Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра информатики

И. И. Пилецкий

ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА И СОПРОВОЖДЕНИЕ БАЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ CASE-СРЕДСТВ

Пособие по курсу
«Методы и технологии программирования»
для студентов специальности 1-31 03 04 «Информатика»
всех форм обучения

Минск БГУИР 2009

УДК 004.65(076)
ББК 32.973.26-018.2 я73
П32

Р е ц е н з е н т :
зав. кафедрой математического обеспечения ЭВМ
Белорусского государственного университета
канд. техн. наук, доцент Л. Ф. Зимянин

Пилецкий, И. И.

П32 Проектирование, разработка и сопровождение баз данных с использованием CASE-средств : пособие по курсу «Методы и технологии программирования» для студ. спец. 1-31 03 04 «Информатика» / И. И. Пилецкий. – Минск : БГУИР, 2009. – 116 с.: ил.

ISBN 978-985-488-324-3

Книга представляет собой пособие по проектированию, разработке и сопровождению баз данных с помощью CASE-средства фирмы Computer Associates AllFusion ERwin Data Modeler, мирового лидера CASE-средств по проектированию баз данных. Данное CASE-средство опирается на стандарт языка графического моделирования и разработки баз данных и хранилищ IDEF1x.

Разработано на реальном материале, полученном при проектировании и сопровождении больших баз данных, и проиллюстрировано большим количеством рисунков с примерами конкретных разработок.

Пособие предназначено для студентов, магистрантов, аспирантов, изучающих дисциплины проектирования информационных систем, а также для специалистов в области информационных систем, аналитиков и проектировщиков баз данных.

УДК 004.65(076)
ББК 32.973.26-018.2 я73

ISBN 978-985-488-324-3

СОДЕРЖАНИЕ

Предисловие 4
1. Общие сведения о среде проектирования AllFusion Erwin Data Modeler....6
2. Основные элементы языка моделирования баз данных IDEF1x 10
2.1. Сущности 10
2.2. Связи, отношения 13
2.3. Мощность связей 13
2.4. Ключи....14
2.5. Внутренние и внешние ключи....16
2.6. Ссылочная целостность 17
2.7. Правила построения диаграмм, виды связей, категории, рекурсии 20
2.8. Домены....33
2.9. Представления 36
3. Проектирование баз данных, логический уровень 41
3.1. Информационные системы, базы данных и модели....41
3.2. Построение логической модели 45
3.2.1. Диаграмма «сущность – связь» 45
3.2.2. Модель данных на основе ключа 47
3.2.3. Полная атрибутивная модель 51
4. Создание физического уровня базы данных на основе логического....57
4.1. Создание новой модели 58
4.2. Редактирование таблиц 58
4.3. Редактирование столбцов таблицы 61
4.4. Редактирование ключей и индексов таблицы....63
4.5. Редактирование связей таблиц 65
4.6. Сохранение модели базы данных 66
4.7. Генерация операторов для создания базы данных 67
4.8. Подготовка исходных данных для разработки новой версии БД....77
4.9. Проектирование новой версии БД и генерация отчетов 82
4.10. Создание новой пустой БД для очередной версии 93
4.11. Выгрузка данных старой версии БД....98
4.12. Загрузка и запуск новой версии БД 99
4.13. Дополнительные требования к построению модели 101
4.14. Пример модификации таблиц БД 104
Приложение 1. ПЕРЕЧЕНЬ ФУНКЦИЙ ПРОЕКТИРОВАНИЯ МОДЕЛИ ДАННЫХ И ИХ ИСПОЛЬЗОВАНИЕ....107
Приложение 2. ПЕРЕЧЕНЬ ФУНКЦИЙ ГРАФИЧЕСКИХ СРЕДСТВ DB2 И ИХ ИСПОЛЬЗОВАНИЕ....111
Приложение 3. ПЕРЕЧЕНЬ ФУНКЦИЙ УТИЛИТ DB2 ДЛЯ КОПИРОВАНИЯ ДАННЫХ И ИХ ИСПОЛЬЗОВАНИЕ....114
ЛИТЕРАТУРА....115

Предисловие

Сегодня уже никто не будет оспаривать тот факт, что для успешной реализации крупного проекта необходимы гибкие и мощные инструментальные средства, которые позволяют автоматизировать жизненный цикл создания и сопровождения программного обеспечения (ПО) и баз данных (БД). Разработка баз данных является сложным циклическим процессом, состоящим из следующих этапов проектирования и сопровождения БД:

анализа предметной области;
построения модели данных;
генерации кода для создания БД;
генерации различных процедур, реализующих бизнес-процессы;
генерации проектной и отчетной документации;
автоматического построения физической модели БД;
сравнения моделей БД;
слияния различных моделей в одну;
построения различных физических моделей по одной логической.

Разработка модели данных – это не разовый, а циклический процесс. Известно, что принятые архитектурные решения при разработке ПО являются определяющими как для быстродействия, так и жизнеобеспечения разрабатываемого ПО. Аналогично и для БД: архитектура БД, отображающая предметную область – это не механический набор плоских таблиц, которые видит в книге или документации рядовой пользователь. Архитектура БД – это в первую очередь модель данных предметной области, модель бизнес-процессов предметной области. Поэтому решения, принятые при разработке архитектуры БД, являются наиболее важными, как по быстродействию, так и по дальнейшей ее надежности при эксплуатации. Существующее у разработчиков мнение, что БД после ее создания остается неизменной, в корне ошибочно. БД развивается в процессе эксплуатации так же, как и ПО. Поэтому сразу нужно проектировать ПО и БД с учетом их дальнейшей модернизации.

«Плохая» архитектура БД, программное обеспечение, «привязанное» к БД, не позволяет обеспечить дальнейшую жизнь информационной системы. ПО нельзя изменить потому, что оно «привязано» к плохой БД, а базу данных нельзя изменить потому, что она «привязана» к плохому ПО.

CASE-средство AllFusion Erwin Data Modeler [1] позволяет автоматизировать процесс циклического построения модели данных, процесс моделирования данных, процесс построения архитектуры БД, отражающий предметную область в терминах стандартного языка моделирования БД, а также позволяет разработчику построить гибкую архитектуру БД. Данный инструмент в разы сокращает время разработки больших баз данных.

Настоящее учебно-методическое пособие адресовано студентам, магистрантам, аспирантам, изучающим основы проектирования информа-

ционных систем, а также системным аналитикам, руководителям проектов и разработчикам БД.

Учебно-методическое пособие состоит из четырех глав и приложений.

В главе 1 дается общее представление о процессе моделирования в среде AllFusion Erwin Data Modeler.

Глава 2 посвящена изложению основ построения и применения стандарта языка моделирования данных – IDEF1x.

В главе 3 излагаются методы построения логической модели данных, общие для проектирования БД в любой среде.

Глава 4 посвящена построению физической модели данных, созданию физической БД и ее сопровождению. В качестве среды реализации физической БД используется СУБД DB2 UDB, которая при практической разработке может быть легко заменена на любую другую СУБД.

Приложения содержат перечни функций проектирования и функций графических средств DB2, а также порядок применения их на каждом из этапов разработки и сопровождения БД.

В основу учебно-методического пособия положены результаты работ, полученные в процессе разработки и сопровождения конкретных больших производственных БД в среде AllFusion Erwin Data Modeler и материалы лекций.

1. Общие сведения о среде проектирования AllFusion Erwin Data Modeler

В настоящее время быстрое и грамотное проектирование и реализация баз данных (БД) без средств графического моделирования данных практически невозможны. Ручные методы разработки и сопровождения БД давно уже исчерпали себя в силу низкого качества и эффективности получаемого результата и низкой производительности труда.

В данной работе в качестве базового компонента проектирования и разработки БД используется готовое CASE-средство AllFusion Erwin Data Modeler [1], опирающееся на стандарты разработки БД FIPS, ISO9001 и язык моделирования БД IDEF1x [2]. CASE-средство AllFusion Erwin Data Modeler реализует структурные подходы к развитию информационной системы и к дизайну данных этих систем. Данное CASE-средство является лидером среди аналогичных систем, около 60 % средств моделирования данных мирового рынка принадлежит AllFusion Erwin Data Modeler.

Среда AllFusion Erwin Data Modeler не только помогает в дизайне (изображении) логической модели данных, она поддерживает дизайн соответствующей физической модели данных и автоматически генерирует структуру физической БД («**forward engineering**»).

AllFusion Erwin Data Modeler включает в себя средства для генерации из функционирующей физической БД соответствующей ей модели данных «**reverse engineering**», поддерживая при этом обе – физическую и логическую/ физическую модели данных. Таким образом, можно поддерживать функционирующие БД или осуществлять миграцию всей БД или ее части (подсхемы БД) на другие серверные платформы.

CASE-средство также включает средства автоматического сравнения моделей и баз данных («**complete compare**»), выдачи и анализа различий между ними, что позволяет в дальнейшем выборочно перемещать эти различия в модель или генерировать их в БД.

Общая функциональная схема проектирования БД с помощью CASE-средства AllFusion Erwin Data Modeler представлена на рис. 1.1.

При проектировании смешанной модели, включающей логический и физический уровень, переключение между ними осуществляется с помощью списка выбора «Model type indicator» – «Указатель типа модели» в панели инструментов AllFusion Erwin Data Modeler (logical / physical). Состояние этого указателя очень важно, так как оно определяет приоритет терминов логического уровня перед физическим. Надо первоначально определять объекты и их характеристики на логическом уровне, затем при первоначальном переключении на физический уровень соответствующие термины для него будут созданы автоматически (рис. 1.2).

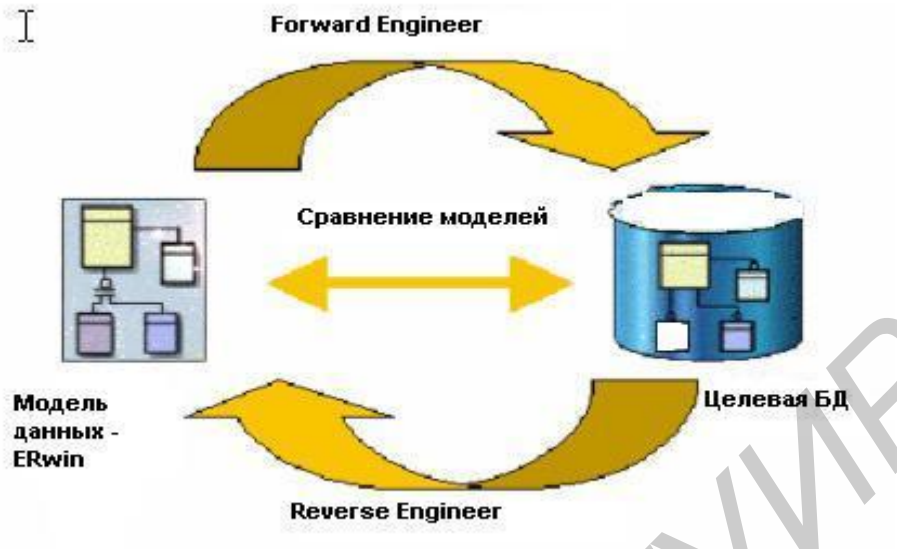

Рис. 1.1

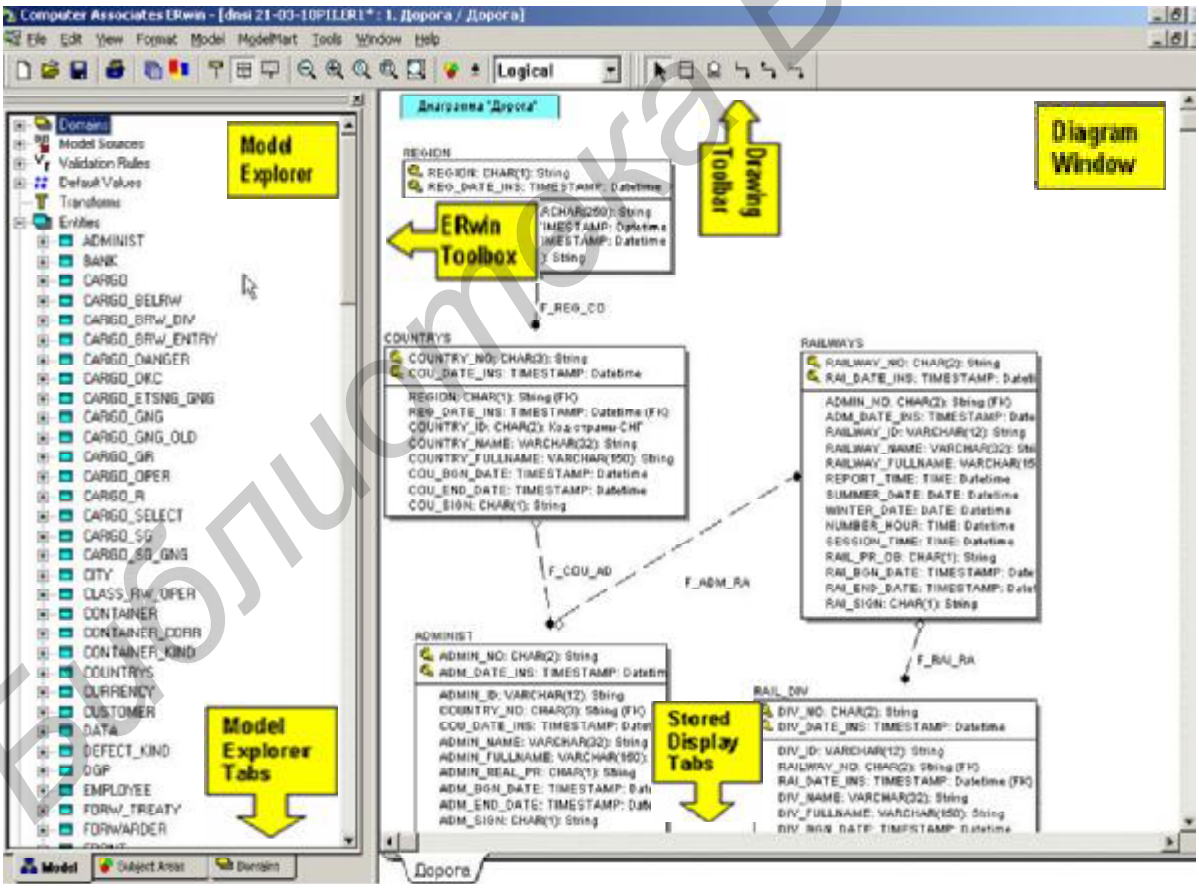

Рис. 1.2

«Указатель типа модели» кроме этого важен тем, что состояние этого указателя определяет описываемые ниже элементы – вид рассматриваемых ниже горизонтального меню и содержимого рабочих пространств.

Работа с компонентом организуется с помощью горизонтального меню «Menu bar» и двух рабочих пространств: «Model Explorer» и «Diagram Window». Горизонтальное меню AllFusion Erwin Data Modeler располагается в верхней части экрана и, хотя оно напоминает типовое для такого рода программ, полный перечень функций меню AllFusion Erwin Data Modeler достаточно большой и требует некоторого времени для поэтапного прохождения по всем операциям и возможностям, некоторые из которых выполняются достаточно редко. Рабочее пространство «Model Explorer» расположено в левой стороне экрана. Оно обеспечивает иерархическое текстовое изображение модели данных, которая синхронно в графическом виде представляется в правой части экрана в рабочем пространстве «Diagram Window».

В любом из этих пространств по своему выбору разработчик БД может осуществлять вызов всех операций по обслуживанию элементов логической и физической схем БД: создание, удаление, переименование, изменение характеристик схем, подсхем, объектов, атрибутов, связей, таблиц, столбцов, ключей, шаблонов, триггеров, скриптов, табличных пространств, пулов, буферов и других характеристик. AllFusion Erwin Data Modeler обеспечивает координацию внесения изменений в обоих рабочих пространствах – как в текстовом, так и в графическом виде.

Вся работа в рабочем пространстве «Model Explorer» выполняется в текстовом виде, т.е. вводом в выдаваемые AllFusion Erwin Data Modeler окна и поля соответствующих значений, либо путем выбора их из предоставленного AllFusion Erwin Data Modeler списка значений.

В «Diagram Window» инструментарий AllFusion Erwin Data Modeler синхронно создает или изменяет объекты схемы в графическом режиме. Разработчик БД может работать в этом окне, используя механизм перетягивания графических элементов схемы (объектов, связей) из AllFusion Erwin Data Modeler Toolbox и Drawing toolbar в окно «Diagram Window». Здесь можно просто щелкнуть мышкой по любому объекту или связи и вызвать соответствующее им окно для выдачи сведений о нем и их корректировки.

Общая функциональная схема процесса создания БД на основе логической и физической схем представлена на рис. 1.3.

Создание модели данных – длительный итерационный процесс по идентификации и документированию предметной области в той части общесистемных требований, которая касается структур данных и взаимосвязей между ними.

В общем случае, разработка и сопровождение БД включает такие работы, как анализ предметной области, создание схемы БД, корректировку и поддержание схемы БД в актуальном состоянии, выпуск новой версии БД, выгрузку и загрузку данных. Жизненный цикл БД (по аналогии с жизненным

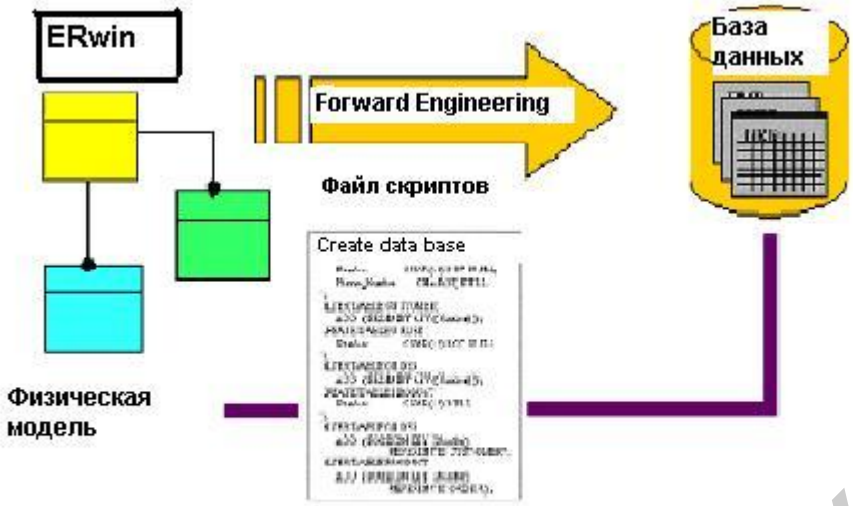

Рис. 1.3

циклом программного обеспечения) включает не только этап создания (проектирования) логической и физической моделей БД, этап генерации БД, этап загрузки данных и тестирования, но и этапы работ, которые выполняются при сопровождении БД:

получение исходных данных для внесения изменений и подготовка документации для внесения изменений;

выполнение подготовительных действий по анализу модели эксплуатируемой БД и планированию создания и запуска ее новой версии;

выполнение редактирования старой схемы и получение новой схемы, получение задания на языке DDL для создания новой версии БД;

запуск задания на создание новой пустой БД;

выполнение выгрузки (экспорта) данных из старой версии БД;

выполнение загрузки (импорта) данных из файлов-копий формата IXF в новую версию БД;

внесение или дополнение необходимых данных в новые объекты новой версии;

выполнение процедуры проверки и опытной эксплуатации новой версии БД;

ввод в промышленную эксплуатацию новой версии БД и объявление о ее выпуске.

В пп. 3 и 4 производится описание технологии выполнения работ и порядок выполнения операций по проектированию и сопровождению БД.

2. Основные элементы языка моделирования баз данных IDEF1x

2.1. Сущности

При моделировании на языке IDEF1x на логическом уровне, объекты реального мира представляются сущностями (одной или более). Сущность отражает набор экземпляров объекта (объектов) реального мира, представляет совокупность или набор экземпляров, похожих по свойствам, но однозначно отличаемых друг от друга. Каждая сущность имеет имя и некоторый набор атрибутов (моделируемых характеристик реального мира). Кроме имени и атрибутов каждая сущность должна иметь текстовое определение (definition) того, что собой представляет данная сущность. В дальнейшем это определение используется как документация к проекту и как комментарий к таблице, физически представляющей данную сущность в конкретной СУБД. Например, ниже в табл. 2.1, 2.2 и 2.3 приведены определения сущности *Страны*, их атрибутов и индексов для работы с данными.

Таблица 2.1

Столбцы таблицы *COUNTRIES – Страны*

Name	**Definition/Comment**
COUNTRY_NO	Код страны
COU_DATE_INS	Дата вставки записи в таблицу COUNTRIES
REGION	Код региона
REG_DATE_INS	Дата вставки записи в таблицу REGION
COUNTRY_ID	Мнемокод страны
COUNTRY_NAME	Наименование страны
COUNTRY_FULLNAME	Полное наименование страны
COU_BGN_DATE	Дата и время начала действия
COU_END_DATE	Дата и время окончания действия
COU_SIGN	Признак состояния записи

Таблица 2.2

Индексы *COUNTRIES Table*

Name	**Definition/Comment**	**Type**
P_K_COU_	Код страны	PK
H_COU_NO	Индекс историчности объекта	AK1
I_COU_NO	Индекс поколения объекта	IE1
F_REGION	Ссылка на «Регионы»	FK

Таблица 2.3

Первичный ключ *COUNTRIES*

Name	**Datatype**	**Definition/Comment**
COUNTRY_NO	CHAR(3)	Код страны
COU_DATE_INS	TIMESTAMP	Дата вставки записи в таблицу COUNTRIES

Конкретное значение объекта представляется конкретным значением набора каждого из атрибутов сущности (или сущностей) и называется экземпляром сущности (instance), другими словами, значением сущности. Каждый экземпляр сущности однозначно идентифицируется с помощью значений одного или более атрибутов. Данные атрибуты образуют первичный ключ. При реализации логической модели средствами конкретной СУБД каждая сущность на физическом уровне, как правило, отображается в таблицу, а набор атрибутов сущности – в колонки таблицы с указанием типа данных каждой колонки. Конкретный экземпляр значений всех колонок представляется записью в данной таблице, т.е. каждая запись в таблице (или таблицах) отражает некоторое значение объекта реального мира. Первичный ключ используется для поиска конкретной записи в таблице. На примере (рис. 2.1), приведена таблица Countries, которая отражает сущность *Страна* и атрибуты (колонки) с комментариями, которые отражают характеристики каждой реальной страны (*Код страны COUNTRY_NO*, *Наименование страны COUNTRY_NAME* и т.д.), а первичным ключом является колонка *COUNTRY_NO* – атрибут *Код страны* и скрытый атрибут «Дата вставки записи в таблицу» (*COU_DATE_INS*).

NSI. COUNTRIES *Страна*

стр. 1 из 20 >> >|

Код страны COUNTRY_NO	Код региона REGI–ON	Мнемокод страны COUNTRY_ID	Наименование страны COUNTRY_NAME	Полное наименование страны COUNTRY_FULLNAME
000	N	NOT	НЕОПРЕДЕЛЕН–НОЕ ЗНАЧЕНИЕ	НЕОПРЕДЕЛЕННОЕ ЗНАЧЕНИЕ
004	O	AF	АФГАНИСТАН	РЕСПУБЛИКА АФГАНИСТАН
008	O	AL	АЛБАНИЯ	НАРОДНАЯ СОЦИАЛИСТИЧЕСКАЯ РЕСПУБЛИКА АЛБАНИЯ
010	O	AQ	АНТАРКТИКА	АНТАРКТИКА
012	O	DZ	АЛЖИР	АЛЖИРСКАЯ НАРОДНАЯ ДЕМОКРАТИЧЕСКАЯ РЕСПУБЛИКА
016	O	AS	ВОСТОЧ. САМОА (США)	АМЕРИКАНСКОЕ САМОА
020	O	AD	АНДОРРА	КНЯЖЕСТВО АНДОРРА
024	O	AO	АНГОЛА	НАРОДНАЯ РЕСПУБЛИКА АНГОЛА (НРА)
028	O	AG	АНТИГУА И БАРБУДА	АНТИГУА И БАРБУДА
031	S	AZ	АЗЕРБАЙДЖАН	АЗЕРБАЙДЖАН
032	O	AR	АРГЕНТИНА	АРГЕНТИНСКАЯ РЕСПУБЛИКА
036	O	AU	АВСТРАЛИЯ	АВСТРАЛИЯ

Рис. 2.1

2.2. Связи, отношения

В процессе моделирования сущности связываются отношениями, например, отдел A *состоит из* сотрудников B, сотрудник из B *пишет* программу C и D. В данном случае, связь является логическим отношением между сущностями. При таком связывании сущность A является родительской для сущности B, а сущность B является потомком (дочерней) для сущности A. Сами сущности могут быть *зависимыми* и *независимыми*, а связи – *идентифицирующими* (между *независимой* и *зависимой* сущностями) или *неидентифицирующими* (дочерняя сущность остается независимой). В случае если потомок связан *идентифицирующей* связью и дочерняя сущность *зависит* от родительской, то каждое значение родительской сущности связано с одним или более значениями дочерней сущности и отображает связи, которых не может быть без родительской сущности. В этом случае любое значение дочерней сущности идентифицируется с помощью некоторого значения родительской сущности, а атрибуты первичного ключа родительской сущности мигрируют в атрибуты первичного ключа дочерней сущности.

Если же связи между двумя сущностями не могут быть идентифицированы, но существуют сущности *независимые*, то любое значение дочерней сущности однозначно определяется без значения родительской сущности и атрибуты первичного ключа родительской сущности мигрируют в состав неключевых атрибутов дочерней сущности. В языке IDEF1x *независимые* сущности изображаются прямоугольником с закругленными углами, а *зависимые* – прямоугольником с овальными углами, сами связи изображаются прямой линией с точкой (рис. 2.2 (сплошная линия – *идентифицирующие*, прерывистая – *неидентифицирующие*)).

2.3. Мощность связей

Спецификация мощности связей (отношений) служит для обозначения количества экземпляров (значений) дочерних сущностей, связанных с соответствующим значением родительской сущности. Здесь приняты следующие обозначения: Pa – родительская сущность, Ch – дочерняя.

Pa ——● Ch 0, 1 или много

Pa ——● Ch 1 или много
p

Pa ——● Ch 0 или 1
z

Pa ——● Ch точно N (7)
7

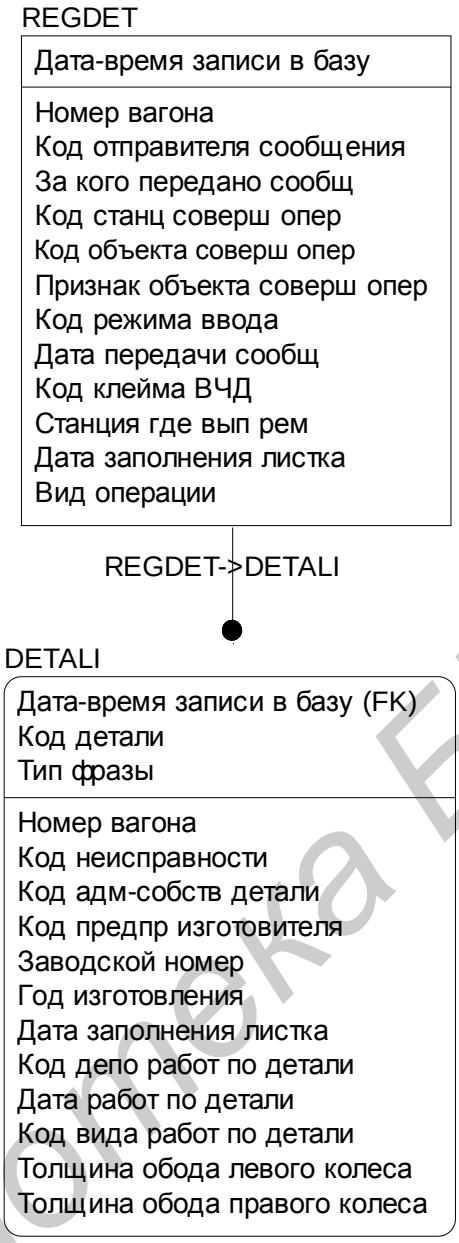

Рис. 2.2

Pa ——● Ch от n до m

N – m

Pa ●—— Ch многие – ко многим, отношения не установлены, должны быть разрешены позже.

На рис. 2.3 приведен диалог установления мощности отношений.

2.4. Ключи

Как уже отмечалось выше, каждый экземпляр сущности должен быть уникален и каждое значение атрибутов должно отличаться от другого.

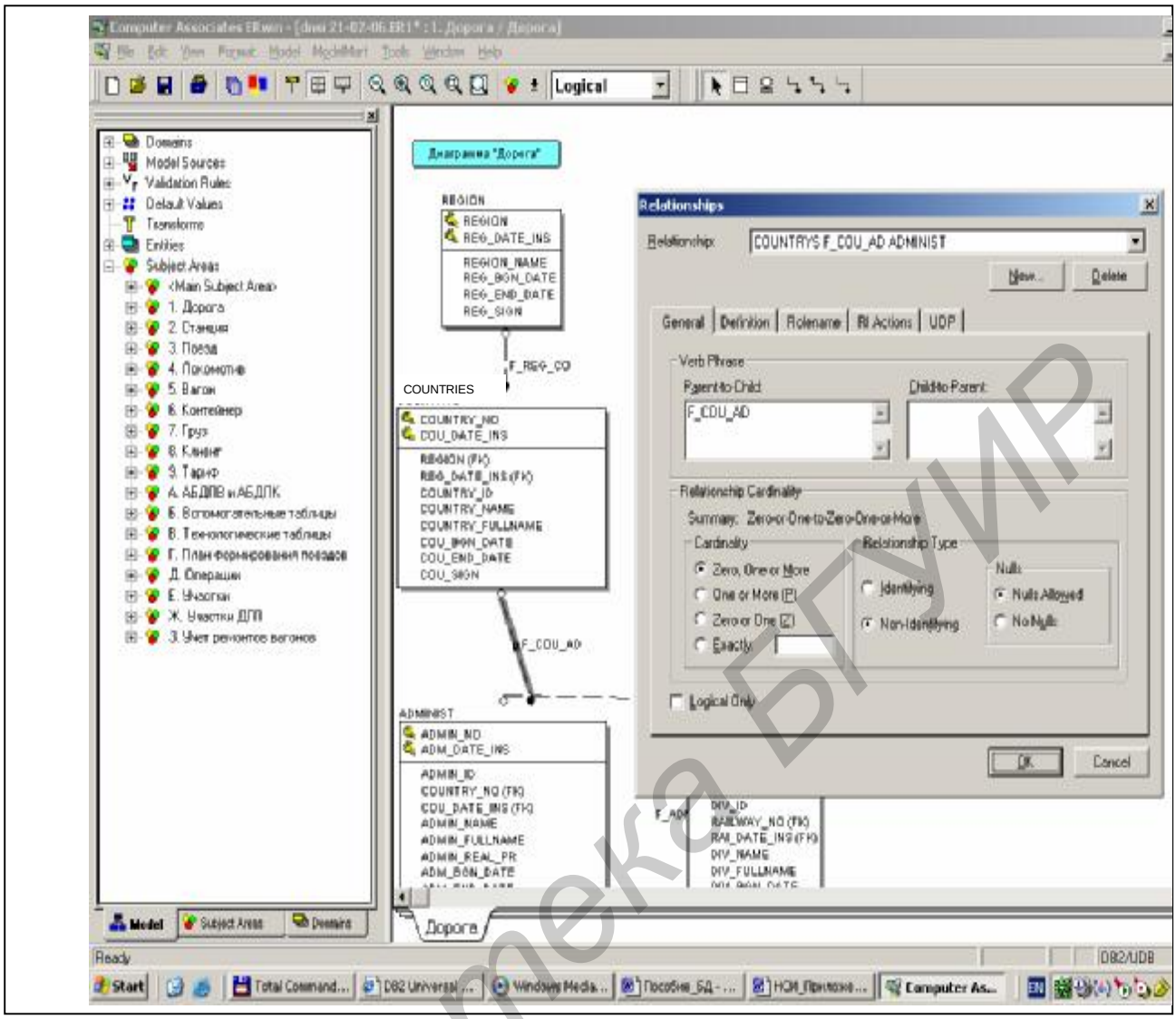

Рис. 2.3

Для идентификации каждого экземпляра сущности используется первичный ключ – PK (Primary Key). Связи между сущностями реализуются с помощью внешних ключей – FK (Foreign Key), которые образуются из PK путем их миграции из родительской сущности в дочернюю. Сущности и их связи, ключи PK и FK – это основные элементы модели данных. Кроме ключей PK и FK при построении модели данных используются альтернативные и инверсные ключи – AK (Alternate Key) и IK (Inversion Key) (см. разд. 3). Альтернативные ключи так же, как и PK, обеспечивают уникальный доступ к значениям сущности, но по другим наборам атрибутов, а инверсные ключи обеспечивают доступ к некоторому набору значений сущности, имеющему общие характеристики.

На рис. 2.4 приведен диалог определения ключей сущности *Countries* с помощью окна для ключевой группы.

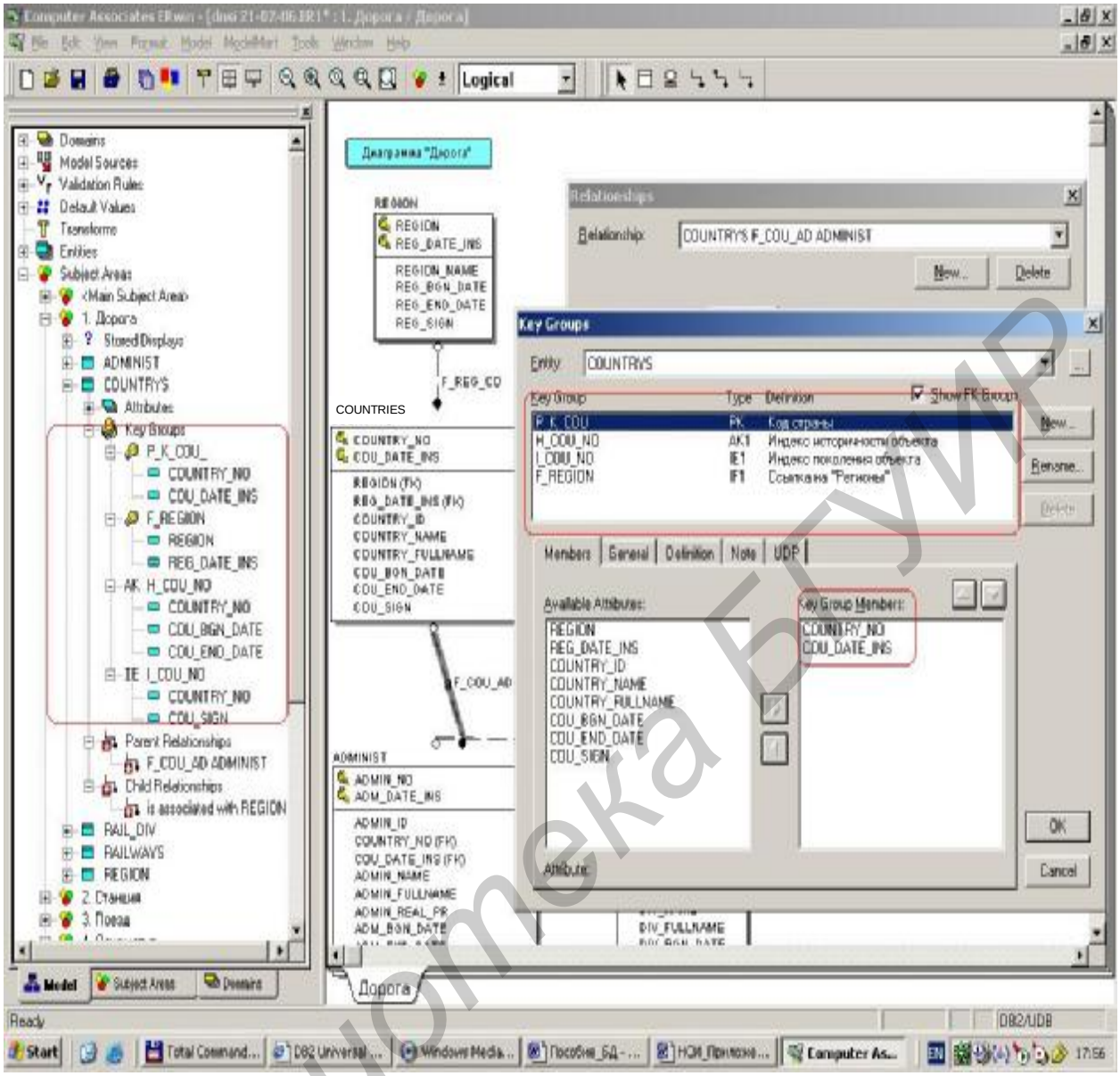

Рис. 2.4

2.5. Внутренние и внешние ключи

Одно из важнейших понятий, используемых при работе с базами данных, – это первичные ключи, с их помощью идентифицируются значения сущности (записи в таблице). В качестве первичного ключа может использоваться естественный ключ или суррогатный.

Естественный ключ – это атрибут или набор атрибутов, которые однозначно идентифицируют значение сущности (записи таблицы) и имеют некий физический смысл вне БД (номер вагона, номер станции, номер контейнера и т.д.). Данный метод построения модели БД основан на естественной идентификации сущностей. Уникальность записи может

достигаться не только значением уникального кода внешнего объекта, но, например, и датой вставки записи в таблицу (первичный ключ PK – состоит из значения кода объекта и даты вставки записи в БД) (см. примеры диаграмм, приведенные выше).

Конечно, естественная идентификация значений сущностей более понятна и предпочтительна, т.к. нет необходимости вводить дополнительные атрибуты, не несущие никакой внешней смысловой нагрузки. Да и при эксплуатации БД в сложных ситуациях можно восстановить их значения. Но не всегда возможно выбрать атрибут или короткий набор атрибутов для идентификации значений. В этом случае без использования суррогатных ключей трудно обойтись.

Суррогатный ключ – автоматически сгенерированное значение, никак не связанное с информационным содержанием записи, которое имеет некий физический смысл только внутри БД; обычно в роли суррогатного ключа могут использоваться данные типа integer или datestamp.

В данном методе построения модели БД суррогатная идентификация значений сущностей позволяет решить проблему идентификации за счет введения внутреннего ID – идентификатора сущности и выработки внутреннего суррогатного ключа (PK) для каждого значения в БД. Такой подход предполагает, что для каждой сущности существует набор определенных характеристик, которые в процессе жизни могут меняться или уточняться. Поэтому в БД каждой сущности на предприятии присваивается своя внутренняя идентификация, уникальная на всем протяжении жизни сущности, и все характеристики (текущие или измененные) уникально идентифицируются для каждой сущности (создается уникальный внутренний суррогатный ID сущности).

Данный уникальный идентификатор каждого значения в БД является первичным ключом записи (PK) в сущностях (таблицах), содержащих значения характеристик сущностей. На рис. 2.5 приведена таблица ID для выработки и хранения суррогатного ID, на основе которого создаются новые суррогатные PK-ключи для значений сущностей (таблиц) всех других таблиц в БД.

2.6. Ссылочная целостность

Одной из главных задач при эксплуатации БД является задача поддержания правил ссылочной целостности, которые были определены при проектировании. Эти правила отражают заданные связи между сущностями и порядок выполнения операций вставки, замены и удаления. Если эти правила не будут соблюдены, то БД быстро деградирует и не будет отражать те бизнес-правила, которые были заложены при ее проектировании. Правила контроля и выполнения этих операций могут быть возложены на специально разработанные пользователем программы, но данное решение крайне ненадежное и трудоемкое для его реализации. Задача поддержания правил

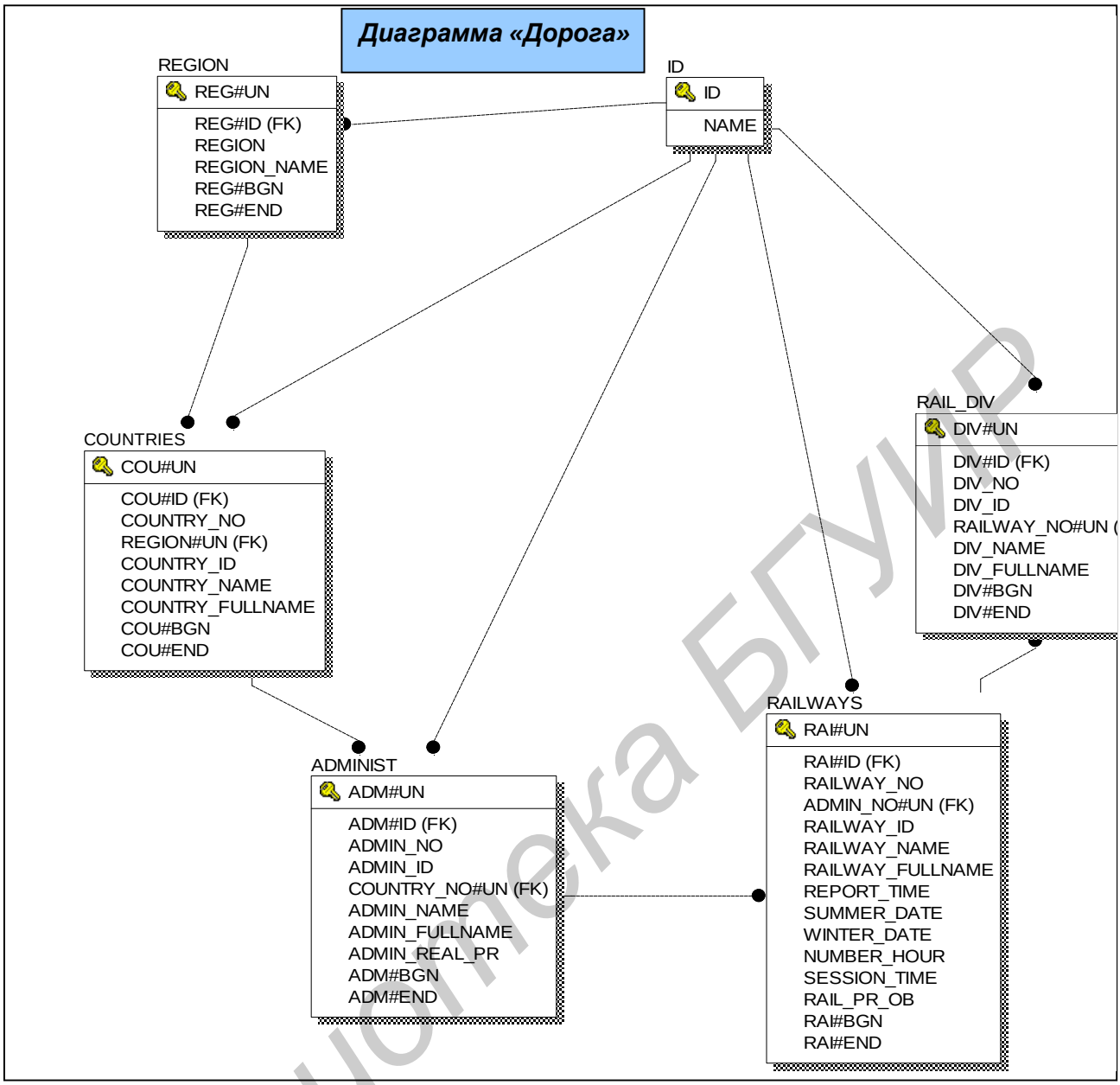

Рис. 2.5

ссылочной целостности обычно реализуется автоматически при генерации схемы БД на основе режимов реализации связей (отношений) с помощью триггеров, обеспечивающих ссылочную целостность. Данные триггера представляют собой программы, выполняемые всякий раз при выполнении операций вставки, замены и удаления. Таким образом, при проектировании автоматически обеспечивается правильная эксплуатация БД.

На рис. 2.6 приведен пример определения правил ссылочной целостности для реализации связи F_COU_AD.

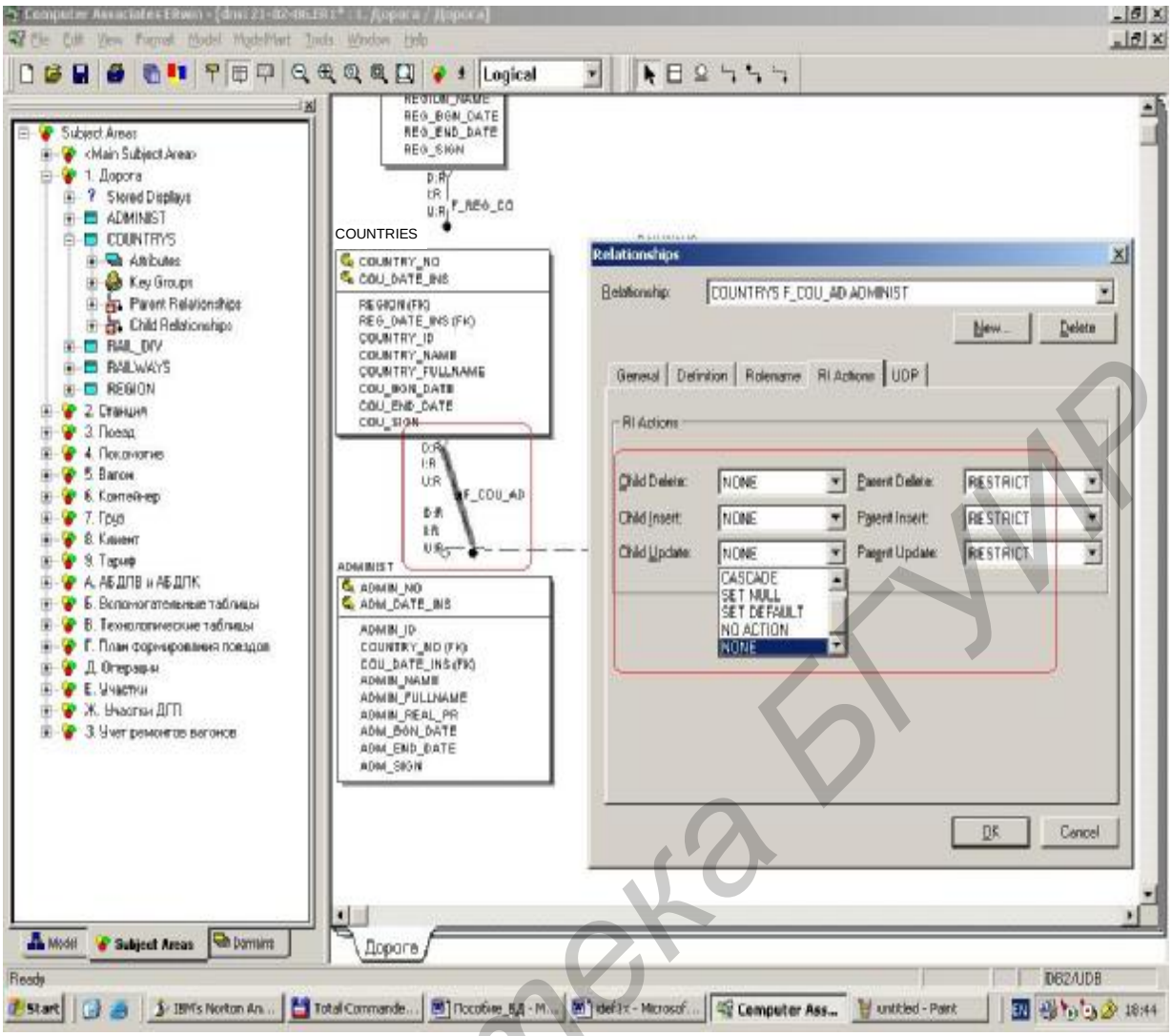

Рис. 2.6

По умолчанию Erwin автоматически генерирует значения режимов ссылочной целостности исходя из контекста схемы БД для реализации каждой связи. Данное значение может быть изменено проектировщиком БД. В общем случае правила ссылочной целостности определяются как **<символ – целостности>:<режим – целостности>** и для родителя, и для потомка (табл. 2.4).

Таблица 2.4

Режимы ссылочной целостности

Операция: вставка, замена и удаление (символ – целостности)		**Режим – целостности**
Родитель	**Потомок**	**C**ascade
Insert	***I***nsert	***R***estrict
Delete	***D***elete	***S***et Null
Update	***U***pdate	***S***et Default
		No ***A***ction
		None

Cascade – задает каскадное действие (например удаление родительской записи влечет удаление всех записей у потомков), ***R***estrict – задает ограничение на выполнение действия, (например, нельзя удалить родительскую запись, пока есть записи у потомков, и наоборот), ***S***et Null – задает действие по установлению FK, равным NULL, при удалении родительской записи, ***S***et Default – задает действие по установлению FK, равным Default, при удалении родительской записи, ***N***o ***A***ction – никаких действий не требуется, если удалены потомки (например, D:R – для родителя, D:S – для потомка или I:R – для потомка, если нет родителя).

2.7. Правила построения диаграмм, виды связей, категории, рекурсии

2.7.1. Диаграммы моделей строятся из сущностей (зависимых и независимых) и связей (идентифицирующих и неидентифицирующих) и отражают моделируемую предметную область. Ниже, на рис. 2.7 приведена диаграмма модели «Участки диспетчирования», которая состоит из сущностей «Диспетчерские участки», «Укрупненные участки» и «Линейные участки» и отношений, связывающих эти сущности. В данном случае дочерние сущности зависят от родительских сущностей и в качестве отношений используются «идентифицирующие» связи. Родительские PK мигрировали в ключевую часть атрибутов потомков.

2.7.2. На рис. 2.8 приведен фрагмент модели топологии железной дороги: в общем случае есть «Администрация дорог», которой подчиняются «Железные дороги» в данном государстве (Россия, Беларусь и др.). «Железные дороги» делятся на «Отделения» и состоят из «Станций», полная административная подчиненность известна только для белорусских станций. В данной диаграмме используются неидентифицирующие отношения, дочерние сущности не зависят от родительских, т.е. некоторые FK могут иметь значение «по умолчанию», неравное значению родительского PK, «нет родительской записи».

2.7.3. На рис. 2.9 сущность «Станция», содержащая все станции стран СНГ, используется для построения других сложных понятий, таких как «Внешний стык» и «Пограничный переход». Сущность «Внешний стык» определяется двумя станциями с одной и другой стороны «Железной дороги», а сущность «Пограничный переход» определяется пятью станциями (рис. 2.10). Общее множество по выделенным свойствам разбивается на другие подмножества, обладающие определенными признаками. Так как ключевой атрибут STATION_NO не может мигрировать больше одного раза как внешний ключ, то в дочерних сущностях данный атрибут переименован в STATION_NO_1, …., STATION_NO_5.

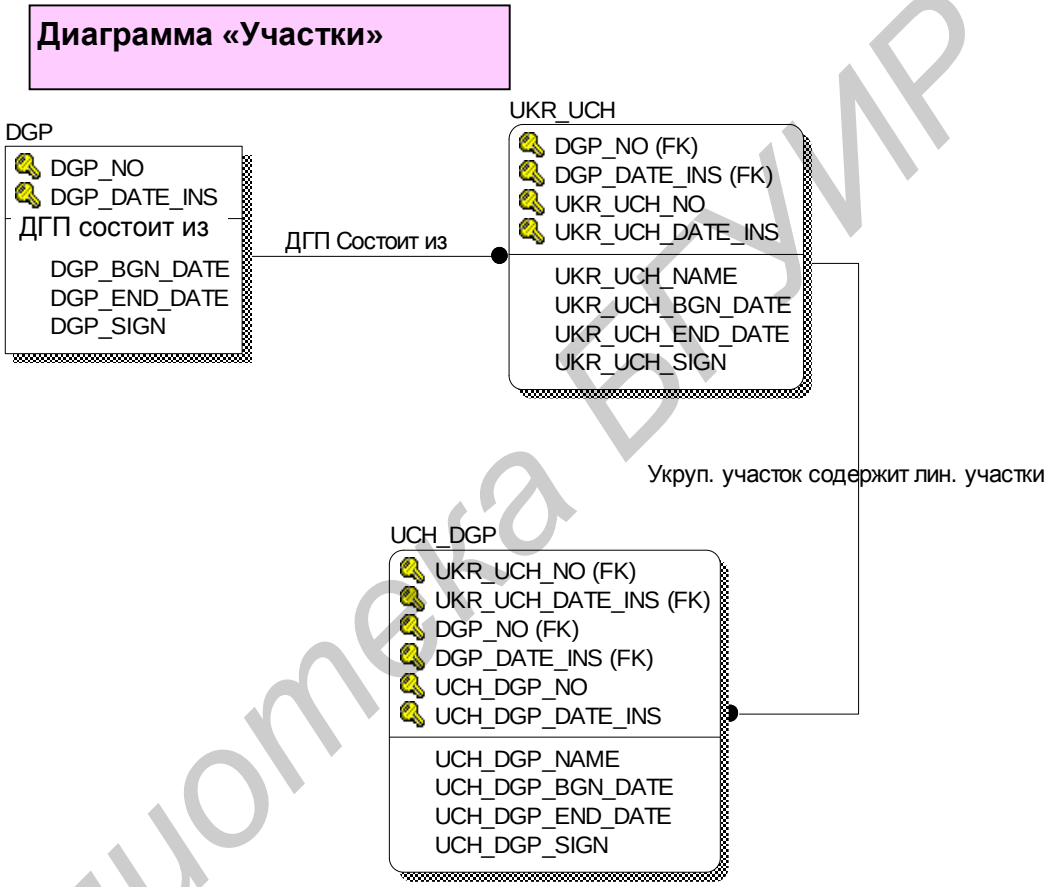

Рис. 2.7

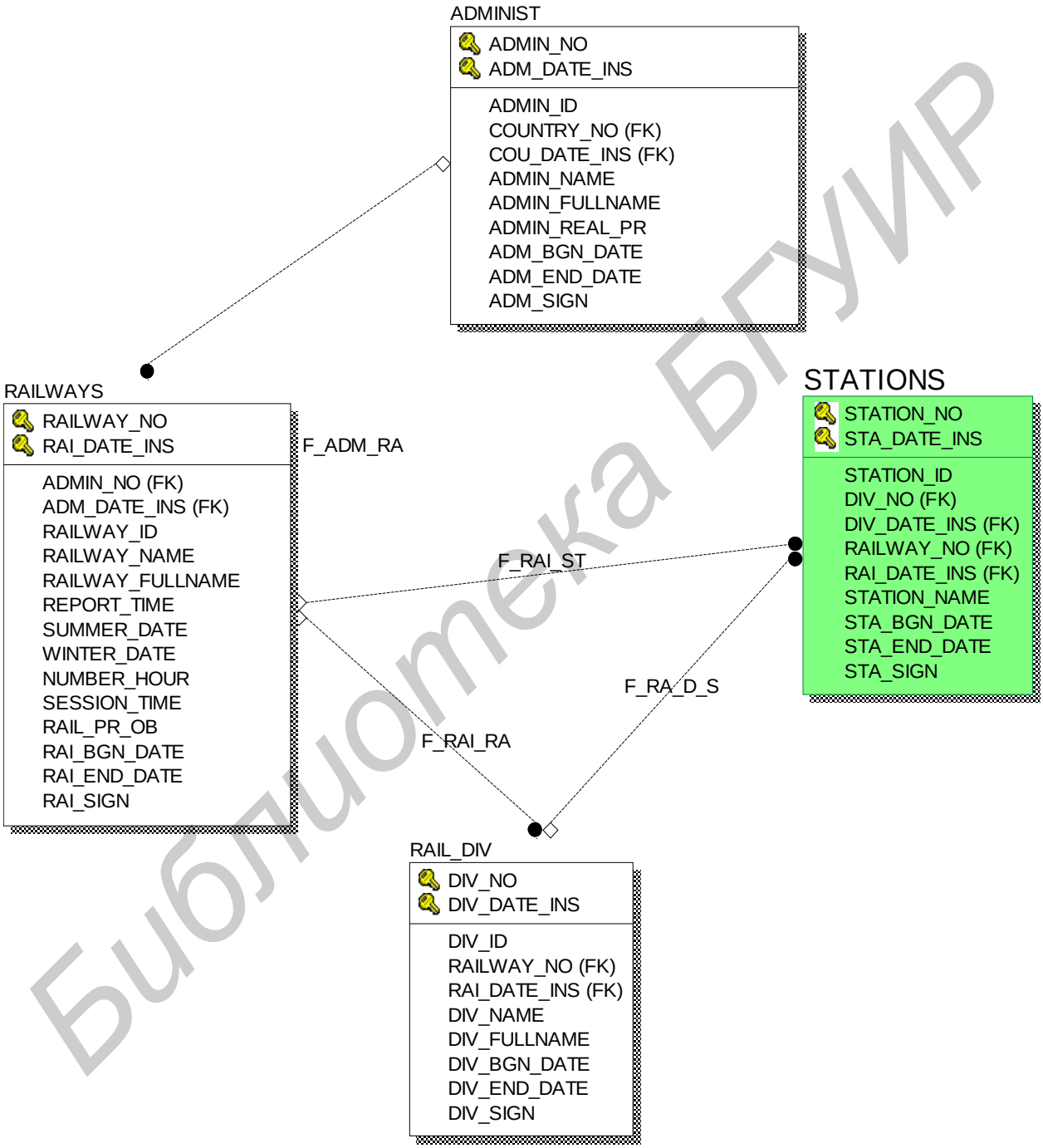

Рис. 2.8

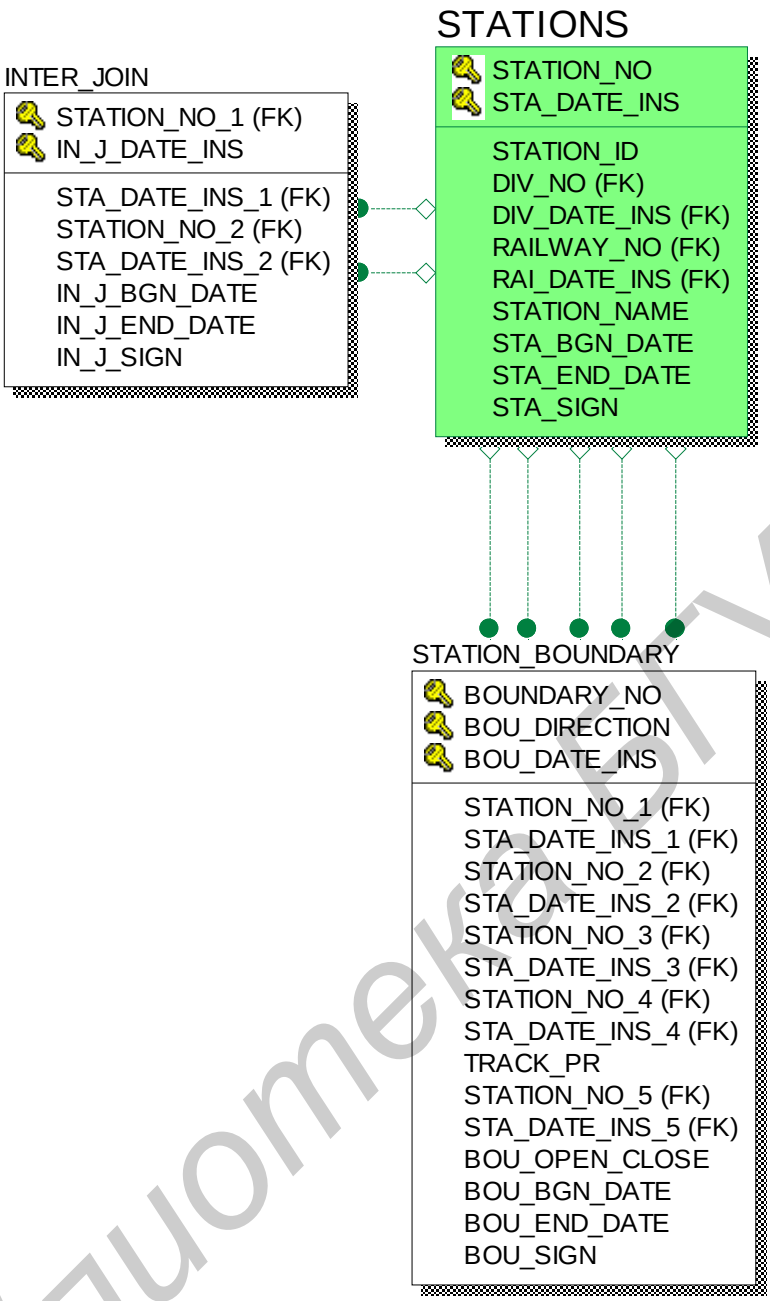

Рис. 2.9

NSI.STATION_BOUNDARY «Пограничный переход»

Номер погран. перехода	Номер направле-ния в погран. переходе	Код станции передачи вагонов БелЖД	Код стыковой погран. станции БелЖД	Код стыковой погран. станции не БелЖД	Код станции передачи вагонов не БелЖД	Код стыкового пункта учета поездов и вагонов
1	1	138507	136605	121101	121008	120518
1	2	137608	136605	121101	121008	120518
1	3	138507	136605	121101	120804	120518
1	4	137608	136605	121101	120804	120518
2	1	135208	135000	NULL	121008	134900
3	1	162900	164107	120202	121008	163000
3	2	162900	164107	120202	120804	163000
4	1	161306	161401	110200	110003	161700
5	1	161306	161202	066900	067000	161607
6	1	160002	161005	067405	067104	160801
7	1	160002	165805	172008	170004	165608
8	1	166704	169100	171401	170004	171346

Рис. 2.10

2.7.4. На рис. 2.11 и 2.12 приведены примеры разрешения связей многие – ко многим. Данный тип отношений возникает в том случае, если между сущностями не удается установить связь типа «родитель – потомок», эти сущности не зависят друг от друга. Так, на рис. 2.11 приведены две сущности: классы операций (OPER_CLASS) и сами операции (RW_OPER). С помощью дополнительной сущности «класс операции – операции» (CLASS_RW_OPER) установлены отношения между сущностями «класс операции» и «операции». Множество операций сущности «операция» классифицировано по определенным признакам и разрешены отношения многие – ко многим. Фактически новая сущность «класс операции – операция» может состоять только из всевозможных пар PK-ключей обеих сущностей, которые образуют PK новой сущности.

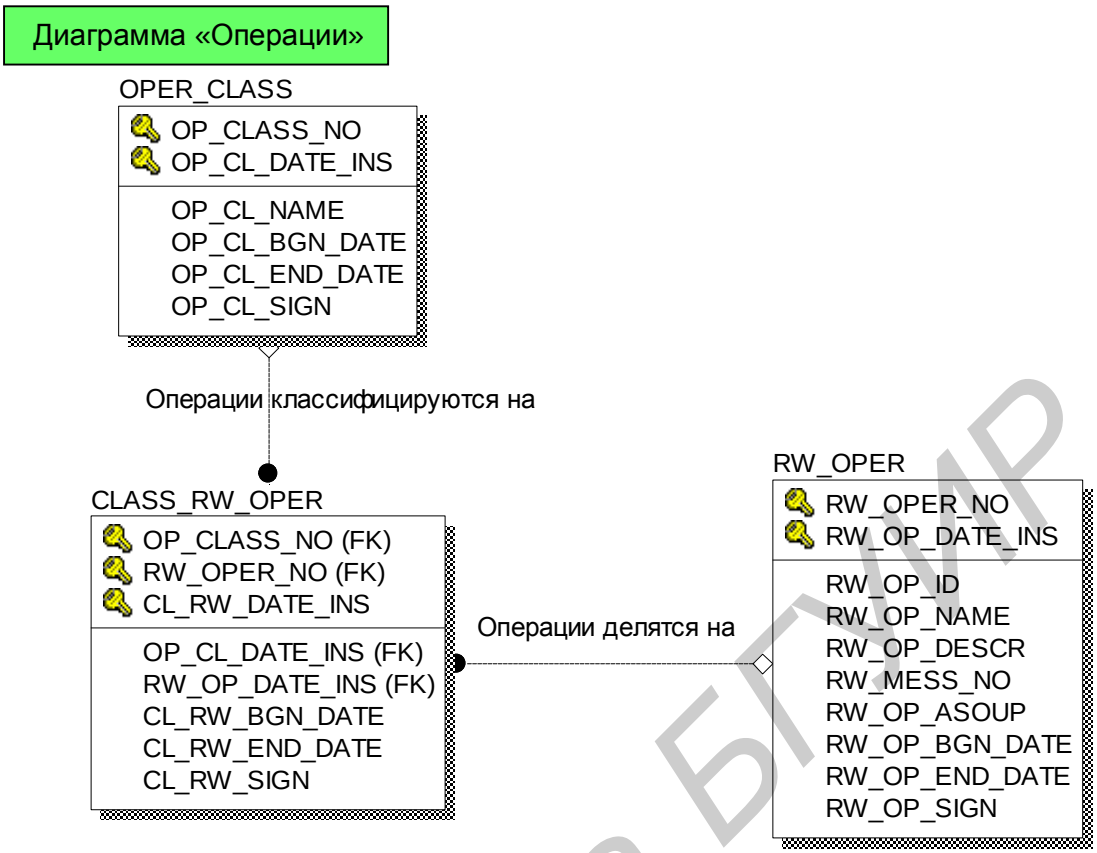

Рис. 2.11

Рис. 2.12 содержит аналогичное решение для проблемы связи между станциями и видами грузовых работ (параграфами станций), допустимых на них. На различных станциях могут выполняться различные виды грузовых работ, а на некоторых выполнение таких работ вообще не допускается. На рис. 2.13–2.15 приведены некоторые конкретные значения данных сущностей, взятые из БД.

2.7.5. На рис. 2.16 приведена не простая диаграмма, отражающая топологию участков железной дороги: кроме того на этой диаграмме отражены участки движения локомотивов. Опять же, основным строительным элементом является сущность «Станция» и ее подмножество «Выделенная станция». Логическая сущность «Участок» объединяет такие объекты, которые содержат информацию об участках железной дороги, принадлежности станций участкам, выделенных станциях, ближайших выделенных станциях, длине участков и времени хода на них, участках обращения локомотивов, принадлежности участков железной дороге, участку обращения локомотивов, расстояниях и времени хода от выделенных до невыделенных станций. Даная диаграмма в терминах языка IDEF1x реализует географическую топологию железной дороги: «Станция» (все станции), «Выделенная станция» (подмножество Станций), «Участок» (участок дороги от одной выделенной станции до другой,

на котором располагаются другие станции), «Участок обращения локомотивов» (состоит из нескольких участков дороги) и содержит все элементы данных этой топологии.

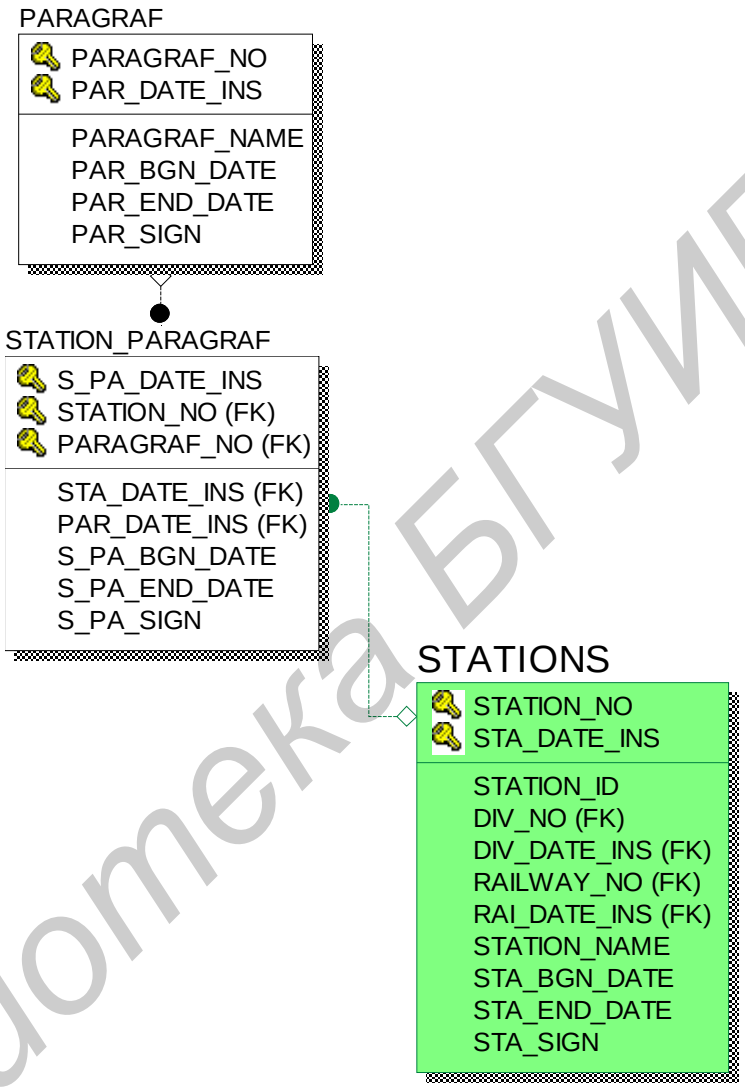

Рис. 2.12

NSI.PARAGRAF «Параграф станции»

Код параграфа PARAGRAF_NO	Наименование параграфа PARAGRAF_NAME
1	ПРИЕМ И ВЫДАЧА ПОВАГОННЫХ ОТПРАВОК ГРУЗОВ, ДОПУСКАЕМЫХ К ХРАНЕНИЮ НА ОТКРЫТЫХ ПЛОЩАДКАХ СТАНЦИЙ
10	ПРИЕМ И ВЫДАЧА ГРУЗОВ В УНИВЕРСАЛЬНЫХ КОНТЕЙНЕРАХ ТРАНСПОРТА МАССОЙ БРУТТО 30 Т НА СТАНЦИЯХ
10Н	ПРИЕМ И ВЫДАЧА ГРУЗОВ В УНИВЕРСАЛЬНЫХ КОНТЕЙНЕРАХ МАССОЙ БРУТТО 24 (30) И 30 Т НА ПОДЪЕЗДНЫХ ПУТЯХ
2	ПРИЕМ И ВЫДАЧА МЕЛКИХ ОТПРАВОК ГРУЗОВ,ТРЕБУЮЩИХ ХРАНЕНИЯ В КРЫТЫХ СКЛАДАХ СТАНЦИЙ
3	ПРИЕМ И ВЫДАЧА ГРУЗОВ ПОВАГОННЫМИ И МЕЛКИМИ ОТПРАВКАМИ, ЗАГРУЖАЕМЫМИ ЦЕЛЫМИ ВАГОНАМИ, ТОЛЬКО НА ПОДЪЕЗДНЫХ ПУТЯХ И МЕСТАХ НЕОБЩЕГО ПОЛЬЗОВАНИЯ
4	ПРИЕМ И ВЫДАЧА ПОВАГОННЫХ ОТПРАВОК ГРУЗОВ,ТРЕБУЮЩИХ ХРАНЕНИЯ В КРЫТЫХ СКЛАДАХ СТАНЦИЙ
5	ПРИЕМ И ВЫДАЧА ГРУЗОВ В УНИВЕРСАЛЬНЫХ КОНТЕЙНЕРАХ ТРАНСПОРТА МАССОЙ БРУТТО 3 И 5 Т НА СТАНЦИЯХ
6	ПРИЕМ И ВЫДАЧА ГРУЗОВ В УНИВЕРСАЛЬНЫХ КОНТЕЙНЕРАХ ТРАНСПОРТА МАССОЙ БРУТТО 3 И 5 Т НА ПОДЪЕЗДНЫХ ПУТЯХ
7	ЗАПРЕЩАЕТСЯ ПРИЕМ И ВЫДАЧА ЛЕГКОВОСПЛАМЕНЯЮЩИХСЯ ГРУЗОВ НА СТАНЦИЯХ
8	ПРИЕМ И ВЫДАЧА ГРУЗОВ В УНИВЕРСАЛЬНЫХ КОНТЕЙНЕРАХ ТРАНСПОРТА МАССОЙ БРУТТО 20 Т НА СТАНЦИЯХ
8Н	ПРИЕМ И ВЫДАЧА ГРУЗОВ В УНИВЕРСАЛЬНЫХ КОНТЕЙНЕРАХ МАССОЙ БРУТТО 20 И 24 Т НА ПОДЪЕЗДНЫХ ПУТЯХ
9	ПРИЕМ И ВЫДАЧА МЕЛКИХ ОТПРАВОК ГРУЗОВ, ДОПУСКАЕМЫХ К ХРАНЕНИЮ НА ОТКРЫТЫХ ПЛОЩАДКАХ СТАНЦИЙ

стр. 1 из 2 >> >|

Рис. 2.13

NSI.STATION_PARAGRAF «Таблица соответствия станций параграфам»

Код станции (ECP) STATION_NO	Код параграфа PARAGRAF_NO
010708	3
010801	3
010905	3
011005	3
011109	3
011202	3
011306	1
011306	2
011306	3
011306	4
011306	9
011400	1

Рис. 2.14

NSI.STATIONS «Станция»

Код станции (ЕСР) STATION_NO	Мнемокод станции STATION_ID	Код отделения DIV_NO	Код дороги RAILWAY_NO	Наименование станции STATION_NAME
137006	СЛОН	02	13	СЛОНИМ
137101	ПЛНК	02	13	ПОЛОНКА
137129	NULL	02	13	БОРОВЦЫ
137203	МОСТ	02	13	МОСТЫ
137307	РОСЬ	02	13	РОСЬ
137400	РОЖН	02	13	РОЖАНКА
137504	СКРБ	02	13	СКРИБОВЦЫ
137608	ЛИДА	02	13	ЛИДА
137612	NULL	02	13	МИНОЙТЫ
137701	ГУДЫ	02	13	ГУДЫ
137805	БАСТ	02	13	БАСТУНЫ
137909	БЕНК	02	13	БЕНЯКОНИ

Рис. 2.15

2.7.6. Категории. В общем случае выделяют следующие типы зависимых сущностей:

характеристическая сущность – дочерняя сущность связана только с одной родительской (хранит информацию о характеристиках родительской сущности (см. рис. 2.7));

ассоциативная сущность – связана с несколькими родительскими сущностями (содержит информацию о связях (см. рис. 2.12, 2.16));

категориальная сущность (иерархия наследования) – сущность с общими атрибутами, которая объединяет несколько разных сущностей, и которые разделяют эти общие атрибуты (рис. 2.17). Здесь общие атрибуты сущности STATIONS наследуются сущностями PARAGRAF и STATION_FLAG. Атрибут STATION_TYPE показывает, как значение одной категориальной сущности отличается от другой.

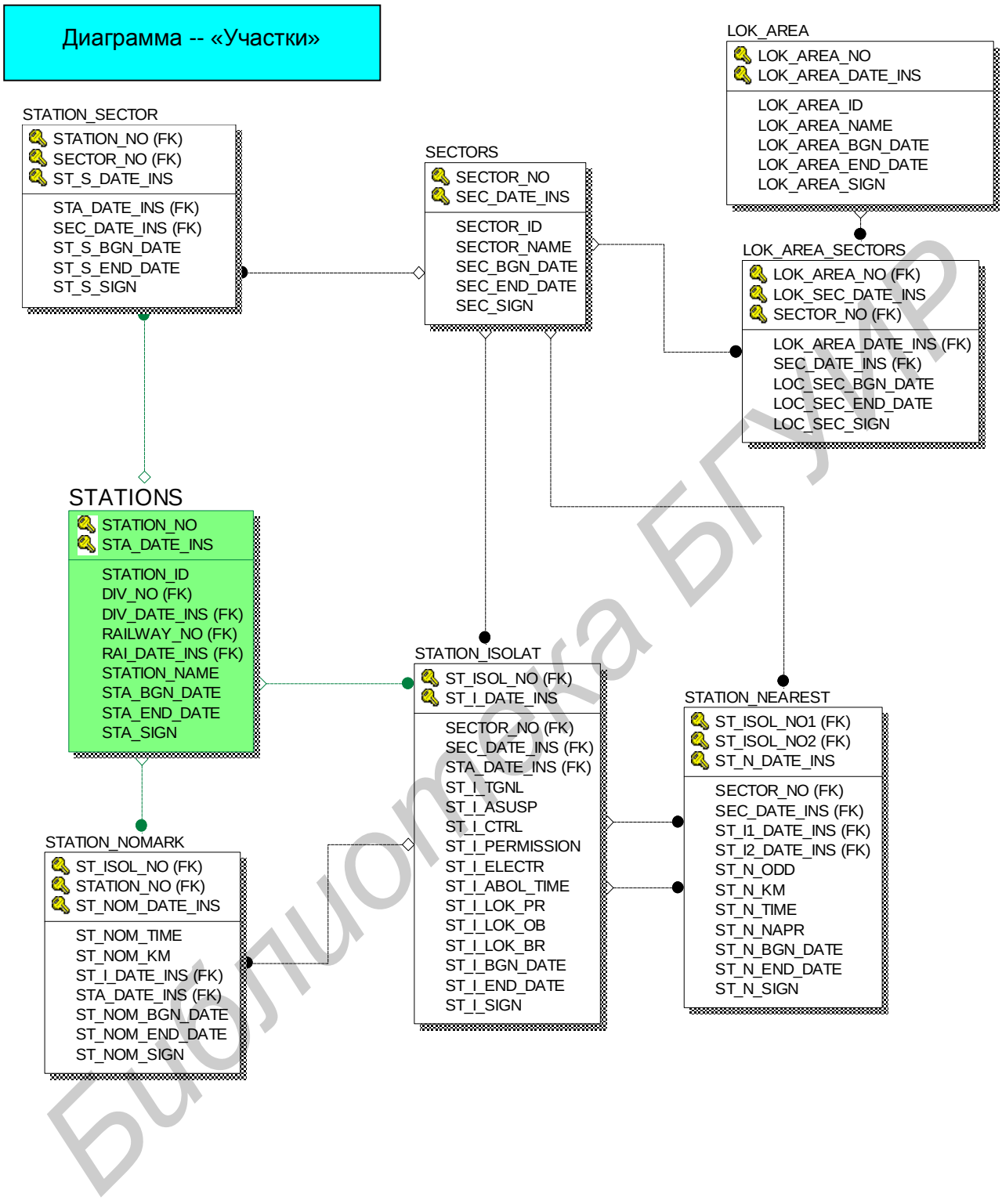

Рис. 2.16

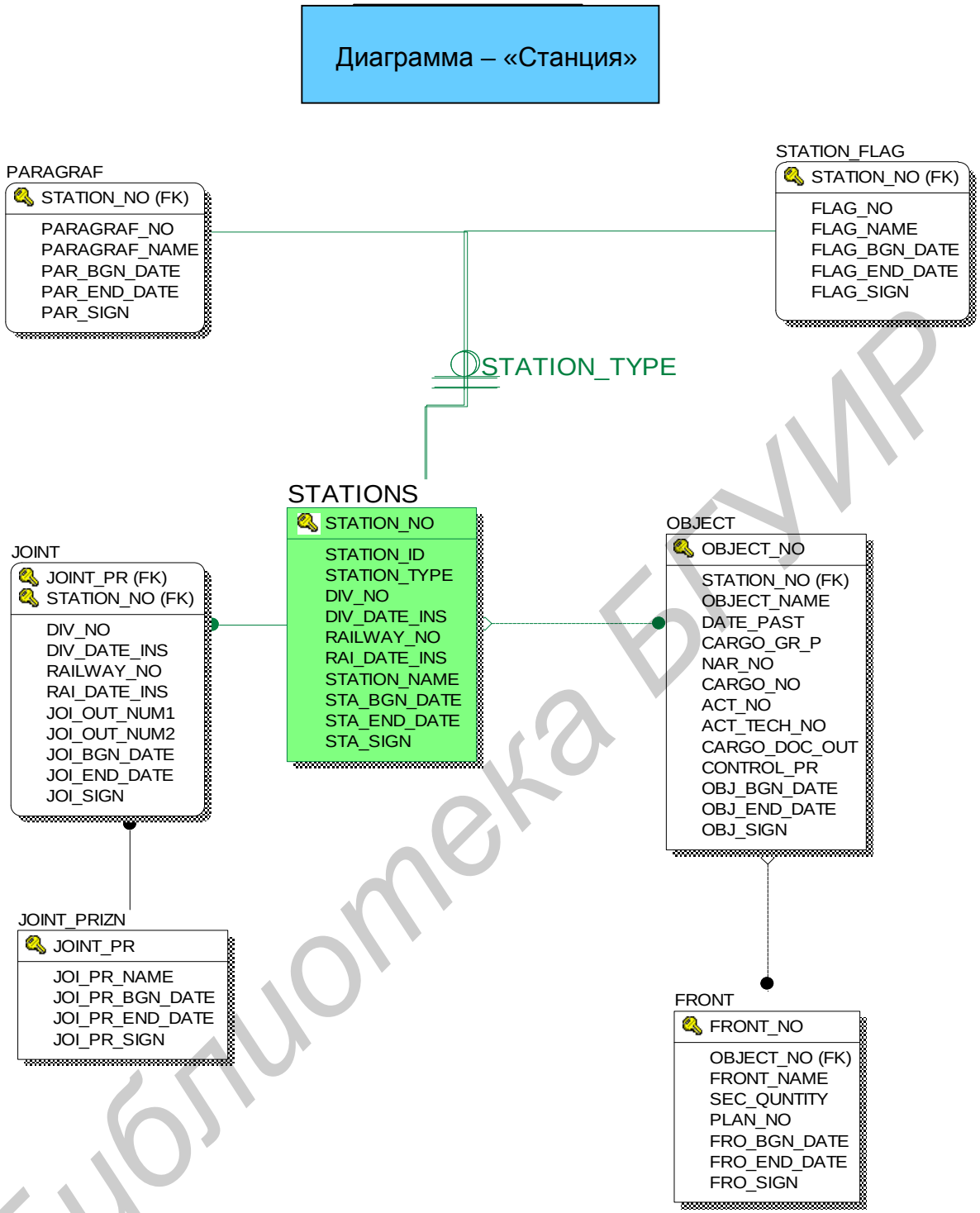

Рис. 2.17

2.7.7. Рекурсия – это такой вид отношений (связей), когда одна и та же сущность является и родительской, и дочерней. При реализации таких связей PK мигрирует в качестве FK в состав неключевых атрибутов, один и тот же атрибут не может появиться дважды под одним и тем же именем. Поэтому

нужно или переименовывать атрибуты FK (см. рис. 2.9), или использовать имена ролей связей при именовании таких FK (рис. 2.18).

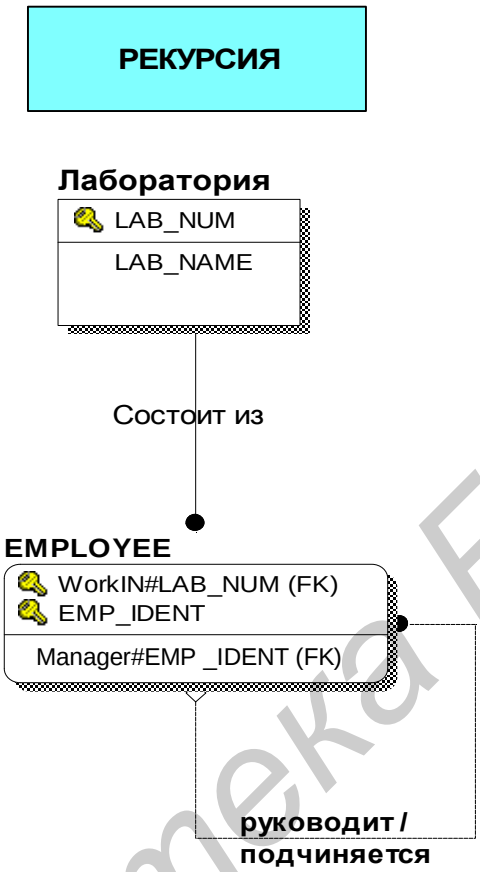

Рис. 2.18

Возможны и другие виды рекурсии: неявные (косвенные), сетевые. Есть сущности, которые находятся сами с собою в связи «многие – ко многим». Примером такой сущности может быть сущность «родственник», когда значения такой сущности связаны с другими значениями этой же сущности. В общем случае, можно говорить о сетевой рекурсии внутри самой сущности. Разрешить такие связи можно с помощью введения новой сущности «отношения родственников», в которой будут заданы пары ключевых атрибутов и атрибут «Тип отношения» «мать–сын», «дед–внук», «тесть–зять» и т.д. Данная сущность будет дважды связана с сущностью «родственник», и PK сущности «родственник» дважды мигрирует в новую сущность «отношения родственников» как FK. Но как и в предыдущем случае, правила именования атрибутов должны быть соблюдены: или должны быть введены разные ролевые отношения, или имена атрибутов должны быть изменены.

2.8. Домены

Стандарт IDEF1x определяет домен как поименованный и определенный набор значений, такой что один или более атрибутов получают значения из этого домена.

Домен, как и класс, фиксирует и определяет допустимое множество значений. Например, домен «Код–страны» (рис. 2.19) определяет код страны, который может состоять из двух специальных букв. Домен – это неизменный класс, значения которого не изменяются во времени. В противоположность этому значения сущности меняются во времени. Каждое значение домена уникально в этом домене.

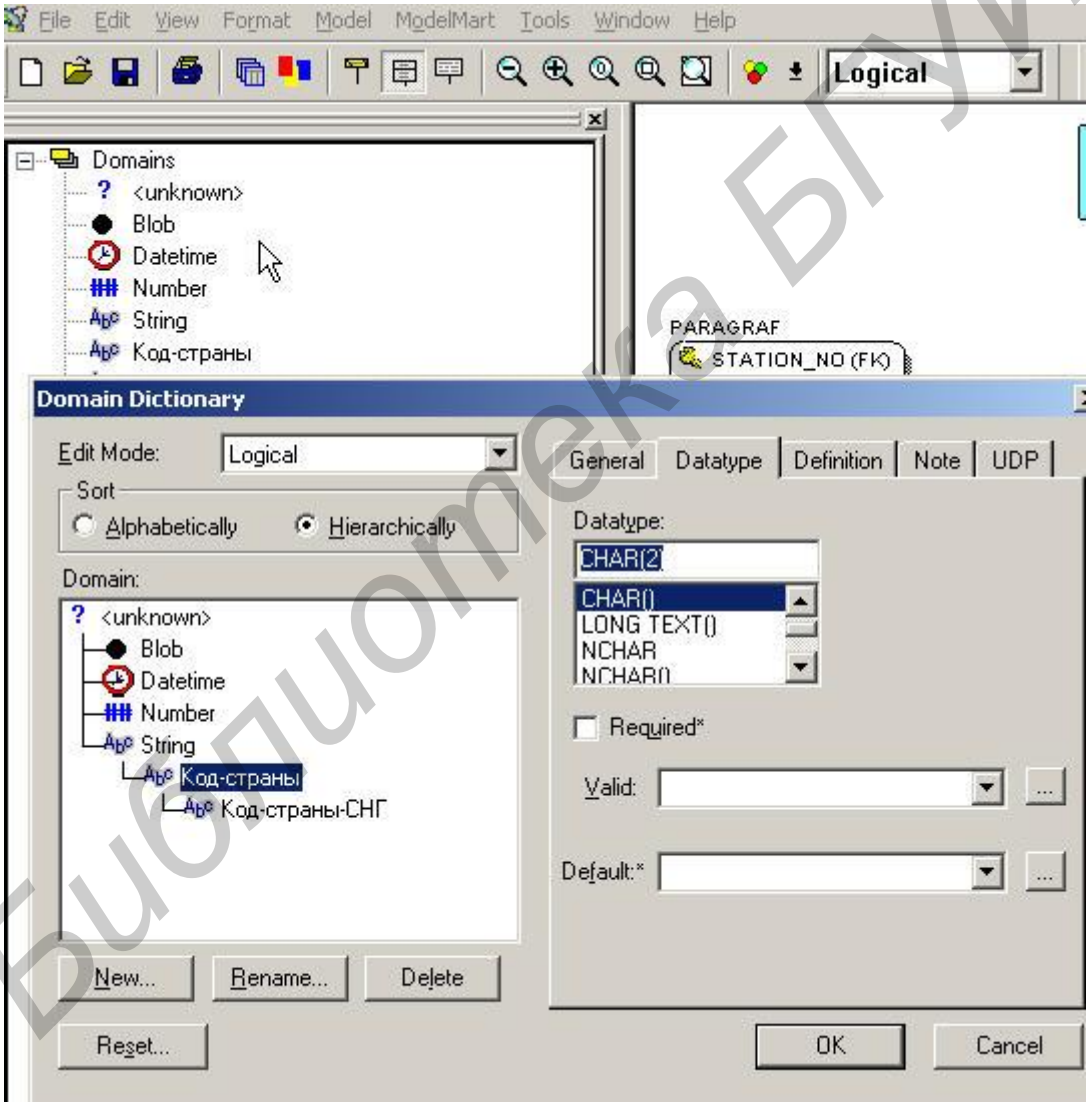

Рис. 2.19

Существуют два типа доменов: базовые домены (Character, Numeric, Boolean в IDEF1x, а в Erwin: String, Number, Datetime, Blob) и типизированные домены (производные), например «Код–страны–СНГ» (рис. 2.20). Все базовые домены имеют правила использования значений данных из домена. Наиболее используемые – это два правила: список значений и диапазоны. Для каждого домена могут быть установлены свои правила определения значений.

Типизированные домены – это подтипы базовых типов или других типизированных доменов.

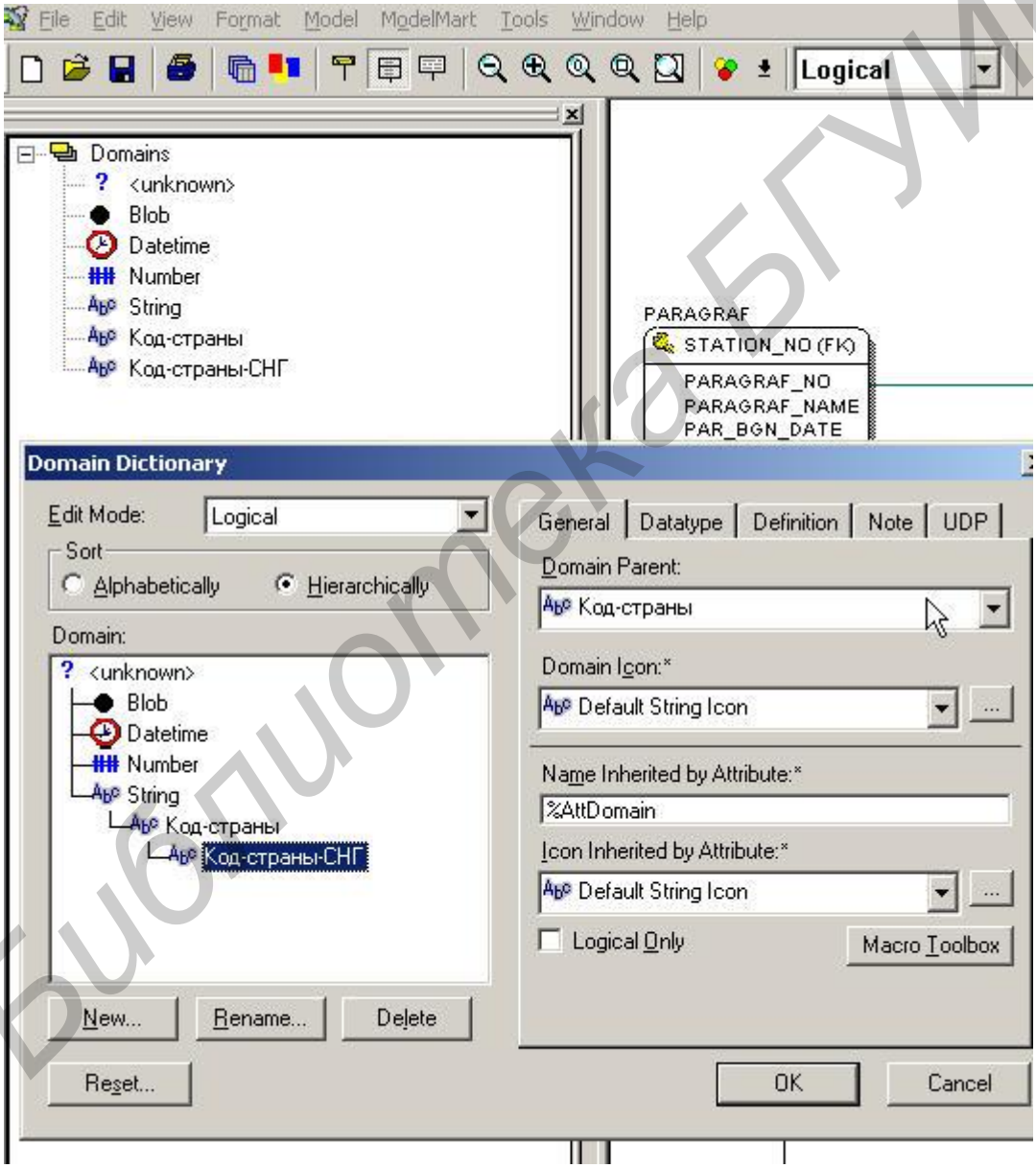

Рис. 2.20

На логическом уровне домены можно описать без конкретных физических свойств, а уже на физическом уровне они получают конкретные специфические свойства. Например, домен «Код–страны–СНГ» (рис. 2.21) на логическом уровне может иметь тип String, а на физическом уровне (DB2) будет присвоен тип Char(2) и указано конкретное множество значений домена и правила валидации. Для другой БД физический уровень значений может быть задан свой.

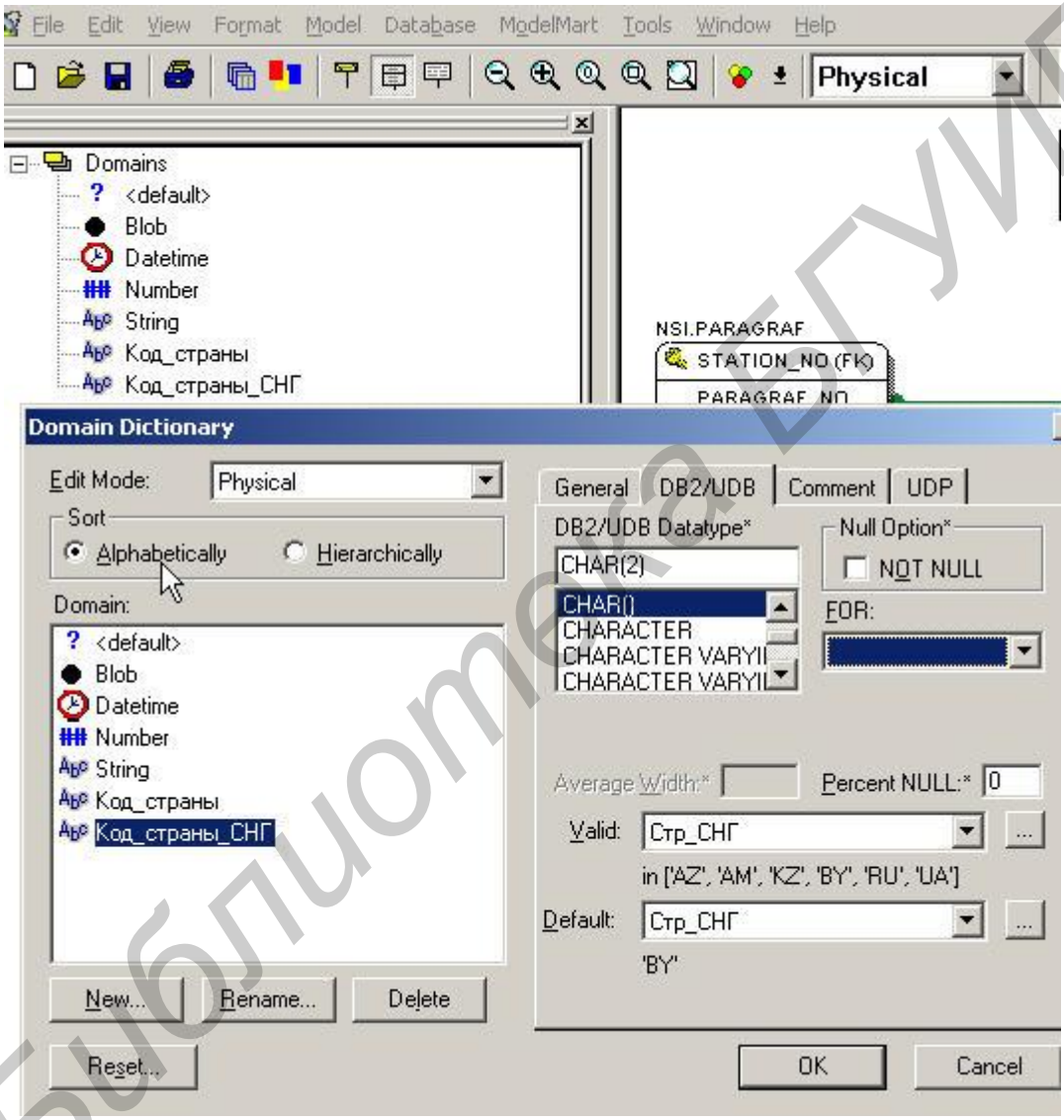

Рис. 2.21

Теперь при определении атрибута «COUNTRY_ID» сущности «COUNTRIES» можно указать его значение из домена «Код-страны-СНГ» (рис. 2.22).

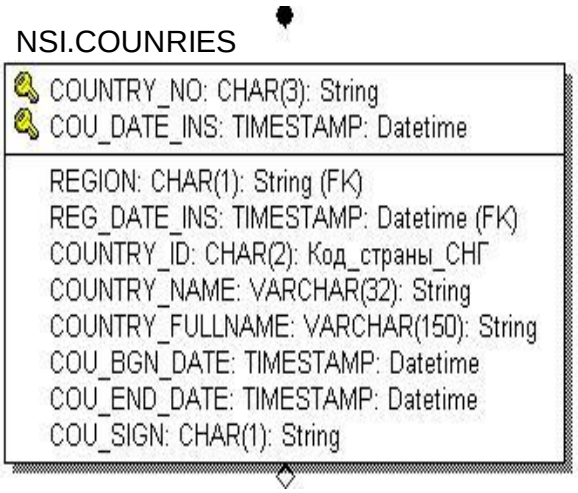

Рис. 2.22

2.9. Представления

Представления (View) являются объектами СУБД, данные в которых формируются динамически при обращении к представлению, т.е. данные не хранятся постоянно, как в таблицах БД. Представление определяется в терминах тех таблиц БД и их атрибутов, которые уже есть в БД. Для создания представления может использоваться одна или более реальных таблиц БД.

Использование представлений (View) – очень удобный механизм:

– для обеспечения безопасности и секретности доступа к данным в БД для каждого пользователя (разработчика) может быть разработано и предоставлено свое представление (видение) данных;

– для разработки интерфейса взаимодействия между подсистемами или системами;

– для разработки пользовательского интерфейса web-сайта;

– для замены сложной системы репликаций данных в другие системы, в которых модели данных существенно отличаются от модели данных исходной системы.

Разработка представлений разрабатывается на физическом уровне и AllFusion Erwin Data Modeler содержит специальные средства для создания таблиц представления и операторов SQL для заполнения их данными.

Так, на рис. 2.23 приведена диаграмма представления для таблиц БД. Эти таблицы содержат внутренние суррогатные PK-ключи идентификации данных xxx#UN, которые для пользователя БД ни о чем не говорят. Пользователю БД необходимы реальные общепринятые данные: код дороги, код отделения дороги, код станции на дороге – поэтому для отображения реальных данных используются представления. Таблицы представлений (прямоугольники с закругленными углами) связаны отношениями с таблицами БД

(прямоугольники) и содержат те атрибуты, которые будут заполнены из соответствующих атрибутов таблиц БД.

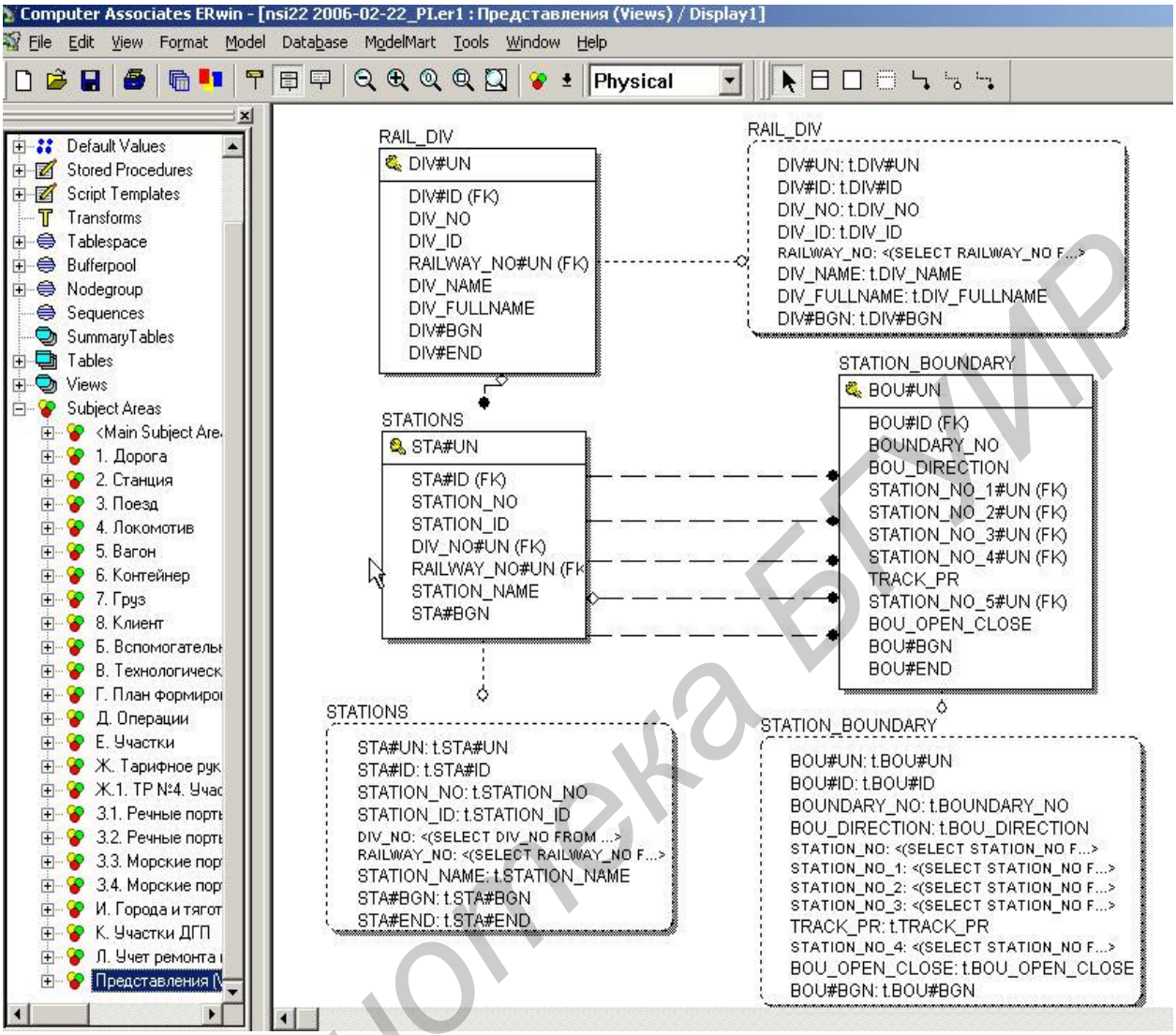

Рис. 2.23

На рис. 2.24 и 2.25 приведен диалог Views для определения представления и генерации оператора Create для его создания.

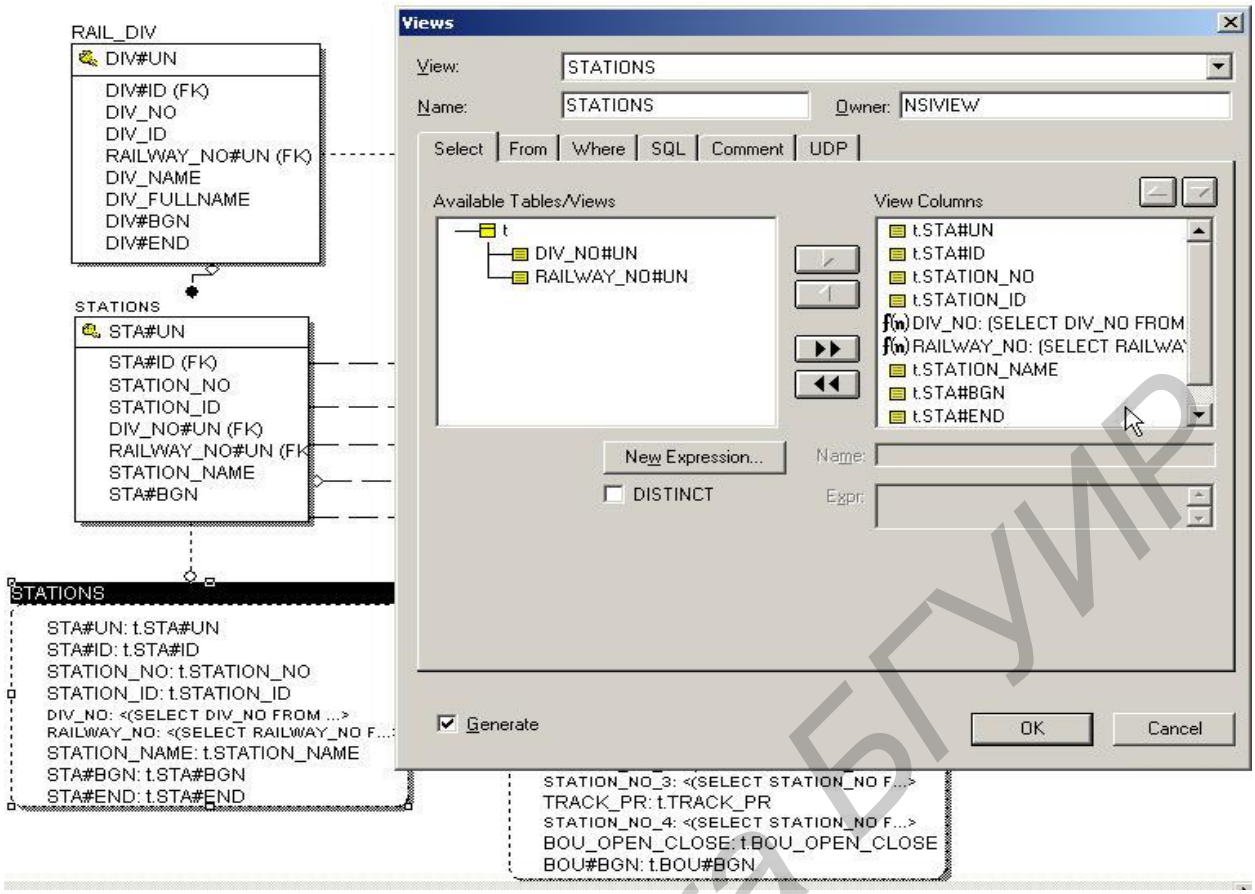

Рис. 2.24

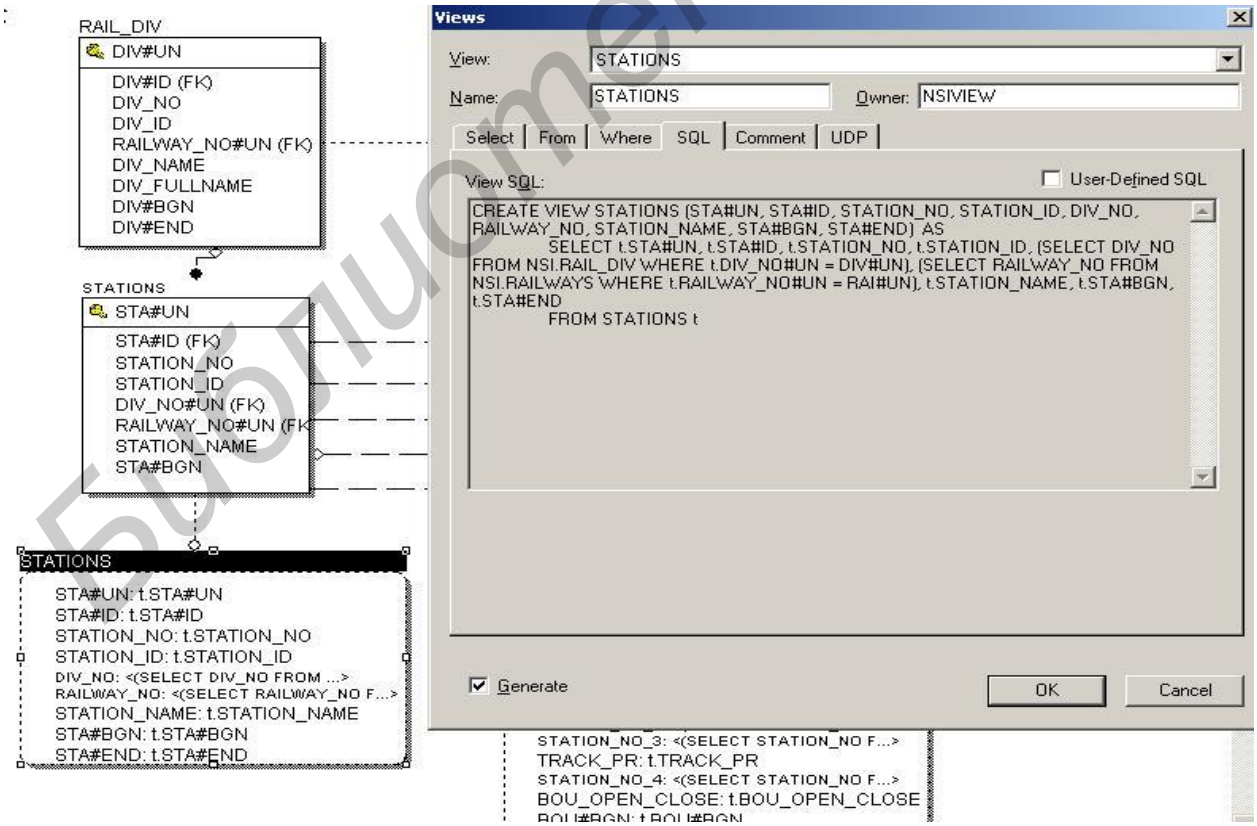

Рис. 2.25

На рис. 2.26 приведен диалог создания оператора Select для заполнения представления.

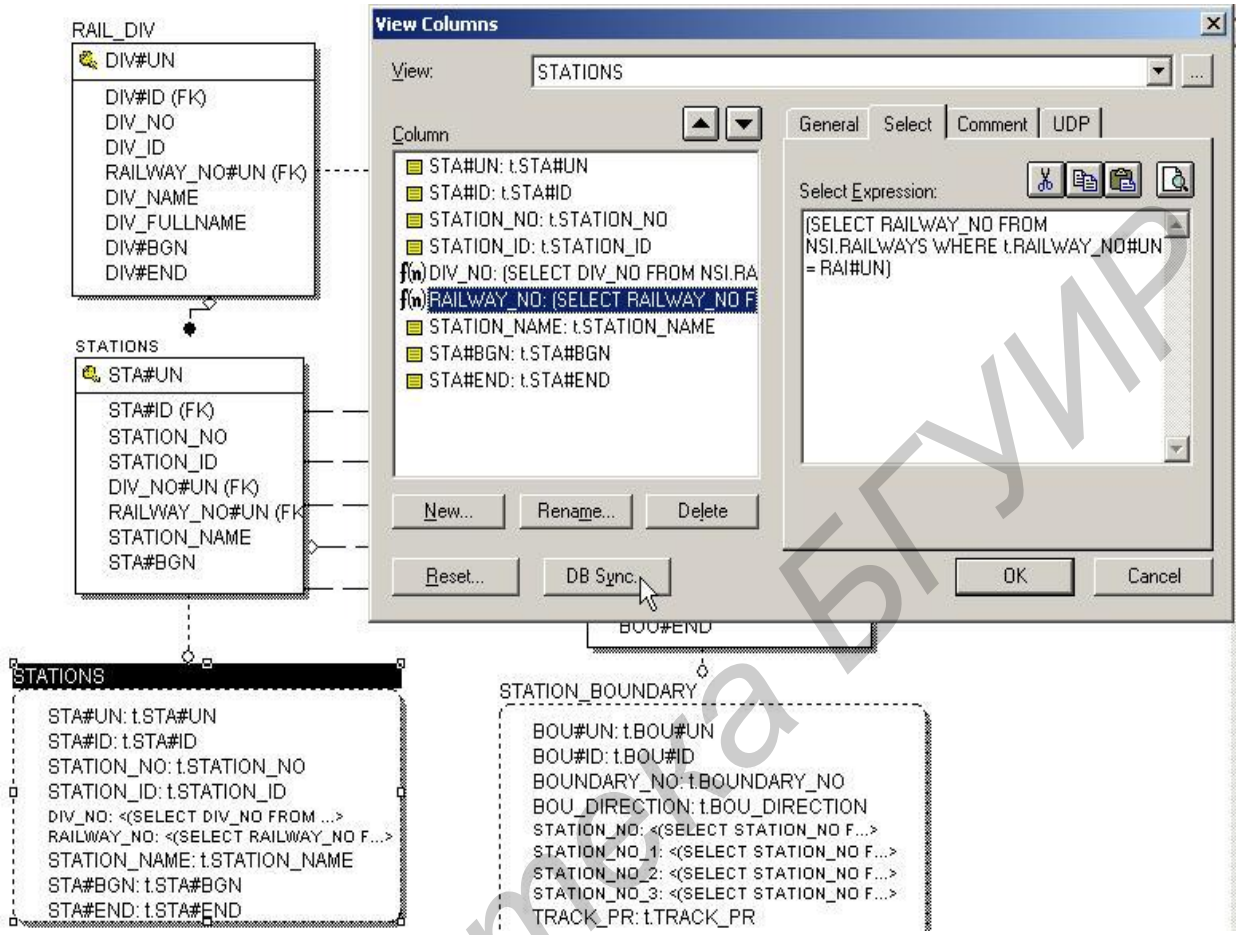

Рис. 2.26

На рис. 2.27 приведена информация, выдаваемая пользователю прямо из таблицы БД, которая содержит внутренние ключи (PK и FK). На рис. 2.28 приведена информация, выдаваемая пользователю через разработанное представление (см. рис. 2.23–2.26).

Таблица: STATIONS

Уникальный ключ записи **STA#UN**	Идентификатор объекта **STA#ID**	Код станции (ЕСР) **STATION_NO**	Мнемокод станции **STATION_ID**	Ссылка на таблицу RAIL_DIV (Код отделения) **DIV_NO#UN**	Ссылка на таблицу RAILWAYS (Код дороги) **RAILWAY_NO#UN**	Наименование станции **STATION_NAME**
192701	1927	140009	МИНС	30501	27001	МИНСК-СОРТ
192801	1928	140102	МИНСВ	30501	27001	МИНСК-СЕВ
192901	1929	140206	МИНП	30501	27001	МИНСК-ПАСС
193001	1930	140304	МИНВ	30501	27001	МИНСК-ВОСТ
193201	1932	140507	МИНЮ	30501	27001	МИНСК-ЮЖНЫЙ

Рис. 2.27

Таблица: STATIONS

Уникальный ключ записи **STA#UN**	Идентификатор объекта **STA#ID**	Код станции (ЕСР) **STATION_NO**	Мнемокод станции **STATION_ID**	Ссылка на таблицу RAIL_DIV (Код отделения) **DIV_NO**	Ссылка на таблицу RAILWAYS (Код дороги) **RAILWAY_NO**	Наименование станции **STATION_NAME**
192700	1927	140009	МИНС	01	13	МИНСК-СОРТ
192800	1928	140102	МИНСВ	01	13	МИНСК-СЕВ
192900	1929	140206	МИНП	01	13	МИНСК-ПАСС
193000	1930	140304	МИНВ	01	13	МИНСК-ВОСТ
193100	1931	140403	СТЕП	01	13	СТЕПЯНКА
193200	1932	140507	МИНЮ	01	13	МИНСК-ЮЖНЫЙ
193300	1933	140600	РЗВ	01	13	РЕЗЕРВНАЯ СТАНЦИЯ
193400	1934	140704	ОЗЕР	01	13	ОЗЕРИЩЕ

Рис. 2.28

3. Проектирование баз данных, логический уровень

3.1. Информационные системы, базы данных и модели

Моделирование данных – это процесс описания информационных структур данных и установления связей и отношений между ними с целью определения информационных потребностей системы.

Модель данных, в наиболее общем понятии, представляет собой баланс между определенными потребностями специфического проекта приложения и общими информационными требованиями предметной области. Модель данных включает все объекты, их атрибуты и отношения, требуемые данным отдельным проектом. Полная модель включает требования томов для объектов, пути доступа и порядок доступа, ожидаемый принцип доступа к структурам данных.

Основные уровни проектирования описаны в стандарте моделирования IDEF1x [2]. Использование этих уровней полезно и эффективно при разработке любой автоматизированной системы, причем для конкретного применения их использование может отличаться путем расширения одних или объединения других.

Для программной поддержки данного стандарта проектирования БД используется CASE-средство AllFusion Erwin Data Modeler.

Использование уровней моделирования позволяет реализовать хорошо структурированный нисходящий сверху вниз подход к разработке модели БД, при котором успех детализации объектов на нижестоящем уровне создается на каждом этапе проектирования.

На верхних уровнях моделирования информационная модель включает общие сведения об объектах, обеспечивая тем самым ее технологическую независимость от конкретной платформы СУБД. Она называется логической моделью. На более низких уровнях проектирования модель дополняется детализированными сведениями об объектах. Это связано с тем, что физическая организация данных, например в СУБД DB2, существенно отличается от физической организации этих данных в СУБД Oracle. Такая детализированная модель называется физической моделью.

AllFusion Erwin Data Modeler обеспечивает два вышеуказанных уровня представления модели – логический и физический. Логический уровень включает наиболее общие сведения о предметной области. Объекты модели логического уровня называются сущностями и атрибутами. Логический уровень модели данных может быть построен на основе диаграмм модели бизнес-процессов предметной области. Физический уровень позволяет описать всю детальную информацию о конкретных физических объектах – таблицах, столбцах, связях между объектами, индексах, процедурах и др. При этом AllFusion Erwin Data Modeler позволяет создавать модели трех типов: модель, имеющую логический уровень представления данных, модель, имеющую

физический уровень представления данных, и модель, имеющую как логический, так физический уровень.

Такое многоуровневое моделирование БД имеет ряд достоинств. Наиболее очевидное достоинство – это систематическое документирование, которое может использоваться постоянно для развития базы данных и прикладных применений, чтобы определить системные требования и связать их непосредственно с требованиями конечного пользователя. Второе достоинство – это обеспечение ясной картины ссылочных ограничений целостности БД. Поддержание ссылочной целостности существенно в реляционной модели, где отношения не кодируются явно. Третье достоинство – это независимая картина базы данных, которая следует из «логической» модели БД. Логическая модель БД может использоваться автоматизированными средствами для генерации различных физических СУБД. Таким образом, можно использовать одну и ту же логическую модель на языке IDEF1x в Erwin для получения из нее схемы таблиц DB2, так же, как и схемы таблиц для других реляционных СУБД.

Создание модели данных, как правило, начинается с создания логического уровня. После описания логического уровня данных разработчик приступает (с учетом выбранной СУБД) к проектированию модели физического уровня.

На рис. 3.1 и 3.2 приведены объекты логической и физической модели данных.

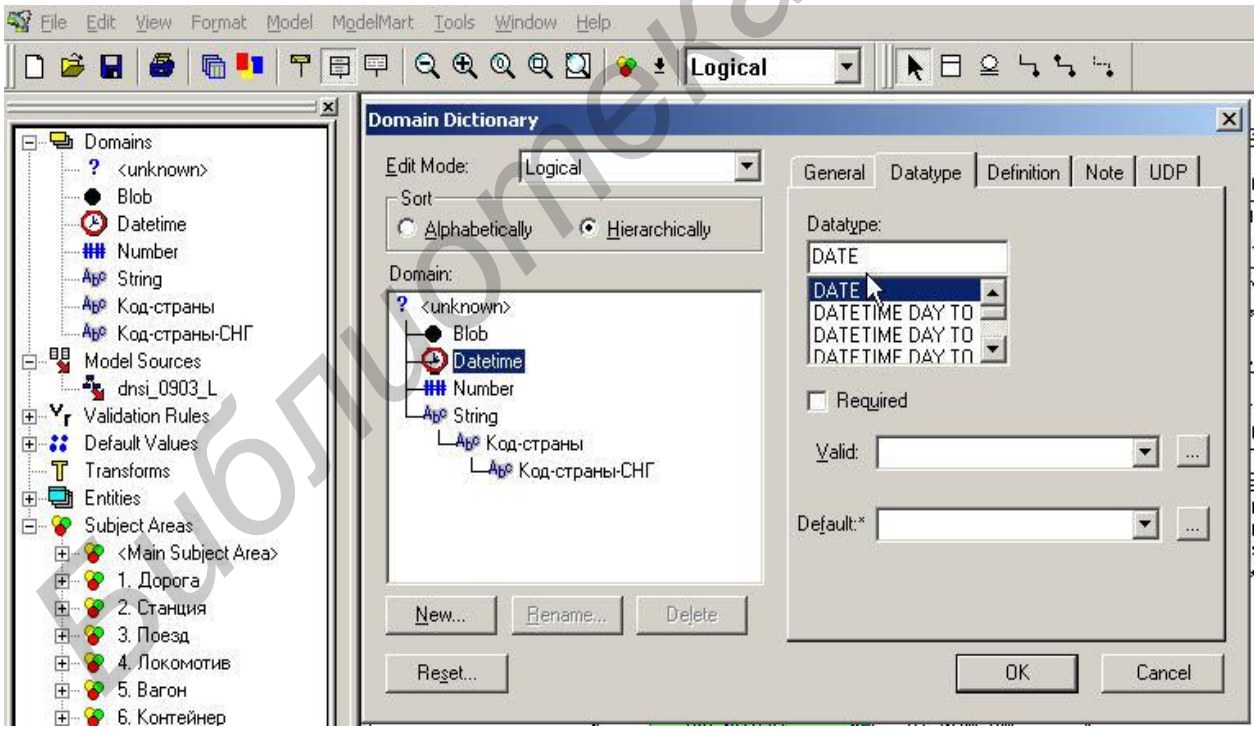

Рис. 3.1

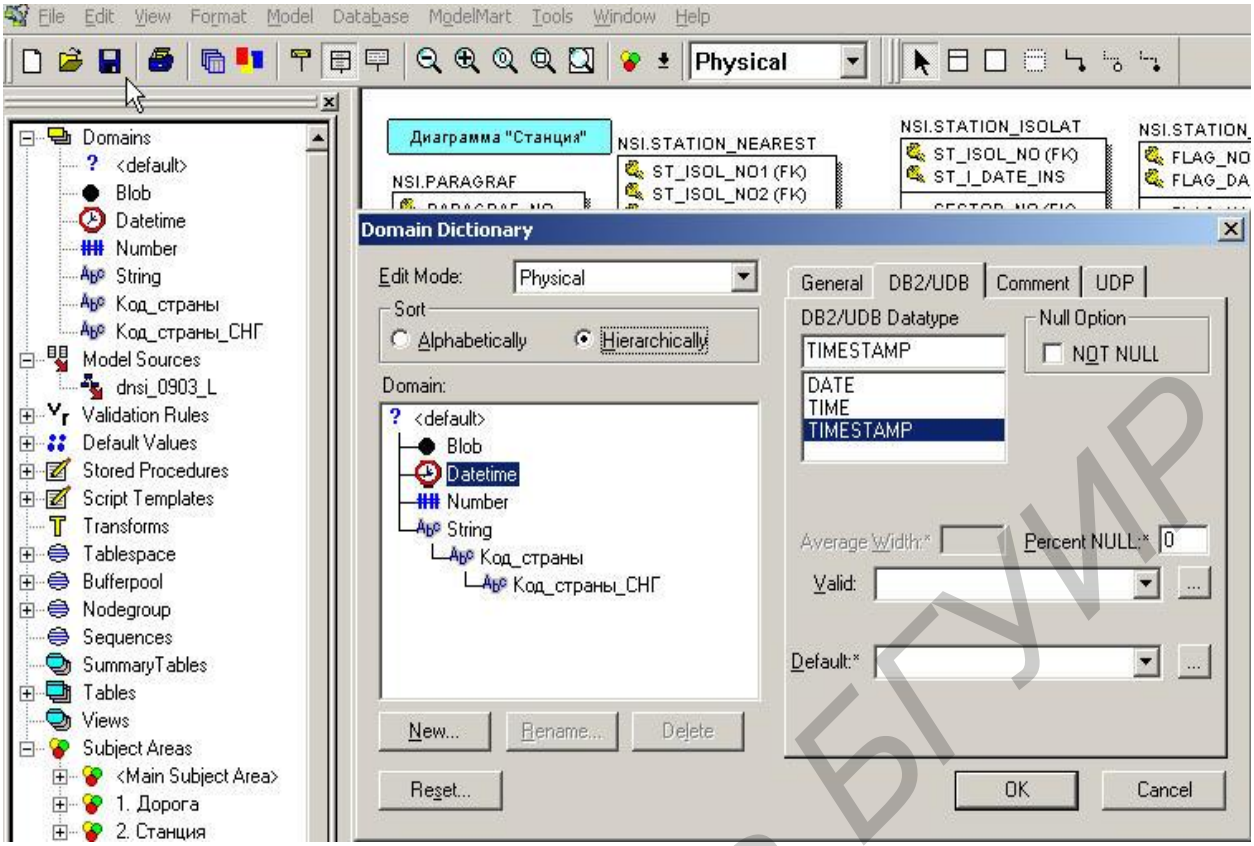

Рис. 3.2

В модели, включающей оба уровня, при проектировании логического уровня Erwin автоматически первоначально создает соответствующие элементы физической модели **(но не наоборот!)**. Это означает, что некоторому элементу логического уровня соответствует определенный элемент физического уровня. Соответствие терминов логического и физического уровней приведено ниже (табл. 3.1). Например, имени сущности (Entity) соответствует имя таблицы (Table), имени атрибута (Attribute) – имя столбца (Column), имени ключевой группы (Key group) – имя индекса (Index). Поэтому при проектировании, а особенно при доработке модели, имеющей логический и физический уровень одновременно, необходимо для их синхронизации (идентичности) определять объекты и их термины на логическом уровне, а затем доопределять элементы соответствующих терминов на физическом уровне. Это позволит избежать рассинхронизации моделей, учитывать приоритетность значений соответствующих терминов из логического уровня перед физическим уровнем модели.

Ниже в таблице приводится перечень общих терминов логической и физической моделей, используемых в дальнейшем при проектировании и сопровождении БД.

Таблица 3.1

Общие термины логической и физической моделей

Логическая модель	***Физическая модель***
Сущность /объект/ (Entity)	Таблица (Table)
Зависимая сущность	Внешний ключ FK является частью PK дочерней таблицы
Независимая сущность	Родительская или дочерняя таблица, у которой FK не является частью ее PK
Атрибут (Attribute)	Столбец (Column)
Логический тип данных (текст, число, дата)	Физический тип данных (зависит от выбранного сервера назначения)
Домен (логический)	Домен (физический)
Первичный ключ (Primary key)	Первичный ключ, PK-индекс
Внешний ключ (Foreign key)	Внешний ключ, FK-индекс
Альтернативный ключ (AK)	AK индекс – уникальный, но не первичный ключ
Инверсионный вход (IE)	IE индекс – не уникальный индекс, созданный для поиска информации в таблице по не уникальному значению
Ключевая группа (Key group)	Индекс (Index)
Бизнес-правило (Business rule)	Триггер или хранимая процедура
Правило проверки данных (Validation rule)	Принудительный контроль данных (Constraint)
Отношение (Relationship)	Отношение, реализованное с использованием FKs
Идентифицированные (определенные) отношения (Identifying)	FK как часть PK дочерней таблицы (выше линии)
Неидентифицированные (неопределенные) отношения (Non-Identifying)	FK не является частью PK дочерней таблицы (ниже линии)
Отношение подтипа (Subtype)	Денормализованные таблицы (Denormalized tables)
Отношение «многие – ко многим» (Many-to-many)	Ассоциативная таблица (Associative table)
Ссылочная (относительная) целостность (cascade, restrict, set null, set default)	Триггеры для операций INSERT, UPDATE и DELETE
Мощность, количество элементов (Cardinality)	Триггеры для операций INSERT, UPDATE и DELETE

3.2. Построение логической модели

Существует три вида логических моделей, которые используются, чтобы охватить информационные требования предметной области: «**Entity Relationship Diagram» (ERD)** – диаграмма «сущность–связь», «**Key–Based Model» (KBM)** – «Модель на основе ключа» и «**Fully Attributed model» (FAM) – «**Полностью атрибутная модель». ERD- и KBM-модели также называются «моделями данных области», потому что они часто охватывают широкую предметную область, которая является, как правило, большей чем бизнес-правила, направленные на единственный отдельный проект автоматизации. ERD-модель используется на первоначальном этапе проектирования для отображения основных сущностей и возможных их связей для обсуждения с экспертами предметной области структур данных. Модель данных, основанная на ключах (KBM), включает описание всех сущностей и ключей, которые соответствуют предметной области. Она дает более подробное описание предметной области. Полностью атрибутная модель (FAM) дает наиболее полное представление о структурах данных. Она состоит из полностью определенных сущностей, их атрибутов и связей и должна представлять данные в третьей нормальной форме.

3.2.1. Диаграмма «сущность–связь»

Первый шаг в построении логической модели – конструирование **диаграммы «сущность–связь»** (ERD) как самого высокого уровня моделирования данных рассматриваемой предметной области. ER-диаграмма показывает основные сущности и связи, которые определяют предметную область в целом. Диаграмма связей (отношений) сущностей использует два главных строительных элемента: сущности и связи (отношения) и, возможно, некоторые атрибуты. Эти термины, проведя аналогию диаграммы ERD с разговорным языком, можно упрощенно интерпретировать следующим образом: сущности – это существительные, атрибуты – это прилагательные или характеристики, связи – это глаголы. Значениями сущностей являются экземпляры подобных объектов реального мира, а атрибуты – их характеристики. Построение модели данных в Erwin – просто вопрос обнаружения правильного собрания существительных, глаголов и прилагательных и размещения их всех вместе.

Цель ERD-моделирования состоит в том, чтобы обеспечить наиболее общее представление о требованиях предметной области, достаточных для планирования дальнейшей разработки информационной системы. Эти модели не очень детализированы, включают только главные сущности, нет многих характеристик объектов (атрибутов), если таковые вообще имеются. Могут присутствовать отношения «многие – ко многим» и неопределенные отношения, ключи обычно вообще отсутствуют. Это прежде всего

первоначальное представление о бизнес-правилах или первоначальная модель для обсуждения.

ERD-модель может быть разделена на подсхемы (subject areas), которые используются для определения части модели, удовлетворяющей интересам отдельного приложения или даже отдельных функций. Использование подсхем позволяет разделить большие модели на меньшие, более управляемые подмножества сущностей, что может быть легче для определения и дальнейшей поддержки (рис. 3.3).

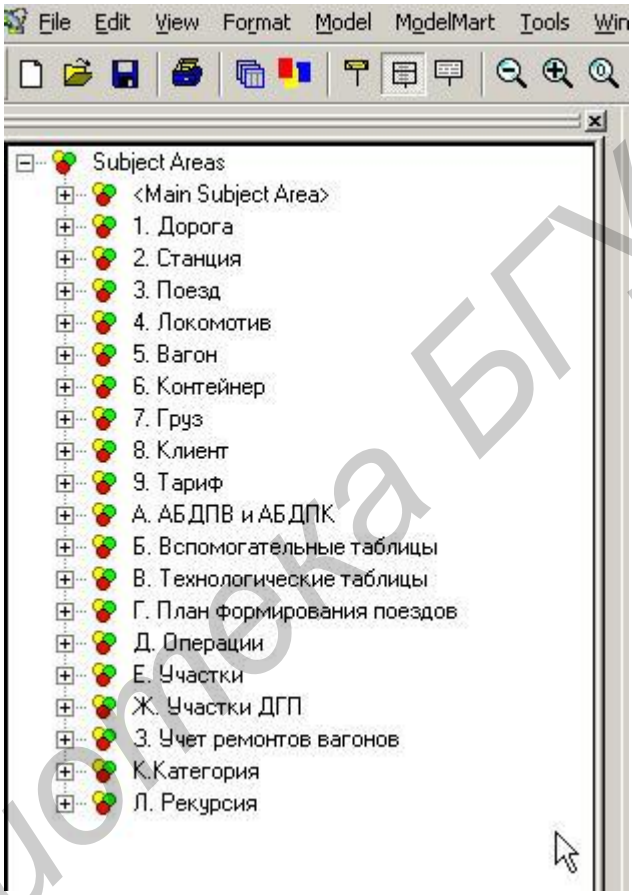

Рис. 3.3

Наиболее фундаментальный компонент реляционной базы данных – таблица. Таблицы используются для организации и хранения информации. Таблица организована в виде множества строк данных одинаковой структуры. Каждая строка, в свою очередь, содержит определенный набор фактов, называемых столбцами таблицы (атрибутами). В реляционной базе данных все значения в столбце данных должны быть элементарными (atomic), что означает, что каждый столбец в таблице может содержать сведения только об одном-единственном факте. Естественно, между таблицами в реляционной базе данных существуют связи (отношения). Каждое отношение представляется в

БД путем совместного использования (разделения) одного или более столбцов в двух связанных таблицах.

Подобно таблицам и столбцам, которые составляют физическую модель реляционной базы данных, диаграмма «сущность – связь» (и все другие логические модели данных) включает эквивалентные компоненты, которые позволяют моделировать структуры данных предметной области, так же, как и система управления базами данных. Так, логический эквивалент таблицы – сущность, а логический эквивалент столбца таблицы – атрибут объекта. Перечень соответствий между эквивалентными понятиями логической и физической моделей приведен в таблице «Общие термины логической и физической моделей» (см. табл. 3.1). В диаграмме ERD сущность (объект) представляется в виде прямоугольника, который содержит название (имя) объекта. Названия сущностей (объектов) всегда желательно употреблять в единственном числе, что облегчает «чтение» диаграммы в дальнейшем. Связь между двумя объектами на диаграмме представляется в виде линии, которая указывает, что данные одной таблицы находятся в некоторой связи (отношении) с данными из другой таблицы. В этой связи всегда присутствует пара таблиц: родительская (Parent) и детская (Child) (рис. 3.4).

Рис. 3.4

На рис. 3.5 приведена ERD-модель, на которой приведены имена сущностей, их определения и отношения между ними.

3.2.2. Модель данных на основе ключа

Модель данных на основе ключа – это модель данных, которая полностью описывает все основные структуры данных, удовлетворяющих требованиям предметной области. Цель модели KBM – включение всех сущностей и атрибутов, которые представляют интерес для бизнеса. Модель на основе

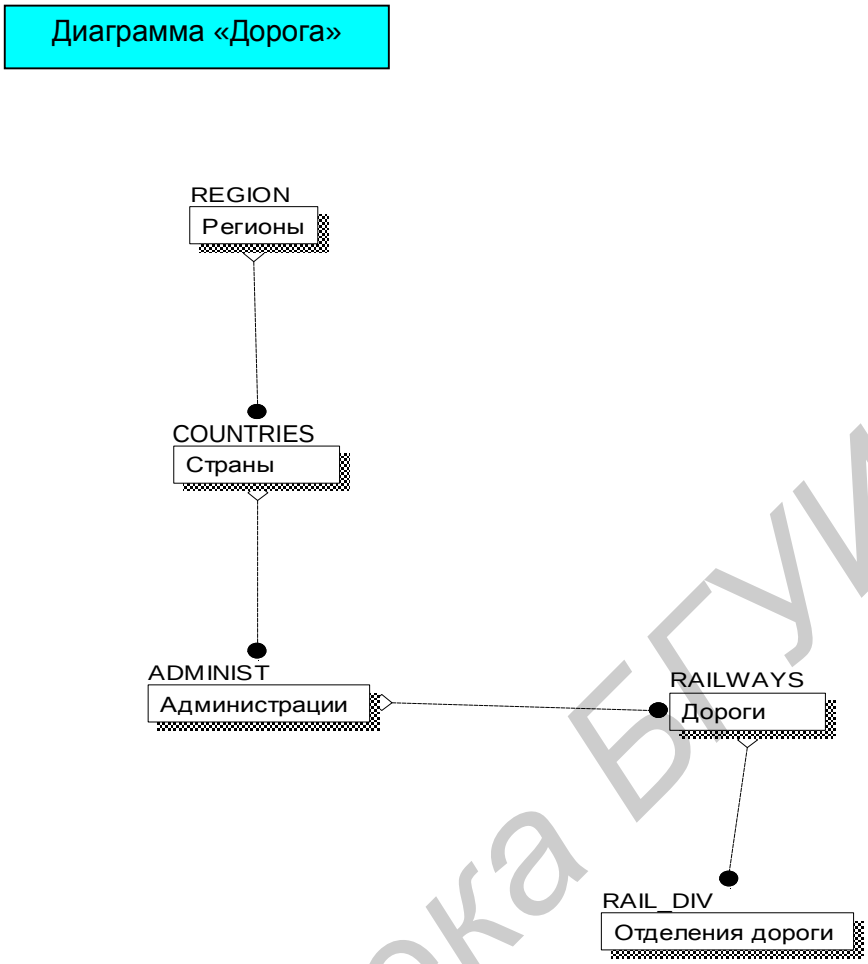

Рис. 3.5

ключа, как и следует из ее названия, в основном содержит в себе ключи. В логической модели ключ однозначно идентифицирует экземпляр данных, т.е. строку внутри сущности (таблицы). В дальнейшем, когда соответствующая физическая модель будет использоваться, применение ключа обеспечит быстрый доступ к нужным данным.

В целом, модель **KBM** охватывает те же самые возможности, что и ERD, но раскрывает их более подробно и детально, насколько это возможно. В порядке развития ERD к правильной логической модели данных необходимо определить ключи, чтобы однозначно идентифицировать каждый элемент в сущности (объекте), т.е. его значение.

В каждом объекте в диаграмме модели данных горизонтальная линия разделяет атрибуты на две группы: ключевую область (выше линии) и неключевую область, или область данных (ниже линии).

Ключевая область содержит первичный ключ (**Primary key PK**) для сущности. Первичный ключ (PK) – это набор атрибутов, которые используются для однозначной идентификации экземпляров сущности. Первичный ключ может включать один или более первичных ключевых атрибутов. Количество их должно быть достаточным, чтобы выбранные атрибуты сформировали

уникальный идентификатор для каждого элемента в сущности. Сущность кроме ключевых атрибутов, как правило, имеет неключевые атрибуты.

Рассмотрение атрибутов – кандидатов для формирования первичного ключа и его выбор (а он может быть только один для таблицы) требуют знания данных предметной области. Существуют строгие правила, использующиеся при выборе первичного ключа. Атрибут (атрибуты) первичного ключа должен:

1) уникально идентифицировать значение сущности;
2) не содержать значение NULL;
3) не изменяться через промежуток времени. Всякий элемент сущности берет свою идентичность от ключа. Если ключ поменяется, то это уже другой элемент;
4) быть как можно более коротким, что облегчать индексацию и восстановление данных. Если нужно использовать ключ, который является комбинацией ключей из других сущностей, то каждая часть ключа должна придерживаться этих правил.

Замечание. Первичный ключ, выбранный для сущности при проектировании логической модели, как исключение, может и не быть первичным ключом в физической модели для более эффективного доступа к таблице. В этом случае ключ может быть изменен, чтобы удовлетворить потребности и требования физической модели базы данных в некотором исключительном случае.

Ниже на рис. 3.6 и 3.7 приведены **KBM**-диаграммы некоторых моделей на основе PK-ключей. В данных моделях присутствуют только имена атрибутов, которые могут быть использованы для идентификации значений сущности.

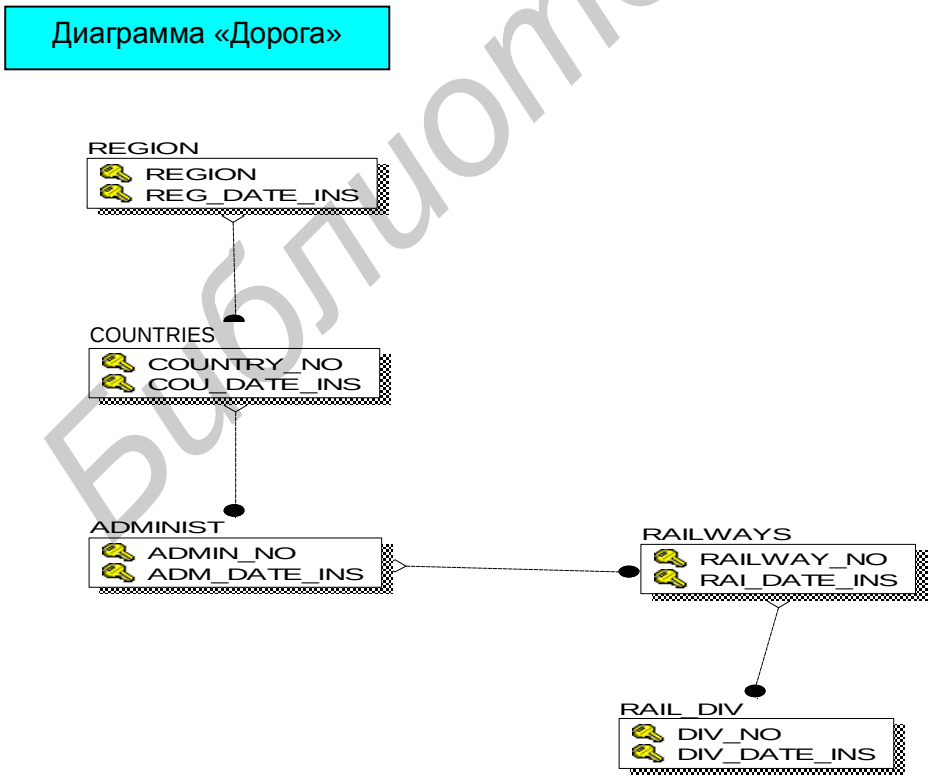

Рис. 3.6

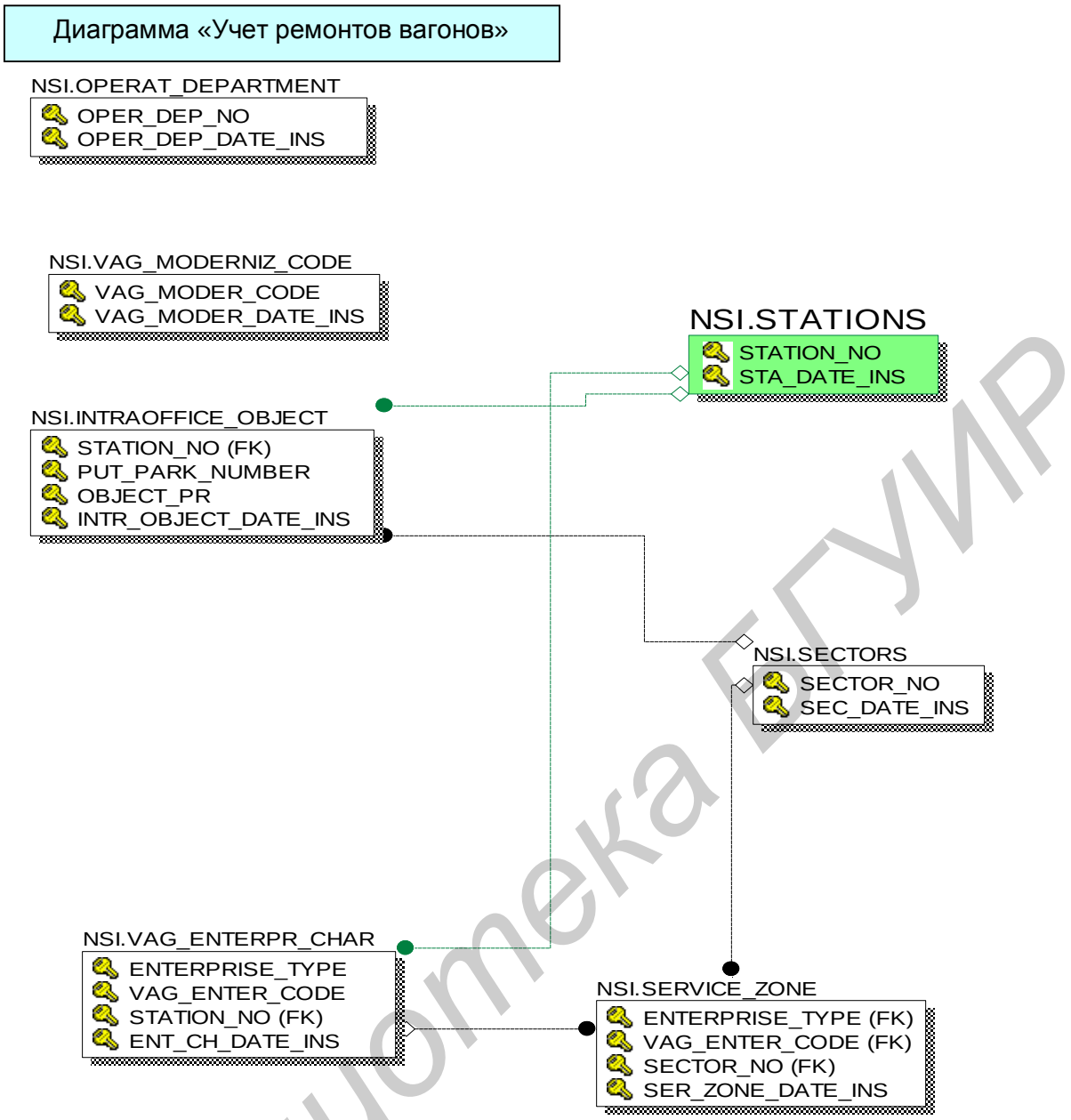

Рис. 3.7

После выбора первичного ключа из списка кандидатов ключей можно определить дополнительные ключи для таблицы **(Alternate Key Attributes – AK),** в том числе и из оставшихся после выбора PK-кандидатов ключей. Дополнительных ключей может быть несколько для одной таблицы. Они часто используются, чтобы идентифицировать дополнительные индексы для более быстрого доступа к данным из приложений. В модели данных дополнительный ключ обозначается символом **AKn,** где **n** – порядковый номер соответствующего AK для данного объекта. На диаграмме объектов это обозначение помещается после атрибутов, которые формируют эту дополнительную ключевую группу (рис. 3.8).

Часто бывает необходимо обеспечить доступ сразу к нескольким экземплярам сущности, т.е. рассматривать несколько записей как группу,

объединенную по каким-либо признакам. В этом случае можно использовать неуникальные ключи (неуникальные индексы). Атрибут или набор атрибутов, которые обычно используются для получения неуникального ключа, называются инверсионными входами (**Inversion Entry – IEn).** В диаграмме модели данных символ **IEn** помещается после атрибута, входящего в инверсионный вход для данной таблицы, где **n** – порядковый номер соответствующего IE для данной таблицы. В базе данных создается неуникальный индекс для инверсионного входа, которых может быть несколько для таблицы.

В соответствии с концепцией отношений объектов IDEF1x, связь между таблицами устанавливается миграцией атрибутов первичного ключа (PK) из родительской таблицы в детскую, где эти атрибуты детской таблицы называются уже внешним ключом **(Foreing Key – FKn)**.

Всем вышеописанным ключам Erwin автоматически присваивает имена индексов и порядковые номера путем добавления специальных кодов перед именем таблицы следующим образом (при желании имена ключей и индексов можно изменить вручную):

- Ключ PK – «X» + «имя таблицы»;
- Ключ AK – «XAKn» + «имя таблицы»;
- Ключ IE – «XIEn» + «имя таблицы»;
- Ключ FK – «XIFn» + «имя таблицы», где n – порядковый номер соответствующего типа ключа для данной таблицы.

Замечание. Некоторый атрибут может одновременно принадлежать как дополнительной ключевой группе AKn, так и группе инверсионного входа IEn.

На рис. 3.8 приведена диаграмма модели, содержащая различные ключи PK (состоят из двух атрибутов), FK (состоят из двух атрибутов), AK (состоят из трех атрибутов), IE (состоят из двух атрибутов) для каждого объекта на диаграмме. Некоторые атрибуты входят в состав различных ключей – PK, AK и IE. Данная модель содержит и другие атрибуты объектов.

3.2.3. Полная атрибутивная модель

Полная атрибутивная модель **(FAM)** получается из модели на основе ключа добавлением в сущности недостающих атрибутов, уточнением связей между сущностями, установлением ссылочной целостности и нормализацией самих сущностей.

В терминологии IDEF1x рассматривается концепция зависимых и независимых сущностей как тип логических отношений, связывающих две сущности. Сущности, которые не зависят ни от какой другой сущности в модели, называются независимыми. В зависимых сущностях устанавливается логическая связь путем использования общих атрибутов в этих сущностях (см. подразд. 2.1 и 2.2).

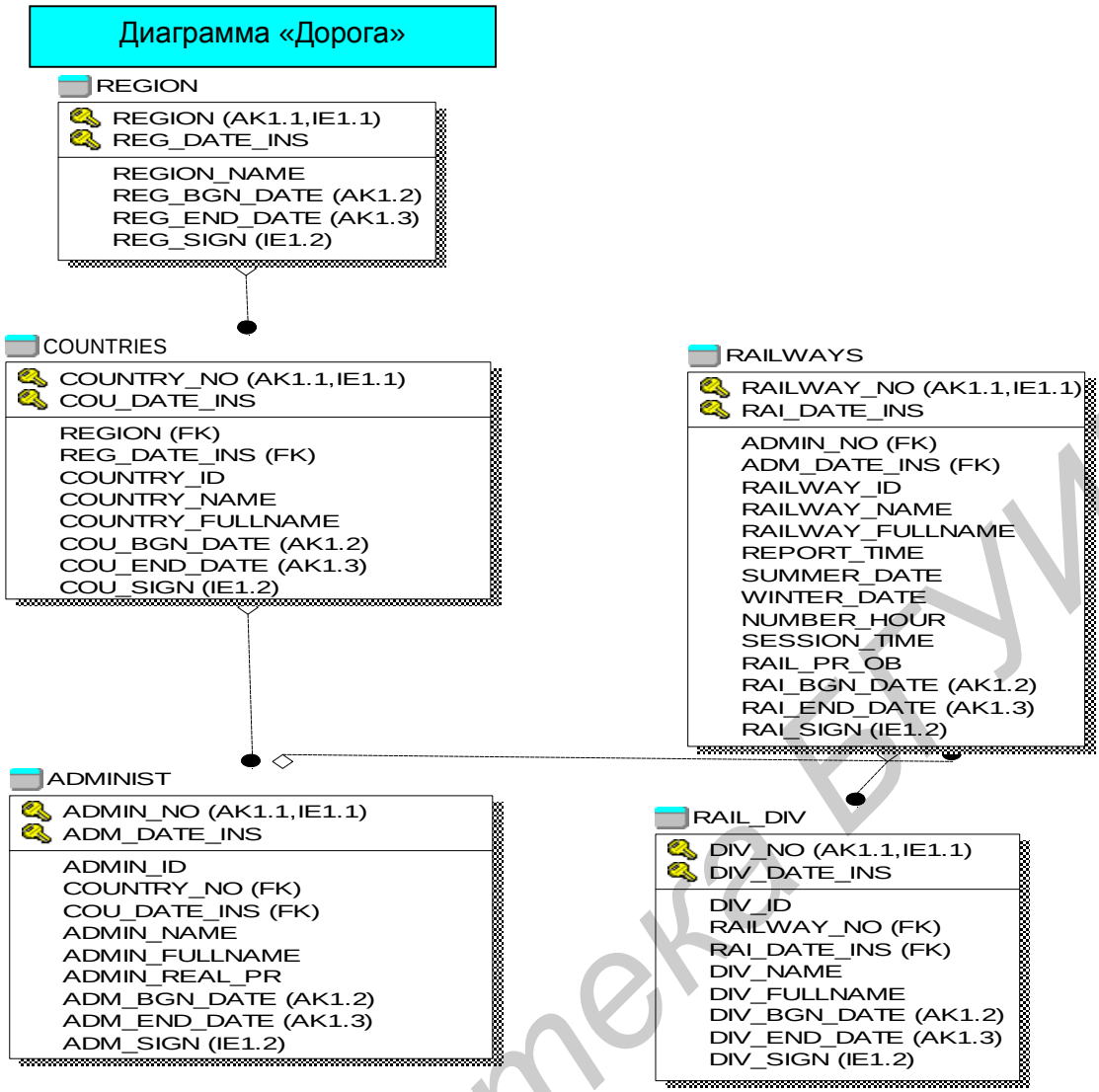

Рис. 3.8

Связевые отношения выступают как «глаголы» между сущностями («существительными») на диаграмме. Правильный подбор этих «связевых глаголов» между зависимыми сущностями позволяет использовать их для чтения отношений от родительских таблиц к дочерним и в обратную сторону, используя пассивную форму соответствующего глагола. Поскольку, как правило, для сложных систем управления модель описываемой предметной области имеет множество сущностей и всевозможных связей между ними, вышеуказанный метод чтения отношений администратором БД и аналитиком данных функциональных приложений позволяет протестировать и, таким образом, проверить правильность проектируемой логической модели (**Validation the design of model**).

Зависимая сущность не может существовать без родителя и без идентификации этой зависимости. Это означает, что зависимая сущность не может быть идентифицирована без использования ключа родителя. Таким образом, в соответствии с концепцией отношений сущностей в IDEF1x, связь

устанавливается миграцией атрибутов первичного ключа (PK) из родительской сущности в дочернюю, где эти атрибуты дочерней сущности называются уже внешним ключом (Foreing Key FK). В общем случае некоторая дочерняя сущность может быть одновременно связана с несколькими родительскими сущностями и иметь несколько внешних ключей (**FKn**). На диаграмме модели данных (рис. 3.9) соответствующие атрибуты связи в дочерней сущности сопровождаются меткой (**FKn**).

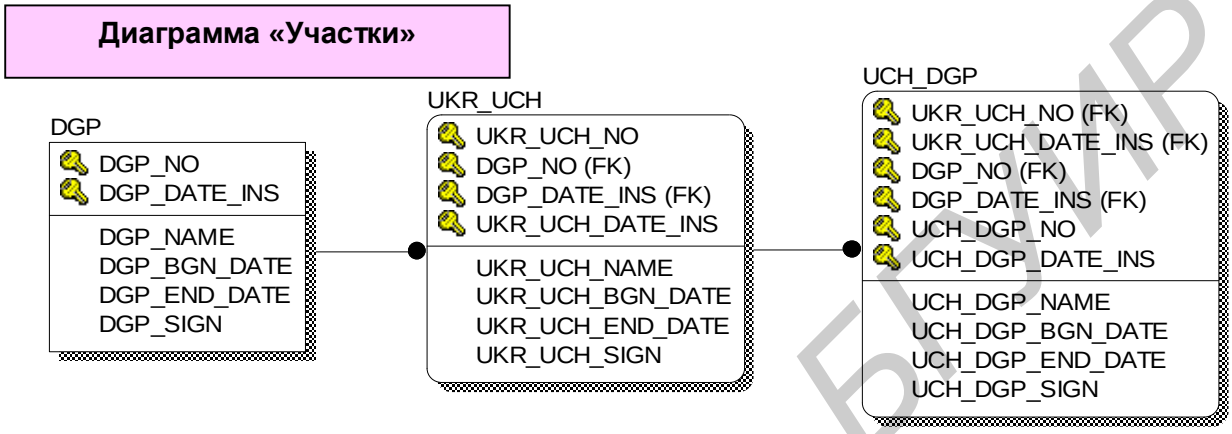

Рис. 3.9

Связь между двумя сущностями может быть определенная и неопределенная (идентифицирующая и неидентифицирующая). Идентифицирующие связи устанавливаются между сущностями, если внешний ключ **FKn** мигрирует в ключевую область зависимой сущности. На диаграмме модели идентифицирующие связи между сущностями изображаются сплошной линией, соединяющей эти сущности. Для таких типов связей и отношений СУБД самостоятельно контролирует ссылочную целостность между записями в родительской и дочерних таблицах на физическом уровне, например могут выполняться каскадные операции удаления.

Неидентифицирующие связи (отношения) устанавливаются, если **FKn** мигрирует в неключевую область зависимой сущности. На диаграмме модели данных неидентифицирующие связи между сущностями изображаются прерывистой линией, соединяющей эти сущности. В обоих случаях на диаграмме помещается точка в конце соединительной линии у дочерней сущности (рис. 3.10).

Для таких типов связей стратегия ссылочной целостности между записями в родительской и дочерних таблицах изменяется. Например, для задачи резервирования билетов (родительская сущность – «заказчик», дочерняя – «места»), при отказе «заказчика» от зарезервированных билетов выполняется операция удаления записи в родительской таблице (отказ от заказанных билетов), записи в дочерней таблице, связанные с записью в родительской

таблице, могут и не удаляться, а их FK принимать значение по умолчанию (например NULL).

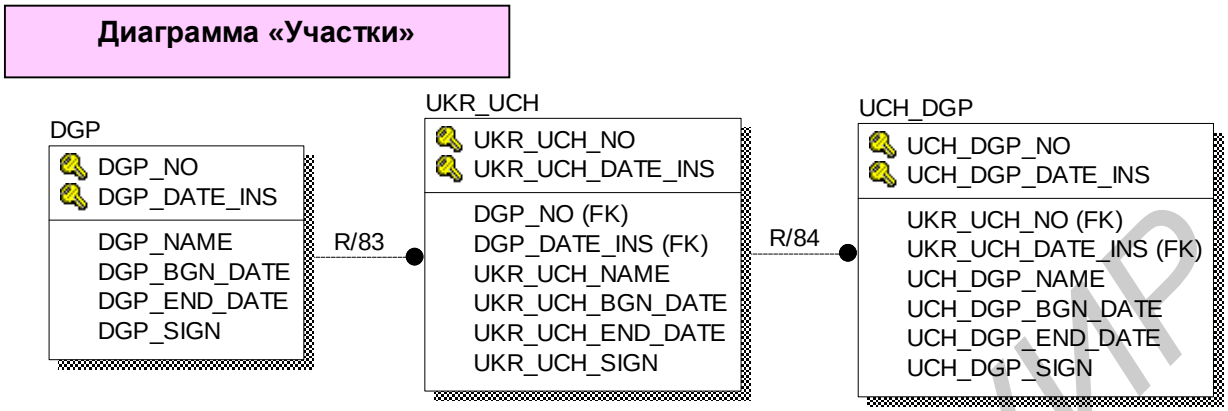

Рис. 3.10

Всем вышеописанным связям Erwin автоматически присваивает имена связи и порядковые номера связи. Порядковый номер связи (глагол связи) формируется следующим образом: «R/n», где n – порядковый номер соответствующей связи для данной схемы. Имя связи между двумя сущностями формируется путем помещения глагола (номера) связи между именами родительской и дочерней сущностей следующим образом: «имя родительской сущности» + «Номер связи» + «имя дочерней сущности».

Пример модели диаграммы с указанием режимов ссылочной целостности приведен на рис. 3.11, значения самих режимов указаны в подразд. 2.5. Здесь D:NOA, U:NOA означает, что при удалении или корректировке записей в родительской сущности никаких действий не требуется, I:R, U:R означает, что вставка или корректировка записей в дочернюю сущность без соответствующей родительской записи недопустима. Сами режимы устанавливаются по умолчанию и могут быть изменены разработчиком БД. На основании этих режимов автоматически генерируются триггеры поддержания ссылочной целостности.

После построения «Атрибутной модели» (FAM) выполняется нормализация данных. Здесь приводятся основные определения нормальных форм данных. Примеры нормализации данных можно найти в любом учебнике по БД. Процесс нормализации производится углубленным анализом данных и отношений между сущностями (таблицами) и их атрибутами (столбцами), последовательным приведением модели к установленным в теории разработки структур баз данных нормальным формам [4]. Известно 6 нормальных форм, из которых на практике ограничиваются тремя.

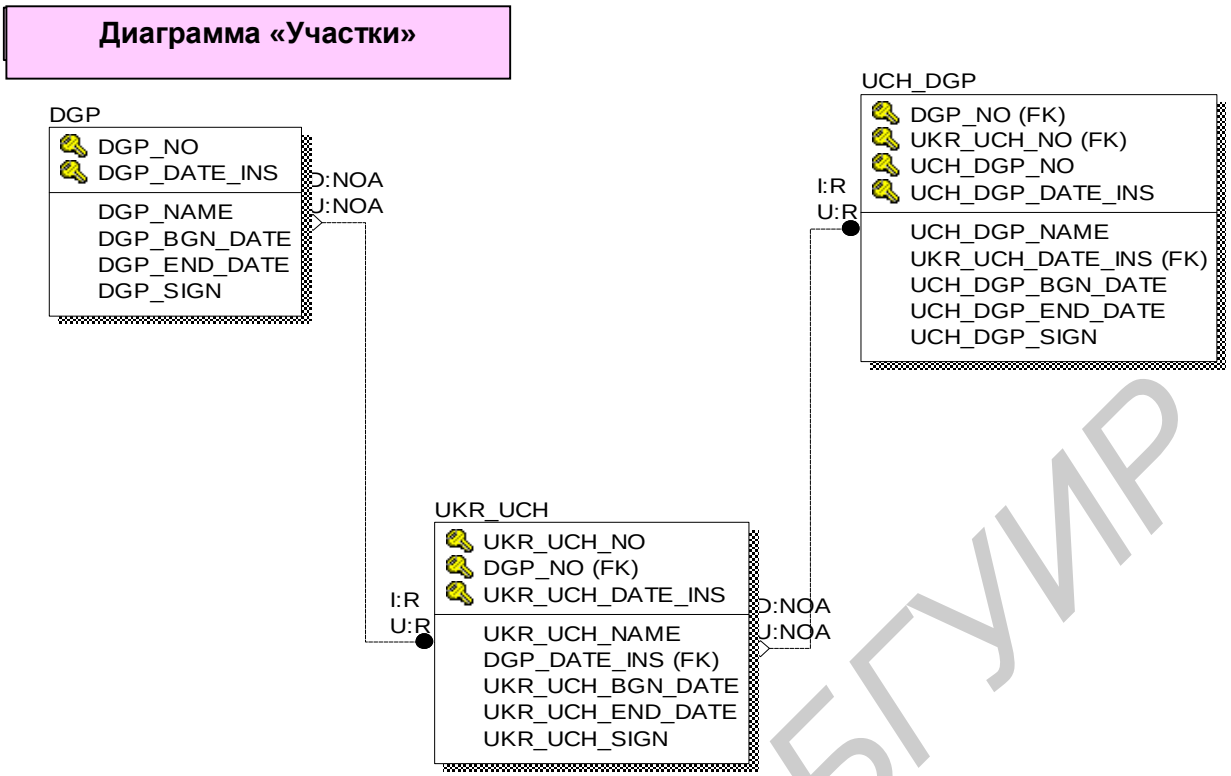

Рис. 3.11

Нормализация данных – это процесс проверки, тестирования и, если требуется, реорганизации сущностей (таблиц) и атрибутов (столбцов) в модели и сведения таблиц к набору столбцов, в котором все неключевые столбцы зависят от столбцов первичного ключа.

Цель нормализации – создание структуры данных, в которой информация о каждом факте хранится в одном месте. Она позволяет устранить недостатки в уже спроектированной модели данных, исключить дублирование данных, это значит, что информация о некотором факте должна храниться только в одном месте, что позволит однозначно и окончательно определить принадлежность каждого столбца только определенной таблице. Нормализация делает БД более устойчивой к изменениям для решения новых задач, позволяет обеспечить целостность самой БД и сокращает объем памяти для хранения информации.

Все нормальные формы основаны на понятии функциональной зависимости.

Атрибут A1 сущности E функционально зависит от атрибута A2 сущности E тогда и только тогда, когда каждое значение A2 в сущности E связано с только с одним значением A1 в сущности E, т.е. для любого значения $x \leq A1$ существует только одно значение $y \leq A2$.

Атрибут A1 сущности E полностью функционально зависит от ряда атрибута A2 сущности E тогда и только тогда, когда A1: 1) функционально зависит от A2; 2) нет зависимости от подмножества A2, т.е. нет транзитивной зависимости.

Первая нормальная форма (1NF) гласит: все атрибуты сущности содержат атомарные (неделимые) значения и среди атрибутов не должно встречаться повторяющихся групп. Это значит, с одной стороны, атрибуты не должны содержать повторяющихся групп (нескольких значений атрибута) и, с другой, – атрибут не должен хранить разные по смыслу значения. Приведение к 1NF форме производится созданием новой таблицы для повторяющихся значений и установлением связи от прежней таблицы (PK) к новой (FK).

Сущность находится во второй нормальной форме (2NF), если она находится в первой нормальной форме (1NF) и любой ее неключевой атрибут полностью зависит от всего первичного ключа, а не от его части. 2NF имеет смысл для объектов со сложным ключом. Сущность приводится к 2NF выделением атрибутов, зависящих от части ключа в отдельную таблицу и установлением связи с основной таблицей.

Сущность находится в третьей нормальной форме (3NF), если при том, что она находится во второй нормальной форме (2NF), любой ее неключевой атрибут не зависит от другого неключевого атрибута этой сущности. Неключевые атрибуты не зависят от неключевых атрибутов, в сущности нет атрибутов значения, которые получены из других атрибутов данной сущности. Сущность приводится к 3NF выделением атрибутов, зависящих от неключевого атрибута в отдельную таблицу и установлением связи с основной таблицей.

Сущность находится в четвертой нормальной форме (4NF), если ни в одной строке нет нескольких многозначных фактов об одном объекте, т.е. нет многозначных зависимостей между атрибутами. Сущность приводится к 4NF выделением атрибутов с многозначными фактами в отдельные таблицы.

CASE-система AllFusion Erwin Data Modeler помогает в создании нормализованной модели БД, контролирует уникальность имени атрибута, поддерживает создание ключей и связей.

4. Создание физического уровня базы данных на основе логического

Как отмечено в подразд. 3.1, нужно сначала определить сущности и их характеристики на логическом уровне, затем при первоначальном переключении на физический уровень соответствующие термины для него будут созданы автоматически. Переключение между уровнями осуществляется с помощью списка выбора «Model type indicator» – «Указатель типа модели» на панели инструментов AllFusion Erwin Data Modeler (logical / physical).

Перечень основных функций проектирования модели данных, их использование в системе AllFusion Erwin Data Modeler и краткое описание приведены в прил. 1. Прил. 1 содержит перечень основных функций проектирования (графа 2) для создания и модификации схемы модели БД. Графа 3 содержит ссылку на пункт меню (***M:***) в системе AllFusion Erwin Data Modeler, а графа 4 содержит ссылку на подпункт меню «Режимы» (***O:***) для выполнения функций через меню. При работе в пространстве «Model Explorer» (***ME:***) графа 3 содержит название элемента схемы и действия разработчика БД по инициированию операции, а графа 4 содержит название элемента графической схемы и действия разработчика БД по инициированию этой операции в пространстве «Diagram Window» (***DW:***). Следующая графа 5 таблицы содержит задаваемые параметры для данной операции. И, наконец, в этой таблице для каждой функции и операции имеется ссылка (графа 6) на документ «AllFusion Erwin Data Modeler. Getting started». Она указывает номер и название главы документа и номер страницы для получения более детальной информации пользователя о выполняемой операции. Кроме этого, при работе с любым компонентом пользователь может в любой момент времени воспользоваться помощью (меню «Help»).

При создании логической и физической схем БД последовательно используются функции, представленные в прил. 1:

1 – создание модели данных;
2 – создание/ добавление таблицы;
3 – именование/ переименование таблиц;
4 – добавление атрибута в таблицу;
5 – добавление связей;
6 – получение новой модели;
7 – сохранение модели данных;
8 – выборка модели БД;
9 – просмотр DDL (Script File);
10 – генерация Script File. Используется далее для получения новой схемы.

Замечание. Выполнение указанных функций приводит к созданию/изменению схемы БД, а не физической БД, которое производится средствами СУБД.

4.1. Создание новой модели

Выполнение ряда функций в AllFusion Erwin Data Modeler на уровне схемы модели БД производится довольно просто. Например, функция 1 – создание новой логической/физической модели данных производится в меню «File» выбором опции «New» (см. п. 1, прил. 1). В появившемся окне необходимо выбрать из предложенного списка тип модели (логическая/физическая), указать используемую базу данных (DB2) и выбрать версию (рис. 4.1).

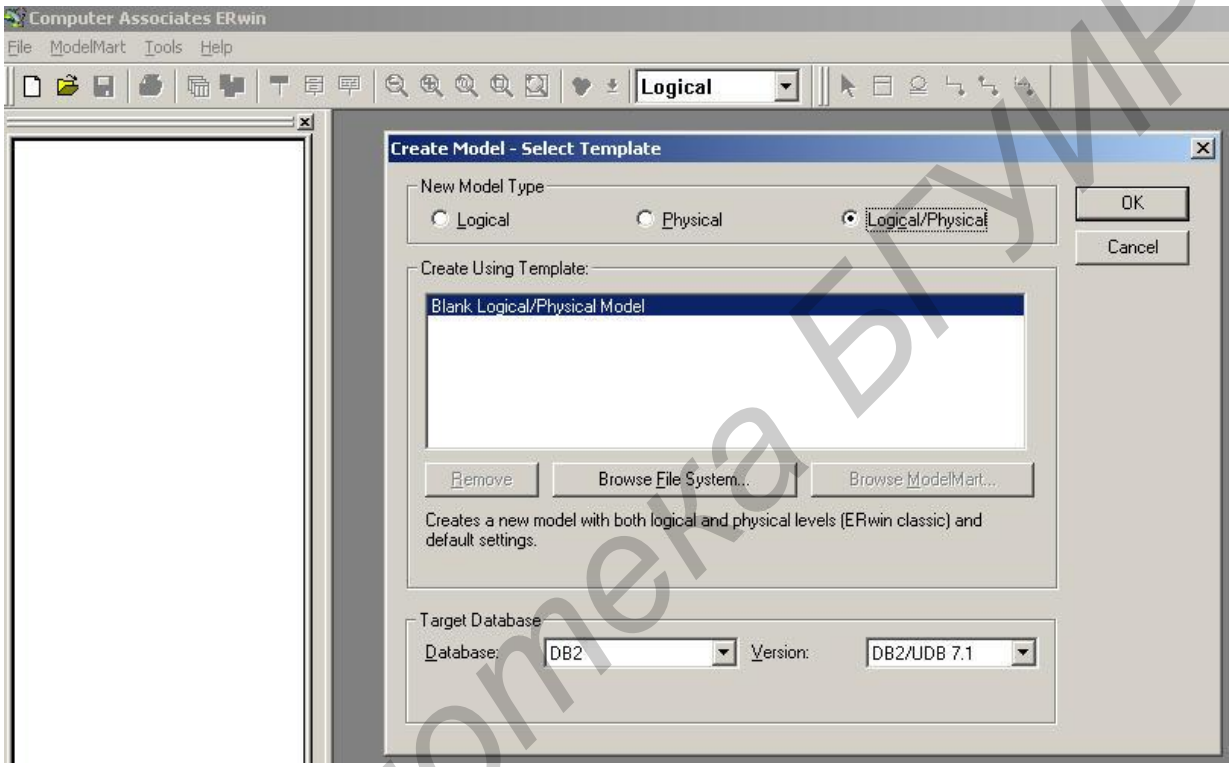

Рис. 4.1

После нажатия кнопки «OK» AllFusion Erwin Data Modeler открывает окна «Model Explorer» и «Diagram Window» для выполнения следующей функции.

4.2. Редактирование таблиц

Функция выполняется в графическом и текстовом режиме. Выполнение функции 2 – создание/добавление таблицы в графическом режиме производится помещением курсора в поле «AllFusion Erwin Data Modeler Toolbox» и нажатием дважды на рисунок, изображающий таблицу «Entity». Затем курсор перемещается в некоторое выбранное место для размещения

таблицы в «Diagram Window» и делается щелчок мышкой. Таблица появляется с временным именем E/1. Такая операция называется перетягиванием и может повторяться для нескольких таблиц.

Поверх временного имени таблицы набирается ее действительное имя. Сразу после этого можно вводить имена атрибутов данной таблицы. Как указано выше, одновременно в окне «Model Explorer» в текстовом виде отражается появление созданной новой таблицы. Выполнение этой функции можно производить и в текстовом режиме в окне «Model Explorer»: правый щелчок по «Entity», во всплывающем меню выбрать «New» и щелкнуть по нему. В появившемся окне поверх имени E/1 набрать нужное имя таблицы. После появления таблицы в окне «Model Explorer» можно выполнять другие функции с таблицей схемы: переименование (Rename), удаление (Delete), переход к таблице на схеме (Go to), а также показать свойства (возможно для их корректировки или дополнения) таблицы (Property). Для этого необходимо сделать правый щелчок по выбранному «Entity», во всплывающем меню выбрать требуемую функцию. Например, выполнение функции 3 – переименование таблицы легко выполняется в окне «Model Explorer»: правый щелчок по «Entity», во всплывающем меню выбрать «Rename» и щелкнуть по нему. Поверх старого имени ввести новое имя таблицы.

Первоначальное определение объекта (Name, definition, note) производится в логической схеме, дальнейшее доопределение параметров объекта/таблицы при создании или добавлении таблицы можно производить в меню «Property» в окне «Model Explorer» по мере развития логической схемы и перехода ее к физической.

Параметры сущности для логической модели:

- название объекта (Name);
- определение/ название таблицы (Definition);
- замечания 1, 2, 3 (Note 1, 2, 3);
- определенные пользователем свойства (UDP);
- значок объекта (Icon);
- история и комментарии (History) – кто отвечает за определение, за ввод, состояние и последнее изменение, связи таблицы и другие особенности;
- признак участия объекта (Logical Only) (рис. 4.2).

Параметры таблицы для физической модели:

- название таблицы (Name);
- схема (Owner);
- комментарии (Comment);
- размерность: начальное количество строк, их максимум, рост (Volumetric: initial row, max, grow by);
- физические характеристики: об именах табличных пространств для таблицы, ее индексов, длинных записей (Phis prop: Table, index, long Tablespace);

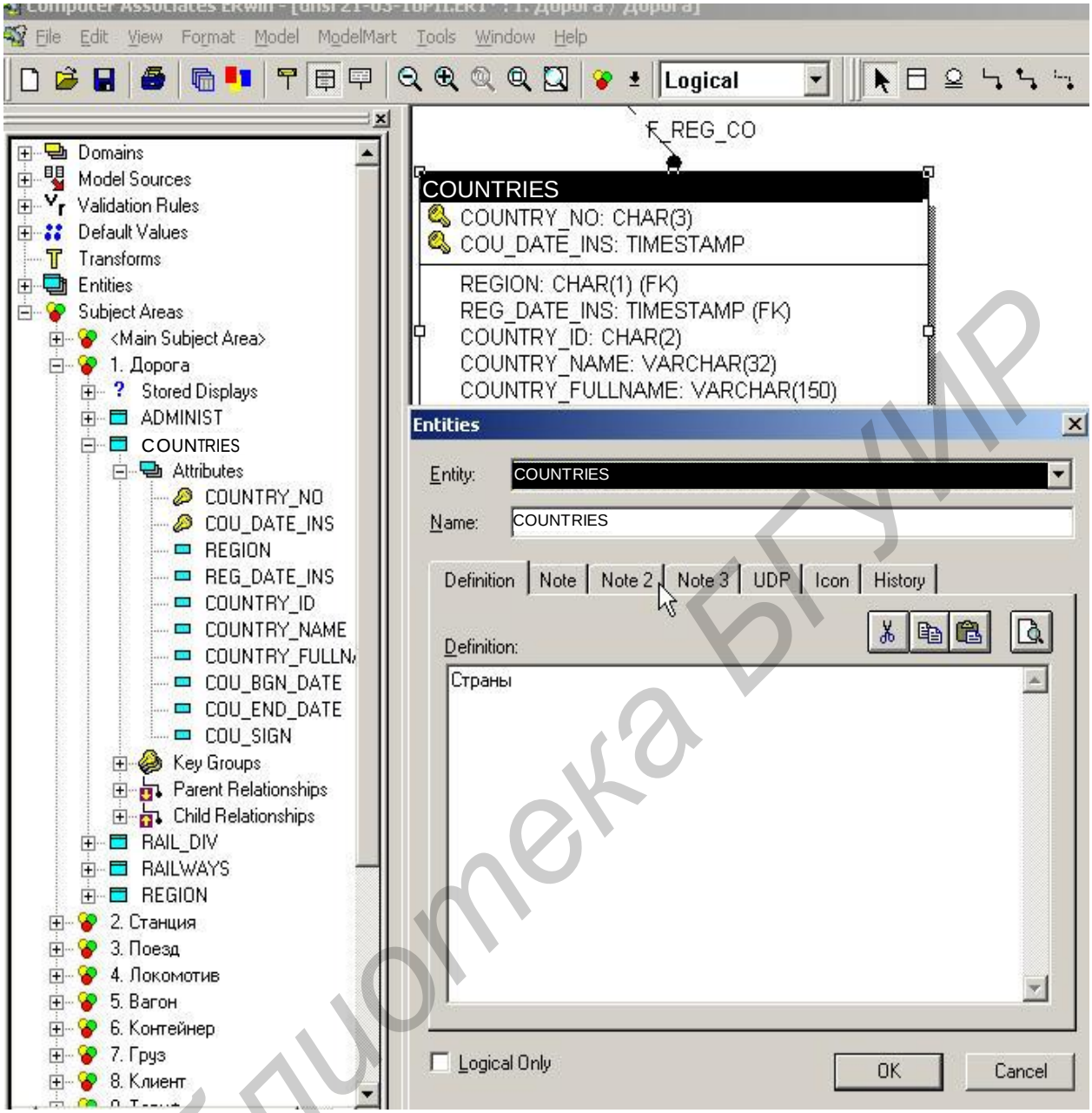

Рис. 4.2

- физические характеристики: захват данных, % свободного места, вставка, репликация, восстановление и т.д., признак таблицы (Data Capture, Free %, Append, Replicated, Refresh, …, Physical Only);
- определенные пользователем свойства (UDP);
- история и комментарии (History);
- ограничение (Validation);
- алиасное имя (Alias) (рис. 4.3).

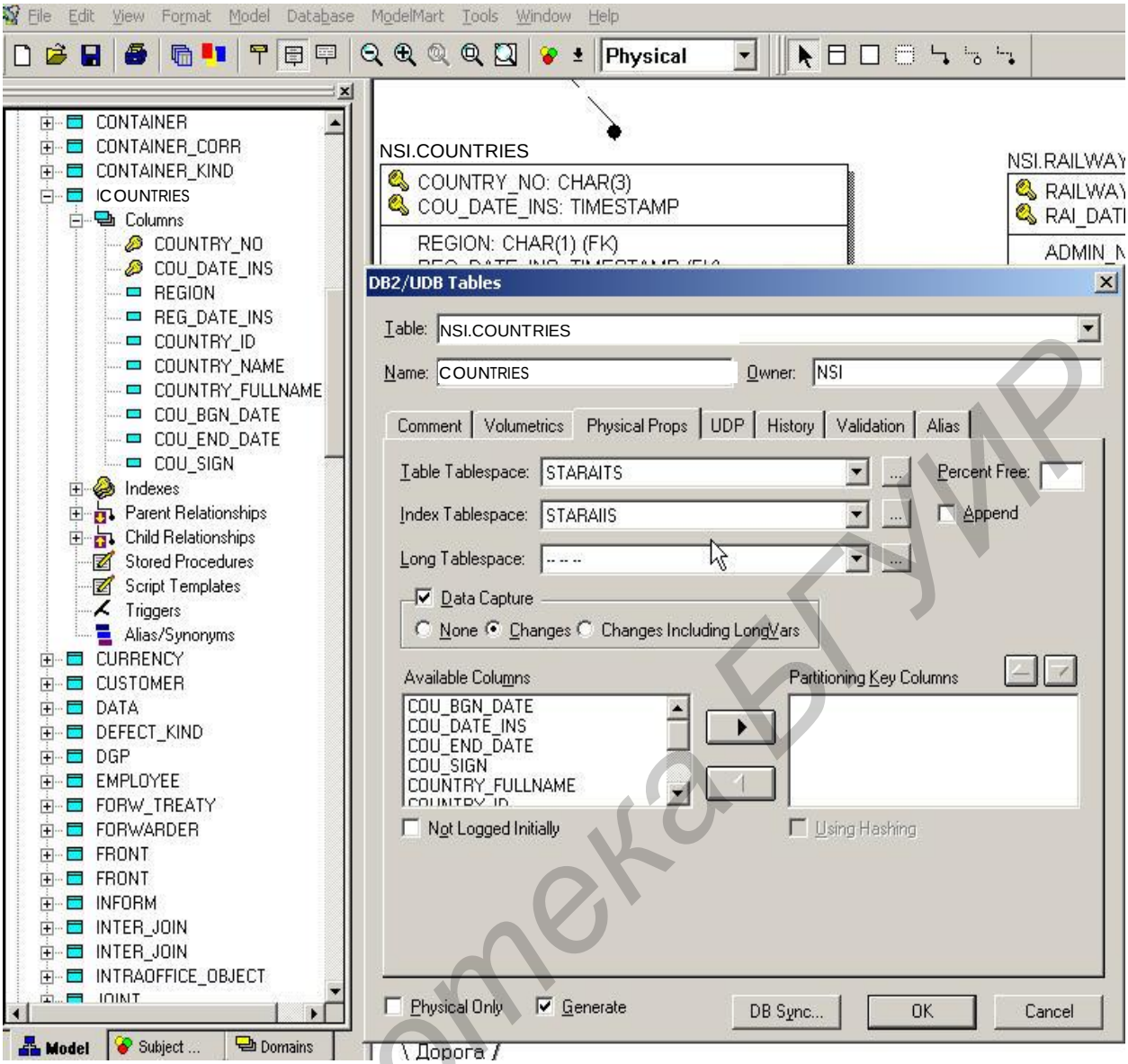

Рис. 4.3

4.3. Редактирование столбцов таблицы

Аналогично вышеизложенному производится выполнение функции 4 по добавлению/переименованию/удалению атрибута таблицы. При создании таблицы и ее атрибутов в соответствии с требованиями реляционных баз данных необходимо обеспечить однозначную идентификацию каждой строки таблицы. Это достигается путем выбора одного или нескольких атрибутов таблицы в качестве первичного ключа (PK). Эти атрибуты образуют область ключа таблицы (Key area). Остальные неключевые атрибуты образуют область данных таблицы (Data area). Эти области разделяются в таблице в окне

«Diagram Window» горизонтальной линией, поэтому при добавлении атрибутов во вновь создаваемую таблицу сначала выше горизонтальной линии вводятся ключевые атрибуты, а затем после нажатия клавиши «Tab» – неключевые. AllFusion Erwin Data Modeler включает удобные средства перемещения атрибутов между вышеуказанными областями и внутри этих областей при корректировке модели.

Первоначальное определение атрибута (Name, General, Datatype, definition, note, Key Group) производится в логической схеме. Дальнейшее доопределение параметров атрибута/столбца таблицы при его добавлении можно производить в меню Property в окне «Model Explorer» по мере развития логической схемы и перехода от нее к физической (рис. 4.4).

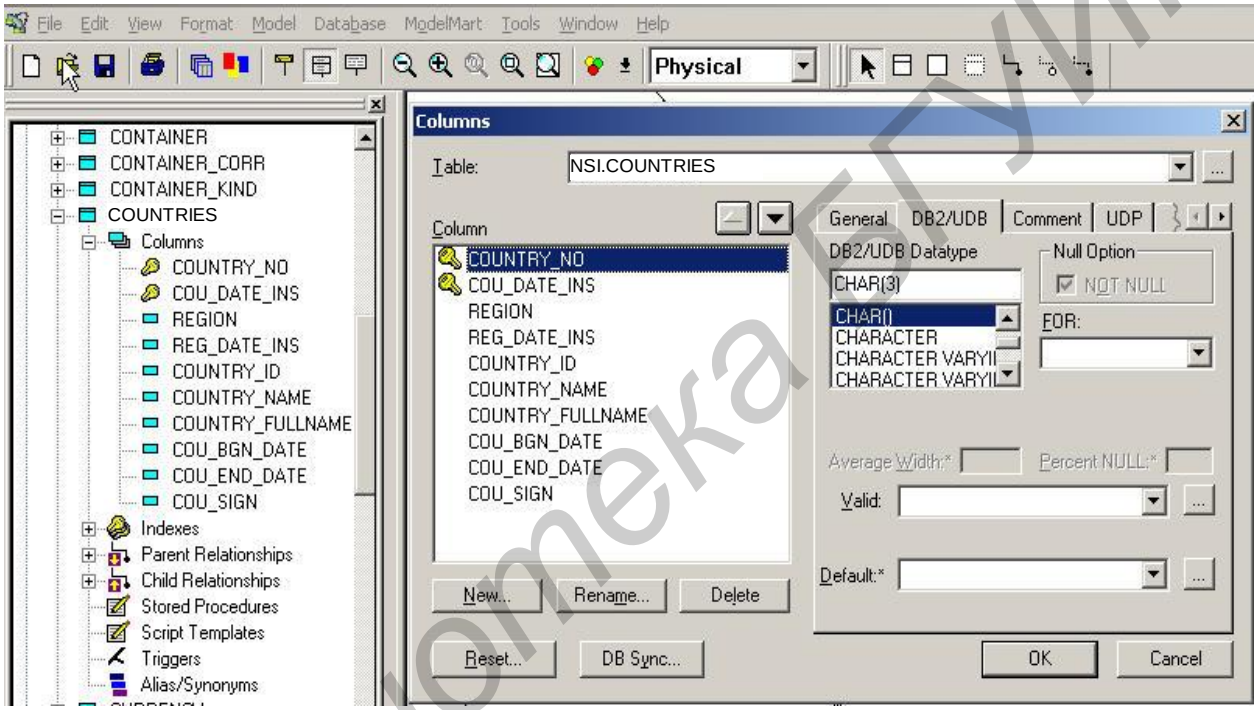

Рис. 4.4

Параметры атрибута объекта для логической модели:
- его имя в перечне имен атрибутов объекта (Name);
- общие свойства (General);
- тип данных (Datatype);
- определение (Definition);
- замечание (Note);
- пользователем определенные свойства (UDP);
- участие в ключах (Key Group);
- история атрибута объекта (History).

Параметры столбца таблицы для физической модели:

- имя его в перечне имен столбцов таблицы (Name);
- характеристики столбца для выбранной СУБД DB2 UDB: тип данных, Null/Not Null, ограничения, значение по умолчанию (DB2 UDB: Datatype, Null/Not Null, Validation, Default);
- комментарии (Comment);
- определенные пользователем свойства (UDP);
- участие столбца в индексах (Index);
- история столбца (History) (рис. 4.5).

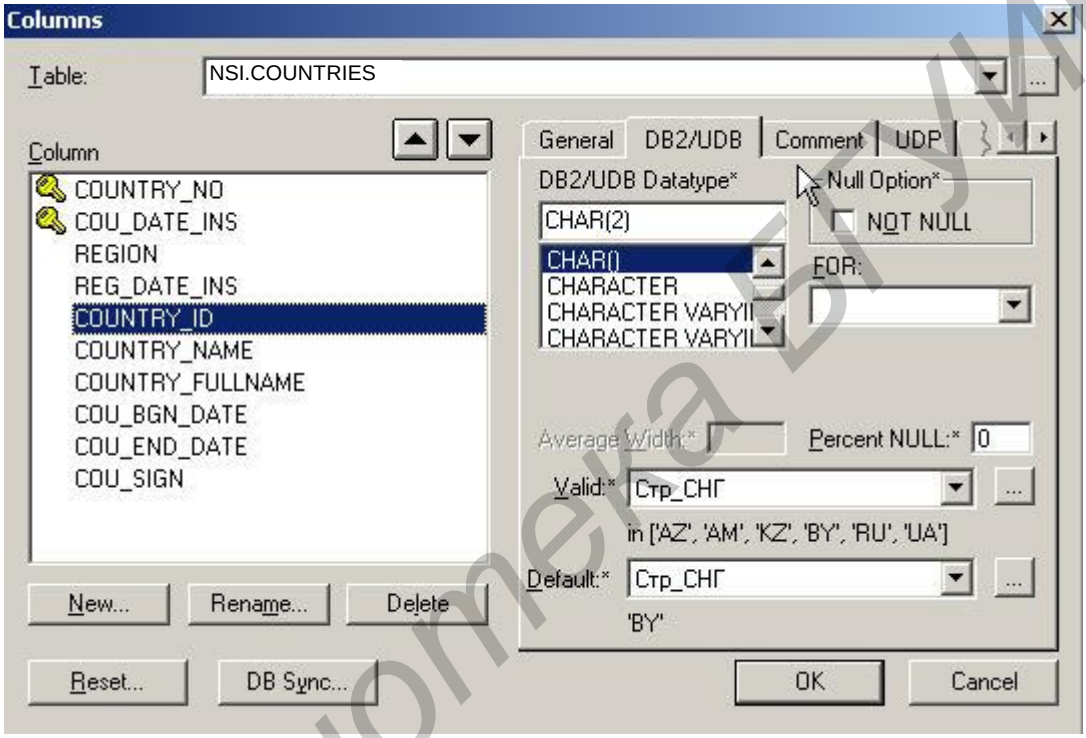

Рис. 4.5

4.4. Редактирование ключей и индексов таблицы

При проектировании логической модели для некоторого объекта может создаваться «Ключевая группа» (см. меню «Key group» для логической модели). Она автоматически включает в себя ключи PK, а также другие виды ключей-индексов (описание см. выше). Первоначальное определение ключей (имя ключевой группы, тип индекса таблицы) производится в логической схеме. Детальное проектирование ключей и индексов (Index), как правило, производится на уровне физической модели.

Как уже излагалось выше, первичный ключ (PK) – это набор атрибутов, которые используются для однозначной идентификации экземпляров объекта. Первичный ключ может включать один или более первичных ключевых

атрибутов. Ключ PK всегда единственный для таблицы и создается автоматически для каждой таблицы при вводе атрибутов в ключевую область.

Выполнение данной функции производится в меню «Key Group»/«Index» в окне «Model Explorer» выбором опции «Property»/«New» (рис. 4.6).

Параметры ключевой группы для логической модели:

- имя ключевой группы (Name);
- тип ключевой группы (PK, AKn, IEn, FKn);
- члены ключевой группы (Members);
- определение (Definition);
- замечание (Note);
- определенные пользователем свойства (UDP).

Параметры индекса таблицы для физической модели:

- имя индекса;
- тип индекса (PK, AKn, IEn, FKn);
- имена столбцов, включенных в индекс;
- свойства СУБД для индекса (уникальность (для PK, AK) или ее отсутствие (для IE), кластеризация, генерация, резервируемый процент свободного места для области индексов);
- комментарий для индекса;
- пользователем определенные свойства (UDP).

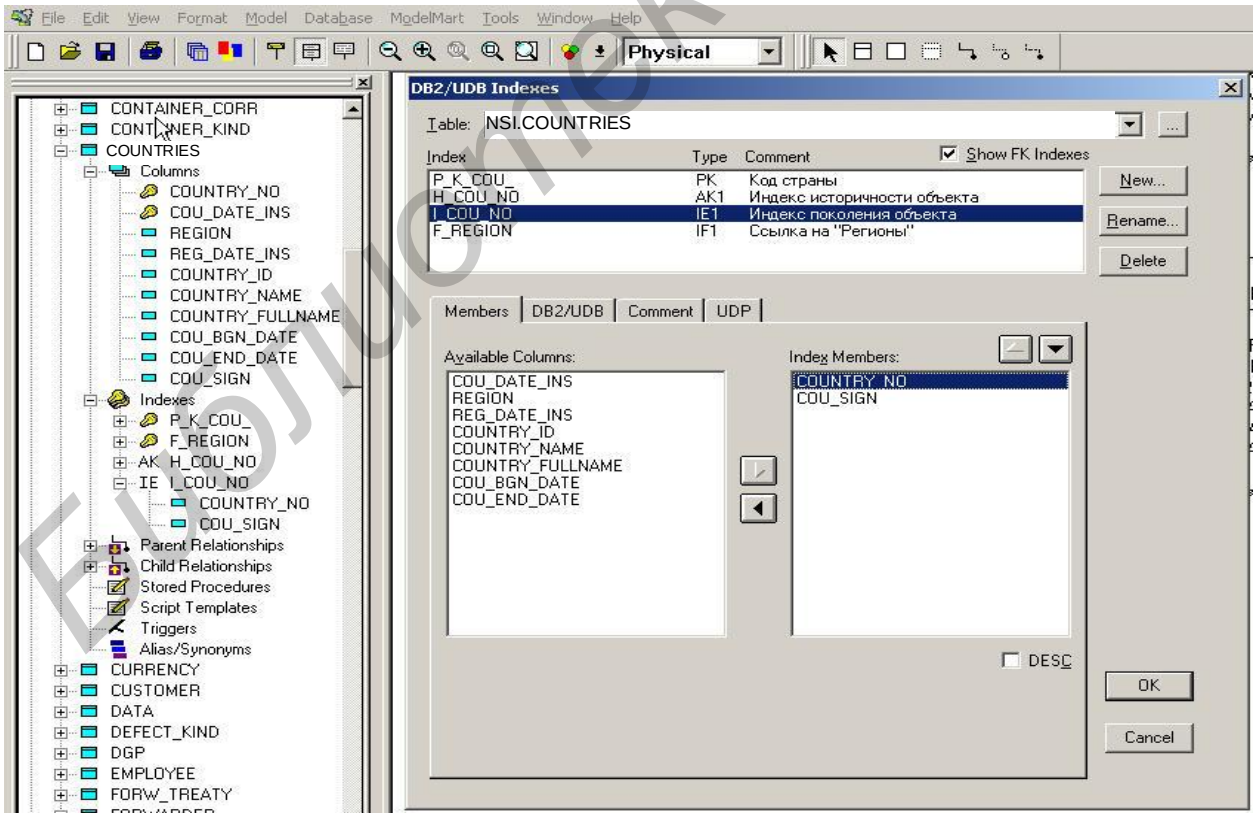

Рис. 4.6

AllFusion Erwin Data Modeler автоматически присваивает имена и порядковые номера вышеуказанным индексам путем добавления специальных кодов перед именем таблицы следующим образом (см. п. 3.2.2):

- ключ PK – «X» + «имя таблицы»;
- ключ AK – «XAKn» + «имя таблицы»;
- ключ IE – «XIEn» + «имя таблицы», где n – порядковый номер соответствующего типа ключа для данной таблицы;
- ключ FK – «XIFn» + «имя таблицы», где n – порядковый номер соответствующего типа ключа для данной таблицы.

Название ключа можно переименовать, используя клавишу «Rename», так же, как и изменить любой из вышеуказанных параметров индекса на уровне логической или физической модели.

4.5. Редактирование связей таблиц

Модель БД имеет сложную древовидную (сетевую) структуру, в которой все сущности (таблицы) связаны между собой некоторыми связями или отношениями (см. подразд. 2.2). Как уже отмечалось, зависимая (дочерняя) сущность не может существовать без родительской и без идентификации этой зависимости. Это означает, что зависимая таблица не может быть идентифицирована без использования ключа родителя. Такая связь изображается сплошной или прерывистой линией между родительской и дочерней таблицами в окне «Diagram Window». Каждая таблица может выступать в какой-то связи как родительская («Parent relationships») и в то же время в какой-то другой связи как дочерняя («Child relationships»). В диаграмме сущностей схемы БД предусмотрено окно «Relationships» для регистрации всех параметров этой связи между сущностями (рис. 4.7).

Активизация этого окна производится в окне «Diagram Window» щелчком мыши по выбранной линии связи или в окне «Model Explorer» выбором нужной таблицы, меню «Parent relationships» или «Child relationships», опции «Property»/«New».

Параметры связи между двумя таблицами для физической модели (см. рис. 4.7 и 4.8):

- имя связи;
- имя глагола связи от родителя к дочерней или наоборот («Parent – to child»/«Child – to parent»);
- тип связи: определенная или неопределенная;
- мощность связи (cardinality): отношение количества экземпляров записей от родительской к детской;
- определение связи;
- имена столбцов связи;

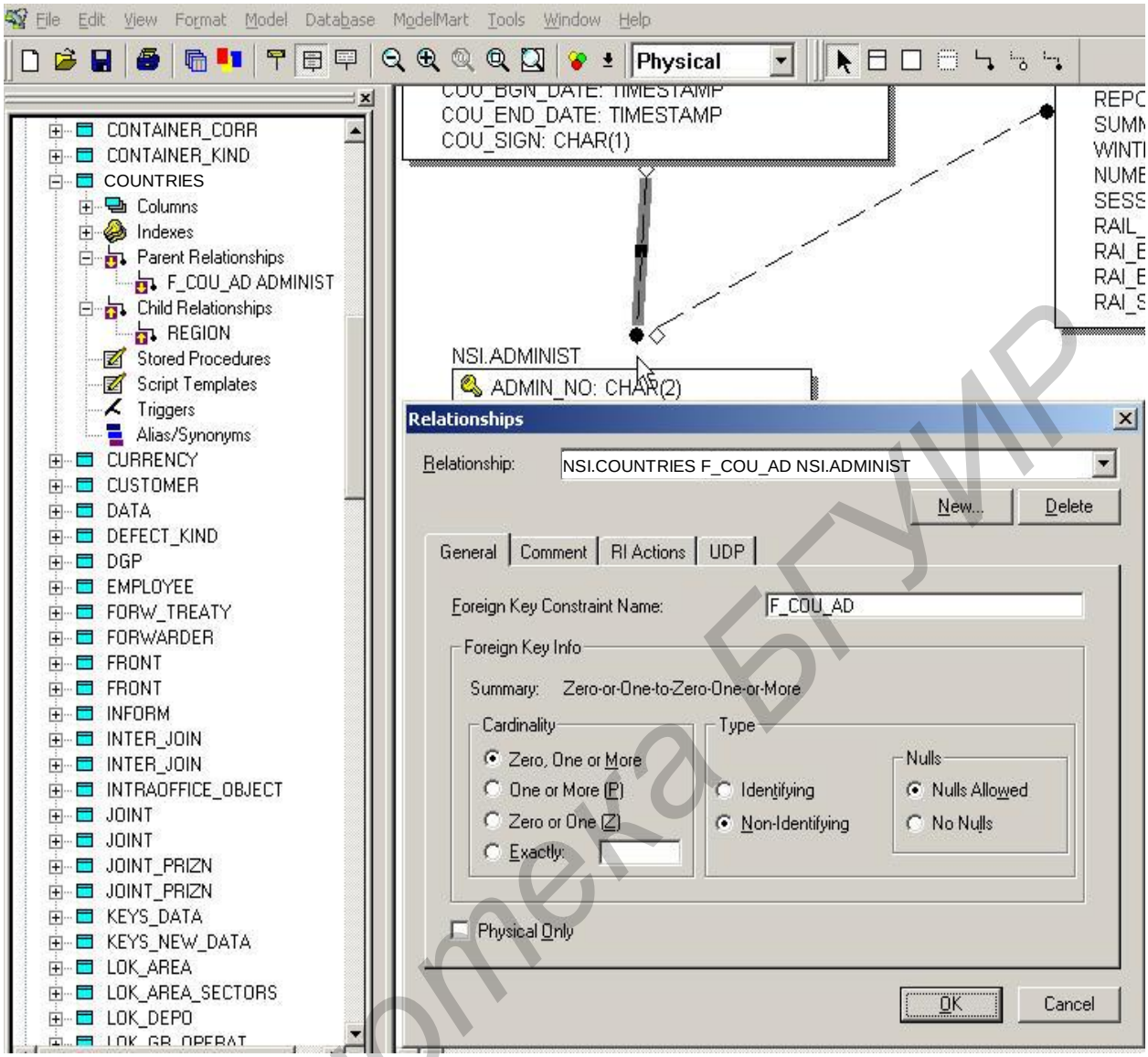

Рис. 4.7

- действия по поддержке относительной целостности связи для операций Insert/Delete/Update;
- определенные пользователем свойства (UDP) (рис. 4.8).

4.6. Сохранение модели базы данных

Создание модели данных – это длительный итерационный процесс, выполнение вышеизложенных функций и операций разработчиком БД может повторяться многократно для получения схемы модели данных, удовлетворяющей требованиям задач и пользователей на некотором этапе сопровождения БД.

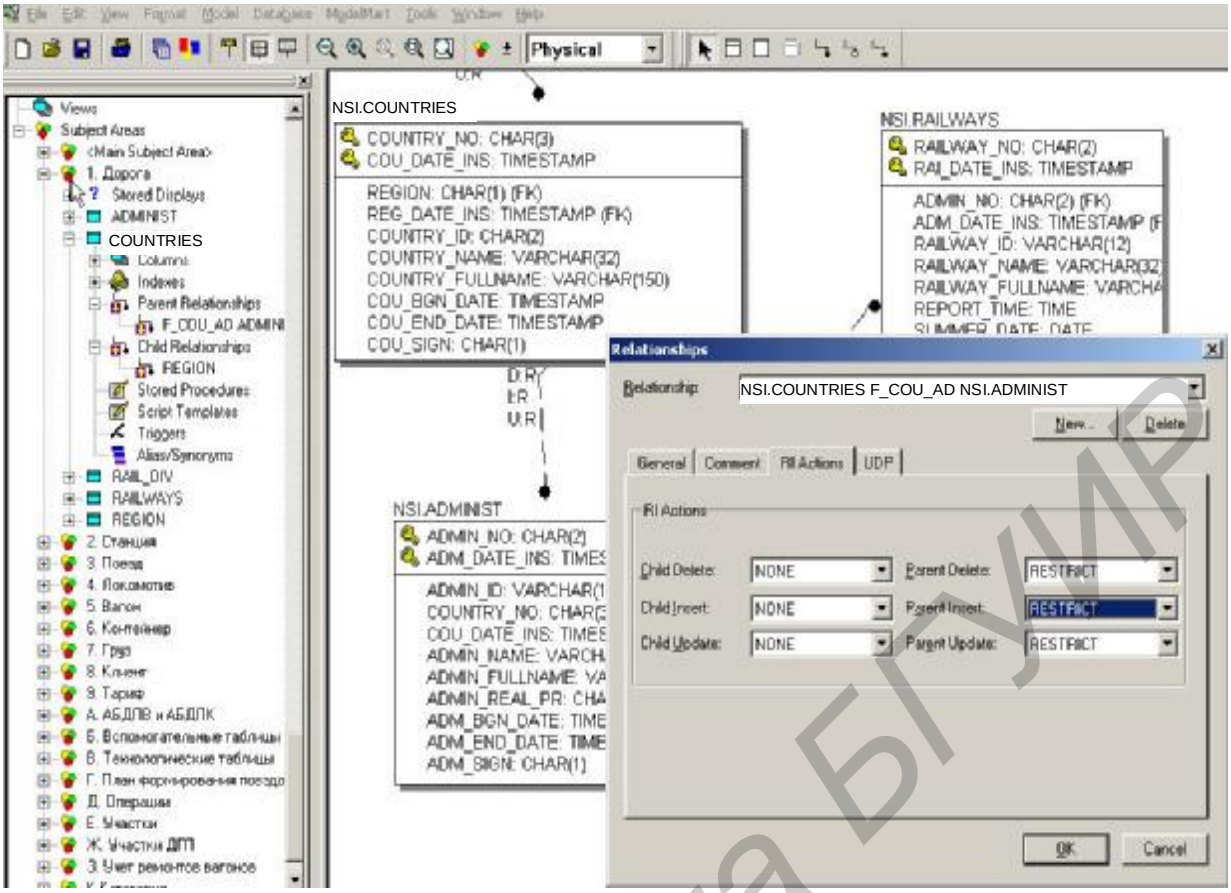

Рис. 4.8

При этом полученная модель всякий раз после окончания корректировки должна быть сохранена с помощью операций «Save» или «Save as» (см. прил. 1, функция 7) в виде AllFusion Erwin Data Modeler файла с расширением *.er1. Эта модель в последующем может быть выбрана для дальнейшей обработки с помощью операции «Open» (см. прил. 1, функция 8). Все эти функции AllFusion Erwin Data Modeler достаточно подробно описаны в документе «AllFusion Erwin Data Modeler. Getting Started».

AllFusion Erwin Data Modeler запоминает имена файлов для создаваемых моделей и их местоположение (пути доступа) и предлагает их для выбора при входе. Тем самым ускоряется процесс выборки требуемой модели и облегчается сопровождение БД.

4.7. Генерация операторов для создания базы данных

После окончательного определения и построения физической модели данных можно перейти к выполнению одной из наиболее важных функций – генерации схемы, а именно – генерации операторов на языке DDL,

описывающих БД. Эти операторы используются затем для создания физической БД и ее поддержки на выбранном для ее размещения сервере.

Ниже приводится последовательность действий разработчика для генерации операторов для создания БД (см. прил. 1, функции 10). AllFusion Erwin Data Modeler обеспечивает ведение следующих групп опций (режимов) для описания (указания) необходимых разработчику вариантов генерации схемы: «Schema», «Summary table», «View», «Table», «Column», «Index», «Referential Integrity», «Trigger», «Other options». Включение параметров для указанных групп опций влияет на характеристики создаваемой физической БД, которые зависят от выбранной платформы СУБД. В связи с этим не все группы опций используются одинаково для различных СУБД и, естественно, не все опции используются в данной разработке БД. Ниже приводится описание наиболее важных из используемых групп опций. Каждая из этих групп опций определяется в соответствующем диалоговом окне. При перечислении опций указывается аббревиатура опции на английском языке, описывается назначение данной опции и необходимость использования данной опции для БД с помощью следующих обозначений: (В) – включается режим при генерации схемы, (Н) – не включается режим при генерации схемы.

Например, опции **«Schema»** позволяют включить некоторые общие управляющие характеристики из физической схемы в генерируемые операторы DDL. К ним относятся:

Create Procedure – включить схемные хранимые процедуры в генерируемую AllFusion Erwin Data Modeler схему (Н);

Drop Procedure – включить операторы DROP PROCEDURE для схемы в генерируемую AllFusion Erwin Data Modeler схему (Н);

Create Tablespace – включить операторы CREATE TABLESPACE в генерируемую AllFusion Erwin Data Modeler схему (Н);

Create Bufferpool – включить операторы CREATE BUFFERPOOL в генерируемую AllFusion Erwin Data Modeler схему (Н);

Create Nodegroup – включить операторы CREATE NODEGROUP в генерируемую AllFusion Erwin Data Modeler схему (Н);

Create Pre/Post Script – создать Pre/Post скрипты для схемы перед/после генерации схемы (Н).

Опции **«Table»** позволяют включить в генерацию схемы операторы определения данных таблиц. К ним относятся:

Create table – включить операторы CREATE TABLE при генерации (В);

Drop table – включить операторы DROP TABLE при генерации (В);

Table Check – включить операторы Check для контроля правил ограничений на уровне таблиц при генерации (В);

Physical Storage – включить объекты и характеристики физической памяти в генерируемую AllFusion Erwin Data Modeler схему (В);

Create Pre/Post Script – включить Pre/Post скрипты для таблиц перед/после генерации схемы (Н);

Create Procedure – включить хранимые процедуры для таблиц в генерируемую AllFusion Erwin Data Modeler схему (Н);

Drop Procedure – включить операторы DROP PROCEDURE для таблиц в генерируемую AllFusion Erwin Data Modeler схему (Н);

Create Alias – включить имена алиасов для таблиц в генерируемую AllFusion Erwin Data Modeler схему (Н);

Drop Alias – включить операторы, которые отменяют ранее определенные алиасы для таблиц в генерируемую AllFusion Erwin Data Modeler схему (Н).

Опции **«Column»** позволяют добавить условия ограничений в операторы определения данных таблиц в процессе генерации схемы. К ним относятся:

Column Check – позволяет включить в операторы утверждения по контролю ограничений для всех столбцов (В);

Physical Order – установить физический порядок следования столбцов при генерации схемы (В);

Default Value – включить в операторы утверждения для установления столбцам значений по умолчанию (В);

Use Distinct Type – включить пользователем определенный тип для столбца в процессе генерации схемы (Н).

Create-опции **«Index»** позволяют Администратору управлять созданием и хранением индексов, указывают, какие ключевые атрибуты образуют индексы. К ним относятся:

Primary Key (PK) – создать индексы PK для каждой таблицы (Н);

Alternate Key (AK) – создать индексы AK для каждой таблицы (В);

Foreing Key (FK) – создать индексы FK для каждой таблицы (Н);

Inversion Entry (IE) – создать индексы IE для каждой таблицы (В);

CLUSTER – создать кластерные индексы (В);

Physical Storage – включить информацию о физической памяти для индексов в схему (В);

Drop PK/AK/FK/IE – включить соответствующие операторы DROP INDEX PK/AK/FK/IE для удаления индексов в схему (Н).

Опции **«Referential integrity»** предоставляют возможность управлять обработкой связанных записей в таблицах Parent/Child при изменении или удалении ключевого поля. Могут быть выбраны следующие режимы:

Primary Key (PK) – установить уникальную идентификацию каждой строки в таблице (В);

Foreing Key (FK) – установить специфицированные (указанные) правила относительной целостности, когда значение в поле FK обновляется (В);

Unique (AK) – установить правила относительной целостности, требующие, чтобы значение альтернативных ключей были уникальными (В);

On Delete/Update – включает выбранный режим обработки относительной целостности соответственно при удалении/обновлении значения в поле PK или FK (В).

Опции **«Other Options»** предоставляют возможность добавить условия ограничений в операторы определения данных таблиц в процессе генерации схемы. К ним относятся:

Constraint Name – включает имена ключей PK или AK и ограничений для проверки их уникальности в операторы CREATE TABLE (ALTER TABLE) при генерации схемы (B);

Comment – включает комментарии для таблиц и столбцов в генерируемую схему (B);

Owner – включает параметр «имя схемы, содержащей таблицу» в операторы CREATE TABLE при генерации (B) (рис. 4.9, 4.10).

Генерация операторов на языке DDL производится при открытой модели БД, при этом необходимо выбрать тип «Physical» для модели. Перед генерацией необходимо выполнить выбор типа БД и ее версии для сервера назначения (см. рис. 4.9, «Выбор версии БД»).

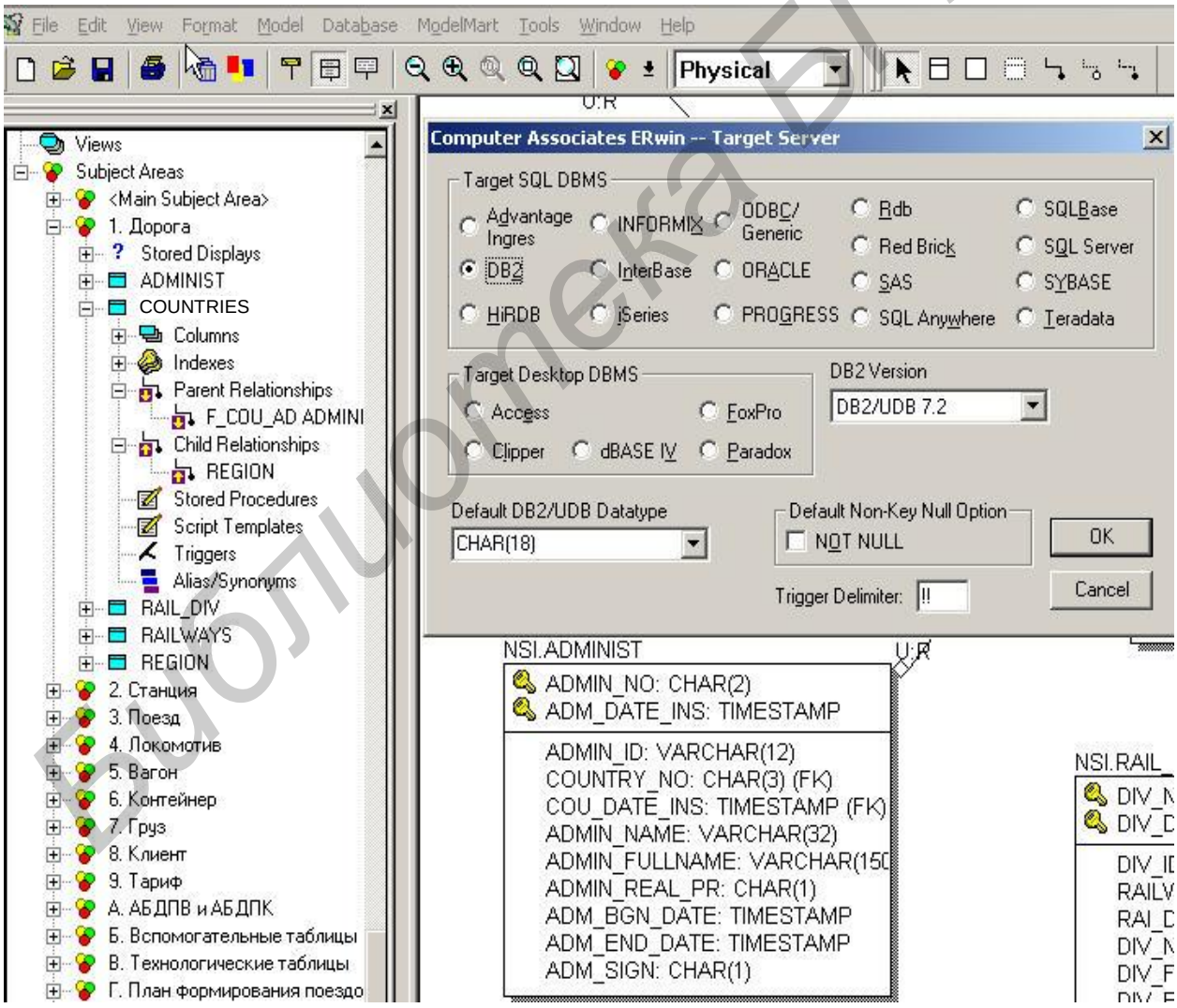

Рис.4.9

Для этого в меню «Database» выбрать «Choose Database», в окне «Target Server» включить кнопки «DB2» и «DB2 version» – выбрать соответствующую версию DB2 для дальнейшей генерации операторов DDL. Следует заметить, что эту операцию необходимо выполнить только первоначально при генерации задания на языке DDL первый раз либо при смене платформы СУБД или ее версии в дальнейшем, так как AllFusion Erwin Data Modeler запоминает предыдущее состояние введенных параметров и использует их при следующем входе.

Далее для продолжения операции получения файла задания на языке DDL необходимо перейти в меню «Tools», подменю «Forward Engineer/Schema generation» – появится окно, изображенное на рис. 4.10. Для указанных в окне «Schema generation» режимов «Schema», «View», «Table», «Columns», «Index», «Referential Integrity», «Trigger», «Other options» необходимо справа в окне «Schema» указать необходимость включения соответствующих последовательностей SQL-операторов в файл задания на языке DDL.

Это операторы Create, Drop и другие, которые будут сгенерированы AllFusion Erwin Data Modeler автоматически для схемы в целом, для табличных пространств и буферных пулов, таблиц, индексов, триггеров и других элементов схемы. Полностью состояние включения элементов в операторы на языке DDL можно посмотреть, щелкнув по кнопке «Summary».

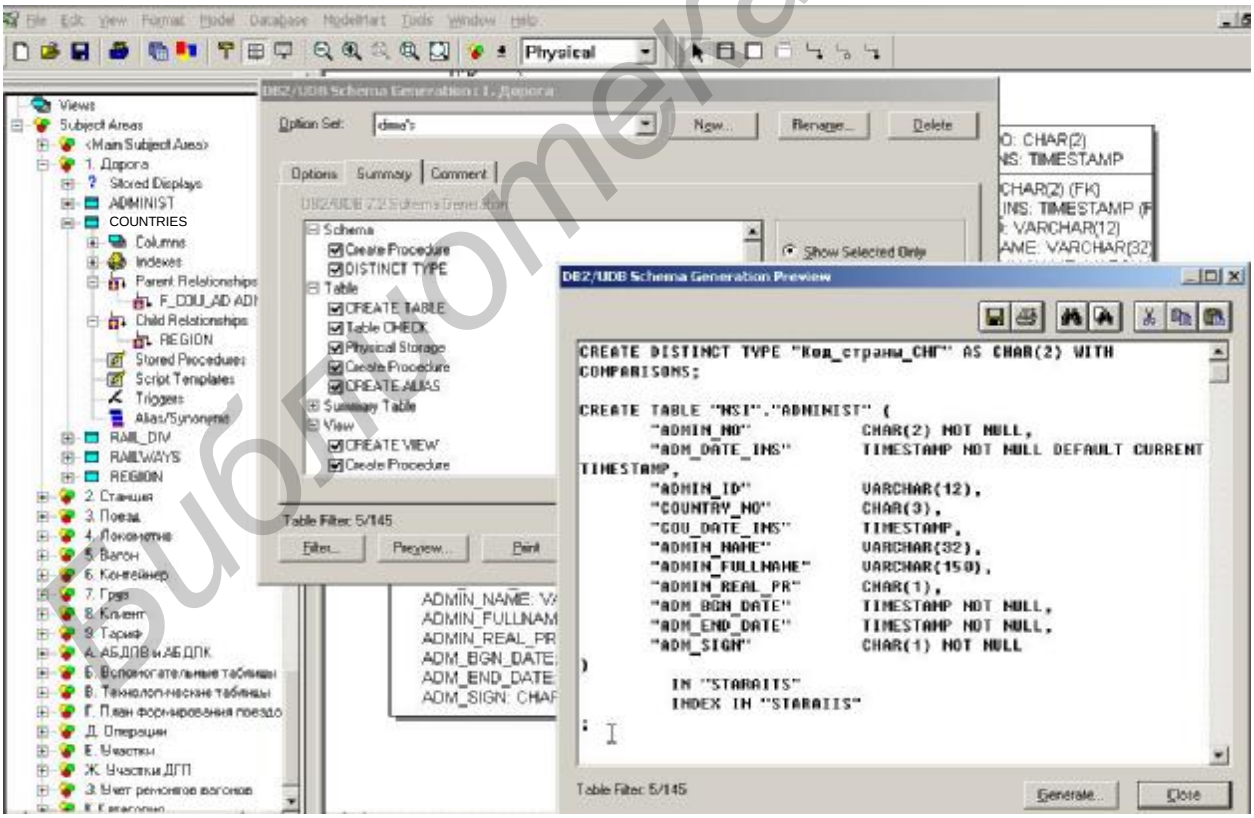

Рис. 4.10

Кроме этого, разработчик может внести свои комментарии для документирования состояния каждого переключателя операторов на языке DDL и ведения истории процесса генерации схемы.

Наряду с выполнением вышеописанных требований поддержки архитектуры таблиц при построении схемы БД можно определить некоторые дополнительные необязательные рекомендации для этапа формирования операторов DDL. Выполнение некоторых из них достигается путем переключения состояния режимов вышеописанных опций генерации схемы.

Как было отмечено выше, AllFusion Erwin Data Modeler запоминает предыдущее состояние введенных параметров переключения операторов DDL для схемы и использует их при следующем входе в эту операцию.

Для завершения генерации файла скриптов и получения операторов DDL в виде файла-задания необходимо нажать кнопку «Report» в нижней части экрана. При появлении окна, изображенного на рис. 4.11, разработчик должен выбрать папку и указать имя файла для сохранения сгенерированных операторов DDL в виде задания (сеанса в терминах DB2) для создания БД на сервере.

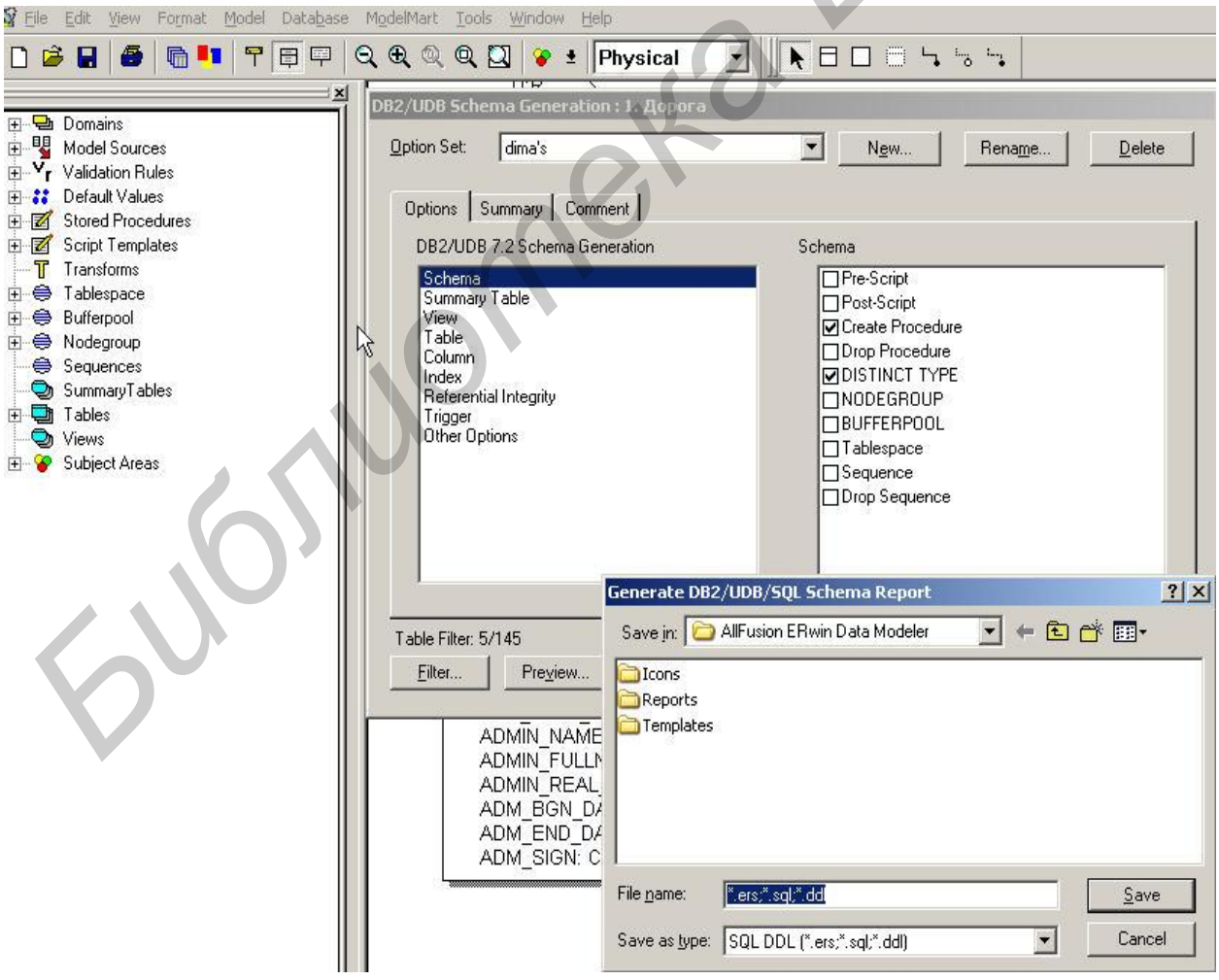

Рис. 4.11

Сгенерированные операторы можно просмотреть, нажав кнопку «Preview» в нижней части экрана (прил. 1, функция 9 – просмотр DDL). При этом используются сохраненные на этапе генерации значения параметров для переключения режимов генерации операторов DDL схемы (см. пример ниже).

```
CREATE TABLE "NSI"."ADMINIST" (
    "ADMIN_NO"          CHAR(2) NOT NULL,
    "ADM_DATE_INS"              TIMESTAMP NOT NULL DEFAULT
CURRENT TIMESTAMP,
    "ADMIN_ID"          VARCHAR(12),
    "COUNTRY_NO"        CHAR(3),
    "COU_DATE_INS"      TIMESTAMP,
    "ADMIN_NAME"        VARCHAR(32),
    "ADMIN_FULLNAME"    VARCHAR(150),
    "ADMIN_REAL_PR"     CHAR(1),
    "ADM_BGN_DATE"      TIMESTAMP NOT NULL,
    "ADM_END_DATE"      TIMESTAMP NOT NULL,
    "ADM_SIGN"          CHAR(1) NOT NULL
)
  IN "STARAITS"
  INDEX IN "STARAIIS"
;
ALTER TABLE "NSI"."ADMINIST"
  DATA CAPTURE CHANGES
 ;
COMMENT ON TABLE "NSI"."ADMINIST" IS 'Администрация';
COMMENT ON COLUMN "NSI"."ADMINIST"."ADMIN_NO" IS 'Код
администрации';
COMMENT ON COLUMN "NSI"."ADMINIST"."ADM_DATE_INS" IS 'Дата
вставки записи в таблицу ADMINIST';
COMMENT ON COLUMN "NSI"."ADMINIST"."ADMIN_ID" IS 'Мнемокод
администрации';
COMMENT ON COLUMN "NSI"."ADMINIST"."COUNTRY_NO" IS 'Код
страны';
COMMENT ON COLUMN "NSI"."ADMINIST"."COU_DATE_INS" IS 'Дата
вставки записи в таблицу COUNTRIES';
COMMENT ON COLUMN "NSI"."ADMINIST"."ADMIN_NAME" IS
'Наименование администрации';
COMMENT ON COLUMN "NSI"."ADMINIST"."ADMIN_FULLNAME" IS
'Полное наименование администрации';
COMMENT ON COLUMN "NSI"."ADMINIST"."ADMIN_REAL_PR" IS
'Признак реального кода администрации';
COMMENT ON COLUMN "NSI"."ADMINIST"."ADM_BGN_DATE" IS
'Дата и время начала действия';
```

```
COMMENT ON COLUMN "NSI"."ADMINIST"."ADM_END_DATE" IS 'Дата и время окончания действия';
COMMENT ON COLUMN "NSI"."ADMINIST"."ADM_SIGN" IS 'Признак состояния записи';
CREATE UNIQUE INDEX "NSI"."H_ADM_NO" ON "NSI"."ADMINIST"
(
    "ADMIN_NO"              ASC,
    "ADM_BGN_DATE"          ASC,
    "ADM_END_DATE"          ASC
);

COMMENT ON INDEX "NSI"."H_ADM_NO" IS 'Индекс историчности объекта';

CREATE INDEX "NSI"."F_COUADM" ON "NSI"."ADMINIST"
(
    "COUNTRY_NO"            ASC,
    "COU_DATE_INS"          ASC
);

COMMENT ON INDEX "NSI"."F_COUADM" IS 'Ссылка на "Страны"';

CREATE INDEX "NSI"."I_ADM_NO" ON "NSI"."ADMINIST"
(
    "ADMIN_NO"              ASC,
    "ADM_SIGN"              ASC
);

COMMENT ON INDEX "NSI"."I_ADM_NO" IS 'Индекс поколения объекта';

ALTER TABLE "NSI"."ADMINIST"
    ADD CONSTRAINT "P_K_ADM_" PRIMARY KEY ("ADMIN_NO",
      "ADM_DATE_INS");

CREATE TABLE "NSI"." COUNTRIES " (
    "COUNTRY_NO"        CHAR(3) NOT NULL,
    "COU_DATE_INS"      TIMESTAMP NOT NULL DEFAULT CURRENT TIMESTAMP,
    "REGION"            CHAR(1),
    "REG_DATE_INS"      TIMESTAMP,
    "COUNTRY_ID"        VARCHAR(12),
    "COUNTRY_NAME"      VARCHAR(32),
    "COUNTRY_FULLNAME"  VARCHAR(150),
```

```
        "COU_BGN_DATE"      TIMESTAMP NOT NULL,
        "COU_END_DATE"      TIMESTAMP NOT NULL,
        "COU_SIGN"          CHAR(1) NOT NULL
    )
      IN "STARAITS"
      INDEX IN "STARAIIS"
    ;

    ALTER TABLE "NSI"." COUNTRIES "
      DATA CAPTURE CHANGES
     ;
    COMMENT ON TABLE "NSI"."COUNTRIES" IS 'Страна';
    COMMENT ON COLUMN "NSI"."COUNTRIES "."COUNTRY_NO" IS 'Код страны';
    COMMENT ON COLUMN "NSI"."COUNTRIES "."COU_DATE_INS" IS 'Дата вставки записи в таблицу COUNTRIES';
    COMMENT ON COLUMN "NSI"."COUNTRIES"."REGION" IS 'Код региона';
    COMMENT ON COLUMN "NSI"."COUNTRIES"."REG_DATE_INS" IS 'Дата вставки записи в таблицу REGION';
    COMMENT ON COLUMN "NSI"."COUNTRIES"."COUNTRY_ID" IS 'Мнемокод страны';
    COMMENT ON COLUMN "NSI"."COUNTRIES"."COUNTRY_NAME" IS 'Наименование страны';
    COMMENT ON COLUMN "NSI"."COUNTRIES"."COUNTRY_FULLNAME" IS 'Полное наименование страны';
    COMMENT ON COLUMN "NSI"."COUNTRIES"."COU_BGN_DATE" IS 'Дата и время начала действия';
    COMMENT ON COLUMN "NSI"."COUNTRIES"."COU_END_DATE" IS 'Дата и время окончания действия';
    COMMENT ON COLUMN "NSI"."COUNTRIES"."COU_SIGN" IS 'Признак состояния записи';
    CREATE UNIQUE INDEX "NSI"."H_COU_NO" ON "NSI"."COUNTRIES"
    (
        "COUNTRY_NO"            ASC,
        "COU_BGN_DATE"          ASC,
        "COU_END_DATE"          ASC
    );

    COMMENT ON INDEX "NSI"."H_COU_NO" IS 'Индекс историчности объекта';

    CREATE INDEX "NSI"."F_REGION" ON "NSI"."COUNTRIES"
    (
```

```
    "REGION"                ASC,
    "REG_DATE_INS"              ASC
);

COMMENT ON INDEX "NSI"."F_REGION" IS 'Ссылка на "Регионы"';

CREATE INDEX "NSI"."I_COU_NO" ON "NSI"."COUNTRIES"
(
    "COUNTRY_NO"            ASC,
    "COU_SIGN"              ASC
);

COMMENT ON INDEX "NSI"."I_COU_NO" IS 'Индекс поколения объекта';

ALTER TABLE "NSI"."COUNTRIES"
    ADD CONSTRAINT "P_K_COU_" PRIMARY KEY ("COUNTRY_NO",
        "COU_DATE_INS");

ALTER TABLE "NSI"."REGION"
    ADD CONSTRAINT "P_K_REGI" PRIMARY KEY ("REGION",
        "REG_DATE_INS");

ALTER TABLE "NSI"."ADMINIST"
    ADD CONSTRAINT "F_COU_AD"
        FOREIGN KEY ("COUNTRY_NO", "COU_DATE_INS")
                REFERENCES "NSI"."COUNTRIES"
                ON DELETE RESTRICT
                ON UPDATE RESTRICT;

ALTER TABLE "NSI"."COUNTRIES"
    ADD CONSTRAINT "F_REG_CO"
        FOREIGN KEY ("REGION", "REG_DATE_INS")
                REFERENCES "NSI"."REGION"
                ON DELETE RESTRICT
                ON UPDATE RESTRICT;
```

Результатом работы на данном этапе является выпуск очередной версии схемы модели БД и оформление ее в документированном виде. Этот вид обеспечивается развитыми в AllFusion Erwin Data Modeler удобными средствами формирования, корректировки, отображения и выдачи отчетов для логической и физической схем БД.

Полная схема БД позволяет AllFusion Erwin Data Modeler сгенерировать операторы на языке описания данных DDL, которые в свою очередь используются в качестве задания – входного потока для создания пустой физической БД на сервере системы средствами СУБД, например DB2 UDB.

4.8. Подготовка исходных данных для разработки новой версии БД

Исходные данные для получения новой версии БД определяются требованиями появляющихся новых функций и модификации задач по мере их развития. Назначение этапа – проверка и обеспечение достоверности схемы модели БД, проверка правильности исходных данных для корректировки и соответствия их схеме модели БД.

Исходные данные для внесения изменений и получения новой версии БД включают:

схему эксплуатируемой предыдущей версии БД в среде AllFusion Erwin Data Modeler (по возможности) или структуру таблиц данных конкретной БД (при первоначальном формировании схемы);

совокупность отчетов по старой схеме БД в среде AllFusion Erwin Data Modeler;

эксплуатируемую физическую БД;

исходные данные для изменений по форме некоторого установленного документа пользователя;

источник ввода данных и порядок их заполнения, порядок установления связей с другими таблицами (для вновь включаемых таблиц, столбцов, связей должен быть определен).

Формы документов, предназначенных для документального оформления требований по внесению изменений в эксплуатирующуюся БД, получению исходных данных для заполнения БД и выпуску новой ее версии, регламентируются внутренними распоряжениями или документами пользователя. Рекомендуется по возможности приближать вид этих документов к отчетам AllFusion Erwin Data Modeler для схем моделей БД.

Функциональная схема технологического процесса этапа подготовки исходных данных для ввода в эксплуатацию новой версии БД представлена на рис. 4.12.

На данном этапе можно выполнить определенные подготовительные работы, которые позволят в дальнейшем сократить время и улучшить качество работ по созданию и запуску новой версии БД. Необходимость их выполнения определяется разработчиком БД в зависимости от того, есть ли достоверная схема БД на данный момент, насколько она соответствует физической БД, насколько велик список изменений в БД и насколько тщательно подготовлены исходные данные для корректировки.

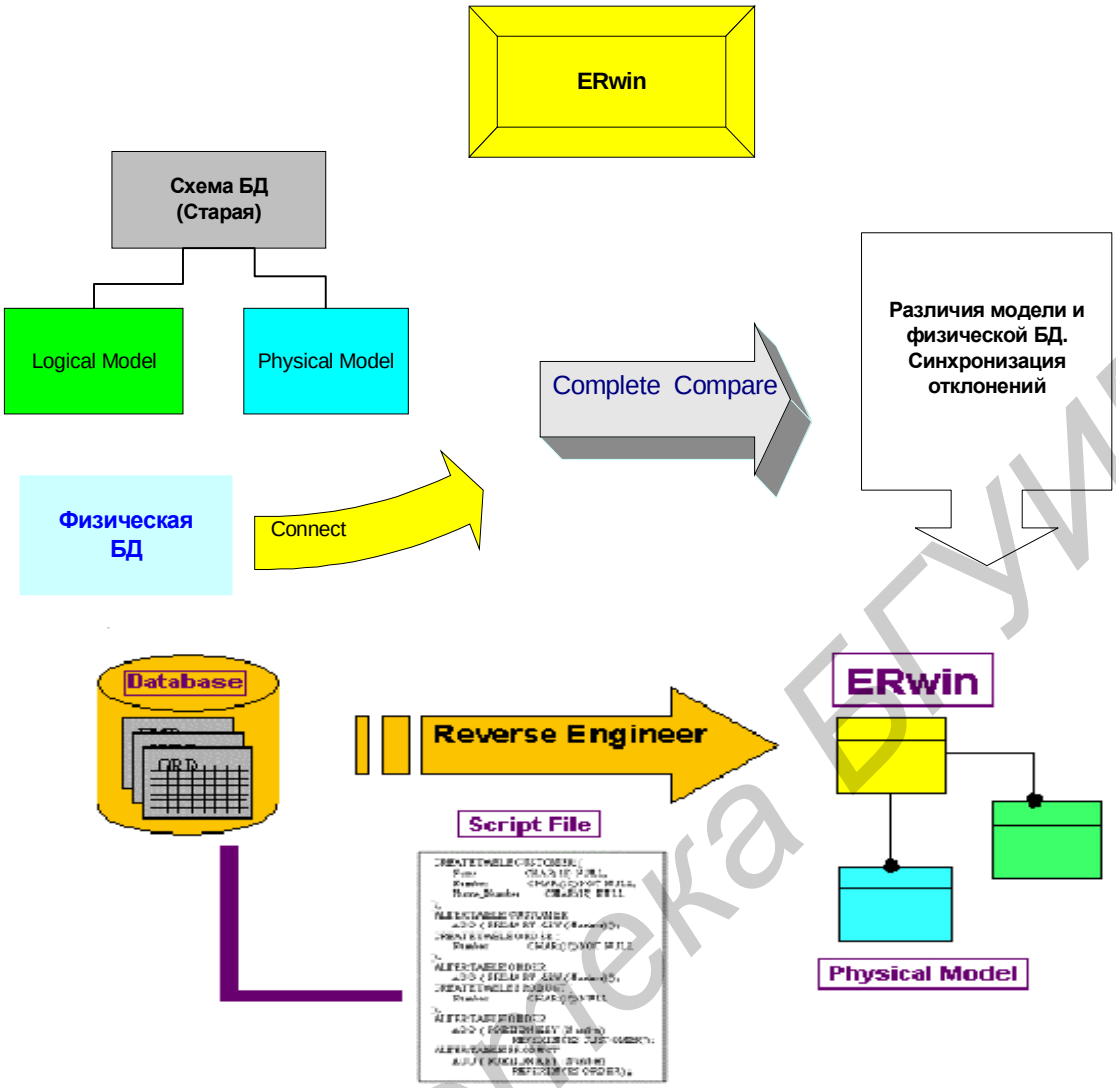

Рис. 4.12

Предлагается выполнить следующий перечень операций, при этом в скобках делается ссылка на номер пункта из приведенных в прил. 1 функций AllFusion Erwin Data Modeler по проектированию схемы БД (см. прил. 1):

запустить CASE-систему AllFusion Erwin Data Modeler;

загрузить существующую модель старой БД в AllFusion Erwin Data Modeler (прил. 1, п. 8) или получить ее средствами AllFusion Erwin Data Modeler, операция «Reverse engineering» (прил. 1, п. 11);

получить доступ к эксплуатируемой БД;

произвести полное сравнение существующей модели БД с физической БД средствами AllFusion Erwin Data Modeler для контроля их соответствия (прил. 1, п. 12);

при наличии отклонений произвести выяснение причин различий и привести в соответствие схемы модели БД со структурой физической БД (прил. 1, пп. 2–7);

получить полную логическую и физическую схемы БД со всеми ее подсхемами, таблицами, столбцами и их характеристиками (прил. 1, пп. 8–15);

произвести анализ исходных данных для внесения изменений и определения оптимального варианта их включения в новую версию БД;

на основе анализа получить более полный исправленный и детализированный перечень вносимых изменений в БД по структуре отчетов AllFusion Erwin Data Modeler для БД (вставка, замена, удаление элементов схемы).

Ниже в качестве примера приводится последовательность действий разработчика для выполнения функции 11 – «Reverse engineering», создание модели данных из существующей БД или файла операторов DDL, сгенерированного средствами СУБД для этой БД. Эта функция позволяет существенно сократить время выпуска новой версии БД и ускорить процесс создания новой версии системы в целом.

Вызвается эта функция из меню «Tools», выбирается операция «Reverse engineering» (рис. 4.13). В диалоговом окне выбирается тип «Physical» или «Logical/ Physical» для модели, в окне «Target Database», необходимо выбрать соответствующую СУБД и ее версию: нажав кнопку «Database» выбрать «DB2», и кнопку «Version» – соответствующую версию DB2 для дальнейшей генерации операторов DDL. Затем нажать кнопку «Next», появляется диалоговое окно «Reverse engineering – Set options» для установки режимов (рис. 4.14) «Reverse engineering».

Операция «Reverse engineering» выполняется на основании физической БД или файла скриптов, сгенерированного средствами СУБД для этой БД, что необходимо здесь выбрать. Необходимо также выбрать из предложенных AllFusion Erwin Data Modeler режимы «Reverse engineering» или использовать их по умолчанию.

В случае восстановления схемы из файла скриптов необходимо указать путь и имя файла SQL. При восстановлении схемы из существующей БД после нажатия кнопки «Next» появляется окно «Reverse engineering – Status» (см. рис. 4.15.). В нем необходимо указать параметры «Имя пользователя», «Пароль» и «Имя БД», после чего AllFusion Erwin Data Modeler выполняет операцию «Connect» для подключения к данной БД и проверку прав доступа пользователя к ней.

Далее AllFusion Erwin Data Modeler в течение некоторого времени выбирает требуемые данные из системных таблиц DB2 и строит модель, схема которой генерируется и появляется на экране автоматически. После этого можно сохранить эту модель – перейти в меню «File», выбрать операцию «Save» и ввести имя AllFusion Erwin Data Modeler-файла *.ER1.

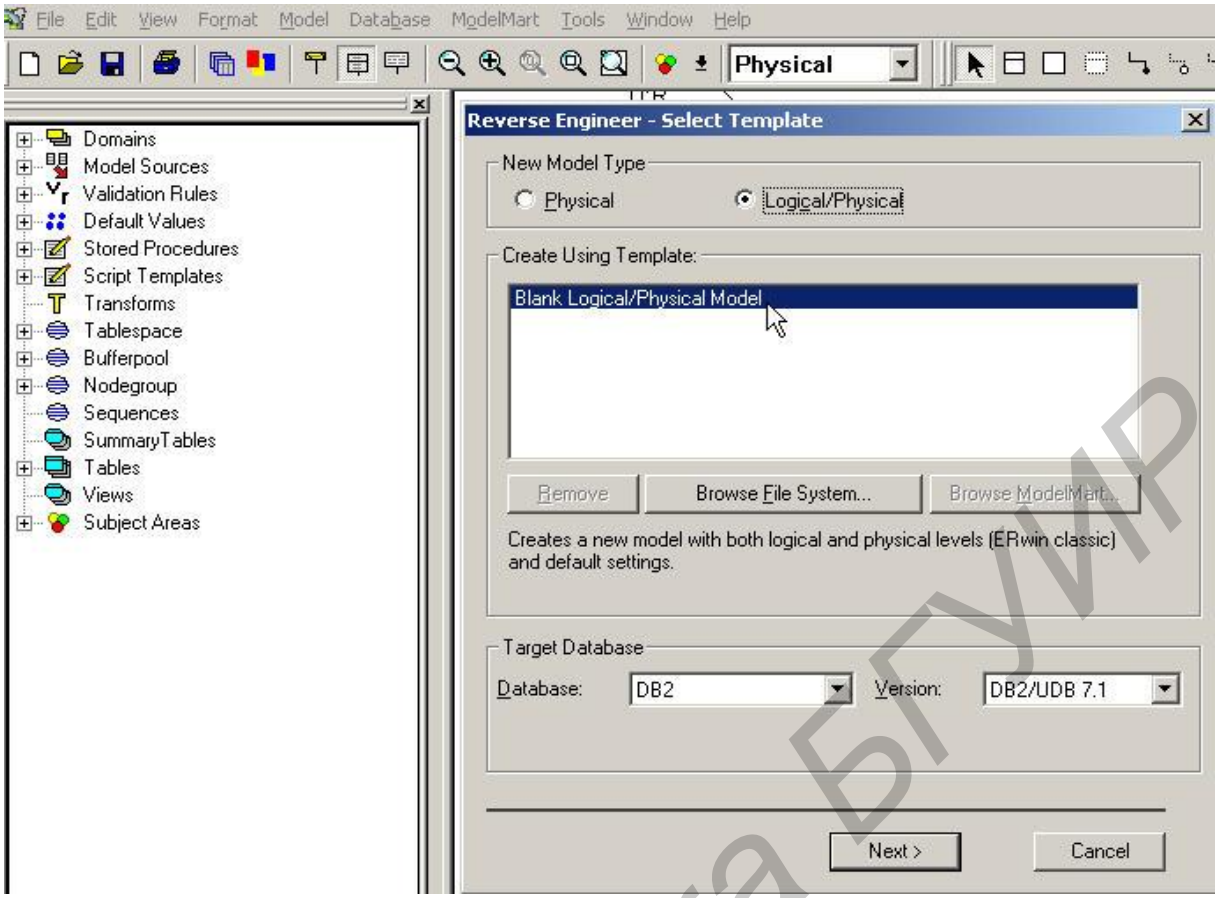

Рис. 4.13

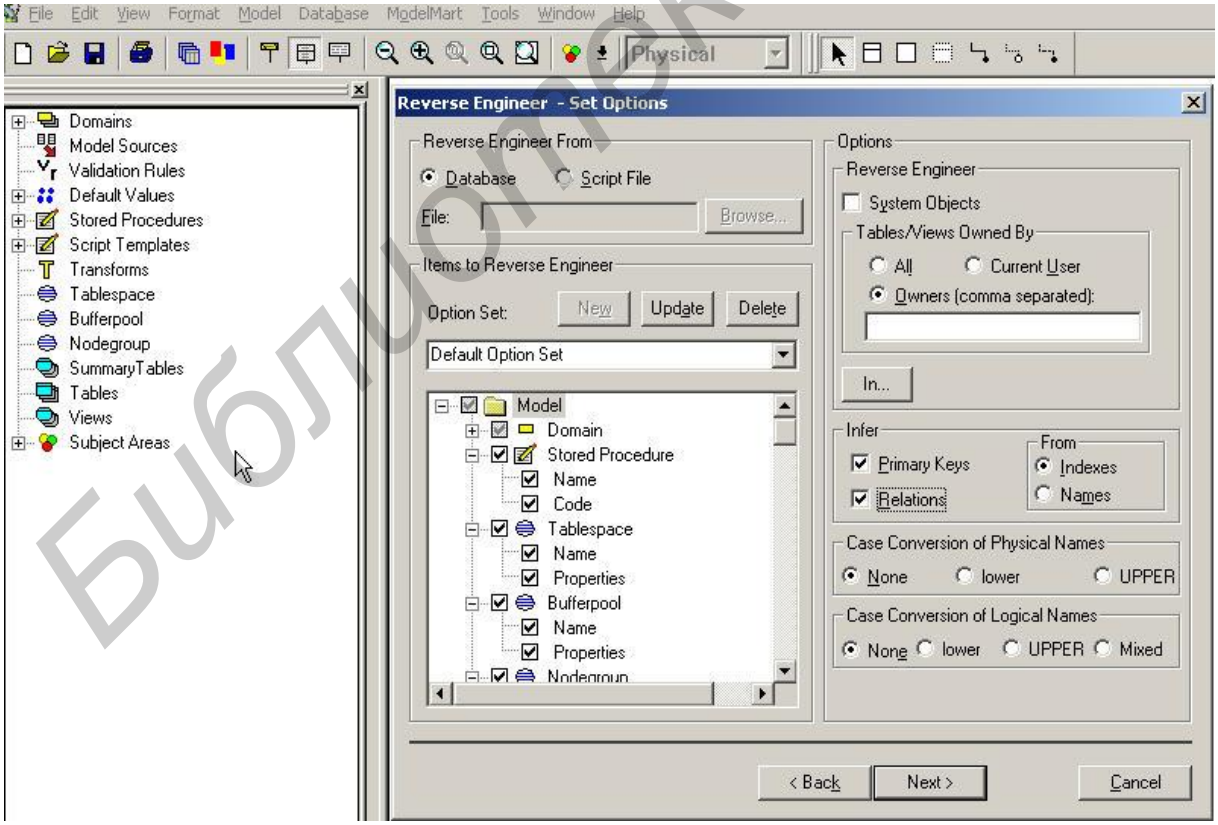

Рис. 4.14

Рис. 4.15

При выполнении операции «Reverse engineering» AllFusion Erwin Data Modeler не восстанавливает подсхем БД, так как эта информация отсутствует в физической БД, поэтому их необходимо создать заново. Для создания подсхемы необходимо сделать щелчок правой клавишей мыши по «Subject Areas» в окне «Model Explorer», выбрать опцию «New» и ввести имя подсхемы, например «Станция». Потом для этой подсхемы указать опцию «Property» и выбрать таблицы, которые включаются в подсхему из «Main Subject Area», содержащей все таблицы БД. AllFusion Erwin Data Modeler автоматически проводит синхронизацию ведения объектов и элементов всех подсхем с общей схемой БД и наоборот.

Полученная описанным выше способом схема БД со всеми ее подсхемами в дальнейшем может использоваться для внесения изменений при выпуске новой версии БД. При наличии рабочей схемы БД, соответствующей функционирующей БД, эту операцию можно и не выполнять. В этом случае можно выполнить сравнение «Complete Compare» схемы и БД. При этом AllFusion Erwin Data Modeler допускает по желанию пользователя выборочное автоматическое внесение изменений в схему БД по выявленным при сравнении отклонениям.

Общим результатом работ на данном этапе является актуализация всех параметров схемы БД, предварительная проверка соответствия ей исходных данных для внесения изменений, подготовка более детализированной информации для предстоящей корректировки схемы БД.

4.9. Проектирование новой версии БД и генерация отчетов

Информационные требования – создание модели данных БД и актуальная поддержка всех ее изменений в процессе жизненного цикла – обеспечиваются средствами CASE-системы AllFusion Erwin Data Modeler. Они обеспечивают развитие, проверку (тестирование) и утверждение модели данных, документирование и наглядность ведения модели БД. Функциональная схема процесса проектирования и выпуска новой версии БД представлена на рис. 4.16.

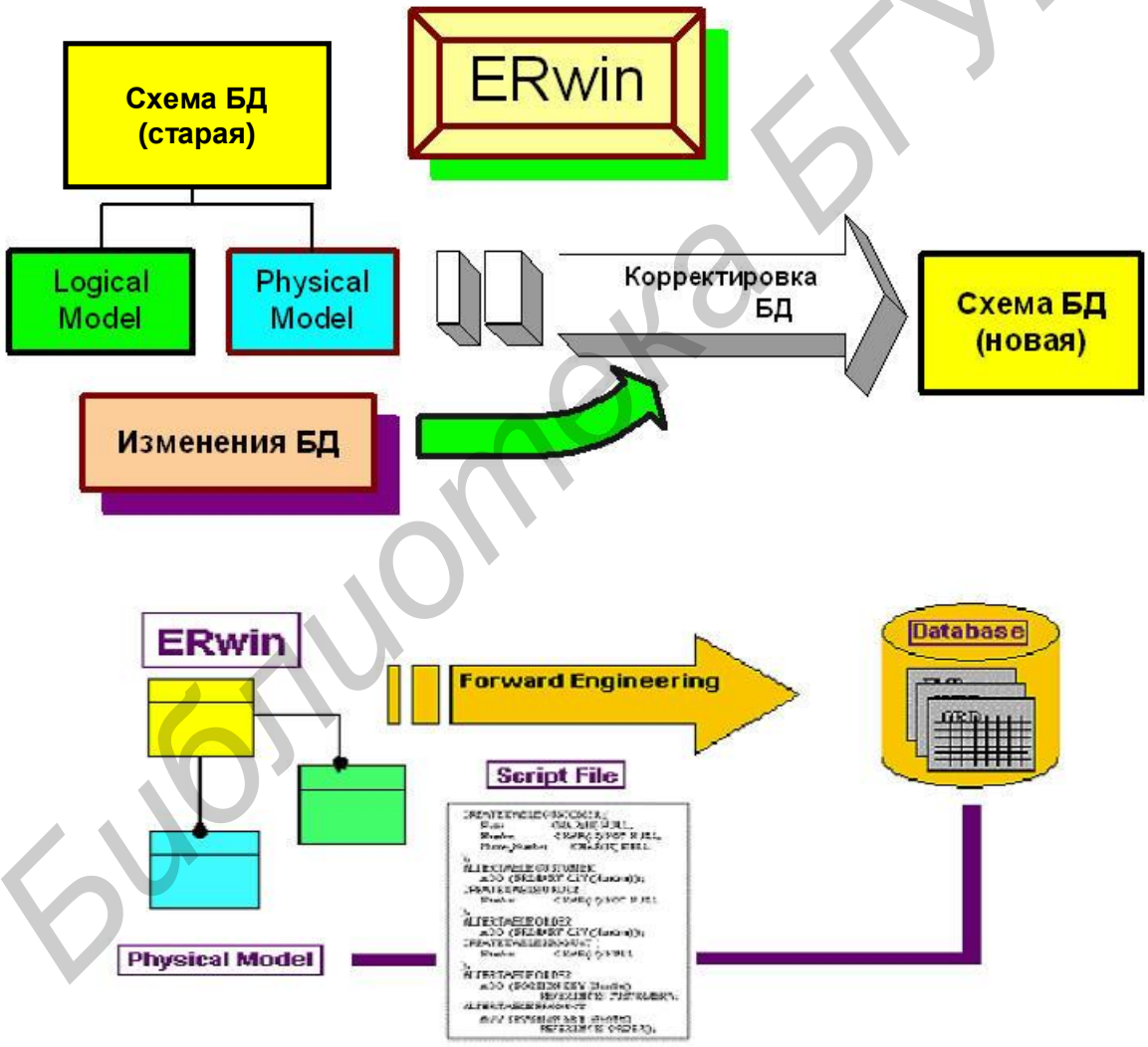

Рис. 4.16

Проектирование новой версии предполагает выполнение в среде AllFusion Erwin Data Modeler следующих работ:

получение новой модели БД на основании старой схемы (см. прил. 1, п. 8.);

выполнение редактирования старой схемы путем введения новых объектов, корректировки и/или удаления старых объектов и таблиц, получение логической схемы базы данных (прил. 1, пп. 2–5, 7);

получение физической схемы базы данных, определение ключей и установление отношений, связей между таблицами и их характеристиками: Cardinality, Referential integrity, Nulls allowed/ not allowed;

проверка схемы и структур данных новой версии (нормализация, оптимизация и т.д.);

получение и просмотр операторов DDL для создаваемой новой БД (прил. 1, п. 9);

генерацию (окончательное оформление) операторов описания данных на языке DDL, сохранение их в виде файла для создания новой БД (прил. 1, п. 10);

документирование новой версии БД путем оформления и получения средствами AllFusion Erwin Data Modeler необходимых отчетов (см. прил. 1, пп. 13–15).

При создании новой версии БД используются те же функции из прил. 1, что и при создании модели, основные из которых были рассмотрены нами ранее. Ниже приводится последовательность действий разработчика для выполнения функций пп. 13–15 из прил. 1 «Построение отчетов из модели данных». Построитель шаблонов отчетов AllFusion Erwin Data Modeler создает предварительные шаблоны отчетов в форматах HTML, RTF и TXT, которые могут быть сохранены и затем многократно использоваться для выдачи отчета по любой модели. Можно посмотреть и сохранить отчеты для общего использования с другими документами, используя стандартный веб-браузер (web-browser).

Обращение к функции формирования отчетов осуществляется из меню «Tools», выбирается операция «Report Builder» (рис. 4.17).

В окне “Report Templates” появляются построенные ранее шаблоны отчетов, которые можно выбрать для редактирования (кнопка «Edit») или запуска (кнопка «Run») и просмотра отчета. Окно «Output type» используется для выбора выходного формата создаваемого шаблона. Для создания нового шаблона необходимо нажать кнопку «New». Появляется панель «Report Template Builder» (рис. 4.18). В этой панели в правом окне «Report Layout» сначала следует дважды щелкнуть по «Document Untitled» для формирования заголовка отчета.

После появления окна «Properties» (рис. 4.19) необходимо щелкнуть по кнопке «Title», убрать слова «Document Untitled» и набрать вместо этих слов текст заголовка отчета, дополнив его в конце пробелом для сцепления заголовка с именем модели. Далее нажать кнопку «Add Macro» в нижней части окна для выполнения указанного сцепления.

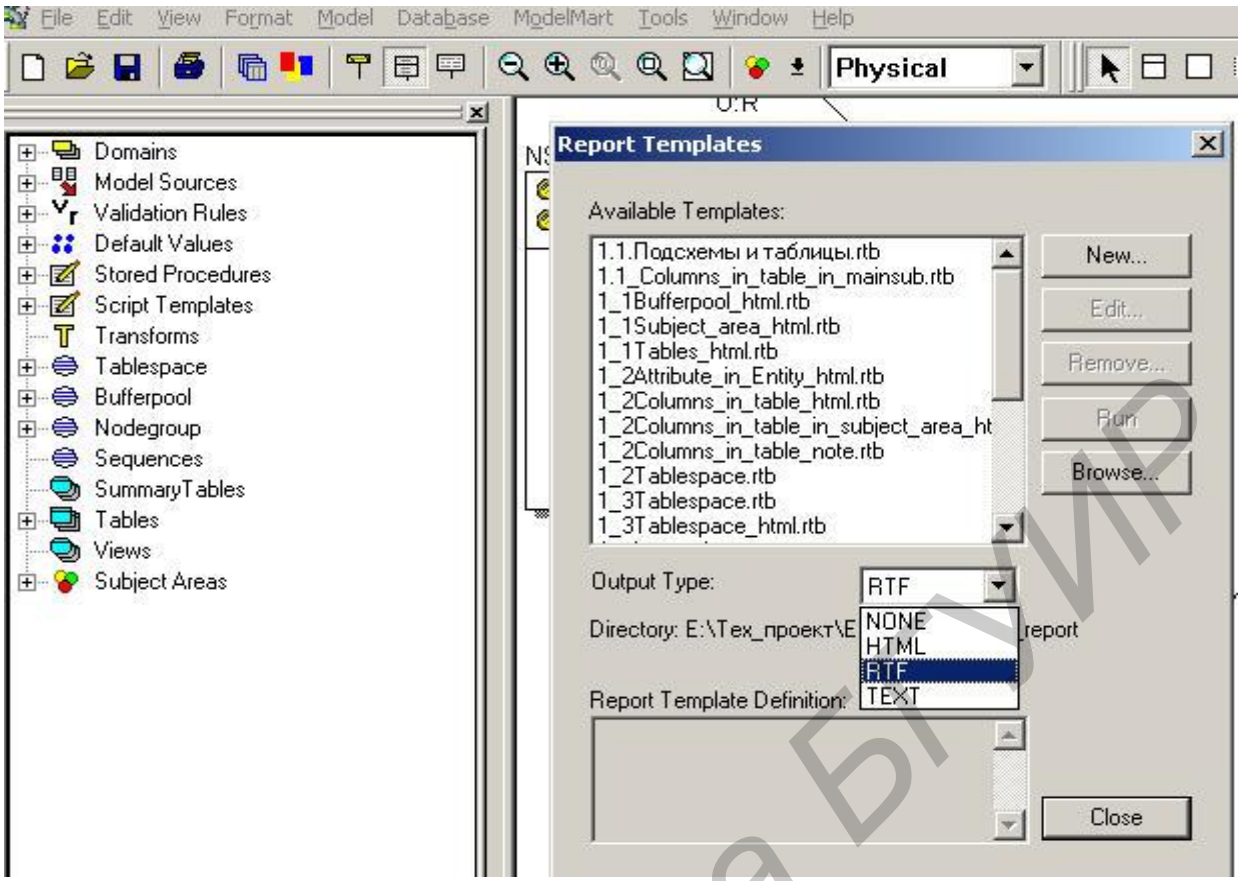

Рис. 4.17

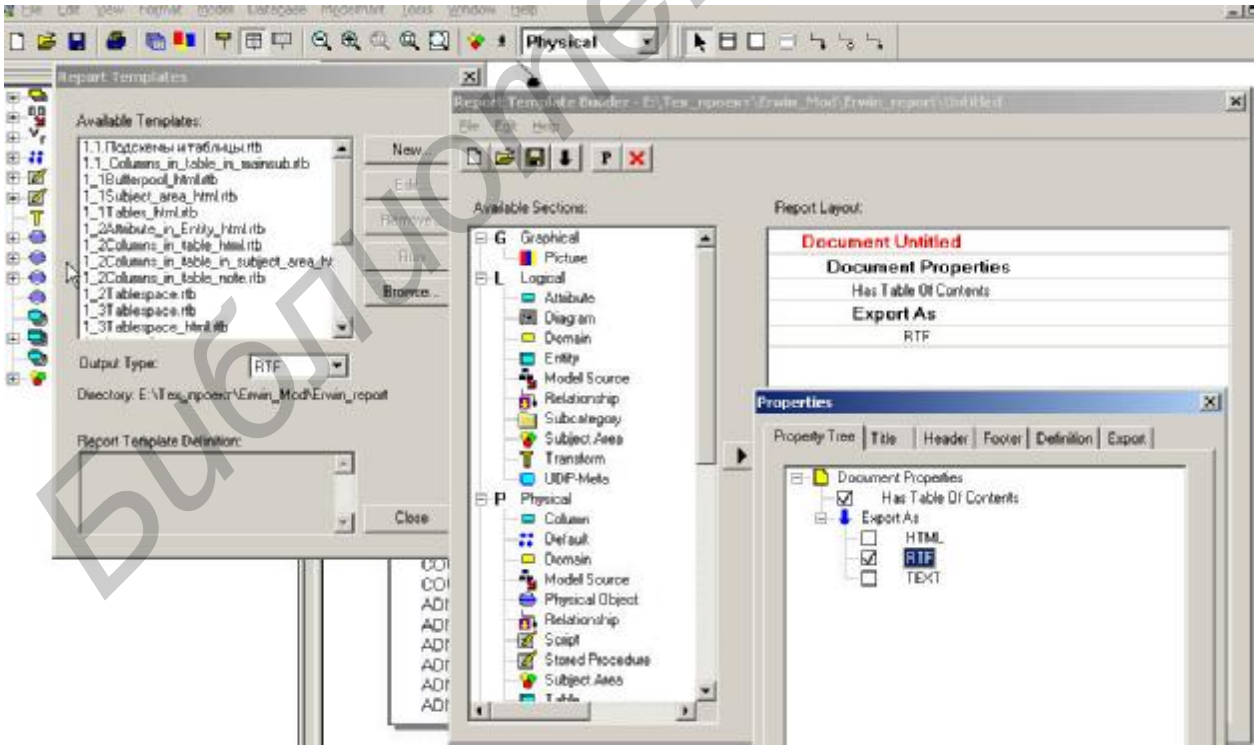

Рис. 4.18

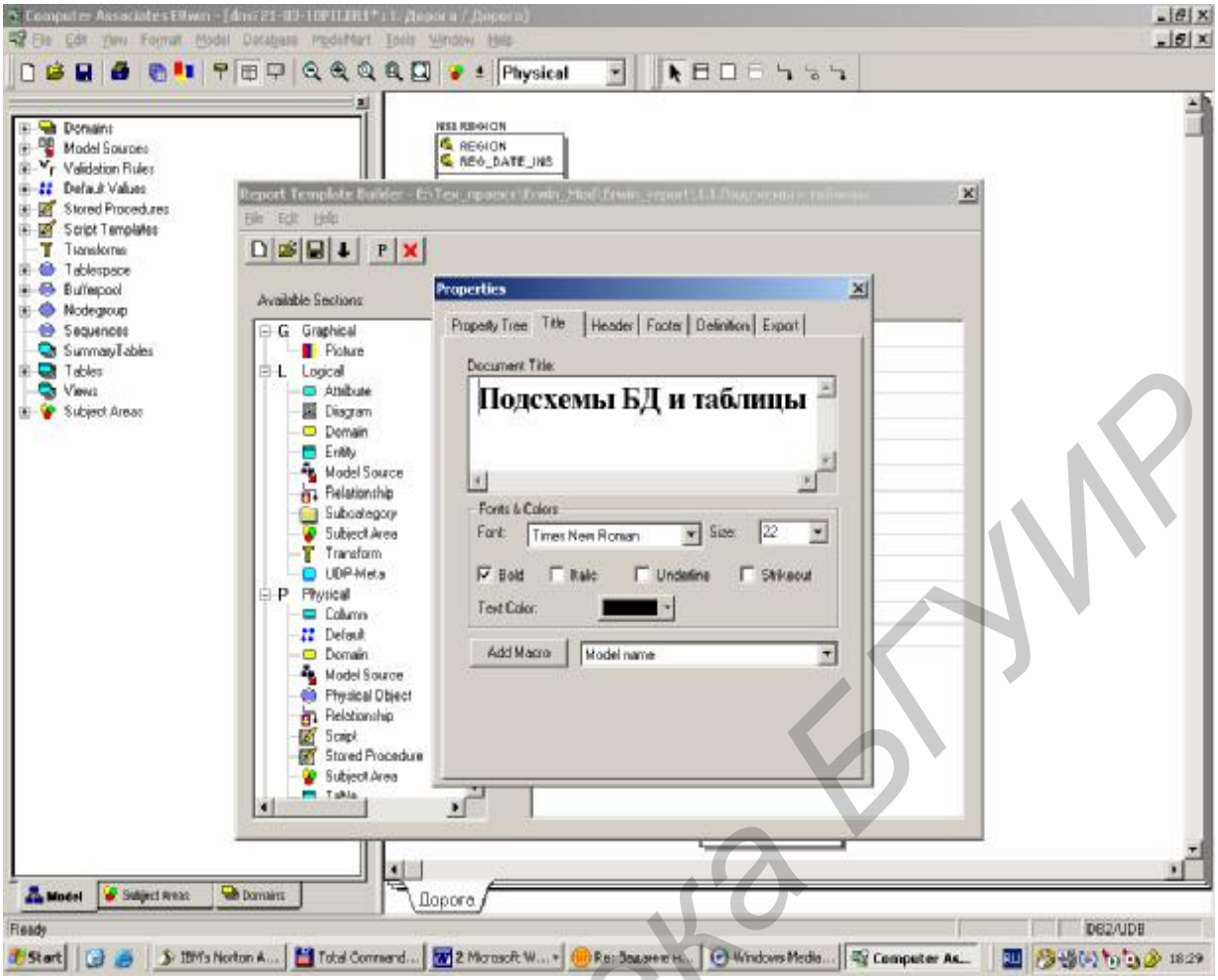

Рис. 4.19

Кроме этого, можно установить цвет заголовка и необходимо выбрать для него шрифт «Кириллица» из предлагаемого в «Font» перечня – «Times New Roman CYR». Предоставляется возможность внести дополнительную информацию в верхнюю и нижнюю части заголовка, используя кнопки «Header», «Footer». Дополнительно можно указать формат экспорта (кнопка «Export»). После формирования заголовка необходимо закрыть данное окно и возвратиться назад в панель «Report Template Builder» (см. рис. 4.18) для продолжения формирования шаблона.

Далее необходимо включить секции отчета в шаблон. Все секции отчета в окне «Available sections» распределены в следующие группы (рис. 4.20):

секция описания схемы базы данных;
группа секций описания логической схемы;
группа секций описания физической схемы;
группа секций описания проверки полноты логической схемы;
группа секций описания проверки полноты физической схемы.

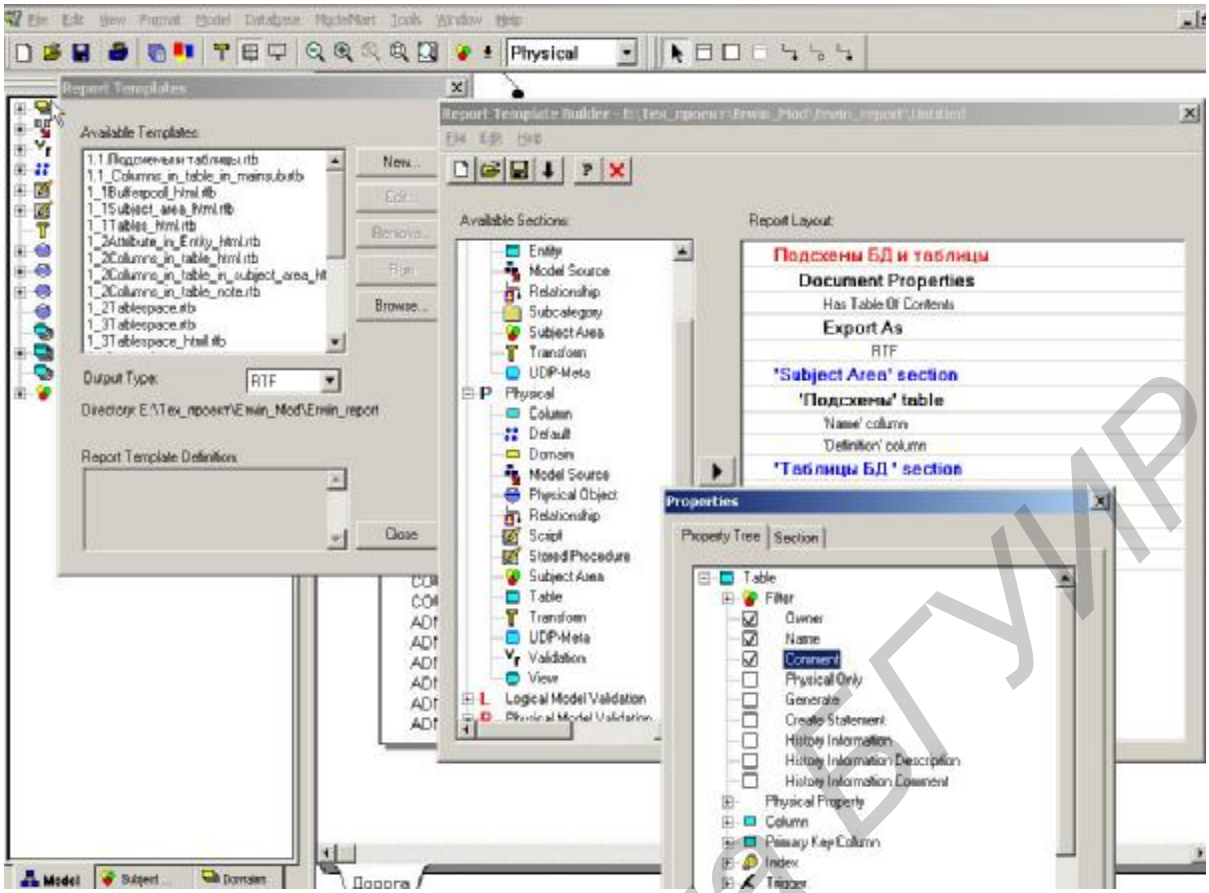

Рис.4.20

Для включения предлагаемого шаблона AllFusion Erwin Data Modeler для выбранной секции необходимо сделать правый щелчок мышью по этой секции в окне «Available sections» и при активизации кнопки «Add section» сделать левый щелчок по кнопке, подтвердив тем самым включение шаблона секции в общий шаблон для этого отчета.

После этого выбранные секции последовательно перемещаются в правое окно «Report Layout», причем для каждой секции там строится несколько строк, образовывая таким образом древовидную структуру отчета. Далее необходимо отредактировать формат каждой секции для собственного удобства пользования отчетом. Для этого в правом окне «Report Layout» необходимо дважды щелкнуть по любой строке любой секции, после чего в окне «Property» появится соответствующее дерево элементов БД. В нем необходимо нажать на знак «+» напротив соответствующей секции или ее составляющей для расширения. Далее нужно отметить включаемые в шаблон элементы, как это показано на рис. 4.20. После выборки и обработки всех требуемых секций производим сохранение шаблона, выбрав в меню «File» операцию «Save as» и введя имя файла.

Таким образом сформированный шаблон отчета готов для подключения к любой модели данных. Как указано выше, при входе в операцию через меню «Tools» при открытой схеме БД выбирается операция «Report Builder» (см. рис. 4.17). Во всплывающем окне «Report Templates», которое содержит построенные шаблоны отчетов, нужно подвести курсор и выбрать шаблон

щелчком. Далее необходимо нажатием кнопки указать операцию: редактирование (кнопка «Edit») или запуск (кнопка «Run») для просмотра данных отчета.

В прил. 1 содержится предлагаемый перечень отчетов БД, который может быть видоизменен или дополнен разработчиком БД.

В целом в результате выполнения всех работ по данному этапу получается полностью документированная схема новой версии БД в среде AllFusion Erwin Data Modeler и соответствующее ей задание с набором операторов DDL для ее создания. Ниже (табл. 4.1–4.3) приведен фрагмент отчета.

Таблица 4.1

Секция описания подсхем		
Подсхемы		
Имя	**Определение**	**Последнее изменение**
1	2	3
<Main Subject Area>		Fri Mar 10 17:47:59 2006
1. Дорога	Подсхема «Дорога»	Tue Jun 01 15:00:50 2004
2. Станция	Подсхема «Станция»	Fri Jun 17 15:51:21 2005
3. Поезд	Подсхема «Поезд»	Fri Oct 15 14:55:37 2004
4. Локомотив	Подсхема «Локомотив»	Mon Nov 28 15:49:36 2005
5. Вагон	Подсхема «Вагон»	Tue Feb 21 15:41:08 2006
6. Контейнер	Подсхема «Контейнер»	Tue Jun 01 15:03:13 2004
7. Груз	Подсхема «Груз»	Tue Feb 21 15:38:52 2006
8. Клиент	Подсхема «Клиент»	Tue Jun 01 15:03:13 2004
9. Тариф	Подсхема «Тариф»	Tue Jun 01 15:03:13 2004
А. АБДПВ и АБДПК	Подсхема «АБДПВ и АБДПК»	Tue Jun 01 15:03:13 2004
Б. Вспомогательные таблицы	Подсхема «Вспомогательные таблицы»	Tue May 24 16:04:05 2005
В. Технологические таблицы	Подсхема «Технологические таблицы»	Mon Jun 14 15:02:22 2004
Г. План формирования поездов	Подсхема «План формирования поездов»	Tue Jun 01 15:03:13 2004
Д. Операции	Подсхема «Операции»	Tue Jun 01 15:03:13 2004

Окончание табл. 4.1

1	2	3
Е. Участки	Подсхема «Участки»	Tue Jun 01 15:03:13 2004
Ж. Участки ДГП	Подсхема «Участки ДГП»	Wed Aug 24 11:49:49 2005
З. Учет ремонтов вагонов	Подсхема «Учет ремонтов вагонов»	Sat Oct 22 19:25:38 2005
И. Категория	Подсхема «Категория»	Fri Mar 10 16:26:48 2006
К. Рекурсия	Подсхема «Рекурсия»	Fri Mar 10 17:47:59 2006

Таблица 4.2

Table/View(s) of "<Main Subject Area>" Subject Area		
Owner	**Name**	**Comment**
1	2	3
NSI	ADMINIST	Администрация
NSI	BANK	Справочник банков
NSI	CARGO	Грузы по ЕТСНГ
NSI	CARGO_BELRW	Выделение грузов из основных номенклатурных групп
NSI	CARGO_BRW_DIV	Выделение грузов по отделениям
NSI	CARGO_BRW_ENTRY	Вхождение групп грузов
NSI	CARGO_DANGER	Карточка опасности груза
NSI	CARGO_DKC	Группы грузов ДКЦ
NSI	CARGO_ETSNG_GNG	Таблица соответствия кодов грузов по ГНГ кодам грузов по ЕТСНГ
NSI	CARGO_GNG	Грузы по ГНГ
NSI	CARGO_GNG_OLD	Грузы по ГНГ (до 2006)
NSI	CARGO_GR	Тарифные группы грузов по ЕТСНГ
NSI	CARGO_OPER	Груз по оперативной номенклатуре
NSI	CARGO_R	Разделы грузов по ЕТСНГ

Продолжение табл. 4.2

1	2	3
NSI	CARGO_SELECT	Наименование группы кодов грузов для БЧ
NSI	CARGO_SG	Подгруппы грузов по ЕТСНГ
NSI	CARGO_SG_GNG	Таблица соответствия кодов грузов по ГНГ подгруппам грузов по ЕТСНГ
NSI	CITY	Города
NSI	CLASS_RW_OPER	Операции с объектами ЖД
NSI	CONTAINER	Контейнеры
NSIPVK	CONTAINER_CORR	Данные о приписке контейнеров
NSI	CONTAINER_KIND	Виды контейнеров
NSI	COUNTRIES	Страна
NSI	CURRENCY	Курсы валют
NSI	CUSTOMER	Клиенты
NSICTRL	DATA	Данные неключевых столбцов
NSI	DEFECT_KIND	Виды неисправностей вагона
NSI	DGP	ДГП
	EMPLOYEE	
NSI	FORW_TREATY	Договоры с экспедиторами
NSI	FORWARDER	Экспедиторы
NSI	FRONT	Фронт станции
NSICTRL	INFORM	Описание вносимых в НСИ изменений
NSI	INTER_JOIN	Справочник межгосударственных стыков
NSI	INTRAOFFICE_OBJECT	Перечень внутристанционных объектов
NSI	JOINT	Справочник стыков
NSI	JOINT_PRIZN	Признак стыка
NSICTRL	KEYS_DATA	Данные ключевых столбцов операции

Продолжение табл. 4.2

1	2	3
NSICTRL	KEYS_NEW_DATA	Новые данные ключевых столбцов (только для «UK» операции)
NSI	LOK_AREA	Участки обращения локомотивов
NSI	LOK_AREA_SECTORS	Принадлежность NBE участку обращения локомотивов
NSI	LOK_DEPO	Локомотивные депо
NSI	LOK_GR_OPERAT	Коды групп операций ЛМД
NSI	LOK_NORM_REPAIR	Нормы простоя в ремонтах по сериям локомотивов и депо
NSI	LOK_PARK_CAT	Учетные категории парка локомотивов
NSI	LOK_PARK_TRAFFIC	Привязка категории парка локомотивов к состоянию и виду движения локомотива
NSI	LOK_SERIES	Серии локомотива и нормы пробега (ТПС) М.476
NSI	LOK_STATE	Состояние локомотива
NSI	LOK_STATE_OPER	Таблица соответствия кодов состояния локомотива кодам операций, используемых в системе ИАС ПУР ГП
NSI	LOK_TRACT_SG	Категории локомотива (подгруппы видов тяги)
NSI	LOK_TRACTION	Виды тяги
NSI	LOK_TYPE_SECTION	Типы секций локомотивов
NSI	LOKOMOTIV	Номера локомотивов (соответствие новой и старой нумерации)
NSILOG	MODIFICATION	Изменения
NSI	OBJECT	Объект станции

Продолжение табл. 4.2

1	2	3
NSI	OPER_CLASS	Объекты, с которыми могут быть совершены операции в системе ИАС ПУР ГП
NSI	OPERAT_DEPARTMENT	Перечень производственных подразделений (ВСЗ, ВРЗ, ВЧД)
NSICTRL	OPERATION	Технологическая таблица операций
NSI	PARAGRAF	Параграф станции
NSI	PARK_TYPE	Тип парка вагонов
NSI	PARK_TYPE_IDLE	Типы вагонов нерабочего парка
NSI	PLF_COURSE	Таблица определения элементарного назначения ПФ по станции назначения вагона
NSI	PLF_ENTRY	Таблица назначений поездов, принимаемых на дорогу
NSI	PLF_OUT	Определение кода ЕСР (стыка сдачи)
NSI	PROFILE_USER	Массив абонентов задачи «Архив вагонов»
NSI	PROPERTY	Вид собственности вагонов
NSI	RAIL_DIV	Отделение дороги
NSI	RAILWAYS	Дорога
NSI	REF_SEC_RANGE	Диапазонный справочник номеров рефрижераторных секций
NSI	REGION	Регион
NSI	RW_OPER	Операции системы ИАС ПУР ГП
NSI	SECTORS	Участки NBE

Окончание табл. 4.2

1	2	3
NSI	SENDER_REPL_DEPO	Таблица соответствия кодов локомотивных депо отправителей сообщения кодам локомотивных депо совершения операций в сообщениях БЖД
NSI	SENDER_REPL_DIV	Таблица соответствия кодов НОД отправителей сообщения кодам станций совершения операций не своего НОД в сообщениях БЖД

Таблица 4.3

Table/View(s) of "1. Дорога" Subject Area		
Owner	**Name**	**Comment**
1	2	3
NSI	ADMINIST	Администрация
NSI	COUNTRIES	Страна
NSI	RAIL_DIV	Отделение дороги
NSI	RAILWAYS	Дорога
NSI	REGION	Регион
Table/View(s) of "2. Станция" Subject Area		
Owner	**Name**	**Comment**
NSI	FRONT	Фронт станции
NSI	INTER_JOIN	Справочник межгосударственных стыков
NSI	JOINT	Справочник стыков
NSI	JOINT_PRIZN	Признак стыка
NSI	OBJECT	Объект станции
NSI	PARAGRAF	Параграф станции
NSI	STA_STA_FLAG	Станции и их признаки
NSI	STATION_BOUNDARY	Пограничный переход
NSI	STATION_FLAG	Признаки станций
NSI	STATION_ISOLAT	Выделенные станции
NSI	STATION_NEAREST	Ближайшие выделенные станции

Окончание табл. 4.3

1	2	3
NSI	STATION_NOMARK	Расстояние и время хода от выделенной до невыделенной станции
NSI	STATION_PARAGRAF	Таблица соответствия станций параграфам
NSI	STATION_PROPERTY	Характеристика станции
NSI	STATION_RANGE	Диапазонный справочник станций
NSI	STATION_REP_NUMBER	Порядок выдачи станций в отчётах ГО
NSI	STATION_REPORT	Отчёты ГО
NSI	STATIONS	Станция
Table/View(s) of "3. Поезд" Subject Area		
Owner	**Name**	**Comment**
NSI	LOK_STATE	Состояние локомотива
NSI	TRAFFIC_KIND	Виды движения поезда
NSI	TRAIN_KIND	Рода поездов
NSI	TRAIN_MARK	Отметка о тяжеловесности и длиносоставности поезда
NSI	TRAIN_RANGE	Поезда
NSI	TRAIN_RANGE_STATE	Соответствие допустимых состояний локомотива диапазонам номеров поездов

4.10. Создание новой пустой БД для очередной версии

Сгенерированный средствами AllFusion Erwin Data Modeler на предыдущем этапе набор DDL операторов используется в качестве сценария компонентами СУБД DB2 UDB для создания новой версии БД. В данном документе в качестве физической БД используется СУБД DB2 UDB. Перечень функций графических средств DB2 «Командный Центр» и «Центр Управления» приведен в прил. 2. В данном приложении в краткой форме аналогичной, как и в прил. 1 приведены основные функции графических средств СУБД DB2 UDB и правила их применения. С целью краткого изложения материала далее по тексту применяются ссылки на пункты, содержащейся в прил. 2. Полное описание всех возможностей этих графических средств приводится в документах “IBM DB2 Universal Database. Руководство администратора.

Планирование”, “IBM DB2 Universal Database. Руководство администратора. Реализация” или в аналогичной документации для другой СУБД.

Создание пустой БД очередной версии производится графическими средствами Центра управления (ЦУ) DB2 (рис. 4.21).

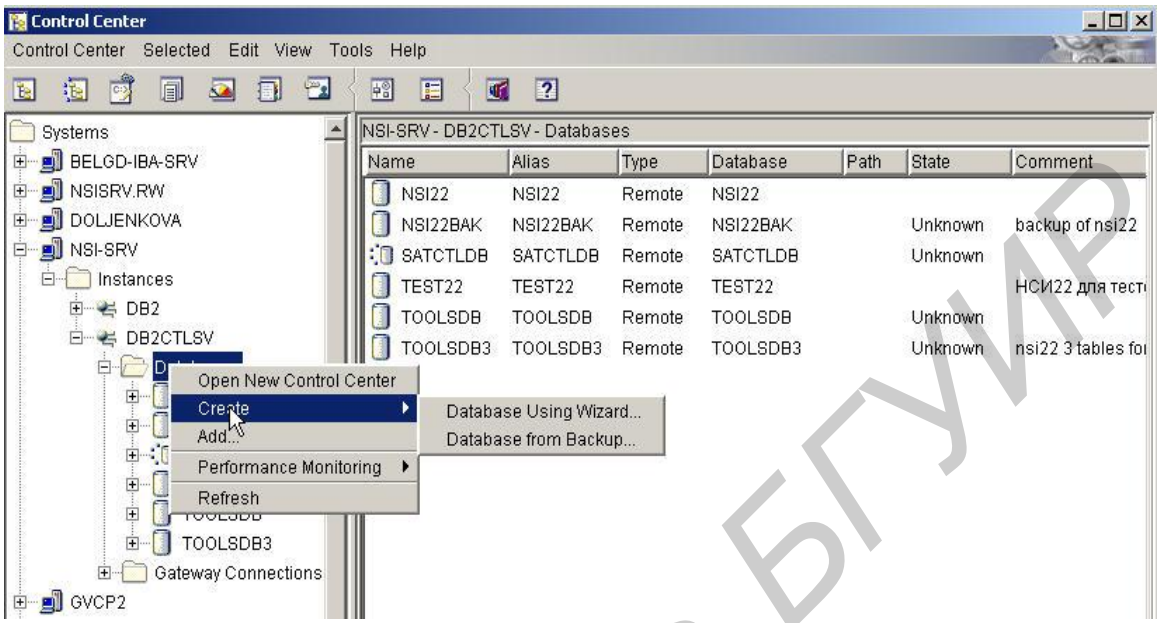

Рис. 4.21

Краткое описание функции «Создание БД» содержится в прил. 2, п. 1. После запуска ЦУ производится выбор системы, экземпляра DB2, правым щелчком по линейке «Базы данных» выбираем операции «Создать» и «БД при помощи мастера». После этого указываем основные параметры БД: имя БД, дисковод, алиас, комментарий (рис. 4.22).

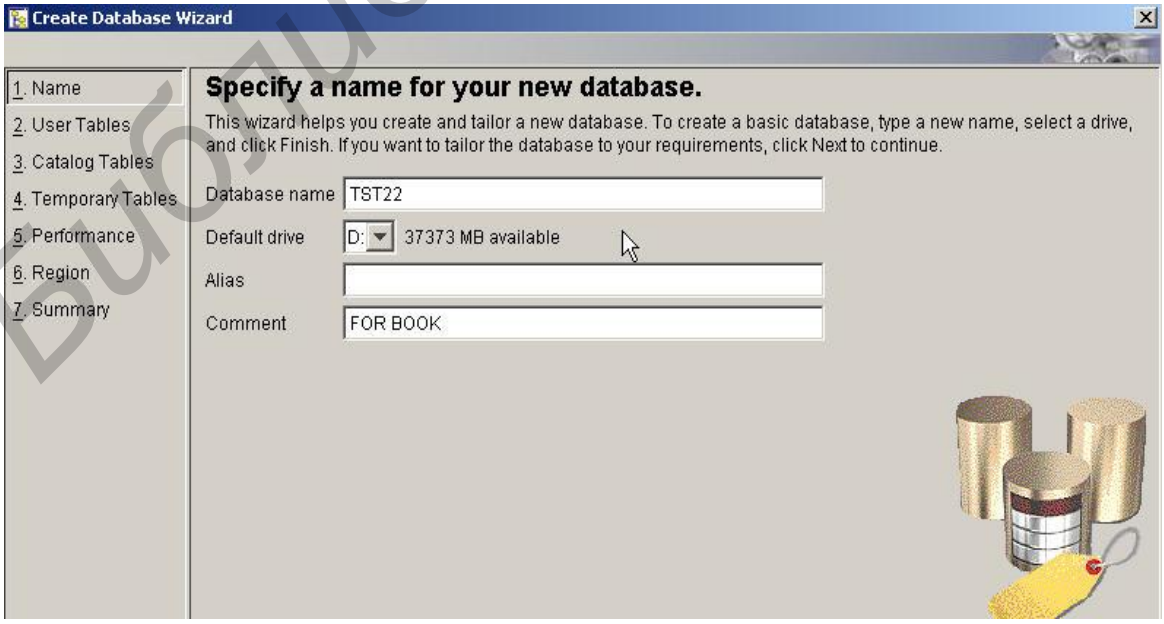

Рис. 4.22

В левой части панели, изображенной на рис. 4.22, перечислены все группы параметров, используемых для определения БД: «Имя», «Пользовательские таблицы», «Таблицы каталога» и т.д. Определение каждой группы параметров осуществляется на отдельной панели (группе панелей). Только определение первой группы параметров «Имя» является обязательным. После нажатия клавиши «Далее» осуществляется переход к следующей панели (экрану). Нажатие кнопки «Завершить» приводит к запуску команды (задания, сеанса) выполнения этой функции (рис. 4.23).

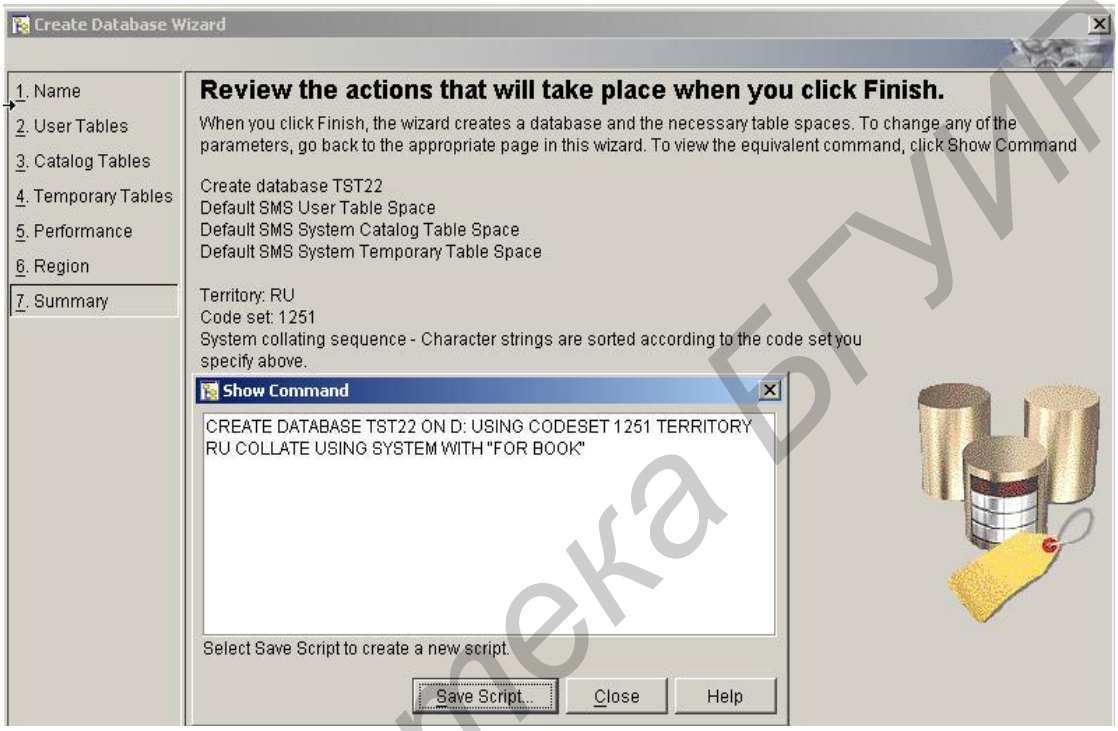

Рис. 4.23

После выполнения операции создается БД с указанным разработчиком именем и заданными им параметрами.

Далее средствами Командного центра (КЦ) СУБД DB2 UDB производится операция «Connect» присоединения к данной БД (рис. 4.24).

Затем тут же выполняется запуск задания, созданного средствами AllFusion Erwin Data Modeler на предыдущем этапе и содержащего операторы DDL для БД. Разработчик производит импорт задания, сохранение его в виде сценария для системы, содержащей ранее созданную БД. Производится запуск данного сценария для создания таблиц, индексов и других элементов БД (меню «Сценарии», операции «Создать», «Сохранить» и «Выполнить») (рис. 4.25., 4.26, прил. 2, п.2).

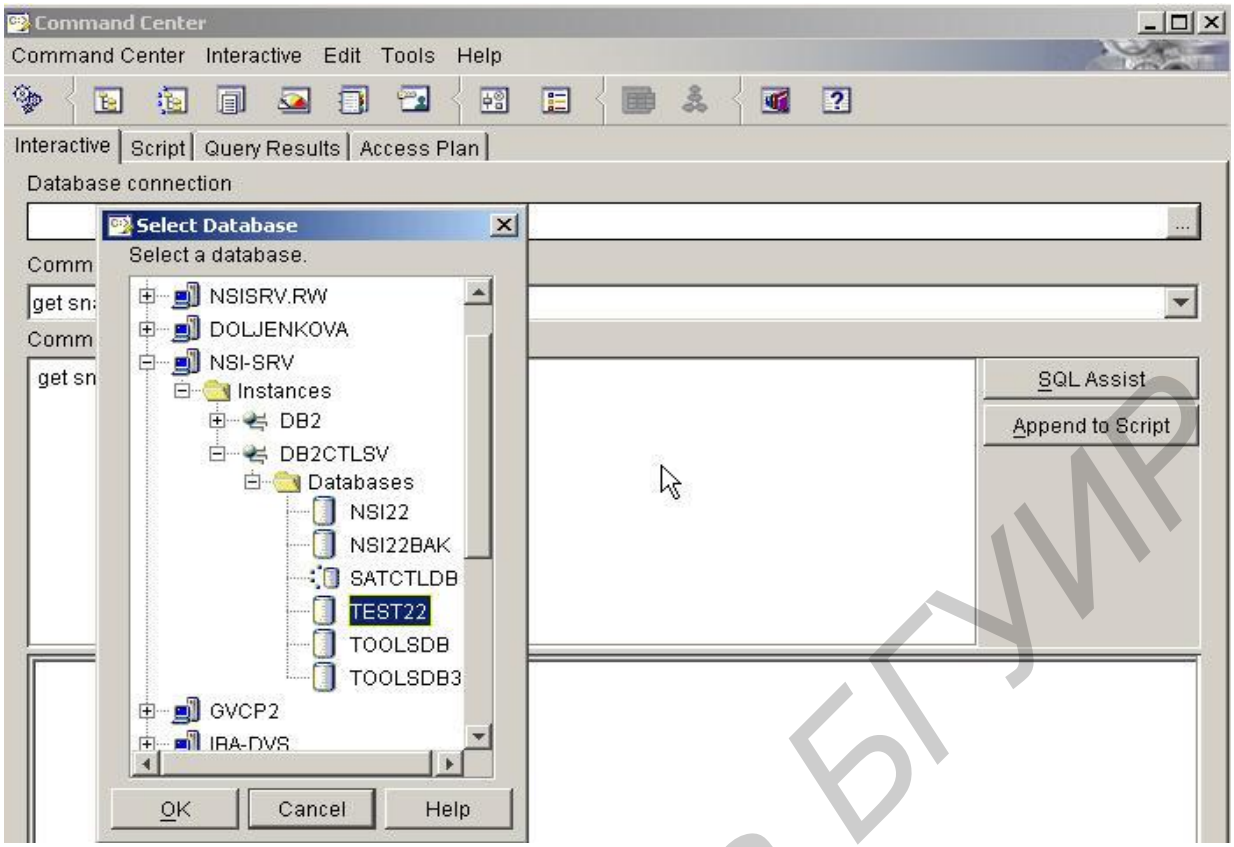

Рис. 4.24

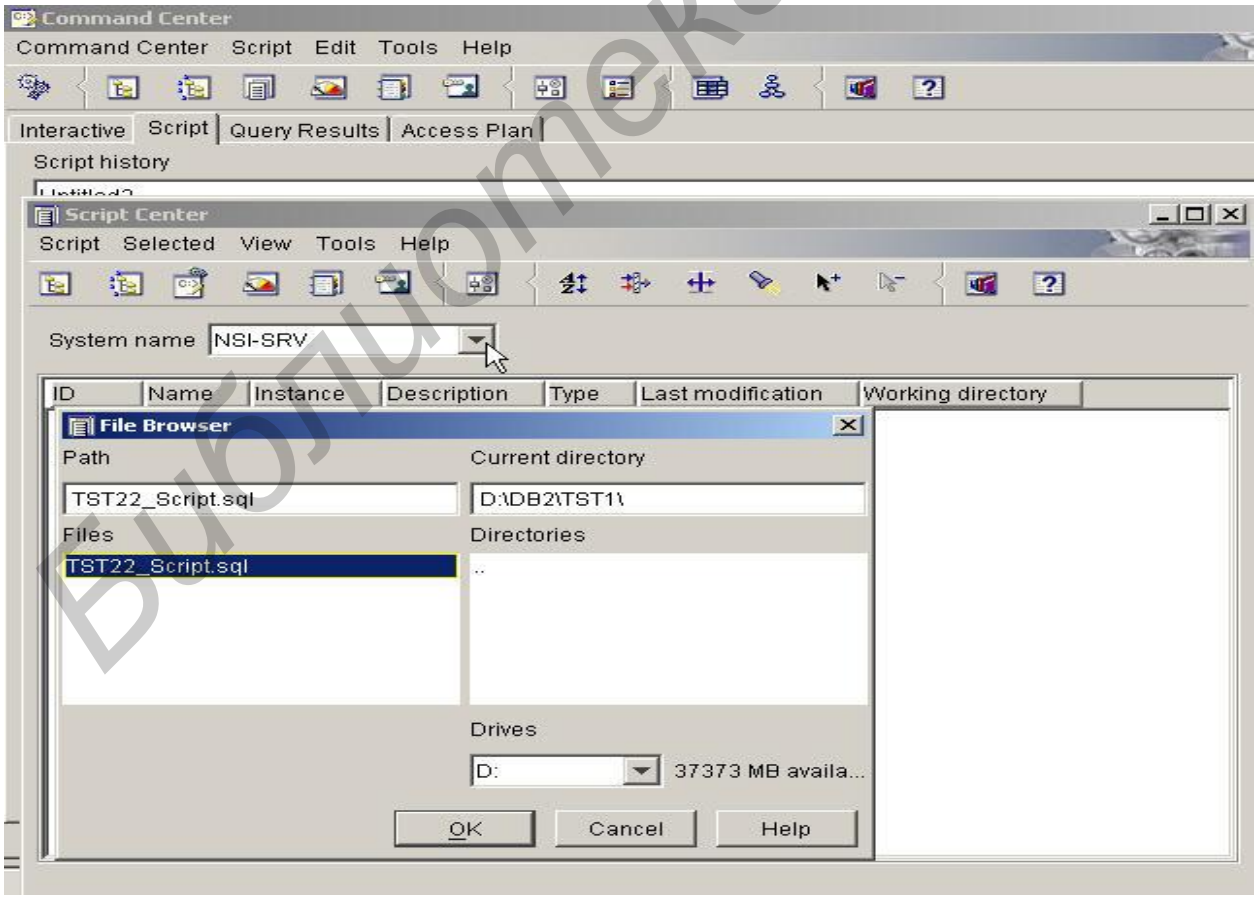

Рис. 4.25

В качестве входного файла указывается созданный на предыдущем этапе набор операторов языка DDL, содержащий операторы CREATE для создания буферных пулов, табличных пространств, таблиц, ключей, триггеров и других элементов вновь создаваемой версии БД. При нормальном завершении данного сценария будет создана новая БД, содержащая распределенные по табличным пространствам на сервере схемы, таблицы для данных БД и другие элементы базы данных DB2, обеспечивающие ее дальнейшее функционирование (см. рис. 4.26).

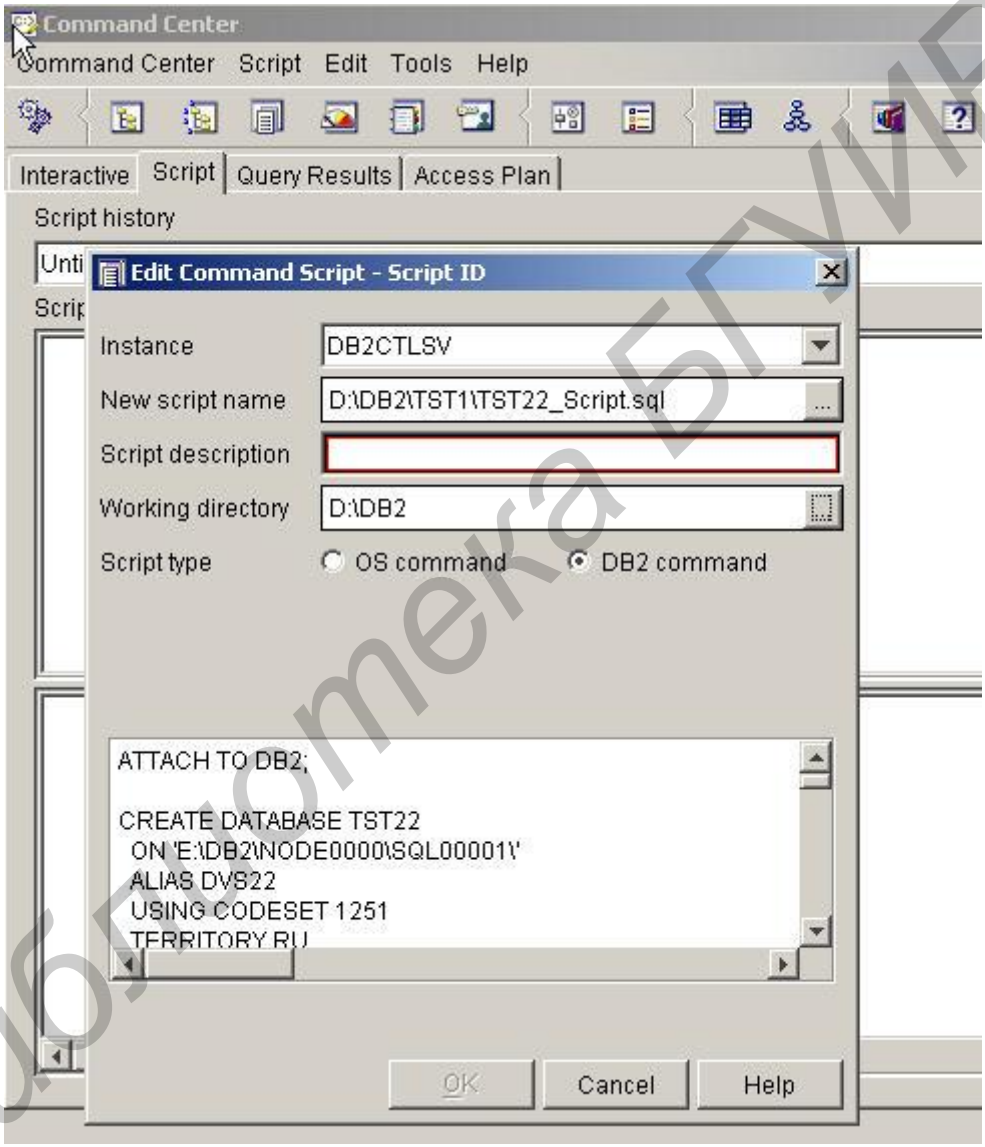

Рис. 4.26

4.11. Выгрузка данных старой версии БД

Для запуска новой версии БД необходимо выполнить выгрузку таблиц из старой версии БД в файлы-копии и затем загрузить данные из этих файлов-копий в новую пустую БД, созданную на предыдущем этапе. Эти работы можно выполнять двумя способами: в графическом и пакетном режимах. Выгрузка (экспортирование) таблиц выполняется следующим образом:

экспортирование всех таблиц старой БД в файлы-копии формата IXF в графическом интерфейсе ЦУ при помощи операции «Экспорт» (см. прил. 2, п. 3);

экспортирование всех таблиц старой БД в файлы-копии формата IXF с помощью соответствующей утилиты EXPORT (см. прил. 3, п. 1).

Ниже в качестве примера приводится последовательность действий разработчика для подготовки и выполнения операции «Экспорт» для таблицы «Страны» с использованием графического интерфейса «Центра управления» DB2 (см. рис. 4.27 и 4.28). Вначале выполняется подготовка (вызов) экспортирования таблицы «Страны» в файл формата IXF.

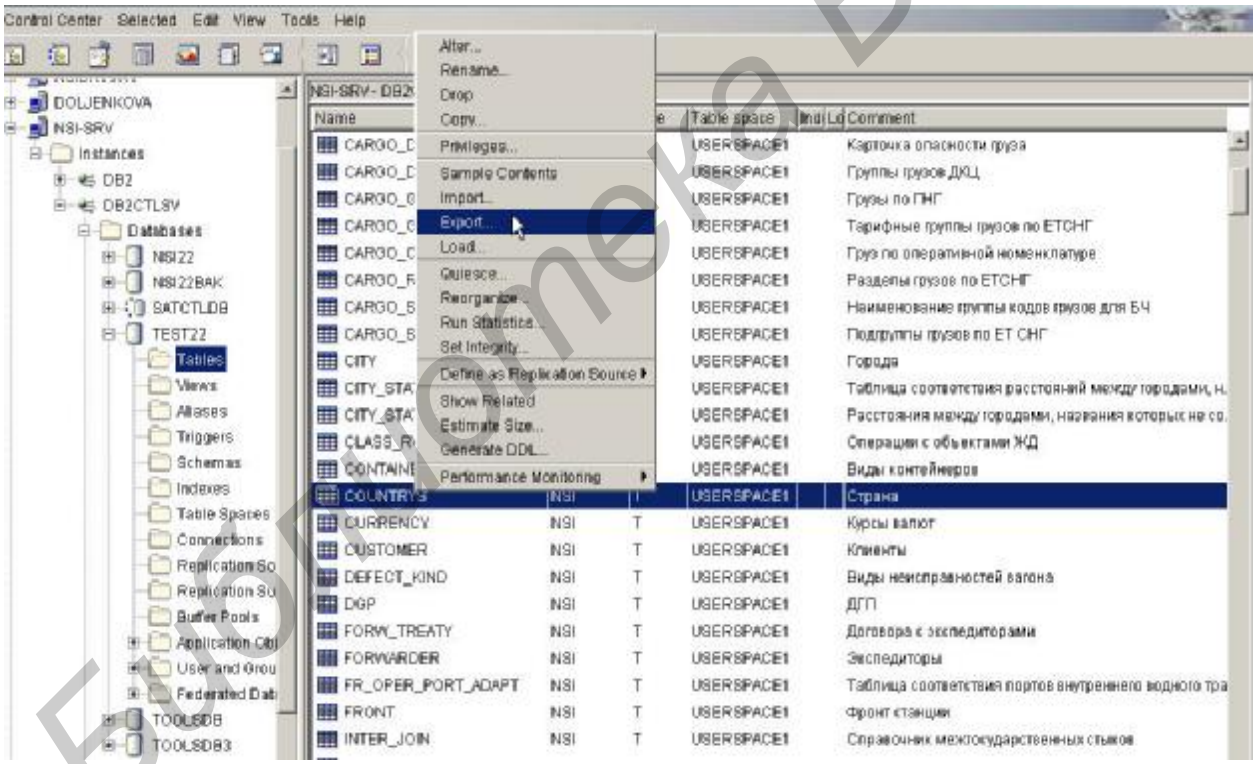

Рис. 4.27

Для вызова операции в «Центре управления» необходимо выполнить подключение к старой базе данных, раскрыть все ее составляющие, нажав знак «+», сделать щелчок левой клавишей мыши по «Table». После некоторого промежутка времени в правом окне выдастся перечень таблиц БД. Необходимо найти и щелкнуть правой клавишей мыши по нужной для экспортирования

таблице. В появившемся перечне операций найти и отметить операцию «Экспортировать». ЦУ DB2 запросит ввести информацию о пути доступа и имени файлов для копии и для сообщений (см. рис. 4.28).

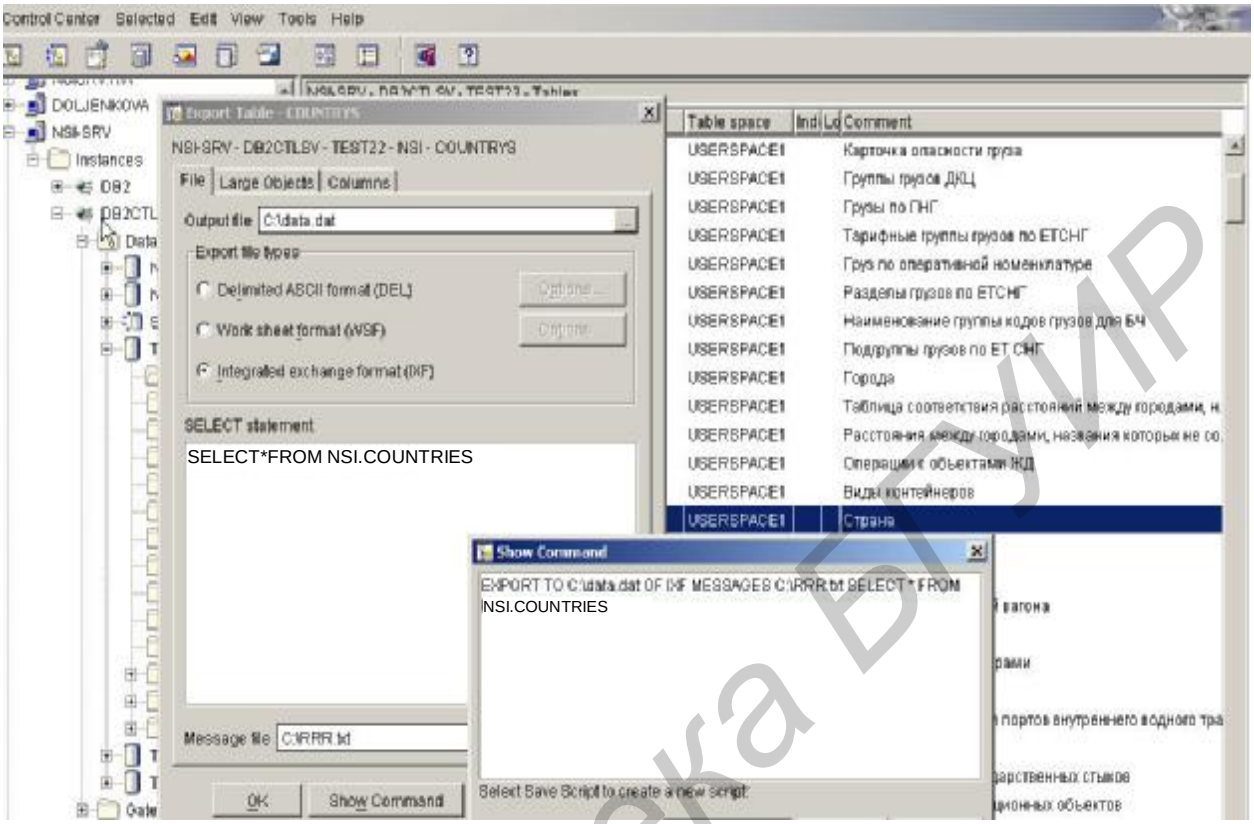

Рис. 4.28

При нормальном завершении операции для всех таблиц старой версии БД будут созданы файлы-копии и файлы с текстом сообщений DB2 о результатах операции выгрузки. Для нормальной работы рекомендуется заранее распределить память для файлов-копий, файлов сообщений, сценариев пусков, установить систему именования папок и файлов с использованием характеристик: какая операция, над каким объектом выполняется и дата выполнения.

4.12. Загрузка и запуск новой версии БД

Загрузку таблиц БД следует производить строго последовательно от родительских таблиц к дочерним в соответствии со схемой БД. Такой порядок загрузки обусловлен необходимостью формирования связей между таблицами по внешним ключам FK дочерней таблицы к первичным ключам PK родительской таблицы. Причем определяется, во-первых, порядок загрузки подсхем и, во-вторых, порядок загрузки таблиц внутри каждой отдельной подсхемы. Этот порядок загрузки определен уровнем расположения таблиц: от

самой нижней к верхней. Таблицы, которые не связаны ни с какими другими, можно загружать в любом порядке.

При загрузке для каждой модифицируемой таблицы необходимо указать имена столбцов, выбираемых для загрузки в новую БД, или порядковые номера загружаемых столбцов во входном файле. Эти работы можно выполнять двумя способами – в графическом и пакетном режимах:

загрузка данных каждой таблицы БД из файлов-копий формата IXF в графическом интерфейсе ЦУ при помощи операции «Загрузить» (см. прил. 2, п. 4 и 5);

загрузка данных каждой таблицы БД из файлов-копий формата IXF с помощью соответствующей утилиты LOAD (см. прил. 3, п. 2);

запуск команд SET INTEGRITY FOR «имя таблицы» IMMEDIATE CHECKED для каждой таблицы в командном центре после установления CONNECT к новой БД.

Ниже в качестве примера приводится последовательность действий разработчика DB2 для подготовки и выполнения операции «Импорт» для таблицы «Countries» с использованием графического интерфейса «Центра управления» DB2 (рис. 4.29 и 4.30).

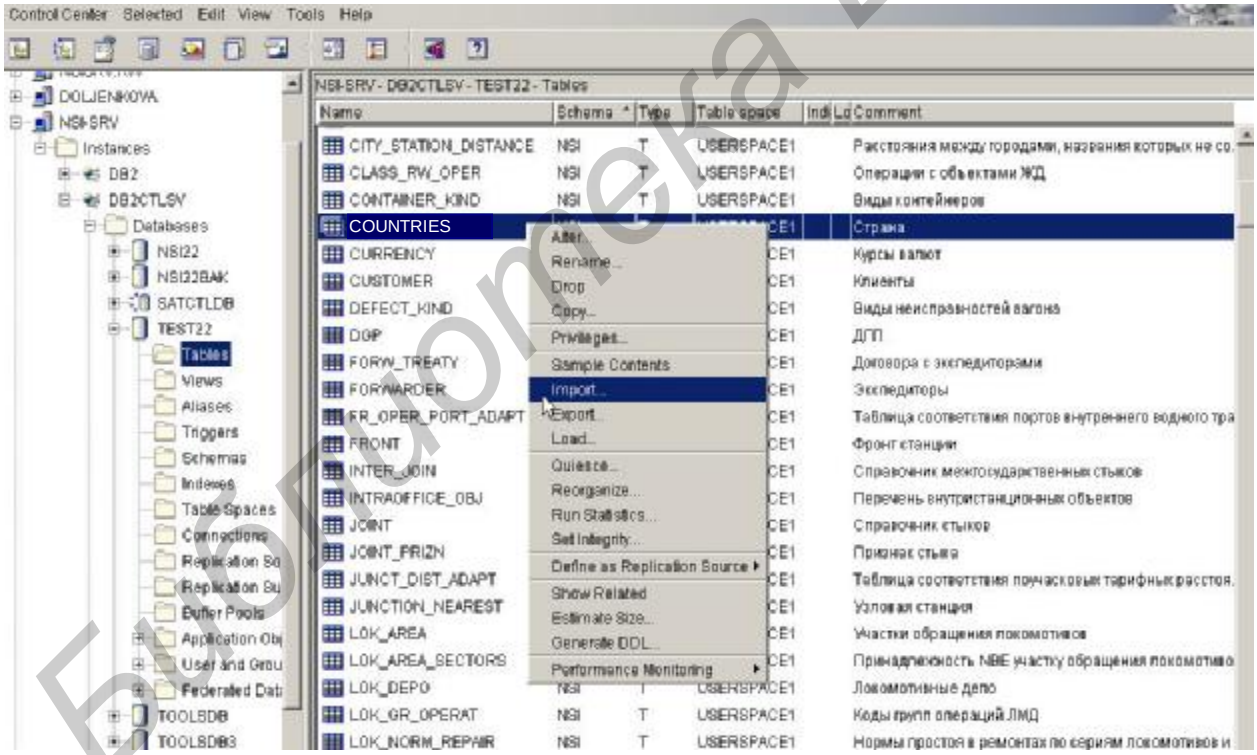

Рис. 4.29

Для вызова операции в «Центре управления» необходимо выполнить подключение к новой базе данных БД, раскрыть все ее составляющие, нажав знак «+», сделать щелчок левой клавишей мыши по «Table». После некоторого промежутка в правом окне появится перечень таблиц БД. Необходимо найти и

щелкнуть правой клавишей мыши по нужной для импортирования таблице. В появившемся перечне операций найти и отметить операцию «Импортировать». СУБД DB2 UDB запросит ввести информацию о пути и имени файлов для копии и для сообщений (см. рис. 4.30).

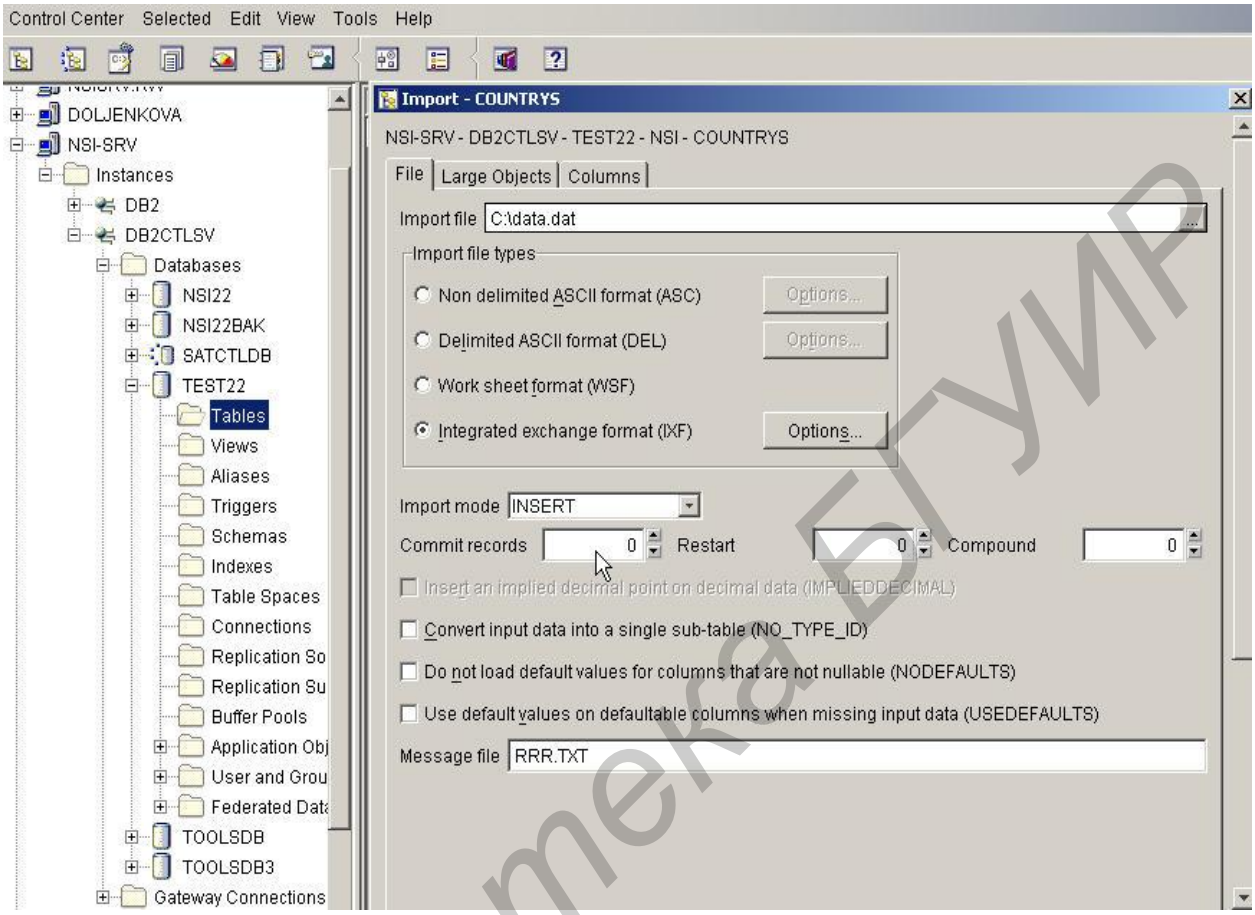

Рис. 4.30

При нормальном завершении операции все таблицы новой версии БД будут загружены из файлов-копий, созданных на предыдущем этапе из таблиц старой БД, при этом DB2 создает файлы с текстом сообщений о результатах операции загрузки. Дальнейшая работа по заполнению (дополнению) таблиц данными производится в процессе ее эксплуатации.

4.13. Дополнительные требования к построению модели

Разработчику БД предлагаются дополнительные необязательные рекомендации для создания файла скриптов при построении модели. При этом в круглых скобках делается ссылка на пункт «Генерация операторов для создания базы данных» для указания опций «AllFusion Erwin Data Modeler», что позволяет автоматически включить требующиеся конструкции при построении операторов DDL.

1. Используя ключевые слова IN и INDEX IN укажите имена табличных пространств, в которых будут размещаться данные таблицы и данные индексов (группы опций «Table» и «Index», опция «Physical Storage»). Если индексное табличное пространство не указано, данные индексов будут располагаться в том же табличном пространстве, что и данные таблицы. Если табличные пространства не указаны, данные таблицы и индексов будут располагаться в табличном пространстве, выбранном сервером DB2.

2. Воспользуйтесь оператором COMMENT ON TABLE для задания комментария к наименованию таблицы (группа опций «Other Options», опция «Comment») .

3. Воспользуйтесь оператором COMMENT ON COLUMN для задания комментария к наименованию столбца (группа опций «Other Options», опция «Comment»).

4. Воспользуйтесь включением конструкции CONSTRAINT «имя первичного ключа таблицы» PRIMARY KEY («имена столбцов первичного ключа») оператора CREATE TABLE для задания первичного ключа таблицы (группа опций «Other Options», опция «Constraint»). Эта конструкция включается в оператор CREATE TABLE, если в группе опций «Referential Integrity»/«Primary Key(PK)» выбрано «Create/PK», или в оператор ALTER TABLE, если там же выбрано «Alter/PK». В качестве наименования первичного ключа используется не более 8 символов, начинающихся с символа «P».

5. Воспользуйтесь оператором CONSTRAINT «имя внешнего ключа таблицы» FOREIGN KEY («имена столбцов внешнего ключа») для задания внешнего ключа. Аналогично предыдущему пункту эта конструкция включается в оператор CREATE TABLE, если в группе опций «Referential Integrity»/«Foreign Key(FK)» выбрано «Create/FK», или в оператор ALTER TABLE, если там же выбрано «Alter/FK». В качестве наименования используется не более 8 символов, начинающихся с символа «F». Внешний ключ более удобно определять с помощью оператора ALTER TABLE.

6. Укажите операции для обработки интеграционной целостности связанных таблиц БД для операций обновления и удаления. Аналогично предыдущему пункту эта конструкция включается в конструкцию CONSTRAINT оператора CREATE TABLE или оператора ALTER TABLE, если там же выбрано «On Delete» и/или «On Update».

7. Укажите правила для обработки интеграционной целостности в целом для модели (меню «Model», опция «Model properties») и для отдельных связей (меню «Model», опция «Relationships»). Рекомендуемое значение параметра – «RESTRICT».

В качестве примера таблицы БД, удовлетворяющей указанным правилам, приводится таблица «Администрация» – ADMINIST:

```
================================================================
    -- Table : ADMINIST
    --
================================================================

    create table NSI.ADMINIST
    (
      ADMIN_NO          CHAR(2)             not null,
      ADM_DATE_INS      TIMESTAMP           not null
        with default,
      ADMIN_ID          VARCHAR(12),
      COUNTRY_NO        CHAR(3),
      COU_DATE_INS      TIMESTAMP,
      ADMIN_NAME        VARCHAR(32),
      ADMIN_FULLNAME    VARCHAR(150),
      ADMIN_REAL_PR     CHAR(1),
      ADM_BGN_DATE      TIMESTAMP           not null,
      ADM_END_DATE      TIMESTAMP           not null,
      ADM_SIGN          CHAR(1)             not null,
      constraint P_K_ADM_ primary key (ADMIN_NO, ADM_DATE_INS)
    )
    in STARAITS
       index in STARAIIS;

    comment on table NSI.ADMINIST is 'Администрация';
    comment on column NSI.ADMINIST.ADMIN_NO is 'Код администрации';
    comment on column NSI.ADMINIST.ADM_DATE_INS is 'Дата совершения
операции';
    comment on column NSI.ADMINIST.ADMIN_ID is 'Мнемокод
администрации';
    comment on column NSI.ADMINIST.COUNTRY_NO is 'Код страны';
    comment on column NSI.ADMINIST.COU_DATE_INS is 'Дата совершения
операции (Таблица "Страны")';
    comment on column NSI.ADMINIST.ADMIN_NAME is 'Наименование
администрации';
    comment on column NSI.ADMINIST.ADMIN_FULLNAME is 'Полное
наименование администрации';
    comment on column NSI.ADMINIST.ADMIN_REAL_PR is 'Признак
реального кода администрации';
```

```
comment on column NSI.ADMINIST.ADM_BGN_DATE is 'Дата и время начала действия';
comment on column NSI.ADMINIST.ADM_END_DATE is 'Дата и время окончания действия';
comment on column NSI.ADMINIST.ADM_SIGN is 'Признак состояния записи (резерв)';

alter table NSI.ADMINIST
  add constraint F_COU_AD foreign key (COUNTRY_NO, COU_DATE_INS)
    references NSI.COUNTRYS (COUNTRY_NO, COU_DATE_INS)
    on delete restrict;
```

4.14. Пример модификации таблиц БД

Ниже приводится конкретный практический пример модификации старой БД и последовательность действий разработчика при получении новой версии БД. Модификация состоит во внесении нового объекта «Regions» в БД, который определяет принадлежность стран некоторым регионам мира: страны СНГ, страны Балтии, страны третьего мира и т.д. Ниже приводится последовательность действий по выполнению этой работы:

1) подготовка модификации: определение места и структуры новой таблицы, перечня столбцов, их форматы и характеристики, наличие ключей, индексов, связей;

2) загрузка старой схемы БД и ее корректировка. Новая таблица добавляется в подсхему «Дорога». Фрагмент подсхемы БД до изменения и после изменения приводится на рис. 4.31 и 4.32;

3) получение новой схемы БД. Генерация файла скриптов на языке DDL для создания всех таблиц новой БД;

4) создание новой БД;

5) выгрузка данных всех таблиц старой БД. Выгрузка производится средствами Центра управления DB2, при помощи операции Export (см. прил. 2, п. 3), или утилитой Export (см. прил. 3, п. 1), или, что предпочтительнее, утилитой DB2MOVE (см. прил. 3, п. 3), где операция выполняется для указанного перечня таблиц (можно указать до 10 таблиц);

6) последовательная загрузка данных старой БД (из их копий) в новую БД по каждой подсхеме. При загрузке необходимо учитывать наличие связей между подсхемами;

7) порядок загрузки таблиц внутри каждой подсхемы – от родительских таблиц к дочерним, начиная со старшей родительской таблицы в подсхеме.

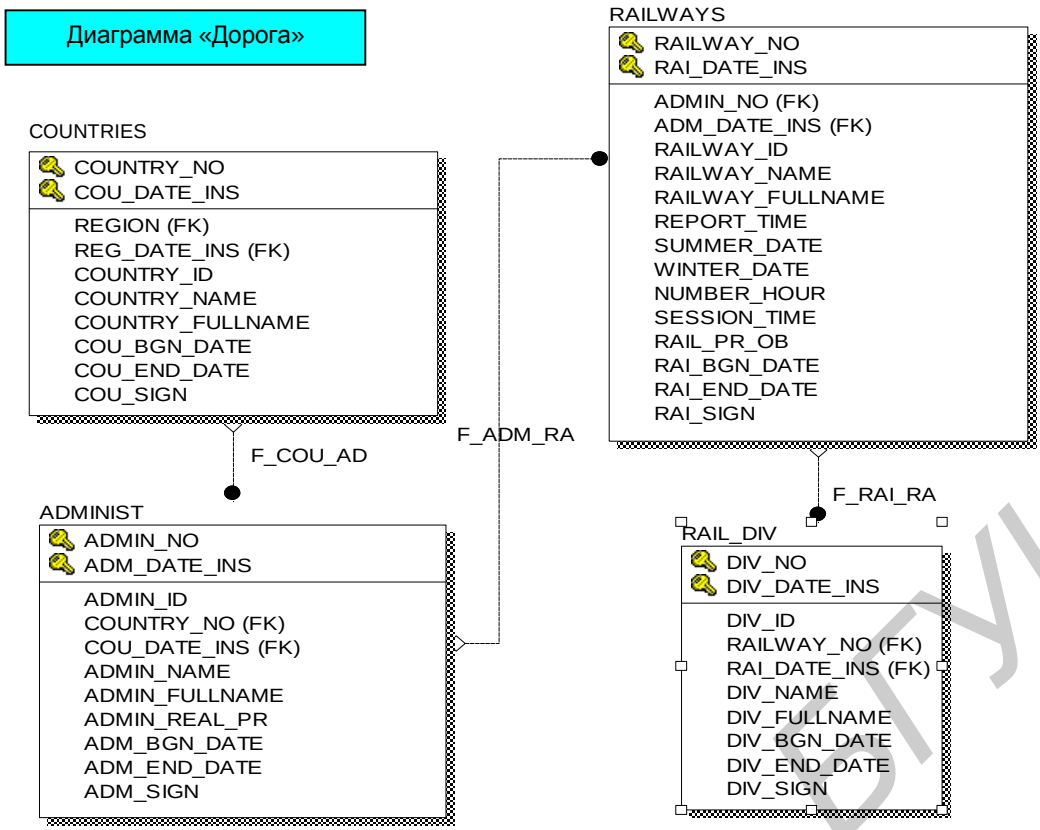

Рис. 4.31

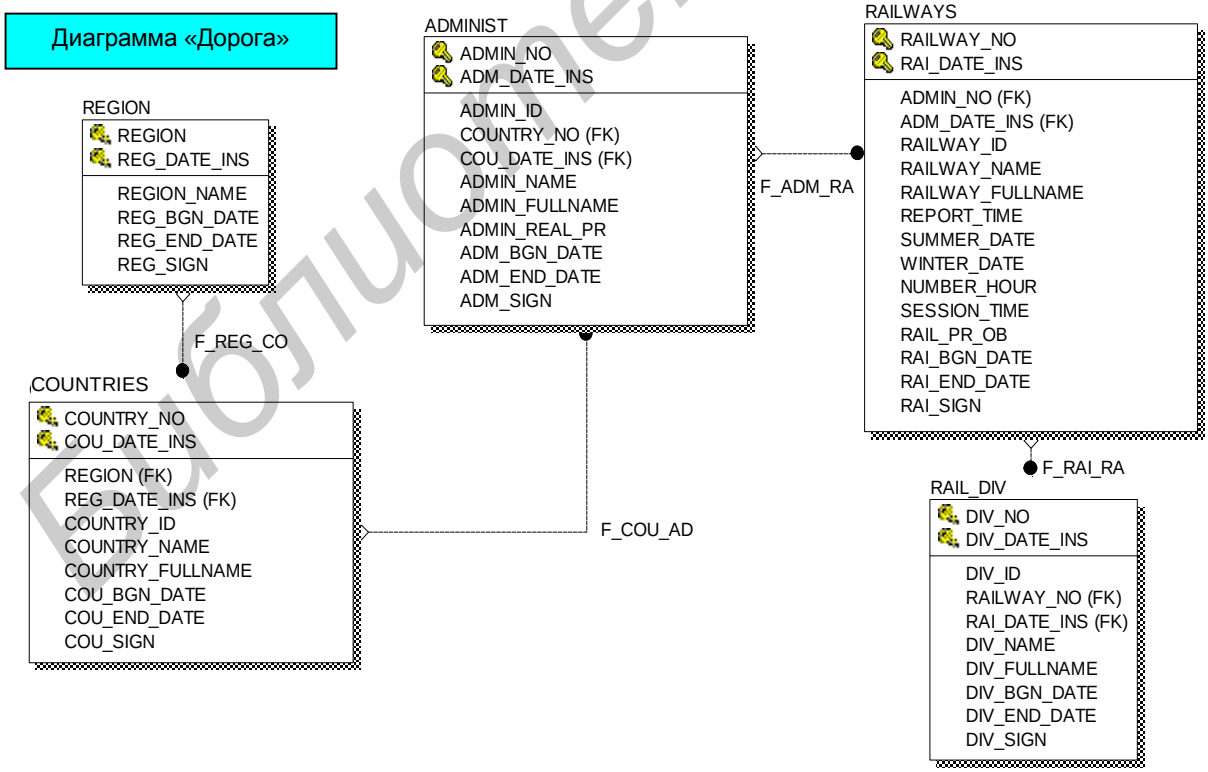

Рис. 4.32

помощи операции Import (см. прил. 2, п. 4), или утилитой Import (см. прил. 3, п. 3), или, что предпочтительнее, утилитой DB2MOVE (см. прил. 3, п. 4), где загрузка выполняется для ранее указанного при выгрузке перечня таблиц. Восстановление производится для всех таблиц до вставляемой (изменяемой) таблицы одним сеансом (заданием);

8) загрузка вставляемой (изменяемой) таблицы с помощью Центра управления DB2, утилиты Import (возможно, части ее столбцов, указывая их номера) или/и заполнение ее с помощью оператора. Выбор конкретного способа (способов) производится администратором DB2 в зависимости от наличия данных для загрузки этой вставляемой (изменяемой) таблицей и типов данных для нее в источнике. Не исключается вариант разработки специальных SQL-операторов для облегчения процесса первоначальной загрузки таблиц из других БД. В нашем примере здесь необходимо заполнить данными таблицу «Regions».

9) загрузка дочерних таблиц ниже уровнем, связанных с этой вставляемой (изменяемой) таблицей родительской связью. Загрузка производится Центром управления DB2, утилитой Import из копий, указывая номера столбцов, без FK-полей, которым присвоится значение NULL. В нашем примере здесь необходимо загрузить таблицу «Countries», указывая номера столбцов. Значение столбца «Код региона» установится NULL;

10) дополнение таблиц ниже уровнем, связанных с этой вставляемой (изменяемой) таблицей родительской связью. С помощью АРМ-оператора заполнить данными столбцы FK этих таблиц. В нашем примере здесь необходимо внести значение столбца «Код региона» с помощью АРМ-оператора;

11) продолжение загрузки оставшихся таблиц подсхемы, расположенных ниже уровнем, непосредственно не затронутых данной модификацией. Загрузка производится средствами Центра управления DB2, операцией Import (см. прил. 2, п. 4), или утилитой Import (см. прил. 3, п. 3), или, что предпочтительнее, утилитой DB2MOVE (см. прил. 3, п. 4), операцией Import для всех оставшихся таблиц, расположенных в схеме после вставляемой (изменяемой) таблицы и связанных с ней дочерних, по возможности одним сценарием (заданием).;

12) выполнение проверки работоспособности БД и ее компонент;

13) выполнение действий по настройке всех программ на новую БД.

Приложение 1

ПЕРЕЧЕНЬ ФУНКЦИЙ ПРОЕКТИРОВАНИЯ МОДЕЛИ ДАННЫХ И ИХ ИСПОЛЬЗОВАНИЕ

№	Функция	Меню для вызова **M**/ Model Explorer **ME**	Опция **O**/ Diagram window **DW**	Параметры функции	Ссылка в «Getting Started»
1	**2**	**3**	**4**	**5**	**6**
1.	Создание модели данных	**M**: File	**O**: New	Выбор типа модели, СУБД и ее версии: Logical, Physical or Logical/ Physical, Database – DB2, Version –DB2/UDB 6.1	Глава 4 «Creating a data Model», p. 4–6
2.	Создание/ добавление таблицы	**ME**: правый щелчок по Entity/Table	**DW**: Entity-перетягивание из Toolbox	**O**: New, Имя таблицы	Глава 4 «To add entity», p. 4–7
3.	Именование/ переименование таблиц	**ME**: правый щелчок по наименованию таблицы (Entity/Table)	**DW**: левый щелчок по Entity 1 раз + Tab n-раз, чтобы подвести имя таблицы	**O**: Rename, Имя таблицы новое ввести поверх старого	Глава 4 «To name entity», p. 4–8
4.	Добавление атрибута в таблицу	**ME**: левый щелчок по «+» для этой таблицы, правый щелчок по Columns	**DW**: 2-й левый щелчок по Entity	**O**: New, Имя и характеристики атрибута добавленного	Глава 4 «To add attributes», p. 4–9
5.	Добавление связей	**ME**: правый щелчок по Parent/Child Relationships для данной таблицы	**DW**: Выбрать значок указания связи, затем левый щелчок по Parent и Child-таблице соответственно	**ME:** Имя таблицы Child/Parent, с которой устанавливается связь, и тип связи Identify/ Non Identify. В результате этого Parent-таблица содержит Primary Key (PK), который автоматически включается в Child-таблицу как Foreing Key (FK)	Глава 4 «To add relationships», p. 4–11

Продолжение прил. 1

1	2	3	4	5	6
6.	Получение новой модели БД	**M**: Tools при открытой старой модели	**O**: Derive New Model	Тип модели: Logical, Physical or Logical–Physical, тип БД и версия, установки по умолчанию для выбора объектов из исходной модели в новую	Глава 4 «To derive a New Model», p. 4–15
7.	Сохранение модели данных	**M**: File при открытой старой модели	**O**: Save / Save as	Папка и Имя файла для сохранения модели в виде Erwin-файла: (*.er1)	Глава 4 «Working with a Data Models», p. 4–7
8.	Выборка модели данных	**M:** File	**O:** Open	Папка и Имя файла (Имя модели) (*.er1)	Глава 4 «Working with a Data Models», p. 4–7
9.	Просмотр DDL (Script File)	**M**: Tools при открытой модели типа Physical	**O**: Forward Engineer/Schema Generation	**Окно «DB2/UDB Schema Generation»:** Нажатие кнопки «Preview» – Показывает Preview Script File для данной схемы	Глава 4 «Forward Engineering. Preview Script File», p. 4–18
10.	Генерация Script File. Исп. далее для получения новой схемы (п.11 – Reverse engineering)	**M**: Tools при открытой модели типа Physical	**O**: Forward Engineer/Schema Generation	**Окно «DB2/UDB Schema Generation»:** Необходимо произвести включение необх. параметров для Options, Summary, Comment. Нажатие кнопки «Report» генерирует ее Script File и сохраняет в виде DDL, «Save» – сохранение в DDL-файле	Глава 4 «Forward Engineering. Generate Script File», p. 4–19

1	2	3	4	5	6
11.	Создание модели данных из БД уже существующей или script файла (DDL-файла)	**M**: Tools	**O**: Reverse Engineer	**Окно «New model type»:** Тип модели: Logical/Physical или Physical , тип БД (DB2) и ее версия (DB2/UDB 7.2); **Окно «Reverse engineer»:** set Options: from: источник – «script file» + имя file Sql; или источник – «database»; Owners by (фильтр owners – имя подсхемы (NSI) для отброса лишнего); включить Infer Primary Keys & Relations (генерацию ключей, индексов и связей); Нажать кнопку Next **Окно «DB2/UDB ODBC connections»:**– указать ID Администратора, его Password, имя Database. , параметр Use ODBC driver connect – не включать.	Глава 5 «Use existing data to build a new Model», p. 5–1–4
12.	Функция «Полное сравнение БД» для получения различий между двумя БД и приведение их в соответствие	**M**: Tools при открытой модели типа Physical	**O**: Complete Compare	**Окно «Compare options»:** 1)Compare Type – «Database/model level compare»; 2) Compare current model with «Database»/«Script file»/«er1» + 2-й источник – file «Sql»/«er1» 3)Action – задает режим обновления сравниваемых моделей:bi–directional/ current/other	Глава 5 «Using Complete compare», p. 5–5
13.	Построение отчетов из модели данных Создание образца:	**M**: Tools при открытой модели типа Logical	**O**: Report Builder	Окно «Report template»: New, Окно «Report Layout»: Document Untiled, Окно «Property»: Tab «Title» и ввести Название report + Tab «Add macro» + Tab «Export», Окно «Properties»: Формат HTML/RTF + Окно «HTML export properties» : «Pict report as popup windows» + Close wind – «X»	Глава 6 «Build reports on your Erwin Models», p. 6–1

1	2	3	4	5	6
14.	Построение отчетов из модели данных Добавление отчета к образцу:	**M**: Tools при открытой модели типа Logical	**O**: Report Builder	Окно «Report template Builder»: right-щелчок по «Picture» – Add sections, right-щелчок по «Entity» – Add sections + Окно»Report layout» – 2 раза left-щелчок по «Entity sections»+Окно «Property»– включить необходимые элементы для отчета(«+» для Entity и выбор Name и Definition, «+» для Attribute и выбор Name, Type и др.), + Close wind – «X» + **M**:File/ Save as «имя отчета» для его сохранения. Запуск HTML отчета:	Глава 6 «Build reports on your Erwin Models», p. 6–3
15.	Построение отчетов из модели данных Запуск HTML отчета:	**M**: Tools при открытой модели типа Logical	**O**: Report Builder	Окно «Report Builder»: подвести под нужный отчет и left-щелчок по нему, вызывается Web Browser для его выдачи – включить необходимые элементы для показа отчета в левой части–содержании. Для закрытия – Close Web Browser wind – «X».	Глава 6 «Build reports on your Erwin Models», p. 6–4

Приложение 2

ПЕРЕЧЕНЬ ФУНКЦИЙ ГРАФИЧЕСКИХ СРЕДСТВ DB2 И ИХ ИСПОЛЬЗОВАНИЕ

№	Функция	Входные данные (пример)	Выходные данные (пример)	Меню ЦУ или КЦ, операция	Параметры	Команда DB2/ (пример)
1	**2**	**3**	**4**	**5**	**6**	**7**
1.	Создание БД	Имя новой версии БД	Созданный объект БД	ЦУ: Базы данных – Создать	Имя БД	
2.	Распределение таблиц в новой версии	Имя файла задания, содержащего операторы DDL для новой версии БД	Созданные пустые таблицы БД	КЦ: Сценарии – Импорт, Создать; Сценарии – выполнение	Имя файла задания с операторами DDL, полученного в Erwin	Зам. В Script-файле Erwin необходимо заменить шрифты на «Кирилл. Window»
3.	Экспорт таблицы в файл	Таблица БД DB2ADMIN.EMPLOYEE;	Файл ОС системы C:\DB2_RAIL_NSI\exp\emp и файл сообщений C:\DB2_RAIL_NSI\exp\emp_soob	(МЦУ) – определение схемы Функция «Экспорт» для таблицы (right-щелчок)	Файл ОС, Файл сообщений	EXPORT TO C:\DB2_RAIL_NSI\exp\emp OF IXF MESSAGES C:\DB2_RAIL_NSI\exp\emp_soob SELECT * FROM DB2ADMIN.EMPLOYEE;
4.	Импорт из файла в таблицу (м.б. указаны столбцы)	Файл ОС C:\DB2_RAIL_NSI\exp\emp Файл сообщений C:\DB2_RAIL_NSI\exp\emp_soob	Таблица БД DB2ADMIN.EMPLOYEE;	(МЦУ) – определение схемы Функция «Импорт» для таблицы (right-щелчок)	Файл ОС, Файл сообщений	IMPORT FROM C:\DB2_RAIL_NSI\exp\emp OF IXF MODIFIED BY indexschema=DB2ADMIN MESSAGES C:\DB2_RAIL_NSI\exp\emp_soob_imp INSERT INTO DB2ADMIN.EMPLOYEE ;

Продолжение прил. 2

1	2	3	4	5	6	7
5.	Загрузка данных таблиц в новую БД Load/Загрузить / Import/Импорт	Имя файла, содержащего данные для загрузки таблицы (ранее экспортированного по Export)	Загруженные данные в таблицу БД	ЦУ: Таблица – Загрузить/ Импорт	Файл: Загрузка из лок.файл.системы +Имя файла копии (Export) Столбцы: Вкл. По номерам столбцов–Y/N +N столбцов указать относительно данных Export-файла	При коррект. таблиц необходимо задать соответствие столбцов результирующей таблицы относительно исходных данных
6.	Удаление старой БД	Объект – База данных	Удаление имени объекта – БД в перечне баз данных ЦУ	Убрать или Отбросить (МЦУ)	Ввести ответ на запрос о подтверждении удаления требуемого объекта и его характеристик	
7.	Резервное копирование БД	Объект – База данных	c:\db2_rail_nsi\exp	Резервное копирование (МЦУ)	Ввести путь доступа к каталогу для копии c:\db2_rail_nsi\exp	BACKUP DATABASE SAMPLE TO C:\\DB2_RAIL_NSI \\exp WITH 2 BUFFERS BUFFER 1024 ;
8.	Резервное копирование БД Online (при помощи мастера)	Объект – База данных	c:\db2_rail_nsi\exp	Резервное копирование (МЦУ)	Ввести параметры БД и путь доступа к каталогу для копии c:\db2_rail_nsi\exp	BACKUP DATABASE SAMPLE ONLINE TO C:\\DB2_RAIL_NSI \\exp WITH 2 BUFFERS BUFFER 1024 ;

1	2	3	4	5	6	7
9.	Восстановление БД	Резервная копия БД В каталоге	c:\db2_rail_nsi\exp	Восстановление (МЦУ)	Ввести путь доступа к каталогу для выбора копии c:\db2_rail_nsi\exp	RESTORE DATABASE SAMPLE FROM C:\\DB2_RAIL_NSI \\exp TAKEN AT 20020403112243 WITH 2 BUFFERS BUFFER 1024 WITHOUT PROMPTING ;
10.	Восстановление БД во вновь созданную	Резервная копия БД В каталоге.	c:\db2_rail_nsi\exp	Восстановление в новое (МЦУ)	Ввести путь доступа к каталогу для выбора копии c:\db2_rail_nsi\exp	RESTORE DATABASE SAMPLE FROM C:\\db2_rail_nsi\\exp\\ TAKEN AT 20020403 WITH 2 BUFFERS BUFFER 1024 WITHOUT ROLLING FORWARD WITHOUT PROMPTING ;
11.	Генерация операторов DDL для БД	Объект – База данных (подсхема NSI только)	Сценарий процессора командной строки DB2 (CLP) сохранен в: C:\PROGRA~1\SQLLIB\db2_5.out	SQL набор операторов	Ввести путь доступа	echo db2look: C:\PROGRA~1\SQLLIB\db2_6.out db2look -d STARAIDB -z NSI -e -l -x -c -o db2_6.out;

Приложение 3

ПЕРЕЧЕНЬ ФУНКЦИЙ УТИЛИТ DB2 ДЛЯ КОПИРОВАНИЯ ДАННЫХ И ИХ ИСПОЛЬЗОВАНИЕ

Функция	Входные данные (Объект)	Выходные данные (пример)	Окно команд DB2 (пример)
Экспорт из таблицы в файл	Таблица БД NSI.ADMINIST;	Файл ОС C:\ПУР_ГП_НСИ\Erwin\staraidb\exp_adm_star Файл сообщений C:\ПУР_ГП_НСИ\Erwin\staraidb\exp_adm_star_soob SELECT * FROM NSI.ADMINIST;	EXPORT TO C:\ПУР_ГП_НСИ\Erwin\staraidb\exp_adm_star OF IXF MESSAGES C:\ПУР_ГП_НСИ\Erwin\staraidb\exp_adm_star_soob SELECT * FROM NSI.ADMINIST;
Импорт из файла в таблицу (м.б. указаны столбцы)	Файл ОС C:\DB2_RAIL_NSI\exp\emp	Таблица БД NSI.ADMINIST; Файл сообщений C:\DB2_RAIL_NSI\exp\ emp_soob_imp	IMPORT FROM C:\DB2_RAIL_NSI\exp\emp OF IXF MODIFIED BY indexschema=DB2ADMIN MESSAGES C:\DB2_RAIL_NSI\exp\emp_soob_imp INSERT INTO DB2ADMIN.EMPLOYEE ;
Пересылка (db2move) БД или таблиц БД между рабочими станциями в т.ч. на различных платформах	БД Sample	Копия БД	db2move sample export -u db2admin -p vfrcbv
Пересылка (db2move import) БД или таблиц БД	Копия БД	БД Sample	db2move sample import -u db2admin -p vfrcbv

ЛИТЕРАТУРА

1. AllFusion® ERwin Data Modeler. Interface [Электронный ресурс]. – 2008. – Режим доступа: http://www.interface.ru/home.asp?artId=7556

2. Integration Definition for Information Modeling (IDEF1x), NIST USA, 148p.

3. Зикопулос, П. DB2 версии 8 : официальное руководство. / П. Зикопулос; пер. с англ. IBM Россия [Электронный ресурс]. – 2004. – Режим доступа: http://www.ibm.com/developerworks/ru/doc/data.html?S_TACT=105AGX99&S_CMP=SIMPLERS

4. Дейт, Дж. Введение в системы баз данных / Дж. Дейт; пер. с англ. – 6-е изд. – Киев : Диалектика, 1998. – 784 с.

Св. план 2008, поз. 54

Учебное издание

Пилецкий Иван Иванович

ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА И СОПРОВОЖДЕНИЕ БАЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ CASE-СРЕДСТВ

Пособие по курсу
«Методы и технологии программирования»
для студентов специальности 1-31 03 04 «Информатика»
всех форм обучения

Редактор Т. Н. Крюкова
Корректор М. В. Тезина
Компьютерная верстка Е. Г. Бабичева

Подписано в печать 19.03.2009.	Формат 60x84 1/16.	Бумага офсетная.
Гарнитура «Таймс».	Печать ризографическая.	Усл. печ. л. 6,86.
Уч.-изд. л. 6,2.	Тираж 180 экз.	Заказ 15.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6