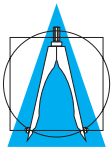


ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

# Базы данных

*Учебник*

ИЗДАТЕЛЬСТВО FOLIANT
Нур-Султан
2019

**УДК 004**
**ББК 32.973.26**
**Б 17**

Авторы:
Дэлер Эльмар
Харди Дирк
Троссман Хуберт

**Рецензенты:**
**Шангытбаева Г. А.** – PhD по специальности «Информационные технологии» Актюбинского регионального государственного университета им. К. Жубанова
**Нурина Г. К. –** преподаватель специальных дисциплин Актюбинского высшего политехнического колледжа, магистр техники и технологии по специальности «Информационные системы»
**Марат Г. М. –** преподаватель специальных дисциплин и производственного обучения Актюбинского высшего политехнического колледжа, магистр техники и технологии по специальности «Информационные технологии»

**Б17** **Базы данных:** *Учебник /* Пер. с немецкого. – Нур-Султан: Фолиант, 2019. – 184 с.

ISBN 978-601-338-415-3

Материалы учебника напрямую связаны с учебной программой модуля «Проектирование баз данных». Модуль относится к циклу профессиональных модулей, обеспечивающий соответствующий уровень знаний и умений в области программирования базы данных. Рассмотрены теоретические и практические основы для планирования, реализации и программирования баз данных с использованием современных программных систем. Большое внимание при этом уделяется разъяснению взаимосвязей.

Предназначен для студентов учебных заведений технического и профессионального образования, обучающихся по специальностям 1305000 «Информационные системы», 1304000 «Вычислительная техника и программное обеспечение».

**УДК 004**
**ББК 32.973.26**



ISBN 978-601-338-415-3

## Предисловие

Информатика и информационные технологии влияют практически на все социальные сферы нашей жизни. Почти все профессиональные действия и процессы в значительной степени осуществляются или поддерживаются благодаря информационным технологиям. Системы баз данных являются важным компонентом, так как от доступности, полноты и точности хранимых данных зависит дееспособность самой компании.

Настоящая книга «Базы данных – разработка, программирование, применение» предоставляет теоретические и практические основы для планирования, реализации и программирования баз данных с использованием современных программных систем. Большое значение при этом уделяется разъяснению взаимосвязей.

Являясь введением в предметную область технологий баз данных, эта книга подойдет обучающимся и студентам профессиональных училищ, колледжей, профессиональных академий, гимназий, специальных высших учебных заведений и университетов.

В отдельных главах наряду с многочисленными примерами также содержатся дифференцированные практические упражнения, которые служат для закрепления и углубления тематических областей.

Будем благодарны всем внимательным читателям, за ценные замечания, которые мы учтем в следующих изданиях нашей книги.

Нам важно Ваше мнение по этой книге!

Мы будем рады принять Ваши отзывы и предложения на электронную почту: lektorat@europa-lehrmittel.de.

Осень 2018 г. Авторы и издатели

СОДЕРЖАНИЕ

# 1 Основные понятия о базе данных

## 1.1 Использование баз данных

**База данных** – это набор сохраненных данных, которые находятся в логической взаимосвязи и управляются **системой управления базами данных (СУБД).** Эти данные используются, например, прикладными программами и пользователями предприятия.

**Примечание**
Базы данных являются логически связанными наборами данных.

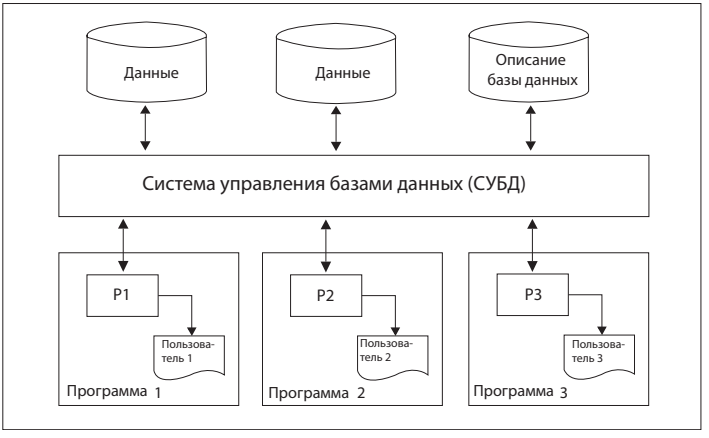
### 1.1.1 Примеры использования баз данных

Базы данных часто играют главную роль в использовании предприятием компьютерных технологий. Везде, где рабочие процессы обрабатываются с помощью компьютера, требуется хранение больших объемов данных.

**Примеры**

- Банки и страховые компании работают с системами управления базами данных. В базе данных структурирована вся информация о счетах, проводках и клиентах. Конфиденциальность и безопасность данных имеют наивысший приоритет.
- В компаниях любого размера и отраслях с системой планирования ресурсов (ERP, Enterprise Resource Planning), данные о клиентах, сотрудниках или товарах хранятся и обрабатываются с помощью системы управления базами данных.
- Автоматизированное управление складом также требует использования баз данных. База данных склада содержит упорядоченную информацию о многочисленных поставщиках, товарах и их запасах.
- Информационные системы в Интернете (например, Wikipedia) управляют своими статьями, используя базы данных.
- Компании хранят в хранилищах данных (Data Warehouses) различную информацию для анализа данных и помощи в принятии решений в различных сферах бизнеса. Таким же образом, например, институт изучения рынков хранит собственны данные и данные третьих лиц для их дальнейшей обработки.

Прикладные программы, например, программы для управления складом или персоналом, могут параллельно обращаться к общим данным посредством СУБД



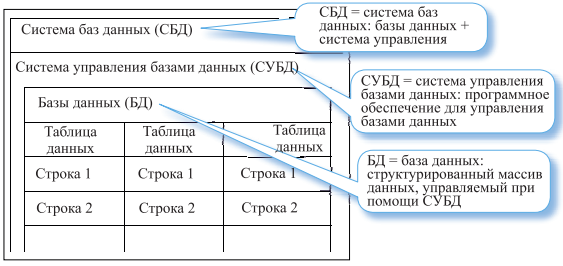

### 1.1.2 Проблемы хранения данных в базах данных

При хранении данных с помощью баз данных могут возникнуть многочисленные проблемы:

- Избыточность
  Данные сохраняются несколько раз (дублируются), что делает дорогостоящим внесение изменений в массив данных, а вероятность ошибок возрастает. Одни и те же данные должны быть изменены несколько раз в разных местах. Например, если изменения многократно сохраненных данных производится только в одном месте, то массив данных будет ошибочным.
- Несоответствия
  Если данные редактируются и изменяются одновременно несколькими пользователями или программами, это может привести к несогласованному состоянию данных. Доступ к данным не синхронизирован, в например, если текущий счет обрабатывается одновременно двумя пользователями, то оба видят одинаковое сальдо: 2000 евро. Теперь, если пользователь A снимет со счета 1000 евро и сохранит эту операцию, а пользователь B одновременно внесет 500 евро и сохранит эту операцию, то оба значения в базе данных: как 1000, так и 2500 евро будут противоречивыми и неверными.
- Защита данных
  Доступ к чтению и записи информации возможен на всем массиве данных. Защита данных – в зависимости от используемой операционной системы – может быть реализована с помощью настройки прав доступа или шифрования.
- Отсутствие независимости данных
  Управление данными обычно возможно только с помощью соответствующего прикладного программного обеспечения. Если необходимо изменить структуру данных, необходимо внести изменения как в прикладном программном обеспечении, так и в программе реструктуризации файлов. Также, для оценки одних и тех же файлов другого приложения необходимо создать собственное управление данными для этого нового приложения.

Для того, чтобы пользователь мог легко и четко управлять данными, ему нужна система управления базами данных – СУБД. Таким образом, **система базы данных (СБД)** состоит из комбинации базы данных (БД) и системы управления.

Наиболее распространенными СУБД являются: Microsoft Access, LibreOffice Base, MySQL, Paradox, Oracle и MS SQL Server.




**Примечание:**
СУБД хранит и организует данные без избыточности, с необходимым уровнем безопасности и защитой данных. СУБД не зависит от прикладных программ (приложений), которые получают доступ к данным.

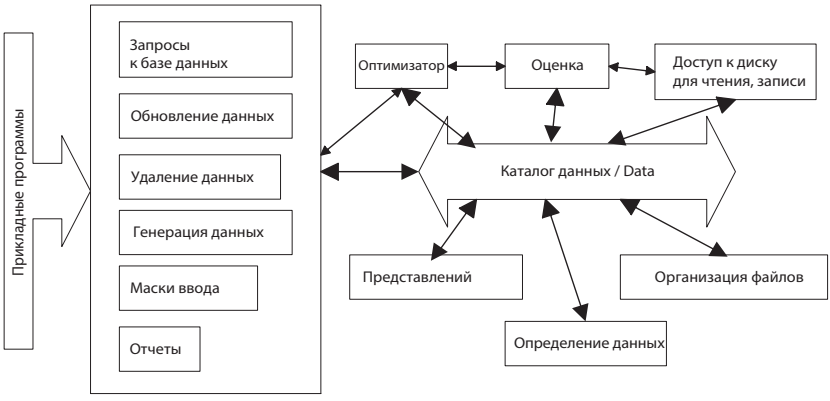
Прикладные программы не имеют прямого доступа к данным, а направляют запросы системе управления базами данных. База данных представляет собой набор логически связанных данных в определенной предметной области, например, данные клиента и данные заказа. СУБД реализует обмен данными между базой данных и их пользователями, например, прикладными программами. Она обеспечивает доступ к данным, предоставляя централизованное управление и контроль. СУБД управляет пользователями, их доступом к базе данных и правами доступа пользователей. Кроме того, при помощи СУБД обеспечивается защита от ошибок аппаратной части и программного обеспечения, поэтому при системном сбое данные не будут потеряны или смогут быть восстановлены. Внесение изменений в структуру базы данных не требует изменения прикладных программ.

### 1.1.3 Задачи СУБД

В 1982 году британский математик Эдгар Франк Кодд разработал 9 пунктов требований к СУБД:

1. Интеграция данных = единое управление всеми необходимыми данными.
2. Операции с данными = массив данных позволяет выполнять поиск, изменение и сохранение данных.
3. Каталог данных = (Data Dictionary) содержит описание базы данных.
4. Пользовательские представления = каждое приложение требует разных представлений (Views) массива данных.
5. Мониторинг согласованности = мониторинг целостности данных гарантирует правильность данных в БД.
6. Контроль доступа = доступ к данным может контролироваться и, при необходимости, пользователю может быть отказано в доступе.
7. Транзакции = изменения в БД могут быть сгруппированы как единицы.
8. Синхронизация = при конкурирующих транзакциях общие данные должны быть синхронизированы.
9. Резервное копирование данных = позволяет восстановить массив данных после конфликта, например, сбоя системы.

Разница между системой баз данных и накоплением отдельных файлов заключается в том, что в системе баз данных данные централизованно управляются системой управления базами данных – СУБД. Прикладные программы получают доступ к общим данным параллельно через СУБД.




Архитектура СУБД

**Каталог данных** (Data Dictionary) описывает, каким образом на внутреннем уровне реализуется хранение данных. Это – центральный каталог всей информации, важной для управления данными. В частности, выделяют следующие компоненты:

- Определение организаций файл, определение путей доступа к файлам,
- Определение концептуальных данных, определение пользовательских представлений, оптимизация дескрипторов базы данных,
- Оценка запросов и изменений и контроль доступа к диску.

Отдельные **транзакции** являются автономными запросами к массиву данных. Например, при переводе денежных средств 100 € снимаются со счета A и вносятся на счет B. Информация о транзакциях хранится **в журнале транзакций**. Журнал транзакций содержит информацию о начале и конце транзакции, а также об измененном массиве данных до и после транзакции. На основе данных журнала, транзакции могут быть отслежены или сторнированы (отменены).

Программное обеспечение для управления транзакциями представляет одновременный доступ к данным. Параллельные транзакции синхронизируются для обеспечения целостности базы данных.

**Пример**

На авиарейс Штутгарт-Берлин было забронировано 1 место. Клиент A в г. Ульм входит в программное обеспечение для бронирования, одновременно с клиентом B в г. Штутгарт и видит то же самое свободное место. Теперь, когда клиент A бронирует место, нажатие кнопки бронирования будет эквивалентно доступу записи к строке базы данных. В этот самый момент строка базы данных доступна клиенту A исключительно для транзакции. Если бронирование и, таким образом, транзакция будет успешно завершено, клиент B не может забронировать то же самое место. Клиент B видит, что место занято.

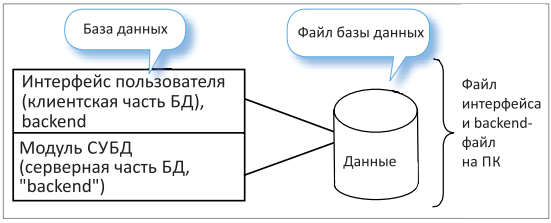
Налицо следующие преимущества:

- Все программы работают с одной базой данных, т. е. данные для всех одинаково актуальны.
- Одноразовое сохранение данных для всех приложений.
- Независимый одновременный доступ к общим данным под центральным управлением.

## 1.2 Системные архитектуры

### 1.2.1 Базы данных для настольных ПК (однопользовательский доступ)

В случае баз данных для настольных ПК, СУБД, например Access или Base, работают с конкретной базой (или базами) данных на ПК пользователя.

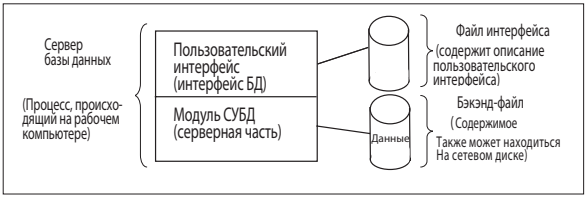



База данных для однопользовательского доступа

### 1.2.2 Базы данных на ПК для нескольких пользователей (многопользовательский доступ)

Если содержимое базы данных (bockend-файл) находится на сетевом диске в интрасети или в интернете, то несколько пользователей могут получить доступ к содержимому базы данных параллельно.

Все данные должны передаваться по сети для обработки, например, для поиска или сортировки.

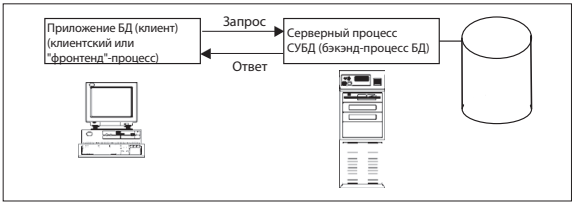



База данных для нескольких пользователей

### 1.2.3 Базы данных клиент-сервер

Система управления базами данных работает на серверном ПК в сети и имеет эксклюзивный доступ к файлам базы данных. Сервер реляционной базы данных также называется SQL-сервером (от **«Structured Query Language»** – структурированный язык запросов). Клиентские программы извлекают и хранят данные.

При этом по сети должны передаваться только запрос (оператор SQL, Query) и ответ. Приложение базы данных может работать на клиентской или серверной части, или распределяться на обе.




Клиент / Сервер базы данных

## 1.3 Модели баз данных

В основном различают четыре модели баз данных. Это реляционные объектно-ориентированные, иерархические, и сетевые базы данных. Различия этих четырех моделей заключаются в способе логического построения базы данных.

### 1.3.1 Реляционные базы данных

Реляционная база данных состоит исключительно из таблиц. Доступ осуществляется через эти таблицы. Поскольку новые таблицы можно легко добавить или удалить, дальнейшие изменения логической структуры базы данных производятся относительно легко. Доступ к таблицам легко реализуется программно, что привело к большой популярности этой модели базы данных.

Взаимосвязи между каждой отдельной таблицей выстраиваются через связи. Эти связи хранятся в таблицах. Построение массива данных с помощью таблиц и их связей друг с другом математически обосновано (реляционная алгебра).

Но реляционные базы данных также имеют недостатки: доступ часто осуществляется через несколько таблиц, что может привести к длительному времени отклика и большому количеству вводов/выводов.

### 1.3.2 Объектно-ориентированные базы данных

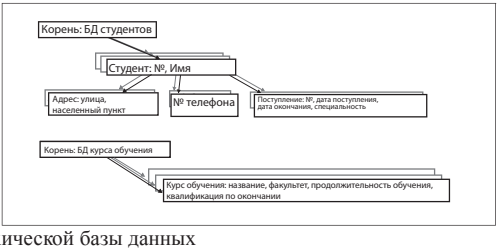
Объектно-ориентированная база данных состоит исключительно из объектов. Объект является либо реальным предметом, например, рейс, человек, либо вообще неким абстрактным предметом: адрес, счет, операция или отдел компании.

Поскольку многие объекты также могут храниться в табличной форме, объектно-ориентированные БД часто считаются расширением реляционных баз данных. Однако это справедливо лишь отчасти. Объектно-ориентированная база данных также включает объектно-ориентированные подходы, такие как классы, инкапсуляции данных или наследования.

Объектно-ориентированные и объектно-реляционные базы данных имеют более сложную структуру, чем собственно реляционные базы данных (т. к. могут содержать практически любые объекты, а не простые таблицы). Как следствие, архитекторы баз данных и программисты приложений должны вкладывать больше усилий в проектирование и программирование. Кроме того, внутреннее управление базой данных более обширно. В качестве преимущества, особенно в технических и мультимедийных приложениях, можно получить более наглядное построение (принудительное отображение сложных объектов в таблицах не требуется). Это может значительно уменьшить время выполнения запроса.

### 1.3.3 Иерархические и сетевые базы данных

Самые старые базы данных представляют собой иерархические базы данных, следующую ступень развития обычной файловой организации, используемой на ПК. Логическое построение этих БД соответствует перевернутой древовидной структуре. Доступ к нужному узлу всегда осуществляется через корень. При этом объект всегда может быть связан только с одним корнем; это называется моноиерархией. Такой способ обеспечивает минимальную избыточность, поскольку доступ осуществляется непосредственно через древовидную структуру и гарантирует кратчайшее время доступа.




Пример иерархической базы данных

Иерархические базы данных связывают данные через фиксированные связи, причем одна запись ссылается на следующую.

Одна из задач СУБД – моделирование реального мира – очень ограничена в иерархической модели. Ограничение на отображение верхних и нижних связей не подходит для реалистичного отображения повседневных ситуаций, в которых чисто иерархические связи можно наблюдать достаточно редко.

Примером иерархической системы управления базами данных является IMS от IBM.

Логическая структура **сетевых баз данных** состоит из данных, которые не являются чисто иерархическими, а связаны друг с другом через произвольно построенную сеть связей. Это значительно увеличивает гибкость, но увеличивает сложность моделирования БД.

Обе эти модели больше не соответствуют сегодняшним требованиям к БД.

Основными представителями сетевых баз данных являются IDMS (Computer Associates) и UDS (Siemens-Nixdorf).

**Преимущества и недостатки различных моделей баз данных**

| Модель | Преимущества | Недостатки |
|---|---|---|
| **Реляционные базы данных** | легкость внесения изменений в структуру базы данных, математически обоснованное, легко программируемое, простое управление | часто требуется много вводов/выводов, для больших объемов данных требуется высокая производительность компьютера |
| **Объектно-ориентированные базы данных** | универсальная объектно-ориентированная конструкция, относительно простой процесс программирования, простота управления | необходимо относительно много вводов/выводов, сложная структура, требуется относительно высокая производительность компьютера |
| **Иерархические и сетевые базы данных** | низкое время отклика, низкая избыточность | изменение структуры практически невозможно, сложное программирование |

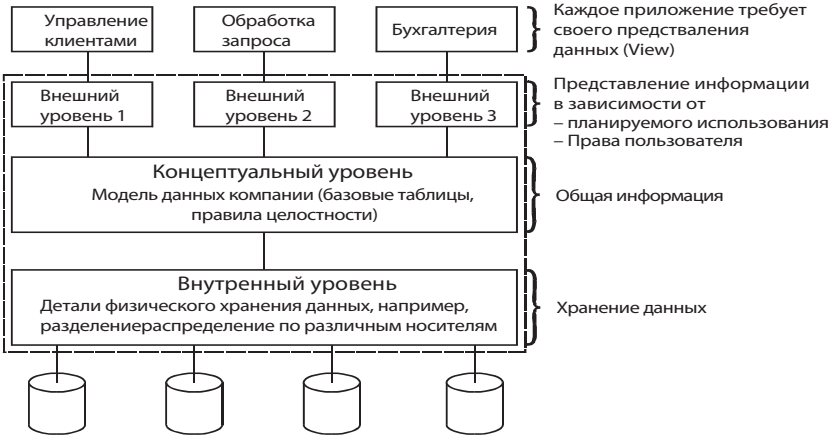
## 1.4 Архитектура системы управления базами данных (СУБД)

Важными особенностями описанного ранее подхода к базе данных являются:

- Изоляция программ и данных (независимость программ, данных и операций)
- Поддержка нескольких представлений пользователей (Views)
- Использование каталога для хранения описания базы данных (схемы).

### 1.4.1 Трехуровневая архитектура (трехслойная архитектура)

Трехуровневая архитектура отделяет пользовательские приложения и детали хранения (физические свойства) друг от друга.




Определены следующие три уровня:

**1. Внутренний уровень**

Внутренний уровень описывает физические структуры памяти базы данных. Внутренняя схема использует физическую модель данных и описывает сведения о доступе к ним при хранении, путях доступа к базе данных и организации файлов.

**2. Концептуальный уровень**

Концептуальный уровень описывает структуру всей базы данных для всех пользователей базы данных. Концептуальная схема скрывает детали физических структур хранения и фокусируется на описании единиц, типов данных, связей, пользовательских операций и ограничений. На этом уровне используется независимая от структур памяти, логическая модель данных.

**3. Внешний слой или слой представлений (View)**

Внешний слой включает внешние пользовательские представления (Views). Каждое представление описывает часть базы данных, в которой заинтересована определенная группа пользователей, и скрывает остальные данные базы этой группы пользователей. На этом уровне (логический) также может использоваться модель данных, не зависимая от структур памяти.

Архитектура уровня, которая лежит в основе практически всех современных систем баз данных, вносит значительный вклад в независимость между прикладными программами и внутренней структурой данных.

Изменения, касающиеся физического хранения данных, делаются в значительной степени невидимыми в СУБД для уровней выше внутренней схемы. Изменения в концептуальной схеме (которые на практике должны производиться как можно реже) могут многократно осуществляться посредством представления информации с помощью внешних схем и быть прозрачными для прикладных программ.

**Примечание:**
Базовые таблицы обычно редактируются только администратором базы данных.

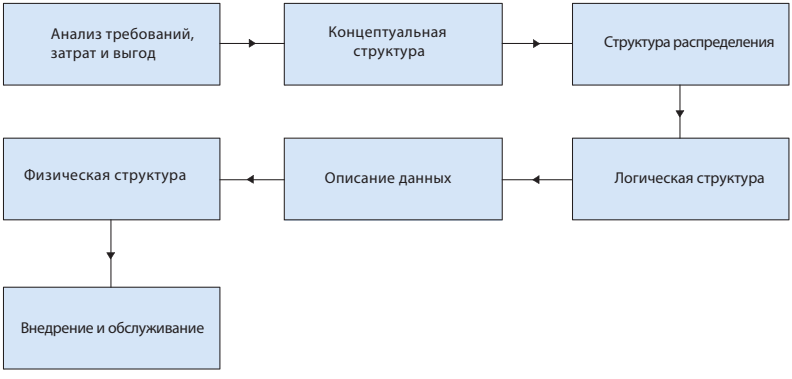
Внешние уровни, прежде всего, гарантируют, что отдельные приложения получат только ту информацию, которую они могут и должны иметь.

**Примечание:**
Пользователи или прикладные программы видят контент БД только с точки зрения внешнего уровня: «внешний вид» (external view).

## 1.5 Этапы проектирования базы данных

Создание прикладной программы для базы данных может быть разделено на несколько этапов. Этапы проектирования базы данных применимы также и к разработке других программных продуктов.

1. Сбор и анализ запросов к новой базе данных.
2. Системно-независимое проектирование базы данных в соответствии с функциями приложения.
3. Для распределенных баз данных: системно-независимая распределенная система проектирования.
4. Выбор модели базы данных и отображение концептуального дизайна модели базы данных.
5. Определение данных, т. е. их кодирование и программирование с использованием СУБД, определение пользовательских представлений.
6. Определение структур доступа в физической модели БД.
7. Установка приложения базы данных, настройка, тестирование.




## 1.6 Упражнения к Главе 1

1. Что такое система управления базами данных?
2. Что такое система баз данных?
3. Назовите примеры использования баз данных.
4. Какие проблемы возникают при использовании баз данных?
5. Опишите несоответствия в базах данных на примере снятия денег при помощи АТМ (банкомата).
6. Какие задачи имеет СУБД?
7. Какие задачи выполняются на внешнем уровне СУБД?
8. Опишите базу данных для настольного ПК с однопользовательским доступом.
9. Опишите базу данных для настольного ПК с многопользовательским доступом.
10. Опишите базу данных типа клиент-сервер.
11. Назовите различные модели базы данных.
12. Опишите реляционную модель.
13. Какие преимущества предоставляют иерархические базы данных?
14. Опишите трехуровневую архитектуру.
15. Укажите задачу на каждом уровне трехслойной структуры.
16. Какое преимущество имеют объектно-ориентированные базы данных?
17. Назовите различия между системой базы данных и хранением данных на ПК.
18. Какие задачи решаются с помощью каталога данных (Data Dictionary)?

# 2 Реляционные системы баз данных

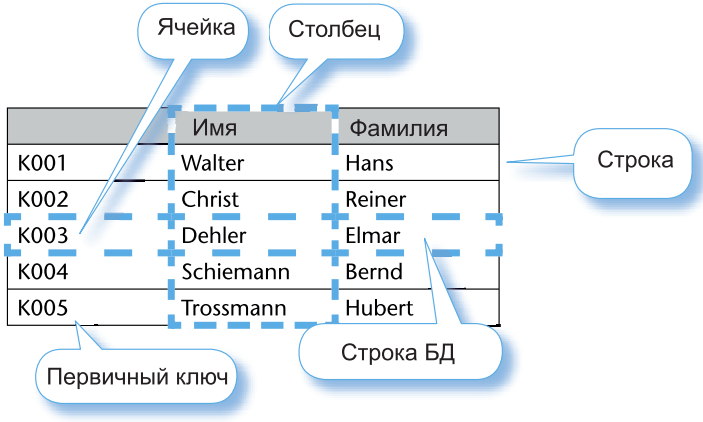
## 2.1 Реляционные системы баз данных

Данные в реляционной базе данных, например, данные клиентов или продукта хранятся в виде таблиц, которые могут быть связаны друг с другом.

### 2.1.1 Таблицы и связи

**Таблицы,** связанные друг с другом, также называются **реляционными таблицами**. Каждая ячейка таблицы содержит одну **запись базы данных.**

**Примечание:**
Реляционные системы баз данных управляют данными в таблицах, связанных друг с другом.




Столбцы таблицы содержат сопоставимые данные каждой записи, например, имя клиента. Значения столбца сохраняются в **поле типа данных** (тип данных), Системы баз данных имеют различные типы данных.

| **Типы данных в системах БД** | | |
|---|---|---|
| **Тип данных** | **Описание** | **Пример** |
| Integer | Целое число | 5 |
| Numeric (x.y) | Десятичное число с X цифр до запятой и Y цифр после запятой | 53,27 |
| Decimal (x.y) | Десятичное число с минимум X цифр до запятой и Y цифр после запятой | 53,2768 |
| Float | Число с плавающей запятой | 8,3E13 |
| Character | Символьная строка | CH80653 |
| Date | Дата | 31.05.1966 |
| Time | Время | 17:55 |

### 2.1.2 Ключи и связи

Отдельные строки таблицы – **строки базы данных**. Строки базы данных описывают, например, человека, объект или событие. Строки базы данных также называют сущностями (англ. «entity») или объектами.

**Примечание:**
Сущность, объект, строка базы данных соответствуют строке в таблице базы данных.

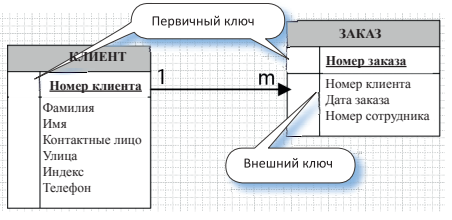
Объекты могут быть перепутаны с другими объектами в таблице, если они имеют одинаковые имена. В таблице Клиенты это могут быть два клиента. Чтобы исключить путаницу, для каждой строки базы данных необходимо установить идентификатор при помощи уникальных полей. Эти поля называются первичным ключом (Primary Key). Они позволяют однозначно идентифицировать соответствующую запись. В таблице Клиенты, это – обычно, номер клиента.

**Примечание:**
Первичный ключ используется для уникального обозначения и идентификации объекта в таблице.

**Связь 1:м**

Если клиент покупает товар, то повторно вводить все данные клиента в другую таблицу – Клиенты нецелесообразно. В таблице Заказы сохраняется только номер клиента.

В графическом представлении линия связи от таблицы с данными клиента, указывает на таблицу с данными заказа.

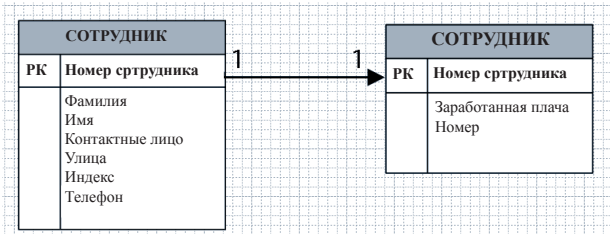



Таблицы Клиенты и Заказы связаны друг с другом при помощи первичного ключа таблицы Клиенты. Каждая строка базы данных в таблице Заказы содержит номер клиента, который привязан к клиенту «1» из таблицы Клиенты. И наоборот, обращение к каждому из клиентов таблицы Клиенты может быть производиться из нескольких «М» строк таблицы Заказы. Таким образом, таблицы Клиенты и Заказы имеют связь 1:m. Строки базы данных, которые однозначно идентифицируются в таблице Клиенты по первичному ключу Номер клиента, также устанавливаются в таблице Заказы по этому номеру клиента. Поле Номер клиента указывает на соответствующее поле таблицы Клиенты и, следовательно, называется **ссылочным ключом или внешним ключом**.

**Примечание:**
Внешний ключ – это поле, указывающее на поле первичного ключа другой таблицы.

**Связь 1:1**

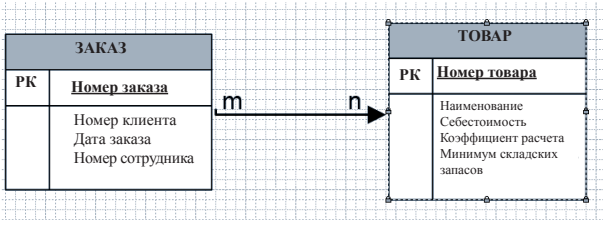
В персональной таблице данных такие данные как, например, заработная плата, не должны передаваться каждому пользователю и не должны находиться в свободном доступе. Таким образом, создается новая таблица Сотрудники_конфиденциально. Поскольку в данной таблице записи также можно найти по первичному ключу Номер сотрудника, то одна строка базы данных в первой таблице соответствует одной строке базы данных во второй таблице. Такая связь называется связью 1:1.



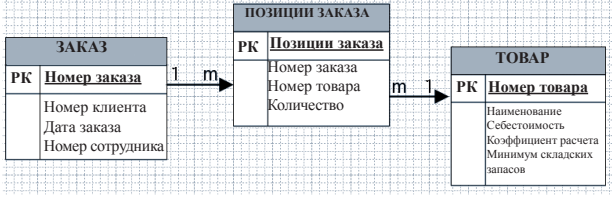

На практике связи 1:1 встречаются редко.

**Связь m:n**

В случае с таблицами Заказы и Товары, заказ может содержать «m», «n» или «∞» товаров, с другой стороны, каждый товар также может содержаться в любом количестве «n», «m» или «∞» заказов (∞ означает бесконечность).




Тогда при помощи таблицы Позиции заказа создаются две связи 1:n.




Товар, который указывается в записи Позиции заказа, может быть однозначно идентифицирован в таблице Товары. И наоборот, любой товар может быть указан в нескольких позициях заказа. Таким образом, между таблицами Товары и Позиции заказа существует связь 1:m. Таблицы Позиции заказа и Заказы также имеют связь 1:m. Заказ имеет столько товарных позиций, сколько требуется, но каждая позиция товара должна быть точно связана с заказом.

Обе внешние таблицы называют главными таблицами (также: сильный объект, англ. strong entity), а среднюю таблицу – дочерней таблицей (таблица соединений, слабый объект, англ. weak entity ).

**Примечание:**
Связь m:n в среде реляционной базы данных должна быть преобразована в связь 1: m и m:1 через подходящую дочернюю таблицу. В дочерней таблице первичные ключи основных таблиц должны вводиться как внешние ключи.

Мощность связи описывает характер связи строк базы данных друг с другом. Можно выделить 16 возможных мощностей.

| | | должно | может | | |
|---|---|---|---|---|---|
| | | 1 | n | c | nc |
| должно | 1 | 1:1 | 1:n | 1:c | 1:nc |
| | m | m:1 | m:n | m:c | m:nc |
| может | c | c:1 | c:n | c:c | c:nc |
| | mc | mc:1 | mc:n | mc:c | mc:nc |

m, n означает один или несколько (больше или равно 1), c означает 1 или 0

Примечание: для создания модели типа «сущность – связь», использование необязательной связи (c), а также использование обозначений min, max не имеет значения, но, тем не менее, автор читает нужным упомянуть их здесь. Существует множество номенклатур для представления диаграмм, которые полностью отказываются от необязательных связей, в угоду упрощению.

## 2.2 Модель типа «сущность – связь«

Модель типа **«сущность – связь» (ERM)** – это стандартизованный метод моделирования данных. ERM отображает структуры данных для их реализации с помощью программного обеспечения. При разработке ERM пользователи (клиент) и разработчики (конструкторы баз данных) поддерживают постоянную связь, чтобы максимально точно отобразить реальную ситуацию.

Отправной точкой ERM является понятие сущности. Сущность – это индивидуальный и идентифицируемый образец вещей, людей, понятий или абстрактных идей реального мира.

| **Примеры сущностей** | |
|---|---|
| **Обозначение** | **Пример** |
| Индивиды | Сотрудник Харди, студент Троссманн, клиент Делер |
| Реальные объекты | Машина 2, комната 7, товар 4711 |
| События | Оплата, бронирование, начало, посадка |
| Абстрактные понятия | Урок, лекции, услуги |

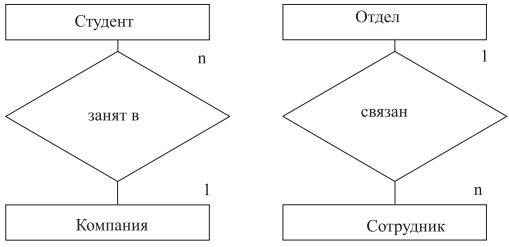
Сущность также часто называют объектом.

Сущности, имеющие одинаковую структуру и одинаковое описание, например, клиенты, группируются в группы **(типы сущностей)**. В реляционной модели каждый тип сущности отображается через таблицу базы данных, например, в таблице Клиенты.

Модель типа «сущность – связь» предназначена для визуализации и описания сохраненных данных и их отношений между собой. Это называется моделированием. Результатом моделирования является диаграмма ERD. Она представлена символами.

| **Символы модели типа «сущность – связь» для визуализации по Чену** | |
|---|---|
| Клиент<br>Счет<br>Заказ | Сводка сущностей с одинаковыми свойствами под единым общим термином представлена в ER-модели прямоугольником. В прямоугольнике находится имя сущности. |
| получает | Взаимодействия и зависимости между сущностями представлены связями (relationships). Связи в основном описываются глаголами.<br>Пример:<br>Клиент Делер — получает — счет № 2 |
| | Взаимосвязи представлены соединительными линиями. |

Диаграммы модели «сущность-связь» читаются сверху вниз или слева направо.




**Атрибуты**

Объекты описываются атрибутами. Например, клиент имеет такие атрибуты как номер, имя, отчество, адрес и место жительства.

Различают:

Ключевые атрибуты или идентификационные атрибуты, например, номер клиента, номер фирмы, номер человека; и описательные атрибуты, такие как имя, фамилия, статья, название, цена.

Если в таблице нет подходящих ключевых атрибутов, то вводится искусственный ключевой атрибут, например, серийный номер, который увеличивается автоматически.

Первичные ключи часто подчеркиваются одинарной линией, а внешние ключи – пунктирной.

**Примечание:**
Атрибуты не являются атомарными, т. е. не могут быть «разобраны» на более мелкие единицы.

**Связи**

Между сущностями существуют связи. Например, Стажер (1) занят в одной из компаний. И наоборот: одна компания использует труд нескольких стажеров. Одному отделу (1) назначено несколько сотрудников (ов), и наоборот, один сотрудник относится только к одному отделу. В модели типа «сущность – связь», связь 1:n указывается рядом с сущностями.


**Связь m:n**

Связь m:n между объектами означает, что к каждому объекту A относится несколько объектов B, и наоборот, к каждому объекту B относится несколько объектов A.

Например, ученик должен прослушать несколько курсов. И наоборот, на одном курсе всегда должно быть несколько учеников. В модели типа «сущность – связь» показана взаимосвязь между учеником и курсом.

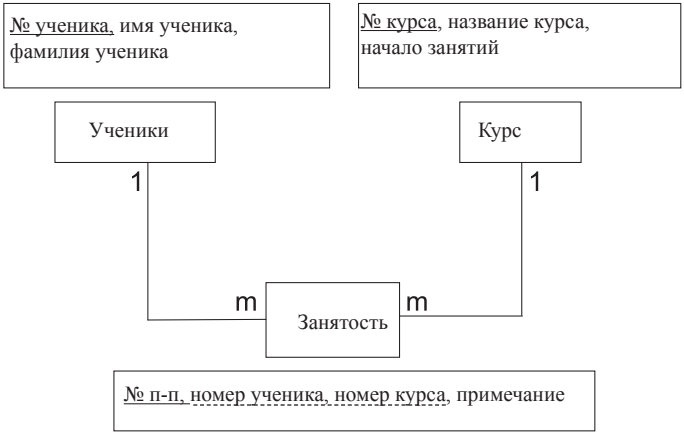

Для ER-модели или ER-диаграммы связи m:n могут быть представлены указаным выше способом.

Связь m:n невозможно представить в табличной форме. Значить, такую связь необходимо разделить на две связи 1:m, т. е. связь выводится через отдельную таблицу (дочернюю таблицу или объект соединения).

Новая дочерняя таблица содержит как минимум первичный ключ двух внешних таблиц (основных таблиц), дополнительные атрибуты также обычно целесообразны. Имя дочерней таблицы вытекает из реального положения вещей, например, Занятость. Если содержательное название найти не представляется возможным, то в основном выбирается сочетание внешнего имени, например, Студенческий курс.

**Примечание:**
Для установления связи m:n требуется объект соединения (дочерняя таблица).

Таблица, представляющая дочернюю таблицу (здесь: Занятость), принимает первичный ключ задействованных таблиц (здесь: Ученики и Курсы) в качестве внешнего ключа.




Обычно присваивается новый первичный ключ (здесь: № п-п). Как исключение, оба ключа вместе иногда могут формировать первичный ключ в новой таблице. Дополнительные атрибуты также могут быть использованы в этой таблице (здесь: примечание).

Модель «сущность – связь» по Питеру Чену:
1. Определение сущностей и связей.
2. Определение ключей идентификаторов для сущностей.
3. Определение типов сущностей и связей.
4. Настройка параметров связей.
5. Определение атрибутов и их диапазонов значений (доменов).
6. Построение диаграмм сущность-связь (ERD) для наборов сущностей и связей.
7. Определение первичных и внешних ключей.
8. Добавление атрибутов и их диапазонов значений в таблицы.

## 2.3 Примеры с решением для модели типа «сущность-связь»:

### 2.3.1 Обработка заказа

**В одной из организаций необходимо организовать обработку заказов с использованием базы данных. За каждый заказ отвечает один сотрудник (=персонал).**

a) Определите связи между этими таблицами.
b) Для сущностей Клиент, Заказ и Персонал создайте таблицы с соответствующими атрибутами.
c) Определите подходящие поля первичного ключа.

Решение:
Варианты дизайна

| **Сущность А** | **Связь** | | **Сущность Б** |
|---|---|---|---|
| Клиент 1 | выдает | m | Заказ |
| Заказ m | обрабатывается | 1 | Персонал |

Клиент: номер клиента, фамилия клиента, имя клиента, адрес, почтовый индекс, телефон клиента
Заказ: номер заказа, номер клиента, номер сотрудника. Данные сотрудника: номер сотрудника, фамилия сотрудника, имя сотрудника

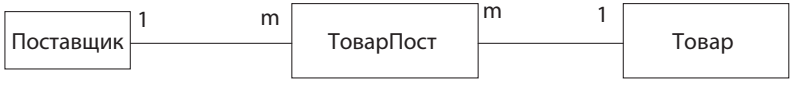

### 2.3.2 Поставщики и товары

**Компания получает свои товары от нескольких поставщиков, один поставщик поставляет несколько товаров в компанию. Каждая поставка включает в себя ровно один товар.**

a) Спроектируйте таблицы с соответствующими атрибутами для сущностей Клиент и Товар. Таблица ТоварПост предназначена для установления связи m:n.
b) Определите подходящие поля первичного ключа.
c) Определите связи между этими таблицами. Варианты дизайна:

| **Сущность А** | | **Связь** | **Сущность Б** | |
|---|---|---|---|---|
| Поставщик | m | поставляет | n | Товар |
| Поставщик | 1 | относится к | m | ТоварПост |
| Товар | 1 | распространяется | m | ТоварПост |

Поставщик: номер поставщика, фамилия поставщика, имя поставщика, адрес, почтовый индекс, телефон поставщика
ТоварПост: номер поставки, номер поставщика, номер товара, дата, количество
Товар: номер товара, название товара, цена, единиц на складе, упаковочные единицы



## 2.4 Упражнения к Главе 2

**1. В обучающей фирме один участник может пройти несколько курсов, в то же время курсы проводятся при минимальном количестве участников 5.**

Спроектируйте таблицы с соответствующими атрибутами для сущностей Участник и Курс. Таблица Заполняемость курса предназначена для установления связи m:n.

Варианты дизайна

| Сущность А | | Связь | | Сущность Б |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

2. В общей мед. практике «Medico» работают несколько врачей. Таким образом, каждый пациент может получать лечение у различных врачей.

Спроектируйте таблицы с соответствующими атрибутами для сущностей Врач и Пациент. Таблица Лечение предназначена для установления связи m:n.

Варианты дизайна

| Сущность А | | Связь | | Сущность Б |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**3. Система электронной обработкой данных**

Вы работаете на предприятии, которое использует большое количество ПК. Вам необходимо разработать ограниченную систему баз данных для технического обслуживания и реализации документооборота на ПК. Информация должна быть доступной для запроса с компьютеров предприятия.

База данных должна содержать следующую информацию:

- Кто из системных администраторов (большинство) является ответственным для каждого конкретного ПК.
- Должен быть доступен номер телефона и номер офиса каждого из администраторов.
- Системный администратор может использовать базу данных, чтобы узнать, где находятся компьютеры, за которые он отвечает (номер офиса).
- Для каждого ПК необходимо сохранить конфигурацию, чтобы можно было узнать емкость жесткого диска компьютера, а также какая операционная система и сетевая карта установлены.
- Руководитель может узнать, сколько часов каждый конкретный системный администратор работал на конкретном ПК.

a) Для указанной базы данных создайте ER-модель (сущность-связь), содержащую всю необходимую информацию.
b) Установите возникающие связи m:n в новой ER-модели (1:n.).
c) Создайте точное описание таблиц базы данных в виде описания связей. В связях укажите все первичные и внешние ключи четко выраженным способом.

**4. Велопрокат Faradiso**

Компания Faradiso предоставляет своим клиентам велосипеды напрокат. Прокат велосипеда можно оформить непосредственно на фирме, либо забронировать заранее. Кроме того, компания организует велотуры по различным направлениям. Направления тура передаются туроператору в виде таблицы (тур, описание, продолжительность, степень сложности, пункт отправления, пункт назначения). Faradiso отвечает только за регистрацию на тур и выбор дат. Вам необходимо разработать систему баз данных для технического обслуживания и реализации бизнес-процессов в компании. Для запроса должна быть доступна информация о клиентах и велосипедах.

База данных должна содержать следующую информацию:

- Какие велосипеды сданы в прокат, кому из клиентов и на какой срок.
- Какие велосипеды были забронированы и кем из клиентов.
- Количество клиентов, которые записались на определенную дату тура.
- Для каждого велосипеда: информацию об изготовителе, название, тип (например, туристический велосипед, горный велосипед, и т. д...), размер рамы, дата покупки, цена покупки и дата последнего техобслуживания.
- Для каждого велосипеда: цена за каждый день аренды. Велосипеды необходимо классифицировать по ценовым категориям.

a) Для указанной базы данных создайте ER-модель (сущность – связь), содержащую всю необходимую информацию.
b) Установите возникающие связи m:n в новой ER-модели (1:n.).
c) Создайте точное описание таблиц базы данных в виде описания связей. В связях укажите все первичные и внешние ключи четко выраженным способом.

**5. База данных для управления отделением больницы**

Создайте базу данных для управления отделением одной из больниц. Для запроса должна быть доступна информация о пациентах, врачах и младшем мед. персонале.

База данных должна содержать следующую информацию:

- Какие врачи лечат каких пациентов?
- Какая из палат (комнат) может быть предоставлена новому пациенту?
- Сколько свободных палат доступно сегодня в хирургическом отделении?
- Сколько свободных палат доступно сегодня в ЛОР-отделении?
- За какие палаты отвечает старшая сестра Хильда?

**Примечания:**

Каждая из палат закреплена за определенными отделениями. Каждый из врачей и членов младшего медицинского персонала (медсестра/медбрат) закреплены за определенными отделениями.

Сущности: пациент, врач, медсестра/медбрат, палата, отделение

**6. Авиакомпания Worldfly**

Необходимо создать базу данных для авиакомпании. База данных должна предоставлять информацию о том, какие самолеты приземляются в определенных аэропортах. Кроме того, необходимо получать информацию о том, у кого из производителей самолетов имеется пункт технического обслуживания в определенных аэропортах. Воздушные судна могут обслуживаться только в пунктах технического обслуживания соответствующего производителя. Авиакомпания Worldfly использует самолеты различных производителей.

База данных должна содержать следующую информацию:

- Где можно обслуживать конкретный самолет?
- В каких аэропортах приземляется конкретный самолет?
- Какова плата за посадку в разных аэропортах для конкретного самолета?

**7. Библиотека**

Вся информация о книжном фонде монастырской библиотеки должна быть собрана в одной базе данных.

- В библиотеке имеется по нескольку экземпляров определенных книг.
- Для каждой книги в базе данных должна быть сохранена следующая информация: автор, название, издательство, год выпуска.
- Существуют относительно старые книги, которые не имеют международного стандартного книжного номера (ISBN).
- У некоторых книг, помимо одного или нескольких авторов, есть еще издатель.

a) Составьте графическое представление связей таблиц посредством ER-диаграммы и укажите тип связи между каждой таблицей.

b) Укажите таблицы со значимыми атрибутами, четко обозначая первичные и внешние ключи.

**8. Врачебная практика**

В одной из врачебных практик работают 8 врачей. Запись на прием к каждому из них осуществляется с помощью базы данных. Для каждого приема назначается только один врач и один пациент. В таблице Врачи сохранены фамилия и номер телефона врача. База данных также должна хранить информацию о пациентах: фамилию, имя и дату рождения. База данных должна содержать точную дату приема, время начала и время окончания.

a) Изобразите связь между сущностями Врач и Пациент с помощью ER-диаграммы.

b) Укажите существующие связи.

c) Установите связь с помощью объекта соединения Прием посредством связей 1:n.

**9. Ресторан быстрого питания BurgerLand**

Данные, поступающие от отдельных филиалов ресторана быстрого питания BurgerLand организованы с помощью базы данных. В базе хранится имя каждого из филиалов, а также наименование позиций, покупки цена продажни. База данных каждого из филиалов должна содержать количество проданных позиций с указанием даты.

a) Изобразите связь между сущностями Филиал и Позиция с помощью ER-диаграммы.
b) Укажите существующие связи.
c) Установите связь посредством объекта соединения Продажа
d) Укажите все первичные и внешние ключи четко выраженным способом.

**10. Склад деталей**

На одном из сборочных предприятий производится установка определенных монтажных узлов. Подлежащие установке компоненты хранятся на складе деталей. Необходимо организовать управление складом с помощью базы данных.

Каждая деталь имеет атрибуты: название, вес, минимальный запас на складе и описание.

Одинаковые детали хранятся на одном определенном складском месте. Каждое складское место имеет свой уникальный номер. Различные складские места рассчитаны на различный допустимый общий вес.

Каждая деталь может быть получена от различных поставщиков. В базе данных хранится имя поставщика, его адрес и номер телефона.

Поставщики периодически отправляют предложения с текущими ценами и текущим временем доставки для каждой детали.

На предприятии уже существует таблица со всеми населенными пунктами Германии. Известны следующие атрибуты: идентификатор населенного пункта, почтовый индекс, название населенного пункта

a) Изобразите связь между сущностями с помощью ER-диаграммы.
b) Укажите существующие связи.
c) Установите возникающие отношения M:N.

**11. База данных футболистов**

Необходимо создать базу данных с именами футболистов и их игровым стажем. База данных хранит следующие сведения об игроке: фамилия, имя, дата рождения. Кроме того, база данных должна содержать следующую информацию: в каких играх принимал участие игрок, сколько времени (от, до) в общей сложности отыграл игрок, на какой позиции и за какую команду.

База данных должна иметь возможность хранить название команды, имя и возраст тренера.

a) Изобразите связь между сущностями с помощью ER-диаграммы.
b) Укажите существующие связи.
c) Установите возникающие отношения M:N.

# 3 Разработка и нормализация базы данных

## 3.1 Разработка базы данных


Этапы работы по разработке базы данных

**Шаг 1: Настройка информационной структуры**

Прежде всего необходимо провести анализ фактов реального мира, с составлением результирующих требований к информации. Какую информацию ожидает пользователь от системы баз данных? После этого, для графического представления взаимосвязей создается модель типа «сущность-связь».

**Шаг 2: Настройка информационной структуры**

На данном этапе необходимо ответить на вопрос о том, какие связи существуют между данными и таблицами. Затем осуществляется проектирование объектов (сущностей) и их отношений друг с другом.

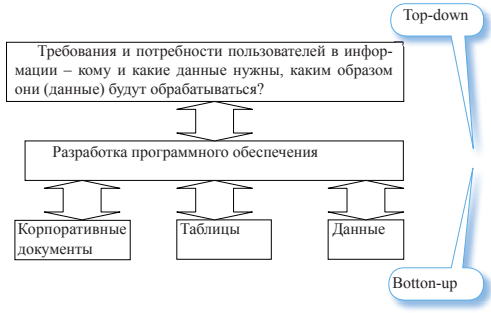
Результатом шагов 1 и 2 является реляционная модель данных, которая находит отражение в третьей нормальной форме (см. следующую главу).

**Шаг 3. Настройка структуры записи**

Теперь определяются отдельные таблицы и поля данных для определенных типов данных. После выбора подходящей СУБД происходит создание будущей базы данных с помощью программного обеспечения.

Построение информационной структуры базы данных, в соответствии с шагом 1, осуществляется методом botton-up, или методом top-down. На практике, как правило, оба этих метода объединяются.

### 3.1.1 Процедуры разработки программного обеспечения




**Метод «Botton-up:**

При использовании метода «botton-up» для извлечения информации, которую необходимо внести в базу данных, необходимо изучить все имеющиеся на предприятии документы и данные. Этот метод проектирования может привести к множеству очень разных решений. Существует риск того, что решения различных задач будут накладываться друг на друга, и что одни и те же данные будут сохраняться несколько раз в разных таблицах (избыточность). Необходимо избегать избыточности. При изменении какого-либо значения, например, адреса, редактирование отдельной строки базы данных должно обеспечить внесение соответствующих изменений во всей базе данных.

**Примечание:**
Наличие не обязательных строк базы данных называется избыточностью. В реляционных базах данных, избыточность возникать не должна.

**Метод «top-down»**

В данном методе модель данных определяется не отдельным приложением, а сначала определяются требования к информации для всех последующих пользователей базы данных. На основании этих различных требований выводится заключение о том, какая информация должна быть включена в базу данных например, информация о клиентах. Таким образом, для всех приложений базы данных существует только одна таблица Клиенты.

## 3.2 Нормализация

Поля данных, заданные информационной структурой, не могут быть перенесены в таблицу без дальнейшей предварительной работы. Необходимо убедиться, что многократная запись не производится (отсутствует избыточность). Процесс, при котором данные поэтапно разбиваются на разные таблицы, называется нормализацией. При этом описываются различные нормальные формы.

**Примечание:**
При нормализации поля данных постепенно распределяются по разным таблицам в соответствии с правилами логики.

### 3.2.1 Нормальные формы

**Первая нормальная форма**

Если в таблице Персонал в поле Место жительства хранятся как почтовый индекс, так и название населенного пункта, то возникает сразу несколько проблем. Большие города, например, г. Ульм, имеют несколько почтовых индексов, во-вторых, это затрудняет сортировку и поиск адресов по названию населенных пунктов.

Поэтому для начала необходимо сохранить составную информацию в разных полях, в качестве простых, недоступных для разделения значений. Такие значения также называются атомарными значениями.

| Таблица: Персонал | | | |
|---|---|---|---|
| **Номер пассажира** | **Фамилия** | **Имя** | **Место жительства** |
| 1 | Палмер | Карло | 89348 Йеттинген |
| 2 | Мюллер | Герта | 86416 Крумбах |
| 3 | Май | Ханс | 89077 Ульм |
| 4 | Шульц | Анна | 88471 Лаупхайм |
| 5 | Winter | Сюзанна | 89312 Гюнцбург |

Поля содержат 2 значения

| Таблица: Персонал | | | | |
|---|---|---|---|---|
| **Номер пассажира** | **Фамилия** | **Имя** | **Место жительства** | **почтовый индекс** |
| 1 | Палмер | Карло | Йеттинген | 89348 |
| 2 | Мюллер | Герта | Крумбах | 86416 |
| 3 | Май | Ханс | Ульм | 89077 |
| 4 | Шульц | Анна | Лаупхайм | 88471 |
| 5 | Winter | Сюзанна | Гюнцбург | 89312 |

Содержимое поля разделено

Таблица Персонал не нормализована и находится в первой нормальной форме

**Примечание:**
Таблица находится в первой нормальной форме, если ни одно из полей данных не содержит значений, которые можно разделить.

Клиенты, детали на складе, заказы, а также другие объекты и операции могут быть четко определены с помощью поля первичного ключа. При поиске по персональному номеру можно получить все сохраненные данные сотрудников, например, имя и адрес. Атрибуты **фамилия, имя, место жительства** функционально зависят от личного номера.

**Вторая нормальная форма**

Первичный ключ таблицы также может быть составлен из нескольких полей. Таким образом, таблица Позиции заказа содержит как номер заказа, так и номер товара. Комбинация этих двух номеров может быть установлена здесь в качестве первичного ключа. Дополнительные сведения о товаре больше не требуются, так как внешний ключ № товара в таблице Товары в качестве первичного ключа обозначает уникальную запись в строке базы данных.

| Таблица: Позиции заказа | | |
|---|---|---|
| | **Название поля** | **Тип данных поля** |
| 🔑 | номер заказа | Текст |
| 🔑 | номер товара | Текст |
| | объем партии | Целое |
| | цена заказа | Валюта |

Составной первичный ключ: номер заказа и номер товара

Таблица Позиции заказа с составным первичным ключом, узнаваемым по значку ключа.

**Примечание:**
Таблица находится во второй нормальной форме, если:

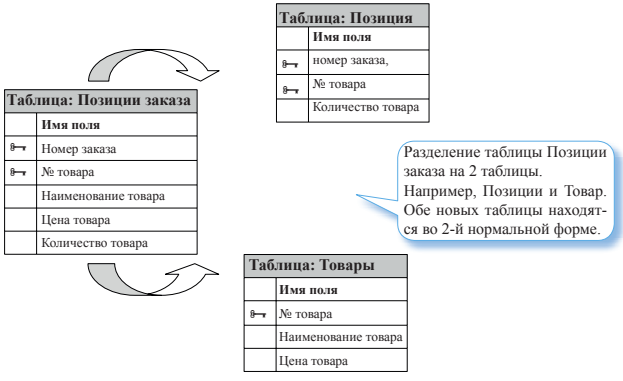
- она находится в первой нормальной форме и
- каждый атрибут можно идентифицировать не только частью первичного ключа, но и целым первичным ключом.

Такая зависимость от ключа называется полной функциональной зависимостью. Таблицы с составным первичным ключом учитываются при разработке второй нормальной формы.

**Примечание:**
Вторая нормальная форма очень важна для таблиц с составными первичными ключами.

Таблица Позиции заказа должна быть переведена во вторую нормальную форму. Для этого, поля, которые не зависят от первичных ключей, хранятся в другой таблице.

В примере таблицы Позиции заказа от обоих ключевых полей зависит только атрибут «Количество». Все данные о товаре уже заданы частью ключа, а именно Номером товара. Они передаются далее в таблицу Товары.




2-я нормальная форма на примере Позиций заказа.

**Третья нормальная форма**

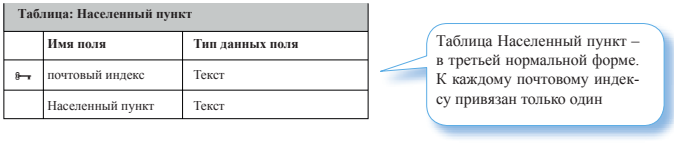
Если таблица находится во второй нормальной форме, то можно начать ее приведение к третьей нормальной форме.

**Примечание:**
Таблица может находиться в третьей нормальной форме при наличии двух условий: таблица уже приведена ко второй нормальной форме и между полями, которые не образуют первичные ключи, отсутствуют связи

Это также называется переходной зависимостью; неключевые атрибуты не зависят функционально от других неключевых атрибутов.

Это требование распространяется на таблицы с простым первичным ключом. Поля данных, зависящие от других неключевых полей, выделяются в отдельную таблицу.

Это возможно в таблице Персонал. В ней атрибут «почтовый индекс» зависит от атрибута «населенный пункт». Данное утверждение верно для небольших населенных пунктов, на большие города, такие как Гамбург или Штутгарт, оно не распространяется. В больших городах почтовый индекс также зависит от улицы или даже почтового ящика клиента. Однако существует и обратная зависимость: населенный пункт зависит от почтового индекса, каждый почтовый индекс точно связан с названием населенного пункта. Если несколько населенных пунктов объединены в одну группу, то это название объединенной общины. Таким образом, поле Населенный пункт может быть выделено из этой таблицы и сохранено в собственной таблице Населенный пункт.



Если все таблицы создаются и редактируются в соответствии с правилами нормализации, то избыточности, безусловно, можно избежать. Нормализация гарантирует правильность логической структуры базы данных.


| Нормальные формы НФ | | |
|---|---|---|
| | **Значение** | **Примечание** |
| 1-я НФ | все значения атомарны (более не делимы) | списки отсутствуют |
| 2-я НФ | каждый неключевой признак зависит от всего первичного ключа | первая нормальная форма достигнута, необходимо обратить внимание только на составные первичные ключи |
| 3-я НФ | каждый неключевой атрибут напрямую зависит от первичного ключа | вторая нормальная форма достигнута, избыточность в таблицах отсутствует |

Недостаток разделения отдельных данных на таблицы состоит в том, что для поиска при помощи СУБД, охватывающих несколько таблиц, требуется много времени. Поэтому в целях увеличения производительности СУБД, может возникнуть необходимость отмены выделения определенных составных частей, например, почтовых индексов. Этот процесс называется **денормализацией**.

### 3.2.2 Пример для нормализации: дистанционная торговля

Компания дистанционной торговли управляет данными счетов с помощью электронных таблиц, например Excel. Счета-фактуры, создаваемые таким образом, содержат все важные данные. Это программное обеспечение может быть заменено базой данных. В данном случае проводится нормализация вплоть до 3-ей нормальной формы.

## ДОСТАВКА ПРОДУКТОВ

П/я 1234, 88471 Лаупхайм, Тел +49739233333, Факс +49739233334

Ральф Бэр
Ан дер Хайде 7
83262 город Н

**Счет-фактура**
Номер клиента
110

Дата: 20.12.2015
Номер счета
342

| Количество | Упаковка | № товара | Обозначение | Наличие на складе | Цена | Стоимость | |
|---|---|---|---|---|---|---|---|
| 2 | 0,75 л | G2 | Красное вино | 3 | 3,99 | 7,98 | € |
| 3 | 6 x 0,5 л | G1 | Пиво | 2 | 6,59 | 19,77 | € |

**Сумма** **27,75 €**
с учетом НДС 5,27 €

Бланк счета-фактуры

**Ненормализованная таблица**

Сначала все данные из бланка счета будут перенесены в не нормализованную таблицу.

| Бланк счета | | Клиенты | | | | | Товар | | | | | | Количество |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| № | Дата | № | Фамилия | Улица | Почтовый индекс | Населенный пункт | № | Упак. | Наименование | Наличие на складе | Цена | НДС | |
| 342 | 20.12.2015 | 110 | Бэр, Ральф | Ан дер Хайде 7 | 83262 | Город Н | G2, G1 | 0,75 л 6 х 0,5 л | Красное вино, пиво | 3, 2 | 3,99 6,59 | 0,19 0,19 | 2, 3 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

**Первая нормальная форма**

Теперь, для того, чтобы привести не нормализованную таблицу к первой нормальной форме, столбцы с неатомарными данными выгружаются в новую таблицу Данные позиции. Результатом являются таблицы Данные заказа и Данные позиции в первой НФ.

| Номер счета | Дата счета | Номер клиента | Имя клиента | Улица | Почтовый индекс клиента | Н/пункт клиента |
|---|---|---|---|---|---|---|
| 342 | 20.12.2015 | 110 | Бэр, Ральф | Ан дер Хайде 7 | 83262 | Город Н |
| ... | ... | ... | ... | ... | ... | ... |

Таблица Данные заказа и Данные позиции в 1-й НФ

| Номер счета | № товара | Упаковка товара | Наименование товара | Наличие товара на складе | Цена товара | НДС | Количество |
|---|---|---|---|---|---|---|---|
| 342 | G2 | 0,75 л | Красное вино | 3 | 3,99 | 0,19 | 2 |
| 342 | G2 | 6 х 0,5 л | Пиво | 2 | 6,59 | 0,19 | 3 |
| ... | ... | ... | ... | ... | ... | ... | ... |

В таблице Данные заказа столбец Номер счета формирует первичный ключ. В таблице Данные позиции первичный ключ содержится в столбце Номер счета и Номер товара, это называется составным первичным ключом.

**Вторая нормальная форма**

В таблице Данные позиции атрибуты товаров зависят от части составного ключа, а именно от номера товара. То есть, эта таблица не находится во второй нормальной форме. Для нормализации и приведения таблицы ко 2-й НФ Номер товара и все зависящие от него атрибуты, объединяются в отдельной таблице с ключом Номер товара. Так формируются таблицы Товар и Позиция.

| № товара | Упак. товара | Наименование товара | Наличие товара на складе | Цена товара | НДС |
|---|---|---|---|---|---|
| **G2** | **0,75 л** | **Красное вино** | **3** | **3,99** | **0,19** |
| G3 | 6 х 0,5 л | Пиво | 72 | 6,59 | 0,19 |
| ... | ... | ... | ... | ... | ... |

| Номер счета | № товара | Количество |
|---|---|---|
| 342 | G2 | 2 |
| 342 | G3 | 3 |
| ... | .. | .. |

Таблицы Товары и Позиции во 2-й нормальной форме

**Примечание:**
2-я нормальная форма важна только для таблицы с составным первичным ключом.

**Третья нормальная форма**

Таблица Данные заказа не находится в 3-й нормальной форме, поскольку ряд атрибутов зависит от Номера клиента. Сведение атрибутов, зависящих от Номера клиента с использованием в качестве ключа номера клиента, в таблицу Клиенты решает эту проблему. В третьей нормальной форме в таблице Сведения о заказе остаются атрибуты номер, выставленный счет и номер клиента.

| Номер клиента | Имя клиента | Улица | Почтовый индекс клиента | Н/пункт клиента |
|---|---|---|---|---|
| 110 | Бэр, Ральф | Ан дер Хайде 7 | 83262 | Город Н |
| ... | ... | ... | ... | ... |

Результатом нормализации базы данных на примере предприятия дистанционной торговли является создание 5 таблиц, каждая из которых находится в третьей нормальной форме. Неключевые атрибуты каждой таблицы полностью функционально зависят от первичного ключа. Соответствующие функциональные зависимости представлены стрелками. Атрибуты указаны нижним регистром. Существуют и более высокие нормальные формы. Однако они практически не применяются.

**Примечание:**
В таблицах, которые находятся в3-ей нормальной форме, все зависимости от ключей или отношений сводятся ко внешним ключам.

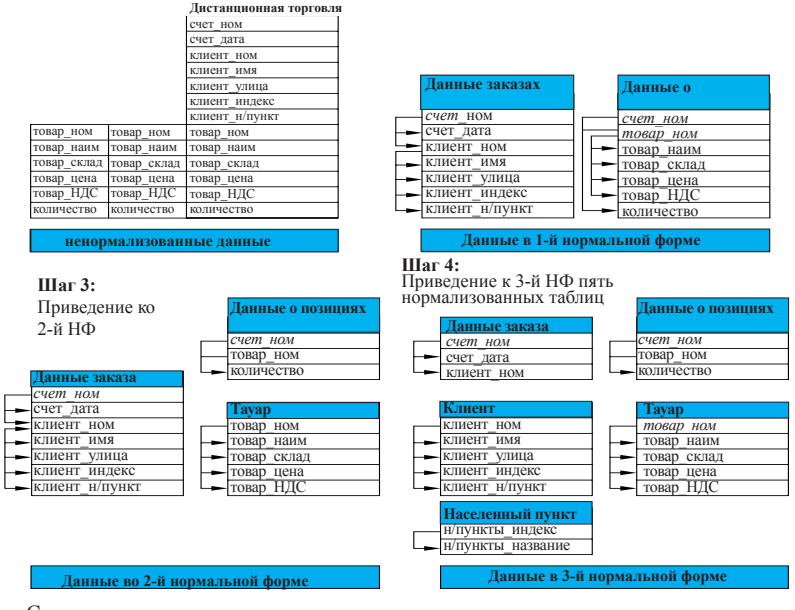
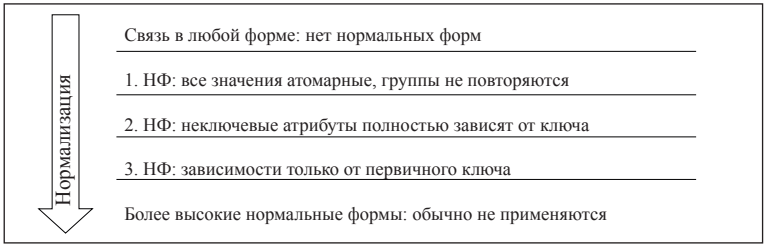



Схема процесса нормализации




### 3.2.3 Другие нормальные формы

Существуют также четвертая и пятая нормальные формы, но они не так популярны, поскольку маловероятно, что в таких ситуациях следует ожидать нарушения нормальных форм. Прежде всего потому, что подобных нарушений можно избежать интуитивно. В частности, если мы выбираем метод, при котором сначала создается ER-модель, которая затем будет преобразована в реляционный проект, то достаточно будет провести тестирование на третьей нормальной форме. Нарушения на уровне второй или третьей нормальной формы возможны, но их в конце концов можно исключить путем проверки. Практически во всех соответствующих случаях можно предположить, что условия четвертой и пятой нормальной форм не будут нарушены.

### 3.2.4 Условия целостности

Требование целостности базы данных нацелено на то, чтобы сохраненный образ данных полностью соответствовал действительности. Неправильные записи, игнорирование обслуживания (обновлений) ставят под сомнение полезность базы данных.

- Типы данных и диапазоны значений являются важным инструментом на уровне отдельных атрибутов для обнаружения и исключения ошибок данных.
- Другой класс возможных ошибок избегается структурными свойствами базы данных в сочетании с проверкой свойств ключа и внешних ключей (ссылочная целостность).

**Ссылочная целостность**

Ссылочная целостность данных – это свод правил, с помощью которых система управления базами данных, напр. MS Access, «убеждается», что связи между записями в детальных таблицах действительны и что связанные данные не были случайно удалены или изменены. Ссылочная целостность достигнута, если выполняются все следующие условия:

- Соответствующее поле из главной таблицы является первичным ключом или имеет уникальный индекс.
- Обе таблицы принадлежат к одной базе данных. Чтобы установить ссылочную целостность, открывается база данных, в которой хранятся таблицы. Ссылочная целостность не может быть применена к связанным таблицам из баз данных других форматов.

При соблюдении ссылочной целостности применяются следующие правила:

- В поле внешнего ключа детальной таблицы нельзя ввести значение, не содержащееся в поле первичного ключа главной таблицы. Однако можно ввести нулевое значение в поле внешнего ключа, указав, что записи не связаны друг с другом. Так, например, невозможно создать заказ, не связанный с существующими клиентами. Однако, может быть создан заказ, который связан с «никем», если поле номер клиента содержит нулевое значение.
- Запись из главной таблицы не может быть удалена, если соответствующие строки базы данных включены в детальную таблицу. Например, невозможно удалить запись о сотруднике из таблицы Персонал, если с этим сотрудником связана строка базы данных в таблице Данные заказа.
- Значение первичного ключа в главной таблице не может быть изменено, если к этой записи имеются записи сведений. Например, невозможно изменить номер сотрудника в таблице Персонал, если с этим сотрудником связаны записи в таблице Заказы.

## 3.3 Упражнения к Главе 3

1.
Ознакомьтесь с таблицей 1:
a) Какие из нормальных форм содержат нарушения?
b) Создайте эквивалентную систему в нормализованной форме. Таблица 1

| ISBN | Авторы | Название | Год выпуска | Страницы |
|---|---|---|---|---|
| 0-201-14192-2 | Дата | Реляционная модель для управления базами данных: Издание 2 | 1990 | 538 |
| 3-89319-117-8 | Финкенцеллер Х., Краке У., Унтерштайн М. | Систематическое применение SQL-Oracle | 1989 | 494 |
| 1-55860-245-3 | Мелтон Й., Саймон А. | Понимание нового SQL | 1993 | 536 |

2.
Ознакомьтесь с таблицей 2:
a) Какие из нормальных форм содержат нарушения?
b) Создайте эквивалентную систему в нормализованной форме. Таблица 2

| Инд. № студента | Студент | № курса | Название курса | Оценка |
|---|---|---|---|---|
| 30321 | Й. Мейер | 70656 | Системы баз данных | 1,0 |
| 30321 | Й. Мейер | 71554 | Разработка программного обеспечения | 1,7 |
| 30346 | Х. Аренс | 71554 | Разработка программного обеспечения | 3,0 |
| 30346 | Х. Аренс | 70656 | Системы баз данных | 2,0 |
| 30346 | Х. Аренс | 71355 | Реляционное и процедурное программирование | 1,7 |
| 30378 | К. Кнудсен | 70656 | Системы баз данных | 2,0 |

3. Учет оборудования

Следующая иллюстрация показывает данные инвентаризации оборудования (например, одной из школ). Данные должны храниться в реляционной базе данных в нормализованной форме. (Данные порта имеют следующее значение: M1 = IDE Port 1 Master, S1 IDE Port1 Slave, ...)

| **Системный блок** | | **Ответственный** | | | **Жесткие диски** | | | **IDE-порт** |
|---|---|---|---|---|---|---|---|---|
| **Инвент. №** | **Место нахождения** | **Инд. №** | **Фамилия** | **Телефон** | **Производитель** | **№ продукта** | **Емкость** | |
| L1001 | B246 | F100 | М. Мейер | 8975 | Western Digital | 102BA | 1TB | M1 |
| | | | | | Western Digital | 102BA | 1TB | S1 |
| | | | | | Fujitsu | MPF3204 AT | 2TB | M2 |
| | | | | | Fujitsu | MPF3204 AT | 2TB | S2 |
| L1003 | B251 | F101 | Ф. Биндер | 5635 | IBM | DTLA-305020 | 2TB | M1 |
| | | | | | IBM | DTLA-307045 | 4TB | M2 |
| | | | | | Western Digital | 102BA | 1TB | S1 |

Для представленного массива данных необходимо разработать нормализованные таблицы для реляционной базы данных. Придерживайтесь следующего алгоритма действий:

1. Шаг 1: создайте первую нормализованную форму, разделив массив данных на 2 таблицы. В качестве результата укажите отношения с их атрибутами и отметьте первичный и внешний ключ.
2. Шаг 2: если необходимо, создайте вторую нормальную форму для отношений из шага 1. В качестве результата снова укажите отношения с их атрибутами и пометьте первичный и внешний ключи.
3. Шаг 3: При необходимости создайте 3-ю нормальную форму для отношений из шага 2 и, в качестве общего результата, укажите все отношения с их атрибутами, которые теперь соответствуют исходному массиву данных в нормализованной форме.

**Вопросы:**

Какие избыточности могли быть устранены путем нормализации?

В таблицах (после шага 1), которые соответствуют 1-й, но не 2-й НФ, возникают проблемы при вставке, изменении и удалении записей. Приведите примеры каждой из форм на основе этих таблиц.

**4. Список багажа авиакомпании**

Произведите пошаговую нормализацию списка багажа авиакомпании и приведите его к третьей нормальнйо форме

| Номер рейса: | *WA876* | **Тип самолета** | B 747 |
|---|---|---|---|
| из: | *Мюнхен* | **КВС** | *Эбелинг* |
| в: | *Гонолулу* | | |
| Дата рейса: | *01.04.2015* | | |

| **Пассажир** | | | | | | **Багаж** | | |
|---|---|---|---|---|---|---|---|---|
| Номер удостоверения личности | Фамилия | Имя | Улица | Почтовый индекс | Населенный пункт | Номер багажа | Тип | Вес (кг) |
| 89021238 | Бреценхубер | Ральф | Бекерштрассе 9 | 81669 | Мюнхен | 1 | Чемодан | 20,5 |
| | | | | | | 2 | Доска для серфинга | 6,5 |
| | | | | | | 3 | Рюкзак | 7 |
| 74921209 | Хиптхалер | Ханс | Бройштрассе 6 | 85049 | Ингольштадт | 4 | Горный велосипед | 35 |
| | | | | | | 5 | Чемодан | 12 |
| 96554367 | Майя | Маттиас | Брайнштрассе 12 | 85049 | Ингольштадт | 6 | Ангельруте | 1,2 |

Ненормализованный массив данных

| № | из: | в: | дата: | FT | FK | Номер пассажира | N | VN | Улица | Почтовый индекс | Населенный пункт | Регистр. номер глобального имени | A | G |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WA876 | MUC | HON | 1.4.15 | B747 | Эбелинг | 89… | Брезень | Ральф | Бекерштрассе 9 | 81669 | Мюнхен | 1 | Чемодан | 20,5 |
| | | | | | | | | | | | | 2 | Доска для серфинга | 6,5 |
| | | | | | | | | | | | | 3 | Рюкзак | 7 |
| | | | | | | 74… | Хиптт | Ханс | Бройштрассе 6 | 85049 | Ингольштадт | 4 | Горный велосипед | 35 |
| | | | | | | | | | | | | 5 | Чемодан | 12 |
| | | | | | | 96… | Майя | Мат. | Брайнштрассе 12 | 85049 | Ингольштадт | 6 | Ангельруте | 1,2 |

**5. Дана ненормализованная таблица**. Выполните пошаговую нормализацию и приведите таблицу к третьей нормальной форме

| Личный номер | Фамилия | Имя | № отдела | Отдел | № проекта | Описание |
|---|---|---|---|---|---|---|
| 1 | Лоренц | София | 1 | Персонал | 2 | Продвижение продаж |
| 2 | Холь | Татьяна | 2 | Покупка | 3 | Конкурентный анализ |
| 3 | Вильшрайм | Тео | 1 | Персонал | 1,2,3 | Опрос клиентов, продвижение продаж, Конкурентный анализ |
| 4 | Рихтер | Ханс | 3 | Продажа | 2 | Продвижение продаж |
| 5 | Визенланд | Вернер | 2 | Покупка | 1 | Опрос клиентов |

6. Даны следующие ненормализованные связи, которые содержат данные о сотрудниках и их принадлежность к отделам. Кроме того, отмечено участие(я) отдельных сотрудников в проектах всей компании, а также то, какой отдел ответственный за определенные проекты.

a) Выполните пошаговую нормализацию и приведите массив данных к третьей нормальной форме.

Примечание. При отображении производных связей ограничьтесь соответствующими именами атрибутов.

b) Вставьте значимые первичные ключи.

c) На основе правил формирования нормальных форм объясните Ваши шаги

| ФАМИЛИЯ | ИМЯ | ДАТА РОЖДЕНИЯ | ДАТА ПРИЕМА НА РАБОТУ | НАЗВАНИЕ ОТДЕЛА | РУКОВОДИТЕЛЬ ОТДЕЛА | ИМЯ ПРОЕКТА | ЗАПУСК ПРОЕКТА | ПРОЕКТНО-КОНСТРУКТОРСКИЙ ОТДЕЛ |
|---|---|---|---|---|---|---|---|---|
| Зойль | Зигги | 04.05.65 | 01.06.90 | Контроль | План | Сбалансированная система показателей для всего концерна, Разработка Системы управления информацией | 01.00 04.01 | Управление информатики |
| Возврат | Руди | 21.12.60 | 01.06.87 | Информатика | Маус | Разработка Системы управления информацией | 04.01 | Информатика |
| План | Петер | 09.03.61 | 01.03.84 | Контроль | План | Подготовка Первичное размещение акций | 12.99 | Контроль |
| ЕхаКt | Эрика | 22.07.69 | 01.09.98 | Контроль | План | Подготовка BSC Первичное размещение акций | 01.00 12.99 | Контроль Контроль |
| Корректировка | Курт | 30.01.55 | 01 01 72 | Финансы | Корректировка | Подготовка Первичное размещение акций | 12.99 | Контроль |
| Маус | Михаэль | 09.10.50 | 01.01.90 | Информатика | Маус | Подготовка Первичное размещение акций | 12.99 | Контроль |
| Excel | Эмиль | 16.04 68 | 01.09.97 | Контроль | План | Подготовка Первичное размещение акций Разработка Системы управления информацией Сбалансированная система показателей, ССП для всего концерна | 12.99 04.01 01 00 | Контроль Информатика Контроль |
| Гиро | Герд | 03 09 73 | 01.09.99 | Финансы | Корректировка | Предварительное размещение акций | 12 99 | Контроль |

**7. Дан ненормализованный массив данных об учебном предложении, содержащем данные о студентах, классах, преподавателях и времени обучения.**

Выполните пошаговую нормализацию и приведите массив данных к третьей нормальной форме. Связь: учебное предложение

| № ученика | Фамилия | Имя | Класс | Преподаватель | № учебного курса | Описание | Кол-во часов |
|---|---|---|---|---|---|---|---|
| 1 | Юргенс | Ина | 11a | Лемпель | 2 | Танцы | 12 |
| 2 | Шмидт | Том | 12a | Брайер | 3 | Хор | 22 |
| 3 | Егер | Франц | 11a | Лемпель | 1, 2, 3 | Электроника, танцы, хор | 15, 12, 2 |
| 4 | Ольсен | Ина | 11b | Зоммер | 2 | Танцы | 5 |
| 5 | Юргенс | Паула | 12a | Брайер | 1 | Электроника | 23 |

# 4 Программное обеспечение для моделирования баз данных

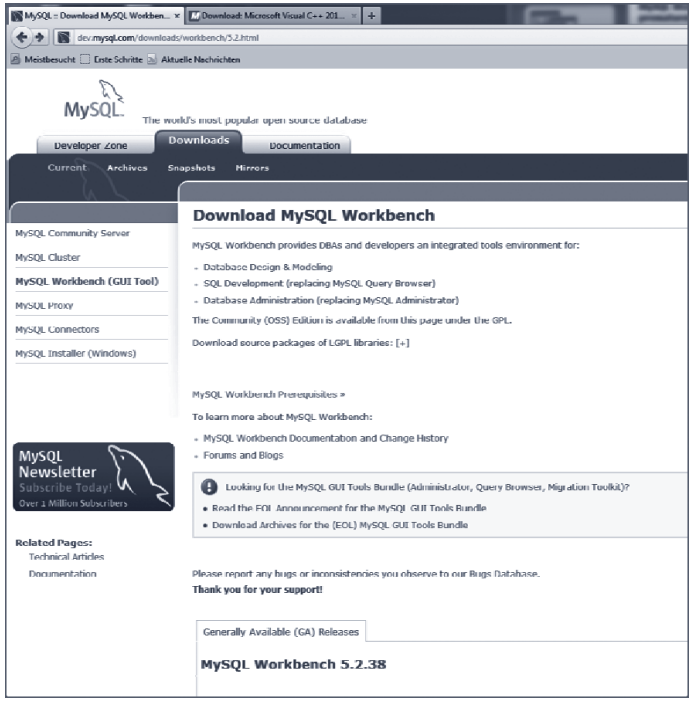
Программное обеспечение для моделирования баз данных обеспечивает поддержку процесса разработки баз данных. С подходящими пакетами программного обеспечения весь процесс разработки осуществляется при помощи вычислительной техники. Типичными программами для проектирования баз данных являются DB-Designer и Microsoft VISIO.

## 4.1 DB-Designer

Программный пакет MySQL Workbench можно получить через Интернет. С помощью программного обеспечения выполняется проектирование и моделирование баз данных, программирование SQL и администрирование баз данных MySQL.

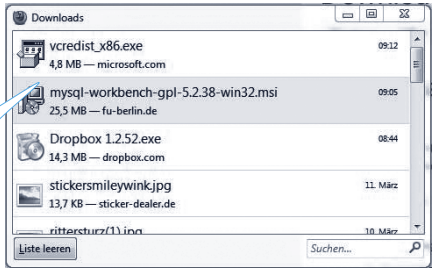
### 4.1.1 Загрузка и установка

Системные требования и полная документация доступны на домашней странице MySQL.



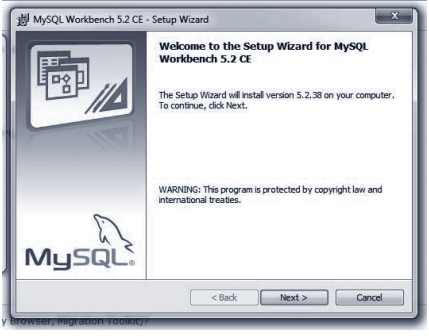

После загрузки msi-пакета (Установщик Windows – Microsoft Software Installer) этот пакет готов к установке.

Дважды кликните на установщик



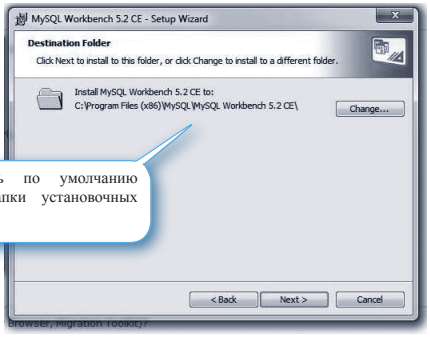

**Экран приветствия**

Процесс установки сопровождается мастером установки (Setup Wizard). Нажмите Next >, чтобы начать процесс автоматической установки.



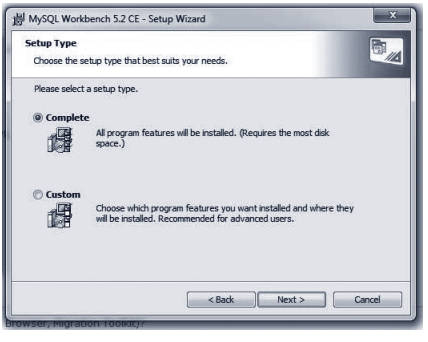

**Destination Folder – папка назначения**

Сначала выберите папку для установки. Если необходимо изменить папку по умолчанию, то Вы можете выбрать любую папку, нажав на Change.



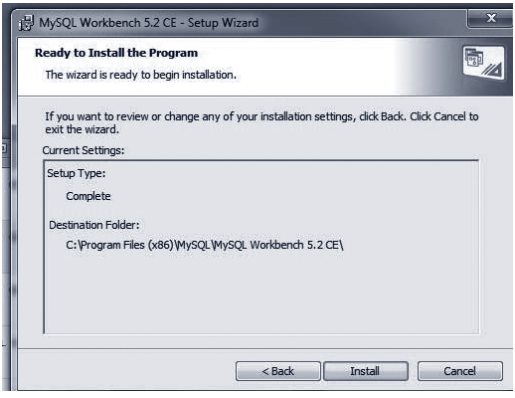

**Setup Type – тип установки**

Далее выберите тип установки: «Complete» для полной установки и «Custom» для установки отдельных программных компонентов на выбор.

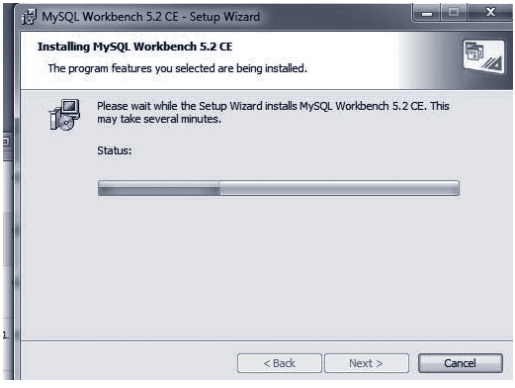


**Ready to Install the Program – Готов к установке программы**

При нажатии на Install начнется копирование файлов в папку по умолчанию или в выбранное вами местоположение.

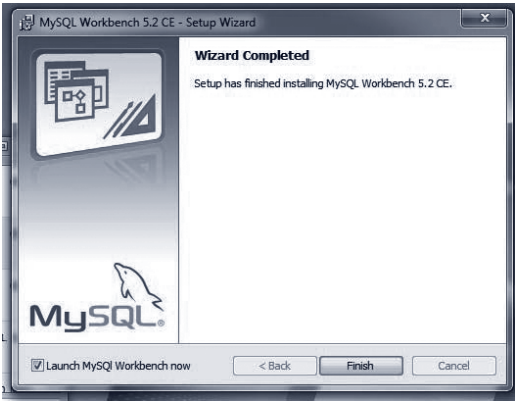



Информация о состоянии установки отображается на следующем этапе установки.




**Wizard Completed – Установка завершена**

После завершения установки мастер сообщает, что установка была завершена. Флажок Launch MySQL Workbench now – «Запустить MySQL Workbench» – стоит по умолчанию. После нажатия на кнопку Finish загружается главный экран MySQL Workbench.




Разработка с помощью SQL: создание связей с уже существующими базами данных SQL.

Моделирование данных с помощью EER (расширенная модель типа «сущность-связь»).

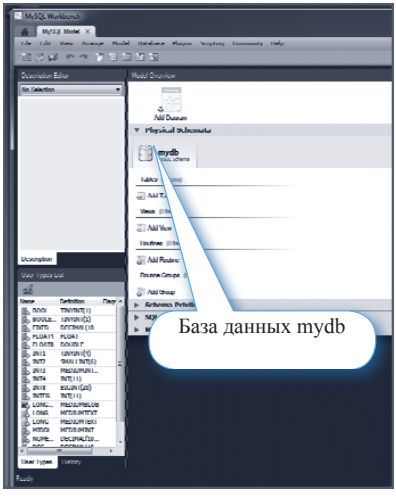
Администрирование SQL-сервера.

Новое подключение к базе данных SQL, редактирование таблиц, редактирование скриптов SQL, управление связями.

Разработка новой EER-модели, разработка новой EER-модели из существующей базы данных, разработка новой EER-модели из сценария SQL.

Создание, управление или экспорт нового экземпляра сервера, управление настройками безопасности.

Полный пакет программ состоит из описанных выше программных компонентов: SQL Development,Data Modeling и Server Administration.

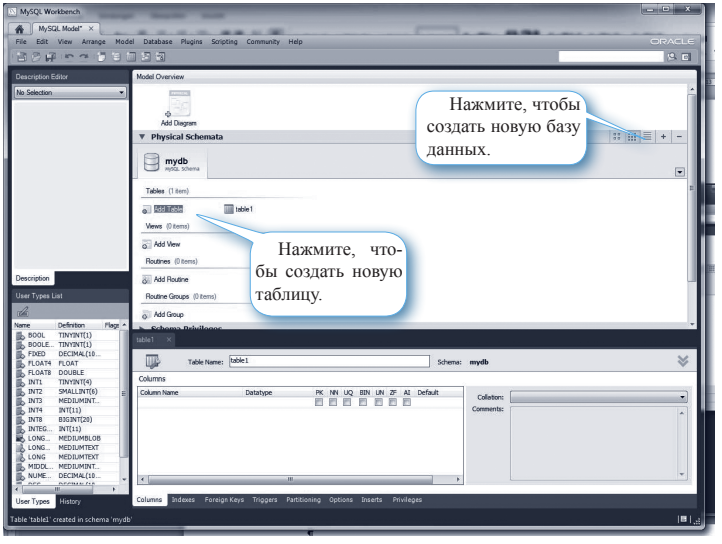



### 4.1.2 Создание таблиц

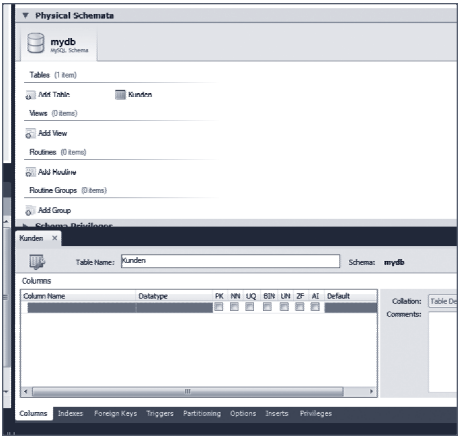
Для моделирования данных выберите пункт меню Create New EER Model (создать новую расширенную модель типа «сущность-связь»).

По умолчанию база данных mydb уже создана.

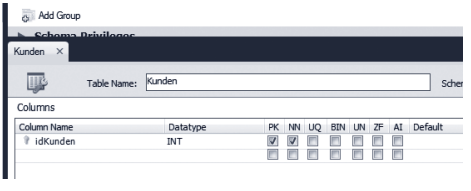
Можно создать новую базу данных, нажав на символ «+». Для создания новой таблицы в базе данных mydb, например, таблицы Клиенты, дважды кликните на Add Table.




После двойного щелчка появляется новая Таблица Table1; название может быть изменено на требуемое, например, Клиенты. Столбец Columns определяет столбцы таблицы с именами столбцов (Column Name) и типом данных (Datatype). В качестве полей выбора доступны параметры PK Primary Key, NN Not Null, UQ Unique, BIN, UN, ZF и AI.

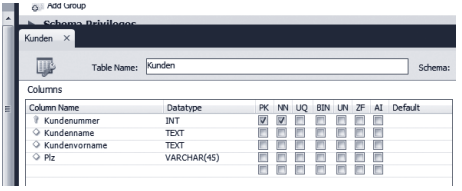


Система автоматически создает столбец с первичным ключом (IdКлиент). Поле первичного ключа имеет флажки PK (Primary Key) и NN (Not Null).

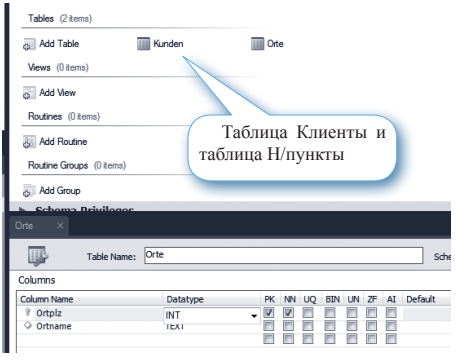


Имя столбца, который по умолчанию называется idКлиент, может быть изменено, например на Номер клиента.

Могут быть созданы дополнительные поля, например, Фамилия клиента, Имя клиента и Почтовый индекс с соответствующими типами данных.

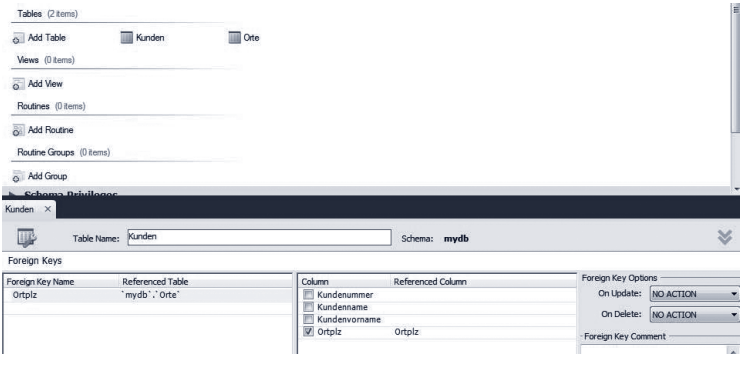



Как описано выше, можно создать вторую таблицу, например Населенные пункты, и задать такие атрибуты, как Почтовый индекс и Населенный пункт.




### 4.1.3 Реляционное связывание таблиц

Две таблицы Клиенты и Н/пункты находятся в отношении 1:m, один Индекс указывает на нескольких Клиентов, но к каждому Клиенту привязан только один определенный Индекс(1).

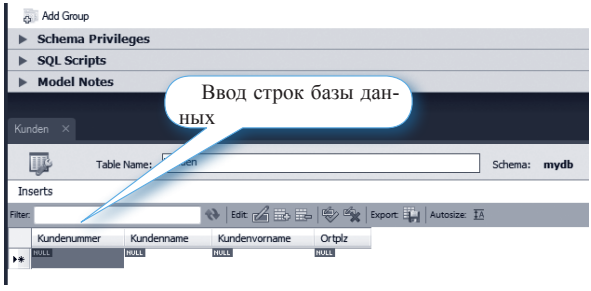



Отношения между таблицами могут быть созданы как вручную, так и автоматизированно с помощью программного обеспечения. На вкладке `Foreign Keys` можно вручную указать внешние ключи.

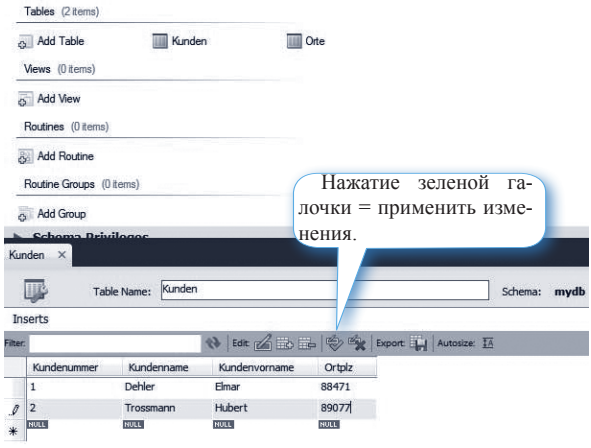
В разделе `Foreign Keys` в качестве `Foreign Key Name` (название внешнего ключа) используется обозначение внешнего ключа, здесь: Индекс и соответствующая таблица (Referenced Table), здесь: Н/пункты. В правой области выбирается соответствующий столбец Column. В разделе Foreign Key Options можно задать поведение ссылочных значений при изменении (`ON Update`) и удалении (`On Delete`) записей. Возможные варианты: `NO ACTION` = действие не требуется, `CASCADE` = переправить дальше, `RESTRICT` = запрет и `SET NULL` = установить на ноль.

### 4.1.4 Запись строк базы данных

Ввод строк базы данных осуществляется посредством нажатия на вкладку `Inserts`.

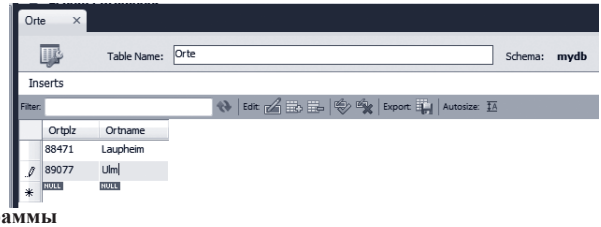



После ввода, например, двух записей, сохранение производится нажатием зеленой галочки Apply changes to data (Применить изменения к данным).




Таким же образом, можно вносить записи в таблицу H/пункты

.

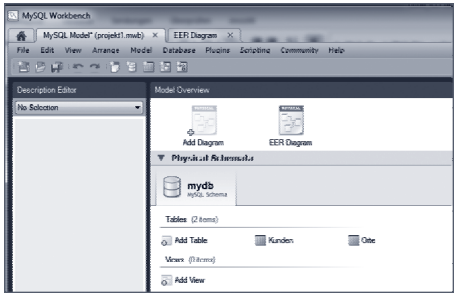



### 4.1.5 Создание ER-диаграммы

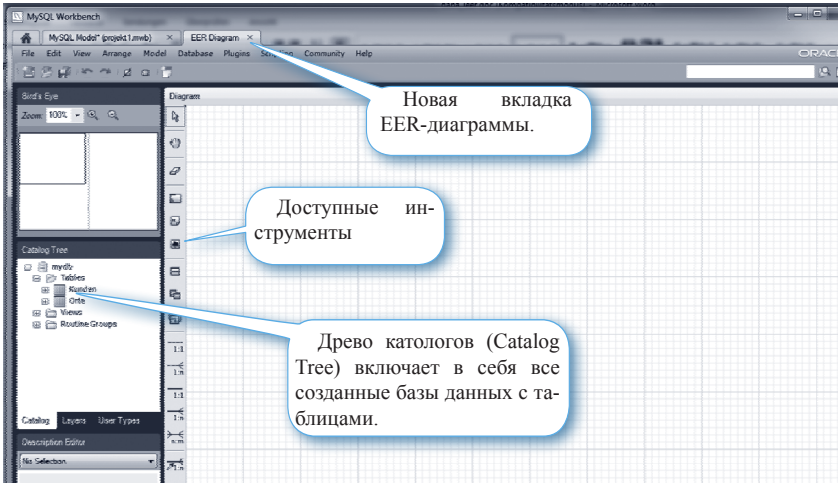
При помощи команды Add Diagram создается диаграмма базы данных, содержащая две таблицы.

Клиенты и H/пункты.

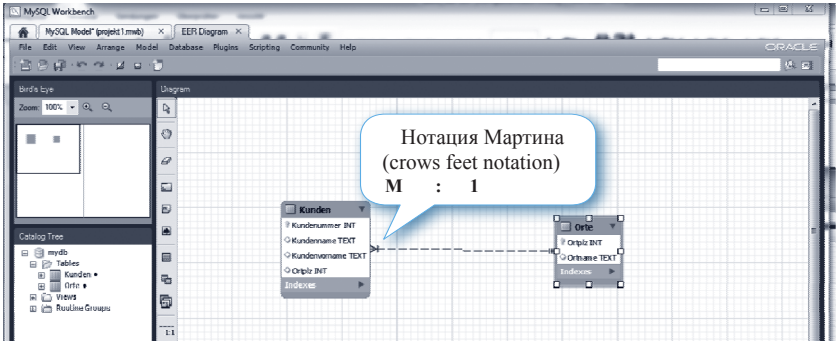
Двойной щелчок на Add Diagram создает новую EER-диаграмму в виде значка и одновременно открывает новую вкладку с надписью EER Diagram.



На рисунке показано окно редактора EER-диаграмм.

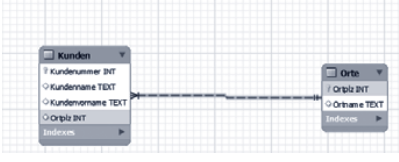



Метод Drag-and-Drop позволяет перетаскивать уже созданные таблицы Клиенты и Н/пункты на рабочий лист.

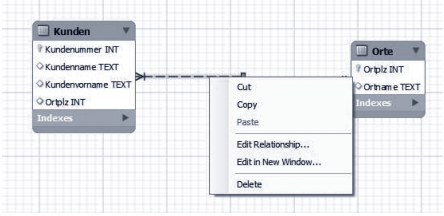



Графическое представление (здесь: в виде нотации Мартина) помогает верно отобразить отношение 1:m между двумя таблицами, «гусиная лапка» обозначает отношение M.

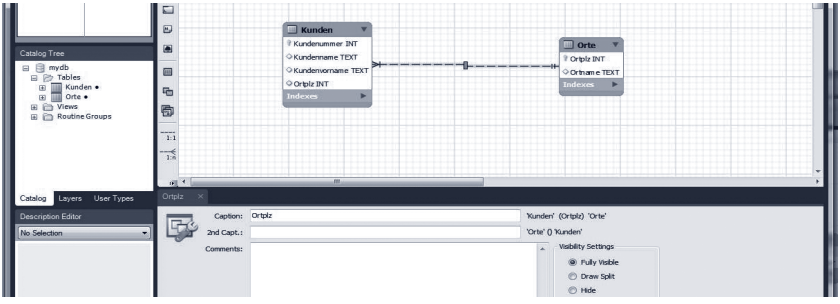
При наведении курсора на линию отношения на диаграмме между таблицами, она становится сплошной и отображает ключевые атрибуты Клиент.Индекс в качестве внешнего ключа и Н/пункт.Индекс в качестве первичного ключа, участвующего в отношении. Для упрощения навигации в левой области («Птичий глаз») отображается расположение двух таблиц по отношению ко всему листу.



При наведении курсора на линии отношений на диаграмме между таблицами и щелчке правой кнопкой мыши появится контекстное меню.




Пункт меню `Edit Relationship` ... позволяет отредактировать отношение (Caption)




В EER-диаграмме, как показано в окне mydb, можно вставлять и создавать таблицы. В левой части окна диаграммы доступны графические значки. Перетащите значок таблицы на рабочий лист для создания новой таблицы.

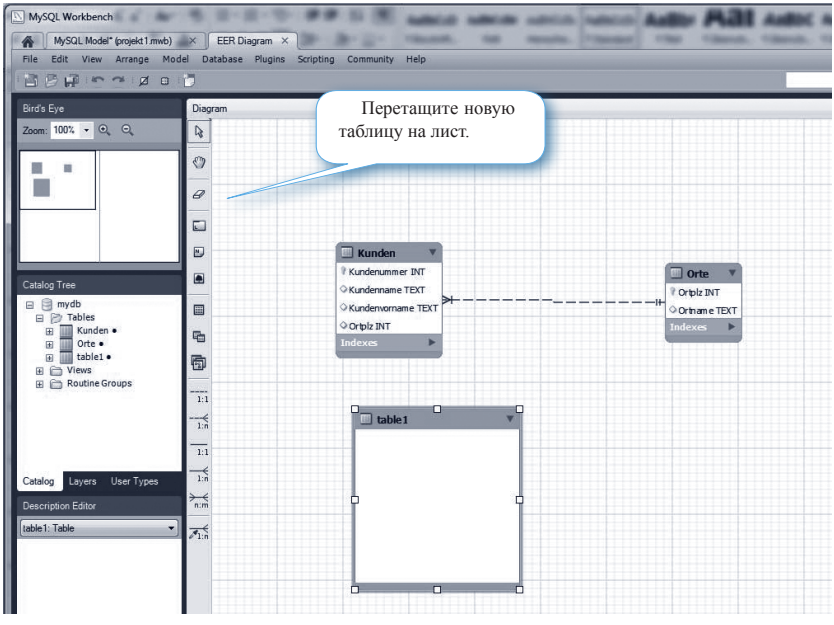
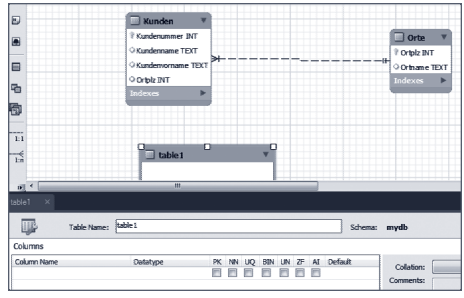
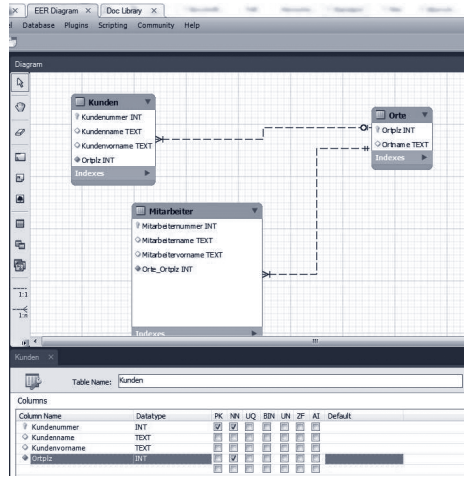


Таблица `table1` открывается двойным щелчком в режиме редактирования. Столбцы таблицы (атрибуты) и их типы данных можно вводить описанным выше способом.



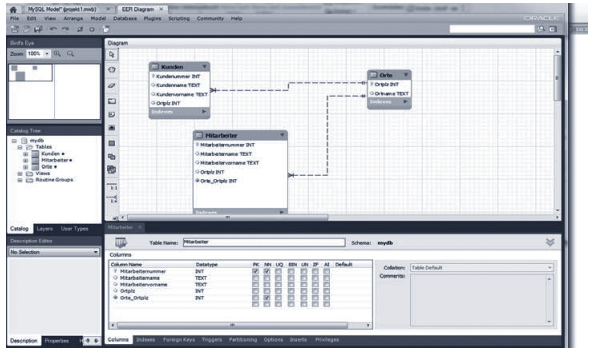

Далее создается новая таблица `table1` в виде таблицы Сотрудники с теми же атрибутами, что и у таблицы Клиенты.



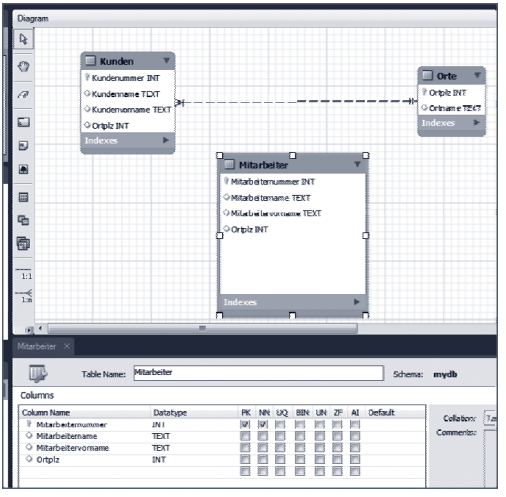

Для реляционного подключения новой таблицы Сотрудники к таблице Н/пункты на панели инструментов в левой части окна Диаграммы. В таблице Сотрудники и таблице Н/пункты отображается индекс населенных пунктов. Результат показан на рисунке. Следует отметить, что система самостоятельно создает новый атрибут внешнего ключа (здесь: `н/пункт_индекс`).

**Примечание:**
При создании БД с помощью EER-диаграммы внешний ключ не должен создаваться с атрибутами, атрибуты устанавливаются при реляционном соединении.

Для полноты верхний атрибут Индекс-удаляется.

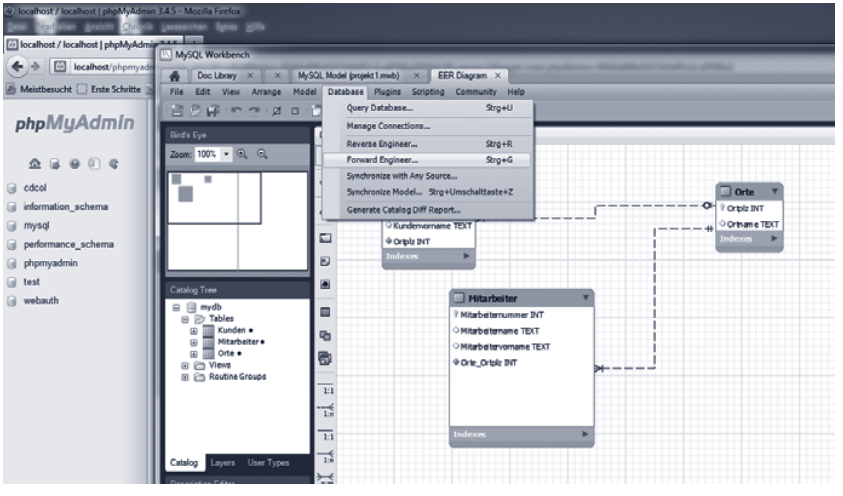


Кроме того, новый внешний ключ визуально выделяется другим цветом шрифта. Таким же образом должны создаваться данные в таблице Клиенты. Двойным щелчком по значку таблицы Клиенты в нижней части окна запускается редактор. При повторном нажатии на ромб перед именем столбца (`Column Name`) цвет ромба изменяется. Белый цвет означает неключевой атрибут, красный цвет – означает внешний ключ (флажки `NN Not Null`), или символ ключа – для обозначения первичного ключа.




### 4.1.6 Forward Engineering (Прямое проектирование)

Forward Engineering предлагает методы и инструменты для разработки систем и программного обеспечения, например: построение реализуемой базы данных на основе эскиза (модели) базы данных. Это включает в себя имплементацию базы данных в СУБД или генерацию исходного SQL-кода.

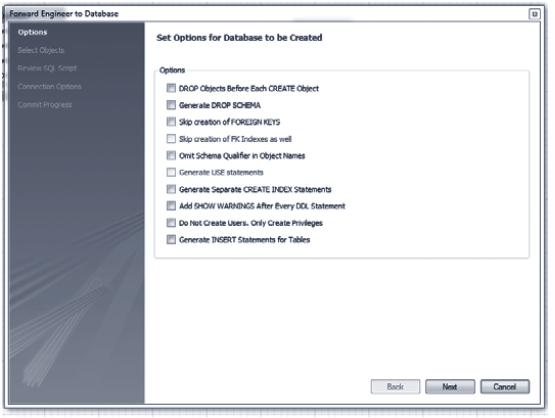



В меню База данных находится подменю Forward Engineer….

Запустить мастер можно, выбрав Forward Engineer или нажав комбинацию клавиш CTRL+G.

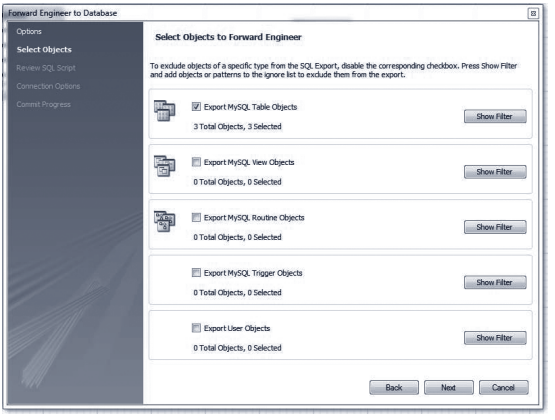
**Параметры / Установить параметры для создаваемой базы данных**

В первом окне будут выбраны параметры для создания базы данных.



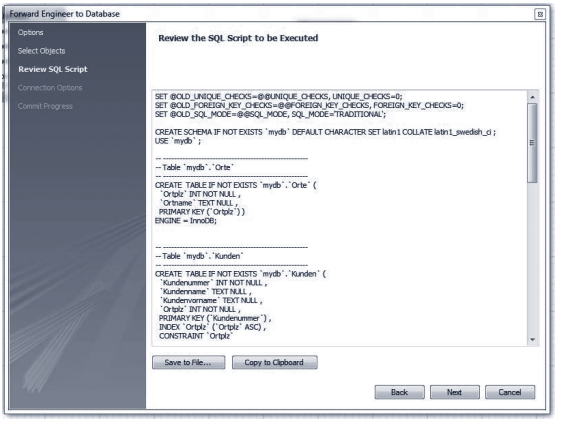

**Select Objects / выберите объекты для Forward Engineer**

После этого будут выбраны объекты, экспортированные с помощью опции Forward Engineering.

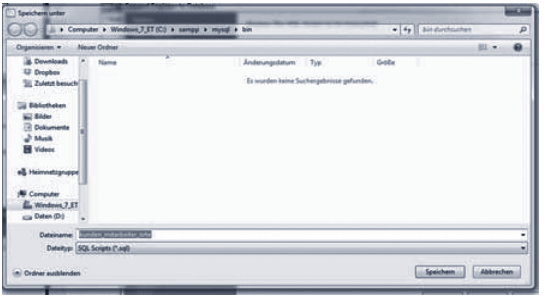



**Review SQL Script / Просмотр SQL-скрипта, который будет выполнен**

Созданный SQL-скрипт можно просмотреть на следующем шаге. При нажатии на кнопку `Save to File...` «Сохранить в файл...» создается файл SQL, а при выборе команды `Copy to Clip board...` «Копировать в буфер обмена» файл SQL копируется на рабочий стол.



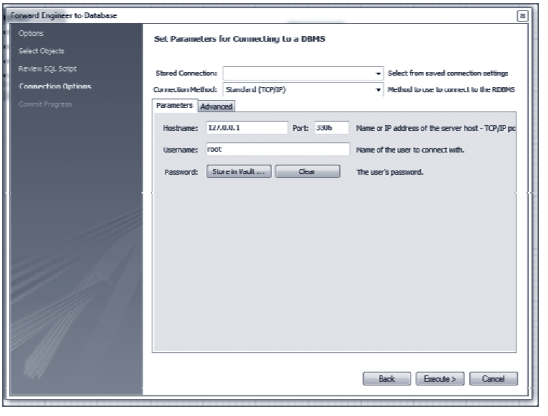
В окне Сохранить как... введите имя файла, например, Клиенты_Сотрудники_Н/пункты и путь хранения.




**Connection Options / Задать параметры для подключения к СУБД**

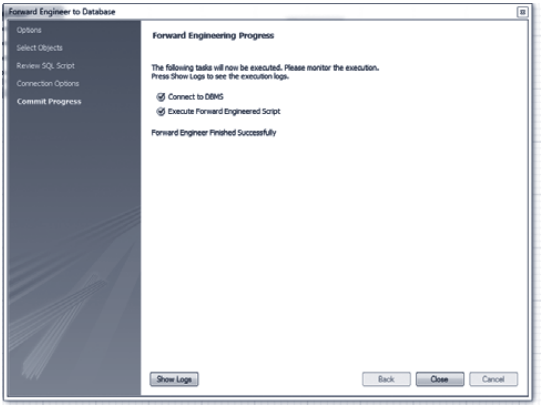
В разделе Параметры подключения вводятся параметры подключенного сервера SQL, например, Имя хоста и Номер порта. При нажатии на Выполнить запускается указанный процесс, т. е. подключение к СУБД и выполнение SQL-сценария.

Важно: веб-сервер и MySQL должны быть запущены.




**Commit Progress / Прогресс операции Forward Engineering**

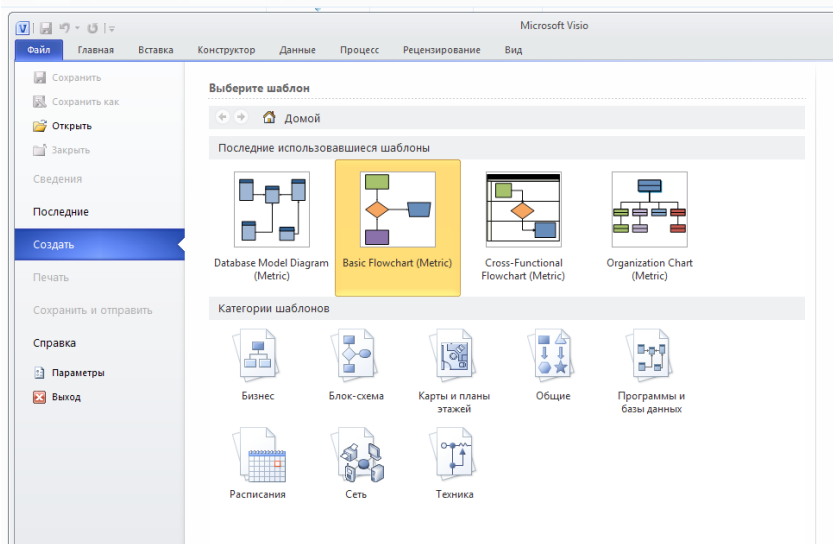
Прогресс процедуры `Forward Engineering` отображается на экране `Forward Engineering Progress`.



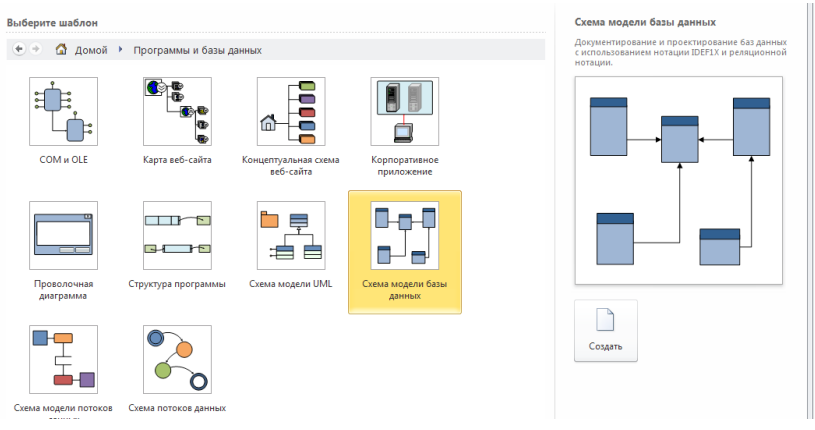
## 4.2 Microsoft VISIO

### 4.2.1 Запуск диаграммы модели базы данных

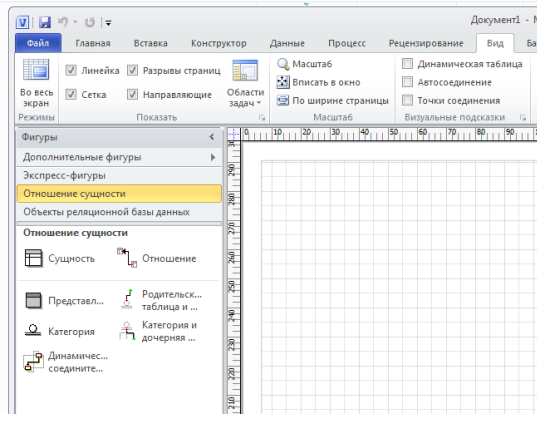
После запуска Microsoft Visio 2010 выберите значок Программное обеспечение и База данных.




Нажмите Выбрать шаблон, чтобы выбрать шаблон диаграммы для модели БД и подтвердите его нажатием кнопки Создать. Шаблон диаграммы для модели базы данных позволяет создать новую модель отношения сущностей (ER-модель) или модель на основе уже существующей базы данных, с использованием обратного проектирования (Reverse Engineering).



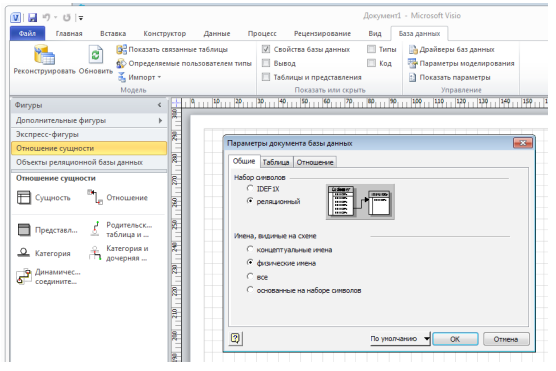
В левой области находятся шаблоны, также называемые фигурами с соответствующими инструментами разработки, в правой области – рабочий лист, на котором создается модель ER.




Шаблон «Сущность–связь» (Entity Relationship) в основном используется для моделирования баз данных на основе стандарта SQL92 и предыдущих стандартов.

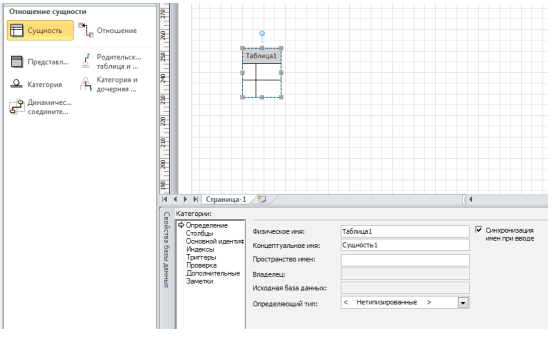
Реляционно-объектный шаблон содержит дополнительные инструменты для моделирования баз данных на основе стандарта SQL99 и более поздних стандартов.

На вкладке База данных в группе Управление можно выбрать параметры отображения. В открывшемся окне Параметры документа – базы данных можно выбрать набор символов в диалоговом окне, например Реляционный и другие параметры таблиц и отношений.

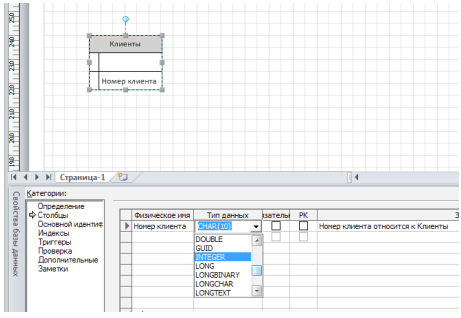



### 4.2.2 Создание таблиц

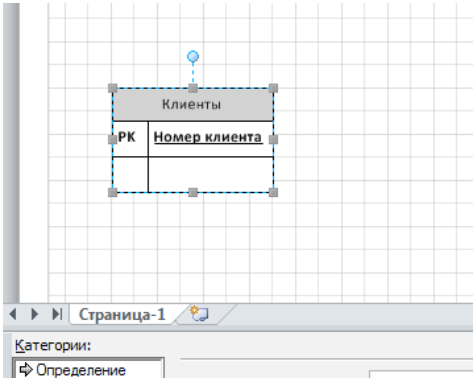
Из шаблона «Отношения сущностей» значок «сущность» можно перетащить на лист (Drag and Drop ). Двойной щелчок по значку открывает окно Свойства базы данных.



В категориях, в разделе Определение, может быть введено имя таблицы (физическое имя), например Клиенты. В разделе Категории / Столбцы можно ввести атрибуты, например, Номер клиента, а тип данных, например, целое число, можно выбрать из выпадающего списка.



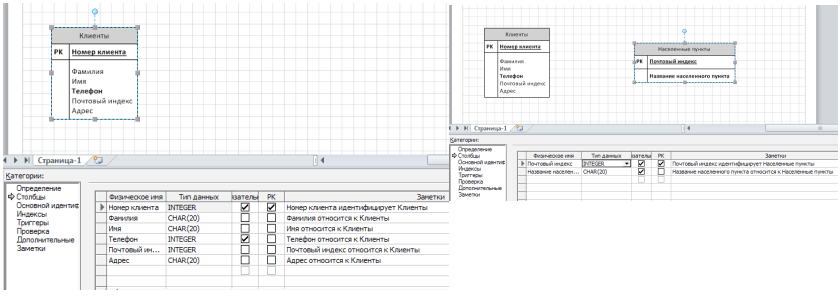

Поскольку номер клиента служит первичным ключом, включение флажка Обязательный столбец предотвращает появление пустых значений (соответствует NN Not Null). При установке флажка PK (Primary Key = первичный ключ) для столбцов отдельные строки в таблице базы данных получают уникальный идентификатор (соответствует UQ Unique). Результат – сокращение первичного ключа и подчеркивание имени.



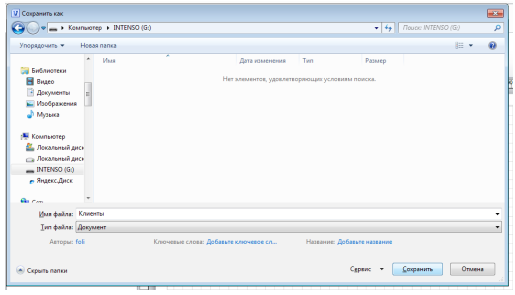

### 4.2.3 Добавление столбцов

В разделе Свойства базы данных/Столбцы можно ввести дополнительные атрибуты и тип данных для них. Установка флажка Обязательно позволяет избежать ввода нулевых значений, например – для даты рождения.

Как описано выше, создается таблица Клиенты и таблица Н/пункты с дополнительными атрибутами.



При выборе команды `Файл / Сохранить как...` чертеж Visio сохраняется под новым именем файла, например, `Клиенты.vsd`.

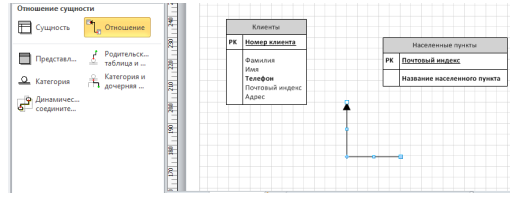



### 4.2.4 Добавление связей

В связях используются первичные и внешние ключи, чтобы базы данных могли связать строку в таблице (например, Н/пункт) со строкой в связанной таблице (например, `клиент.plz`). Эти связи могут отображаться на диаграмме. Кроме того, можно указать мощность связи (например, 1:n) и отобразить ее, используя нотацию Мартина, реляционную нотацию или нотацию IDEF1X.

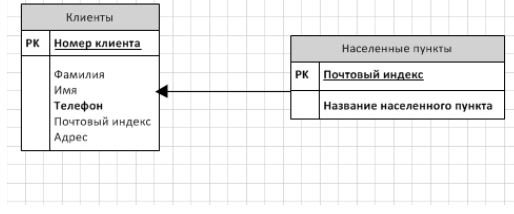
**Примечание:**
Отношения M:N (множества ко множеству) не могут быть отображены в шаблоне диаграммы модели базы данных с использованием нотации Мартина, реляционной нотации или нотации IDEF1X.

Из шаблона связь – сущность значок связи перетаскивается на лист и помещается на свободное место.



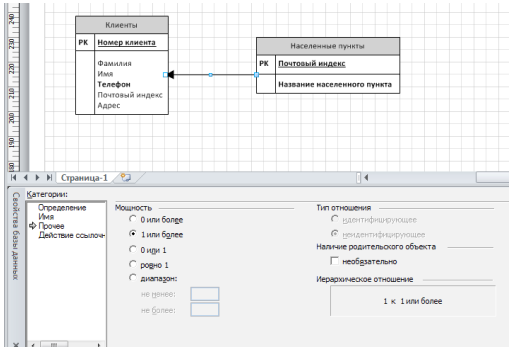

Конец со стрелкой связан с дочерней таблицей (подчиненной таблицей), например, Клиенты, а другой конец – с основной таблицей (родительской таблицей), например, Н/пункты.

Если дочерняя таблица еще не содержит столбец с тем же именем, что и первичный ключ, Visio добавляет внешний ключ в дочернюю таблицу.



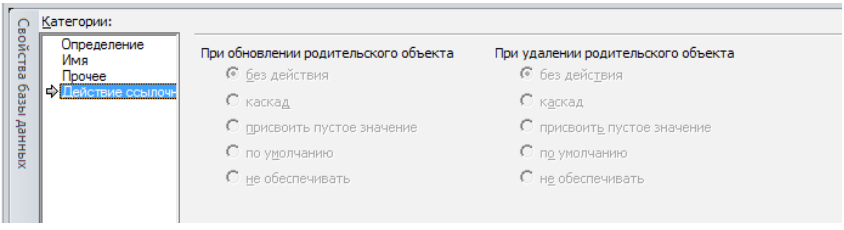

**Примечание**
Если линии отношений больше не отображаются, на вкладке База данных в группе Управление выберите Параметры отображения. На вкладке Связь в разделе Показать установите флажок Связи.

**Установка мощности связи:**
Двойной щелчок по строке отношения открывает окно свойств базы данных. В окне «Свойства базы данных» выберите Категории: разное. В параметре Мощность связи можно выбрать один или множество.




**Примечание**
Для связей 1:n подходит значение ноль, или множество либо один или множество.
Для связей 1:1 подходит значение ноль или один, либо один.

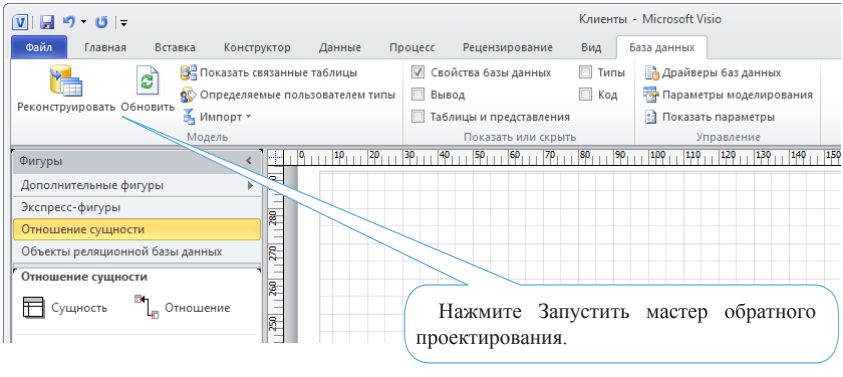
Ссылочное действие определяет способ обработки родительского элемента при его удалении или обновлении, например Установить ноль (SET NULL), или действие не требуется (NO ACTION).



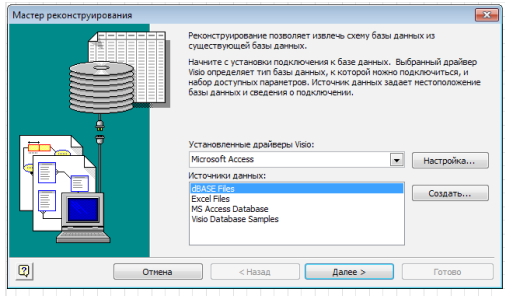

### 4.2.5 Реверсивный инжиниринг

Существующую базу данных можно импортировать с помощью Ассистента обратного проектирования (`Reverse Engineering`) Visio и представить как модель типа «сущность-связь».

На вкладке База данных нажмите на значок Обратное проектирование (`Reverse Engineering`).

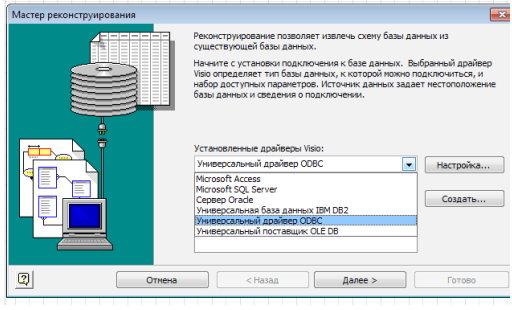


В разделе Источники данных выбирается соответствующий источник и устанавливается драйвер базы данных для системы управления базами данных (СУБД). Если драйвер базы данных Visio еще не был связан с конкретным источником данных ODBC, это можно сделать в разделе Настройка.

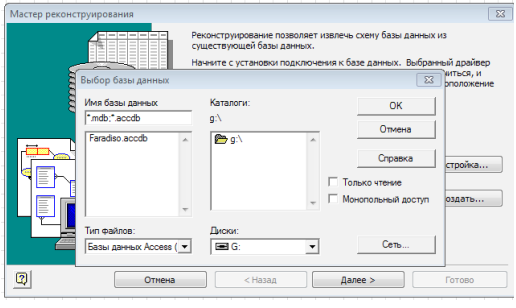



Если требуется выполнить обратное проектирование для листа Excel, то следует выбрать универсальный ODBC-драйвер.

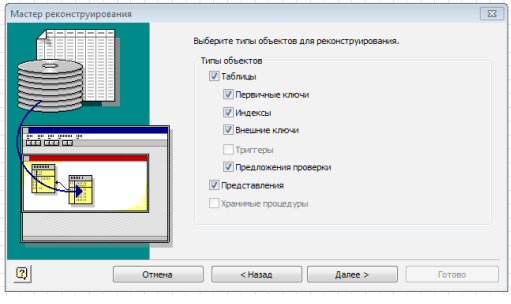
Определение соответствующего драйвера Visio гарантирует, что мастер правильно сопоставит собственные типы данных и что весь код, извлеченный мастером, будет правильно отображаться в окне кода.



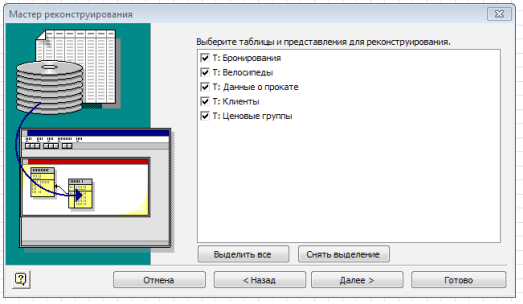

В следующем окне выберите источник данных базы данных (например, Faradiso.accdb), который должен быть импортирован в Visio. Если для существующей базы данных еще не создан источник данных, это можно сделать, нажав кнопку Создать.



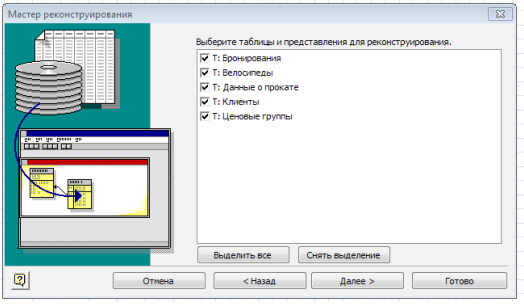

Теперь мастер запрашивает типы объектов, которые должны быть выбраны с помощью обратного проектирования.



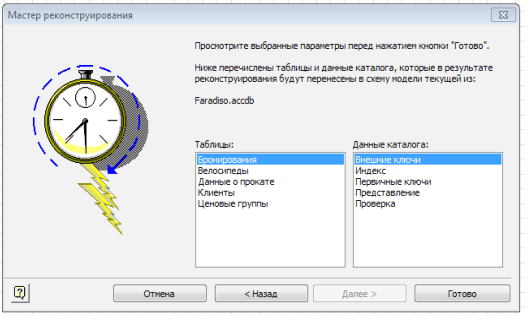
После выбора типов объектов отображаются таблицы. Можно выбрать некоторые объекты, установив флажки по отдельности, либо все вместе с помощью опции Выбрать все.



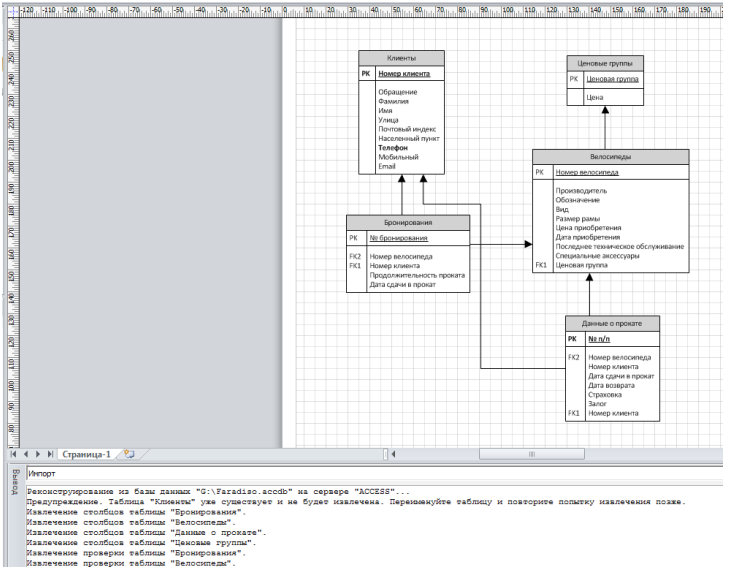

После нажатия кнопки Готово необходимо принять решение о том, следует ли автоматически отображать лист чертежа, или же данная возможность будет добавлена позже.




Сводная информация о выбранных настройках отображается в окне завершения работы мастера.




Мастер импортирует выбранную информацию и отображает информацию об импорте в окне вывода ниже. Шаг за шагом можно проверить все таблицы и связи.

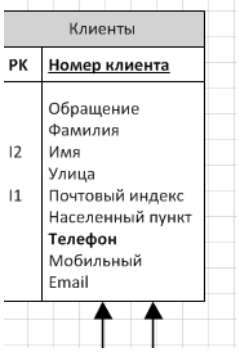


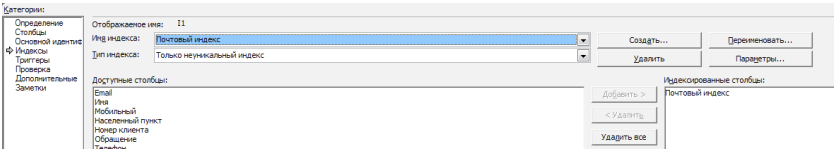

### 4.2.6 Создание индексов

Установка индексов повышает производительность или скорость работы базы данных при выполнении запроса. Индекс определяется как путь доступа, СУБД обычно хранит таблицу, отсортированную по индексу, для ускорения запроса, посредством индексированного значения, например – почтового индекса. Ключевые значения индексируются всегда.

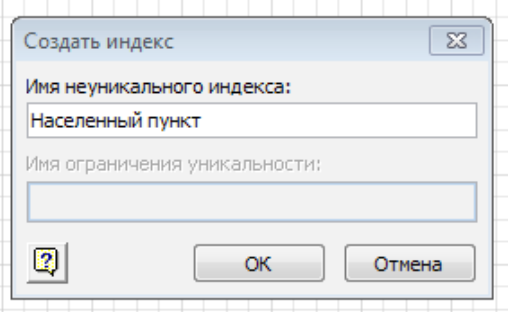
Ниже приведены индексы, созданные в таблице Клиенты базы данных faradiso.

Уже созданные индексы можно узнать по буквам I1 и I2 на ER-диаграмме. В данном примере индексируются почтовый индекс и фамилия. Если щелкнуть таблицу Клиенты на рабочем листе и выбрать Индексы в области Категории, отобразятся назначенные индексы, которые также могут быть отредактированы.

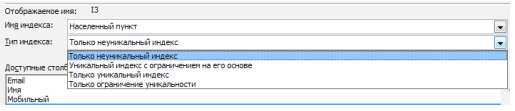


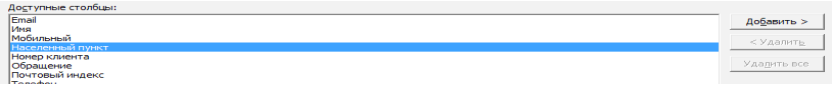



При нажатии кнопки Создать создается новый индекс. В диалоговом окне Создать Индекс вводится имя для индекса, например Н/пункт. При нажатии кнопки OK, диалоговое окно Создать индекс закрывается.




В списке Тип индекса выберите параметр, чтобы создать уникальный или однозначный индекс. Доступны 4 варианта.



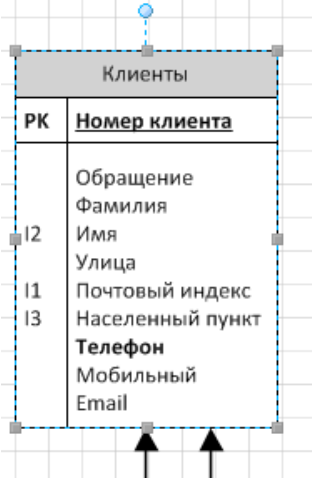





В списке Доступные столбцы выберите имена столбцов, которые будут включены в новый индекс. Нажмите кнопку Добавить, чтобы выбрать соответствующий столбец.

В списке Индексированные столбцы установите флажок Вкл, чтобы создать индекс в порядке возрастания сортировки. При отключении флажка создается индекс в порядке сортировки по убыванию.

На диаграмме модели базы данных новый индекс I3 затем отображается в атрибуте Н/пункт.

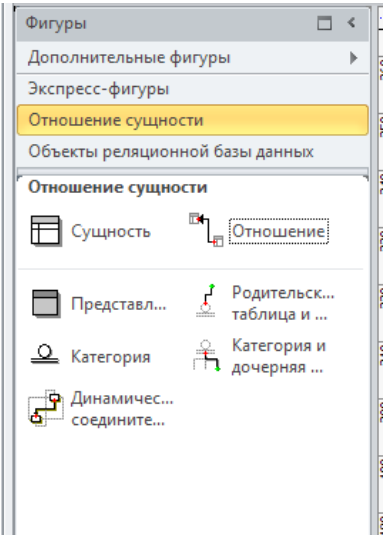


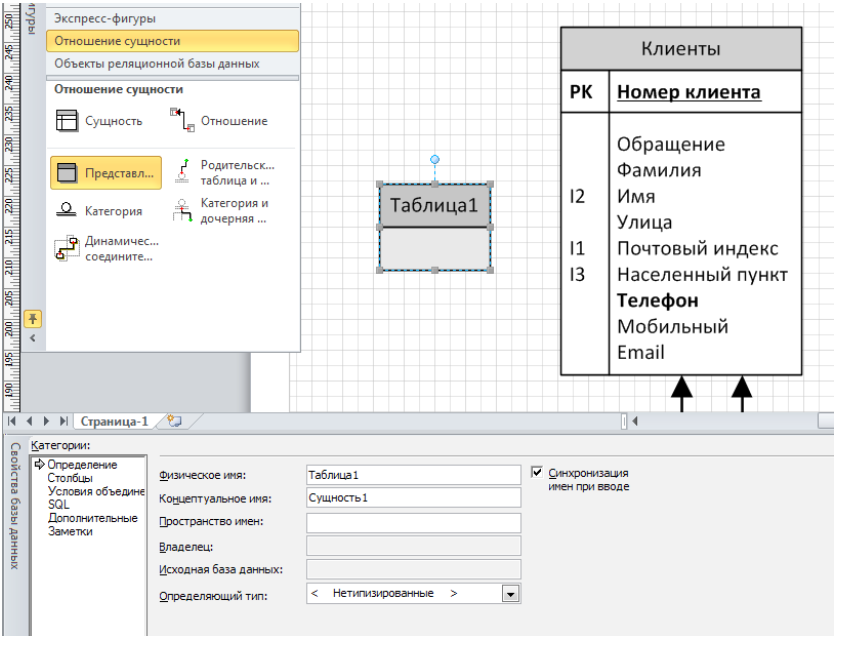
### 4.2.7 Создание представлений (Views)

Представление (View) представляет собой сохраненный запрос. Представления особенно полезны при повторном запросе одной и той же информации из нескольких таблиц. Представления могут использоваться для отображения разных пользователей базы данных (например, данных для клиентов) без возможности фактического изменения таблиц.

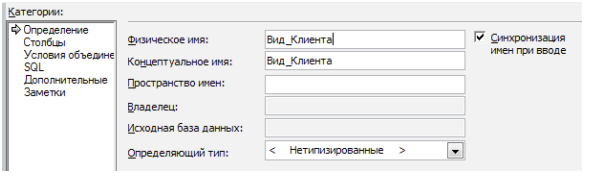
Для создания представления, из шаблона Сущность-связь или из шаблона Объект

реляционный перетащите форму представления на страницу чертежа.

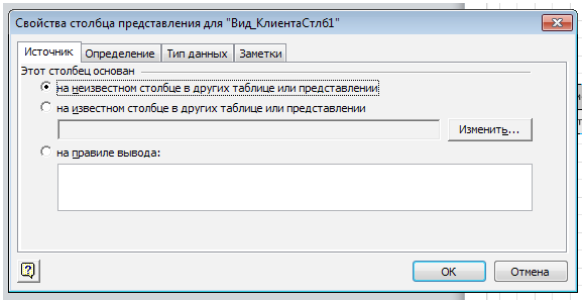


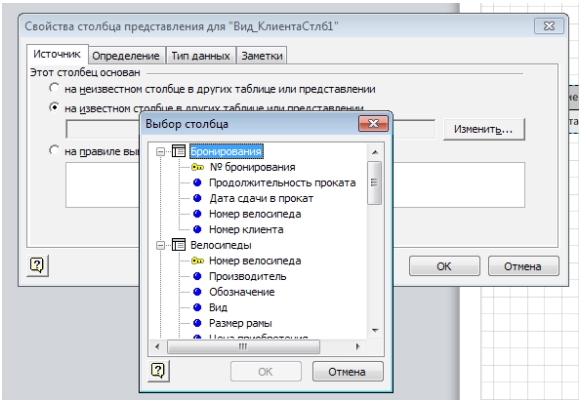
Свойства View_ Kunden, то есть представление таблицы Клиенты, теперь перечислены в Категориях. В пункте меню Определение устанавливается имя, например View_ Kunden.



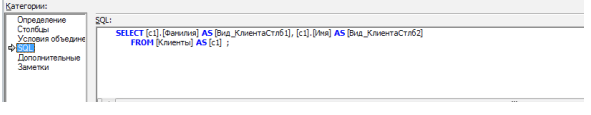

В разделе столбцы вставляется новый столбец. После добавления свойства столбца нового столбца устанавливаются в разделе Редактирование.




Источник можно открыть, выбрав Известный столбец в другой таблице, или в окне выбора Просмотр и изменение Выбрать столбец. Здесь можно выбрать, например, параметр Фамилия. При двойном подтверждении с помощью OK столбец переносится в представление. Таким образом можно добавить столбец за столбцом.



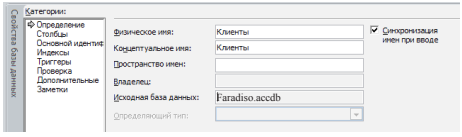

В Категории SQL связи между клиентами View_ Kunden снова отображаются в виде SQL-кода.



### 4.2.8 Создание условий проверки полей

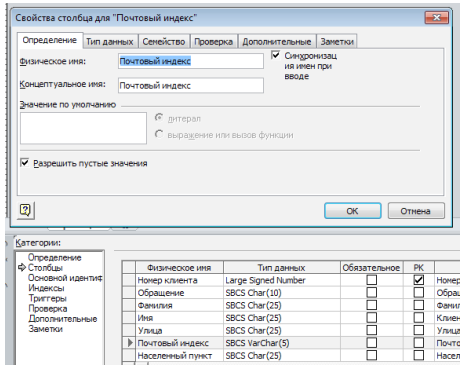
Условия проверки полей гарантируют, что данные, введенные в столбец, находятся в определенном диапазоне значений. Например, можно установить, что почтовый индекс имеет длину в 5 цифр.

Двойной щелчок по таблице (например, Клиенты) открывает окно свойств базы данных.

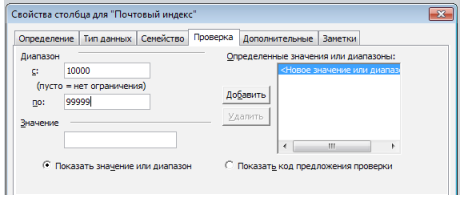



Затем в Категории Столбцы нажмите Столбец Индекс.

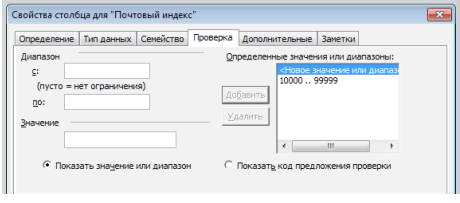
Затем нажмите Изменить, и откроется окно Свойства столбца для Почтового индекса. Требуемое ограничение вводится на вкладке Проверка.



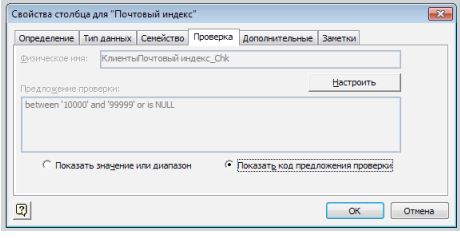

Почтовый индекс должен находиться в диапазоне от 10000 до 99999.




Область добавляется после ввода значений и подтверждения нажатием кнопки Добавить.

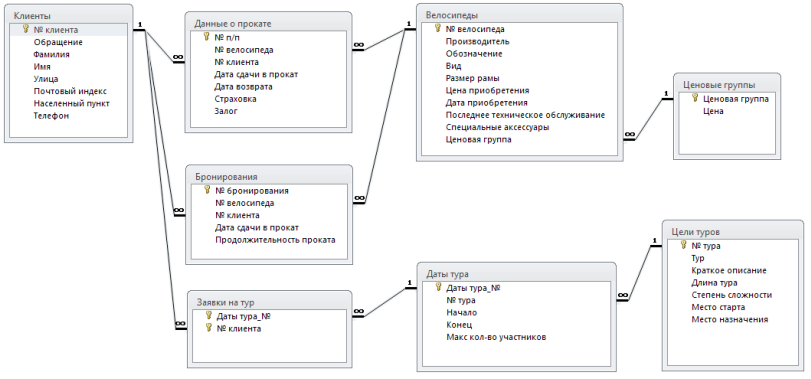



Проверка результатов может быть выполнена с помощью Кода для условия проверки полей. Здесь было добавлено от 10000 до 99999 или NULL.



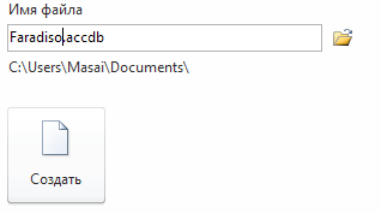
# 5 Разработка базы данных в среде Access

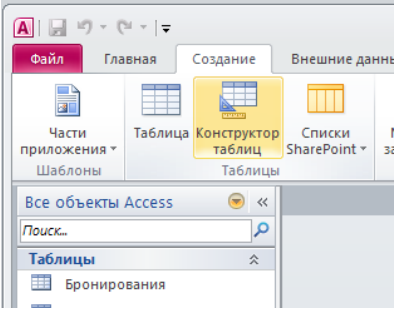
С помощью системы управления базами данных Access постепенно создается база данных для управления прокатом велосипедов `Faradiso`. При этом регистрируются операции проката и бронирования. Также можно управлять регистрациями запланированных поездок на велосипеде. Таким образом, расширенная ERM-модель базы данных должна получить следующую структуру:




## 5.1 Создание таблиц

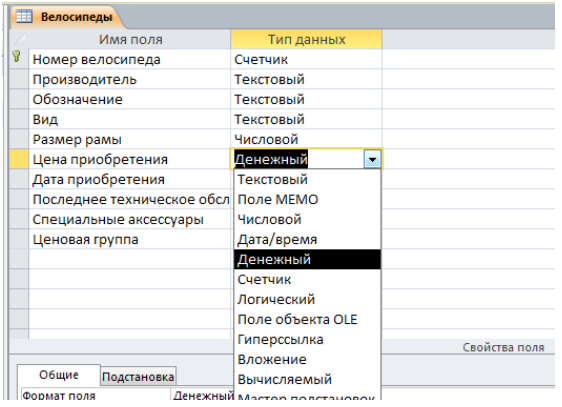
При запуске Access открывается окно с запросом, с какой базой данных следут работать. Для создания новой базы данных выберите пункт Пустая база данных. После ввода нужного места хранения и имени Faradiso новая база данных создается нажатием кнопки Создать.




Данные хранятся в таблицах в реляционных базах данных. Перед вводом данных для этих таблиц задаются параметры их столбцов и типов данных.

После выбора вкладки Создать и нажатия кнопки Макет таблицы откроется окно конструктора новой таблицы.




Чтобы определить поля таблицы, введите имена полей в столбце Имя поля по порядку и задайте тип данных в столбце Тип данных поля. В правом столбце можно указать описание поля.

**Установка типов данных**

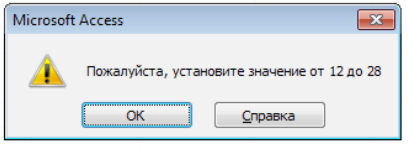
Для каждого столбца таблицы необходимо задать свой тип данных.

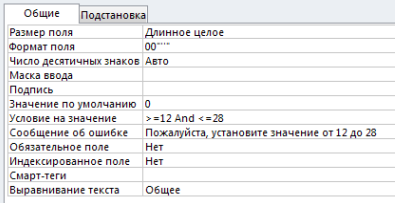
Например, для поля Номер велосипеда может быть выбран тип данных Автоматически. Номер записи автоматически увеличивается на один при вводе записей. Для полей Производитель, Марка и Тип выбирается тип данных Текст и устанавливается размер поля, например 20 символов. Для содержимого поля Размер рамы тип данных – Число, а размер поля – Байт (целое число от 0 до 255). В поле Цена покупки в качестве типа данных задается валюта, а в качестве единицы валюты – евро. Полям Дата покупки и Последнее техобслуживание назначаются типы данных Дата/время. Для поля данных Специальные аксессуары устанавливается тип данных Memo для возможности записи более длинных описаний.

**Правила проверки**

Чтобы избежать неправильного ввода в числовом поле, можно задать правила. Если введенное значение не соответствует заданному правилу, на экране появится окно с текстом сообщения.

Например, для поля Размер рамки задается область от 12 до 28 дюймов. При попытке ввести значение 36 отображается указанное сообщение.




Чтобы дополнить числовое значение в поле Размер рамы символом «дюймы», необходимо определить формат. После двух заполнителей для чисел 00 отображаемые символы (здесь символы″) заключаются в кавычки.

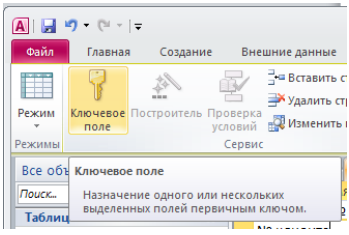
При вводе стандартного значения, например 0, программа автоматически предлагает поставить значения при вводе новой записи данных. Это облегчает ввод для часто используемых значений.

**Первичный ключ**

Чтобы однозначно идентифицировать отдельные велосипеды, в качестве поля первичного ключа устанавливается поле Номер велосипеда, содержание которого отличается в каждой записи. После нажатия на строку Номер велосипеда на панели инструментов будет нажата кнопка Первичный ключ.

Перед именем поля появится значок ключа.

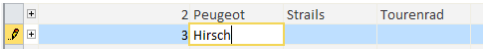
Первичный ключ, состоящий из нескольких полей, определяется путем выделения соответствующих строк при нажатой клавише управления и нажатия кнопки Первичный ключ.




Для сохранения таблицы нажмите на значок диска на панели инструментов. В качестве имени, например, вводится значение Велосипеды и подтверждается нажатием на Сохранить.

Чтобы ввести данные, откройте таблицу, дважды щелкнув имя таблицы или используя кнопку Просмотр – Просмотр таблицы. При вводе данных, в соответствующей строке появляется символ пера.

При выходе из записи перо исчезает, и запись автоматически сохраняется.



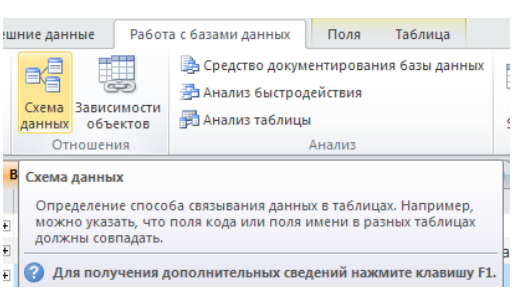

**Примечания:**

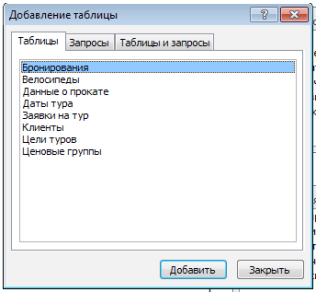
Данные сразу же сохраняются на жестком диске после выхода из записи, чтобы избежать потери данных.

Функция отмены ввода данных, как в текстовых редакторах, отсутствует.

## 5.2 Определение связей и ссылочной целостности

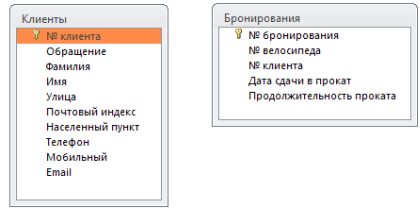
Связи определяют отношения полей различных таблиц. В среде Access связи создаются при помощи графических средств. Например, нужно указать, что номера клиентов в таблице Резервирование связаны с номерами клиентов таблицы Клиенты. Для этого в строке меню выберите вкладку Инструменты базы данных и нажмите кнопку Связи. Откроется окно Показать таблицу, чтобы выбрать таблицы.



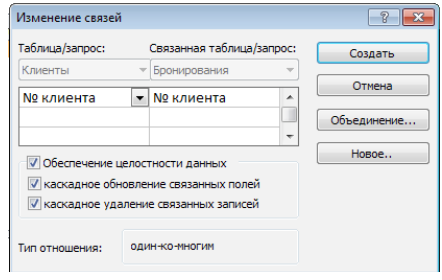




Выберите все отображаемые таблицы, удерживая клавишу Shift и кликнув по первой и последней таблице в списке, затем нажмите клавишу Добавить. Окно Показать таблицу будет закрыто.

Чтобы установить связь, кликните запись Номер клиента в таблице Клиенты и при нажатой кнопке мыши перетащите запись Номер клиента в таблицу Резервирование.




Откроется окно для редактирования свойств связей.



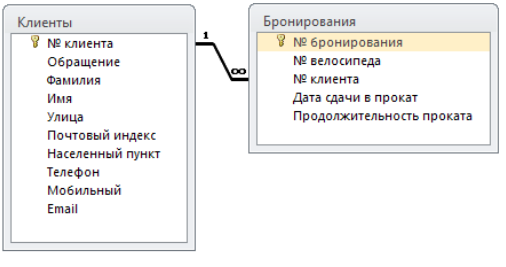

Флажок Учитывать ссылочную целостность заставляет систему базы данных проверять логические отношения между записями. Первичный ключ клиента, который еще не был добавлен в таблицу Клиенты, не может быть введен в таблицу Резервирование. И наоборот, запись данных клиента, на которую ссылаются записи данных другой таблицы, не может быть удалена без удаления связанных записей данных, например, в таблице Резервирование.

**Примечание:**
Ссылочная целостность гарантирует, что связи между таблицами являются логически правильными, а несвязанные массивы данных отсутствуют.

Чтобы иметь возможность удалять данные (например, бывших клиентов), установите флажок Перенос удаления на связанные записи данных. Если запись удаляется из таблицы, поле первичного ключа которой (например, номер клиента) ссылается на другую запись, все записи в других таблицах, которые содержат тот же номер клиента, также удаляются. Таким образом предотвращаются ситуации, когда при удалении первичных записей остаются забытие Записи, в результате чего могут возникнуть аномалии удаления.

Активация Переноса обновления в связанные поля означает, что при изменении ссылочного поля (например, номера клиента в таблице Клиенты), содержимое других полей таблицы со ссылкой на это поле также изменяется.

Если нажать кнопку Создать, связь 1:n между двумя таблицами отображается линией и символами 1 и ∞ в окне «Связи».

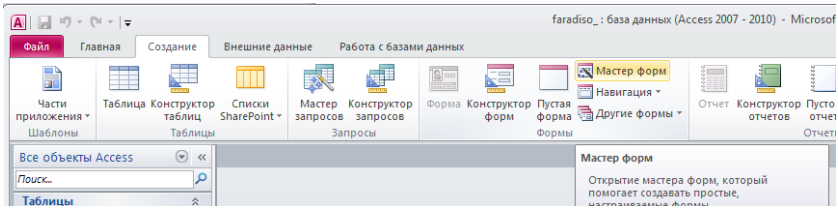


## 5.3 Формы

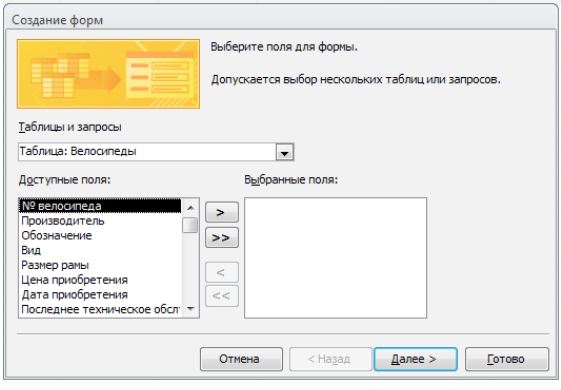
Для ввода данных предусмотрены формы. Они используются как для ввода данных, так и для просмотра, редактирования и навигации.

### 5.3.1 Создание формы

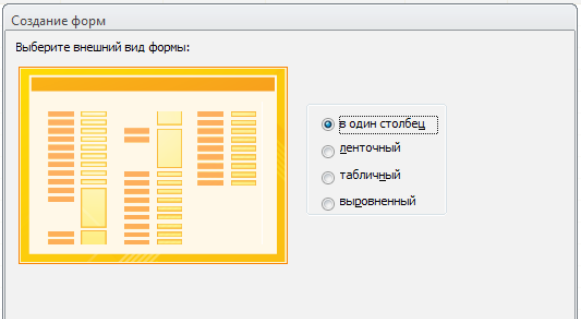
Чтобы создать новую форму, перейдите на вкладку Создать, в подпункте Формы выберите Мастер создания формы.



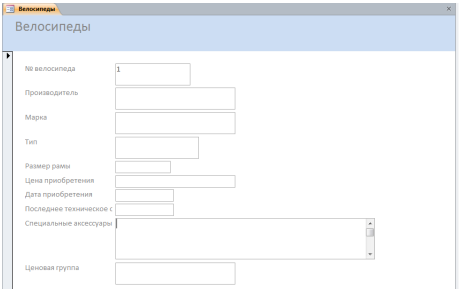

В следующем диалоговом окне задается внешний вид формы. На примере одной формы для таблицы Велосипеды:




После выбора таблицы Велосипеды кнопка >> переносит все поля таблицы для формы. После нажатия кнопки Далее > устанавливается макет формы, например, одностолбцовый.



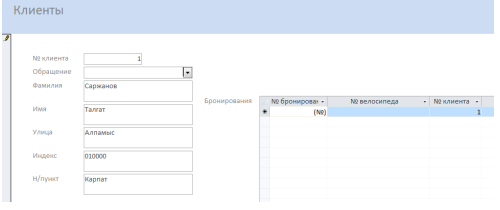
После нажатия кнопки Завершить процесс создания формы завершается. Он используется, например, для выбора или изменения данных.




### 5.3.2 Подчиненные формы

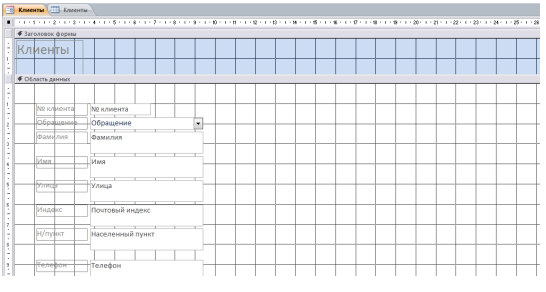
Подчиненные формы – это инструменты для отображения связей 1:n таблиц в формах. В то время как основная форма отображает запись, которая содержит поле первичного ключа отношения, подчиненная форма подробно отображает связи между записями.

Далее создается форма Клиенты с подчиненной формой, в которой также отображаются связанные резервирования для соответствующего отображаемого клиента.

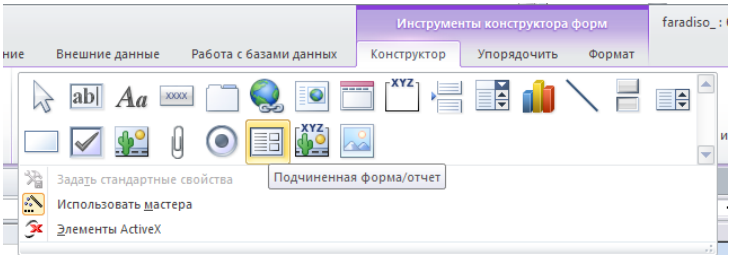



В качестве подчиненной формы сначала создается форма, которая может отображать записи резервирования. Для этого на вкладке Создать нажмите кнопку мастер создания форм. В качестве источника данных установлена таблица Резервирование, а при помощи кнопки >> добавляются все поля. Форма сохраняется под именем Подч_форма_Резервирование.

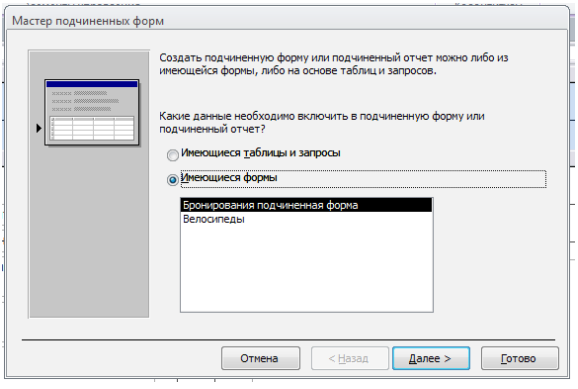
Теперь форма Клиенты с одним столбцом создается с помощью мастера форм. Чтобы найти достаточно места для подчиненной формы, область детализации формы должна быть соответственно увеличена. Это можно сделать, щелкнув по правому краю области детализации в режиме конструктора и перетащив ее вправо с помощью мыши.



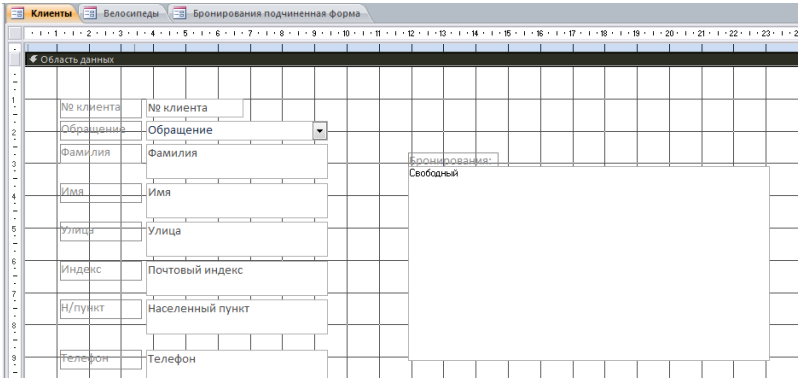
На вкладке Инструменты проектирования формы – вкладка Дизайн выбирается кнопка Подчиненная форма.




Теперь на свободном пространстве формы рисуется подходящий большой прямоугольник. Запускается мастер (конструктор), который позволяет выбрать подчиненную форму.




После сохранения формы информация о каждом клиенте будет отображаться с соответствующим Резервированием. Конструктор выглядит следующим образом:

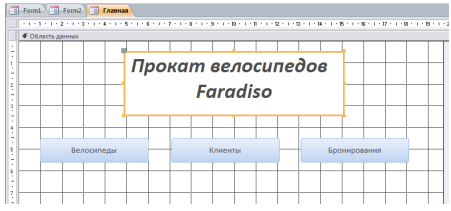


### 5.3.3 Управление базами данных с помощью кнопок

Используя кнопки на формах пользователь может переходить к различным рабочим областям приложения базы данных.

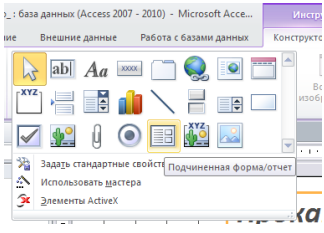
**Пример**

Создается форма Центральное отделение с полем для Проката велосипедов Faradiso и тремя кнопками управления Велосипеды, Клиенты и Резервирование. Готовая форма, например, выглядит так:

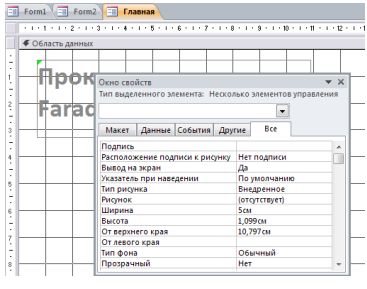



Имеется возможность управления обработкой различных таблиц с помощью командных кнопок.

Сначала создается новая форма, которая не связана с конкретной таблицей. Чтобы создать заголовок, щелкните значок поля метки Аа во вкладке «Элементы управления макетом», затем с помощью мыши перетащите границы вокруг области сведений формы и введите заголовок.




Щелчок правой кнопкой мыши на ярлыке открывает меню, в котором можно выбрать окно Свойства. Здесь можно определить, например, ширину и высоту поля и свойства шрифта.

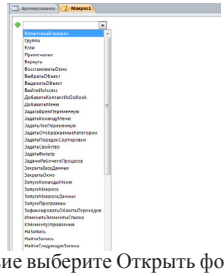



Кнопки команд встраиваются в форму путем клика на Командную кнопку на вкладке Инструменты макета формы – Дизайн и изменения размера кнопки в форме.

Такие командные кнопки, например, хранятся в макросах для выполнения определенных действий.

## 5.4 Макросы

Макросы создаются для того, чтобы с помощью нажатия на Велосипеды центральной формы открыть форму Велосипеды и одновременно закрыть центральную форму. Если щелкнуть вкладку Создать в окне База данных и нажать кнопку Макрос, то откроется окно для создания макросов.



Для этого в поле Действие выберите Открыть форму. Теперь в Области имеются аргументы, например, имя формы Велосипеды и представление формы.

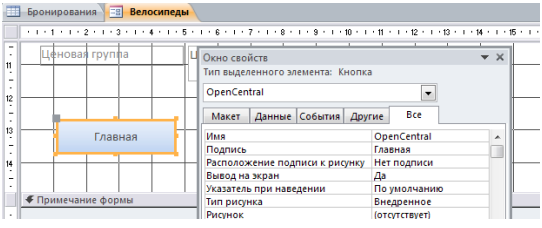
Центральная форма закрывается с помощью второго действия макроса, например, Закрыть и задаются аргументы для формы типа объекта Форма и имени объекта Центральная форма. Закройте окно макроса и сохраните его под значимым именем, например Доступные велосипеды.

**Примечание:**
Значимые имена объектов облегчают последующее администрирование базы данных.

Теперь макрос OpenWheels связан с соответствующей кнопкой в центральной форме. Для этого откройте центральную форму в режиме конструктора и откройте страницу свойств, нажав на командную кнопку Велосипеды. На вкладке Событие для события Открыть при нажатии задан макрос Открыть велосипеды. Если теперь нажать на эту командную кнопку в центральной форме, то макрос и, следовательно, записанные в нем действия будут выполнены.

Чтобы вернуться в Центральную форму, необходимо дополнить форму Велосипеды с помощью командной кнопки и макроса.

Для этого создайте макрос OpenCentral таким же образом, как и макрос Открыть велосипеды. Первое действие, которое будет выполнено – Закрыть без каких-либо дополнительных аргументов для закрытия текущего объекта. Второе действие Открыть форму устанавливается с аргументами действия центральная и Форма.




Командная кнопка расположена в конструкторе формы Велосипеды и снабжена соответствующими свойствами, вызываемыми двойным кликом. Вместо заголовка в качестве свойства можно задать изображение (например, информативное растровое изображение). В свойстве При нажатии устанавливается макрос Открыть центральную форму.

## 5.5 Создание отчета

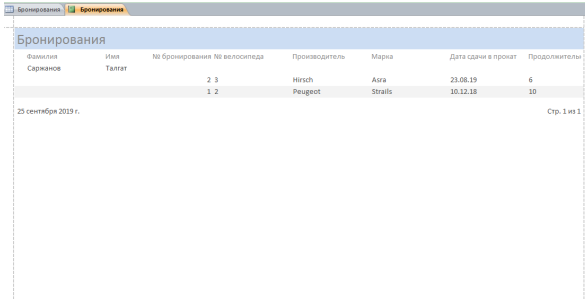
Отчеты предназначены для обработки данных для вывода на печать. Например, создаются и распечатываются каталоги или счета. В одном отчете можно легко реализовать обширную группировку данных с промежуточными суммами и другими расчетами.

Отчеты можно создавать с помощью Мастера отчетов.

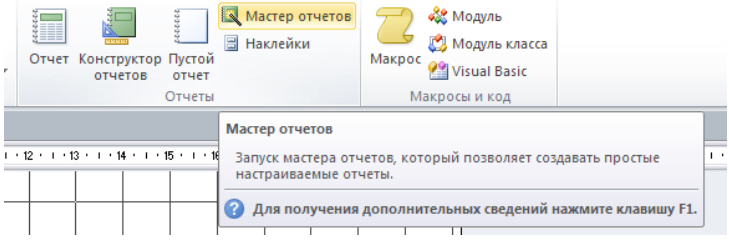
**Пример**

Создается отчет о Резервированиях каждого клиента, хранящихся в базе данных. Должны выводиться фамилия и имя, номер бронирования, информация о велосипеде, дата и планируемый срок проката. После каждого клиента должна отображаться сумма за все дни периода аренды.

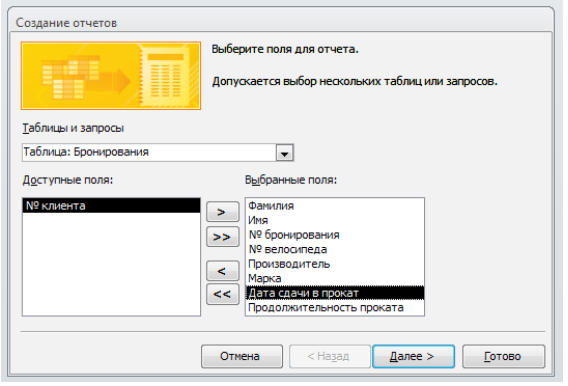
Отчет может выглядеть следующим образом:



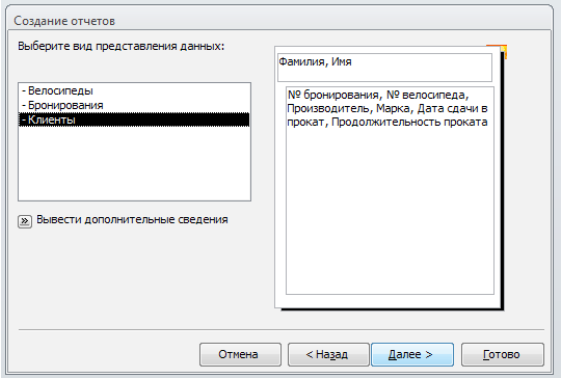

После выбора вкладки Создать нажмите кнопку Мастер отчетов в области Отчеты на ленте.



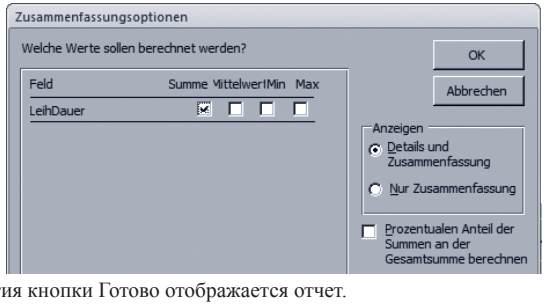

В следующем окне можно определить поля, которые будут включены в отчет. Чтобы принять фамилию и имя клиента, в окне Таблицы / Запросы выберите опцию Таблица: Клиенты и нужные поля, нажав соответствующую кнопку > переместить в правую половину изображения.




Из таблицы Резервирование будут взяты поля № рез., № велосипеда, Дата сдачи в прокат и Срок проката, а поля Производитель и Обозначение будут взяты из таблицы Велосипеды. После нажатия кнопки Далее производится группировка записей данных. Чтобы просмотреть все записи, относящиеся к одному клиенту, сгруппируйте записи по клиентам, и нажмите Далее > для подтверждения.




Чтобы просмотреть стоимость всего срока аренды для каждого клиента, выберите в следующем окне Параметры сводки, нажмите на флажок Сумма и подтвердите выбор нажатием на OK.




После нажатия кнопки Готово отображается отчет.

## 5.6 Создание запросов к базе данных

Запросы используются для оценки и обработки данных. Так, например, можно произвести поиск в таблицах по определенным критериям.

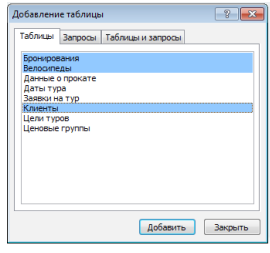
**Пример**

Необходимо разработать запрос Прокат_2019, который покажет всех клиентов 2019 года с данными аренды и данными велосипедов, сдававшихся в прокат.

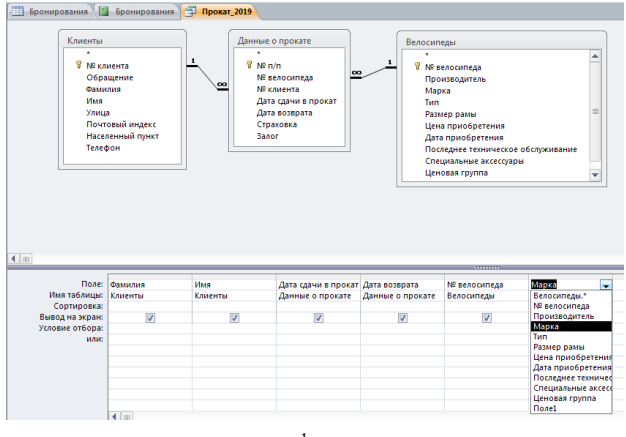
Прокат_2019

| Фамилия | Имя | Дата сдачи | Дата возврата | № велосипеда | Марка |
|---|---|---|---|---|---|
| Арканов | Михаил | 15.06.18 | 18.02.19 | 2 | Strails |
| Саржанов | Талгат | 18.10.18 | 19.12.18 | 1 | A25 |

После выбора вкладки Создать в разделе Запросы на ленте меню среды Access выбирается кнопка Конструктор запросов. Таблицы для редактирования доступны для выбора в окне Показать таблицу.



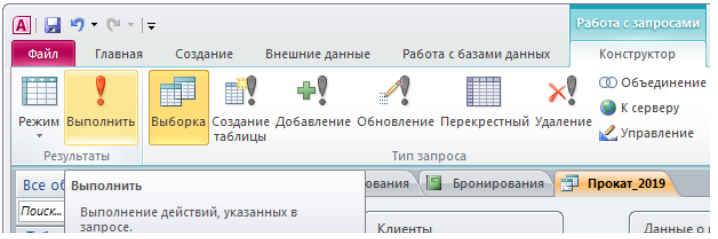

При нажатой клавише управления выбираются таблицы Клиенты, Данные проката и Велосипеды. После подтверждения с помощью Добавить открывается представление конструктора запроса.




Желаемые поля, такие как, например, фамилия, имя, дата сдачи в прокат выбираются в отдельных столбцах с помощью выпадающих меню или двойным щелчком мыши.

Чтобы ограничить запрос сдачей в аренду в 2019 году, в столбце Дата сдачи в прокат, в качестве критерия вводится значение с #01.01.2019 по #31.12.2019. В Access значения даты устанавливаются между символами #.

После сохранения запроса (например, под названием прокат_2019) можно использовать кнопку ! Запуск выполняется.




**Примечание:**

Сохраненные запросы содержат не результат запроса, а SQL-инструкцию для выполнения запроса.

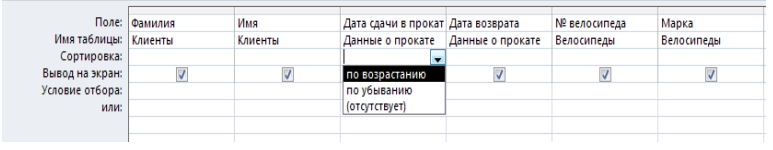
Сортировка вывода:

Для сортировки записей по столбцу в режиме конструктора запроса в соответствующей строке задается порядок сортировки по возрастанию или по убыванию.

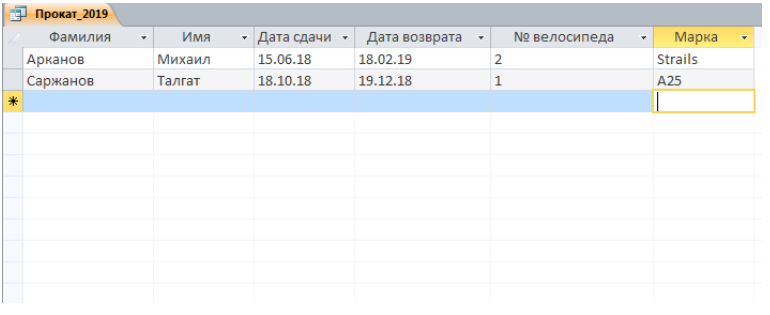
**Пример**

Запрос Прокат_2019 должен быть отсортирован по возрастанию по дате сдачи в прокат.

Для этого нажмите на выпадающее меню поля Сортировка в столбце Дата сдачи в прокат и выберите По возрастанию.




Затем строки базы данных отображаются в порядке возрастания даты проката после выполнения запроса:



## 5.7 Упражнения к Главе 5

**Задача 1**

**Управление проектами**

В одной из компаний необходимо с помощью базы данных создать ограниченную систему управления проектами с возможностью расширения функций в будущем.

Должны соблюдаться следующие условия:

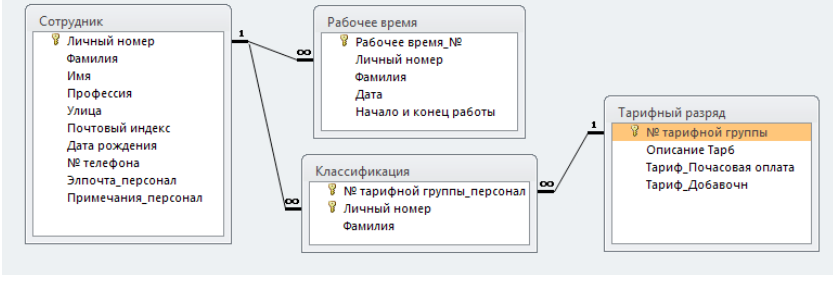
- Доступ Каждого сотрудника в систему должен осуществляться с помощью пароля. Пароль хранится в таблице сотрудников.
- Для каждого проекта необходимо сохранить имя, описание, дату начала и окончания, а также руководителя проекта.
- Для каждого проекта в качестве руководителя проекта выбирается именно один сотрудник.
- Предполагается, что для каждого сотрудника будет записана продолжительность рабочего времени (продолжительность в часах) и описание выполненной работы, которую он выполнял в определенный день для конкретного проекта.

a) Составьте графическое представление связей таблиц посредством ER-диаграммы и укажите тип связей между таблицами.
Создайте таблицы с необходимыми атрибутами в 3-й нормальной форме. Создайте первичные ключи для каждой таблицы и связей.
Связи должны быть проверены на ссылочную целостность при вводе данных. Кроме того, предполагается, что будет возможно расширение удаления и обновление подробных записей.
Для чего требуется ссылочная целостность?

b) Создайте форму для ввода данных проекта.

**Задача 2**

Для расчета заработной платы персонала в компании используется база данных с приведенной ниже расширенной ER-диаграммой,




a) Реализуйте диаграмму модели типа «сущность – связь» в Access.

b) В каких моментах этот проект нарушает правила нормализации первой, второй и третьей нормальной формы? Предложите, как можно избежать нарушений.

c) Почему разумно и важно соблюдать правила нормализации?

# 6 Язык базы данных SQL

SQL (Structured Query Language = структурированный язык запросов) – это язык базы данных для определения структур данных в реляционных базах данных, а также для редактирования (вставки, изменения, удаления) и запросов к хранилищам данных.

## 6.1 Стандарты SQL

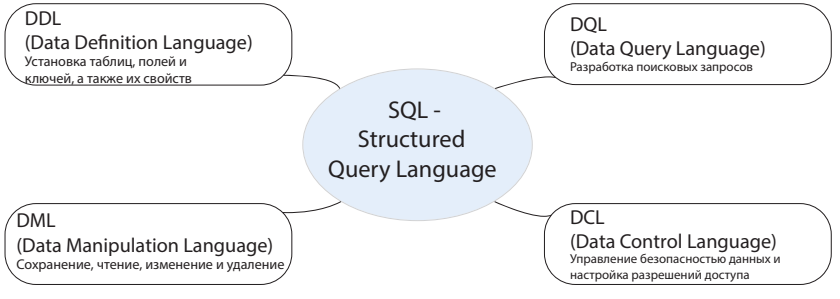
Сначала SQL был языком для конечного пользователя систем реляционных баз данных. Здесь его заменили широко распространенные графические интерфейсы. Но поскольку SQL более мощный, чем многие графические системы для доступа к данным, он является важным программным обеспечением для разработчиков баз данных. Он также имеет большое значение как язык интерфейса и доступа к другим системам баз данных, поскольку он практически не зависит от операционной системы, режима работы и интерфейса.

Различают различные стандарты:

| Год выпуска | Норма | Стандарт |
|---|---|---|
| 1986 г. | ANSI-SQL1 | Стандарт SQL-1 |
| 1989 | ISO SQL-1 | SQL-1 с расширениями целостности |
| 1992 г. | ISO SQL-92 | Стандарт SQL-2, три этапа: |
| – Entry-Level<br>– Intermediate-Level<br>– Full-Level | | |
| 1999 г. | ISO / IEC 9075: 1999 | Стандарт SQL-3 |
| 2003 г. | ISO/IEC 9075:2003 | Связь с XML (расширяемый язык разметки) |
| 2007 г. | ISO/IEC 9075:2007 | Расширение типа данных XML |
| 2011 | ISO/IEC 9075:2011 | Текущая Версия |

В настоящее время в системах баз данных реализован, прежде всего, стандарт SQL-2, в то время как новые стандарты реализуются только в отдельных системах.

Язык SQL состоит из относительно небольшого количества команд, но их можно использовать различными способами. Помимо прочего, он служит как для определения таблицы и поля, так и для запроса и изменения данных.



## 6.2 Создание, изменение и удаление таблиц

**Создание таблиц**

Команда `CREATE TABLE` имя таблицы создает таблицу базы данных. Общий синтаксис:

```
CREATE TABLE имя таблицы (
Spalte_1 Datentyp_für_Spalte_1,
Spalte_2 Datentyp_für_Spalte_2,
...);
```

**Пример**

Создать новую таблицу `Клиенты`.

Решение:

```
CREATE TABLE Клиенты
№ клиента INT NOT NULL auto_increment,
фамилия VARCHAR(45) NOT NULL
имя VARCHAR(45) NULL ,
улица VARCHAR(45) NULL ,
Почтовый индекс VARCHAR (6) NULL,
телефон VARCHAR(25) NULL ,
);
```

**Примечание:**

Командой CREATE таке ожно создать новую базу данных, напр. команда

CREATE Database faradiso_neu;

создаст базу данных с именем faradiso_neu.

**Изменение таблицы**

Команда `ALTER TABLE` имя таблицы изменяет свойства или столбцы таблицы.

Общий синтаксис:

```
ALTER TABLE Имя_таблицы
[ Спецификация Изменений]
```

Изменение спецификаций зависит от характера требуемых Изменений, например:

- Добавить столбец: `ADD столбец_1 тип данных_для_столбец_1`
- Удалить столбец: `DROP столбец_1`
- Изменить имя столбца: `CHANGE старое_имя столбца новое имя столбца тип данных_для_нового_имени столбца`
- Изменение типа данных столбца: `MODIFY Столбец_1 новый тип_данных`

**Пример**

В таблице Клиенты добавляется столбец Email.

Решение:

```
ALTER TABLE Клиенты
ADD Email VARCHAR(30);
```

Удаление таблицы

Команда `DROP TABLE` имя таблицы удаляет таблицу.

**Пример**

Необходимо удалить таблицу `Велосипеды_стар`.

Решение:

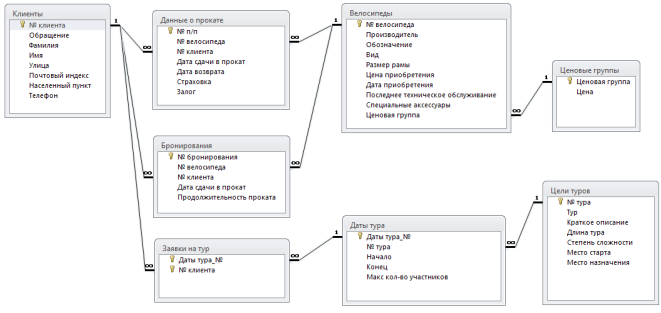
```
DROP TABLE Велосипеды_стар;
```

**Примечание:**
Удаленная таблица больше не может быть восстановлена с помощью команды undo из-за условий целостности данных.

## 6.3 Запросы выбора с помощью SELECT

Для запросов по выбору имеется команда SELECT. Эта команда обладает обширным синтаксисом, что делает ее универсальной.

**Примечания:**
Следующие примеры относятся к базе данных «Faradiso», которая отображает прокат велосипедов.




В дополнение к основным данным клиентов и велосипедов, он включает в себя как данные аренды, так и данные бронирования.
Кроме того, велосипедные туры предлагаются по различным направлениям, на которые клиенты могут зарегистрироваться.

### 6.3.1 Ограничение запросов на выбор с условиями

**Пример**
Создать SQL-запрос для выбора всех велосипедов с группой цен менее 5.
Решение:

```
SELECT номер велосипеда, производитель, название, ценовая
группа FROM Велосипеды
WHERE ценовая группа < 5;
```

После ключевого слова `SELECT` указываются столбцы для отображения, разделенные запятыми, например, номер велосипеда, производитель, название, ценовая группа. Вывод выполняется в порядке после оператора SELECT. Обратите внимание на то же написание имен полей, что и в таблицах, иначе они не будут распознаны. Также некоторые специальные символы, такие как дефис «-» интерпретируются как математический оператор «минус».

**Примечание:**
Специальные символы и пробелы вызывают ошибки выполнения в именах таблиц и полей.

Символ * может использоваться как заполнитель для всех полей таблицы, например, показывает инструкцию

```
SELECT * FROM Велосипеды;
```

все записи в таблице Велосипеды.

Таблица, с которой необходимо работать, например, Велосипеды, указывается после ключевого слова FROM (= с, по). Если используется несколько таблиц, то они разделяются запятыми. После зарезервированного слова `WHERE` (= где) возникает условие, ограничивающее выбор отображаемых записей. Обычно он состоит из имени столбца, за которым следует оператор сравнения и значение сравнения.

| Оператор | Пример | Значение, эффект |
|---|---|---|
| = | № клиента = 24 | тот же |
| < | Ценовая группа < 5 | меньше чем |
| > | | больше, чем |
| <> | | не равно |
| <= | | меньше или равно |
| >= | | больше или равно |
| LIKE | Н/пункт LIKE 'U%' | Сравнивает поле Местоположение с текстовым шаблоном; учитываются все населенные пункты, начинающиеся с U. |
| И | ПОЧТОВЫЙ ИНДЕКС = 89077 Н/пункт AND | Строка базы данных учитывается только в том случае, если выполнено условие 1 и условие 2 |
| | № > 200<br>OR<br>№ < 100 | Запись учитывается, если выполнены условие 1 или условие 2. |
| NOT | NOT (ПОЧТОВЫЙ ИНДЕКС = 89077) | Запись учитывается, если условие не выполнено |

**Примечание:**
условия «WHERE» должны всегда состоять из логического сравнения, на которое можно ответить «`true`» или «`false`».

В примере используется условие Группа цен < 5. Для каждой записи в таблице Велосипеды, система управления базой данных проверяет, выполняется ли для нее условие «истина» или «ложь». В результате запроса на экране отображаются записи, для которых ценовая группа <5 является истиной.

### 6.3.2 DISTINCT

Ключевое слово `DISTINCT` (= разные) скрывает дубликаты записей в запросе. Если, например, необходимо отобразить только названия производителей, у которых были приобретены велосипеды, то при выборе

```
SELECT DISTINCT FROM
```

Велосипеды, каждый соответствующий производитель отображается только один раз.

### 6.3.3 Представление содержимого поля в условии WHERE

Содержимое числовых полей, включая поля валюты, используется без дополнительной маркировки, например, `Цена = 150.`

Содержимое текстовых полей заключено в кавычки внутри условия WHERE, например, `Место жительства = 'Штутгарт'.`

Текстовые поля можно искать по текстовым шаблонам. В качестве плейс-холдеров в Jet-SQL (например, Access), используется знак вопроса ? для обозначения одного любого символа или * – для одного или нескольких любых символов.

| Пример | Результат |
|---|---|
| 'M??er' | 'Meier', 'Maier', 'Mayer', 'Meyer'; заменяются ровно два символа. |
| 'M*er' | Maier, Mittersberger, Müller ... ; все поля, содержимое которых начинается с 'M' и заканчивается на 'er'. |

**Пример**

Оператор SELECT используется для вывода всех клиентов, чье место жительства начинается с буквы U.

**Решение:**

```
ВЫБЕРИТЕ фамилию, имя, улицу, почтовый индекс, город, телефон
FROM  Клиенты
WHERE Населенный пункт LIKE 'U*';
```

Сравнивая текстовое поле с текстовым шаблоном оператор LIKE проясняет, что текстовое поле должно сравниваться с текстовым шаблоном, и, таким образом, в примере должны быть выведены все места с начальной буквой U.

**Примечание:**

При сравнении текстовых полей используйте оператор `LIKE` вместо =.
Поля даты вводятся и отображаются по-разному в системах баз данных.

| База данных | Формат поля даты |
|---|---|
| Oracle, Informix, DB2 | ’2020/10/03’ |
| Access | #2020/10/03# или #03.10.2020# |
| MySQL | ’2020-10-03’ |

Хотя и существуют способы преобразования между немецким и американским правописанием, американское правописание (год /месяц/день), напр. #2020/10/04#, предпочтительное.

### 6.3.4 Оператор BETWEEN

Оператор `BETWEEN` (= между) используется для разработки запросов диапазона. Он может использоваться для текстовых полей, полей даты и числовых полей. BETWEEN выбирает данные между нижним и верхним пределами. Предельные параметры включаются в выборку.

**Пример**

Необходимо отобразить все велосипеды, приобретенные в 2018 году при помощи SQL-оператора

**Решение (в Access):**

```
SELECT № велосипеда, производитель, обозначение, дата
покупки, цена покупки
FROM Велосипеды
WHERE дата покупки Between #2018/01/01 # And # 2018/12/31#;
```

**Результат:**

| Номер велосипеда | Производитель | Обозначение | Дата приобретения | Цена приобретения |
|---|---|---|---|---|
| 3 | Panasonic | FirstClass | 17.01.2018 | 1 960,00 € |
| 4 | Miyata | Devant | 17.03.2018 | 2 800,00 € |
| 6 | Scott | Executive | 18.03.2018 | 2 350,00 € |
| 19 | Hirsch | Klettergemse | 16.06.2018 | 2 400,00 € |
| 24 | Ivecu | Stralys | 31.12.2018 | 1 270,00 € |

### 6.3.5 Оператор IN

Оператор IN используется для сравнения содержимого поля со списком возможного содержимого. Чтобы захватить всех клиентов из агломерации г. Штутгарт, например, может быть использовано следующее условие:

```
... WHERE Н/пункт IN ('Stuttgart', 'Esslingen', 'Fellbach',
'Waiblingen');
```

**Пример**
Запрос ищет всех клиентов из районных центров земли Баден-Вюртемберг.
**Решение:**

```
SELECT фамилия, имя, улица, почтовый индекс, город
FROM Клиенты
WHERE Н/пункт IN ('Stuttgart', 'Karlsruhe', 'Tübingen',
'Freiburg');
```

**Примечание:**
Оператор IN может быть применен к тексту, дате и числовым полям.

### 6.3.6 Работа с нулевыми значениями

Столбцы, в которые не было введено никаких значений, имеют значение NULL (читай: нал). Значение NULL не является ни числом 0, ни строкой с пробелами. Арифметические операции с полем, содержащим значение NULL, приводят к значению NULL. Свойство NOT NULL ищет поля с записью.

**Пример**
**Запрос**

```
SELECT фамилия, имя, улица, почтовый индекс, город, Email
FROM Клиенты
WHERE Email is NOT NULL;
```

выбор всех клиентов с адресом электронной почты.

Если поле таблицы не должно содержать значение NULL, т. Е. Не может быть пустым, это следует учитывать при проектировании таблицы. Поля первичного ключа, напр. не могут быть пустыми. Для других полей, например, фамилия или почтовый индекс, может быть целесообразно запретить значения NULL.

### 6.3.7 Сортировка данных

Инструкция ORDER BY позволяет сортировать выходные данные по содержимому полей. Порядок сортировки указывается с добавлением аргумента ASC (по возрастанию) или DESC (по убыванию). По умолчанию (без аргумента) происходит сортировка в порядке возрастания.

**Пример**
Клиенты отображаются в алфавитном порядке:

```
SELECT *
FROM Клиенты
ORDER BY фамилия, имя ASC
```

| № клиента | Обращение | Фамилия | Имя | Улица | Почтовый индекс | Населенный пункт | Телефон |
|---|---|---|---|---|---|---|---|
| 16 | Госпожа | Адлер | Астрид | Гюнтербургштрассе 56 | 40213 | Дюссельдорф | 0211 543210 |
| 25 | Господин | Амини | Ханс | Бухенландвег 120 | 89075 | Ульм | 0731 4538254 |
| 23 | Господин | Бадер | Михаэль | Блаутальалле 235 | 89075 | Ульм | 4567890 |
| 6 | Госпожа | Мюллер | Андреа | Адельштрассе 13 | 88446 | Биберах | 07364/894623 |
| 3 | Господина | Мюллер | Ханс | Фридбергер Ландштрассе 204 | 89077 | Аугсбург | 0821/434343 |
| 2 | Госпожа | Мюллер | Герта | Рингельштассе 2 | 86416 | Крумбах | 08282/465432 |
| 5 | Госпожа | Winter | Сюзанна | Ам Вальдранд 1 | 89077 | Ульм | 0731/4892 |
| 8 | Господина | Цвибель | Карл | Хюгельштрассе 123 | 89123 | Эльхинген | 07321/745645 |

Если в качестве критериев сортировки указано два содержимого поля, то сортировка происходит после первого (здесь фамилия), с тем же содержимым поля (например, «Мюллер») дополнительно после второго поля (здесь имя).

Если после слова SELECT следует символ «*», то для отображения выбираются все поля. Порядок сортировки может быть по двум критериям

**Пример**

Велосипеды должны быть отсортированы в соответствии с датой их покупки, начиная с самых новых до самых старых. На ту же дату покупки заказ должен быть отсортирован в порядке возрастания по производителю:

```
SELECT *
FROM Велосипеды
ORDER BY дата покупки DESC, производитель;
```

| Номер велосипеда | Производитель | Обозначение | Тип | Цена приобретения | Дата приобретения | Последнее техническое обслуживание |
|---|---|---|---|---|---|---|
| 23 | Hercules | Davos | Дорожный велосипед | 890,00 € | 18.08.2017 | 08.09.2018 |
| 24 | Peugeot | Stralis | Дорожный велосипед | 1 270,00 € | 18.08.2017 | 01.01.2018 |
| 26 | Yamaha | Sutra | Гоночный велосипед | 4 560,00 € | 18.08.2017 | 01.01.2018 |
| 7 | Hercules | GoClimb | Горный велосипед | 850,00 € | 26.04.2016 | 08.01.2018 |
| 8 | Panasonic | EasyRide | Дорожный велосипед | 960,00 € | 26.04.2018 | 08.05.2019 |
| 18 | Hirsch | Schneller Hirsch | Гоночный велосипед | 2 100,00 € | 13.01.2018 | 19.06.2019 |
| 20 | Panasonic | FirstClass | Дорожный велосипед | 1 325,00 € | 05.10.2009 | 10.03.2019 |

### 6.3.8 Ограничение результатов запроса

Ключевое слово TOP ограничивает результаты запроса определенным количеством записей. Например, TOP 3 выводит только первые 3 записи запроса.

**Пример**

Необходимо сдать в прокат 5 велосипедов с самыми дорогими покупными ценами.

**Решение:**

```
SELECT TOP 5 *
FROM Велосипеды
ORDER BY цена покупки DESC;
```

**Результат**:

| Номервелосипеда | Производитель | Обозначение | Тип | Цена приобретения | Дата приобретения | Последнее обслуживание |
|---|---|---|---|---|---|---|
| 5 | Mercier | Excalibur Ultra Light | Гоночный велосипед | 3 750,00 € | 27.04.2016 | 08.01.2020 |
| 15 | Techno- bike | Supertandem | Дорожный велосипед | 3 200,00 € | 28.12.2016 | 03.06.2020 |
| 17 | Staiger | Supertandem | | 2 950,00 € | 14.01.2016 | 19.06.2020 |
| 13 | Systemo | Hurrican | Горный велосипед | 2 665,00 € | 30.07.2016 | 12.04.2019 |
| 4 | Miyata | Devant | Дорожный велосипед | 2 585,00 € | 17.03.2018 | 29.11.2020 |

Поскольку записи в таблице обычно хранятся в несортированном виде, запрос дополняется функцией сортировки `ORDER BY` покупная Цена приобретения DESC.

Функция TOP также может выводить определенный процент записей. Пример синтаксиса: SELECT TOP 10 percent …

**Задача**

Напишите запрос, чтобы сдать в прокат 10 процентов новых велосипедов.

Решение:

```
SELECT TOP 10 percent *
FROM Велосипеды
ORDER BY дата покупки DESC;
```

Оператор сортировки также является DESC, потому что новые значения даты считаются большими, чем предыдущие значения.

### 6.3.9 Функции в SELECT-запросах

**Агрегатные функции**

Агрегатные функции (также: групповые функции) оценивают один или несколько столбцов таблицы по определенным критериям. Результат, например, сумма столбца выводится в поле.

| Функция | Результат |
|---|---|
| MIN (имя столбца) | Минимальное значение содержимого поля столбца |
| MAX (имя столбца) | Максимальное значение |
| COUNT (имя столбца) | Количество существующих записей в столбце |
| COUNT (*) | Количество всех записей |
| SUM (имя столбца) | Сумма содержимого поля столбца |
| AVG (имя столбца) | Среднее арифметическое содержимого поля, AVG от англ.: average = среднее, |

**Пример**

Необходимо создать SQL-запрос для вывода суммарной покупной стоимости всех велосипедов.

**Решение:**

```
SELECT SUM (начальная цена) AS summe
FROM Велосипеды;
```

**Результат:**

| Сумма |
|---|
| 41 337,00 € |

Чтобы обеспечить вывод столбца с желаемым заголовком при выдаче результата используется оператор AS (= как), за которым следует новый заголовок, например, сумма.

Результатом статистической функции обычно является одно поле. Попытка использовать ту же команду для вывода содержимого другого столбца, например, номера велосипеда, приводит к сообщению об ошибке

**Примечание:**

В простом операторе `SELECT` с агрегатной функцией не может быть отображено больше значений столбца.

Функция `COUNT` (имя столбца) подсчитывает количество полей в столбце, которые не являются пустыми. Инструкция `SELECT COUNT` (Email) FROM Клиенты определяет количество клиентов, для которых зарегистрирован адрес электронной почты.

Инструкция `SELECT COUNT(*) FROM` Клиенты учитывает все записи таблицы Клиенты.

**Примечания:**

Чтобы охватить все записи в таблице, поиск выполняется с помощью COUNT(*).

Функции `SUM () и AVG ()` могут применяться только к содержимому числового столбца.

Функции `MIN () и MAX ()` можно использовать с полями текста и даты, причем текстовые поля обрабатываются в соответствии с кодом ASCII: A «меньше», чем Z, а строчные буквы «больше», чем верхний регистр, но «меньше», чем специальные символы. Более поздняя дата «больше», чем предыдущая.

**Пример**

Следующая инструкция SQL определяет дату приобретения последнего велосипеда.

```
SELECT MAX (дата покупки)
FROM Велосипеды;
```

**Арифметическая операция**

Числовые значения столбцов могут использоваться для выполнения арифметических операций, таких как сложение, вычитание, умножение и деление.

Это также относится к значениям даты, поскольку они хранятся как целые числа.

**Функции даты**

Функции даты позволяют обрабатывать значения даты. Например, номер года можно отфильтровать по дате или текущую дату можно обработать динамически.

**Пример**

Для того, чтобы перечислить все велосипеды, дата обслуживания которых состоялась более 100 дней назад, необходима следующая инструкция:

```
SELECT Номер велосипеда, обозначение, последнее обслуживание
FROM Велосипеды
WHERE Последнее обслуживание < DATE()-100;
```

Функция DATE () отображает системную дату. Из этого значения вычитается число 100 и результат сравнивается с содержимым поля Последнее Обслуживание. Выводятся все записи с меньшим значением.

**Примечание:**

Функции даты не являются частью стандарта SQL, но они существуют практически во всех системах баз данных. Однако они существенно различаются. Они очень зависят от используемой системы баз данных. При необходимости обратитесь к соответствующему руководству или справочной функции.

В Access кроме функции DATE (), которая отображает текущую дату, есть дополнительные функции даты:

| Функция | Результат |
|---|---|
| `DATE()` | Текущая дата |
| `DAY (дата)` | День из содержимого поля Дата в виде числа от 1 до 31 |
| `MONTH (дата)` | Месяц в виде числа от 1 до 12 |
| `YEAR (дата)` | Год в виде четырехзначного числа |
| `ISDATE (дата)` | Проверяет правильность даты |
| `Формат (выражение, 'format')` | Форматирует дату в выражении по умолчанию:<br>D, DD: номер дня,<br>DDD, DDDD: номер дня недели,<br>M, MM: номер месяца,<br>MMM, MMMM: имя месяца<br>YY, YYYY: номер года |

Оператор `FORMAT (выражение «формат»)` может использоваться для вывода значений даты в отформатированном виде.

**Пример**

Дата полей возврата в таблице данных аренды должна отображаться в форме «Пятница, 26 апреля 2019 г.».

```
SELECT FORMAT (дата возврата, 'DDDD, DD. MMMM YYYY') AS
['дата счета-фактуры']
FROM даты_проката;
```

**Результат:**

| Дата возврата |
|---|
| Вторник, 06 февраля 2018 г. |
| Среда, 04 октября 2017 г. |
| Вторник, 12 марта 2019 г. |
| Суббота, 27 апреля 2019 г. |

**Функции символьной строки**

Различные системы баз данных предоставляют разные функции для обработки и преобразования символьных строк. Однако действие соответствующих функций очень похоже. Например, в системе баз данных Oracle функция ASCII (n) возвращает кодовый номер символа ASCII, в Access эта функция – ASC (n) делает то же самое.

| Функция | Действие, примеры |
|---|---|
| ASC (n) | Отображает номер кода n, ASC(‘M’) в → 77. |
| LCASE (строка) | Преобразует символы в нижний регистр, LCASE (’Müller’) в → ’müller’ |
| LEFT (строка, длина) | Дает выровненную по левому краю символьную подстроку, LEFT(‘Müller’, 2) в → ‘Mü’. |
| LEN (строка) | Выводит число символов, LEN(‘Müller’) → 6. |
| RIGHT (строка, длина) | Дает символьную строку, выровненную по правому краю, см. LEFT (). |
| UCASE (строка) | Преобразует символы в врехний регистр, UCASE(’text’) в →’TEXT’ |
| & | Оператор для объединения (в Access)’Hans’ & ’ ’ & ’Müller’ в → ’Hans Müller’ |

**Пример**

Независимо от орфографии требуется оператор, который выведет фамилии клиента, чтобы первая буква отображалась как заглавная буква, а следующие буквы отображались строчными буквами.

```
SELECT Left(UCase(Nachname),1) & LCase(Right(Nachname,
Len(Фамилия) -1)) AS [ 'фамилия']
FROM Клиенты;
```

Случайный ввод фамилия «Мюллер» преобразуется этой инструкцией в «Мюллер».

**Функции преобразования**

Например, с помощью функций преобразования символьные строки преобразуются в числа и наоборот. Примеры:

| Функция | Действие, примеры |
|---|---|
| CDATE ('строка даты‘) | Символьная строка, число в типе даты<br>DATE CDATE ('12 .03.2019 ') → 12.03.2019 |
| STR (число) | Число в символьную строку,<br>STR(89077) → `89077’ |
| VAL (строка) | Символьную строку в число,<br>VAL(‘34’) →34 |
| CINT (номер) | Конвертирует в INTEGER, округляет до<br>CINT (45,8) → 46<br>CINT (-67,4) → 67 |

Например, функция CDATE («дата») преобразует символьную строку чисел или целое число в дату, насколько это возможно в зависимости от исходного типа.

Функция CINT() округляет до следующего более высокого целого численного значения и отображает его в результате.

**Пример**

Следующий оператор SQL отображает цены аренды каждого велосипеда в таблице групп цен, увеличенные на 5%, округленные до полной суммы в евро.

```
SELECT CINT(цена*1.05)
FROM Ценовые группы;
```

**Математические и логические функции**

Для решения математических задач, например, для формирования квадратного корня содержимого поля, используются следующие функции

| Функция | Результат |
|---|---|
| ABS(число) | Абсолютное значение (без отрицательного знака) |
| SQR(число) | Корень числа: SQR (16) = 4 |
| RND() | Случайное число от 0 до 1 |
| ISNULL (выражение) | Значение = 1, если содержимое поля пусто, значение = 0, если содержимое поля не пусто. |
| EXP (значение) | Отображает значение e с номером Эйлера e = 2.71828 ... в качестве базы. |
| LOG (значение) | Натуральный логарифм, т. е. LN(EXP(значение)) = значение. |
| LOG10 (значение) | Логарифм числового значения в основание 10 |
| SIGN (значение) | -1 для отрицательных значений и 1 для положительных значений |
| MOD (значение, делитель) | Остаточная стоимость (целого численного) деления значения/делителя |

**Пример**

Рассчитать диаметр из таблицы с поперечным сечением кабеля.

**Решение:**

```
SELECT площадь, 2*SQR(площадь/(3.1415*2)) AS диаметр
FROM Кабельные линии;
```

Функция RND() генерирует случайное число с плавающей запятой число >0 и <1, например, 0,3452419.

Функция ISNULL(выражение) проверяет, является ли выражение, например, значение поля пустым. Если это так, функция возвращает значение 1 (‘true’), иначе – значение 0 ('false’).

### 6.3.10 Группировка данных

Группировка означает группирование данных в поля с одинаковым содержимым поля. Это делается с помощью SQL-выражения `GROUP BY,` за которым следует имя поля.

**Пример**

Следующая инструкция идентифицирует зону обслуживания компании Faradiso при указании места жительства и количества клиентов из каждого населенного пункта:

```
SELECT н/пункт, COUNT (№ клиента) AS количество
FROM Клиенты
GROUP BY н/пункт;
```

| Населенный пункт | Количество |
|---|---|
| Берлин | 2 |
| Биберах | 2 |
| Йеттинген-Шеппах | 2 |
| Дюссельдорф | 35 |
| Нехарсулм | 1 |
| Хан-Грютен | 12 |
| Ульм | 11 |
| Цвиккау | 1 |
| ... | ... |

Сначала используется выражение `GROUP BY` Н/пункт для формирования группы из всех строк базы данных, имеющих одинаковую запись в поле н/пункт. Впоследствии в каждой из этих групп используется функция COUNT (№ клиента) для подсчета количества записей исходя из № клиента.

**Примечание:**
Все поля, перечисленные после ключевого слова `SELECT` в дополнение к статистической функции, также должны быть указаны после оператора группировки `GROUP BY`, в противном случае появляется сообщение об ошибке.

**Условия для группировки – HAVING:**

Отображение отдельных групп также можно сделать условным с помощью оператора условия HAVING.

**Пример**

SELECT Производитель, SUM (Покупная цена) AS Сумма FROM Велосипеды
GROUP BY Производители
HAVING SUM (начальная цена) > 4000;

| Производитель | Сумма |
|---|---|
| Hirsch | 5 932,00 € |
| Panasonic | 6 095,00 € |

Например, утверждение HAVING SUM (цена покупки)> 4000 приводит к тому, что будут выпущены только группы, чья общая цена покупки («общая сумма») составляет более 4000 €.

**Примечание:**
Условие после ключевого слова HAVING всегда должно быть применимо ко всей группе.

### 6.3.11 Запросы по нескольким таблицам (JOIN)

Joins (to join = связать) создает логические связи между несколькими таблицами.

Например, чтобы вывести все данные об аренде с соответствующими именами клиентов, должны отображаться поля двух таблиц Клиенты и Резервирование. Инструкция выглядит таким образом:

SELECT.№ Клиента, Фамилия, Имя, Резервирование,
Дата сдачи в аренду
FROM Клиенты, Бронирование;

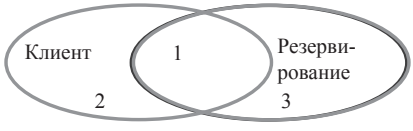
Результатом является вывод, который объединяет всех клиентов со всеми данными аренды. Из таблицы с 50 записями и из таблицы с 400 записями генерируется набор результатов с 20000 записей.

Количество записей умножается на количество записей каждой таблицы, которую вы добавляете.

**Типы объединений**

На примере двух таблиц Клиенты и Резервирование объясняются различные объединения.

- клиенты, которые заказали резервирование (1),
- клиенты, которые не заказывали резервирование (2) и
- есть резервирования, не связанные с клиентами (3).



| Фамилия | Представление | Пояснение | Пример синтаксиса |
|---|---|---|---|
| INNER JOIN, EQUI JOIN |  | Отображение клиентов, зарезервировавших велосипеды, со связанными резервированиями | SELECT * FROM Клиенты INNER JOIN Резервирование ON клиенты.№ клиента = резервирования.№ клиента; |
| NATURAL JOIN |  | Результат INNER JOIN | SELECT * FROM клиенты, бронирование WHERE № клиента= резервирование.№ клиента; |
| LEFT JOIN, |  | Отображение всех клиентов и (если таковые имеются) связанные резервирования. | SELECT * FROM Клиенты LEFT JOIN Резервирования ON клиенты.№ клиента = резервирования.№ клиента; |
| RIGHT JOIN, |  | Вывод всех резервирований – и – соответствующих клиентов | SELECT * FROM Клиенты RIGHT JOIN Резервирование ON клиенты.№ клиента = резервирования.№ клиента; |
| LEFT OUTER JOIN |  | Вывод только тех клиентов, которые не заказывали резервирование. | SELECT * FROM Клиенты LEFT JOIN Резервирование ON № клиента = резервирование.№ клиента WHERE Резервирования.№ клиента is NULL; |
| RIGHT OUTER JOIN |  | Вывод резервирований, которые не привязаны ни к одному из клиентов. | SELECT * FROM Клиенты RIGHT JOIN Резервирование ON № клиента = резервирование.№ клиента WHERE Резервирования.№ клиента is NULL; |
| FULL OUTER JOIN |  | Вывод всех клиентов без резервирования и всех резервирований, не привязанных к клиентам | SELECT * FROM Клиенты FULL JOIN Резервирование ON № клиента = резервирование.№ клиента WHERE Резервирование. № клиента is NULL; (Не поддерживается ACCESS-SQL) |

**Inner-Joins, Equi-Joins, Natural Joins**

Для запросов, включающих несколько таблиц, объединения равенства, внутренних соединений. В этом примере могут быть приняты во внимание только те клиенты, чей номер указан в таблице Резервирования. Для этого в команде SQL должно быть определено уникальное правило связывания. Это реализуется условием

```
WHERE № клиента = Резервирование.№ клиента.
```

Таким образом, полная инструкция имеет следующий вид:

```
SELECT № клиента, Фамилия, Имя, Отчество, Резервирование,
Дата сдачи в прокат
FROM Клиенты, Резервирование
WHERE №клиента = № Резервирования;
```

Чтобы различать поля с одинаковыми именами в разных таблицах, к имени столбца добавляется префикс имени таблицы, например № клиента.

Объединения нескольких таблиц описываются в запросе SQL, например, в форме условия WHERE.

**Пример**

Выводятся все резервирования с данными клиента (номер клиента, фамилия, имя, место жительства) и данные велосипеда (номер велосипеда, производитель, название) Инструкция:

```
SELECT K.№ клиента, фамилия, имя, н/пункт, дата сдачи в прокат,
срок проката AS срок проката, № велосипеда, производитель
FROM Клиенты AS K, Велосипеды AS F, Резервирования AS R
WHERE K.№ клиента=R.№ клиента И
F.№ велосипеда=R.№ велосипеда;
```

В дополнительном поле Общ. стоим указывается общая стоимость проката P.Цена*L длительность проката AS Общ. стоим высчитывается и выдается. Результат:

```
SELECT K.№ клиента, фамилия, имя, н/пункт, дата сдачи в прокат,
длительность проката AS Lдлительность, F.велосипеда AS №,
производитель,
P.цена *Lсрок аренды AS Общ.стоим
FROM Клиенты AS K, ценовые группы AS P, велосипед AS F,
Резервирования AS R
WHERE к. номер клиента = R. KdNr AND F. велосипед номер=R.
FRadNr AND
P. ценовая группа=F ценовая группа;
```

**Вывод выглядит следующим образом:**

| № клиента | Фамилия | Имя | Срок сдачи в прокат | № | Производитель | Общ. стоим. |
|---|---|---|---|---|---|---|
| 2 | Мюллер | Герта | 9 | 23 | Hercules | 108,00 € |
| 3 | Мюллер | Ханс | 18 | 13 | Systemo | 486,00 € |
| 5 | Винтер | Сюзанна | 12 | 8 | Panasonic | 132,00 € |
| 5 | Винтер | Сюзанна | 9 | 21 | Hercules | 135,00 € |
| 8 | Цвибель | Карл | 9 | 8 | Panasonic | 99,00 € |
| 10 | Хазе | Ханна | 12 | 21 | Hercules | 180,00 € |

Поскольку таблица Резервирование связана с таблицей клиентов с помощью объединения, это также указывается как условие WHERE в соответствии с отношениями таблиц Резервирование – циклы – Ценовые группы. Поэтому при разработке сложного запроса SQL для нескольких таблиц необходимо знать природу отношений и затронутых полей.

Чтобы упростить синтаксис SQL, имена таблиц в запросе могут быть переименованы. Для этого в операторе FROM имени таблицы присваивается характерная буква, например, K, P, R, F (то есть велосипеды F). Это назначит таблице новое (временное) имя. В этом случае таблица доступна в запросе только по этой букве.

Ключевое слово AS используется для переименования полей в выходных данных. Это – важно для вычисляемых полей, для которых никакое имя не определено иначе.

В приведенном выше примере это: P.Цена*LСрок аренды AS Общ. стоим.

Внутренние объединения также могут быть описаны с помощью ключевых слов INNER JOIN.

**Пример**

Все зарезервированные велосипеды (`номер велосипеда, название, производитель`) должны отображаться с важными данными резервирования (`ResNr, LoanDate, RentalDuration`). Инструкция:

```
SELECT F.№велосипеда, F.Название, F.Производитель, F.последнее
   техобслуживание, R.№ резервирования, R.дата сдачи в прокат,
   R.срок проката
FROM Велосипеды F INNER JOIN Резервирования ON R F.№ велосипеда = R.
   № велосипеда;
```

Выходные результаты:

**Вывод результатов**:

| Номер велосипеда | Название | Производитель | Последнее Обслуживание | № резервирования | Дата сдачи в прокат | Продолжительность проката |
|---|---|---|---|---|---|---|
| 4 | Devant | Miyata | 29.08.2018 | 20 | 22.01.2021 | 2 |
| 4 | Devant | Miyata | 29.11.2018 | 22 | 01.04.2020 | 5 |
| 8 | Galata | Panasonic | 08.05.2018 | 7 | 18.01.2019 | 9 |
| 13 | Hurrican | Systemo | 12.04.2019 | 5 | 16.06.2020 | 3 |
| 14 | DownHill Racer | Hitachi | 21.05.2019 | 8 | 15.04.2021 | 9 |
| 19 | Klettergemse | Hirsch | 27.02.2018 | 12 | 15.12.2022 | 6 |

**Примечание:**

INNER-Joins, EQUI-Joins und Natural Joins – каждый дает одинаковый результат. Они описываются только с различным синтаксисом. Для естественных объединений описывается соединение таблиц в предложении WHERE, для соединений INNER (= объединений EQUI) это делается условием FROM.

**Left-Joins (также Left-Inner-Joins)**

LEFT-JOIN берет все записи одной таблицы (например, Клиенты) и объединяет их либо со всеми действительными записями в другой таблице (Резервирования), либо с NULL, если во второй таблице не найдено соответствующей записи. Важно знать соответствующую ER-модель для двух таблиц:


Отношение гласит, что у каждого клиента может быть несколько резервирований или вовсе не быть резервирований. В случае, если резервирование не назначено, результат LEFT JOIN в столбце Дата сдачи в прокатсодержит значение NULL.

Если вы хотите включить в запрос резервирования также клиентов, которые еще не зарезервировали велосипед, то условие ссылки должно быть изменено. В то время как Equi-Join соответствует пересечению двух диапазонов: Клиенты и Резервирование, левая область двух величин должна быть включена.


Такой аргумент Left Join может использоваться в Access с общим синтаксисом

```
... FROM tab1 LEFT JOIN tab2 ON tab1.поле = tab2.поле...
```

**Пример**

```
SELECT K. № клиента, фамилия, имя, место, дата сдачи в прокат AS
дата сдачи в прокат, резервирование AS Lсрок сдачи в прокат
FROM клиенты K LEFT JOIN Резервирования R ON K. № клиента=R. №
клиента;
```

**Вывод**:

| № клиента | Фамилия | Имя | Населенный пункт | Дата сдачи в прокат | Срок сдачи в прокат |
|---|---|---|---|---|---|
| 1 | Пальмерт | Карло | Йеттинген | | |
| 2 | Мюллер | Герта | Крумбах | 16.10.2019 | 9 |
| 2 | Мюллер | Герта | Крумбах | 22.01.2020 | 3 |
| 3 | Мюллер | Ханс | Ульм | 27.02.2022 | 18 |
| 4 | Шульце | Анна | Ульм | | |
| 5 | Винтер | Сюзанна | Ульм | 16.10.2021 | 9 |
| 5 | Винтер | Сюзанна | Ульм | 01.05.2021 | 12 |
| 6 | Мюллер | Андреа | Биберах | | |
| 7 | Цвибель | Карл | Эльхинген | 18.01.2021 | 9 |

**Outer-Joins**

Outer-Joins (например, Left Outer Join) выводят в качестве результата только те записи, на которые нет справочных данных во второй таблице.

**Пример**

отобразить данные клиентов, которые еще не сделали резервирование.

```
SELECT K.№ клиента, фамилия, имя, отчество, место, даты проката AS
дата сдачи в прокат, длительность сдачи AS срок сдачи
FROM Клиенты K LEFT JOIN резервирование R ON K. .№ клиента=
WHERE R № клиента .№ клиента IS NULL;
Вывод:
```

| № клиента | Фамилия | Имя | Населенный пункт | Дата сдачи в прокат | Срок сдачи в прокат |
|---|---|---|---|---|---|
| 1 | Пальмерт | Карло | Йеттинген | | |
| 4 | Шульце | Анна | Ульм | | |
| 6 | Мюллер | Андреа | Биберах | | |
| 9 | Уинклер | Клаус | Тальфинген | | |
| 10 | Хазе | Ханна | Гюнцбург | | |
| 11 | Хан | Изабель | Зенден | | |

**Заметка:**

Inner-Join Equi-Join, Natural Join: запросы к нескольким таблицам, где отображается только пересечение данных.

Left-Join / Right Join (также Left-Inner-Join и Right-Inner-Join): в дополнение к пересечению также включается левый/правый раздел подмножеств.

Outer-Join: исключает пересечение и отображает только те записи, которые не имеют записей в подключенной таблице. Задается в Access путем добавления оператора `WHERE [имя поля] NULL`.

## 6.3.12 Подзапросы

Подзапросы (Subqueries) необходимы, если условие поиска зависит от результата другого запроса. Выводятся только те данные, которые приводят к логически оцениваемому условию в разделе WHERE, в противном случае появляется сообщение об ошибке.

**Пример**

Создать запрос, который отображает имя клиента, бронирование которого должно быть выполнено в следующем порядке. Вывести номер и название велосипеда. Полезно сначала сформировать запрос, который выводит следующую дату бронирования. Функция MIN (дата сдачи в прокат) выводит самую раннюю сохраненную дату резервирования, затем условие WHERE> = DATE () ограничивает запрос наименьшей датой на сегодняшний день.

Прошлые резервирования Будущие резервирования:
WHERE Дата сдачи в прокат > = DATE()


Таким образом, запрос для поиска желаемого значения даты, который затем служит подзапросом:
SELECT MIN (дата сдачи в прокат) FROM резервирование
Где дата сдачи в прокат> = Date();
Результат этого запроса представляет значение наименьшей будущей датой сдачи, например

| Подзапрос |
|---|
| 16.10.201X |

Теперь соответствующая запись в таблице Резервирование должна быть найдена по условию WHERE Borrow Date = [Subquery]. Поскольку данные считываются из нескольких таблиц, таблицы Клиенты, Резервирование и Велосипеды должны быть связаны необходимыми уравнениями.
Весь запрос:

```
SELECT F.№ велосипеда, F.название, фамилия, дата сдачи в прокат
FROM Велосипеды AS F, Резервирование AS R, Клиенты AS K
WHERE K.№ клиента= R.№ клиента
AND F. № велосипеда = R. № велосипеда
AND дата сдачи в прокат= (
        SELECT MIN (дата сдачи в прокат
        FROM Резервирование
        WHERE Дата сдачи в прокат > = DATE()
        );
```

Natural Joins

Subquery

Результат:

| Следующая сдача в прокат | | | |
|---|---|---|---|
| **№** | **Обозначение** | **Фамилия** | **Дата сдачи в проект** |
| 21 | StormRide | Winter | 16.10.201X |

**Примечание:**
Результат подзапроса должен соответствовать типу данных с полем сравнения. Кроме того, подзапрос может давать только несколько результатов, если, например, он является частью условия с оператором IN и соответствует списку полей.

Сложные запросы всегда должны быть модульными. В результате ошибки могут быть своевременно обнаружены и легко устранены.

**Пример**
Вывести данные клиентов, которые проживают в н/пункте производителей велосипедов. Обязательным условием является то, что сначала создается таблица Производители с данными производителей велосипедов.

```
SELECT *
     FROM Клиенты
     WHERE Клиенты. Н/пункт IN
         (
         SELECT Производитель.н/пункт
         FROM Производитель
         );
```

## 6.4 Редактирование данных с помощью SQL

Язык манипулирования данными (DML) в SQL предоставляет возможность обрабатывать данные, вставляя новые записи, удаляя существующие или изменяя их.

### 6.4.1 Вставка строк базы данных

Для обновления данных в соответствии с текущими бизнес-процессами новые данные добавляются, а ненужные удаляются. Добавление записей происходит в SQL с помощью ключевого оператора INSERT.

Общий синтаксис INSERT:

```
INSERT INTO Таблица (Столбец1, Столбец2, ...)
VALUES (Содержание1, Содержание2, ... );
```

Если данные для ввода выбираются из существующей таблицы, включается соответствующий оператор SELECT. Структура инструкции INSERT:

```
INSERT INTO Таблица (Столбец1, Столбец2, ...)
SELECT (Столбец1, Столбец2, ...) ...;
```

**Пример**
Добавить при помощи оператора из SQL новую строку базы данных для велосипеда «Easygo» производителя Stiegl и № велосипеда в таблице Велосипеды.

**Решение:**

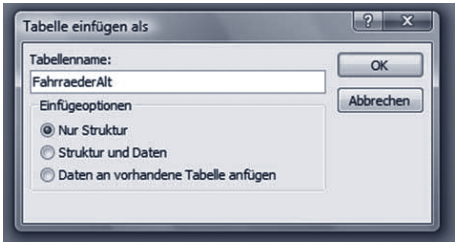
```
INSERT INTO Велосипеды (№ велосипеда, производитель, наименоваие,
тип) VALUES (12, 'Stiegl', 'Easygo', 'Tourenrad');
```

**Примечание:**
Порядок полей, перечисленных после INSERT, соответствует значениям, перечисленным с помощью VALUES. Тип данных полей должен совпадать с типом данных соответствующего значения.

**Пример**
Велосипеды, приобретенные до 2004 года, должны быть выведены из эксплуатации. Чтобы не потерять данные навсегда, вам нужна инструкция, которая вставляет данные велосипеда в таблицу ВелосипедыСтар для целей резервного копирования.

```
INSERT INTO ВелосипедыСтар
SELECT *
FROM Велосипеды
WHERE дата покупки <#1/1/2004#;
```

Необходима таблица ВелосипедыСтар с той же структурой. Ее можно создать с помощью оператора CREATE TABLE или существующей таблицы Велосипеды. например в Access можно копировать, используя команду контекстного меню Копироватьа потом Вставить. выбрана опция Только структура.




В этом примере оператор SELECT ищет все записи о велосипедах, приобретенных до 2004 года, а затем сохраняет их в новой таблице.

**Примечание:**
Типы данных и имена соответствующих полей обеих таблиц должны соответствовать этому методу.

### 6.4.2 Удаление строк базы данных

Удаление выгружаемых записей происходит с помощью команды SQL DELETE. Общий синтаксис:

```
DELETE Имя атрибута/ * FROM таблица
WHERE условие;
```

**Пример**
Инструкция SQL предназначена для вывода велосипедов, приобретенных до 2004 года, из таблицывелосипеды.
**Решение:**

```
DELETE *
FROM Велосипеды
WHERE дата покупки <#1/1/2004#;
```

### 6.4.3 Обновление данных

Чтобы изменить содержимое поля, используйте команду обновления UPDATE, например если многие записи должны быть изменены в одном направлении.

Общий синтаксис:

```
UPDATE Таблица
SET Attributname = значение
WHERE условие;
```

**Пример**
Группы цен используются в примере базы данных Faradiso для сводки велосипедов с одинаковой ценой аренды.
Цена аренды велосипедов, приобретенных до 2006 года, должна быть уменьшена на одну ценовую группу. В соответствующих записях должны сохраниться изменения.
**Решение:**

```
UPDATE велосипеды
SET ценовая группа = ценовая группа – 1
WHERE YEAR (дата покупки) < 2006;
```

Здесь содержимое поля ценовой группы уменьшается на 1 во всех записях данных в таблице Велосипеды, которые соответствуют условию WHERE и были приобретены до 2006 года.

## 6.5 Согласованность базы данных

Согласованность базы данных описывает правильность внутренних структур хранения и путей доступа, чтобы всегда иметь возможность использовать массив данных без противоречий.

Поскольку базы данных обычно управляются несколькими пользователями одновременно, то операции удаления данных не могут быть отменены. Потому что после удаления другие пользователи могут работать с уже измененным массивом данных. Невозможность отмены удаления, таким образом, предотвращает аномалии (логические противоречия) в базе данных, и данные остаются последовательными (не противоречивыми).

**Пример**

Структура базы данных должна гарантировать, что однажды сохраненный адрес клиента не противоречит другому адресу этого клиента. В противном случае отправка счета будет невозможна.

При планировании базы данных необходимо предотвратить аномалии удаления, изменения и аномалии вставки.

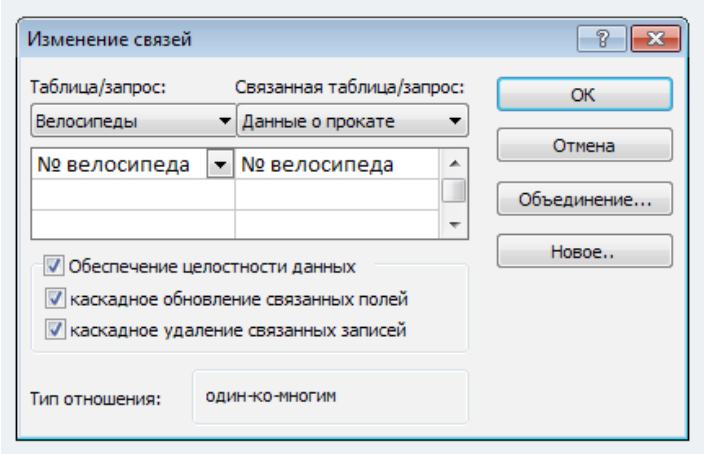
Аномалии удаления возникают, когда удаляется запись, значение первичного ключа которой является зависимой записью из другой таблицы. Если, например, запись данных о клиенте с номером 15 была удалена, то данные аренды, относящиеся к номеру клиента 15, больше не могут быть выведены.

Аномалии изменения возникают, например, из-за многократного сохранения данных. Если данные (например, адрес клиента) изменяются в одной таблице, то они противоречат данным в другой таблице.

Аномалии вставки возникают тогда, когда после вставки строк базы данных значения появляются в базе данных несколько раз и конфликтуют между собой.

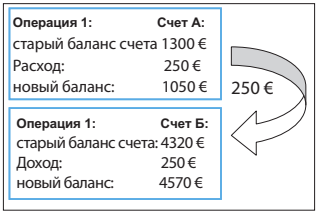
Чтобы предотвратить аномалии, связи устанавливаются уже на этапе проектирования базы данных, так что процедуры удаления могут легко контролироваться.

Если, например, велосипед удален в таблице Велосипеды, а на номер этого велосипеда ссылается запись данных в таблице Прокат, то запись данных проката велосипеда под этим номером также должна быть удалена. В Access это можно сделать, установив флажок Перенос удаления на связанные записи в окне Редактировать отношения:



## 6.6 Транзакции

Базы данных должны оставаться согласованными всегда, даже после аппаратного или программного сбоя. В случае перевода между счетами двух банков перевод может быть действительным только в том случае, если счет A был уменьшен на конкретную сумму и на счет B была зачислена эта сумма.




Если после проведения операции 1 произойдет сбой питания, то остатки счета больше не будут согласованными, поскольку остаток на счете A уже уменьшен на 250 евро, а остаток счета B еще не увеличен.

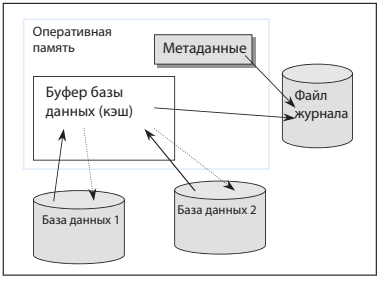
Чтобы убедиться, что передача завершена, две операции всегда выполняются в форме транзакции.

**Определение:**
**Транзакция** – это последовательность SQL-операторов, которые логически связаны друг с другом и обеспечивают согласованность данных.

**Примечание:**
При выполнении транзакции необходимо убедиться, что транзакция выполняется полностью или полностью отменена в случае ошибки.
Последующие инструкции будут выполнены только в том случае, если предыдущие были выполнены успешно.

Чтобы обеспечивать действительность Массива данных, предпринимаются следующие шаги:

| Шаг | Задача |
|---|---|
| 1. Считать данные | Данные считываются с носителя. |
| 2. Запомнить текущее значение данных | Данные, подлежащие изменению, записываются в файл журнала (Before-Image). |
| 3. Изменить данные | Изменение данных в памяти, блокировка этих записей для других пользователей (запись в журнале). |
| 4. Запомнить измененные значения данных | Измененные данные записываются в файл журнала (After-Image). |
| 5. Завершить транзакцию с помощью COMMIT (= Подтверждение) | Запись всех образов и метаданных в файл журнала. Отметьте конец транзакции в файле журнала. Снятие блокировки. |
| 6. Транзакции с `ROLLBACK` (= поворачивать назад) | Сброс метаданных транзакции. Измененные данные, уже записанные в базу данных, будут аннулированы. Снятие блокировки. |
| 7. Сохранить изменения | Измененные данные записываются в базу данных. |




Сначала считываются данные, то есть баланс счета A: 1300 €. Еще не измененные данные теперь записываются как Before-Image (= образ до изменения) для резервного копирования в файле журнала. Это энергонезависимый файл в системе базы данных.

Теперь выполняются изменения данных в оперативной памяти с необходимыми командами (UPDATE, DELETE, INSERT)

. При этом создаются метаданные (= предварительные измененные данные), которые заблокированы для пользователей. Таким образом, остаток на счете A уменьшается с обновлением на 250 €, т. е. до 1050 €. Этот измененный баланс будет сохранен в файле журнала как After-Image (= образ после изменения).

Соответствующие операции теперь должны выполняться с балансом B. Таким образом, новые данные доступны как After-Image в файле журнала.

Если до сих пор не произошла ошибка, команда COMMIT отметит конец транзакции в файле журнала. Снятие блокировок. Метаданные записываются в базу данных.

Если перед командой COMMIT произошла ошибка, инструкция ROLLBACK сбрасывает всю транзакцию со всеми метаданными. Предыдущие данные все еще сохраняются в качестве Before-Images в файле журнала. Поскольку в конце транзакции в файл журнала записывается примечание, его можно обнаружить в любой момент в случае сбоя системы, изменения которого относятся к уже завершенной транзакции.

Если файл журнала не создается на том же носителе, на котором расположена база данных, очень маловероятно, что оба файла данных, файл журнала и база данных, будут удалены одновременно. Таким образом, с помощью последней резервной копии и файла журнала, набор данных может быть восстановлен в любое время для получения согласованного набора данных.

## 6.7 Упражнения к Главе 6

**Задача 1**

Разработать базу данных для врачебного кабинета со следующими требованиями:

► Необходимо управлять информацией об отдельных процедурах пациентов. Для этого необходимо сохранить дату, диагноз и, возможно, комментарии.

► Каждый пациент является участником только одного фонда медицинского страхования. Пациенты с частной страховкой не учитываются

► В кабинете предоставляются такие услуги как (сбор крови, мониторинг артериального давления, лабораторные анализы, детальное обследование, выписка направлений, выдача справок о состоянии здоровья, рецептов). Здесь иногда выдаются несколько назначений на лечение. Услуги должны быть записаны в соответствии с каталогом с номером, описанием услуги и ценой.

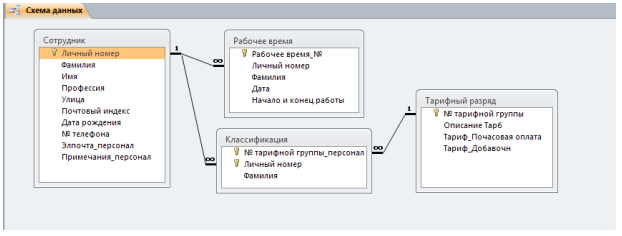
► В отношении рецептов должны сохраняться данные о назначенном препарате, производителе, а также о цене и основном действующем веществе.

► Должна быть доступна информация о стоимости лечения для каждого пациента, включая предоставляемые услуги. (Фиксация возможного увеличения стоимости не требуется.)

Создайте подходящую базу данных. Установите необходимые связи.

**Задача 2**

Для расчета заработной платы сотрудников в кабинете врача используется следующее расширение базы данных «Кабинет врача».




a) В каких моментах этот проект нарушает правила нормализации первой, второй и третьей нормальной формы? Предложите, как можно избежать нарушений.

Создайте SQL-запросы, выполняющие следующие задачи:

b) Вывести информацию о всех сотрудниках с почтовым индексом 46*** .
c) Сколько сотрудников работает в кабинете врача?
d) Какие сотрудники работали 23.12.2015?

**Задача 3**

В одной из организаций необходимо создать управление проектами с помощью базы данных. Доступ каждого сотрудника в систему должен осуществляться с помощью пароля.

Пароль хранится в таблице сотрудников. Для каждого проекта должны быть сохранены имя, а также дата начала, дата окончания и менеджер проекта.

Для каждого проекта в качестве руководителя проекта выбирается именно один сотрудник.

Предполагается, что для каждого сотрудника будет записана продолжительность рабочего времени (в часах) и описание выполненной работы, которую он выполнял в определенный день для конкретного проекта.

a) Составьте графическое представление связей таблиц посредством ER-диаграммы и укажите тип связей между таблицами.
b) Сколько сотрудников участвуют в проектах? Создайте SQL-запрос, который выводит номер и имя проекта, а также количество сотрудников.

**Задача 4**

Даны следующие соотношения (таблицы) базы данных оператора мобильной связи.

| **Клиенты** | **Договоры** | **Разговоры** | **Тарифы** |
|---|---|---|---|
| клиент_№ | договор_№ | разговор_№ | тариф_№ |
| клиент_фамилия | № договор_клиента | № договор_разг | тариф_минуты |
| клиент_имя | договор_арт | разг_целев номер | |
| клиент_улица | договор_начало | разговор_дата | |
| клиент_индекс | договор_окончан | разг_начало | |
| клиент_дата рожд | договор_цена | разговора_продолжительность (в мин.) | |
| | | разг_тариф_№ | |

Создайте SQL-запросы, выполняющие следующие задачи:

a) Какой клиент провел разговор № 34890?
b) Должна выводиться информация о всех разговорах с началом и продолжительностью, а также тарифом и соответствующей стоимостью, которые провел клиент Герман Майер 25.03.2015.
c) Какой контракт (выдача с именем клиента) заканчивается следующим?
d) Вывести информацию о количестве разговоров каждого клиента.
e) Что делает SQL-оператор » HAVING ...» (пример)?

**Задача 5:**

Совместный врачебный кабинет предлагает своим пациентам записаться на прием к врачу онлайн. Необходимо разработать базу данных, которая, кроме всего прочего, содержит следующие отношения:

Врач (№ врача, имя, специальность, № телефона)

Прием (№ приема, № врача, Дата, Время начала, Время окончания, № пациента)

Пациент (№ пациента, Фамилия, Дата рождения)

a) Отметьте поля первичного ключа и поля внешнего ключа.

b) Помощнику врача (фельдшеру) необходим список всех приемов у доктора Галленбиттера на сегодня в хронологическом порядке.

Сформулируйте SQL-запрос для требуемого списка.

c) Необходимо ввести данные о 20-минутном приеме пациента Питера Пирсинга (№ пациента = 27) у доктора Мюллера (№ врача=2) 17 июня в 16:30

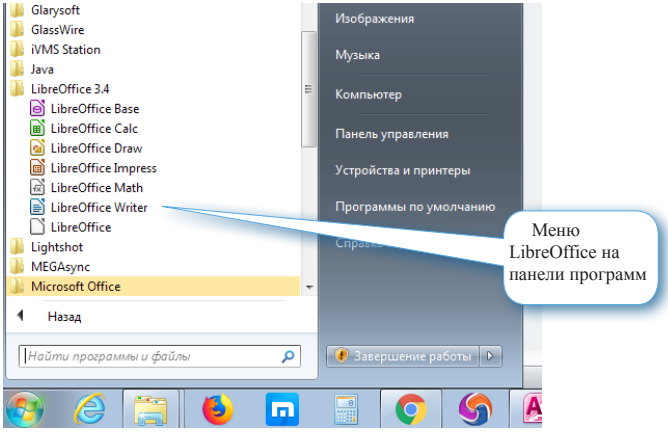
Создайте SQL-инструкцию, которая введет эту запись.

(№ приема автоматически присваивается системой управления базами данных.)

# 7 LibreOffice Base

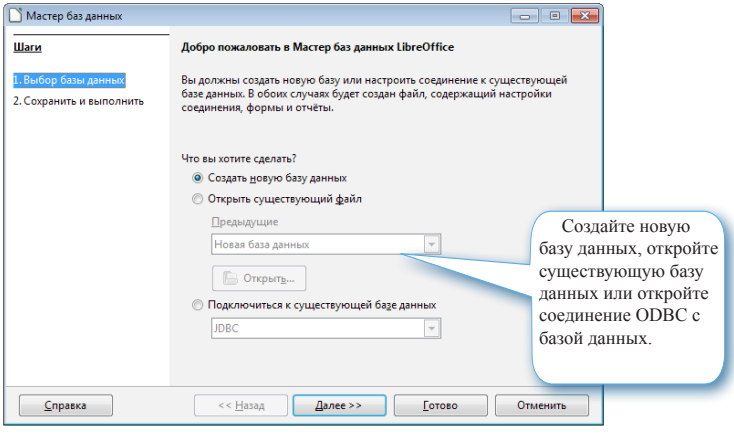
## 7.1 Создание базы данных

LibreOffice Base позволяет создавать и управлять базами данных.

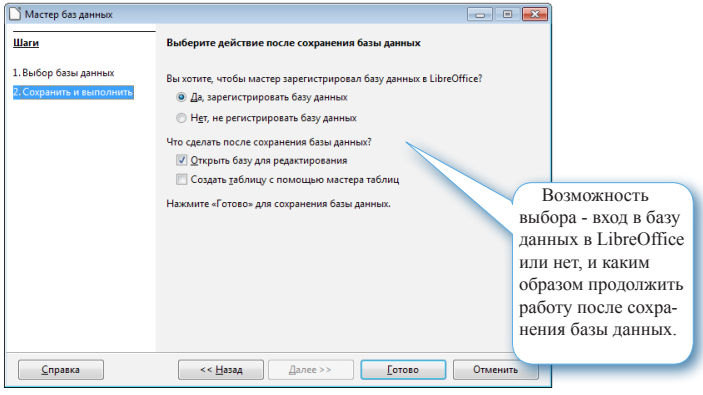



При запуске LibreOffice Base запускается Мастер базы данных. В окне Мастера базы данных доступны следующие опции:

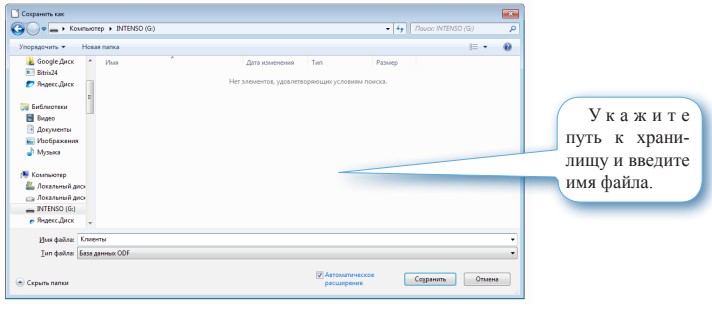
- Создать новую базу данных,
- Открыть существующую базу данных или
- Создать подключение к существующей базе данных.



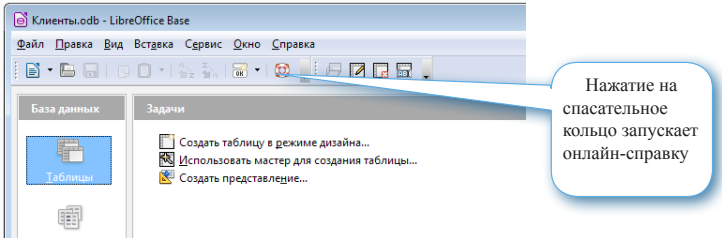
Выбрав Создать новую базу данных и нажав Далее, вы перейдете на 2-е окно выбора Завершить и продолжить.



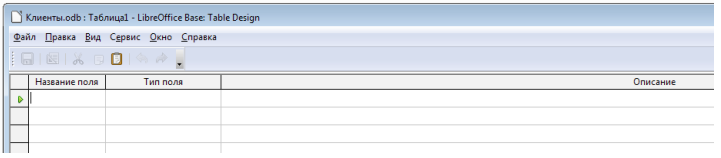

После нажатия кнопки Готово задается имя базы данных и ее местоположение. В этом примере указана база данных с именем файла Клиент. Тип файла по умолчанию для базы данных odb .



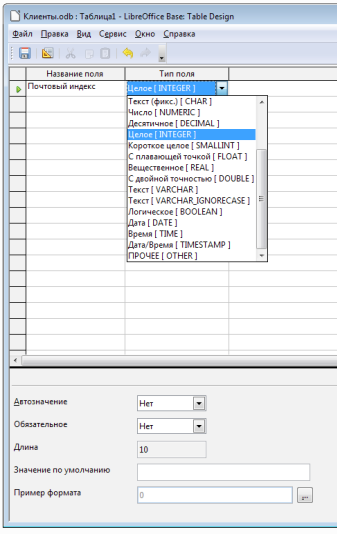

Окно базы данных содержит слева область База данных с иконками выбора Таблицы, Запросы, Формы и Отчеты. На правой панели в разделе Задачи находятся пункты меню Создать таблицу в режиме конструктора.., Создать таблицу с помощью мастера... и Создать представление... При нажатии на значок со спасательным кругом (при подключении к Интернету будет) загружена онлайн-справка.



При нажатии кнопки Создать таблицу в конструкторе... запускается окно конструктора, в котором создается структура таблицы, то есть Имена полей для атрибутов, Тип поля для типа данных и, при необходимости, Описание.




После ввода имени поля (например, индекс) вы можете выбрать множество типов данных в поле «Тип поля» с помощью выпадающего меню.




В качестве типов данных доступны следующие варианты выбора:

Числовые поля

Целые числа

Tiny Integer [ITINYINT ], 1-байтовое целое число в диапазоне значений от -128 до 127, также называемое Byte.

Small Integer [ SMALLINT ], 2-байтовое целое число в диапазоне от -32 768 до 32 767, называется Integer.

Целое [ INTEGER ], 4-байтовые целочисленные значения в диапазоне от -2 147 483 648 до 2 147 483 647, также называемое Long Integer.

BigInt [ BIGINT ], 8-байтовое целое число в диапазоне значений + / – 9,2 триллионов, часто не поддерживается проприетарными программами баз данных.

**Десятичные дроби**

DECIMAL, десятичная дробь до 10 цифр, десятичные результаты округляются до указанного количества десятичных разрядов, особенно в случае денежных сумм, также называемых *Валюта.*

Число NUMERIC, десятичная дробь с 646 456, 993 цифрами и 32 767 десятичными разрядами, подходит для больших денежных сумм.

**Числа с плавающей запятой**

[FLOAT], число с плавающей запятой, позволяет указывать числа (17) и числа после запятой.

[REAL ], 4-байтовое число с плавающей запятой в диапазоне значений от + / – 3,4E38 до -1,40 E-45, также обозначаемое как Single.

[DOUBLE], 8-байтное число с плавающей запятой в диапазоне значений от +/- 1,80E308 до 4,94E-324, также обозначаемое как Double.

**Текстовые поля**

[ VARCHAR ], текст до 65 534 знаков, называется Text.

[ VARCHAR_IGNORE_CASE ], текст до 65 534 знаков, при сравнении не учитывается регистр.

[ CHAR ], текст с фиксированной длиной до 65 534 символов, ввод дополняется пробелами до значения по умолчанию.

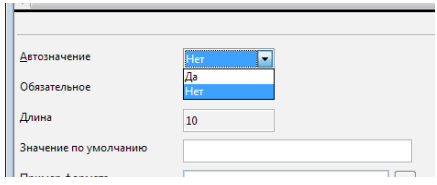
[LONGVARCHAR ], может хранить до 2 гигабайт символов, также обозначается как *Memo.*

Если количество используемых символов известно, следует использовать текстовые поля ограниченной длины.

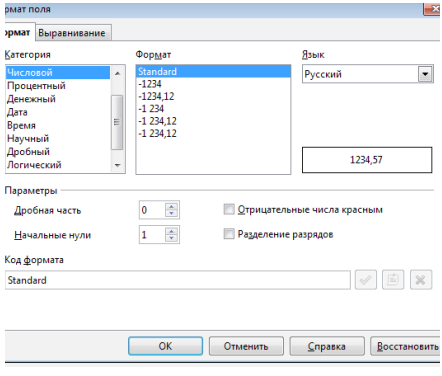
**Другие Поля**

| Поля данных | Двоичные Поля |
|---|---|
| Да / нет [ BOOLEAN ], истинные значения даты DATE, ввод даты<br>Время [TIME ], во времени дата/время [TIMESTAMP], ввод даты и времени. | Двоичные данные – это, например, изображения или звуковые данные, которые не могут обрабатываться базой данных.<br>Двоичное поле [ VARBINARY ], 2 гигабайта двоичное поле (fix) [BINARY], 2 гигабайта изображения [LONGVARBINARY ], 2 гигабайта<br>Other [OTHER ], следует использовать только для хранения объектов Java. |

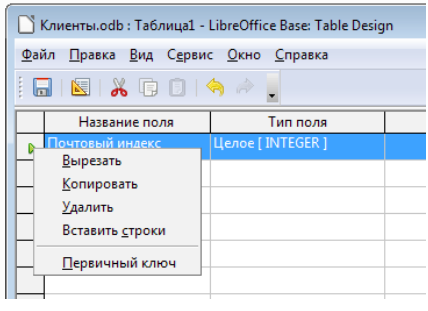
В нижней части окна вы можете определить дополнительные свойства для каждого атрибута (имя поля). Например, автоматическое значение, требуется ввод (соответствует NN = NOT NULL), максимальная длина, значение по умолчанию и форматирование могут быть указаны с помощью примера формата.



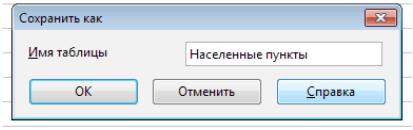

В разделе Пример формата можно выбрать форматирование готовых полей, нажав кнопку .... Предложения по формату выбираются в Категориях, предлагаются готовые форматы. Дальнейшее форматирование поля можно выбрать в разделе «Параметры». Можно создавать свои собственные форматы.

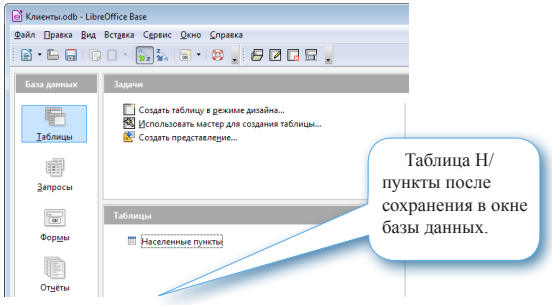



Щелкнув правой кнопкой мыши по строке поля можно выбрать свойство Первичный ключ в контекстном меню Если поле первичного ключа не указано, программное обеспечение базы данных запросит, следует ли создать новое поле в качестве первичного ключа.

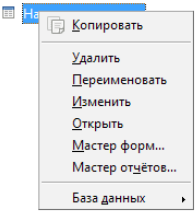


В качестве последнего шага сохраняется таблица под именем Н/пункт в базе данных.

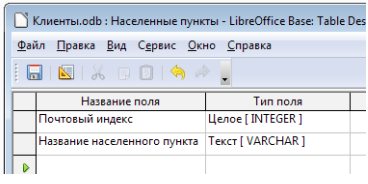


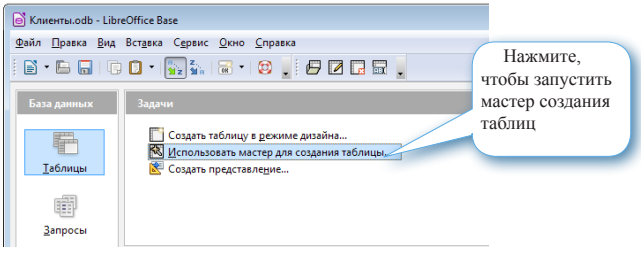

При щелчке по любой таблице правой кнопкой мыши, можно снова открыть ее в контекстном меню в разделе Правка в Конструкторе и изменить ее соответствующим образом.



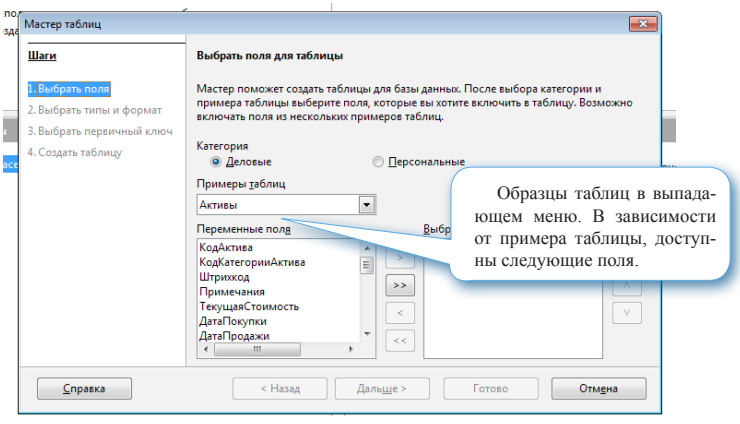

В таблице Н/пункты в качестве другого атрибута дополнительно к индексу вставляется текстовое поле с названием н/пункта. Нажмите на значок Сохранить, чтобы завершить проект.



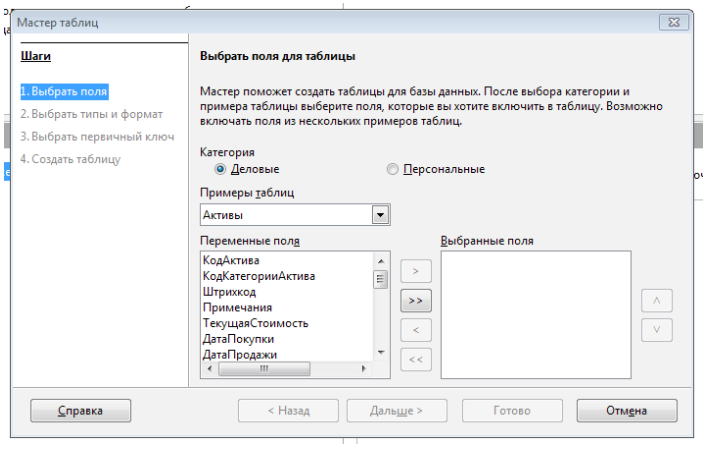

Новая таблица Клиенты теперь должна быть создана с помощью мастера создания таблиц.



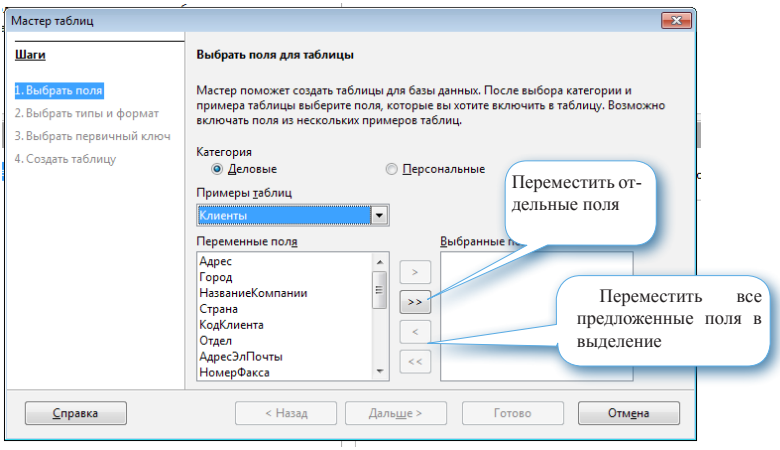
С помощью мастера создания таблица создается в четыре шага.



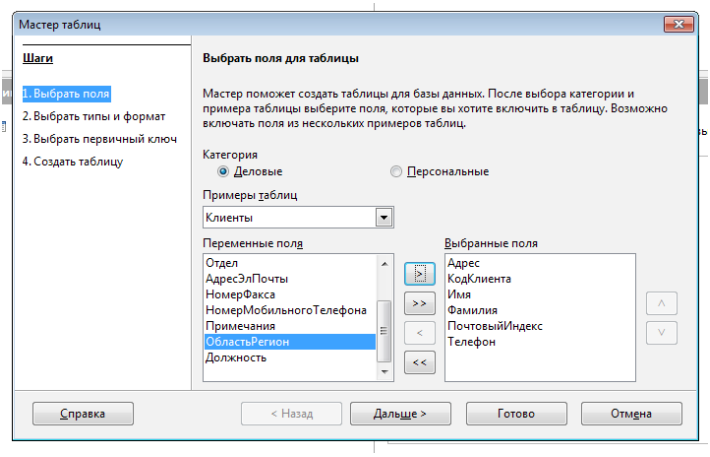

В категориях Бизнес и Частные, которые активируются кнопками выбора, доступны многочисленные шаблоны с возможными именами полей. В выпадающем меню образцов таблиц теперь выбираются клиенты.



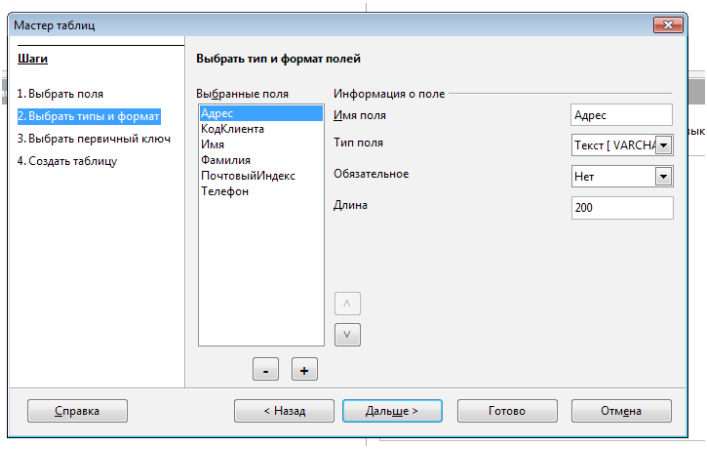

Мастер таблиц предлагает возможные поля, подходящие для выбранной таблицы. Поля выбираются двойным щелчком или щелчком, а затем нажатием кнопки со стрелкой и появляются в правом окне. Все предложенные поля можно переместить в выбранные поля с помощью клавиши двойной стрелки.



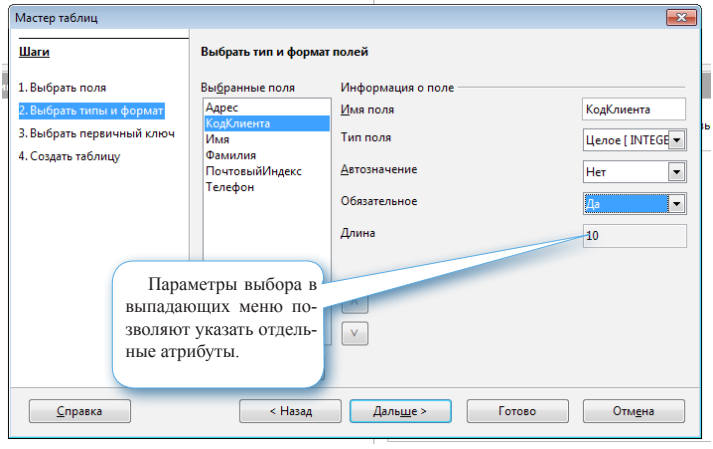

Как описано выше, последовательно выбираются поля Адрес, № клиента, Имя, Фамилия, Почтовый индекс и Номер телефона. Нажав кнопку Далее вы попадете на 2 экран выбора полей типа данных.




Теперь для каждого атрибута (имени поля) можно определить тип данных поля, требуемый ввод Y/N (соответствует NN Not Null) и длину отдельного поля.

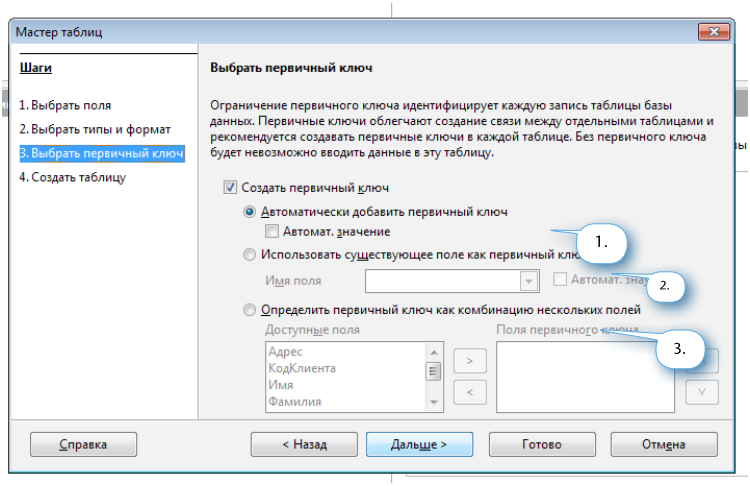



Теперь вы можете, например, установить для поля № клиента требуется значения; это также можно сделать в окне 3 установить первичные ключи.

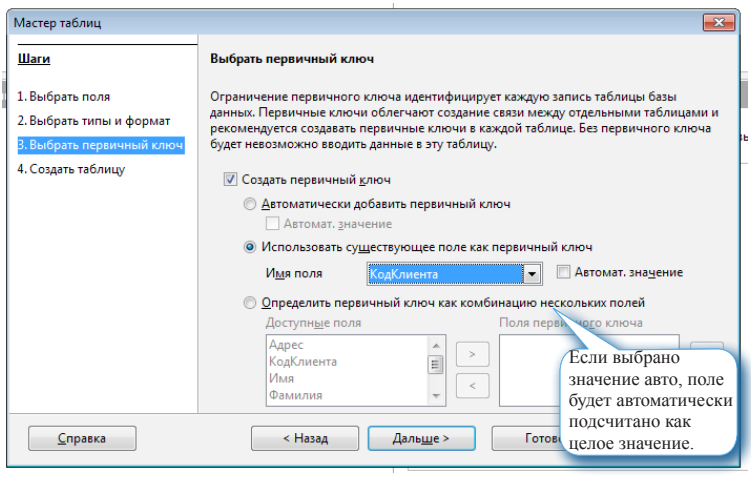



«Задать первичный ключ» устанавливает поле в качестве первичного ключа. Ключ может:

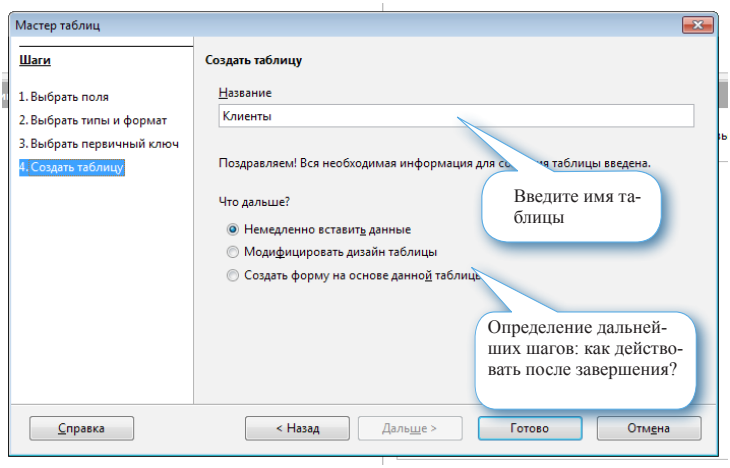
1. Быть добавлен автоматически в качестве авто-значения,
2. Быть выбран в качестве существующего поля, или
3. Создан из нескольких полей в виде составного первичного ключа.



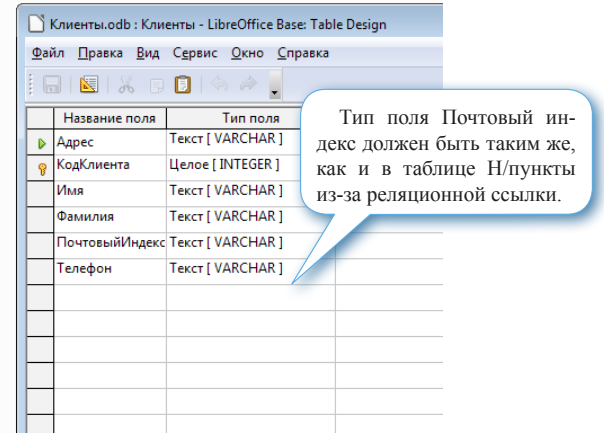

В этом примере поле № клиента устанавливается со вторым параметром (в качестве первичного ключа используется существующее поле).




По завершении работы мастера таблиц будет указано имя и дальнейшие действия.

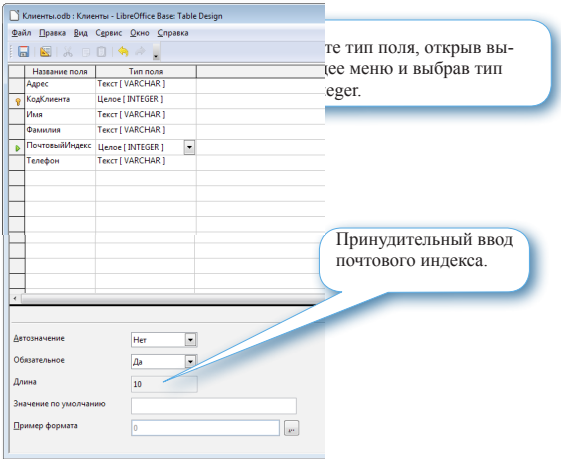



После нажатия на Завершитьбудет создана таблица Клиенты. Если открыть таблицу в Режиме редактирования, то можно заметить, что почтовый индекс должен быть целочисленным, поскольку почтовый индекс реализует реляционную связь между двумя таблицами. Изменение типа поля можно произвести в окне мастера в разделе 2 Тип данных поля.



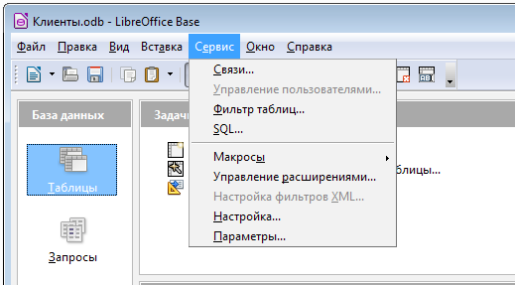

**Примечание:**
Тип поля реляционно связанных ключевых поелй (первичный ключ-внешний ключ) должен быть одинаковым. Термины «тип поля» и «тип данных поля» означают в LibreOffice одно и то же.

Имеет смысл, выбрав параметр Требуется ввод, принудительно ввести индекс.

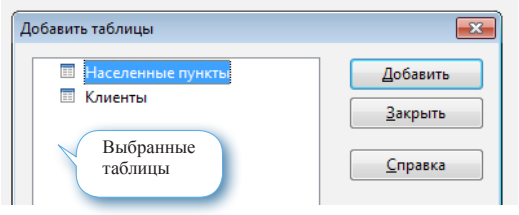



## 7.2 Создание связей между таблицами

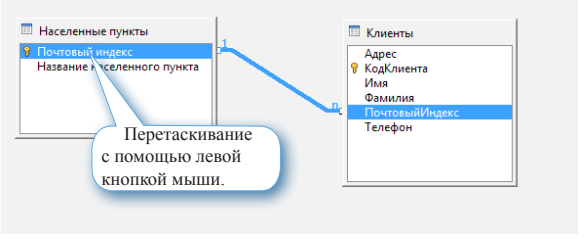
В разделе Дополнительные связи созданными таблицами.



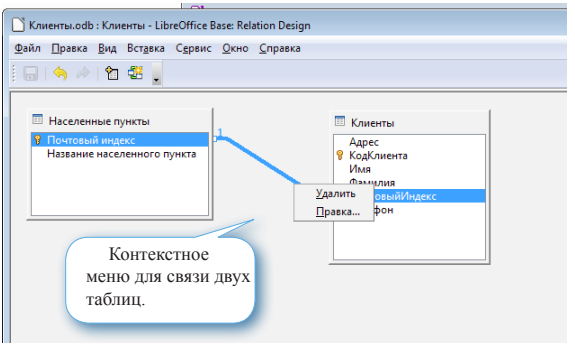

Выберите таблицы кликон на таблицы и нажатием Добавить или посредством двойного клика по нужным именам таблиц.



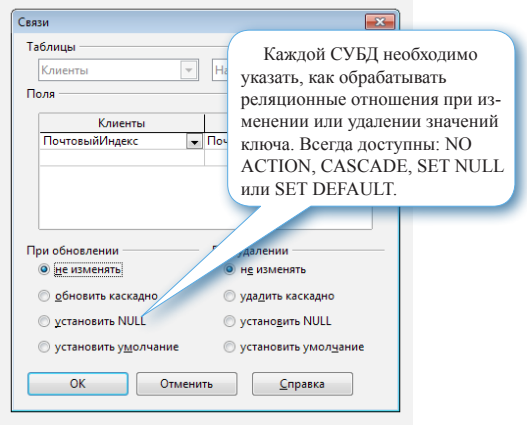

При нажатой левой кнопке мыши (перетаскивание) переместите элемент из н/пункты. индекс в клиенты.индекс и снова отпустите кнопку мыши в клиенты.индекс **Линия связи** между таблицами и **мощностью связи** (1:n) добавляется автоматически. Таблица с полем первичного ключа всегда получает мощность 1.




После клика правой кнопкой мыши по линии отношений в контекстном меню появится пункт Удалить или Редактировать.




При выборе Изменить... вы увидите задействованные таблицы и поля. Кроме того, можно задать параметры обновления (ON UPDATE) и параметры удаления (ON DELETE).

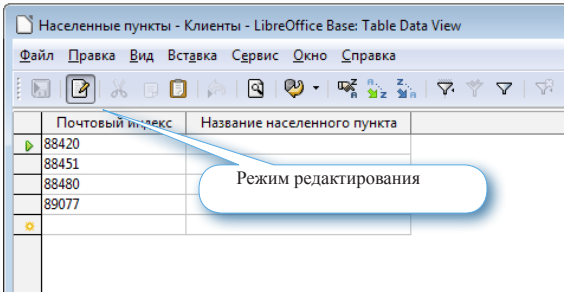



При помощи ввода срок базы данных необходимо проверить, действительно ли СУБД устанавливает ключевую связь.

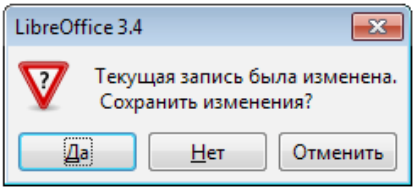
## 7.3 Запись строк базы данных

Дважды щелкните таблицу Н/пункты, чтобы открыть ее для редактирования.

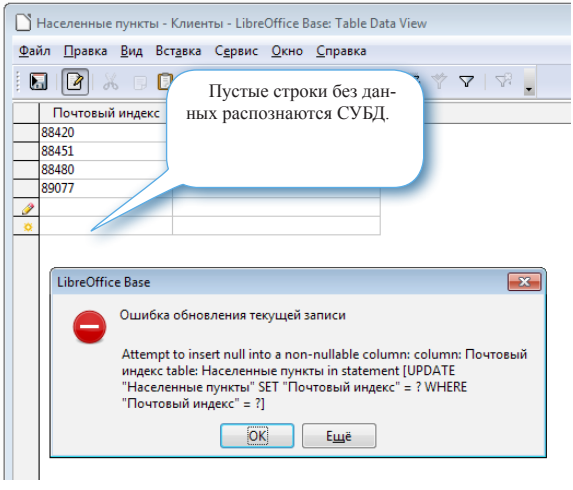
Теперь можно вводить любые записи. Неправильные записи, например, запись индекса в качестве текстовых данных перехватываются СУБД и заменяются значением по умолчанию (например, 0).




При закрытии таблицы СУБД запрашивает, следует ли сохранить данные.

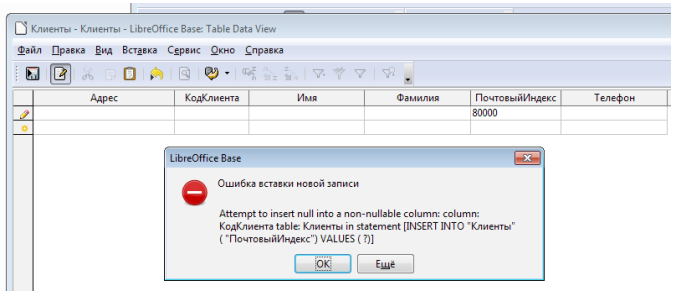



Поскольку в последнюю строку не было введено никаких данных, но поле индекс имеет свойство NN = NOT NULL, сообщение об ошибке указывает на это. Удаление пустой строки позволяет устранить ошибку и произвести сохранение.




После ввода записей данных в основную таблицу Индекс записи вводятся в дочернюю таблицу Клиенты. Однако для обеспечения ссылочной целостности могут быть назначены только те почтовые индексы, которые уже существуют в ссылочной таблице Индекс.

После ввода произвольной записи данных в таблицу Клиенты почтовый индекс, который противоречит ссылочной целостности (Integrity constraint violation), подтверждается сообщением об ошибке (**no parent FK** = foreign key).

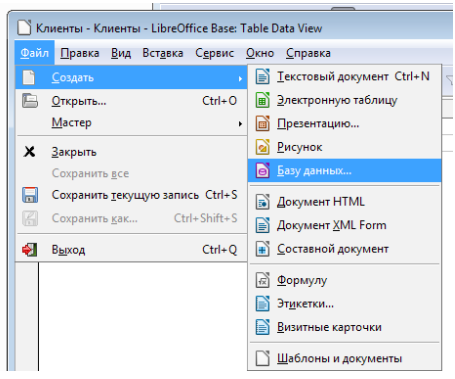



Изменение почтового индекса на реально существующий, например, 89077, позволяет сохранить данные.

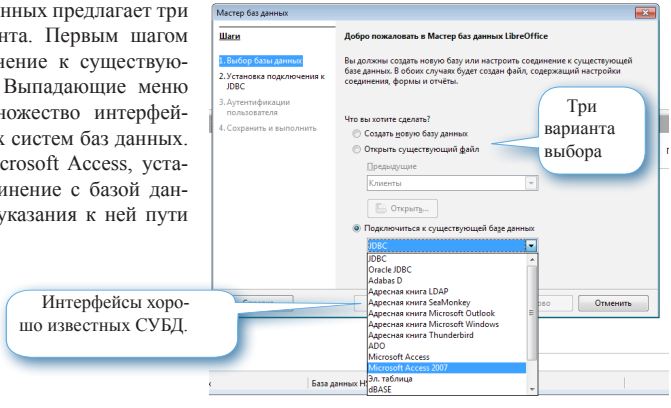
## 7.4 Подключение к другим базам данных

**Подключение к базе данных Access**

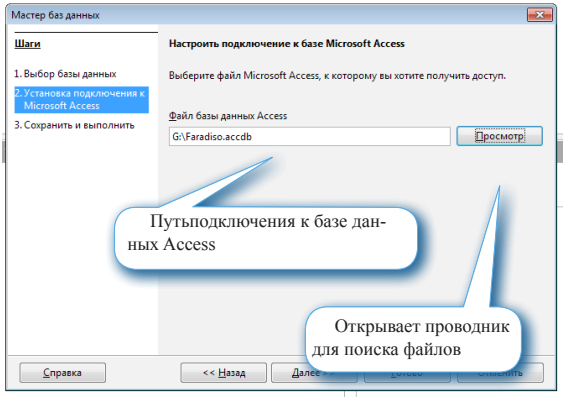
В дополнение к созданию собственных баз данных LibreOffice Base также позволяет подключаться к базам данных, созданным в сторонних приложениях. С помощью панели быстрого доступа или пункта меню Файл – Новый или кнопки Новый выбирается тип документа База данных.



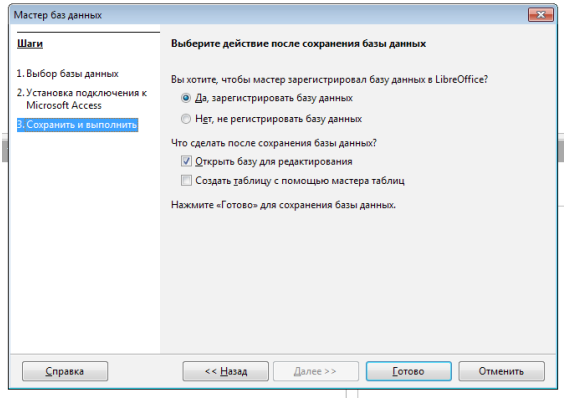

Мастер базы данных предлагает три различных варианта. Первым шагом является подключение к существующей базе данных. Выпадающие меню предоставляет множество интерфейсов для известных систем баз данных. При клике на Microsoft Access, устанавливается соединение с базой данных с помощью указания к ней пути ее хранения.

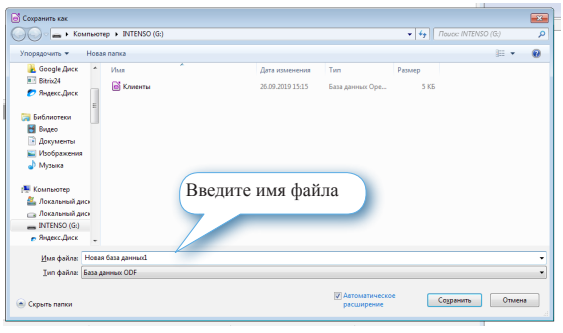


После нажатия кнопки Далее мастер базы данных запрашивает путь к базе данных Access, к которой необходимо подключиться. С помощью опции Обзор вы можете найти путь к БД, как при использовании с помощью программы Проводник на ПК.

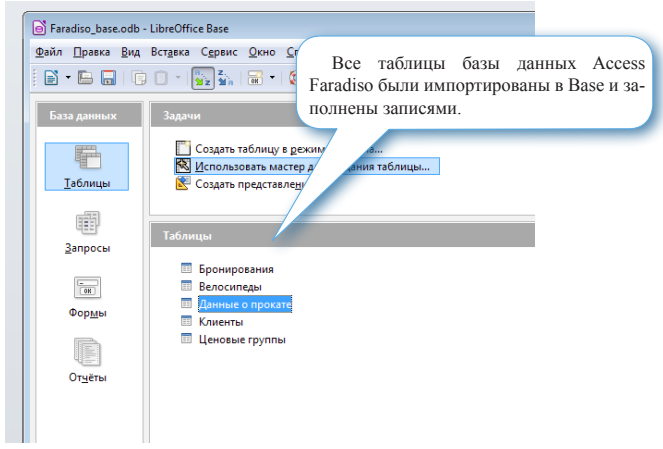


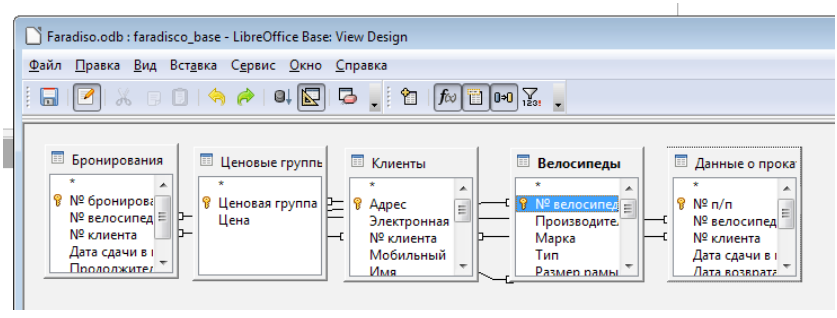

На третьем и последнем шаге вы выбираете Да, подключить базу данных. Рекомендуется не использовать мастер создания таблиц, а выбирать «Открыть базу данных для редактирование». Пустое подключение к базе данных теперь может быть сохранено в любом месте.



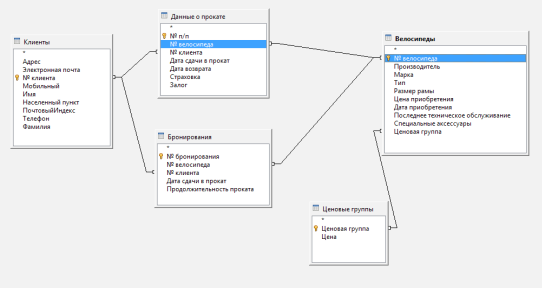

После ввода имени файла, например, faradiso_base и нажатия кнопки Сохранить Вы находитесь в окне документа базы данных.




Связи корректно переносятся с помощью импорта, например, Дополнительно -> Связи. Die Tabellen müssen hier nur noch geordnet werden, z. B. Kunden links, Fahrraeder rechts, restliche Tabel- len in die Mitte.

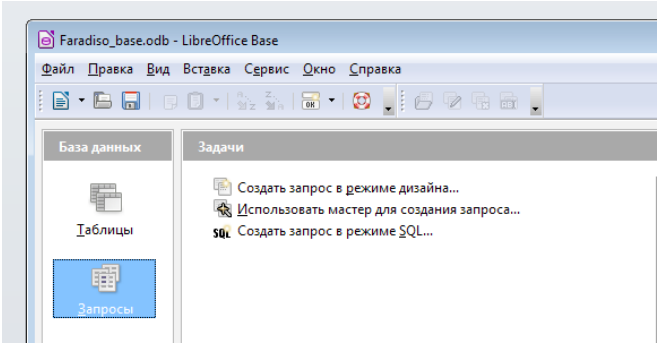
Таблицы должен быть упорядочены, напр., Клиенты слева, Велосипеды справа, остальные таблицы по центру.



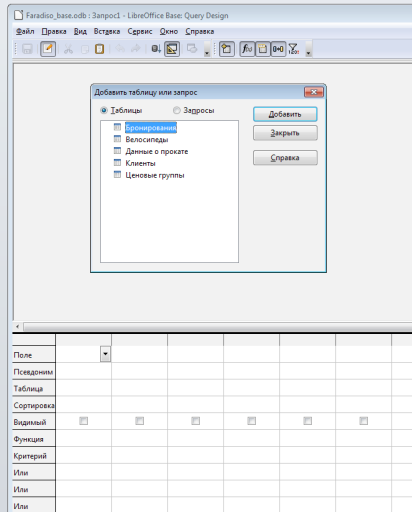
## 7.5 Создание запросов

При нажатии на Запросы открывается окно запроса. Здесь запросы можно создавать в режиме Конструктора, с помощью Мастера или непосредственно в виде SQL-кода.

Создание запроса в режиме конструктора




**В режиме конструктора запросы создаются** графически. Запросы могут выполняться как для таблиц, так и для существующих запросов. Выбрав имена таблиц и нажав кнопку Добавить, или дважды щелкнув имя таблицы, перенесите соответствующие атрибуты в нижнее окно выбора.




В окне выбора доступны следующие строки:

Псевдоним: это имя, поле которого отличается от имени поля для запроса.

Таблица: определение целевой таблицы, что особенно полезно, если, например, поле Имя существует в нескольких таблицах.

Сортировка: задает восходящие (Ascending) или нисходящие (Descending) параметры сортировки результатов запроса.

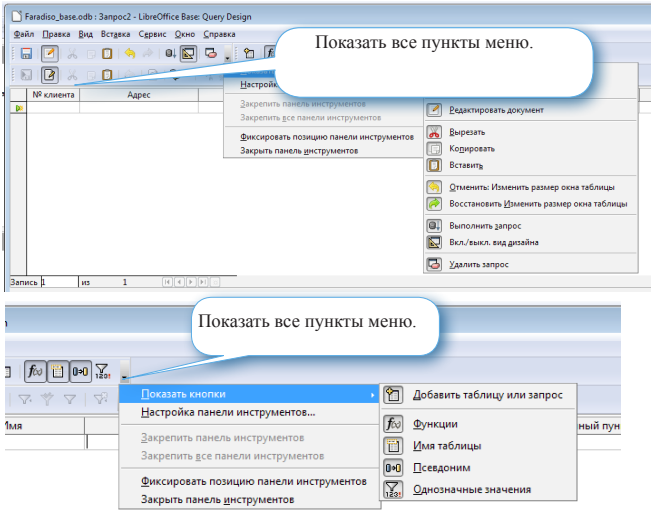
Видимый: без установленного флажка, этот запрос в результате не выводится.

Функция: предлагает функции для расчета, например, SUM (сумма).

Критерий: задает критерии фильтрации для запроса.

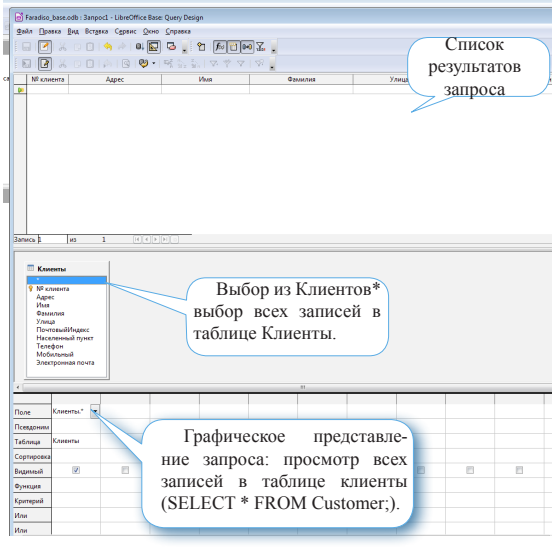
Или: позволяет осуществлять ИЛИ-вложения отдельных критериев.

Двойным щелчком или щелчком по таблице и последующим ее добавлением вы выбираете таблицы, используемые в запросе.

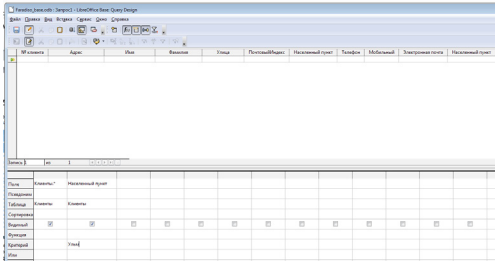



При нажатии на видимые кнопок отображаются все доступные команды.

В первом запросе должны быть представлены все данные всех клиентов. При двойном клике на звездочку Клиенты.* будут добавлены в поле ячейки . Запрос выполняется нажатием клавиши F5 или нажатием кнопки Выполнить запрос. В таблице перечислены все клиенты.



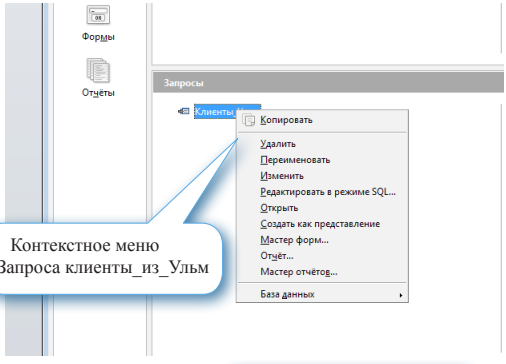
Если, например, необходимо отобразить только клиентов н/пункта Ulm (г. Ульм), то в запрос должен быть добавлен критерий.

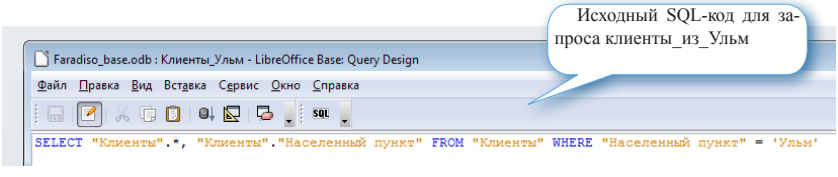


Ввод Н/пункта во втором столбце поля строки и критерия Ulm в результате запроса выдает информацию о всех клиентах из г. Ульм. Теперь этот запрос может быть сохранен и закрыт например, под названием Клиенты_ из_Ульм.

Клик правой кнопкой мыши по запросу открывает контекстное меню.

После выбора Изменить представление SQL отображается исходный код SQL-запроса.




**Формулирование условий фильтрации**

Для формулирования условий фильтрации используются различные операторы и команды. Кроме реляционных операторов существуют также команды, специфичные для SQL, для запроса содержимого полей базы данных. Эти команды создаются графически в режиме конструктора и автоматически переносятся в соответствующий синтаксис SQL. Команды SQL также могут быть введены напрямую.

В следующей таблиц представлен обзор операторов и команд:

| Оператор | Значение | Условие выполнено, если… |
|---|---|---|
| = | равно | ... содержимое поля совпадает с указанным выражением. Оператор = не отображается в полях запроса. Если ввести значение без оператора, предполагается оператор = |
| <> | не равно | ... содержимое поля не совпадает с указанным выражением |
| > | больше | ... содержимое поля больше указанного выражения |
| < | меньше | ... содержимое поля меньше указанного выражения |
| >= | больше или равно | ... содержимое поля больше или равно заданному выражению |
| <= | меньше или равно | ... содержимое поля меньше или равно заданному выражению |

| OpenOffice.org Команда | SQL-Команда | Значение | Условие выполнено, если… |
|---|---|---|---|
| ПУСТО | IS NULL | пусто | ... поле данных пусто. Для да/нет – полей с тремя состояниями эта команда запрашивает неопределенное состояние (ни да, ни нет). |
| НЕ ПУСТО | IS NOT NULL | не пусто | ... поле данных не пусто. |
| КАК (Заполнитель * для любого количества символов Заполнитель (плейсхолдер ? ровно для одного знака) | LIKE (Подстановочный знак % для любого количества символов Плейсхолдер_ для точно одного знака) | является составной частью | ... поле данных содержит указанное выражение. Подстановочные знаки (*) указывают, встречается ли выражение x в начале (x*), в конце (*x) или внутри содержимого поля (*x*). В качестве заполнителей можно вводить символ SQL% в SQL-запросах. В интерфейсе OpenOffice.org вы можете использовать обычные подстановочные знаки файловой системы (*). Подстановочный знак * или% обозначает любое количество символов. Знак вопроса (?) Используется только для одного символа в интерфейсе OpenOffice.org или подчеркивания (_) в качестве заполнителя в SQL-запросах. |
| НЕ | NOT LIKE | не является частью | … поле данных не содержит указанное выражение. |
| Между X и Y | BETWEEN x AND y | находится в интервале [x,y] | ... поле данных содержит значение, которое находится между двумя значениями x и y. |
| Не между X и Y | NOT BETWEEN x AND y | не находится в интервале [x, y] | ... поле данных содержит значение, лежащее между значениями x и y. |
| В (a; b; c…) Обратите внимание на точку с запятой в качестве разделителя во всех списках значений! | В (a, b, c…) | содержит a, b, c… | … поле данных содержит одно из указанных ... выражений a, b, c. Можно указать любое количество выражений, результат запроса определяется операцией ИЛИ. Выражения a, b. c ... могут быть как цифрами, так и символами. |
| НЕ В (a; b; c…) | NOT IN (a, b, c…) | не содержит a, b, c… | ... поле данных не содержит одно из заданных выражений a, b, c, .... |
| = ИСТИНА | = TRUE | имеет значение True | ... поле данных имеет значение True. |
| = ЛОЖЬ | = FALSE | имеет значение False | ... поле данных имеет значение False. |

**Примеры**

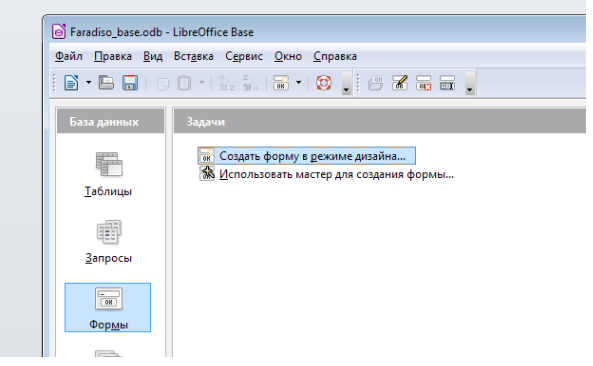
| | |
|---|---|
| =’Женщина’ | поля данных с содержимым поля «женщина». |
| КАК ‘H?llo’ | поля данных с содержимым поля, такие как «Hallo» и «Hello». |
| КАК ‘S*’ | поля данных с содержимым поля, например «SUN». |
| ОТ 10 ДО 20 | поля данных с содержимым поля между значениями 10 и 20. (Это могут быть как текстовые, так и числовые поля). |
| В (1; 3; 5; 7) | поля данных со значениями 1, 3, 5, 7. Например, если поле данных содержит номер статьи, вы можете создать запрос, который возвращает конкретные статьи с указанным номером. |
| НЕ В (‘Müller‘) | поля данных, которые не содержат «Müller». |

## 7.6 Формы

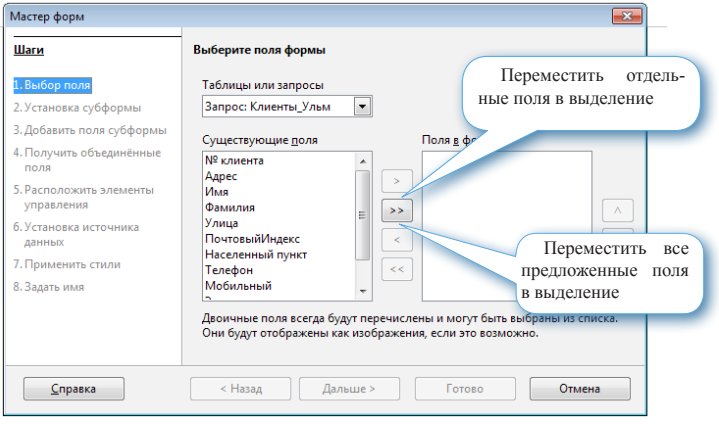
Форма – это текстовый документ со специальными характеристиками. Он может содержать, например, поля ввода, кнопки, элементы управления или списки. После нажатия на элемент меню Sotype, можно выбрать между созданием формы в режиме конструктора или с помощью Мастера.

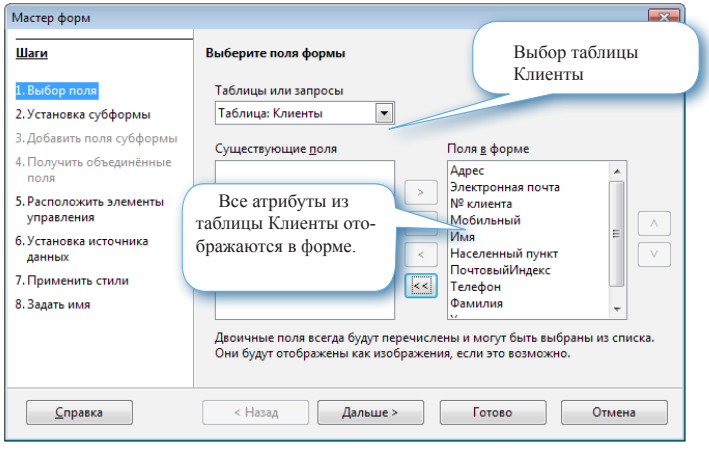
Мастер постепенно создаст форму для ввода данных клиента.

В левой части окна Базы данных доступен пункт меню Формы. После нажатия кнопки Задачи вы можете создать форму в Конструкторе» и выбрать Форму с помощью мастера создания.

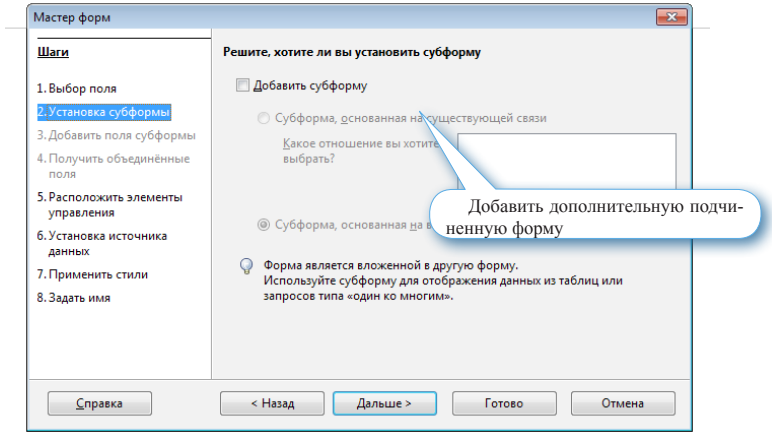



В Мастере форм сначала выбираются соответствующие таблицы или запросы, а затем поля, которые они содержат. Это можно сделать, нажав кнопку со стрелкой или кнопку с двойной стрелкой. С помощью кнопки с двойной стрелкой все атрибуты таблицы Клиенты переносятся в форму.

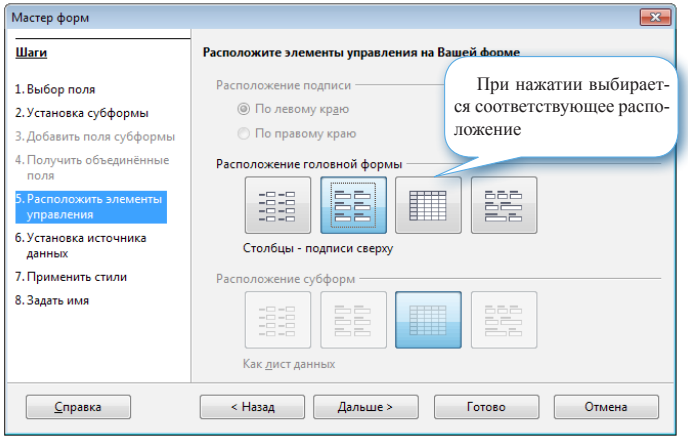




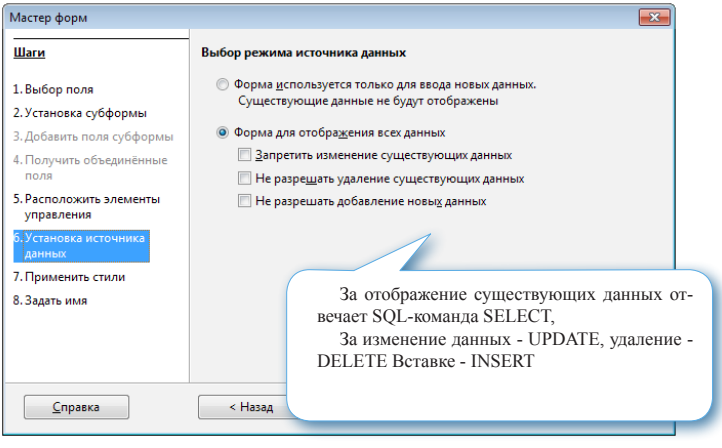

На втором шаге можно добавить необязательную подчиненную форму. С помощью подчиненной формы все связанные данные заказа могут отображаться для каждого клиента. Необходима реляционная взаимосвязь задействованных таблиц.



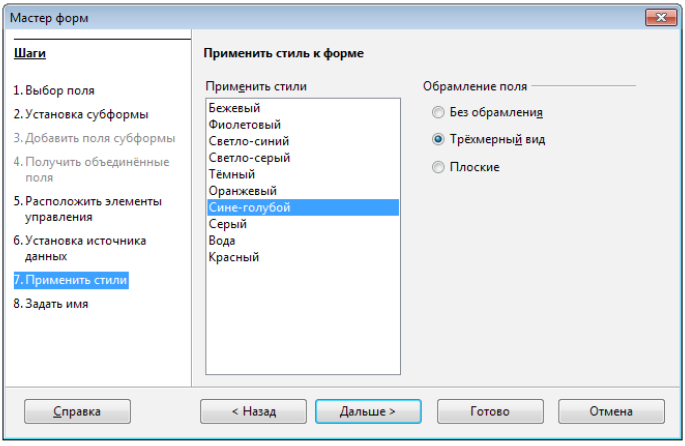

Поскольку для ввода данных о клиенте не требуется подчиненной формы, шаги 3 и 4 отображаются серым цветом и недоступны для выбора. Шаг 5 определяет порядок полей базы данных. Порядок может быть изменен позже.



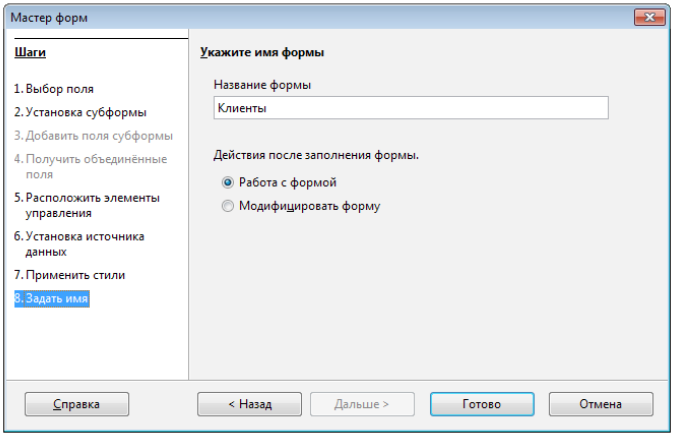

При вводе данных определяется, отображаются ли существующие данные клиента или нет. Также определяется, разрешены ли манипуляции с данными, такие как изменение, удаление существующих данных в форме. Таким способом можно решать будет ли это форма ввода для новых данных, или возможны более обширные действия

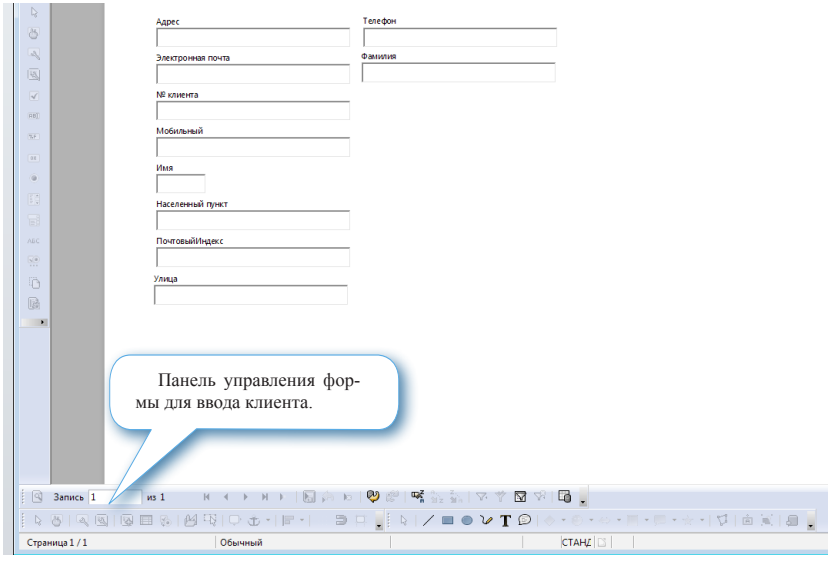



В заключение доступны различные цветовые стили и границы.




После определения имени формы она по умолчанию открывается для ввода данных.






В режиме конструктора внешний вид формы также может быть впоследствии изменен.

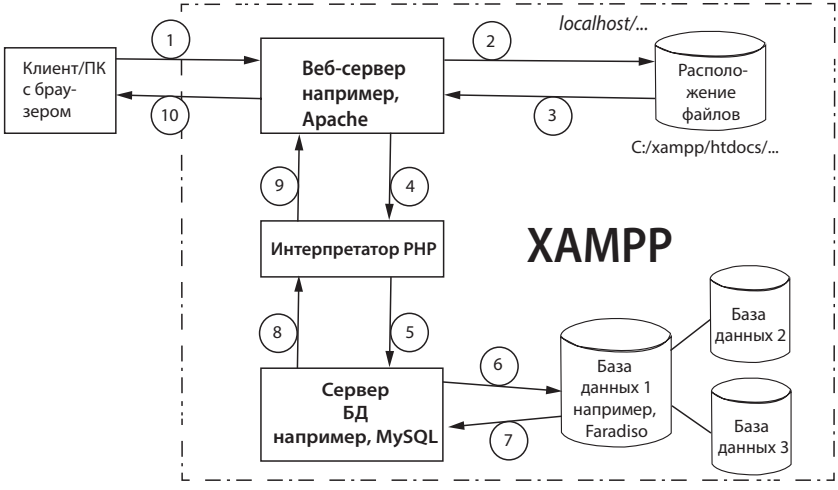
# 8 Базы данных в сети Интернет

## 8.1 Среда разработки XAMPP

Веб базы данных – это базы данных, которые подходят для размещения и использования в Интернете. Например, базы данных MySQL. Для разработки веб-базы данных необходима локальная среда разработки. Она может состоять, например, из

- **Веб-сервера Apache**, который отправляет клиенту данные, запрошенные в интернете,
- **СУБД MySQL** (**M**y **S**tructured **Q**uery **L**anguage)
- и языка программирования **PHP** (PHP Hypertext Preprocessor).

Все эти элементы собраны в среде разработки **XAMPP** (см. рисунок).




## 8.2 Работа компонентов

### 8.2.1 Веб-сервер

Веб-сервер, напр. Apache организует взаимодействие отдельных интернет-компонентов, например, браузера и сервера, на котором хранятся данные.

URL (= niform Resource Locator), запрошенный клиентом①, идентифицируется веб-сервером как адрес и извлекается в месте хранения② ③. По расширению файла веб-сер-вер распознает, является ли эта страница чистой HTML-страницей, или запрашиваются другие типы файлов, например файлы PHP.

Файлы с расширением .PHP сначала перенаправляются веб-сервером в PHP-программу для интерпретации④. Там все PHP-коды распознаются и интерпретируются. Если, напри-мер, в базе данных содержатся запросы, то они передаются в систему базы данных интер-претатором PHP⑤ Система базы данных доводит операторы SQL в базу данных (⑥ и⑦) и передает данные в интерпретатор PHP⑧ Эти результаты (например, SQL-запрос) пре-образуются в код HTML и возвращаются на веб-сервер⑨ Оттуда клиент получает чистый HTML-документ ⑩, который отображается на экране веб-браузера, напр. Mozilla Firefox.

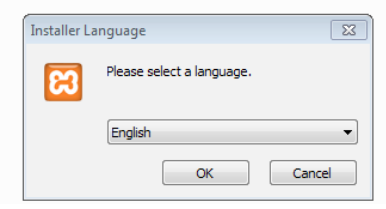
Веб-сервер, программы-интерпретаторы и серверы баз данных не обязательно должны находиться на одном ПК, а могут располагаться по всему миру на различных серверах в Интернете.

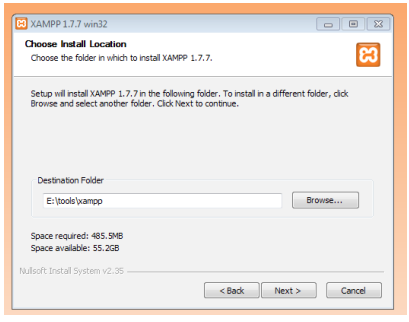
### 8.2.2 Установка среды разработки XAMPP

После загрузки бесплатной среды разработки (например, с www.apachefriends.org/ru/xampp-windows.html) запустите файл XAMPP xampp-win32-1.x.x-VCxx-installer.exe.

Выберите язык с помощью выпадающего меню:

Целевой диск установки указывается в окне Выбрать целевую папку (каталог).

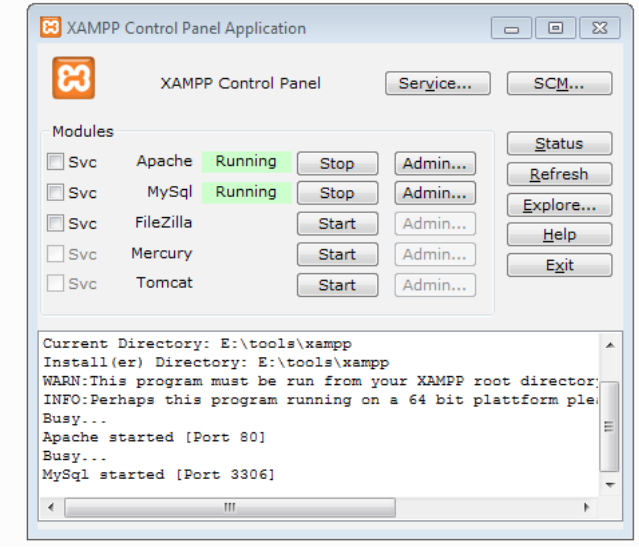



Чтобы установить путь к этому каталогу, дважды щелкните файл setup_xampp.bat для его запуска.

### 8.2.3 Запуск компонентов

Файл xampp-control.exe запускает среду разработки XAMPP, и открывается панель управления XAMPP. Нажатие кнопки Start запускает веб-сервер Apache, а также сервер базы данных MySQL.



## 8.3 Язык сценариев PHP

### 8.3.1 Введение

PHP – это язык сценариев (от scriptum = document), который может быть встроен в код HTML. Язык сценариев не компилируется заранее, сценарий интерпретируется и выполняется во время исполнения. Однако, в отличие от JavaScript, веб-сервер не передает этот код клиенту, а выполняет его на стороне сервера.

Клиент получает результат программы в виде HTML-файла, который не содержит исходный код PHP.

**Примечание:**
PHP предотвращает кражу кода, не передавая исходный код (скрипт) клиенту, а интерпретируя его на сервере.

В то время как HTML-браузер клиента должен быть подходящим и установленным для обработки страниц с помощью Java-кода, PHP не предъявляет никаких требований к браузеру.

В среду разработки XAMPP включен интерпретатор PHP. Чтобы запустить исходный код PHP в тестовой среде XAMPP,

- сценарии должны храниться в подкаталоге C:\...\xampp\htdocs с расширением .php
- должен быть запущен веб-сервер и
- вызов скрипта осуществляться должен через веб-сервер с IP 127.0.0.1/... или URL localhost/...

**Примечание:**
Если файл test.php читается непосредственно из браузера по адресу памяти, например, C:\xampp\ht- docs ..., то интерпретатор PHP игнорируется, а исходный код PHP отображается непосредственно в браузере. Доступ к базе данных не представляется.

### 8.3.2 Написание сценария PHP

Языковые элементы PHP в основном происходят от языков программирования C, Java и Perl.

**Пример**
Скрипт должен отображать содержимое текстовой переменной, например, «Привет, эксперты по базам данных!».

```
<html>
   <head>
          <title>Упражнение 1</title>
   </head>
   <body>

          <?php
          $Data = "Привет, эксперты по базам данных!"
          //Объявление переменной
          echo $daten;          // Вывод переменной
          ?>


   </body>
</html>
```

HTML-Code

PHP-Skript

HTML-Code

**Принцип:**

С помощью текстового или HTML-редактора вы создаете основу HTML-файла.

PHP-скрипт устанавливается между тегами (= маркеры) <?php и ?> в тело HTML-скрипта.

Каждая строка инструкции должна закрываться точкой с запятой. Комментарии начинаются с символов //.

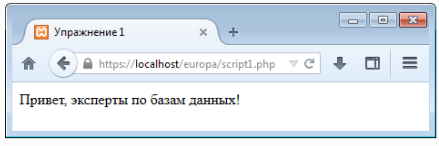
**Примечание:**

Скрипты PHP всегда должны создаваться и тестироваться постепенно, чтобы своевременно обнаружить, локализовать и исправить ошибки.

### 8.3.3 Переменные в PHP

Переменные в PHP обозначаются префиксом $, напр. `$daten`. Они не должны объявляться явно, но им присваивается соответствующий формат данных посредством присвоения значения. Вывод переменных $daten на экран осуществляется при помощи оператора print $daten или echo $daten.

Файл со сценарием PHP должен быть сохранен с расширением .php, например, script1.php, в каталоге публикации веб-сервера. Вызов выполняется не в автономном режиме в браузере, путем открытия файла по пути `C:\...\xampp\htdocs...`, а в онлайн-режиме. Например, по адресу `http://localhost/script1.php`.




Для интерпретации сценариев PHP на страницах HTML файлы должны быть сохранены, с расширением .php и откаты через веб-сервер.

### 8.3.4 Массивы (Arrays)

Массив (Arrays) (= поле) состоит из определенного числа элементов, которые не обязательно должны иметь один тип данных. Так, например, можно объединить поля, текстовые поля и объекты изображений в массив.

Массивы могут состоять из элементов поля с разными типами данных.

Вот почему массивы PHP особенно подходят для обработки записей в базах данных.

Массив обладает, как и другие переменные, именем, напр. $Клиенты. Отдельные элементы поля отличаются непрерывным индексом в числовых массивах, например.

$Клиенты[1] = «Хубер». Индекс в PHP начинается по умолчанию с[0].

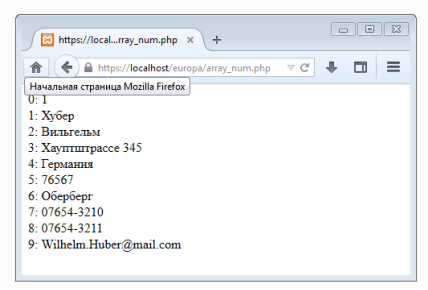
**Пример**

С помощью скрипта создается числовой массив, содержащий следующие данные о клиенте Хубер:

№: 1;
Фамилия: Хубер;
Имя: Вильгельм;
Адрес: Хауптштрасе 345;
Страна: Германия;
Почтовый индекс: 76567;
город: Оберберг;
Тел.: 07654-3210; Факс: 07654-3211;
E-Mail: Wilhelm.Huber@mail.com;

```
<?php
$клиенты[ 0] = 1;
$клиенты[ 1] = "Хубер";
$клиенты[ 2] = "Вильгельм";
$клиенты[ 3] = "Хауптштрассе 345";
$клиенты[ 4] = "Германия";
$клиенты[ 5] = "76567";
$клиенты[ 6] = "Оберберг";
$клиент[ 7] = "07654-3210";
$клиент[ 8] = "07654-3211";
$клиенты[ 9] = "Wilhelm.Huber@mail.com";
?>
```

**Вывод массива на экран:**




**Ассоциативные массивы**

Ассоциативные (связанные с представлениями) массивы используют в качестве индекса своих элементов не числа, а строки или слова, называемые ключами. Вместо $клиенты[0] и $клиенты[1] элементы массива получают имя$клиенты[Nr] и $клиенты[Name].

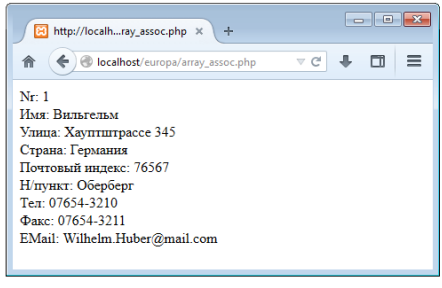
При индексировании с помощью этого ключа, например, [Nr] или [Name], значение отдельного элемента передается легче.

**Пример**

С помощью PHP-скрипта данные клиента Хубер хранятся и выводятся в ассоциированном массиве.

```
<?php
    $клиент[ Nr] = 1;
    $клиенты[ имя] = «Хубер«;
    $клиенты[ имя] = «Вильгельм«;
    $клиенты[ улица] = «Хауптштрассе 345«;
    $клиенты[ страна] = «Германия«;
    $клиенты[ почтовый индекс] = «76567«;
    $клиенты[ н/пункт] = «Оберберг«;
    $клиенты[ тел] = «07654-3210«;
    $клиент[ факс] = «07654-3211«;
    $клиенты[ EMail] = «Wilhelm.Huber@mail.com«;

foreach(Клиент as $key => $element)
    echo $key,': ', $element,'<br>';
?>
```

Вывод на экран:



В этом примере содержимое поля выводится через цикл foreach. Общий синтаксис этого цикла (для ассоциативного массива):

foreach ($arrayname as $keyname => $elementname) инструкция;

Для числовых массивов спецификация ключа опускается. Общий синтаксис цикла:

```
foreach ($arrayname as $elementname)
   {
   Инструкция 1;
   Инструкция 2;
   Инструкция 3
}
```

Вместо отдельных инструкций, несколько фигур в скобках можно объединить в один блок.

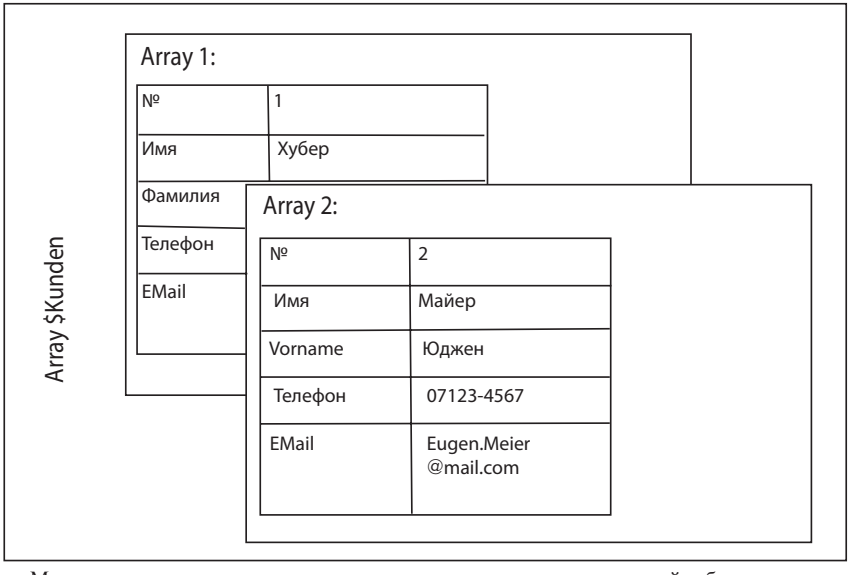
Чтобы упорядочить ассоциативные массивы, перед именем массива следует указать ключевое слово array. Ключи отдельных элементов указываются в кавычках, например "Имя", и после оператора присваивания => заполняются значениями:

```
$клиены=array (
  "№" = > 1,
  "Name"=>"Хубер",
  "Name"=> "Вильгельм",
  ...
  "EMail"=>"Wilhelm.Huber@mail.com");
```

**Многомерные массивы**

Простые числовые или ассоциативные массивы могут быть расширены по желанию. Однако, чтобы сохранить обзор нескольких клиентов, рекомендуется создать новый массив для каждого клиента. Массивы отдельных клиентов (соответствующие записям данных базы данных) объединяются в массив более высокого уровня.

Принцип многомерного массива можно представить следующим образом:




Многомерные массивы предназначены для хранения содержимого всей таблицы реляционной базы данных.

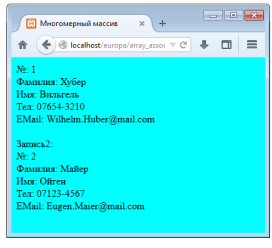
**Пример**

PHP-скрипт сохраняет две записи с двумя адресами и выводит их на экран изображения.

```
<html>
   <head>
          <title>многомерный массив</title>
   </head>
   <body text=#000000     bgcolor=#00FFFF
          <?php
                 $клиент[ 1]  = array(
                    «№«=> 1,
                    «Фамилия«=> «Хубер»,
                    «Имя«=> «Вильгель»,
                    «Тел«=> «07654-3210»,
                    «EMail«=> «Wilhelm.Huber@mail.com«
                    );
                 $клиент[ 2]  = array( «№«=> 2,
                    «Фамилия«=> «Майер»,
                    «Имя«=> «Ойген»,
                    «Тел«=> «07123-4567»,
                    «EMail«=> «Eugen.Maier@mail.com«
                    );
                 foreach ($клиент как $счетчик =>
$запись)
                 {  echo 'запись', $счетчик,':','<br>';
                 foreach($запись as $индекс=>$содержимое поля)
          {  echo $index,':', $содержимое поля,'<br>';
                    }
                 echo '<br>';}
          ?>
   </body>
</html>
```

**Пояснение:**

После указания обоих массивов `$клиент [ 1]` и `$клиент [ 2]`, данные выводятся через цикл foreach с двойным вложениием. Во внешнем цикле внутренний счетчик $счетчик сначала устанавливается на первый элемент. Этот первый дочерний массив $клиент[1] теперь передается переменной $запись. После вывода «записи1:» начинается внутренний цикл. Здесь запускается счетчик $index и первый элемент активного массива $запись переносится в переменную $содержимое поля. С помощью echo... индекс и содержимое поля отображаются на экране:




Если числитель $index прошел через все элементы массива набора данных, то счетчик цикла $счетчик верхнего уровня увеличивается на 1, и процесс повторяется со следующим подмассивом $клиент[2].

### 8.3.5 Работа с массивами

PHP имеет специальные функции для обработки массивов:

| Функция | Пример | Эффект |
|---|---|---|
| array_walk | $результат = array_walk ($числа, 'квадрат'); | Изменяет элементы массива, применяя пользовательскую функцию (например, ‚квадрат‘)) к каждому элементу. |
| key | $ключ = ключ($array); | Возвращает ключ текущего местоположения указателя массива. |
| array_pop | array_pop ($клиенты); | Удаляет последний элемент массива. |
| array_push | array_push ($клиенты, $elem); | Добавляет $elem в массив в качестве другого элемента. |

**Пример**

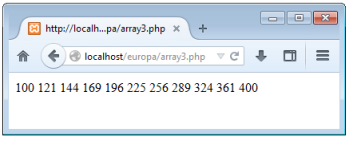
В массиве хранятся числа от 10 до 20. Затем элементы этого массива возводятся в квадрат с помощью пользовательской функции, и результаты отображаются на экране.

```
<?
   function квадрат (&$x){
          $x=$x*$x;
   }
   $числа = массив (10,11,12,13,14,15,16,17,18,19,20);
   $результат = array_walk ($числа, 'quadrat');
   foreach ($числа as $element)  {
          echo $element,'        ' ;
   }
?>
```

Сначала определяется функция возведения в квадрат элементов.

После установки исходного массива $числа оператор array_walk ($числа, 'square') возводит в квадрат элементы массива и записывает их в их исходные местоположения. Вывод осуществляется через инструкции в цикле foreach.

Результат выглядит следующим образом:




### 8.3.2 Редактирование символьных строк

PHP предоставляет различные функции для редактирования символьных строк (см. таблицу). Например, можно управлять регистрами в базах данных:

| Функция | Пример | Описание |
|---|---|---|
| strtolower() | $name = strtolower ($name); | Преобразует текст в нижний регистр. |
| strtoupper() | $name = strtoupper ($name); | Преобразует текст в верхний регистр. |
| ucfirst() | $name = ucfirst ($name); | Преобразует первый символ в верхний регистр. |
| ucwords() | $str1 = strtolower ($str); | Преобразует первый символ каждого слова в верхний регистр. |
| ord() | $значение = ord («A») | Возвращает значение ASCII символа. |

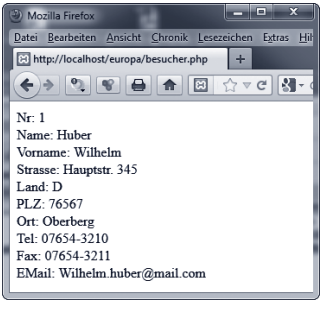
**Пример**

Независимо от регистра, данные клиента Хубера хранятся в массиве. При выводе первая буква текстовых элементов должна быть заглавной, все остальные – строчными.

```
<?
    $клиент = array( «№«=> 1,
       «Фамилия«=> «Хубер»,
       «Имя«=> «Вильгельм»,
       «Улица«=> «Хауптштрассе 345»,
        «Страна«=> «Германия»,
       «Почтовый индекс» = > » 76567»,
       «Н/пункт«=> «Оберберг»,
       «Тел«=> «07654-3210»,
       «Факс» = > » 07654-3211»,
       «EMail«=> «Wilhelm.Huber@mail.com«
       );
    foreach ($клиент как $счетчик = > $содержимое поля) {
       $ содержимое поля = strtolower ($содержимое поля);
       $содержимое поля = ucfirst($содержимое поля);
       echo $счетчик,':    ', $содержимое поля,'<br>';
       }
    ?>
```

В цикле foreach функция strtolower ($содержимое поля) преобразует содержимое переменной в нижний регистр и снова сохраняет его под тем же именем. Числовые переменные не обрабатываются.

Функция ucfirst() преобразует первую букву в заглавную, а затем выводит содержимое поля на экран:




### 8.3.7 Операции с файлами с помощью PHP

Есть несколько функций, доступных для файловых операций:

| Функция | Пример | Значение |
|---|---|---|
| fopen() | $fp =fopen(‘имя файла‘, $режим); | Открывает файл определенного типа доступа |
| fclose() | fclose($fp); | Закрывает открытый файл |
| fgets() | fgets($fp,12); | Читает строку с максимальной длиной 12 из файла. |
| fgetc() | fgetc($fp); | Считывает один символ файла. |
| fwrite() | fwrite($fp,$счетчик); | Записывает символьную строку $str в файл. |
| file_exists() | if(!file_exists($файл счетчика) ...; | Проверяет, существует ли файл. |

| Функция | Пример | Значение |
|---|---|---|
| rewind() | rewind($fp); | Устанавливает указатель на начало файла. |
| unlink() | unlink('filename'); | Удаляет файл. |
| require() | require($файл счетчика); | Вставляет файл в скрипт PHP. |

**Пример**

Для подсчета посетителей домашней страницы запрограммирован счетчик, который сохраняет статус подсчета в текстовом файле counter.txt и выводит его на экран.

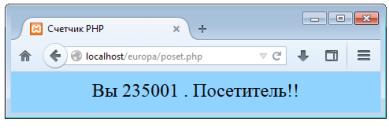
```
<html>
<head>
   <title>счетчик PHP</title>
</head>
<body bgcolor=«#87ceff«>
   <?php
          $zaehldatei=«counter.txt«;
                   if( file_exists($файл счетчика))     {
                  $fp=fopen(файл счетчика,«w+»);
                  fwrite($fp,«0»);
                  fclose($fp);
                  }
          $fp=fopen($zaehldatei,«r+»);
          $zaehler=(int)fgets($fp,12);
          $zaehler++; rewind($fp);
          fwrite($fp,$zaehler);
          fclose($fp);
   ?>
<div align=«Center«>
    <font size="+2">
<? require($файл счетчика); ?>
.          Посетитель  </font></div>
</body>
</html>
```

В файле counter.txt в качестве целого числа хранится только показание счетчика. Имя этого файла указывается в переменной.

Если этот файл не существует (еще не существует), выполняется оператор if. Этот файл создается и открывается в режиме w+. Этот режим работы файла определяет, что он будет открыт для чтения и записи и, если он не существует, то сначала будет создан. Функция fwrite () записывает число 0 в файл. fclose () закрывает файл.

Функция fopen() открывает файл, который теперь существует в режиме r+ для чтения и записи. С помощью fgets() максимальное количество из 12 символов считывается как строка и преобразуется в целое число с помощью int(). Он кэшируется в переменной $zaehler и $ zaehler++ увеличивается на один. Функция rewind() устанавливает внутренний указатель на начало файла. Там число увеличивается на единицу с помощью fwrite(), и файл снова закрывается.

Выходные данные представлены в формате HTML, при этом показания счетчика отображаются путем интеграции файла с помощью функции require ($count).



Файлы могут быть открыты в разных режимах с помощью функции `fopen()`. Режим a открывает файл и помещает внутренний указатель файла в конец файла. Там разрешены только операции записи. В режиме a+ это ограничение не существует, чтение возможно. В обоих режимах выбранный файл создается, если он еще не существует.

Обзор возможных режимов работы с файлами:

| Режим работы | Пояснение |
|---|---|
| "r" | Открывает файл только для чтения. |
| "r+" | Открывает файл для чтения и записи. |
| "w" | Открывает файл только для записи. |
| "w+" | Открывает файл для чтения и записи. Сначала удаляется содержание. Если файл не существует, то он будет создан. |
| "a" | Открывает файл только для записи и помещает указатель файла в конец файла Если файл не существует, то он будет создан. |
| "a+" | Открывает файл для чтения и записи и помещает указатель файла в конец файла. |

### 8.3.8 Права доступа к файлам

Важной информацией при работе с файлами, особенно с базами данных, являются права доступа. Вы узнаете о разрешениях различных пользователей, связанных с этими файлами.

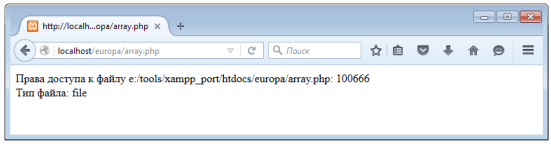
PHP предоставляет функции, подходящие для управления файлами и правами доступа:

| Функция | Описание |
|---|---|
| filegroup(); | Определяет группу владельца файла. |
| fileowner(); | Определяет владельца файла. |
| fileperms(); | Определяет права доступа к файлу. |
| filesize(); | Определяет размер файла. |
| filetype(); | Определяет тип файла. |
| decoct(); | Преобразует десятичное число в восьмеричное. |

**Пример**

Права доступа и тип файла, например, для файла `D:\tools\xampp_port\htdocs\europa\array.php`, определяются с помощью сценария PHP.

```
<?php
      $файл = «d:/tools/xampp_port/htdocs/europa/array.php«;
      $erg=fileperms ($файл);
      echo «Права доступа к файлу $datei: ", decoct($erg),"<br>";
      echo «тип файла:    ", filetype($файл);
?>
```




После определения пути к файлу в переменной функция fileperms() определяет права доступа к этому файлу. Поскольку результат находится в десятичной записи, но может правильно интерпретироваться только в восьмеричной записи, его необходимо преобразовать с помощью функции decoct(). Первые три цифры (здесь 100) информируют о типе файла. Функция filetype() выводит тип файла в виде текстовой переменной.

Возможные типы файлов:

- block: заблокировать специальное устройство
- char: символьное специальное устройство
- dir: директория
- fifo: FIFO (именованный канал)
- file: обычный файл
- link: символическая ссылка
- unknown: неизвестный тип файла

Последовательность цифр 666 информирует о правах владельца, группы и других пользователей. Первая цифра 6 означает, что владелец файла может прочитать (4), записать (2), но не исполнять. Вторая цифра 6 информирует о равных правах группы, и последняя цифра – о правах всех остальных пользователей.

Возможные права каждого пользователя:

| **Владелец** | | | **Группа** | | | **Другой пользователь** | | |
|---|---|---|---|---|---|---|---|---|
| r | w | x | r | w | x | r | w | x |
| 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 |
| r – read = чтение, w от write = запись,<br>x – execute = выполнение | | | | | | | | |

**Пример**

Для файла в качестве ключа для прав доступа используется число 754.
Индивидуальные права и полномочия определяются следующим образом:
7 = владелец: чтение (4) + запись (2) + запуск (1);
5 = группа: чтение (4) + запуск (1);
4 = другие: чтение (4).

### 8.3.9 Работа с формами

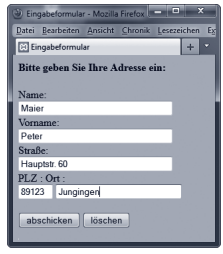
PHP-скрипты позволяют обрабатывать данные из HTML-форм.

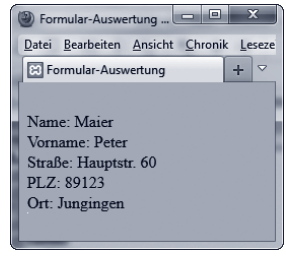
**Пример**

В HTML-форме вводится адрес (фамилия, имя, улица, почтовый индекс, город). Командная кнопка используется для отправки данных в скрипт PHP и отображения их там.

Код формы ввода выглядит следующим образом:

```
<html>
    <head><title>форма ввода</title></head>
    <body text=«#000000» bgcolor=«#00E0FF«>
      <h4>пожалуйста, введите свой адрес:</h4>
      <form action=«form_aus.php» method=«POST«>
         name:<br><input type=«Text» name=«Фамилия» size=«35«>
         <br> name:<br><input type=«Text» name=«Имя» size=«35«><br>
         Улица:<br><input type=«Text» name=«Улица«      size= «35«>
    <br>
    Почтовый индекс: н/пункт :<br><input type=«Text» name=«Индекс»
size= «5» >
                                          < input type= «Text» name=
«н/пункт» size= " 28 " ><br><br>
       <input type="submit" name=«submit» value=«отправить«>
       <input type="reset"    name=loeschen value=удалить>
            </form>
         </body>
</html>
```




Сначала создается форма в HTML-коде. С помощью метода POST данные формы будут отправлены в скрипт PHP, например, form_aus.php. Это делается с помощью инструкции HTML

<form action=«form_aus.php» method=«post«>.

При нажатии кнопки Отправить данные передаются в файл form_aus.php

.

Пример кода для файла form_aus.php :

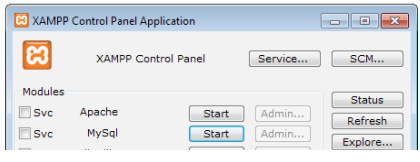
```
<html><head>
      <title>оценка формы</title></head>
<body text=«#000000» bgcolor=«#00E0FF«>
        <?php
      $фамилия        =$_POST[ «Имя»] ;
      $имя            =$_POST[ «Фамилия»] ;
      $улица          =$_POST [ «улица»] ;
      $почтовый индекс       =$_POST [ «почтовый индекс»] ;
      $н/пункт               =$_POST[ «н/пункт»] ;
      echo "<br>Name:        ", $фамилия;
      echo "<br>             имя: ",
      $имя; echo             "<br>улица: ",
      $улица; echo           "<br>индекс: ", $индекс;
      echo "<br>н/пункт:     ", $н/пункт;
        ?>
</body > </html>
```

Стандартная переменная $_POST [] передает данные в формате массива. Элементы считываются с помощью, например, $Фамилия=$_POST[«Фамилия»] и присваиваются локальной переменной. Они могут быть отображены с помощью команды echo...

## 8.4 Система управления базами данных MySQL

MySQL – это реляционная система баз данных. Используется архитектура клиент-сервер, где сервер базы данных mysqld.exe работает как фоновый процесс на сервере. Доступ к серверу может получать неограниченное число клиентов. Концепция многопоточности позволяет обрабатывать различные запросы к серверу.

Дистрибутив MySQL включен в бесплатный инструмент разработки баз данных XAMPP. Запустить сервер можно, например, через XAMPP Control Panel, нажав кнопку Пуск рядом с записью MySql.

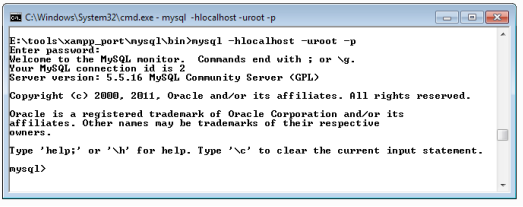


### 8.4.1 Работа с клиентами MySQL работает

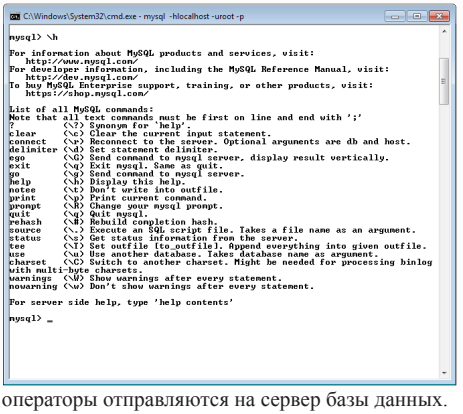
Клиентская часть mysql

Клиентская программа mysql (пишется строчными буквами, в отличие от системы баз данных MySQL) связывается с сервером MySQL и позволяет отправлять команды на сервер.

Клиент mysql запускается из командной строки Windows в каталоге установки XAMPP, имя файла mysql. Параметры подключения указываются при обращении к серверу. После атрибута -h указывается расположение сервера базы данных, например: Localhost, после -u расположение пользователя, например, root . Пароль следует за ключом –p. По умолчанию для пользователя root пароль не установлен. Чтобы повысить безопасность, имеет смысл установить пароль.

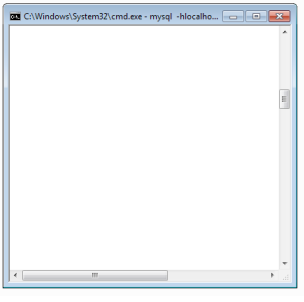



После приветственного сообщения в Клиент могут вводиться команды. Команды всегда заканчиваются символом ';'. Вызов справки осуществляется с помощью команды \h.




По команде mysql операторы отправляются на сервер базы данных.

Например, команда show database; выводит все базы данных на сервер. На картинке это 11 баз данных.

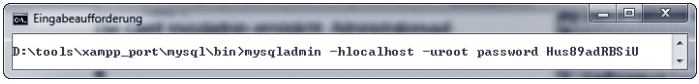


**Клиент mysqladmin**

Клиент mysqladmin позволяет выполнять задачи администрирования на сервере баз данных.

**Пример**

Для пользователя root установлен пароль Hus89adRBSiU.




В командной строке каталога mysql вызывается программа mysqladmin, с атрибутами для сервера localhost и пользователя root. Ключевое слово password задает конечный пароль для сервера базы данных.

**Пример**

Пароль конфиденциально меняется на новый пароль.

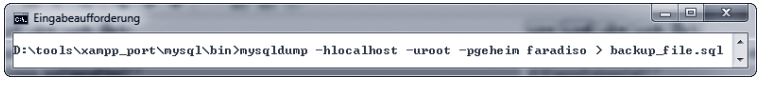

Для второго входа на сервер базы данных требуется предварительно определенный пароль после атрибута –p, чтобы впоследствии изменить этот пароль

**Клиент mysqldump**

Клиент mysqldump используется для резервного копирования данных. Программа записывает содержимое баз данных в текстовый файл.

**Пример**

Содержимое базы данных faradiso сохранено в файле backup_file.sql с целью резервного копирования.

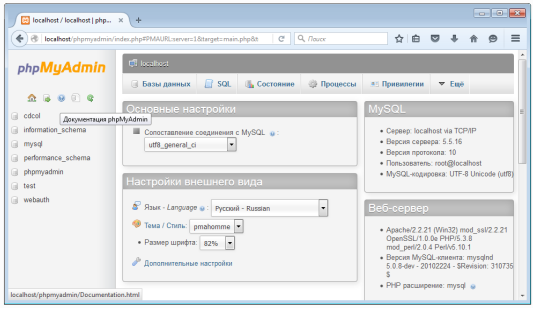



После атрибутов входа в систему указывается база данных, и после знака > указывается имя для файла экспорта, например, backup_file.sql. В этом текстовом файле записаны команды SQL, которые снова генерируют таблицы и вставляют данные при выполнении.

Опция --all-databases выполняет резервное копирование всех баз данных сервера БД, например, в файл sicherung.sql:


**Клиент phpMyAdmin**

Среда разработки XAMPP предлагает клиент phpMyAdmin по умолчанию. Он запускается с адреса localhost/phpmyadmin в браузере. Почти все задачи администрирования на сервере базы данных MySQL могут быть удобно выполнены с помощью этого инструмента. Поэтому, в дальнейшем будем работать с этим клиентом.

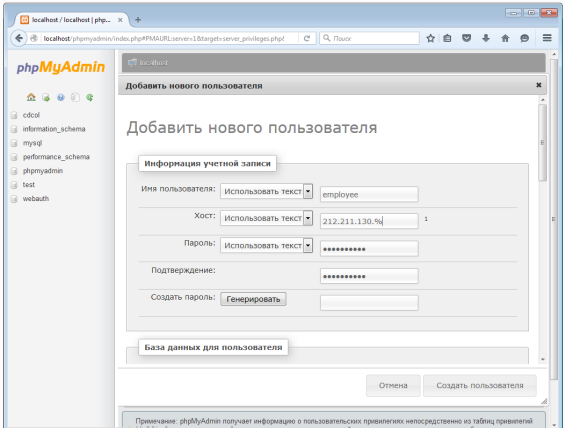



### 8.4.2 Предоставление и отмена прав доступа

MySQL хранит права пользователей в базе данных mysql очень дифференцированно.

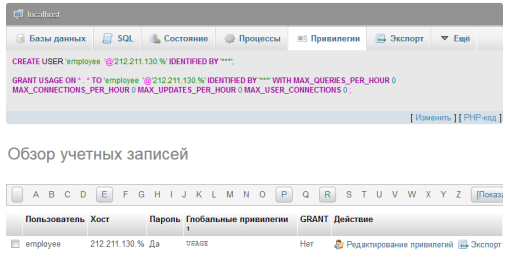
**Пример**

В клиенте phpMyAdmin, создается новый пользователь с именем `Angestellter` и паролем `Angestellter`. Он должен иметь возможность входить только с хостов домена 212.211.130.xxx.



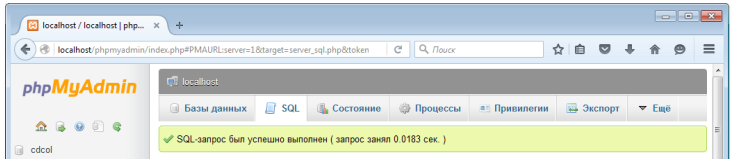

На вкладке Права на главной странице можно просмотреть пользователей, существующих в базе данных mysql. Здесь вы выбираете Добавить нового пользователя, и открываете вышеуказанный шаблон ввода данных. Производится ввод информации, необходимой для входа в систему. В строке Хост необходимо установить параметр Использовать текстовое поле в раскрывающемся окне, чтобы ввести конкретный хост, с которого пользователю разрешено входить на сервер базы данных. Если новый пользователь должен иметь возможность входить со всех компьютеров определенной сети, то вместо последнего байта адреса укажите заполнитель %.

В области Глобальные права устанавливаются права пользователя. Например, SELECT для чтения данных, INSERTдля вставки записей в таблицы и UPDATE для изменения существующих данных. При нажатии кнопки Создать пользователя новый пользователь создается оператором CREATE и ему назначаются права пользователя. Об успешном выполнении сообщается путем отображения выполненной команды в коде SQL.



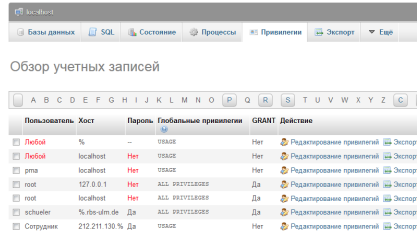

**Пример**

С помощью команды SQL создайте нового пользователя schueler с паролем schueler2012. Он должен иметь возможность входить только с хостов домена rbs-ulm.de и редактировать только базу данных faradiso со всеми правами.




Ключевое слово GRANT инициирует SQL -команду для предоставления прав пользователя. Слово ALL устанавливает все права для пользователя. С атрибутом ON faradiso.* эти права ограничиваются всеми таблицами базы данных faradiso. Атрибут *.* означает, что пользователь может редактировать все существующие базы данных. После атрибута TO стоит имя нового пользователя для входа в систему, например schueler, а после символа @ – хост или область сети, из которой пользователю разрешено входить в систему. Пароль для входа следует за ключевым словом IDENTIFIED BY.

Преимущество использования команд SQL над прямыми манипуляциями с базой данных mysql заключается в том, что администратору не нужно знать содержимое таблиц в этой базе данных и не нужно принимать решение, в какой из этих таблиц должны быть распределены записи.




Права каждого пользователя можно просмотреть в обзоре пользователей клиента phpmyadmin.

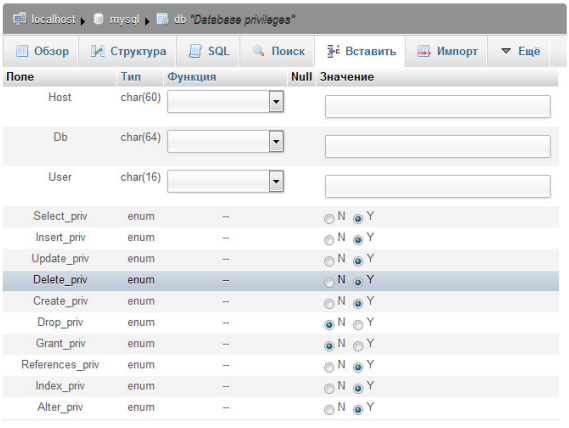
Права пользователя schueler (ученика) в примере заданы термином USAGE . Данное ключевое слово говорит о том, что пользователь schueler не имеет глобальных прав на этом сервере базы данных.

Его права на базу данных faradiso хранятся в таблице db базы данных mysql. Вот фрагмент записей в этой таблице:

| Host | Db | User | Select_priv | Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv | Grant_priv |
|---|---|---|---|---|---|---|---|---|---|
| % | test | | Y | Y | Y | Y | Y | Y | N |
| % | test_% | | Y | Y | Y | Y | Y | Y | N |
| localhost | phpmyadmin | pma | Y | Y | Y | Y | N | N | N |
| %.rbs-ulm.de | faradiso | schueler | Y | Y | Y | Y | Y | Y | N |

Например, ученик не получил права Grant_priv (запись – N), в противном случае он мог бы назначать права другим пользователям и таким образом обходить свои ограничения.

Например, чтобы пользователь schueler не смог удалить таблицы или базы данных, этого пользователя необходимо лишить права drop_priv. Это делается путем редактирования его прав в клиенте phpmyadmin. Там необходимо выбрать и подтвердить N в строке Drop_priv.




**Примечание:**
Глобальные разрешения применяются ко всем базам данных на сервере.

### 8.4.3 Редактирование базы данных MySQL с помощью PHP

PHP предоставляет различные функции для работы с MySQL:

| Функция PHP | Описание | Пример |
|---|---|---|
| mysql_connect() | Подключение к серверу БД. | $erg=mysql_connect ($host, $user, $password); |
| mysql_db_query() | Отправляет SQL-запрос на сервер базы данных. | $erg=mysql_db_query (‘faradiso’, ‘select * from клиенты’); |
| mysql_num_rows() | Количество записей в запросе. | $колво= mysql_ num_rows($erg); |
| mysql_num_fields() | Количество полей записи запроса. | $колво= mysql_num_ fields($erg); |
| mysql_close() | Закрывает подключение к серверу базы данных. | mysql_close(); |
| mysql_fetch_array() | Возвращает запись в виде массива. | $liste=mysql_fetch_array ($результат, $тип) |

Подключение к серверу базы данных и запрос данных выполняется в несколько этапов:

1. Вход пользователя на сервер базы данных
2. Настройка базы данных
3. Отправка запроса на сервер базы данных, сохранение результата в переменной
4. Закрытие соединения

Вход пользователя на сервер базы данных осуществляется с помощью функции `mysql_connect` ($host, $user, $passwd).

SQL-запрос отправляется с помощью функции mysql_db_query ($db, $sql). Например, функция mysql_num_rows ($erg_sql) считывает количество записей данных в таблице и сохраняет их в переменной $anz. Функция mysql_close () отключает соединение с сервером базы данных.

Сценарий вывода количества записей:

```
<?
    $db       = mysql_connect('localhost','root','');
    $erg_sql = mysql_db_query('faradiso', 'select * из клиенты')
    $anz = mysql_num_rows($erg_sql);
     echo $anz;
    mysql_clos
?>
```

Подключение к серверу базы данных

Просмотр

Считывание результата SQL-команды

Закрытие подключения к БД

**Пример**

Требуется разработать PHP-скрипт, который выводит таблицу клиенты базы данных faradiso на экран в виде таблицы HTML.

Сначала скрипт устанавливает соединение и отправляет SQL-инструкцию (команду). Количество требуемых столбцов определяется с помощью функции PHP mysql_num_fields($erg_sql) и задается переменной $number. С помощью цикла for создается заголовок таблицы, который помечается именами полей с помощью функции mysql_field_name($erg_sql,$i).

В цикле while функция mysql_fetch_array ($erg_sql, MYSQL_ASSOC) считывает запись результата запроса в ассоциативный массив $zeile при каждом запуске. Цикл Foreach обеспечивает доступ к отдельным полям записи и вывод в таблице HTML для каждой отдельной записи. После того, как были указаны конечные теги таблицы, соединение с сервером базы данных закрывается mysql_close ().

Пример скрипта:

```
<?php
   $db = mysql_connect('localhost','root','');
   $erg_sql = mysql_db_query('faradiso', 'select *
клиенттерден');
?>
   <кесте ені=80% border=1>
     <tr bgcolor='#CFCFCF'>
     <? $число=mysql_num_field
     for ($i=0;$i<$anzahl; $i++)
?> < th><?
echo mysql_field_n
?> </th> <?
</tr>
<tr>       <?
while ($zeile = mysql_fetc foreach ($zeile as $elem)   {
echo «<td bgcolor='#EFEFEF'><font size='-1'> $elem </
font></td>»;
}
?></tr><?
}  ?>
</table> <? mysql_close();
?>
```

Вывод заголовков столбцов

Чтение отдельных записей

Вывод содержания строки базы данных

## 8.5 Обмен данными через интерфейсы ODBC

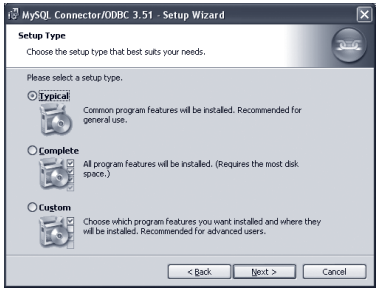
Прямой обмен данными из баз данных различных форматов, как правило, невозможен. Поэтому был создан стандарт ODBC (Open Database Connectivity = открытое соединение базы данных), который позволяет обмениваться данными через специальные драйверы ODBC. При этом СУБД, например, MySQL передает данные в ODBC драйвер, который преобразует их в формат второй СУБД, например Access.


**Примечание:**
Различные системы баз данных могут работать вместе через интерфейсы ODBC.

Для обмена информацией между Access и MySQL в Интернете доступны подходящие драйверы для Linux и Windows, например, в виде установочного файла `mysql-connector-odbc-` 3.51.23-win32.msi.

**Установка драйвера MyODBC**

Установка выполняется путем запуска установочного файла. Экран приветствия подтверждается кнопкой Далее>. Затем программа запрашивает желаемый тип установки (см. рисунок). Установите флажок `Typical` и подтвердите, нажав Далее>. В открывшемся окне отображается история установки и подтверждается успешное завершение работы.




**Редактирование базы данных MySQL с помощью Access**

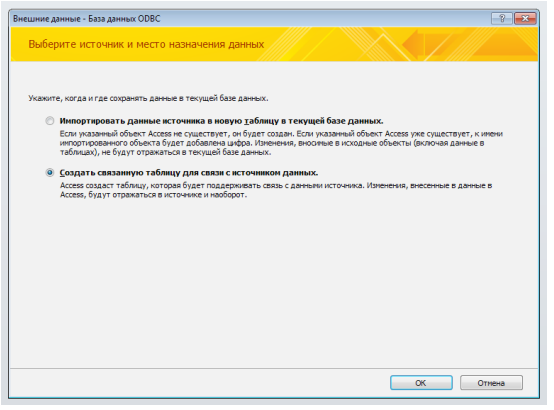
После запуска Access и создания новой базы данных через интерфейс ODBC осуществляется доступ к таблицам существующей базы данных MySQL.

**Пример**

Таблица Клиенты базы данных MySQL faradiso обрабатывается системой баз данных Access.

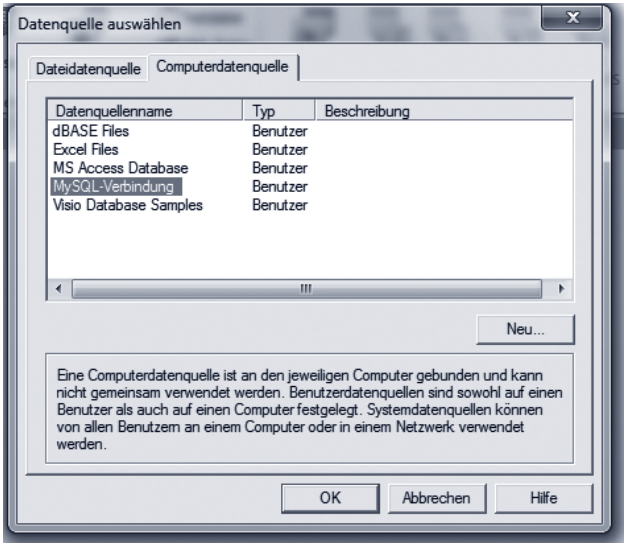
Для этого после запуска системы базы данных Access выберите в меню Внешние данные параметр База данных ODBC.


На следующем экране вы можете выбрать между импортом данных и созданием ссылки. Нажмите Создать ссылку… и подтвердите при помощи OK.

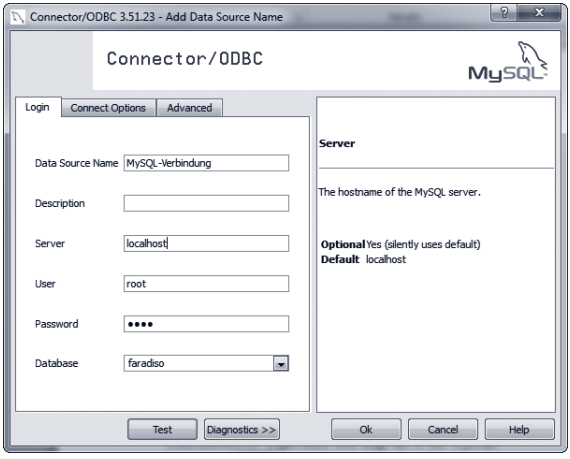



Таким образом, данные не импортируются в базу данных Access, остаются в базе данных MySQL, но могут быть отредактированы из Access посредством интерфейса ODBC. Если выбрана опция Импорт... данные будут скопированы как независимые таблицы в базу данных Access. База данных MySQL не будет изменена.

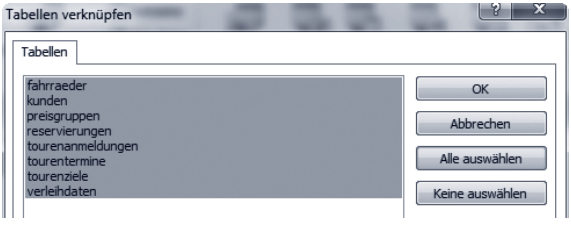
Следующее окно предоставляет выбора источника данных. На вкладке Источник данных компьютера выбирается имя источника данных MySQL connection и подтверждается нажатием OK.

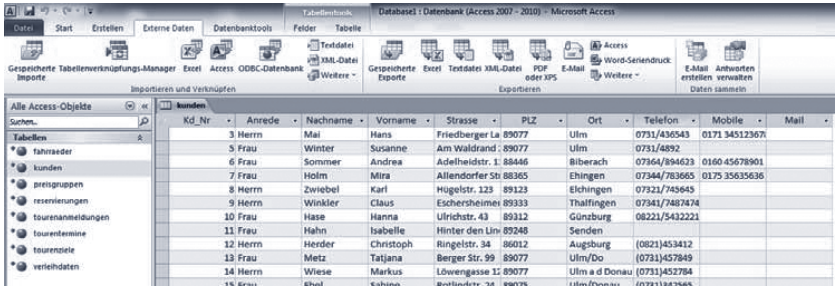



Чтобы войти на сервер MySQL, следующее окно должно содержать уникальное имя соединения, имя сервера MySQL, имя существующей базы данных, например faradiso, а также имя пользователя, з. например root. При необходимости необходимо указать еще один пароль.




После подтверждения с помощью OK Access регистрируется на сервере базы данных MySQL и отображает таблицы, доступные в базе данных MySQL faradiso. После нажатия кнопки Выбрать все и подтверждения с помощью OK все таблицы базы данных MySQL доступны для редактирования.




Изменения, которые производятся сейчас, например, в таблице Клиенты, не хранятся в базе данных Access, а передаются в базу данных MySQL через интерфейс ODBC и сохраняются там. Эти связанные таблицы базы данных MySQL обозначаются значком перед именем таблицы. 

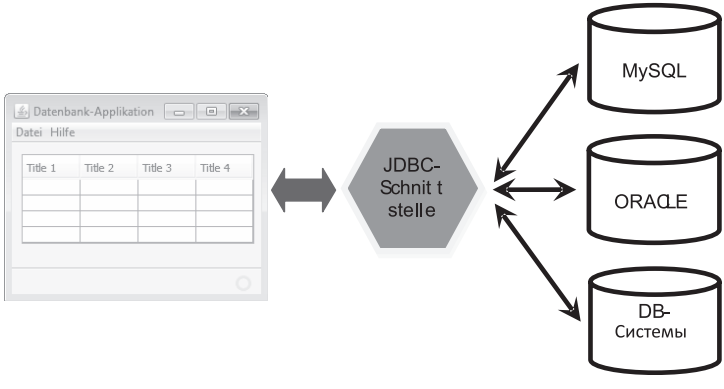
# 9 Доступ к базе данных через Java

## 9.1 Доступ к базе данных через Java

Сохранение данных является важной задачей прикладных программ. Конечно, приложение может самостоятельно хранить данные, используя файловые операции. Для небольшого объема данных это, вероятно, лучший выбор поскольку сохраняется относительная независимость данных. Однако, если требуется сохранять большой объем данных (или записей), а сами данные имеют сложную структуру, то следует рассмотреть вопрос об их хранении в базе данных. Большим преимуществом подключения к базе данных является независимость приложения от технической реализации хранилища данных. Это делается с помощью базы данных в фоновом режиме. Изменение или удаление данных также легко осуществить с помощью соответствующих команд базы данных. Язык запросов **SQL**, который уже подробно обсуждался, играет здесь важную роль.

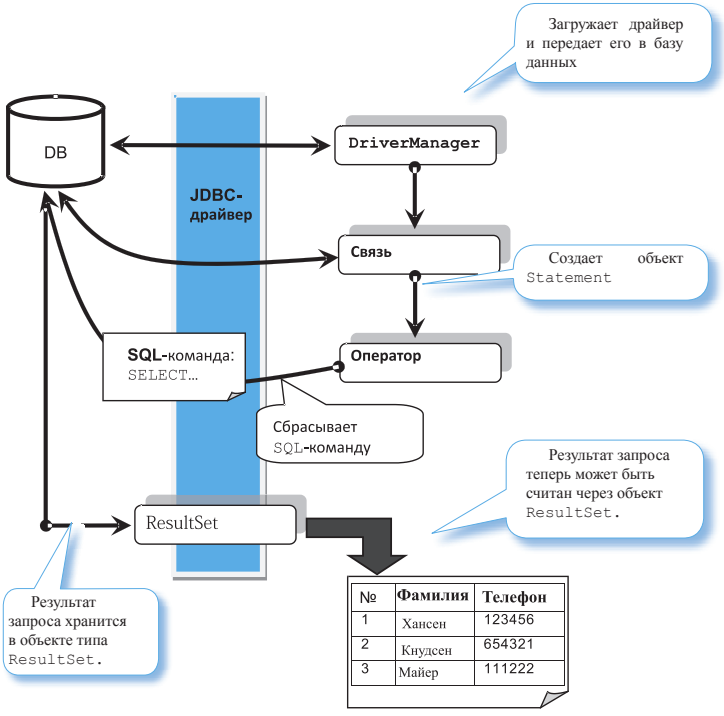
### 9.1.1 Привязка базы данных с помощью JDBC

Java предлагает множество классов для подключения к базе данных. Эти классы собраны под общим термином **JDBC** (**J**ava **D**atabase **C**onnectivity) Большинство баз данных имеют **JDCB-интерфейсы**, которые преобразуют доступ к базе данных из приложения Java в соответствующие команды базы данных. В принципе, для программиста Java нет принципиального значения, какая система баз данных используется в фоновом режиме – доступ реализуется одним способом. На следующем рисунке показан основной принцип реализации этого доступа:




### 9.1.2 Загрузка драйверов JDBC и соединение

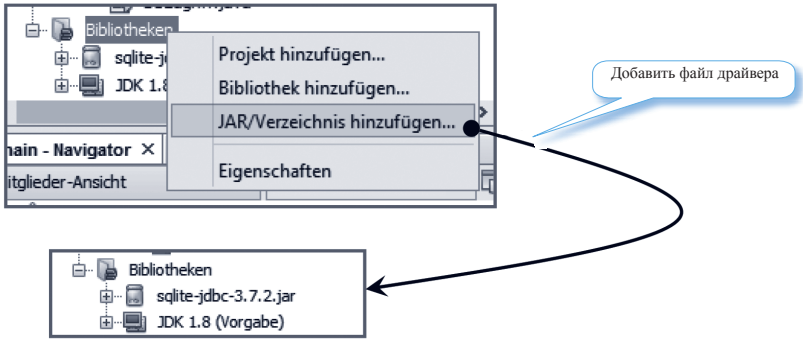
Для подключения к базе данных необходимо предварительно установить соответствующий драйвер. Затем загрузить драйвер в память можно с помощью метода класса `DriverManager`. Этот класс является частью пакета java.sql, который следует импортировать в проект Java. После успешной загрузки драйвера может быть установлено соединение с базой данных. Это делается с помощью класса Connection, который используется DriverManager для подключения к базе данных. В зависимости от базы данных необходимо указать имя пользователя и пароль. Затем объект Statement может запускать запрос и считывать его с помощью объекта ResultSet. Следующий рисунок отображает схему:




### 9.1.3 Доступ к базе данных SQLite

В следующем разделе описан доступ к базе данных SQLite. Описанные принцип можно применить и к другим реляционным базам данных, таким как базы данных MySQL или Oracle. При этом пакет должен быть импортирован пакет java.sql В этом пакете находятся все соответствующие классы для доступа к базе данных.

В следующих примерах используется среда разработки NetBeans. Этот принцип можно перенести и на другие среды (например, Eclipse). После загрузки требуемого драйвера (например sqlite-jdbc-XXX.jar) файл архива Java интегрируется в проект:



После успешного добавления драйвера, соединение может быть создано с помощью класса Class:

Строка базы данных = «jdbc:sqlite:/путь/база данных»;

Class.forName(«org.sqlite.JDBC»);

Укажите путь и файл базы данных.

Загрузить драйвер.

Установить подключение

Connection подключение;

подключение = DriverManager.getConnection(dat enbank);

Основная база данных Клиенты.sqlite находится в папке «C:\temp». Она содержит таблицу-пример Клиенты с атрибутами ID (тип Число) и Имя (тип VARCHAR):

| ID | Name |
|---|---|
| Filter | Filter |
| 1 | Maier |
| 2 | Knudsen |
| 3 | Kaiser |
| 4 | Franzen |
| 5 | Knobloch |
| 6 | Laufer |

**Совет:**

База данных SQLite является бесплатной и портируемой базой данных, которая включает полную логику базы данных и сами данные в один файл. Это позволяет передавать программы Java с собственной базой данных. Для небольших проектов с относительно небольшим объемом данных это – отличная альтернатива крупным СУБД, таким как Oracle или MySQL. С помощью бесплатных инструментов, таких как «DB-Browser for SQLite», эти базы данных можно легко администрировать.

```
package db_zugriff_java;
import java.sql.*;

public class DBДоступ{
public static void main(String[] args) { try {
```

Определите строку подключения с информацией о драйвере и источнике данных.

**Строка базы данных = «jdb**

Загрузить драйвер

**Class.f JDBC»);**

Запросите объект подключения, используя статический метод getConnection.

**Connection подключение DriverManager**

Объект подключения создает объект для команды SQL.

**Оператор sql-команда = ve**

Метод executeQuery выполняет SQL-запрос (или команду SELECT) и выдает результат в виде объекта типа ResultSet.

```
ResultSet sql-команда
```

`ResultSet` предоставляет методы для запроса таблицы результатов. метод next показывает, есть ли еще записи, а метод getString считывает следующую запись из нужного столбца (здесь Name).

```
while (результат.next() == true) {
        System.out.println(«Name: » +
                                результат.getString («Name»));
}
соединение.close();
}
catch (Exception e) {
System.out.println(e.getMessage());
}
}
}
```

**ВАЖНО:** подключения к базе данных должны быть снова закрыты.

**ВНИМАНИЕ:** При запросах к базам данных особенно важно использовать обработку исключений!

После запуска появится следующий экран:


База данных была опрошена без ошибок, а все имена клиентов были считаны и отображены.

**Примечание:**

Доступ к значениям столбца таблицы с результирующим объектом зависит от соответствующего типа данных. Для каждого типа данных доступен подходящий метод, который принимает либо индекс столбца, либо имя столбца:

- getString (int индекс столбца)
- getString ( String имя столбца)
- getDouble( int индекс столбца)
- getDouble ( String имя столбца)
- getInt( int индекс столбца)
- getInt ( String имя столбца)
- … дополнительные типы

Например, первый столбец таблицы Клиенты может быть прочитан с использованием метода getInt, т. к. это целочисленный числовой тип:

```
while (результат.next() == true) { System.out.
println(«ID: » +
результат.getInt(0));
}
```

### 9.1.4 Отмена команд, отличных от SELECT

Чтение любой таблицы можно выполнить с помощью инструкций, описанных выше. Однако если вы хотите не выбирать, а вставлять, изменять или удалять столбцы, то можно отменить так называемую команду executeUpdate. Желаемая SQL-команда должна быть ранее создана в символьной строке. В следующем примере вставляется новая строка в таблицу Клиенты, изменяется существующая строка и удаляется одна из строк:

```
package db_доступ_java; import java.sql.*;
public class DBДоступ{
        public static void main(String[] args) { try {
        Строка базы данных = «jdbc:sqlite:/c:/temp/клиенты.sqlite«;
        Class.forName(«org.sqlite.JDBC»);
        Connection соединение = DriverManager.getConnection (база данных,«»,«»);
        Statement sqlBefehl = соединение.createStatement();
```

Отменить команду SQL и получить количество затронутых строк.

SQL-команда для вставки строки.

```
uteUpdate(«IVALUES(7,'Koenig');»);

System.out.println(«Количество вставленных строк: «
+ количество);
```

Команда UPDATE

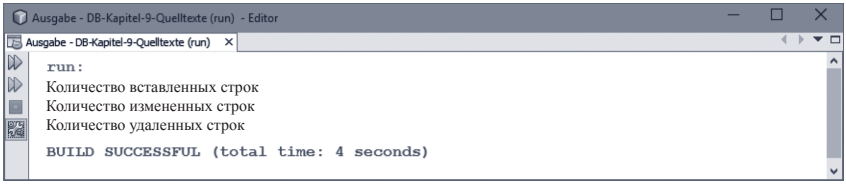
```
количество = sqlBefehl.executeUpdate(«Фамилия ='Kunden SET'
WHERE Name = 'Knobloch;»);
System.out.println(«количество измененных строк: «+ количе-
ство);
```

Команда DELETE

```
количество = sqlBefehl.executeUpdate(«DRE Name = 'Kunden
SET';»);

System.out.println(«количество удаленных строк: «+ количе-
ство);
```

```
                соединение.close();
            }
            catch (Exception e) { System.out.println(e.
getMessage());
            }
        }
}
```

После запуска выдаются три команды SQL No-Select и выводится число затронутых строк.




Для сравнения: таблица Customers (клиенты) до и после команд SQL:

**До:**

| ID | Name |
|---|---|
| Filter | Filter |
| 1 | Maier |
| 2 | Knudsen |
| 3 | Kaiser |
| 4 | Franzen |
| 5 | Knobloch |
| 6 | Laufer |

**После:**

| ID | Name |
|---|---|
| Filter | Filter |
| 1 | Maier |
| 3 | Kaiser |
| 4 | Franzen |
| 5 | Knoblauch |
| 6 | Laufer |
| 7 | Koenig |

### 9.1.5 Получение метаданных

Для легкого доступа к базе данных часто важно получить информацию о базе данных, драйвере и таблицах базы данных. Эта информация может быть получена с использованием классов метаданных БД. На основании этой информации программа может принимать дальнейшие решения. Например, таблица в базе данных может постоянно получать новые столбцы. Это означает, что доступ к этой таблице должен быть гибким, в противном случае это приводит к ошибке или сбою в работе. Для чтения метаданных: важны два класса DatabaseMetaData и ResultSetMetaData

В следующем примере показано, как использовать два класса для чтения базы данных SQLite из приведенного выше примера

```
package db_доступ_java; import java.sql.*;

public class DBДоступ{
      public static void main(String[] args) { try {
      Строка базы данных = «jdbc:sqlite:/c:/temp/клиенты.sqlite«;
            Class.forName(«org.sqlite.JDBC»);
```

```
Connection подключение =
            DriverManager.getConnection (база данных,«»,«»);
```

Объект Connection запрашивает метаданные базы данных.

**// Метаданные Базы Данных DatabaseMetaDa**

**System.out.println(«Метаданные Базы Данных:»); System.out.println(«Имя Базы Данных : «**

```
      +     dbinfos.getDatabaseProductName());     System.out.
println(«Имя драйвера: «
      + dbinfos.getDriverName()); System.out.println();
```

**// Таблицы Метаданных**

```
Statement sqlBefehl = соединение.createStatement(); ResultSet
результат=
sqlBefehl.executeQuery(«SELECT * FROM Клиенты;»);
```

**ResultSetMetaData tb**

System.out.println(«Метаданные таблицы Клиенты:»); for ( int i = 1; i <= tbinfos. getColumnCount(); i++ ) {

Посредством объекта ResultSet запрашиваются метаданные таблицы

```
System.out.println(«Имя столбца: » +
tbinfos.getColumnLabel(i)); System.out.println(«Тип столбца: »
+
tbinfos.getColumnTypeName(i));
}
соединение.close();
}
catch (Exception e) {
System.out.println(e.getMessage());
}
}
```

Запрашивать количество столбцов.
ACHTUNG: индекс начинается с 1!

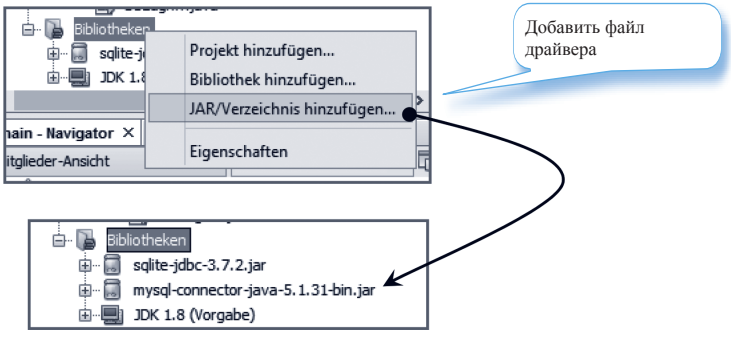
После запуска в NetBeans появится следующий экран:


## 9.2 Обратиться к другим базам данных

Большинство баз данных имеют драйверы, поэтому к ним можно получить доступ с помощью Java и JDBC. Как правило, драйвер нужно только загрузить и добавить в проект. После этого драйвер может быть зарегистрирован как обычно.

### 9.2.1 Добавление драйвера

После загрузки требуемого драйвера (например, mysql-connector-java-XXX-bin.jar) файл Java-архива интегрируется в проект:




После успешного добавления драйвера MySQL может быть установлено соединение с классом Class

**Строка базы данных = «jdbc:mysql://Servername/Datenbank«;**

Укажите имя сервера базы данных MySQL. Для локальной установки просто введите localhost.

Укажите имя базы данных.

Class.forName(«com.mysql.jdbc.Driver»);

Загрузить драйвер.

Подключиться к базе данных (с указанием пользователя и пароля).

Connection подключение;
подключение = DriverManager.getConnection (база данных

### 9.2.2 Другие драйверы базы данных

В следующей таблице приведен обзор общих баз данных и имя соответствующего драйвера Java. Соответствующий файл драйвера может быть загружен с веб-сайта провайдера, как описано выше, или иным образом.

| База данных | Имя драйвера Java |
|---|---|
| Firebird (бесплатные БД, преемник Borland Interbase) | org.firebirdsql.jdbc.FBDriver |
| DB2 (IBM) | com.ibm.db2.jcc.DB2Driver |
| Informix (IBM) | com.informix.jdbc.IfxDriver |
| Microsoft SQL-Server | com.microsoft.jdbc.sqlserver.SQLServerDriver |
| MySQL | com.mysql.jdbc.Driver |
| Oracle | oracle.jdbc.OracleDriver |
| SQLite | org.sqlite.JDBC |

**Примечание:**
Перед интеграцией базы данных, как правило, не остается альтернативы всеобъемлющему поиску в интернете или или обзору соответствующей литературы по БД.

## 9.3 Упражнения к Главе 9

**Задача 1**

В одной из компаний комиссионные торговых представителей хранятся в простой базе данных (в данном примере – в базе данных SQLite). Создайте такую базу данных с таблицей и соответствующим содержимым. Впоследствии из таблицы необходимо будет считывать следующие статистические данные:

- ► торговый представитель с самой высокой комиссией
- ► торговый представитель с наименьшей комиссией
- ► сумма всех комиссий
- ► среднее значение всех комиссий

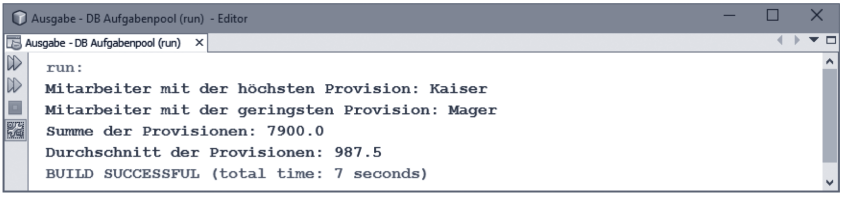
Таблица в базе данных SQLite выглядит следующим образом:

| Имя | Фамилия |
|---|---|
| Filter | Filter |
| Maier | 1200.0 |
| Knudsen | 800.0 |
| Laufer | 600.0 |
| Kaufhold | 1400.0 |
| Mager | 350.0 |
| Kaiser | 1900.0 |
| Lehmberg | 950.0 |
| Katernberg | 700.0 |

Удобное администрирование базы данных SQLite может выполняться с помощью бесплатного инструмента «DB-Browser for SQLite».

**Примечание:**
Вычисление параметров может выполняться либо с помощью соответствующих функций SQL (например SUM, AVG, MIN и MAX), либо с помощью программной логики Java.

После запуска вывод на экран может выглядеть следующим образом:




**Задача 2**

**Начальные условия:**

В одной из компаний данные заказа клиента хранятся в двух таблицах базы данных (например, SQLite). Необходимо разработать для сотрудников простое Java-приложение с графическим интерфейсом, с помощью которого можно четко отображать данные заказа клиента.

Базовые таблицы выглядят следующим образом:

**Таблица клиентов:**

| ID | Name |
|---|---|
| Filter | Filter |
| 1 | Maier |
| 2 | Knudsen |
| 3 | Kaiser |
| 4 | Franzen |
| 5 | Knobloch |
| 6 | Laufer |

Связь таблиц:

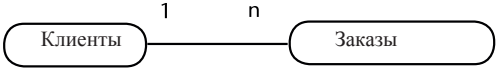


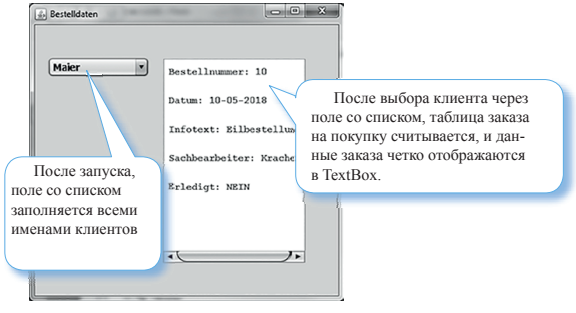

Таблица заказов:

| Клиенты | Код заказа | Дата | Инфо | Сотрудник | Выполнено |
|---|---|---|---|---|---|
| Filter | Filter | Filter | Filter | Filter | Filter |
| 1 | 10 | 10-05-2018 | Eilbestellung | Kracher | true |
| 3 | 11 | 10-06-2018 | gefährliche Fr... | Klauber | true |
| 4 | 12 | 10-07-2018 | guter Kunde | Hütter | false |

Таблица заказов имеет внешний идентификатор клиента, который реализует связь «1:n» двух таблиц.

**Постановка задачи:**

Создайте две таблицы в подходящей базе данных (например, SQLite) и заполните таблицы соответствующими значениями. Затем разработайте приложение Java, которое обращается к базе данных и считывает данные таблиц. Интерфейс приложения должен выглядеть так:



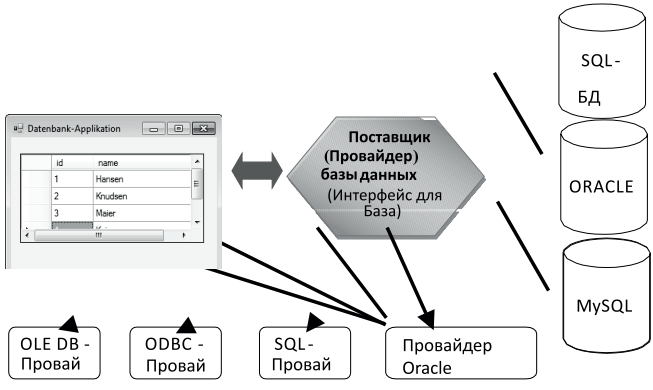
# 10 Доступ к базе данных с помощью .NET и C#

## 10.1 Доступ к базе данных с помощью .NET и C #

Язык программирования C# предоставляет множество файловых операций для постоянного хранения или чтения данных. Для данных большого объема или сложной структуры имеет смысл рассмотреть возможность их хранения в базе данных (как в языке программирования Java в предыдущей главе). Система .NET Framework предлагает удобные варианты. Язык запросов SQL также играет здесь важную роль.

### 10.1.1 Подключение базы данных под .NET Framework

.NET Framework предоставляет множество классов для подключения к базе данных. Эти классы собраны под общим термином ADO.NET. ADO расшифровывается как ActiveX Data Objects и является расширением существующей технологии Microsoft. ADO.NET обеспечивает доступ к источникам данных, таким как SQL Server или источники данных OLE DB и ODBC. На следующем рисунке показан основной принцип ADO.NET:




Отдельные поставщики (провайдеры) предназначены для определенного вида подключения к базам данных:

Поставщик ► OLE DB: OLE DB расшифровывается как Object Linking and Embedded Database и представляет собой метод, используемый в приложениях Microsoft Office. Например, можно включить электронную таблицу Excel в документ Word, чтобы изменения в исходной электронной таблице всегда были видны в электронной таблице Word (и наоборот). OLE DB может использоваться всякий раз, когда существует такой поставщик, доступный для базы данных (например, ACCESS).

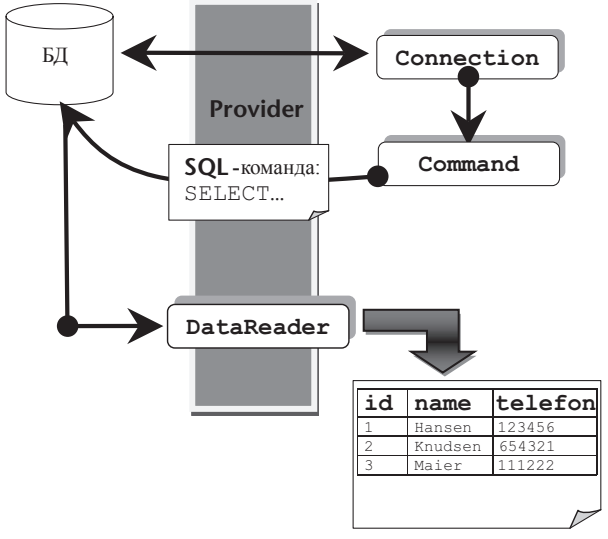
► ODBC-Provider: ODBC от Open Database Connectivity был одним из первых интерфейсов для реализации унификации доступа к базе данных. Каждая база данных должна предоставлять только один интерфейс ODBC и поэтому может использоваться приложением Windows.

► SQL: Этот поставщик предоставляет функции для доступа к Microsoft SQL Server.

►ORACLE: этот поставщик предоставляет функциональные возможности для доступа к базе данных ORACLE.

### 10.1.2 Использование провайдеров и установление соединения

Для подключения к базе данных должен иметься в наличии соответствующий поставщик базы данных. Некоторые драйверы уже присутствуют в стандартной установке Visual C # (например, поставщик OLE DB). Другие поставщики должны быть приобретены у соответствующих производителями. Затем выполняется фактическое соединение с объектом класса Connection. В зависимости от базы данных, указываются имя пользователя и пароль. При использовании объекта Command, запрос может быть запущен и считан с помощью объекта DataReader. На следующем рисунке показана связь:




### 10.1.3 Пример доступа к базе данных на ACCESS

► В следующем разделе описывается доступ к базе данных ACCESS с поставщиком OLE DB. Этот принцип можно перенести на другие системы реляционных баз данных, такие как Microsoft SQL Server или MySQL. Соединение с базой данных осуществляется с помощью объекта OleDbConnection, выдача SQL-команды – с помощью объектом OleDBCommand, чтение результата – с с помощью объекта типа OleDbDataReader.

►Среда разработки, используемая в следующих примерах, является бесплатной версией Microsoft Visual C # 2017.

В следующем исходном коде показано подключение к базе данных ACCESS Customers.accdb, которая доступна в папке (здесь: C:\temp). В данной БД есть таблица-пример Customers с аттрибуиами id (ТипZahl) и Фамилия, улица, н/пункти телефон(Text):

```
using System;
using System.Data;
using System.Data.OleDb;

namespace DB_Zugriff_CSharp
{

    class CDBZugriff
    {

    static void Main(string[] args)
    {

        string verbindungsstring =
                    "Provider=Microsoft.ACE.OLEDB.12.0;
                     Data Source=C:\\Temp\\Kunden.accdb";

        OleDbConnection dBVerbindung = null;
        OleDbCommand befehl = null;
        OleDbDataReader datenleser = null;
        bool offen = false;

        try
        {
            dBVerbindung =
                  new OleDbConnection(verbindungsstring);
            dBVerbindung.Open();
            offen = true;

            befehl = dBVerbindung.CreateCommand();
            befehl.CommandText = "SELECT * FROM Kunden";

            datenleser = befehl.ExecuteReader();

            while (datenleser.Read())
            {
                Console.WriteLine("Name: "
                            + datenleser.GetString(1));

            }
        }
```

Включите необходимое пространство имен!

Указать соединение спецификации провайдера и источника данных

Ссылка на одно OLEDB-соединение

Ссылка на команду OLEDB

Ссылка на считыватель данных OLEDB

ВАЖНО: обработка ошибок

Пример связи

Открытие БД

Установите метку

Создание объекта команды

Назначение команды SQL (все из таблицы)

На основе экземпляра чтения данных команды SQL

Последовательное считывание

Метод GetString () возвращает Значение текущей строки и переданный индекс столбца

```
catch(исключение)
{
 }
finally
{

}
}
}
}

Console.WriteLine(«ошибка базы данных: «
+ исключение.Message);

if (offen == true) dBсоединение.Close();
```


После запуска таблица клиента считывается, и значения 2-го столбца (индекс 1) выводятся шаг за шагом с использованием устройства чтения данных:

**Примечания:**
Доступ к значениям столбцов таблицы с помощью устройства чтения данных зависит от соответствующего типа данных. Подходящий метод доступен для каждого типа данных:
X GetDateTime( Индекс столбца) X GetString(Индекс столбца) X GetInt32( Индекс столбца )
X ... дополнительные типы
Например, первый столбец таблицы customer может быть прочитан с помощью метода GetInt32 (), потому что это – целочисленный числовой тип (типа ACCESS):

```
while (считыватель данных.Read())
{
Console.WriteLine («Первый Столбец: «
+ считыватель данных.GetInt32(0));
```

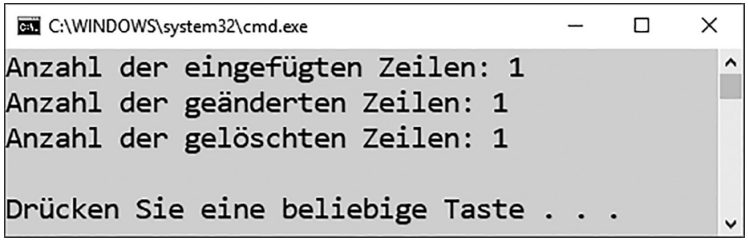
### 10.1.4 Отмена команд, отличных от SELECT

Считывание любой таблицы можно выполнить с помощью инструкций, описанных выше. Однако если вы хотите не выбирать, а вставлять, изменять или удалять столбцы, то можно отменить команду ExecuteNonQuery. Желаемая SQL-команда должна быть ранее создана в символьной строке. В следующем примере вставляется новая строка в таблицу Клиенты, изменяется существующая строка и удаляется одна из строк:

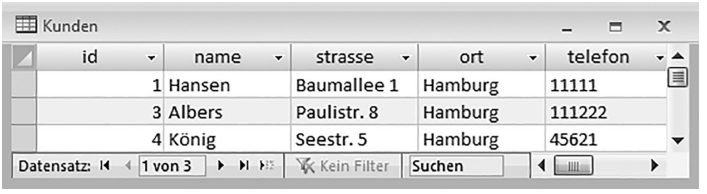
```
using System; using System.Data;
using System.Data.OleDb;


namespace DB_доступ_CSharp
{

КлассCDBДоступ
{

static void Main(string[] args)
{

string строка подключения =
«Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\\Temp\\Customers.accdb«;
OleDbConnection dBсоединение= null; OleDbCommand = null;
bool открыто = false; int число = 0;
try
{
соединение DB = new
OleDbConnection (строка подключения); Подключение DB.Open();
offen = true;
команда = dBсоединение.CreateCommand();




команда.CommandText = » вставить в customers
VALUES (4, ‘Кениг’, ‘Зеештрассе 5’,
‘Гамбург’, ‘45621’ );«;




номер=команда.ExecuteNonQuery();
Console.WriteLine («количество вставленных строк: «
+ количество);



команда.CommandText =«UPDATE Customers SET Телефон=
‘11111’ WHERE name = ‘Хансен’;«;
```

```
номер=команда.ExecuteNonQuery();
Console.WriteLine («количество измененных строк: «
+ количество);


команда.CommandText = «D
WHERE name = 'Кнудсен';«;

номер=команда.ExecuteNonQuery();
Console.WriteLine («количество удаленных строк: «
+ количество); Console.WriteLine();

}
catch(исключение)
{
Console.WriteLine(«ошибка базы данных: «
+ исключение.Message);
}
Закрытие подключения к базе данных.
finally
{
if (offen == true) dBсоединение.Close();
}
}
}
}
```

После запуска выдаются три SQL-команды, отличные от Select, и после каждой команды выводится число затронутых строк:



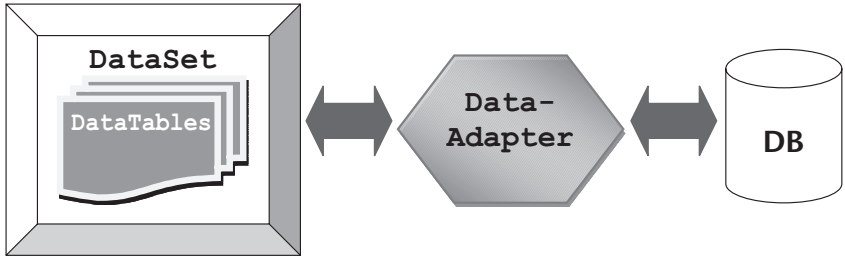

Для сравнения: таблица Customers (клиенты) до и после команд SQL:
Раньше:




Новая запись автоматически получает ID 4 от ACCESS. ACCESS игнорирует полные идентификаторы после удаления и вставки записей.

### 10.1.5 DataAdapter и набор данных

Ранее база данных открывалась и запрашивалась последовательно или изменялась с использованием соответствующих команд SQL. Хотя такой подход и практикуется, но он не очень удобен. По этой причине можно сохранить полный результат запроса в специальном объекте класса DataSet. Этот объект может быть отредактирован независимо от базы данных, и только после этого он будет синхронизирован с базой данных. Поэтому такие объекты еще называют несоединенными. Для правильного обмена данными между этими объектами и базой данных необходимы объекты -адаптеры. Следующий рисунок иллюстрирует эту взаимосвязь:




В следующем примере показано использование DataSet и DataAdapter:

```
using System; using System.Data;
using System.Data.Ole


namespace DB_доступ_CSharp
{
КлассCDBДоступ
{

static void Main(string[] args)
{
string строка подключения = «Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\\
Temp\\Customers.accdb«;
```

```
OleDbConnection dBсоединение= null; OleDbCommand = null;
bool offen = false;


try
{



dBсоединение =
new OleDbConnection(строка подключения); dBсоединение.Open();
offen = true;

команда = dBсоединение.CreateCommand(); команда.CommandText = «SELECT * FROM
Customers«;

OleDbDataAdapter da = new OleDbDataAdapter(команда);



DataSet ds = new DataSet(); da.Fill(ds);

for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
{
```


```
ds.Tables[0].Rows[0][«name»] = «Laufer«;
```

```
DataRow строка = ds.Tables[0].NewR zeile[«id»] = 10;
строка[«name»] = «Kaiser«; ds.Tables[0].Rows.Add(строка);



ds.Tables[0].Rows[1].Delete();



OleDbCommandBuilder cmb =
new OleDbCommandBuilder(da); da.Update(ds);
}
catch(исключение)
{
Console.WriteLine(«ошибка базы данных: «
+ исключение.Message);
}
finally
{
if (offen == true) dBсоединение.Close();
}
}
}
}
```

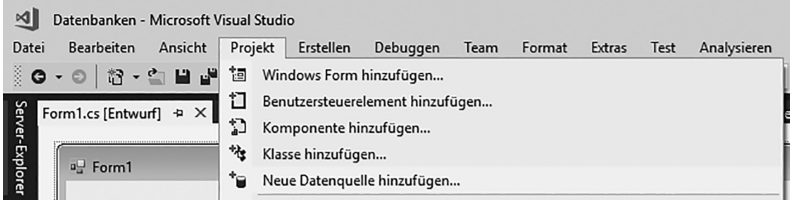
После синхронизации таблица клиентов выглядит так:

## 10.2 Использование мастера базы данных Visual C#

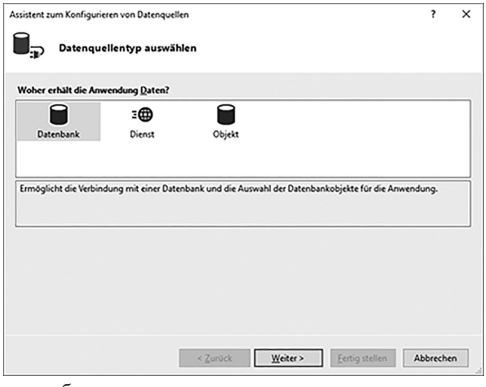
### 10.2.1 Подключение базы данных

Среда разработки Visual C # предлагает работу с мастером, который можно использовать для автоматической интеграции баз данных в проект. Это значительно сокращает время разработки по сравнению с описанными выше методами. Тем не менее, разработчик также имеет меньше свободы, потому что мастер автоматически генерирует большое количество исходного кода.

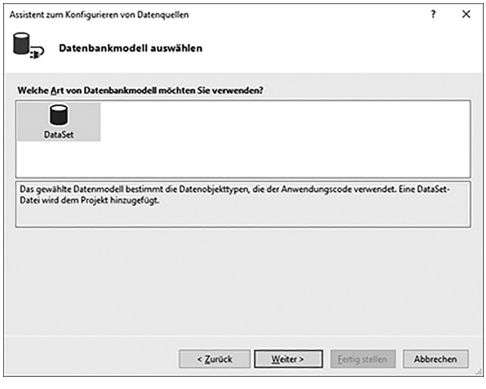
На первом этапе в проект необходимо добавить источник данных. Это достигается с помощью пункта меню «Projekt  Добавить новый источник данных…»:




На следующем шаге будет выбрана база данных:




Затем выберите модель базы данных:

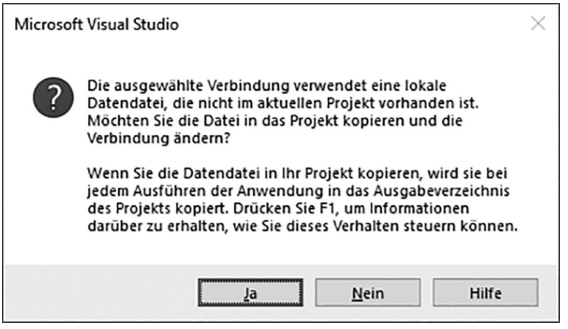


После этого выберите новое соединение:


Измените источник данных на нужную базу данных (здесь база данных ACCESS):


После проверки соединения его можно подтвердить нажатием на «OK». Затем вы можете завершить процесс с помощью кнопки «Продолжить» исходного диалога.

Однако до этого мастер запрашивает использование базы данных в качестве копии:

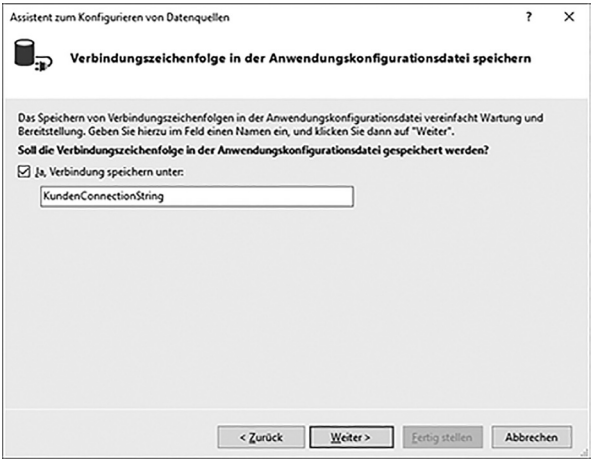



Онлайн-справка Microsoft предоставляет информацию об этой опции:

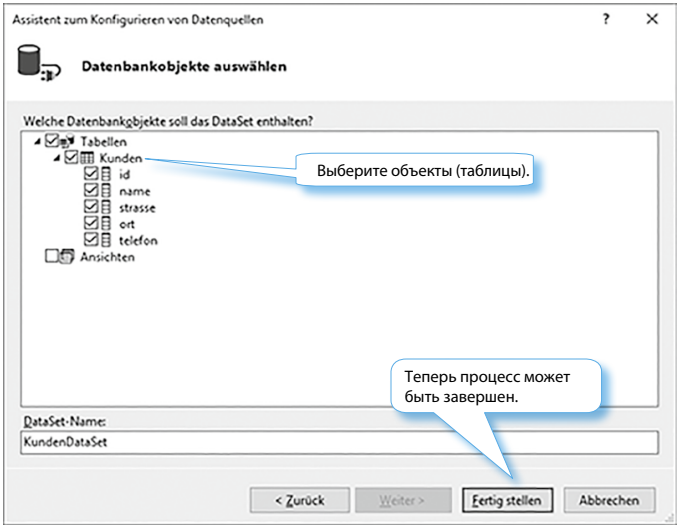
> Файл локальной базы данных может быть включен в проект в виде файла. При первом подключении приложения к локальному файлу базы данных вы можете сделать копию базы данных в своем проекте или подключиться к файлу базы данных в его текущем местоположении. Когда вы подключаетесь к существующему файлу, соединение устанавливается так же, как и к любой другой удаленной базе данных, и файл базы данных остается в своем первоначальном расположении. Если вы хотите скопировать базу данных в ваш проект, Visual Studio создает копию файла базы данных, добавляет ее в проект и изменяет соединение так, чтобы оно указывало на базу данных в проекте, а не на исходное местоположение файла базы данных.

В данном примере – выбрано «нет», программа будет работать с исходной базой данных (без копии).

Затем соединение может быть сохранено под именем:

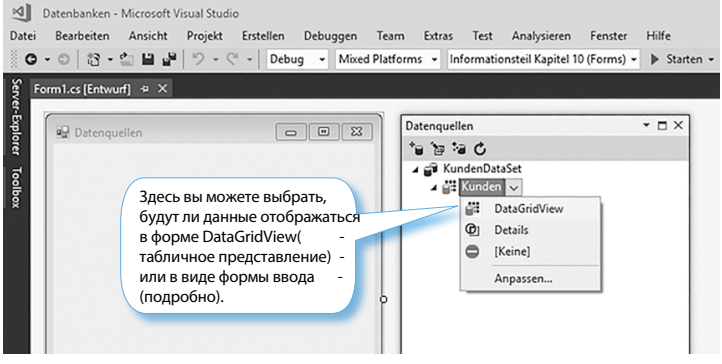


Теперь можно выбрать объекты базы данных. В этом случае выбирается полная таблица клиентов:

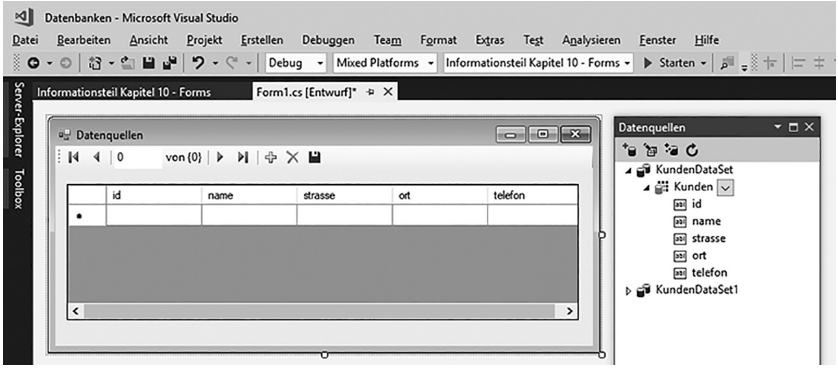



### 10.2.2 Автоматическое подключение элементов управления Windows Forms

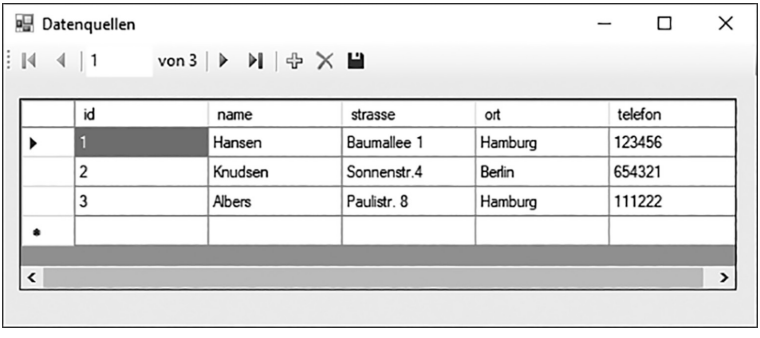
Используйте пункт меню Вид □ Другие окна □ Источники данных чтобы отобразить источник данных, а затем использовать его для приложения Forms.



Теперь таблицу «Клиенты» можно перетащить на форму в нужном представлении (здесь DataGridView) с помощью функции Drag&Drop. Мастер автоматически создает соответствующий элемент управления и соединение с базой данных и таблицей:




После запуска доступно функциональное представление таблицы базы данных.

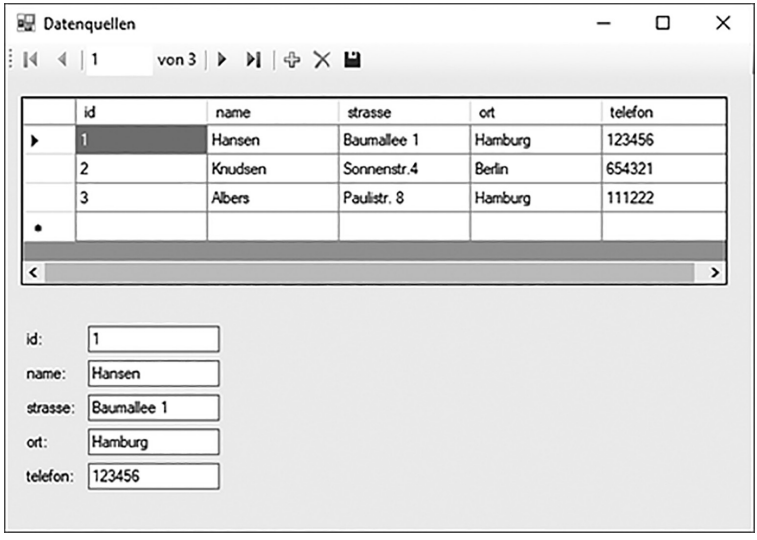



| | id | name | strasse | ort | telefon |
|---|---|---|---|---|---|
| ▶ | 1 | Hansen | Baumallee 1 | Hamburg | 123456 |
| | 2 | Knudsen | Sonnenstr.4 | Berlin | 654321 |
| | 3 | Albers | Paulistr. 8 | Hamburg | 111222 |
| * | | | | | |

Записи могут быть изменены, а также удалены. Также можно добавлять новые записи. Конечно, разработчик имеет в своем распоряжении множество событий и функций, доступных для программирования элемента. Например, двойным щелчком по ячейке можно создать обработчик события CellContentClick, который реагирует на щелчок по содержимому ячейки.

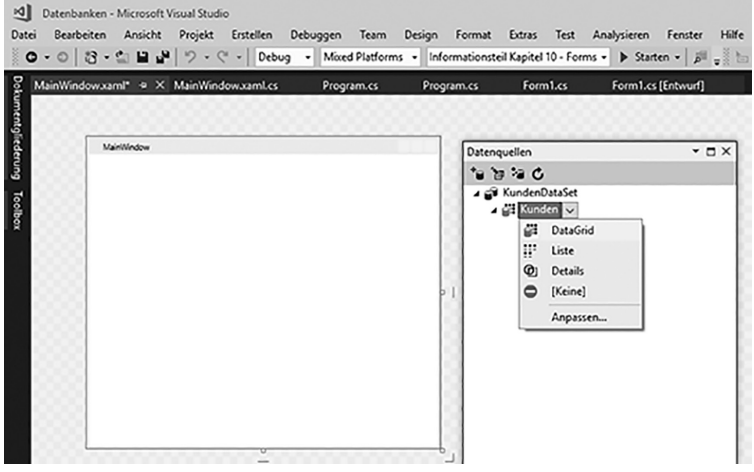
```
private void customersDataGridView_CellContentClick(object ,
DataGridViewCellEventArgs e)
{
MessageBox.Show («нажатие на содержимое ячейки!»);
}
```

В дополнение к представлению GridView, подробное представление также можно перетаскивать (Drag&Drop ) на форму. Это позволяет пользователю просматривать не только таблицу, но и форму. Отображение данных синхронизируется автоматически:

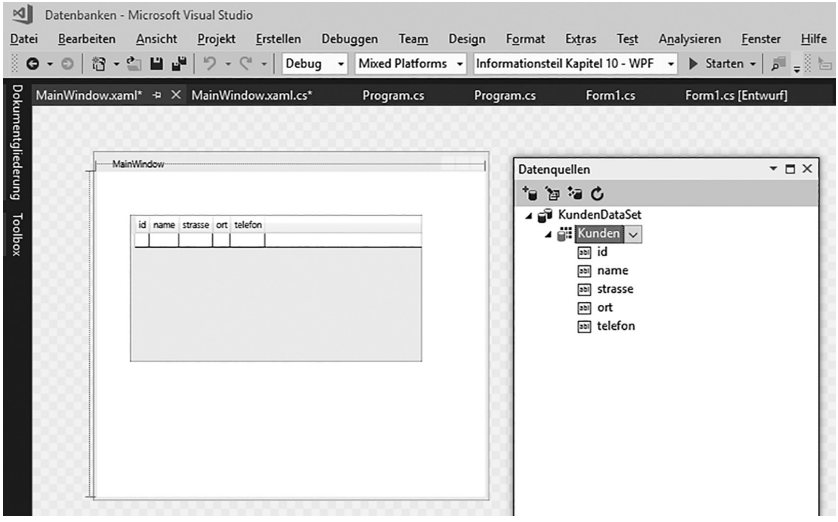



### 10.2.3 Автоматическое подключение элементов управления WPF

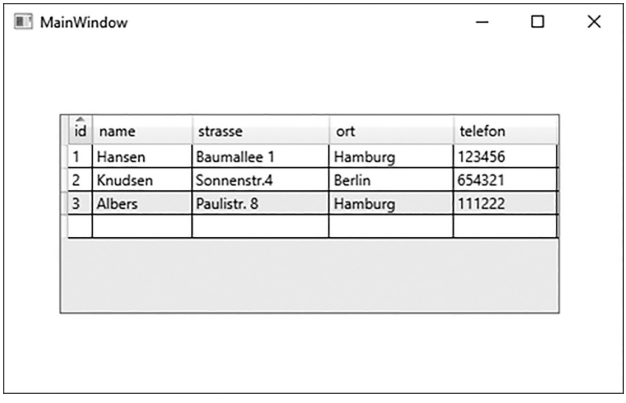
Используйте пункт меню Вид → Другие окна→ Источники данных» для отображения источника данных, а затем используйте его для приложения WPF.



Теперь таблицу Customers (Клиенты) можно перетащить в окно в желаемом виде (здесь DataGrid). Мастер автоматически создает соответствующий элемент управления и соединение с базой данных и таблицей:




После запуска будет доступно представление таблицы базы данных.



Связанный XAML-код выглядит следующим образом:

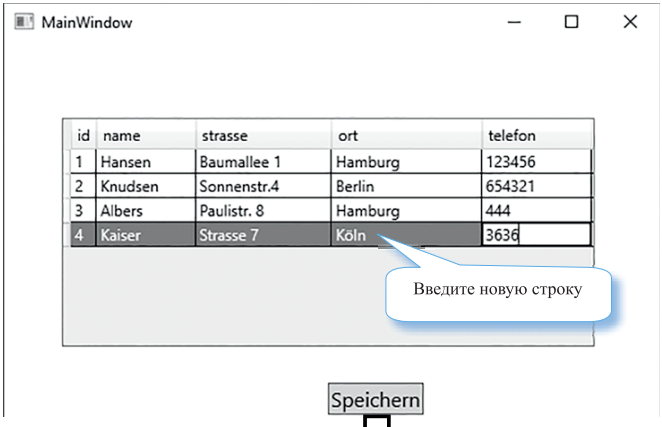
```
<DataGrid x:Name="kundenDataGrid" AutoGenerateColumns="False"
EnableRowVirtualization="True" ItemsSource="{Binding}" Margin="45,59,51,64"
RowDetailsVisibilityMode="VisibleWhenSelected">
        <DataGrid.Columns>
                <DataGridTextColumn x:Name="idColumn"
                Binding="{Binding id}" Header="id" Width="SizeToHeader"/>
                <DataGridTextColumn x:Name="nameColumn" Binding="{Binding na
                    me}" Header="name" Width="SizeToHeader"/>
                <DataGridTextColumn x:Name="strasseColumn" Binding="{Binding
                    strasse}" Header="strasse" Width="SizeToHeader"/>
                <DataGridTextColumn x:Name="ortColumn" Binding="{Binding ort}"
                    Header="ort" Width="SizeToHeader"/>
                <DataGridTextColumn x:Name="telefonColumn" Binding="{Binding
                    telefon}" Header="telefon" Width="SizeToHeader"/>
        </DataGrid.Columns>
</DataGrid>
```

Столбцы соответствуют заголовку.

Привязка данных WPF!

В таблицу также могут быть добавлены новые значения, но автоматическая синхронизация не выполняется. Для этого в коде файла должен быть вызван определенный метод класса DatenAdapterklasse. Следующий метод событий показывает способ синхронизации.

private void Button_Click(object sender, RoutedEventArgs e)




```
ate void Button_Click(object sender, RoutedEventArgs e)
```

При создании экземпляра объекта CommandBuilder, экземпляр адаптера получает всю необходимую информацию для обновления!

```
    try
    {
          OleDbCommandBuilder cmbKunden = new
                OleDbCommandBuilder(kundenDataSetKundenTableAdapter.
                                                          Adapter);

             kundenDataSetKundenTableAdapter.Update(kundenDataSet);
             MessageBox.Show("Update erfolgreich");

          }
          catch (System.Exception ex)
          {
             MessageBox.Show("Update-Fehler:" + ex);
          }
```

Произведите Обновление!

Однако в файле коде, исходный код должен быть изменен таким образом, чтобы ссылки для соединения с базой данных создавались как атрибуты. В противном случае доступ к объекту адаптера был бы невозможен, как в приведенном выше методе.

Настроенный исходный код выглядит так:

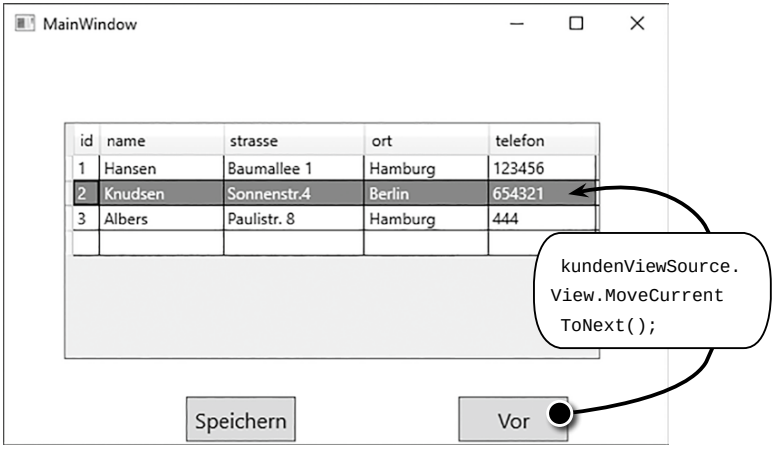
```
public partial class MainWindow : Window
{
       public MainWindow()
       {
              InitializeComponent();
       }

       private KundenDataSet kundenDataSet;

       private KundenDataSetTableAdapters.KundenTableAdapter
                                          kundenDataSetKundenTableAdapter;

       private System.Windows.Data.CollectionViewSource kundenViewSource;

       private void Window_Loaded(object sender, RoutedEventArgs e)
       {
              kundenDataSet =
              ((KundenDataSet)(this.FindResource(„kundenDataSet“)));

              kundenDataSetKundenTableAdapter = new
              KundenDataSetTableAdapters.KundenTableAdapter();

              kundenDataSetKundenTableAdapter.Fill(kundenDataSet.Kunden);

              kundenViewSource =
              ((Sysem.Windows.Data.CollectionViewSource)
                     (this.FindResource(„kundenViewSource“)));

              kundenViewSource.View.MoveCurrentToFirst();
       }
       :
       :
}
```

Создать ссылки как частные атрибуты.

### Методы навигации

В дополнение к методу обновления для фиксации изменений, можно использовать Move-методы класса ViewSource для создания движений в записях:

```
kundenViewSource.View.MoveCurrentToNext();
kundenViewSource.View.MoveCurrentToPrevious();
kundenViewSource.View.MoveCurrentToFirst();
kundenViewSource.View.MoveCurrentToLast();
```




## 10.3 задачи к главе 10

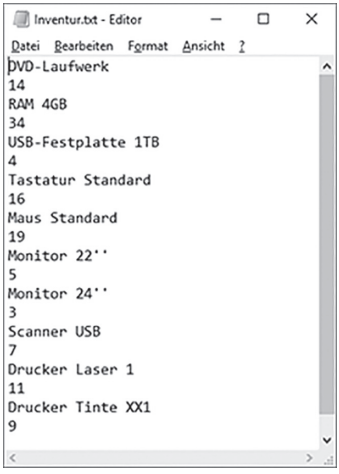
### Задача 1

В одной из компаний для инвентаризации используется мобильный сканер, который сканирует все товары со штрих-кодом. Кроме того, количество товраов может быть отсканировано с помощью листа штрих-кода. После инвентаризации все данные сохраняются в виде текстового файла на микросхеме памяти сканера. Напишите простое консольное приложение, которое считывает такой текстовый файл и сохраняет его в таблице базы данных. Таблица базы данных ранее создавалась в подходящей базе данных (например, ACCESS) с соответствующими командами SQL. Впоследствии из таблицы считываются следующие статистические данные:

X Три товара, которые доступны в наибольшем количестве. X Три товара, которые доступны в наименьшем количестве. X среднее количество товаров.

10 доступ к базе данных с помощью .NET и C#

После инвентаризации текстовый файл может выглядеть так:

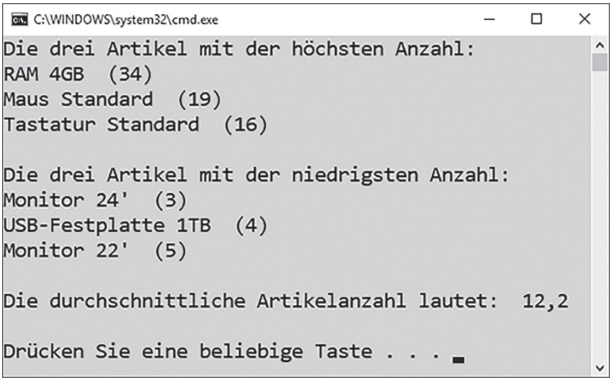



**Примечания:**

X для вставки записей используйте SQL-оператор или объект DataSet.

X Расчет данных выполняется либо с помощью соответствующих функций SQL (например, AVG), либо непосредственно в программе C#.

После запуска, вывод на экран может выглядеть следующим образом:




### Задача 2

**Начальные условия:**

В одной из компаний данные заказа клиента хранятся в двух таблицах базы данных (например, Access). Необходимо разработать для сотрудников простое приложение с графическим интерфейсом, с помощью которого можно четко отображать данные заказа каждого клиента.

Базовые таблицы выглядят следующим образом:

Таблица клиентов:

| Kunden | |
|---|---|
| 1 ID | n Name |
| 1 | Maier |
| 2 | Knudsen |
| 3 | Kaiser |
| 4 | Franzen |
| 5 | Knobloch |

Связь таблиц:

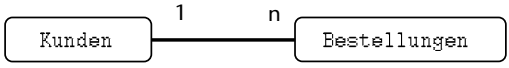


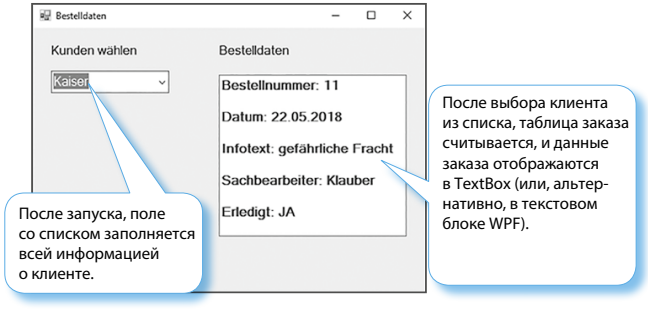

Таблица заказов:

| Bestellungen | | | | | |
|---|---|---|---|---|---|
| KundenID | Bestellnummer | Datum | Infotext | Sachbearbeiter | erledigt |
| 1 | 10 | 10.05.2018 | Eilbestellung | Kracher | Ja |
| 3 | 11 | 22.05.2018 | gefährliche Fracht | Klauber | Ja |
| 4 | 12 | 20.05.2018 | guter Kunde | Hütter | Nein |

Таблица заказов имеет внешний идентификатор Клиент_ID, который реализует связь 1:n для двух таблиц.

Постановка задачи:

Создайте две таблицы в подходящей базе данных (например, ACCESS) и заполните таблицы соответствующими значениями. Затем внедрите приложение Windows Forms или WPF, которое обращается к базе данных и считывает таблицы. Интерфейс приложения должен выглядеть так:



**Предметный указатель**

**Индекс**

*Серия «Профессиональное образование»*

# БАЗЫ ДАННЫХ

*Учебник*

***Художественный редактор*** Ж. Казанкапов
***Технический редактор*** Э. Заманбек
***Дизайнер*** Д. Жылкышин
***Верстка*** А. Кадикеновой
***Корректор*** Н. Мордвинцева

Подписано в печать 10.10.2019.
Формат 60x90 1/16. Бумага офсетная.
Печать офсетная. Усл. п.л. 11,5
Тираж 50 экз. Заказ № 0206.

Издательство «Фолиант».
010000, г. Нур-Султан, ул. Ш. Айманова, 13.

Отпечатано в типографии «Регис-СТ Полиграф».
010000, г. Нур-Султан, ул. Ш. Айманова, 13.