

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования Российской Федерации

РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Н.Н. Попов, Л.В. Александрова, В.М. Абрамов



**Инновационные технологии геоинформационного обеспечения
управления данными предприятия**

СпецЛит
Санкт-Петербург

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования Российской Федерации

**РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ**

Н.Н. Попов, Л.В. Александрова, В.М. Абрамов

**Инновационные технологии геоинформационного обеспечения
управления данными предприятия**

Учебное пособие

СпецЛит
Санкт-Петербург
2017

УДК 322.24(073)

ББК 32.97

П76

Одобрено методической комиссией Института информационных систем и геотехнологий РГГМУ в качестве учебного пособия по направлениям подготовки. 090302, 090303, 380305

Инновационные технологии геоинформационного обеспечения управления данными предприятия / Н.Н. Попов, Л.В. Александрова, В.М. Абрамов, – СПб.: СпецЛит, 2017. - 51 с.

Ответственный редактор: Истомин Е.П. – д-р техн. наук, профессор, директор Института информационных систем и геотехнологий РГГМУ.

Учебное пособие соответствует программам очной формы обучения студентов РГГМУ по дисциплинам: «Управление данными предприятия», «Объектно-ориентированное проектирование», «Геоинформационные системы», «Системы управления базами данных», «Основы управления рисками», «Геориски», «Управление рисками». Материал включает в себя основные сведения о языке SQL, а также основы работы с базами данных и основными СУБД, используемыми при геоинформационном обеспечении поддержки решений при рациональном природопользовании, включая управление рисками. Учебное пособие предназначено для студентов, обучающихся по направлениям подготовки 090302 - Информационная безопасность телекоммуникационных систем, 090303 – Прикладная информатика, 380305 - Бизнес-информатика. Оно также может быть полезно для студентов, аспирантов, научных работников и преподавателей в области гидрометеорологических и экологических специальностей.

Настоящее учебное пособие создано при финансовой поддержке Минобрнауки России (проект 5.4425.2017/NM).

Innovative technologies of geoinformation support within enterprise data control

ISBN 978-5-299-00-923-1

© Н.Н. Попов, Л.В. Александрова, В.М. Абрамов, 2017

ОГЛАВЛЕНИЕ

	Стр.
Предисловие	4
Введение	5
1 Основы систем управления базами данных (СУБД)	6
1.1 Основные понятия СУБД	6
1.2 Как развивались СУБД	9
1.3 Основные функции СУБД	12
2 Проектирование базы данных	14
2.1 Вступительное слово	14
2.2 Как создать базу данных	15
2.3 Практическая часть	22
3 Язык SQL	29
3.1 Определение	29
3.2 Базовый синтаксис языка SQL	29
3.2.1 Диалекты SQL	29
3.2.2 Операторы SQL	31
3.3 Пример использования SQL для работы с базой студентов	38
4 MySQL	46
4.1 Общие сведения о MySQL	46
4.2 Начало работы с системой	48
5 Big Data	56
5.1 История происхождения термина	56
5.2 Принципы работы	57
5.3 Применение Big Data	59
5.4 Big Data в политике	59
Заключение	62
Список литературы	63

ПРЕДИСЛОВИЕ

В учебном пособии изложены в систематизированном виде сведения об основах языка SQL, принципах построения реляционных баз данных, основах систем управления реляционными СУБД. Приведен пример создания простейшей СУБД.

Материал состоит из пяти глав и начинается с рассмотрения основных принципов работы реляционных баз данных, принципов работы систем управления базами данных. Далее приводится описание основ языка SQL и одной из популярных СУБД - MySQL. Значительное внимание в пособии уделяется практической части, которая присутствует в первых четырех главах, что позволяет использовать пособие для подготовки и проведения практикумов и лабораторных работ. Пятая глава, посвященная Big Data, является обзорной и предназначена, в первую очередь, для расширения кругозора и понимания сложности и многообразия систем сбора и анализа разнородных данных.

Учебное пособие соответствует программам обучения студентов РГГМУ очной формы обучения дисциплин «Управление данными предприятия», «Объектно-ориентированное проектирование», «Основы управления рисками», «Геориски», «Управление рисками» и предназначено для студентов, обучающихся по направлениям подготовки 090302 - Информационная безопасность телекоммуникационных систем, 090303 – Прикладная информатика, 380305 - Бизнес-информатика. Оно также может быть полезно для студентов, аспирантов, научных работников и преподавателей в области гидрометеорологических и экологических специальностей.

ВВЕДЕНИЕ

С давних времен человечество занималось накоплением, обработкой, хранением и передачей информации. Сведения о том, когда и как охотиться, ловить рыбу, выращивать и собирать урожай, копились, анализировались и бережно передавались из поколения в поколение, обеспечивая тем самым устойчивое развитие цивилизаций. В современном мире стоимость информации может измеряться суммами, значительно превышающими стоимость оборудования, на котором хранятся данные. Примером может служить внезапно вышедший из строя банковский сервер. В данной ситуации организация не только вынуждена восстанавливать весь объём утраченных данных, но и компенсировать убытки, вызванные крахом системы. Таким образом, системы хранения и распространение информации являются важным объектом, рассматриваемым специалистами в области информационной безопасности.

Адекватный подход к проектированию, разработке, внедрению и поддержанию данных систем является залогом информационной безопасности, и позволяет организации максимально эффективно заниматься своей деятельностью.

1. Основы систем управления базами данных (СУБД)

1.1 Основные понятия СУБД

Говоря о системах управления базами данных (СУБД), нельзя обойти стороной некоторые определения. Что же такое данные, база данных (БД) и СУБД?

Данные – это полезная информация. Необходимо сделать оговорку, что всё, что мы получаем с помощью органов чувств, является информацией. Одна информация несет нам полезные сведения о среде, на основе которых мы можем сделать определенные выводы и принять определенные решения, другая не содержит никаких прикладных полезных сведений. Давайте определимся, что в этом курсе данными мы будем называть только полезную информацию. Только ту, которую мы целенаправленно собираем, обрабатываем и храним.

База данных (БД) – это упорядоченный набор данных. Здесь предполагается, что собранная полезная информация была каким-то образом структурирована, упорядочена и сохранена в списках.

Система управления базами данных (СУБД) – это программа, которая позволяет хранить и управлять собранными данными. Она отвечает за своевременное резервное копирование блоков данных, обрабатывает запросы пользователей и обеспечивает различные уровни доступа к данным.

В настоящее время СУБД используются практически во всех отраслях современной экономики, так как предприятия постоянно оперируют значительными объемами информации. Примером может

служить банковское дело или работа провайдера телекоммуникационных услуг (рисунок 1.1).



Рисунок 1.1 – Пример систем, использующих СУБД

В самом простом случае работа СУБД, отображенной на рисунке 1.1, может быть описана следующим образом:

1. Есть таблица, в которой хранится информация о номере карты (CardID), фамилии и имени клиента (LastName и FirstName), балансе карты (Balance), а также сведения об удостоверении личности для возможности идентификации клиента (PassportID);

2. Каждый раз при совершении транзакции (например, оплате покупок или начислении зарплаты) происходит обращение к этой таблице и изменяется значение, указанное в графе «Balance»: из первоначальной суммы вычитается стоимость

покупки или прибавляется какое-то количество денег, поступившее в виде зарплаты. После чего полученное значение записывается обратно в графу «Balance». В упрощенном виде по тому же принципу происходит работа с базами операторов мобильной связи и Интернета.

С развитием Интернета СУБД начали массово применяться при создании сайтов и порталов. При рассмотрении практически любого функционального интернет-проекта мы всегда увидим базу данных, на основе которой он работает. Примерами могут являться порталы социальных сетей (ВКонтакте, Facebook, Instagram, Tweeter), информационно-справочные системы (Kinopoisk, Kinoafisha.spb.ru), всевозможные системы продажи билетов (РЖД, Kassir.ru, Aviasales) или любых других ресурсов, в чей функционал заложено выведение справочной информации, которая может потребоваться посетителю портала (рисунок 1.2).

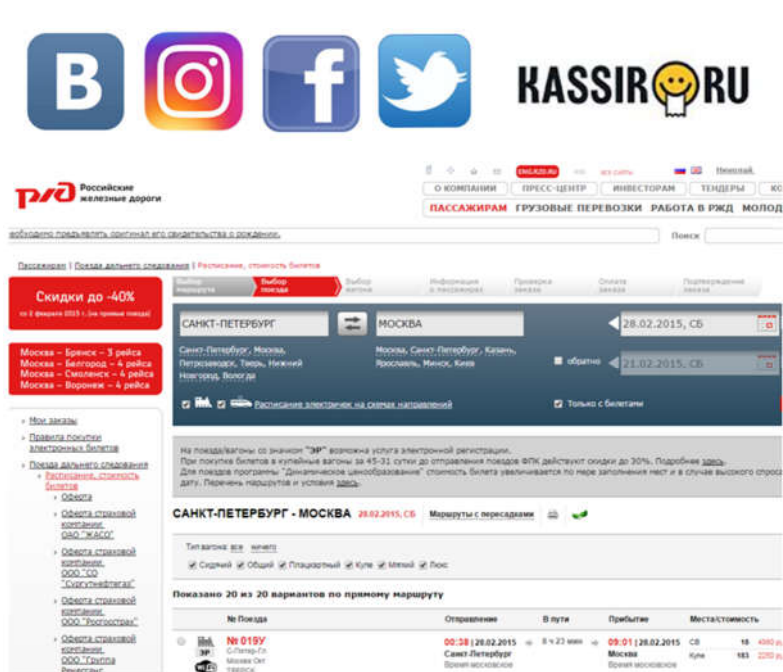


Рисунок 1.2 – Пример сайта, использующего в своей работе СУБД

1.2 Как развивались СУБД

Изначально информация хранилась в виде учетных карточек на бумажном носителе (рисунок 1.3). Примером могут быть записные книжки и каталоги. Главным и основным неудобством при таком хранении информации является то, что подобный каталог достаточно сложно скопировать. Представьте, как сложно будет изготовить копию учетных карточек книг в библиотеке. И, конечно же, при неправильном хранении или чрезвычайном происшествии этот банк данных может быть безвозвратно потерян. Кроме того, оперировать с такими носителями крайне неудобно современному пользователю: на их основе невозможно реализовать автоматический поиск или сортировку по нужному элементу.



Рисунок 1.3 – Хранение информации в картотеке

С развитием компьютерной техники появилась возможность хранить интересующую нас информацию в виде самых простых

текстовых файлов. Это позволило осуществлять простое копирование и хранение данных, а также их распространения по сети, в случае необходимости. Но всё равно при хранении информации только в обычном текстовом формате отсутствует возможность сортировки.

Рассмотрим ситуацию, когда необходимо вести учет того, что мы одолжили нашему товарищу (рисунок 1.4). В этом случае на каждого абонента записной книжки придётся создать дополнительный текстовый файл, который будет связан с основным списком условно.

Появление электронных таблиц в какой-то мере решила задачу упорядочивания и сортировки массивов данных и позволила хранить однотипную информацию в отведенных для этого столбцах (рисунок. 1.5), что позволило проводить поиск необходимой информации не во всем документе, а только в его определенной области – это значительно ускорило работу системы.

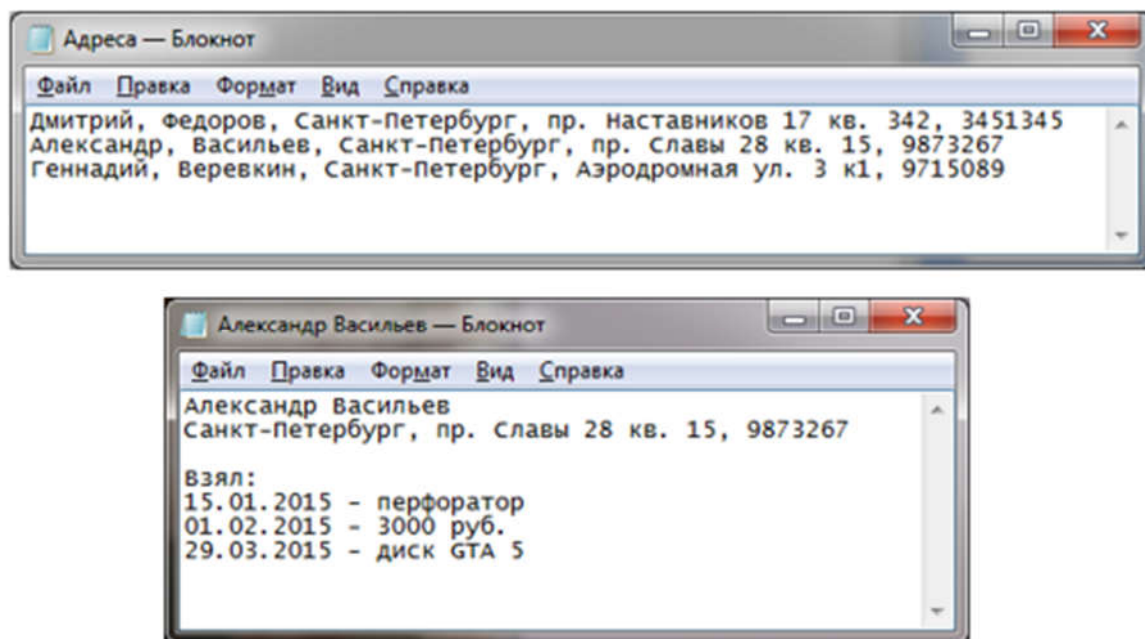


Рисунок 1.4— Хранение информации в текстовых файлах

	A	B	C	D	E	F	G	H
1	Фамилия	Имя	Город	Адрес	Телефон			
2	Федоров	Дмитрий	Санкт-Петербург	пр. Наставников 17 кв. 342	3451345			
3	Васильев	Александр	Санкт-Петербург	пр. Славы 28 кв. 15	9873267			
4	Веревкин	Геннадий	Санкт-Петербург	Аэродромная ул. 3 к1	9715089			
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								

Рисунок 1.5 – Хранение информации в электронной таблице

На ранних этапах развития систем управления базами данных отсутствовали технические возможности для реализации графического интерфейса, позволяющего оператору в интерактивном режиме общаться с базой данных. Для этих целей был разработан язык SQL, структура которого очень похожа на обычный английский язык. Чтобы понять, как работает SQL, давайте рассмотрим пример, приведенный на рисунке 1.6. Здесь представлена таблица базы данных, в которой хранятся сведения о людях. Необходимо вывести список фамилий и имен тех, кто родился после 1980 года. Запрос, приведенный на рисунке, читается очень легко: первое слово описывает то, что следует сделать СУБД – выбрать (SELECT) все записи из полей «LastName» и «FirstName» из (FROM) таблицы «MyDB» (название всей таблицы), где (WHERE) значение в поле год (Year) больше 1980. Просто, не правда ли? Эта интуитивная понятность в свое время позволила популяризировать компьютерную технику и базы данных среди рядовых клерков.

LastName	FirstName	City	Address	Phone	Day	Month	Year	Sex
Федоров	Дмитрий	Санкт-Петербург	пр. Наставников 17 кв. 342	3451345	12	6	1981	М
Васильев	Александр	Санкт-Петербург	пр. Славы 28 кв. 15	9873267	6	11	1970	М
Веревкин	Геннадий	Санкт-Петербург	Аэродромная ул. 3 к1	9715089	30	12	1947	М
Харламов	Павел	Москва	Алтайская ул. 17, кв. 5	4569632	11	10	1983	М
Ляховец	Юлия	Тель-Авив	пл. Моше Даяна 12	1298654	17	3	1981	Ж



SELECT LastName, FirstName **FROM** MyDB **WHERE** Year>1980



LastName	FirstName
Федоров	Дмитрий
Харламов	Павел
Ляховец	Юлия

Рисунок 1.6 – Работа SQL запроса

1.3 Основные функции СУБД

Принято выделять следующие основные функции СУБД:

Обеспечение хранения данных – основная и важнейшая функция СУБД;

Обеспечение доступа к данным – СУБД должна давать возможность одновременной работы с данными пользователям, которые могут находиться в разных местах (например, бронирование билетов на самолет);

Обеспечение высокой скорости отклика – применение передовых алгоритмов поиска позволяет значительно сократить время ожидания ответа, что может быть критичным при принятии оперативных решений управления;

Обеспечение разграничения привилегий – необходима для разграничения уровней доступа. В качестве примера, рассмотрим базу данных магазина, который занимается розничной торговлей техники. Самые широкие права для работы с базой находятся у администратора. Он занимается созданием и наполнением БД, следит за правильным ее функционированием. Бухгалтерия, которая отвечает за закупки товара, учет позиций на складе и т.д, имеет уже ограниченные права. Бухгалтер должен иметь доступ только к своей части базы. Он ни в коем случае не должен иметь возможность удалять или модифицировать информацию, которая не относится к его области деятельности. Консультант-продавец торгового зала должен иметь возможность поиска по базе информации о наличии и количестве товара, а также месте его хранения. Данный пользователь БД должен иметь возможность только просматривать отдельные таблицы базы данных и ни в коем случае не должен иметь прав на внесения изменений в них. Правильное делегирование привилегий обеспечивает стабильность работы.

Обеспечение резервного копирования и восстановления – данная функция позволяет избежать полной потери информации. Безвозвратно теряется только та часть, которая была внесена в базу с момента последнего резервирования. Адекватный выбор интервала резервного копирования позволяет в значительной степени снизить риски.

Обеспечение распределенного хранения и взаимодействия – удобная функция, позволяющая хранить информацию на разных серверах. Например, база данных РЖД о времени отправления локальных электричек может быть распределена по региональным дата-центрам.

Поддержка универсального языка управления – все базы данных, доступные для широкой аудитории, поддерживают язык SQL.

2. Проектирование базы данных

2.1 Вступительное слово

Если уважаемый читатель подойдет к полке с технической литературой в книжном магазине и посмотрит на учебники, посвященные простым и интуитивно понятным инструментам, входящим в состав MS Office (таким как MS Word или MS Excel), он очень удивится, если обратит внимание на объем данных изданий. Например, учебник, посвященный формулам в MS Excel может быть объемом более 700 страниц, а самоучитель по текстовому процессору MS Word, который является достаточно простым для самостоятельного изучения, может быть объемом до 500 страниц. То же касается и книг, посвященных разным аспектам, касающимся СУБД. Если мы посчитаем суммарный объем книг, необходимых для составления представления о СУБД нам придется прочитать более 3000 - 5000 страниц текста. К счастью, это никак не связано со сложностью данной темы. В мире писателей является престижным выпускать толстые и очень объемные фолианты. В то время как для понимания СУБД можно обойтись куда менее значительными объемами литературы. Авторы данного короткого учебного пособия ставили перед собой задачу максимально лаконично и просто объяснить азы использования СУБД и языка SQL. Конечно же, изложенного здесь материала недостаточно для профессиональной работы с базами данных, но целью данного издания является формирование костяка знаний о СУБД, на котором читатель будет собирать необходимые ему сведения в зависимости от используемых на практике программных пакетов и необходимой глубины познаний.

2.2 Как создать базу данных

Давайте спроектируем базу студентов РГГМУ, которая будет использоваться сотрудниками деканата для учета всех обучающихся на факультете. В первую очередь, нам необходимо определиться с теми данными которые будут храниться в таблице (для простоты мы будем проектировать базу данных, состоящую из одной таблицы):

1. Номер студенческого билета;
2. Фамилия и имя;
3. Номер группы, в которой учится студент;
4. Телефон, по которому сотрудники деканата могут связаться со студентом.

Пример таблицы приведён на рисунке 2.1.



Рисунок 2.1 – Таблица учета студентов

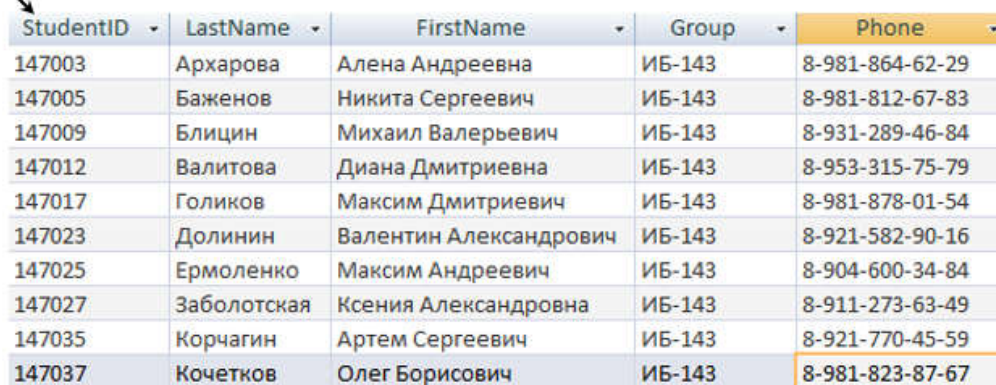
В среде специалистов по базам данных принято горизонтальные записи таблицы называть записями, а вертикальные - полями. В названии полей лучше не использовать пробелы и русский язык. Это обусловлено особенностями языка SQL. Если в заголовке поля были

использованы пробелы, то такое поле в тексте SQL запроса обособливается квадратными скобками, что не удобно:

При проектировании баз данных используются несколько простых, но очень важных правил, нарушение которых приведет к логическим и функциональным ошибкам в работе с СУБД:

1. **Целостность** – каждая таблица должны иметь уникальный ключ, который может состоять из одного или нескольких полей. Его значение не может повторяться в этой таблице (рисунок 2.2). С помощью этого ключа СУБД унифицирует данные, хранящиеся в записях. Если допустить, что ключ повторяется, то это приведет к логической ошибке: один студенческий билет числится за разными людьми или сведения об одном человеке дублируются. Обычно, СУБД следит за тем, чтобы одинаковые значения ключевого поля не вносились пользователем. Если пользователь попытается ввести в ключевое поле значение, которое уже содержится в поле, СУБД выдаст ошибку.

Ключ



StudentID	LastName	FirstName	Group	Phone
147003	Архарова	Алена Андреевна	ИБ-143	8-981-864-62-29
147005	Баженов	Никита Сергеевич	ИБ-143	8-981-812-67-83
147009	Блицин	Михаил Валерьевич	ИБ-143	8-931-289-46-84
147012	Валитова	Диана Дмитриевна	ИБ-143	8-953-315-75-79
147017	Голиков	Максим Дмитриевич	ИБ-143	8-981-878-01-54
147023	Долинин	Валентин Александрович	ИБ-143	8-921-582-90-16
147025	Ермоленко	Максим Андреевич	ИБ-143	8-904-600-34-84
147027	Заболотская	Ксения Александровна	ИБ-143	8-911-273-63-49
147035	Корчагин	Артем Сергеевич	ИБ-143	8-921-770-45-59
147037	Кочетков	Олег Борисович	ИБ-143	8-981-823-87-67

Рисунок 2.2 – Таблица с указанием ключевого поля

2. **Упорядоченность** – данные одного вида должны находиться в одном столбце (рисунок 2.3). Допустим, секретарь

делает ошибку и вносят имя студента в столбец, предназначенный для фамилии, или номер группы в столбец, предназначенный для имени. При поиске по фамилии в столбце LastName правильный ответ не будет получен, так как одна фамилий будет находиться в другом столбце. Частично можно решить данную проблему путем указания типа данных поля, что не позволит вводить телефон (цифры) в поле, предназначенное для хранения фамилии (строка).

StudentID	LastName	FirstName	Group	Phone
147003	Архарова	Алена Андреевна	ИБ-143	8-981-864-62-29
147005	Баженов	Никита Сергеевич	ИБ-143	8-981-812-67-83
147009	Михаил Валерьевич	Блицин	ИБ-143	8-931-289-46-84
147012	ИБ-143	Диана Дмитриевна	Валикова	8-953-315-75-79
147017	Голиков	Максим Дмитриевич	ИБ-143	8-981-878-01-54
147023	Долинин	Валентин Александрович	ИБ-143	8-921-582-90-16
147025	Ермоленко	ИБ-143	Максим Андреевич	8-904-600-34-84
147027	Заболотская	Ксения Александровна	ИБ-143	8-911-273-63-49
147035	Корчагин	Артем Сергеевич	ИБ-143	8-921-770-45-59
147037	Кочетков	Олег Борисович	ИБ-143	8-981-823-87-67

Рисунок 2.3 – Неправильное занесение сведений в таблицу

А что делать если у студента два телефонных номера? Добавлять дополнительные поля, как это указано на рисунке 2.4? Но у студента может быть 5 телефонов, а постоянно менять структуру таблицы занятие сложное и не рациональное. Добавить еще одну запись, как это показано на рисунке 2.5? Но в этом случае мы нарушаем первое правило – у нас появляется повторяющиеся значение в ключевом поле. Мы, конечно, можем добавить еще одно поле, которое назначить ключевым вместо «StudentID», назвав его PrimaryKey (рисунок 2.6), но в этом случае у нас возникает дублирование информации.

StudentID	LastName	FirstName	Group	Phone	Phone2
147003	Архарова	Алена Андреевна	ИБ-143	8-981-864-62-29	
147005	Баженов	Никита Сергеевич	ИБ-143	8-981-812-67-83	
147009	Блицин	Михаил Валерьевич	ИБ-143	8-931-289-46-84	
147012	Валитова	Диана Дмитриевна	ИБ-143	8-953-315-75-79	
147017	Голиков	Максим Дмитриевич	ИБ-143	8-981-878-01-54	
147023	Долинин	Валентин Александрович	ИБ-143	8-921-582-90-16	8-953-320-46-11
147025	Ермоленко	Максим Андреевич	ИБ-143	8-904-600-34-84	
147027	Заболотская	Ксения Александровна	ИБ-143	8-911-273-63-49	
147035	Корчагин	Артем Сергеевич	ИБ-143	8-921-770-45-59	
147037	Кочетков	Олег Борисович	ИБ-143	8-981-823-87-67	

Рисунок 2.4 – Два поля для ввода телефонных номеров

StudentID	LastName	FirstName	Group	Phone
147003	Архарова	Алена Андреевна	ИБ-143	8-981-864-62-29
147005	Баженов	Никита Сергеевич	ИБ-143	8-981-812-67-83
147009	Блицин	Михаил Валерьевич	ИБ-143	8-931-289-46-84
147012	Валитова	Диана Дмитриевна	ИБ-143	8-953-315-75-79
147017	Голиков	Максим Дмитриевич	ИБ-143	8-981-878-01-54
147023	Долинин	Валентин Александрович	ИБ-143	8-921-582-90-16
147025	Ермоленко	Максим Андреевич	ИБ-143	8-904-600-34-84
147027	Заболотская	Ксения Александровна	ИБ-143	8-911-273-63-49
147035	Корчагин	Артем Сергеевич	ИБ-143	8-921-770-45-59
147037	Кочетков	Олег Борисович	ИБ-143	8-981-823-87-67
147037	Кочетков	Олег Борисович	ИБ-143	9-921-437-84-12

Рисунок 2.5 – Дублирование записей

PrimaryKey	StudentID	LastName	FirstName	Group	Phone
1	147005	Баженов	Никита Сергеевич	ИБ-143	8-981-812-67-83
2	147003	Архарова	Алена Андреевна	ИБ-143	8-981-864-62-29
3	147009	Блицин	Михаил Валерьевич	ИБ-143	8-931-289-46-84
4	147012	Валитова	Диана Дмитриевна	ИБ-143	8-953-315-75-79
5	147017	Голиков	Максим Дмитриевич	ИБ-143	8-981-878-01-54
6	147023	Долинин	Валентин Александрович	ИБ-143	8-921-582-90-16
7	147025	Ермоленко	Максим Андреевич	ИБ-143	8-904-600-34-84
8	147027	Заболотская	Ксения Александровна	ИБ-143	8-911-273-63-49
9	147035	Корчагин	Артем Сергеевич	ИБ-143	8-921-770-45-59
10	147037	Кочетков	Олег Борисович	ИБ-143	8-981-823-87-67
11	147037	Кочетков	Олег Борисович	ИБ-143	8-921-782-60-56

Рисунок 2.6 – Избыточное хранение информации

И здесь нам на помощь приходит третье правило:

3. **Нормализация** – удаление повторяющихся данных, путем их переноса в новые таблицы (рисунок 2.7). Обратите внимание, у нас

теперь в базе данных две таблицы. В первой таблице хранится информация о номере студенческого билета, фамилии и имени студента, а также номер группы. Во второй таблице хранится номер студенческого билета и номер телефона. В данной таблице ключевым является новое поле «Code», его значения не повторяются, а вот номера студенческих в этой таблице повторяться могут. Таким образом, минимальное дублирование данных позволяет повысить эффективность работы СУБД.

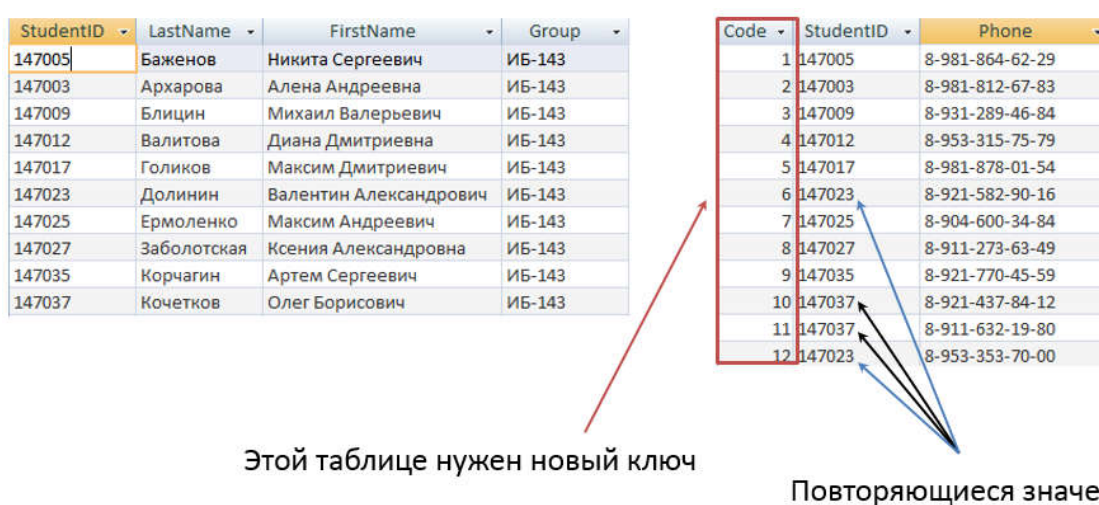


Рисунок 2.7 – Нормализация

Между нашими таблицами возникает связь, которая может быть двух видов:

1. «Один к одному» - одной записи таблицы соответствует одна запись другой таблицы;
2. «Один ко многим» - одной записи соответствует несколько записей в другой (рисунок 2.8).

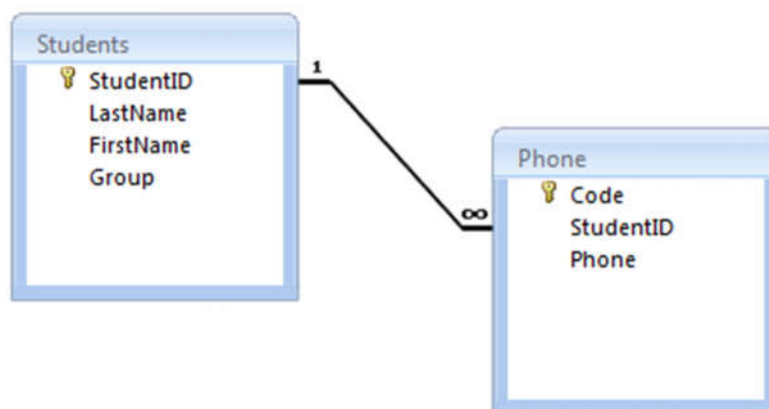


Рисунок 2.8 – Связь один ко многим

Связь между таблицами называются отношениями (от английского relations). Отсюда название «**реляционные** базы данных».

Основными этапы создания базы данных являются:

1. **Формулировка цели.** Необходимо определить какие данные будет содержать наша база, кто с ней будет работать и какие запросы будут выполняться;
2. **Распределение данных по таблицам.** Нужно логично продумать структуру так, чтобы в дальнейшем не приходилось ничего менять. *Данный шаг является основополагающим при проектировании базы данных;*
3. **Задание первичных ключей.** Необходимо решить, какие поля будут ключевыми;
4. **Настройка связей между таблицами** – продумать взаимоотношения таблиц.

Тренироваться в написании своей первой база данных удобно с помощью инструмента, входящего в состав пакета Microsoft Office, который называется MS Access. Это реляционная СУБД, данные в которой представляются простейшими двумерными объектами (таблица. 2.1).

Таблица 2.1 – Элементы MS Access

Элементы	Назначение
Таблицы	Хранение данных
Запросы	Управление данными
Формы	Вывод данных на экран
Отчеты	Печать информации

2.3 Практическая часть

Давайте рассмотрим процесс создания БД студентов, описанной выше в разделе 2.2, по шагам:

1. Для начала работы с MS Access, найдите его ярлык в списке программ и запустите его (рисунок 2.9);



Рисунок 2.9 – Ярлык программы MS Access

2. В открывшемся меню следует выбрать «Пустая база данных рабочего стола», ввести имя БД и нажать «Создать»;
3. MS Access работает в нескольких режимах (рисунок 2.10):

Режим таблицы – предназначен для просмотра и наполнения;

Конструктор – используется для проектирования таблиц и назначения типов данных для полей.

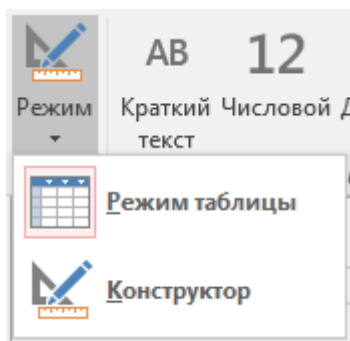


Рисунок 2.10 – Режимы работы MS Access

Давайте переключимся в режим конструктора, введем имя таблицы «Students» и нажмем «Ок»; Обратите внимание, что при создании таблицы, ключевое поле создается автоматически, т.к. является основным (рисунок 2.11).

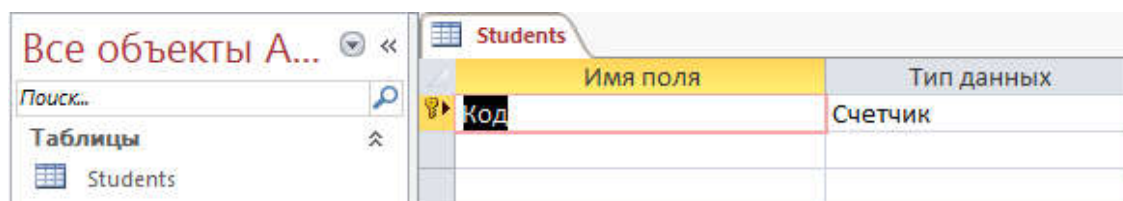


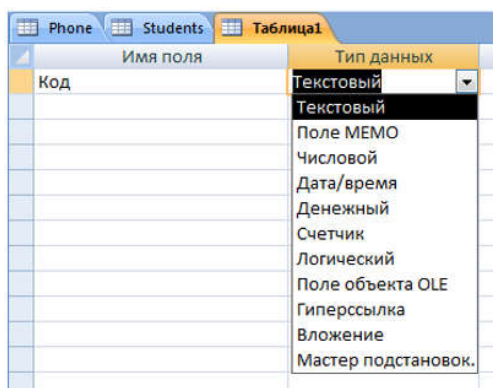
Рисунок 2.11 – Создание новой таблицы

4. Заполним имя полей согласно описанной концепции (рисунок 2.12), заменив поля «LastName» и «FirstName» полем «FIO» и добавив в него поле «Photo», которое будет содержать фотографию студента.

	Имя поля	Тип данных
🔑	StudentID	Числовой
	FIO	Короткий текст
	Group	Короткий текст
	Photo	Поле объекта OLE

Рисунок 2.12 – Заполнение имен полей

5. Обратите внимание на то, что каждому создаваемому полю автоматически присваивается тип данных «Короткий текст». При необходимости его можно изменить (рисунок 2.13) или настроить (рисунок 2.14)



Текстовый	Алфавитно-цифровые данные	До 255 байт
Мемо	Алфавитно-цифровые данные большого объема	До 64 Кбайт
Числовой	Числовые данные	1,2,4,8 байт
Дата/Время	Дата и время	8 байт
Денежный	Числовые данные с 4 точками после запятой	8 байт
Счетчик	Уникальное длинное целое, генерируемое ACCESS при запросе нового значения	4 байта
Логический	Логические данные	1 байт
Объект OLE	Всевозможные OLE-объекты из приложений <u>Windows</u>	До 1 Гбайт

Рисунок 2.13 – Изменение типа данных полей

Общие	Подстановка
Размер поля	255
Формат поля	
Маска ввода	
Подпись	
Значение по умолчанию	
Правило проверки	
Сообщение об ошибке	
Обязательное поле	Нет
Пустые строки	Да
Индексированное поле	Нет
Сжатие Юникод	Да
Режим IME	Нет контроля
Режим предложений IME	Нет
Выравнивание текста	Общее

Рисунок 2.14 – Свойства поля

6. Сохраним таблицу (рисунок 2.15), нажав правую кнопку мыши на заголовке, и перейдем в режим таблицы (рисунок 2.10).

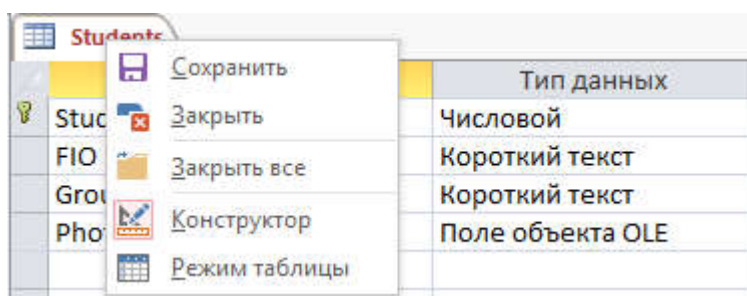
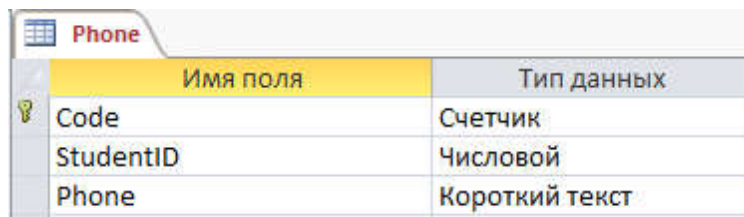


Рисунок 2.15 – Сохранение таблицы

7. Теперь по аналогии необходимо создать вторую таблицу, где будут храниться телефоны (рисунок 2.7). В ней ключевым полем будет «Code», а тип поля «Phone» – короткий

текст, чтобы пользователь мог вводить в это поле не только цифры, но и символы «+» и «-» (рисунок 2.16):



Имя поля	Тип данных
Code	Счетчик
StudentID	Числовой
Phone	Короткий текст

Рисунок 2.16 – Таблица «Phone»

8. Две таблицы созданы. Необходимо внести в них часть информации. Для этого используйте файл «Исходные данные.docx». Вставьте в таблицы сведения о первых трех студентах (поля «FIO» и «Group» заполняются копированием ячеек, а «Photo» – перетаскиванием фотографии из папки Photo). Обратите внимание, что у студента Ермолова два телефона, запись с его номером студенческого повторяется в таблицу Phone дважды. Результат должен выглядеть так (рисунок 2.17):

StudentID	FIO	Group	Photo
137115	Бачище Леонтий Яковлевич	ИБ-х49	Bitmap Image
137116	Воронин Артур Русланович	ИБ-х49	Bitmap Image
137117	Ермолов Михаил Дмитриевич	ИБ-х49	Bitmap Image

Code	StudentID	Phone
1	137115	89213070248
2	137116	89213070249
3	137117	89213070250
4	137117	89213070330

Рисунок 2.17 – Образцы заполненных таблиц

9. Пришло время связать таблицы. Для этого их надо закрыть, перейти на закладку «Работа с базами данных», выбрать «Схема данных», добавить обе таблицы, захватить мышкой поле «StudentID» в одной таблице и перетащить его на «StudentID» во

второй. Должно появиться окно «Изменение связей», где нужно поставить галки как, это показано на рисунке 2.18:

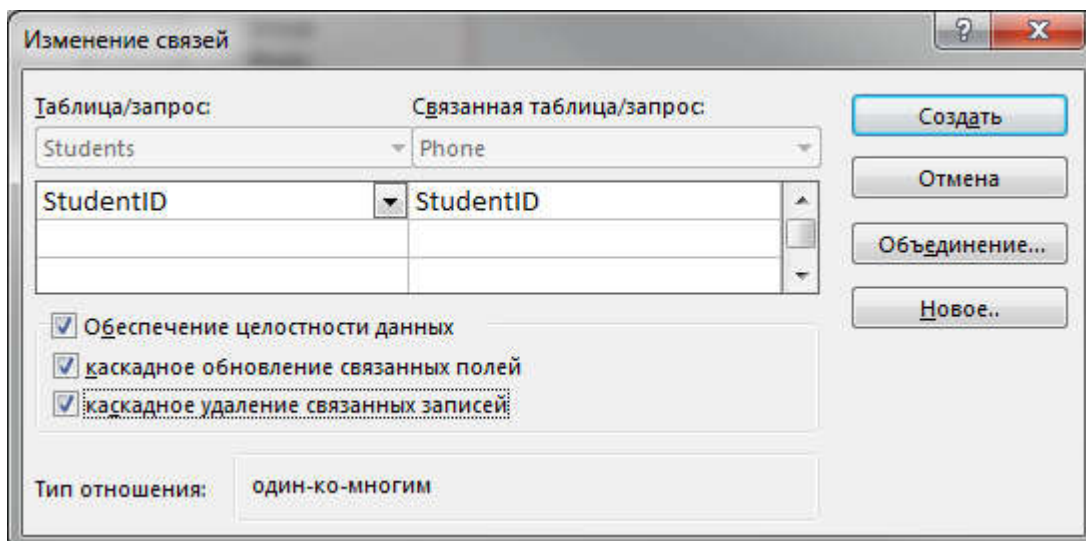


Рисунок 2.18 – окно «Изменение связей»

Выбор пункта «Обеспечение целостности данных» позволяет осуществлять синхронное обновление и удаление связанных записей.

После нажатий кнопки «Создать» откройте таблицу «Students». В левой части каждой записи появился «+». Нажав на него (рисунок 2.19), мы увидим связанную информацию. Это означает, что наша база функционирует правильно.

StudentID	FIO	Group	Photo	
+	137115	Бачище Леонтий Яковлевич	ИБ-х49	Bitmap Image
-	137116	Воронин Артур Русланович	ИБ-х49	Bitmap Image
	Code	Phone	Щелкните	
		2 89213070249		
*	(№)			
+	137117	Ермолов Михаил Дмитриевич	ИБ-х49	Bitmap Image

Рисунок 2.19 – Связанная информация

10. Остался последний штрих – создать интерфейс. Перейдите во вкладку «Создание» и выберите «Мастер форм». Добавьте поля, которые следует отобразить: «StudentID», «FIO», «Group» и «Photo» из таблицы «Students» и «Phone» из таблицы «Phone» (рисунок 2.20), нажмите «Готово» (рисунок 2.21).

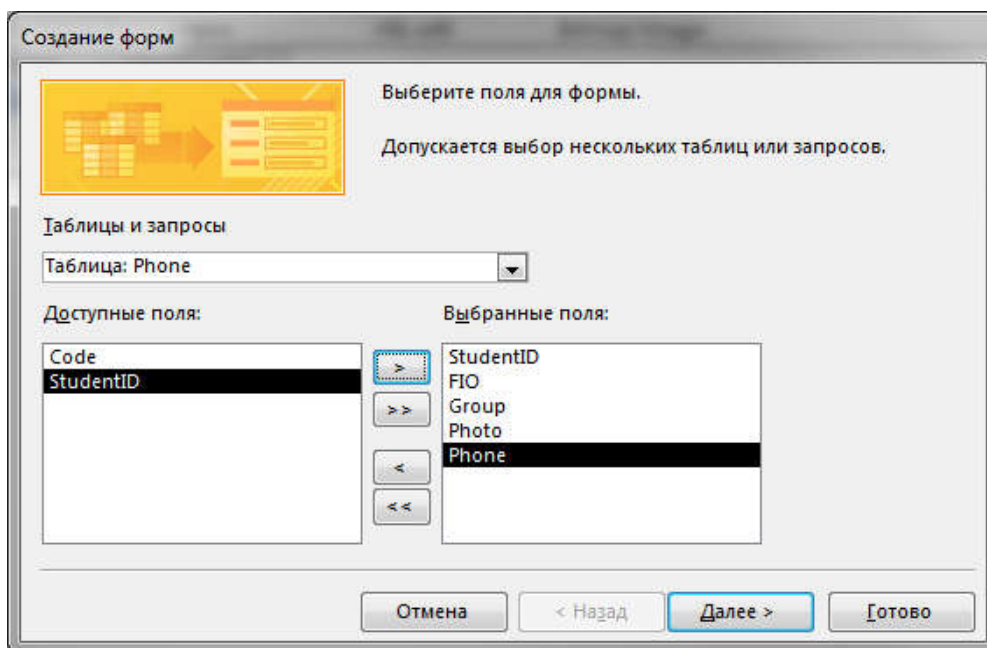


Рисунок 2.20 – Выбор необходимых данных для представления в форме

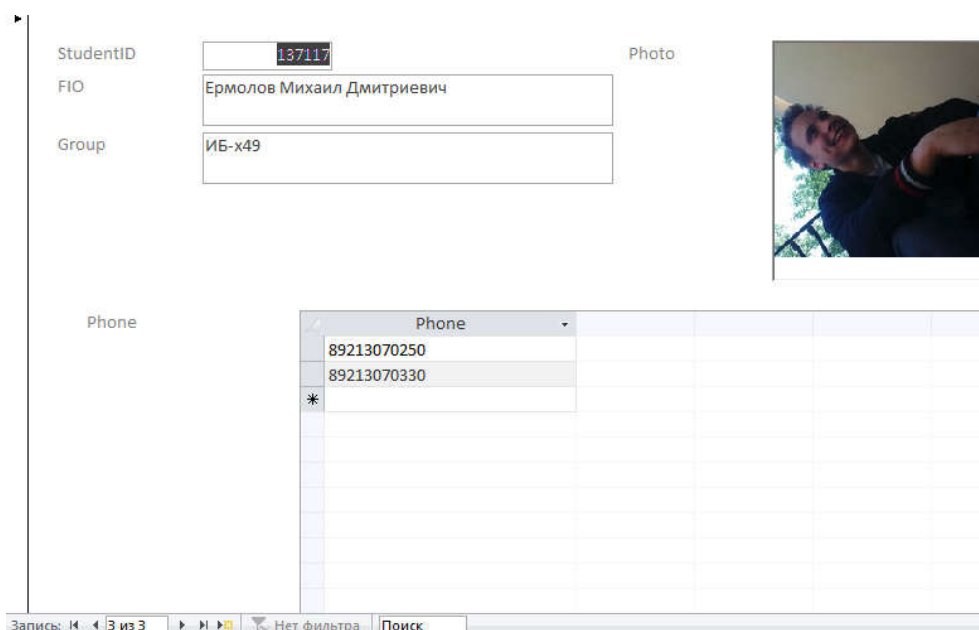


Рисунок 2.21 – Готовая форма

Используя меню внизу формы можно:

1. Создать пустой бланк для записи сведений о новом студенте;
2. Перейти к следующей/предыдущей записи;
3. Перейти в начало/конец списка;
4. Найти студента по его имени или фамилии.

Поздравляем, вы только что успешно создали первую базу данных. Предлагаем закрепить материал и дополнить базу функционально. Попробуйте доработать ее так, чтобы ее могли использовать следующие подразделения университета:

1. Отдел режима: номер карты пропуска, замечания о нарушениях (если есть);
2. Отдел кадров: паспорт, информация о аттестате и сертификатах;
3. Информационный отдел: логин, e-mail;
4. Библиотека: книги на руках;
5. Общежитие: номер комнаты проживания.

3. Язык SQL

3.1 Определение

В начале 70-х годов прошлого века компания IBM разрабатывала первые системы управления реляционными базами данных, а вместе с ними и язык, предназначенный для работы в этих системах. Экспериментальная версия называлась SEQUEL — Structured English QUERy Language (структурированный английский язык запросов). Однако официальная версия была названа короче — SQL (Structured Query Language). Он задумывался как простой язык, близкий к естественному. Предполагалось, что эта близость сделает SQL средством, доступным для широкого круга пользователей, а не только программистов [1].

3.2 Базовый синтаксис языка SQL

3.2.1 Диалекты SQL

Несмотря на существование международного стандарта, многие компании, занимающиеся разработкой СУБД (например, Oracle, Sybase, Microsoft, MySQL AB), вносят изменения в язык SQL, применяемый в разрабатываемой СУБД, тем самым отступая от стандарта. Таким образом, появляются специфичные для каждой конкретной СУБД

диалекты языка SQL (табл.1). Данные различия не касаются основных операторов SQL, а в основном синтаксиса и процедурных расширений.

Таблица 1. Диалекты языка SQL

СУБД	Краткое название	Расшифровка
InterBase/Firebird	PSQL	Procedural SQL
IBM DB2	SQL PL (англ.)	SQL Procedural Language (расширяет SQL/PSM); также в DB2 хранимые процедуры могут писаться на обычных языках программирования: Си, Java и т. д.
MS SQL Server/ Sybase ASE	T-SQL	Transact-SQL
MySQL	SQL/PSM	SQL/Persistent Stored Module
Oracle	PL/SQL	Procedural Language/SQL (основан на языке Ada)
PostgreSQL	PL/pgSQL	Procedural Language/PostgreSQL Structured Query Language (очень похож на Oracle PL/SQL)

3.2.2 Операторы SQL

Операторы [2] SQL делятся на:

- операторы определения данных (*Data Definition Language, DDL*)

CREATE создает объект БД (саму базу, таблицу, представление, пользователя и т. д.)

ALTER изменяет объект

DROP удаляет объект

- операторы манипуляции данными (*Data Manipulation Language, DML*)

SELECT считывает данные, удовлетворяющие заданным условиям

INSERT добавляет новые данные

UPDATE изменяет существующие данные

DELETE удаляет данные

- операторы определения доступа к данным (*Data Control Language, DCL*)

GRANT предоставляет пользователю (группе) разрешения на определенные операции с объектом

REVOKE отзывает ранее выданные разрешения

DENY задает запрет, имеющий приоритет над разрешением

- операторы управления транзакциями (*Transaction Control Language, TCL*)

COMMIT применяет транзакцию.

ROLLBACK откатывает все изменения, сделанные в контексте текущей транзакции.

SAVEPOINT делит транзакцию на более мелкие участки.

Транзакция – совокупность каких-либо операций над данными.

Когда все операции были выполнены, транзакция считается успешной,

если хотя бы одна из операций была неудачной, транзакция не выполняется и все откатывается в исходное состояние.

Пример: Необходимо перевести с банковского счёта номер 5 на счёт номер 7 сумму в 10 денежных единиц. Этого можно достичь, к примеру, приведённой последовательностью действий:

- Начать транзакцию

прочитать баланс на счёту номер 5 уменьшить баланс на 10 денежных единиц сохранить новый баланс счёта номер 5 прочитать баланс на счёту номер 7 увеличить баланс на 10 денежных единиц сохранить новый баланс счёта номер 7

- Окончить транзакцию

Эти действия представляют из себя логическую единицу работы «перевод суммы между счётами», и таким образом, являются транзакцией. Если прервать данную транзакцию, к примеру, в середине, и не аннулировать все изменения, легко оставить владельца счёта номер 5 без 10 единиц, тогда как владелец счёта номер 7 их не получит.

Базовый синтаксис операторов DML (выбор, обновление, добавление и удаление данных в таблицы)

Оператор SELECT – выбор данных

SELECT * FROM table_name WHERE (выражение) [order by field_name [desc][asc]]

Эта команда ищет все записи в таблице *table_name*, которые удовлетворяют выражению *выражение*.

Если записей несколько, то при указанном предложении *order by* они будут отсортированы по тому полю, имя которого *field_name* записывается правее этого ключевого слова (если задано слово *desc*, то

упорядочивание происходит по убыванию). В предложении *order by* могут также задаваться несколько полей.

Особое значение имеет символ звёздочка *. Он предписывает, что из отобранных записей следует извлечь все поля, когда будет выполнена команда получения выборки. С другой стороны, вместо звездочки можно через запятую непосредственно перечислить имена полей, которые требуют извлечения. Но чаще всего все же пользуются именно *.

Пример: нам надо найти в таблице *db_quest*, содержащей сообщения в гостевой книге, все записи, которые оставил определенный пользователь *user*. В этой таблице поле, содержащее имена пользователей, называется *name*

```
SELECT * FROM db_guest WHERE name='user'
```

Теперь представим, что у нас есть база сообщений гостевой книги, и нам надо отсортировать сообщения по времени, когда они были оставлены.

Для этого одно из полей таблицы базы должно содержать время записи сообщения *time*.

Сортировка по какому-либо столбцу осуществляется при помощи конструкции *order by*.

В нашем случае, при учете, что более "свежие" сообщения будут сортироваться верхними:

```
SELECT * FROM db_guest ORDER BY time DESC
```

В запросах со сложным условием WHERE условия заключаются в скобки. Например,

```
SELECT * FROM db_guest WHERE (name='user') AND (age=26)
```

Полный синтаксис оператора SELECT в MySQL (подробности в соответствующих справочниках):

```
SELECT [STRAIGHT_JOIN]
```

[SQL_SMALL_RESULT] [SQL_BIG_RESULT]
[SQL_BUFFER_RESULT]
[SQL_CACHE | **SQL_NO_CACHE]**
[SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
[DISTINCT | DISTINCTROW | ALL]
 expression,...
[INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
[FROM table_references
[WHERE where_definition]
[GROUP BY {unsigned_integer | field_name | formula} [ASC | DESC],
 ...]

[HAVING where_definition]
[ORDER BY {unsigned_integer | field_name | formula} [ASC | DESC],
 ...]

[LIMIT [offset,] rows] [PROCEDURE procedure_name]
[FOR UPDATE | LOCK IN SHARE MODE]]

Оператор UPDATE – обновление данных

UPDATE tbl_name SET
 field_name1=значение1[,field_name2=значение2,
 ...] **[WHERE where_definition] [LIMIT #]**

tbl_name - задает имя таблицы, в которой будут обновляться записи.

На момент запуска команды UPDATE таблица с таким именем должна существовать в базе данных.

SET - после этого ключевого слова должен идти список полей таблицы, которые будут обновлены и непосредственно сами новые значения полей в виде:

имя поля='значение'

WHERE - задает условие отбора записей, подлежащих изменению.

LIMIT - задает максимальное количество строк, которые могут быть изменены.

Пример:

Следующий пример производит обновление поля *country* у ВСЕХ записей в таблице *users*:

```
UPDATE users SET country='Russia'
```

Следующий пример изменит название города в записях пользователей с «Ленинград» на «Санкт-Петербург»:

```
UPDATE users SET city='Санкт-Петербург' WHERE  
city='Ленинград'
```

Полный синтаксис оператора UPDATE в MySQL (подробности в соответствующих справочниках):

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
```

```
SET col_name1=expr1 [, col_name2=expr2, ...]
```

```
[WHERE where_definition]
```

```
[LIMIT #]
```

Оператор INSERT – вставка данных

```
INSERT INTO tbl_name
```

```
[(field_name,...)]VALUES(expression,...),(...),...
```

либо

```
INSERT INTO tbl_name SET
```

```
field_name=expression,field_name=expression, ...
```

либо

```
INSERT INTO tbl_name [(field_name,...)] SELECT ...
```

tbl_name - задает имя таблицы, в которую будет вставлена новая строка. На момент запуска команды INSERT таблица с таким именем должна существовать в базе данных.

INSERT ... VALUES - в этом случае в команде четко указывается порядок следования устанавливаемых полей и их значений.

Пример:

Следующая команда вставит в таблицу *users* новую запись, присвоив полям *name*, *age*, *country*, *city* значения *Evgen*, *26*, *Russia*, *Ryazan* соответственно:

```
INSERT INTO users (name, age, country, city) VALUES ('Evgen', 26, 'Russia', 'Ryazan')
```

INSERT ... SET - в этом случае в команде каждому полю, присутствующему в таблице, присваивается значение в виде имя поля='значение'.

Пример:

```
INSERT INTO users SET name = 'Evgen', age = 26, country = 'Russia', city = 'Ryazan'
```

INSERT ... SELECT - такой синтаксис позволяет внести в таблицу большое количество записей за один раз, причем из разных таблиц.

Пример:

записать в таблицу *users_new* все записи из таблицы *users*, в которых поле *country* равно «*Russia*»

```
INSERT INTO users_new SELECT * FROM users WHERE country = 'Russia'
```

Оператор DELETE – удаление данных из таблиц

```
DELETE FROM table_name [WHERE where_definition] [ORDER BY ...]
```

[LIMIT rows]

Команда DELETE удаляет из таблицы *table_name* все записи, удовлетворяющие условию *where_definition*. Если условие *WHERE where_definition* не задано, то из таблицы *table_name* удаляются ВСЕ записи.

Команда DELETE возвращает количество удаленных записей.

Следует учитывать, что если в команде DELETE не задано условие *WHERE where_definition*, то команда возвратит 0, хотя записи были удалены.

table_name - имя таблицы в базе данных, из которой будет происходить удаление строк.

LIMIT - задает максимальное количество строк, которые могут быть удалены за текущий запрос.

Пример:

следующий запрос удалит из таблицы *users* только 5 строк:

```
DELETE FROM users LIMIT 5
```

ORDER BY - задает имя поля, или имена полей через запятую, по которым происходит сортировка удаляемых записей. Эта возможность актуальна при необходимости удалить только определенное кол-во записей, отсортированных по какому-либо свойству.

Пример:

Допустим, таблица *users* состоит из 2 полей – имя (*name*) и возраст (*age*). Следующий запрос удалит из таблицы *users* запись самого молодого человека. Поскольку нам нужно удалить только одного человека, используем ограничитель LIMIT:

```
DELETE FROM users ORDER BY age LIMIT 1
```

3.3 Пример использования SQL для работы с базой студентов

Давайте рассмотрим запрос к таблице «Students» рисунок 3.1) на поиск фамилии и имени по номеру студенческого билета (рисунок 3.2).

StudentID	LastName	FirstName	Group	Phone
147003	Архарова	Алена Андреевна	ИБ-143	8-981-864-62-29
147005	Баженов	Никита Сергеевич	ИБ-143	8-981-812-67-83
147009	Блицин	Михаил Валерьевич	ИБ-143	8-931-289-46-84
147012	Валитова	Диана Дмитриевна	ИБ-143	8-953-315-75-79
147017	Голиков	Максим Дмитриевич	ИБ-143	8-981-878-01-54
147023	Долинин	Валентин Александрович	ИБ-143	8-921-582-90-16
147025	Ермоленко	Максим Андреевич	ИБ-143	8-904-600-34-84
147027	Заболотская	Ксения Александровна	ИБ-143	8-911-273-63-49
147035	Корчагин	Артем Сергеевич	ИБ-143	8-921-770-45-59
147037	Кочетков	Олег Борисович	ИБ-143	8-981-823-87-67

Рисунок 3.1 – Таблица базы данных



Рисунок 3.2 – Запрос к таблице

Читается он очень просто – выбрать значения полей «LastName» и «FirstName» из таблицы «Students», для которых значение поля «StudentID» равно 170027.

Давайте рассмотрим дипломный проект выпускника кафедры информационных технологий и систем безопасности РГГМУ, разработавшего электронную библиотечную систему – ЭБС (elib.rshu.ru). На рисунках 3.3 и 3.4 приведены интерфейс сайта и часть

главной таблицы базы данных, на основе которой работает электронная библиотека. Чтобы найти книгу по фамилии автора или по названию, пользователь должен ввести запрос или его часть в строке поиска (рисунок 3.3).

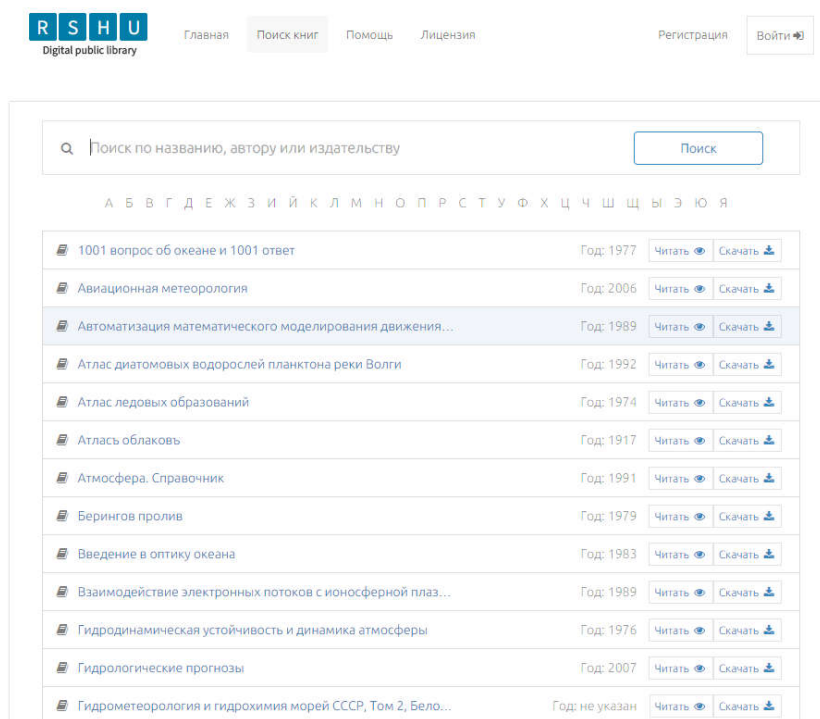


Рисунок 3.3 – Интерфейс электронной библиотеки РГГМУ

Number	Author	Label	Year
1	Барышников Н.Б.	Гидравлические сопротивления речных русел.	2003
2	Богородский В.В.	Гидроакустическая техника исследования и освое	1984
3	Михайлов Л.Е.	Гидрогеология	1985
4	ПАВЛОВ А.Н.	ГИДРОГЕОЛОГИЯ. ОБЩИЙ КРУГОВОРОТ ВОДЫ НА	1975
5	ПЛАЩЕВ А.В., ЧЕКМАРЕВ В.А.	ГИДРОГРАФИЯ СССР	1967
6	МИРОНЕНКО З.И.	ГИДРОДИНАМИКА БЕРЕГОВОЙ ЗОНЫ И ЭСТУАРИИ	1970
7	РУСИН И.Н.	ГИДРОДИНАМИЧЕСКИЕ МЕТОДЫ ДОЛГОСРОЧНОГО	1984
8	ЦЫЦЕНКО К.В.	ГИДРОЛОГИЧЕСКИЕ ОСНОВЫ ОРОСИТЕЛЬНЫХ МЕ	1990
9	Попов Е.Г.	ГИДРОЛОГИЧЕСКИЕ ПРОГНОЗЫ	1979
10	Георгиевский Ю.М., Шаночкин С.В.	Гидрологические прогнозы.	2007
11	Воскресенский К.П.	Гидрологические расчеты при проектировании о	1956
12	Коваленко В.В.	Гидрологическое обеспечение надежности строи	2009
13	Угренинов Г.Н.	Гидрологическое обеспечение народного хозяйс	1986
14	ИВАНОВ к.Е.	ГИДРОЛОГИЯ БОЛОТ	1953
15	Голченко Е.Д., Гушля А.В.	Гидрология с основами мелиорации	1989
16	Грузинов В.М.	Гидрология фронтальных зон Мирового океана	1989
17	Брейтфус Л.	ГИДРОМЕТЕОРОЛОГИЧЕСКАЯ СЛУЖБА в Карском	1916
18	Субботин А.С.	ГИДРОМЕТРИЧЕСКИЕ СООРУЖЕНИЯ	1989
19	КОВАЛЕНКО В.В.	ГИДРОМЕТРИЧЕСКОЕ ОЦЕНИВАНИЕ РЕЧНОГО СТО	1986
20	Карасев И.Ф., Шумков И.Г.	Гидрометрия	1985
21	Быков В.Д., Васильев А.В.	Гидрометрия	1977
22	Белевич М.Ю.	Гидромеханика. Основы классической теории.	2006
23	Барышников Н.Б.	Гидромеханический анализ турбулентного русло	1985
24	Догановский А.М., Малинин В.Н.	ГИДРОСФЕРА ЗЕМЛИ	2004
25	Одрова Т.В.	ГИДРОФИЗИКА ВОДОЕМОВ СУШИ	1979
26	Шишкина Л.А.	Гидрохимия	1974

Рисунок 3.4 – Часть таблицы БД электронной библиотеки РГГМУ

При написании сайта ЭБС, был сформирован SQL запрос, в который подставляется введенная пользователем строка (вместо «ТЕКСТ_ЗАПРОСА»):

```
SELECT Library.Author, Library.Label FROM Library WHERE  
Label LIKE '*ТЕКСТ_ЗАПРОСА*' OR Author LIKE  
 '*ТЕКСТ_ЗАПРОСА*'
```

Перед текстом запроса и после него стоит «*», который заменяет собой любые символы. Если пользователь не помнит точное название книги или фамилию автора, он вводит только часть. Например, если пользователь введет «метео», то система выдаст все книги, в названии которых есть это сочетание букв: гидрометеорология, метеоры, метеоролог и пр.

В этом запросе появляются две новых функции, которые мы не встречали ранее:

LIKE (похож) – просит у базы данных выдать все записи, которые похожи на введенный запрос;

OR (или) – так как система не знает ввел пользователь фамилию автора или название книги, запрос ищется как в поле «Label» (название), так и в поле «Author» (авторы).

В облачном хранилище вы найдете базу данных RSHU. Давайте напишем этот запрос к таблице «Library». Для этого нужно:

1. Открыть базу данных RSHU;
2. Перейти в раздел «Создание»;
3. Выдрать «Конструктор запросов», в появившемся окне нажмите «Заккрыть»;
4. В разделе выбора режима, выберите «Режим SQL»;
5. Напишите запрос, заменив «Текст_запроса» на «Гидро» и нажмите «Выполнить». Вы должны получить список книг, в названии которых встречается «гидро».

Попробуйте изменить запрос так, чтобы запрос искал не только по названию, но и по автору (подсказка – функция AND).

Результатом SQL запроса является таблица, которая существует только во время сеанса работы с базой данных и не присоединяется к числу таблиц, входящих в нее. Она либо содержит запрошенные данные, либо пуста, если данных, соответствующих запросу, не нашлось.

В общем случае запрос выглядит следующим образом:

SELECT СписокСтолбцов **FROM** СписокТаблиц **WHERE**
УсловиеПоиска

Вместе с оператором WHERE могут быть использованы следующие символы (рисунок 3.5):

Символ	Описание
=	Равно
<>	Не равно
<	Меньше
<=	Меньше или равно
>	Больше
>=	Больше или равно

Рисунок 3.5 – Таблица символов сравнения

Допустим, нас попросили реализовать систему оплаты, позволяющую студентам получать скидки при оплате любых товаров и услуг в кампусе университета (еда в столовых, услуги типографии, оплата общежитий). Оплата принимается как наличными, так и электронными картами студента (ЭКС), привязанными к электронному кошельку. В случае оплаты ЭКС, студент получает скидку в 50%. Баланс ЭКС можно пополнить в любом терминале на территории университета. Реализованную базу можно найти в облачном хранилище. Ее схема и

пример таблицы, содержащий сведения о счете студента приведены на рисунках 3.6 и 3.7, соответственно.

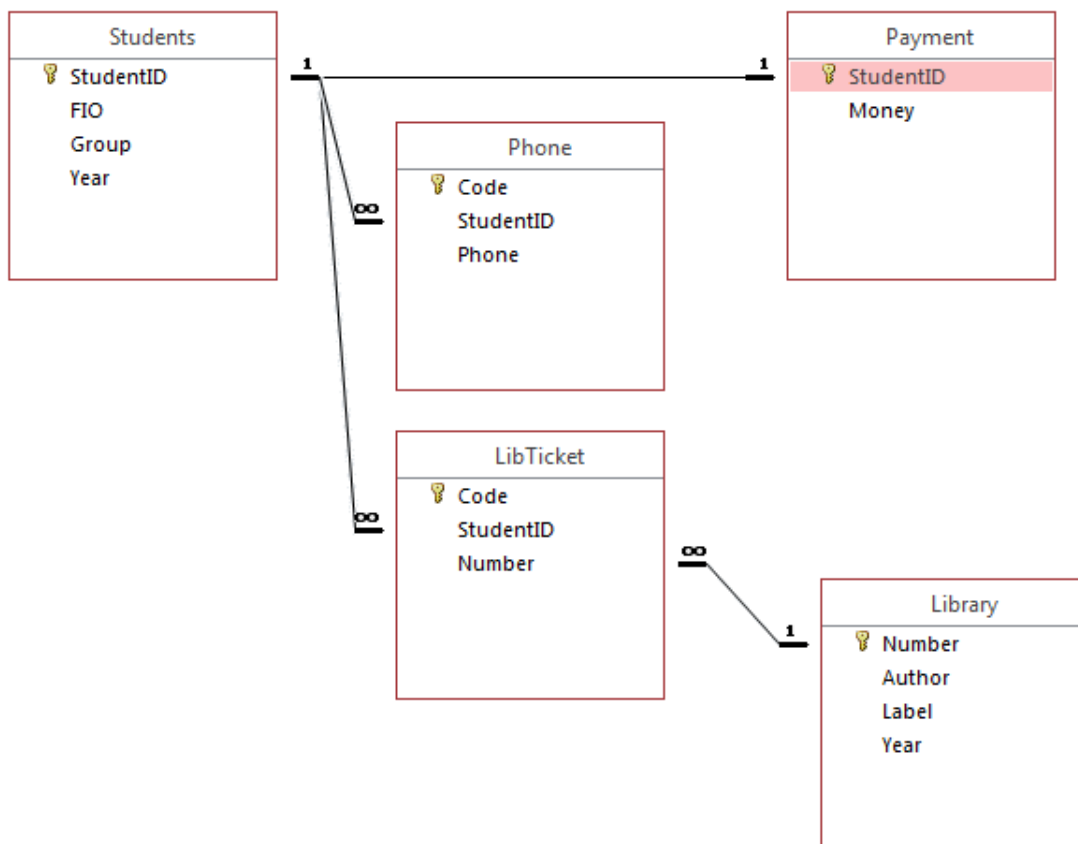


Рисунок 3.6 – Схема данных

Payment		
	StudentID	Money
+	137002	14
+	137004	440
+	137005	347
+	137006	353
+	137007	273
+	137008	30
+	137009	83
+	137010	169
+	137011	342
+	137013	370

Рисунок 3.7 – Баланс счета студентов

Давайте напишем запрос, которые проверяет баланс счёта и, если он меньше 100 рублей, отправляет уведомление (реализуем только первую часть этой процедуры):

```
SELECT Money, StudentID FROM Payment WHERE Money < 100
```

Оператор BETWEEN позволяет найти значения, лежащие в интервале. Выведем список студентов, у которых на счету больше 100, но меньше 300 рублей:

```
SELECT Money, StudentID FROM Payment WHERE Money  
BETWEEN 100 AND 300
```

Для проверки существования значения в списке используются операторы IN (в) и NOT IN (не в). Если нужно вывести всех студентов, приехавших из Петербурга, Краснодарского края и Мурманской области, необходимо выполнить следующий запрос:

```
SELECT FIO, Address FROM Students WHERE Region IN ('Санкт-  
Петербург', 'Краснодарский край', 'Мурманская область')
```

Для поиска всех, кто не из указанных регионов следует заменить «IN» на «NOT IN»:

```
SELECT FIO, Address FROM Students WHERE Region NOT IN  
( 'Санкт-Петербург', 'Краснодарский край', 'Мурманская область')
```

Язык SQL позволяет проводить вычисления. Рассмотрим несколько математических операторов [2]:

SUM (Имя поля) — возвращает сумму значений. Рассчитаем сколько всего денег хранится в системе:

```
SELECT SUM (Money) FROM Payment
```

AVG (Имя поля) — возвращает среднее арифметическое всех значений. Выведем среднюю сумму на счетах студентов:

```
SELECT AVG (Money) FROM Payment
```

MAX (Имя поля) / **MIN** (Имя поля) — возвращают максимальное / минимальное значение в столбце:

```
SELECT MAX (Money) FROM Payment
```

```
SELECT MIN (Money) FROM Payment
```

В SQL сложные запросы являются комбинацией простых SQL-запросов. Каждый простой запрос в качестве ответа возвращает таблицу, а комбинация простых запросов возвращает результат тех или иных операций над ответами на простые запросы. Давайте найдем всех студентов, у которых на счету хранится больше денег, чем в среднем по ВУЗу. Для этого нам сначала надо найти среднее арифметическое, как это было показано выше, а потом найденный результат использовать при выведении списка студентов:

```
SELECT * FROM Payment WHERE Money > (SELECT AVG  
      (Money) FROM Payment)
```

Пока мы рассматривали запросы, выводящие информацию только из одной таблицы, но что делать, если нам необходимо получить ответ, состоящий из данных нескольких таблиц. Для этих целей используют операции соединения «JOIN ... ON». Давайте выведем список фамилий студентов и сумму денег на счетах. Информация находится в таблицах Students и Payment:

```
SELECT Students.FIO, Payment.Money FROM Students INNER  
JOIN Payment ON Students.[StudentID] = Payment.[StudentID]
```

Конструкция «**INNER JOIN** Payment **ON** Students.[StudentID] = Payment.[StudentID]» связывает обе таблицы на основе общего – поля «StudentID» (как мы это делали в прошлом разделе с помощью инструментов MS Access).

Усложним запрос и выведем фамилии тех, у кого на счету больше, чем в среднем у остальных:

SELECT Students.FIO, Payment.Money FROM Students INNER JOIN Payment ON Students.[StudentID] = Payment.[StudentID] WHERE Payment.Money > (SELECT AVG (Money) FROM Payment)

Стоит отметить, что SQL-запрос не всегда начинается со слова SELECT. Если запрос должен что-то вставить в таблицу, то он начинается со слова INSERT, удалить – DELETE, изменить – UPDATE.

Как уже говорилось в начале, язык SQL был придуман в качестве инструмента общения между пользователем и СУБД. Но зачем он нужен сейчас, когда нам доступны средства графического интерфейса? Язык SQL используется для написания сайтов, работающих с базами данных. Примером может служить сайт Auto.ru (рисунок 3.8). Пользователь выбирает необходимые опции, которые он хотел бы видеть в объявлении, а сайт формирует SQL запрос, который далее отправляется в СУБД.

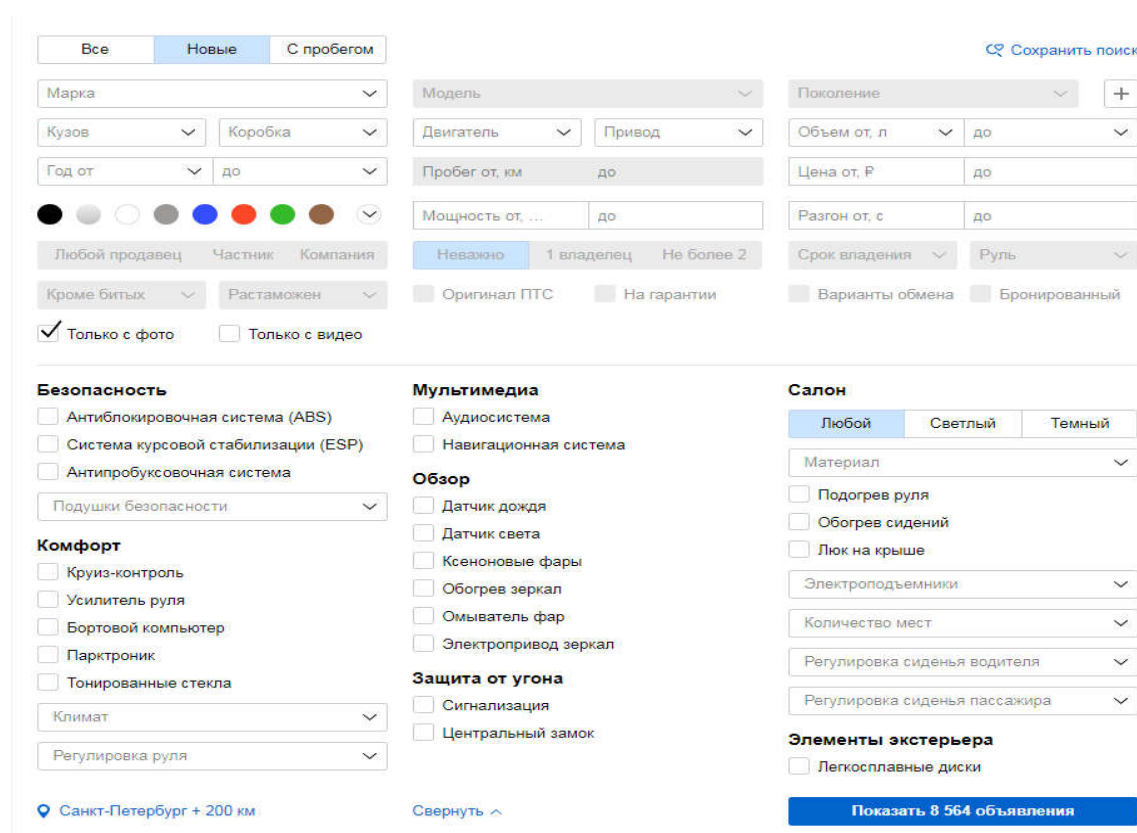


Рисунок 3.8 – Пример страницы, формирующей SQL-запрос

4. MySQL

4.1 Общие сведения о MySQL

В предыдущих разделах мы познакомились с простой СУБД MS Access и с языком SQL. Пришло время поговорить о более сложной системе управления базами данных, которая широко используется на практике – MySQL. Это свободная реляционная СУБД [3], которая первоначально разрабатывалась шведской компанией MySQL AB (1995 - 2008). Набрал популярность у пользователей благодаря своей простоте, надежности, объему (установленная версия занимала порядка 200 Мб, а сервер с этой базой данных требовал самых минимальных ресурсов) и доступности, в 2008 году компания (и проект) была поглощена американской компанией Sun Microsystems (2008-2010). С 2010 разработку и поддержку MySQL осуществляет корпорация Oracle.

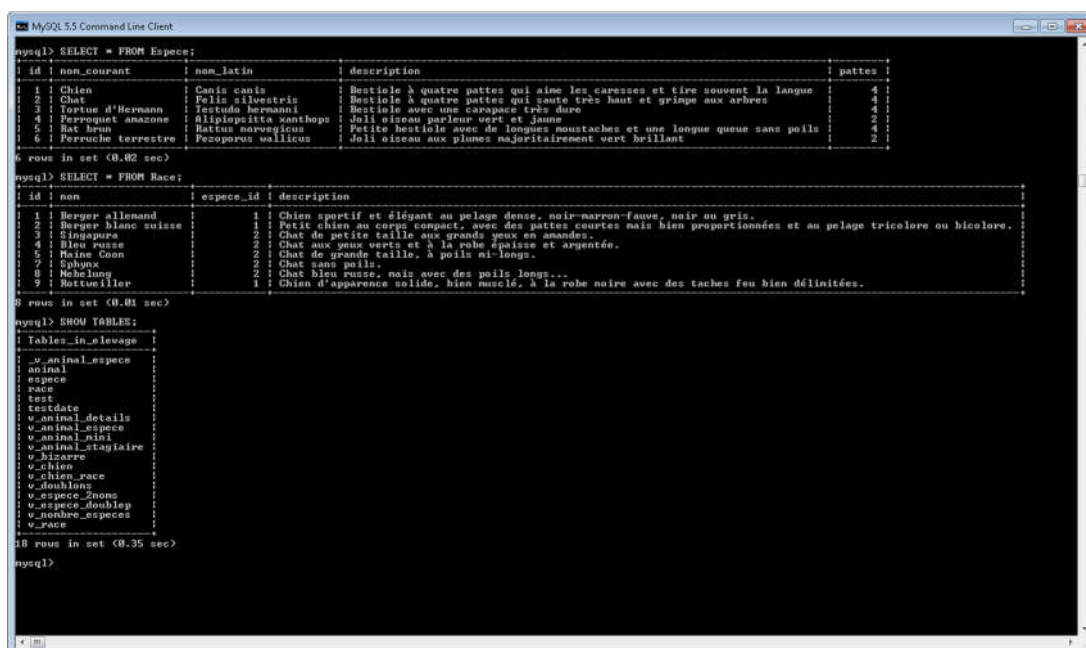
Продукт распространяется как под GNU (General Public License), так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

Хотя MySQL заявлялась как решение для малых и средних приложений, Facebook вполне успешно работает именно на основе этой СУБД [4].

При аренде хостинга для размещения своего сайта пользователь практически всегда получает предустановленную MySQL. Именно поэтому данной СУБД будет посвящен раздел настоящего учебного пособия.

Что нужно для превращения обычного компьютера в web-сервер:

1. Программа, которая будет на все http запросы отвечать, выдавая странички сайта. Часто используется ПО Apache;
2. Программа, позволяющая интерпретировать язык сценариев PHP;
3. СУБД (например, MySQL);
4. Web-интерфейс для администрирования СУБД MySQL (опционально). Интерфейс MySQL в чистом виде представлен на рисунке 4.1. Аскетично, не правда ли? С другой стороны, это позволяет СУБД быть очень лояльной к ресурсам сервера. В таком виде вполне можно работать, вводя SQL-запросы в командную строку и получая таблицы в ответ. Гораздо удобнее современному искушенному пользователю работать с графическим интерфейсом БД, который обеспечивается установкой phpMyAdmin (рисунок 4.2). Он интуитивно понятен большинству пользователей, что обеспечивает минимальное время подготовительной работы и привыкания перед началом полноценного использования системы.



```
mysql> SELECT * FROM Espece;
+----+-----+-----+-----+
| id | non_courant | nom_latin | description | pattes |
+----+-----+-----+-----+
| 1 | 1 | Canis canis | Bestiole à quatre pattes qui aime les caresses et lève souvent la langue | 4 | |
| 2 | 1 | Felis silvestris | Bestiole à quatre pattes qui saute très haut et grimpe aux arbres | 4 |
| 3 | 1 | Testudo hermanni | Bestiole avec une carapace très dure | 4 |
| 4 | 1 | Perrequet amazone | Joli oiseau parlant vert et jaune | 2 |
| 5 | 1 | Rat brun | Rattus norvegicus | Petite bestiole avec de longues moustaches et une longue queue sans poils | 2 |
| 6 | 1 | Perruche terrestre | Pezoporus wallicus | Joli oiseau aux plumes majoritairement vert brillant | 2 |
+----+-----+-----+-----+
6 rows in set (0.02 sec)

mysql> SELECT * FROM Race;
+----+-----+-----+-----+
| id | non | espece_id | description |
+----+-----+-----+-----+
| 1 | 1 | Berger allemand | Chien sportif et élégant au pelage dense, noir-marron-jaune, noir ou gris. |
| 2 | 1 | Berger blanc suisse | Petit chien au corps compact, avec des pattes courtes mais bien proportionnées et au pelage tricolore ou bicolore. |
| 3 | 1 | Shogapura | Chat de petite taille aux grands yeux en amandes. |
| 4 | 1 | Bleu russe | Chat aux yeux verts et à la robe soignée et argentée. |
| 5 | 1 | Maine Coon | Chat de grande taille, à poils ni-longs. |
| 6 | 1 | Sphinx | Chat sans poils. |
| 7 | 1 | Nebelung | Chat bleu russe, mais avec des poils longs... |
| 8 | 1 | Rottweiler | Chien d'apparence solide, bien musclé, à la robe noire avec des taches feu bien délimitées. |
+----+-----+-----+-----+
8 rows in set (0.01 sec)

mysql> SHOW TABLES;
+-----+
| Tables_in_oleveage |
+-----+
| v_animal_espece |
| animal |
| espece |
| race |
| test |
| testdate |
| v_animal_details |
| v_animal_espece |
| v_animal_midi |
| v_animal_stagiaire |
| v_bicarre |
| v_chien |
| v_chien_race |
| v_double |
| v_espece_2noms |
| v_espece_doublep |
| v_nombre_especes |
| v_race |
+-----+
18 rows in set (0.35 sec)

mysql>
```

Рисунок 4.1 – Интерфейс MySQL

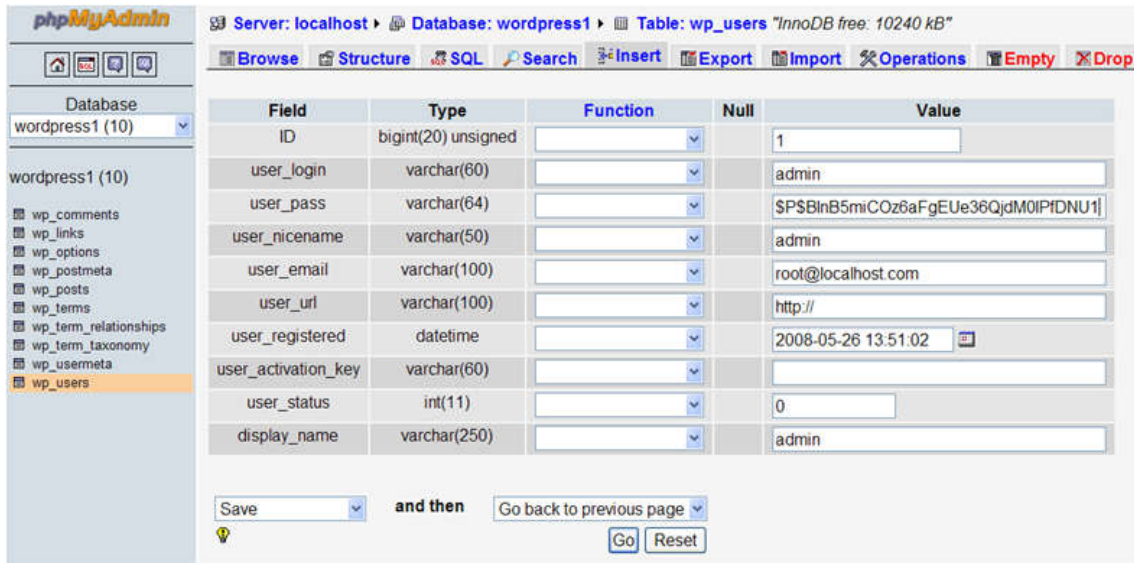


Рисунок 4.2 – Интерфейс phpMyAdmin

Давайте запустим виртуальную машину, находящуюся в облачном хранилище, и поработаем с MySQL. В контейнере виртуальной машины вы найдете комплект разработчика Denwer [5], объединяющий в себе все необходимое ПО для разработки сайтов и работы с СУБД MySQL:

Для того, чтобы начать работы с Denwer, запустите приложение «Start Denwer», откройте браузер Opera, находящийся на рабочем столе виртуальной машины, и введите адрес <http://localhost/tools/phpmyadmin/>

4.2 Начало работы с системой

Создадим пользователей и назначим им привилегии:

1. Перейдите в закладку «Пользователи» и нажмите «Добавить пользователя» (рисунок 4.3);

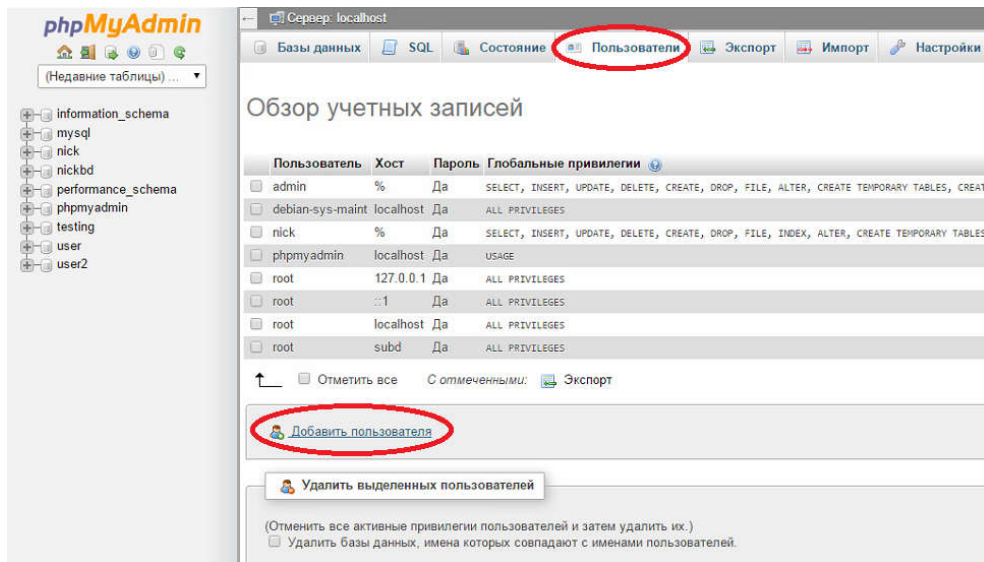


Рисунок 4.3 – Меню добавления пользователей

2. Введите имя пользователя (обычно, имя чувствительно к регистру букв) и пароль (рисунок 4.4). Ниже можно указать те функции, которые будут доступны пользователю. Например, если нужно дать пользователю возможность только просматривать данные, стоит выбрать только пункт «SELECT» в таблице «Данные»;

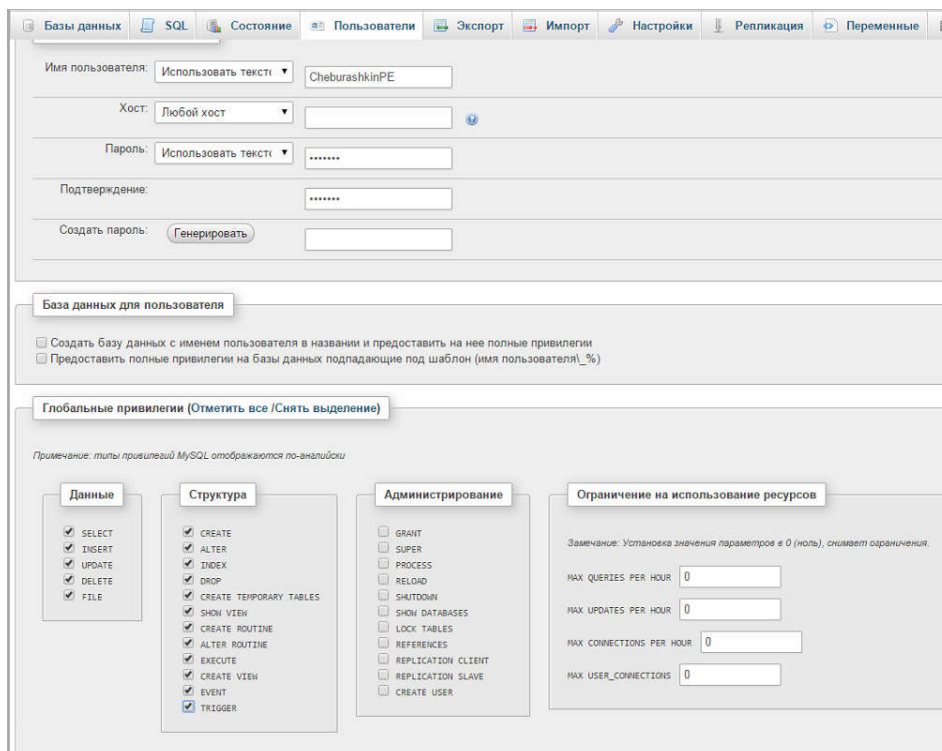


Рисунок 4.4 – Регистрация пользователя и назначение привилегий

3. Выйдем из системы и войдем в нее, используя созданную учетную запись (рисунок 4.5):

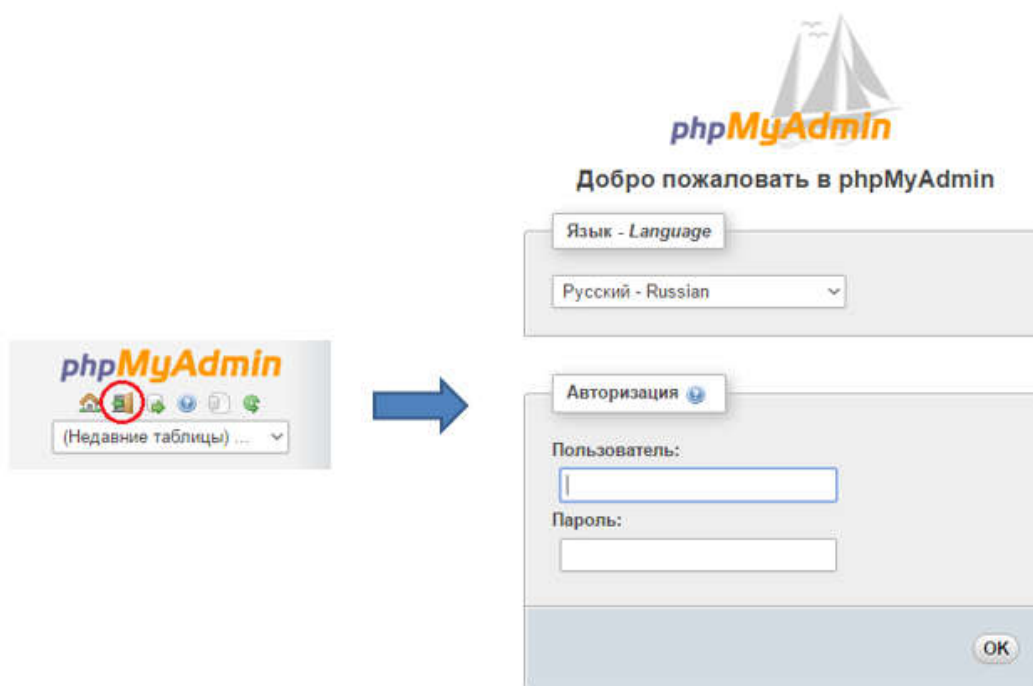


Рисунок 4.5 – Вход под созданной учетной записью

Создадим пустую базу данных:

4. Перейдите в раздел «Базы данных», введите имя БД, укажите кодировку «utf8_general_ci» и нажмите «Создать» (рисунок 4.6):

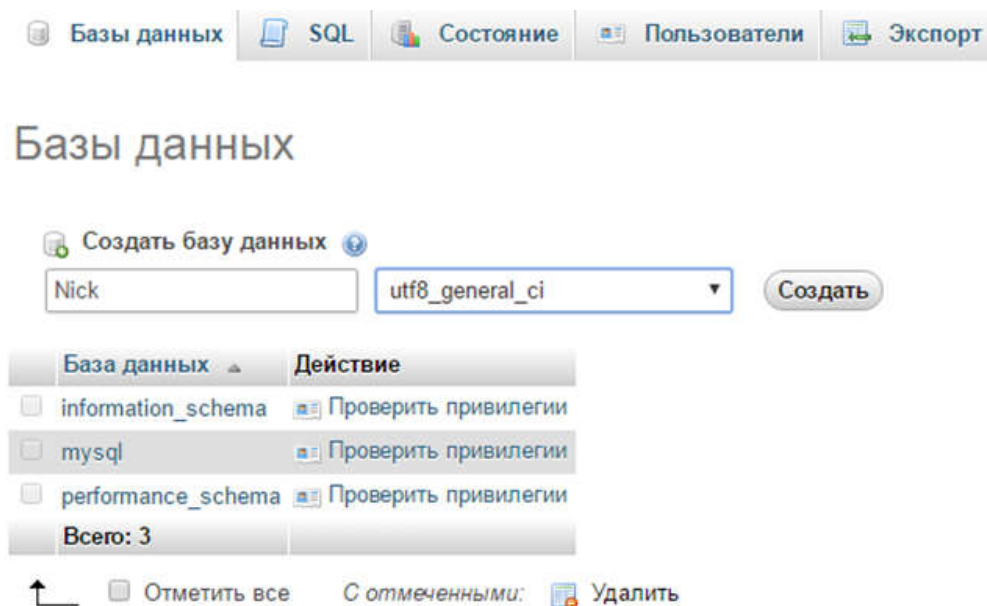


Рисунок 4.6 – Создание базы данных

Так же можно создать базу данных и с помощью SQL-запроса, перейдя в пункт «SQL», и введя команду «Create database rshu» (рисунок 4.7)

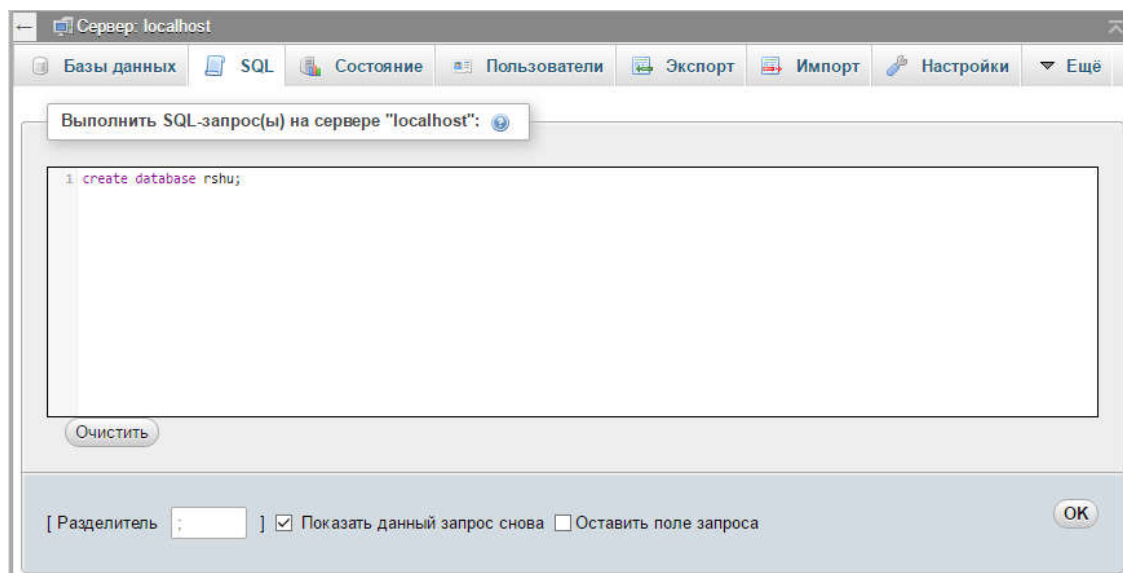


Рисунок 4.7 – Создание базы данных с помощью SQL-запроса

Просмотреть имеющиеся базы данных в системе можно с помощью запроса «show databases», а перейти к нужной с помощью «use имя_базы»;

5. Давайте создадим простую базу, которую мы создавали при рассмотрении MS Access. В меню «Структура» введите название первой таблицы «Students» и укажите количество столбцов (4), как это показано на рисунке 4.8.

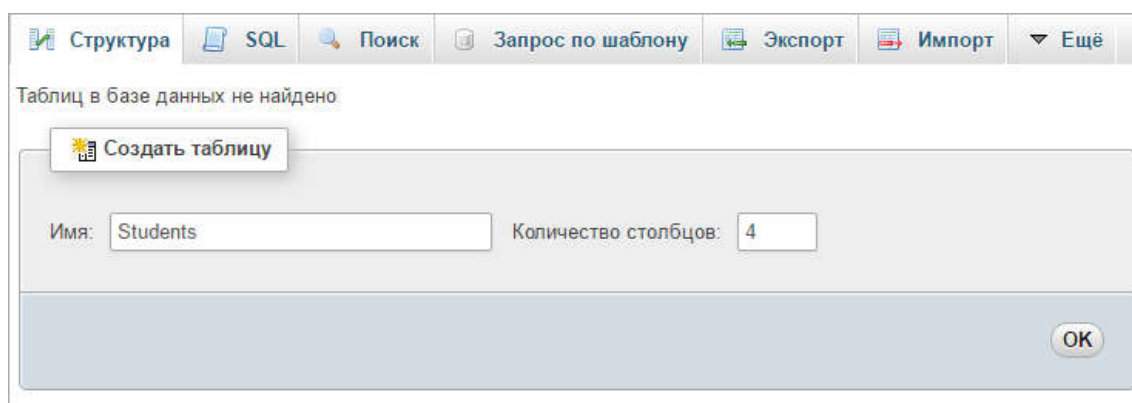


Рисунок 4.8 – Создание таблицы

6. После нажатия «ОК», система предложит заполнить название полей (рисунок 4.9), выбрать их тип, задать максимальную длину строки, и указать ключевое поле (указывается выставлением метки «Primary» в столбце «Индекс»).

Имя	Тип	Длина/значения	По умолчанию	Сравнение	Атрибуты	Null	Индекс
StudentID	INT		Нет			<input type="checkbox"/>	PRIMARY
FIO	CHAR	40	Нет			<input type="checkbox"/>	---
StudGroup	CHAR	10	Нет			<input type="checkbox"/>	---
NYear	INT		Нет			<input type="checkbox"/>	---

Рисунок 4.9 – Определение структуры таблицы

Так же можно создать таблицу, используя SQL (рисунок 4.10). Для этого следует ввести следующую команду:

```
create table Students (StudentID int not null primary key, FIO
char(40), StudGroup char(10), NYear int)
```

Здесь int указывает на то, что поле «StudentID» будет содержать только целые числа. Если пользователь попытается ввести буквы или символы препинания, СУБД сообщит об ошибке ввода. Команда «Not null» запрещает оставлять это поле пустым, т.к. оно является ключевым (primary key).

7. Создадим таблицу «Phone» с помощью SQL-запроса:

```
create table Phone (Code int not null AUTO_INCREMENT primary
key, StudentID int, Phone bigint)
```

8. Перейдите во вкладку «Вставить», чтобы ввести несколько записей из файла «Исходные данные». Введем данные о первых трех студентах, скопировав сведения в поля «Значение» и нажав «ОК»:

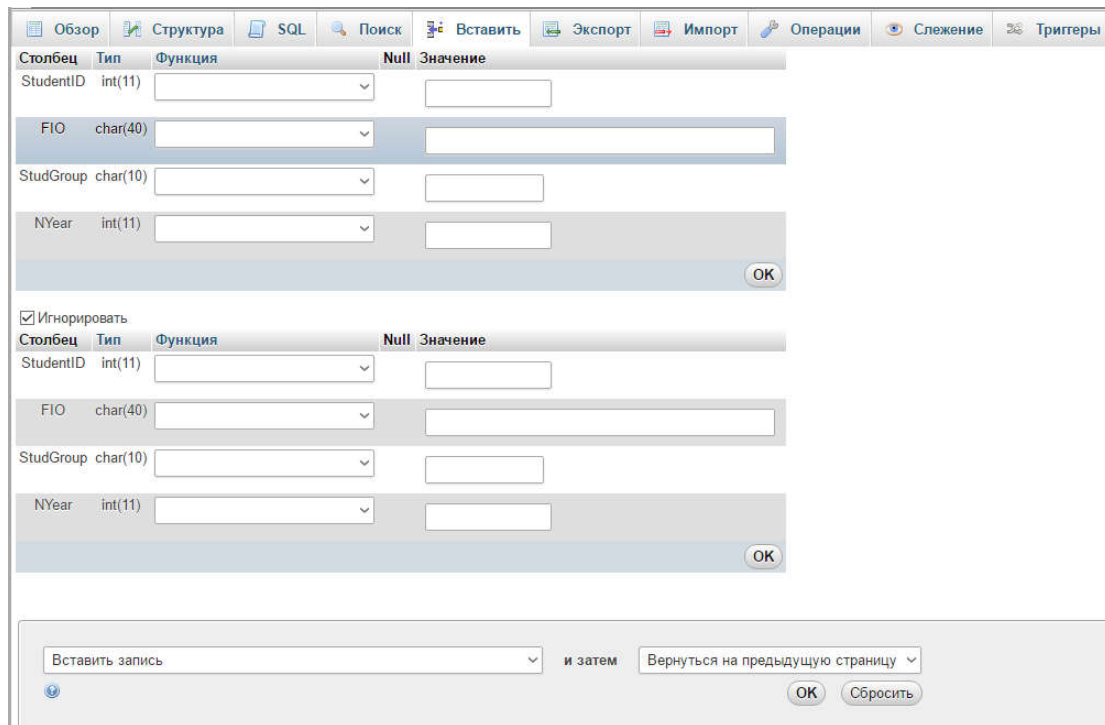


Рисунок 4.10 – Заполнение таблицы

9. Данных в Excel-файле много и вручную их скопировать будет сложно. Автоматизировать процесс ввода данных можно с помощью импорта CSV файлов (рисунки 4.11 и 4.12), которые можно подготовить в Excel:

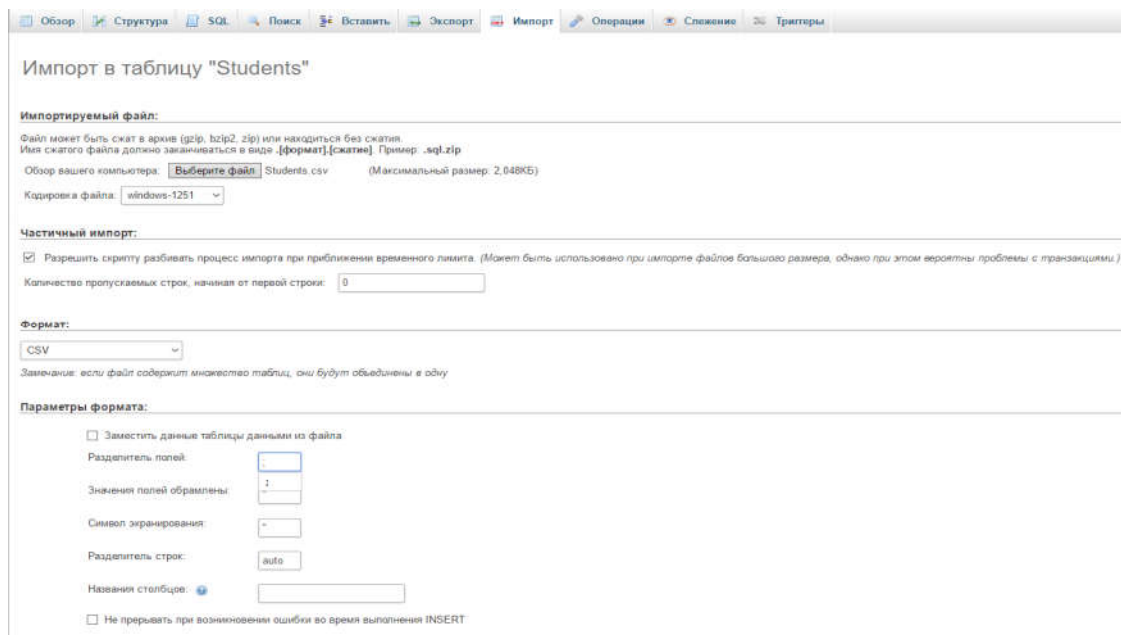


Рисунок 4.11 – Импорт данных в таблицу Students

Обзор Структура SQL Поиск Вставить Экспорт Импорт Операции Слежение Триггеры

Импорт в таблицу "Phone"

Импортируемый файл:

Файл может быть сжат в архив (gzip, bzip2, zip) или находиться без сжатия. Имя сжатого файла должно заканчиваться в виде `.[формат].[сжатие]`. Пример: `.sql.zip`

Обзор вашего компьютера: Phone.csv (Максимальный размер: 2,048КБ)

Кодировка файла:

Частичный импорт:

Разрешить скрипту разбивать процесс импорта при приближении временного лимита. (Может быть использовано при импорте файлов большого размера)

Количество пропускаемых строк, начиная от первой строки:

Формат:

Замечание: если файл содержит множество таблиц, они будут объединены в одну

Параметры формата:

Заместить данные таблицы данными из файла

Разделитель полей:

Значения полей обрамлены:

Символ экранирования:

Разделитель строк:

Названия столбцов:

Не прерывать при возникновении ошибки во время выполнения INSERT

Рисунок 4.12 – Импорт данных в таблицу Phones

10.

11. Чтобы проверить правильность работы нашей базы данных, введем запрос на вывод списка фамилий и телефонов:

```
SELECT Students.FIO, Phone.Phone FROM Students INNER JOIN
Phone ON Students.StudentID=Phone.StudentID
```

Если вы все сделали правильно, то результатом работы должна быть следующая таблица (рисунок 4.13):

Структура SQL Поиск Запрос по шаблону Экспорт Импорт Опера

Отобразить поле запроса

✔ Отображение строк 0 - 8 (9 всего, Запрос занял 0.0000 сек.)

```
SELECT student.FIO, phone.Phone FROM student INNER JOIN phone ON student.StudentID=phone.StidentID
```

Показать все | Количество строк: 25 | Фильтровать строки: Поиск в таблице

Сортировать по индексу: Ниодного

+ Параметры

FIO	Phone
Бачище Леонтий Яковлевич	89213070248
Воронин Артур Русланович	89213070249
Ермолов Михаил Дмитриевич	89213070250
Ермолов Михаил Дмитриевич	89213070330
Калатур Николай Петрович	89213070251
Клюева Ксения Николаевна	89213070252
Корзинева Юлия Владимировна	89213070253
Корзинева Юлия Владимировна	89213070335
Кудзаев Тимур Ирбекович	89213070254

Показать все | Количество строк: 25 | Фильтровать строки: Поиск в таблице

Рисунок 4.12 – Вывод списка ФИО и номеров телефонов студентов

5. BIG DATA

5.1 История происхождения термина

Big Data — один из наиболее часто упоминаемых сегодня терминов не только в ИТ-публикациях, но и в материалах, посвященных политической, экономической и социальной проблематике. Однако, как это часто бывает с новыми понятиями, при краткости термина его смысл весьма расплывчат. Трактовок понятия достаточно много, поэтому будем исходить, во-первых, из истории возникновения термина, во-вторых, из практических вопросов, где этот термин применим.

Таким образом, Big data – это сверхбольшие объемы структурированных и неструктурированных данных, с которыми трудно работать с помощью традиционных средств.

Впервые термин появился 3 сентября 2008 года, когда Клиффорд Линч, редактор научного журнала Nature [6] впервые использовал его для описания проблем накопления научных данных.

Источниками больших данных являются интернет-документы, социальные сети, блоги, измерительные устройства, радиочастотная идентификация, устройства аудио и видеорегистрации. Прогнозируется значительное расширение полей источников больших данных в грядущую эру так называемого Интернета вещей (Internet of things, IoT), тесно связанного с внедрением мобильной связи пятого поколения [7, 8].

Невозможность классифицировать большие данные и систематизировать их с помощью обычных статистических средств и традиционных СУБД привели к необходимости разработки инновационных технологий сбора, хранения и анализа информации и

появлению новых инструментов, таких как Hadoop, MapReduce, NoSQL баз данных, и фактически к появлению новой парадигмы анализа данных [8].

Таким образом, под Big Data понимают не какой-то конкретный объём данных и даже не сами данные, а методы их обработки, которые позволяют распределённо обрабатывать информацию. Эти методы можно применить как к огромным массивам данных, так и к маленьким.

5.2. Принципы работы

Основными принципами работы с Big Data являются следующие:

1. Горизонтальная масштабируемость. Любая система, которая подразумевает обработку больших данных, должна быть расширяемой. Как физические ресурсы (количество железа, объём кластера), так и программные (способность софта к усвоению большего объёма информации) должны увеличиваться пропорционально необходимости дополнительных ресурсов для обработки появившихся новых данных.

2. Отказоустойчивость. Принцип горизонтальной масштабируемости подразумевает, что машин в кластере может быть много. Например, Hadoop-кластер Yahoo имеет более 42000 машин. Это означает, что часть этих машин будет гарантированно выходить из строя. Методы работы с большими данными должны учитывать возможность таких сбоев и переживать их без каких-либо значимых последствий.

3. Локальность данных. В больших распределённых системах данные распределены по большому количеству машин. Если данные физически находятся на одном сервере, а обрабатываются на другом –

расходы на передачу данных могут превысить расходы на саму обработку.

К основным инструментам относится, например, **Hadoop** - система распределенных вычислений, проект фонда Apache Software Foundation, которая содержит свободно распространяемые функции, утилиты для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов. Разработка Hadoop началась в 2005 г. под влиянием публикации сотрудников Google о вычислительной концепции MapReduce. На сегодняшний день Yahoo содержит более 4 тыс. узлов, 15 Пбайт каждый), Facebook около 2 тыс. узлов на 21 Пбайт, Ebay – 700 узлов на 16 Пбайт [9].

Типовыми для Hadoop являются: компонентная структура, большое количество (до десятков тысяч) узлов, на основе относительно дешевого оборудования, возможность каждого узла быть сервером и хранения, и обработки, обработка данных в массивно-параллельном режиме, MapReduce, хранение данных в нескольких копиях (обычно в трех), т.е. отказ узла или двух не ведет к потере данных, практически неограниченная масштабируемость системы.

NoSQL базы данных являются нереляционными СУБД, у них нет традиционной структурированной схемы: данные часто хранятся в схеме Key-Value. Интерпретация смысла данных содержится в приложении, целостность данных поддерживается не так жестко, как в традиционных СУБД, то есть по сути нет никаких стандартов. Всего существует около 130 NoSQL СУБД, из которых значительное количество так называемых открытых систем (Open Source Systems). Преимуществом их является скорость обработки.

Также к инструментам можно отнести **Язык статистических исследований R** – OpenSource проект для углубленной аналитики на основе статистических исследований (временные ряды,

прогнозирование, классификация, кластеризация) с визуализацией результатов. К инструментам относят и различные **системы для семантического разбора текста** и визуализации, которые позволяют определить: содержательный портрет текста, упоминания персон и организаций, упоминания особых объектов, ситуации в тексте, отношение к объекту в тексте. Позволяют провести разбор частично структурированного текста, тематическое рубрицирование текстов, кластеризацию новостей, поиск документов, выявить заимствования и поиск похожих текстов.

5.3 Применение Big Data

Наиболее активно работа с большими данными ведется в банках и финансовых организациях, в компаниях телекома, в госсекторе, в недвижимости и ритейле.

Большие данные в российских банках: Сбербанк, Газпромбанк, ВТБ 24, Альфа Банк, Райффайзенбанк, Промсвязьбанк, Ситибанк.

Большие данные в ритейле: Лента, Азбука вкуса, Глория Джинс, Юлмарт, X5 Retail Group.

5.4 Big Data в политике

Швейцарский журнал Das Magazin опубликовал расследование [10] о том, как технологии персонализированной рекламы в сети Facebook

повлияли на итоги выборов в США и референдума о выходе Великобритании из ЕС.

По мнению издания, ключевые факторы победы Трампа – инновационные технологии, ежедневная слежка за каждым человеком и инновационные методы анализа собираемой информации с целью ее дальнейшего использования. Специалист по Big Data и психометрии Михал Косински, сотрудник Cambridge Analytica, понял, что победа Трампа связана с его научными разработками. Как только по ТВ объявили результаты выборов, фирма Cambridge Analytica выпустила пресс-релиз, в котором говорилось: «Мы поражены тем, что наш революционный подход к основанным на данных коммуникациям внес такой существенный вклад в победу Дональда Трампа».

Косински научился оценивать людей в рамках разработанной им модели по юзерпику (user picture), количеству друзей и пр. Он также анализировал приватные данные пользователя – на это владелец давал согласие при установке приложения-теста. Таким образом, можно говорить о том, что Косински создал поисковую систему по людям.

В 2014 году компания Strategic Communications Laboratories, которая занималась предвыборными кампаниями, предложила Косински проанализировать 10 млн людей. Позднее разоблачения Wikileaks показали, что за компанией стояли корпорации, система была завязана на «налоговые гавани». Одной из дочерних компаний SCL была та самая Cambridge Analytica, которая организовывала кампании в поддержку Brexit и Трампа.

В дни дебатов Трампа и Клинтон в социальные сети отправили 175 тыс. вариантов посылов. Материалы продвигали как положительные качества Трампа, так и приводили аргументы, разубеждавшие голосовать за Клинтон. Компания сконцентрировалась на 17 штатах и выделила 32 психотипа людей. Услуги Cambridge Analytica в течение

предвыборной кампании Трампа обошлись в 15 млн долларов. Известно, что сотрудничать с Cambridge Analytica планируют видные политики европейских стран.

ЗАКЛЮЧЕНИЕ

В пособии изложены основные сведения о языке SQL, а также об основах работы с базами данных и основных СУБД, используемых при геоинформационном обеспечении поддержки решений при рациональном природопользовании, включая управление рисками.

Учебное пособие соответствует программам обучения студентов РГГМУ очной формы обучения дисциплин «Аппаратные средства вычислительной техники», «Основы управления рисками», «Геориски», «Управление рисками».

Настоящее учебное пособие создано при финансовой поддержке Минобрнауки России (проект 5.4425.2017/NM).

СПИСОК ЛИТЕРАТУРЫ

1. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: полное руководство, 3-е издание = SQL: The Complete Reference, Third Edition. — М.: «Вильямс», 2014. — 960 с.
2. Мартин Грабер. Справочное руководство по SQL. - М.: Издательство "Лори", 1997
3. Роберт Шелдон, Джоффри Мойе. MySQL 5: базовый курс = Beginning MySQL. — М.: «Диалектика», 2007. — 880 с.
4. Pete Pachal. Why Facebook Uses MySQL for Timeline. - <http://mashable.com/2011/12/15/facebook-timeline-mysql/#qBdPORSsu8q8>
5. Джентльменский набор web-разработчика («Денвер»): интервью с создателем // КомпьютерПресс. — 2007. — № 10.
6. Clifford A. Lynch, “Big data: How do your data grow?” Nature, vol. 455, no. 7209 (September 3, 2008).
7. Velev D., Zlateva P. An analysis of the relation between natural disasters and Big Data. – Int. J. Data Science, Vol.1, No 4, 2016.
8. Igor Jurčić, IoT: opportunities and threats for modern mobile operators, 2017. - https://www.researchgate.net/publication/317013810_IoT_opportunities_and_threats_for_modern_mobile_operators
9. Donald Miner, Adam Shook. MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems.- 2012. — 230 с.

10.Hadoop. - <http://ru-wiki.org/wiki/Hadoop>

11.Ich habe nur gezeigt, dass es die Bombe gibt. Von Hannes Grassegger
und Mikael Krogerus. - Das Magazin N°48 – 3. Dezember 2016