МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационных систем и технологий

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«15» 12 2017 г.

АДМИНИСТРИРОВАНИЕ БАЗ ДАННЫХ

Методические указания к выполнению лабораторных работ для студентов направления 09.03.02 Информационные системы и технологии

Курск 2017

УДК 004.6

Составитель Д.О. Бобынцев

Рецензент: к.т.н., доцент Ватутин Э.И.

Администрирование баз данных: методические указания к выполнению лабораторных работ / Юго-Зап. гос. ун-т; сост.: Д.О. Бобынцев. Курск, 2017. 57 с. Библиогр.: с. 57.

Содержит методические указания к выполнению лабораторных работ по дисциплине «Администрирование баз данных». Указывается порядок выполнения работ, контрольные вопросы. Предназначен для студентов направления подготовки «Информационные системы и технологии».

Текст печатается в авторской редакции

Подписано в печать 15.12.2017. Формат 60х84 1/16.
Усл.печ. л. 3,31. Уч.-изд. л. 3. Тираж 100 экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Содержание

Концептуальное моделирование базы данных

Цель работы: освоить первые этапы проектирования базы данных – анализ предметной области и построение инфологической модели.

Теоретический материал

Процесс конструирования базы данных (ее проектирования и реализации) состоит из последовательности преобразований модели данных одного уровня в модель данных другого уровня.

Последовательности преобразований модели данных:

- систематизация объектов реального мира;
- создание информационных структур, описывающих систему объектов реального мира;
- создание структуры данных, которая используется для представления информационных структур в базе данных и прикладных программах;
- представление структуры памяти, используемой для хранения структур данных.

Процесс проектирования базы данных включает три этапа. Первый этап **-** анализ предметной области или *этап концептуального проектирования.* На этапе концептуального проектирования осуществляется сбор, анализ и редактирование требований к данным**.**

Первым этапом проектирования БД любого типа является анализ предметной области, который заканчивается построением информационной структуры (концептуальной схемы). На данном этапе анализируются запросы пользователей, выбираются информационные объекты и их характеристики, которые предопределяют содержание проектируемой БД. На основе проведенного анализа структурируется предметная область. Анализ предметной области не зависит от программной и технической сред, в которых будет реализовываться БД.

Анализ предметной области целесообразно разбить на три фазы:

1) анализ требований и информационных потребностей;

2) выявление информационных объектов и связей между ними;

3) построение модели предметной области и проектирование схемы БД.

Рассмотрим каждую фазу данного этапа проектирования подробно.

На этапе анализа концептуальных требований и информационных потребностей необходимо выполнить**:**

1) анализ требований пользователей к базе данных (концептуальных требований);

2) выявление имеющихся задач по обработке информации, которая должна быть представлена в базе данных (анализ приложений),

3) выявление перспективных задач (перспективных приложений);

4) документирование результатов анализа

Требования пользователей к разрабатываемой БД представляют собой список запросов с указанием их интенсивности и объемов данных. Эти сведения разработчики БД получают в диалоге с ее будущими пользователями. Здесь же выясняются требования к вводу, обновлению и корректировке информации Требования пользователей уточняются и дополняются при анализе имеющихся и перспективных задач.

Рассмотрим примерный состав вопросника, требований к базе данных при анализе различных предметных областей.

Пример 1. Пусть предлагается разработать систему вопросов к БД «Сессия студентов колледжа»:

1. Сколько студентов учится в колледже?

2. Сколько отделений в данном колледже?

3. Как распределены студенты по отделениям отделений и курсам?

4. Сколько дисциплин читается на каждом курсе по каждой специальности?

5. Как часто обновляется информация в базе данных?

6. Сколько преподавателей?

7. Сколько иногородних студентов живет в общежитии, на частных квартирах?

8. Какая преемственность существует между читаемыми курсами?

9. Сколько лекционных аудиторий и аудиторий для проведения практических занятий, лабораторий?

10. Как информация, представленная в п.п. 1-9, используется в настоящее время (расписание занятий, экзаменов, зачетов и т.д.) и как ее собираются использовать?

11. Сколько раз в день, сколько человек и кто пользуются БД?

Пример 1 (продолжение). Выполним *анализ требований* к БД «Сессия студентов». *Вопрос 1.* Для каких типов задач (приложений) проектируется БД? *Ответ.* Для трех типов задач: Задача 1. Информация о студентах. Задача 2. Информация о преподавателях. Задача 3. Информация об успеваемости студентов. Задача 4. Информация о предметах.

Вопрос 2. Какими информационными объектами характеризуются эти задачи? *Ответ.* Задача 1 характеризуется информационным объектом: *личные дела студентов.* Задача 2 характеризуется информационным объектом: *личные дела преподавателей.* Задача 3 характеризуется одним информационным объектом - *сессия.* Задача 4 характеризуется одним информационным объектом - *предметы.*

Вопрос 3. Каким текущим запросам должны удовлетворять данные информационные объекты?

Вопрос 4. Каким перспективным запросам должны удовлетворять информационные объекты в БД «Сессия студентов»?

Пример 2. Пусть требуется разработать требования к локальной БД «Аэропорт»**.**

Вопрос 1. Для каких типов задач (приложений) проектируется БД? *Ответ.* Для трех типов задач: Задача 1. Информация об обслуживающем персонале. Задача 2. Информация о полетных средствах Задача 3. Информация о графике движения самолетов.

Вопрос 2. Какими информационными объектами характеризуются эти задачи? *Ответ.* Задача 1 характеризуется тремя информационными объектами: *летный состав, диспетчеры, технический персонал.* Задача 2 характеризуется двумя информа-

ционными объектами: *самолет, взлетное поле.* Задача 3 характеризуется одним информационным объектом - *рейсы.*

Вопрос 3. Каким текущим запросам должны удовлетворять данные информационные объекты? *Ответ.*

1. ФИО, звание, должность членов экипажа самолета.
2. Списочный состав диспетчеров.
3. Состав смены технического персонала.
4. Тип самолета, который может обслуживать тот или иной пилот.
5. Номер самолета, который обслуживает данный пилот, данная смена диспетчеров и технического персонала.
6. Номер личного дела сотрудника аэропорта.
7. Номер смены диспетчеров и технического персонала, обслуживающего аэропорт в заданном интервале времени.
8. Готовность самолета с номером № к полету.
9. Количество часов налета самолета с № ...
10. Готовность данной взлетной полосы в настоящее время.
11. Длина данной полосы.
12. Номер (номера) рейса до данного пункта назначения.
13. Какие промежуточные посадки совершает рейс № …?
14. Время вылета и расчетное время прибытия рейса № …
15. Время и место регистрации рейса №
16. Время посадки на рейс № …
17. До какого времени задерживается рейс № …?
18. Какие типы самолетов обслуживают рейс №. …?
19. Какой номер самолета обслуживает рейс *№ …*?

Вопрос 4. Каким перспективным запросам должны удовлетворять информационные объекты в БД «Аэропорт»?

Ответ.

1. С какого года используется самолет с № в аэропорту, тип самолета?
2. Какое количество часов полета у члена экипажа, ФИО?

Расчетное время отпуска члена экипажа, диспетчера, технического работника.

Выявление информационных объектов и связей между ними

Вторая фаза анализа предметной области состоит в выборе информационных объектов, задании необходимых свойств для

каждого объекта, выявлении связей между объектами, определении ограничений, накладываемых на информационные объекты, типы связей между ними, характеристики информационных объектов.

При выборе информационных объектов следует ответить на следующие вопросы:

1. На какие классы можно разбить данные, подлежащие хранению в БД?
2. Какое имя можно присвоить каждому классу данных?
3. Какие наиболее интересные характеристики (с точки зрения пользователя) каждого класса данных можно выделить?
4. Какие имена можно присвоить выбранным наборам характеристик?

В ходе выявления связей между информационными объектами следует ответить на следующие вопросы:

1. Какие типы связей между информационными объектами?
2. Какое имя можно присвоить каждому типу связей?
3. Каковы возможные типы связей, которые могут быть использованы впоследствии?
4. Имеют ли смысл какие-нибудь комбинации типов связей?

Далее проектировщик пытается задать ограничения на объекты и их характеристики. Под ограничением целостности обычно понимают логические ограничения, накладываемые на данные. *Ограничение целостности* - это такое свойство, которое проектировщик задает для некоторого информационного объекта или его характеристики, и которое должно сохраняться для каждого их состояния.

При выявлении условий ограничения целостности проектировщик пытается ответить на следующие вопросы:

1. Какова область значений для числовых характеристик?
2. Каковы функциональные зависимости между характеристиками одного информационного объекта?
3. Какой тип отображения соответствует каждому типу связей?

Пример 1 (продолжение). Для БД «Сессия студентов» выберем следующие сущности: *институт, факультет, студент, преподаватель, дисциплина, ведомость.* Каждую сущность зададим набором атрибутов (ключевые атрибуты подчеркнем):

институт (сокращение, название, подчиненность, адрес, телефон, ФИО ректора).

факультет (код Факультета, название, код специальности, декан). ***кафедры факультета*** (код кафедры, название, код факультета, зав кафедрой). *студент* (номер зачетной книжки, ФИО, группа, пол, дата рождения, домашний адрес, телефон).

преподаватель {№ страхового свидетельства. ФИО, дата рождения, домашний адрес, телефон, должность, ученое звание, ученая степень, код кафедры, стаж).

дисциплина (шифр дисциплины, название, число часов, виды занятий, число читаемых семестров, на каких курсах преподается).

ведомость (№ п/п, номер зачетной книжки студента, код дисциплины, семестр, форма сдачи, дата сдачи, отметка, преподаватель). Определим связи между сущностями.

Имя связи учится изучает принадлежит

Связи между объектами

студент, факультет студент, дисциплина институт, факультет

учится

изучает

принадлежит

работает преподает экзамен

преподаватель, факультет

преподаватель, дисциплина

студент, дисциплина

преподаватель, студент

Рассмотрим некоторые ограничения на характеристики объектов:

1. Значение атрибута "телефон" (сущность - *институт)* задается целым положительным шестизначным числом, задавать значение будем по маске __-__-__ .

2. Значение атрибута "код факультета" (сущность *факультет)* лежит в интервале 0-10.

3. Значение атрибута "курс" (сущность - *студент)* лежит в интервале 1-6 и хранится первая цифра номера группы.

4. Значение атрибута "семестр" (сущность - *студент, дисциплина)* лежит в интервале 1-12.

5. Значение атрибута "число часов" (сущность - *дисциплина)* лежит в интервале 1-300.

6. Одному студенту может быть приписана только одна группа.

7. Один студент может учиться только на одном факультете.

8. Один студент в семестре сдает от 3 до 10 дисциплин.

9. Один студент изучает в семестре от 6 до 12 дисциплин.

10.Одному преподавателю приписывается только одна кафедра.

11.Один студент может пересдавать одну дисциплину не более трех раз.

Ключи: сокращение (названия института), код факультета, номер зачетной книжки, № страхового свидетельства преподавателя, шифр дисциплины, № п/п.

Построение концептуальной (инфологической) модели предметной области

Заключительная фаза анализа предметной области состоит в проектировании ее информационной структуры или концептуальной модели.

Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в, рассматриваемой предметной области (ПО) и выявляемых в результате анализа данных. Концептуальная модель применяется для структурирования предметной области с учетом информационных интересов пользователей системы. Она дает возможность систематизировать информационное содержание предметной области, позволяет как бы "подняться вверх" над ПО и увидеть ее отдельные элементы.

При этом, уровень детализации зависит от выбранной модели. Концептуальная модель является представлением точки зрения пользователя на предметную область и не зависит ни от программного обеспечения СУБД, ни от технических решений. Концептуальная модель должна быть стабильной. Могут меняться прикладные программы, обрабатывающие данные, может меняться организация их физического хранения, концептуальная модель

остается неизменной или увеличивается с целью включения дополнительных данных.

Одной из распространенных моделей концептуальной схемы является модель «сущность-связь». Остановимся на наиболее известной модели данного типа, названной по фамилии автора, - модели П. Чена, или ER-модели. Основными конструкциями данной модели являются сущности и связи.

Под *сущностью* понимают основное содержание объекта ПО, о котором собирают информацию. В качестве сущности могут выступать место, вещь, личность, явление. *Экземпляр* сущности - конкретный объект. Например, сущность (объект) - студент, экземпляр сущности - Иванов А. В.; сущность (объект) - институт, экземпляр сущности - КГУ.

Сущность принято определять *атрибутами* - поименованными характеристиками. Например, сущность - студент, атрибуты - ФИО, год рождения, адрес, номер группы и т.д.

Чтобы задать атрибут в модели, ему надо присвоить имя и определить область допустимых значений. Одно из назначений атрибута - идентифицировать сущность.

Контрольные вопросы

1. Из каких фаз состоит анализ предметной области?
2. Как выбираются информационные объекты?
3. Как выявляются связи между объектами?
4. Что такое ключ?
5. Что включает концептуальная модель?
6. Что понимается под сущностью?
7. Что такое атрибут?
8. Что такое экземпляр сущности?

Логическое моделирование базы данных

Цель логического этапа проектирования - организация данных, выделенных на этапе инфологического проектирования в форму, принятую в выбранной СУБД. Задачей логического этапа проектирования является отображение объектов предметной области в объекты используемой модели данных, чтобы это отображение не противоречило семантике предметной области и было по возможности наилучшим (эффективным, удобным и т.д.). С точки зрения выбранной СУБД задача логического проектирования реляционной базы данных состоит в обоснованном принятии решений о том:

- из каких отношений должна состоять база данных;
- какие атрибуты должны быть у этих отношений;
- какие ограничения должны быть наложены на атрибуты и отношения базы данных, чтобы обеспечить ее целостность.

Требования к выбранному набору отношений и составу их атрибутов должны удовлетворять следующим условиям:

- отношения должны отличаться минимальной избыточностью атрибутов;
- выбранные для отношения первичные ключи должны быть минимальными;
- между атрибутами не должно быть нежелательных функциональных зависимостей;
- выбор отношений и атрибутов должен обеспечивать минимальное дублирование данных;
- не должно быть трудностей при выполнении операций включения, удаления и модификации данных;
- время выполнения запросов на выборку данных (см. описание запросов из варианта задания учебного пособия "Введение в проектирование реляционных баз данных") должно удовлетворять предъявляемым требованиям;
- перестройка набора отношений при введении новых типов должна быть минимальной.

Удовлетворение отмеченных требований обеспечивается аппаратом нормализации отношений. Нормализация отношений - это пошаговый обратимый процесс композиции или декомпозиции исходных отношений в отношения, обладающие лучшими

свойствами при включении, изменении и удалении данных, назначение им ключей по определенным правилам нормализации и выявление всех возможных функциональных зависимостей.

Процесс получения реляционной схемы базы данных из ER-диаграммы включает следующие шаги:

1. Каждая простая сущность превращается в отношение. Простая сущность - сущность, не являющаяся подтипом и не имеющая подтипов. Имя сущности становится именем отношения.

2. Каждый атрибут становится возможным столбцом с тем же именем; может выбираться более точный формат исходя из возможностей СУБД. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы, соответствующие обязательным атрибутам, - не могут.

3. Компоненты уникального идентификатора сущности превращаются в первичный ключ отношения. Если имеется несколько возможных уникальных идентификатора, выбирается наиболее используемый.

4. Связи M:1 (и 1:1) становятся внешними ключами. Для этого делается копия уникального идентификатора с конца связи "один" и соответствующие столбцы составляют внешний ключ. Необязательные связи соответствуют столбцам, допускающим неопределенные значения; обязательные связи - столбцам, не допускающим неопределенные значения.

5. В таблицах, построенных на основе ассоциаций, внешние ключи используются для идентификации участников ассоциации, а в таблицах, построенных на основе характеристик и обозначений, использовать внешние ключи используются для идентификации сущностей, описываемых этими характеристиками и обозначениями. Специфицировать ограничения, связанные с каждым из этих внешних ключей.

6. Если в концептуальной схеме присутствовали подтипы, то возможны два способа:

а) все подтипы размещаются в одной таблице;

б) для каждого подтипа строится отдельная таблица.

При применении способа (а) таблица создается для наиболее внешнего супертипа. В таблицу добавляется по крайней мере один столбец, содержащий код ТИПА, и он становится частью

первичного ключа. Для работы с подтипами могут создаваться представления. При использовании метода (б) супертип воссоздается с помощью конструкции *UNION*.

При обработке данных необходима гарантия сохранения целостности данных в базе, поэтому важным этапом проектирования реляционной базы данных является обеспечение целостности базы данных.

Выделяют три группы правил целостности:

- целостность по сущностям;
- целостность по ссылкам;
- целостность, определяемая пользователем.

Обеспечение целостности базы данных обеспечивается заданием ограничений целостности. Ограничение целостности - это некоторое утверждение, которое может быть истинным или ложным в зависимости от состояния базы данных.

По способам реализации ограничения целостности делятся на:

- декларативные, выполняемые средствами языка SQL;
- процедурные, выполняемые посредством триггеров и хранимых процедур.

При выполнении этой лабораторной работы в процессе построения реляционной модели данных должны быть обеспечены декларативные ограничения целостности. Декларативные ограничения целостности должны обеспечивать:

- задание первичных ключей для обеспечения целостности по сущностям;
- определение необходимых внешних ключей для обеспечения целостности по ссылкам;
- контроль функциональных ограничений на значения атрибутов, определяемых требованиями предметной области;
- задание неопределенных значений и значений по умолчанию;

задание условий каскадного удаления и пр.

Последовательность выполнения лабораторной работы:

1. Составить описание предметной области и построить модель «Сущность-связь».

2. Изучить вопросы теории нормализации, условия нахождения отношения в той или иной нормальной форме

3. Выполнить процедуру построения реляционной модели данных из ER-модели, построив необходимый набор отношений. Определить состав атрибутов отношений.

4. Определить первичные и внешние ключи отношений.

5. Выполнить шаги по нормализации полученных отношений, приведя модель к третьей нормальной форме.

6. Задать необходимые декларативные ограничения целостности исходя из специфики предметной области.

7. Представить связи между первичными и внешними ключами в виде вертикальной диаграммы.

8. Средствами имеющейся СУБД создать спроектированную базу данных, ее таблицы, задать необходимые ограничения целостности.

9. На языке SQL записать выражения для указанных в варианте задания запросов на выборку данных из созданной базы данных. Проверить работоспособность написанных запросов в интерактивном режиме.

Контрольные вопросы

8. Каковы задачи, решаемые на этапе логического проектирования?

9. Каковы базовые свойства реляционной модели данных?

10. В чем состоят требования структурной части реляционной модели данных?

11. В чем состоят требования манипуляционной части реляционной модели данных?

12. В чем состоят требования целостной части реляционной модели данных?

13. Каковы общие свойства нормальных форм?

14. Что такое функциональная, функционально полная зависимость?

15. Каковы условия нахождения отношений в первой нормальной форме?

16. Каковы условия нахождения отношений во второй нормальной форме?

17. Каковы условия нахождения отношений в третьей нормальной форме?

18. Каковы условия нахождения отношений в третьей усиленной нормальной форме?

19. Что понимается под многозначной зависимостью?

20. Каковы условия нахождения отношений в четвертой нормальной форме?

21. Что понимается под понятием "проецирование без потерь"?

22. Каковы условия нахождения отношений в пятой нормальной форме?

23. В чем состоят общие требования обеспечения ограничений целостности?

24. Каковы средства задания ограничений целостности в языке SQL?

Физическое моделирование базы данных

Разберем принципы формирования модели базы данных в трёх типовых СУБД на примере приложения «Деканат». Модель будет создаваться с помощью инструментов моделирования данных в различных оболочках.

Описание задачи. Пусть требуется хранить и управлять информацией о результатах обучения студентов: об учебных группах; студентах, обучающихся в этих группах; дисциплинах, которые изучаются и сдаются в разные семестры; преподавателях, которые ведут эти дисциплины; оценках, которые были получены студентами при сдаче зачетов/экзаменов.

Существует несколько концепций моделей баз данных (иерархическая, сетевая, объектная, реляционная). Наиболее распространенной моделью является реляционная модель, которая очень тесно переплетается с принципами объектно-ориентированного анализа и еще одного популярного подхода в моделировании данных – ER-модели (модель «сущность-связь»).

ER-модель удобна для начального проектирования, поскольку она интуитивно понятна большинству пользователей. В ней выделяются понятия сущности (основные объекты базы), атрибуты (свойства сущности) и связи (взаимодействия между сущностями). В ряде оболочек именно в этих терминах и создан сервис создания модели данных.

Реляционная модель представляет всю базу данных как набор связанных таблиц – отношений. Большинство таблиц отвечает за хранение информации о сущностях (столбцы таблиц характеризуют их атрибуты). Среди атрибутов сущности выделяют ключевые атрибуты – атрибуты, которые являются идентифицирующими, точно определяющими запись, объект сущности. С помощью внедрения ключевых атрибутов одних сущностей (родительские таблицы) в качестве столбцов в другие таблицы (дочерние) реализуются различные связи между сущностями.

Построение модели с помощью оболочки MySQL Workbench

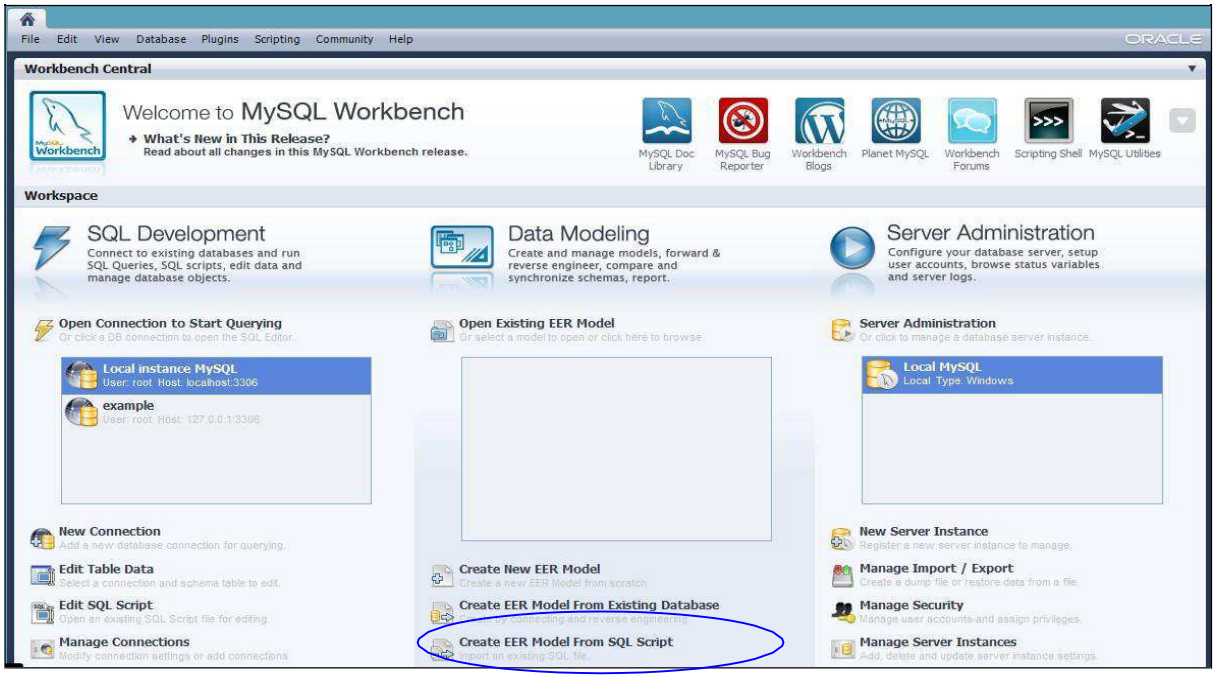


Рис. 1 - Создание модели базы данных в MySQL Workbench

Создаем новую ER-модель и диаграмму в модели. В полученном окне модели представлено полотно, на которое можно наносить новые таблицы, с помощью визуальных средств редактирования, создать столбцы (атрибуты) таблиц и с помощью панели инструментов создать связи между таблицами. При установке связи ключевые поля родительских сущностей добавляются в дочерние таблицы автоматически.

Проведем анализ состава таблиц для решаемой задачи. При описании столбцов таблицы поля, входящие в первичный ключ, будут подчеркнуты.

Имеется таблица **Студенты (Students): (№Зач.книжки, ФИО Студента, №Группы).**

Для хранения групп не будем выделять отдельную таблицу.

Имеется таблица Преподаватель (Teachers): (№Преподавателя, ФИО Преподавателя, Должность, №Кафедры).

Чтобы избежать дублирования информации с названием кафедры введем справочную таблицу кафедр: таблица **Кафедра (Departments): (№Кафедры, Название, Телефон).**

Имеется таблица учебных дисциплин **Дисциплина (Subjects): (№Дисциплины, Название).**

Таблица **Сессия** содержит информацию о том, каков состав зачетов и экзаменов для каждой конкретной группы по семестрам, каким преподавателям следует сдавать зачеты и экзамены: **Sessions (№Группы, №Семестра, №Дисциплины, Отчетность, №Преподавателя).** Заметим, что отчетность может определяться номером дисциплины и номером семестра, но в предположении наличии нескольких специальностей один и тот же предмет может сдаваться в разных семестрах разными группами. Поэтому отчетность и преподаватель зависят и от группы тоже.

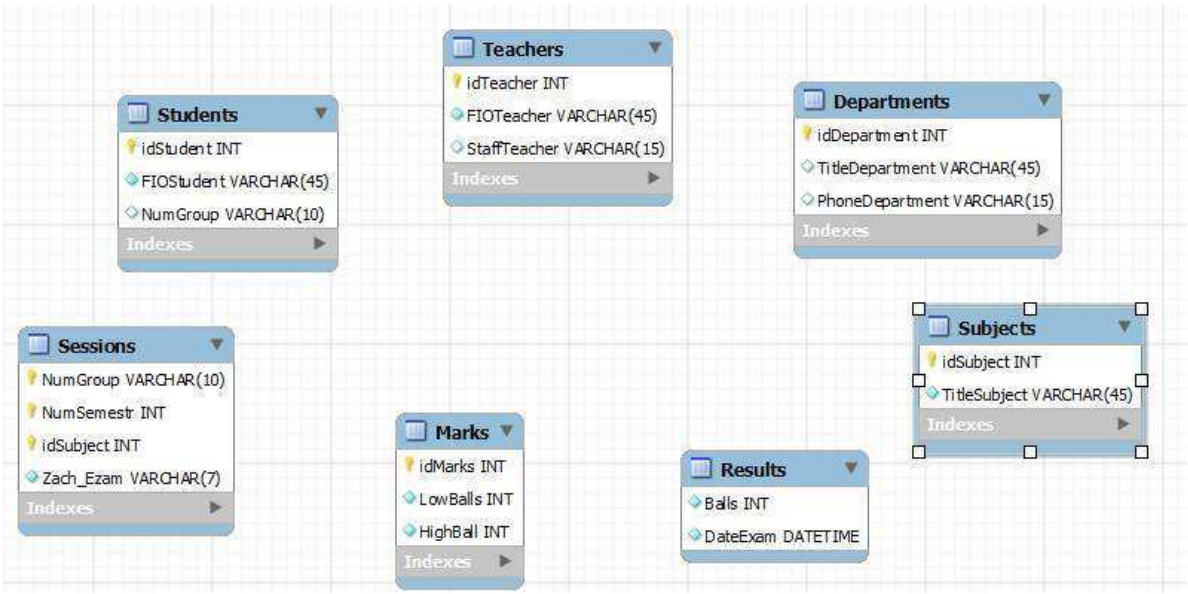
Наконец, результаты сдачи сессии хранятся в таблице результатов **Results (№Студента, №Группы, №Семестра, №Дисциплины, Баллы, ДатаСдачи, Оценка).** Окончательную оценку хранить не требуется, так как она определяется количеством набранных баллов и таблицей оценок.

Marks (Оценка, НижняяГраница, ВерхняяГраница) – эта таблица является справочной и не связана с основными таблицами базы. Ее роль заключается в определении правильной оценки по набранным баллам.

В результате данного анализа задачи получится следующая модель:

- сначала формируется состав таблиц без связующих атрибутов:

Рис. 2 - Модель таблиц базы данных «Деканат» без указания



связей

- затем устанавливаем связи. Заметим, что можно было бы все связующие атрибуты сразу добавить в таблицы. Тогда все связи можно было бы добавить, как связи «один-ко-многим» для существующих столбцов. Отметим также, что связь таблицы результатов и сессии не является очевидной, так как сессия зависит от номера группы, а в таблице результатов указываются оценки конкретных студентов. Поэтому эту связь можно сделать идентифицирующей, а потом удалить из таблицы результатов атрибут номера группы. Другой вариант решения этой проблемы, добавить все поля в таблицу результатов и не устанавливать связь на уровне модели. Далее после создания таблиц в базе данных добавить ограничения внешних ключей для полей номера дисциплины и номера преподавателя.

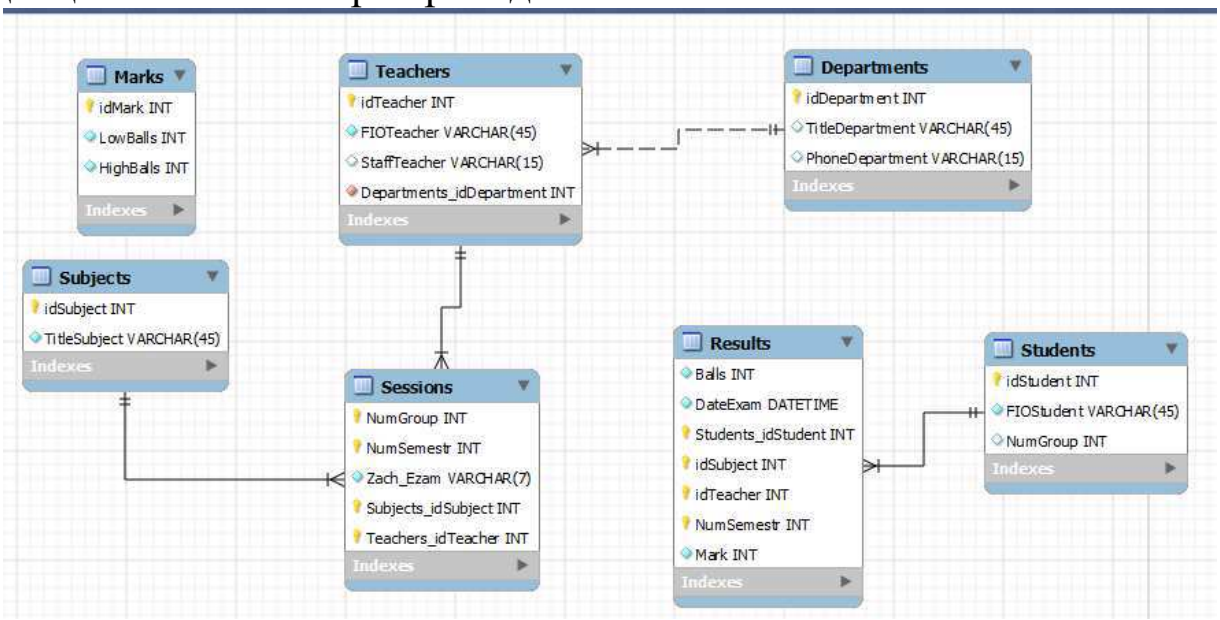


Рис. 3 - Модель таблиц базы данных «Деканат» с указанием связей

Отметим некоторую избыточность таблицы результатов относительно номера группы. Требуется обеспечить, чтобы номер группы и студенты были согласованы по таблицам студентов и результатов сессии.

Построение модели в оболочке dbForge Studio для SQL Server

Новую модель (диаграмму) базы данных можно создать с помощью меню «База данных»-> «Диаграмма БД».

Окно редактирования новой диаграммы состоит из полотна, на которое можно наносить новые таблицы с помощью визуальных средств редактирования, создать столбцы (атрибуты)

таблиц и с помощью панели инструментов создать связи между таблицами.

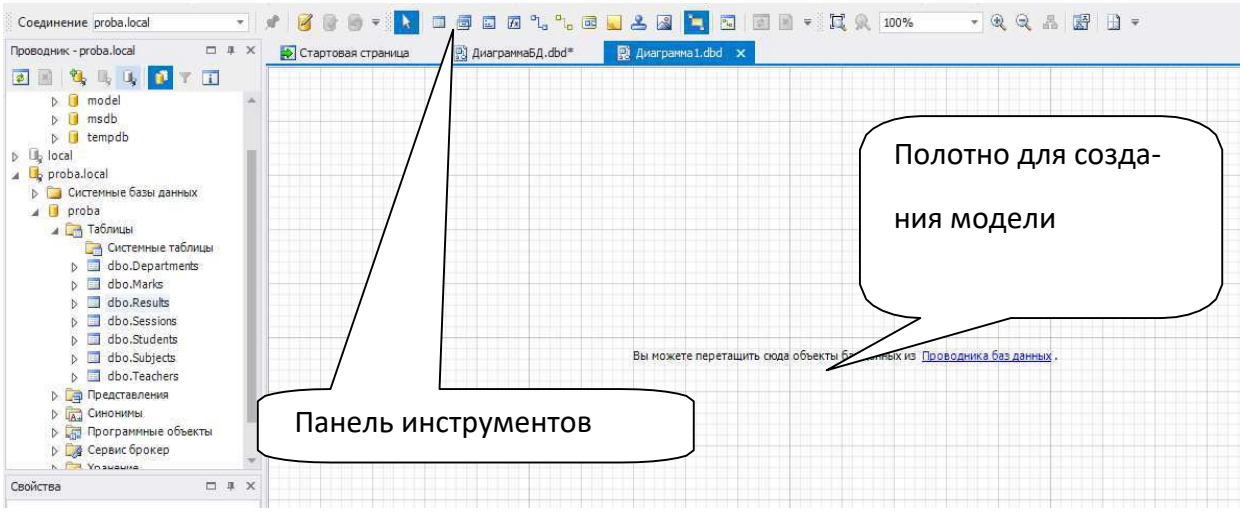


Рис. 4 - Окно редактирования диаграммы базы данных

На панели инструментов следует отметить пока только две кнопки «Новая таблица», «Новая связь», которые позволяют создать новую таблицу, определив ее состав столбцов, первичные ключи и основные ограничения, и создать связи между таблицами, определив тем самым ограничения внешнего ключа.

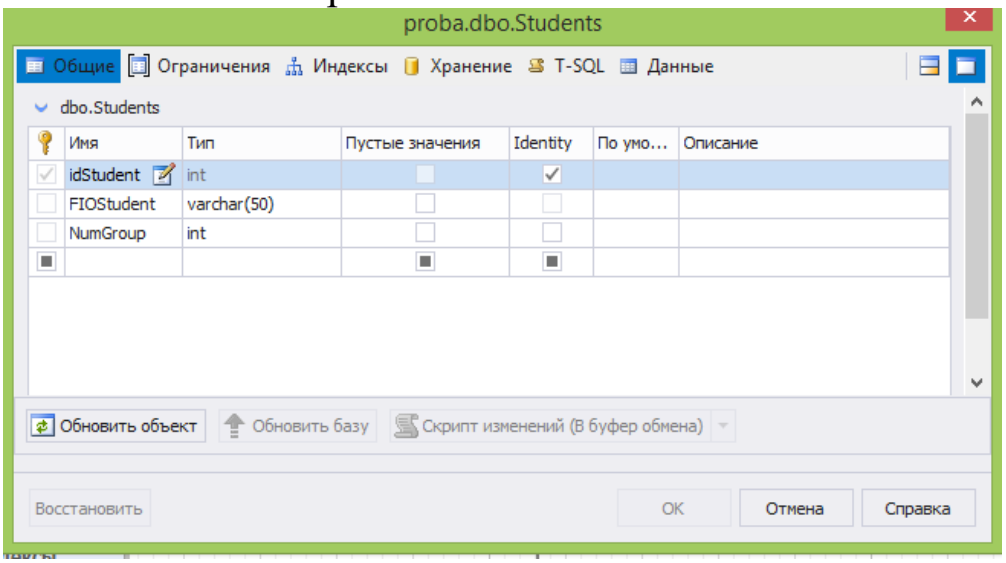


Рис. 5 - Окно создания столбцов таблицы

Для столбцов можно задать простые ограничения: допустимы ли пустые значения и определяет ли столбец поле-счетчик. Кроме того, можно выбрать столбцы, определяющие первичный ключ.

При создании связи требуется «нарисовать» мышью линию от дочерней таблицы к родительской. Для подтверждения параметров связи будет показано окно, в котором нужно уточнить имена полей родительской и дочерней таблиц, которые будут

связаны ограничением внешнего ключа:

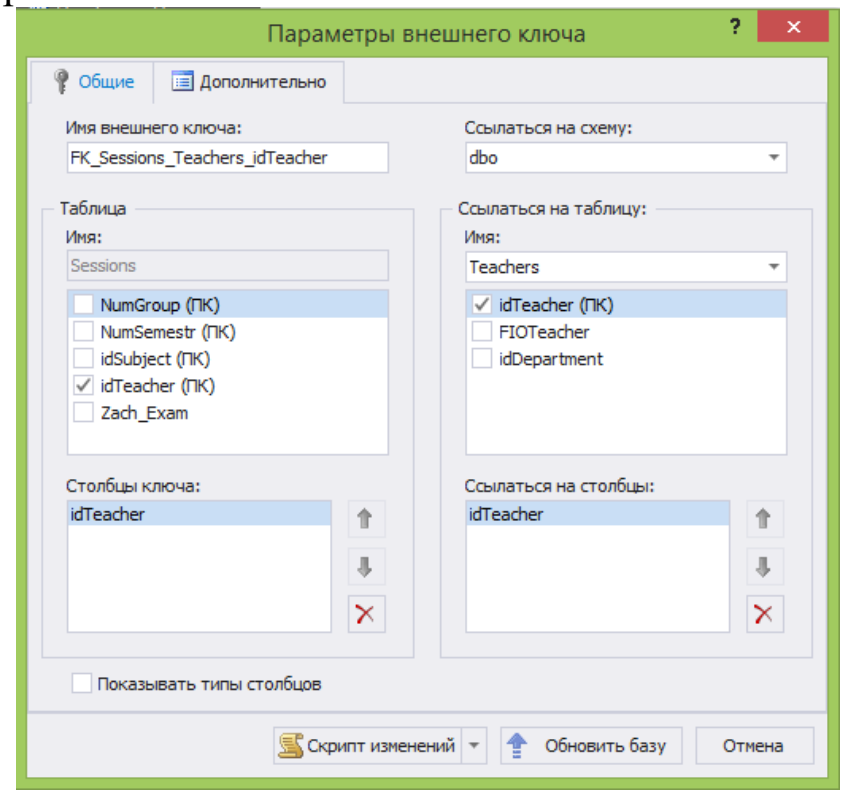


Рис. 6 - Окно задания параметров внешнего ключа

На вкладках «Ограничения» и «Индексы» можно увидеть все ограничения, которые сгенерируются в базе данных применительно к этой таблице. На вкладке T-SQL можно увидеть SQL-команду, выполнение которой эквивалентно выполнению всех сделанных настроек.

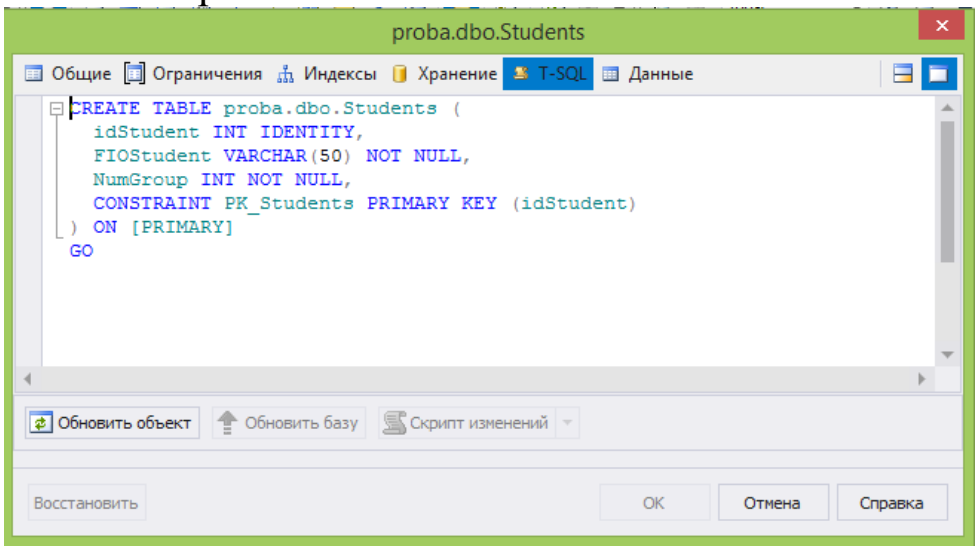


Рис. 7 - Команда SQL создания таблицы «Студенты»

Отметим, что построитель модели синхронизирует все действия пользователя с базой данных, создавая указанные таблицы вместе со всеми ограничениями.

Таким образом, будет получена следующая модель:

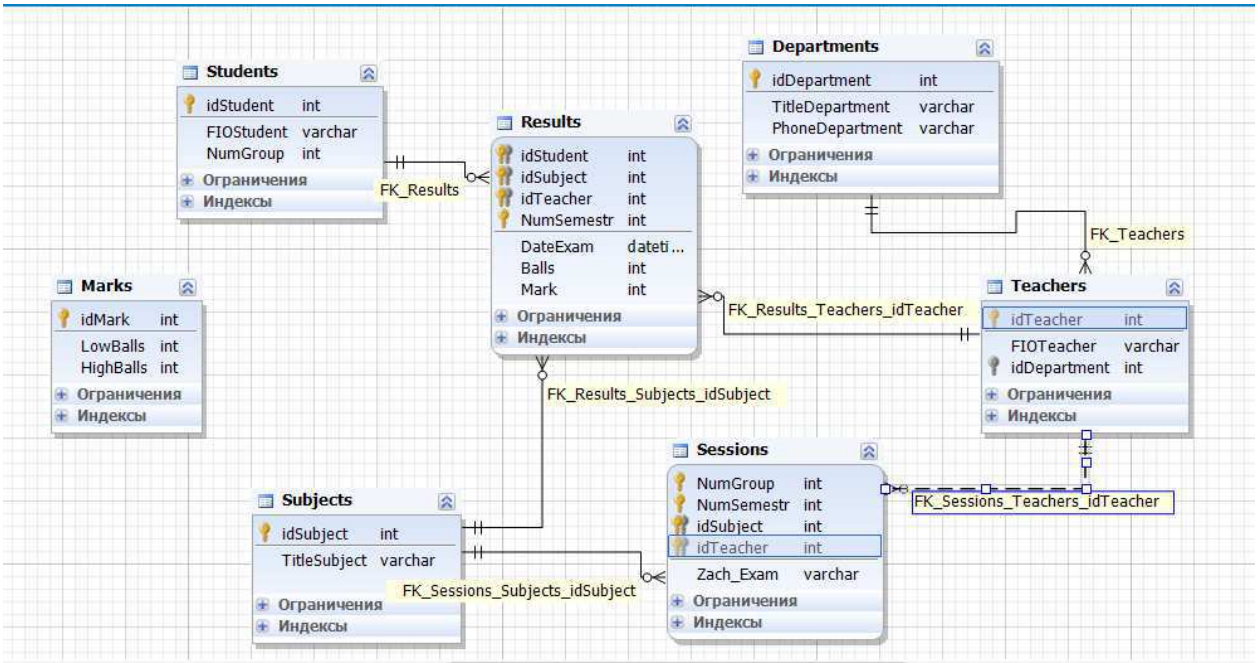


Рис. 8 - Модель данных, построенная с помощью dbForge Studio

Замечания относительно синхронизации номера группы в таблицах «Сессия» и «Студент» остаются на уровне модели нерешенным.

Аналогичным образом создается модель и, соответственно, база данных в среде dbForge Studio для MySQL.

Для PostgreSQL в стандартный набор инструмент формирования модели данных не входит. Поэтому состав таблиц нужно будет создать или с помощью специального SQL-оператора, или с помощью конструкторов таблиц:

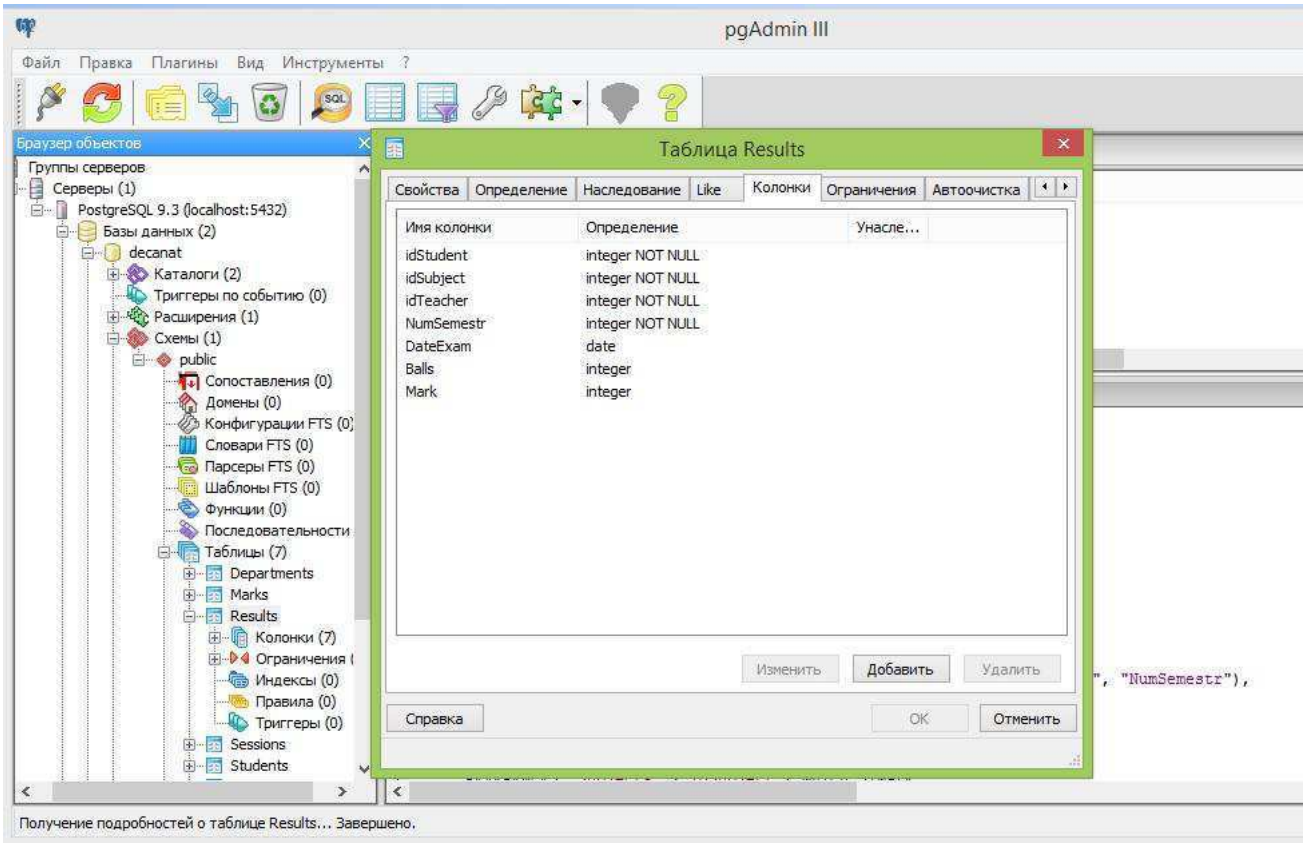


Рис. 9 - Вид окна редактирования таблицы

Действия с базой данных можно производить с помощью контекстного меню соответствующего элемента (таблицы, столбца, ограничения) в дереве объектов сервера. Настройки любого элемента производятся с помощью пункта контекстного меню «Свойства».

Задание: составить реляционную модель из работы 2 в трёх описанных СУБД.

1. Каковы задачи, решаемые на этапе логического проектирования?
2. Каковы базовые свойства реляционной модели данных?
3. В чем состоят требования структурной части реляционной модели данных?
4. В чем состоят требования манипуляционной части реляционной модели данных?
5. В чем состоят требования целостной части реляционной модели данных?

Разработка и управление серверной частью базы данных

Цель работы: изучить основные средства управления базами данных, используемые в языке SQL.

Теоретические положения

Для получения данных из реляционной базы данных предназначен оператор SELECT, который имеет следующий формат:

SELECT [ALL | DISTINCT] (<список полей> | *)
FROM <список таблиц>
[WHERE <предикат условия>]
[GROUP BY <список полей результата>]
[HAVING <предикат условия>]
[ORDER BY <список полей>]

Здесь используются следующие обозначения:

- ALL– результирующая таблица содержит все строки, в том числе повторяющиеся;
- DISTINCT– результирующая таблица содержит только различающиеся строки;
- * – результирующая таблица содержит все столбцы, полученные из части FROM;
- FROM– предложение, в котором указывается исходная таблица или операция декартово произведения или операция соединения таблиц;
- WHERE– предложение, которое содержит предикат выборки;
- GROUPBY– предложение, в котором указываются столбцы, по которым выполняется группирование строк;
- HAVING– предложение, которое содержит предикат для фильтрации групп (а не отдельных строк);
- ORDERBY– предложение, содержащее список столбцов по которым должно выполняться упорядочивание (ASC – по возрастанию, DESC– по убыванию).

Для получения связанной информации из нескольких таблиц используют условные соединения. Предусмотрены следующие виды условных соединений:

- **INNER**– внутреннее соединение. Выбираются пары строк, для которых выполняется условие соединения, заданное предикатом в предложении ON.
- **LEFT**– левое внешнее соединение. В результат включается внутреннее соединение таблиц, к которому добавляются строки из левой таблицы, не вошедшие во внутреннее соединение. Строки из левой таблицы, не вошедшие во внутреннее соединение, дополняются значениями NULLв соответствии со схемой результирующей таблицы.
- **RIGHT**– правое внешнее соединение. В результат включается внутреннее соединение таблиц, к которому добавляются строки из правой таблицы, не вошедшие во внутреннее соединение. Строки из правой таблицы, не вошедшие во внутреннее соединение, дополняются значениями NULLв соответствии со схемой результирующей таблицы.
- **FULL**– полное открытое соединение. В результат включается внутреннее соединение таблиц, к которому добавляются строки из левой таблицы, не вошедшие во внутреннее соединение, и строки из правой таблицы, не вошедшие во внутреннее соединение, которые дополняются значениями NULL в соответствии со схемой результирующей таблицы.

В языке SQL предусмотрены следующие основные агрегатные функции:

- COUNT({[ALL|DISTINCT] <имя атрибута> | *}) – количество строк с непустыми значениями атрибута. Если *, то количество всех строк таблицы, не зависимо от содержания. Для числовых и символьных атрибутов.
- SUM([ALL|DISTINCT] <имя атрибута>) – сумма значений. Для числовых атрибутов.
- AVG([ALL|DISTINCT] <имя атрибута>) – среднее значение. Для числовых атрибутов.
- MIN([ALL|DISTINCT] <имя атрибута>) – минимальное значение. Для числовых и символьных атрибутов.
- MAX([ALL|DISTINCT] <имя атрибута>) – максимальное значение. Для числовых и символьных атрибутов.

Хранимая процедура (Stored Procedure) – это именованный набор команд языка Transact-SQL, хранящийся на сервере в

качестве самостоятельного объекта БД. Синтаксис оператора создания хранимой процедуры следующий:

CREATE PROC[EDURE] <имя процедуры>
[{@<параметр> <тип>} [= <значение по умолчанию>]
[OUTPUT]] [,…n]
AS<SQLоператор> [,…n]

Здесь <имя процедуры> – уникальное имя; @<параметр> – имя формального параметра, которое должно содержать символ @; <SQLоператор> – операторы SQL, составляющие тело процедуры. Формальные параметры могут быть входными, если для них не указывается ключевое слово OUTPUT и входными-выходными, если указывается ключевое словоOUTPUT. Если для входного параметра указывается значение по умолчанию, то при вызове хранимой процедуры значение этого параметра может не указываться.

Для вызова хранимой процедуры необходимо использовать оператор вызова:

EXEC[UTE]
[@<переменная> =] <имя процедуры>
[[@<параметр> =] {<значение> | @<переменная> [OUT[PUT]] | [DEFAULT] }] [,…n]

Если при вызове указывается имена формальных параметров, то порядок следования передаваемых параметров может быть любым. Если же имена формальных параметров не указываются, то порядок следования фактических параметров должен быть такой же, как в объявлении хранимой процедуры. Выходные параметры должны быть помечены ключевым словом OUTPUT. Значение выходного параметра может быть присвоено переменной.

Получать данные из процедуры можно следующими способами:

- Посредством выходных параметров хранимой процедуры, которые помечаются ключевым словом OUTPUT.
- Через набор строк или набор данных. Набор строк возвращается из хранимой процедуры, если в теле процедуры был выполнен SELECT-запрос. Если в теле процедуры более одного запроса, то возвращается набор данных – набор данных.

- В виде кода завершения. Процедура всегда возвращает целочисленное значение. Его можно задать параметром в операторе RETURN в теле процедуры.

Триггер– процедура, связанная с таблицей или представлением, которая автоматически выполняется при выполнении операции вставки, изменения или удаления строки этой таблицы или представления. Триггер создается командой

CREATE TRIGGER <имя триггера>
ON <имя таблицы> | <имя представления>
{FOR | AFTER | INSTEAD OF}
{ [DELETE] [,] [INSERT] [,] [UPDATE]}
AS
<SQL оператор> […n]

Классификация триггеров по типу действия:

- INSERTTRIGGER – запускаются при выполнении команды INSERT;
- UPDATETRIGGER – запускаются при выполнении команды UPDATE;
- DELETETRIGGER– запускаются при выполнении команды DELETE.

Классификация триггеров по типу поведения:

- AFTER – триггер выполняется ***после успешного выполнения*** команды;
- INSTEADOF – триггер вызывается ***вместо выполнения*** команды. Для представлений можно использовать только триггер INSTEADOF.

AFTER триггер выполняется после того, как действие команды было завершено. Поэтому, если необходимо отменить действие команды, то в AFTER триггере надо использовать конструкцию ROLLBACK TRANSACTION. В этой же ситуации в INSTEAD OF триггере не надо отменять действие, т.к. оно не выполняется (т.е. не надо использовать ROLLBACK TRANSACTION). Но для фиксации операции сам триггер должен выполнить соответствующую операцию (INSERT, DELETE, UPDATE).

В теле триггера могут использоваться две псевдо-таблицы с именами **inserted** и **deleted**. Схема этих псевдо-таблиц совпадает со

схемой той таблицы или представления, с которой связан триггер. При выполнении команды INSERT псевдо-таблица inserted содержит все вставляемые строки, deleted – пустая. При выполнении команды DELETE псевдо-таблица inserted – пустая, deleted содержит удаленные строки. При выполнении команды UPDATE псевдо-таблица inserted содержит новые значения строк, deleted – старые (заменяемые) значения.

Порядок выполнения работы

1. Разработать схемы таблиц для предметной области. Таблиц должно быть не менее трех, например, две таблицы сущностей с первичными ключами и связующая таблица, включающая внешние ключи к первым двум таблицам. Согласовать с преподавателем схемы таблиц.
2. Разработать запросы, полезные для работы в предметной области. Запросы должны содержать реализации следующих операций реляционной алгебры:

    1) выборка из всех трех таблиц; проекция и выборка из всех трех таблиц;
    2) внутреннее условное соединение двух таблиц; естественное соединение двух таблиц;
    3) условное соединение (внутреннее) трех таблиц;
    4) внешнее соединение двух таблиц (левое, правое, полное);
    5) объединение двух таблиц (можно использовать объединение выборок из одной и той же таблицы);
    6) вычисление всех агрегатных функций;
    7) упорядочивание данных;
    8) группирование данных.

4. Запустить консольную программу для работы с СУБД.
5. Создать базу данных.
6. Создать таблицы.
7. Заполнить таблицы значениями. В каждой таблице должно быть не менее 10 строк

8. В редакторе SQL запросов выполнить разработанные запросы. Текст SQL-скриптов с запросами сохранить в файл. Результаты запросов также сохранять в файл.
9. Создать дополнительную таблицу для хранения изменений. Назначение журнала изменений (пусть соответствующая таблица называется LOG) состоит в следующем. Если в какой-то таблице происходит изменение, то в LOG записывается характер изменения и время его выполнения.
10. Для одной из таблиц разработать по одному триггеру на каждую операцию, которые записывали бы в журнал записи об изменениях таблицы при добавлении, изменении и удалении записей.
11. Разработать триггер, выполняющий каскадное удаление записей из связных таблиц при удалении записи из родительской таблицы.
12. Разработать хранимую процедуру архивации данных одной из таблиц. Для этого создать таблицу для хранения архивных данных. В исходную таблицу добавить атрибут с индикатором, указывающим на то, что запись была занесена в архивную таблицу.

Рекомендации

1. Для создания базы данных надо выполнить команду CREATE DATABASE <имя базы данных>.
2. Для того, чтобы созданная база данных стала текущей для последующих действий, выполнить команду USE <имя базы данных>
3. Создать таблицы используя команду CREATETABLE. Например, CREATE TABLE E (fio CHAR(20), disc CHAR (30), mark INTEGER)
4. Для добавления новых значений в базу данных использовать оператор INSERT, например, INSERT INTO g (fio, gr) VALUES ('Petrov', 'IVT-200').
5. Для изменения данных использовать оператор UPDATE, а для удаления записей – оператор DELETE.

6. Для сохранения результатов запроса в файл следует в окне результирующей таблицы выполнить команду локального меню «Select All» и команду сохранения в файл.

Содержание отчета

1. Все таблицы с исходными данными.
2. Тексты всех разработанных SQL скриптов. Команды создания таблиц должны сопровождаться письменным объяснением назначения таблиц и их атрибутов.
3. Результаты выполнения запросов и объяснение полученных результатов.
4. Скрипты создания хранимых процедур и триггеров.
5. Описание результатов выполнения запросов, в которых задействованы хранимые процедуры и триггеры.
6. Выводы по работе.

Контрольные вопросы

1. Операции реляционной алгебры.
2. Оператор SELECT и реализация с его помощью операций реляционной алгебры.
3. Обобщающие функции.
4. Группирование записей.
5. Хранимые процедуры.
6. Триггеры.
7. Процедурный способ обеспечения целостности БД.
8. Декларативный способ обеспечения целостности БД.
9. Расширение языка SQL.

Разработка клиентской части базы данных

Цель работы: разработать программу на прикладном языке программирования для работы с базой данных.

Спецификация программы:

Программа должна подключаться ранее созданной базе данных и должна позволять пользователю выполнять предусмотренные требования к базе данных операции. Соединение с базой данной с помощью технологии ADO (от англ. ActiveX Data Objects — «объекты данных ActiveX») — интерфейс программирования приложений для доступа к данным, разработанный компанией Microsoft.

Для создания формы использовать компоненты:

Label – для подписей

Button – для инициирования действий

Edit – для вывода количества полей (колонок) и записей (строк) таблицы

ADOConnection –компонент для подключения к базе данных

ADOTable – компонент для работы со структурой и данными таблицы базы данных

DataSource – компонент для передачи данных компоненту DBGrid и DBNavigator.

DBGrid – компонент для визуализации таблицы из БД

DBNavigator – компонент для редактирования записей подключенной таблицы БД

Примерная компоновка формы программы представлена на рисунке 1.

Рекомендации для выполнения лабораторной работы:

1) сохранить ранее созданную базу данных в папке программы;

2) запустить C++ Builder. При запуске автоматически создается новый проект. Окно C++ Builder показано на рисунке 2. Для создания нового проекта, в случае если он не создался автоматически или вы его закрыли, выполнить команду меню File / New /Application;

3) сохранить проект в свою рабочую папку, выполнив команду меню File / Save Project As. Будет сохранено несколько файлов проекта;

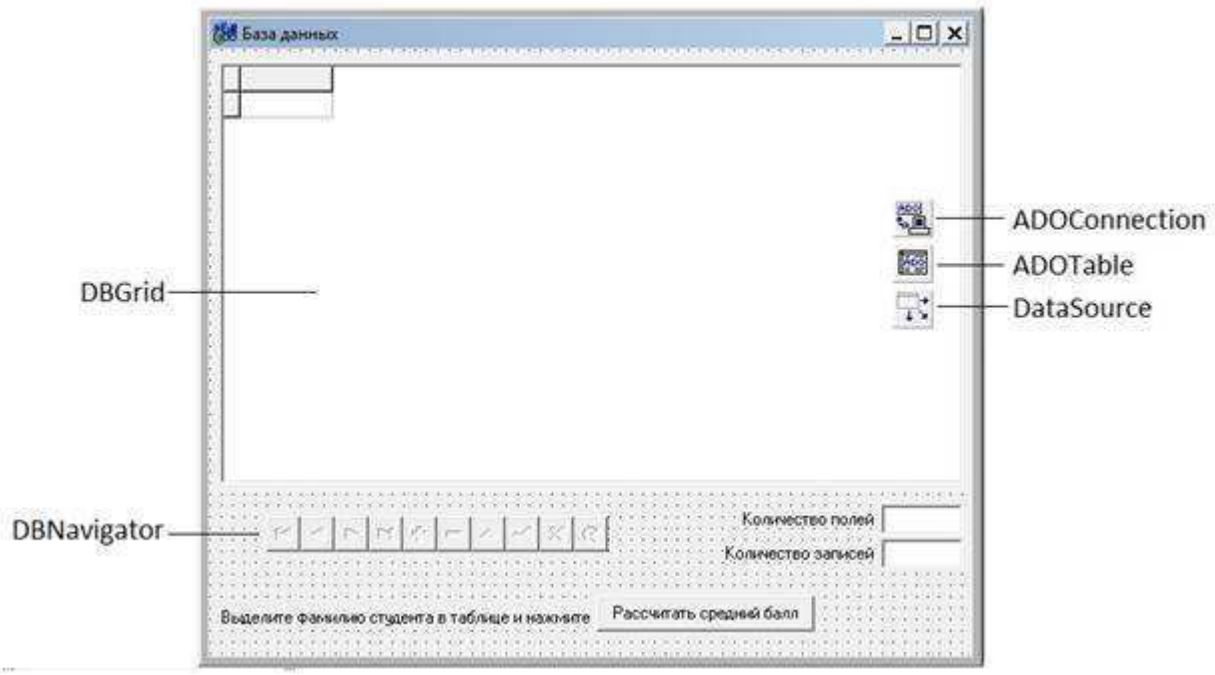


Рис. 1 - Примерная компоновка формы

4) расположить на форме требуемое количество объектов;

Вкладка Standard: Label, Button, Edit.

Вкладка ADO: ADOConnection, ADOTable.

Вкладка DataAccess: DataSource.

Вкладка DataControls: DBGrid, DBNavigator.

5) изменить подписи объектов Label и пользовательской формы Form1. Для этого необходимо у перечисленных объектов отредактировать свойство Caption в соответствии с рисунком 1;

6) у объектов Edit и ComboBox очистить поле свойства Text;

7) поскольку объекты Edit используются только для вывода, то необходимо присвоить свойству ReadOnly для этих объектов значение true;

8) настроить подключение к БД следующим образом.

Выделите компонент ADOConnection1. Установить значение false для свойств Connected (соединение) и LoginPromt (вход с паролем) Сформировать строку подключения ConnectionString (строка параметров подключения к базе данных), нажав на кнопку с тремя точками.

Рис. 2 - Настройка свойств объекта ADOConnection1

В появившемся окне выберите пункт «Use Connection String» и нажмите на кнопку «Build»:

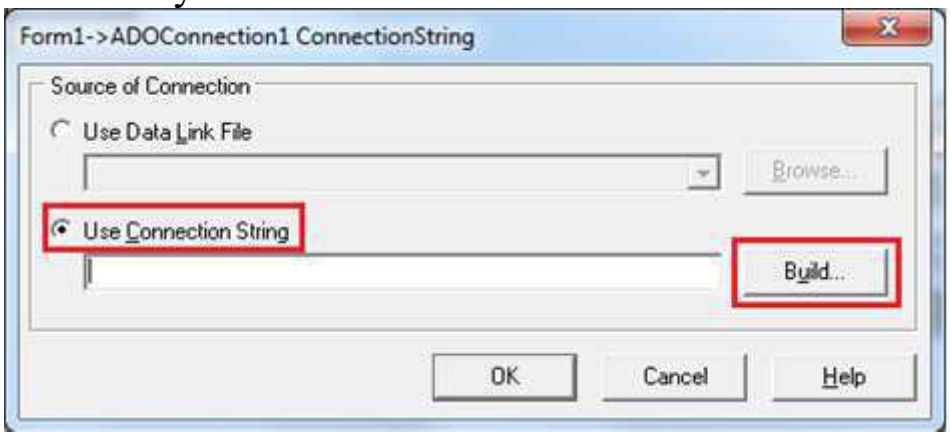


Рис. 3 - Окно настройки подключения

Далее необходимо выбрать поставщика данных и нажать на кнопку далее:

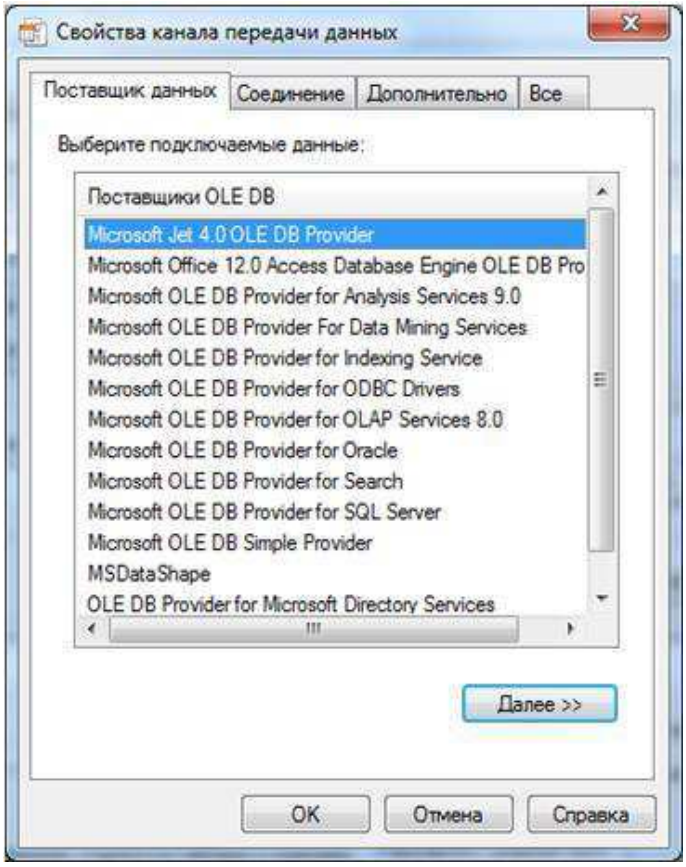


Рис. 4 - Выбор поставщика данных

Указать путь к базе данных и проверить соединение.

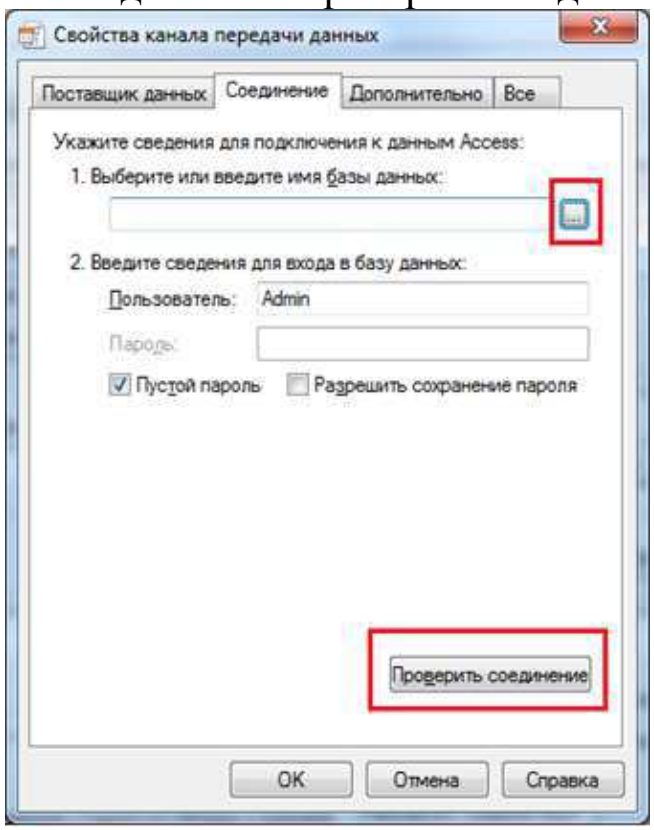


Рис. 5 - Выбор файла базы данных

Примените все изменения и поменять значение свойства Connected на true.

Важное примечание: Строка подключения представляет собой обычную строку, в которой перечислены параметры подключения программы к базе данных. Строка подключения содержит путь к базе данных, который при необходимости можно заменять программно. Это необходимо для подключения различных баз данных одного типа к программе.

На этом настройка компонента ADOConnection1 закончена.

Выделите объект ADOTable и в окне «Object Inspector» в поле Connection выберите объект ADOConnection1, а в поле TableName выбрать таблицу из базы данных. Если при выборе таблицы возникает, то соединение с базой данных не было установлено. В этом случае следует проверить строку подключения в объекте ADOConnection1. В самую последнюю очередь установить переключатель Active в положение true.

Рис. 6 - Настройка объекта ADOTable1

Выделите объект DataSource1 и в списке свойств в поле DataSet выберите объект ADOTable1.

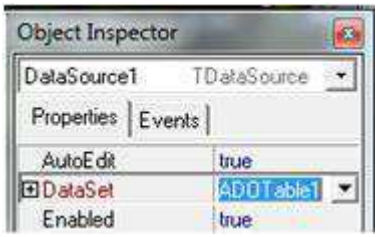


Рис. 7 - Настройка объекта DataSource1

Для объектов DBGrid и DBNavigator в поле свойства DataSource выберите объект DataSource1

При правильном выполнении всех вышеперечисленных операций в объект DBGrid должна быть загружена таблица из базы данных. Настройка подключения таблицы базы данных к программе завершена. Далее необходимо написать код для работы с базой.

Контрольные вопросы

1. Операции реляционной алгебры.
2. Оператор SELECT и реализация с его помощью операций реляционной алгебры.
3. Обобщающие функции.
4. Группирование записей.
5. Хранимые процедуры.
6. Триггеры.
7. Процедурный способ обеспечения целостности БД.
8. Декларативный способ обеспечения целостности БД.

Разработка и управление хранилищем данных

Хранилища данных – это специальным образом сконструированные базы данных, которые предназначены не столько для хранения информации, сколько для быстрого получения сложных аналитических данных. Перечислим основные характерные моменты, связанные с проектированием и эксплуатацией хранилищ данных, построенных на реляционной модели:

1. Хранилища данных строятся на особой модели базы данных, которая пренебрегает многими аспектами нормализации. В основном, используются схемы типа «Снежинка» и «Звезда». В обеих схемах выделяется центральная **таблица фактов**, которая и содержит данные для анализа, и множество **таблиц измерений** (обычно сильно ненормализованных), содержащие информацию об объектах, в разрезе которых осуществляется анализ, и которые соединены с таблицей фактов связью типа «один-ко-многим».

2. Хранилища не предназначены для большого количества операций модификации данных. Обычно в хранилищах дублируются данные из операционных баз данных (базы, в которые попадают первичные данные). Зато они существенно зависят от операций экспорта из различных источников (не обязательно из баз данных).

3. Большая часть запросов к хранилищу данных связана с таблицей фактов, соединенной только с теми измерениями, которые необходимы для целей анализа.

В качестве учебного примера рассмотрим особенности проектирования и использования хранилища для примера базы с данными об учебном процессе. Для определенности будем использовать в качестве СУБД MS SQL Server. Рассмотрим основные этапы проектирования и использования хранилищ данных на примере схемы «Звезда». В качестве таблицы фактов будем использовать таблицу результатов сдачи экзаменов студентами. В качестве таблиц измерений будут фигурировать таблицы «Студенты», «Преподаватели», «Дисциплины».

Итак, у нас получилась довольно простая модель данных с тремя таблицами измерений. Отметим денормализацию на уровне таблицы измерений «Преподаватели» (в нее включается название кафедры, а не ее номер, как это было ранее). Кроме того, отметим наличие еще одного измерения, для которого таблица не создается. Этим измерением является дата сдачи экзамена. Фактами же в данном случае являются набранные баллы и соответствующая им оценка.

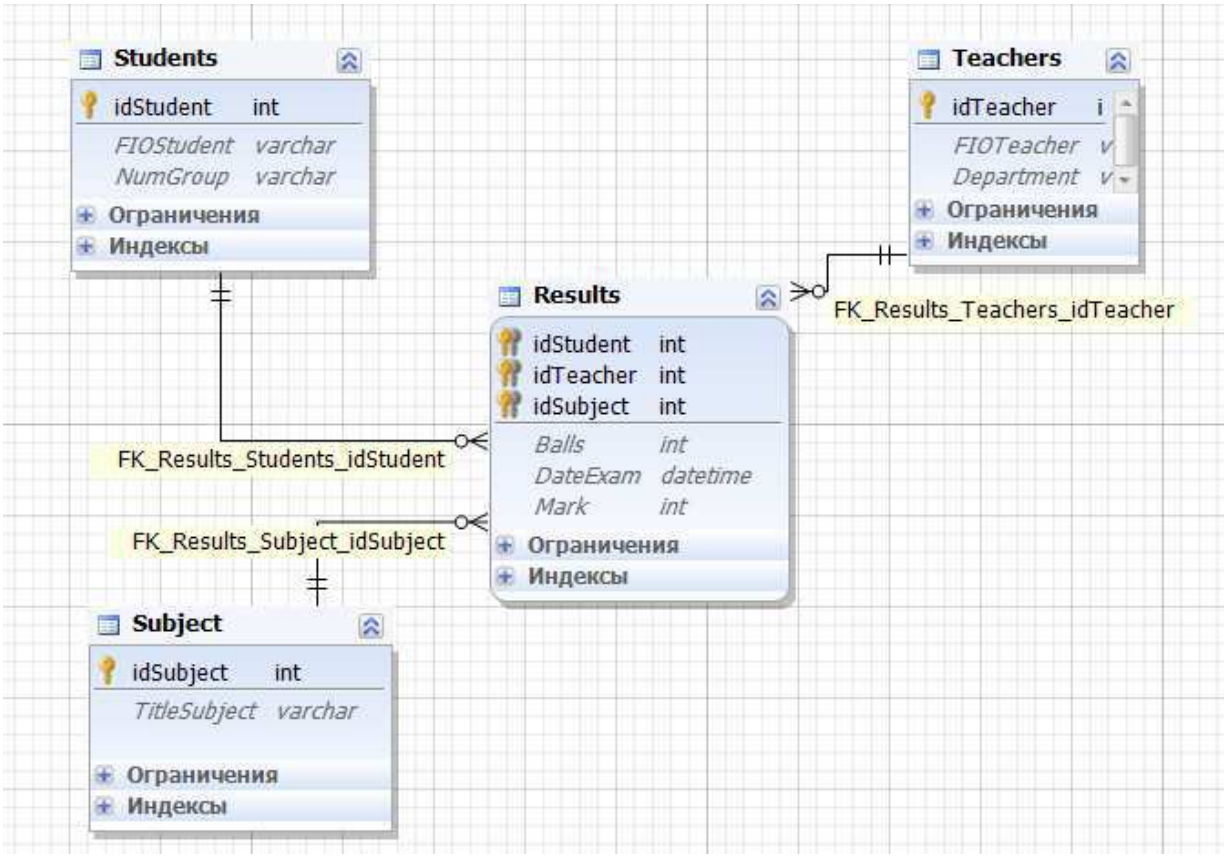


Рис. 1 - Схема «Звезда» для хранилища данных.

Произведем загрузку данных из базы данных через файлы различного текстового формата. Начнём с загрузки данных в таблицы измерений.

Таблица «Students». Таблица студентов должна быть загружена в том же виде, что и хранится в базе данных деканата. Используем для загрузки файл формата CSV. Этот формат определяет текстовый файл, в котором в первой строке задаются имена столбцов, а каждая следующая строка содержит данные об одной записи, поля разделяются с помощью специального символа- разделителя.

Чтобы создать такой файл оболочка dbForge Studio содержит специальный конструктор экспорта данных. Доступ к нему можно

получить, например, с помощью вкладки «Миграция данных» и опции «Экспорт данных». При экспорте придется указать источник данных для экспорта (таблицы или представления) и файл, в который произойдет сохранение информации. В результате будет получен следующий файл:

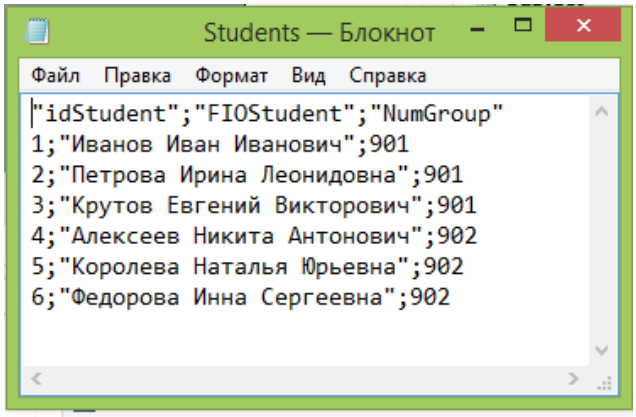


Рис. 2 - Файл «Student.csv»

При импорте данных в базу хранилища данных следует воспользоваться той же вкладкой «Миграция данных» и опцией «Импортировать внешние данные». Сначала потребуется выбрать формат и файл с данными, затем определить таблицу, в которую осуществляется импорт, далее будет показан источник данных и можно будет сделать необходимые настройки (пропустить строки,

установить символы-разделители), далее устанавливается соответствие между столбцами источника и приемника данных (в нашем случае они называются одинаково) и далее следуют уточнения по поводу операций импорта (какие операции произвести – добавление новых записей, модификация старых), интерпретация типов данных и прочее.

Таблица «Subjects». Таблица дисциплин также должна быть

загружена в том же виде. Используем для ее загрузки файл формата XML. Этот формат определяет текстовый файл, в котором структура и данные размечены с помощью специальных тегов. Как и в предыдущем случае следует использовать опции «Экспорт данных» и «Импортировать внешние данные». В результате будет сгенерирован следующий файл:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<xs:schema id="Root"
xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
<xs:element name="Root"
msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
                                        <xs:complexType>
                                <xs:choice minOccurs="0" maxOccurs="unbounded">
<xs:element name="Table">
<xs:complexType>
<xs:attribute name="idSubject" type="xs:int" />
<xs:attribute name="TitleSubject" type="xs:string" />
</xs:complexType>
</xs:element>
                                        </xs:choice>
                                    </xs:complexType>
</xs:element>
</xs:schema>
<Table idSubject="1" TitleSubject="Математический анализ" />
<Table idSubject="2" TitleSubject="Алгебра и геометрия" />
<Table idSubject="3" TitleSubject="Теория вероятностей" />
```

<Table idSubject="4" TitleSubject="Методы оптимизации" />

<Table idSubject="5" TitleSubject="Вычислительная математика" />

<Table idSubject="6" TitleSubject="Алгоритмизация" />

<Table idSubject="7" TitleSubject="Программирование" />

<Table idSubject="8" TitleSubject=

"Объектно-ориентированное программирование" />

<Table idSubject="9" TitleSubject="Базы данных" />

<Table idSubject="10" TitleSubject="Web-программирование" />

</Root>

Таблица «Teachers». При экспорте таблицы преподавателей мы должны на самом деле экспортировать результат его естественного соединения с таблицей кафедр. Для этого придется создать специальное представление данных, так как запрос не может быть использован в качестве источника данных для экспорта.

```
CREATE VIEW Teacher_for_export AS
SELECT idTeacher, FIOTeacher, TitleDepartment FROM Teachers INNER JOIN Departments
ON Teachers.idDepartment=Departments.idDepartment;
```

Используем для загрузки данных о преподавателях файл формата Excel, указав при экспорте в качестве источника данных созданное представление. При импорте этих данных следует обратить внимание на настройки относительно строки с заголовками столбцов – требуется явно указать, что первая строка является строкой с именами столбцов, тогда данные будут считываться только со следующей строки. Не забудьте при импорте настроить соответствие для имен столбцов с названием кафедры.

	A	B	C
1	idTeacher	FIOTeacher	TitleDepartment
2	17	Федосеев Александр Иванович	Кафедра математики
3	18	Александрова Анна Петровна	Кафедра математики
4	19	Кириллов Дмитрий Викторович	Кафедра информатики
5	20	Никитин Владимир Иванович	Кафедра информатики
6			

Рис. 3 - XLS-файл с данными о преподавателях

Отметим, что экспортировать данные можно и в другие форматы: RTF, PDF и т.д., но для импорта в другую базу эти форматы не будут пригодными.

Вполне естественно, что файл для импорта не обязательно должен быть сгенерирован каким-либо СУБД. Так как таблицу результатов сдачи зачетов- экзаменов в базе данных мы практически не заполняли, экспортировать в таблицу фактов нам пока нечего.

Напишем приложение, которое генерирует файл с оценками, и далее импортируем его в нашу базу данных. Для простоты не будет рассматривать ситуацию, когда разные группы должны сдавать разный набор дисциплин. Будем предполагать, что у каждого студента имеются оценки по всем дисциплинам, которые есть в таблице измерений.

Программный код такого проекта по генерации данных будет следующим. Комментариев в коде должно быть достаточно для понимания алгоритма генерации:

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.Data; using System.Data.Odbc; using System.IO;
```

```
namespace generator
{
class Program
{
// в набор данных будут загружены измерения
// для правильной генерации кодов объектов static DataSet ds = new DataSet();

// вспомогательная функция перевода баллов в оценку static int GetMark(int b)
{
if (b < 55)
        return 2; if (b < 71)
        return 3; if (b < 86)
return 4;
return 5;
}

static void Main(string[] args)
{
OdbcConnection con = new OdbcConnection("DSN=proba"); con.Open();

// загрузка таблицы студентов OdbcDataAdapter adapt1 =
        new OdbcDataAdapter("select * from Students", con); adapt1.Fill(ds, "Students");

// загрузка таблицы дисциплин OdbcDataAdapter adapt2 =
        new OdbcDataAdapter("select * from Subjects", con); adapt2.Fill(ds, "Subjects");
// загрузка таблицы преподавателей OdbcDataAdapter adapt3 =
        new OdbcDataAdapter("select * from Teachers", con); adapt3.Fill(ds, "Teachers");

con.Close();

//создание текстового файла для записи
//сгенерированных данных StreamWriter tf = new StreamWriter
```

```
(new FileStream("Results.csv", FileMode.Create));

// запись строки заголовка таблицы.
// Для вывода кавычек используется \” tf.WriteLine("\"idStudent\";\"idSubject\";
\"idTeacher\";\"DateExam\";
\Balls\";\"Mark\"");

// вызов функции генерации данных GenerateData(tf);

// закрытие файла tf.Close();
}

// функция генерации оценок экзаменов по всем дисциплинам static void
GenerateData(StreamWriter tf)
{
Random r = new Random();

// цикл перебора всех дисциплин
foreach (DataRow dr_subj in ds.Tables["Subjects"].Rows)
{
// генерируем преподавателя,
// которому сдается данный предмет
int i = r.Next(ds.Tables["Teachers"].Rows.Count); int nom_teach =
(int)ds.Tables["Teachers"].Rows[i]["idTeacher"];

//генерируем три даты для сдачи экзамена DateTime [] dates=new DateTime[3];

//определяем зимнюю или весенюю сессии if (r.Next(100) % 2 == 0)
{
// зимняя сессия dates[0] =
        new DateTime(2014, 1, 5 + r.Next(20)); dates[1] =
        new DateTime(2014, 1, 5 + r.Next(20)); dates[2] =
new DateTime(2014, 1, 5 + r.Next(20));
```

```
}
else
{
// весенняя сессия dates[0] =
                new DateTime(2014, 6, 3 + r.Next(24)); dates[1] =
                new DateTime(2014, 6, 3 + r.Next(24)); dates[2] =
new DateTime(2014, 6, 3 + r.Next(24));
}

// цикл перебора всех студентов foreach (DataRow dr_stud in
ds.Tables["Students"].Rows)
{
// генерируем оценку - будем оптимистами,
// все оценки положительные int balls = 55 + r.Next(45); int mark = GetMark(balls);

// генерируем дату сдачи экзамена int nom_date = r.Next(120) % 3;

// формируем строку вывода сгенерированной
// информации tf.WriteLine("{0};{1};{2};{3};{4};{5}",
dr_stud["idStudent"], dr_subj["idSubject"], nom_teach, dates[nom_date], balls, mark);
}
}
}
}
}
```

Таким образом, получим следующий csv-файл, который импортируем в базу данных для организации хранилища:

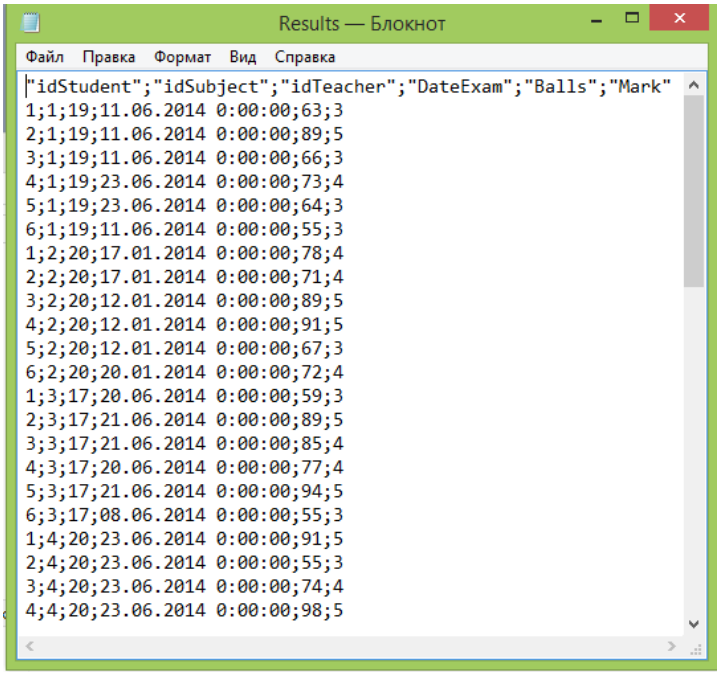


Рис. 4 - CSV-файл с оценками студентов.

По своей сути таблица фактов представляет собой OLAP-куб (Online Analytical Processing), который позволяет быстро находить интересующую информацию в разных аспектах. Существуют стандартные операции с OLAP-кубами: срез, вращение, консолидация и детализация. Кратко опишем суть этих операций:

- Операция среза обычно определяет все возможные данные с фиксацией одного или нескольких измерений.
- Операция вращения предполагает переупорядочивание измерений.
- Операция консолидации предполагает выполнение агрегирования данных по отдельным измерениям.
- Операция детализации предполагает получение нового куба, содержащего не все значения измерений.

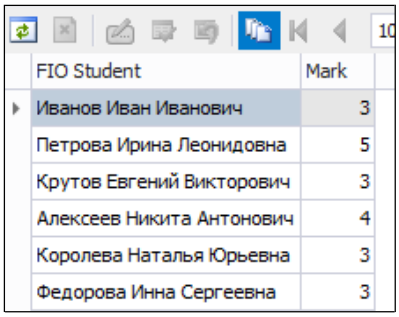
А теперь продемонстрируем на примере ряда запросов применение этих операций для получения информации. Обычно все запросы к хранилищу данных обращаются к таблице фактов

или к естественному соединению таблицы фактов с необходимыми для запроса таблицами измерений:

- Получить все оценки по дисциплине «Математический анализ». В этом случае мы используем операцию среза. Можно сказать, что это одновременно является срезом по измерению «Преподаватель» (так как, согласно генерации данных, дисциплина ведется только одним преподавателем):

```
SELECT FIOStudent, Mark FROM Results INNER JOIN Students ON
Results.idStudent=Students.idStudent
WHERE idSubject =
(SELECT idSubject FROM Subjects
WHERE TitleSubject='Математический анализ');
```

Результатом является другой гиперкуб, в данном случае с одним измерением FIOStudent и фактом - оценкой.

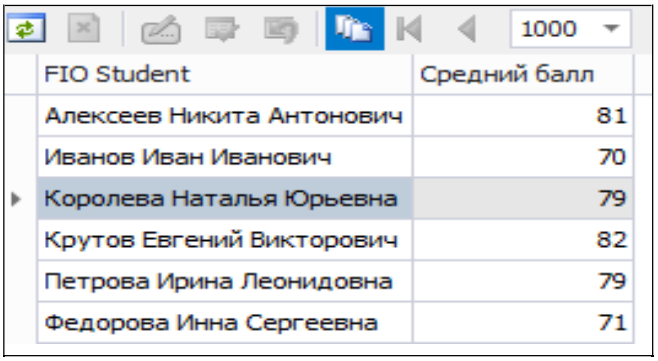


FIO Student	Mark
Иванов Иван Иванович	3
Петрова Ирина Леонидовна	5
Крутов Евгений Викторович	3
Алексеев Никита Антонович	4
Королева Наталья Юрьевна	3
Федорова Инна Сергеевна	3

Рис. 5 - Оценки студентов по дисциплине «Математический анализ»

- Получить средние баллы всех студентов. Здесь демонстрируется операция консолидации, которая выражается укрупнением группировок по измерениям.

```
SELECT FIOStudent, AVG(Balls) AS "Средний балл" FROM Results INNER JOIN Students
ON Results.idStudent=Students.idStudent GROUP BY FIOStudent;
```



FIO Student	Средний балл
Алексеев Никита Антонович	81
Иванов Иван Иванович	70
Королева Наталья Юрьевна	79
Крутов Евгений Викторович	82
Петрова Ирина Леонидовна	79
Федорова Инна Сергеевна	71

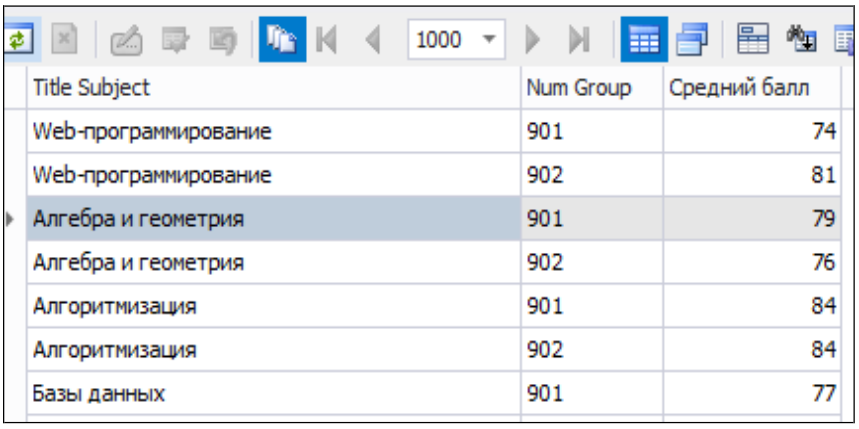
Рис. 6 - Средние баллы студентов

- Получить средние баллы сдачи экзаменов по группам студентов. В данном случае производится консолидация по более крупной группе – номерам групп студентов. Таким образом, результатом является гиперкуб, в котором два измерения – дисциплина и номер учебной группы. В каком-то смысле здесь производится переупорядочивание измерений, поэтому можно говорить и о применении операции вращения.

```
SELECT TitleSubject, NumGroup, AVG(Balls) AS "Средний балл" FROM Results INNER JOIN Subjects

ON Results.idSubject=Subjects.idSubject

INNER JOIN Students ON Results.idStudent=Students.idStudent GROUP BY TitleSubject, NumGroup ORDER BY TitleSubject;
```



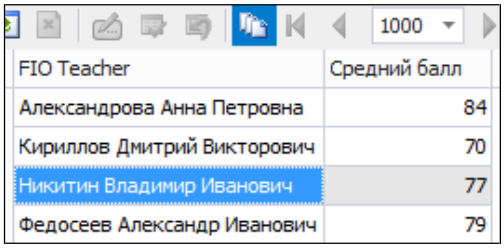
Title Subject	Num Group	Средний балл
Web-программирование	901	74
Web-программирование	902	81
Алгебра и геометрия	901	79
Алгебра и геометрия	902	76
Алгоритмизация	901	84
Алгоритмизация	902	84
Базы данных	901	77

Рис. 7 - Средние баллы по предметам и группам

- Получить средние баллы по преподавателям. Какому студенту не хотелось знать перед экзаменом, какой преподаватель

более лояльный? Ситуация аналогичная запросу о средних баллах студентов. Как видно, нашелся один заметно более строгий преподаватель.

```
SELECT FIOTeacher, AVG(Balls) AS "Средний балл" FROM Results INNER JOIN Teachers
ON Results.idTeacher=Teachers.idTeacher GROUP BY FIOTeacher;
```



FIO Teacher	Средний балл
Александрова Анна Петровна	84
Кириллов Дмитрий Викторович	70
Никитин Владимир Иванович	77
Федосеев Александр Иванович	79

Рис. 8 - Средние баллы по преподавателям

- Получить количество пятерок, которые поставили преподаватели.

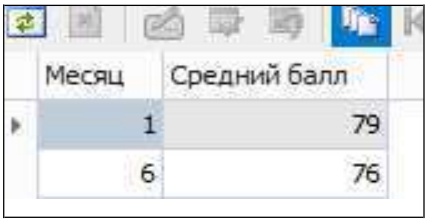
```
SELECT FIOTeacher, COUNT(Mark) AS "Количество пятерок" FROM Results INNER JOIN
Teachers
ON Results.idTeacher=Teachers.idTeacher WHERE Mark=5
GROUP BY FIOTeacher;
```

FIO Teacher	Количество пятерок
Александрова Анна Петровна	3
Кириллов Дмитрий Викторович	2
Никитин Владимир Иванович	7
Федосеев Александр Иванович	8

Рис. 9 - Количество пятерок, которые поставили преподаватели

- Узнать, в какую сессию (зимнюю или летнюю) студенты сдают экзамены лучше. Для этого мы рассчитаем средние баллы сдачи экзаменов зимой и летом. Здесь мы опять сталкиваемся с операцией консолидации, но по нестандартному измерению – дате сдачи экзамена.

```
SELECT DATEPART(month,DateExam) AS "Месяц", AVG(Balls) AS "Средний балл"
FROM Results GROUP BY DATEPART(month,DateExam);
```

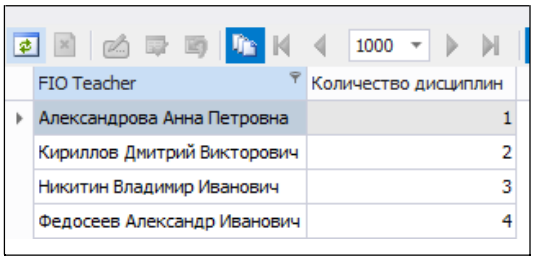


Месяц	Средний балл
1	79
6	76

Рис. 10 - Средние баллы в зимнюю и летнюю сессии

- Получить количество дисциплин, которые ведет каждый из преподавателей. Здесь в качестве факта выступает измерение таблицы фактов idSubject. В данном случае сначала применяем операцию детализации, выбирая для нового гиперкуба из таблицы фактов только некоторые данные, после чего применяется операция консолидации посредством группировки по измерению «Преподаватель».

```
SELECT FIOTeacher, COUNT(idSubject) AS "Количество дисциплин" FROM
(SELECT DISTINCT FIOTeacher, idSubject FROM Results
INNER JOIN Teachers
ON Results.idTeacher=Teachers.idTeacher) AS Q1
GROUP BY Q1.FIOTeacher;
```



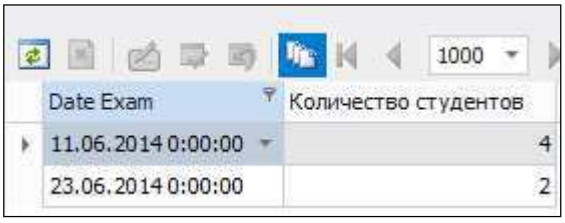
FIO Teacher	Количество дисциплин
Александрова Анна Петровна	1
Кириллов Дмитрий Викторович	2
Никитин Владимир Иванович	3
Федосеев Александр Иванович	4

Рис. 11 - Количество дисциплин каждого из преподавателей.

- Получить количество студентов, сдавших дисциплину «Математический анализ» на каждую дату сдачи этого предмета. Здесь используется операция среза, после чего применяется операция консолидации по дате сдачи экзамена.

```
SELECT DateExam, COUNT(*) AS "Количество студентов" FROM Results WHERE
idSubject=(SELECT idSubject FROM Subjects WHERE

TitleSubject='Математический анализ') GROUP BY DateExam;
```



Date Exam	Количество студентов
11.06.2014 0:00:00	4
23.06.2014 0:00:00	2

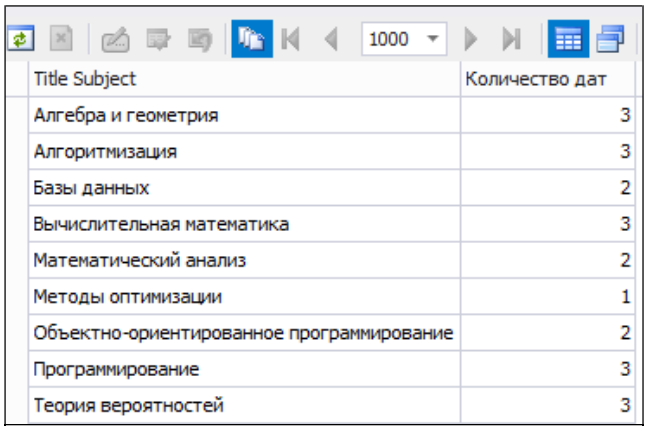
Рис. 12 - Количество студентов, сдавших математический анализ на разные даты

- Получить количество дат, в которые проводились экзамены по каждому из предметов. Данный запрос будет связан с последовательностью операции детализации и консолидации.

```
SELECT TitleSubject, COUNT(DateExam) AS "Количество дат" FROM (SELECT DISTINCT
TitleSubject, DateExam FROM Results

INNER JOIN Subjects

ON Results.idSubject=Subjects.idSubject) AS Q1 GROUP BY Q1.TitleSubject;
```



Title Subject	Количество дат
Алгебра и геометрия	3
Алгоритмизация	3
Базы данных	2
Вычислительная математика	3
Математический анализ	2
Методы оптимизации	1
Объектно-ориентированное программирование	2
Программирование	3
Теория вероятностей	3

Рис. 13 - Количество дат сдачи каждого из экзаменов

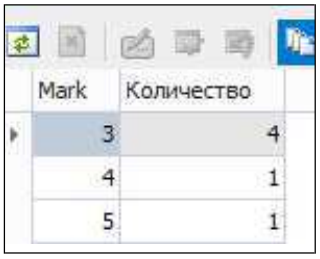
- Получить количество оценок каждого типа по дисциплине «Математический анализ». В данном примере применение среза и последующей детализации приведет к построению гиперкуба, в котором можно сделать факт (оценку) измерением. Затем к этому новому гиперкубу применяется операция консолидации.

```
SELECT Mark, COUNT(*) AS "Количество" FROM Results WHERE
                                    idSubject =

(SELECT idSubject FROM Subjects

WHERE TitleSubject='Математический анализ')

GROUP BY Mark;
```



Mark	Количество
3	4
4	1
5	1

Рис. 14 - Количество оценок каждого типа по математическому анализу

Оболочка dbForge Studio содержит собственные средства для создания отчетов. Отчеты формируются в виде файлов с расширением rbd. Они содержат разметку элементов данных в отчете. Файл можно создать с помощью конструктора, который вызывается с помощью вкладки «Анализ данных» стартовой страницы и пункта «Дизайн нового отчета».

На первой странице дизайнера (мастера) предлагается выбрать базу данных и источник данных (таблицу или запрос). На следующей странице можно будет выбрать таблицу или написать текст запроса для отображения в отчете. Далее можно будет выбрать поля для отображения в отчете. На следующем шаге задаются принципы группировки записей (группировок может быть несколько, и они могут образовывать иерархию). Далее следуют несколько шагов с заданием характеристик внешнего вида отчета.

Например, на основании запроса, представляющего собой

соединение таблицы фактов и всех таблиц измерений, сгенерируем отчет:

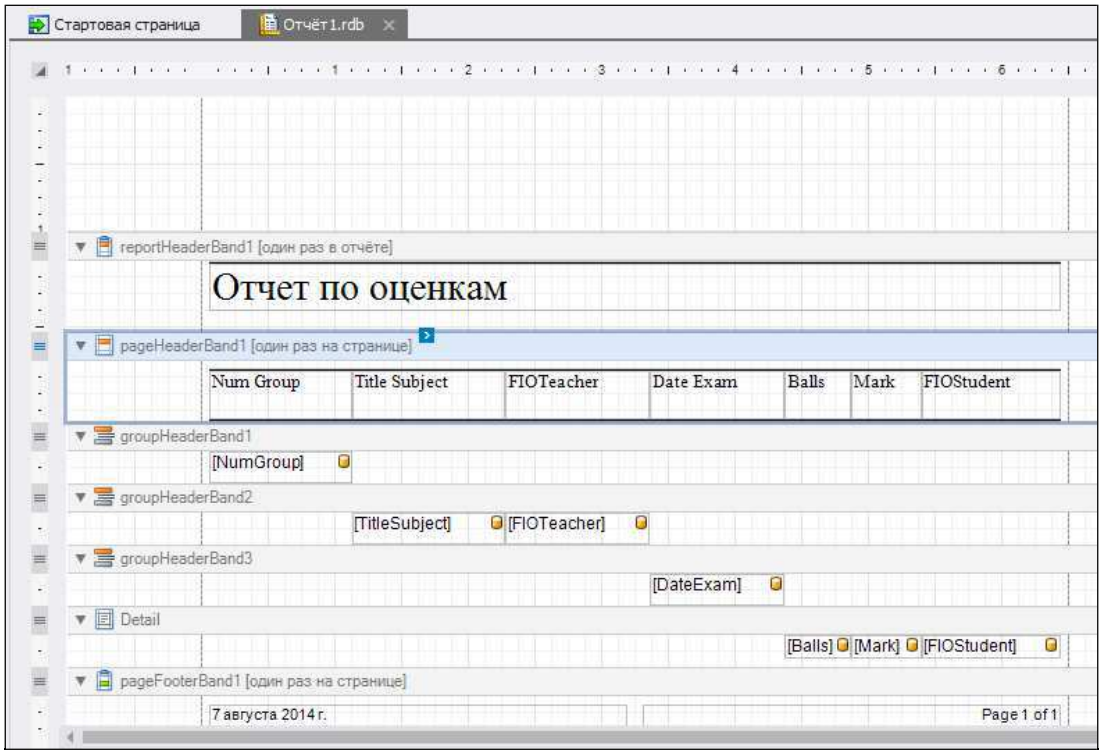


Рис. 15 - Вид отчета

Основная группировка данных производится по номеру группы, далее по иерархии идут группировки по дисциплине (преподавателю), дате экзамена. Детализацией записи являются ФИО студента, набранные баллы и полученная оценка.

Сгенерированный отчет можно редактировать с помощью специальных панелей и окон свойств.

Результат генерации отчета можно посмотреть, войдя в режим просмотра (вкладка внизу области отчета).

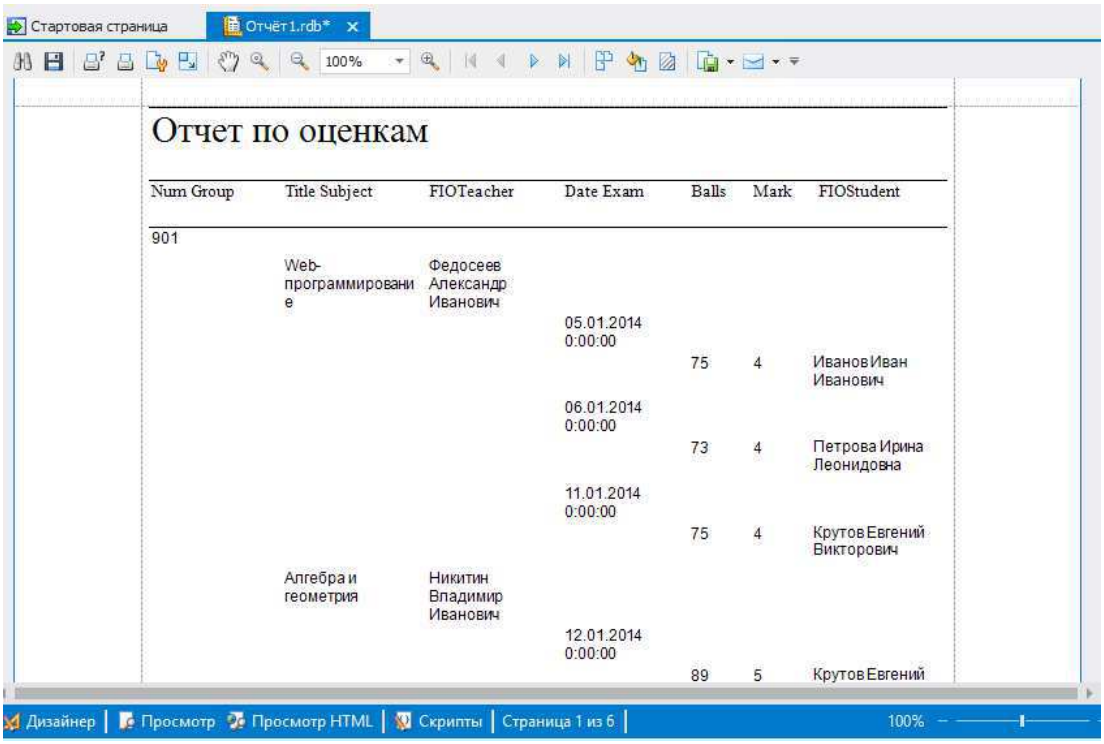


Рис. 16 - Сгенерированный отчет

Задание: спроектировать хранилище данных по заданию преподавателя.

Контрольные вопросы

1. Что такое набор данных?
2. Что такое хранилище данных?
3. Назовите наиболее распространённые схемы построения хранилищ данных.
4. Что такое таблица фактов?
5. Что такое таблица измерений?
6. Опишите стандартные операции с OLAP-кубами.

Список литературы

1. Громов, Ю.Ю. Управление данными [Электронный ресурс] : учебник / Ю.Ю. Громов и др. - Тамбов : Изд-во ФГБОУ ВПО "ТГТУ", 2015. - 192 с. - Режим доступа / http://biblioclub.ru/index.php?page=book_view_red&book_id=444642.
2. Цехановский, В.В. Управление данными [Текст] : учебник / В. В. Цехановский, В. Д. Чертовской. - Санкт-Петербург : Лань, 2015. - 432 с.
3. Васюков, О.Г. Управление данными [Электронный ресурс] : учебно-методическое пособие / О.Г. Васюков. - Самара : СГАСУ, 2014. - 161 с. - Режим доступа / http://biblioclub.ru/index.php?page=book_view_red&book_id=438334.
4. Кузовкин, А.В. Управление данными [Текст]: учебник / А.В. Кузовкин, А.А. Цыганов, Б.А. Щукин. – М.: Академия, 2010. - 256 с.