УДК 517.956.223

ОБ ОДНОМ ПОДХОДЕ К АВТОМАТИЗАЦИИ СОЗДАНИЯ ПРИЛОЖЕНИЙ, ОРИЕНТИРОВАННЫХ НА РАБОТУ СО СЛОЖНЫМИ СТРУКТУРАМИ ДАННЫХ

О. Б. Арушанян[1], Н. А. Богомолов[1], А. Д. Ковалев[1], Н. И. Волченскова[1]

Описана архитектура разработанного авторами базового программного обеспечения, предназначенного для автоматизации создания приложений, ориентированных на интерактивную работу со сложными структурами данных. Рассматриваются концепция разработки и основные компоненты программного обеспечения. Работа выполнена при поддержке РФФИ (код проекта 04–07–90288).

Ключевые слова: программное обеспечение, инструментальные программные средства, виртуальная память, язык Паскаль, базы данных, Delphi.

1. Общие замечания. Работа посвящена вопросам создания инструментального программного комплекса (далее ИК), предназначенного для автоматизации разработки широкого класса приложений, ориентированных на интерактивное создание и редактирование в однопользовательском режиме сложных структур данных, общий объем которых не превышает размера доступной виртуальной памяти компьютера (1 – 3 гигабайта для современных персональных компьютеров). К этой категории относятся, в частности, приложения, предназначенные для обработки и сопровождения больших массивов текстов программ на алгоритмических языках и документации этих программ на естественном языке.

Автоматизация создания такого рода приложений достигается за счет использования на всех этапах разработки описываемых ниже компонент инструментального комплекса.

Следует отметить, что использование единого инструментального программного обеспечения для создания ряда независимых приложений из смежных областей существенно облегчает обмен данными между этими приложениями, а также, в случае необходимости, и задачу их последующей интеграции в единое комплексное приложение.

Описываемый инструментальный комплекс является надстройкой над базовым языком программирования приложения в виде библиотек классов и программ. В качестве такого языка был выбран язык Object Pascal [1], используемый в среде быстрой разработки приложений Delphi [2].

2. Средства создания базы данных приложения. Принятые ограничения на объем данных позволяют во время работы приложения разместить все содержимое базы данных (БД) в виртуальной памяти компьютера, что обеспечивает возможность хранения связей между объектами в виде прямых ссылок по памяти. Это оказывает существенное влияние на реализацию средств организации БД, позволяя значительно ускорить работу процедур поиска и манипулирования данными.

Модель данных, которую можно реализовать с помощью средств организации данных описываемого ИК, является объектно-ориентированной. Однако по сравнению с объектно-ориентированной моделью данных стандарта ODMG-99 [3], который был разработан исследовательской группой Object Data Management Group, модель данных ИК можно охарактеризовать как *неполную*. Неполнота объектно-ориентированной модели данных ИК состоит, прежде всего, в отсутствии типизации связей между объектами на уровне языка определения данных. Таким образом, модель данных ИК занимает промежуточное положение между объектно-ориентированной моделью данных стандарта ODMG и моделью *полуструктурированных данных* (semistructured data) [4, 5], для которой характерно полное отсутствие схемы данных, задаваемой отдельно от самих данных.

Отсутствие в языке определения данных ИК схемы связей между объектами обусловлено стремлением поддержать работу с плохо формализуемыми структурами данных, а также потребностью агрегирования баз данных приложений, имеющих различные схемы данных. Предполагается, что обеспечение корректности объектных связей должно реализовываться на уровне бизнес-логики конкретного приложения.

2.1. Организация БД. Базовые понятия. Структуры данных, создаваемые средствами организации данных ИК, представляют собой множество объектов, объединенных направленными связями. Такие

[1] Научно-исследовательский вычислительный центр, Московский государственный университет им. М. В. Ломоносова, 119992, Москва; e-mail: arush@srcc.msu.su, nbogom@srcc.msu.su, kovalev@srcc.msu.su

структуры могут быть описаны как упорядоченный направленный граф, вершинам которого соответствуют объекты базы данных, а дугам — связи между ними.

Будем называть начальную вершину дуги *родительской* для конечной вершины, а конечную вершину — *дочерней* для начальной вершины. Вершину, у которой нет родительской вершины, будем называть *корневой вершиной*, или *корнем графа*.

Ориентированным путем из некоторой начальной вершины к некоторой конечной вершине называют последовательность дуг, для которых конечная вершина любой дуги является либо начальной вершиной следующей дуги, либо конечной вершиной пути. Ориентированный путь называют *простым*, если начальные вершины всех этих дуг различны и их конечные вершины также различны.

Граф, имеющий единственную корневую вершину, от которой до всех остальных вершин существует единственный простой ориентированный путь, является *ориентированным непересекающимся деревом*, или просто *деревом*. Если же от корня до какой-либо вершины существует два или более простых ориентированных пути, то будем называть такой ориентированный граф *пересекающимся деревом*.

Граф, имеющий несколько корневых вершин, состоит из нескольких деревьев. Будем называть такой граф *лесом*. Если два и более дерева имеют хотя бы одну общую вершину, то будем называть такие деревья *взаимно-пересекающимися*, а эту вершину — *общей вершиной* этих деревьев. Поддерево, корнем которого является общая вершина, будем называть *общим поддеревом* нескольких взаимно-пересекающихся деревьев.

Средства ИК позволяют порождать структуры, которые могут содержать пересекающиеся и взаимно-пересекающиеся деревья (рис. 1). При воплощении БД в памяти компьютера в процессе работы приложения создается специальный объект — *технологический корень БД*, дочерними объектами которого становятся объекты, соответствующие корневым вершинам графа БД. Добавление технологического корня (ТК) превращает граф БД в дерево, которое будем называть *полным* деревом БД (рис. 2).

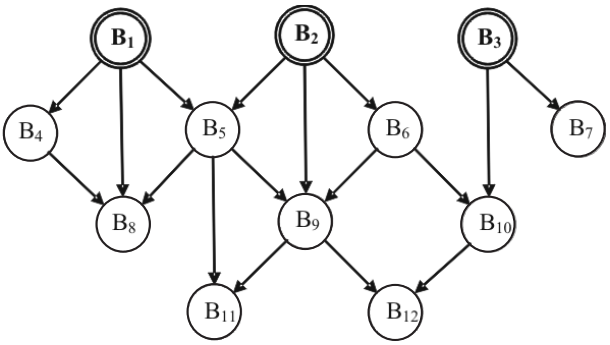

Рис. 1. Граф БД

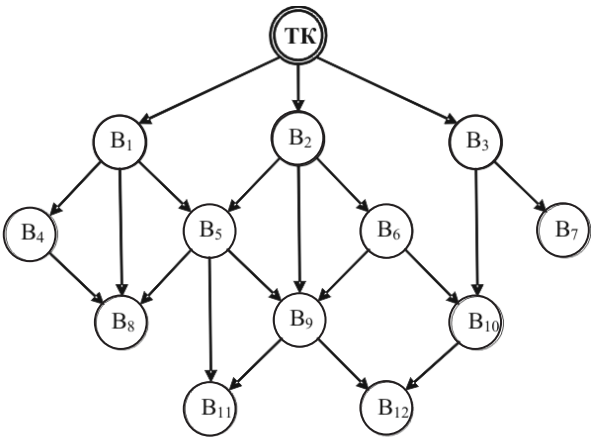

Рис. 2. Полное дерево БД

Полное дерево БД является, как правило, пересекающимся. Однако средства организации БД обеспечивают выделение в полном дереве БД непересекающегося поддерева, включающего все вершины. Будем называть такое поддерево *скелетным деревом* БД (рис. 3). Дуги, соединяющие вершины скелетного дерева, будем называть *скелетными дугами* (скелетные дуги изображены на рисунке утолщенными стрелками). Скелетное дерево БД может использоваться для гарантированного однократного обхода всех объектов БД.

Простой ориентированный путь от объекта, не являющегося технологическим корнем, до любого другого объекта БД будем называть *относительным* путем БД. Для любой пары объектов БД могут существовать несколько относительных путей или путь может отсутствовать. Например, между объектами B1 и B8 на рис. 3 существует три пути: B1>B4>B8, B1>B8 и B1>B5>B8, а между объектами B1 и B6 пути нет. Путь, принадлежащий скелетному дереву БД, будем называть *скелетным* путем. Между двумя объектами может существовать только один скелетный путь. Путь от технологического корня к любому объекту БД будем называть *абсолютным* путем. Для любого объекта БД всегда существует хотя бы один абсолютный путь и единственный абсолютный скелетный путь. Например, от технологического корня к объекту B8 существует четыре абсолютных пути: ТК>B1>B4>B8, ТК>B1>B8, ТК>B1>B5>B8,

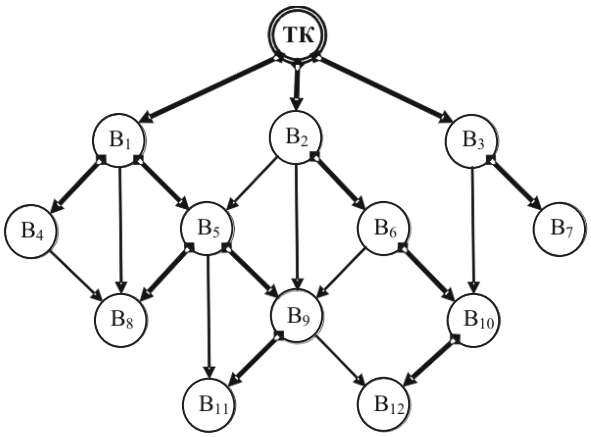

Рис. 3. Скелетное дерево БД

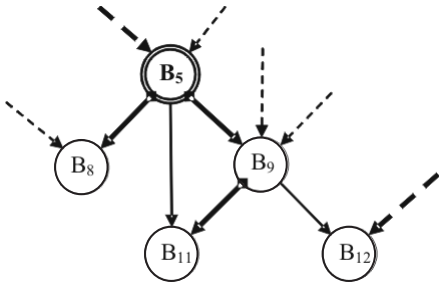

Рис. 4. Поддерево объекта B5

ТК>B2>B5>B8, и один абсолютный скелетный путь ТК>B1>B5>B8.

Поддерево полного дерева БД, корнем которого является некоторый объект, будем называть *поддеревом* этого *объекта* (рис. 4). Поддерево любого объекта БД содержит фрагмент скелетного дерева БД — *скелетное поддерево*. Таким образом, каждому объекту БД соответствует единственное скелетное поддерево. Полное дерево БД является поддеревом ее технологического корня.

2.2. Структура объектов БД. Информация, содержащаяся в каждом объекте, состоит из двух частей: общей (структурной) и индивидуальной. Набор полей общей части одинаков для всех объектов и содержит сведения о самом объекте, как части структуры БД, и о его структурных связях с другими объектами. Состав структурных связей объекта может динамически изменяться при редактировании БД. С помощью структурных связей задается информация, в том числе и о скелетном дереве БД.

Сведения о самом объекте как части структуры содержат, прежде всего:

— идентификатор объекта;

— параметры визуализации объекта и его дочерних объектов при отображении структуры БД;

— данные о типе прикладной информации, хранящейся в объекте.

Структурные связи объекта с другими объектами БД устроены следующим образом. Каждый объект может быть родительским для нескольких дочерних объектов. Информация о составе дочерних объектов хранится в родительском объекте в виде динамически изменяемого списка ссылок на дочерние объекты. Любой объект может иметь несколько родительских объектов. Однако не все родительские объекты являются равноправными. Один из родительских объектов является главным и называется *владельцем* дочернего объекта. Каждый объект БД имеет поле, в котором хранится ссылка на родительский объект, являющийся владельцем данного объекта. Если объект не имеет владельца (ссылка на владельца не определена), то он является корнем БД. Структурная связь между владельцем и его дочерним объектом является скелетной.

Набор полей индивидуальной части объекта определяется его типом. Поля индивидуальной части объекта также могут содержать ссылки на другие объекты БД (неструктурные связи объекта). Состав возможных индивидуальных связей определяется типом объекта. Индивидуальные связи объекта не могут быть скелетными.

2.3. Сериализация БД. Сериализация БД используется при сохранении состояния БД и ее фрагментов во внешнем файле, а также при копировании фрагментов БД в процессе редактирования. Базовая процедура сериализации обеспечивает превращение поддерева любого объекта БД (в том числе и полного дерева БД) в последовательность байтов. При этом в поток сериализации попадает содержимое не всех объектов выгружаемого поддерева, а только тех, которые входят в состав скелетной части выгружаемого поддерева. Значения полей, содержащих ссылки на объекты БД, сериализуются в виде скелетных путей к этим объектам. Информация о скелетном пути позволяет однозначно идентифицировать эти объекты при восстановлении связей в процессе десериализации (превращения последовательности байтов во фрагмент БД).

Результат сериализации может быть представлен как в двоичном, так и в текстовом виде. Синтаксис текстового сериализованного представления БД соответствует стандарту XML [6].

2.4. Базовые классы объектов БД ИК. При проектировании объектной модели БД приложения необходимо за основу выбирать базовые классы объектов БД ИК, которым придается необходимая прикладная функциональность и требуемые для параметризации этой функциональности прикладные структуры данных. Существует два базовых класса объектов БД ИК, которые предназначены для создания прикладных объектов приложения:

— главный базовый класс TN_UDBase,

— базовый класс контейнеров прикладных данных TK_UDRArray.

Базовый класс контейнеров прикладных данных TK_UDRArray является наследником главного базового класса TN_UDBase. Объект этого класса может содержать некоторую прикладную структуру данных, тип которой задается независимо от типа объекта-контейнера на этапе выполнения программы. В базовых классах объектов БД ИК воплощена вся необходимая функциональность по организации структуры БД, поддержанию ее целостности при редактировании, сериализации и десериализации.

Сериализация БД обеспечивается за счет наличия у каждого объекта методов, которые осуществляют сериализацию самого объекта и его дочерних объектов. Аналогично, десериализация БД обеспечивается за счет наличия у каждого объекта методов, которые осуществляют десериализацию самого объекта и его дочерних объектов.

Добавление новых полей в производные классы объектов БД приложения может идти двумя путями. Первый путь — это *статическое* включение дополнительных полей в базовый класс за счет наследования классов базового языка программирования. Такое наследование возможно и от TN_UDBase, и от TK_UDRArray. Второй путь — *динамическое* включение дополнительных прикладных полей в объект, являющийся экземпляром класса TK_UDRArray или его наследником, на этапе выполнения программы за счет определения структуры дополнительных полей средствами описания данных языка SPL (см. ниже).

Если осуществляется статическое дополнение класса прикладными структурами данных, то необходимо перегружать методы сериализации/десериализации для обработки дополнительных полей. Методы базового класса TK_UDRArray автоматически обеспечивают сериализацию/десериализацию дополнительных полей любого типа. Кроме того, в ИК реализована поддержка визуализации и редактирования прикладных структур данных, встраиваемых в объекты-контейнеры. Эта поддержка состоит в автоматической генерации экранных форм, которые обеспечивают наглядное представление и редактирование содержимого этих структур данных.

Таким образом, выбор базового класса в процессе разработки объектной модели БД приложения целесообразно выполнять по следующим правилам. Если объект БД приложения не должен содержать прикладные структуры данных, требующие сериализации и интерактивного редактирования, то в качестве базового класса для наследования нужно выбирать главный базовый класс TN_UDBase.

Если же объект БД приложения должен содержать прикладные структуры данных, которые необходимо просматривать и редактировать в интерактивном режиме, а также сохранять их состояние от сеанса к сеансу, то наиболее подходящим для наследования базовым классом является класс TK_UDRArray.

2.5. Организация объединения баз данных. Реализованные в ИК средства организации БД приложения обеспечивают работу с данными в однопользовательском режиме. В этой ситуации особенно актуальна задача объединения независимо редактируемых баз данных. В состав ИК входят средства поддержки такого объединения. В основе процедуры объединения лежит понятие *системного каталога*. Системным каталогом считается объект БД, для которого определены правила объединения поддерева этого объекта, заданного структурными связями, с поддеревом некоторого соответствующего ему системного каталога другой БД. Системным каталогом может быть объявлен любой объект БД независимо от его типа. Объекты, являющиеся системными каталогами БД, образуют *дерево объединения*, корнем которого является технологический корень БД. Дерево объединения является поддеревом скелетного дерева БД.

В процедуре объединения участвуют две БД. Базу данных, в которую осуществляется добавление объектов, будем называть целевой или результирующей. Базу данных, объекты которой добавляются к результирующей БД, будем называть добавляемой или источником данных.

При объединении баз данных осуществляется добавление к объектам дерева объединения результирующей БД соответствующих объектов дерева объединения БД-источника. Фактически объединение происходит в два этапа. Вначале выполняется анализ структур деревьев объединения обеих баз данных, в результате которого составляется список пар соответствующих друг другу системных каталогов двух баз. Затем, уже независимо от структуры объединяемых баз данных, осуществляется объединение отобранных на первом этапе пар системных каталогов.

Для каждого системного каталога можно задать свои индивидуальные правила объединения. В основе управления объединением системных каталогов лежит разделение всех дочерних объектов каталога-источника на две группы:

— дочерние объекты каталога-источника, для которых найдены соответствующие объекты результирующего каталога (первая группа);

— дочерние объекты каталога-источника, для которых не найдены соответствующие объекты результирующего каталога (вторая группа).

Для каждой группы управление объединением осуществляется независимо. Добавление дочерних объектов первой группы происходит по одному из следующих правил:

— не добавлять в результирующий каталог;

— добавлять в результирующий каталог;

— заменять соответствующие объекты результирующего каталога;

— заменять только устаревшие соответствующие объекты результирующего каталога.

Дочерние объекты второй группы могут либо добавляться, либо не добавляться в результирующий каталог.

Поиск соответствующих объектов в системных каталогах может происходить с учетом нескольких признаков:

— совпадение идентификаторов,

— совпадение типов объектов,

— совпадение типов дополнительных прикладных полей объектов-контейнеров.

Реализованная в ИК поддержка объединения баз данных эффективно работает при объединении баз данных, имеющих одинаковую или "похожую" структуру деревьев объединения, что характерно для объединения баз данных, созданных в рамках одного приложения. Кроме того, средства ИК позволяют в случае наличия в БД приложения дерева объединения автоматизировать процедуру создания копии БД, содержащей только определенные, выбранные пользователем объекты с сохранением структуры дерева объединения копируемой БД.

2.6. Работа с векторами данных. В средствах управления БД ИК реализована поддержка хранения и манипулирования данными, организованными в вектора с кодовой привязкой элементов. Каждый элемент такого вектора имеет идентифицирующий код, уникальный в рамках некоторой группы кодов, называемой *кодовым пространством*. Одна БД может содержать несколько кодовых пространств. Коды элементов одного вектора могут принадлежать только одному кодовому пространству.

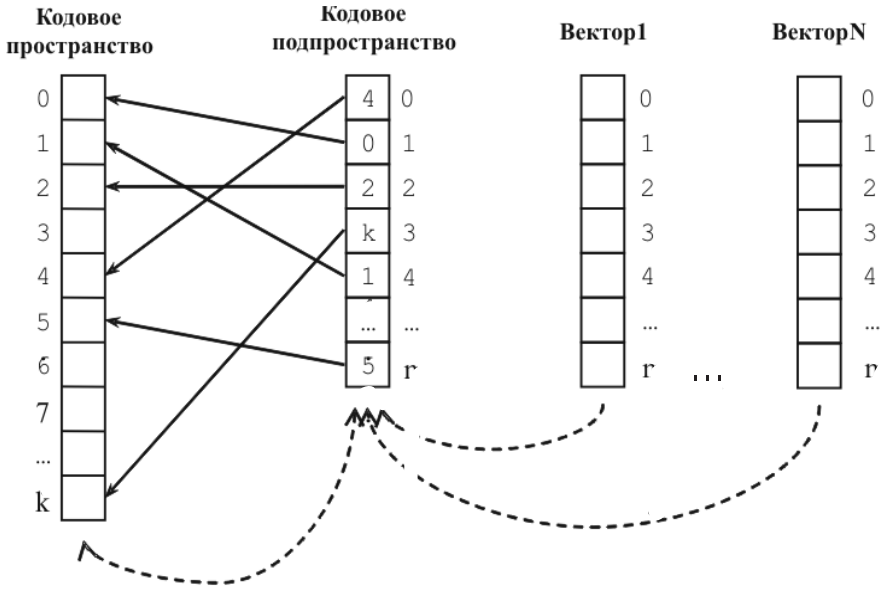

Рис. 5. Вектора данных с кодовой привязкой

Значением идентифицирующего кода может быть произвольная строка. Установление соответствия между идентифицирующими кодами и элементами векторов данных с кодовой привязкой организовано следующим образом. Объект "кодовое пространство БД" содержит вектор значений идентифицирующих кодов, соответствующий элементам кодового пространства. Для установления связи между элементами

вектора данных и элементами кодового пространства используется специальный вид объектов БД — *кодовые подпространства*. Кодовое подпространство задает связь между элементами кодового пространства и множеством векторов, имеющих одинаковый состав и порядок элементов. Оно представляет собой синхронный с вектором данных вектор индексов по вектору кодов кодового пространства (рис. 5). Таким образом, для вектора данных V с кодами элементов из пространства CS, связь с которым осуществляется с помощью некоторого кодового подпространства CSS, справедливо следующее — код элемента вектора $V[i]$ можно определить как $CS\big[CSS[i]\big]$. Каждый вектор данных с кодовой привязкой элементов хранит ссылку на соответствующее ему кодовое подпространство, а объект "кодовое подпространство", в свою очередь, содержит информацию об объекте "кодовое пространство", к которому он относится.

Наличие идентифицирующих кодов позволяет автоматически устанавливать соответствие между элементами различных векторов, относящихся к разным подпространствам одного кодового пространства.

Существует еще один вид объектов БД, позволяющий устанавливать соответствие между элементами одного или различных кодовых пространств — *кодовая проекция*. Кодовую проекцию можно представить как однозначную функцию, областью определения которой являются множество всех элементов одного кодового пространства, а областью значений — подмножество элементов другого кодового пространства. Кодовая проекция реализована как вектор с кодовой привязкой элементов, который синхронен элементам кодового пространства, являющегося ее областью определения (рис. 6). Значениями элементов этого вектора служат индексы элементов кодового пространства, которое задает область значений кодовой проекции. Таким образом, кодовая проекция является одновременно не только вектором с кодовой привязкой элементов кодового пространства своей области определения, но и кодовым подпространством кодового пространства своей области значений.

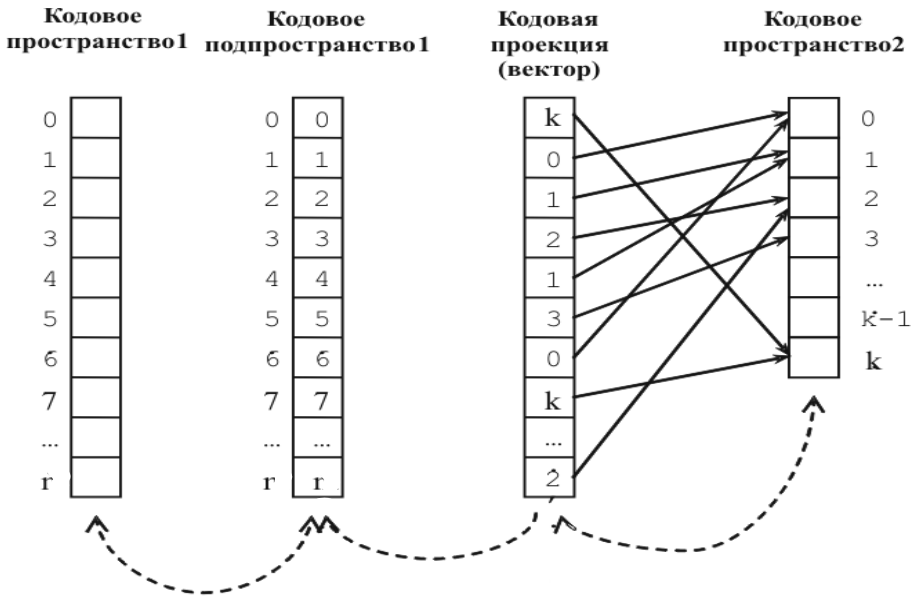

Рис. 6. Соответствие между кодовыми пространствами

Наличие кодовых проекций расширяет возможности установления соответствия между элементами векторов данных с кодовой привязкой. Они позволяют устанавливать отношение "один к одному" и "один ко многим" как между элементами векторов данных различных кодовых пространств, так и между элементами векторов данных, относящихся к одному кодовому пространству, в том числе и между элементами одного вектора данных.

Для установления соответствия между элементами двух векторов данных средства ИК позволяют вычислять *проекцию данных*. Проекцию данных, как и кодовую проекцию, можно представить как однозначную функцию, областью определения которой является множество всех элементов подпространства одного вектора данных, а областью значений — подмножество элементов подпространства другого вектора данных. Проекция данных представляет собой массив индексов, синхронный с одним из векторов данных, который задает для каждого элемента этого вектора данных индекс соответствующего элемента другого вектора данных (рис. 7).

Вектора данных с кодовой привязкой элементов, кодовые подпространства и кодовые проекции являются наследниками объекта TK_UDRArray.

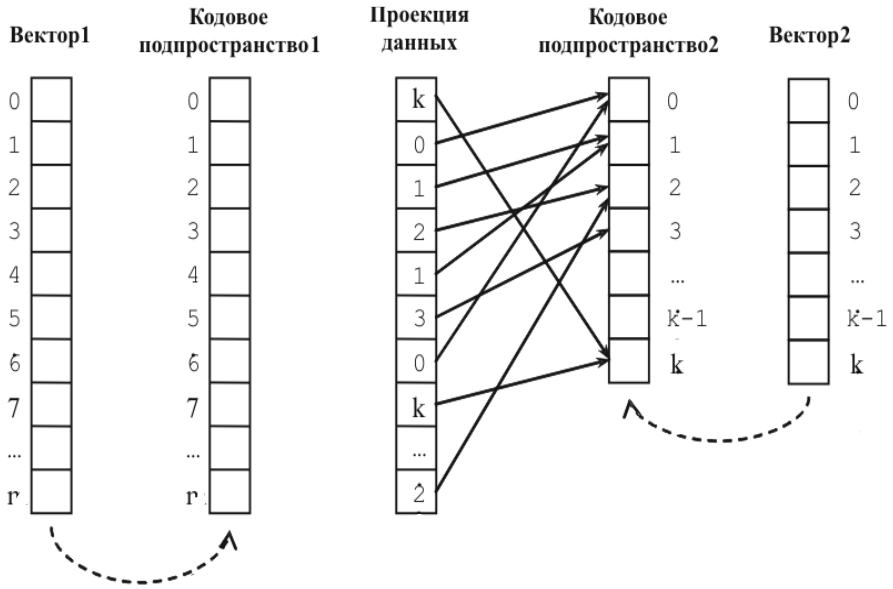

Рис. 7. Соответствие между элементами векторов данных

3. Язык описания данных и сценариев их обработки. В состав ИК входит встроенный язык описания данных и сценариев их обработки, названный SPL. Язык SPL является универсальным процедурным объектно-ориентированным языком программирования, синтаксис которого основан на синтаксисе языка Object Pascal. Он обладает характерными для современных языков программирования средствами описания типов данных и алгоритмов, а также средствами оформления программных модулей (классы, процедуры, функции).

Язык SPL выполняет в инструментальном комплексе две основные функции:

— описание типов прикладных данных для объектов-контейнеров данных БД приложения (TK_UDRArray и его наследники),

— описание сценариев обработки данных.

Средства описания типов данных в SPL позволяют определять структуры данных, являющихся типичными для большинства современных языков программирования (C, Pascal). Типы данных в SPL можно разделить на две категории:

— предопределенные (встроенные);

— производные (определяемые пользователем).

В предопределенные типы SPL входят различные виды целых (1, 2, 4, 8 байт) и действительных чисел (4, 8 байт), строка, ссылка на объект БД и др.

Средства описания производных типов данных позволяют определять следующие производные типы:

— перечисления,

— множества,

— записи,

— динамические массивы.

Результатом работы SPL-компилятора являются структура данных, содержащая описания заданных в SPL-программе производных типов, и байт-код SPL-процедур. Выполнение откомпилированных SPL-программ осуществляется SPL-интерпретатором байт-кода.

Программы, написанные на языке SPL, могут эффективно взаимодействовать с программами, реализованными на языке Object Pascal. Это достигается за счет того, что результат компиляции встроенных и производных типов данных языка SPL в точности соответствует результату компиляции аналогичных конструкций базового языка программирования. Такое соответствие позволяет без каких-либо накладных расходов обрабатывать структуры данных SPL-программ из программ на Object Pascal и, наоборот, обращаться к полям структур данных Object Pascal из SPL-программ. Кроме того, экземпляр любого SPL-класса, создаваемый в процессе работы SPL-программы, является объектом базового класса контейнеров прикладных данных TK_UDRArray.

Интеграция программ, реализованных на SPL и Object Pascal, осуществляется не только на уровне доступа к данным, но и на уровне программного кода. Так, имеется возможность непосредственного

вызова из SPL-программ процедур и функций, реализованных на языке Object Pascal. Эти процедуры должны быть написаны по определенным правилам с учетом того, что их вызов осуществляет из SPL-интерпретатора. Синтаксис вызова таких процедур в SPL совпадает с синтаксисом вызова процедур и функций самого языка SPL. Встраивание в SPL подобных процедур, реализованных на языке Object Pascal, позволяет расширять возможности базового SPL с учетом потребностей конкретного приложения.

В ИК также имеются средства вызова SPL-процедур и SPL-функций из программ на Object Pascal. При этом вызываемым процедурам и функциям можно передать необходимые параметры.

4. Базовые компоненты пользовательского интерфейса. Средства инструментального комплекса обеспечивают автоматизацию построения пользовательского интерфейса приложения за счет наличия программных модулей визуализации пользовательских объектов и модулей поддержки редактирования объектов БД приложения.

Программная компонента, обеспечивающая визуализацию фрагментов БД, позволяет отобразить структурные связи выбранного фрагмента БД в виде дерева с возможностью управления глубиной раскрытия. Каждый объект БД отображается в виде вершины дерева, которому соответствует определенная иконка и текст. Визуальное представление структуры БД позволяет при отображении структурных связей объектов БД специальным образом выделить скелетные связи.

Построение визуального представления фрагмента структуры БД осуществляется следующим образом. Для выбранного объекта БД строится вспомогательная структура данных — *дерево визуализации*. Дерево визуализации представляет собой дерево специальных объектов — визуальных вершин. Каждой визуальной вершине соответствует один объект БД. Однако одному объекту БД может соответствовать несколько визуальных вершин. Это происходит в том случае, когда объект БД имеет несколько родительских узлов, поскольку каждому родительскому объекту БД соответствует в дереве визуализации своя родительская визуальная вершина, а каждая родительская визуальная вершина имеет свою дочернюю визуальную вершину, соответствующую тому же дочернему объекту БД. Дерево визуализации используется для генерации реального визуального представления структуры БД. Средства генерации дерева визуализации позволяют с помощью функции фильтрации управлять составом отображаемых объектов БД.

Средства инструментального комплекса позволяют одновременно отображать несколько визуальных представлений структуры БД, для каждого из которых строится свое дерево визуализации. Административная система инструментального комплекса обеспечивает автоматическое изменение всех отображаемых деревьев визуализации и соответствующих визуальных представлений при изменении структурных связей БД.

Каждое дерево визуализации поддерживает возможность визуального выделения одной текущей и нескольких выбранных вершин. Программный интерфейс позволяет приложению с помощью дерева визуализации получить доступ к объектам БД, соответствующим как текущей визуальной вершине, так и списку выбранных визуальных вершин.

Программный интерфейс взаимодействия с компонентой, обеспечивающей визуализацию структуры БД, позволяет приложению определить собственные реакции на следующие основные программные события:

— выбор текущего объекта;

— активизация текущего объекта (двойной щелчок мыши на объекте или нажатие клавиши Enter на уже выбранном объекте);

— нажатие кнопки мыши на объекте;

— начало переименования объекта;

— завершение переименования объекта.

В ИК реализована поддержка визуализации и редактирования в табличной форме массивов записей, структура которых может быть описана средствами языка SPL. Программная компонента, обеспечивающая наглядное представление и редактирование содержимого этих структур данных, функционирует на основе управляющей информации, которая состоит из матрицы указателей на отдельные поля редактируемых записей и массива описателей полей, задающих способы их визуального представлении и редактирования. Описатель поля, в частности, содержит:

— тип данных;

— название;

— формат вывода;

— параметры цветового оформления;

— объект, отвечающий за визуализацию данных;

— объект, отвечающий за редактирование данных.

Эта управляющая информация создается на основе сведений о составе и типе полей редактируемой структуры данных, которые создаются в результате работы SPL-компилятора. Кроме того, при генерации данной управляющей информации можно использовать специальные объекты "*описатели визуального представления*". Эти объекты являются экземплярами SPL-класса TK_FrameDescription. Они позволяют задать состав и порядок отображения отдельных полей, а также изменять типовые способы их визуального представления и редактирования.

Программный интерфейс взаимодействия приложения с компонентой, обеспечивающей визуализацию и редактирование массивов записей, позволяет приложению определить собственные реакции на следующие основные программные события:

— выбор текущего поля записи,
— выбор текущей записи массива,
— завершение редактирования поля,
— добавление записи,
— удаление записи,
— перемещение записи.

В административной системе инструментального комплекса реализованы механизмы установления соответствия между типами данных, формами редактирования и описателями визуального представления. Эти механизмы позволяют автоматизировать написание кода приложения, обеспечивающего вызов необходимых редакторов для объектов БД, выбранных пользователем в интерактивном режиме.

Реализация модулей, обеспечивающих автоматизацию построения пользовательского интерфейса, в большой мере ориентирована на средства создания визуальных компонент среды быстрой разработки приложений Delphi.

5. Заключение. Использование Object Pascal в качестве базового языка программирования не оказало существенного влияния на функциональность описанных выше компонент ИК. Она складывалась, прежде всего, на основе опыта реализации ряда независимых приложений, ориентированных на работу со сложными структурами данных. Подобная функциональность вполне могла бы быть реализована и на каком-нибудь другом объектно-ориентированном языке программирования — C++, Java и др.

Инструментальный комплекс был успешно использован при создании ряда приложений, ориентированных на подготовку публикаций в Интернет разнородной территориально привязанной информации, включая представления числовых данных на картах и схемах. Примеры сайтов, подготовленных с помощью этих приложений, можно найти в Интернет по адресу http://www.srcc.msu.su/ivis/.

СПИСОК ЛИТЕРАТУРЫ

1. *Архангельский А.Я.* Object Pascal в Delphi. М.: БИНОМ, 2002.
2. *Кэнту М.* Delphi 7: Для профессионалов. СПб.: Питер, 2004.
3. The Object Data Base Standard: ODMG-99. Ed. by Cattell R.G.G. San Francisco: Morgan–Kaufmann, 1999.
4. *Гарсиа-Моллина Г., Ульман Дж., Уидом Дж.* Системы баз данных. Полный курс. М.: Издательский дом "Вильямс", 2004.
5. *Abiteboul S., Suciu D., Buneman P.* Data on the Web: from relations to semistructure data and XML. San Francisco: Morgan–Kaufmann, 1999.
6. Extensible Markup Language (XML) 1.0 (Third Edition).W3C Recommendation. 04 February 2004. Ed. by T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau. Available at http://www.w3.org/TR/REC-xml/

Поступила в редакцию
03.01.2005