На правах рукописи

Минобрнауки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Оренбургский государственный университет»**

Кафедра прикладной информатики в экономике и управлении

Методические указания для обучающихся по освоению дисциплины

*«Б.1.Б.16 Базы данных»*

Уровень высшего образования

БАКАЛАВРИАТ

Направление подготовки

*38.03.05 Бизнес-информатика*

(код и наименование направления подготовки)

*Информационные системы в экономике*

(наименование направленности (профиля) образовательной программы)

Тип образовательной программы

*Программа академического бакалавриата*

Квалификация

*Бакалавр*

Форма обучения

*Очная*

Год набора 2022

Составители Панова Н.Ф

Методические указания рассмотрены и одобрены на заседании кафедры прикладной информатики в экономике и управлении

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Жук М.А.

Методические указания является приложением к рабочей программе по дисциплине «Базы данных», зарегистрированной в ЦИТ под учетным номером\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
|  |
|  |

**Содержание**

|  |  |
| --- | --- |
| 1 Методические указания по лекционным занятиям ………………..... | 4 |
| 2 Методические указания по практическим занятиям…………………. | 6 |
| 3 Методические указания по лабораторным занятиям …..…………..... | 7 |
| 4 Методические указания по выполнению курсового проекта……..... | 9 |
| 5 Методические указания по промежуточной аттестации по дисциплине………………………………………………………………….. | 10 |

**1 Методические указания по лекционным занятиям**

Лекции являются одним из основных методов обучения по дисциплине «Базы данных», решающим следующие задачи:

* изложить важнейший материал программы курса;
* познакомить с основными понятиями технологии баз данных, моделями данных, способами проектирования баз данных;
* познакомить с основами работы в среде реляционных СУБД;
* развивать у обучающихся потребность к самостоятельной работе над учебниками и научной литературой.

Изучение дисциплины следует начинать с проработки рабочей программы, особое внимание необходимо уделить целям и задачам, структуре и содержанию курса.

При конспектировании лекций обучающиеся должны излагать услышанный материал на лекции своими словами. Необходимо выделять важные места в своих записях. Каждый раз, когда что-либо не понятно, необходимо записывать свои вопросы. По возможности можно сравнивать свои конспекты с конспектами двух-трех других обучающихся, при этом дополняя и исправляя свои записи.

Успешное освоение курса предполагает активное, творческое участие обучающегося путем планомерной, повседневной работы.

Лекционный материал необходимо кратко записывать, обращая внимание, на логику изложения материла, аргументацию и приводимые примеры.

Лекционный материал следует просматривать в тот же день, когда читалась лекция, помечая непонятные места. Если самостоятельно не удалось разобраться в материале, сформулируйте вопросы и обратитесь за консультацией на ближайшей лекции к преподавателю.

Рекомендуемую дополнительную литературу следует прорабатывать после изучения данной темы по учебнику и материалам лекции.

При подготовке материала необходимо обращать внимание на точность определений, последовательность изучения материала, аргументацию, собственные примеры.

Каждую неделю рекомендуется отводить время для повторения пройденного материала, проверяя свои знания, умения и навыки по контрольным вопросам и тестам.

**2 Методические указания по практическим занятиям**

Практические занятия по дисциплине имеют большое значение. Именно на этих занятиях обучающиеся отрабатывают навыки проектирования реляционных баз данных с помощью правил нормализации, учатся разрабатывать инфологичекие модели баз данных, преобразовывать их в даталогичекские, конструировать запросы с помощью языка SQL. Концептуальное моделирование данных- процесс сложный и творческий. Необходимо научиться использовать готовые шаблоны проектирования, например, моделирование иерархических структур с помощью рекурсивных связей, шаблон для моделирования адреса и т.д.

Подготовку к каждому практическому занятию студент должен начать с изучения или повторения теоретического материала. Очень важно приходить на занятие со знанием теории, чтобы быть готовым к выполнению практических заданий. Результат такой работы должен проявиться в способности свободно ответить на теоретические вопросы практикума, выступать и участвовать в коллективном обсуждении вопросов изучаемой темы, правильно выполнять практические задания и контрольные работы.

В процессе подготовки к практическим занятиям, студенту необходимо регулярно выполнять домашние задания. Практических занятий в семестре не много, поэтому невыполнение даже одного задания приведет к большому пробелу в знаниях. Следует обратить особое внимание на самостоятельное изучение рекомендованной литературы. При всей полноте конспектирования лекции в ней невозможно изложить весь материал из-за лимита аудиторных часов. Поэтому самостоятельная работа с учебниками, учебными пособиями, материалами периодических изданий и Интернета является наиболее эффективным методом получения дополнительных знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала, формирует у студента собственное отношение к конкретной проблеме.

**3 Методические указания по лабораторным занятиям**

Выполнение лабораторных работ обучающимися является необходимым условием успешного освоения дисциплины «Базы данных». Выполнение лабораторных работ способствует укреплению теоретического материала и освоению практических навыков решения экономических и расчетных задач с применением современных средств обработки данных.

Программа курса, согласно рабочей программе, включает несколько лабораторных работ, охватывающих весь учебный курс. Каждой лабораторной работе предшествует теоретический материал, пример реализации задач, варианты заданий для самостоятельного решения и контрольные вопросы по рассматриваемой теме.

Перед началом выполнения лабораторной работы необходимо ознакомиться с целью и содержанием работы, изучить требуемый для выполнения работы теоретический материал. Результаты лабораторной работы оформляются в виде отчета. При защите работы студент показывает преподавателю отчет, демонстрирует работу соответствующей программы, отвечает на вопросы.

Для выполнения лабораторных работ требуется программное обеспечение. В начале семестра обучающиеся приобретают навыки создания баз данных и приложений для доступа к базам данных в среде СУБД Access. Во второй половине семестра обучающиеся знакомятся с удаленными СУБД. В компьютерных классах установлено свободное ПО – СУБД PostgreSQL. В домашних условиях обучающиеся также могут воспользоваться этой СУБД или свободно-распространяемой СУБД Firebird.

**3 Методические указания по выполнению курсового проекта**

Целью выполнения курсового проекта по дисциплине "Базы данных" является приобретение навыков проектирования и реализации баз данных для решения практических задач в области обработки данных с использованием современных СУБД, а также закрепление и расширение знаний, полученных при изучении теоретического материала. При выполнении курсового проекта студент должен показать умение анализировать предметную область, и на практике применить теоретические знания в области проектирования и реализации реляционных баз данных.

Курсовой проект выполняется по актуальным вопросам систем обработки данных согласно индивидуального задания и оформляется в виде пояснительной записки и программной реализации с использованием современных СУБД. Основой для успешного выполнения курсового проекта является глубокое изучение учебной и научно-технической литературы.

Курсовой проект – самостоятельная творческая завершенная работа студента, выполненная под руководством преподавателя. Она обычно состоит из двух частей: теоретической и практической. Практической частью работы по дисциплине «Базы данных» является разработанная база данных, а в теоретической части оформляется отчет. Основные цели проекта: развитие активной творческой личности, способной самостоятельно приобретать новые знания, развитие навыков самостоятельной исследовательской работы у студентов.

Выполнение курсового проекта состоит из нескольких этапов:

1. Проектирование базы данных на основе нисходящего подхода
2. Реализация базы данных в среде целевой СУБД
3. Обеспечение семантической и реляционной целостности декларативным и процедурным путем
4. Заполнение базы данных
5. Реализация запросов

Этапы курсового проекта выполняются последовательно, после изучения соответствующих тем дисциплины. При этом рекомендуется придерживаться следующего порядка работы:

1. Ознакомиться с заданием этапа и индивидуальной темой проекта.

2. Исследовать и проанализировать возможности выполнения этапа, оценить требуемые знания и навыки.

3. Выработать несколько идей для осуществления этапа и выбрать наиболее удачную.

4. Выполнить практическую часть этапа.

5. Провести самооценку полученного результата в соответствии с предложенными критериями.

Защита курсового проекта студентом осуществляется после завершения последнего этапа. Отчет предоставляется в текстовом файле и в распечатанном виде, разработанная БД – в электронном виде.

Студенту необходимо охарактеризовать основные этапы работы, обобщить собственную проектную деятельность, ответить на поставленные вопросы преподавателя в рамках темы курсового проекта.

Критерии оценки отчета:

– оценка «5» – отчет составлен в полном соответствии с требованиями;

– оценка «4» – отчет составлен в полном соответствии с требованиями, имеются частичные замечания по оформлению документа;

– оценка «3» –имеются неточности и замечания при оформлении документа;

– оценка «2» – отчет не соответствует требованиям.

**4 Методические указания по промежуточной аттестации по дисциплине**

Изучение дисциплины «Базы данных» в 3 семестре завершается зачетом. Для допуска к зачету необходимо представить отчет по лабораторным работам. Отчет по лабораторным работам описывает процесс создания базы данных и приложения для работы с базой данных в СУБД Access.

Изучение дисциплины «Базы данных» в четвертом семестре завершается экзаменом. Для допуска к экзамену необходимо представить отчет по лабораторным работам. Отчет по лабораторным работам должен содержать описание этапов проектирования реляционной базы данных по нисходящему подходу. Кроме того, необходимо защитить курсовой проект.

Подготовка к экзамену способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к экзамену, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На экзамене студент демонстрирует то, что он приобрел в процессе обучения по данной учебной дисциплине.

В период подготовки к экзамену студенты вновь обращаются к пройденному учебному материалу. При этом они не только скрепляют полученные знания, но и получают новые. Подготовка студента к экзамену включает в себя три этапа:

-самостоятельная работа в течение семестра;

-непосредственная подготовка в дни, предшествующие экзамену по темам курса;

-подготовка к ответу на вопросы, содержащиеся в методическом комплексе.

Подготовка к экзамену осуществляется на основании методических рекомендаций по дисциплине и списка вопросов изучаемой дисциплины, конспектов лекций, учебников и учебных пособий, научных статей, информации среды интернет.

Студент должен иметь хороший собственный конспект лекций. Даже в том случае, если была пропущена какая-либо лекция, необходимо во время ее восстановить, обдумать, снять возникшие вопросы для того, чтобы запоминание материала было осознанным.

Кроме того, при подготовке к экзамену у студента должен быть хороший учебник или конспект литературы, прочитанной по указанию преподавателя в течение семестра.

Вначале следует просмотреть весь материал по дисциплине, отметить для себя трудные вопросы. Обязательно в них разобраться. В заключение еще раз целесообразно повторить основные положения.

На правах рукописи

Минобрнауки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Оренбургский государственный университет»**

Кафедра прикладной информатики в экономике и управлении

Методические указания по лабораторным работам для обучающихся

по дисциплине

*«Б.1.Б.16 Базы данных»*

Уровень высшего образования

БАКАЛАВРИАТ

Направление подготовки

*38.03.05 Бизнес-информатика*

(код и наименование направления подготовки)

*Информационные системы в экономике*

(наименование направленности (профиля) образовательной программы)

Тип образовательной программы

*Программа академического бакалавриата*

Квалификация

*Бакалавр*

Форма обучения

*Очная*

Год набора 2022

Составители Панова Н.Ф

Методические указания рассмотрены и одобрены на заседании кафедры прикладной информатики в экономике и управлении

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Жук М.А.

Методические указания является приложением к рабочей программе по дисциплине «Базы данных», зарегистрированной в ЦИТ под учетным номером\_\_\_\_\_\_\_\_\_\_\_

Оглавление

[2 Лабораторные работы 15](#_Toc69761714)

[2.1 Лабораторная работа №1. Сравнительный анализ СУБД 15](#_Toc69761715)

[2.2 Лабораторная работа №2. Основы работы в среде СУБД ACCESS. Возможности системы 15](#_Toc69761716)

[2.3 Лабораторная работа №3. Создание БД в среде СУБД ACCESS в режиме мастера и конструктора 17](#_Toc69761717)

[2.4 Лабораторная работа №4. Создание форм для ввода и редактирования данных. Связанные и подчиненные формы 18](#_Toc69761718)

[2.5 Лабораторная работа №5. Создание отчетов 19](#_Toc69761719)

[2.6 Лабораторная работа №6. Реализация запросов в среде СУБД ACCESS. Параметрические запросы. Использование обобщающих функций в запросах 21](#_Toc69761720)

[2.7 Лабораторная работа №7 Разработка приложения для работы с БД (главная форма) 24](#_Toc69761721)

[2.8 Лабораторная работа №8. Знакомство с СУБД PostgreSQL. Установка. Создание новых пользователей в PSQL. Создание БД. Управление правами доступа 28](#_Toc69761722)

[2.9 Лабораторная работа №9-10. Знакомство с программой pgAdmin 35](#_Toc69761723)

[2.10 Лабораторная работа №11. Создание объектов БД 36](#_Toc69761724)

[2.11 Лабораторная работа №12. Реализация запросов. Оператор Select 43](#_Toc69761725)

[2.12 Лабораторная работа №13 Вставка, удаление и обновление данных с помощью хранимых процедур 46](#_Toc69761726)

[2.13 Лабораторная работа №14. Обеспечение целостности данных с помощью триггеров 48](#_Toc69761727)

[Вставка, удаление и обновление осуществляется с помощью операторов манипулирования данными, описанными ниже. 52](#_Toc69761728)

[2.14-15 Лабораторная работа №14 Технология ADO.net. Подключение к БД. Разработка приложений для доступа к БД 54](#_Toc69761729)

# 2 Лабораторные работы

## 2.1 Лабораторная работа №1. Сравнительный анализ СУБД

Цель: знакомство с существующими на рынке ПО СУБД и сравнительный анализ основных характеристик нескольких СУБД.

Для выполнения работы обучающиеся должны найти информацию в сети Интернет , выбрать 3 СУБД, определить перечень характеристик, по которым будет осуществляться анализ, составить сводную таблицу и сделать выводы. По итогам выполнения работы сформировать отчет и предоставить преподавателю в распечатанном виде.

## 2.2 Лабораторная работа №2. Основы работы в среде СУБД ACCESS. Возможности системы

Microsoft Access является настольной СУБД (система управления базами данных) реляционного типа. Достоинством Access является то, что она имеет очень простой графический интерфейс, который позволяет не только создавать собственную базу данных, но и разрабатывать приложения, используя встроенные средства.

В отличие от других настольных СУБД, Access хранит все данные в одном файле, хотя и распределяет их по разным таблицам, как и положено реляционной СУБД. К этим данным относится не только информация в таблицах, но и другие объекты базы данных, которые будут описаны ниже.

Для выполнения почти всех основных операций Access предлагает большое количество Мастеров (Wizards), которые делают основную работу за пользователя при работе с данными и разработке приложений, помогают избежать рутинных действий и облегчают работу неискушенному в программировании пользователю.

Создание многопользовательской БД Access и получение одновременного доступа нескольких пользователей к общей базе данных возможно в локальной одноранговой сети или в сети с файловым сервером. Сеть обеспечивает аппаратную и программную поддержку обмена данными между компьютерами. Access следит за разграничением доступа разных пользователей к БД и обеспечивает защиту данных. Так как Access не является клиент- серверной СУБД, возможности его по обеспечению многопользовательской работы несколько ограничены. Обычно для доступа к данным по сети с нескольких рабочих станций, файл БД Access выкладывается на файловый сервер. При этом обработка данных ведется в основном на клиенте – там, где запущено приложение, в силу принципов организации файловых СУБД. Этот фактор ограничивает использование Access для обеспечения работы множества пользователей (более 15–20) и при большом количестве данных в таблицах, так как многократно возрастает нагрузка не сеть.

В плане поддержки целостности данных Access отвечает только моделям БД небольшой и средней сложности. В нем отсутствуют такие средства как триггеры и хранимые процедуры, что заставляет разработчиков возлагать поддержание бизнес логики БД на клиентскую программу.

В отношении защиты информации и разграничения доступа Access не имеет надежных стандартных средств. В стандартные способы защиты входит защита с использованием пароля БД и защита с использованием пароля пользователя. Снятие такой защиты не представляет сложности для специалиста.

Однако, при известных недостатках MS Access обладает большим количеством преимуществ по сравнению с системами подобного класса.

В первую очередь можно отметить распространенность, которая обусловлена тем, что Access является продуктом компании Microsoft, программное обеспечение и операционные системы которой использует большая часть пользователей персональных компьютеров. MS Access полностью совместим с операционной системой Windows, постоянно обновляется производителем, поддерживает множество языков.

В целом MS Access предоставляет большое количество возможностей за сравнительно небольшую стоимость. Также необходимо отметить ориентированность на пользователя с разной профессиональной подготовкой, что выражается в наличии большого количества вспомогательных средств - Мастеров, развитую систему справки и понятный интерфейс. Эти средства облегчают проектирование, создание БД и выборку данных из нее.

MS Access предоставляет в распоряжение непрограммирующему пользователю разнообразные диалоговые средства, которые позволяют ему создавать приложения не прибегая к разработке запросов на языке SQL или к программированию макросов или модулей на языке VBA.

## 2.3 Лабораторная работа №3. Создание БД в среде СУБД ACCESS в режиме мастера и конструктора

1. Разработать реляционную базу данных, состоящую из трех и более таблиц, связанных друг с другом. Полученные схемы отношений проверить у преподавателя.

2. При создании таблиц в окне конструктора использовать разнообразные типы полей - числовой, текстовый, дата и т. д. Одна из таблиц должна содержать фотографию. В таблицах предусмотреть ограничения на значения атрибутов и вывод сообщения в случае нарушения этого ограничения. Например, возраст абитуриента не должен превышать 30 лет. Указать атрибуты, которые не могут быть пустыми. Для текстовых атрибутов задать маску ввода. Например, для поля «фамилия» запретить ввод всех символов, кроме букв русского алфавита, первую букву преобразовать к верхнему регистру. Маски задать также для поля «телефон» и аналогичным полям. Для полей, имеющих ограниченный неизменяемый список значений, таких как «пол», «образование», «семейное положение» и т.д. использовать мастер подстановок из набора фиксированных значений.

3. Ключевые поля. При описании простых ключевых полей в качестве их типа выбирать тип «счетчик». Хотя бы в одной из таблиц ключевое поле описать сделать составным.

4. Внешние ключи. Для обеспечения ссылочной целостности значения всех внешних ключей не должны вводиться вручную. Эти значения должны выбираться из списка значений соответствующих первичных ключей. Поэтому в качестве типа внешнего ключа выбирается мастер подстановок значений из другой таблицы или запроса.

5. В окне схемы данных обеспечить ссылочную целостность , задав определенную стратегию, как показано на рисунке 1.

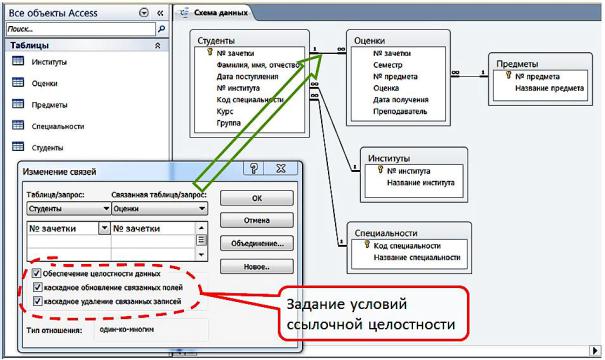


Рис.1 – Окно схемы данных

## 2.4 Лабораторная работа №4. Создание форм для ввода и редактирования данных. Связанные и подчиненные формы

В форме Access, как на витрине магазина, удобно просматривать и открывать нужные элементы. Так как форма — это объект, с помощью которого пользователи могут добавлять, редактировать и отображать данные, хранящиеся в базе данных классического приложения Access, ее внешний вид играет важную роль. В ACCESS формы можно создавать в режиме конструктора или в режиме мастера. Если форма создана в режиме мастера, ее можно открыть и изменить в окне конструктора. Можно создать форму для работы с одной таблицей или сразу с несколькими. В одной форме можно отображать содержимое 2-х и более таблиц с помощью связанных/подчиненных форм. При этом данные основной таблицы выводятся по одной записи, а в нижней части формы в табличном виде отображаются все записи подчиненной таблицы, связанные с текущей записью основной, как показано на рис.2.

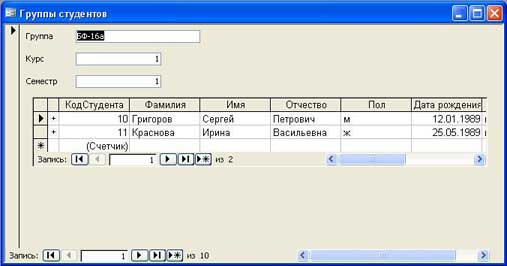


Рис.2 - Связанные формы

1. На формах стандартные кнопки навигации убрать и создать свои кнопки, как показано на рис.3.

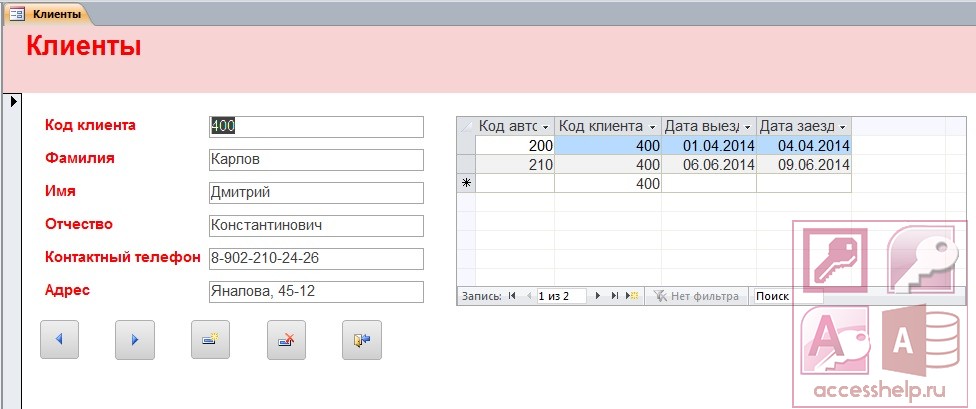


Рис.3 – Связанные формы с кнопками

## 2.5 Лабораторная работа №5. Создание отчетов

Для того чтобы иметь возможность просмотреть нужную нам информацию, возможность распечатать данные, имеющиеся в базе данных, используются отчеты. На первом этапе освоения программы Access целесообразно воспользоваться командой «Мастер отчетов» расположенной на ленте «Создать».

В процессе работы «Мастер отчетов» предложит выбрать из списка доступные поля для отчета, вид представления данных, порядок сортировки и вычисления, мает для отчета, стиль и имя нашего отчета.

На рис. 4 представлен вид простейшего отчета без группировки и подведения итогов.

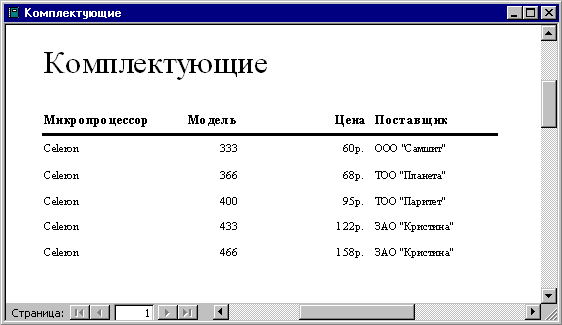


Рис. 4 – Простейший отчет

Для того чтобы улучшить внешний вид нашего отчета, а также добавить в него какие-либо итоговые данные, следует воспользоваться «Конструктором отчетов»

Так как мастер отчетов некоторые поля создал очень узкими, их можно расширить их, используя курсор мыши. А также мы можем регулировать высоту строк в отчете. Для добавления итогов используется пункт меню «Группировка и итоги», где делаются следующие шаги: выбирается поле, по которому будут группироваться данные (значения в этом поле должны повторяться, например поле студенческая группа); далее выбирается пункт «с итогами» и выбирается тип итоговой операции – подсчет количества, суммирование, нахождение среднего арифметического и т.д. Можно также обеспечить подведение общего итога, например, по всем студенческим группам подсчитать количество студентов. Кроме того, можно выбрать место, где будет размещен итог по каждой группе: в примечании группы или в заголовке группы.На рис. 5 представлен пример отчета с группировкой и итогами по каждой группе.



Рис. 5 – Отчет с итогами по полю «должность»

1. Необходимо создать простой отчет по нескольким таблицам и отчет с группировкой и итогами, также по двум и более таблицам.

## 2.6 Лабораторная работа №6. Реализация запросов в среде СУБД ACCESS. Параметрические запросы. Использование обобщающих функций в запросах

Существует несколько типов запросов: на выборку, на обновление, на добавление, на удаление, перекрестный запрос, создание таблиц. Наиболее распространенным является запрос на выборку. Запросы на выборку используются для отбора нужной пользователю информации, содержащейся в таблицах. Они создаются только для связанных таблиц. Запрос можно создать в режиме Мастера или Конструктора.

Окно Конструктора запросов состоит из двух частей – верхней и нижней. В верхней части окна размещается схема данных запроса, которая содержит список таблиц – источников и отражает связь между ними. В нижней части окна находится бланк построения запроса QBE (Query by Example), в котором каждая строка выполняет определенную функцию (рисунок 6):

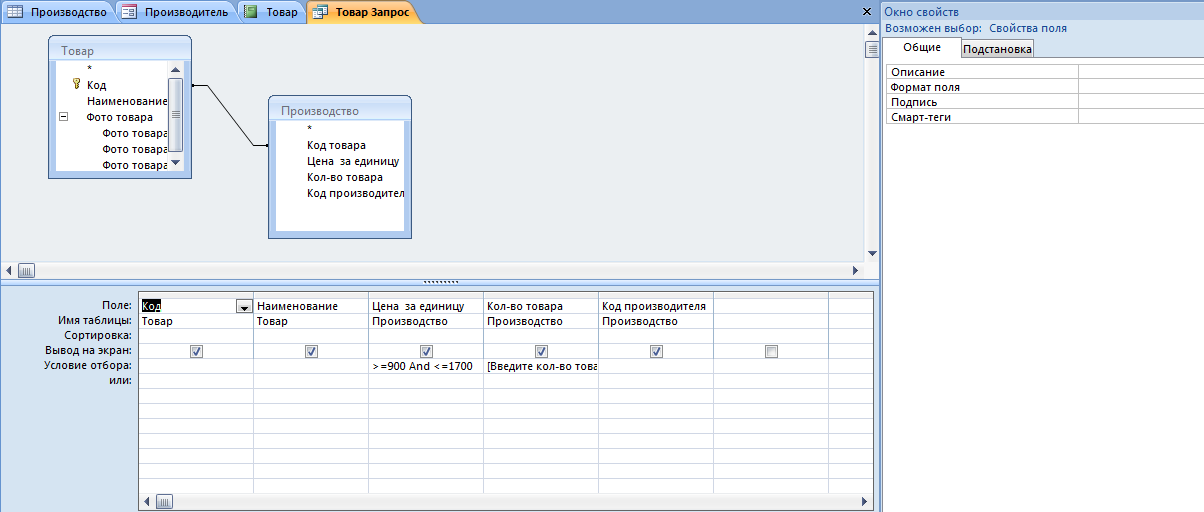


Рисунок 6– Окно конструктора запросов

* поле – указывает имена полей, которые участвуют в запросе;
* имя таблицы – имя таблицы, с которой выбрано это поле;
* сортировка – указывает тип сортировки;
* вывод на экран – устанавливает флажок просмотра поля на экране;
* условия отбора  - задаются критерии поиска;
* или – задаются дополнительные критерии отбора.

Если запрос создается с помощью Конструктора, в окне  «Запрос: запрос на выборку» с помощью инструментов формируем запрос:

* выбрать таблицу – источник, из которой производится выборка записей;
* переместить имена полей с источника в бланк запроса;
* задать принцип сортировки. Курсор мыши переместить в строку Сортировка для любого поля, появится кнопка открытия списка режимов сортировки: по возрастанию и по убыванию;
* в строке вывод на экран автоматически устанавливается флажок просмотра найденной информации в пол;
* в строке «Условия» отбора и строке «Или» необходимо ввести условия ограниченного поиска – критерии поиска;
* после завершения формирования запроса закрыть окно «Запрос» на выборку. Откроется окно диалога «Сохранить» – ответить «Да», щелкнуть ОК и вернуться в окно базы данных.

Чтобы создать запрос с помощью мастера нужно в окне база данных выбрать вкладку «Создание» и щелкнуть на пиктограмме «Мастер запросов» (рисунок 7).

В первом появившемся окне будут перечислены допустимые типы запросов, выберем простой запрос. Далее определяются поля, которые будут входить в запрос, вид отчета (итоговый или подробный) и имя запроса. Созданный простой запрос можно дополнить, усложнить или изменить с помощью Конструктора запросов.

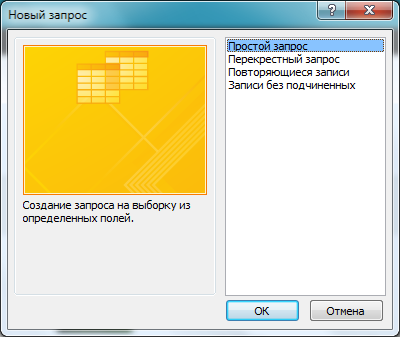


Рисунок 7– Окно выбора типа запроса

Чаще всего простой запрос преобразуется в параметрический. Для этого в стоке Условие отбора в том столбце, на которое накладывается условие, задается одна из операций отношения (=, >, < и т.д.). Если в условии задать конкретную границу, запрос будет статическим, т.е. не изменяемым. Поэтому желательно при инициации запроса вводить требуемые значения. Для этого после операции отношения указывается текст сообщения в квадратных скобках. Например =[Введите наименование товара].

Любой запрос в окне конструктора запросов можно представить в виде SQL-оператора. Для этого в контекстном меню, вызванном нажатием правой кнопки мыши на пустой области верхней части конструктора, следует выбрать команду «режим SQL». В этом режиме можно создать новый запрос или отредактировать существующий.

1. Данная лабораторная работа предполагает создание запросов на выборку, включающих следующие типы:

- параметрический запрос по нескольким таблицам;

-запрос с группировкой и условием на группы;

- запрос с итогами, включающий обобщающие функции;

- запросы с подзапросами.

## 2.7 Лабораторная работа №7 Разработка приложения для работы с БД (главная форма)

Главная кнопочная форма создается для обеспечения удобного доступа к функциям приложения, т.е. она может использоваться в качестве главного меню.

Форма должна иметь привлекательный вид. Кнопки на ней должны быть выровнены, цвета согласованы. В качестве фона формы можно использовать произвольное изображение. Элементами главной кнопочной формы являются объекты форм и отчётов.

Запросы и таблицы не являются элементами главной кнопочной формы. Поэтому для создания кнопок Запросы или Таблицы на кнопочной форме можно использовать макросы. Сначала в окне базы данных создают макросы «Открыть Запрос» или «Открыть Таблицу» с уникальными именами, а затем в кнопочной форме создают кнопки для вызова этих макросов.

Для создания главной кнопочной формы и ее элементов необходимо открыть базу данных, и в меню «Работа с базами данных» выбрать пиктограмму «Диспетчер кнопочных форм» (рисунок 8).

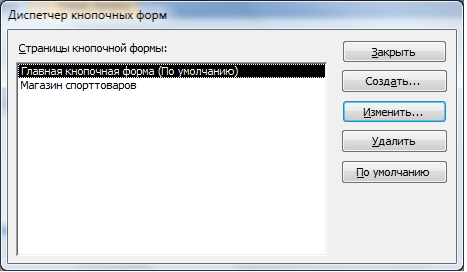


Рисунок 8 – Окно диспетчера кнопочных форм

После этого создаются элементы кнопочной формы, для этого в «Окне диспетчер кнопочных форм» выделяется страница «Главная кнопочная форма» и с помощью кнопки «Изменить» открывается новое окно «Изменение страниц кнопочной формы». Щелчком кнопки мыши на кнопке «Создать» открывается новое окно «Изменение элемента кнопочной формы», в котором выполняются следующие действия:

* вводится текст: Формы для ввода данных;
* из раскрывающегося списка выбирается команда: Перейти к кнопочной форме;
* выбирается из списка кнопочная форма: Ввод данных в формы.

В окне «Изменение страницы кнопочной формы» отобразится элемент кнопочной формы «Формы для ввода данных» (рисунок 9).

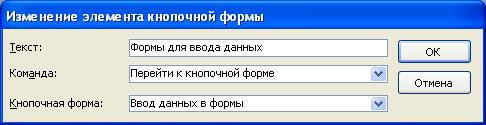


Рисунок 9 – Окно для изменения элементов кнопочной формы

В результате в окне «Изменение страницы кнопочной формы» будут отображаться все элементы главной кнопочной формы (рисунок 10).

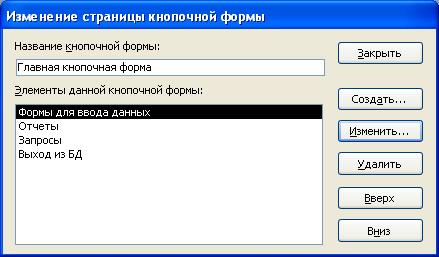


Рисунок 10 – Окно изменения страниц кнопочной формы

На рисунке 11 представлена кнопочная форма в режиме выполнения.

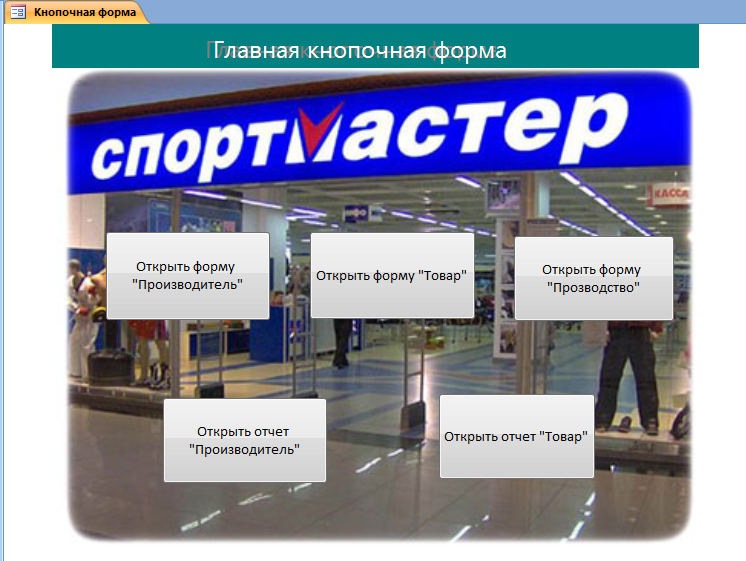


Рисунок 11 – Главная кнопочная форма в режиме выполнения

Отчет по лабораторным работам должен содержать соответствующие пункты:

- таблицы;

- формы;

-отчеты;

-запросы;

-кнопочная форма.

В пункте «таблицы» описать тиры полей и ограничения, а также маски ввода. Следует описать, каким образом обеспечена ссылочная целостность данных, какие стратегии выбраны.

В пункте «формы» представить скриншоты экранных форм. Формы должны обеспечивать доступ ко всем созданным таблицам. Доступ к связанным таблицам осуществляется с помощью связанных форм.

Аналогично описываются отчеты.

Кнопочная форма должна обеспечивать доступ ко всем созданным объектам базы данных. Создать ее можно с помощью диспетчера кнопочных форм или с помощью конструктора.

## 2.8 Лабораторная работа №8. Знакомство с СУБД PostgreSQL. Установка. Создание новых пользователей в PSQL. Создание БД. Управление правами доступа

Postgre SQL – объектно-реляционная система управления базами данных, разработка которой ведется начиная с 1977 года. Работа началась с проекта Ingres в Калифорнийском университете (Беркли). В настоящее время над проектом Postgre SQL активно роаботает группа разработчиков со всего мира. Postgre SQL распаространяется под свободной лицензией и с открытым исходным кодом.

Имя суперпользователя - Postgres. Пароль задается при установке.

В окне установщик предлагает указать порт TCP/IP для PostgreSQL. В большинстве случаев можно оставить тот, который предлагается по умолчанию.



На рисунке представлено окно выбора локали, т.е. кодировки БД. Локаль Russian, Russian определяет кодировку windows – 1251.

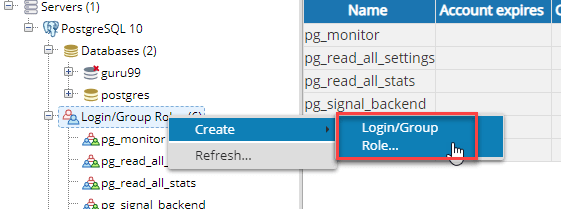


После определения указанных параметров начинается процесс копирования файлов в определенный ранее каталог установки PostgreSQL. Необходимо обратить внимание на слова в процессе установки СУБД: «Initialising database cluster» (инициализация кластера баз данных), означающие, что копирование файлов закончено и создается кластер баз данных, содержащий базу данных, которая будет использоваться как шаблон для вновь создаваемых баз. Через некоторое время эта надпись сменится на «starting database server», что означает запуск службы сервера. Если запустить «Диспетчер задач» и перейти на вкладку «Службы», можно убедиться, что СУБД запущена.

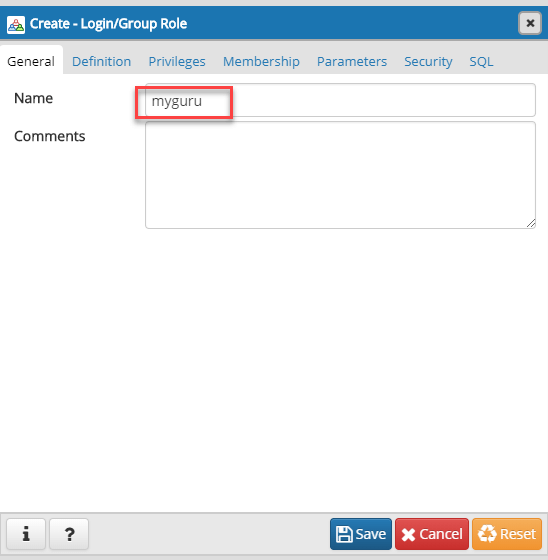
- область отчетов – отображает различные виды отчетов о выбранном объекте СУБД PostgreSQL.

**Создание новых пользователе**

**Шаг 1)** Щелкните правой кнопкой мыши на Login / Group Role -> Create -> Click на Login / Group Role…

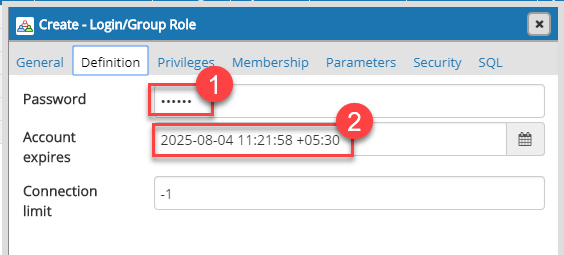


**Шаг 2)** Введите имя для входа



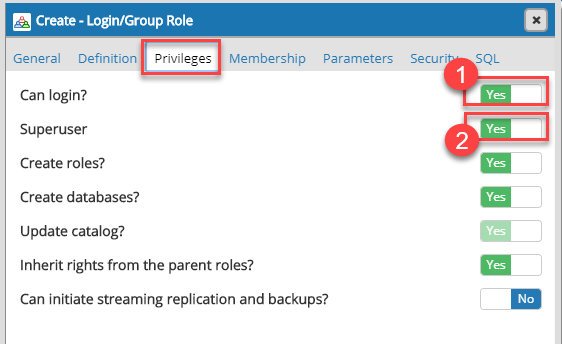
**Шаг 3)** В области определения

1. Введите пароль
2. Дата истечения срока действия аккаунта



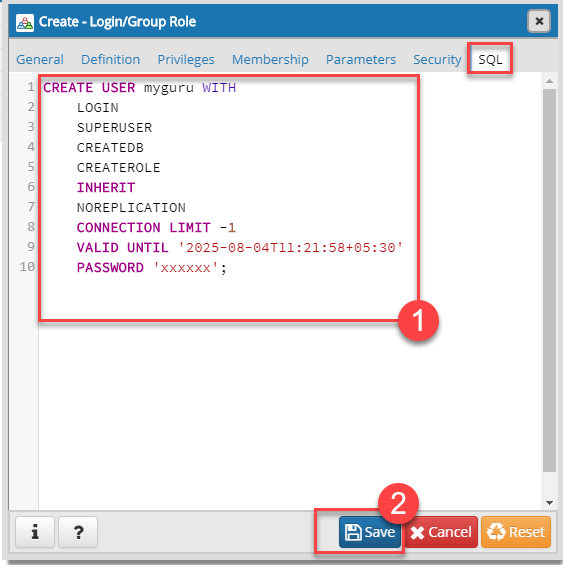
**Шаг 4)** В разделе привилегий

1. Переключить кнопку «Войти» на «Да»
2. Переключить суперпользователя на ДА

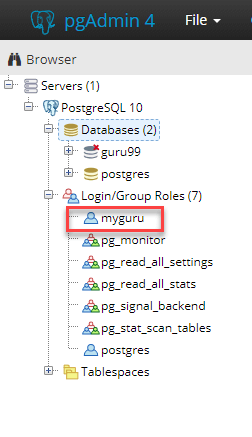


**Шаг 5)** В разделе SQL

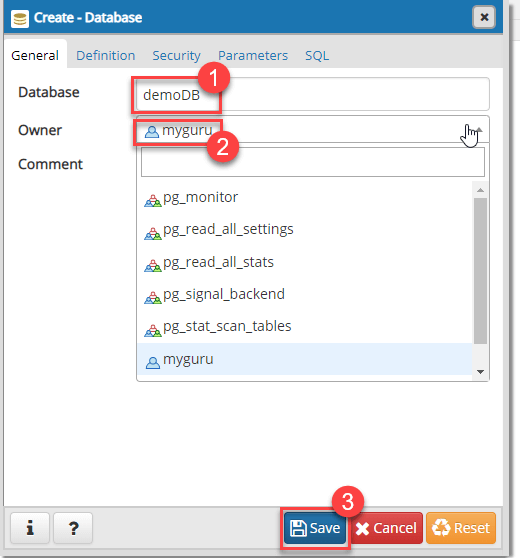
1. Вы увидите запрос SQL для создания пользователя в соответствии с выборами, сделанными в предыдущих шагах
2. Нажмите кнопку Сохранить



**Шаг 6)** Роль создана и отражена в дереве объектов



**Шаг 7)** Создайте базу данных и назначьте на нее владельца myguru.



Вы можете создать пользователя с помощью командной строки

СОЗДАТЬ ПОЛЬЗОВАТЕЛЯ

В PostgreSQL этот метод более удобен для программистов и администраторов, поскольку они имеют доступ к консоли сервера PostgreSQL. Более того, их необходимо выполнить с помощью одной команды вместо входа в систему и использования интерфейса клиента PostgreSQL.

Syntax:

CREATE USER name WITH option

where the option can be:

|SUPERUSER | NOSUPERUSER

| CREATEROLE | NOCREATEROLE

| CREATEDB | NOCREATEDB

| INHERIT | NOINHERIT

| LOGIN | NOLOGIN

| REPLICATION | NOREPLICATION

| BYPASSRLS | NOBYPASSRLS

| CONNECTION LIMIT

| ( ENCRYPTED ] PASSWORD 'password.'

| VALID UNTIL 'timestamp1

| IN ROLE role\_name [, ...J

| IN GROUP role\_name [, ...]

| ROLE role\_name [, ...]

| ADMIN role\_name [, ...)

| USER role\_name [, ...]

| SYSID uid

Пример:

CREATE USER tom;

создаст пользователя Том

CREATE USER tom WITH SUPERUSER;

Создаст пользовательский том с привилегиями суперпользователя. Давайте посмотрим на следующий пример.

**Шаг 1)** Мы создаем суперпользователя сроком действия до 3 — го Apri 2025 11:50:38 IST. Введите следующую команду

CREATE USER mytest WITH

LOGIN

SUPERUSER

CREATEDB

CREATEROLE

INHERIT

NOREPLICATION

CONNECTION LIMIT -1

VALID UNTIL '2025-04-03T11:50:38+05:30'

PASSWORD '123456';

## <https://coderlessons.com/wp-content/uploads/images/gur/403aa28507fc5ba06ea8c7716b2c444a.png>

## 2.9 Лабораторная работа №9-10. Знакомство с программой pgAdmin

Администрирование СУБД PostgreSQL может осуществляться через графическое приложение pgAdmin III. Использование подобных средств облегчает администрирование за счет наличия привычного для неподготовленного пользователя графического интерфейса. Далее предсчтавлено основное окно pgAdmin. Оно содержит несколько областей:

- браузер объектов – древовидная структура, содержащая основные объекты сервера. Браузер позволяет просматривать, сохдавать, редактировать и удалять различные объекты: базы данных, таблицы и т.д.

- панель SQL - отображает представление выбранного объекта PostgreSQL в виде SQL-запроса для повышения наглядности.

область отчетов – отображает различные виды отчетов о выбранном объекте СУБД PostgreSQL.





## 2.10 Лабораторная работа №11. Создание объектов БД

Современные СУБД поддерживают такие объекты как домены, таблицы, хранимые процедуры, триггеры, роли.

Домен – множество допустимых значений атрибутов. Доменом также называют пользовательский тип, создаваемый для нескольких атрибутов. Общий вид команды создания домена:

create domain <имя>

as

<тип> [not null] [check (value <условие>)];

Общая структура оператора создания таблицы:

create table <имя>

(<описание столбца 1>,

….

<описание столбца n>,

[<ограничения таблицы>] )

Описание столбца:

<имя> <тип> [not null] [unique]

[check <условие>] [primary key]

или

<имя> computed by <выражение> (используется для вычисляемых столбцов)

Фразу primary key  можно задать тогда и только тогда, когда первичный ключ простой, то есть состоит только из одного этого столбца.

Фраза check <условие> обеспечивает проверку логического ограничения, определенного для столбца в заданной предметной области

В качестве типа столбца можно использовать имя домена.

В ограничениях таблицы задаются составные первичные ключи, внешние ключи, стратегии обеспечения ссылочной целостности, ограничения семантической целостности, требующие проверки значений нескольких столбцов таблицы.

Primary key (<список столбцов ПК>)

Foreign key (<список столбцов ВК>)

references <имя основной таблицы> (<список столбцов ПК основной таблицы>) on delete <стратегия> on update <стратегия>

В качестве стратегий можно задать следующие:

* cascade. Каскадное удаление строк подчиненной таблицы, связанных со строкой, удаляемой из основной таблицы или каскадное обновление столбцов внешнего ключа в строках подчиненной таблицы, связанных с теми строками основной, в которых происходит обновление первичного ключа;
* set null подразумевает, что столбцам внешнего ключа будет присвоено значение null при удалении строк из основного отношения или обновлении первичного ключа в строках основного отношения;
* set default означает, что столбцам внешнего ключа будет присвоено значение по умолчанию при удалении строк из основного отношения или обновлении первичного ключа в строках основного отношения;
* restrict означает запрет на удаление строк или обновление первичного ключа в основном отношении, если им в подчиненном отношении соответствуют некоторые строки.

Если при создании таблицы не задана ни одна стратегия, возможно, ссылочная целостность обеспечена процедурным путем, т.е. созданы необходимы триггеры. Если нет, то СУБД будет придерживаться стратегии Restrict.

SQL - скрипты выполняются в режиме интерпретации, поэтому сначала создаются основные таблицы, а потом – подчиненные, которые ссылаются на основные.

В качестве примера рассмотрим предметную область «Морской порт», инфологическая модель которой представлена на рисунке 14.

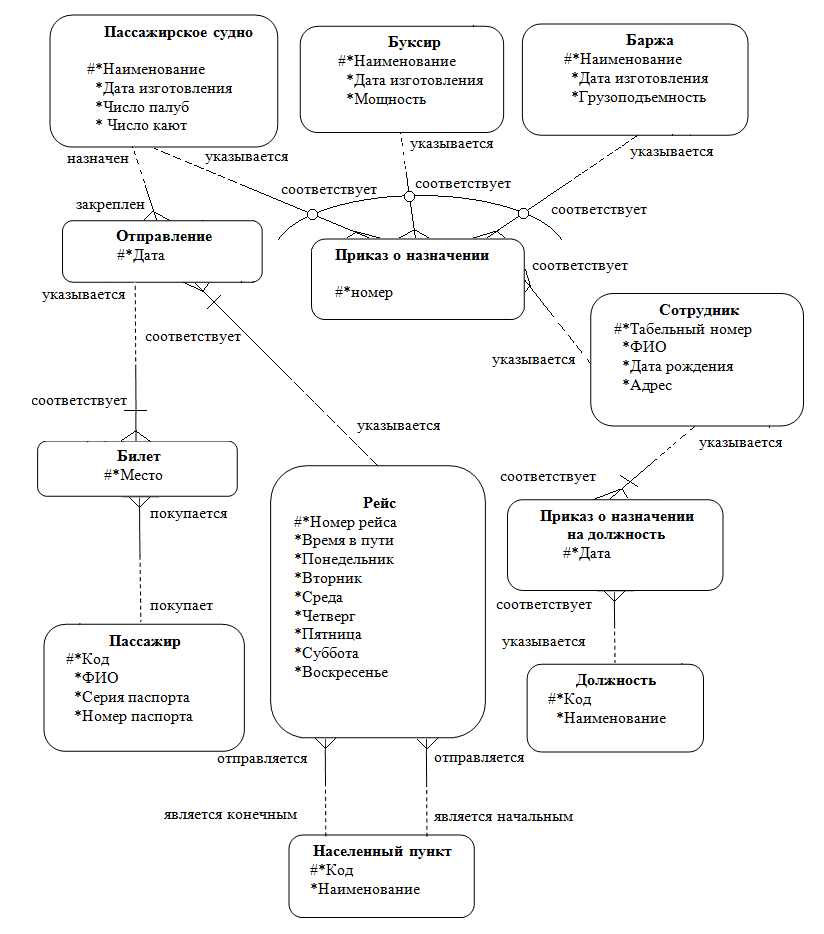


Рисунок 14 – Инфологическая модель предметной области

Преобразуем ER-диаграмму в даталогическую модель реляционной базы данных, добавляя столбцы внешних ключей в подчиненные отношения:

Баржа (наименование, дата изготовления, грузоподъемность)

Буксир (наименование, дата изготовления, мощность)

Пассажирское судно (наименование, дата изготовления, количество кают, количество палуб)

Населенный пункт (код пункта, наименование)

Пассажир (код пассажира, ФИО, серия паспорта, номер паспорта)

Должность (код должности, наименование)

Сотрудник (табельный номер, ФИО, дата рождения, адрес)

Приказ о назначении должности (дата, код должности, табельный номер сотрудника)

Приказ о прикреплении к судну (номер, дата, табельный номер сотрудника, наименование пассажирского судна, наименование баржи, наименование буксира)

Отправление (дата, номер рейса, наименование пассажирского судна)

Билет (место, дата рейса, номер рейса, код пассажира)

Строка расписания рейсов (номер рейса, время в пути, понедельник, вторник, среда, четверг, пятница, суббота, воскресенье, код пункта отправки, код пункта прибытия)

Напишем SQL-скрипты для создания доменов и таблиц:

create domain code as integer not null check (value>0);

create domain stroka as char (100) not null;

create table Barja

(Name stroka Primary key,

Data\_izg Date not null check(Data\_izg< current\_date),

Gruz integer not null check(Gruz>0));

create table Buksir

(Name stroka Primary key,

Data\_izg Date not null check(Data\_izg< current\_date),

Mos integer not null check(Mos>0));

create table Pas\_sudno

(Name stroka Primary key,

Data\_izg Date not null check(Data\_izg< current\_date),

Kol\_vo\_palub integer not null check(Kol\_vo\_palub>0),

Kol\_vo\_kayut integer not null check(Kol\_vo\_kayut>0));

create table Sotrudnik

(Tab\_nomer Code Primary key,

FIO Stroka,

Dr Date not null check((Dr<'01.01.1996') and (Dr>'01.01.1910')),

Adres Stroka);

create table Dolgnost

(Kod Code Primary key, Name Stroka);

create table Pas

(Kod Code Primary key,

FIO Stroka, Seriya\_pasporta integer not null,

Nomer\_pasporta integer not null);

create table Gorod

(Kod Code Primary key, Name Stroka);

create table Stroka\_raspisaniya

(Nomer\_reisa Code Primary key,

Vremya\_v\_puti Stroka,

Ponedelnik integer not null, Vtornik integer not null,

Sreda integer not null, Chetverg integer not null, Pyatnica integer not null,

Subbota integer not null, Voskresenye integer not null,

Code\_otprav Code, Code\_pribt Code,

Foreign key (Code\_otprav) References Gorod (Kod) on update cascade,

Foreign key (Code\_pribt) References Gorod (Kod) on update cascade);

create table Prikaz2

(Nomer Code Primary key,

Data Date not null check((Data< current\_date) and (Data>'01.01.1910')),

Tab\_nomer Code, Name\_sudno\_Pas char(15),

Name\_sudno\_Barja char(15), Name\_sudno\_Buksir char(15),

Foreign key (Tab\_nomer) References Sotrudnik (Tab\_nomer)

on update cascade,

Foreign key (Name\_sudno\_Barja) References Barja (Name)

on update cascade,

Foreign key (Name\_sudno\_Buksir) References Buksir (Name)

on update cascade,

Foreign key (Name\_sudno\_Pas) References Pas\_sudno (Name)

on update cascade);

create table Otpravlenie

(Nomer\_reisa code, Data Date not null, Name\_pas\_sudno stroka,

Primary key (Nomer\_reisa, Data),

Foreign key (Name\_pas\_sudno) References Pas\_sudno (Name)

on update cascade,

Foreign key (Nomer\_reisa) References Stroka\_raspisaniya (Nomer\_reisa)

on update cascade);

create table Bilet

(Nomer\_reisa integer not null check(Nomer\_reisa>0),

Data Date check((Data<'01.01.2016') and (Data>'01.01.1910')),

Mesto integer not null check(Mesto>0),

Kod\_Pas Code, primary key (Mesto, Data, Nomer\_reisa),

Foreign key (Kod\_Pas) References Pas (Kod) on update cascade,

Foreign key (Nomer\_reisa, Data)

References Otpravlenie (Nomer\_reisa, Data) on update cascade);

create table Prikaz1

(Data Date not null check((Data< current\_date) and (Data>'01.01.1910')),

Nomer\_dol Code, Tab\_nomer Code,

Primary key (Data, Nomer\_dol, Tab\_nomer),

Foreign key (Nomer\_dol) References Dolgnost (Kod)

on update cascade,

Foreign key (Tab\_nomer) References Sotrudnik (Tab\_nomer)

on update cascade);

Далее при рассмотрении операторов манипулирования данными, оператора запросов select, хранимых процедур и триггеров будут использоваться примеры, основанные на базе данных морского порта.

## Лабораторная работа №12. Реализация запросов. Оператор Select

Запросы к реляционной базе данных реализуются с помощью оператора select. Данный оператор реализует всевозможные запросы к БД. Один оператор select заключает в себе все операции реляционной алгебры.

Общий вид оператора select:

Select [all | distinct] \* | < список столбцов >

From < список таблиц >

[ where < условие выборки или соединения > ]

[ group by < список группируемых столбцов >

[ having < условие отбора групп > ] ]

[ order by < список сортируемых столбцов > ]

Примеры:

1. выборка всех строк и столбцов. Вывести всю информацию о сотрудниках

Select \*

from Sotrudnik;

1. выборка отдельных столбцов. Вывести фамилии сотрудников.

Select FIO

from Sotrudnik;

1. вывести фамилии сотрудников моложе 30

Select FIO

from Sotrudnik

where Data < ’01.01.1983’;

1. вывести фамилии сотрудников, рожденных в 1985 г.

Select FIO

from Sotrudnik

where Data between ’01.01.1985’ and ’31.12.1985’;

1. вывести адреса сотрудников Иванова, Петрова и Сидорова

Select Adres

from Sotrudnik

where FIO in (‘Иванов’, ‘Петров’, ‘Сидоров’);

1. вывести фамилии сотрудников, проживающих в г. Орск

Select FIO

from Sotrudnik

where Adres like ‘% г. Орск %’;

1. Подсчитать количество барж и их суммарную грузоподъемность, изготовленных за каждый день сентября 2012 года.

Select count (\*) as kolvo,

sum (Gruz) as gpod, Data\_izg

from Barja

where Data\_izg between ’01.09.2012’ and ’30.09.2012’

group by Data\_izg;

1. Вывести сведения о баржах, упорядочив их по дате изготовления, а при одинаковых значениях дат – по грузоподъемности.

Select \* from Barja

order by Data\_izg , Gruz;

1. Вывести фамилии сотрудников, на которых не был издан ни один приказ в сентябре 2012.

Select s.FIO

from Sotrudnil s,

where not exists

(select p.\* from Prikaz2 p

where p.Data between '01.09.2012' and '30.09.2012'

and p.Tab\_nomer = s.Tab\_nomer);

1. Вывести те даты, в которые в общей сложности были изготовлены баржи с наибольшей грузоподъемностью.

Select Data\_izg

from Barja

group by Data\_izg

having sum (Gruz) > = all

(select sum (Gruz) from Barja

group by Data\_izg );

1. Вывести полную информацию о назначении сотрудников на пассажирское судно с указанием наименований пассажирского судна, а также фамилий сотрудников.

Select s . FIO, p . Data, p . Tab\_nomer, p. Name\_sudno\_Pas

From Sotrudnik s, Prikaz2 p

where p . Tab\_nomer = s . Tab\_nomer;

1. Подсчитать количество приказов, изданных в 2012 году, в которых упоминался сотрудник Иванов.

Select count (\*) as kol

from Sotrudnik s, Prikaz2 p

where p.Tab\_nomer = s. Tab\_nomer

and s. FIO = ’Иванов’ and p. Data between ’01.01.2012’ and ’31.12.2012’;

## 2.12 Лабораторная работа №13 Вставка, удаление и обновление данных с помощью хранимых процедур

Хранимые процедуры – это объекты базы данных, которые представляют собой программу, написанную на языке, встроенном в СУБД.

Хранимые процедуры выполняются на сервере и используются в режиме «клиент-сервер» для проверки сложных ограничений целостности, для реализации запросов, а также для выполнения операций вставки, удаления и обновления.

Они содержат управляющие конструкции - циклы, ветвление и операторы языка SQL. Процедуры могут иметь входные и выходные параметры. Создается процедура командой:

CREATE [ OR REPLACE ] PROCEDURE ***имя***

( [ ***режим\_аргумента*** ] [ ***имя\_аргумента*** ] ***тип\_аргумента*** ] )

LANGUAGE ***имя\_языка***

AS

'***определение***’

Команда CREATE PROCEDURE определяет новую процедуру.  CREATE OR REPLACE PROCEDURE создаёт новую процедуру либо заменяет определение уже существующей.

**Имя  *-*** имя создаваемой процедуры;

**режим\_аргумента *-*** режим аргумента: IN (входной) или INOUT  (выходной). По умолчанию подразумевается IN.

**тип\_аргумента  *-*** тип данных аргумента процедуры

**имя\_языка *-*** имя языка, на котором реализована функция. Это может быть sql, c, internal либо имя процедурного языка, определённого пользователем, например, plpgsql.

**Определение  *-*** строковая константа, определяющая реализацию процедуры; её значение зависит от языка. Это может быть имя внутренней процедуры, путь к объектному файлу, команда SQL или код на процедурном языке.

Часто бывает полезно заключать определение процедуры в доллары, а не в традиционные апострофы. Если не использовать доллары, все апострофы и обратные косые черты в определении процедуры придётся дублировать.

Рассмотрим в качестве примера хранимую процедуру вставки новой строки в таблицу Student, имеющую структуру Student(kod, fam, dat).

Create procedure Add(pkod integer, pfam char(30), pdat date)

Language SQL

As

$$

Insert into student values (pkod,pfam,pdat);

$$

Процедура, реализующая обновление строки в этой таблице, имеет вид:

Create procedure Upd(pkod integer, pfam char(30), pdat date)

Language SQL

As

$$

Update table student set fam=pfam, dat=pdat where kod=pkod;

$$

Процедура удаления строки из таблицы имеет вид:

Create procedure Del(pkod integer)

Language SQL

As

$$

Delete from student where kod=pkod;

$$

Вызов ХП осуществляется с помощью следующего оператора:

Call <имя процедуры> (<список фактических параметров>)

Call Add(3, ‘Самохин’, ’12.12.2001’)

Объявляются переменные в теле процедуры с помощью конструкции:

declare <имя переменной> <тип>;

Для объявления нескольких переменных (даже одного типа) необходимо несколько таких конструкций:

declare a integer;

declare b integer;

Список фактических и входных параметров должны совпадать по количеству, типу и порядку следования.

## 2.13 Лабораторная работа №14. Обеспечение целостности данных с помощью триггеров

Триггер - это особый вид хранимой процедуры, которая выполняется автоматически при вставке, удалении или модификации записи таблицы или представления (view).

Триггеры могут "срабатывать" непосредственно до или сразу же после указанного события. Триггеры широко используются для обеспечения ссылочной и семантической целостности, реализации операций в связанных таблицах.

Создается триггер следующей командой:

Create trigger имя {before | after | INSTEAD OF insert|update|delete ON table\_name

[ FOR [ EACH ] { ROW | STATEMENT } ]

[ WHEN ( ***условие*** ) ] EXECUTE PROCEDURE ***имя\_функции*** ( ***аргументы*** )

Триггер можно настроить так, чтобы он срабатывал до операции со строкой (до проверки ограничений и попытки выполнить INSERT, UPDATE или DELETE) или после её завершения (после проверки ограничений и выполнения INSERT, UPDATE или DELETE), либо вместо операции (при добавлении, изменении и удалении строк в представлении).

Триггер с пометкой FOR EACH ROW вызывается один раз для каждой строки, изменяемой в процессе операции. Например, операция DELETE, удаляющая 10 строк, приведёт к срабатыванию всех триггеров ON DELETE в целевом отношении 10 раз подряд, по одному разу для каждой удаляемой строки.

 Триггер с пометкой FOR EACH STATEMENT, напротив, вызывается только один раз для конкретной операции, вне зависимости от того, как много строк она изменила.

Кроме того, в определении триггера можно указать логическое условие WHEN, которое определит, вызывать триггер или нет. В триггерах на уровне строк условия WHEN могут проверять старые и/или новые значения столбцов в строке.

Триггер сработает ДО того, как вставка физически осуществлена. Поэтому данные, присланные на вставку, еще не занесены в таблицу. Они находятся в некотором промежуточном буфере. И у триггера есть возможность обращаться к этому буферу с помощью контекстной переменной NEW

NEW. <имя поля>

Кроме контекстной переменной NEW, существует ее зеркальный аналог - переменная OLD.

В отличие от NEW, OLD содержит старые значения записи, которые удаляются или изменяются.

В триггере INSERT можно использовать только NEW.col\_name.

В триггере UPDATE вы можете использовать OLD.col\_name для ссылки на столбцы строки перед ее обновлением и NEW.col\_name для ссылки на столбцы строки после ее обновления.

В триггере DELETE может использоваться только OLD.col\_name; нового ряда нет.

Команда RAISE предназначена для инициирования ошибок и исключений в функциях PL/pgSQL. Она передает заданную информацию механизму PostgreSQL elog (стандартное средство ведения протокола ошибок – данные обычно направляются в файл /var/log/messages или $PGDATA/serverlog с одновременным выводом в поток stderr).

В команде RAISE также указывается уровень ошибки и строка, передаваемая PostgreSQL. Кроме того, в команду можно включить переменные и выражения, значения которых будут содержаться в выходных данных. Соответствующие позиции строки помечаются знаками процента (**%**).

Допустимые уровни ошибок

|  |  |
| --- | --- |
| **Значение** | **Описание** |
| **DEBUG** | направляет заданный текст в виде сообщения DEBUG: в журнал PostgreSQL и клиентской программе, если клиент подключен к кластеру базы данных, работающему в отладочном режиме. Команды RAISE уровня DEBUG игнорируются базами данных, работающими в режиме реальной эксплуатации |
| **NOTICE** | направляет заданный текст в виде сообщения NOTICE: в журнал PostgreSQL и клиентской программе. Сообщение передается в любом режиме работы PostgreSQL |
| **EXCEPTION** | направляет заданный текст в виде сообщения ERROR: в журнал PostgreSQL и клиентской программе. Ошибка уровня EXCEPTION также вызывает откат текущей транзакции |

Пример:

RAISE NOTICE 'Вызов функции cs\_create\_job(%)', v\_job\_id;

Здесь символ % будет заменён на значение v\_job\_id

Триггерные функции создаются следующей командой:

CREATE OR REPLACE FUNCTION имя()

RETURNS trigger

AS

$$

BEGIN

Тело функции

RETURN NEW;

END;

$$

LANGUAGE 'plpgsql';

Пример триггерной функции, которая инициирует исключение, если значение атрибута отрицательно.

CREATE OR REPLACE FUNCTION tr\_update()

RETURNS trigger

AS

$$

BEGIN

If (new.kod<0) then raise exception ‘Отрицательный код!’);

END IF;

RETURN NEW;

END;

$$

LANGUAGE 'plpgsql';

Команда создания триггера:

CREATE TRIGGER stud\_update

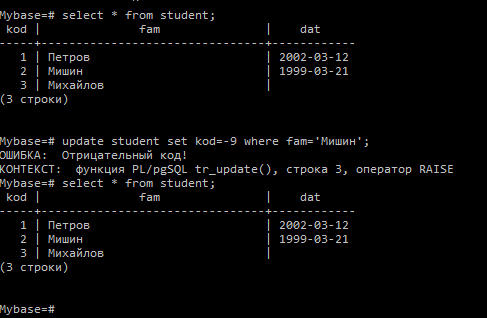
BEFORE UPDATE

ON student

FOR EACH ROW

EXECUTE PROCEDURE tr\_update();

Результат выполнения триггера:



## Вставка, удаление и обновление осуществляется с помощью операторов манипулирования данными, описанными ниже.

Insert into <имя таблицы>

[(<список столбцов>)]

values

(<список выражений>)

Оператор осуществит вставку новой строки в таблицу. Если список столбцов опущен, подразумевается все столбцы таблицы. Значения выражений в порядке следования присваиваются соответствующим столбцам

Если тип выражения не совпадает с типом столбца, возникает ошибка. Если в списке столбцов заданы не все столбцы, пропущенные столбцы примут значения null.

Пример:

Barja (name, data\_izg , gruz )

1) insert into Barja

values

('Fortune', '15.06.1967', 15000);

2) insert into Barja

values

('Riga', '22.09.1998', 25000);

Обновление данных в таблице осуществляется с помощью оператора update. Общий вид оператора приведен ниже.

Update < имя таблицы>

set

< столбец\_1 > = < выражение\_1>,

…..,

< столбец\_n > = < выражение\_n >,

[ where < условие > ];

Оператор обновляет таблицу, присваивая новые значения заданным столбцам. Если фраза where опущена, то обновляется все строки таблицы, в противном случае только те строки, которые удовлетворяют заданному условию.

Пример:

Update barja

set

name = ’ Fortune’

where name = ’Kaluga’;

Удаление данных в таблице осуществляется с помощью оператора delete. Общий вид оператора приведен ниже.

Delete from < имя таблицы>

[where <условие>]

Оператор удаляет строки из таблицы. Если фраза where опущена, удаляются все строки, но таблица остается.

Drop barja – удаляет таблицу полностью.

Приведенный ниже оператор удаляет из таблицы только одну строку, удовлетворяющую заданному условию:

Delete from barja

where name = ’ Fortune’

## 2.14-15 Лабораторная работа №14 Технология ADO.net. Подключение к БД. Разработка приложений для доступа к БД

***Источник данных DataSet***

Основным объектом ADO является *источник данных*, представленный

объектом DataSet. DataSet состоит из объектов типа DataTable и объектов

DataRelation. В коде к ним можно обращаться как к свойствам объекта DataSet, то есть, используя точечную нотацию. Свойство Tables возвращает объект типа DataTableCollection, который содержит все объекты DataTable используемой базы данных.

***Таблицы и поля (объекты DataTable и DataColumn)***

Объекты DataTable используются для представления одной из таблиц базы

данных в DataSet. В свою очередь, DataTable составляется из объектов

DataColumn.

DataColumn – это блок для создания схемы DataTable. Каждый объект

DataColumn имеет свойство DataType, которое определяет тип данных, содержащихся в каждом объекте DataColumn. Например, можно ограничить тип данных до целых, строковых и десятичных чисел. Поскольку данные, содержащиеся в DataTable, обычно переносятся обратно в исходный источник данных, необходимо согласовывать тип данных с источником.

***Объекты DataRelation***

Объект DataSet имеет также свойство Relations, возвращающее коллекцию

DataRelationCollection, которая в свою очередь состоит из объектов DataRelation.

Каждый объект DataRelation выражает отношение между двумя таблицами (сами таблицы связаны по какому-либо полю (столбцу)). Следовательно, эта связь осуществляется через объект DataColumn.

***Строки (объект DataRow)***

Коллекция Rows объекта DataTable возвращает набор строк (записей) за-

данной таблицы. Эта коллекция используется для изучения результатов запроса к базе данных. Мы можем обращаться к записям таблицы как к элементам простого массива.

***DataAdapter***

DataSet – это специализированный объект, содержащий образ базы дан-

ных. Для осуществления взаимодействия между DataSet и собственно источником данных используется объект типа DataAdapter. Само название этого объекта – адаптер, преобразователь, – указывает на его природу. DataAdapter содержит метод Fill() для обновления данных из базы и заполнения DataSet.

***Объекты DBConnection и DBCommand***

Объект DBConnection осуществляет связь с источником данных. Эта связь

может быть одновременно использована несколькими командными объектами.

Объект DBCommand позволяет послать базе данных команду (как правило, команду SQL или хранимую процедуру). Объекты DBConnection и DBCommand иногда создаются неявно в момент создания объекта DataSet, но их также можно создавать явным образом.