновидность СЗ-стратегии может рассматриваться смешанная (СМ) стра-тегия, заключающаяся в организации системы таблиц по заданной прикладной модели при одновременном использовании словарей метаданных. При опреде-ленной избыточности в представлении данных такая стратегия обеспечивает более эффективную реализацию сложных запросов непосредственно средс-твами реляционной СУБД, поскольку вся необходимая метаинформация может также храниться в виде таблиц и быть доступной при обработке запросов.

## 3.3. Отображение наследования классов

Паттерны, предназначенные для отображения отношений простого наследования между классами, являются хорошо известными [13, 14]. Мы обсудим возможные варианты отображений в рамках развитой модели множественного наследования языка EXPRESS.

### *3.3.1. Паттерн OneInheritanceHierarchy–OneTable*

Первый, наиболее простой паттерн ***OneInheritanceHierarchy–OneTable*** соответствует случаю отображения всех конкретных родственных классов, имеющих общий набор прародителей, в одну таблицу **<Hierarchy>_Instances**. Прародителем будем называть класс-предок, у которого нет собственных родителей.

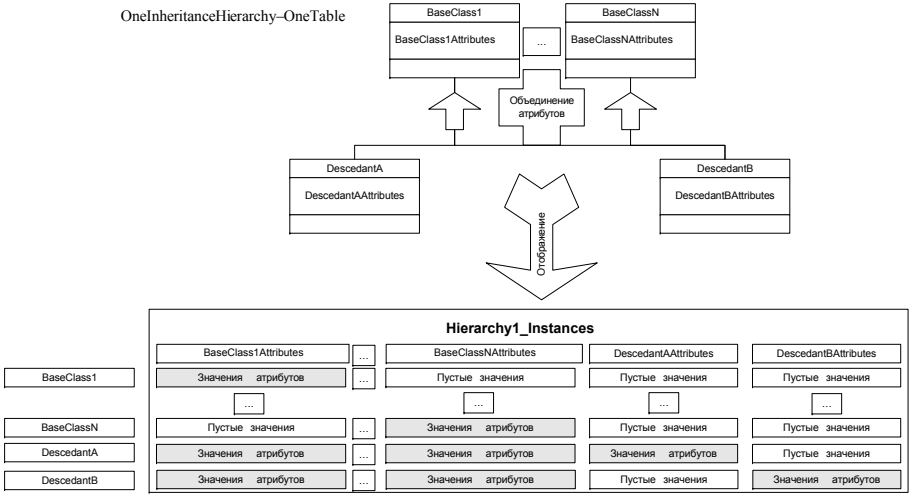


*Рис. 6. Иллюстрация паттерна наследования* ***OneInheritanceHierarchy–OneTable****.*



В случае простого наследования данный паттерн трансформируется в стратегию представления конкретных классов в каждом дереве наследования одной реляционной таблицей (см. рис. 6). Атрибуты всех родственных классов, являющихся вершинами дерева, отображаются в столбцы данной таблицы. Если иерархия наследования классов в прикладной модели представлена несколькими деревьями, то каждому такому дереву будет соответствовать отдельная таблица.

В общем случае множественного наследования иерархия классов представляется набором таблиц, каждая из которых соответствует одному из сочетаний прародителей. Все атрибуты классов, объединенных единым набором прародителей, отображаются в столбцы конкретной таблицы из этого набора. Для записей объектов, в которых не определены какие-либо атрибуты, в соответствующих столбцах прописываются нулевые значения.

Достоинством паттерна является возможность эффективной реализации базовых операций над произвольными объектами без дополнительных расходов на сборку значений атрибутов из разных таблиц и их обратное распределение по ним. Также непосредственно реализуется полиморфное чтение. Единственная сложность состоит в определении типа запрашиваемых объектов. Простота поддержки и развития такой СЗ-стратегии делает ее довольно привлекательной. Недостатком является излишнее потребление памяти за счет избыточного хранения нулевых значений, а иногда и необходимость индексирования большого числа столбцов для ускорения выполнения запросов по значениям отдельных атрибутов. При большой глубине наследования классов, что является типичным в научных и промышленных моделях STEP, это может оказаться критичным как для потребления памяти, так и для производительности.

### *3.3.2. Паттерн OneClass–OneTable*

В паттерне ***OneClass–OneTable*** каждый конкретный или абстрактный классы в иерархии наследования представляются отдельной таблицей **<Class>_Instances**, при этом собственные атрибуты данного класса отображаются в ее столбцы. Для связи с наследуемыми атрибутами она хранит вторичные ключи соответствующих записей в таблицах родительских классов. В случае простого наследования — один вторичный ключ, в случае множественного наследования — несколько вторичных ключей, каждый из которых соответствует таблице одного из родителей (см. рис. 7).

Поскольку при множественном наследовании возможны неоднозначные ситуации, когда атрибуты одного и того же базового класса наследуются несколько раз, реализация данного паттерна сопряжена с анализом и разрешением подобных ситуаций. В языке EXPRESS допустимо лишь виртуальное наследование, при котором любые атрибуты базовых классов должны наследоваться единожды. Поэтому при анализе реконструируется основное дерево наследования, а ветви, приводящие к циклам, игнорируются в резу-

льтате обнуления соответствующих вторичных ключей к записям в таблицах родительских классов. Случаи многократного наследования атрибутов обрабатываются автоматически и не требуют дополнительного анализа.

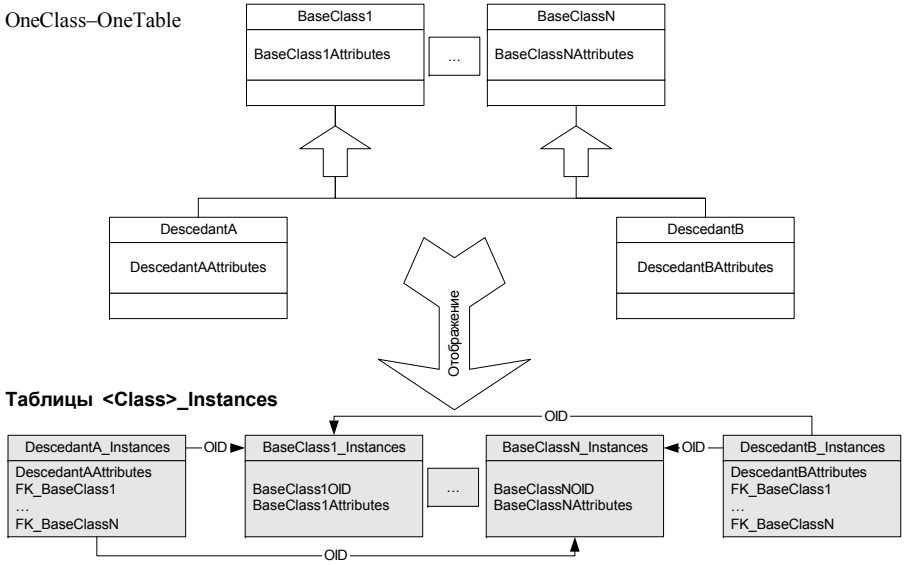


*Рис. 7. Иллюстрация паттерна наследования **OneClass—OneTable**.*

Паттерн обеспечивает хорошую производительность на операциях полиморфного чтения, однако реализация базовых операций над объектами сопряжена с расходами на сборку значений атрибутов из нескольких таблиц при чтении и их рассылку по таблицам при записи и модификации. При глубокой иерархии наследования такие издержки могут оказаться существенными.

Затраты памяти в реализации данного паттерна близки к оптимальным, поскольку хранение вторичных ключей не требует больших накладных расходов. Как элемент СЗ-стратегии паттерн не обеспечивает простоту поддержки и эволюции сложных прикладных моделей с развитой иерархией наследования.

### *3.3.3. Паттерн OneInheritancePath–OneTable*

Некоторые недостатки предыдущего паттерна компенсируются в результате сериализации таблиц классов по отношениям наследования. В паттерне ***OneInheritancePath–OneTable*** каждому конкретному классу соответствует своя таблица **<Concrete_Class>_Instances**, в столбцы которой отображаются все атрибуты класса, включая наследуемые от родителей (см. рис. 8).

Паттерн обеспечивает хорошую эффективность на базовых операциях чтения, записи, модификации, удаления, однако полиморфные операции оказываются достаточно дорогими, поскольку сопряжены с просмотром всех таблиц



классов, наследуемых от заданного. Затраты памяти здесь оптимальны, поскольку не требуется хранение избыточных полей. Важным достоинством паттерна в ряде случаев оказывается равномерное распределение запросов и сопряженных с ними блокировок по таблицам схемы. В отличие от предыдущих паттернов наследования, в которых превалирующая нагрузка приходится на корневые таблицы прародителей, данный паттерн обеспечивает большую эффективность за счет более равномерного распределения записей.

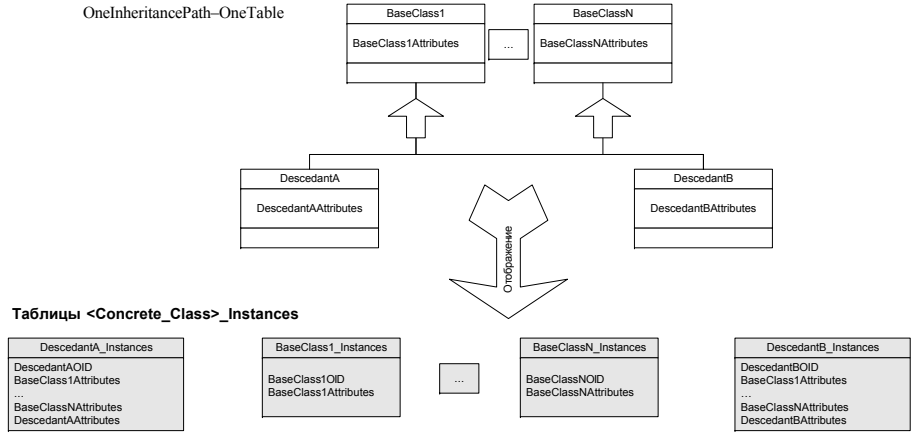


*Рис. 8. Иллюстрация паттерна наследования **OneInheritancePath–OneTable**.*

Однако в отношении поддержки эволюции схем паттерн довольно критичен, поскольку все запросы, основанные на полиморфных операциях, требуют модификации с учетом каждого нового наследуемого класса, включаемого в прикладную модель.

### *3.3.4. Паттерн AllClasses–OneTable*

Паттерн ***AllClasses–OneTable*** предполагает использование единой таблицы **Instances** для представления дескрипторов объектов всех классов. В столбцах таблицы хранятся идентификатор объекта и его тип (см. Рис. 9). Контекст использования паттерна связан с представлением атрибутов классов в виде самостоятельных таблиц. В этом случае связь между таблицами экземпляров классов и значений их атрибутов осуществляется через внешние ключи записей объектов (см. раздел 3.4.2.2). Предполагается, что значения атрибутов одного и того же типа хранятся в единой таблице независимо от их вхождения в состав того или иного класса. Тем самым достигается существенная для СН-стратегии инвариантность реляционной схемы по отношению к прикладным моделям. Связь простого объекта с его классом осуществляется через внешний ключ записи в таблице классов **Entities** (см. раздел 3.5). Для сложных объектов предусмотрен внешний ключ записи в соответствующей таблице сложных классов **Complex_Entities**.

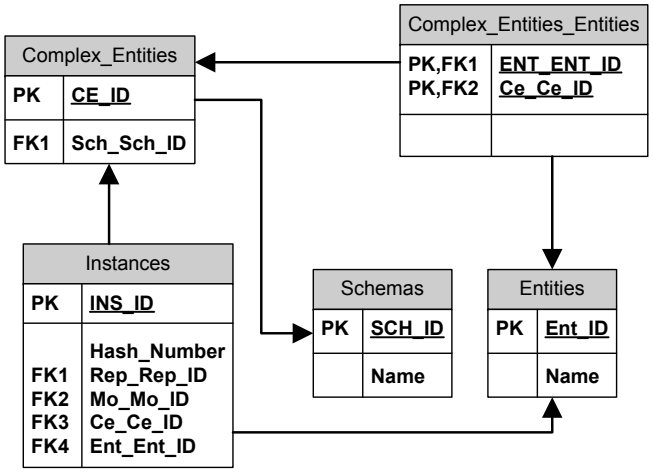


*Рис. 9. Представление сложных объектов и их связь с метаданными в паттерне **AllClasses–OneTable**.*

Более подробно варианты представления атрибутов в виде отдельных таблиц рассматриваются в следующих разделах (см. разделы 3.4.2.2, 3.4.3.3, 3.4.4.4, 3.4.5.3).

### *3.3.5. Паттерн BLOB*

Паттерн ***BLOB*** также предполагает использование единой таблицы **BLOB_Instances** для представления объектов всех классов. Однако в отличие от паттерна ***AllClasses–OneTable*** в данной таблице используется дополнительный столбец для хранения значений атрибутов, упакованных в бинарную или текстовую строку переменной длины. Задача упаковки значений в строку и их распаковки для клиентских приложений ложится непосредственно на промежуточный слой программного обеспечения. Хотя чтение и запись данных объекта осуществляются за одну операцию обращения к таблице, дополнительные расходы приходятся на обработку строк промежуточным слоем. По существу, в этом случае ***BLOB***-стратегия объединяет в себе паттерны наследования, агрегации и ассоциации.

Возможны разновидности данного паттерна, связанные с различными способами представления строки значений атрибутов как в бинарном формате, так и в одном из текстовых метаформатов. В частности, применительно к мета-модели EXPRESS стандарт STEP определяет формат текстового кодирования прикладных данных (ISO-10303-21) и несколько альтернативных способов XML разметки документов (ISO-10303-28), порождаемых соответствующей прикладной моделью данных, специфицируемой на языке EXPRESS.

Главным недостатком паттерна ***BLOB*** является невозможность разрешения запросов и реализации объектных операций непосредственно средствами



реляционной СУБД. В данном случае она играет роль простого хранилища, а эти функции выполняет промежуточный слой. Как паттерн СН-стратегии он не требует больших затрат на поддержку реляционной схемы при эволюции прикладной модели, поскольку связанные с этими изменениями функции затрагивают лишь промежуточный слой.

BLOB

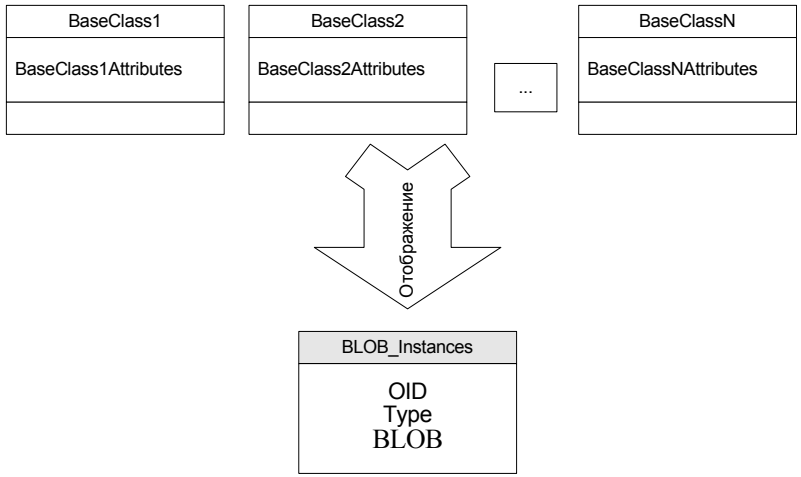


*Рис. 10. Иллюстрация паттерна **BLOB**.*

## 3.4. Отображение атрибутов

Атрибуты классов представляются либо столбцами соответствующих таблиц объектов классов, либо самостоятельными таблицами. Как и в случае паттернов отображения классов, альтернативы представления атрибутов во многом определяются применяемой схемо-зависимой или схемо-независимой стратегией объектно-реляционного отображения. Рассмотрим их, следуя введенной классификации паттернов отображения атрибутов простых, селективных, агрегатных типов и ассоциаций.

### *3.4.1. Представление простых типов*

Соответствие базовых типов языка EXPRESS типам данных в SQL [17] достаточно прозрачно. В таблицу 1 сведены базовые типы EXPRESS и возможные способы их представления в некоторых популярных коммерческих и свободно распространяемых реляционных СУБД.

Помимо указанных вариантов представления точности числовых значений и ограничений длины строковых и двоичных переменных, важные отличия затрагивают способы представления атрибутов типа BOOLEAN, LOGICAL и ENUMERATION. С точки зрения семантики языка EXPRESS значения этих типов упорядочены, и для них определены операции сравнения. Поэтому, если

предполагается реализация запросов с использованием подобных операций, более предпочтительным выглядит представление переменных этих типов как целых чисел. В этом случае функция интерпретации значений в терминах исходной прикладной модели возлагается на промежуточный программный слой, возможно, с использованием словарей метаданных. Значения, представленные в виде символьных строк, интерпретируются непосредственно клиентскими приложениями. Средствами СУБД может контролироваться также и корректность данных, для чего должны быть наложены соответствующие ограничения на область допустимых значений.

| EXPRESS | MySQL | PostgreSQL | Oracle |
|---|---|---|---|
| INTEGER | INTEGER | INTEGER | INTEGER |
| REAL | REAL,<br>DOUBLE PRECISION | FLOAT8,<br>DOUBLE PRECISION | NUMBER,<br>DOUBLE PRECISION |
| REAL(n) | FLOAT(n) | NUMERIC(n) | NUMBER(n) |
| NUMBER | REAL,<br>DOUBLE PRECISION | FLOAT8,<br>DOUBLE PRECISION | NUMBER,<br>DOUBLE PRECISION |
| BOOLEAN | CHAR(1),<br>TINYINT | CHAR(1),<br>SMALLINT | CHAR(1),<br>INTEGER |
| LOGICAL | CHAR(1),<br>TINYINT | CHAR(1),<br>SMALLINT | CHAR(1),<br>INTEGER |
| ENUMERATION | VARCHAR(128),<br>INTEGER | VARCHAR(128),<br>INTEGER | VARCHAR2(128),<br>INTEGER |
| STRING | TEXT (up to 64K),<br>LONGTEXT (up to 4Gb) | TEXT (about 1Gb) | VARCHAR2(4000),<br>LONG (up to 2Gb) |
| STRING(n) | VARCHAR(n) | VARCHAR(n) | VARCHAR2(n) |
| STRING(n) FIXED | CHAR(n) | CHAR(n) | CHAR(n) |
| BINARY | BLOB (up to 64K),<br>LONGBLOB (up to 4Gb) | BYTEA | LONG RAW (up to 2Gb) |
| BINARY(n) | VARCHAR(n) BINARY | VARBIT(n) | RAW(n) |
| BINARY(n) FIXED | CHAR(n) BINARY | BIT(n) | RAW(n) |
| ENTITY (reference) | VARCHAR(128),<br>FOREIGN KEY | VARCHAR(128),<br>FOREIGN KEY | VARCHAR2(128),<br>FOREIGN KEY |

*Таблица 1. Соответствие базовых типов EXPRESS и SQL в реляционных СУБД.*

### *3.4.2. Отображение атрибутов простых типов*

#### 3.4.2.1. Паттерн Attribute–Column

Реализация паттерна представления атрибутов простых типов в виде столбцов соответствующих таблиц объектов довольно прозрачна и естественна для применения в рамках СЗ-стратегии. В этом случае каждая таблица объектов



классов включает в себя соответствующий столбец для представления значений простого атрибута. Тип столбца определяется возможными вариантами представления базовых типов языка EXPRESS, рассмотренными в предыдущем разделе. В качестве имени столбца могут быть выбраны либо имя атрибута, уникальное в пределах класса, либо конкатенация имен класса и атрибута, уникальная в пределах информационной схемы.

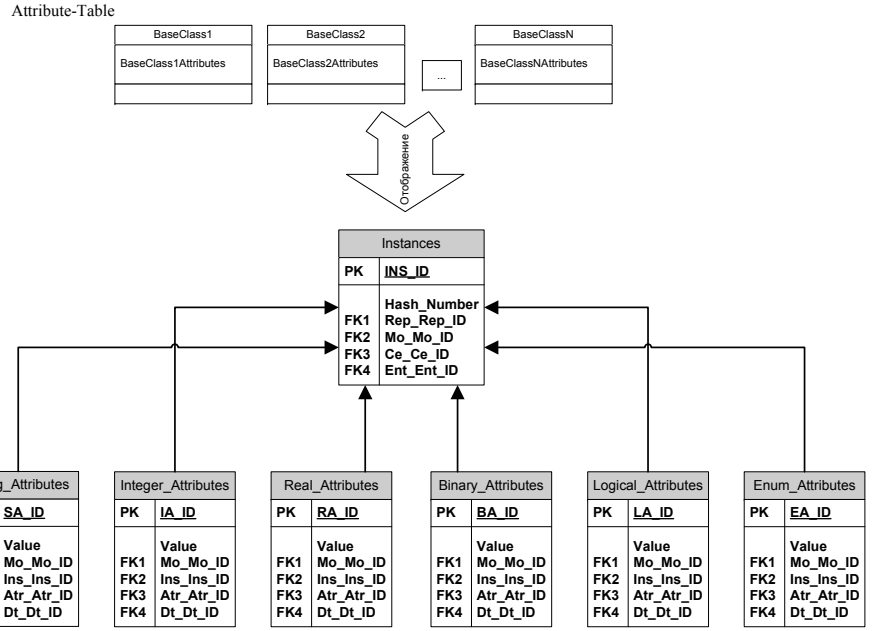


*Рис. 11. Организация реляционных таблиц для представления атрибутов простых типов в паттерне **Attribute–Table**.*

#### 3.4.2.2. Паттерн Attribute–Table

Данный паттерн предполагает использование самостоятельных таблиц для представления простых однотипных атрибутов. Его применение возможно лишь в рамках СН- и СМ-стратегий с одновременным использованием таблиц метаданных. Привязка значений атрибутов к дескрипторам объектов осуществляется по внешним ключам записей объектов в таблице **Instances**, представленным в таблицах атрибутов. Для идентификации хранимых величин как значений атрибутов определенных классов в них также хранятся внешние ключи записей метаинформации о соответствующих атрибутах в таблице **Attributes** в составе системы таблиц метаданных.

Независимо от принадлежности различным классам значения однотипных атрибутов хранятся в одной таблице. Для представления всех атрибутов простых типов, допускаемых метамоделью языка EXPRESS, в общей реляционной схеме достаточно иметь фиксированный набор из шести таблиц:

**Integer_Attributes**, **Real_Attributes**, **Logical_Attributes**, **String_Attributes**, **Binary_Attributes** и **Enum_Attributes** (см. рис. 11). Предполагается, что в схеме хранения данные типа NUMBER всегда представимы типом REAL, а данные типа BOOLEAN — типом LOGICAL.

### *3.4.3. Отображение ассоциаций*

Язык EXPRESS допускает определение разного рода ассоциативных отношений между классами, отличающихся как по типу, так и по кратности. Ассоциации однонаправлены в том смысле, что ассоциируемый объект может быть получен по соответствующей ссылке из объекта, содержащего ассоциативную связь. Вместе с тем, допускается задание двунаправленных связей с помощью определения обратных атрибутов (INVERSE) в ассоциируемом классе. При определении прямой и обратной ассоциации допустимо указание диапазона кратности связи как ограничения, налагаемого на количество объектов, участвующих в ней как со стороны ассоциируемых, так и со стороны ассоциирующих объектов (см. Рис. 1).

Ассоциации “один к одному” реализуются как простые атрибуты, имеющие тип объектной ссылки. Ассоциации “один ко многим” представляются средствами языка как агрегаты простых ассоциативных отношений. Ассоциации вида “многие ко многим” непосредственно конструкциями языка не поддерживаются, однако могут быть представлены в прикладной модели в виде дополнительных объектов, через которые на основе множественных ассоциаций могут быть установлены требуемые отношения между прикладными объектами.

Рассмотрим задачу отображения множественных ассоциаций вида “один ко многим” как наиболее типовой случай, к которому могут быть непосредственно редуцированы ассоциации “один к одному” и “многие ко многим”. Видятся три наиболее содержательных случая представления ассоциаций и соответствующих им паттерна отображения.

#### 3.4.3.1. Паттерн ForeignKeyAssociation

Данный паттерн применим к прямым множественным ассоциациям при условии, что соответствующая обратная ассоциация является простой. Иными словами, с каждым объектом, участвующим в связи, может быть ассоциировано некоторое множество объектов. Однако каждый объект таких множеств может участвовать только в одной обратной ассоциации этой связи. Паттерн реализуется в рамках СЗ-стратегии путем включения в таблицу ассоциированного класса **<Associated_Class>** (класса, на который содержится ссылка в классе ассоциации) внешнего ключа на таблицу **<Associating_Class>**. Имя ключа может соответствовать имени связи в соответствующей спецификации **<Associating_Class>** (см. Рис. 12). В случае упорядоченных множественных ассоциаций (LIST или ARRAY OF ENTITY) может потребоваться дополнительный столбец в таблице **<Associated_Class>** для хранения индекса ассоциации. Если связь реализуется как элемент вложенной



агрегатной конструкции, то в данной таблице предусматривается необходимое число столбцов индексов для каждого из упорядоченных множеств, участвующих в ней. Аналогично, если связь реализуется как элемент селективной конструкции, то в таблицу добавляется соответствующий столбец для представления дискриминатора. Более подробно эти случаи описаны в паттернах отображения селективных и агрегатных конструкций.

ForeignKeyAssociation

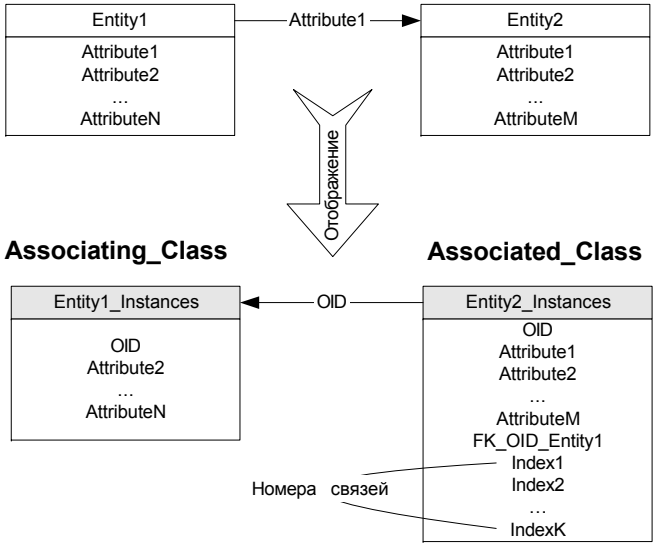


*Рис. 12. Представление ассоциативных отношений в паттерне **ForeignKeyAssociation**.*

Чтение ассоциирующего объекта реализуется посредством одной операции соединения или двух операций запроса к таблицам **<Associated_Class>** и **<Associating_Class>**, один из которых является множественным. Запись объекта также сопряжена с множественной операцией модификации внешних ключей в записях ассоциированных объектов. Затраты памяти в реализации паттерна близки к оптимальным, поскольку издержки приходятся лишь на хранение дополнительного внешнего ключа, а иногда и индекса ассоциации в таблице **<Associated_Class>**, для каждой связи, в которой ассоциируемый класс участвует.

#### 3.4.3.2. Паттерн ClassAssociation

Данный паттерн позволяет непосредственно представить множественные ассоциативные отношения в результате использования отдельной таблицы в рамках СЗ-стратегии. Такая таблица **<Class1>_<Class2>_Associations** создается для каждой пары классов, участвующих в ассоциативной связи, и содержит

пары внешних ключей записей в таблицах классов ассоциируемых и ассоциирующих объектов (см. рис. 13).

В отличие от предыдущего, данный паттерн обеспечивает представление произвольных ассоциативных отношений, не ограниченных кратностью обратных ассоциаций.

ClassAssociation

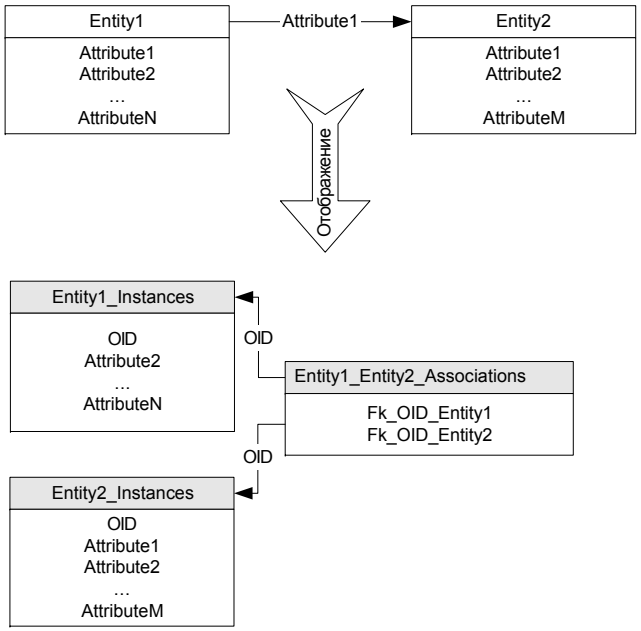


*Рис. 13. Представление ассоциативных отношений в паттерне **ClassAssociation**.*

### 3.4.3.3. Паттерн GenericAssociation

Данный паттерн соответствует CH-стратегии. Он реализует идею унифицированного представления всех видов ассоциаций, участвующих в прикладной модели, одной таблицей **Associations**. Ассоциативные связи в таблице устанавливаются через ссылки на дескрипторы прикладных объектов в таблице **Instances** в виде внешних ключей соответствующих записей в ней. Поскольку для реализации CH стратегии важна привязка элементов данных к прикладной модели, для каждой ассоциации в таблице **Associations** хранится также ссылка на метаинформацию о соответствующем атрибуте, представленную в таблице **Attributes**. Если ассоциация устанавливается как элемент агрегатной или селективной конструкции, то указывается также ссылка на соответствующую запись в таблицах **Aggregates** или **Selections** (см. Рис. 14). Таким образом, таблица **Associations** хранит множество записей обо всех видах ассоциаций прикладных данных, контексте их использования в составе агрегатных или селективных конструкций и их привязке к прикладной информационной модели.



GenericAssociation

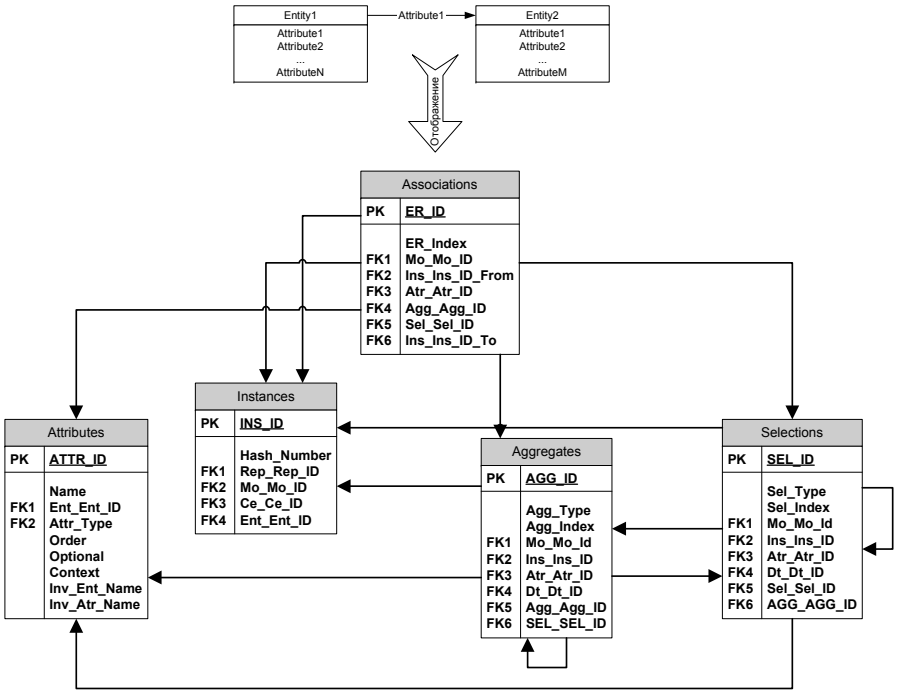


*Рис. 14. Представление ассоциативных отношений в паттерне **GenericAssociation**.*

### *3.4.4. Отображение селективных типов*

Поскольку селективные типы данных в языке EXPRESS по существу являются альтернативным представлением других базовых типов, паттерны их отображения тесно связаны с соответствующими паттернами отображения тех базовых типов, на которых они основаны. Важно отметить, что селективные типы могут быть основаны на простых типах данных, ассоциативных типах, агрегатах различного вида и на ранее предопределенных селективных типах иной организации. Дискриминатор установки селективного элемента данных в одно из альтернативных состояний в реляционной схеме представим столбцом в таблице хранения значений атрибутов объектов. Тип столбца дискриминатора соответствует рассмотренным способам отображения данных перечислимого типа ENUMERATION (см. раздел 3.4.1). А способ представления самого состояния элемента определяется одним из паттернов отображения атрибутов базовых типов. Поскольку в качестве базовых могут выступать пользовательские типы данных, эквивалентные с точки зрения способов представления

в реляционной схеме, то для исключения избыточности и минимизации затрат памяти целесообразно выделить подмножество неэквивалентных базовых типов и предусмотреть способы их адекватного реляционного представления. При этом дискриминатор селективного элемента позволит однозначно определить, в каком именно состоянии хранимые данные находятся.

Удобно различать два встречаемых на практике случая определения селективного типа. Первый случай соответствует селективным типам, базируемым только на простых и ассоциативных типах данных, учитывая возможный вложенный характер составных типов. Второй общий случай охватывает все возможные варианты определения селективного типа на основе произвольной комбинации базовых, в том числе и с участием агрегатов.

### 3.4.4.1. Паттерн Select–Columns

Данный паттерн применим к отображению атрибутов селективных типов, относящихся к первому случаю. В реляционной схеме селективный элемент такого вида представляется набором столбцов в таблице объектов класса. Один из столбцов резервируется для хранения дискриминатора селективных элементов. А остальные используются для хранения всех возможных альтернативных неэквивалентных состояний самих селективных данных. В случаях, когда селективный тип является составным вложенным, в таблице резервируется необходимое число столбцов под дискриминаторы для каждого вложенного селективного типа, участвующего в определении составного типа, и под каждое неэквивалентное состояние, в котором селективные данные могут находиться (см. Рис. 15).

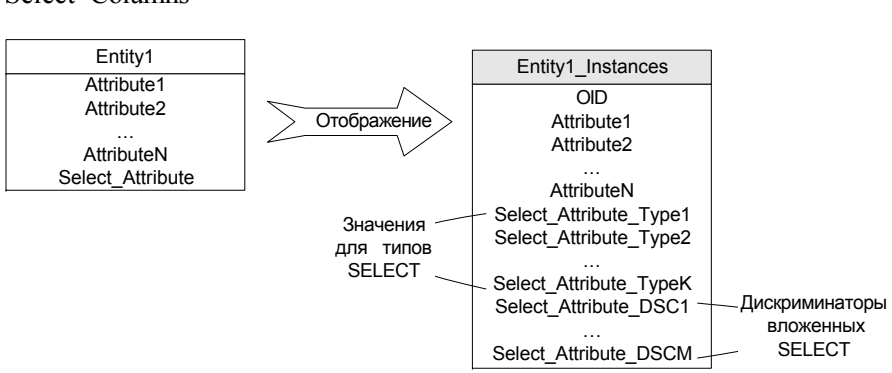


*Рис. 15. Представление селективных конструкций в паттерне **Select–Columns**.*

Достоинством данного паттерна является возможность эффективной реализации базовых операций над объектами с участием атрибутов селективного типа путем непосредственной адресации к таблице объектов. Он



может использоваться в сочетании со всеми рассмотренными выше паттернами отображения классов в рамках СЗ стратегии.

### 3.4.4.2. Паттерн ClassSelect

В тех случаях, когда в определении атрибута селективного типа участвуют агрегатные конструкции, применение рассмотренного выше паттерна является проблематичным и возникает необходимость представления значений атрибута класса отдельной таблицей **<Class>_<Attribute>_Select**. Организация данной таблицы повторяет структуру столбцов в предыдущем паттерне отображения за исключением того, что резервируются дополнительные столбцы для хранения индексов элементов агрегатов, участвующих в определении селективного типа. Число столбцов, необходимое для этого, определяется максимальной глубиной вложенности используемых конструкций упорядоченных агрегатов. Для связи с объектами используется ссылка из таблицы **<Class>_<Attribute>_Select** на соответствующую таблицу объектов классов в виде внешнего ключа записей в ней (см. Рис. 16).

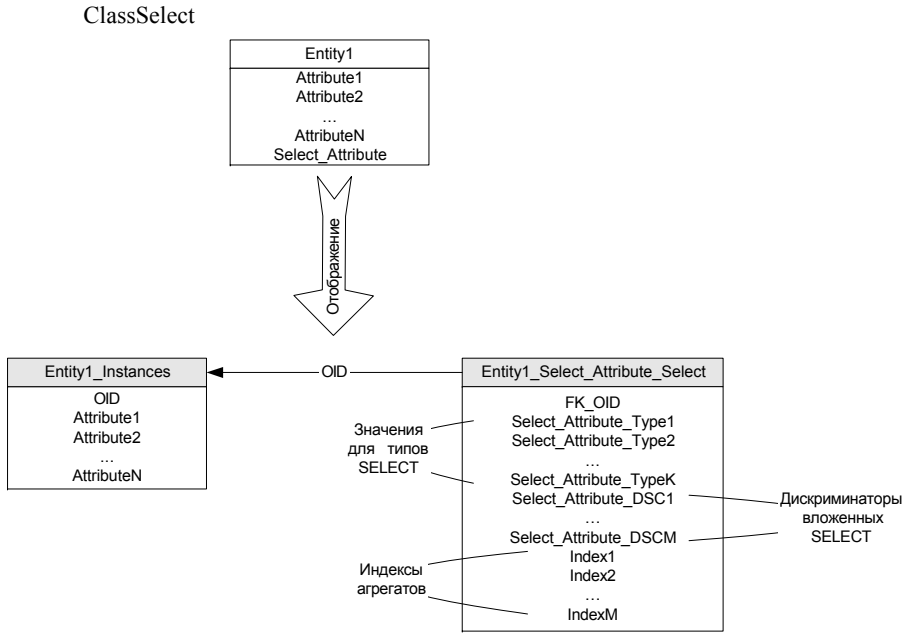


*Рис. 16. Представление селективных конструкций в паттерне **ClassSelect**.*

По сравнению с предыдущим паттерном для реализации базовых операций над объектами возникает необходимость доступа к нескольким таблицам, однако при этом он охватывает наиболее общий случай определения атрибутов произвольного селективного типа. Отметим, что вспомогательные операции со

значениями селективных атрибутов выполняются достаточно эффективно, поскольку в таблице хранятся только значения, соответствующие одному атрибуту (или нескольким однотипным атрибутам) одного класса. Однако число реляционных таблиц при отображении масштабных прикладных моделей может быть значительным с учетом повторяемости используемых селективных типов в определениях классов.

### 3.4.4.3. Паттерн HierarchySelect

Настоящий паттерн устраняет отмеченный выше недостаток за счет использования одной таблицы для каждого селективного типа, встречаемого в определении самостоятельной иерархии наследования классов. Однако контекст его использования ограничивается единственным паттерном отображения классов ***OneInheritanceHierarchy–OneTable***. Организация таблицы для каждого селективного типа повторяет предыдущий случай. Для связи с объектами используется внешний ключ записей объектов в таблице **<Hierarchy>_Instances**. Данный паттерн позволяет существенно сократить число таблиц, необходимых для реляционного представления масштабных прикладных моделей, за счет хранения однотипных селективных данных в единых таблицах. Их размер естественно возрастает, что приводит к замедлению операций работы с хранимыми селективными данными, однако общее число таблиц, критичное для большинства современных реализаций реляционных СУБД, снижается.

### 3.4.4.4. Паттерн GenericSelect

Данный паттерн предоставляет типовое обобщенное решение для реляционного представления произвольных атрибутов селективного типа. Он применяется в сочетании с паттерном отображения классов ***AllClasses–OneTable*** в рамках CH-стратегии.

Таблица **Selections** хранит записи дескрипторов селективных данных, включая их дискриминаторы и ссылки на объекты в таблице **Instances**, атрибутами которых они являются. Сами данные хранятся в таблицах представления простых типов **Integer_Elements**, **Real_Elements**, **String_Elements**, **Binary_Elements**, **Logical_Elements**, **Enum_Elements** и в таблице ассоциаций **Associations**.

Возможны случаи, когда сами селективные данные являются элементом другой родительской селективной или агрегатной конструкции. Для этого в таблице дескрипторов **Selections** предусмотрены ссылки на соответствующие родительские селективные и агрегатные элементы данных в виде столбцов внешних ключей записей в таблицах **Selections** и **Aggregates**. Такая организация таблиц (см. рис. 17) позволяет представить вложенные конструкции, составленные из данных произвольных типов. Для получения метаинформации о селективном элементе в таблице **Selections** также хранятся ссылки на соответствующие записи в таблице метаданных **Attributes**.



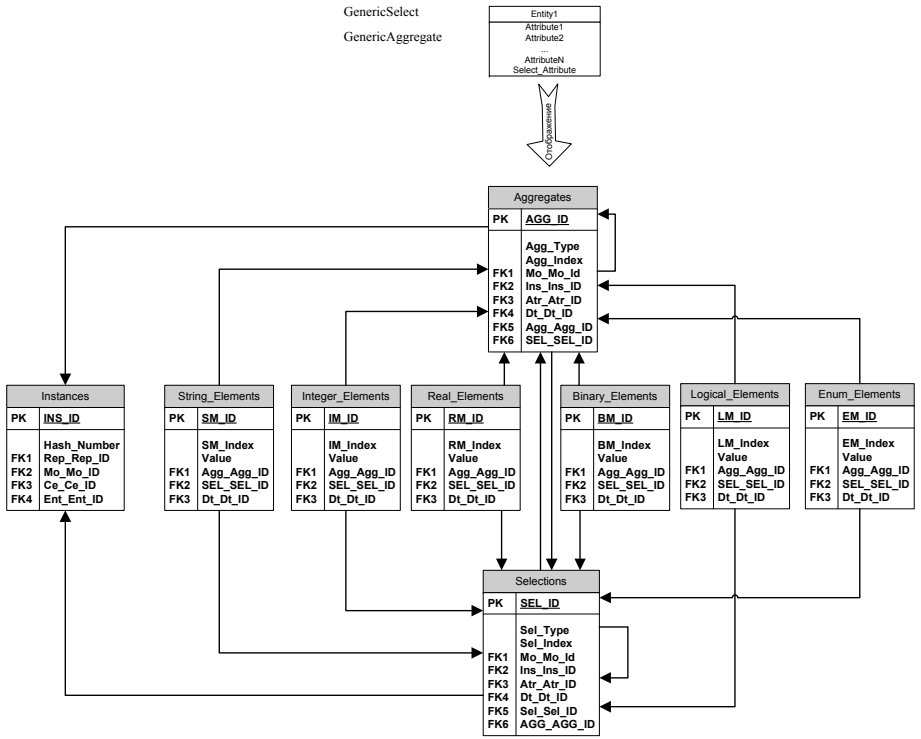


*Рис. 17. Представление составных селективных и агрегатных конструкций паттернами **GenericSelect** и **GenericAggregate**.*

### *3.4.5. Отображение агрегатов*

Паттерны отображения атрибутов агрегатных типов в реляционную схему во многом аналогичны паттернам представления селективных типов. Основные отличия затрагивают необходимость представления размерности агрегата вместо дискриминаторов в случае селективных элементов, а также организации специальных таблиц для хранения значений агрегируемых элементов, а также и их индексов в случае упорядоченных множеств и мультимножеств. Число столбцов, необходимое для представления размерности агрегата, определяется глубиной вложенности агрегатных конструкций. Число столбцов, необходимое для представления индексов агрегируемых элементов, определяется глубиной вложенности упорядоченных агрегатов. Организация столбцов для представления значений самих элементов определяется их типом. Для простых типов данная организация тривиальна. Для элементов селективного типа организация столбцов следует описаниям рассмотренных выше паттернов.

В отношении способов представления вложенных типов данных, в том числе основанных на предопределенных конструкциях, паттерны отображения

селективных и агрегатных типов довольно близки. Паттерны ***ClassAggregate*** и ***HierarchyAggregate*** предназначены для использования с паттернами отображения классов ***OneClass–OneTable***, ***OneInheritancePath–OneTable*** и ***OneInheritanceHierarchy–OneTable*** в рамках СЗ-стратегии. Паттерн ***GenericAggregate*** применяется вместе с паттерном ***AllClasses–OneTable***, соответствуя СН-стратегии отображения.

#### 3.4.5.1. Паттерн ClassAggregate

Данный паттерн предполагает организацию специальных столбцов для хранения размерностей агрегата непосредственно в таблице объектов класса, а также создание отдельной таблицы **<Class>_<Attribute>_Aggregate** для хранения значений и индексов элементов агрегата. Такая таблица создается для каждого агрегатного атрибута каждого класса. Для связи с объектами используется ссылка из **<Class>_<Attribute>_Aggregate** на соответствующую таблицу объектов классов в виде внешнего ключа записей в ней.

Паттерн охватывает наиболее общий случай определения атрибутов произвольного агрегатного типа. При этом число реляционных таблиц при отображении масштабных прикладных моделей обычно велико с учетом повторяемости эквивалентных агрегатных типов в определениях классов.

#### 3.4.5.2. Паттерн HierarchyAggregate

Настоящий паттерн уменьшает число таблиц, необходимых для представления агрегатных данных за счет использования одной таблицы для каждого агрегатного типа, встречаемого в определении самостоятельной иерархии наследования классов. Размер таких таблиц при этом увеличивается, что приводит к замедлению операций работы с хранимыми агрегатными данными, однако общее число таблиц, критичное для большинства современных реализаций реляционных СУБД, снижается. Контекст использования паттерна ограничивается соответствующей схемой отображения классов ***OneInheritanceHierarchy–OneTable***. Для связи с объектами используется внешний ключ записей объектов в таблице **<Hierarchy>_Instances**.

#### 3.4.5.3. Паттерн GenericAggregate

Данный паттерн предоставляет типовое решение для обобщенного реляционного представления произвольных атрибутов агрегатного типа (см. Рис. 17). Он применяется в сочетании с паттерном отображения классов ***AllClasses–OneTable*** в рамках СН-стратегии.

Таблица **Aggregates** хранит записи дескрипторов агрегатных данных, включая их размерности и ссылки на объекты в таблице **Instances**, атрибутами которых они являются. Таблица является родительской для элементов агрегата, хранящихся в других таблицах **Integer_Elements**, **Real_Elements**, **String_Elements**, **Binary_Elements**, **Logical_Elements**, **Enum_Elements** и **Associations**. В свою очередь, каждая запись в таблице **Aggregates** может иметь ссылку на запись в этой же таблице, если агрегат является элементом,



вложенным в другой родительский агрегат, а также хранить соответствующий индекс в нем. Если агрегат является одним из значений селективного типа, то он ссылается на соответствующую родительскую конструкцию, представляемую записью в таблице **Selections**. В остальных случаях записи таблицы **Aggregates** ссылаются на соответствующие записи в таблице **Attributes** для получения метаинформации об агрегатном атрибуте.

## 3.5. Отображение метаданных

Для представления метаданных в рамках схемонезависимой и смешанной стратегий объектно-реляционного отображения необходимо предусмотреть специальную систему таблиц, которая в свою очередь может рассматриваться как результат отображения объектно-ориентированной метамодели языка EXPRESS на реляционную модель. Поскольку полная метамодель языка EXPRESS довольно сложна, а объектно-реляционное отображение допускает множество реализаций, в том числе на основе вышеописанных паттернов, ограничимся рассмотрением следующего возможного варианта организации таблиц. Система таблиц, приведенная на Рис. 18, позволяет представить необходимую метаинформацию об EXPRESS схеме, ее составе в виде определяемых простых и объектных типов данных и организации атрибутов в классах объектов. Допустимы расширения реляционной схемы, обеспечивающие представление различного рода ограничений, допускаемых языком EXPRESS.

Таблица **Schemas** предназначена для представления информационных схем языка EXPRESS, зарегистрированных в реляционной базе данных. Она хранит первичные ключи записей и уникальные имена схем. **Defined_Types** — это таблица простых типов данных, определяемых пользователем, которая хранит первичный ключ типа, его имя, а также ссылку на базовый тип в виде внешнего ключа записи в этой же таблице. Одиннадцать предопределенных типов, соответствующих семи элементарным типам языка EXPRESS, обобщенным ассоциативному и перечисляемому типам, а также селективному и агрегатному супертипам, заносятся заранее при инициализации таблицы. Предопределенные типы являются листьями в дереве иерархии сложных типов, рекурсивно определяемых пользователем и заносимых в данную таблицу в виде отдельных записей. **Defined_Types_To_Schemas** — это таблица соответствия определяемых типов данных конкретным схемам. Связь между пользовательским типом и информационной схемой устанавливается через отдельную таблицу, а не через внешний ключ в таблице **Defined_Types**, поскольку один и тот же тип может включаться в разные схемы, если в спецификации на языке EXPRESS для него определены директивы импорта. Таблица хранит внешние ключи определяемых пользователем типов и соответствующих им информационных схем. Пара внешних ключей «тип–схема» формирует составной первичный ключ записи в таблице **Defined_Types_To_Schemas**, чем контролируется уникальность включения типа в одну и ту же схему.

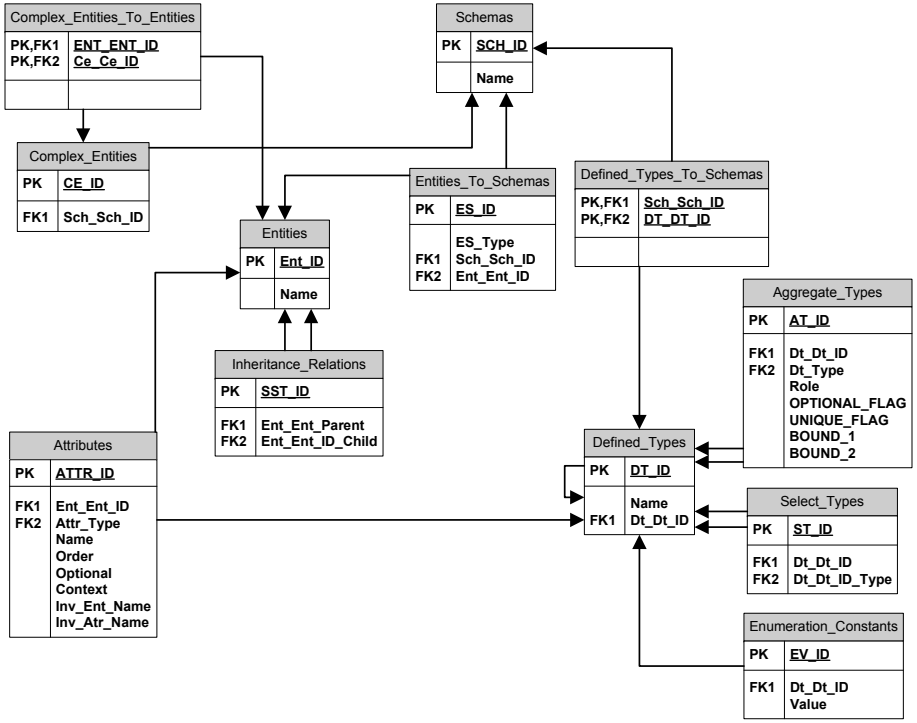


*Рис. 18. Возможное реляционное представление метаданных для объектно-ориентированных моделей.*

Для детального описания перечислимых, селективных и агрегатных типов дополнительно используются таблицы **Enumeration_Constants**, **Select_Types** и **Aggregate_Types**. Таблица **Enumerations_Constants** содержит списки возможных значений для каждого перечислимого пользовательского типа. Ее столбцы хранят символьные значения перечислимого типа и внешний ключ соответствующей записи типа в таблице **Defined_Types**. Аналогичным образом, таблица **Select_Types** представляет списки базовых типов, входящих в определение каждого конкретного селективного типа, в виде внешних ключей записей типов в таблице **Defined_Types**. В столбцах таблицы агрегатных типов **Aggregate_Types** содержится информация о типе агрегата (ARRAY, LIST, SET или BAG), базовом типе его элементов в виде внешнего ключа соответствующей записи в таблице **Defined_Types**, разрешенных значениях нижней и верхней границ агрегата, признаках допустимости наличия уникальных и неопределенных элементов.

Таблица классов **Entities** предназначена для представления объектных типов информационной схемы, зарегистрированной в базе данных. Ее столбцы хранят первичные ключи записей и уникальные имена классов. Аналогично



определяемым типам, привязка классов к схеме осуществляется через отдельную таблицу соответствия **Entities_To_Schemas**. Для реконструкции отношений наследования между классами используется таблица **Inheritance_Relations**, в которой данные отношения представлены парами внешних ключей записей классов родителей и потомков в таблице **Entities**. Поскольку язык EXPRESS допускает множественное наследование с признаками AND или ANDOR, важно иметь альтернативное представление иерархии наследования в виде множеств всех родительских классов, данные которых включаются в конструируемые объекты сложных классов. Для этой цели используется таблица **Complex_Entities**. Для единообразия обработки объектов простые классы также представляются записями этой таблицы. Таблица **Complex_Entities_To_Entities** хранит все соответствия сложных классов родительским классам в виде пары внешних ключей записей в таблицах **Complex_Entities** и **Entities**.

Наконец, таблица атрибутов **Attributes** содержит столбцы для представления имени атрибута, класса, в котором данный атрибут определяется (или переопределяется), в виде соответствующего внешнего ключа записи в таблице **Entities**, типа атрибута в виде внешнего ключа записи в таблице **Defined_Types**, признака обязательности значений и контекста использования (EXPLICIT, DERIVED или INVERSE).

В заключение отметим, что приведенная схема достаточно удобна для реализации промежуточного объектно-реляционного слоя в рамках CH- и CM-стратегий непосредственно средствами реляционной СУБД. Вместе с тем, возможен ряд ее модификаций, связанных с иными способами реляционного представления метамодели EXPRESS путем использования альтернативных паттернов отображения прикладных объектно-ориентированных моделей, рассмотренных выше.

## *4. Реализация промежуточного OP-слоя в среде Oracle9*

В настоящее время в рамках проекта создания программной платформы для интеграции приложений ведется разработка промежуточного объектно-реляционного слоя общего назначения. OP-слой предназначен для работы с произвольными прикладными объектно-ориентированными данными, модели которых описаны на языке EXPRESS.

Для интегрируемых приложений OP-слой предоставляет программные объектно-ориентированные интерфейсы доступа к прикладным данным на некоторых популярных языках реализации (см. Рис. 19). Организация этих интерфейсов следует перечисленным выше принципам прозрачного манипулирования хранимыми данными, декларируемым Манифестом объектно-ориентированных баз данных. Интерфейсы предоставляют функционально развитый набор операций для манипулирования хранимыми и временными объектами, включая операции создания, модификации, удаления объектов, навигации по их однонаправленным и двунаправленным

ассоциативным связям и выборки объектов на основе языка запросов. Запросы базируются на конструкции QUERY языка EXPRESS, позволяющей задать произвольный предикат на множестве объектов и отобрать те из них, которые удовлетворяют условию данного предиката. Интерфейсы предусматривают несколько пессимистических и оптимистических моделей транзакций с различными способами изоляции на уровне отдельных прикладных объектов и самостоятельных объектных популяций, содержательных для коллективных пользовательских сессий и участвующих в них приложений.

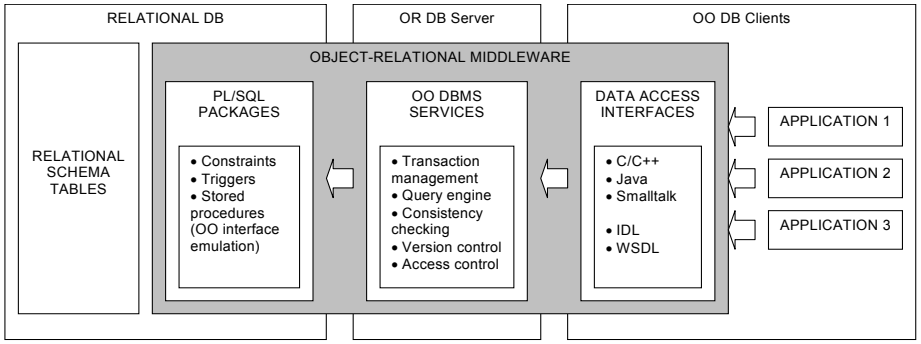


*Рис. 19. Общая организация промежуточного объектно-реляционного слоя.*

По спецификации данные интерфейсы совместимы с соответствующими частями международного информационного стандарта по интероперабельности STEP и поэтому обеспечивают интегрируемость широкого класса унаследованных и вновь создаваемых программных систем научного и промышленного назначения.

В качестве хранилища данных реализация OP-слоя предусматривает использование реляционных СУБД со схемами, основанными на рассмотренных выше паттернах объектно-реляционного отображения. При этом функции по управлению транзакциями, разрешению запросов, контролю целостности данных, управлению версиями и контролю прав доступа распределяются между сервером OP-слоя, через который непосредственно взаимодействуют приложения, и реляционной СУБД, выступающей в роли вторичного хранилища данных.

Важнейшими функциями, реализуемыми непосредственно средствами реляционной СУБД, являются операции манипулирования хранимыми объектами и выполнения простых объектных запросов. С этой целью на языке PL/SQL разрабатываются пакеты программ, эмулирующие объектно-ориентированные интерфейсы доступа к данным путем предоставления функциональных средств для создания, модификации, поиска и удаления объектов. Поскольку полная поддержка объектного языка запросов средствами реляционной СУБД представляется проблематичной с учетом разнообразия императивных конструкций языка EXPRESS, пакеты программ выполняют



простейшие виды запросов на основе хранимых объектных идентификаторов (PID), объектных типов и навигационных маршрутов в виде графов переходов по ассоциативным связям типизированных объектов. Поддерживая кэширование объектов, посредник в ряде случаев разрешает запросы самостоятельно, а иногда переадресовывает их реляционной СУБД. При этом происходит редукция клиентского запроса, представленного в общей форме, к запросу упрощенного вида, расширяющего множество объектов и выполнимого пакетом программ реляционной СУБД. Результаты затем обрабатываются посредником с целью исключения объектов, полученных в результате интерпретации упрощенного запроса и не удовлетворяющих исходному.

Поскольку выбор стратегии отображения для реализации OP-слоя подобной функциональности крайне неоднозначен с учетом разнообразия потенциальных приложений, предполагается реализация и поддержка нескольких альтернативных стратегий, а именно: схемонезависимого, схемозависимого и BLOB подходов. Они базируются на тех или иных сочетаниях рассмотренных выше паттернов OP-отображения и используют собственные пакеты программ на PL/SQL для реализации базовой функциональности OP-слоя. Хотя все пакеты реализуют семантически эквивалентные наборы операций для манипулирования объектами и их поиска, их внешние интерфейсы не допускают унификацию в силу ограниченных возможностей языка SQL при формировании клиентских запросов со стороны OP-посредника для специфических реляционных схем представления объектно-ориентированных моделей данных. Для адаптации посредника к иным OP-стратегиям в его архитектуре предусмотрены специальные компоненты-адаптеры, обеспечивающие требуемую виртуализацию хранилищ данных. Каждый адаптер реализуется с учетом специфики конкретного OP-отображения.

В качестве целевой платформы реализации промежуточного OP-слоя выбрана СУБД Oracle9 [18].

## 4.1. CH-стратегия

Разработанная схемонезависимая стратегия состоит в применении обобщенных паттернов ***AllClasses–OneTable***, ***Attribute–Table***, ***GenericAssociation***, ***GenericSelect***, ***GenericAggregate*** для отображения схем, классов и атрибутов, а также паттерна представления соответствующих метаданных прикладной модели реляционными таблицами.

Реализованные в среде Oracle9 PL/SQL пакеты обеспечивают выполнение всего базового набора операций с хранимыми объектно-ориентированными данными и запросов к ним. Обобщенная, независимая от конкретных прикладных моделей реализация PL/SQL процедур и функций основана на совместном одновременном использовании данных и метаданных, хранимых в системах таблиц в соответствии с перечисленными паттернами отображения. Приведем описание основных пакетных методов для объектно-реляционного отображения в качестве иллюстрации CH-стратегии.

Пакет **lb_defined_types** для работы с метаинформацией о пользовательских типах данных, определенных EXPRESS схемой:

- procedure Register_Defined_Type – регистрация в БД пользовательского типа схемы;
- procedure Save_Enum_Type – сохранение метаданных для перечислимого типа;
- procedure Save_Select_Type – сохранение метаданных для селективного типа;
- procedure Save_Aggregate_Type – сохранение метаданных для агрегатного типа;
- function Get_Type – выдача метаинформации о пользовательском типе данных.

Пакет **lb_entity** предназначен для работы с метаинформацией, относящейся к объектным типам EXPRESS схемы:

- function Register_Entity – регистрация объектного типа схемы;
- procedure Save_Attribute – сохранение метаданных для атрибута, определяемого в регистрируемом объектном типе;
- procedure Save_Inheritance_Relations – сохранение информации о подтипах и супертипах регистрируемого объектного типа;
- function Add_Entity_From_Schema – экспортирование информации об объектном типе из другой схемы;
- function Get_Entity – выдача метаинформации об объектном типе;
- function Get_Attribute – выдача метаинформации об атрибуте, определяемом в объектном типе схемы.

Пакет **lb_instance** предназначен для работы с данными: занесения данных в базу, а также для извлечения данных из нее на основе запросов:

- function Init_Instance – инициация сохранения объекта;
- procedure Init_Attribute_List – инициация сохранения значений атрибутов объекта;
- procedure Put_Simple_Attribute_{R, I, S, B, L, E} – сохранение значения атрибута вещественного, целочисленного, символьного, бинарного, логического, перечислимого типа соответственно;
- procedure Put_Association – сохранение значения атрибута ассоциативного типа;
- function Put_Aggregate – инициация сохранения элементов агрегата;
- function Put_Select – инициация сохранения селективного элемента;
- procedure Put_Element_{R, I, S, B, L, E} – сохранение значения элемента агрегатной или селективной конструкции вещественного, целочисленного, символьного, бинарного, логического, перечислимого типа соответственно;
- procedure Get_Instances_By_ID – выборка объектов по заданным идентификаторам;
- procedure Get_Instances_By_Type – выборка объектов по заданному типу;



- procedure Get_Instances_By_Route – выборка объектов по заданному навигационному маршруту;
- procedure Add_Route_Path – метод формирования навигационного маршрута;
- procedure Get_All_Instances – выборка всех объектов;
- procedure Delete_Instances – удаление объектов по заданным идентификаторам.

До начала работы с прикладными данными соответствующие таблицы метаданных должны быть проинициализированы. С этой целью разработан CASE инструмент, позволяющий автоматически сгенерировать скрипт инициализации соответствующих таблиц на языке PL/SQL по заданной EXPRESS спецификации прикладной модели. Фрагмент данного скрипта для информационной схемы `ActorResource`, приведенной на Рис. 1, представлен ниже.

```
declare
  l_Sch_ID Schemas.sch_id%TYPE;
  l_Ent_ID Entities.ent_id%TYPE;
begin
…
  lb_defined_types.register_defined_type('Label','string',l_Sch_ID);
  lb_defined_types.register_defined_type('ActorRole','Label',l_Sch_ID);
  lb_defined_types.register_defined_type('AddressTypeEnum','enumeration',
l_Sch_ID);
  lb_defined_types.save_enum_type('OFFICE');
  lb_defined_types.save_enum_type('HOME');
  lb_defined_types.save_enum_type('USERDEFINED');
  l_Ent_ID := lb_entity.register_entity('Organization',l_Sch_ID,false);
  lb_entity.save_attribute('Id','integer',1,'','',0,'E');
  lb_entity.save_attribute('Name','Label',2,'','',0,'E');
  lb_entity.save_attribute('Description','string',3,'','',1,'E');
  lb_entity.save_attribute('Roles','aggregate',4,'','',0,'E');
  lb_entity.save_attribute('Addresses','aggregate',5,'','',0,'E');
  lb_entity.save_attribute('IsRelatedBy','OrganizationRelationship',6,'Organ
izationRelationship','RelatedOrganizations',0,'I');
  lb_entity.save_attribute('Relates','OrganizationRelationship',7,'Organizat
ionRelationship','RelatingOrganization',0,'I');
  lb_entity.save_attribute('Engages','Person',8,'Person','EngagedIn',0,'I');
  l_Ent_ID := lb_entity.register_entity('Address',l_Sch_ID,false);
  lb_entity.save_attribute('Purpose','AddressTypeEnum',1,'','',0,'E');
  lb_entity.save_attribute('UserDefinedPurpose','string',2,'','',1,'E');
  lb_entity.save_attribute('OfPerson','Person',3,'Person','Addresses',0,'I');
  lb_entity.save_attribute('OfOrganization','Organization',4,'Organization',
'Addresses',0,'I');
  l_Ent_ID := lb_entity.register_entity('PostalAddress',l_Sch_ID,false);
  lb_entity.save_inheritance_relations('Address',false);
  lb_entity.save_attribute('AddressLines','aggregate',1,'','',0,'E');
…
end;
```

При непосредственной работе с данными адаптер CH стратегии осуществляет динамическую трансляцию базовых операций манипулирования объектами того или иного типа в соответствующую последовательность вызовов PL/SQL функций и процедур. На следующем примере можно проследить логику генерации подобных последовательностей. Аналогичным образом реализуются операции модификации, удаления и поиска объектов на основе хранимых идентификаторов, объектных типов и маршрутов навигации.

```
-- Фрагмент исходного файла с данными в формате ISO-10303-21
#10=POSTALADDRESS(.OFFICE., $, ('25, B.Kommunisticheskaya str., Moscow,
109004, Russia'));
#11=ORGANIZATION(770901001, 'ISP RAS', $,
('Research','Development','Teaching'), (#10));

-- Фрагмент PL/SQL скрипта для занесения данных в БД
DECLARE
   type Agg_Level_Type is table of Aggregates.agg_type%TYPE index by
binary_integer;
   type Agg_Level_ID is table of Aggregates.agg_id%TYPE index by
binary_integer;
   l_Sch_ID Schemas.sch_id%TYPE;
   l_Ent_ID Entities.ent_id%TYPE;
   l_Ins_ID Instances.ins_id%TYPE;
   l_Atr_ID Attributes.atr_id%TYPE;
   l_Agg_Type Agg_level_Type;
   l_Agg_ID Agg_level_ID;
…
BEGIN
…
   l_Ent_ID := lb_entity.get_entity('Organization',l_Sch_ID);
   l_Ins_ID := lb_instance.init_instance(l_Ent_ID,l_MO_ID,l_Rep_ID,'#11');
   lb_instance.init_attribute_list(l_Ent_ID);
   l_Atr_ID := lb_entity.get_attribute(1);
   lb_instance.put_simple_attribute_i(l_Ins_ID,l_Atr_ID,770901001);
   l_Atr_ID := lb_entity.get_attribute(2);
   lb_instance.put_simple_attrib_s(l_Ins_ID,l_Atr_ID,'ISP RAS');
   l_Atr_ID := lb_entity.get_attribute(4);
   l_Agg_Type(1) := lb_defined_types.get_type('ActorRole',l_Sch_ID);
   l_Agg_ID(1) :=
lb_instance.put_aggregate(l_Agg_Type(1),NULL,NULL,l_Atr_ID,l_Ins_ID,l_MO_ID,
NULL,NULL);
   lb_instance.put_element_s(0,'Research',NULL,l_Agg_ID(1),NULL);
   lb_instance.put_element_s(1,'Development',NULL,l_Agg_ID(1),NULL);
   lb_instance.put_element_s(2,'Teaching',NULL,l_Agg_ID(1),NULL);
   l_Atr_ID := lb_entity.get_attribute(5);
   l_Agg_Type(1) := lb_defined_types.get_type('Address',l_Sch_ID);
   l_Agg_ID(1) :=
lb_instance.put_aggregate(l_Agg_Type(1),NULL,NULL,l_Atr_ID,l_Ins_ID,l_MO_ID,
NULL,NULL);
   lb_instance.put_association(l_Ins_ID,l_Atr_ID,'#10',0,l_Agg_ID(1),NULL);
```



```
   l_Ent_ID := lb_entity.get_entity('PostalAddress',l_Sch_ID);
   l_Ins_ID := lb_instance.init_instance(l_Ent_ID,l_MO_ID,l_Rep_ID,'#10');
   lb_instance.init_attribute_list(l_Ent_ID);
   l_Atr_ID := lb_entity.get_attribute(1);
   lb_instance.put_simple_attribute_e(l_Ins_ID,l_Atr_ID,'office');
   l_Atr_ID := lb_entity.get_attribute(3);
   l_Agg_Type(1) := lb_defined_types.get_type('Label',l_Sch_ID);
   l_Agg_ID(1) :=
lb_instance.put_aggregate(l_Agg_Type(1),NULL,NULL,l_Atr_ID,l_Ins_ID,l_MO_ID,
NULL,NULL);
   lb_instance.put_element_s(0,'25, B.Kommunisticheskaya str., Moscow,
109004, Russia',NULL,l_Agg_ID(1),NULL);
…
END;
```

## 4.2. C3-стратегия

Альтернативу рассмотренному способу реализации OP-отображения составляет разработанный вариант схемозависимой стратегии, основанный на использовании паттернов отображения классов и атрибутов ***OneInheritancePath–OneTable***, ***Attribute–Column***, ***ClassAssociation***, ***ClassSelect*** и ***ClassAggregate***. Данный вариант представляет собой попытку оптимизировать реляционную схему для наиболее эффективной работы с данными внутри одной прикладной модели. Подобная постановка задачи возникает довольно часто на практике и представляет интерес для приложений, оперирующих с одной, возможно масштабной, междисциплинарной прикладной моделью. Указанное сочетание паттернов отображения приводит к большому количеству таблиц, требуемых для адекватного представления объектно-ориентированных данных. Однако при этом оно обеспечивает более эффективную реализацию базовых операций манипулирования объектами. Паттерн ***OneInheritancePath–OneTable*** использует преимущества сериализованного представления атрибутов конкретных классов и упрощает компоновку наследуемых атрибутов для объектов выбранных типов. Перечисленные паттерны отображения исключают многоуровневую косвенную адресацию при доступе к таблицам атрибутов и обеспечивают высокую эффективность реализации вспомогательных операций сборки значений из таблиц атрибутов при чтении объектов и их рассылку по соответствующим таблицам при записи и модификации объектов.

Реализация адаптера посредника для C3-стратегии существенно отличается от реализации CH-стратегии. Во-первых, реляционная схема для СУБД генерируется соответствующим CASE инструментом для каждой прикладной модели, специфицированной на языке EXPRESS. Ниже представлен фрагмент описания такой схемы на языке SQL для прикладной модели `ActorResource`, приведенной на Рис. 1. Во-вторых, одновременно со схемой генерируются исходные тексты пакета процедур на языке PL/SQL для манипулирования объектами данной прикладной модели. Каждая процедура пакета ориен-

тирована на работу с объектами определенного типа и имеет специфический для него интерфейс. Поддержка подобных хранимых процедур со стороны реляционной СУБД существенно упрощает реализацию адаптера для посредника и позволяет повысить эффективность его работы за счет компиляции соответствующих директив манипулирования объектами в среде самой СУБД. В-третьих, не требуется какая-либо работа с метаданными, поскольку организация таблиц данных следует структурным особенностям прикладной информационной модели и позволяет явно адресоваться к ним при работе.

```
-- создание таблицы для описателей объектов схемы ActorResource
CREATE TABLE actorresource_instance (
        PID INTEGER DEFAULT 1 NOT NULL PRIMARY KEY,
        Title VARCHAR2(128) NOT NULL,
        Entity VARCHAR2(128) NOT NULL,
        Model INTEGER NOT NULL,
        Commentary VARCHAR2(4000),
        FOREIGN KEY (Model) REFERENCES model(PID) ON DELETE CASCADE
);

CREATE SEQUENCE sq$actorresource_instance;
CREATE UNIQUE INDEX i$actorresource_title_model ON actorresource_instance
(Title, Model);
CREATE INDEX i$actorresource_entity_model ON actorresource_instance (Entity,
Model);
CREATE INDEX i$actorresource_model ON actorresource_instance (Model);

-- таблица для объектов типа Organization
CREATE TABLE actorresource_organization (
        PID INTEGER DEFAULT 1 NOT NULL PRIMARY KEY,
        Instance INTEGER NOT NULL,
        Id_ INTEGER NOT NULL,
        Name_ VARCHAR2(255) NOT NULL,
        Description_ VARCHAR2(4000),
        FOREIGN KEY (Instance) REFERENCES actorresource_instance(PID) ON
DELETE CASCADE
);
CREATE SEQUENCE sq$actorresource_organization;
CREATE INDEX i$actorresource_organization ON actorresource_organization
(Instance);

CREATE TABLE actorresource_organizat_3 (
        PID INTEGER DEFAULT 1 NOT NULL PRIMARY KEY,
        Parent INTEGER NOT NULL,
        Element_Index1 INTEGER,
        Element_Value VARCHAR2(255),
        FOREIGN KEY (Parent) REFERENCES actorresource_organization(PID) ON
DELETE CASCADE
);
CREATE SEQUENCE sq$actorresource_organizat_3;
CREATE INDEX i$actorresource_organizat_3 ON actorresource_organizat_3
(Parent);
```



```
CREATE TABLE actorresource_organizat_4 (
        PID INTEGER DEFAULT 1 NOT NULL PRIMARY KEY,
        Parent INTEGER NOT NULL,
        Element_Index1 INTEGER,
        Element_Value VARCHAR2(128),
        FOREIGN KEY (Parent) REFERENCES actorresource_organization(PID) ON
DELETE CASCADE
);
CREATE SEQUENCE sq$actorresource_organizat_4;
CREATE INDEX i$actorresource_organizat_4 ON actorresource_organizat_4
(Parent);
…
```

### 4.3. BLOB-стратегия

Наконец, третий разработанный вариант реализации адаптера основан на применении BLOB&Text&XML_Encoding паттернов для реляционного представления объектов классов и их атрибутов. Этот вариант воспроизводит упрощенную схему CH-стратегии за счет упакованного представления значений атрибутов объекта в виде бинарной или текстовой строки. Сами строки хранятся как элементы записей в таблице объектов **BLOB_Instances**. Как следствие, таблицы представления простых и сложных атрибутов отсутствуют. Модифицированная таблица **Associations** хранит ассоциации всех видов и используется для реализации навигационных запросов по ним. Из таблиц метаданных поддерживаются лишь **Schemas**, **Entities**, **Entities_To_Schemas**, **Inheritance_Relations**, **Complex_Entities** и **Complex_Entities_To_Entities**, записи которых воспроизводят отношения наследования классов в прикладной модели и используются для реализации запросов по типам объектов. Соответствующий CASE инструмент позволяет сгенерировать скрипт инициализации таблиц метаданных по спецификации прикладной информационной модели на языке EXPRESS.

Разработанный на языке PL/SQL пакет процедур предоставляет базовый набор операций манипулирования объектами и их поиска по идентификаторам, типам и навигационным маршрутам в рамках BLOB стратегии. Поскольку значения атрибутов объекта представлены в БД единой строкой, функции по упаковке и распаковке строк целиком ложатся на адаптер посредника. В силу этой же причины в рамках BLOB стратегии невозможно выполнение более тонких запросов на основе атрибутных свойств объектов непосредственно средствами СУБД.

## *5. Заключение. Рекомендации использования*

Таким образом, на основе выделенных паттернов проведена систематизация методов объектно-реляционного отображения. Паттерны отображения информационных схем, классов, атрибутов, метаданных и их сочетания приводят к существенным различиям в организации реляционных таблиц и способах реализации промежуточного OP-слоя. Получаемые решения

обладают разной степенью эффективности, гибкости и адаптируемости к развитию прикладных моделей.

Так, приведенный пример реализации схемонезависимой стратегии ОР-отображения, основанный на фиксированной системе таблиц, может быть рекомендован для использования в приложениях, оперирующих одновременно с несколькими перманентно изменяемыми моделями либо с масштабными промышленными моделями, включающими тысячи классов. Однако эффективность выполнения запросов, а также базовых операций манипулирования объектами при использовании данной стратегии оказывается низкой, поскольку их реализация связана с необходимостью дополнительного анализа таблиц метаданных, а также сборки значений атрибутов объектов из нескольких таблиц при чтении и их обратной рассылки по таблицам при записи.

Частично компенсировать данные недостатки, а также сократить количество необходимых реляционных таблиц позволяет упрощенный вариант реализации схемонезависимой стратегии, основанный на представлении значений атрибутов, упакованных в единую бинарную или текстовую строку (BLOB) и хранимых как элементы записей в таблице объектов. Недостатком данной стратегии является невозможность непосредственной реализации запросов и базовых операций манипулирования объектами средствами самой СУБД. Вся нагрузка здесь ложится на промежуточный слой, выполняющий операции упаковки/распаковки строк со значениями атрибутов. Реализованный вариант BLOB-стратегии, описанный в настоящей статье, позволяет разгрузить слой-посредник и выполнить простые запросы по идентификаторам объектов, их типам, а также навигационным маршрутам средствами СУБД, поскольку использует дополнительную систему таблиц для хранения отношений наследования и ассоциативных связей между отдельными объектами.

Наиболее эффективную реализацию запросов и операций манипулирования объектами обеспечивает разработанный вариант схемозависимой стратегии, основанный на сериализованном представлении атрибутов конкретных классов. Данная стратегия рекомендуется для использования в приложениях, оперирующих с одной прикладной моделью, включающей несколько сотен классов. Ее недостатками являются большое количество используемых таблиц, критичное для большинства реализаций современных реляционных СУБД, чувствительность к эволюции прикладной модели, а также необходимость применения CASE инструментария для генерации реляционной схемы и процедур, реализующих запросы и операции манипулирования объектами. Подобный инструментарий позволяет существенно упростить сопровождение и администрирование базы данных, эксплуатирующей данную стратегию объектно-реляционного отображения.

Таким образом, разрабатываемый программно-инструментальный комплекс предоставляет развитые средства для эффективной организации промежуточного ОР-слоя в типовых прикладных контекстах. Представленные рекомендации могут служить конструктивной основой для выбора наиболее



оптимальных решений.

**Литература**


1. W. Keller. Object/Relational Access Layers — A Roadmap, Missing Links and More Patterns. // Proceedings of the 3rd European Conference on Pattern Languages of Programming and Computing (EuroPLoP), 1998, http://www.objectarchitects.de/ObjectArchitects/papers/Published/ZippedPapers/or06_proceedings.pdf.
2. В.П. Иванников, С.С. Гайсарян, К.В. Антипин, В.В. Рубанов. Объектно-ориентированное окружение, обеспечивающее доступ к реляционным СУБД. // Труды Института системного программирования РАН, том 2, 2001, с. 89–114.
3. M. P. Atkinson, F. Bancilhon, D. J. DeWitt, K. R. Dittrich, D. Maier, S. B. Zdonik. The Object-Oriented Database System Manifesto. // Proceedings of the First International Conference on Deductive and Object-Oriented Databases (DOOD'89), 1989, pp. 223–240.
4. J. Eggers. Implementing EXPRESS in SQL. Document ISO TC184/SC4/WG1/N292, October 1988.
5. M. Mead, D. Thomas. Proposed Mapping from EXPRESS to SQL. Technical Report, Rutherford Appleton Laboratory, May 1989.
6. K.C. Morris. Translating EXPRESS to SQL: A User's Guide. Technical Report NISTIR 4341, National Institute of Standards and Technology, Gaithersburg, Maryland, May 1990.
7. D. Sanderson, D. Spooner. Mapping between EXPRESS and Traditional DBMS Models. Proceedings of EUG‘93 — The Third EXPRESS Users Group Conference, Berlin, October 2–3, 1993.
8. L. Klein, A. Stonis, D. Jancauskas. EXPRESS/SQL white paper. Document ISO TC184/SC4/WG11/N144, February 2001.
9. ISO 10303: 1994, Industrial automation systems and integration — Product data representation and exchange.
10. ISO 10303-11: 1994, Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.
11. Unified Modeling Language (UML), Version 1.5, 2003, http://www.omg.org/cgi-bin/doc?formal/03-03-01.
12. The Object Data Standard: ODMG 3.0. eds. R.G.G. Cattel, D.K. Barry. Morgan Kauffmann, 2000.
13. W. Keller. Mapping Objects to Tables. A Pattern Language. // Proceedings of the 2nd European Conference on Pattern Languages of Programming and Computing (EuroPLoP), 1997, http://www.objectarchitects.de/ObjectArchitects/papers/Published/ZippedPapers/mappings04.pdf.
14. K. Brown, B.G. Whitenack. Crossing Chasms: A Pattern Language for Object-RDBMS Integration. White Paper, Knowledge Systems Corp., 1995, http://www.ksc.com/articles/staticpatterns.htm.
15. J. Coldewey. Decoupling of Object-Oriented Systems — A Collection of Patterns, Version 1.1. Coldewey Consulting, Munich, July 2000, http://www.coldewey.com/publikationen/Decoupling.1.1.PDF.
16. P. Heinckiens. Building Scalable Database Applications: Object-Oriented Design, Architectures, and Implementations. Addison-Wesley, 1998, ISBN 0-2013101-3-9.
17. ANSI X3.135-1992, American National Standard for Information Systems — Database Language — SQL, November 1992.
18. PL/SQL User’s Guide and Reference, Release 2 (9.2), Part No. A96624-01, Oracle Corp., March 2002.