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

*«Б1.Д.Б.24 Базы данных»*

Уровень высшего образования

БАКАЛАВРИАТ

Направление подготовки

*09.03.03 Прикладная информатика*

(код и наименование направления подготовки)

*Прикладная информатика в экономике*

(наименование направленности (профиля) образовательной программы)

Квалификация

*Бакалавр*

Форма обучения

*Очная*

Год набора 2025

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 лабораторным занятиям …..…………..... | 6 |
| 3 Методические указания по практическим занятиям | 7 |
| 4 Методические указания по самостоятельной работе …..…………..... | 8 |
| 4.1 Методические указания по выполнению индивидуального творческого задания………………………………………..………………. | 8 |
| 4.2 Методические указания по выполнению курсового проекта | 9 |
| 5 Методические указания по промежуточной аттестации по дисциплине………………………………………………………………….. | 11 |

**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 обучающимся, необходимо 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 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. Также предлагаются рекомендации по выполнению и оформлению лабораторных работ и список рекомендуемой литературы.

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 3 семестре обучающиеся приобретают навыки создания баз данных и приложений для доступа к базам данных в среде СУБД LibreOffice Base. В 4 семестре обучающиеся знакомятся с удаленными СУБД. В компьютерных классах установлена свободная СУБД PostgreSQL.

**3 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 инфологические модели баз данных, преобразовывать их в даталогические, 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.

**4 Методические указания по самостоятельной работе**

**4.1 Методические указания по выполнению индивидуального творческого задания**

Целью выполнения индивидуального задания по дисциплине "Базы данных" является приобретение навыков реализации баз данных в среде СУБД LibreOffice Base, а также приобретению практических навыков реализации запросов и формированию отчетов. При выполнении индивидуального задания студент должен показать умение обоснованно и последовательно изложить материал разрабатываемой темы.

Индивидуальное задание выполняется по актуальным вопросам систем обработки данных согласно индивидуального или комплексного задания и оформляется в виде пояснительной записки и программную реализацию с использованием современных СУБД. Основой для успешного выполнения индивидуального задания является глубокое изучение научно-технической литературы.

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

Выполнение задания состоит из нескольких этапов:

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

Отчет предоставляется в текстовом файле и в распечатанном виде, разработанная БД – в электронном виде.

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

**4.2 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 курсовой работы студент должен показать умение анализировать предметную область, и на практике применить теоретические знания в области проектирования и реализации реляционных баз данных.

Основой для успешного выполнения курсового проекта является глубокое изучение учебной и научно-технической литературы.

Практической частью курсового проекта 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. Разработка приложения для доступа к базе данных
6. Реализация запросов

Этапы 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.

**5 Методические указания по промежуточной аттестации по дисциплине**

Изучение дисциплины «Базы данных» в 3 семестре завершается экзаменом. Для допуска к экзамену в 3 семестре необходимо представить отчет по индивидуальному творческому заданию. Этот отчет описывает включает описание процесса создания базы данных и приложения для работы с базой данных в СУБД LibreOffice Base. Кроме того, обучающийся должен ответить на вопросы по теоретическому материалу, пройденному в семестре.

Изучение дисциплины в 4 семестре завершается экзаменом. Подготовка к нему 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. Для допуска к зачету студент должен защитить курсовой проект. Отчет включает описание результатов проектирования и реализации системы управления базами данных в среде удаленной СУБД (PostgreSQL, MySql).

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.

Чтобы хорошо подготовиться к экзамену, 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

*«Б1.Д.Б.24 Базы данных»*

Уровень высшего образования

БАКАЛАВРИАТ

Направление подготовки

*09.03.03 Прикладная информатика*

(код и наименование направления подготовки)

*Прикладная информатика в экономике*

(наименование направленности (профиля) образовательной программы)

Квалификация

*Бакалавр*

Форма обучения

*Очная*

Год набора 2025

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

[1 Лабораторные работы 14](#_Toc193994867)

[1.1 Лабораторная работа №1. Обзор современных СУБД 14](#_Toc193994868)

[1.2 Лабораторная работа №2. Основы работы в среде СУБД LIBREOFFICE BASE. Возможности системы 14](#_Toc193994869)

[1.3 Лабораторная работа №3-4. Создание БД в среде СУБД LIBREOFFICE BASE в режиме мастера и конструктора. Реализация ограничений целостности. Обеспечение ссылочной целостности 15](#_Toc193994870)

[1.4 Лабораторные работы №5-6. Создание форм для ввода и редактирования данных. Связанные и подчиненные формы 18](#_Toc193994871)

[1.5 Лабораторная работа №7. Создание отчетов 20](#_Toc193994872)

[1.6 Лабораторные работы №8-9. Реализация запросов в среде СУБД LIBREOFFICE BASE. Параметрические запросы. Использование обобщающих функций в запросах 21](#_Toc193994873)

[1.7 Лабораторная работа №10. Создание завершенного приложения 22](#_Toc193994874)

[1.8 Лабораторные работы №11-13. Знакомство с СУБД PostgreSQL. Утилита psql. Ввод и выполнение SQL-команд. Создание и регистрация БД 23](#_Toc193994875)

[1.9 Лабораторная работа №14. Знакомство с pgAdmin 25](#_Toc193994876)

[1.10 Лабораторные работы №14-17. Создание основных объектов БД в PostgreSQL. Заполнение БД. Реализация запросов с помощью оператора Select 26](#_Toc193994877)

[1.11 Лабораторные работы №17-20 Создание хранимых процедур. Создание функций и триггеров для обеспечения семантической и ссылочной целостности 30](#_Toc193994878)

[1.12 Лабораторная работа №21. Уровни изоляции транзакций в PostgreSQL 35](#_Toc193994879)

[1.13 Лабораторная работа №22. Резервное копирование и восстановление данных 36](#_Toc193994880)

[1.14 Лабораторная работа №23. Работа с форматом json 36](#_Toc193994881)

[1.15 Лабораторная работа №24 Подключение к БД PostgreSQL 37](#_Toc193994882)

[1.16 Лабораторные работы №25-29. Работа с БД с помощью объектов подключенного уровня ADO.NET 39](#_Toc193994883)

[1.17 Лабораторная работа №16. Работа с БД с помощью объектов отключенного уровня ADO.NET 39](#_Toc193994884)

# 1 Лабораторные работы

# 1.1 Лабораторная работа №1. Обзор современных СУБД

В ходе выполнения работы студент должен осуществить анализ интернет-источников или периодических изданий, найти информацию о 3-4 СУБД и привести их краткую характеристику, указав следующие данные:

-фирму-производителя;

-год выпуска;

-назначение;

-поддерживаемую модель данных;

-платформу;

-сферу использования;

-вид лицензии и т.д.

# 1.2 Лабораторная работа №2. Основы работы в среде СУБД LIBREOFFICE BASE. Возможности системы

Microsoft LibreOffice Base является настольной СУБД (система управления базами данных) реляционного типа. Достоинством LibreOffice Base 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, LibreOffice Base 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 LibreOffice Base 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 LibreOffice Base 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. LibreOffice Base следит за разграничением доступа разных пользователей к БД и обеспечивает защиту данных. Так как LibreOffice Base 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 LibreOffice Base 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 LibreOffice Base 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 LibreOffice Base 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 LibreOffice Base 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.

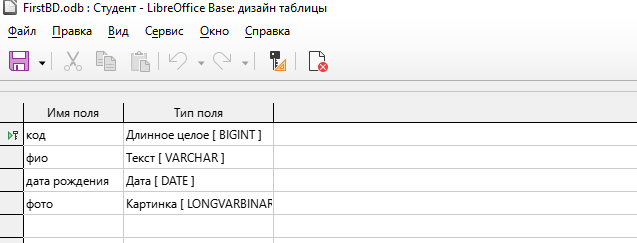
Однако, при известных недостатках MS LibreOffice Base обладает большим количеством преимуществ по сравнению с системами подобного класса.

## ***1.3 Лабораторная работа №3-4. Создание БД в среде СУБД LIBREOFFICE BASE в режиме мастера и конструктора. Реализация ограничений целостности. Обеспечение ссылочной целостности***

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

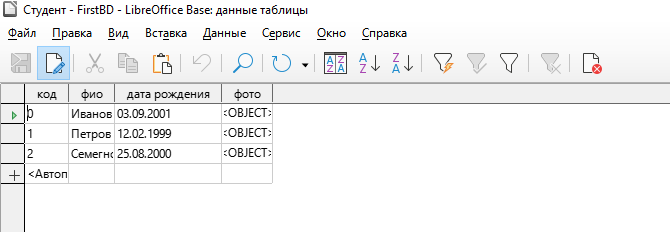
2. При создании таблиц в окне конструктора использовать разнообразные типы полей - числовой, текстовый, дата и т. д. Одна из таблиц должна содержать фотографию. Указать атрибуты, которые не могут быть пустыми.

На рисунке представлено окно конструктора таблиц. При вводе даты указать нужный формат, т.к. даты могут храниться в кратком формате дд.мм.гг или длинном дд.мм.гггг. Структуру таблицы можно в любой момент изменить, щелкнув правой кнопкой мыши на нужной таблице и выбрав команду «Правка».



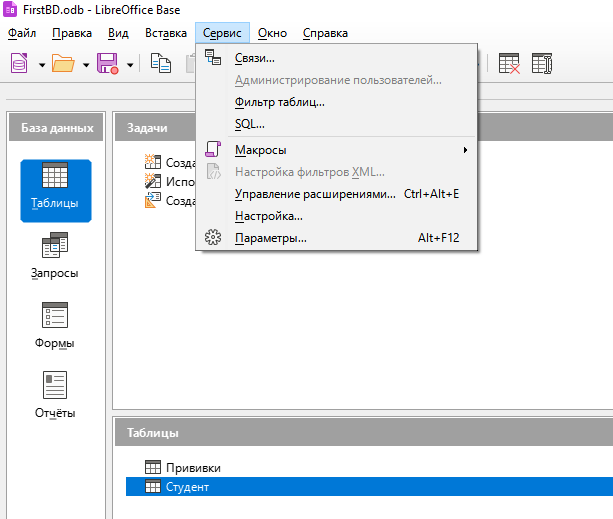
3. Ключевые поля. Чтобы один из столбцов определить в качестве первичного ключа, в разделе «Свойства поля» устанавливаем для параметра «Автозначение» поля первичного ключа значение Да.  С функцией Автоназначение (AutoValue), Base будет автоматически вводить следующий инкрементный номер для каждой новой записи.

4. Заполните таблицы. Для этого щелкнуть правой кнопкой мыши на нужной таблице и выбрать команду «открыть». Откроется окно.

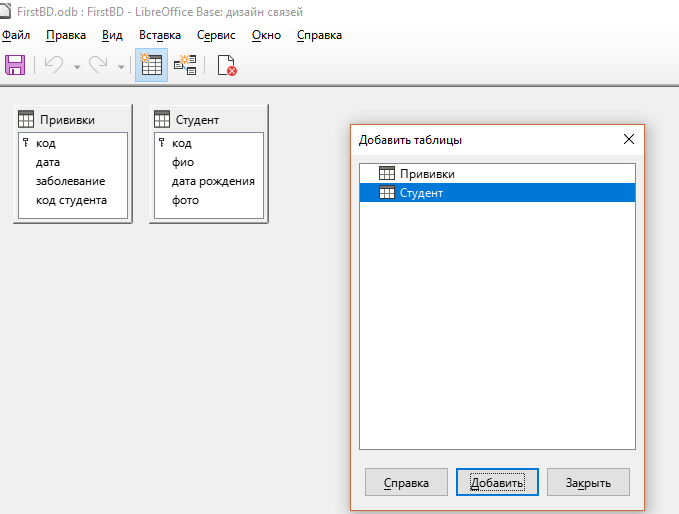


На этом этапе нельзя добавить данные в поле типа «Картинка», это делается с помощью формы.

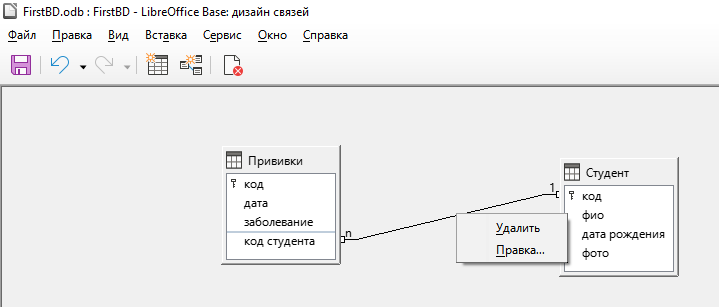
Создадим еще одну таблицу «Прививки», которая будет содержать сведения о прививках студентов. Очевидно, что у каждого студента может быть несколько прививок, поэтому таблица «Прививки» содержит больше строк, чем таблица «Студент» и является подчиненной таблицей, а таблица «Студент» является главной. Внешний ключ «Код студента» помещаем в таблицу «Прививки». Свяжем эти таблице с помощью пункта меню Сервис->Связи.



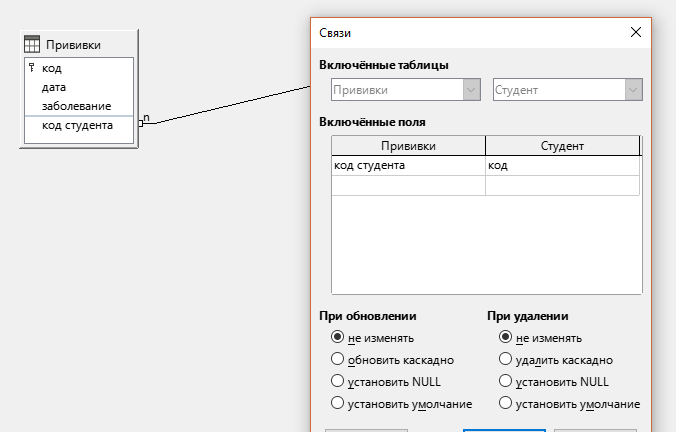
В появившемся окне выберем нужные таблицы



Для установления связи необходимо выделит поле первичного ключа основной таблицы «Студент» перенести в поле внешнего ключа подчиненной таблицы «Прививки». Эту связь в любой момент можно изменить или удалить, как показано на рисунке.



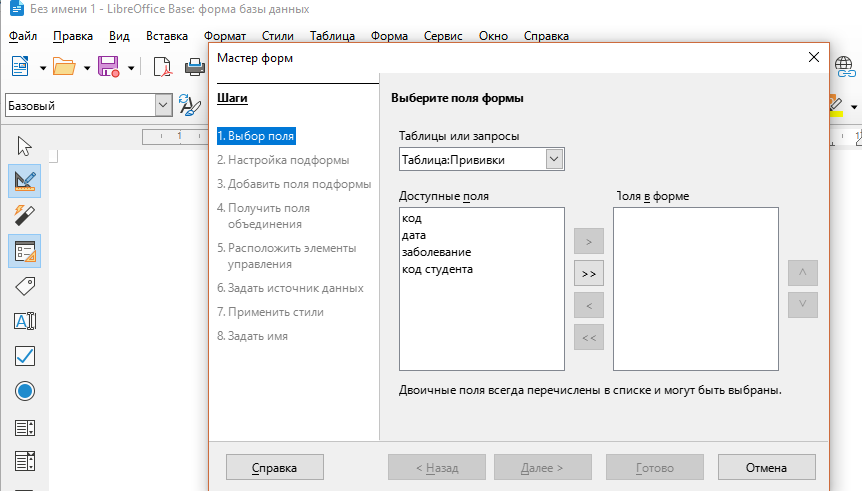
Связь имеет мощность «1:N», т.е. у каждого студента может быть несколько прививок. Здесь задаем стратегии обеспечения ссылочной целостности.



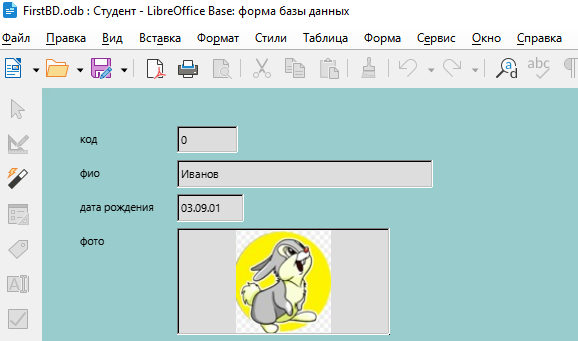
## **1.4 Лабораторные работы №5-6. Создание форм для ввода и редактирования данных. Связанные и подчиненные формы**

В форме LibreOffice Base, 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 LibreOffice Base, ее внешний вид играет важную роль. В LIBREOFFICE BASE формы можно создавать в режиме дизайнера или в режиме мастера.

Рассмотрим процесс создания формы на основе одной таблицы, например «Студент». Выбираем команду «Создание формы с помощью мастера» Мастер позволяет выбрать столбцы, которые будут отображаться на форме, внешний вид формы и т.д.



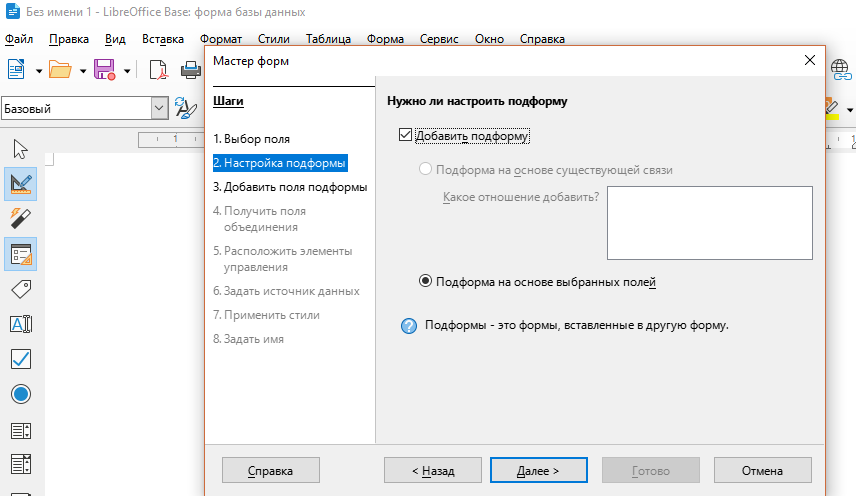
Последовательно выполняя указания мастера, получим следующую форму



В нижней части формы расположены кнопки навигации, которые позволяют перемещаться по строкам таблицы, изменять содержимое строк, добавлять новые и удалять имеющиеся.



Теперь создадим форму на основе двух таблиц. На этой форме при перемещении по строкам основной таблицы будут показаны только те строки подчиненной, которые связаны с текущей записью основной таблицы. Для этого в мастере форм следует поставить галочку при ответе на вопрос «Нужно ли настроить подформу». После этого выбрать нужные поля из подчиненной таблицы.



Форма имеет вид, представленный на рисунке. На форме можно передвигаться как по строкам основной, так и подчиненной таблицы, изменять, удалять и добавлять записи в обе таблицы.

# 

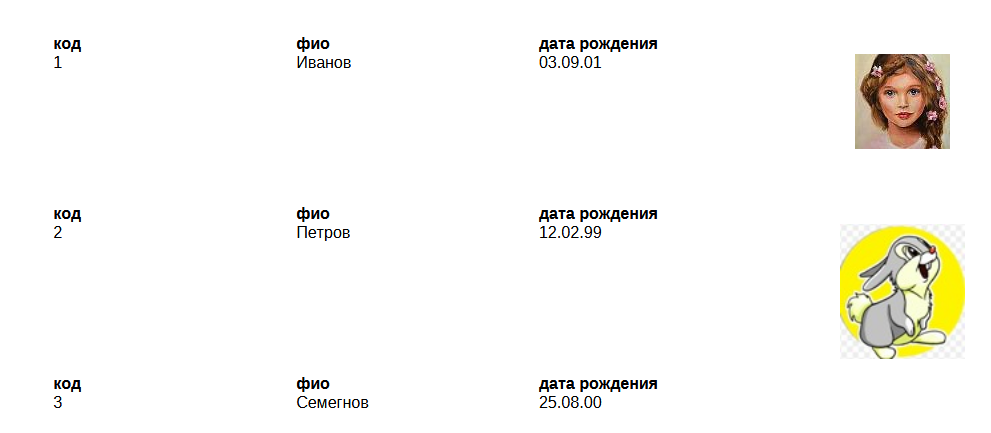
# 1.5 Лабораторная работа №7. Создание отчетов

Для того чтобы иметь возможность просмотреть нужную нам информацию, возможность распечатать данные, имеющиеся в базе данных, используются отчеты. На первом этапе освоения программы LibreOffice Base целесообразно воспользоваться командой «Мастер отчетов» расположенной на ленте «Создать». Отчёт — это неизменяемый документ, который предоставляется конечному потребителю для просмотра или печати. Отчёт формируется на основе таблицы или запроса и может быть статическим, либо динамическим. Статический отчёт будет включать в себя только ту информацию, которая была актуальна на момент его создания, и при повторном выполнении изменяться на будет. Такой тип удобен, например, для создания отчётности по прошедшим периодам. В динамическом отчёте данные обновляются во время каждого выполнения и могут использоваться для мониторинга информации в интерактивном режиме.

Создать отчёт можно, традиционно, в режиме дизайна или с помощью мастера.

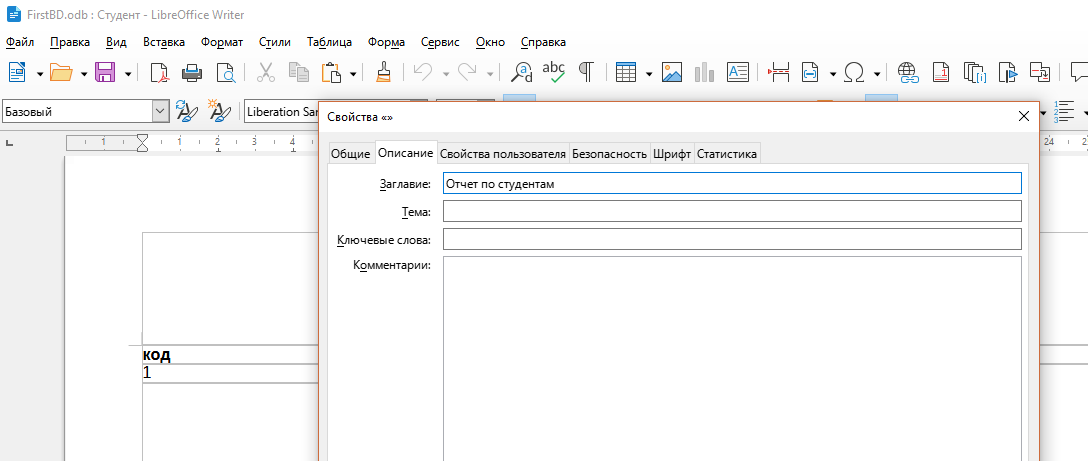
В процессе работы «Мастер отчетов» предложит выбрать из списка доступные поля для отчета, вид представления данных, порядок сортировки и вычисления, мает для отчета, стиль и имя нашего отчета.

На рисунке представлен вид простейшего отчета без группировки и подведения итогов.



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

Чтобы вписать в отчёт имя автора и его название, выполним следующие настройки: пункт меню Файл->Свойства..., вкладка Описание, поле Заголовок:



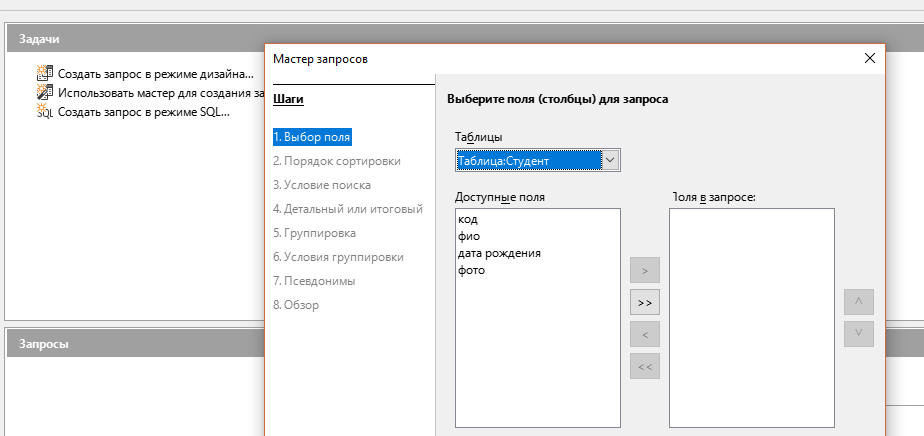
1. Необходимо создать простой отчет по нескольким таблицам и отчет с группировкой и итогами, также по двум и более таблицам.

# 1.6 Лабораторные работы №8-9. Реализация запросов в среде СУБД LIBREOFFICE BASE. Параметрические запросы. Использование обобщающих функций в запросах

Запросы используют для извлечения какой-либо информации из базы данных. Обычно данные в запросе должны отвечать условиям, определяемым пользователем, и содержат данные сразу из нескольких таблиц. Результат запроса обычно выводится в виде таблицы, все записи которой удовлетворяют заданным условиям.

В LibreOffice Base создавать запросы можно в режиме дизайна запросов, мастера запросов и SQL. Для начала рассмотрим его второй метод. Выберем пункт Запросы в окне База данных и пункт Использовать мастер для создания запросов... в окне Задачи. Откроется окно Мастера запросов, в котором на первом шаге нам будет предложено выбрать поля, выводимые в запросе.

Различают запросы на выборку, вычисляемые запросы, итоговые запросы, запросы на изменение. На рисунке представлено окно мастера запросов.



Здесь можно выбрать столбцы, которые войдут в результирующую таблицу. Далее можно задать правило сортировки и условие отбора строк.

Запросы к БД можно составить непосредственно на языке SQL.

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;

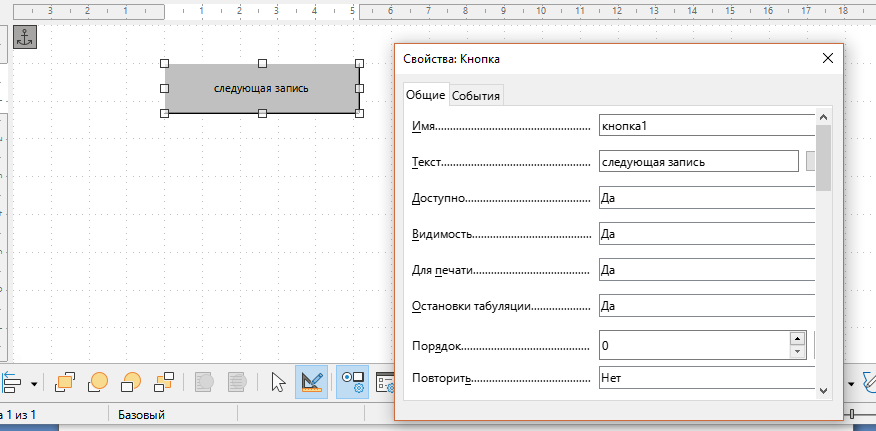
- запросы с подзапросами.

# 1.7 Лабораторная работа №10. Создание завершенного приложения

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.

Такая форма разрабатывается в режиме дизайна. При добавлении кнопки на форму при двойном щелчке мыши на ней открывается окно свойств.



В этом окне можно задать текст кнопки, текст всплывающей подсказки, цвет фона, шрифт. Действия, производимые по щелчку на кнопке, задаются в этом же окне в пункте «Действия».

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.

# 1.8 Лабораторные работы №11-13. Знакомство с СУБД PostgreSQL. Утилита psql. Ввод и выполнение SQL-команд. Создание и регистрация БД

PostgreSQL – 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. В процессе установки появится сообщение «Initialising database cluster», 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.

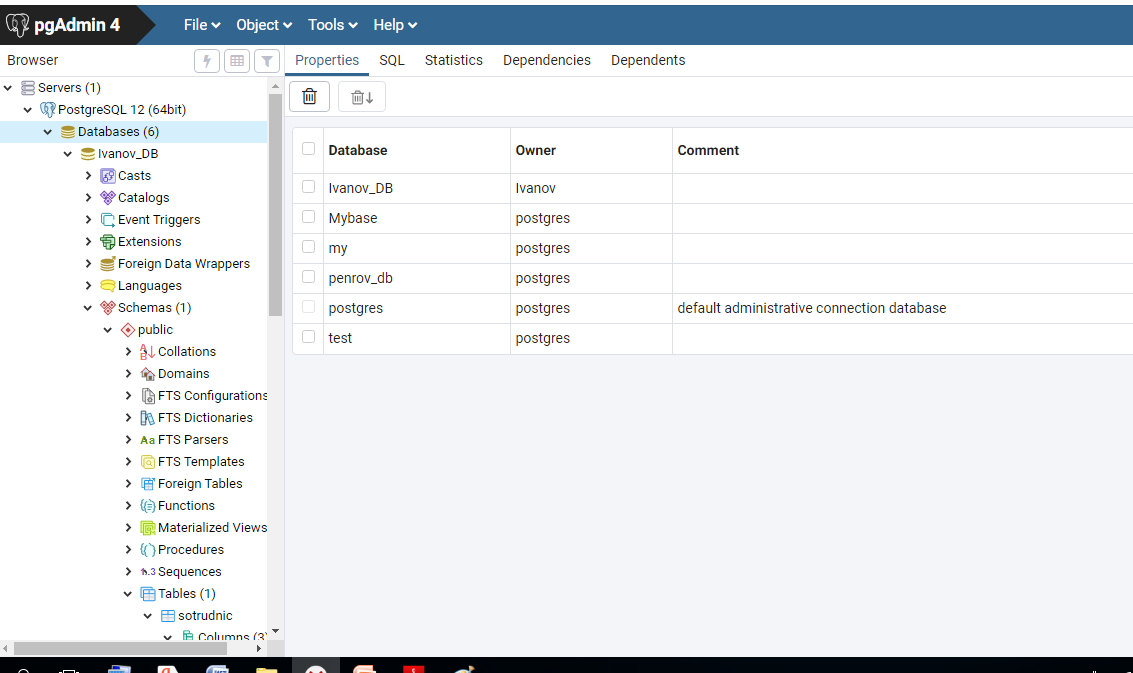
## 1.9 Лабораторная работа №14. Знакомство с pgAdmin

Администрирование СУБД PostgreSQL может осуществляться через графическое приложение pgAdmin. Далее представлено основное окно pgAdmin. Оно содержит несколько областей:

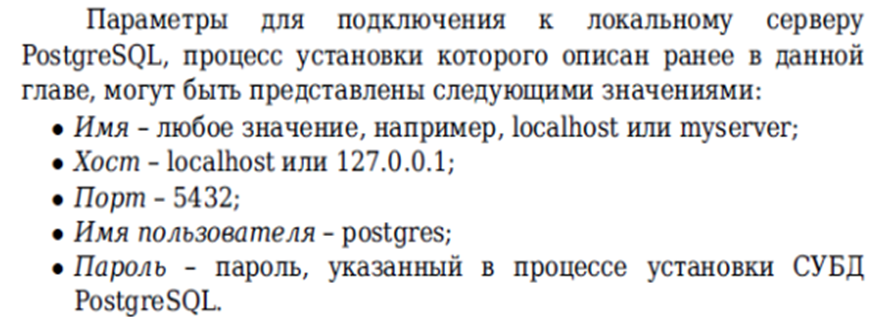
- браузер объектов – древовидная структура, содержащая основные объекты сервера. Браузер позволяет просматривать, сохдавать, редактировать и удалять различные объекты: базы данных, таблицы и т.д.

- 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.



Для начала необходимо создать новое подключение к серверу. При выборе пункта меню Сервер- >Добавить сервер откроется диалог для добавления сервера:





# 1.10 Лабораторные работы №14-17. Создание основных объектов БД в PostgreSQL. Заполнение БД. Реализация запросов с помощью оператора Select

СУБД PostgreSQL, как и все современные клиент-серверные СУБД, поддерживает 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. Ниже представлена схема отношения и оператор создания таблицы.

Баржа ( имя, дата изготовления, груз);

create table Barja

(

Name stroka Primary key,

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

Gruz integer not null check(Gruz>0)

);

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.

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’

Оператора Select 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 > ]

Пример:

Даны схемы отношений

Sotrudnik (Tab\_nomer, FIO, Dr Date, Adres)

Prikaz2 (Nomer, Data, Tab\_nomer, Name\_sudno\_Pas, Name\_sudno\_Barja, Name\_sudno\_Buksir)

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 %’;

## 1.11 Лабораторные работы №17-20 Создание хранимых процедур. Создание функций и триггеров для обеспечения семантической и ссылочной целостности

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.

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

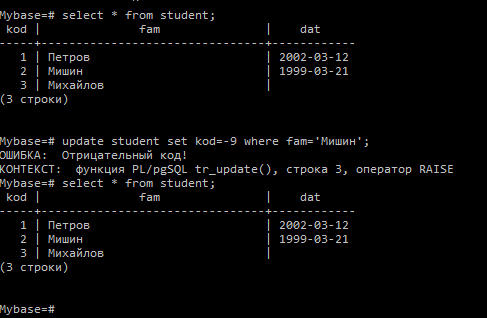
BEFORE UPDATE

ON student

FOR EACH ROW

EXECUTE PROCEDURE tr\_update();

Результат выполнения триггера:

******

## 1.12 Лабораторная работа №21. Уровни изоляции транзакций в PostgreSQL

Цель работы: закрепить теоретические навыки по данной теме.

Задачи: изучить влияние параллельных транзакций друг на друга при различных уровнях изоляции.

Практическое задание:

**1**

**Уровень изоляции READ UNCOMMITED**

1. Установить уровень изоляции READ UNCOMMITED.
2. В командном окне начать транзакцию, обновив содержимое таблицы. Просмотреть содержимое.
3. Выполнить откат транзакции. Сделать выводы.
4. Начать новую транзакцию, обновив запись в таблице.
5. Не завершая транзакцию, открыть еще одно командное окно, подключившись к БД под другим пользователем. Установить уровень изоляции. Просмотреть содержимое таблицы, обновленной в предыдущем окне. Сделать выводы.
6. Откатить обе транзакции.

**2**

**Уровень изоляции READ COMMITED**

**Убедитесь, что на этом уровне нет проблемы потерянного чтения**

1. Установить уровень изоляции READ COMMITED
2. В командном окне начать транзакцию, увеличив значение некоторого столбца на 200. Просмотреть содержимое.
3. Не завершая транзакцию, открыть еще одно командное окно, подключившись к БД под другим пользователем. Установить уровень изоляции. Начать новую транзакцию, пытаясь увеличить значение этого же столбца на 300.
4. В первом окне завершить транзакцию.
5. Во втором окне просмотреть содержание таблицы.
6. Завершить транзакцию во втором окне. Сделать выводы.

**Убедитесь, что на этом уровне остается проблема неповторяемого чтения**

1. В первом командном окне вывести содержимое таблицы
2. Во втором окне удалить строку в этой таблице и завершить транзакцию командой Commit.
3. Вернуться в первое окно и вновь вывести содержимое таблицы. Завершить транзакцию. Сделать выводы.

**3**

**Уровень изоляции REPEATABLE READ**

1. Установить уровень изоляции REPEATABLE READ
2. В первом окне вывести содержимое таблицы
3. Во втором командном окне установить уровень изоляции REPEATABLE READ. Добавить строку в таблицу, обновить существующую строку в таблице, завершить транзакцию командой Commit.
4. Вернуться в первое окно и вновь вывести содержимое таблицы.
5. Завершить транзакцию, зафиксировав изменения.
6. Вновь вывести содержимое таблицы. Сделать выводы.

## Лабораторная работа №22. Резервное копирование и восстановление данных

Цель: знакомство с возможностями PostgreSQL для резевного копирования и восстановления данных

Содержание:

1. Создать копию базы в логическом формате с помощью утилиты pg\_dump.
2. Создать копию базы данных в архивном формате с помощью pgAdmin.
3. Выполнить восстановление данных из логической копии с помощью утилиты psql.
4. Выполнить восстановление данных из архивной копии с помощью pgAdmin.
5. Выполнить восстановление данных из архивной копии с помощью утилиты pg\_restore.
6. Оформить отчет.

## Лабораторная работа №23. Работа с форматом json

Цель: знакомство с возможностями PostgreSQL для работы с форматом Json

Содержание:

1. Создать базу данных, состоящую из двух связанных таблиц. Главная таблица должна содержать сведения о некоторых объектах, подчиненная – более детальную информацию о каждом объекте в формате json;
2. Разработать Json-схему и проверить ее на валидность. Схему из лекции не повторять!!
3. Заполнить таблицы. При заполнении ввести в отдельные строки иерархические данные;
4. Реализовать следующую обработку:

- выборку по отдельным ключам, в том числе по ключам нижних иерархий (не менее 4 запросов);

- выборку по условию на значения отдельных ключей, в том числе ключей нижних иерархий (не менее 4 запросов).

**Варианты объектов**

1. Студенты
2. Фильмы
3. Писатели
4. Мобильные телефоны
5. Автомобили
6. Книги
7. Учебные дисциплины
8. Животные
9. Породы собак
10. Породы кошек
11. Футбольные игроки
12. Актеры кино
13. Страны
14. Певицы (Певцы)
15. Языки программирования
16. СУБД
17. Музыкальные группы
18. Моря
19. Отели
20. Морские суда
21. Танки

# 1.15 Лабораторная работа №24 Подключение к БД PostgreSQL

Провайдерами PostgreSQL являются Npgsql ODBC. ODBC (Open DataBase Connectivity) – это программный интерфейс (API) доступа к БД. ODBC позволяет единообразно оперировать с разными источниками данных, отвлекаясь от особенностей взаимодействия в каждом конкретном случае.

ODBC API позволяет разработчикам создавать приложения, работающие с БД разных производителей. Можно скачать с сайта Devart.com драйвер Devart - бесплатно на 3 месяца по ссылке:

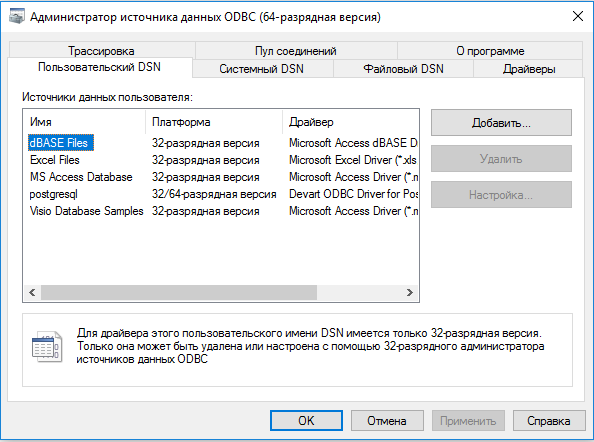
https://freesoft.ru/windows/postgresql\_odbc\_drayver/download/44985 - бесплатно.

Для подключения необходимо выполнить следующие действия.

В Пуск->Средства администрирования->Администрирование выбрать Источники данных (ODBC).

В результате откроется окно:

Администратор источников данных ODBC



На вкладке Пользовательский DNS необходимо нажать кнопку Добавить и выбрать нужный драйвер.

В данном примере это драйвер Devart ODBC Driver for PostgreSQL и нажать кнопку Готово.

В открывшемся окне параметров подключения необходимо указать: имя источника данных (Data Sourсe);

имя БД, с которой устанавливается соединение (Database);

имя сервера, на котором работает PostgreSQL (Server);

порт (Port);

имя пользователя БД (User Name) и пароль (Password), с использованием которых будет осуществляться подключение к БД (это значит, что на сервере с PostgreSQL должен быть создан пользователь именно с таким именем и паролем).

В приложении для подключения используем класс ODBC.Connection из пространства имен System.Data.Odbc (указать в Using)

OdbcConnection my\_Con = new OdbcConnection();

my\_Con.ConnectionString = "Dsn=My\_postgre\_SQL";

my\_Con.Open();

# 1.16 Лабораторные работы №25-29. Работа с БД с помощью объектов подключенного уровня ADO.NET

Класс Command позволяет выполнить SQL-оператор любого типа.

Класс Command можно использовать для создания и изменения баз данных, таблиц и индексов, а также для выполнения задач манипулирования данными - извлечения и обновления записей в таблице.

Существует 3 типа команд класса Command

|  |  |
| --- | --- |
| **значение** | **описание** |
| *CommandType.Text* | Команда будет выполнять прямой оператор SQL, указанный в свойстве CommandText. Используется по умолчанию |
| *CommandType.StoredProcedure* | Команда будет выполнять хранимую процедуру, имя которой указано в свойстве CommandText |
| *CommandType.TableDirect* | Команда будет опрашивать все записи таблицы. CommandText — имя таблицы. Эта опция предназначена только для обратной совместимости с некоторыми драйверами OLE DB |

Ниже приведен пример создания простого запроса

OdbcConnection my\_Con = new OdbcConnection();

my\_Con.ConnectionString = "Driver={Devart ODBC Driver for postgresql};Server=localhost;Port=5432;User=postgres;

Password=123;Database=Mybase";

string s = "select \* from student";

OdbcCommand my\_Command = new OdbcCommand(s, my\_Con);

# 1.17 Лабораторная работа №16. Работа с БД с помощью объектов отключенного уровня ADO.NET

ADO.net позволяет моделировать данные из БД с помощью классов пространства имен System.Data.

Все операции над данными осуществляются с локальной копией данных.

Ниже представлен принцип работы на отключенном уровне:



**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) заданной 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.