

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Г.П. Токмаков

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ С ПОМОЩЬЮ CASE-СРЕДСТВ

**Учебно-методические указания
по выполнению лабораторных работ**

Ульяновск
УлГТУ
2016

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
ГЛАВА 1. ВАРИАНТЫ ЗАДАНИЙ ПО ЛАБОРАТОРНЫМ РАБОТАМ И ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТНЫХ МАТЕРИАЛОВ	7
1.1 ВАРИАНТЫ ЛАБОРАТОРНЫХ РАБОТ	7
1.1 ОПИСАНИЕ ЛАБОРАТОРНЫХ РАБОТ	9
1.2 ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТНЫХ МАТЕРИАЛОВ.....	10
ГЛАВА 2. СОДЕРЖАНИЕ РАБОТ ПО ВЫПОЛНЕНИЮ ЗАДАНИЙ	17
2.1 ЛАБОРАТОРНАЯ РАБОТА 1. ИЗУЧЕНИЕ, УСТАНОВКА И НАСТРОЙКА СУБД POSTGRESQL И CASE-СРЕДСТВА ERWIN.....	17
2.1.1 Установка СУБД POSTGRESQL	17
2.1.2 Установка CASE-средства ERWIN	18
2.1.3 Создание пустой БД	18
2.1.4 Создание псевдонима БД	19
2.2 ЛАБОРАТОРНАЯ РАБОТА 2. РАЗРАБОТКА КОНЦЕПТУАЛЬНОЙ СХЕМЫ ПРО.....	22
2.2.1 Выделение сущностей ПРО	23
2.2.2 Определение типов сущностей.....	23
2.2.3 Выделение реквизитов	24

2.2.4	Выделение связей между объектами и определение их типов	25
2.2.5	Определение типов связей	26
2.2.6	Концептуальная модель ПРО.....	27
2.3	ЛАБОРАТОРНАЯ РАБОТА 3. РАЗРАБОТКА ЛОГИЧЕСКОЙ СХЕМЫ ПРО	29
2.3.1	Создание сущностей	29
2.3.2	Установка атрибутов сущности.....	32
2.3.3	Установка свойств связей между сущностями	33
2.3.4	Диаграмма логической схемы	41
2.4	ЛАБОРАТОРНАЯ РАБОТА 4. РАЗРАБОТКА ФИЗИЧЕСКОЙ СХЕМЫ	43
2.4.1	Выбор сервера	43
2.4.2	Таблицы и колонки	44
2.4.3	Редактирование свойств полей.....	46
2.4.4	Проверка результатов выполнения скриптов	49
2.5	ЛАБОРАТОРНАЯ РАБОТА 5. РАЗРАБОТКА ГЕНЕРАТОРОВ И ТРИГГЕРОВ ДЛЯ КЛЮЧЕВЫХ АТТРИБУТОВ ТАБЛИЦ БД.....	51
2.5.1	Краткие сведения о языке макросов ERWIN и последовательностях, генераторах и триггерах	51
2.5.2	Создание пользовательского свойства.....	54
2.5.3	Создание шаблона скрипта последовательности, генератора и триггера	55
2.5.4	Проверка результатов выполнения скриптов	58
2.6	ЛАБОРАТОРНАЯ РАБОТА 6. РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР ДЛЯ ВСТАВКИ, ОБНОВЛЕНИЯ И УДАЛЕНИЯ ЗАПИСЕЙ ТАБЛИЦ БД	60
2.6.1	Краткие сведения о Хранимых процедурах.....	60
2.6.2	Создание шаблона скрипта функций хранимых процедур	62
2.6.3	Проверка результатов выполнения скриптов	65

2.7	ЛАБОРАТОРНАЯ РАБОТА 7. ГЕНЕРИРОВАНИЕ СКРИПТА И СОЗДАНИЕ БД В POSTGRESQL.....	67
2.7.1	Подключение CASE-средства ERWIN К СУБД POSTGRESQL.....	67
2.7.2	Генерирование SQL-сценария создания БД	68
2.7.3	Создание схемы БД.....	71
2.7.4	Проверка результатов выполнения скриптов	72
	ЗАКЛЮЧЕНИЕ.....	75
	БИБЛИОГРАФИЧЕСКИЙ СПИСОК	76

ВВЕДЕНИЕ

На современном этапе развития технологии машинного моделирования реальной действительности, сфера применения «чистого программирования», когда и данные и код создаются в пределах одной среды программирования, сужается. На смену этой технологии приходят более сложные приемы моделирования реальной действительности, согласно которым средствами систем управления базами данных (СУБД) разрабатываются модели реальной действительности коллективного пользования в виде баз данных (БД), а затем, используя ту или иную среду программирования, создаются программные средства и пользовательские интерфейсы для различных категорий пользователей.

Такие системы создаются по клиент-серверной архитектуре, в рамках которой приложение может взаимодействовать с другими приложениями по локальной сети путем обмена данными через сервер баз данных. В архитектуре «клиент-сервер» СУБД принимает запросы от различных программных средств и возвращает им результаты по сети. Такая СУБД называется сервером БД.

Переход на клиент-серверную архитектуру повысил эффективность функционирования информационных систем, но одновременно возросла их сложность и объемы данных, хранящихся в базах.

В настоящее время разработка БД информационных систем – это задача для коллективов разработчиков, выполнение которой разбивается на следующие этапы:

- анализа предметной области (ПрО);
- проектирования концептуальной схемы;
- разработки логической и физической схемы, для размещения элементов моделируемых информационных объектов предметной области, выявленных на предыдущих этапах, в памяти машины.

Этап анализа предметной области включает в себя выделение информационных объектов и их атрибутов, установление связей между информационными объектами и определение их характеристик. В результате выполнения анализа предметной области создается концептуальная модель будущей базы данных.

На этапе разработки создается логическая и физическая модель БД. Логическая модель представляет собой машинную реализацию концептуальной модели, а физическая, в отличие от логической модели, строится с привязкой к конкретной СУБД.

Время разработки современных информационных систем сравнимо с предполагаемым циклом их жизни, и сокращение времени их разработки становится одной из важнейших задач современной индустрии разработки информационных систем.

Поэтому будущему специалисту в области разработки информационных систем крайне необходимо владеть инструментарием быстрой разработки БД. С этой целью в данных учебно-методических указаниях описываются содержание и порядок выполнения лабораторных работ, направленных на усвоение студентами практических приемов разработки БД с помощью CASE-средства [ERwin](#), существенно сокращающего трудоемкость разработки информационных систем.

Данные учебно-методические указания предназначены для использования бакалаврами направлений [09.03.04](#), [09.04.01](#), а также студентами других родственных специальностей, специализирующихся в области автоматизированного проектирования информационных систем.

ГЛАВА 1. ВАРИАНТЫ ЗАДАНИЙ ПО ЛАБОРАТОРНЫМ РАБОТАМ И ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТНЫХ МАТЕРИАЛОВ

Лабораторные работы, описываемые в данных учебно-методических пособиях, охватывают все этапы разработки БД информационных систем, описанные во введении. Кроме лабораторных работ, связанных с собственно разработкой информационной системы, в данных указаниях содержится описание работы, направленной на изучение и освоение практических навыков работы с CASE-средством **ERwin**, с помощью которой создается SQL-сценарий создания БД, и с СУБД, с помощью которой создается и управляется целевая БД системы.

1.1 ВАРИАНТЫ ЛАБОРАТОРНЫХ РАБОТ

В качестве вариантов ПрО, бизнес-процессы которых требуется автоматизировать с помощью информационной системы, рассматриваются следующие.

Вариант 1. Автоматизированная разработка БД АС для учета деятельности аптеки: поступление и реализация лекарств.

Вариант 2. Автоматизированная разработка БД АС для учета деятельности продуктового магазина: поступление и реализация продовольственных товаров.

Вариант 3. Автоматизированная разработка БД АС для учета деятельности обувного магазина: поступление и реализация обуви.

Вариант 4. Автоматизированная разработка БД АС для учета деятельности магазина одежды: поступление и реализация одежды.

Вариант 5. Автоматизированная разработка БД АС для учета деятельности компьютерного магазина: поступление и реализация компьютеров.

Вариант 6. Автоматизированная разработка БД АС для учета деятельности аэропорта: учет выполнения рейсов.

Вариант 7. Автоматизированная разработка БД АС для учета деятельности железнодорожного вокзала: учет выполнения рейсов.

Вариант 8. Автоматизированная разработка БД АС для учета деятельности больницы: учет больных и наличия лекарств.

Вариант 9. Автоматизированная разработка БД АС для учета результатов сдачи экзаменов по изучаемым дисциплинам: заполнение экзаменационных ведомостей.

Вариант 10. Автоматизированная разработка БД АС для учета деятельности мастерской по ремонту часов: учет поступления и выполнения заказов и поступления и использования комплектующих в ходе выполнения заказов.

Вариант 11. Автоматизированная разработка БД АС для учета деятельности мастерской по ремонту обуви: учет поступления и выполнения заказов и поступления и использования расходных материалов в ходе выполнения заказов.

Вариант 12. Автоматизированная разработка БД АС для учета деятельности кондитерской фабрики: учет выпуска продукции.

Вариант 13. Автоматизированная разработка БД АС для учета деятельности обувной фабрики: учет выпуска продукции.

Вариант 14. Автоматизированная разработка БД АС для учета деятельности фармакологической фирмы: учет выпуска продукции.

Вариант 15. Автоматизированная разработка БД АС для учета деятельности фирмы по сборке компьютеров: учет выпуска продукции.

Вариант 16. Автоматизированная разработка БД АС для учета процесса изучения дисциплин учебной программы: заполнение журнала проведения занятий и учета посещаемости студентов.

Вариант 17. Автоматизированная разработка БД АС для учета деятельности университетской библиотеки: учет поступления книг в фонд библиотеки и их списания, а также выдачи и возврата книг студентами.

Вариант 18. Автоматизированная разработка БД АС для учета деятельности автомобильного вокзала: учет выполнения рейсов.

Вариант 19. Автоматизированная разработка БД АС для учета деятельности речного порта: учет выполнения рейсов.

Вариант 20. Автоматизированная разработка БД АС для учета деятельности гостиницы: учет поступающих и выбывающих постояльцев и их размещения по номерам.

Вариант 21. Автоматизированная разработка БД АС для учета деятельности оптового склада: учет поступления и отпуска товара.

Вариант 22. Автоматизированная разработка БД АС для учета деятельности магазина спортивной одежды и инвентаря: поступление и реализация товара.

Вариант 23. Автоматизированная разработка БД АС для учета деятельности мебельного магазина: поступление и реализация товара.

Вариант 24. Автоматизированная разработка БД АС для учета деятельности магазина строительных материалов: поступление и реализация товара.

Вариант 25. Автоматизированная разработка БД АС для учета деятельности отделов универсального магазина: поступление и реализация товара.

Примечание. Для каждого варианта определите не менее 7 сущностей, а для каждой сущности варианта – не менее 5 свойств. При сдаче лабораторных работ для каждой сущности должны быть заполнены не менее 5 записей.

1.1 ОПИСАНИЕ ЛАБОРАТОРНЫХ РАБОТ

Для всех приведенных выше вариантов предметных областей в процессе разработки БД выполняются следующие этапы проектирования:

- создание и настройка рабочего места для выполнения лабораторных работ;
- анализ ПрО, в ходе которого выявляются информационные объекты, соответствующие требованиям нормализации данных. Затем определяются связи между выделенными объектами и, таким

образом, строится информационно-логическая (концептуальная) модель ПрО в виде бумажного документа;

- на основе построенной концептуальной модели строится логическая модель БД, сохраняющаяся в памяти машины, где каждый информационный объект отображается в виде сущности, а связи между сущностями соответствуют связям между информационными объектами;

- логическая модель строится без учета требований СУБД, с помощью которой создается БД для хранения данных создаваемой модели. На следующем этапе выбирается СУБД, а модель ПрО приводится в соответствии с требованиями этой СУБД. Таким образом создается физическая модель ПрО, по которой генерируется схема БД;

- с использованием языка макросов [ERwin](#), разрабатываются шаблоны для генерирования [SQL](#)-скриптов создания последовательностей, генераторов и триггерных функций;

- с использованием языка макросов [ERwin](#), разрабатываются шаблоны для генерирования функций хранимых процедур, обеспечивающих выполнение операций вставки, обновления и удаления во всех таблицах создаваемой БД;

- генерирование [SQL](#)-сценария создания БД и его выполнение в СУБД [PostgreSQL](#).

Содержание работ по выполнению перечисленных заданий приведено в Главе 2 настоящих методических указаний.

1.2 ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТНЫХ МАТЕРИАЛОВ

По каждой лабораторной работе должен быть оформлен отчет, содержащий следующие разделы:

1. Цель выполнения работы.
2. Описание содержания выполненных работ.
3. Выводы.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Проектирование
информационных систем
с помощью CASE-средств**

**Отчет
по выполнению лабораторной работы №2
«Разработка концептуальной схемы БД»**

Отчет составил:
Бакалавр группы ИВТВМбд-4
Петров П.П.

Отчет принял:
Преподаватель кафедры ИКТРС
д.т.н. Токмаков Г.П.

Ульяновск
УлГТУ
2016

Рис. 1.1 Титульный лист отчета

В разделе описания содержания работ, структура которого зависит от выполняемой работы, должны быть приведены:

- текст, поясняющий ход выполнения работы, разбивается на разделы и подразделы в зависимости от выполняемой лабораторной работы (см. ниже);
- в тексте описания должны быть приведены листинги разработанного кода (при их наличии);
- в тексте описания должны быть приведены скриншоты, иллюстрирующие практический ход выполнения и результаты работы.

В заключении (раздел «Выводы») должны быть подытожены результаты выполненной работы.

Образец титульного листа отчета приведен на Рис. 1.1, а ниже приведен перечень разделов отчетов по каждой лабораторной работе.

Лабораторная работа №1. Изучение свободно распространяемых версий СУБД PostgreSQL и CASE-средства ERwin

1. Цель выполнения работы.
2. Описание содержания выполненных работ.
 - 2.1. Краткое описание СУБД PostgreSQL.
 - 2.2. Краткое описание CASE-средства ERwin (описание интерфейсного окна).
 - 2.3. Описание процесса установки СУБД PostgreSQL и CASE-средства ERwin с приведением скриншотов.
 - 2.4. Описание процесса создания пустой БД и ее псевдонима.
3. Выводы.

Лабораторная работа №2. Разработка концептуальной схемы ПрО (Вариант №)

1. Цель выполнения работы.
2. Описание содержания выполненных работ.
 - 2.1. Описание процесса выделения сущностей ПрО в рамках требований Вашего варианта.

Для каждой сущности должен быть определен ее тип с обоснованием.

Привести обоснование для каждой сущности с точки зрения ее необходимости для организации бизнес-процессов автоматизации деятельности Вашей ПрО.

2.2. Определение перечня атрибутов для каждой сущности. В перечень должны быть включены только атрибуты, необходимые для организации бизнес-процессов автоматизации деятельности Вашей ПрО.

2.3. Определение связей между сущностями. Для каждой связи определите тип (идентифицирующая или не идентифицирующая).

3. Выводы.

Лабораторная работа №3. Разработка логической схемы

1. Цель выполнения работы

2. Описание содержания выполненных работ.

2.1. Описание процесса создания сущностей с помощью CASE-средства ERwin.

2.2. Описание процесса создания атрибутов сущностей с помощью CASE-средства ERwin.

2.3. Описание процесса определения связей между сущностями. Для каждой связи реализуйте правила сохранения ссылочной целостности (определенные в работе №2) с помощью CASE-средства ERwin.

3. Выводы.

Лабораторная работа №4. Разработка физической схемы

1. Цель выполнения работы.

2. Описание содержания выполненных работ.

2.1. Выбор сервера для реализации БД по разработанной физической модели БД.

2.2. Описание процесса создания таблиц реляционной БД с помощью CASE-средства ERwin на основе сущностей, определенных в работе №3.

2.3. Описание процесса создания колонок таблиц реляционной БД с помощью CASE-средства ERwin на основе атрибутов сущностей, определенных в работе №3.

2.4. Описание процесса определения связей между сущностями. Для каждой связи реализуйте правила сохранения ссылочной целостности (определенные в работе №2) с помощью CASE-средства ERwin.

2.5. Описание процессов создания пустой БД с помощью СУБД PostgreSQL и источника данных с помощью средств ОС Windows.

2.6. Установка связи между CASE-средством ERwin и СУБД PostgreSQL (т.е. с созданным источником данных).

2.7. Генерирование и просмотр SQL-сценария создания БД.

Сгенерируйте код с помощью кнопки «Preview...» и убедитесь, что в нем присутствуют скрипты для формирования таблиц, первичных и вторичных ключей.

Опишите этот процесс.

3. Выводы.

Лабораторная работа №5. Разработка последовательностей, генераторов и триггеров для ключевых атрибутов таблиц БД

1. Цель выполнения работы.

2. Описание содержания выполненных работ.

2.1. Общее описание генераторов и триггеров (копирование материалов лекций не допускается).

2.2. Краткое описание языка макросов ERwin в части макросов, требуемых для разработки генераторов, триггеров и хранимых процедур (описание языка можно найти в Интернете).

2.3. Разработка кода шаблона для формирования:

- последовательностей для всех первичных ключей Вашей БД;
- генераторов для формирования значений последовательностей;
- триггеров для занесения сформированных значений в поля ключевых атрибутов.

2.4. Подключение к СУБД PostgreSQL и генерирование SQL-сценария создания БД.

Сгенерируйте код с помощью кнопки «Preview...» и убедитесь, что в нем присутствуют скрипты для формирования:

- последовательностей для всех первичных ключей Вашей БД;

- генераторов для формирования значений последовательностей;
- триггеров для занесения сформированных значений в поля ключевых атрибутов.

Опишите этот процесс.

3. Выводы

Лабораторная работа №6. Разработка хранимых процедур для вставки, обновления и удаления записей таблиц БД

1. Цель выполнения работы.

2. Описание содержания выполненных работ.

2.1. Общее описание хранимых процедур (копирование материалов лекций не допускается).

2.2. Разработка кода шаблонов формирования хранимых процедур для:

- вставки записей таблиц БД;
- обновления записей таблиц БД;
- удаления записей таблиц БД.

2.3. Описание SQL-сценарий создания БД.

Подключитесь к СУБД PostgreSQL и сгенерируйте SQL-сценарий создания БД с помощью кнопки «Preview...». Просмотрите полученный код и убедитесь, что в нем присутствуют скрипты формирования хранимых процедур для:

- вставки записей таблиц БД;
- обновления записей таблиц БД;
- удаления записей таблиц БД.

Опишите этот процесс.

3. Выводы.

Лабораторная работа №7. Генерирование SQL-сценария и создание схемы БД в СУБД PostgreSQL

1. Цель выполнения работы.

2. Описание содержания выполненных работ.

2.1. Установка соединения между CASE-средством ERwin и СУБД PostgreSQL генерирование SQL-сценария создания БД с помощью кнопки «Generate...».

2.2. Просмотр результата выполнения SQL-сценария создания БД с помощью интерфейсного приложения pgAdmin III СУБД PostgreSQL.

При корректном выполнении SQL-сценарий создания БД в должны быть созданы требуемое количество:

- таблиц;
- последовательностей;
- триггерных функций;
- функций хранимых процедур.

В данном разделе приведите краткое описание структуры БД, включающей в свой состав скрипты для формирования таблиц, первичных и вторичных ключей, последовательностей, триггерных функций, функций хранимых процедур.

Приведите также скриншот левой панели интерфейсного приложения pgAdmin III с раскрытыми узлами «Таблицы ()», «Последовательности ()», «Триггерные функции ()», «Функции ()» в узле дерева Вашей БД.

3. Выводы.

Для демонстрации выполненной работы ее результаты представляются на машинном носителе. Фрагменты кода, используемые для пояснения работы, должны быть представлены в тексте отчета в виде листингов.

Примечание. Запрещается использовать в текстах отчетов фрагменты настоящих методических указаний, а также учебного пособия по курсу лекций, т.е. отчеты нужно составлять «своими словами».

ГЛАВА 2. СОДЕРЖАНИЕ РАБОТ ПО ВЫПОЛНЕНИЮ ЗАДАНИЙ

Для демонстрации порядка и содержания выполнения лабораторных работ, описываемых в данных методических указаниях, рассмотрим пример разработки схемы БД для организации учебного процесса. Подробное описание этой ПрО приведено в работе [1].

2.1 ЛАБОРАТОРНАЯ РАБОТА 1. ИЗУЧЕНИЕ, УСТАНОВКА И НАСТРОЙКА СУБД POSTGRESQL И CASE-СРЕДСТВА ERWIN

Целью данной работы является овладение навыками работы по установке и настройке свободно распространяемых программных средств СУБД [PostgreSQL](#) и CASE-средства [ERwin](#) (trial-версии), создании пустой БД с помощью СУБД и создании ее псевдонима для доступа к ней с [ERwin](#).

2.1.1 УСТАНОВКА СУБД POSTGRESQL

Для установки СУБД [PostgreSQL](#) войдите на сайт и скачайте одну из последних версий для [Windows](#).

Установка данного программного средства не требует отдельного описания, так как в рамках изучения дисциплины «Базы данных» Вы эту установку уже производили.

Желательно выбрать для установки полную версию дистрибутива, включающую в свой состав необходимые драйверы. Полные версии дистрибутивов можно определить по объему памяти. Например, для версии [postgresql-9.5.3-1-windows-x64.exe](#) объем занимаемой памяти больше 60 Мб.

2.1.2 УСТАНОВКА CASE-СРЕДСТВА ERWIN

Для установки CASE-средства ERwin войдите на сайт по адресу <https://www.ca.com/us/register/forms/ca-erwin-data-modeler-evaluation-software.aspx> и заполните предложенную форму.

В ответ на указанный в форме почтовый ящик приходит письмо с инструкцией и ссылками на дистрибутив и файл лицензии.

Инструкция на английском, но ее суть довольно простая, которая заключается в выполнении следующих шагов:

1. Скачайте и скопируйте дистрибутив и лицензионный файл на Ваш компьютер.
2. Если лицензионный файл сохраняется как **.txt**, то переименуйте его в файл с расширением **.lic**.
3. Установите со скачанного дистрибутива, а затем запустите программное средство **CA ERwin Data Modeler**.
4. Выберите в меню «**Help**» пункт **Licensing**, затем пункт «**Select the Licensing**».
5. Щелкните по кнопке «**Install License File**» и укажите путь к сохраненному лицензионному файлу (см. Шаг 1).

2.1.3 СОЗДАНИЕ ПУСТОЙ БД

После построения логической, а затем физической модели ПрО Вашего варианта с помощью CASE-средства ERwin генерируется SQL-сценарий для формирования по этому сценарию БД в СУБД PostgreSQL. Процесс формирования БД запускается с CASE-средства ERwin, а для этого требуется, чтобы оно было подключено к соответствующей БД. Поэтому на данный момент должна быть создана пустая БД, т.е. выполнен запрос **CREATE DB uchPrc** для нашего примера. Для этого выполните следующие шаги.

1. Запустите программное средство pgAdmin. На левой панели «**Браузер объектов**» главного окна раскройте узел сервера, затем щелкните по узлу «**Базы данных**» и в контекстном меню выберите пункт «**Новая база данных ...**».
2. В окне «**Новая база данных**» в поле **Name** введите имя Вашей БД. Для рассматриваемой в методических указаниях ПрО введем наименование **uchPrc**.

3. На левой панели «Браузер объектов» под узлом «Базы данных» появляется дочерний узел с именем **uchPrc**. Щелкните по этому узлу и на правой нижней панели с названием «Панель SQL» сможете увидеть скрипт для создания БД.

На этом создание пустой БД завершается.

2.1.4 СОЗДАНИЕ ПСЕВДОНИМА БД

Сложность использования CASE-средства **ERwin** для генерирования SQL-сценария создания БД заключается в том, что скрипт создается в **ERwin**, а выполняется в СУБД.

Поэтому после реализации физической модели ПрО с помощью **ERwin** необходимо подключить его к пустой БД для формирования ее схемы, включающей все необходимые компоненты, такие как таблицы, первичные и вторичные ключи, последовательности, триггерные функции и функции хранимых процедур.

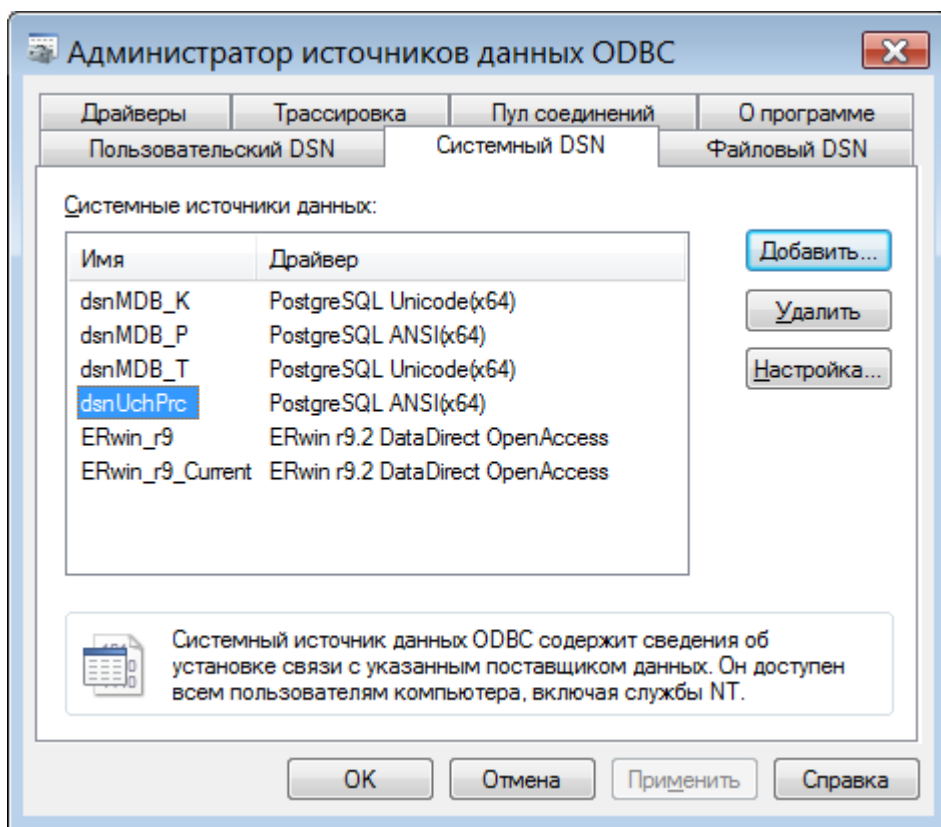


Рис. 2.1 – Диалог «Администратор источников данных ODBC»

Для облегчения подключения сторонних программных средств к БД создается ее псевдоним, связанный с параметрами подключения к БД. Именно этот псевдоним будет использоваться при подключении **ERwin** к СУБД **PostgreSQL** для выполнения в ней **SQL**-сценария создания БД. Это гораздо удобнее, чем иметь дело с множеством параметров подключения к БД.

Для создания псевдонима БД (или источника данных в терминологии **Windows**) вызовите Панель управления **Windows**, раскройте в нем раздел «Система и безопасность», а затем раздел «Администрирование» и на правой панели Проводника дважды щелкните по узлу «Источник данных (ODBC)».

В диалоговом окне «Администратор источников данных ODBC» перейдите на вкладку «Системный DSN» и нажмите на кнопку «Добавить» (см. Рис. 2.1).

В окне «Создание нового источника данных» выберите пункт **PostgreSQL ANSI(x64)** и нажмите кнопку «Готово» (см. Рис. 2.2).

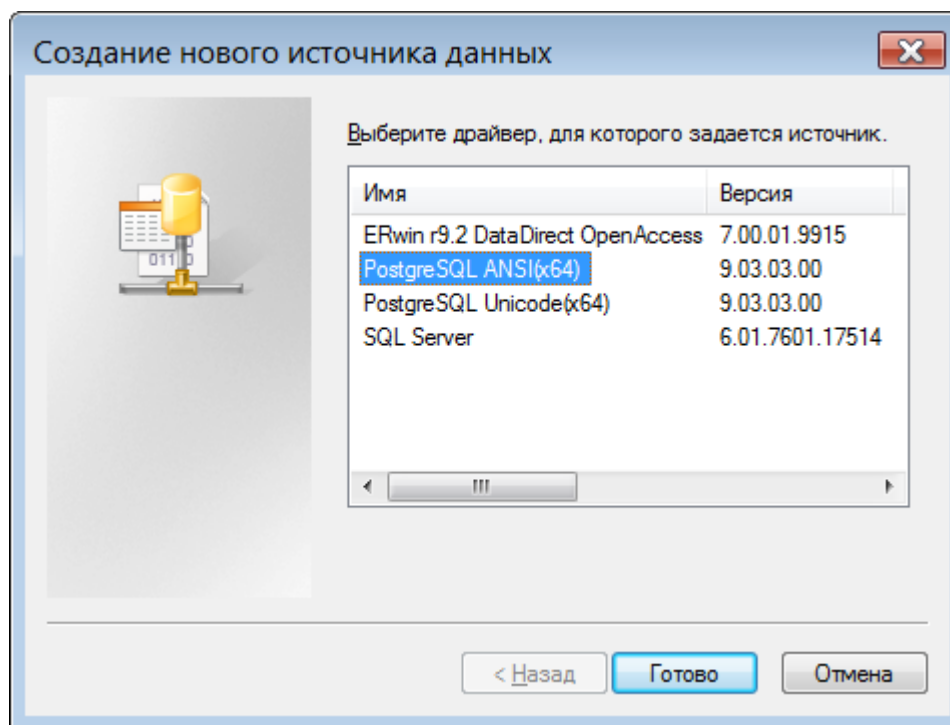


Рис. 2.2 – Диалог «Создание нового источника данных»

После этого появляется окно «PostgreSQL ANSI ODBC Driver (psqlODBC) Setup», в котором введите значения параметров подключения к Вашей БД как показано на (см. Рис. 2.3).

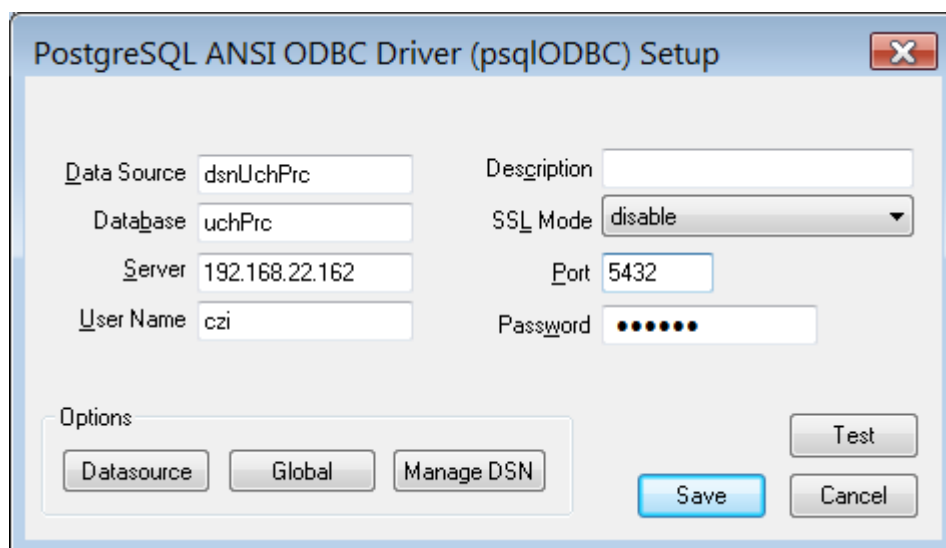


Рис. 2.3 – Диалог «PostgreSQL ANSI ODBC Driver (psqlODBC) Setup»

Обратите внимание на то, что псевдоним БД вводится в поле «Data Source», а в остальных полях вводятся параметры подключения, связанные с этим псевдонимом.

Для проверки корректности введенных данных нажмите на кнопку «Test» и в случае получения сообщения «Connection successful» нажмите на кнопку «Save» и введенный комплекс данных будет сохранен под псевдонимом.

2.2 ЛАБОРАТОРНАЯ РАБОТА 2. РАЗРАБОТКА КОНЦЕПТУАЛЬНОЙ СХЕМЫ ПРО

Целью данной работы является овладение навыками анализа предметной области и разработки концептуальной схемы БД для заданного варианта предметной области.

Перед выполнением данной лабораторной работы внимательно изучите раздел 5.1 работы [1].

Определение сущностей, их свойств и связей между ними путем анализа ПрО осуществляется без использования средств автоматизированного проектирования, поэтому такая модель используется в качестве исходных данных для разработки логической и физической моделей с помощью CASE-средства ERwin и называется концептуальной моделью ПрО. Элементы и связи концептуальной модели обозначаются средствами естественного языка.

Сведения, необходимые для построения концептуальной модели БД, извлекаются путем дискретизации информации, содержащейся в документах, сопровождающих деятельность в автоматизируемой ПрО. Любой документ, используемый в ходе деятельности организации или предприятия, содержит информацию, описывающую сущности, представленные набором количественных параметров, называемых показателями.

С целью выявления показателей документа проводится его анализ, в ходе которого определяется совокупность описываемых сущностей, их характеристик или реквизитов. На основе реквизитного состава определяются показатели, отражающие содержание документа.

При этом выполняются следующие этапы:

1. Анализ системы документов предметной области.
2. Выделение сущностей и их реквизитов (характеристик).
3. Определение связей между сущностями, т.е. выявление зависимых реквизитов сущностей.

2.2.1 ВЫДЕЛЕНИЕ СУЩНОСТЕЙ ПРО

Для выполнения перечисленных этапов работ по разработке концептуальной схемы необходимо располагать описанием заданной предметной области (ПрО), охватывающее ее реальные сущности и процессы, определяющее все необходимые источники информации для удовлетворения предполагаемых запросов пользователя и потребности в обработке данных.

В работе [1] анализ ПрО заключается в изучении соответствующих документов, вырабатываемых в ходе реализации учебного процесса текущего семестра на кафедре.

В результате выполнения анализа ПрО выявляются следующие сущности рассматриваемой ПрО:

Преподаватель – список преподавателей кафедры;

Кафедра – список кафедр;

Студент – список студентов, обучающихся на кафедре;

Группа – список групп студентов, выпускаемых кафедрой;

Предмет – список предметов, изучаемых студентами кафедры;

Изучение – сущность, описывающая процесс изучения студентами предметов и содержащая данные о преподавателях, ведущих занятия;

Успеваемость – сущность, описывающая процесс тестирования студентов по изучаемым предметам и содержащая сведения о результатах тестирования студентов по изучаемым предметам.

При описании выполняемой лабораторной работы каждая выделенная сущность должна быть обоснована с точки зрения необходимости ее использования при автоматизации бизнес-процессов ПрО.

2.2.2 ОПРЕДЕЛЕНИЕ ТИПОВ СУЩНОСТЕЙ

Анализ выявленных сущностей показывает, что не все из них имеют одинаковую природу. Сущности **Преподаватель**, **Кафедра**, **Студент**, **Группа**, **Предмет** отражают объекты, воспринимаемые органами чувств и не требующие данных других сущностей для идентификации их экземпляров. Например, экземпляры объекта **Предмет** (допу-

стим, предмет «Физика») существуют и идентифицируются независимо от того, кто их преподает и изучает. Такие объекты в схемах БД называются независимыми. Независимыми в нашей концептуальной схеме являются и другие объекты описываемой группы, а именно, **Преподаватель**, **Кафедра**, **Студент**, **Группа**.

С другой стороны, сущности **Изучение** и **Успеваемость** описывают не реально существующие объекты, а процессы, совершаемые независимыми объектами. Для идентификации экземпляров таких объектов требуется привлечь данные других сущностей. Например, экземпляры сущности **Изучение** определяются тем:

- какой предмет изучается (идентификатор экземпляра объекта **Предмет**);
- кто преподает этот предмет (идентификатор экземпляра объекта **Преподаватель**);
- кто изучает (идентификатор экземпляра объекта **Группа**).

Такие сущности в схемах БД называются зависимыми. Эта зависимость определяется тем, что их первичные ключи состоят из первичных ключей независимых сущностей, являющихся субъектами и объектами процесса, описываемого данной зависимой сущностью. Таким образом, в нашей концептуальной схеме сущности **Изучение**, **Успеваемость** являются зависимыми.

2.2.3 ВЫДЕЛЕНИЕ РЕКВИЗИТОВ

Путем изучения документов ПрО легко определяется состав реквизитов каждой сущности. В общем случае для сущностей можно определить бесконечно большое количество реквизитов, но для составления схемы БД требуется выбрать только те, которые необходимы для автоматизации бизнес-процессов ПрО. Для сущностей нашей ПрО можно определить следующий состав реквизитов:

- **ПРЕПОДАВАТЕЛЬ**: Код преподавателя, Фамилия И.О., Уч. степень, Уч. звание, Код кафедры;
- **КАФЕДРА**: Код кафедры, Название кафедры, Телефон, Заведующий;
- **СТУДЕНТ**: Код студента, Фамилия И.О., Год рождения, Адрес, Балл, Код группы;

- **ГРУППА**: Код группы, Количество студентов, Средний балл;
- **ПРЕДМЕТ**: Код предмета, Название предмета, Часов лекций, Часов практики;
- **ИЗУЧЕНИЕ**: Вид занятия, Часы занятия, Код предмета, Код преподавателя, Код группы;
- **УСПЕВАЕМОСТЬ**: Код группы, Код студента, Код предмета, Код преподавателя, Оценка.

Допускается также определение состава и структуры данных исходя из знания ПрО (по жизненному опыту, путем опроса специалистов), хотя использование документов для этой цели является предпочтительнее.

2.2.4 ВЫДЕЛЕНИЕ СВЯЗЕЙ МЕЖДУ ОБЪЕКТАМИ И ОПРЕДЕЛЕНИЕ ИХ ТИПОВ

Далее для построения концептуальной схемы требуется установить связи между выявленными объектами ПрО, а для этого необходимо установить функциональные связи между атрибутами объектов или реквизитами документов. Функциональные связи между атрибутами объектов можно использовать также для уточнения состава объектов ПрО.

Связь устанавливается между двумя информационными объектами. Связи между объектами существуют, если логически взаимосвязаны экземпляры этих информационных объектов. Например, возьмем два объекта: **СТУДЕНТ** и **ГРУППА**.

Связь экземпляров этих объектов определяется тем обстоятельством, что каждый студент обучается в какой-либо группе, что является одним из свойств студента. В описываемом примере этим свойством является «Код группы». С другой стороны, этим свойством обладает и объект **ГРУППА** (совпадение наименований таких свойств необязательно, хотя и облегчает понимание схемы).

Таким образом, связь экземпляров этих объектов определяется совпадением значений этих свойств этих объектов. Например, если значение свойства «Код группы» экземпляра объекта **СТУДЕНТ** равно «1», то это означает, что данный конкретный студент связан с эк-

экземпляром объекта **ГРУППА**, свойство «Код группы» которого также равно «1».

Связь между парой объектов **КАФЕДРА → ПРЕПОДАВАТЕЛЬ** имеет аналогичную природу, отражающей принадлежность экземпляров одного объекта экземплярам другого объекта.

Другая группа связей определяется вокруг сущности **ИЗУЧЕНИЕ**.

В каждой группе в течение семестра проводится ряд занятий, что отражается связью **ГРУППА → ИЗУЧЕНИЕ**.

Занятия проводятся по конкретным предметам, что определяет связь между объектами **ПРЕДМЕТ → ИЗУЧЕНИЕ**.

Занятия по конкретным предметам проводят конкретные преподаватели, что описывается связью **ПРЕПОДАВАТЕЛЬ → ИЗУЧЕНИЕ**.

Объект **УСПЕВАЕМОСТЬ** является еще одной сущностью, определяющей связи с другими сущностями. Эта сущность, содержащая данные об успеваемости (оценку) конкретного студента по конкретному виду занятия (виды занятий описываются сущностью **ИЗУЧЕНИЕ**), определяет связи **СТУДЕНТ → УСПЕВАЕМОСТЬ** и **ИЗУЧЕНИЕ → УСПЕВАЕМОСТЬ**.

2.2.5 ОПРЕДЕЛЕНИЕ ТИПОВ СВЯЗЕЙ

Связь **ГРУППА → СТУДЕНТ** определяют ассоциации между независимыми сущностями и называется не идентифицирующей, т.к. для идентификации экземпляра группы не требуются данные обучающихся в ней студентов, а для идентификации студентов не требуется знать, в какой группе он обучается. Хотя свойство «Код группы», определяющее в какой группе учится каждый студент, но это свойство не используется для идентификации студента. Для этого можно использовать такие свойства, как «Номер студенческого билета» или «Номер зачетной книжки».

Аналогичный характер имеет связь **КАФЕДРА → ПРЕПОДАВАТЕЛЬ**, т.е. данная связь также называется неидентифицирующей.

Совершенно другую природу имеют связи между парами объектов: **ГРУППА → ИЗУЧЕНИЕ**, **ПРЕДМЕТ → ИЗУЧЕНИЕ**, **ПРЕПОДАВАТЕЛЬ → ИЗУЧЕНИЕ**. Отличие этих связей от рассмотренных определяется тем фактом, что объект **ИЗУЧЕНИЕ** описывает процесс, а информацион-

ные объекты **ГРУППА**, **ПРЕПОДАВАТЕЛЬ**, **ПРЕДМЕТ** являются субъектами и объектами этого процесса.

Связи **СТУДЕНТ → УСПЕВАЕМОСТЬ** и **ИЗУЧЕНИЕ → УСПЕВАЕМОСТЬ** определяются сущностью-процессом **УСПЕВАЕМОСТЬ**, и также определяют зависимость ее экземпляров от соответствующих экземпляров сущностей **СТУДЕНТ** и **ИЗУЧЕНИЕ**.

Таким образом, для идентификации экземпляров объекта **ИЗУЧЕНИЕ** требуются значения первичных ключей объектов **ГРУППА**, **ПРЕДМЕТ**, **ПРЕПОДАВАТЕЛЬ**, а для идентификации экземпляров объекта **УСПЕВАЕМОСТЬ** – значения первичных ключей объектов **СТУДЕНТ** и **ИЗУЧЕНИЕ**.

Описанные факторы позволяют установить, что связи **ГРУППА → ИЗУЧЕНИЕ**, **ПРЕДМЕТ → ИЗУЧЕНИЕ**, **ПРЕПОДАВАТЕЛЬ → ИЗУЧЕНИЕ**, **СТУДЕНТ → УСПЕВАЕМОСТЬ** и **ИЗУЧЕНИЕ → УСПЕВАЕМОСТЬ** являются идентифицирующими.

Обратите внимание на то, что объект: –

- **ИЗУЧЕНИЕ** выполняет роль объекта-связки в отношениях типа **М:М** между сущностями **ГРУППА → ПРЕДМЕТ**, **ГРУППА → ПРЕПОДАВАТЕЛЬ**, **ПРЕДМЕТ → ПРЕПОДАВАТЕЛЬ**;

- **УСПЕВАЕМОСТЬ** выполняет роль объекта-связки в отношениях типа **М:М** сущностей **СТУДЕНТ** и **ИЗУЧЕНИЕ**. Отношения между этими сущностями определяются тем, что одному студенту соответствует много видов занятий, отображаемых объектом **ИЗУЧЕНИЕ**, а один вид занятия проводится со многими студентами.

Можно сделать еще один важный вывод: дочерние сущности идентифицирующих связей становятся зависимыми. При первоначальной установке сущностей на диаграмме с помощью кнопки **Entity** все сущности по умолчанию определяются как независимые.

2.2.6 КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ ПРО

На основе выявленных объектов их свойств и связей между объектами строится концептуальная модель, которая отображает данные Про в виде совокупности информационных объектов и связей между ними. Эта модель является основой для разработки логической модели БД.

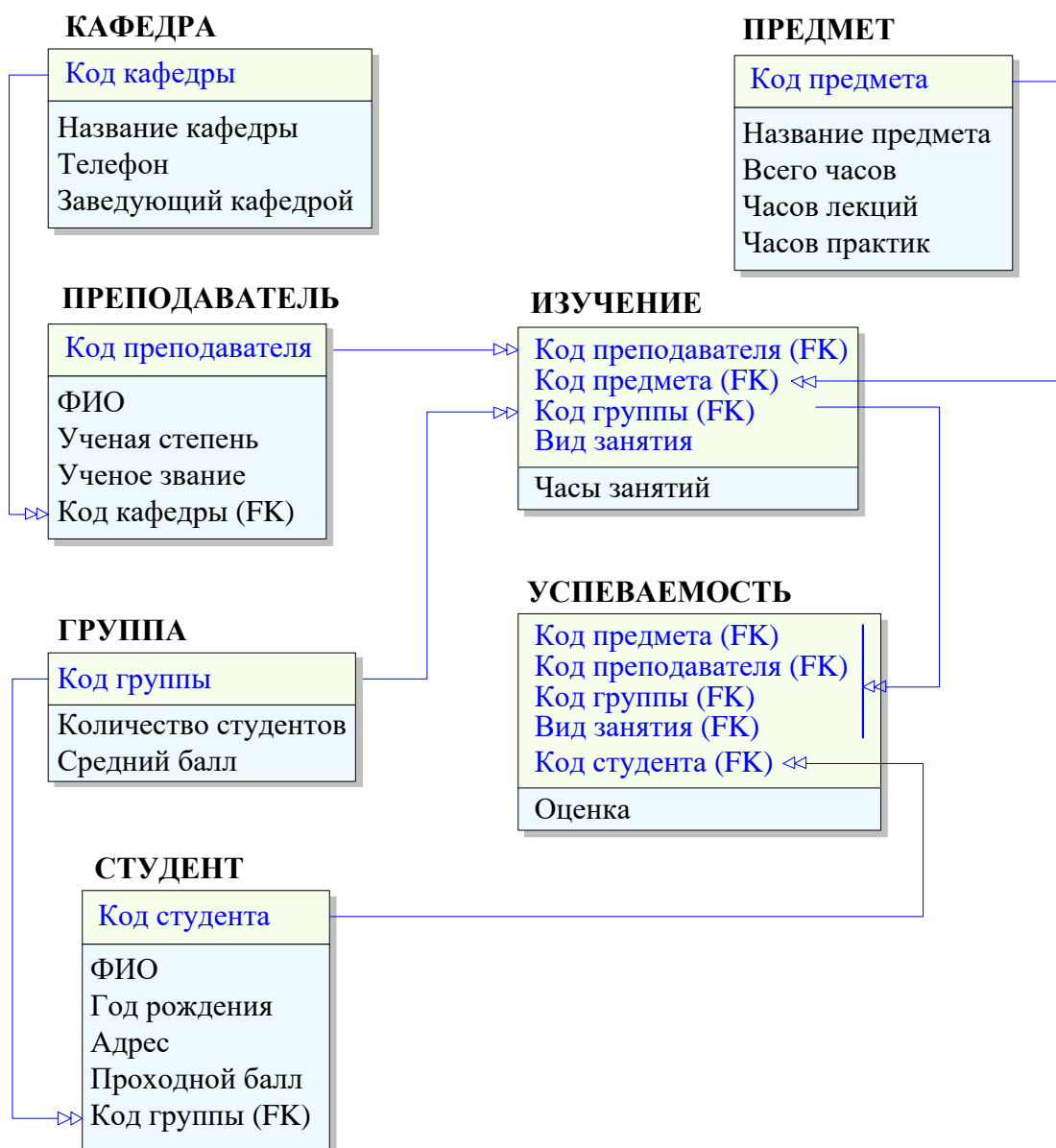


Рис. 2.4 – Концептуальная модель ПрО «Учебный процесс»

На Рис. 2.4 приведена схема данных, наглядно отображающая логическую структуру БД. На этой схеме прямоугольники отображают таблицы БД с полным списком их полей, а связи показывают, по каким полям осуществляется взаимосвязь таблиц. Имена ключевых полей для наглядности выделены и находятся в верхней части полного списка полей каждой таблицы.

По полученной модели далее строится логическая модель ПрО, которая описывается в следующей лабораторной работе.

2.3 ЛАБОРАТОРНАЯ РАБОТА 3. РАЗРАБОТКА ЛОГИЧЕСКОЙ СХЕМЫ ПРО

Цель данной лабораторной работе заключается в реализации концептуальной модели, разработанной при выполнении лабораторной работы №2, в виде логической схемы созданной с помощью CASE-средства ERwin.

Перед выполнением данной лабораторной работы внимательно изучите раздел 5.2 работы [1].

Основные компоненты диаграммы ERwin – это сущности, атрибуты и связи. Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Общий порядок создания логической схемы ПрО с помощью ERwin описывается следующим образом.

Работа по созданию логической схемы начинается с введения в диаграмму сущностей, определенных на концептуальной схеме в виде информационных объектов. В рассматриваемом нами примере это объекты Преподаватель, Кафедра, Студент, Группа, Предмет, Изучение, Успеваемость.

Определив сущности, необходимо ввести в схему атрибуты этих сущностей. Задав атрибуты сущностей, мы определяем основу для создания таблиц БД на этапе разработки физической схемы ПрО.

На последнем этапе определяются связи между введенными таблицами. В отличие от этапа разработки концептуальной схемы на данном этапе определяются свойства связей, поддерживаемые после создания БД на уровне СУБД.

2.3.1 СОЗДАНИЕ СУЩНОСТЕЙ

Набор сущностей предметной области определяется на логическом уровне работы ERwin. Поэтому в первую очередь перейдите на закладку «Уровень сущностей» с помощью выпадающего списка на панели инструментов интерфейсного окна ERwin.

Для построения корректной логической схемы необходимо иметь в виду, что ERwin поддерживает следующие типы сущностей: независимые и зависимые.

Когда вы добавляете сущность в диаграмму, ее тип определяется по типу связи, в которую она входит. Например, когда вы добавляете новую сущность, она изначально определяется как независимая. Когда вы соединяете эту сущность с другой сущностью, именно тип связи определяет ее тип. Если эта сущность является дочерней, а тип связи определяется как идентифицирующая, то сущность становится зависимой. Если при этом тип связи определяется как неидентифицирующая, то сущность остается независимой.

Определение типов связей, а также типов сущностей – непростые задачи для обучаемых, т.к. критерии их определения неоднозначны и их очень сложно применить на практике.

Поэтому предлагается следующий способ определения типов сущностей и связей при выполнении лабораторных работ.

В соответствии с определением зависимые сущности для идентификации своих экземпляров требуют данные других сущностей. Например, для определения экземпляра сущности «Изучение» необходимо определить код изучаемого предмета, код преподавателя, ведущего занятия по данному предмету, а также код группы, которая изучает данный предмет.

Таким образом, сущность «Изучение» описывает процесс, а сущности («Предмет», «Преподаватель» и «Группа»), используемые для определения ее экземпляра, являются субъектами и объектами этого процесса.

Отсюда следует, что связи между сущностью-процессом и определяющими ее сущностями-субъектами, сущностями-объектами должны быть идентифицирующими.

Другими словами, сущности, описывающие процессы, всегда являются зависимыми, что и определяет природу связей этой сущности с сущностями, участвующими в этом процессе в качестве субъектов и объектов.

1. Выделите в палитре инструментов кнопку сущности. Затем щелкните по чистой области диаграммы, и на поле появится прямоугольник, изображающий новую сущность с именем «E/№», где №, номер текущей сущности.

2. Измените название по умолчанию на «Студент», введя это имя с клавиатуры.

3. Таким же образом вставьте в диаграмму еще шесть сущностей – «Группа», «Преподаватель», «Кафедра», «Предмет», «Изучение» и «Успеваемость».

Каждая из сущностей, как и всякий объект ER-диаграммы, обладает контекстным меню, для вызова которого необходимо щелкнуть по прямоугольнику сущности правой кнопкой мыши.

Выберите пункт меню **Entity Properties** для вызова редактора сущности, позволяющего изменять свойства выбранной сущности.

В верхней части окна редактора сущностей находится список всех сущностей диаграммы. Далее имеется поле **Name**, в котором высвечивается имя сущности, которое может редактироваться (Рис. 2.5).

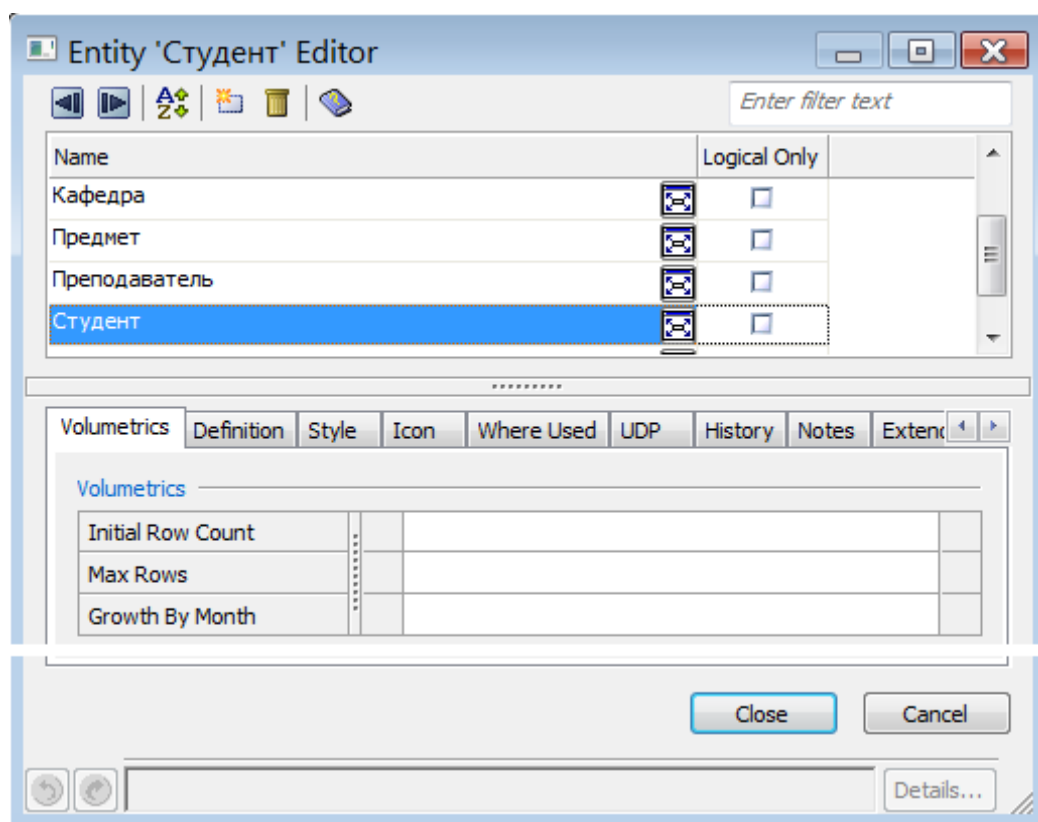


Рис. 2.5– Редактор сущности

Ниже в окне редактора находится ряд закладок:

- **Definition** – на этой странице вводится определение сущности;
- **Note, Note2, Note3** – используются для ввода произвольного текста, связанного с сущностью;
- **UDP** – используется для определения свойств сущности;
- **Icon** – для наглядности каждой сущности может быть присвоена иконка, которая выводится рядом с ее названием.

2.3.2 УСТАНОВКА АТТРИБУТОВ СУЩНОСТИ

Теперь необходимо определить атрибуты введенных сущностей диаграммы. Для этого можно щелкнуть правой кнопкой мыши (ПКМ) по прямоугольнику сущности на диаграмме и выбрать в контекстном меню (КМ) пункт «**Attribute Properties ...**».

С этой же целью можно воспользоваться возможностями левой панели, которая, хотя и является более трудоемкой, но обеспечивает большую наглядность процесса.

В этом случае:

- на левой панели выберите закладку «**Model**»;
- в дереве «**Model_1**» раскройте узел «**Entities**»;
- раскройте узел с наименованием сущности, например, «**Студент**»;
- в раскрывшемся списке выберите узел «**Attributes**» и нажатием ПКМ на этом узле выберите единственный пункт «**New**»;
- в появившемся узле введите наименование атрибута, например, «**Номер студента**»;
- вызовите редактор атрибутов щелчком ПКМ на наименовании атрибута и выбором пункта КМ «**Properties ...**» (см. Рис. 2.6).

С помощью вызванного диалога в таблице, расположенной в верхней части, с помощью выпадающего списка в столбце «**Logical Data Type**» выберите тип данных «**INTEGER**». Если для данного атрибута определен домен, то в столбце «**Pattern Domain**» с помощью выпадающего списка выберите этот домен; в этом случае тип данных определяется автоматически как база для определения домена.

Аналогично определяем остальные атрибуты сущности «**Студент**», а также всех сущностей на диаграмме.

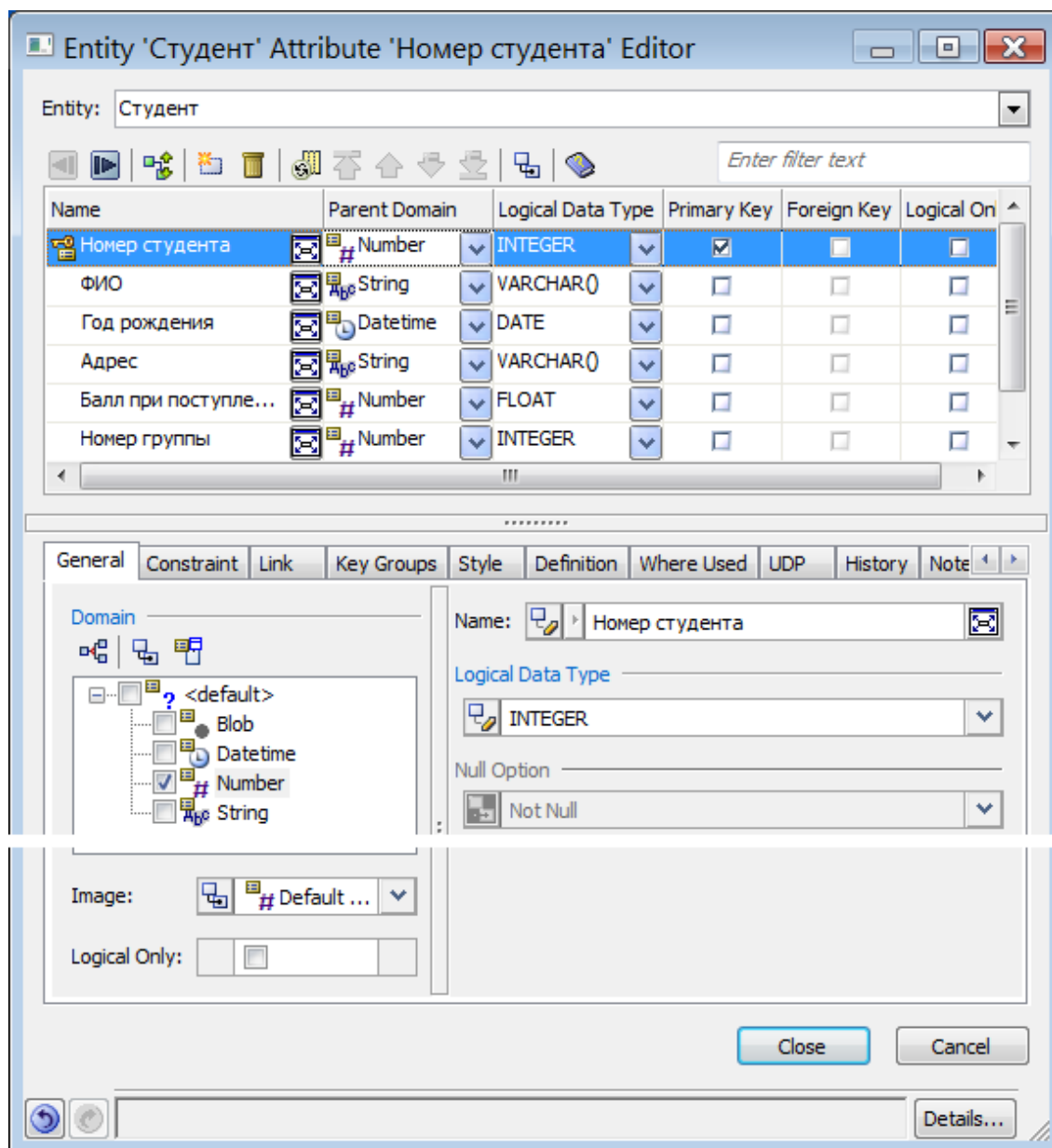


Рис. 2.6 – Редактор атрибутов сущностей


2.3.3 УСТАНОВКА СВОЙСТВ СВЯЗЕЙ МЕЖДУ СУЩНОСТЯМИ


Связь используется в логической модели для указания наличия ассоциации между двумя сущностями. Установка связей вносит определенные изменения на диаграмме. Во-первых, связанные сущности соединяются линией. Вид линии зависит от типа связи: для не идентифицирующих связей используется штриховая линия, а для идентифицирующих – сплошная.


Кроме этого, при установке не идентифицирующей связи атрибут первичного ключа родительской сущности мигрирует в дочернюю в качестве вторичного ключа, а при установке идентифицирующей связи – в качестве первичного ключа.

2.3.3.1 Типы связей

В логической модели с помощью соответствующих кнопок на панели инструментов можно создать следующие типы связей:

- идентифицирующая (кнопка  «Identifying relationship» на ПИ), используется, если необходимо идентифицировать экземпляр дочерней сущности через ее связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, входят в первичный ключ дочерней сущности. Идентифицирующая связь изображается сплошной линией;

- не идентифицирующая (кнопка  «Non-identifying relationship» на ПИ), используется, если экземпляр дочерней сущности идентифицируется иначе, чем через связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в состав неключевых атрибутов дочерней сущности. Неидентифицирующая связь изображается пунктирной линией;

- многие-ко-многим (кнопка  «Many-to-many relationship» на ПИ), используется для моделирования ситуации, когда одной сущности соответствует один или несколько экземпляров второй сущности, а экземпляру второй сущности соответствует один или несколько экземпляров первой сущности, отражается логической моделью «многие-ко-многим» между данными сущностями. Связь данного типа возможна только на логическом уровне. На физическом уровне связь этого типа заменяется сущностью.

2.3.3.2 Установка связей между сущностями

Рассмотрим процесс установления связи между сущностями «Группа» и «Студент» и редактирования ее свойств.

1. Щелкните по одной из кнопок (например, «Non-identifying relationship») для установления связи на панели инструментов.

2. В окне диаграммы щелкните сначала по родительской сущности («Группа»), затем – по дочерней («Студент»).

3. Между сущностями «Группа» и «Студент» появляется штриховая линия с не закрашенным ромбиком (свойство «Nulls Allowed») со стороны родительской сущности и закрашенным кружочком (свойство «Zero, One or More») со стороны дочерней сущности.

Для редактирования свойств отношений используется редактор «Relationship 'R/№' Editor» (см. Рис. 2.7). Открываем этот диалог путем выделения редактируемой связи и выборе пункта «Properties» в КМ.

После установки связи определяются его свойства по умолчанию. В открывшемся диалоге мы можем отредактировать эти свойства в соответствии с требованиями технического задания. Рассмотрим часть из этих возможностей.

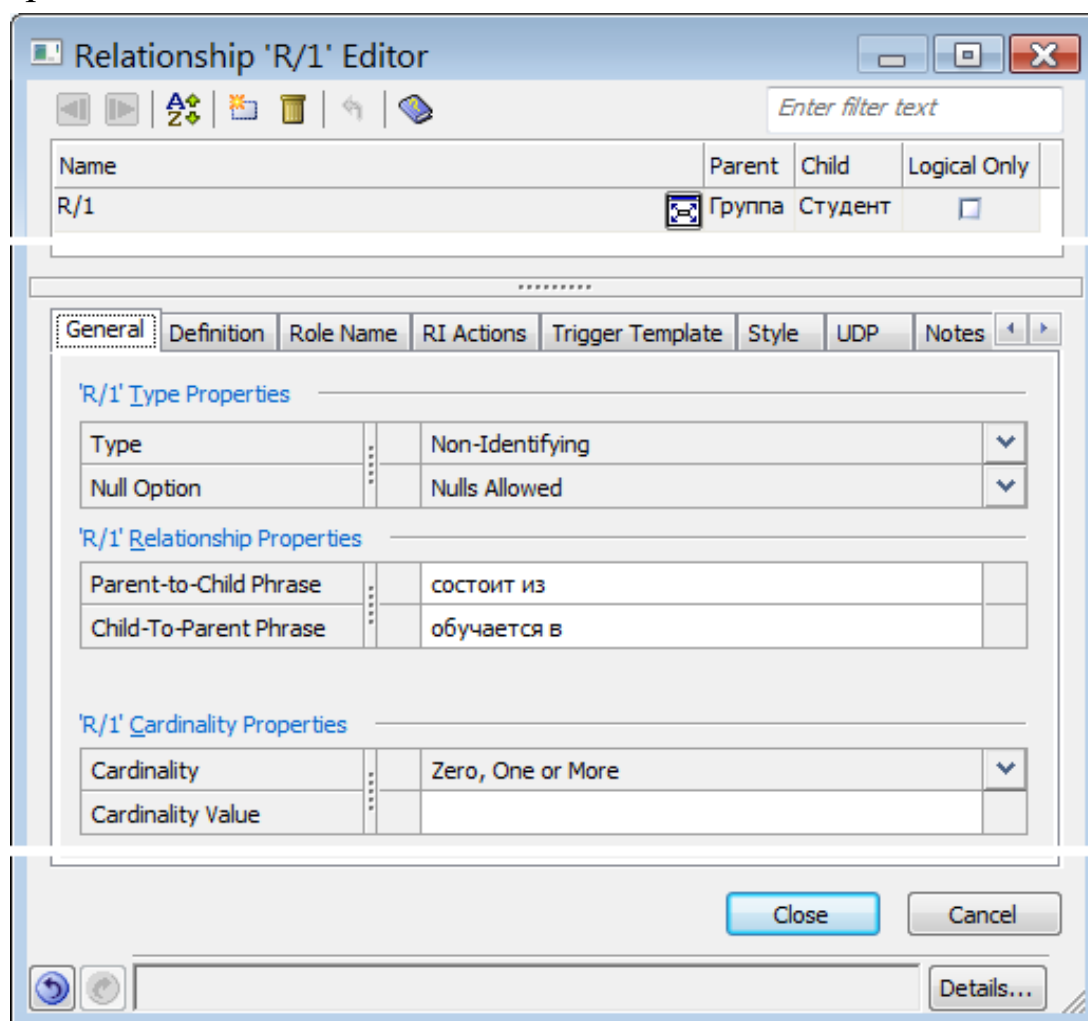


Рис. 2.7 – Редактор отношений «Relationship 'R/№' Editor»

2.3.3.3 Редактирование типа связи

Связь может быть идентифицирующей и неидентифицирующей. Кроме того, для неидентифицирующей связь может быть либо обязательной (**Nulls Not Allowed**), либо необязательной (**Nulls Allowed**).

В разделе '**R/1**' **Type Properties** вкладки **General** диалога **Relationship Editor** содержатся:

- свойство **Type** можно изменить на значения **Non-identifying** или **Identifying**;
- свойство **Null Option** можно изменить на значения **Nulls Allowed** или **Nulls Not Allowed**. Допустимость пустых значений (**Nulls Allowed**) в неидентифицирующих связях **ERwin** изображает пустым ромбиком на дуге связи со стороны родительской сущности.

2.3.3.4 Редактирование наименования связи

Связи на диаграмме лучше воспринимаются, если им присваиваются глагольные фразы. Это также можно сделать с помощью редактора связей.

В разделе '**R/1**' **Relationship Properties** вкладки **General** диалога **Relationship Editor**:

- свойство **Parent-To-Child** позволяет задать фразу для прямого направления связи;
- свойство **Child-To-Parent** позволяет задать фразу для обратного направления связи.

2.3.3.5 Редактирование мощности связи

Мощность связи – представляет собой отношение количества экземпляров родительской сущности к соответствующему количеству экземпляров дочерней сущности.

В разделе '**R/1**' **Cardinality Properties** вкладки **General** диалога **Relationship Editor** свойство **Cardinality** позволяет выбрать следующие степени связи:

- **Zero, One or More** – каждый экземпляр родительской сущности связан с нулем, одним или более экземпляров дочерней сущности;

- **One or More (P)** – каждый экземпляр родительской сущности связан с одним или более экземпляров дочерней сущности;
- **Zero or One (Z)** – каждый экземпляр родительской сущности не связан ни с одним экземпляром или одним экземпляром дочерней сущности.

Свойство **Cardinality** в этом же разделе позволяет задать количество экземпляров (**Exactly**) дочерней сущности, связанных с экземплярами родительской сущности. Рядом находится поле, где необходимо ввести это поле.

2.3.3.6 Редактирование свойств связей по обеспечению ссылочной целостности

Под ссылочной целостностью понимается обеспечение требования, чтобы значения внешнего ключа экземпляра дочерней сущности соответствовали значениям первичного ключа родительской сущности.

Установка ссылочной целостности – это логические конструкции, которые выражают ограничения использования данных. Они определяют, какие действия должна выполнять СУБД при удалении, вставке или изменении строки таблицы (экземпляра сущности), связанной с другой таблицей, если при этом произошло нарушение ссылочной целостности.

Заданные таким образом действия для каждой связи могут использоваться впоследствии при автоматической генерации триггеров, поддерживающих целостность данных.

В разделе **'R/1' RI Actions** вкладки **RI Actions** диалога **Relationship Editor** правила по обеспечению ссылочной целостности устанавливаются для следующих операций:

- **Child Delete Rule**;
- **Child Insert Rule**;
- **Child Update Rule**;
- **Parent Delete Rule**;
- **Parent Insert Rule**;
- **Parent Update Rule**.

Для каждой из перечисленных операций можно установить следующие значения, определяющие правила обеспечения ссылочной целостности:

- **None** – отсутствие проверки;
- **No Action** – отсутствие действий;
- **Restrict** – запрет операции;
- **Cascade** – каскадное выполнение операции;
- **Set null** – установка пустого значения;
- **Set default** – установка заданного значения по умолчанию.

Для корректного определения перечисленного набора параметров ссылочной целостности требуется знание характера связей между объектами ПрО.

Например, для определения значения параметра ссылочной целостности для операции **Parent Delete Rule** (правило выполнения операции **Delete** при возникновении ошибки удаления записи из родительской таблицы) связи **ГРУППА → СТУДЕНТ** необходимо четко представлять ситуацию, при которой требуется удалить экземпляр родительской сущности **ГРУППА**, а также знать последствия такого удаления в реальной жизни.

В качестве примера опишем анализ ситуации, приводящей к удалению записи родительской таблицы для связи **ГРУППА → СТУДЕНТ**. Удаление записи из таблицы **ГРУППА** может быть осуществлено при расформировании группы, например, из-за большого числа отчисленных студентов. В БД после этого оказывается, что оставшиеся студенты, обучаемые в расформированной группе, не относятся ни к одной из существующих групп.

Это происходит по той причине, что вторичные ключи соответствующих записей в таблице **СТУДЕНТ** ссылаются на несуществующую (уже удаленную) запись в таблице **ГРУППА**. Другими словами, происходит нарушение ссылочной целостности, и это нарушение выявляется СУБД автоматически благодаря наличию соответствующих триггерных функций. В данных методических указаниях подробное описание работы таких триггеров не приводится, так как данная тема изучается в рамках дисциплины «Базы данных».

Наша задача заключается в определении реакции СУБД, направленной на исправление данной ошибочной ситуации. Действия СУБД можно определить путем анализа возникшей ситуации, которая характеризуется тем, что студенты расформированной группы должны быть распределены по оставшимся группам потока.

После проведенного анализа легко определить значение параметра ссылочной целостности для операции **Parent Delete Rule**, определяющего характер реакции СУБД на описанное нарушение. В качестве значения параметра ссылочной целостности может быть определено значение **Set null** (или **Set default**). Установленное значение определяет следующую реакцию СУБД и действия пользователя на возникшую ошибку.

1. После выполнения операции удаления родительской записи в таблице **ГРУППА**, для дочерних записей в таблице **СТУДЕНТ** значения вторичных ключей устанавливаются псевдозначение **NULL** (или значение по умолчанию, определенное при создании таблицы).

2. Далее для вторичных ключей соответствующих записей таблицы **СТУДЕНТ** устанавливаются значения, равные значениям первичных ключей записей о тех группах, в которые распределяются не отчисленные студенты. Эта установка производится пользователем «вручную» по ходу распределения студентов по новым группам.

Рассмотрим еще одну операцию **Child Insert Rule** (правило выполнения операции **Insert** при возникновении ошибки ввода в дочернюю таблицу данной связи), которая может привести к нарушению ссылочной целостности. Смысл этой операции заключается в вводе новой записи в таблицу **СТУДЕНТ** при зачислении студента (например, студента, вернувшегося из академического отпуска) в одну из уже существующих групп.

В данной ситуации нарушение ссылочной целостности может произойти, если при вводе данных об этом студенте заносится значение вторичного ключа, не совпадающее ни с одним значением первичного ключа родительской таблицы **ГРУППА**.

Это приводит к ситуации, когда данный студент оказывается не зачисленным ни в одну из существующих групп, т.е. к нарушению ссылочной целостности.

Для отработки этой ситуации, как и в предыдущем примере, необходимо определить значение параметра сохранения ссылочной целостности. Таким значением может быть **Restrict**, запрещающее выполнение операции **Insert** в таблицу **СТУДЕНТ**. В этом случае выполнение операции вставки в таблицу запрещается с выдачей сообщения о характере ошибки. После этого пользователь должен исправить ошибку и повторить операцию ввода.

Таким же образом определите правила отработки нарушений ссылочной целостности для оставшихся четырех правил: **Child Delete Rule**, **Child Update Rule**, **Parent Insert Rule**, **Parent Update Rule**.

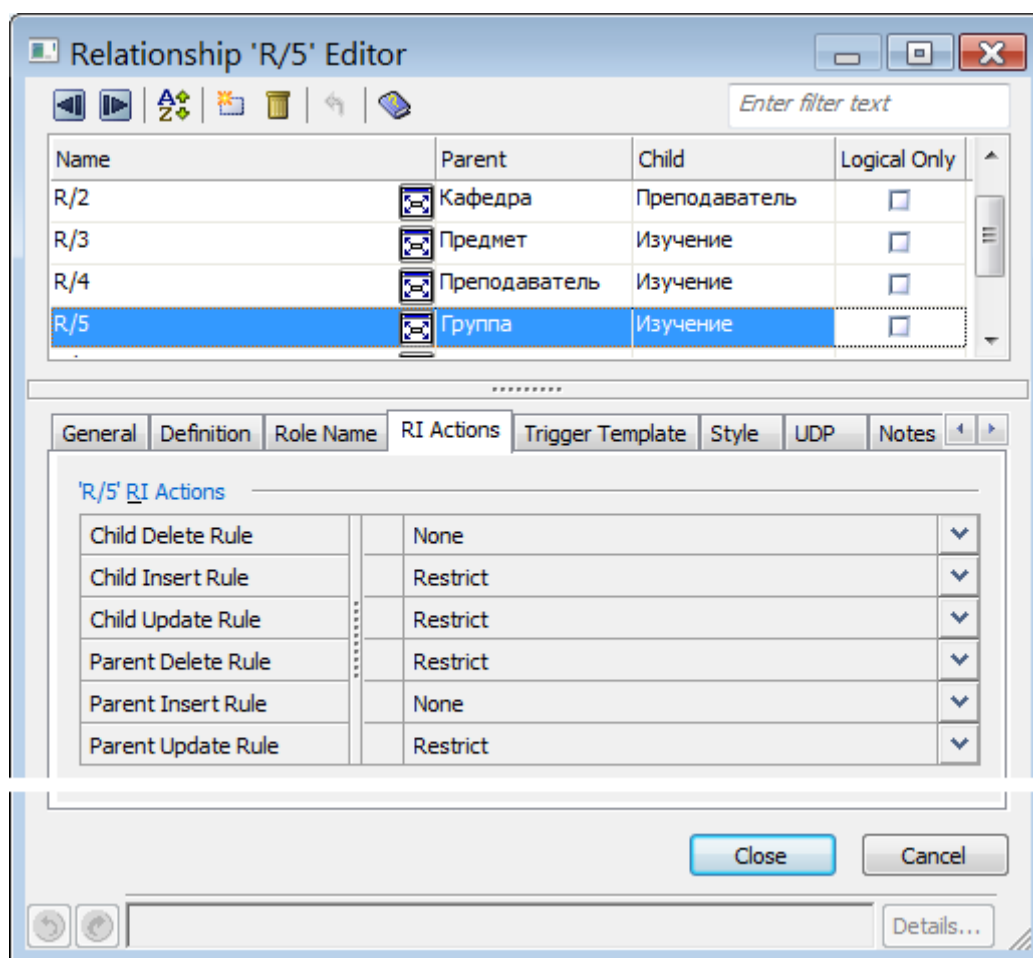


Рис. 2.8 – Окно Relationship 'R/5' Editor

Определение свойств связей по обеспечению ссылочной целостности осуществляется следующим образом.

1. Щелкните по линии связи на диаграмме правой кнопкой мыши и в контекстном меню выберите пункт **Properties**.

2. В окне **Relationship 'R/Номер связи' Editor** (см. Рис. 2.8) перейдите на вкладку **RI Actions** и установите значения для каждого правила ссылочной целостности.

2.3.4 ДИАГРАММА ЛОГИЧЕСКОЙ СХЕМЫ

После определения всех сущностей и установки всех связей между сущностями диаграмма схемы базы данных будет иметь вид, показанный на Рис. 2.9.

На диаграмме графическое изображение связи зависит от типа. Сплошными линиями показаны идентифицирующие связи, а штриховыми линиями – неидентифицирующие.

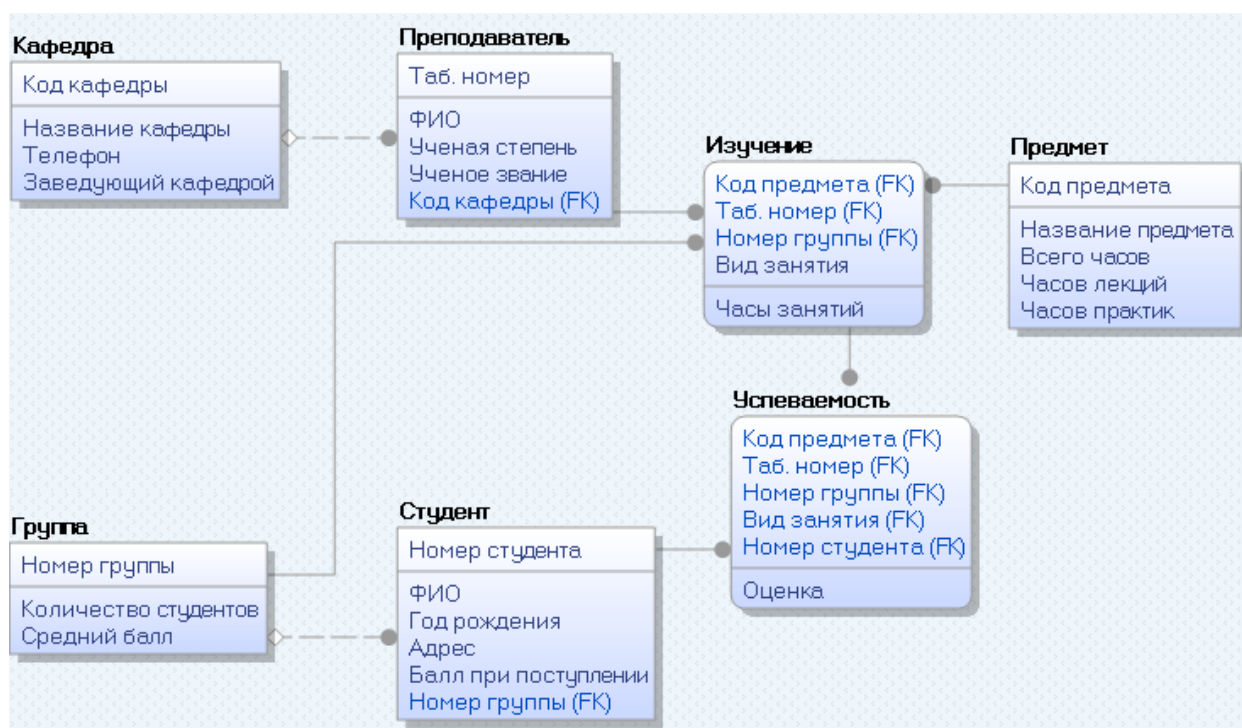


Рис. 2.9 – Сущности, их атрибуты и связи между ними

Типы сущностей также выделяются графическими средствами: зависимые сущности отличаются от независимых по закругленным углам.

По сравнению с концептуальной моделью, приведенной на Рис. 2.4, данная диаграмма является более информативной и функциональной. Сущности, их атрибуты и связи между сущностями, типы связей и сущностей, а также правила отработки нарушений ссылочной целостности, приведенные на диаграмме и сохраненные в памяти машины, могут быть использованы на последующих этапах построения модели ПрО для генерирования SQL-сценария создания схемы БД.

Примечание. С целью исключения загромождения диаграммы, правила отработки нарушений ссылочной целостности на данной диаграмме не приведены. Для отображения этих правил на диаграмме выполните следующие действия.

1. Нажмите на свободном месте диаграммы на правую кнопку мыши и в контекстном меню выберите пункт **Properties...**

2. В окне **ER Diagram 'Учебный процесс' Editor** перейдите на вкладку **Relationship** и в таблице **Relationship Logical Display Options** для свойства **Display Logical Referential Integrity** установите «галочку».

2.4 ЛАБОРАТОРНАЯ РАБОТА 4. РАЗРАБОТКА ФИЗИЧЕСКОЙ СХЕМЫ

Цель работы – разработка физической модели ПрО на основе логической, разработанной при выполнении лабораторной работы №3. Если логическая модель была составлена в терминах ПрО, то физическая модель должна быть определена с учетом требований используемой СУБД по именованию объектов БД и моделированию данных.

Перед выполнением данной лабораторной работы внимательно изучите раздел 5.3 работы [1].

2.4.1 ВЫБОР СЕРВЕРА

Конечный этап моделирования БД – реализация физической модели, которая также представлена в виде диаграммы. Физический уровень представления модели зависит от выбранного сервера, поэтому в первую очередь нужно определить сервер, для которого создается физическая модель.

Переключите диаграмму на физический уровень и выберите пункт меню **Actions\Target Database**, доступный только на физическом уровне. При этом откроется окно, показанное на Рис. 2.10.

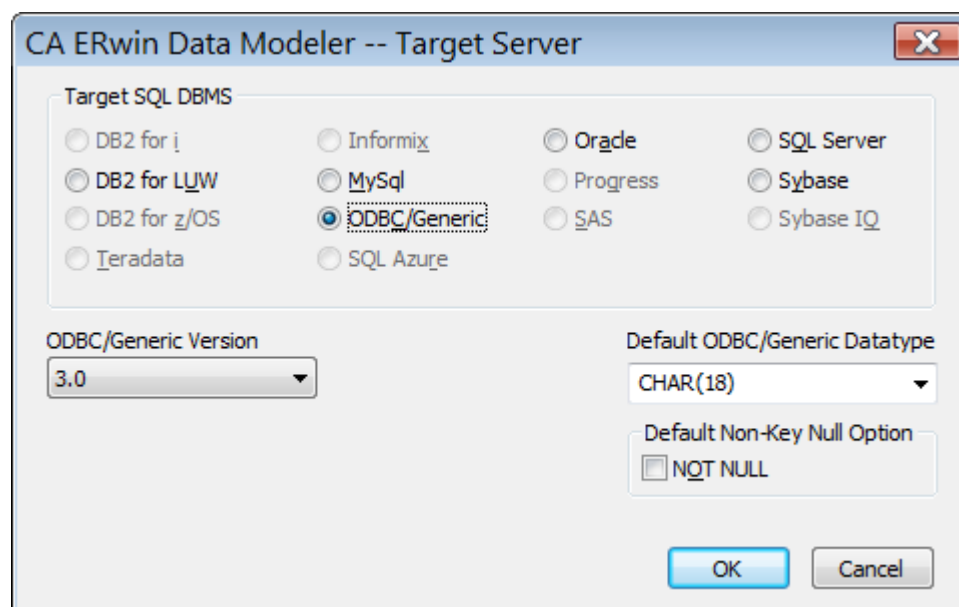


Рис. 2.10 – Диалоговое окно для выбора сервера

В категории **Target SQL DBMS** можно указать тип конечного сервера. Так как мы используем СУБД **PostgreSQL**, не представленную в этом списке, необходимо выбрать **ODBC**.

Open Database Connectivity – это программный интерфейс (API) доступа к БД, разработанный фирмой **Microsoft**, призванный упростить подключение к серверу БД из клиентского приложения, не беспокоясь о тонкостях взаимодействия с несколькими источниками. Также в данном окне можно выбрать версию конечного сервера, тип данных сервера по умолчанию и соответствие **NOT NULL** для **Default Non-Key Null Option**.

2.4.2 ТАБЛИЦЫ И КОЛОНКИ

В диалоге «**Target Server**» предусмотрено поле для шаблона имени таблицы, в котором по умолчанию стоит вызов макроопределения **%EntityName()**. Это макроопределение устанавливает в качестве имени таблицы имя сущности логической диаграммы, что не может нас устраивать, так как сервер **PostgreSQL** не допускает символов кириллицы в именах объектов метаданных.

Поэтому нам необходимо изменить имена таблиц, сгенерированные **ERwin** автоматически, на новые, не содержащие символов кириллицы. Пример переименования логических наименований для рассматриваемой ПрО приведен в нижеследующем списке:

Студент – **Student**;
Группа – **Group**;
Преподаватель – **Teacher**;
Кафедра – **Cathedral**;
Предмет – **Subject**;
Изучение – **Studying**;
Успеваемость – **Advancing**.

Для переименования таблицы необходимо вызвать пункт «**Table**» ее контекстного меню и или выбрать в главном меню пункт «**Model\Tables...**». При этом появится диалог «**ODBC Table 'Имя таблицы' Editor**», который после переименования всех таблиц схемы базы данных выглядит, как показано на Рис. 2.11.

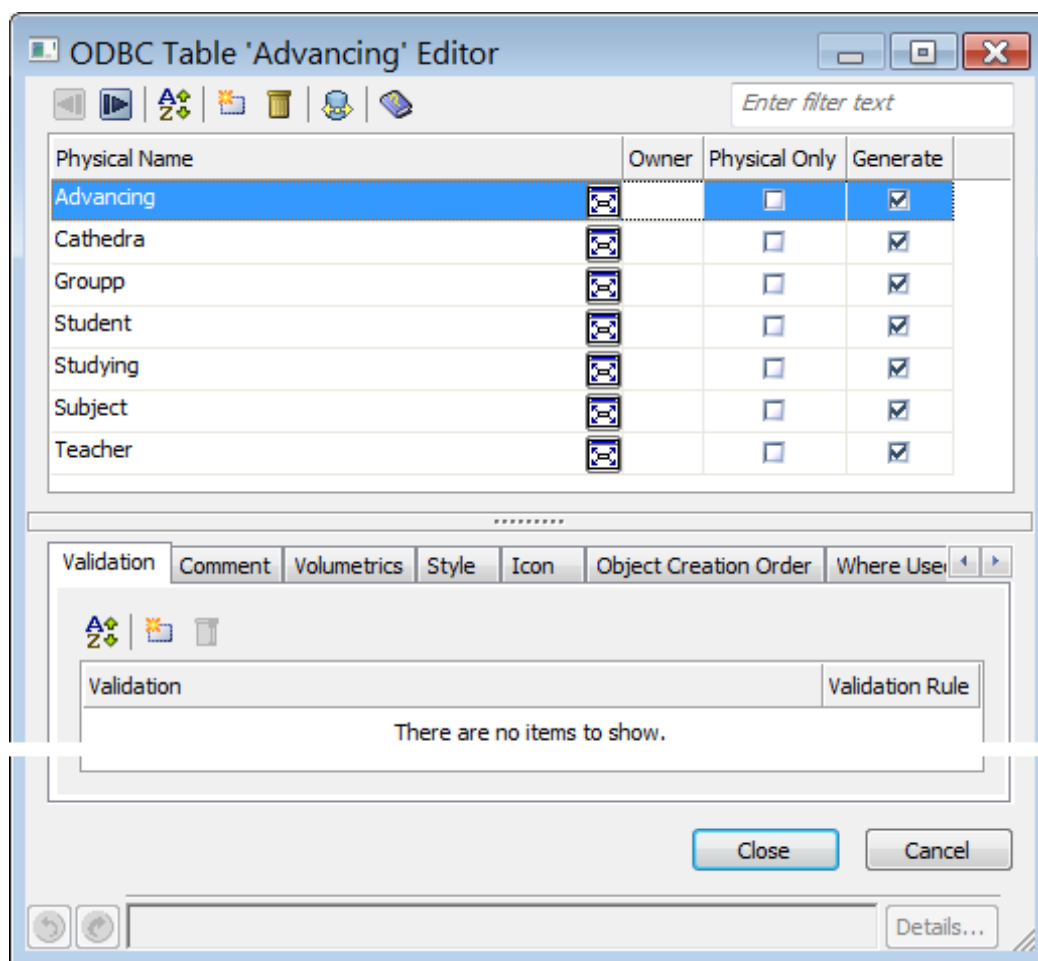


Рис. 2.11 – Редактор таблиц

Диалог «ODBC Table 'Имя таблицы' Editor» имеет следующую конструкцию. В верхней части диалога содержится таблица с полями:

«Physical Name», в котором выбирается необходимая таблица для редактирования имени;

«Owner», в котором можно указать владельца таблицы, если он отличается от пользователя, генерирующего схему БД;

«Physical Only», в котором можно установить флажок, указывающий на то, что таблица существует только на физическом уровне;

«Generate», в котором можно установить флажок, указывающий на то, что при генерации схемы БД будет выполняться запрос CREATE TABLE.

Кроме того, редактор таблиц содержит страницы с закладками. Основными из этих закладок являются следующие:

Validation – служит для задания правил и корректности ввода данных в таблицу;

Comment – страница для ввода комментариев к таблице;

UDP – обеспечивает ввод значений пользовательских свойств, связанных с таблицей.

2.4.3 РЕДАКТИРОВАНИЕ СВОЙСТВ ПОЛЕЙ

После приведения имен таблиц в соответствие с требованиями, займемся именами и типами данных полей. Сервер **PostgreSQL** поддерживает типы данных, приведенные в Табл. 2.1.

Таблица 2.1. Типы данных, поддерживаемые сервером **PostgreSQL**

Тип данных	Размер	Описание	Диапазон
smallint	2 байта	целые числа	от -32768 до +32767
integer	4 байта	обычные целые числа	от -2147483648 до +2147483647
bigint	8 байт	целые числа большого диапазона	от -9223372036854775808 до 9223372036854775807
decimal	пере- менный	числа с точностью, указываемой пользователем, точное число	до 131072 разрядов перед десятичной точкой; до 16383 разрядов после десятичной точки
numeric	пере- менный	числа с точностью, указываемой пользователем, точное число	до 131072 разрядов перед десятичной точкой; до 16383 разрядов после десятичной точки
real	4 байта	переменная точность, неточное число	точность 6 десятичных разрядов
double precision	8 байт	переменная точность, неточное число	точность 15 десятичных разрядов
serial	4 байта	целое число с автоувеличением	от 1 до 2147483647
bigserial	8 байт	большое целое число с автоувеличением	от 1 до 9223372036854775807
date	4 байта	дата (без часового пояса)	от 4713 BC до 5874897 AD с шагом в 1 день
Boolean	1 байт	в состоянии истина или ложь	

Тип данных	Размер	Описание	Диапазон
character varying(n), varchar(n)		символьный тип переменной длины с ограничением	
char(n), character(n)		символьный тип заданной длины с заполнением пробелами	

Для редактирования свойств вызовите редактор колонок ([Column Editor](#)). Для этого выберите пункт [Columns Properties](#) контекстного меню таблицы.

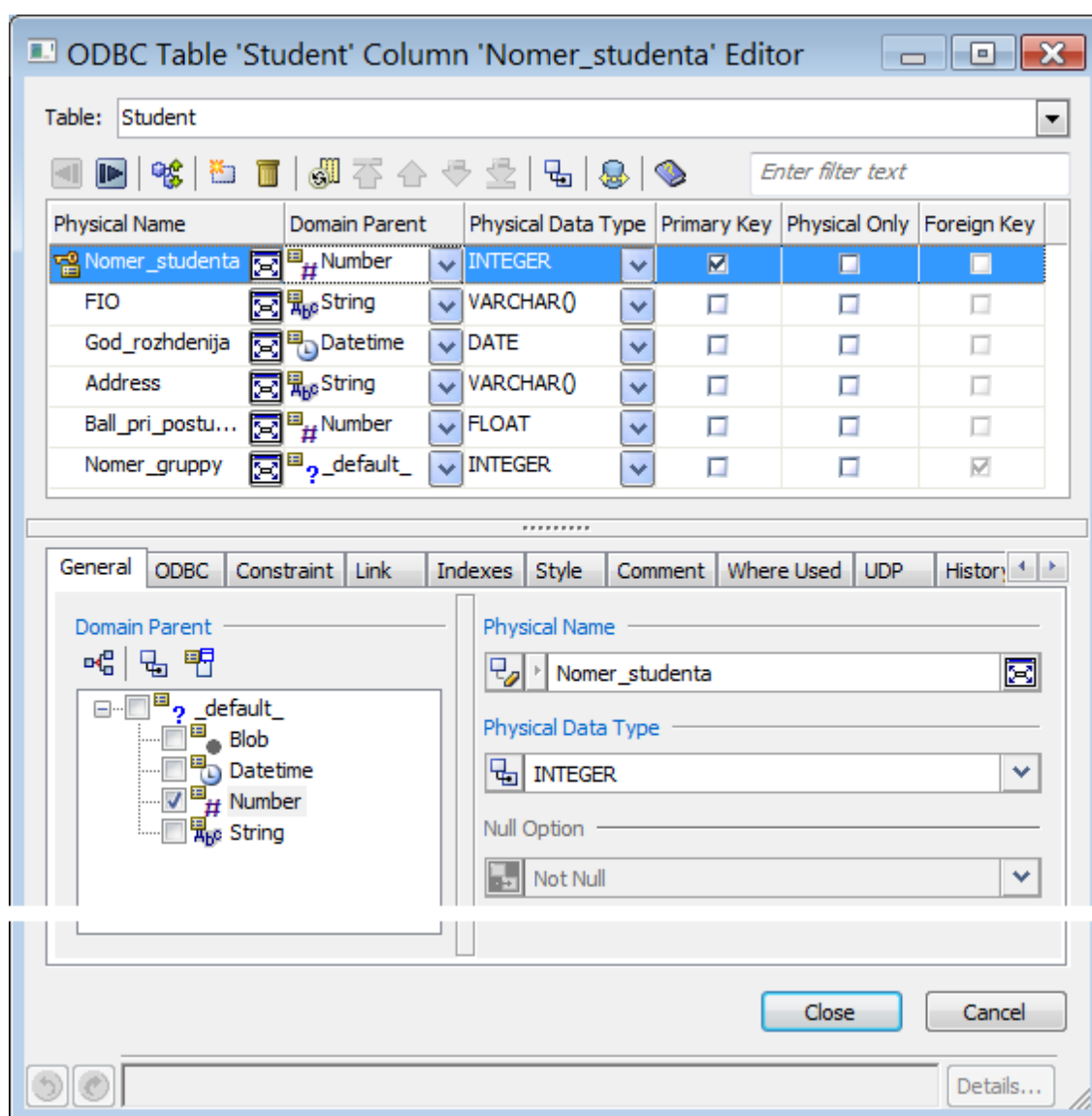


Рис. 2.12 – Редактор колонок

Редактор колонок (см. Рис. 2.12) внешне напоминает редактор атрибутов, рассмотренный ранее. Редактируемая таблица выбирается в списке «Table», находящемся в верхней части диалога.

Переключение редактируемых таблиц осуществляется с помощью выпадающего списка в верхней части окна. С помощью таблицы в верхней части диалогового окна можно выбрать редактируемое поле, изменить его имя, тип, сделать его первичным ключом. В нижней части окна расположены следующие вкладки (ниже перечислены только основные):

General – позволяет выбрать имя и тип данных на физическом уровне модели;

ODBC – вкладка, зависящая от целевой СУБД. Позволяет выбрать тип данных и **Null Option** на уровне сервера;

Indexes – позволяет редактировать и добавлять новые индексы в таблицу. В нижней части имеется флаг **Show FK Indexes**, при включе-

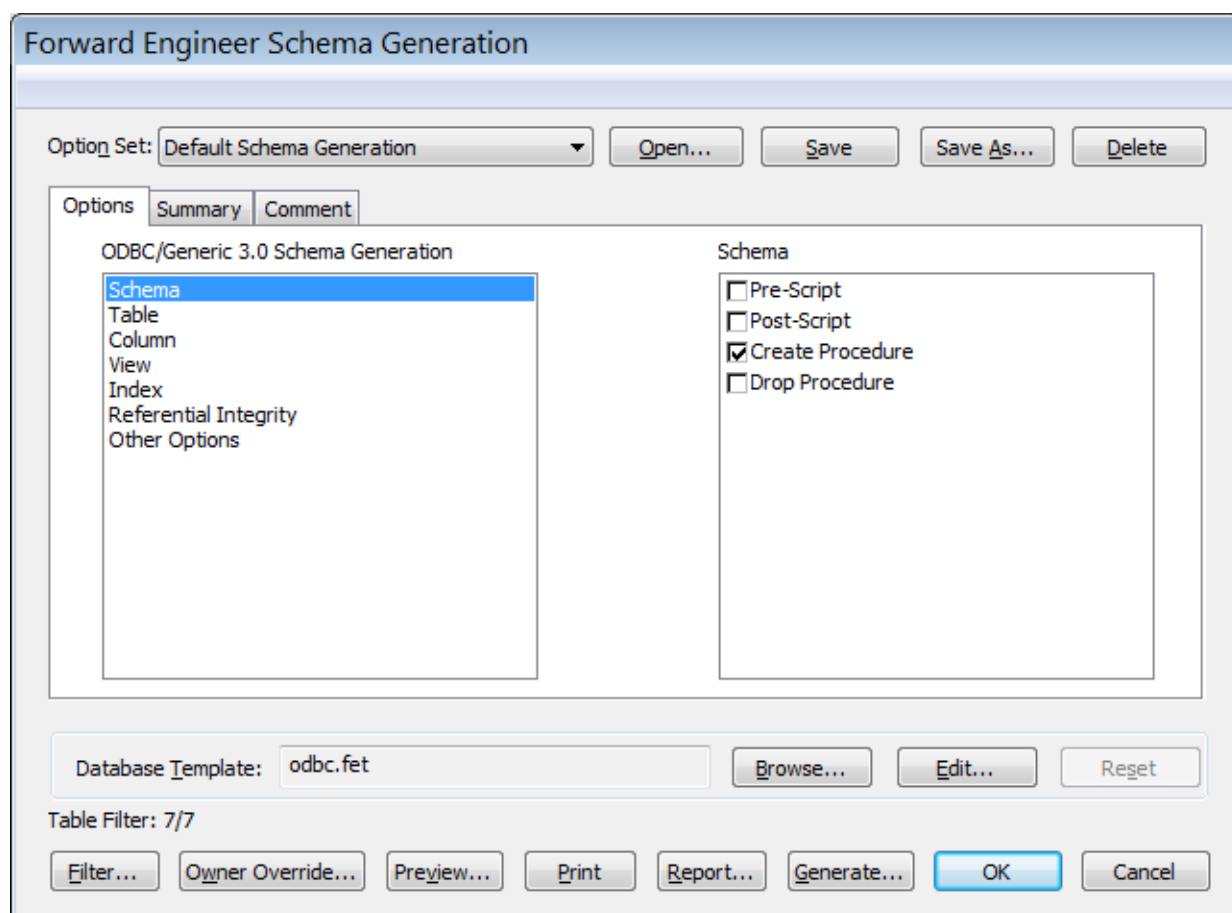


Рис. 2.13 – Окно «Forward Engineer Schema Generation»

нии которого показываются как первичный, так и внешние индексы. Те индексы, в которых участвует выбранная колонка, помечены флажком.

Остальные вкладки аналогичны таким в ранее рассмотренных редакторах доменов и таблиц. Вторая кнопка под таблицей данного окна позволяет вызвать редактор индексов ([Indexes Editor](#)).

2.4.4 ПРОВЕРКА РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ СКРИПТОВ

Теперь можно проверить результаты проведенной работы. Для этого выполните следующие шаги:

1. Выберите пункт меню «[Actions\Forward Engineer\Schema...](#)» или нажмите на кнопку [Forward Engineer Schema Generation](#) на панели инструментов.

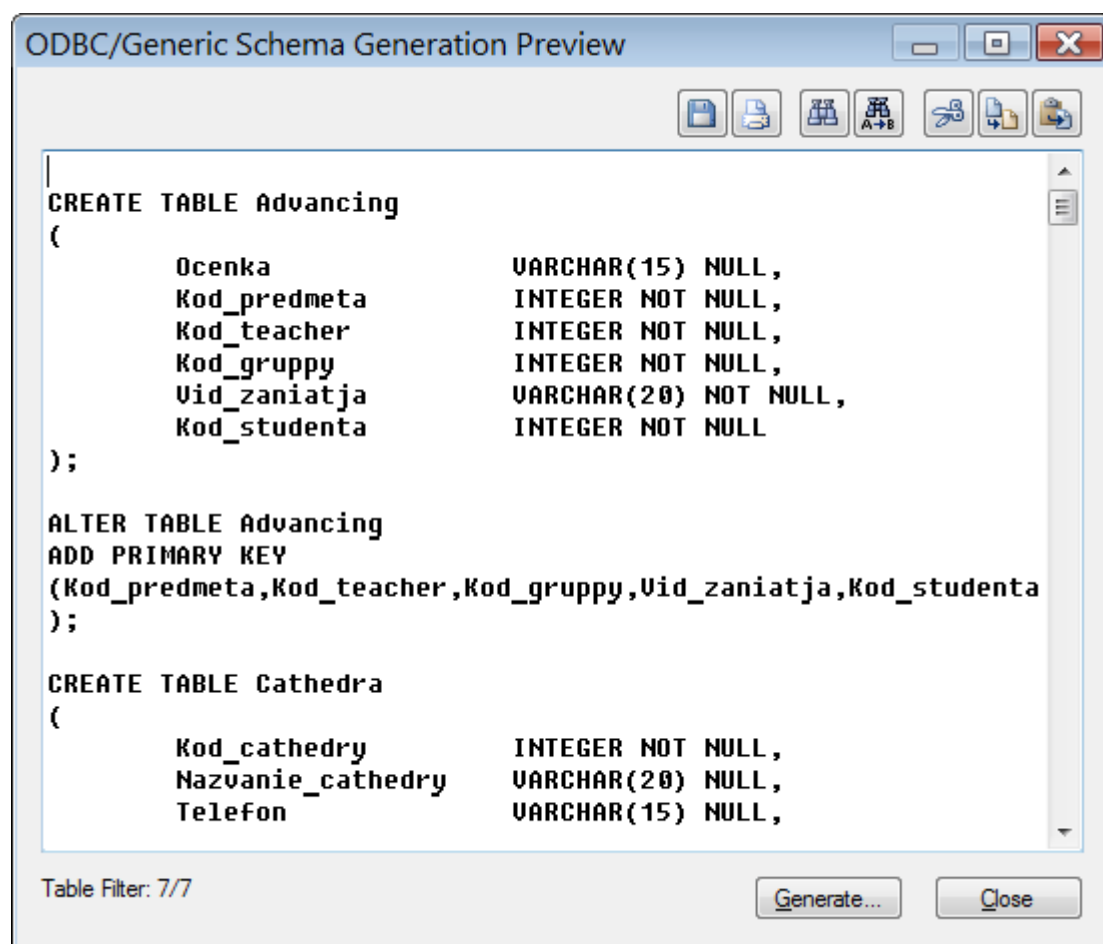


Рис. 2.14 – Окно «Forward Engineer Schema Generation»

2. В окне **Forward Engineer Schema Generation** (см. Рис. 2.13) нажмите на кнопку «**Preview ...**».

3. В окне **ODBC/Generic Schema Generation Preview** (см. Рис. 2.14) внимательно изучите сгенерированный код **SQL**-сценария создания схемы БД.

Убедитесь, что в сгенерированном коде содержатся скрипты для создания всех таблиц схемы БД, а также их первичных и вторичных ключей.

2.5 ЛАБОРАТОРНАЯ РАБОТА 5. РАЗРАБОТКА ГЕНЕРАТОРОВ И ТРИГГЕРОВ ДЛЯ КЛЮЧЕВЫХ АТТРИБУТОВ ТАБЛИЦ БД

Цель работы – Разработка с использованием языка макросов **ERwin** последовательностей, генераторов и триггеров для автоматического генерирования уникальных целых числовых значений и вставки их в поля ключевых атрибутов таблиц БД.

Перед выполнением данной лабораторной работы внимательно изучите раздел 5.5.1 работы [1].

2.5.1 КРАТКИЕ СВЕДЕНИЯ О ЯЗЫКЕ МАКРОСОВ ERWIN И ПОСЛЕДОВАТЕЛЬНОСТЯХ, ГЕНЕРАТОРАХ И ТРИГГЕРАХ

Встроенный в **ERwin** макроязык содержит около 200 макросов, имена которых начинаются с символа «%». С описанием этих макросов можно ознакомиться на сайте разработчика **ERwin**. Здесь будут приведены описания только тех макросов, которые будут использованы для выполнения данной и следующей лабораторных работ.

В текстах скриптов, создаваемых в рамках выполнения лабораторных работ, используются следующие макросы:

- **%==(value1, value2)** – оператор сравнения. Выдает значение 1 при совпадении значений **value1** и **value2**, и 0 – при их несовпадении;
- **%AttProp(nameUDP)** – извлекает значение свойства с именем «**nameUDP**», подключенного к атрибуту;
- **%TableName** – при раскрытии шаблона разворачивается в имя текущей таблицы;
- **%ForEachAtt (%TableName)** – макрос-оператор цикла. Осуществляет перебор всех атрибутов таблицы **%TableName**. Например, конструкция **%ForEachAtt (%TableName) {...}** выполняет все, что находится внутри скобок {...}, для каждого атрибута таблицы **%TableName**;
- **%if (проверка условия)** – условный макрос является аналогом условного оператора **if** языков программирования. Например, конструкция **%if (%==(%AttProp(generate_id), yes)) {...}** проверяет значение **generate_id** атрибута (**generate_id** – это свойство атрибута, устанавливаемое

пользователем, описанное ниже), и если оно равно «yes», то выполняется все, что находится в фигурных скобках.

Приведем также краткие сведения о последовательностях, генераторах и триггерах, изученных Вами в рамках дисциплины «Базы данных».

СУБД PostgreSQL поддерживает так называемые последовательности (**sequence**), представляющие собой автоматически увеличивающееся число и, как правило, использующиеся для присваивания уникальных значений идентификаторов или первичных ключей в таблицах. Последовательность определяется текущим числовым значением и набором характеристик, определяющих алгоритм автоматического увеличения (или уменьшения) используемых данных.

В нашей модели имеются поля, введенные в качестве первичного ключа. Эти поля должны содержать генерируемые автоматически уникальные для таблицы целые числовые значения. Для реализации этой задачи необходимы два компонента:

- генератор уникального значения, в качестве которого создадим последовательность;
- триггер, присваивающий это значение при вставке новой записи.

Для создания генератора воспользуемся уже упомянутым запросом **CREATE SEQUENCE**, а для присвоения уникального значения первичному ключу – запросом **CREATE TRIGGER**.

В учебных целях создадим аналог последовательности, включающий в себя генератор уникального значения и триггер, который присваивает это значение при вставке новой записи.

Примем правило, что генераторы будут называться по имени поля с прибавлением префикса **_gen**.

Такое же правило примем относительно имен генераторов, которые образуются от имени таблиц с добавлением **_GEN_ID.**, и имен триггеров, образуемых от имени таблиц с добавлением **_TRIG_ID**.

С учетом изложенного запросы на создание последовательности, генератора и триггера должны иметь вид:

```

CREATE SEQUENCE Kod_gruppy_gen;

CREATE OR REPLACE FUNCTION Groupp_GEN_ID()
RETURNS TRIGGER AS
$Groupp_GEN_ID$
BEGIN
    NEW.Kod_gruppy =
        nextval('Kod_gruppy_gen'::regclass);
    return NEW;
END;
$Groupp_GEN_ID$
LANGUAGE plpgsql;

CREATE TRIGGER Groupp_TRIG_ID
BEFORE INSERT ON Groupp
FOR EACH ROW EXECUTE PROCEDURE Groupp_GEN_ID();

```

Такие запросы должны быть в сценарии создания БД после SQL-запросов создания таблиц **CREATE TABLE** для каждой таблицы, имеющей автоинкрементный ключ.

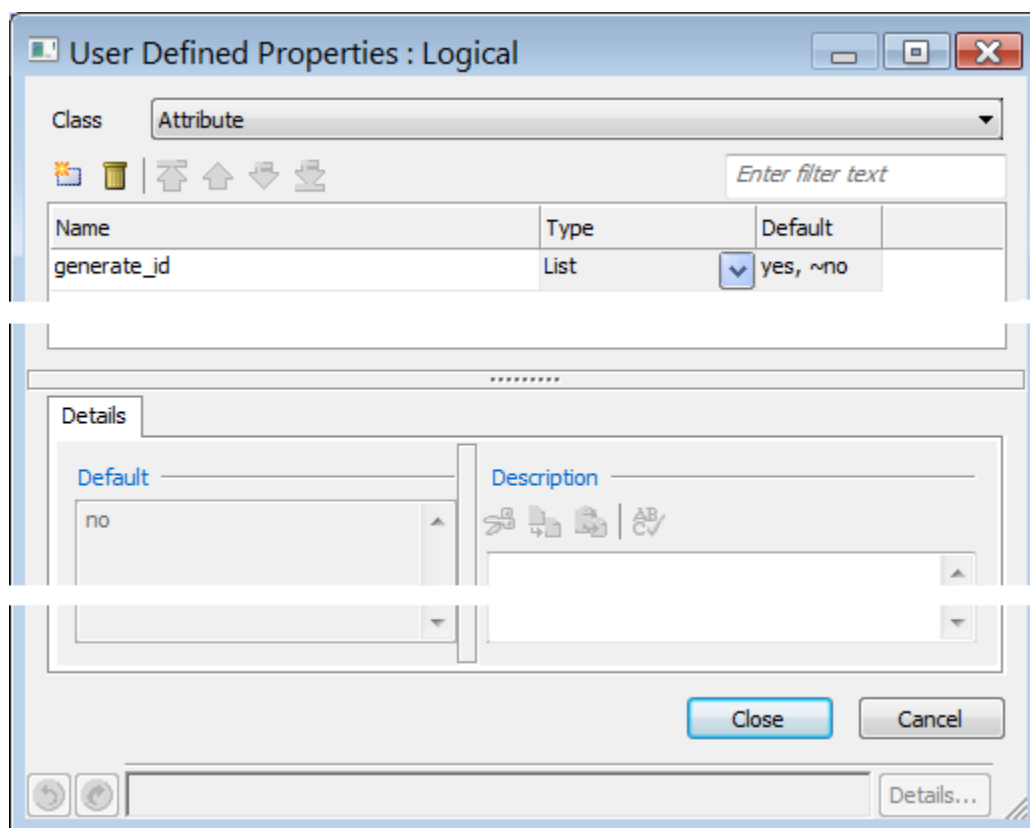


Рис. 2.15 – Установка свойства generate_id – признака автоинкрементного поля

2.5.2 СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО СВОЙСТВА

А теперь для всех атрибутов создадим специальное пользовательское свойство (**User Definition Property – UDP**), которое будет служить признаком генерирования значений. Для этого перейдите в режим логической схемы и выберите пункт меню **Model\User Defined Properties**. В диалоге свойств установите в списке **Class** объект **Attribute**, создайте свойство с именем **generate_id** типа **List** и назначьте ему допустимые значения по **yes, ~no** (см. Рис. 2.15). Здесь знак ‘~’ определяет значение по умолчанию.

Это пользовательское свойство будет видно у всех атрибутов модели, а значение его по умолчанию будет равно **no**. Для генерируемых ключевых атрибутов поменяйте это значение на **yes**. С этой целью вызовите диалог **Attribute Editor** и на вкладке **UDP** поменяйте значение **no** свойства **generate_id** на **yes** (см. Рис. 2.16).

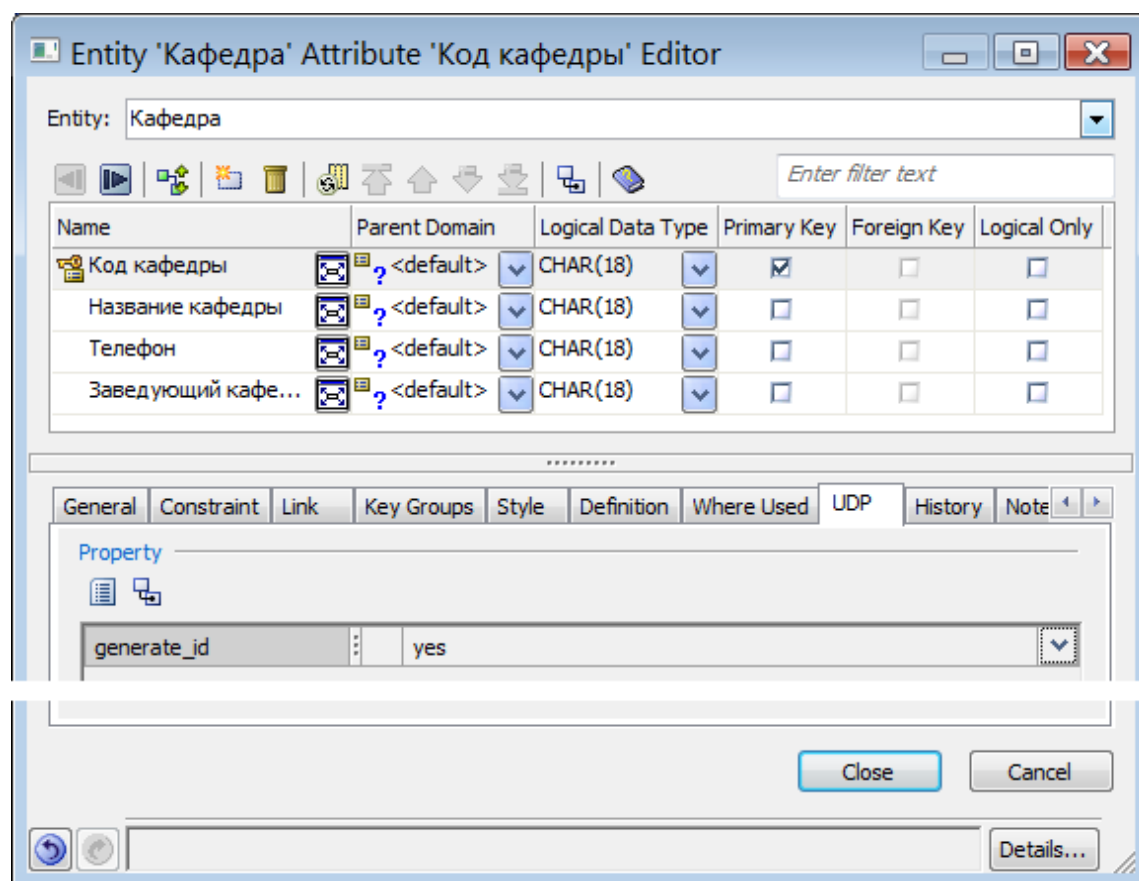


Рис. 2.16 – Установка значения свойства *generate_id*

Перечень ключевых автоинкрементных атрибутов приведен в табл. 2.2.

Табл. 2.2. Ключевые автоинкрементные атрибуты

Сущность	Ключевой атрибут
Студент	Код студента
Группа	Код группы
Кафедра	Код кафедры
Преподаватель	Код преподавателя
Предмет	Код предмета
Изучение	-
Успеваемость	-

2.5.3 СОЗДАНИЕ ШАБЛОНА СКРИПТА ПОСЛЕДОВАТЕЛЬНОСТИ, ГЕНЕРАТОРА И ТРИГГЕРА

Итак, мы поместили автоинкрементные атрибуты, теперь необходимо создать шаблон скрипта для создания генератора и подключить его к указанным таблицам.

Для этого перейдите в режим физической модели, выберите **Model\Tables** (см. Рис. 2.17) и на панели инструментов закладки **Object Creation Order** нажмите кнопку **Script Template Editor**.

Появляется окно «**ODBC Script Template Editor**» (см. Рис. 2.18). Необходимо в этом редакторе создать скрипт и подключить к таблице. На этот раз скрипт должен иметь тип «**после генерации**», так как он будет содержать запрос на создание триггера, а таблица, к которой этот триггер относится, должна уже существовать.

Щелкните по кнопке «**Script Template**» и перейдите в редактор шаблонов, уже рассмотренный выше. Создайте новый скрипт с названием «**Создание генератора**». Для этого в окне «**Table Script Template**» наберите текст:

```
%ForEachAtt (%TableName)
{%if (%==( %AttProp (generate_id) , yes) )
{ CREATE SEQUENCE %AttFieldName_gen;
  CREATE OR REPLACE FUNCTION %TableName_GEN_ID ()
```

```

RETURNS TRIGGER AS
$%TableName_GEN_ID$
BEGIN
    NEW.%AttFieldName =
        nextval('%AttFieldName_gen'::regclass);
    return NEW;
END;
$%TableName_GEN_ID$
LANGUAGE plpgsql;
CREATE TRIGGER %TableName_TRIG_ID
BEFORE INSERT ON %TableName FOR EACH ROW
EXECUTE PROCEDURE %TableName_GEN_ID();
}}

```

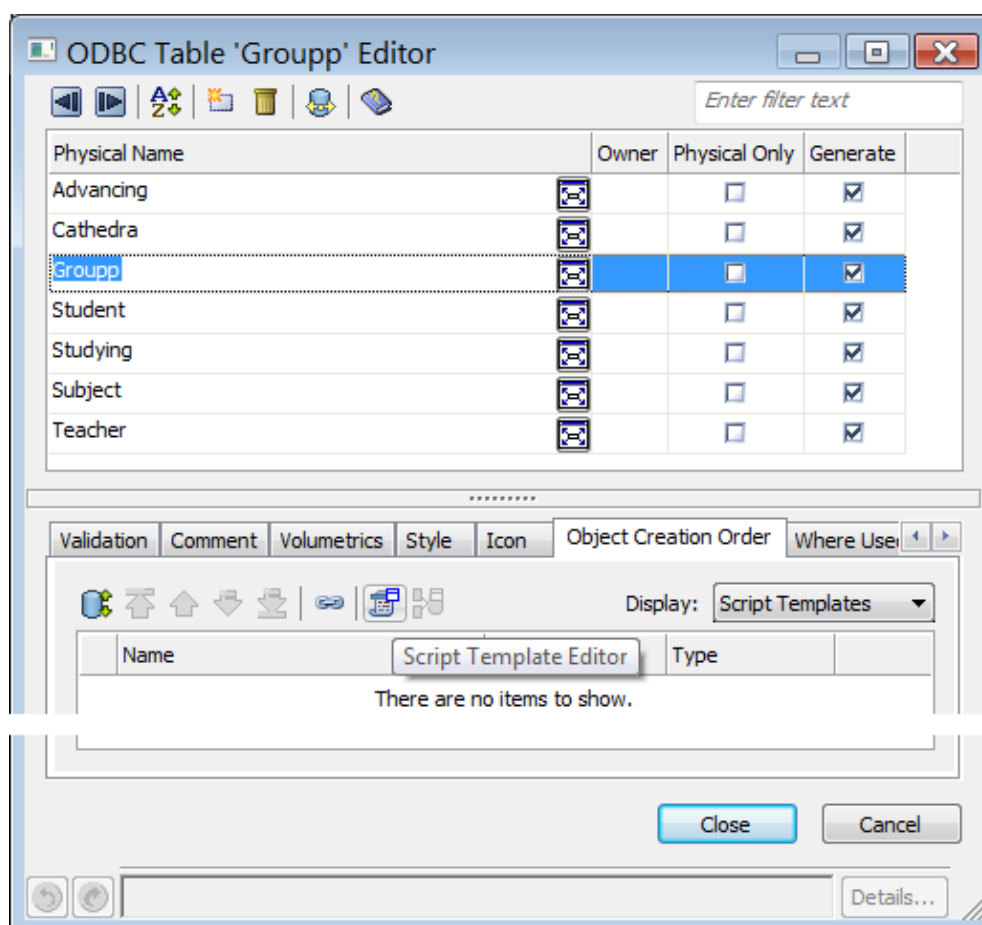


Рис. 2.17 – Создание шаблона скрипта генератора

Необходимо отметить, что фрагмент кода между макросами \$%TableName_GEN_ID\$, отмеченными спецсимволом \$, должен быть

записан в одну строку во избежание ошибки «строка с незавершённым спецсимволом \$» при генерации как показано ниже

```

    ${TableName}_GEN_ID$ BEGIN NEW.%AttFieldName = next-
val('%AttFieldName_gen'::regclass); return NEW; END;
${TableName}_GEN_ID$.

```

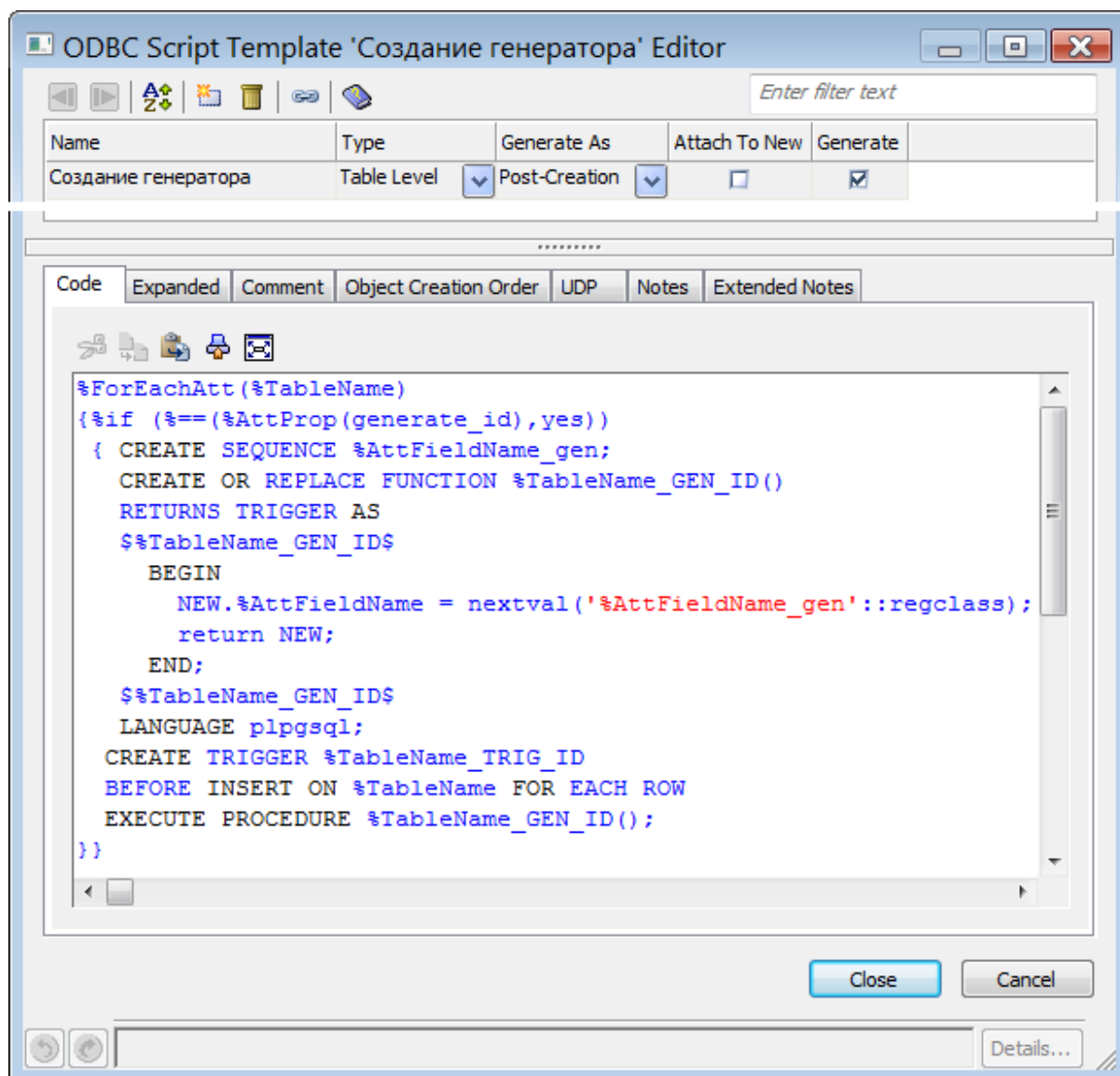


Рис. 2.18 – Создание скрипта триггера

Этого правила следует придерживаться и при разработке кода шаблонов хранимых процедур при выполнении следующей лабораторной работы.

Обратите внимание на то, что в выше приведенном тексте, а также на Рис. 2.18, данный фрагмент кода представлен в разных

строках. Но это сделано лишь для улучшения восприятия текста; при выполнении лабораторных работ следует придерживаться описанного выше правила.

Проверьте, чтобы **Generate As** был установлен в положении **Post-Creation**, а **Type** – как **Table Level** и нажмите **Close**. В редакторе таблицы нажмите кнопку **Script Template Browser**, и прикрепите скрипт к таблицам, перенеся их из левого (**Unattached Object**) списка в правый (**Attached Object**) список. После этого можно закрыть диалог.

2.5.4 ПРОВЕРКА РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ СКРИПТОВ

По схеме, описанной в подразделе 2.4.4 описания выполнения предыдущей лабораторной работы, проверьте результаты проделанной работы.

Убедитесь, что в дополнение к скриптам SQL-запросов создания таблиц **CREATE TABLE** и их первичных и вторичных ключей, появились строки создания генераторов и триггеров. Например, часть сценария, относящаяся к таблице **Groupp**, выглядит теперь следующим образом:

```
CREATE TABLE Groupp
(
    Kod_gruppy          INTEGER NOT NULL,
    Kol_studentov       INTEGER NULL,
    Srednij_ball        FLOAT NULL
);

ALTER TABLE Groupp
ADD PRIMARY KEY (Kod_gruppy);

CREATE SEQUENCE Kod_gruppy_gen;

CREATE OR REPLACE FUNCTION Groupp_GEN_ID()
RETURNS TRIGGER AS
$Groupp_GEN_ID$
BEGIN
    NEW.Kod_gruppy=nextval('Kod_gruppy_gen'::regclass);
    return NEW;
END;
$Groupp_GEN_ID$
LANGUAGE plpgsql;
```

```
CREATE TRIGGER Groupp_TRIG_ID  
  BEFORE INSERT ON Groupp  
  FOR EACH ROW EXECUTE PROCEDURE Groupp_GEN_ID();
```

2.6 ЛАБОРАТОРНАЯ РАБОТА 6. РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР ДЛЯ ВСТАВКИ, ОБНОВЛЕНИЯ И УДАЛЕНИЯ ЗАПИСЕЙ ТАБЛИЦ БД

Цель работы – Разработка с использованием языка макросов **ERwin** последовательностей, генераторов и триггеров для автоматического генерирования уникальных целых числовых значений и вставки их в поля ключевых атрибутов таблиц БД.

Перед выполнением данной лабораторной работы внимательно изучите раздел 5.5.2 работы [1].

2.6.1 КРАТКИЕ СВЕДЕНИЯ О ХРАНИМЫХ ПРОЦЕДУРАХ

Рассмотренный в предыдущей работе язык макросов очень удобен, когда при генерации схемы требуется повторить одну и ту же операцию для нескольких объектов модели.

Воспользуемся этим языком и для создания хранимых процедуры вставки, изменения и удаления записей, имеющих одинаковую структуру для всех таблиц. Поэтому для хранимых процедур можно создать шаблоны для вставки, изменения и удаления записей, а затем использовать их для всех таблиц базы.

Напомним, что запрос на создание процедуры вставки записи имеет вид

```
CREATE FUNCTION <имя процедуры> (<параметр тип>, ...) AS  
BEGIN  
    INSERT INTO <имя таблицы> (<имя поля>, <имя поля>...)   
    VALUES (<: параметр>, <: параметр> ...);  
END;
```

Примем правило, согласно которому имя процедуры образуется от имени таблицы, путем добавления перед ним префикса «ins_». Параметрами процедуры будут все поля таблицы, кроме автоинкрементного поля, значение которого генерируется триггером.

Для таблицы **Groupp** процедура вставки создается запросом

```
CREATE FUNCTION ins_groupp (Kol_studentov integer,  
    Srednij_ball FLOAT)  
RETURNS void AS  
BEGIN
```

```

INSERT INTO Groupp(Kol_studentov, Srednij_ball)
VALUES (Kol_studentov, Srednij_ball);
END;
LANGUAGE plpgsql;

```

Процедура изменения записи таблицы имеет вид

```

CREATE FUNCTION <имя процедуры> (<параметр тип>, ...) AS
BEGIN
    UPDATE <имя таблицы>
    SET <имя поля> = <:параметр>,
        <имя поля> = <:параметр>,
        ...
        <имя поля> = <:параметр>,
    WHERE <ключ> = <параметр> AND
        <ключ> = <параметр> AND
        ...
        <ключ> = <параметр>;
END;

```

Как и в предыдущем случае, имя этой процедуры создадим из имени таблицы, добавив к нему префикс «upd_». В отличие от предыдущей процедуры, параметры здесь должны содержать и ключевое поле, по которому ставится условие, в запросе UPDATE. Применительно к таблице Groupp процедура выглядит так:

```

CREATE FUNCTION upd_groupp(Kod_gruppy int4,
    Kol_studentov int4, Srednij_ball FLOAT)
RETURNS void AS
$BODY$
BEGIN
    UPDATE Groupp
    SET Kol_studentov=Kol_studentov,
        Srednij_ball=Srednij_ball
    WHERE Kod_gruppy=Kod_gruppy;
END;
$BODY$
LANGUAGE plpgsql;

```

Процедура удаления содержит в качестве параметров только ключевые реквизиты и имеет вид

```

CREATE FUNCTION del_groupp(Kod_gruppy int4)
RETURNS void AS

```

```

$BODY$
BEGIN
    DELETE FROM Groupp
    WHERE Kod_gruppy=Kod_gruppy;
END;
$BODY$
LANGUAGE plpgsql;

```

2.6.2 СОЗДАНИЕ ШАБЛОНА СКРИПТА ФУНКЦИЙ ХРАНИМЫХ ПРОЦЕДУР

Для создания шаблонов и привязки их к таблицам выберете пункт главного меню **Model\Stored Procedures**.

В вызванном редакторе создайте три новых шаблона (тип – **Table Level**) с соответствующими именами и добавьте в них код, указанный ниже (Рис. 2.19).

Вставка записи

```

CREATE FUNCTION ins_ %Lower(%TableName)
(%ForEachAtt(%TableName, ','))
{ %if (%Not(%AttIsPK))
    {%AttFieldName %AttDataType } } )
RETURNS void AS
$BODY$
BEGIN
    INSERT INTO %TableName
    (%ForEachAtt(%TableName, ','))
    {%if (%Not(%AttIsPK)) {%AttFieldName} } )
VALUES (%ForEachAtt(%TableName, ','))
    {%if (%Not(%AttIsPK)) {%AttFieldName} } );
END;
$BODY$
LANGUAGE plpgsql

```

Изменение записи

```

CREATE FUNCTION upd_ %Lower(%TableName)
(%ForEachAtt(%TableName, ','))
    {%AttFieldName %AttDataType})
RETURNS void AS
$BODY$

```

```

BEGIN
  UPDATE %TableName
  SET %ForEachAtt(%TableName, ',')
    { %if (%Not(%AttIsPK)) { %AttFieldName=%AttFieldName } }
  WHERE %ForEachAtt(%TableName, 'AND')
    { %if (%AttIsPK) { %AttFieldName=%AttFieldName } };
END;
$BODY$
LANGUAGE plpgsql

```

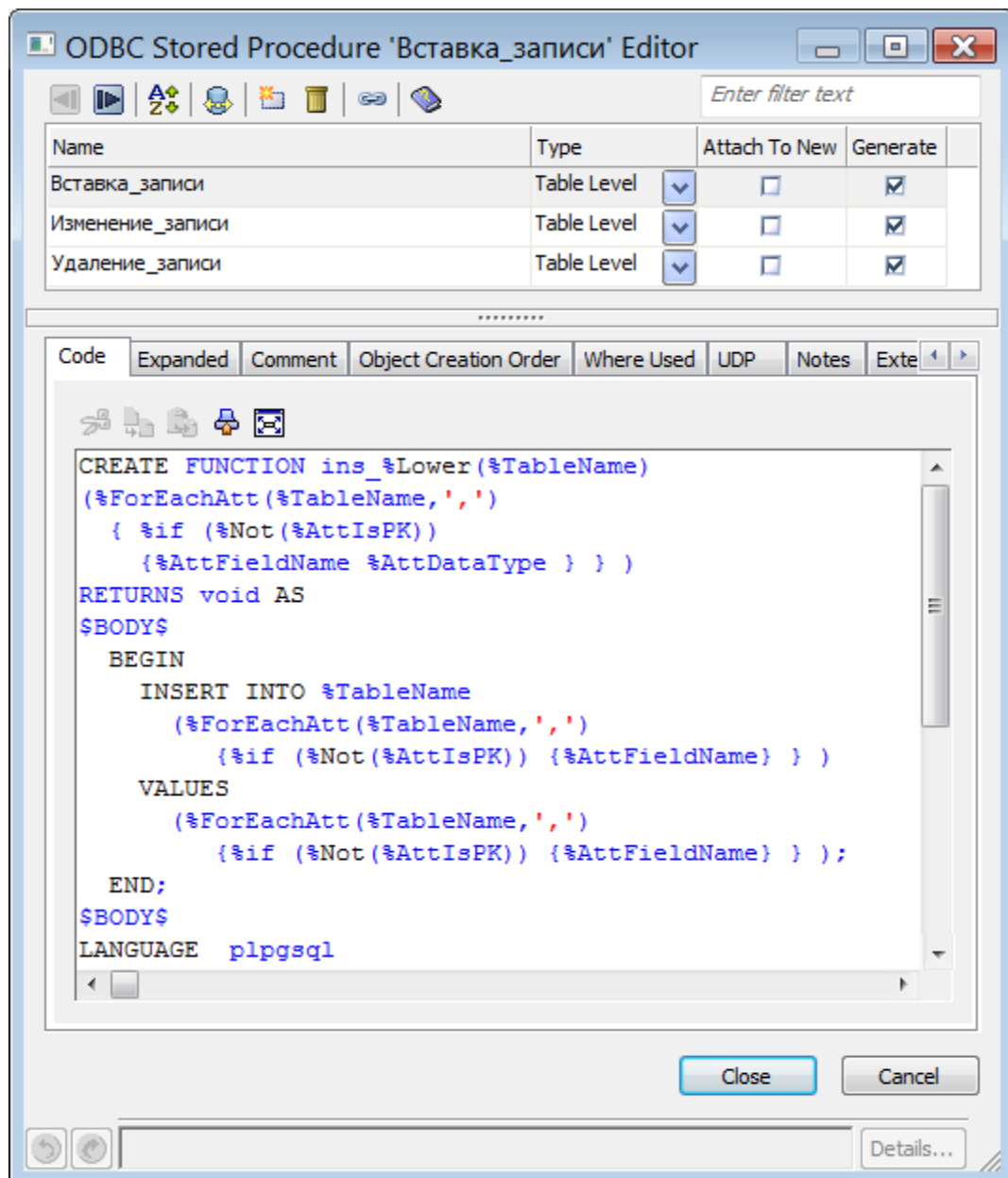


Рис. 2.19 – Создание скрипта хранимой процедуры

Удаление записи

```
CREATE FUNCTION del_ %Lower(%TableName)
(%ForEachAtt(%TableName, ','))
{%if (%AttIsPK) {%AttFieldName %AttDataType %}}
RETURNS void AS
$BODY$
BEGIN
DELETE FROM %TableName
WHERE %ForEachAtt(%TableName, 'AND')
{%if (%AttIsPK) {%AttFieldName=%AttFieldName%}};
END;
$BODY$
LANGUAGE plpgsql
```

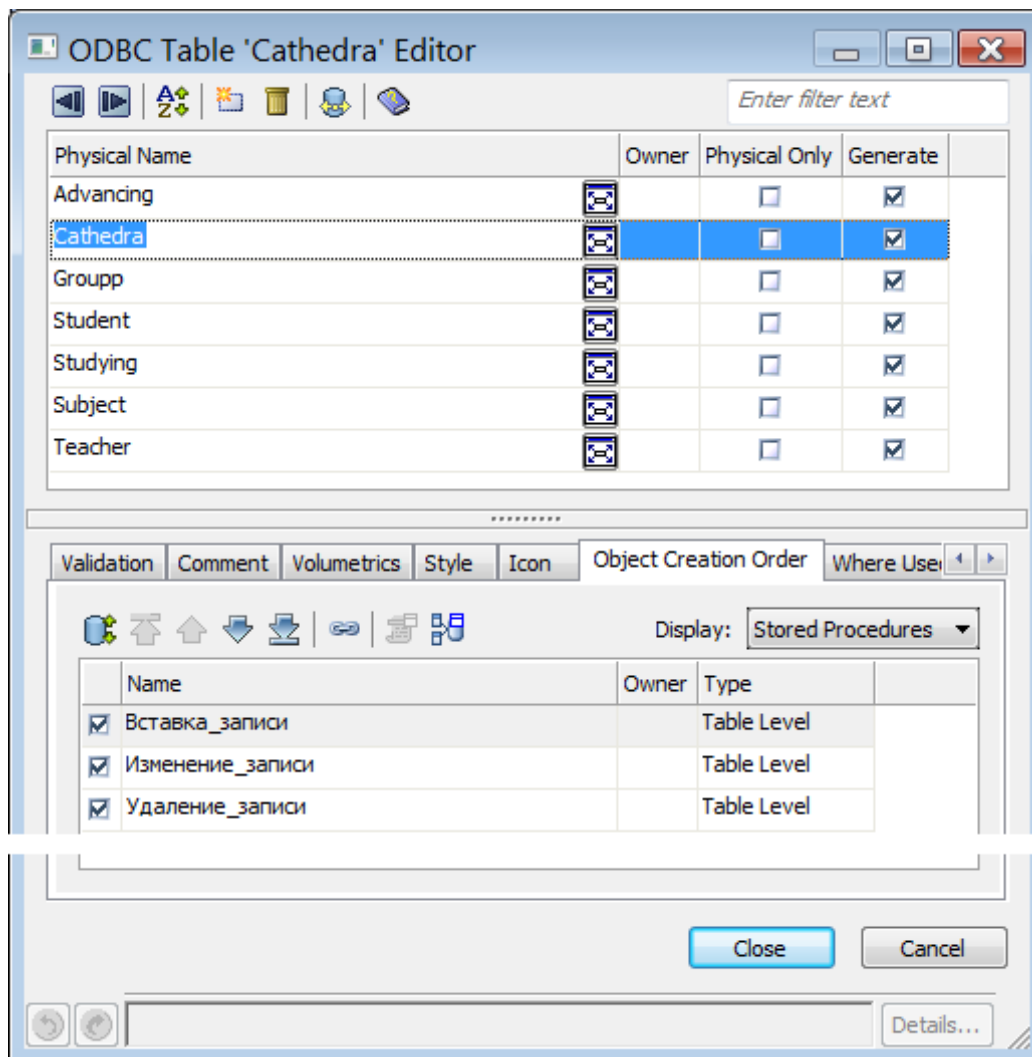


Рис. 2.20 – Подключение шаблонов скриптов к таблицам

На приведенном рисунке показан код для шаблона функции «Вставка записи» `CREATE FUNCTION ins_%Lower(%TableName)`. Остальные описанные шаблоны функций вводятся аналогично.

После этого откройте редактор и подключите созданные шаблоны к таблицам: `Advancing`, `Groupp`, `Student`, `Cathedra`, `Teacher`, `Subject`, `Studying`. Для этого откройте редактор таблиц (`Model\Tables`) перейдите в на вкладку `Object Generation Order` и, переключив уровень отображения (`Display`) на `Stored Procedures` расставьте флажки напротив процедур для каждой таблицы (см. Рис. 2.20).

2.6.3 ПРОВЕРКА РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ СКРИПТОВ

По схеме, описанной в подразделе 2.4.4 описания выполнения предыдущей лабораторной работы, проверьте результаты проделанной работы.

Убедитесь, что в окне `ODBC/Generic Schema Generation Preview` появились выражения запросов `CREATE FUNCTION` на создание функций хранимых процедур. Например, часть сценария, относящаяся к таблице `Groupp`, должна выглядеть следующим образом:

```
CREATE FUNCTION ins_groupp (Kol_studentov INTEGER,
                           Srednij_ball FLOAT)
RETURNS void AS
$BODY$
BEGIN
    INSERT INTO Groupp (Kol_studentov, Srednij_ball)
    VALUES (Kol_studentov, Srednij_ball);
END;
$BODY$
LANGUAGE plpgsql;

CREATE FUNCTION upd_groupp (Kod_gruppy INTEGER,
                           Kol_studentov INTEGER, Srednij_ball FLOAT)
RETURNS void AS
$BODY$
BEGIN
```

```

UPDATE Groupp
SET Kol_studentov=Kol_studentov,
    Srednij_ball=Srednij_ball
WHERE Kod_gruppy=Kod_gruppy;
END;
$BODY$
LANGUAGE plpgsql;

CREATE FUNCTION del_group (Kod_gruppy INTEGER)
RETURNS void AS
$BODY$
BEGIN
    DELETE
    FROM Groupp
    WHERE Kod_gruppy=Kod_gruppy;
END;
$BODY$
LANGUAGE plpgsql;

```

2.7 ЛАБОРАТОРНАЯ РАБОТА 7. ГЕНЕРИРОВАНИЕ СКРИПТА И СОЗДАНИЕ БД В POSTGRESQL

Цель работы – Создание схемы БД путем генерирования SQL-сценария по физической модели ПрО, созданной с помощью CASE-средства **ERwin**, и его выполнения в СУБД **PostgreSQL**.

Перед выполнением данной лабораторной работы внимательно изучите раздел 5.4.1 и 5.4.2 работы [1].

2.7.1 ПОДКЛЮЧЕНИЕ CASE-СРЕДСТВА ERWIN К СУБД POSTGRESQL

Для подключения к вашей целевой СУБД вы используете **ODBC**, поэтому необходимо настроить драйвер **ODBC**, чтобы указать базу данных, к которой Вы хотите подключиться. Другими словами, необходимо в **ERwin** указать на псевдоним БД или источник данных, созданный Вами при выполнении лабораторной работы №1.

Для этого вернитесь в **ERwin**. В режиме редактирования модели на физическом уровне перейдите в пункт меню **Actions\Database Connection**.

1. В появившемся диалоге **ODBC Connection** (см. Рис. 2.21) введите следующую информацию:

- **Database** – определяет версию **ODBC** для базы данных, к которой вы хотите подключиться (для **ODBC Data Source – ODBC 3.x**);
- **Authentication** – указывает на тип проверки подлинности, используемый для подключения (для **ODBC Data Source** – неактивно);
- **User Name** – логин администратора сервера (**postgres**), определенный Вами при установке СУБД **PostgreSQL**;
- **Password** – пароль администратора сервера (**admin**), определенный Вами при установке СУБД **PostgreSQL**.

В нижеследующей таблице для значений колонки **Parameters** введите следующие значения в колонке **Value**:

- **ODBC Data Source** – псевдоним БД;
- **Invoke ODBC Administrator** – поставьте «галочку».

2. После ввода перечисленных данных нажмите кнопку **Connect** – откроется диалог **Выбор источника данных** (см. Рис. 2.21).

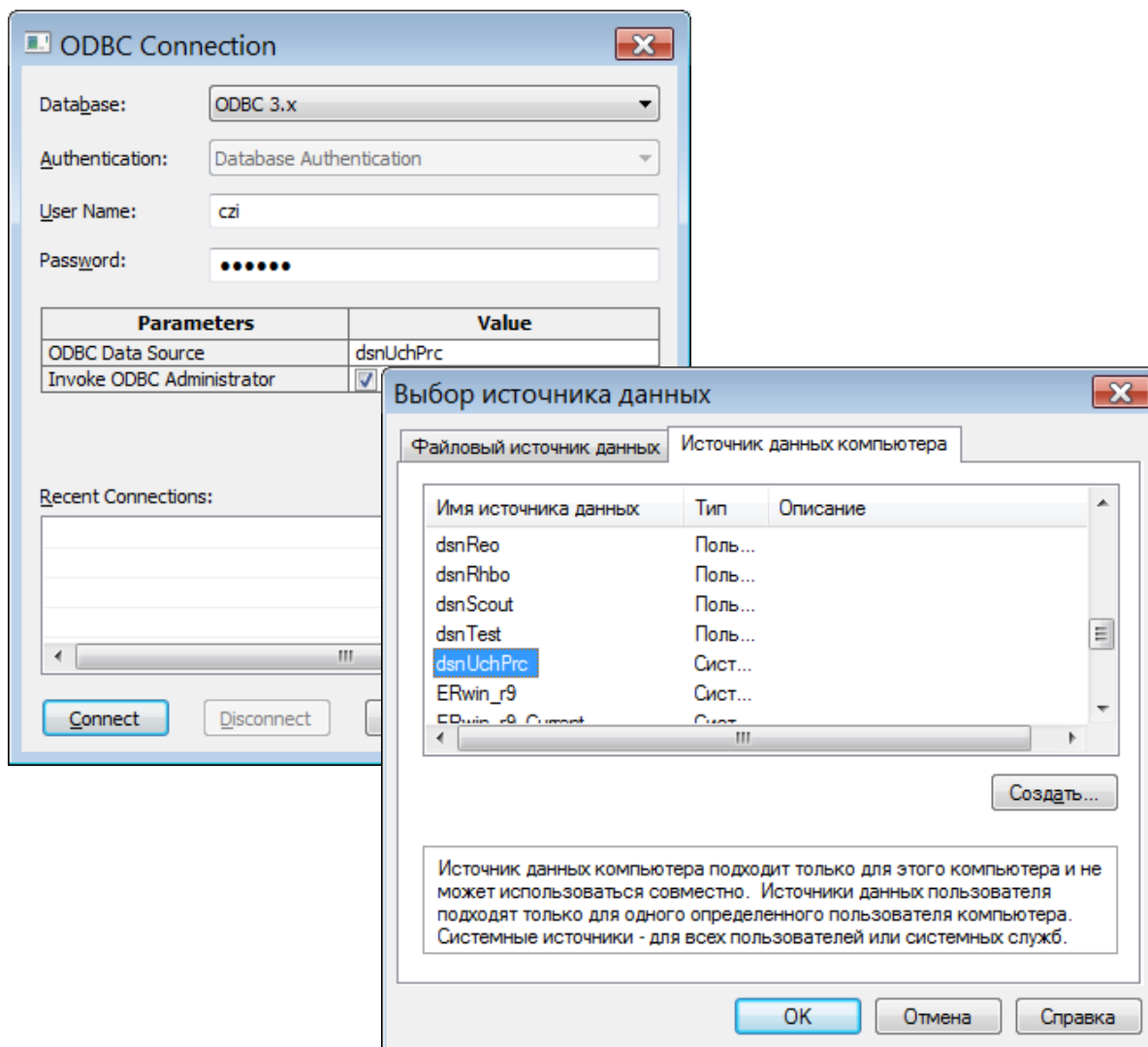


Рис. 2.21 – Выбор источника данных

На вкладке **Источник данных компьютера** выберете созданный ранее псевдоним или источник данных (в нашем примере – **dsnUchPrc**) и нажмите кнопку **ОК**.

После подключения к выбранной целевой СУБД закройте диалоговое окно соединения. Вы остаетесь в состоянии подключения к базе данных, пока вы не нажмете кнопку **Disconnect** в диалоговом окне **ODBC Connection**.

2.7.2 ГЕНЕРИРОВАНИЕ SQL-СЦЕНАРИЯ СОЗДАНИЯ БД

Основной целью процесса проектирования является генерация физической схемы БД. Для генерации схемы БД следует выбрать пункт меню «**Actions\Forward Engineer\Schema...**».

Физическая схема БД генерируется на основе логической схемы и набора установок, задаваемых в диалоговом окне генератора схем (см. Рис. 2.22). Эти установки определяют, какие элементы должны войти в схему БД. Для каждой логической схемы можно создать несколько таких наборов установок.

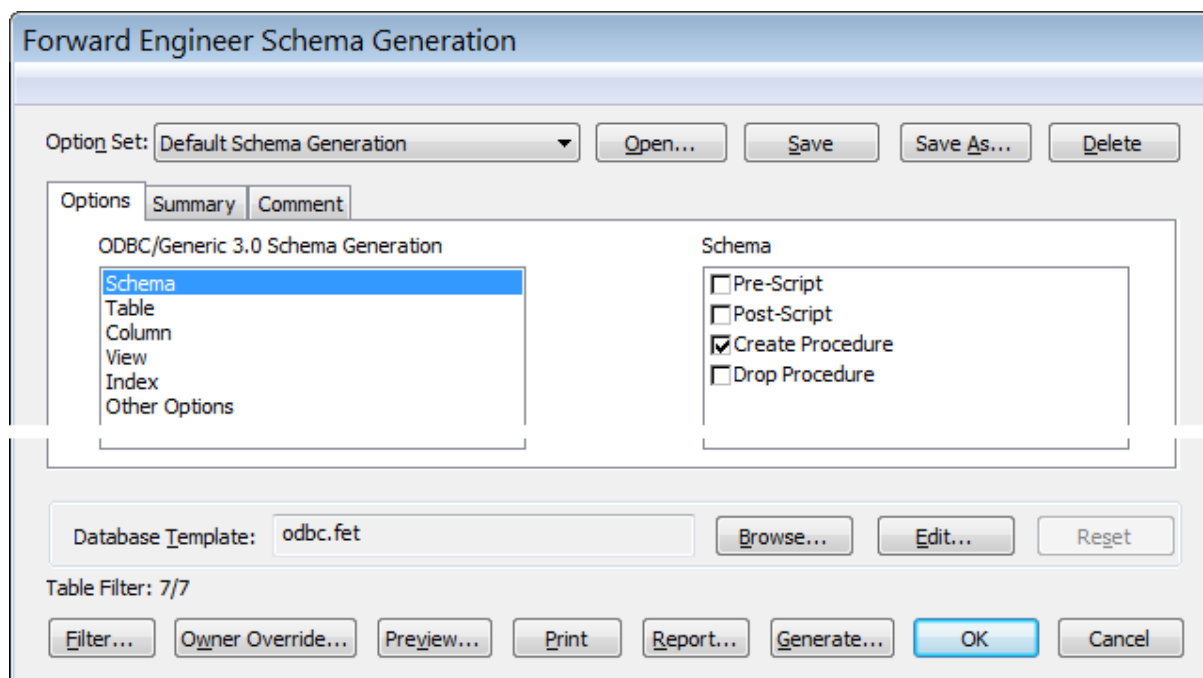


Рис. 2.22 – Диалог генератора физической схемы БД

Текущий набор установок выбирается в списке **Option Set**. Кнопки, расположенные рядом с выпадающим списком, позволяют сохранить или удалить текущий набор настроек, открыть существующий набор.

Для выбора установок перейдите на вкладку **Options** окна генератора схемы. Элементы генерируемой схемы организованы по разделам, список которых находится в левом окне страницы. В правом окне находится список элементов отмеченного раздела с флажками. Для выбора элемента, который должен использоваться при генерации схемы БД, следует проставить флажок рядом с этим элементом.

На вкладке **Summary** показываются элементы, выбранные на странице **Options**.

Вкладка **Comment** обеспечивает ввод комментариев к каждой из схем генерации.

В нижней части генератора схемы имеется ряд кнопок:

- **Filter** – вызывает редактор фильтра таблицы, с помощью которого выбираются таблицы (сущности), входящие в схему (см. Рис. 2.23). Диалоговое окно фильтра состоит из двух списков, содержащих имена таблиц (переключатель «**Display Names**» в положении **Physical**) или сущностей (переключатель «**Display Names**» в положении **Logical**). В левом списке находятся имена таблиц, исключенные из схемы генерации, в правом – вошедшие в схему;

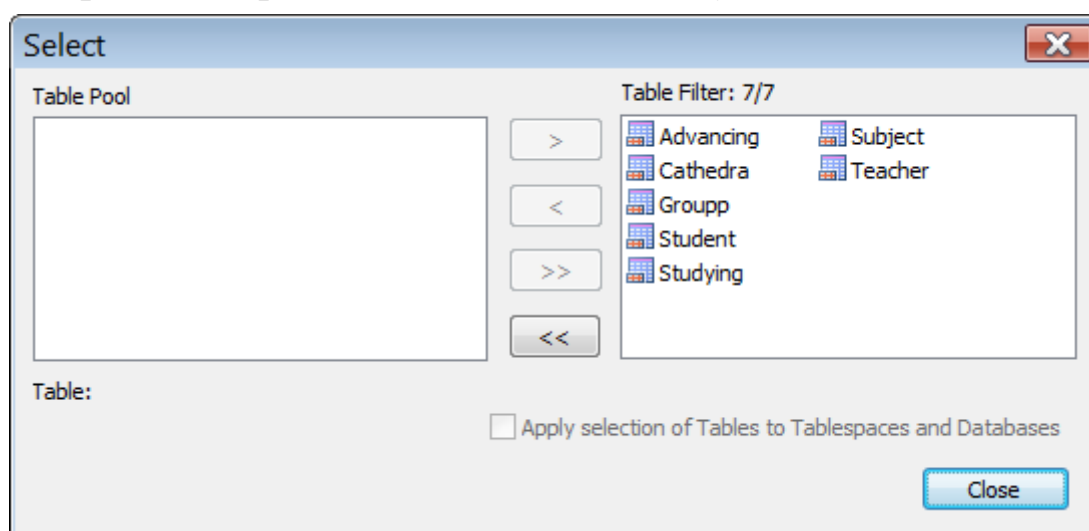


Рис. 2.23 – Диалоговое окно фильтра таблиц

- **Owner Override** – позволяет принудительно назначить нового владельца таблицы, представления или всей схемы целиком;

- **Preview** – обеспечивает просмотр сгенерированного SQL-сценария создания БД. Вызванное окно содержит стандартное текстовое окно и набор кнопок для редактирования, просмотра и печати текста сценария. Полученный сценарий можно сохранить в файле. Кнопка **Generate** диалога **Generic Schema Generation Preview** вызывает диалог генерации системного каталога БД;

- **Print** – обеспечивает печать SQL-сценария на принтере;
- **Report** – обеспечивает сохранение SQL-сценария в файле;
- **Generate** – запускает процесс генерации физической схемы БД.

2.7.3 СОЗДАНИЕ СХЕМЫ БД

Используя созданное ранее подключение, выполните сгенерированный скрипт на сервере. Для этого на физическом уровне на панели инструментов нажмите на кнопку **Forward Engineer Schema Generation** (или выберите пункт меню **Forward Engineer/Schema...**).

В окне **Forward Engineer Schema Generation** (Рис. 2.22) нажмите на кнопку **Preview ...** и в появившемся окне **ODBC/Generic Schema Generation Preview** просмотрите сгенерированный код. Прокрутив код, убедитесь, что скрипты для создания всех компонентов схемы БД (таблиц, последовательностей, триггерных функций, функций хранимых процедур) имеются в наличии.

При отсутствии скриптов для создания каких-либо компонентов схемы БД еще раз внимательно изучите созданные ранее шаблоны и установки, и при обнаружении ошибок устраните их. После этого снова вызовите окно диалога **ODBC/ Generic Schema Generation Preview**.

При наличии скриптов создания всех перечисленных компонентов схемы БД закройте окно **ODBC/Generic Schema Generation** и в окне **Forward Engineer Schema Generation** нажмите кнопку **Generate....**

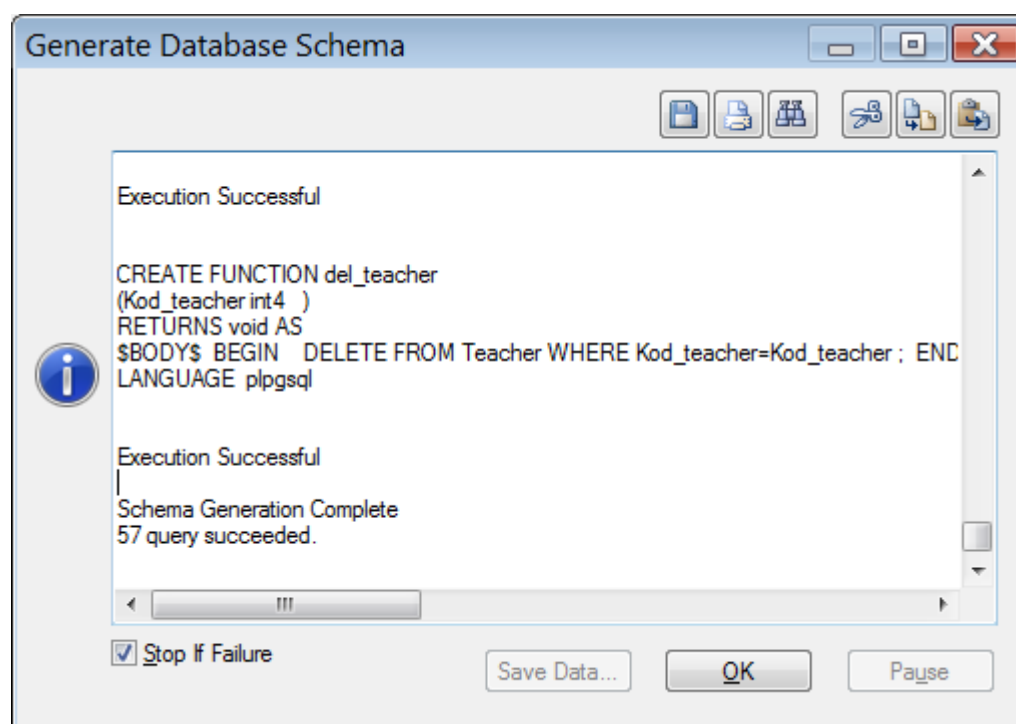


Рис. 2.24 – Окно *Generate Database Schema*

В ходе выполнения скрипта могут появиться сообщения о том, что какой-либо триггер или генератор уже существует - пропустите их. Это происходит потому, что некоторые ключи, для которых создаются генераторы (последовательности) и триггерные функции присутствуют в нескольких таблицах и сгенерированный скрипт содержит одинаковые блоки.

В случае успешного выполнения скрипта, рассмотренного выше, результат обращения к серверу, выводится сообщение примерно следующего содержания (см. Рис. 2.24):

Schema Generation Complete
57 query succeeded.

2.7.4 ПРОВЕРКА РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ СКРИПТОВ

Проверка результатов выполнения скриптов осуществляется путем просмотра состава сгенерированной БД в СУБД PostgreSQL.

Для этого откройте интерфейсное приложение pgAdmin III для доступа к объектам БД, созданной СУБД PostgreSQL.

В браузере объектов pgAdmin III («Браузер объектов» на левой панели) найдите Вашу БД, созданную в рамках выполнения предыдущих лабораторных работ. Для этого раскройте узел требуемого сервера и выберите узел БД uchPrc.

Далее раскройте узел «Схемы(1)», найдите в нем схему по умолчанию public. Раскройте на дереве узел public и в случае успешного создания схемы разрабатываемой БД Вы должны обнаружить в числе прочих узлы:

- Таблицы (7);
- Последовательности (5);
- Триггерные функции (5);
- Функции (21).

Это означает, что при выполнении скрипов, сгенерированных с помощью CASE-средства ERwin, созданы:

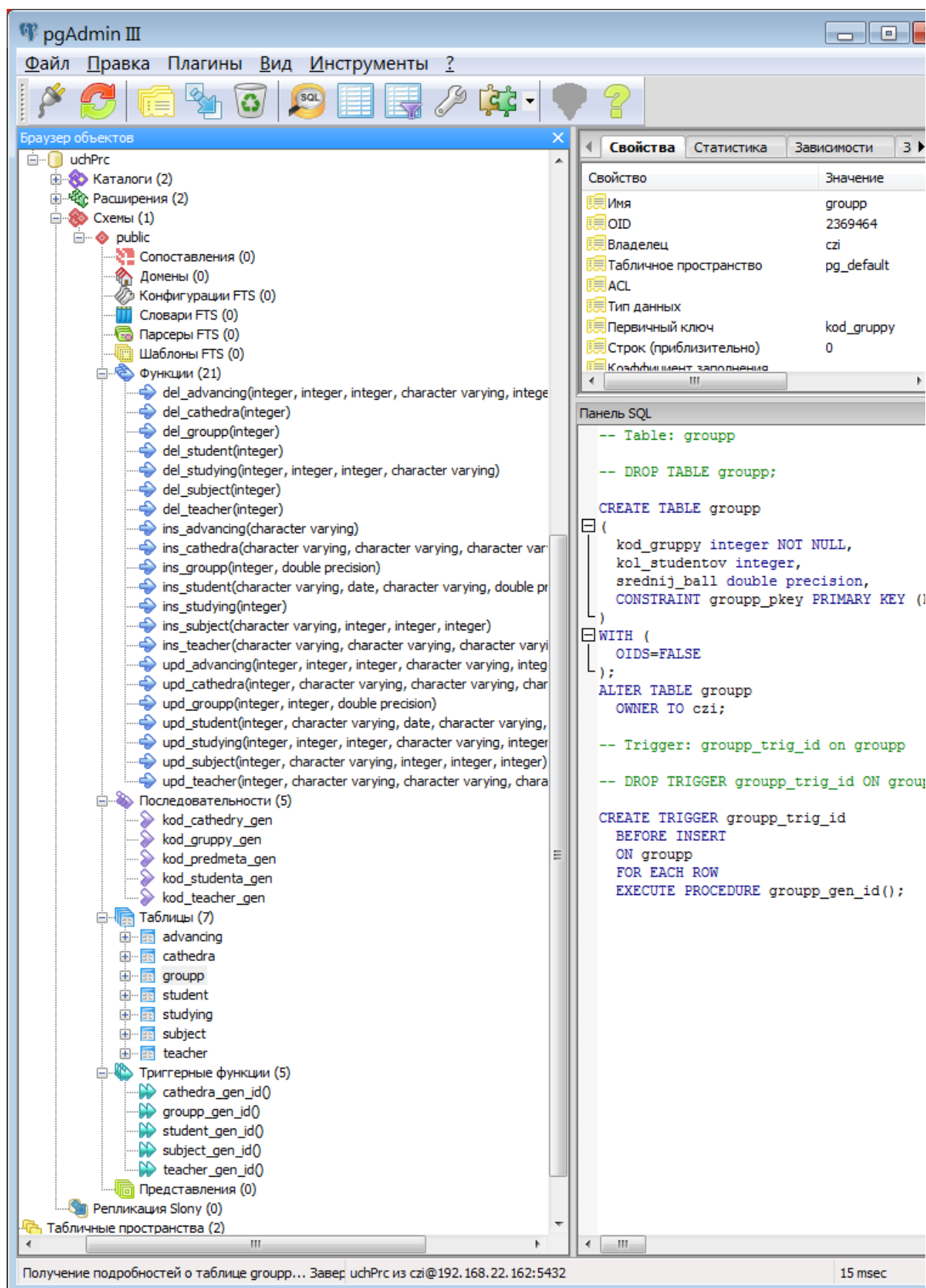


Рис. 2.25 – Результат генерации схемы БД в pgAdmin

- 7 таблиц;
- 5 последовательностей для автоматического наращивания значений первичных ключей 5 таблиц.

Для таблиц, первичные ключи которых формируются из вторичных ключей, последовательности не создаются;

- 5 триггерных функций. Именно эти триггерные функции осуществляют наращивание значений первичных ключей после вставки записей в соответствующие таблицы;

– 21 функция хранимых процедур, вызываемых из кода приложений БД для осуществления вставки, обновления и удаления. Число 21 определяется количеством таблиц (в нашем случае 7) и количеством создаваемых для каждой из них функций (в нашем случае 3).

Раскройте эти узлы и изучите их состав. Окно «Браузер объектов» с созданными объектами схемы БД `uchPrc`, размещенное на левой панели интерфейсного окна приложения `pgAdmin`, приведено на Рис. 2.25.

ЗАКЛЮЧЕНИЕ

Предметом изучения данных учебно-методических указаний является освоение навыков автоматизированного проектирования БД информационных систем в ходе выполнения лабораторных работ. В качестве средства разработки рассматривается CASE-средство разработки БД ERwin.

Этот выбор объясняется тем, что ERwin – наиболее популярное средство автоматизированного создания БД среди российских разработчиков и является одним из лучших в своем классе (среди CASE-средств среднего класса по мощности) по соотношению «цена/качество».

В предлагаемом пособии подробно изложены все этапы проектирования базы данных путем разработки ее логической и физической модели с помощью ERwin. В рамках разработки этих моделей было выполнено:

- создание сущностей предметной области;
- установка атрибутов сущности;
- установка связей между сущностями;
- создание пользовательских свойств;
- разработка генераторов, триггеров, хранимых процедур.

Изложение материала методических указаний сопровождается демонстрацией примером разработки небольшой предметной области «Учебный процесс».

Усвоение навыков разработки БД с помощью CASE-средства ERwin поможет студентам сравнительно легко разрабатывать и поддерживать довольно сложные модели реальной действительности. Весьма ценное качество таких средств – масштабируемость разработок. Благодаря этому качеству разработанный код можно легко перенести от одной СУБД к другой, практически не внося в него изменения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Токмаков Г. П. Проектирование информационных систем с помощью CASE-средств: учебное пособие. – Ульяновск: УлГТУ, 2017.

Учебное издание

ТОКМАКОВ Геннадий Петрович

ПРОЕКТИРОВАНИЕ
ИНФОРМАЦИОННЫХ СИСТЕМ
С ПОМОЩЬЮ CASE-СРЕДСТВ

Учебно-методические указания

Редактор М. В. Теленкова

Подписано в печать 20.07.2004. Формат 60×84/16.

Усл. печ. л. 8,37. Уч.-изд. л. 8,00.

Тираж 100 экз. Заказ .

Ульяновский государственный технический университет

432027, г. Ульяновск, ул. Сев. Венец, 32.

Типография УлГТУ. 432027, г. Ульяновск, ул. Сев. Венец, 32.