Известия Гомельского государственного университета
имени Ф. Скорины, №5(50), Ч1, 2008

УДК 681.3.07

Об одном подходе к реализации баз данных ERP-систем

В. А. Короткевич, Л. И. Короткевич, А. В. Белевич

Введение. В настоящее время важной тенденцией в автоматизации управления предприятиями является переход от так называемой «лоскутной» автоматизации, предполагающей создание разрозненных подсистем для отдельных подразделений и конкретных рабочих мест сотрудников предприятия, к внедрению комплексных решений по автоматизации всех видов управленческой деятельности. Реализация таких проектов ведется на основе программных систем, относящихся к классу так называемых ERP-систем (от Enterprise Resource Planning – Планирование ресурсов предприятия). К наиболее известным системам этого класса относятся SAP R/3 [1], Oracle E-Buisness Suite, BAAN и др. Реализация проектов на основе таких систем представляет собой сложный, длительный и дорогостоящий процесс, но окупается в дальнейшем за счет повышения оперативности и точности принимаемых управленческих решений, оптимизации материальных и финансовых потоков на предприятии.

Концепция разработки и внедрения ERP-систем предполагает создание единого комплексного хранилища данных, содержащего всю корпоративную бизнес-информацию и обеспечивающего одновременный доступ к ней любого необходимого количества сотрудников предприятия, наделенных соответствующими полномочиями. В данной статье рассматривается возможный подход к реализации такого хранилища на основе реляционной системы управления базой данных (СУБД).

Конструирование бизнес-объектов. Единое хранилище данных ERP-системы должно содержать в себе сведения по самым различным бизнес-объектам: подразделениям и сотрудникам предприятия, контрагентам, материально-техническим ресурсам и продукции предприятия, документам различных видов и др. В такой ситуации стандартный подход к проектированию реляционной базы данных, связанный с сопоставлением каждому бизнес-объекту отдельной таблицы БД, приводит к очень сложной структуре базы данных, состоящей из большого количества взаимосвязанных таблиц. При этом многие из этих таблиц, например, соответствующие документам различных видов, будут иметь близкую структуру, отличаясь только несколькими атрибутами. В противоположность такому подходу, авторами предлагается конструировать бизнес-объекты на основе небольшого количества базовых таблиц (см. рис. 1).

Каждый экземпляр любого бизнес-объекта представляется записью в таблице *ITEMS*, которая одновременно связывает все экземпляры объектов в древовидную структуру. Поле *ID* является уникальным идентификатором экземпляра объекта, *PARENTID* – указывает на владельца (предок в дереве), ссылаясь на запись в той таблице *ITEMS*. Любой экземпляр бизнес-объекта может быть представлен в дереве произвольное количество раз: вновь созданный экземпляр считается оригиналом объекта, для него в поле *BASEID* устанавливается значение, равное *ID*, а в последующем может быть создано множество записей – ярлыков объекта, с тем же значением поля *BASEID*, что и у оригинала. Таким образом, например, для сотрудника предприятия запись–оригинал может являться потомком узла дерева «Личные карточки сотрудников», а ярлыки являться потомками других узлов: узла для подразделения предприятия, где работает сотрудник, узла «Получатели пособий на детей», узла «Сотрудники, находящиеся в командировках» и т.п.

Все экземпляры бизнес-объектов через поле *BASEID* ссылаются на таблицу *NODES*, указывающую к какому типу относится тот или иной элемент дерева. Поле *TYPEID* этой таблицы содержит числовое значение – тип бизнес-объекта, которому соответствует запись таблицы *ITEMS*.

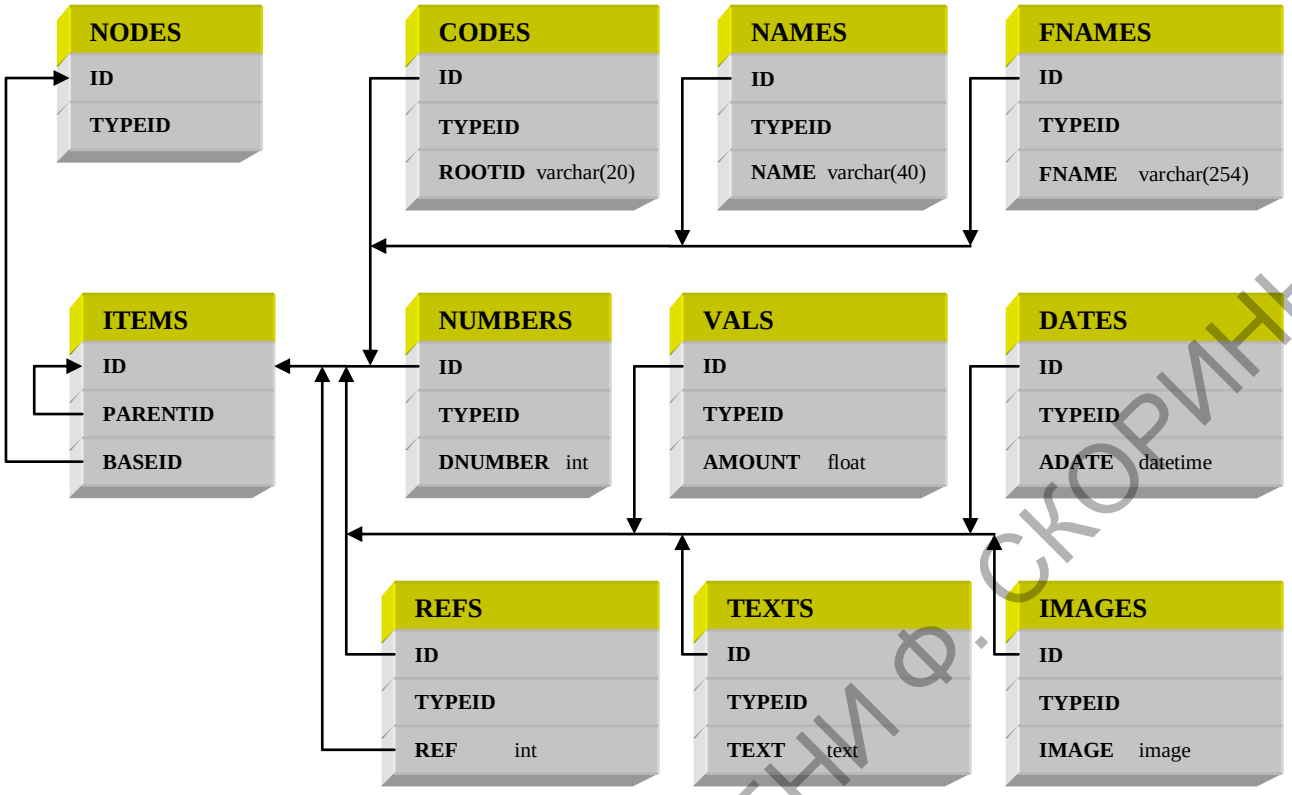

Рисунок 1 – Базовые таблицы для конструирования бизнес-объектов

Для хранения всех остальных атрибутов бизнес-объектов служат следующие базовые таблицы, соответствующие атрибутам различных типов:
CODES, *NAMES*, *FNAMES* – строковые атрибуты, длиной соответственно до 20, 40, 254 символов:
NUMBERS – целочисленные атрибуты;
VALS – числовые атрибуты со значениями в плавающем формате;
DATES – дата / время;
REFS – ссылки на родительские объекты, атрибуты соответствуют внешним ключам таблиц при традиционной структуре реляционной базы данных;
TEXTS, *IMAGES* – длинные текстовые и двоичные данные.

Все эти таблицы имеют аналогичную структуру, различаясь только форматом поля со значением атрибута. Поле *ID* каждой таблицы является идентификатором экземпляра объекта, которому принадлежит атрибут, поле *TYPEID* определяет тип атрибута. Значение *TYPEID* может быть произвольным, служит для того, чтобы различать атрибуты совпадающих типов (хранящиеся в одной и той же таблице), входящие в структуру одного и того же бизнес-объекта (хотя естественно использовать одинаковые значения типов для аналогичных атрибутов различных бизнес-объектов).

Значение атрибута для оригинала объекта принимается по умолчанию в качестве значения для всех его ярлыков и не дублируется в базе данных. При необходимости для ярлыка может быть назначено собственное перекрывающее значение атрибута. В базе данных не хранятся NULL-значения атрибутов: если ранее занесенное значение атрибута меняется на NULL, то из соответствующей таблицы просто удаляется запись со старым значением.

В качестве примера рассмотрим реализацию структуры объекта «Личная карточка сотрудника» с использованием указанных базовых таблиц (см. табл. 1).

Доступ к бизнес-объектам. Предлагаемая структура базы данных абсолютно не зависит от состава и структуры хранящихся в базе бизнес-объектов: сама структура объектов ни-

Таблица 1 – Личная карточка сотрудника

Поле	Содержимое	Базовая таблица
ID	Идентификатор	ITEMS
PARENTID	Предок элемента дерева	ITEMS
BASEID	Идентификатор оригинала	ITEMS
TYPEID	Тип объекта	NODES
TABNUM	Табельный номер	CODES (TypeID = 100)
FAM	Фамилия	NAMES (TypeID = 200)
IM	Имя	CODES (TypeID = 101)
OTCH	Отчество	CODES (TypeID = 102)
DATER	Дата рождения	DATES (TypeID = 300)
ADDR	Домашний адрес	FNAMES (TypeID = 400)
OBRAZOV	Образование (ссылка на экземпляр объекта «Уровни образования»)	REFS (TypeID = 500)
и т.д.	…	…

как не отражена в базе данных. В то же время каким-то образом должны быть реализован доступ к экземплярам объектов как к реально существующим единицам, в том числе: получение перечней экземпляров объекта, создание и уничтожение объектов, изменение значений атрибутов объектов.

Проблема решается путем создания для доступа к объектам соответствующих хранимых функций и процедур – программных модулей, хранящихся в базе данных. Создание таких модулей в той либо иной форме поддерживается всеми современными СУБД [2], [3].

Для каждого бизнес-объекта создается:

1) функция, выполняющая «сборку» экземпляров объекта из содержимого базовых таблиц и возвращающая перечень экземпляров в виде набора данных;

2) процедура, получающая в качестве параметров значения атрибутов и выполняющая создание или обновление экземпляра объекта;

3) процедура уничтожения экземпляра объекта по идентификатору.

В процессе разработки указанных процедур и функций могут быть использованы уже разработанные стандартные средства:

1) скалярные функции для получения значения атрибута объекта (для каждого типа атрибута);

2) процедуры обновления значения атрибута объекта (для каждого типа атрибута);

3) процедура уничтожения экземпляра объекта, удаляющая сведения о нем из таблиц *ITEMS*, *NODES* и всех таблиц атрибутов (универсальная процедура, используемая при уничтожении экземпляра любого бизнес-объекта).

Наследование бизнес-объектов. Назовем бизнес-объект *B* наследником бизнес-объекта *A*, если множество атрибутов *B* полностью включает в себя множество атрибутов *A*. Существование отношения наследования позволяет при разработке функций и процедур доступа для объекта *B* использовать функции и процедуры доступа к объекту *A*.

Пусть, например, бизнес-объекта «Документ» содержит атрибуты: «дата», «номер», «поставщик», «получатель» и др., и для получения перечня документов разработана соответствующая функция *GetDocs*. Тогда при разработке функции доступа к бизнес-объекту «Платежный документ», содержащему дополнительные атрибуты «сумма» и «назначение платежа», может быть использован SQL-запрос следующего вида (синтаксис MS SQL Server):

```
SELECT DOCS.*
      ,dbo.SF_VAL(DOCS.ID,700) SUMMA -- сумма
      ,dbo.SF_FNAME(DOCS.ID,800) NAZNACH -- назначение платежа
  FROM dbo.GetDocs() DOCS
```

где SF_VAL, SF_FNAME – скалярные функции для получения значений атрибутов объектов, 700, 800 – типы атрибутов.

Аналогично, процедура обновления объекта «Платежный документ» может вызывать процедуру объекта «Документ», после чего выполнять сохранение только дополнительных атрибутов. Как следствие, представление всей совокупности бизнес-объектов в виде иерархического дерева наследования, определяющего отношения «предок» – «потомок», позволяет существенно упростить и автоматизировать создание SQL-кода для доступа к объектам. При этом совокупность базовых объектов, созданных в ходе реализации одного проекта, может использоваться как основа для создания бизнес-объектов в других проектах.

Заключение. Предлагаемый в работе подход к построению базы данных ERP-системы обеспечивает максимальную гибкость в определении состава и структуры бизнес-объектов системы при наличии простой, фиксированной структуры базы данных на SQL-сервере. Этот подход апробирован в программном комплексе SBC (ООО «Системные бизнес-компоненты», г. Гомель) в ходе реализации ряда крупных проектов на территории Республики Беларусь и Российской Федерации с использованием таких серверов баз данных как MS SQL Server и ORACLE.

Abstract. The method of realization of the ERP-system database is considered in the paper. The construction of business objects using base tables containing object attributes is also suggested.

Литература

1. Хагеман, С. SAR R/3. Системное администрирование / С. Хагеман, Л. Вилл. – М.: Лори, 2007.
2. Хендерсон, К. Профессиональное руководство по SQL Server: хранимые процедуры, XML, HTML / К. Хендерсон. – СПб.: Питер, 2005.
3. Урманн, С. Oracle 9i. Программирование на языке PL/SQL / С. Урманн. – М.: Лори, 2006.

Гомельский государственный
университет имени Ф.Скорины

Поступило 10.05.08