МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт энергетики и электротехники
(институт)
Кафедра «Промышленная электроника»

27.03.04 Управление в технических системах
(код и наименование направления подготовки)

Системы и технические средства автоматизации и управления
(направленность (профиль)

БАКАЛАВРСКАЯ РАБОТА

на тему «Разработка специализированного модуля контроля для дистанционной подготовки студентов направления управление в технических системах»

Студент(ка)	Н.Д. Морунов	
	(И.О. Фамилия)	(личная подпись)
Руководитель	О.Ю. Копша	
	(И.О. Фамилия)	(личная подпись)
Консультанты		
	(И.О. Фамилия)	(личная подпись)
	(И.О. Фамилия)	(личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.А. Шевцов _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2016

Аннотация

Тема данной бакалаврской работы – «Разработка специализированного модуля контроля для дистанционной подготовки студентов направления управление в технических системах», в рамках которой был рассмотрен один из методов моделирования информационных систем и разработана информационная система контроля успеваемости студентов.

Задача состоит в проектировании информационной системы, включающей в себя базу данных контрольно-измерительных материалов и программный модуль для прохождения тестирования студентами и добавления в базу новых тестовых опросов преподавателями.

Исходя из поставленной задачи, в бакалаврской работе рассмотрены ряд следующих вопросов: выбор стратегии конструирования программного обеспечения, проектирование информационной системы контроля успеваемости студентов, методы создания специализированного программного модуля. Система была разработана и испытана.

Содержание

Введение

Ни для кого не секрет, что информационные технологии охватили уже практически все сферы человеческой жизни. Компьютеризации и информатизации подверглось и обучение. Для объективной оценки знаний, а также для дистанционного обучения стало необходимым разрабатывать специальные автоматизированные системы информационного контроля знаний. Сейчас активно развиваются интеллектуальные системы тестирования, которые, используя математические модели и специальные алгоритмы, предполагают интеллектуализацию обучения.

Цель данной бакалаврской работы – разработать специализированный модуль контроля знаний учащихся.

1 Выбор стратегии конструирования программного обеспечения

Проектируемый программный модуль должен запускаться на локальных, а возможно и удалённых, системах и получать доступ к базе данных, расположенной на одном сервере. За неимением большого числа компьютеров для чисто вычислительных целей, для такой информационной системы следует выбрать архитектуру «клиент-сервер».

Сервер возьмёт на себя хранение и частично обработку данных. Так как сервер хранит данные о тестах и об успеваемости студентов, к которым необходимо организовать доступ с других клиентов, на сервере должна быть установлена СУБД – система управления базами данных. Также СУБД разграничит уровень доступа различных пользователей. Сегодня наиболее известными СУБД являются: Oracle, Microsoft SQL Server, MySQL, PostgreSQL и др. Самые лидирующие позиции занимают Oracle и MS SQL, не столько отличающиеся по возможностям, сколько кроссплатформенностью. Следует учесть, если база данных будет располагаться на компьютере с установленной операционной системой Windows, то лучшим выбором, конечно, будет Microsoft SQL, если же на другой системе или база данных должна быть переносима на другие платформы, то лучше выбрать Oracle.[1]

Программный модуль подключается к серверу с помощью специальных драйверов, на операционных системах Microsoft уже установлены драйвера, для доступа к SQL Server. Так же к SQL Server`у можно обращаться через другие платформы, с предварительно установленными на них драйверами. Для взаимодействия программы с драйверами были разработаны программные интерфейсы доступа к базам данных – OLE DB и ODBC.

Object Linking and Embedding, Database (OLE DB) – наиболее современная разработка самой компании Microsoft, которая получила широкое применение. Отличительной особенностью является использование в основе объектной модели компонентов(COM), воплощающей в себе идеи инкапсуляции и полиморфизма.

Open Database Connectivity (ODBC) – что в переводе открытый механизм взаимодействия с базой данных, использует в своей основе программный стандарт, интерфейс уровня вызовов(CLI). Разрабатывался вместе с компанией Microsoft, и

они посчитали его более удачным, чем новый OLE DB. В недалёком будущем Microsoft хочет отменить поддержку OLE DB и вернуться к ODBC. Также следует отметить, что OLE DB поддерживается только на Windows, а ODBC способен обращаться к базе данных и с других платформ.[2]

Так как проектируемая система контроля успеваемости студентов будет организована в локальной сети университета, с возможностью удалённого доступа, а в университете на большинстве компьютеров установлена Windows, выберем MS SQL Server и ODBC, в перспективе на будущую доработку системы тестирования и создание программных моделей для смартфонов, планшетов и других гаджетов.

В качестве среды, в которой будет разрабатываться программный модуль, выберем Microsoft Visual Studio. В настоящее время пакет программ Visual Studio Community 2015 является полностью бесплатной некорпоративной средой программирования.

Приложение будет писаться на C++ с использованием библиотеки MFC(Microsoft Foundation Classes). MFC упрощает процесс написания приложений под Windows. Все приложения Windows пишутся с помощью интерфейса программирования приложений (API). Но при создании даже простого окна потребуется писать много шаблонного кода: регистрация класса окна, создание палитры, создание самого окна, цикл обработки сообщений и др. MFC была разработана как раз с целью ускорения и упрощения создания таких приложений. MFC представляет собой иерархию классов, в которых содержатся самые основные структуры и функции API. Библиотека классов была разработана так, что может без конфликтов использоваться вместе со стандартными API функциями. Библиотека лишь слегка уступает по производительности чистым функциям, она была максимально оптимизирована разработчиками. [3]

2 Проектирование информационной системы контроля успеваемости студентов

2.1 Общеструктурная схема и определение структурных связей

Вся информационная система контроля успеваемости студентов подразделяется на два основных блока, это клиентская часть (программный модуль и драйвер), и серверная часть(SQLServer и база данных тестов).

В программном модуле составляются запросы и обрабатываются результирующие наборы. Для этого модуль вызывает функции ODBC, которые обрабатывает драйвер, и отправляет сформированные запросы на SQLServer или принимает данные и возвращает результаты запроса приложению.[4]

База данных тестов содержит контрольно-измерительные материалы, информацию об успеваемости студентов, внутренние процедуры для обработки результатов. SQLServer принимает запросы от клиента и обрабатывает их, выполняя необходимые действия над базой данных, а также определяет роли управления данными и доступа к ним.

Всего взаимодействуют три основные роли:

1) Администратор – запускает сервер, настраивает его, и поддерживает его в исправном состоянии.
2) Преподаватель – имеет возможность создавать и редактировать тесты, а также допускать к ним студентов и редактировать траектории обучения.
3) Студент – может только проходить тестирование и просматривать результаты, для остального доступ закрыт.

Администратор имеет полный доступ ко всему серверу, в том числе и к базе данных тестов. Он может напрямую добавлять, изменять или удалять данные из таблиц, вызывать все пользовательские процедуры, просматривать представления, добавлять новые элементы в базу данных (таблицы, представления, хранимые процедуры). Так как администратор вправе изменить названия существующих элементов, в программном модуле необходимо добавить файл конфигурации, в котором будут сопоставляться виртуальные объекты приложения с физическими объектами на сервере.

Взаимодействие администратора с базой данных может происходить как напрямую (отправка SQL запросов на сервер), так и через SQL Server Management Studio, которая входит в пакет программ SQL Server. Программа облегчает взаимодействие администратора с базами данных, благодаря визуальному интерфейсу. Для основных операций студия сама формирует запросы и отправляет их на сервер (выполнение хранимых процедур, создание представлений), для других же в программе имеются шаблоны, которые администратор заполняет сам или пишет запросы вручную, во встроенном в программу текстовом редакторе.

Через программный модуль преподаватель может создавать новые тесты, прописывать к ним вопросы и ответы, редактировать уже созданные тесты, задавать общую (для всей группы) или индивидуальную траекторию обучения, формируя порядок прохождения тестов. Для него открыт не полный доступ к таблицам (только чтение), а запись и добавление новых данных производится через представления, с предварительно заданными триггерами добавления и вставки. Доступ к полным таблицам ограничен, по причине нарушения целостности информации каждого преподавателя в отдельности. Тогда бы один преподаватель мог попросту удалить или испортить тесты других преподавателей или назначить допуск к тесту любому студенту по любой дисциплине.

Доступ студента к базе максимально урезан, в целях защиты целостности базы данных и ответов на тесты. Даже если студент додумается отправлять на сервер SQL запросы, вместо честного прохождения теста в программе, он не получит доступа к основным таблицам тестов. Через приложение студент может узнать информацию о своей группе, о себе, он может просмотреть учебный план, ФИО (фамилия имя отчество) преподавателя.

Схема информационной системы со связями между её элементами представлена на рисунке 2.1.

Сервер необходимо расположить на персональном компьютере с установленной операционной системой Windows, как писалось ранее, SQL Server не поддерживается на других системах, так как является разработкой Microsoft. На одном сервере может быть несколько баз данных. Поскольку программный модуль

определяет базу через файл конфигурации, в системе может быть и несколько баз данных тестов.

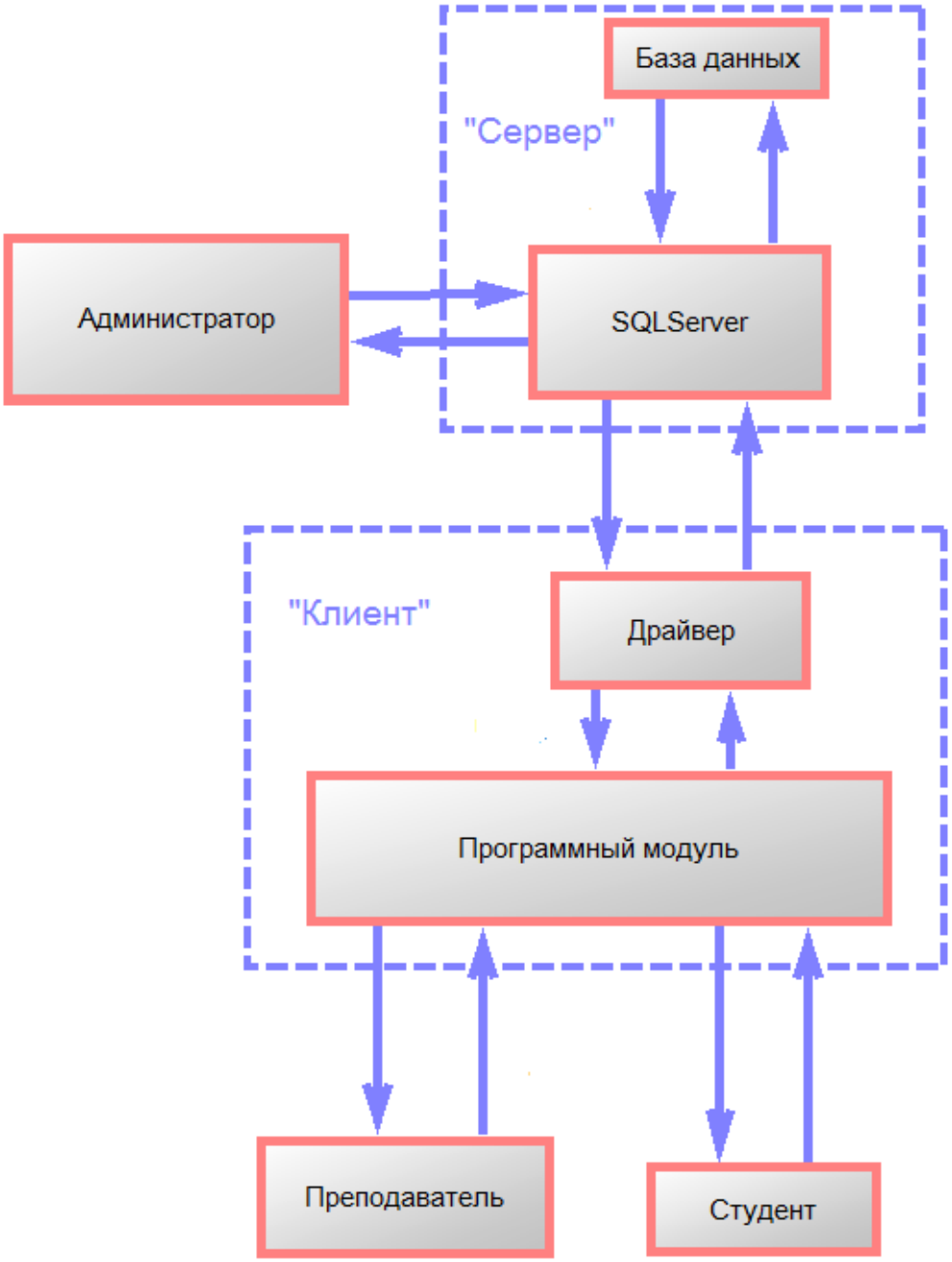

Рисунок 2.1 – Схема информационной системы контроля успеваемости студентов

Для удобства клиент-приложение унифицировано, и за одним компьютером может работать как преподаватель, так и студент. В целях защиты доступа, приложение должно скрывать пароль во время входа и удалять данные о паролях сразу после подключения к серверу.

2.2 Определение функций и назначений программного модуля

Для определения функций модуля воспользуемся диаграммой прецедентов. Диаграмма прецедентов (вариантов использования) это средство унифицированного моделирования информационных систем (UML), наглядно демонстрирующая функции информационной системы, отношения между экторами и их возможностями (прецедентами).[5]

Эктор – это система, класс или человек, представленный как множество ролей, исполняемых при взаимодействии с сущностями.

Прецеденты – это возможности ролей, которые может предоставить информационная система.

Преподаватель может создавать, редактировать и удалять тесты, может настраивать траекторию обучения, просматривать успеваемость студентов, допускать студентов до теста. Студент может проходить тест, просматривать результаты тестирования и успеваемость всей группы.

В обязанности администратора входит создание базы данных на основе разработанной реляционной физической модели. Добавление необходимых процедур и регистрация в базе всех пользователей (преподавателей и студентов). Также исправление возможных ошибок и поддержание базы в рабочем состоянии. Перед началом работы администратор должен установить пакет программ SQL Server.

Экторами в данном случае являются студент, преподаватель, администратор, а также SQLServer и сам программный модуль.

Программный модуль подключается к базе данных тестов, отправляет запросы на сервер и принимает результаты в виде результирующего набора, далее обрабатывает их, и выполняет запрашиваемые пользователями (преподавателем или студентом) функции.

SQLServer подключает пользователя к базе данных в соответствии с его ролью, принимает запросы и выполняет необходимые операции (указанные в запросе) с базой данных, далее отправляет результат запроса программному модулю.

Построим диаграмму прецедентов, для этого воспользуемся программным пакетом Rational Roses.[6]

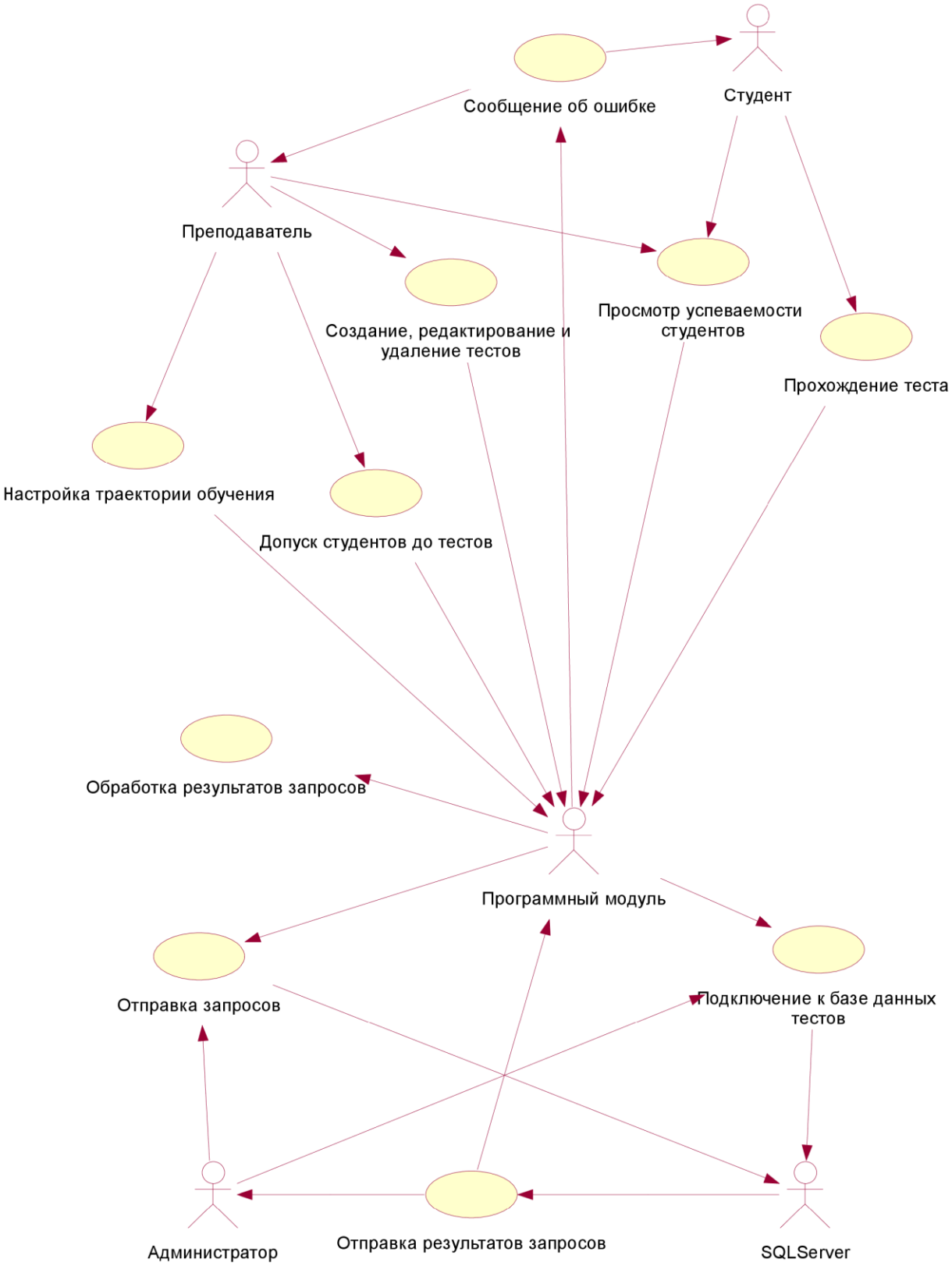

Рисунок 2.2 – Диаграмма прецедентов информационной системы контроля успеваемости студентов

3 Методы создания специализированного программного модуля

3.1 Проектирование базы данных для тестирования студентов

База данных – это систематизированный набор данных, составленный таким образом, чтобы его потом можно было обработать на компьютере.

Проектирование любой базы данных начинается с построения ER-модели(entity-relationship). Фактически модель «сущность-связь» представляет собой концептуальную модель базы данных, и используется при высокоуровневом проектировании баз данных. Модель можно представить в виде ER-диаграммы, на которой выделяются ключевые сущности и проводятся связи между ними.[7]

Сущностью называю некоторый объект, который идентифицирован неким способом, отличающим его от остальных объектов, а связь – это соединение между двумя сущностями.

В базе данных присутствуют следующие сущности:

1) Пользователь – сюда входят и студент, и преподаватель. При разделении сущности на две: сущность студент и сущность преподаватель, возникает проблема с неправильным содержанием поля «логин», а именно возможность записи одного пользователя в студенты и в преподаватели одновременно, сущность пользователь объявлена для защиты базы данных от подобных ошибок. Сущность содержит два свойства: логин (соответствующий логину пользователя базы данных) и ФИО (Фамилия Имя Отчество).
2) Группа – группа студентов, содержит два свойства – код группы (специальная аббревиатура группы) и расшифровка (расшифровка этой аббревиатуры).
3) Дисциплина – предмет, по которому необходимо провести тестирование, состоит из свойств код дисциплины и название.
4) Успеваемость – сущность определяет успеваемость студентов по каждой дисциплине, состоит из свойств номер оценки (номер контрольной точки), допуск, сырые баллы (соответствуют числу правильно отвеченных вопросов), приведённые баллы (оценка студента от 0 до 100).

5) Тест – самая большая и основная сущность, включает следующие свойства: номер теста, название теста, номер вопроса, вопрос, номер ответа, ответ и правильность.

Все сущности связаны связями:

«Состоит в» – связь связывает пользователя и группу, если пользователь студент, то он относится к некоторой группе. Так как в группе может быть много студентов, и каждый студент состоит только в одной группе, связь «многие к одному»;

«Преподаёт в» – тройная связь, связывает сразу 3 сущности, пользователь, как преподаватель, преподаёт в некоторой группе, некоторую дисциплину. Считаем, что один преподаватель может преподавать в нескольких группах, несколько дисциплин, одна группа может обучаться несколькими преподавателями и имеет некоторый учебный план (несколько дисциплин), одна дисциплина может преподаваться в разных группах, и разными преподавателями, все три связи «многие ко многим»;

«Учится с» – связь связывает пользователя, как студента, с его успеваемостью, связь «один к одному»;

«По» – связь успеваемости и дисциплины, так как успеваемость определяется по каждой дисциплине, связь «многие к одному»;

«Относится к» – связь дисциплины и тестов, в каждой дисциплине есть некоторое количество тестов – «многие к одному».

Для проектирования воспользуемся такой CASE – системой, как PowerDesigner. Средство проектирования PowerDesigner позволяет создавать различные модели данных и баз данных, анализировать взаимодействие этих моделей, конвертировать из одной в другую, генерировать новую и многое другое.[8]

В среде создаём новую концептуальную модель, и заполняем её в соответствии с уже разработанной ER-моделью. Полученная в итоге концептуальная модель базы данных представлена на рисунке 3.1.

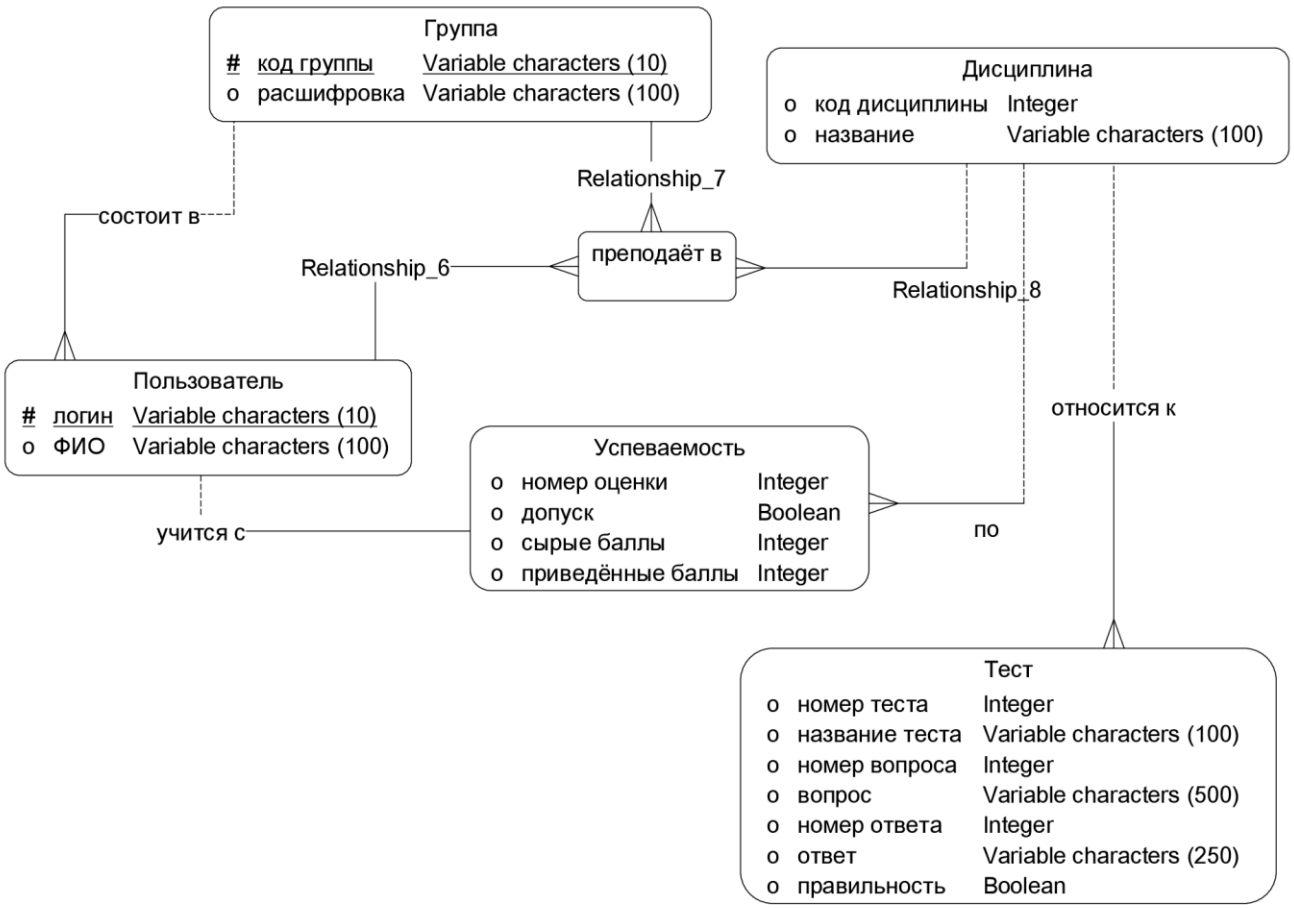

Рисунок 3.1 – Концептуальная модель проектируемой базы данных

Следующим шагом необходимо создать логическую модель по полученной концептуальной модели. На первый взгляд модели практически не отличаются, отличие заключается в том, что логическая модель определяет тип базы данных (иерархическая, реляционная, сетевая и т.д.). Распространенный и наиболее простой тип баз данных это реляционные базы данных. Логическая модель реляционной базы данных добавляет к сущностям внешние ключи.

Далее просто воспользуемся функционалом программы PowerDesigner и сгенерируем логическую модель на базе концептуальной. Программа сама по заданным связям определит и допишет внешние ключи. Затем самостоятельно выделим первичные ключи в каждой сущности. Результат представлен на рисунке 3.2.

Следующим шагом необходимо создать физическую модель базы данных, заменить абстрактные сущности, реальными таблицами (отношениями). Воспользуемся всё тем же средством PowerDesigner, и сгенерируем физическую модель из логической модели. Спроектированная физическая модель представлена на рисунке 3.3.

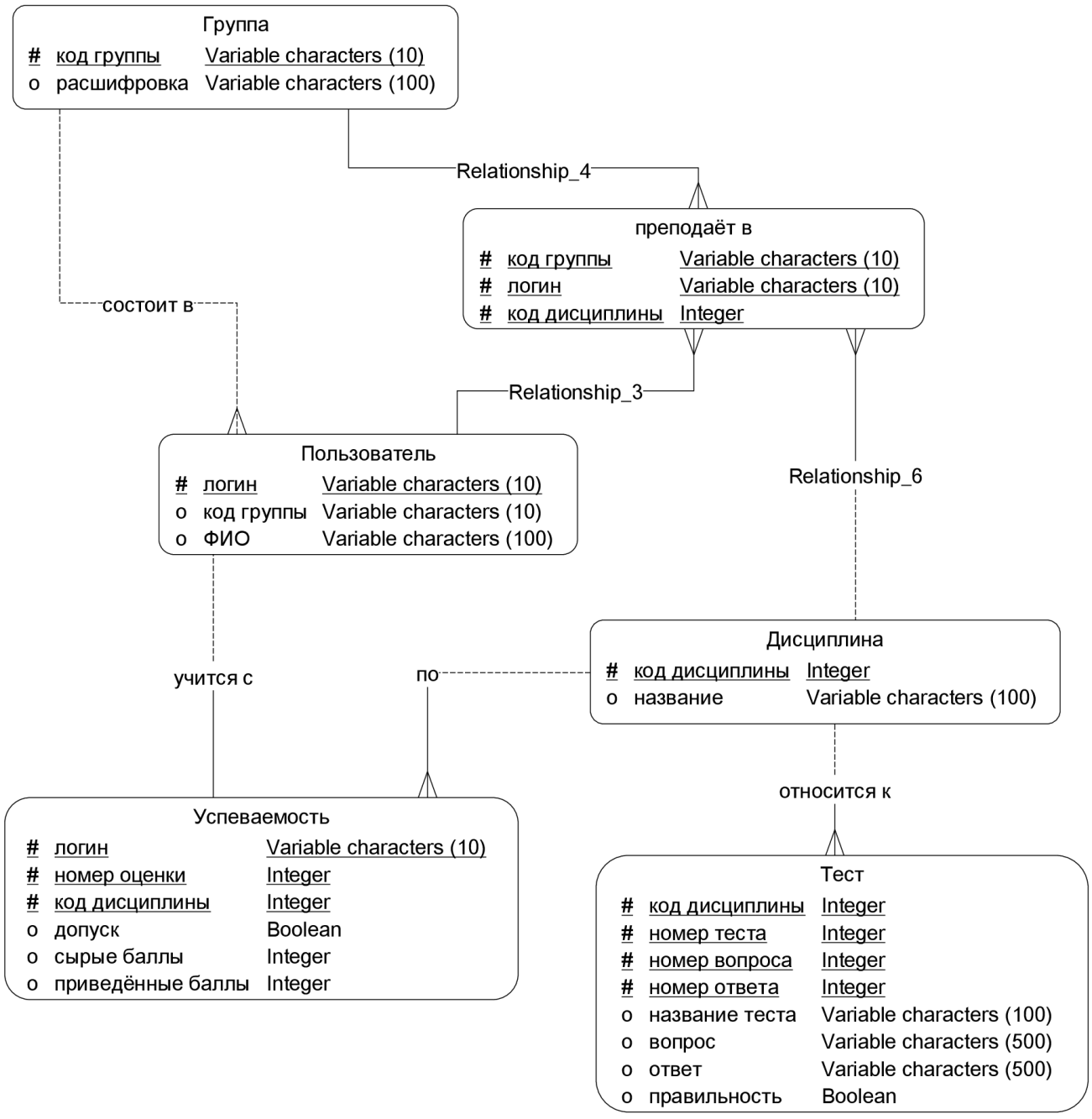

Рисунок 3.2 – Логическая модель проектируемой базы данных

Теперь после того как у нас есть физическая модель реляционной базы данных, необходимо провести нормализацию её отношений. Нормализация – приведение базы данных к структуре с минимальной избыточностью, т.е. уничтожение повторяющихся и лишних данных. На практике целью нормализации не является именно уменьшение избыточности, скорее её последствия, а именно отсутствие аномалий, и высокое быстродействие такой базы данных. При увеличении степени нормализации, быстродействие растёт нелинейно. Быстродействие увеличивается только где-то до четвёртой нормальной формы, а после падает.[9]

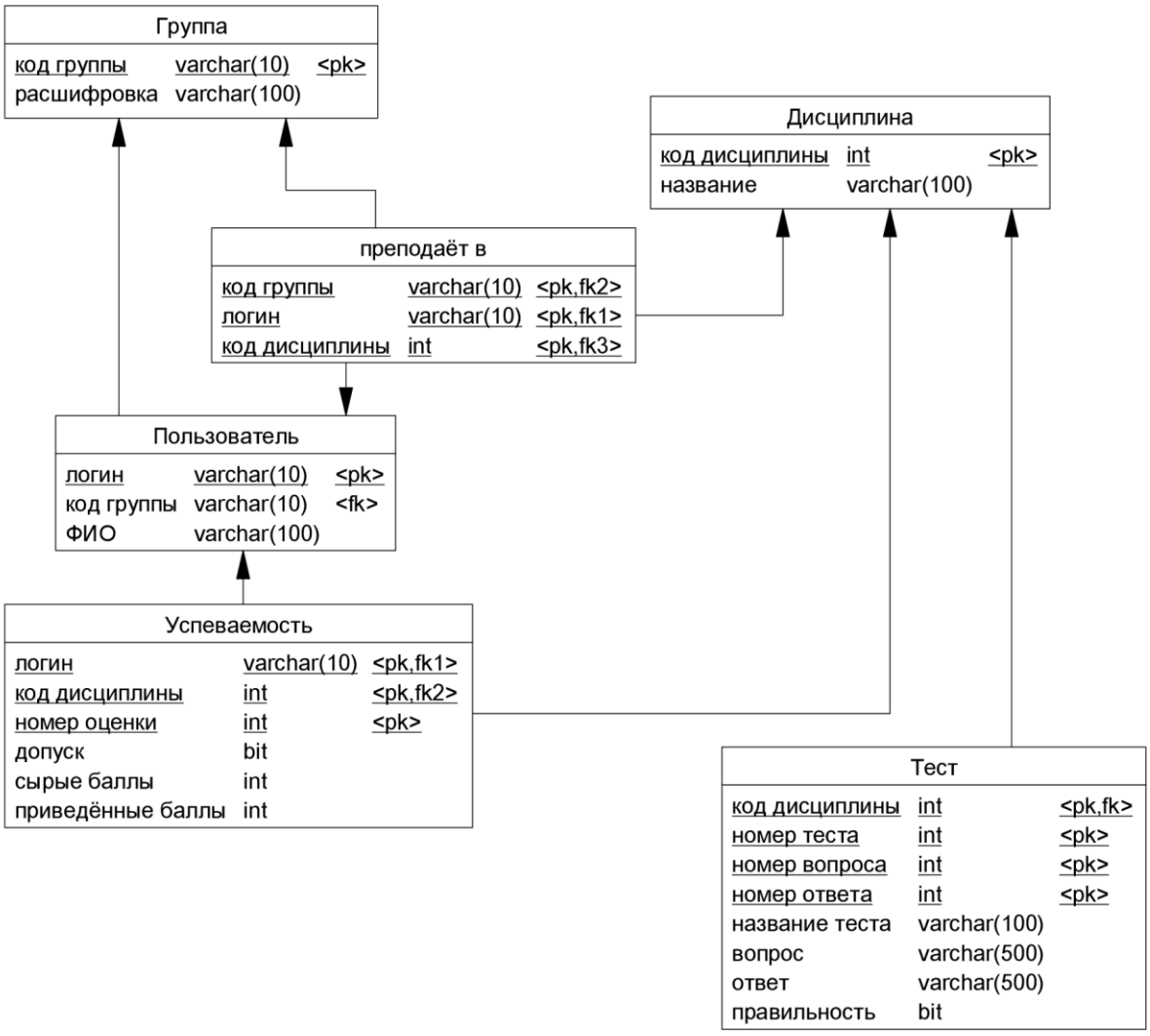

Рисунок 3.3 – 1НФ физической модели проектируемой базы данных

База данных поэтапно проходит от первой нормальной формы до шестой, каждая нормальная форма ограничивается определённым типом функциональных

зависимостей (транзитивные зависимости, многозначные зависимости и т.д.) и устранением соответствующих номеру нормальной формы аномалий. Опять же на практике базу данных не приводят к шестой форме, четвёртая форма и выше не способствует оптимизации базы данных, а иногда даже, наоборот, на много увеличивает количество отношений и снижает быстродействие, из-за чего обычно используют только первые три - четыре формы, а остальные представляют скорее теоретический интерес.

3.1.1 Первая нормальная форма

Переменная отношения находится в первой нормальной форме тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.[10]

Первую нормальную форму ещё называют базовой, так как по определению самого отношения, любое отношение должно находиться в первой форме. Рассмотрим, что же такое отношение и чем оно отличается от простых таблиц. Во-первых, отношение не упорядочено ни по строкам, ни по столбцам. Отношение не содержит повторяющихся строк (кортежей). Каждое пересечение столбца и строки сдержит атомарное (неделимое) значение из соответствующего домена. Все столбцы (атрибуты) являются обычными, т.е. не содержат скрытых компонентов, доступных только при вызове специальных операторов. Проще говоря, отношение это не массив.

Проверим нашу модель базы данных на соответствие первой форме. Первое условие – атомарность атрибутов. Так как ни фамилии, ни имя нигде отдельно не фигурируют, будем считать атрибут ФИО неделимым (в дальнейшем поле всегда можно будет разбить на три).

Второе условие - отсутствие повторяющихся групп.

Отношение Тест уже приведено к виду нормального отношения, исключены массивы ответов и вопросов. Следовательно, ошибок нет, база данных уже находится в первой форме, рисунок 3.3.

3.1.2 Вторая нормальная форма

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме, и каждый неключевой атрибут неприводимо зависит от её потенциального ключа (уникального набора атрибутов)[10]

Под термином неприводимость подразумевается неделимость ключа для всех атрибутов, все неключевые атрибуты зависят только от полного ключа. Очевидно, что данное условие не выполняется в отношении Тест. Название теста зависит только от кода дисциплины и номера теста, вопрос от кода дисциплины, номера вопроса и номера ответа, и только ответ и правильность зависит от целого составного ключа. Разобьем отношение Тест на три составляющих: Тест (название), Тест (вопросы) и Тест (ответы). Условие второй нормальной формы следует проверять только у отношений с составными ключами (с двумя и более атрибутами). Отношение Успеваемость удовлетворяет условию: допуск, сырые баллы и приведённые баллы зависят только от конкретной контрольной точки для конкретного студента, в конкретной дисциплине. Можно считать, что вся база данных находится во второй форме, рисунок 3.4.

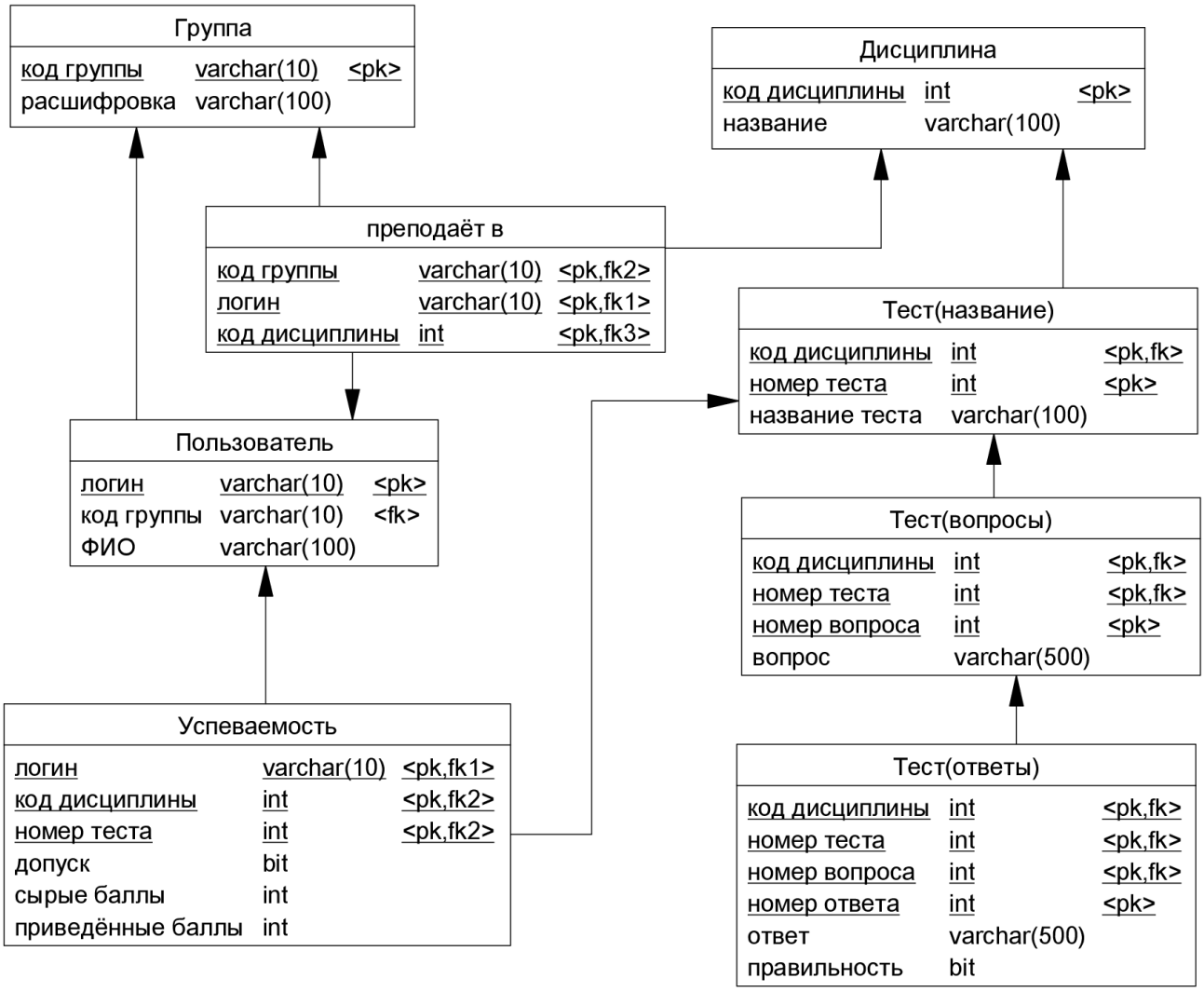

Рисунок 3.4 – 2НФ физической модели проектируемой базы данных

3.1.3 Третья нормальная форма

Переменная отношения R находится в 3НФ тогда и только тогда, когда выполняются следующие условия:

- R находится во второй нормальной форме.

- ни один неключевой атрибут R не находится в транзитивной функциональной зависимости от потенциального ключа R.[10]

Следует пояснить, что такое транзитивная функциональная зависимость. Когда некоторый неключевой атрибут функционально зависит от другого неключевого атрибута, а тот в свою очередь уже от потенциального ключа, такую зависимость называют транзитивной.

В нашей базе данных есть только один атрибут такого типа, это приведённые баллы, ведь приведённые баллы можно легко высчитать, зная сырые баллы и

количество вопросов в тесте. Следовательно, необходимо избавится от этого атрибута. Для этого введём некоторые поправки, считаем, что тест состоит из некоторого множества вопросов, и из этого множества студенту случайным образом отбирается некоторое ограниченное количество вопросов, введём новый атрибут в отношение Тест (название) – количество вопросов.

Так как список вопросов формируется индивидуальный для каждого студента, введём новое отношение Текущие вопросы. Отношение содержит некоторые тестовые опросы для каждого студента и по каждому тесту, проведём связи с отношениями Пользователь и Тест (вопросы), чтобы гарантировать, что каждый вопрос реально существует в базе. Получаем в отношении два внешних ключа, один из которых состоит из трёх атрибутов. Так как база скрыта для пользователей, и программный модуль не сможет определить количество ответов в вопросе и порядок вопроса, введём два новых атрибута порядковый номер вопроса (integer) и несколько ответов (bit). Атрибуты логин, код дисциплины, номер теста и порядковый номер вопроса определяют составной первичный ключ. В отношении также имеется альтернативный ключ состоящий из внешних ключей.

Кроме того, что вопросы перемешиваются, должны перемешиваться и ответы. Введём новое отношение Текущие ответы, связанное с отношением Текущие вопросы, в атрибут добавим 3 новых атрибута, порядковый номер ответа (integer), номер ответа (integer) и правильность (bit). Первичный ключ определяют внешние ключи и порядковый номер ответа, альтернативный ключ определяют также внешние ключи и номер ответа.

Результат преобразований представлен на рисунке 3.5.

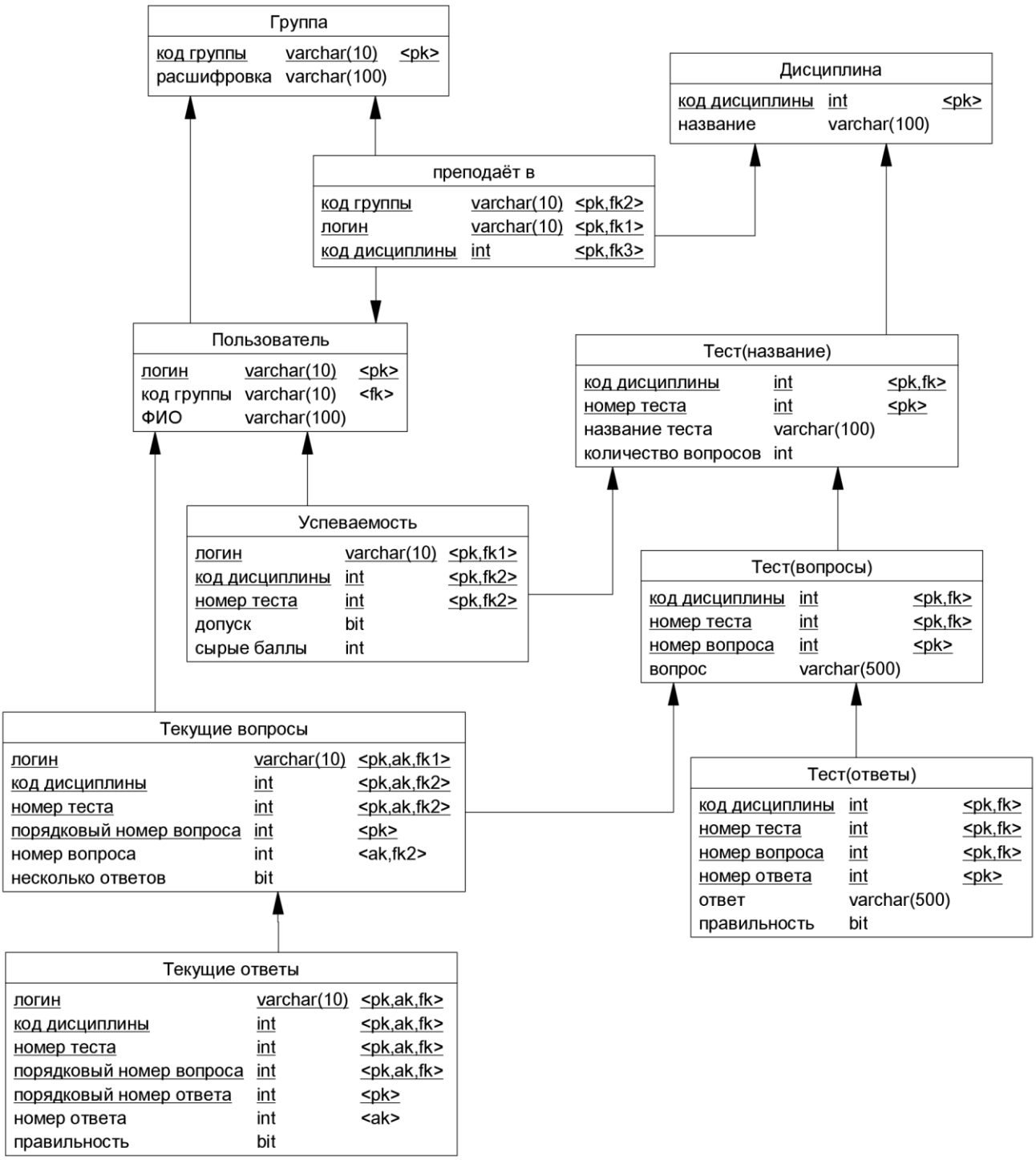

Рисунок 3.5 – 4НФ физической модели проектируемой базы данных

3.1.4 Нормальная форма Бойса-Кодда

Переменная отношения находится в НФБК тогда и только тогда, когда каждая её нетривиальная и неприводимая слева функциональная зависимость имеет в качестве своего детерминанта некоторый потенциальный ключ. Проще говоря, отношение находится в нормальной форме Бойса-Кодда, когда каждый детерминант

является возможным ключом. Детерминант - любой атрибут, от которого полностью функционально зависит некоторый другой атрибут.[11]

Ни одно отношение в базе не имеет ни два, ни более, составных потенциальных ключей, поэтому вся база данных уже соответствует нормальной форме Бойса-Кодда.

3.1.5 Четвёртая нормальная форма

Переменная отношения R находится в четвёртой нормальной форме, если она находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от её потенциальных ключей.[11]

Разберём новый термин многозначная зависимость. Фактически функциональная зависимость является частным проявлением многозначной зависимости. Один атрибут многозначно зависит от другого, тогда и только тогда когда множество значений первого атрибута, соответствует заданной паре полей второго и некоторого третьего атрибутов, зависит от второго атрибута, но не зависит от третьего.

Такая многозначная зависимость часто встречается в тройных связях, поэтому первым делом проверяем отношение (таблицу-связку) «преподаёт в». Предположим, что атрибут код группы зависит от логина, тогда при постоянном логине и коде дисциплины, мы получим некоторое множество групп, в которых преподаёт конкретный преподаватель конкретную дисциплину. Но при изменении любого атрибута множество групп меняется, меняем преподавателя, меняется весь список групп, меняем предмет, и также меняется группа. Что свидетельствует о зависимости обеих атрибутов от кода группы, и значит, зависимость не является многозначной, таким же образом можно проверить две оставшиеся пары логин – код дисциплины и код группы – код дисциплины, но результат будет таким же, всё потому что отношение нельзя разбить ровно на два отношения. Есть учебный план, закреплённый за каждой конкретной группой, т.е. каждой группе соответствует некоторое количество дисциплин, и каждую дисциплину может проходить несколько групп, связь «многие ко многим». Каждому пункту этого учебного плана

может соответствовать несколько преподавателей, но ограничимся одним преподавателем. База данных находится в четвёртой нормальной форме.

Пятую форму выводить не имеет смысла, так как это уже понизит быстродействие. На этом нормализация базы данных завершена, рисунок 3.5.

3.1.6 Роли пользователей

Теперь необходимо проверить, как будут взаимодействовать с базой пользователи разных ролей.

Проверяется роль пользователя, если роль - студент, считаем, что атрибут код группы отношения Пользователь содержит номер группы, в которой учится этот студент, далее определяем список всех дисциплин для данной группы и список всех тестов, к которым имеет доступ студент. Студент выбирает номер контрольной точки, которой соответствует конкретный номер теста, но ведь работа включала в себя возможность создания индивидуальных траекторий обучения, а значит и номер контрольной точки должен содержать свой собственный номер теста, для каждого студента. Исправим и допишем в отношение Успеваемость дополнительное поле номер контрольной точки, и заменим первичный ключ. Далее после выбора теста, идёт проверка, а были ли уже добавлены вопросы из этого теста во временную таблицу, и если нет, то в отношении Текущие вопросы составляется случайный список вопросов, а в Текущие ответы случайный список ответов, и выдаётся студенту. После правильного ответа на вопрос зачисляется один бал в сырые баллы, в конце ответа на вопрос, вопрос удаляется из списка текущих вопросов, поэтому студент не сможет исправить ответ на вопрос. Когда в списке текущих вопросов по пройденному тесту не останется ни одного вопроса, допуск для данного теста снимается, и студент не может заново пройти этот тест, без нового допуска от преподавателя.

Преподаватель выбирает одну из тех дисциплин, которую преподаёт, и имеет возможность добавлять новые тесты, или изменять и дополнять уже существующие. Также преподаватель может составлять график обучения (расставлять тесты на контрольные точки), и проставлять допуски. Ранее мы уже ввели индивидуальный график обучения, за счёт атрибута номер теста в отношении Успеваемость, но

теперь не обходимо организовать стандартный (общий) график, для этого введём ещё одно отношение Тест для группы, связанное с отношением Тест (название) и Группа. В новом отношении составляются графики обучения сразу для целых групп. В отношение необходимо добавить атрибут номер контрольной точки и выделить первичный ключ: код группы, код дисциплины, номер контрольной точки.

В результате проверки были выявлены и исправлены некоторые недочёты, физическая модель базы данных приняла новый вид, представленный на рисунке 3.6.

Пользователи (студент и преподаватель) не имеют доступа для записи и изменения таблиц баз данных, такими правами обладает только администратор, пользователи же взаимодействуют с базой посредством представлений и хранимых процедур. К каждым, из которых разные роли пользователей имеют разный доступ. Создадим в среде PowerDesigner три роли student, sys_admin и teacher. Роль sys_admin получает полный доступ, и все привилегии. Роль student не имеет никаких привилегий, а из таблиц ему открыта только Дисциплина, и то только для чтения, к остальным доступ закрыт. Роль teacher имеет доступ для чтения к таблицам Группа, Дисциплина и Пользователь, к остальным доступ закрыт, привилегий у преподавателя, так же как и у студента нет.

3.1.7 Процедурная часть базы данных

Приложения студента и преподавателя имеет простейший алгоритм, и практически никаких расчётов не выполняет, все, что делает приложение, это отправляет запросы, и получает таблицы, которые затем отображает. Все основные расчёты выполняются в самой базе данных, это сильно её нагружает, но это также крайне необходимо для высокой защиты данных.

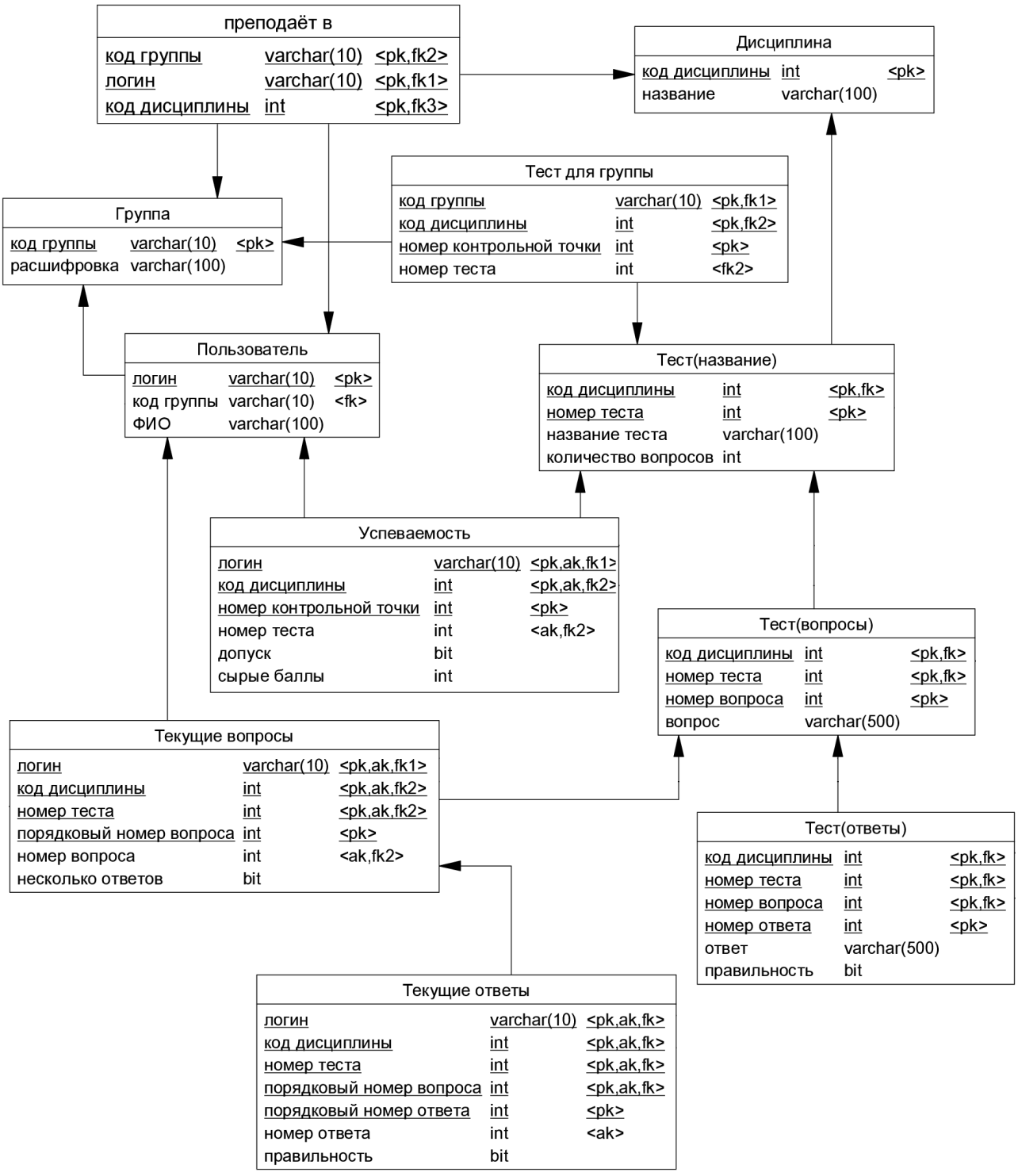

Рисунок 3.6 – Конечный вид физической модели проектируемой базы данных

Преподавателю не зачем ограничивать полный доступ ко всем таблицам. Таблицы Группа, Дисциплина, Пользователь не содержат конфиденциальных данных, и их может просматривать любой преподаватель. А доступ к Тест для группы, Успеваемость, Преподаёт в, Тест (название), Тест (вопросы), Тест (ответы),

необходимо ограничить через представления, чтобы один преподаватель не мог залезть в данные другого преподавателя. Также преподаватель имеет право сбросить тест студенту, такая процедура необходима для устранения ошибок (например, переключение номера теста в траектории обучения, пока студент решал тест).

Для преподавателя необходимо создать следующие элементы:

- представления (Тест для группы(огранич), Успеваемость(огранич), преподаёт в(огранич), Тест(название)(огранич), Тест(вопросы)(огранич), Тест(ответы)(огранич));
- процедура Очистка текущих вопросов;
- триггеры INSERT и DELETE для Тест(название)(огранич), для Тест(вопросы) (огранич) и для Тест(ответы) (огранич) и триггер UPDATE для таблицы [преподаёт в].

Представление – виртуальная таблица, образованная одним оператором SELECT. Через представление можно обновлять данные (UPDATE) непроизводных атрибутов. Если представление составлено из двух и более таблиц, то к нему не применимы операции вставки (INSERT) и удаления (DELETE).[11]

Представление [Тест для группы(огранич)] объединяет 2 таблицы [Тест для группы] и [преподаёт в] по кодам групп и кодам дисциплин с помощью оператора внутреннего соединения INNER JOIN. Атрибут логин таблицы [преподаёт в] должен быть равен текущему пользователю (логину преподавателя зашедшего на сервер), Изменять в этом представлении можно только атрибут номер теста. Представление доступно только для преподавателя (выборка и обновление).

```
CREATE VIEW [Тест для группы(огранич)] AS
SELECT      [Тест для группы].[код группы],
            [Тест для группы].[код дисциплины],
            [Тест для группы].[номер контрольной точки],
            [Тест для группы].[номер теста]
FROM        [Тест для группы]
INNER JOIN [преподаёт в] ON [преподаёт в].логин = CURRENT_USER
```

AND [Тест для группы].[код группы] = [преподаёт в].[код группы]
AND [Тест для группы].[код дисциплины] = [преподаёт в].[код дисциплины]

Представление [Успеваемость(огранич)] связывает 3 таблицы [Пользователь] и [преподаёт в] по коду группы, [Успеваемость] с [Пользователь] по логину и [Успеваемость] с [преподаёт в] по коду дисциплины с помощью оператора INNER JOIN. Изменять можно атрибуты номер теста и допуск.

CREATE VIEW [Успеваемость(огранич)] AS
SELECT s.логин, s.[код дисциплины],
s.[номер контрольной точки], s.[номер теста], s.допуск, ROUND(CAST(s.[сырые баллы] AS float)/(SELECT [количество вопросов] FROM [Тест(название)] AS t WHERE t.[код дисциплины] = s.[код дисциплины] AND t.[номер теста] = s.[номер теста])*100, 0) AS [приведённые баллы]
FROM Пользователь AS u INNER JOIN
[преподаёт в] AS p
ON p.логин = CURRENT_USER AND u.[код группы] = p.[код группы] INNER JOIN Успеваемость AS s ON u.логин = s.логин
AND p.[код дисциплины] = s.[код дисциплины]

Каждое из представлений [Тест(название)(огранич)], [Тест(вопросы)(огранич)], [Тест(ответы)(огранич)] через код дисциплины связывает по 2 таблицы [преподаёт в] и [Тест(…)].

CREATE VIEW [Тест(название)(огранич)] AS
SELECT [Тест(название)].[код дисциплины], [Тест(название)].[номер теста],
[Тест(название)].[название теста],
[Тест(название)].[количество вопросов]
FROM [преподаёт в] INNER JOIN [Тест(название)]
ON [преподаёт в].логин = CURRENT_USER

```
AND [преподаёт в].[код дисциплины] = [Тест(название)].[код дисциплины]
```

Процедура очистки текущих вопросов в соответствии с параметрами: код дисциплины, номер теста и логин студента, удаляет все записи в таблице [Текущие вопросы] и [Текущие ответы].

```
CREATE PROCEDURE ClearQuestions
    @login varchar(10), @subject_code int, @test_number int
AS
BEGIN
SET NOCOUNT ON;
DELETE    [Текущие вопросы]
FROM      [преподаёт в] INNER JOIN
Пользователь ON [преподаёт в].логин = CURRENT_USER AND
[преподаёт в].[код группы] = Пользователь.[код группы] INNER JOIN
[Текущие вопросы] ON Пользователь.логин = @login AND
Пользователь.логин = [Текущие вопросы].логин AND
[преподаёт в].[код дисциплины] = @subject_code AND
[преподаёт в].[код дисциплины] = [Текущие вопросы].[код дисциплины] AND
[Текущие вопросы].[номер теста] = @test_number
END
```

Так как представления [Тест(название)(огранич)], [Тест(вопросы)(огранич)], [Тест(ответы)(огранич)] состоят из 2 таблиц, нельзя сделать вставку строки или удаление. Для этого необходимо ввести триггеры.

Триггер – это хранимая процедура, выполняющаяся при определённых изменениях в базе данных (вставка строки, удаление строки, обновление строки).

```
CREATE TRIGGER DELETE_TEST ON [Тест(название)(огранич)]
INSTEAD OF DELETE
AS
```

```
IF EXISTS(SELECT * FROM [Тест(название)] JOIN deleted
ON [Тест(название)].[код дисциплины] = deleted.[код дисциплины]
AND [Тест(название)].[номер теста] = deleted.[номер теста])
BEGIN
SET NOCOUNT ON
DELETE test
FROM [Тест(название)] AS test, deleted AS d
WHERE test.[код дисциплины] = d.[код дисциплины]
AND test.[номер теста] = d.[номер теста]
END

CREATE TRIGGER INSERT_TEST ON [Тест(название)(огранич)]
INSTEAD OF INSERT
AS
IF EXISTS(SELECT * FROM  [преподаёт в] INNER JOIN inserted AS i
                    ON [преподаёт в].логин = CURRENT_USER
                    AND [преподаёт в].[код дисциплины] = i.[код дисциплины])
BEGIN
SET NOCOUNT ON
INSERT INTO [Тест(название)] ([код дисциплины],[номер теста],
                    [название теста], [количество вопросов])
SELECT i.[код дисциплины], i.[номер теста], i.[название теста],
              i.[количество вопросов]
FROM  [преподаёт в] INNER JOIN inserted AS i
                    ON [преподаёт в].логин = CURRENT_USER
                    AND [преподаёт в].[код дисциплины] = i.[код дисциплины]
END
ELSE ROLLBACK TRANSACTION
```

В базе данных существует атрибут номер контрольной точки, но неизвестно, сколько всего существует контрольных точек для заданной единицы учебного плана. По этой причине в таблицах [Успеваемость] и [Тест для группы] не определено начальное количество записей для каждого студента (группы) по каждой дисциплине. Преподаватель может менять траекторию обучения, а значит и количество контрольных точек, но если дать преподавателю доступ для вставки, и удаления в таблицы [Успеваемость] и [Тест для группы], могут возникнуть нарушения в последовательности контрольных точек. Для этого в таблицу [преподаёт в] добавляем атрибут количество контрольных точек, перед этим изменим первичный ключ на код группы, код дисциплины. Как уже писалось выше, для одного элемента учебного плана в рамках этой бакалаврской работы не требуется несколько преподавателей, а также нельзя определить количество контрольных точек для разных преподавателей, этот новый атрибут относится только к учебному плану.

Такого рода проблему можно решить добавлением триггера на обновление, и задать автоматическое добавление и удаление записей в таблицах [Успеваемость] и [Тест для группы]. После изменение количества контрольных точек, срабатывает триггер, обрезает лишние контрольные точки и добавляет недостающие, при этом не затрагивая номера уже существующих.

```
CREATE TRIGGER UPDATE_PLAN ON [преподаёт в]
AFTER UPDATE
AS
BEGIN
SET NOCOUNT ON
DELETE t FROM [Тест по умолчанию] AS t, inserted AS i
WHERE t.[код группы] = i.[код группы]
        AND t.[код дисциплины] = i.[код дисциплины]
        AND (t.[номер контрольной точки] > i.[количество контрольных точек]
                                        OR  t.[номер контрольной точки] < 1)
```

```
DELETE s FROM [Успеваемость] AS s INNER JOIN
Пользователь AS u ON s.логин = u.логин, inserted AS i
WHERE u.[код группы] = i.[код группы]
AND s.[код дисциплины] = i.[код дисциплины]
AND (s.[номер контрольной точки] > i.[количество контрольных точек]
OR  s.[номер контрольной точки] < 1)
DECLARE @count int = 1
DECLARE @point_number int = (SELECT [количество контрольных точек]
                                        FROM inserted)
DECLARE @group_code varchar(10) = (SELECT [код группы] FROM inserted)
DECLARE @subject_code int = (SELECT [код дисциплины] FROM inserted)
WHILE @count <= @point_number
BEGIN
IF NOT EXISTS(SELECT * FROM [Тест по умолчанию] AS t
            WHERE t.[код группы] = @group_code
            AND t.[код дисциплины] = @subject_code
            AND t.[номер контрольной точки] = @count)
INSERT INTO [Тест по умолчанию] ([код группы], [код дисциплины],
                    [номер контрольной точки])
      VALUES(@group_code, @subject_code, @count)
INSERT INTO Успеваемость (логин, [код дисциплины], [номер контрольной точки])
SELECT u2.логин, @subject_code, @count FROM Пользователь AS u1 INNER JOIN
Успеваемость AS s ON u1.[код группы] = @group_code AND u1.логин = s.логин
AND s.[номер контрольной точки] = @count
AND s.[код дисциплины] = @subject_code RIGHT OUTER JOIN Пользователь AS u2
ON s.логин = u2.логин WHERE u2.[код группы] = @group_code
AND u1.логин IS NULL
SET @count = @count + 1
END
END
```

Представление [преподаёт в(огранич)] составлено из одной таблицы [преподаёт в] и ограничивается по логину, который должен быть равен текущему.

```
SELECT
    [код группы],
    [код дисциплины],
    [количество контрольных точек]
FROM [преподаёт в]
WHERE логин = CURRENT_USER
```

Для студента необходимо создать следующие элементы:

- представления (Оценки группы, Текущий тест(вопросы), Информация о студенте, Текущий тест(ответы), Список доступных тестов). Все представления студента кроме Текущий тест(ответы) только для чтения.
- процедуры начала тестирования и завершения тестирования.

Представление [Оценки группы] объединяет 3 таблицы [Пользователь], [Успеваемость] и [Тест(название)].

```
CREATE VIEW [Оценки группы] AS
SELECT Пользователь.ФИО, Успеваемость.[код дисциплины], Успеваемость.[номер контрольной точки], Успеваемость.допуск, ROUND(CAST(Успеваемость.[сырые баллы]
AS float) / [Тест(название)].[количество вопросов] * 100, 0)
        AS [приведённые баллы]
FROM    Пользователь INNER JOIN Пользователь AS Пользователь_1
ON Пользователь_1.логин = CURRENT_USER AND
Пользователь.[код группы] = Пользователь_1.[код группы] INNER JOIN
Успеваемость ON Пользователь.логин = Успеваемость.логин INNER JOIN
[Тест(название)]
ON Успеваемость.[код дисциплины] = [Тест(название)].[код дисциплины]
```

AND Успеваемость.[номер теста] = [Тест(название)].[номер теста]

Представление [Текущий тест(вопросы)] связывает 2 таблицы [Текущие вопросы], [Тест(вопросы)] по коду дисциплины, по номеру теста, по порядковому номеру с помощью оператора INNER JOIN.

```
CREATE VIEW "Текущий тест(вопросы)" AS
SELECT
  cq.[код дисциплины], cq.[номер теста],
  cq.[порядковый номер вопроса], qq.[вопрос]
FROM [Текущие вопросы] cq
INNER JOIN [Тест (вопросы)] qq on  cq.[логин] = CURRENT_USER
AND cq.[код дисциплины] = qq.[код дисциплины]
AND cq.[номер теста] = qq.[номер теста]
AND cq.[номер вопроса] = qq.[номер вопроса]
```

Представление [Текущий тест(ответы)] связывает 2 таблицы [Текущие ответы], [Тест(ответы)] по коду дисциплины, по номеру теста, по порядковому номеру с помощью оператора INNER JOIN.

```
CREATE VIEW "Текущий тест(ответы)" AS
SELECT
  ca.[код дисциплины], ca.[номер теста], ca.[порядковый номер вопроса],
  ca.[порядковый номер ответа], aa.[ответ], ca.[правильность]
FROM [Текущие ответы] ca
  INNER JOIN [Тест (ответы)] aa
  ON  ca.[логин] = CURRENT_USER
  AND ca.[код дисциплины] = aa.[код дисциплины]
  AND ca.[номер теста] = aa.[номер теста]
  AND ca.[номер вопроса] = aa.[номер вопроса]
  AND ca.[номер ответа] = aa.[номер ответа]
```

Представление [Информация о студенте] связывает 2 таблицы [Пользователь] и [Группа] по коду группы с помощью оператора INNER JOIN.

```
CREATE VIEW "Информация о студенте" AS
SELECT us.ФИО, us.[код группы], gr.расшифровка
FROM Пользователь AS us INNER JOIN Группа AS gr
ON us.логин = CURRENT_USER AND us.[код группы] = gr.[код группы]
```

Представление [Список доступных тестов] связывает 5 таблиц: [Пользователь], [преподаёт в], [Успеваемость], [Тест для группы], [Тест название] с помощью оператора INNER JOIN.

```
CREATE VIEW "Список доступных тестов" AS
SELECT Успеваемость.[код дисциплины],
    ISNULL(Успеваемость.[номер теста], [Тест для группы].[номер теста]) AS
    [номер теста], [Тест(название)].[название теста],
    [Тест(название)].[количество вопросов]
FROM    Пользователь INNER JOIN
        [преподаёт в] ON Пользователь.логин = CURRENT_USER
        AND Пользователь.[код группы] = [преподаёт в].[код группы] INNER
JOIN Успеваемость ON Пользователь.логин = Успеваемость.логин
AND [преподаёт в].[код дисциплины] = Успеваемость.[код дисциплины]
AND Успеваемость.допуск = 1 INNER JOIN [Тест для группы]
ON [преподаёт в].[код группы] = [Тест для группы].[код группы]
AND [преподаёт в].[код дисциплины] = [Тест для группы].[код дисциплины] AND
Успеваемость.[номер контрольной точки] =
    [Тест для группы].[номер контрольной точки]
INNER JOIN [Тест(название)]
ON Успеваемость.[код дисциплины] = [Тест(название)].[код дисциплины]
```

```
AND [Тест(название)].[номер теста] = ISNULL(Успеваемость.[номер теста],
    [Тест для группы].[номер теста])
```

Процедура загрузки вопросов в соответствии с параметрами: код дисциплины, номер теста, перемешивает и добавляет выборку новых вопросов в [Текущие вопросы], и также перемешивает ответы и добавляет их в [Текущие ответы].

```
CREATE PROCEDURE ChoiceTest
    @subject_code int, @test_number int
AS
IF EXISTS(SELECT * FROM Успеваемость WHERE логин = CURRENT_USER
AND [код дисциплины] = @subject_code AND [номер теста] = @test_number
AND допуск = 1) AND NOT EXISTS(SELECT * FROM [Текущие вопросы]
WHERE логин = CURRENT_USER AND [код дисциплины] = @subject_code
    AND [номер теста] = @test_number)
BEGIN
    SET NOCOUNT ON;
    DECLARE @count_quests int =(SELECT [количество вопросов]
FROM [Тест(название)] WHERE [код дисциплины] = @subject_code
    AND [номер теста] = @test_number)
INSERT INTO [Текущие вопросы] (логин, [код дисциплины], [номер теста],
[порядковый номер вопроса], [номер вопроса], [несколько ответов])
    SELECT TOP (@count_quests) CURRENT_USER, q.[код дисциплины],
q.[номер теста], ROW_NUMBER() OVER(ORDER BY NEWID()),
q.[номер вопроса],CAST((SELECT COUNT(*) FROM [Тест(ответы)] AS a
        WHERE a.[код дисциплины] = @subject_code
        AND a.[номер теста] = @test_number
        AND a.[номер вопроса] = q.[номер вопроса]
        AND a.правильность = 1)/2 AS bit) AS [несколько ответов]
        FROM [Тест(вопросы)] AS q
WHERE q.[код дисциплины] = @subject_code
```

```
AND q.[номер теста] = @test_number
SET @count_quests = (SELECT COUNT(*) FROM [Текущие вопросы]
    WHERE логин = CURRENT_USER AND [код дисциплины] = @subject_code
        AND [номер теста] = @test_number)
WHILE @count_quests > 0
BEGIN
INSERT INTO [Текущие ответы] (логин, [код дисциплины], [номер теста], [порядковый номер вопроса], [порядковый номер ответа], [номер ответа])
SELECT CURRENT_USER, a.[код дисциплины], a.[номер теста],
        q.[порядковый номер вопроса],
        ROW_NUMBER() OVER(ORDER BY NEWID()), a.[номер ответа]
    FROM [Тест(ответы)] AS a INNER JOIN [Текущие вопросы] AS q
        ON a.[код дисциплины] = q.[код дисциплины]
        AND a.[номер теста] = q.[номер теста]
        AND a.[номер вопроса] = q.[номер вопроса]
    WHERE a.[код дисциплины] = @subject_code
        AND a.[номер теста] = @test_number
        AND q.[порядковый номер вопроса] = @count_quests
        AND q.логин = CURRENT_USER
SET @count_quests = @count_quests - 1
 END
END
```

Процедура решения теста сверяет правильность ответов, если хоть один из ответов студента не правильный, бал не засчитывается. Далее делит сырые баллы на количество вопросов в тесте и получает приведённые баллы.

```
CREATE PROCEDURE PassTest
    @subject_code int, @test_number int
AS
IF EXISTS(SELECT * FROM dbo.[Текущие вопросы]
```

```
WHERE логин = CURRENT_USER AND [код дисциплины] = @subject_code
        AND [номер теста] = @test_number)
BEGIN
SET NOCOUNT ON;
DECLARE @count_right int = (SELECT COUNT(*)
        FROM (SELECT COUNT(*) AS count FROM
        [Текущие ответы] AS a INNER JOIN [Текущие вопросы] AS q
        ON a.логин = q.логин AND a.[код дисциплины] = q.[код дисциплины]
        AND a.[номер теста] = q.[номер теста]
        AND a.[порядковый номер вопроса] = q.[порядковый номер вопроса]
        INNER JOIN [Тест(ответы)] AS aa
        ON q.[код дисциплины] = aa.[код дисциплины]
        AND q.[номер теста] = aa.[номер теста]
        AND q.[номер вопроса] = aa.[номер вопроса]
        AND a.[номер ответа] = aa.[номер ответа]
        AND a.правильность = aa.правильность
        WHERE q.логин = CURRENT_USER
        GROUP BY q.[код дисциплины], q.[номер теста], q.[номер вопроса]
        HAVING COUNT(*) = (SELECT COUNT(*) FROM [Тест(ответы)] AS aa2
                WHERE aa2.[код дисциплины] = q.[код дисциплины]
                AND aa2.[номер теста] = q.[номер теста]
                AND aa2.[номер вопроса] = q.[номер вопроса]) ) AS counts)
UPDATE Успеваемость
SET [сырые баллы] = @count_right, допуск = 0
WHERE логин = CURRENT_USER
        AND [код дисциплины] = @subject_code
        AND [номер теста] = @test_number
DELETE FROM [Текущие вопросы]
        WHERE логин = CURRENT_USER
        AND [код дисциплины] = @subject_code
```

AND [номер теста] = @test_number

END

Результат добавления процедур и представлений представлен на рисунке 4.7.

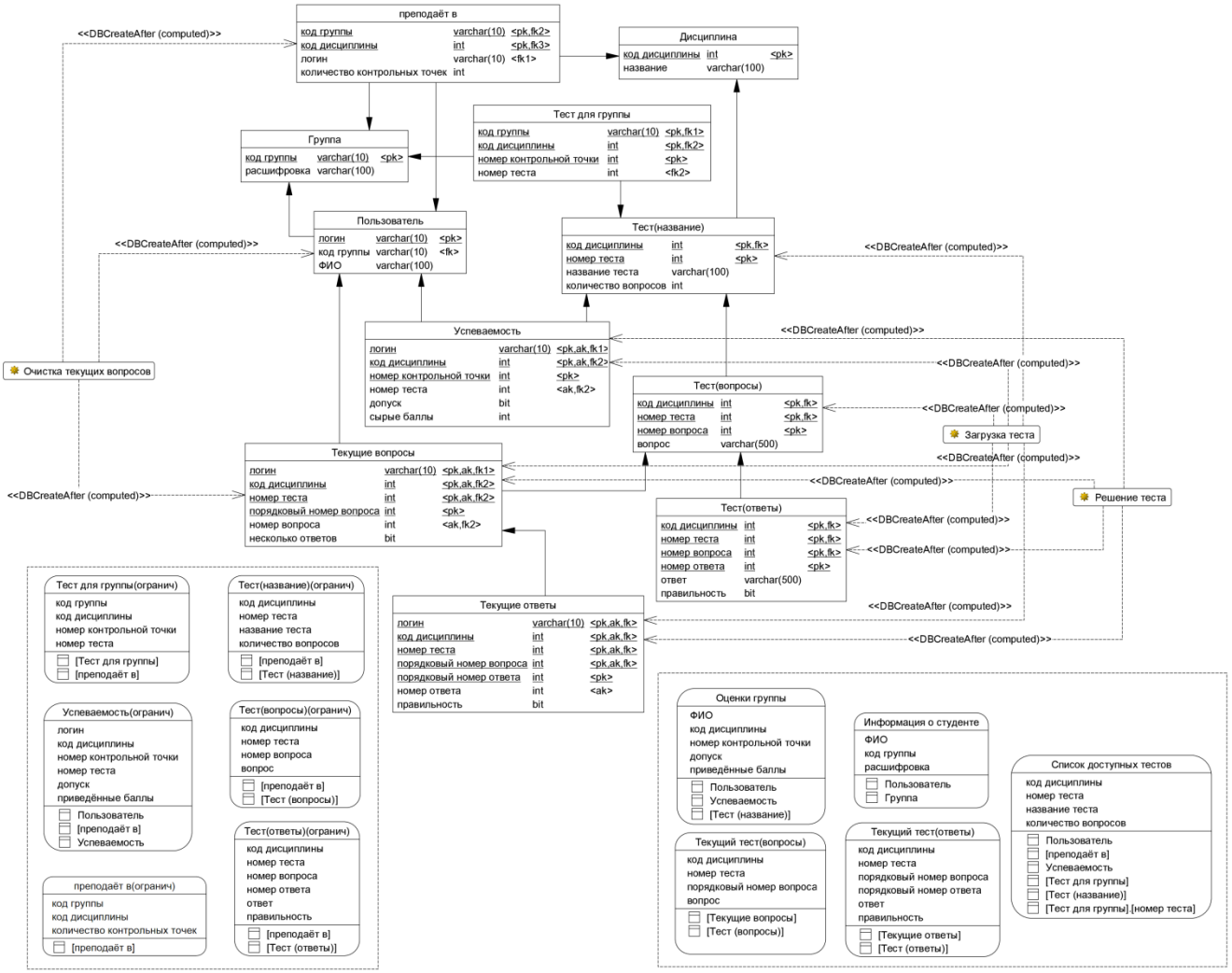

Рисунок 4.7 – Общая схема базы данных

3.2 Проектирование программного модуля для тестирования студентов

Программный модуль написан с использованием библиотеки MFC. Для запуска приложения необходимо объявить производный класс приложения от CWinApp и создать экземпляр этого класса. После чего назначить полю m_pMainWnd (указатель на главное окно) указатель на экземпляр класса окна (CWnd), это может быть как фреймовое окно (CFrameWnd), так и диалоговое (CDialog). Приложение Windows работает так, что не позволяет менять главное окно после запуска программы. Если работать без главного окна, очередь сообщений

напрямую пойдёт к приложению, без обработки сообщений диалоговых окон, в результате чего приложение завершит работу.[12]

Лучший вариант решения этой проблемы - за главное окно поставить пустое диалоговое и скрыть его, а остальные диалоговые окна вызывать как дочерние к главному, тогда можно легко разрушать диалоговые окна и соответственно переключаться между ними.

В классе приложения создадим метод для переключения между окнами (ChangeDialog). Вначале каждому классу диалогового окна присваивается свой код. Метод принимает код нового окна, удаляет предыдущий диалог и по коду создаёт новое диалоговое окно.

Во встроенном редакторе ресурсов в Visual Studio рисуем все необходимые диалоги, всего их 12 и один отдельный пустой диалог для главного окна. Все окна определённым образом связаны друг с другом. Эту связку можно представить в виде дерева диалоговых окон, представленного на рисунке 3.8.

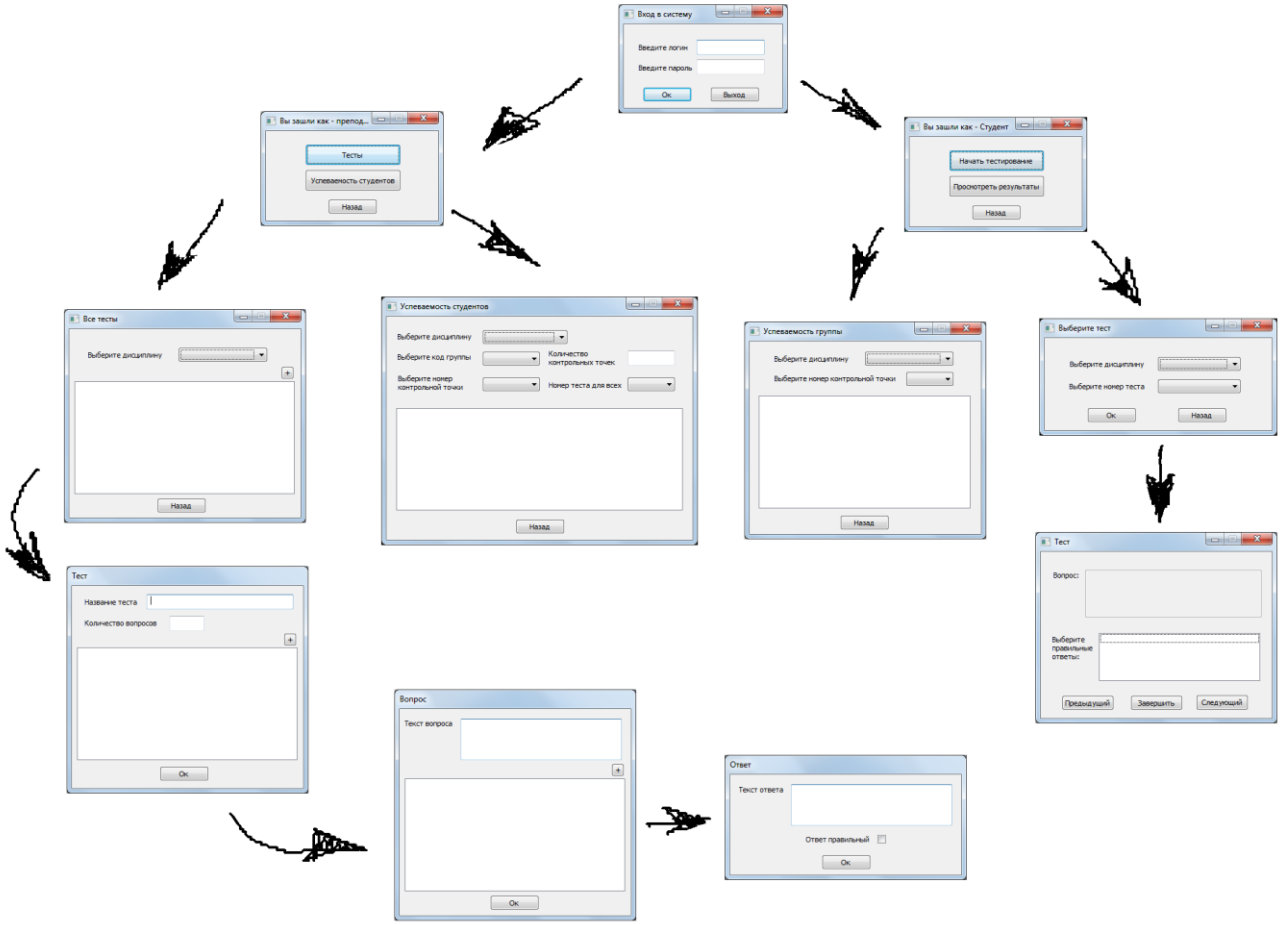

Рисунок 3.8. Структура программного модуля

Рассмотрим структуру программного модуля. При моделировании объектно-ориентированных программ, составляют диаграмму классов. В приложении можно выделить 15 основных классов, из которых 12 – классы диалоговых окон, а другие 3 – класс приложения, класс главного окна и класс подключения к серверу. Диаграмма представлена на рисунке 3.9.

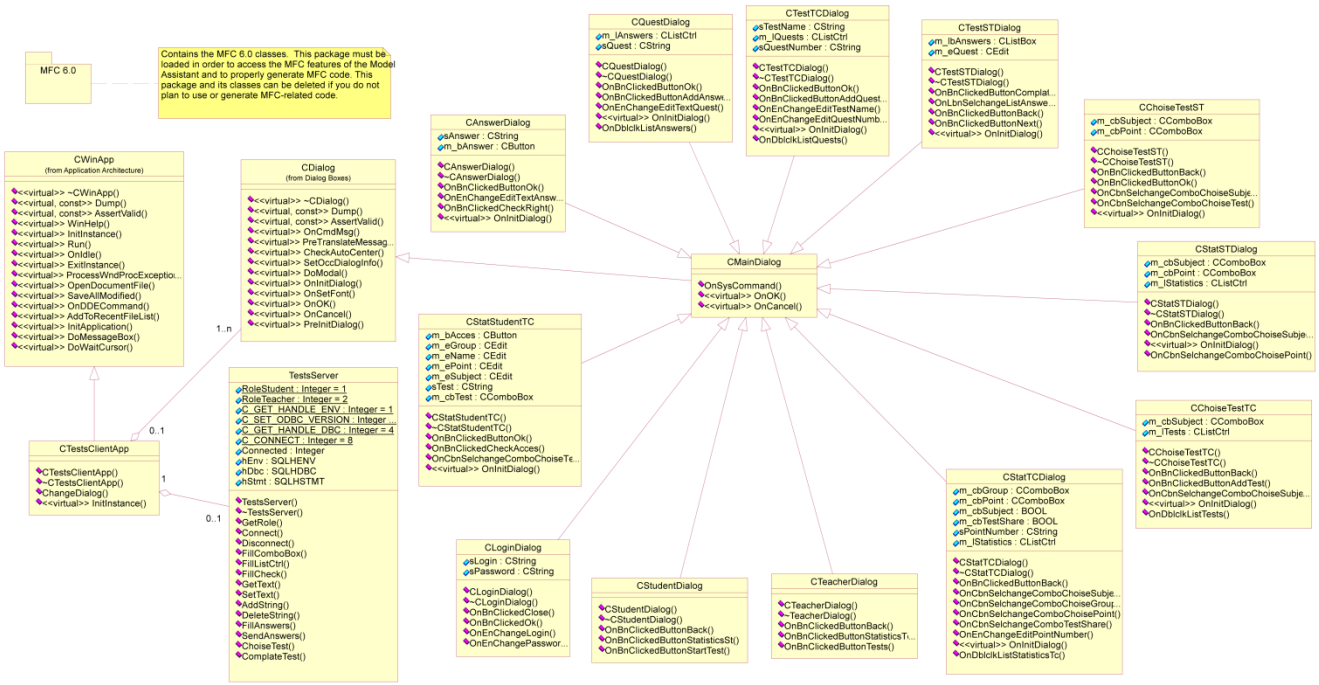

Рисунок 3.9 – Диаграмма классов для программного модуля

Отдельный класс для работы с базой данных (TestsServer) выделен для разделения кода диалоговых окон от кода обращения к серверу баз данных. Класс содержит специальные константы для определения ролей и состояния подключения. А также необходимые дескрипторы для подключения к SQL серверу. Методы класса представляют собой упрощённые классифицированные по типу окон запросы:

1) Методы подключения. Методы Connect и Disconnect – соединение, разъединение с сервером. Метод GetRole определяет и возвращает роль пользователя.
2) Методы заполнения. Метод FillComboBox заполняет выпадающий список данными полученными в результате запроса (параметры запроса указаны в

аргументах). Данные собираются в одну строку, если в результате запроса в результирующем наборе оказалось несколько колонок, они разделяются точкой. Метод FillListCtrl заполняет список в соответствии с запросом, в отличие от выпадающего списка контрольный список может содержать колонки свойств, которые можно использовать для отображения всех полей результирующего набора. Метод FillCheck, устанавливает или сбрасывает флажок, в зависимости от значения поля, на которое направлен запрос. Метод FillAnswers заполняет список - перечень ответов, во время прохождения теста. Все методы заполнения принимают указатель на элемент, который необходимо заполнить.

3) Методы обновления. Метод SendAnswers принимает указатель на элемент, и база данных обновляет данные в соответствии с этим элементом.
4) Методы работы с отдельными ячейками. GetText – прочитать текст одного поля, описанного в запросе. SetText – установить текст в заданное поле.
5) Методы работы со строками. Метод AddString – добавляет заданную строку в таблицу, указанную в запросе. Метод DeleteString – удаляет строку. Методы применимы для представлений тестов.
6) Методы выполнения хранимых процедур. ChoiseTest – выполняет процедуру выбора теста (загрузки вопросов). ComplateTest – выполняет процедуру завершения тестирования.

Disconnect – только отключается от сервера, но не освобождает дескрипторы соединения и среды. Эту обязанность берёт на себя деструктор.

Рассмотрим особенности работы с SQL сервером. Для подключения к серверу через ODBC необходимо сделать следующее:

1) Выделить дескриптор среды;
2) Выделить дескриптор соединения;
3) Подключиться к базе данных.

Для отключения - отключиться от сервера и освободить дескрипторы в обратном порядке.

После отправки запроса, SQL Server возвращает результат запроса в виде результирующего набора. Результирующий набор представляет собой некоторый двумерный массив данных, разбитый на колонки и строки. Прежде чем считать набор необходимо привязать переменные, соответствующего типа, к соответствующим колонкам, и далее вызывая функцию чтения строки поочерёдно переписывать поля каждой строки в привязанные к ним переменные. [13]

3.2.1 Вход в систему

Первое диалоговое окно должно создать подключение к базе данных и проверить роль пользователя. Для подключения необходимо составить строку подключения вида:

DRIVER={SQL Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s,

где SERVER = (ip адрес)\(имя сервера), UID = (логин), PWD = (пароль), DATABASE = (имя базы данных).

После чего подключиться к серверу с помощью функции SQLDriverConnect, если подключение не удалось (функция возвратила SQL_ERROR = -1), можно получить код и текст последней ошибки, используя функцию SQLError.

Для проверки роли на SQL Server посылается запрос:

SELECT IS_MEMBER('student'), IS_MEMBER('teacher').

В случае если пользователь студент - результирующий набор будет {1, 0}. Если преподаватель, то {0, 1}. При комбинации {1, 1} – ошибка доступа, при {0, 0} – неизвестная роль пользователя. После того как результирующий набор прочитан, необходимо очистить дескриптор инструкции, иначе нельзя будет отправить другой запрос на сервер используя тот же дескриптор. Это производится с помощью функции SQLFreeStmt с параметрами SQL_CLOSE, SQL_UNBIND, SQL_RESET_PARAMS. Для каждого параметра вызывается отдельная функция:

SQLFreeStmt(hStmt, SQL_CLOSE);//Сбрасывает курсор

SQLFreeStmt(hStmt, SQL_UNBIND); //Отменяет привязку к столбцам

SQLFreeStmt(hStmt, SQL_RESET_PARAMS);//Сбрасывает параметры инструкции.

3.2.2 Главное меню

После того как вход выполнен перед пользователем появится окно с двумя кнопками: тест или успеваемость. Тест – это прохождение теста или создание тестов (в зависимости от роли), а успеваемость – успеваемость группы или допуск и редактирование траекторий обучения. При выходе или при переходе обратно на окно входа, приложение должно отключиться от сервера. У каждого класса окна есть методы OnOK и OnCancel, которые закрывают диалоговое окно. Если они определены по умолчанию и наследуются от класса CDialogEx, то вызываются при нажатии клавиш ENTER(OnOK) и ESCAPE(OnCancel). Но так как программа имеет другую архитектуру, то при простом закрытии дочернего диалогового окна, программа продолжает выполняться в скрытом главном окне. Для того чтобы избежать ошибок необходимо переопределить методы и сбросить их.

3.2.3 Успеваемость группы

Каждое окно включает в себя другие дочерние окна-элементы. Некоторые, из которых необходимо определить в момент инициализации диалогового окна. В диалоге успеваемость группы есть два выпадающих списка (ComboBox) которые необходимо заполнить данными с базы данных

Сначала список необходимо привязать к переменной Control (это можно сделать средствами Visual Studio через мастер классов), далее при инициализации отправить запрос:

SELECT [код дисциплины], название FROM [Оценки группы] GROUP BY [код дисциплины], название. Прямые запросы на сервер (без предварительной обработки) посылаются с помощью функции SQLExecDirect. Далее необходимо привязать колонки результирующего набора к переменным, делается это через функцию SQLBindCol. И перебором получаем строки данных, вызывая SQLFetch. Каждую строку нужно соединить в одну и добавить в выпадающий список с помощью AddString (метод класса CComboBox).

Таким образом, можно записать метод по добавлению записей из базы данных в выпадающий список. И таким же методом добавляем записи в выпадающую строку номера контрольной точки, со следующим запросом:

SELECT [номер контрольной точки] FROM [Оценки группы]

WHERE [код дисциплины] = (выбранный код дисциплины)

GROUP BY [номер контрольной точки].

Выбранный код дисциплины можно узнать через методы GetLBText и GetCurSel. Также стоит заметить, что после каждого выбора нового текущего элемента в выпадающем списке элементы не обновляются, а просто добавляются, для удаления старых элементов воспользуемся ResetContent.

В диалоговом окне успеваемость группы есть элемент управляющий список, который также необходимо заполнить записями из таблицы [Оценки группы]. В отличие от выпадающего списка, управляющий может содержать колонки. Добавление колонок производится с помощью функции InsertColumns, добавление новой записи InsertItem, запись можно оставить пустой, так как обновляется текст только у первой колонки, для доступа ко всем колонкам, включая первую, воспользуемся функцией SetItemText. Для удаления всех строк нужно вызвать функцию DeleteAllItems, а столбцы можно удалять только по одному, с помощью функции DeleteColumn. Функция удаляет один крайний столбец (параметр 0), для того чтобы узнать количество столбцов воспользуемся методом заголовка элемента управления GetItemCount(), указатель на заголовок можно получить вызвав функцию GetHeaderCtrl().

3.2.4 Выбор теста

Окно выбор теста содержит только два выпадающих списка: выбор дисциплины и выбор номера теста. Выпадающий список выбора дисциплины уже был рассмотрен ранее. С помощью уже описанного выше метода отправляется запрос на сервер:

SELECT [код дисциплины], название

FROM [Список доступных тестов]

GROUP BY [код дисциплины], название

Для номера теста запрос будет выглядеть следующим образом:

SELECT [номер теста]

FROM [Список доступных тестов]

GROUP BY [номер теста]

После нажатия кнопки ОК, выполняется хранимая процедура ChoiseTest, для которой нужны два параметра код дисциплины и номер теста.

Заключение

Программный модуль может применяться не только как средство самоконтроля и самообучения студентов, но и как полноценная система тестирования. Вся информационная система построена с перспективой на будущее развитее. Можно разработать мобильное приложение, подключающееся к той же базе данных.

В отличие от старых систем, в новой системе все тесты можно заносить в базу через программный модуль, не обращаясь к системному администратору. Так же можно сразу их редактировать, пополнять новыми вопросами. Мало того преподаватели могут сами разрабатывать учебный план для своей дисциплины и выстраивать любые траектории обучения.

Вся информационная система отлично подходит для дистанционного обучения студентов. Программный модуль работает как терминал, удобный для доступа к базе данных. Благодаря чему студент никак не может навредить базе данных, и все тесты скрыты от него.

Список используемых источников

1) Lee J. Oracle vs. MySQL vs. SQL Server: A Comparison of Popular RDBMS [Электронный ресурс]: https://blog.udemy.com/oracle-vs-mysql-vs-sql-server/
2) Kurt Mackie Microsoft Shifting Future SQL Server APIs from OLE DB Back to ODBC [Электронный ресурс]: https://mcpmag.com/articles/2011/09/01/microsoft-shifting-to-open-database-connectivity-for-sql-server.aspx
3) Давыдов В.Г. Visual C++. Разработка Windows-приложений с помощью MFC и API функций. – СПб.: БХВ-Петербург, 2008. – 576 с.: ил.
4) Роберт Сигнор, Михаэль О. Стегман Использование ODBC для доступа к базам данных: Пер. с англ. – М.: БИНОМ; НАУЧНАЯ КНИГА. – 384 с.: ил.
5) Ларман, Крэг. Применение UML и шаблонов проектирования. 2-е издание.: Пер. с англ. – М. : Издательский дом «Вильямс», 2004. – 624 с. : ил. – Парал. тит. англ.
6) Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование: Пер. с англ. — М.: ДМК Пресс, 2001. — 176 с.: ил. (Серия «Объектно-ориентированные технологии в программировании»)
7) В. И. Грекул, Г.Н. Денищенко, Н. Л. Коровкина. Проектирование информационных систем: курс лекций: учеб. пособие.– М.: Интернет-Ун-т Информ технологий, 2005. – 304 с.: ил.
8) Анна Нартова PowerDesigner 15 Моделирование данных. – М.: Издательство «Лори», 2012. – 468 с.: ил.
9) Ржеуцкая, С.Ю. Базы данных. Язык SQL: учеб. Пособие. /С.Ю. Ржеуцкая– Вологда: ВоГТУ, 2010. – 159 с.
10) Дейт, К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильяме", 2005. — 1328 с.: ил. — Парал. тит. англ.
11) Ицик Бен-Ган Microsoft SQL Server 2008. Основы T-SQL: Пер. с англ. – СПб.: БХВ-Петербург, 2009. – 432 с.: ил.
12) Сидорина, Т. Л. С34 Самоучитель Microsoft Visual Studio C++ и MFC. / Т.Л. Сидорина— СПб.: БХВ-Петербург, 2009. — 848 с.: ил.
13) SQL Server Native Client (ODBC) [Электронный ресурс]: https://msdn.microsoft.com/en-gb/library/ms131415.aspx
14) Н. Джонсьютис. C++ Стандартная библиотека. Для прфессионалов. – СП Питер, 2004. – 730 с.: ил.
15) Ким, В.С. Тестирование учебных достижений. Монография./В.С. Ким – Усурийск: Издательство УГПИ, 2007. – 214 с.: ил.
16) Object Management Group Unified Modeling Language Version 2.5 Ссылка на нормативный документ: http://www.omg.org/spec/UML/2.5

17) Integration definition for information modeling (IDEF1X) – Стандарт программного обеспечения, Методы моделирования
18) Литвиненко, Н. А. Технология программирования на C++. Win32 API-приложения./Н.А. Литвиненко — СПб.: БХВ-Петербург, 2010. — 288 с.: ил. — (Учебное пособие)
19) Руссинович М., Соломон Д. Внутреннее устройство Microsoft Windows. 6-е изд. — СПб.: Питер, 2013. — 800 с.:ил. — (Серия «Мастер-класс»).
20) Иванов, В.А. Введение в теорию моделирования и параметризации педагогических тестов./ В.А. Иванов – М : Прометей, 2000. – 169 с.