ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ

Кафедра информационных систем и технологий

Бедняк С.Г., Захарова О.И.

БАЗЫ ДАННЫХ

Методические указания
к лабораторным занятиям по дисциплине
для направлений 38.03.05, 27.03.04, 09.03.04
очной формы обучения
(часть 2)

Самара 2018

УДК 004.65+ 004.43
ББК 32.973

Рекомендовано к изданию методическим советом ПГУТИ,
протокол № 49, от 8.05.2018 г.

Захарова, О. И.

Базы данных: методические указания к лабораторным занятиям для бакалавров по направлениям подготовки 38.03.05 – «Бизнес – информатика», 27.03.04 – «Управление в технических системах», 09.03.04 – «Программная инженерия» по дисциплине «Базы данных» (часть 2) / О. И. Захарова, С.Г. Бедняк – Самара: ПГУТИ, 2018. – 60 с.

Целью преподавания дисциплины «Базы данных» является обучение студентов основным принципам и методам построения баз данных, необходимых при создании, исследовании и эксплуатации информационных систем различной природы.

В курсе изучаются теоретические основы реляционных баз данных, структурированный язык запросов SQL, принципы создания баз данных с использованием реляционной СУБД.

Рассматриваются практические аспекты выполнения запросов на языке SQL в реляционных базах данных.

В методических рекомендациях к лабораторным занятиям даны задания к лабораторным работам, помогающие закрепить на практике полученные знания.

Методические указания разработаны в соответствии с Федеральным государственным образовательным стандартом высшего профессионального образования по направлениям подготовки 38.03.05 – «Бизнес – информатика», 27.03.04 – «Управление в технических системах», 09.03.04 – «Программная инженерия» и предназначены для студентов третьего курса факультета информационных систем и технологий, а также для студентов других специальностей, изучающих и использующих базы данных.

© Захарова О.И.
© Бедняк С.Г.
2018

Оглавление

Лабораторная работа № 6

Создание таблиц базы данных, полей, ключей, индексов, реляционных связей

1. Цель работы

Изучить создание таблицы базы данных в Web-интерфейсе phpMyAdmin.

2. Теоретическая часть

Реляционные базы данных хранят все данные в таблицах. Таблица это структура, состоящая из множества неупорядоченных горизонтальных строк (rows), каждая из которых содержит одинаковое количество вертикальных столбцов (colums). Пересечение отдельной строки и столбца называется полем (field), которое содержит специфическую информацию. Многие принципы работы реляционной базы данных взяты из определений отношений (relations) между таблицами.

Создание таблицы главным образом подразумевает определение столбцов таблицы. Главные атрибуты столбца включают:

Имя столбца;

Тип данных столбца или домен на котором он базируется;

Может или нет поле столбца принимать значение NULL;

Факультативно справочные ограничения целостности (referential integrity constraints).

Типы данных (Data types)

В каждой таблице БД должен существовать *первичный ключ*. Под первичным ключом понимают поле или набор полей, однозначно (уникально) идентифицирующих запись. Первичный ключ должен быть минимально достаточным: в нем не должно быть полей, удаление которых из первичного ключа не отразится на его уникальности.

Данные таблицы «Преподаватель»

Таб. №	ФИО	Уч. степень	Уч. звание	Код кафедры
101	Андреев А.П.	Д-р техн. наук	Профессор	01
102	Апухтин И.С.	Канд. техн. наук	Доцент	01
103	Глухов И.Л.	Канд. техн. наук	Доцент	01
104	Сеченов Ю.Б.	Канд. техн. наук	Доцент	01

В качестве первичного ключа в таблице «Преподаватель» может выступать только «Таб. №», значения других полей могут повторяться внутри данной таблицы.

При разработке структур баз данных нужно всегда определять первичный ключ для таблицы базы данных.

Индексы это механизм для улучшения быстродействия поиска данных. Индекс определяет столбцы которые могут быть использованы для эффективного поиска и сортировки в таблице.

Методические указания и примеры выполнения работы

Создание базы данных

В открывшемся окне phpMyAdmin сразу можно создавать базу данных, или выбрать слева одну из ранее созданных. (Рис. 1)

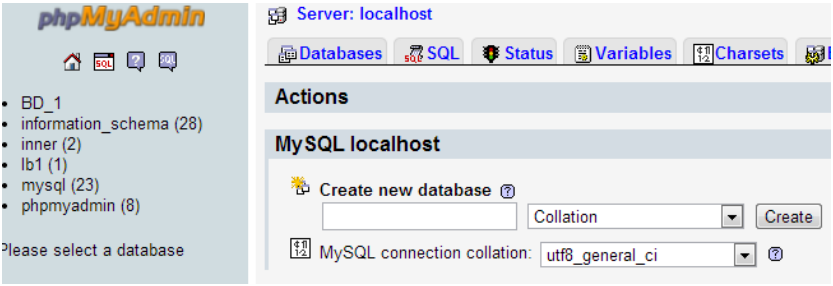


Рис. 1.

После добавления название базы данных, необходимо нажать кнопку «Creat». База данных будет создана. (Рис. 2)

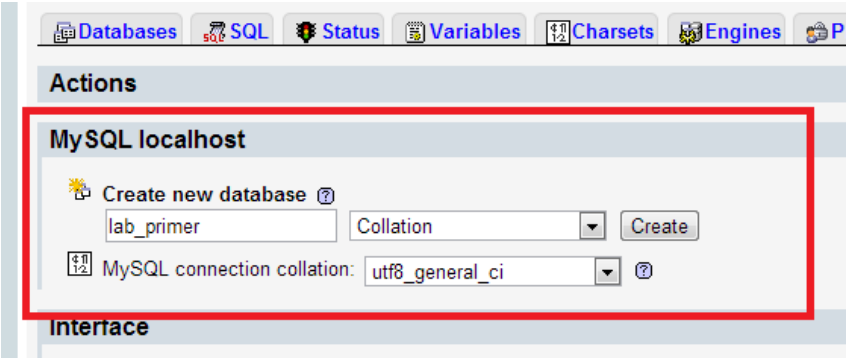


Рис. 2

Если на экране нет сообщения об ошибке (например, о том, что такое же имя базы данных уже используется), открывается окно с возможностями работы с БД. Здесь можно сразу создать первую таблицу (Рис. 3).

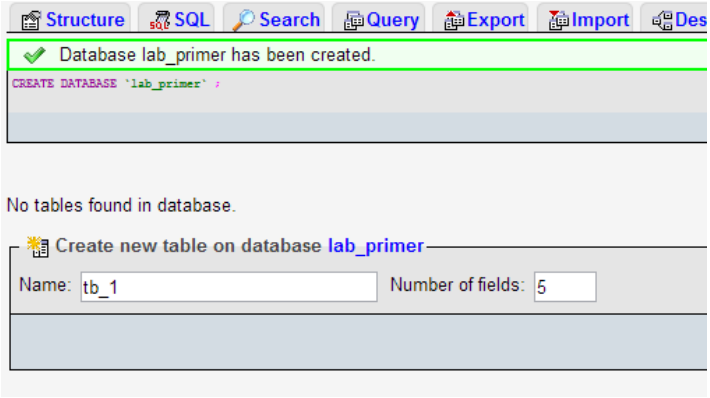


Рис. 3

В полученном окне создается первая таблица tb_1 с количеством атрибутов равное 5. Добавив эти значения в соответствующие поля для заполнения необходимо нажать кнопку «GO». Далее, указывается название атрибутов и выбирается для них длину и тип данных, а так же назначается первичный ключ. Закончив заполнение, необходимо нажать кнопку «Save» (Рис. 4).

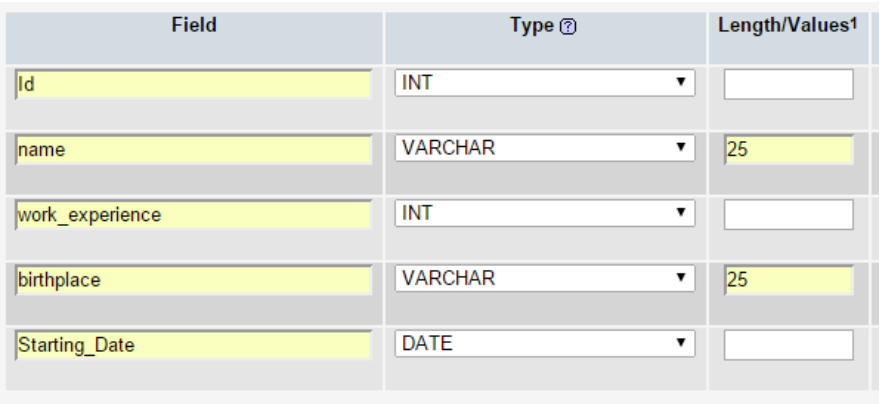


Рис. 4.

Таблица создана. Теперь необходимо ее заполнить. Для этого нужно выбрать вкладку Insert (Рис5):

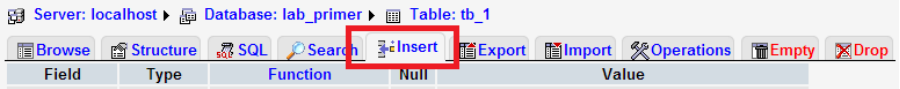


Рис. 5

В открывшемся меню заполнить записи. По умолчанию программа позволяет добавить 2 записи, если этого мало – внизу, во вкладке «Restart insertion with» можно выбрать количество необходимое добавить в таблицу (Рис. 6):

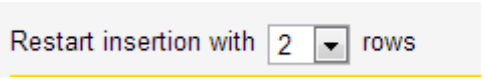


Рис. 6

Значения добавляются в столбце «Value» (Рис. 7).

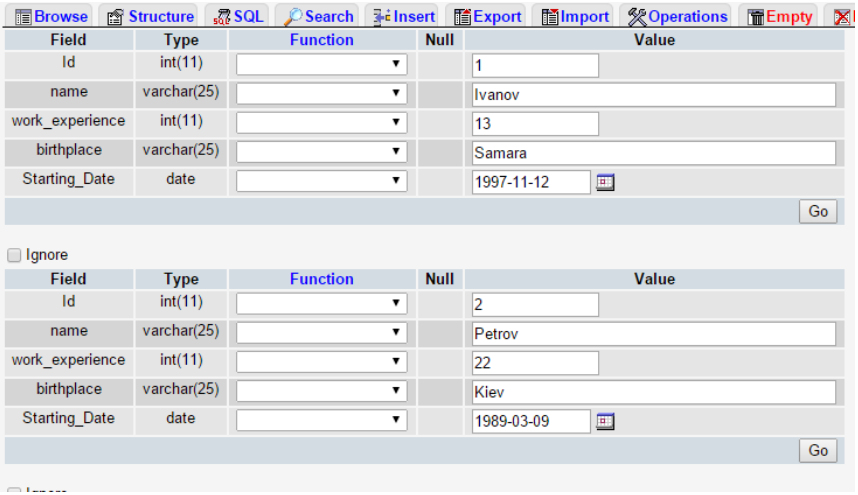


Рис. 7

3. Подготовка к работе

3.1 Изучить методические указания и рекомендованную литературу.

3.2 Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

4.1 Запустите СУБД MySQL.

4.2 Создайте базу данных.

4.3 Создайте необходимое количество таблиц (не менее пяти по теме варианта). В каждой таблице не менее 20 записей.

4.4 При выполнении шагов 2,3 необходимо документировать ход работы.Подготовить отчет.

Варианты заданий

Все лабораторные работы выполняются по одной теме. Вариант задания (тема) выбирается один раз, в лабораторной работе № 3. Тема выбирается согласно номеру студента в списке группы.

5. Требование к отчету

Структура отчёта зависит от типа задания и индивидуального стиля работы студента. Однако, все работы должны обладать некоторыми общими чертами и включать следующие основные элементы:

- Титульный лист.
- Лист рецензии.
- Содержание.
- Цель лабораторной работы.
- Ответы на контрольные вопросы.
- Введение.
- Теоретическая (практическая) часть. Инфологическая модель базы данных согласно варианту.
- Поэтапно описанный процесс создания базы данных и таблиц, представленный не только текстовой частью но и скриншотами.
- Заключение (выводы по работе).
- Список источников информации.
- Приложения.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЁТА:

- Текстовый материал должен быть подготовлен с использованием MS Word.
- Текст должен быть набран с учетом следующих параметров:
    1. шрифт Times New Roman;
    2. размер шрифта заголовков – 14, основного текста - 12;
    3. межстрочный интервал - 1;
    4. межбуквенный интервал - Normal;
    5. поля: левое – 2.5, правое – 2, верхнее и нижнее – 1.5;
    6. выравнивание – по ширине страницы;
    7. нумерация страниц – в нижем правом углу;
    8. в верхнем колонтитуле указать название работы, Ф.И.О. автора;
    9. образец оформления титульного листа приведен в приложении №1
- Объем работы не менее 10 страниц.
- Использовать электронное оглавление.
- Рисунки должны быть пронумерованы и подписаны.
- Список источников должен содержать как список литературы, так и список ресурсов Интернет.

6. Контрольные вопросы

1.1 Что такое таблица?

1.2 Для чего требуется ключи?

1.3 Каким типом данных должны быть ключи?

1.4 Для чего требуется индекс?

7. Рекомендуемая литература

7.1 Диго С. М. Базы данных: проектирование и использование [Text] : учебник для вузов / Диго С. М. - М. : Финансы и статистика, 2005. - 592 с. : ил. - 20 экз. - ISBN 5-279-02571-2 (18)

7.2 Андон Ф. Язык запросов SQL. Учебный курс [Text] / Андон Ф. - СПб. : Питер, 2006. - 416 с. : ил. - 30 экз. - ISBN 5-469-00394-9 : Б. ц. УДК 004.6004.65(075) (30)

7.3 Рудикова, Л. В. Базы данных. Разработка приложений для студента [Text] / Рудикова, Л. В. - СПб. : БХВ-Петербург, 2006. - 496 с. : ил. - 30 экз. - ISBN 5-94157-805-9 (30)

7.4 Карпова, И. П. Базы данных. Курс лекций и материалы для практических занятий [Текст] : учеб. пособие для вузов / И. П. Карпова . - СПб. : Питер, 2013. - 240 с.

7.5 Малыхина, М. П. Базы данных: основы, проектирование, использование [Text] : учеб. пособие для вузов / Малыхина, М. П. - СПб. : БХВ-Петербург, 2004. - 499 с. : ил. - 99 экз. - (Учебное пособие). - Библиогр.: с. 491-493 (28 назв.). - ISBN 5-94157-310-4 (98)

7.6 Мотев, А. А. Уроки MySQL [Text] : самоучитель / Мотев, А. А. - СПб. : БХВ-Петербург, 2006. - 208 с. : ил. - 20 экз. - (Самоучитель). - ISBN 5-94157-658-7(20)

Лабораторная работа № 7

Простые SQL-запросы на выборку данных

1. **Цель работы**

Изучить простые SQL-запросы в web-интерфейсе phpMyAdmin.

2. **Теоретическая часть**

Описание языка

SQL символизирует собой *Структурированный Язык Запросов*. Это - язык, который дает Вам возможность создавать и работать в реляционных базах данных, являющихся наборами связанной информации, сохраняемой в таблицах. Информационное пространство становится более унифицированным. Это привело к необходимости создания стандартного языка, который мог бы использоваться в большом количестве различных видов компьютерных сред. Стандартный язык позволит пользователям, знающим один набор команд, использовать их для создания, нахождения, изменения и передачи информации - независимо от того, работают ли они на персональном компьютере, сетевой рабочей станции, или на универсальной ЭВМ. В нашем все более и более взаимосвязанном компьютерном мире, пользователь снабженый таким языком, имеет огромное преимущество в использовании и обобщении информации из ряда источников с помощью большого количества способов. Элегантность и независимость от специфики компьютерных технологий, а также его поддержка лидерами промышленности в области технологии реляционных баз данных, сделало SQL (и, вероятно, в течение обозримого будущего оставит его) основным стандартным языком. По этой причине, любой, кто хочет работать с базами данных 90-х годов, должен знать SQL. Стандарт SQL определяется ANSI (*Американским Национальным Институтом Стандартов*) и в данное время также принимается ISO (*Международной Организацией по Стандартизации*). Однако, большинство коммерческих программ баз данных расширяют SQL без уведомления ANSI, добавляя различные особенности в этот язык, которые, как они считают, будут весьма полезны. Иногда они несколько нарушают стандарт языка, хотя хорошие идеи имеют тенденцию развиваться и вскоре становиться стандартами "*рынка*" сами по себе в силу полезности своих качеств.

Оператор SELECT

Язык запросов в SQL состоит из единственного оператора – SELECT. Синтаксис оператора SELECT имеет следующий вид:

SELECT [ALL| DISTINCT] <Список полей>|* FROM <Список таблиц> [WHERE <Предикат-условие выборки или соединения>] [GROUP BY <Список полей результата>] [HAVING <Предикат-условие для группы>] [ORDER BY <Список полей, по которым упорядочить вывод>];

SELECT – ключевое слово, которое сообщает СУБД, что эта команда – запрос. Все запросы начинаются этим словом, с последующим пробелом. За ним может следовать способ выборки. Здесь ключевое слово ALL означает, что в результирующий набор строк включаются все строки, удовлетворяющие условиям запроса. Значит, в результирующий набор могут попасть одинаковые строки. Это нарушение принципов теории отношений (в отличие от реляционной алгебры, где по умолчанию предполагается отсутствие дубликатов в каждом результирующем отношении). Ключевое слово DISTINCT означает, что в результирующий набор включаются только различные строки, то есть дубликаты строк результата не включаются в набор. Список полей – это список перечисленных через запятую столбцов, которые выбираются запросом из таблиц. Символ * (звездочка) означает, что в результирующий набор включаются все столбцы из исходных таблиц запроса. После ключевого слова SELECT можно выводить не просто названия столбцов, но и выполнять с ними арифметические действия, как по отдельности, так и используя несколько столбцов одновременно.

В разделе FROM задается перечень исходных отношений (таблиц) запроса. В случае, если указано более одного имени таблицы, неявно подразумевается, что над перечисленными таблицами осуществляется операция декартова произведения.

Разделы SELECT и FROM являются обязательными, все другие разделы являются необязательными.

В разделе WHERE задаются условия отбора строк результата или условия соединения кортежей исходных таблиц, подобно операции условного соединения в реляционной алгебре. В выражении условий раздела WHERE могут быть использованы следующие предикаты сравнения (=, <>, >, >=, <, <=), которые имеют традиционный смысл.

3. Подготовка к работе

3.1 Изучить методические указания и рекомендованную литературу.

3.2 Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

4.1 Запустить СУБД MySQL.

4.2 Выбрать созданную ранее базу данных.

4.3 Выбрать вкладку SQL.

4.4 Выполнить 4 простых запроса, сохранив их в текстовом документе.

4.5 Сделать скриншоты, полученных в результате запросов данных.

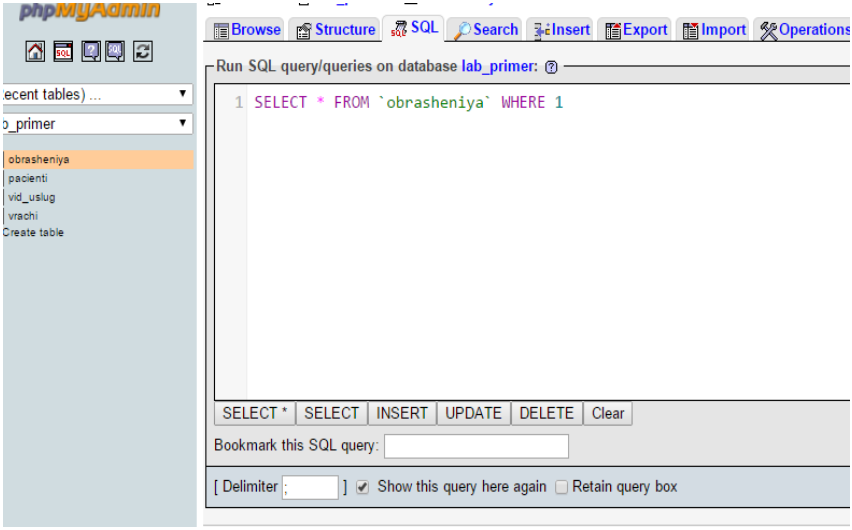
4.6 Подготовить отчет.

Варианты заданий

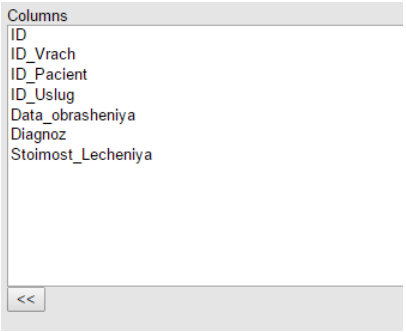
Все лабораторные работы выполняются по одной теме. Вариант задания (тема) выбирается один раз, в лабораторной работе № 3. Тема выбирается согласно номеру студента в списке группы.

Пример выполнения лабораторной работы

Запустив Денвер, и перейдя в нужную базу данных, переходим во вкладку SQL.



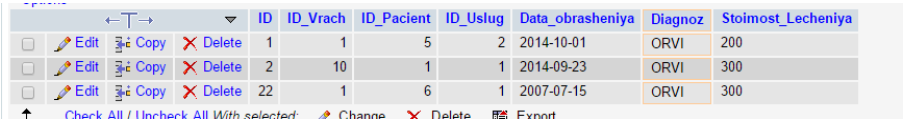
При переходе во вкладку SQL из какой-либо таблицы, будет создан простейший запрос, который будет отображать все данные из таблицы, так же справа будет меню, для быстрого вывода атрибутов:



Для вывода данных из таблицы «Обращения» тех пациентов, что обратились с диагнозом ORVI, пишется во вкладку SQL:

SELECT * FROM `obrasheniya` WHERE `Diagnoz` = 'ORVI'

СУБД представляет нам информацию по данному запросу:



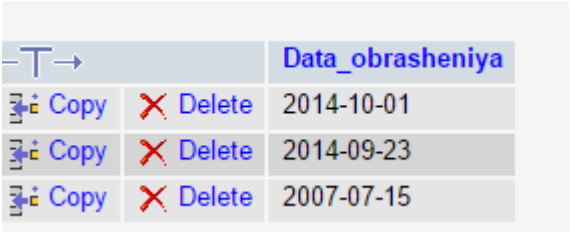
	ID	ID_Vrach	ID_Pacient	ID_Uslug	Data_obrasheniya	Diagnoz	Stoimost_Lecheniya
Edit Copy Delete	1	1	5	2	2014-10-01	ORVI	200
Edit Copy Delete	2	10	1	1	2014-09-23	ORVI	300
Edit Copy Delete	22	1	6	1	2007-07-15	ORVI	300

Check All / Uncheck All With selected: Change Delete Export

Учитывая, что звездочка после SELECT позволяет выводить всю информацию из перечисленных таблиц, это порой бывает излишне. Если требуется узнать только дату обращения, можно указать её в условии:

SELECT `Data_obrasheniya` FROM `obrasheniya` WHERE `Diagnoz` = 'ORVI'

Данных из таблицы станет намного меньше:



	Data_obrasheniya
Copy Delete	2014-10-01
Copy Delete	2014-09-23
Copy Delete	2007-07-15

Содержание отчёта

Титульный лист.

Цель лабораторной работы.

Ответы на контрольные вопросы.

Выводы по работе.

5. Требование к отчету

Структура отчёта зависит от типа задания и индивидуального стиля работы студента. Однако, все работы должны обладать некоторыми общими чертами и включать следующие основные элементы:

- Титульный лист.
- Лист рецензии.
- Содержание.
- Цель лабораторной работы.
- Ответы на контрольные вопросы.
- Введение.
- Теоретическая (практическая) часть. Инфологическая модель базы данных согласно варианту.
- Результаты выполнения простых запросов (код и скриншоты).
- Заключение (выводы по работе).
- Список источников информации.
- Приложения.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЁТА:

- Текстовый материал должен быть подготовлен с использованием MS Word.
- Текст должен быть набран с учетом следующих параметров:
  1. шрифт Times New Roman;
  2. размер шрифта заголовков – 14, основного текста - 12;
  3. межстрочный интервал - 1;
  4. межбуквенный интервал - Normal;
  5. поля: левое – 2.5, правое – 2, верхнее и нижнее – 1.5;
  6. выравнивание – по ширине страницы;
  7. нумерация страниц – в нижем правом углу;
  8. в верхнем колонтитуле указать название работы, Ф.И.О. автора;
  9. образец оформления титульного листа приведен в приложении №1
- Объем работы не менее 10 страниц.
- Использовать электронное оглавление.
- Рисунки должны быть пронумерованы и подписаны.
- Список источников должен содержать как список литературы, так и список ресурсов Интернет.

6. Контрольные вопросы

6.1 Что такое SQL? Область его применения?

6.2 Для чего используются предикаты сравнения?

6.3 Перечислите и опишите основные элементы вкладки SQL.

6.4 Каких ключевых слов достаточно для простого запроса?

7. Рекомендуемая литература

7.1 Диго С. М. Базы данных: проектирование и использование [Text] : учебник для вузов / Диго С. М. - М. : Финансы и статистика, 2005. - 592 с. : ил. - 20 экз. - ISBN 5-279-02571-2 (18)

7.2 Андон Ф. Язык запросов SQL. Учебный курс [Text] / Андон Ф. - СПб. : Питер, 2006. - 416 с. : ил. - 30 экз. - ISBN 5-469-00394-9 : Б. ц. УДК 004.6004.65(075) (30)

7.3 Рудикова, Л. В. Базы данных. Разработка приложений для студента [Text] / Рудикова, Л. В. - СПб. : БХВ-Петербург, 2006. - 496 с. : ил. - 30 экз. - ISBN 5-94157-805-9 (30)

7.4 Карпова, И. П. Базы данных. Курс лекций и материалы для практических занятий [Текст] : учеб. пособие для вузов / И. П. Карпова . - СПб. : Питер, 2013. - 240 с.

7.5 Малыхина, М. П. Базы данных: основы, проектирование, использование [Text] : учеб. пособие для вузов / Малыхина, М. П. - СПб. : БХВ-Петербург, 2004. - 499 с. : ил. - 99 экз. - (Учебное пособие). - Библиогр.: с. 491-493 (28 назв.). - ISBN 5-94157-310-4 (98)

7.6 Мотев, А. А. Уроки MySQL [Text] : самоучитель / Мотев, А. А. - СПб. : БХВ-Петербург, 2006. - 208 с. : ил. - 20 экз. - (Самоучитель). - ISBN 5-94157-658-7(20)

Лабораторная работа № 8

Использование реляционных и булевых операторов для создания предикатов SQL

1. **Цель работы**

Изучить SQL-запросы с использованием реляционных и булевых операторов в web-интерфейсе phpMyAdmin.

2. **Теоретическая часть**

Реляционный оператор - математический символ который указывает на определенный тип сравнения между двум значениями. Предположим что вы хотите видеть всех Продавцов с их комиссионными выше определенного значения. Вы можете использовать тип сравнения "больше чем" - (>). Реляционные операторы которыми располагает SQL :

= Равный к

> Больше чем

< Меньше чем

>= Больше чем или равно

<= Меньше чем или равно

< > Не равно

Эти операторы имеют стандартные значения для числовых значений. Для значения символа, их определение зависит от формата преобразования, ASCII или EBCDIC, который вы используете. SQL сравнивает символьные значения в терминах основных номеров, как определено в формате преобразования. Даже значение символа, такого как "1", который представляет номер, не обязательно равняется номеру, который он представляет. Вы можете использовать реляционные операторы чтобы установить алфавитный порядок - например, "a" < "n" где средство а первое в алфавитном порядке - но все это ограничивается с помощью параметра преобразования формата.

И в ASCII и в EBCDIC, символы - по значению: меньше чем все другие символы которым они предшествуют в алфавитном порядке и имеют один вариант(верхний или нижний). В ASCII, все символы верхнего регистра - меньше чем все символы нижнего регистра, поэтому "Z" < "a", а все номера - меньше чем все символы, поэтому "1" < "Z". То же относится и к EBCDIC. Чтобы сохранить обсуждение более простым, мы допустим что вы будете использовать текстовый формат ASCII. Проконсультируйтесь с вашей документацией системы, если вы неуверенны какой формат вы используете или как он работает. Значения сравниваемые здесь называются - скалярными значениями. Скалярные

значения производятся скалярными выражениями; 1 + 2 - это скалярное выражение которое производит скалярное значение 3. Скалярное значение может быть символом или числом, хотя очевидно, что только номера используются с арифметическими операторами, такими как +(плюс) или *(звезда). Предикаты обычно сравнивают значения скалярных величин, используя или реляционные операторы или специальные операторы SQL, чтобы увидеть верно ли это сравнение. Предположим, что вы хотите увидеть всех заказчиков с оценкой(rating) выше 200. Так как 200 - это скалярное значение, как и значение в столбце оценки, для их сравнения вы можете использовать реляционный оператор.

Основные Булевы операторы также распознаются в SQL. Выражения Буля - являются или верными или неверными, подобно предикатам. Булевы операторы связывают одно или более верных/неверных значений и производят единственное верное/или/неверное значение. Стандартными операторами Буля распознаваемыми в SQL являются: AND, OR, и NOT.

Использование больше чем (>) Существуют другие, более сложные, операторы Буля (типа " исключенный или "), но они могут быть сформированы из этих трех простых операторов - AND, OR, NOT. Как вы можете понять, Булева верна / неверна логика - основана на цифровой компьютерной операции; и фактически, весь SQL(или любой другой язык) может быть сведен до уровня Булевой логики.

Операторы Буля и как они работают:

* AND берет два Буля (в форме A AND B) как аргументы и оценивает их по отношению к истине, верны ли они оба.

* OR берет два Буля (в форме A OR B) как аргументы и оценивает на правильность, верен ли один из них.

* NOT берет одиночный Булев (в форме NOT A) как аргументы и заменяет его значение с неверного на верное или верное на неверное.

Связывая предикаты с операторами Буля, вы можете значительно увеличить их возможности.

3. Подготовка к работе

3.1 Изучить методические указания и рекомендованную литературу.

3.2 Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

4.1 Запустить СУБД MySQL.

4.2 Выбрать созданную прежде базу данных.

4.3 Выбрать вкладку SQL.

4.4 Выполнить 4 запроса, используя реляционные и булевы операторы, сохранив их в текстовом документе.

4.5 Сделать скриншоты, полученных в результате запросов данных.

4.6 Подготовить отчет.

Варианты заданий

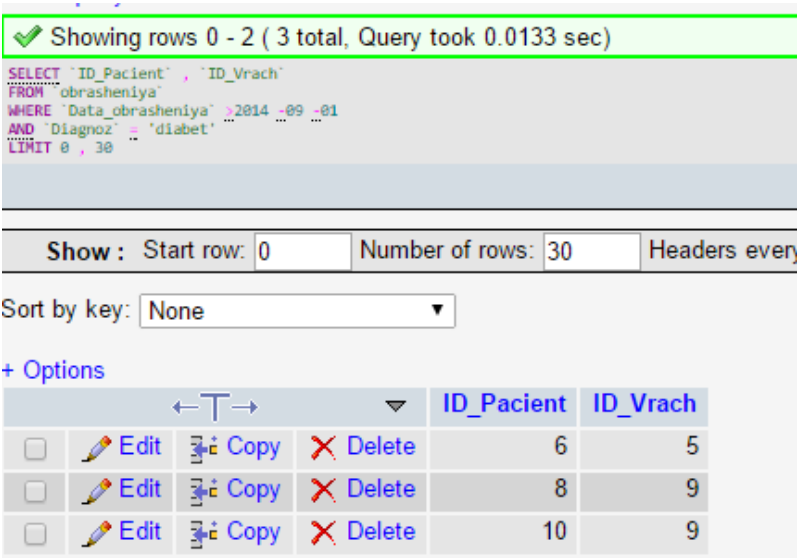
Все лабораторные работы выполняются по одной теме. Вариант задания (тема) выбирается один раз, в лабораторной работе № 3. Тема выбирается согласно номеру студента в списке группы.

Пример выполнения лабораторной работы

Иногда для выполнения запроса, необходимо составить условие, в котором требуется несколько значений переменных, например дата обращения не раньше 1 сентября 2014 года и диагноз диабет. Используя оператор AND, создается запрос:

SELECT `ID_Pacient`,`ID_Vrach` FROM `obrasheniya` WHERE `Data_obrasheniya` > 2014-09-01 and `Diagnoz` ='diabet'

По искомым значениям получаем трёх пациентов:



.

5. Требование к отчету

Структура отчёта зависит от типа задания и индивидуального стиля работы студента. Однако, все работы должны обладать некоторыми общими чертами и включать следующие основные элементы:

- Титульный лист.
- Лист рецензии.
- Содержание.
- Цель лабораторной работы.
- Ответы на контрольные вопросы.
- Введение.
- Теоретическая (практическая) часть. Инфологическая модель базы данных согласно варианту. Результаты выполнения запросов (код и скриншоты).
- Заключение (выводы по работе).
- Список источников информации.
- Приложения.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЁТА:

- Текстовый материал должен быть подготовлен с использованием MS Word.
- Текст должен быть набран с учетом следующих параметров:
  1. шрифт Times New Roman;
  2. размер шрифта заголовков – 14, основного текста - 12;
  3. межстрочный интервал - 1;
  4. межбуквенный интервал - Normal;
  5. поля: левое – 2.5, правое – 2, верхнее и нижнее – 1.5;
  6. выравнивание – по ширине страницы;
  7. нумерация страниц – в нижем правом углу;
  8. в верхнем колонтитуле указать название работы, Ф.И.О. автора;
  9. образец оформления титульного листа приведен в приложении №1
- Объем работы не менее 10 страниц.
- Использовать электронное оглавление.
- Рисунки должны быть пронумерованы и подписаны.
- Список источников должен содержать как список литературы, так и список ресурсов Интернет.

6. Контрольные вопросы

6.1 Откуда взято название булевы операторы?

6.2 Напишите булевы операторы по их приоритету выполнения.

7. Рекомендуемая литература

7.1 Диго С. М. Базы данных: проектирование и использование [Text] : учебник для вузов / Диго С. М. - М. : Финансы и статистика, 2005. - 592 с. : ил. - 20 экз. - ISBN 5-279-02571-2 (18)

7.2 Андон Ф. Язык запросов SQL. Учебный курс [Text] / Андон Ф. - СПб. : Питер, 2006. - 416 с. : ил. - 30 экз. - ISBN 5-469-00394-9 : Б. ц. УДК 004.6004.65(075) (30)

7.3 Рудикова, Л. В. Базы данных. Разработка приложений для студента [Text] / Рудикова, Л. В. - СПб. : БХВ-Петербург, 2006. - 496 с. : ил. - 30 экз. - ISBN 5-94157-805-9 (30)

7.4 Карпова, И. П. Базы данных. Курс лекций и материалы для практических занятий [Текст] : учеб. пособие для вузов / И. П. Карпова . - СПб. : Питер, 2013. - 240 с.

7.5 Малыхина, М. П. Базы данных: основы, проектирование, использование [Text] : учеб. пособие для вузов / Малыхина, М. П. - СПб. : БХВ-Петербург, 2004. - 499 с. : ил. - 99 экз. - (Учебное пособие). - Библиогр.: с. 491-493 (28 назв.). - ISBN 5-94157-310-4 (98)

7.6 Мотев, А. А. Уроки MySQL [Text] : самоучитель / Мотев, А. А. - СПб. : БХВ-Петербург, 2006. - 208 с. : ил. - 20 экз. - (Самоучитель). - ISBN 5-94157-658-7(20)

Лабораторная работа № 9

Использование специальных предикатов NULL, IN, LIKE, BETWEEN в SQL-запросах

1. **Цель работы**

Изучить SQL-запросы совместно со специальными предикатами в web-интерфейсе phpMyAdmin.

2. **Теоретическая часть**

Специальные предикаты

Предикаты представляют собой выражения, принимающие истинностное значение. Они могут представлять собой как одно выражение, так и любую комбинацию из неограниченного количества выражений, построенную с помощью булевых операторов AND, OR или NOT. Кроме того, в этих комбинациях может использоваться SQL-оператор IS, а также круглые скобки для конкретизации порядка выполнения операций.

Предикат в языке SQL может принимать одно из трех значений TRUE (истина), FALSE (ложь) или UNKNOWN (неизвестно). Исключение составляют следующие предикаты: NULL (отсутствие значения), UNIQUE (уникальность) и MATCH (совпадение), которые не могут принимать значение UNKNOWN.

Предикат Between A and B – принимает значения между A и B. Предикат истинен, когда сравниваемое значение попадает в заданный диапазон, включая границы диапазона. Одновременно в стандарте задан и противоположный предикат Not Between A and B, который истинен тогда, когда сравниваемое значение не попадает в заданный интервал, включая его границы.

Предикат вхождения в множество IN (множество) истинен тогда, когда сравниваемое значение входит в множество заданных значений. При этом множество значений может быть задано простым перечислением или встроенным подзапросом. Одновременно существует противоположный предикат NOT IN (множество), который истинен тогда, когда сравниваемое значение не входит в заданное множество.

Предикаты сравнения с образцом LIKE и NOT LIKE. Предикат LIKE требует задания шаблона, с которым сравнивается заданное значение, предикат истинен, если сравниваемое значение соответствует шаблону, и ложен в противном случае.

Предикат NOT LIKE имеет противоположный смысл. Шаблон может содержать % для обозначения любого числа любых символов; _ для обозначения любого одного символа.

Предикат сравнения с неопределенным значением IS NULL. Для выявления равенства значения некоторого атрибута неопределенному значению применяют специальные стандартные предикаты: <имя атрибута> IS NULL и <имя атрибута> IS NOT NULL

Для таблиц и полей можно задавать псевдонимы (alias). Для этого необходимо использовать предлог AS. Например, Select [Цена за единицу] * [Количество] as [Стоимость покупки] from Продажа; - здесь определяется псевдоним для вычисляемого поля (операция умножение).

3. Подготовка к работе

3.1 Изучить методические указания и рекомендованную литературу.

3.2 Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

4.1 Запустить СУБД MySQL.

4.2 Выбрать созданную ранее базу данных.

4.3 Выбрать вкладку SQL.

4.4 Выполнить 4 запроса, используя предикаты, затем сохранив их в текстовом документе.

4.5 Сделать скриншоты полученных в результате запросов данных.

4.6 Подготовить отчет.

Варианты заданий

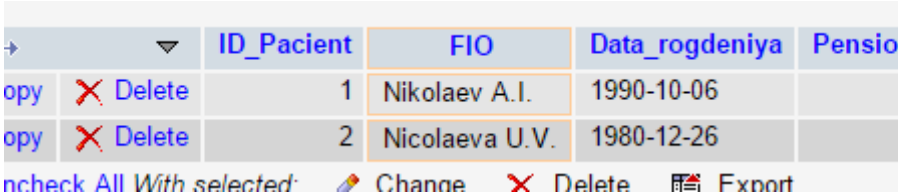
Все лабораторные работы выполняются по одной теме. Вариант задания (тема) выбирается один раз, в лабораторной работе № 3. Тема выбирается согласно номеру студента в списке группы.

Пример выполнения лабораторной работы

Предположим, что необходимо узнать фамилию пациента, но по каким- то причинам, известно лишь только начало. В данной проблеме поможет оператор LIKE:

SELECT * FROM `pacienti` WHERE `FIO` LIKE 'Ni%'

В итоге СУБД вывела две записи:



		ID_Pacient	FIO	Data_rogdeniya	Pensio
opy	Delete	1	Nikolaev A.I.	1990-10-06	
opy	Delete	2	Nicolaeva U.V.	1980-12-26	

ncheck All *With selected:* Change Delete Export

5. Требование к отчету

Структура отчёта зависит от типа задания и индивидуального стиля работы студента. Однако, все работы должны обладать некоторыми общими чертами и включать следующие основные элементы:

- Титульный лист.
- Лист рецензии.
- Содержание.
- Цель лабораторной работы.
- Ответы на контрольные вопросы.
- Введение.
- Теоретическая (практическая) часть. Инфологическая модель базы данных согласно варианту.
- Результаты выполнения запросов (код и скриншоты).
- Заключение (выводы по работе).
- Список источников информации.
- Приложения.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЁТА:

- Текстовый материал должен быть подготовлен с использованием MS Word.
- Текст должен быть набран с учетом следующих параметров:
  1. шрифт Times New Roman;
  2. размер шрифта заголовков – 14, основного текста - 12;
  3. межстрочный интервал - 1;
  4. межбуквенный интервал - Normal;
  5. поля: левое – 2.5, правое – 2, верхнее и нижнее – 1.5;
  6. выравнивание – по ширине страницы;
  7. нумерация страниц – в нижем правом углу;
  8. в верхнем колонтитуле указать название работы, Ф.И.О. автора;
  9. образец оформления титульного листа приведен в приложении №1
- Объем работы не менее 10 страниц.

- Использовать электронное оглавление.
- Рисунки должны быть пронумерованы и подписаны.
- Список источников должен содержать как список литературы, так и список ресурсов Интернет.

6. Контрольные вопросы

6.1 Что такое предикат?

6.2 Для чего используются предикаты?

6.3 Перечислите и опишите известные вам предикаты.

6.4 Опишите принцип работы предикатов.

7. Рекомендуемая литература

7.1 Диго С. М. Базы данных: проектирование и использование [Text] : учебник для вузов / Диго С. М. - М. : Финансы и статистика, 2005. - 592 с. : ил. - 20 экз. - ISBN 5-279-02571-2 (18)

7.2 Андон Ф. Язык запросов SQL. Учебный курс [Text] / Андон Ф. - СПб. : Питер, 2006. - 416 с. : ил. - 30 экз. - ISBN 5-469-00394-9 : Б. ц. УДК 004.6004.65(075) (30)

7.3 Рудикова, Л. В. Базы данных. Разработка приложений для студента [Text] / Рудикова, Л. В. - СПб. : БХВ-Петербург, 2006. - 496 с. : ил. - 30 экз. - ISBN 5-94157-805-9 (30)

7.4 Карпова, И. П. Базы данных. Курс лекций и материалы для практических занятий [Текст] : учеб. пособие для вузов / И. П. Карпова . - СПб. : Питер, 2013. - 240 с.

7.5 Малыхина, М. П. Базы данных: основы, проектирование, использование [Text] : учеб. пособие для вузов / Малыхина, М. П. - СПб. : БХВ-Петербург, 2004. - 499 с. : ил. - 99 экз. - (Учебное пособие). - Библиогр.: с. 491-493 (28 назв.). - ISBN 5-94157-310-4 (98)

7.6 Мотев, А. А. Уроки MySQL [Text] : самоучитель / Мотев, А. А. - СПб. : БХВ-Петербург, 2006. - 208 с. : ил. - 20 экз. - (Самоучитель). - ISBN 5-94157-658-7(20)

Лабораторная работа № 10

SQL-запросы с агрегатными функциями. Группирование записей и использование предикатов вывода

1. Цель работы

Изучить SQL-запросы совместно с агрегатными функциями, а так же группировку записи в web-интерфейсе phpMyAdmin.

2. Теоретическая часть

Довольно часто требуется узнать, сколько записей соответствует тому или иному запросу, какова сумма значений некоторого числового столбца, его максимальное, минимальное и среднее значение. Для этого служат так называемые итоговые (статистические, агрегатные) функции. Агрегатные функции – особый класс функций, применяемых сразу к нескольким записям набора данных, но возвращающим одно значение. Обычно агрегатные функции используются в запросах с группировкой, но также встречается их применение и в запросах без группирования. В этом случае агрегатная функция обрабатывает *все* записи итогового набора.

avg(выражение) Среднее арифметическое значений выражения для всех записей в группе

count(выражение) Количество записей в группе, для которых значение выражения отлично от NULL

max(выражение) Максимальное значение выражения в группе

min(выражение) Минимальное значение выражения в группе

sum(выражение)Сумма всех значений в группе

Термин *выражение* означает любой столбец в итоговом наборе или любое выражение, выполняющее операцию с этим столбцом.

При использовании итоговых функций в списке столбцов в операторе SELECT заголовки соответствующих им столбцов в результатной таблице имеют вид Expr1001, Expr1002 и т.д. (или что-нибудь аналогичное, в зависимости от реализации SQL). Однако возможно задать заголовки для значений итоговых функций и других столбцов по своему усмотрению. Для этого достаточно после имени столбца в операторе SELECT указать выражение вида AS заголовок_столбца.

Count (параметр) – возвращает количество записей, указанных в параметре. Если требуется получить количество всех записей итогового набора, то в качестве параметра следует указать символ звездочки (*). Если в качестве параметра указать имя столбца, то функция вернет количество

записей, в которых этот столбец имеет значения, отличные от NULL. Чтобы узнать, сколько различных значений содержит столбец, перед его именем следует указать ключевое слово DISTINCT.

Например:

```
SELECT count(location) AS set_locs,
        count(ALL location) AS all_locs,
        count(DISTINCT location) AS unique_locs,
        count(*) AS all_rows
FROM subjects;
```

Результат запроса:

```
set_locs all_locs  unique_locs      all_rows
15       15        7          16
```

Примеры использования других агрегатных функций:

```
SELECT AVG(retail) AS средняя_цена
FROM stock;

SELECT MIN(reatail * 28.8) AS
                        минимальная_цена_в_долларах
FROM stock;

SELECT SUM(retail) AS общая_стоимость_редких_книг
FROM stock
WHERE stock < 10;
```

Группировка записей

Предложение GROUP BY в инструкции SELECT задает столбцы, используемые для формирования групп из выбранных строк. Строки каждой группы содержат одно и то же значение заданного столбца (столбцов). Выражение за ключевым словом GROUP BY может быть простым полем таблицы, оно также может представлять собой произвольную операцию с полем. При перечислении нескольких полей или выражений, разделенных запятыми, группировка записей производится по совпадению значений во всех перечисленных выражениях. Появление секции GROUP BY в запросе SQL приводит к тому, что все записи с одинаковым значением выражений, заданных в предложении GROUP BY, группируются в *одну* запись. Если предложение GROUP BY расположено после предложения WHERE, то создаются группы из строк, выбранных после применения WHERE.

Необходимо понимать, что все целевые поля, указанные в секции SELECT, участвующие в запросе с группировкой, но не указанные в секции GROUP BY, доступны лишь при выборке через агрегатную функцию.

Другими словами, при включении предложения GROUP BY в инструкцию SELECT список выбора может состоять только из выражений, указанных в предложении GROUP BY или из агрегатных функций.

Выведем количество книг, хранящихся в базе данных booktown для каждого издательства:

```
SELECT name AS publisher,
        count(isbn) AS number_of_books
FROM editions AS e INNER JOIN publishers AS p
        ON (e.publisher_id = p.id)
GROUP BY name;
```

Секция GROUP BY указывает на то, что записи объединенного набора данных должны группироваться по имени издательства. Все записи с одинаковым названием издательства группируются, после чего функция count() подсчитывает в каждой группе количество непустых значений поля isbn и возвращает результат – количество записей, объединенных в каждую группу для одного издательства.

Получим количество книг, написанных авторами по каждой теме:

```
SELECT last_name, first_name, subject,
count(title) AS number_of_books
FROM books AS b INNER JOIN authors AS a
                ON (b.author_id = a.id)
        INNER JOIN subjects AS s
                ON(b.subject_id = s.id)
GROUP BY last_name, first_name, subject;
```

Отбор групп записей

Предложение HAVING, за которым следует условие отбора, определяет группы строк, которые включаются в результатную таблицу. Условие отбора будет применяться к каждой из групп, сформированных с помощью секции GROUP BY. Если некоторая группа не удовлетворяет условию отбора, то она не включается в результатную таблицу.

Разница между предложениями HAVING и WHERE заключается в том, что условие отбора, заданное в предложении WHERE, применяется к отдельным записям перед объединением их в группы, а условие отбора предложения HAVING применяется к группам строк. Секция WHERE не может содержать агрегатных функций. Условия же секции HAVING, наоборот, основаны на агрегатных функциях, а не на условиях для отдельных записей.

Выведем количество книг, хранящихся в базе данных booktown, для тех издательств, которые представлены двумя и более книгами:

```
SELECT name AS publisher,
```

```
        count(isbn) AS number_of_books
FROM editions AS e INNER JOIN publishers AS p
        ON (e.publisher_id = p.id)
GROUP BY name
HAVING count(isbn)>1;
```

3. Подготовка к работе

3.1 Изучить методические указания и рекомендованную литературу.

3.2 Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

4.1 Запустить СУБД MySQL.

4.2 Выбрать созданную ранее базу данных.

4.3 Выбрать вкладку SQL.

4.4 Выполнить 4 запроса, используя функции агрегирования и группировку записей, затем сохранив их в текстовом документе.

4.5 Сделать скриншоты полученных в результате запросов данных.

4.6 Подготовить отчет.

Варианты заданий

Все лабораторные работы выполняются по одной теме. Вариант задания (тема) выбирается один раз, в лабораторной работе № 3. Тема выбирается согласно номеру студента в списке группы.

Пример выполнения лабораторной работы

В таблице обращений необходимо узнать, сколько человек посетило каждого врача, для этого используется COUNT и группировка записей:

SELECT `ID_Vrach`, count(`ID_Pacient`) FROM `obrasheniya` group by `ID_Vrach`

ID_Vrach	count(`ID_Pacient`)
1	3
2	4
3	3

Данные были выведены, но название столбца с количеством обращений выглядит ненаглядно, а так же нужно условие отбора – все обращения с id услуги = 3 :

SELECT `ID_Vrach`, count(`ID_Pacient`) as 'количество обращений' FROM `obrasheniya` where `ID_Uslug` = 3 group by `ID_Vrach`

ID_Vrach	количество обращений
10	1

5. Требование к отчету

Структура отчёта зависит от типа задания и индивидуального стиля работы студента. Однако, все работы должны обладать некоторыми общими чертами и включать следующие основные элементы:

- Титульный лист.
- Лист рецензии.
- Содержание.
- Цель лабораторной работы.
- Ответы на контрольные вопросы.
- Введение.
- Теоретическая (практическая) часть. Инфологическая модель базы данных согласно варианту.
- Результаты выполнения запросов (код и скриншоты).
- Заключение (выводы по работе).
- Список источников информации.
- Приложения.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЁТА:

- Текстовый материал должен быть подготовлен с использованием MS Word.
- Текст должен быть набран с учетом следующих параметров:
  1. шрифт Times New Roman;
  2. размер шрифта заголовков – 14, основного текста - 12;
  3. межстрочный интервал - 1;
  4. межбуквенный интервал - Normal;
  5. поля: левое – 2.5, правое – 2, верхнее и нижнее – 1.5;
  6. выравнивание – по ширине страницы;
  7. нумерация страниц – в нижем правом углу;
  8. в верхнем колонтитуле указать название работы, Ф.И.О. автора;
  9. образец оформления титульного листа приведен в приложении №1
- Объем работы не менее 10 страниц.
- Использовать электронное оглавление.
- Рисунки должны быть пронумерованы и подписаны.
- Список источников должен содержать как список литературы, так и список ресурсов Интернет.

6. Контрольные вопросы

6.1 Какие еще бывают функции агрегирования?

6.2 Для чего необходимы функции агрегирования?

6.3 Если группировать записи, но не выполнять с ними каких-то действий, что за информацию нам выдаст СУБД? Почему?

6.4 Можно ли в HAVING работать с агрегатными записями? Как?

7. Рекомендуемая литература

7.1 Диго С. М. Базы данных: проектирование и использование [Text] : учебник для вузов / Диго С. М. - М. : Финансы и статистика, 2005. - 592 с. : ил. - 20 экз. - ISBN 5-279-02571-2 (18)

7.2 Андон Ф. Язык запросов SQL. Учебный курс [Text] / Андон Ф. - СПб. : Питер, 2006. - 416 с. : ил. - 30 экз. - ISBN 5-469-00394-9 : Б. ц. УДК 004.6004.65(075) (30)

7.3 Рудикова, Л. В. Базы данных. Разработка приложений для студента [Text] / Рудикова, Л. В. - СПб. : БХВ-Петербург, 2006. - 496 с. : ил. - 30 экз. - ISBN 5-94157-805-9 (30)

7.4 Карпова, И. П. Базы данных. Курс лекций и материалы для практических занятий [Текст] : учеб. пособие для вузов / И. П. Карпова . - СПб. : Питер, 2013. - 240 с.

7.5 Малыхина, М. П. Базы данных: основы, проектирование, использование [Text] : учеб. пособие для вузов / Малыхина, М. П. - СПб. : БХВ-Петербург, 2004. - 499 с. : ил. - 99 экз. - (Учебное пособие). - Библиогр.: с. 491-493 (28 назв.). - ISBN 5-94157-310-4 (98)

7.6 Мотев, А. А. Уроки MySQL [Text] : самоучитель / Мотев, А. А. - СПб. : БХВ-Петербург, 2006. - 208 с. : ил. - 20 экз. - (Самоучитель). - ISBN 5-94157-658-7(20)

Лабораторная работа № 11

Вложенные SQL-запросы. Использование оператора EXISTS. Использование операторов ANY, ALL И SOME

1. Цель работы

Изучить вложенные SQL-запросы, использовать операторы EXISTS, ANY и ALL.

2. Теоретическая часть

Подзапрос или вложенный запрос – это запрос на выборку данных, вложенный в другой запрос.

В свою очередь подзапрос может содержать другой подзапрос, но в первую очередь выполняется подзапрос, имеющий самый глубокий уровень вложения.

Часто, но не всегда, внешний запрос обращается к одной таблице, а подзапрос - к другой.

Подзапросы характеризуются тем, что они формально никак не связаны с содержащими их внешними запросами, что позволяет сначала выполнить подзапрос, результат которого используется для выполнения внешнего запроса.

Три вида подзапросов:

- подзапросы, возвращающие единственное значение;
- подзапросы, возвращающие список значений, из одного столбца таблицы;
- подзапросы, возвращающие набор записей.

Подзапросы, возвращающие единственное значение

Вывести оценки, которые больше среднего значения всех оценок

SELECT * FROM USPEV WHERE OCENKA > (SELECT AVG(OCENKA) FROM USPEV)

Подзапросы, возвращающие список значений, из одного столбца таблицы. Определить коды родителей студента Маркова и Иванова.

SELECT KOD_RODITEL FROM RODDETI WHERE KOD_STUDENT IN (SELECT KOD_STUDENT FROM DANNIE WHERE FAM='МАРКОВ' OR FAM='ИВАНОВ’)

Определить название дисциплин, которые сдавал студент с кодом 2.

SELECT NAZVANIE FROM DISCHIPLINA WHERE KOD_DISCHIPLINA IN (SELECT KOD_DISCHIPLINA FROM USPEV WHERE KOD_STUDENT='2')

Вывести фамилию, и место работы родителей студента с кодом 3.

SELECT FIO_ROD, RABOTA FROM RODITELI WHERE KOD_RODITEL IN (SELECT KOD_RODITEL FROM RODDETI WHERE KOD_STUDENT=3)

Использование операций EXISTS и NOT EXISTS

Ключевые слова *EXISTS* и NOT EXISTS предназначены для использования только совместно с *подзапросами*. Результат их обработки представляет собой логическое значение TRUE или FALSE. Для ключевого слова *EXISTS* результат равен TRUE в том и только в том случае, если в возвращаемой *подзапросом* результирующей таблице присутствует хотя бы одна строка. Если результирующая таблица *подзапроса* пуста, результатом обработки операции *EXISTS* будет значение FALSE. Для ключевого слова NOT EXISTS используются правила обратные по отношению к ключевому обработки, слову *EXISTS.* Поскольку по ключевым словам *EXISTS* и NOT EXISTS проверяется лишь наличие строк в результирующей таблице *подзапроса*, то эта таблица может содержать произвольное количество столбцов.

Список групп, не сдающих философию.

SELECT num_gr, form FROM groups

WHERE NOT EXISTS (SELECT ex_id FROM exams,subjects

WHERE exams.subj_id=subjects.subj_id AND groups.gr_id=exams.gr_id

AND subj_name='Философия')

Использование ключевых слов ANY и ALL

Ключевые слова ANY и ALL могут использоваться с *подзапросами*, возвращающими один столбец чисел.

Если *подзапросу* будет предшествовать ключевое слово ALL, условие сравнения считается выполненным, только когда оно выполняется для всех значений в результирующем столбце *подзапроса*.

Если записи *подзапроса* предшествует ключевое слово ANY, то условие сравнения считается выполненным, когда оно выполняется хотя бы для одного из значений в результирующем столбце *подзапроса*.

Если в результате выполнения *подзапроса* получено пустое значение, то для ключевого слова ALL условие сравнения будет считаться выполненным, а для ключевого слова ANY – невыполненным.

Примеры.

Выбор студентов-отличников (при условии, что у каждого студента есть хотя бы одна оценка).

```
SELECT s.surname, s.name,s.patron FROM student s
WHERE 5=ALL (SELECT mark FROM rating r WHERE s.stud_id=r.stud.id)
```

Выбор студентов, получивших хотя бы одну пятерку.

```
SELECT s.surname, s.name,s.patron FROM student s
WHERE 5=ANY (SELECT mark FROM rating r WHERE s.stud_id=r.stud.id)
```

3. Подготовка к работе

3.1 Изучить методические указания и рекомендованную литературу.

3.2 Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

4.1 Запустить СУБД MySQL.

4.2 Выбрать созданную ранее базу данных.

4.3 Выбрать вкладку SQL.

4.4 Выполнить 4 запроса, используя вложенные запросы, а так же операторы ANY и ALL, затем сохранив их в текстовом документе.

4.5 Сделать скриншоты полученных в результате запросов данных.

4.6 Подготовить отчет.

Варианты заданий

Все лабораторные работы выполняются по одной теме. Вариант задания (тема) выбирается один раз, в лабораторной работе № 3. Тема выбирается согласно номеру студента в списке группы.

5. Требование к отчету

Структура отчёта зависит от типа задания и индивидуального стиля работы студента. Однако, все работы должны обладать некоторыми общими чертами и включать следующие основные элементы:

- Титульный лист.
- Лист рецензии.
- Содержание.
- Цель лабораторной работы.
- Ответы на контрольные вопросы.
- Введение.

- Теоретическая (практическая) часть. Инфологическая модель базы данных согласно варианту.
- Результаты выполнения запросов (код и скриншоты).
- Заключение (выводы по работе).
- Список источников информации.
- Приложения.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЁТА:

- Текстовый материал должен быть подготовлен с использованием MS Word.
- Текст должен быть набран с учетом следующих параметров:
  1. шрифт Times New Roman;
  2. размер шрифта заголовков – 14, основного текста - 12;
  3. межстрочный интервал - 1;
  4. межбуквенный интервал - Normal;
  5. поля: левое – 2.5, правое – 2, верхнее и нижнее – 1.5;
  6. выравнивание – по ширине страницы;
  7. нумерация страниц – в нижем правом углу;
  8. в верхнем колонтитуле указать название работы, Ф.И.О. автора;
  9. образец оформления титульного листа приведен в приложении №1
- Объем работы не менее 10 страниц.
- Использовать электронное оглавление.
- Рисунки должны быть пронумерованы и подписаны.
- Список источников должен содержать как список литературы, так и список ресурсов Интернет.

6. Контрольные вопросы

6.1 Что такое вложенный запрос?

6.2 В каком порядке выполняется обработка данных при использовании вложенного запроса?

6.3 Объясните отличие EXISTS от ANY.

6.4 Что такое оператор SOME?

7. Рекомендуемая литература

7.1 Диго С. М. Базы данных: проектирование и использование [Text] : учебник для вузов / Диго С. М. - М. : Финансы и статистика, 2005. - 592 с. : ил. - 20 экз. - ISBN 5-279-02571-2 (18)

7.2 Андон Ф. Язык запросов SQL. Учебный курс [Text] / Андон Ф. - СПб. : Питер, 2006. - 416 с. : ил. - 30 экз. - ISBN 5-469-00394-9 : Б. ц. УДК 004.6004.65(075) (30)

7.3 Рудикова, Л. В. Базы данных. Разработка приложений для студента [Text] / Рудикова, Л. В. - СПб. : БХВ-Петербург, 2006. - 496 с. : ил. - 30 экз. - ISBN 5-94157-805-9 (30)

7.4 Карпова, И. П. Базы данных. Курс лекций и материалы для практических занятий [Текст] : учеб. пособие для вузов / И. П. Карпова . - СПб. : Питер, 2013. - 240 с.

7.5 Малыхина, М. П. Базы данных: основы, проектирование, использование [Text] : учеб. пособие для вузов / Малыхина, М. П. - СПб. : БХВ-Петербург, 2004. - 499 с. : ил. - 99 экз. - (Учебное пособие). - Библиогр.: с. 491-493 (28 назв.). - ISBN 5-94157-310-4 (98)

7.6 Мотев, А. А. Уроки MySQL [Text] : самоучитель / Мотев, А. А. - СПб. : БХВ-Петербург, 2006. - 208 с. : ил. - 20 экз. - (Самоучитель). - ISBN 5-94157-658-7(20)

Лабораторная работа № 12
Сложные SQL-запросы на объединение нескольких таблиц. Комбинированные запросы

1. Цель работы

Изучить сложные SQL-запросы на объединение нескольких таблиц.

2. Теоретическая часть

В секции FROM указывается источник данных – таблица или итоговый набор. Секция может содержать несколько источников, разделенных запятыми. Результат подобного перечисления функционально эквивалентен перекрестному объединению.

Использование нескольких источников данных требует осторожности. В результате выполнения команды SELECT для нескольких источников без секций WHERE и JOIN, уточняющих связи между источниками, возвращается полное декартово произведение источников. Итоговый набор будет содержать все возможные комбинации записей из всех источников.

Обычно для уточнения связей между источниками, перечисленными через запятую в секции FROM, используется секция WHERE.

Например, получим сведения о книгах и их авторах:

```
SELECT books.id, title, authors.id, last_name
FROM books, authors
WHERE books.author_id = authors.id;
```

При использовании ссылок на имена полей, относящихся к разным источникам, может возникнуть неоднозначность. Предположим, команда SELECT получает исходные данные из таблиц books и authors. В каждой из этих таблиц имеется поле с именем id. Без дополнительных уточнений невозможно определить, к какой таблице относится ссылка на поле id в следующей команде:

```
SELECT id
FROM books, authors;
```

Для предотвращения неоднозначности в "полные" имена столбцов включается имя таблицы. При этом используется специальный синтаксис, называемый точечной записью (название связано с тем, что имя таблицы отделяется от имени поля точкой). Например, books.id означает поле id таблицы books.

Точечная запись обязательна только при наличии неоднозначности между наборами данных. Ссылка может состоять только из имени поля – при условии, что это имя уникально во всех наборах данных, перечисленных в секции FROM. В приведенном выше примере поле title присутствует только в таблице books, а поле last_name входит только в таблицу authors, поэтому на их имена можно ссылаться без уточнения.

Источникам данных в секции FROM – таблицам, подзапросам и т.д. – можно назначать синонимы в секции AS (по аналогии с отдельными полями). Синоним часто используется для упрощения точечной записи. Наличие синонима для набора данных позволяет обращаться к нему при помощи точечной записи, что делает команды SQL более компактными и наглядными.

Например, получим сведения о книгах и их авторах с упрощением точечной записи при помощи секции AS:

```
SELECT b.id, title, a.id, last_name
FROM books AS b, authors AS a
WHERE b.author_id = a.id;
```

Ключевое слово AS не является обязательным при назначении синонима:

```
SELECT b.id, title, a.id, last_name
FROM books b, authors a
WHERE b.author_id = a.id;
```

Операции соединения

Операции соединения наборов записей возвращают таблицы, записи в которых получаются путем некоторой комбинации записей соединяемых таблиц. Для этого используется оператор JOIN.

Довольно часто операции, основанные на операторе JOIN, называют объединением таблиц (наборов записей). Однако термин "объединение" лучше подходит для UNION – оператора теоретико-множественного объединения записей, при котором записи исходных наборов не комбинируются (не соединяются) друг с другом, а просто к одному набору записей добавляется другой набор. В случае оператора JOIN в результатную таблицу попадают записи, полученные из разных наборов путем присоединения одной из них к другой. Поэтому операции, основанные на операторе JOIN, будем называть операциями соединения таблиц (наборов записей).

Перекрестное соединение

Существуют несколько разновидностей соединения, которым соответствуют определенные ключевые слова, добавляемые к слову JOIN. Так, например, декартово произведение является операцией перекрестного соединения. В SQL-выражении для обозначения этой операции используется оператор CROSS JOIN. Результат перекрестного соединения принципиально не отличается от перечисления источников через запятую.

Пусть в базе данных имеются следующие две таблицы:

Сотрудники (Номер_сотрудника, Фамилия, Имя, Номер_отдела);

Отделы (Номер_отдела, Название).

Общим столбцом для этих таблиц является Номер_отдела. Декартово произведение этих таблиц получается с помощью следующих эквивалентных запросов:

```
SELECT *
FROM Сотрудники CROSS JOIN Отделы;
```

или

```
SELECT *
FROM Сотрудники, Отделы;
```

На следующих рисунках показаны примеры таблиц Сотрудники и Отделы, а также результат их декартового произведения.

Сотрудники			
Номер_сотрудника	Фамилия	Имя	Номер_отдела
1	Иванов	Иван	1
2	Петров	Петр	2
3	Сидоров	Сидор	1

Отделы	
Номер_отдела	Название
1	Отдел автоматизации
2	Отдел кадров

Декартово произведение					
Номер_ сотрудника	Фамилия	Имя	Сотрудники. Номер_ отдела	Отделы. Номер_ отдела	Название
1	Иванов	Иван	1	1	Отдел автоматизации
1	Иванов	Иван	1	2	Отдел кадров

2	Петров	Петр	2	1	Отдел автоматизации
2	Петров	Петр	2	2	Отдел кадров
3	Сидоров	Сидор	1	1	Отдел автоматизации
3	Сидоров	Сидор	1	2	Отдел кадров

Синтаксис CROSS JOIN всего лишь более формально выражает отношение между двумя наборами данных. Между синтаксисом CROSS JOIN и простым перечислением таблиц через запятую нет никаких функциональных различий.

В основе любого соединения наборов записей лежит операция их декартового произведения.

Соединение по именам столбцов

Соединение по именам столбцов похоже на естественное соединение. Отличие состоит в том, что можно указать, какие одноименные столбцы должны проверяться. В естественном соединении проверяются *все* одноименные столбцы.

Допустим, имеются две таблицы с одинаковыми структурами:

Коробки (Размер, Количество, Цвет);

Крышки (Размер, Количество, Цвет).

Предположим, нам нужны комплекты, в каждом из которых количества коробок и крышек одного размера совпадают, а их цвета могут быть различными. Естественное соединение в этом случае не подойдет, поскольку в нем проверяются все одноименные столбцы, а потому в таблицу результатов не попадут комплекты из разноцветных коробок и крышек. Поэтому следует использовать соединение по одинаковым именам только столбцов Размер и Количество:

```
SELECT *
FROM Коробки JOIN Крышки USING (Размер, Количество);
```

После ключевого слова USING в круглых скобках указывается список одноименных столбцов соединяемых таблиц, которые необходимо проверять.

Приведенный выше запрос можно сформулировать иначе:

```
SELECT *
FROM Коробки, Крышки
```

WHERE (Коробки.Размер = Крышки.Размер)

AND (Коробки.Количество = Крышки.Количество);

На следующих рисунках показаны таблицы Коробки и Крышки, а также результат рассмотренного запроса.

Коробки		
Размер	Количество	Цвет
30×20	13	Белый
30×30	7	Белый
30×15	20	Синий
20×20	26	Красный
20×20	29	Белый

Крышки		
Размер	Количество	Цвет
30×20	13	Белый
30×30	3	Белый
30×15	20	Синий
20×20	26	Желтый
20×20	28	Красный

Комплекты					
Размер	Количество	Цвет	Размер	Количество	Цвет
30×20	13	Белый	30×20	13	Белый
30×15	20	Синий	30×15	20	Синий
20×20	26	Красный	20×20	26	Желтый

Условное соединение

Условное соединение позволяет соединить таблицы по произвольным столбцам, не обязательно имеющим одинаковые имена. Кроме того, в качестве условия соединения может выступать не только равенство, но вообще любое логическое выражение, которое записывается после ключевого слова ON. Если условие выполняется для текущей записи декартового произведения исходных таблиц, то она входит в результатную таблицу.

Пусть в базе данных имеются следующие две таблицы:

Сотрудники (Номер, Фамилия, Имя, Номер_отдела);

Отделы (Номер, Название).

Тогда эти таблицы можно соединить:

SELECT *

FROM Сотрудники JOIN Клиенты

ON (Номер_отдела = Отделы.Номер);

Допустимо также использовать для условного соединения инструкцию INNER JOIN ON.

Конструкция JOIN была включена в стандарт SQL для того, чтобы условия соединения источников данных (условия ON) можно было

отличить от условий принадлежности записей к итоговому набору (условия WHERE).

Например:

```
SELECT *
FROM Сотрудники, Клиенты
WHERE (Номер_отдела = Отделы.Номер)
        AND (Фамилия = 'Иванов');
```

и

```
SELECT *
FROM Сотрудники JOIN Клиенты
        ON (Номер_отдела = Отделы.Номер)
WHERE Фамилия = 'Иванов';
```

Две синтаксические формы функционально идентичны и возвращают одинаковые результаты. Синтаксис JOIN позволяет отделить критерий связи источников от критерия выбора записей, поскольку связи определяются только в секции ON. Это существенно упрощает чтение и модификацию запросов, так как не приходится разбираться в смысле каждого условия в секции WHERE.

Сложные соединения

Хотя одна секция JOIN соединяет всего два набора данных, на практике соединения не ограничиваются двумя источниками. За набором, созданным посредством соединения, может следовать новая секция JOIN – по аналогии с тем, как перечисляются через запятую источники данных.

Пример соединения нескольких источников данных:

```
SELECT last_name, publisher, e.isbn, subject
FROM authors AS a
        JOIN books AS b ON a.id = b.author_id
        JOIN editions AS e ON b.id = e.book_id
        JOIN publishers AS p ON e.publisher_id = p.id
        JOIN subjects AS s ON b.subject_id = s.id;
```

Хотя таблица books участвует в соединении, ни одно из ее полей не входит в итоговый набор. Включение таблицы books в секции JOIN предоставляет критерии для соединения других таблиц. Все таблицы, поля которых возвращаются в результате запроса, связываются с другими таблицами через поле id таблицы books (кроме таблицы publishers, которая связывается с таблицей editions по полю publisher_id).

Объединение наборов записей

Нередко требуется объединить наборы записей двух или более таблиц с похожими структурами в одну таблицу. Иначе говоря, к набору записей, возвращаемых одним запросом, требуется добавить записи, возвращаемые другим запросом. Для этого служит оператор UNION:

```
Запрос1
UNION
Запрос2;
```

При этом в результатной таблице остаются только отличающиеся записи. Чтобы сохранить в ней все записи, следует после оператора UNION написать ключевое слово ALL.

Например, таблицы Коробки и Крышки имеют однотипные столбцы Размер, Количество и Цвет. Тогда, чтобы получить общий список данных и о коробках, и о крышках, достаточно выполнить следующий запрос:

```
SELECT Размер, Количество, Цвет
FROM Коробки
UNION
SELECT Размер, Количество, Цвет
FROM Крышки;
```

Оператор UNION можно применять только к таблицам, удовлетворяющим следующим условиям совместимости:

количества столбцов объединяемых таблиц должны быть равны;

данные в соответствующих столбцах объединяемых таблиц должны иметь совместимые типы данных. Например, символьные (строковые) типы CHAR и VARCHAR совместимы, а числовой и строковый типы не совместимы.

Имена соответствующих столбцов и их размеры могут быть различными. Важно, чтобы количества столбцов были равны, а их типы были совместимы.

Пусть требуется получить сведения о том, в какие пункты можно попасть, сделав не более одной пересадки (т.е. без пересадок или с одной пересадкой). Для этого достаточно объединить записи исходной таблицы Рейсы с результатом запроса о достижимости через один промежуточный пункт:

```
SELECT Начальный_пункт, Конечный_пункт
FROM Рейсы
UNION
SELECT T1.Начальный_пункт, T2.Конечный_пункт
FROM Рейсы T1, Рейсы T2
WHERE T1.Конечный_пункт = T2.Начальный_пункт;
```

Внешние соединение

Все соединения таблиц, рассмотренные ранее, являются *внутренними*. Во всех примерах вместо ключевого слова JOIN можно писать INNER JOIN. Из таблицы, получаемой при внутреннем соединении, удаляются все записи, у которых нет соответствующих строк одновременно в обеих исходных таблицах. При *внешнем* соединении (OUTER JOIN) несоответствующие строки сохраняются. В этом и заключается отличие внешнего соединения от внутреннего.

В запросе, имеющем соединение, будем называть таблицу *левой*, если ее имя в операторе запроса предшествует ключевому слову JOIN, и *правой*, если ее имя следует за словом JOIN.

Внешнее соединение может сохранить записи, для которых не находится соответствия в другом наборе. В этом случае недостающие поля заполняются значением NULL. Решение о том, войдет ли такая запись в результат внешнего соединения, зависит от того, в каком из соединяемых наборов (таблиц) отсутствуют данные, и от типа внешнего соединения.

Существуют три разновидности внешних соединений. Использоваться в данной лабораторной работе будет лишь две.

Левое внешнее соединение

Операция LEFT JOIN (LEFT OUTER JOIN) возвращает все строки из левой таблицы, соединенные с теми строками из правой таблицы, для которых выполняется условие соединения. Если во второй таблице нет таких строк, то в качестве значений столбцов правой таблицы будут установлены значения NULL.

В базе данных booktown в таблице books содержится общая информация о книгах, а в таблице editions хранятся данные, относящиеся к конкретному изданию – код ISBN, издатель и дата публикации. В таблицу editions входит поле book_id, связывающее ее с полем id, которое является первичным ключом таблицы books.

Допустим, требуется информация о каждой книге вместе со всеми имеющимися кодами ISBN:

```
SELECT title, isbn
FROM books INNER JOIN editions
        ON (books.id = editions.book_id);
```

Если у книги нет печатного издания (или информация об этом издании еще не занесена в базу данных), информация о ней не будет включена в результат данного запроса.

Чтобы получить данные о *каждой* книге, следует воспользоваться следующим запросом:

```
SELECT title, isbn
FROM books LEFT OUTER JOIN editions
        ON (books.id = editions.book_id);
```

Теперь в итоговом наборе будут присутствовать и те книги, у которых отсутствуют коды ISBN. В этом запросе использовано левое внешнее соединение. Выбор объясняется тем, что запрос должен вернуть названия книг, для которых существуют или не существуют коды ISBN. Поскольку таблица books стоит слева от ключевого слова JOIN, задача решается при помощи левого внешнего соединения.

Правое внешнее соединение

Операция RIGHT JOIN (RIGHT OUTER JOIN) возвращает все строки из правой таблицы, соединенные с теми строками из левой таблицы, для которых выполняется условие соединения. Если во второй таблице нет таких строк, то в качестве значений столбцов правой таблицы будут установлены значения NULL.

В качестве примера получим названия книг, у которых нет кодов ISBN:

```
SELECT title
FROM editions RIGHT JOIN books
        ON (editions.book_id = books.id)
WHERE isbn IS NULL;
```

3. Подготовка к работе

3.1 Изучить методические указания и рекомендованную литературу.

3.2 Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

4.1 Запустить СУБД MySQL.

4.2 Выбрать созданную прежде базу данных.

4.3 Выбрать вкладку SQL.

4.4 Составить 4 сложных запроса, применив внутреннее, внешнее соединение.

4.5 Выполнить 2 произвольных запроса, использовав множественные операцииСделать скриншоты, полученных в результате запросов данных.

4.6 Подготовить отчет.

Варианты заданий

Все лабораторные работы выполняются по одной теме. Вариант задания (тема) выбирается один раз, в лабораторной работе № 3. Тема выбирается согласно номеру студента в списке группы.

Пример выполнения лабораторной работы

Учитывая, что информация в таблицах после нормализации уникальна, для получения полной информации нужно использовать несколько таблиц, которые могут быть связаны лишь косвенно с собой.

Существует таблица обращения, которая связана с таблицами врачи и пациенты. Нам необходимо вывести всех пациентов, кто был на приеме у психолога. Для этого все таблицы объединяем вместе и добавляем условие:

```
SELECT `pacienti`.`FIO`, `vrachi`.`FIO`, `vrachi`.`Specialnost`
 FROM `pacienti` inner join `obrasheniya` on `pacienti`.`ID_Pacient` = `obrasheniya`.`ID_Pacient` inner join `vrachi` on `vrachi`.`ID_Vrach`=`obrasheniya`.`ID_Vrach` WHERE `Specialnost` = 'Psiholog'
```

FIO	FIO	Specialnost
Ushakov V.I.	Tomina T.N.	Psiholog
Malinina T.E.	Tomina T.N.	Psiholog
Letnokov T.I.	Tomina T.N.	Psiholog

5. Требование к отчету

Структура отчёта зависит от типа задания и индивидуального стиля работы студента. Однако, все работы должны обладать некоторыми общими чертами и включать следующие основные элементы:

- Титульный лист.
- Лист рецензии.
- Содержание.
- Цель лабораторной работы.
- Ответы на контрольные вопросы.
- Введение.
- Теоретическая (практическая) часть. Инфологическая модель базы данных согласно варианту.
- Результаты выполнения запросов (код и скриншоты).

- Заключение (выводы по работе).
- Список источников информации.
- Приложения.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЁТА:

- Текстовый материал должен быть подготовлен с использованием MS Word.
- Текст должен быть набран с учетом следующих параметров:
  1. шрифт Times New Roman;
  2. размер шрифта заголовков – 14, основного текста - 12;
  3. межстрочный интервал - 1;
  4. межбуквенный интервал - Normal;
  5. поля: левое – 2.5, правое – 2, верхнее и нижнее – 1.5;
  6. выравнивание – по ширине страницы;
  7. нумерация страниц – в нижем правом углу;
  8. в верхнем колонтитуле указать название работы, Ф.И.О. автора;
  9. образец оформления титульного листа приведен в приложении №1
- Объем работы не менее 10 страниц.
- Использовать электронное оглавление.
- Рисунки должны быть пронумерованы и подписаны.
- Список источников должен содержать как список литературы, так и список ресурсов Интернет.

6. Контрольные вопросы

6.1 Какое отличие внешнего соединения от внутреннего?

6.2 В каком порядке выполняется обработка данных при использовании вложенного запроса?

6.3 Что такое ALL UNION?

6.4 Что такое декартово множество, применительно реляционных баз данных?

7. Рекомендуемая литература

7.1 Диго С. М. Базы данных: проектирование и использование [Text] : учебник для вузов / Диго С. М. - М. : Финансы и статистика, 2005. - 592 с. : ил. - 20 экз. - ISBN 5-279-02571-2 (18)

7.2 Андон Ф. Язык запросов SQL. Учебный курс [Text] / Андон Ф. - СПб. : Питер, 2006. - 416 с. : ил. - 30 экз. - ISBN 5-469-00394-9 : Б. ц. УДК 004.6004.65(075) (30)

7.3 Рудикова, Л. В. Базы данных. Разработка приложений для студента [Text] / Рудикова, Л. В. - СПб. : БХВ-Петербург, 2006. - 496 с. : ил. - 30 экз. - ISBN 5-94157-805-9 (30)

7.4 Карпова, И. П. Базы данных. Курс лекций и материалы для практических занятий [Текст] : учеб. пособие для вузов / И. П. Карпова . - СПб. : Питер, 2013. - 240 с.

7.5 Малыхина, М. П. Базы данных: основы, проектирование, использование [Text] : учеб. пособие для вузов / Малыхина, М. П. - СПб. : БХВ-Петербург, 2004. - 499 с. : ил. - 99 экз. - (Учебное пособие). - Библиогр.: с. 491-493 (28 назв.). - ISBN 5-94157-310-4 (98)

7.6 Мотев, А. А. Уроки MySQL [Text] : самоучитель / Мотев, А. А. - СПб. : БХВ-Петербург, 2006. - 208 с. : ил. - 20 экз. - (Самоучитель). - ISBN 5-94157-658-7(20)

Лабораторная работа № 13

SQL-запросы на добавление, модификацию и удаление данных. Использование со вложенными запросами

1. Цель работы

Изучить команды изменения данных.

2. **Теоретическая часть**

При создании и дальнейшем сопровождении базы данных обычно возникает задача добавления новых и удаления ненужных записей, а также изменения содержимого ячеек таблицы. В SQL для этого предусмотрены операторы INSERT (вставить), DELETE (удалить) и UPDATE (изменить). Запросы, начинающиеся с этих ключевых слов, не возвращают данные в виде виртуальной таблицы, а изменяют содержимое уже существующих таблиц базы данных. Запросы на модификацию данных могут содержать вложенные запросы на выборку данных из той же самой таблицы или из других таблицы, однако сами не могут быть вложены в другие запросы.

Добавление новых записей

Для добавления (вставки) записи в таблицу служит оператор INSERT, который имеет несколько форм:

INSERT INTO имя_Таблицы VALUES (список_Значений) – вставляет пустую запись в указанную таблицу и заполняет эту запись значениями из списка, указанного за ключевым словом VALUES. При этом первое в списке значение вводится в первый столбец таблицы, второе значение – во второй столбец и т.д. Порядок столбцов задается при создании таблицы. Данная форма оператора INSERT не очень надежна, поскольку нетрудно ошибиться в порядке вводимых значений.

INSERT INTO имя_Таблицы (список_Столбцов) VALUES (список_Значений) – вставляет пустую запись в указанную таблицу и вводит в заданные столбцы значения из указанного списка. При этом в первый столбец из список_Столбцов вводится первое значение из список_Значений, во второй столбец – второе значение и т.д. Порядок имен в списке можеть отличаться от их порядка, заданного при создании таблицы. Столбцы, которые не указаны в списке, заполняются значениями NULL.

INSERT INTO имя_Таблицы (список_Столбцов) SELECT ... – вставляет в указанную таблицу записи, возвращаемые запросом на выборку. На практике нередко требуется загрузить в одну таблицу данные из другой таблицы. Например,

INSERT INTO books

(id, title, author_id, subject_id)

```
SELECT book_id, title, author_id, subject_id
FROM book_queue
WHERE subject_id = 4;
```

Удаление записей

Для удаления записей из таблицы применяется оператор DELETE (удалить):

```
DELETE
FROM имя_Таблицы
WHERE условие;
```

Данный оператор удаляет из указанной таблицы записи (а не отдельные значения столбцов), которые удовлетворяют указанному условию.

Следующий запрос удаляет записи из таблицы stock, в которых значение столбца stock равно нулю. Если подходящих записей несколько, все они будут удалены.

```
DELETE
FROM stock
WHERE stock = 0;
```

В операторе WHERE может находиться подзапрос на выборку данных (оператор SELECT). Подзапросы в операторе DELETE работают точно так же, как и в операторе SELECT.

Операция удаления записей из таблицы является опасной в том смысле, что связана с риском необратимых потерь данных в случае ошибок. Чтобы избежать неприятностей, перед удалением записей рекомендуется сначала выполнить соответствующий запрос на выборку, чтобы просмотреть, какие записи будут удалены. Так, например, перед выполнением рассмотренного ранее запроса на удаление не помешает выполнить соответствующий запрос на выборку:

```
SELECT *
FROM stock
WHERE stock = 0;
```

Для удаления *всех* записей из таблицы достаточно использовать оператор DELETE без ключевого слова WHERE. При этом сама таблица со всеми определенными в ней столбцами остается и готова для вставки новых записей. Например,

```
DELETE
FROM stock;
```

Изменение данных

Для изменения значений столбцов таблицы применяется оператор UPDATE (изменить, обновить). Чтобы изменить значения в одном столбце таблицы в тех записях, которые удовлетворяют некоторому условию, следует выполнить такой запрос:

```
UPDATE имя_Таблицы
SET имя_Столбца = значение
WHERE условие;
```

За ключевым словом SET (установить) следует выражение равенства, в левой части которого указывается имя столбца, а в правой – выражение, значение которого следует сделать значением данного столбца. Эти установки будут выполнены в тех записях, которые удовлетворяют условию в операторе WHERE.

Чтобы одним оператором UPDATE установить новые значения сразу для нескольких столбцов, вслед за ключевым словом SET записываются соответствующие выражения равенства, разделенные запятыми. Например:

```
UPDATE publishers
SET name = 'O\'Reilly & Associates',
address = 'O\'Reilly & Associates, Inc. '
                || '101 Morris St, Sebastopol, CA 95472'
WHERE id = 113;
```

Использование оператора WHERE в операторе UPDATE не обязательно. Если он отсутствует, то указанные в SET изменения будут произведены для всех записей таблицы.

Операция изменения записей, как и их удаление, связана с риском необратимых потерь данных в случае ошибок. Чтобы избежать подобных неприятностей, перед обновлением записей рекомендуется выполнить соответствующий запрос на выборку, чтобы просмотреть, какие записи будут изменены. Так, например, перед выполнением приведенного ранее запроса на обновление данных не помешает выполнить соответствующий запрос на выборку:

```
SELECT *
FROM publishers
WHERE id = 113;
```

Условие в операторе WHERE может содержать подзапросы, в том числе и связанные.

Некоторые СУБД (например, PostgreSQL) имеют расширение стандарта SQL, позволяющее обновлять одну таблицу данными из другой. В этом случае команда UPDATE дополняется поддержкой секции FROM.

Секция FROM позволяет получать входные данные из других наборов данных (таблиц и подзапросов).

Например, обновим данные таблицы stock по данным таблицы stock_backup:

```
UPDATE stock
SET retail = stock_backup.retail
FROM stock_backup
WHERE stock.isbn = stock_backup.isbn;
```

Секция WHERE описывает связь между обновляемой таблицей и источником. Каждый раз, когда в таблицах находятся совпадающие значения isbn, поле retail в таблице stock обновляется значением из резервной таблицы stock_backup.

Секция FROM поддерживает все разновидности синтаксиса JOIN, что открывает широкие возможности обновления данных в существующих наборах.

3. Подготовка к работе

3.1 Изучить методические указания и рекомендованную литературу.

3.2 Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

4.1 Запустить СУБД MySQL.

4.2 Выбрать созданную ранее базу данных.

4.3 Выбрать вкладку SQL.

4.4 Дополнить БД новой таблицей, и с помощью команд модификаций и (или) вложенных запросов выполнить в ней 4 SQL-запроса.

4.5 Сделать скриншоты полученных в результате запросов данных.

4.6 Подготовить отчет.

Варианты заданий

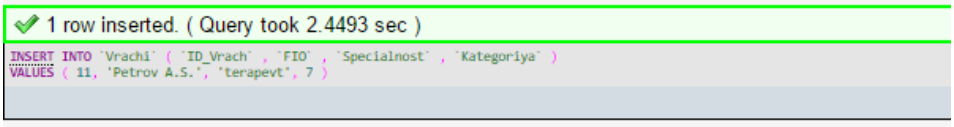
Все лабораторные работы выполняются по одной теме. Вариант задания (тема) выбирается один раз, в лабораторной работе № 3. Тема выбирается согласно номеру студента в списке группы.

Пример выполнения лабораторной работы

В таблице врачи хранится 10 записей. Нужно добавить еще одну запись, в которой будет хранится информация о новом враче, для этого напишем:

INSERT INTO `Vrachi` (`ID_Vrach`, `FIO`, `Specialnost`, `Kategoriya`) VALUES (11, 'Petrov A.S.', 'terapevt', 7)

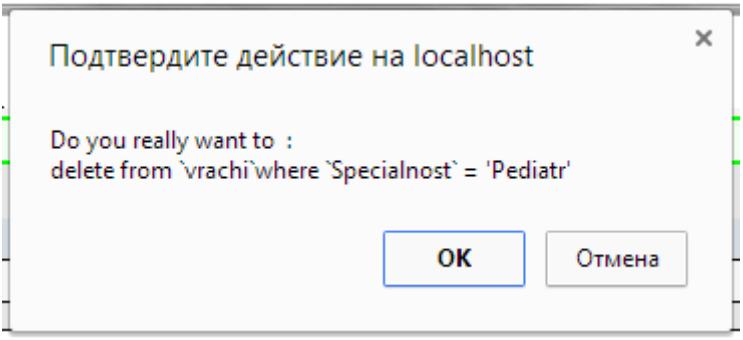
СУБД сообщит нам что 1 запись добавлена:



Теперь необходимо удалить всех педиатров:

DELETE FROM `vrachi` WHERE `Specialnost` = 'Pediatr'

СУБД выводит запрос о подтверждении удалении записей:



Информация была удалена из таблицы:

ID_Vrach	FIO	Specialnost	Kategoriya
2	Porashenko I.V.	Hirurg	10
3	Leonova E.E.	Ortoped	5
4	Iosif K.I.	Okulist	3
5	Mishin M.M.	Laborant	6
7	Umihina T.T.	Gastrointerolog	2
8	Tomina T.N.	Psiholog	1
9	Inova L.U.	Laborant	9
11	Petrov A.S.	terapevt	7

selected: Change Delete Export

5. Требование к отчету

Структура отчёта зависит от типа задания и индивидуального стиля работы студента. Однако, все работы должны обладать некоторыми общими чертами и включать следующие основные элементы:

- Титульный лист.
- Лист рецензии.

- Содержание.
- Цель лабораторной работы.
- Ответы на контрольные вопросы.
- Введение.
- Теоретическая (практическая) часть. Инфологическая модель базы данных согласно варианту.
- Результаты выполнения запросов (код и скриншоты).
- Заключение (выводы по работе).
- Список источников информации.
- Приложения.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЁТА:

- Текстовый материал должен быть подготовлен с использованием MS Word.
- Текст должен быть набран с учетом следующих параметров:
  10. шрифт Times New Roman;
  11. размер шрифта заголовков – 14, основного текста - 12;
  12. межстрочный интервал - 1;
  13. межбуквенный интервал - Normal;
  14. поля: левое – 2.5, правое – 2, верхнее и нижнее – 1.5;
  15. выравнивание – по ширине страницы;
  16. нумерация страниц – в нижем правом углу;
  17. в верхнем колонтитуле указать название работы, Ф.И.О. автора;
  18. образец оформления титульного листа приведен в приложении №1
- Объем работы не менее 10 страниц.
- Использовать электронное оглавление.
- Рисунки должны быть пронумерованы и подписаны.
- Список источников должен содержать как список литературы, так и список ресурсов Интернет.

6. Контрольные вопросы

6.1 Какие условия должны выполняться при использовании команды DELETE? UPDATE?

6. 2 Можно ли объединять несколько команд модификаций в один запрос?

6.3 Можно ли модифицировать одновременно несколько таблиц?

6.4 Опишите процедуру использования вложенных запросов с командами модификаций.

7. Рекомендуемая литература

7.1 Диго С. М. Базы данных: проектирование и использование [Text] : учебник для вузов / Диго С. М. - М. : Финансы и статистика, 2005. - 592 с. : ил. - 20 экз. - ISBN 5-279-02571-2 (18)

7.2 Андон Ф. Язык запросов SQL. Учебный курс [Text] / Андон Ф. - СПб. : Питер, 2006. - 416 с. : ил. - 30 экз. - ISBN 5-469-00394-9 : Б. ц. УДК 004.6004.65(075) (30)

7.3 Рудикова, Л. В. Базы данных. Разработка приложений для студента [Text] / Рудикова, Л. В. - СПб. : БХВ-Петербург, 2006. - 496 с. : ил. - 30 экз. - ISBN 5-94157-805-9 (30)

7.4 Карпова, И. П. Базы данных. Курс лекций и материалы для практических занятий [Текст] : учеб. пособие для вузов / И. П. Карпова . - СПб. : Питер, 2013. - 240 с.

7.5 Малыхина, М. П. Базы данных: основы, проектирование, использование [Text] : учеб. пособие для вузов / Малыхина, М. П. - СПб. : БХВ-Петербург, 2004. - 499 с. : ил. - 99 экз. - (Учебное пособие). - Библиогр.: с. 491-493 (28 назв.). - ISBN 5-94157-310-4 (98)

7.6 Мотев, А. А. Уроки MySQL [Text] : самоучитель / Мотев, А. А. - СПб. : БХВ-Петербург, 2006. - 208 с. : ил. - 20 экз. - (Самоучитель). - ISBN 5-94157-658-7(20)

Лабораторная работа № 14

Создание представлений для работы с базой данных

1. Цель работы

Изучить представление реляционных таблиц в Web-интерфейсе phpMyAdmin

2. Теоретическая часть

Представления (view) - это одно из мощных средств языка SQL, предназначенное для реализации механизм подсхем пользователей базы данных. Представления позволяют скрыть от пользователей схему базы данных. Они представляют собой хранимые в базе данных запросы, выраженные операторами SELECT. На базе одних представлений могут быть созданы новые представления, которые наследуют все свойства базовых представлений. Формировать представления могут пользователи с привилегиями SELECT для используемых в представлениях таблиц (базовых таблиц).

Для пользователя представления предстают как объекты очень похожие на таблицы данных. Это выражается тем, что:

обращение к представлениям осуществляется также как и к таблицам;

ко всем представлениям применим оператор SELECT;

для некоторых представлений могут применяться операторы INSERT, UPDATE и DELETE.

Однако в соответствие со стандартом ANSI SQL/89 в SYBASE SQL Anywhere таблицы данных и представления имеют некоторые различия:

запрос, именованный через представление выполняется только в момент обращения к представлению;

для представления невозможно определить ограничения целостности и первичный ключ;

в операторе SELECT, на базе которого создается представление, нельзя устанавливать сортировку его результатов;

не ко всем представлениям могут применяться операторы INSERT, UPDATE и DELETE.

Представление может быть модифицировано (т.е. по отношении к нему можно использовать операторы INSERT, UPDATE и DELETE) в том, и только в том случае, если для оператора SELECT, на базе которого создано представление, выполняются каждое из следующих специфических условий:

не используется служебное слово DISTINCT;

при выполнении запроса данные извлекаются только из таблицы;

в списка полей этого оператора отсутствуют арифметические выражения. Элементами списка могут быть только поля базовой таблицы или базового представления. В свою очередь на поля этого представления накладывается такое же ограничение;

в запросе не применяются подзапросы;

для результирующих данных не определено группирование.

Для создания представления используется оператор CREATE VIEW, имеющий следующий синтаксис:

```
CREATE [OR REPLACE]
[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
VIEW view_name [(column_list)]
AS select_statement
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

view_name — имя создаваемого представления. select_statement — оператор SELECT, выбирающий данные из таблиц и/или других представлений, которые будут содержаться в представлении

Оператор CREATE VIEW содержит 4 необязательные конструкции:

OR REPLACE — при использовании данной конструкции в случае существования представления с таким именем старое будет удалено, а новое создано. В противном случае возникнет ошибка, информирующая о существовании представления с таким именем и новое представление создано не будет. Следует отметить одну особенность — имена таблиц и представлений в рамках одной базы данных должны быть уникальны, т.е. нельзя создать представление с именем уже существующей таблицы. Однако конструкция OR REPLACE действует только на представления и замещать таблицу не будет.

ALGORITM — определяет алгоритм, используемый при обращении к представлению (подробнее речь об этом пойдет ниже).

column_list — задает имена полей представления.

WITH CHECK OPTION — при использовании данной конструкции все добавляемые или изменяемые строки будут проверяться на соответствие определению представления. В случае несоответствия данное изменение не будет выполнено. Обратите внимание, что при указании данной конструкции для необновляемого представления возникнет ошибка и представление не будет создано.

3. Подготовка к работе

3.1 Изучить методические указания и рекомендованную литературу.

3.2 Подготовить ответы на контрольные вопросы.

4. Задание на выполнение работы

4.1 Изучить назначение и возможности применения представлений.

4.2 Создать представление.

4.3 Внести изменения в представление.

4.4 Подготовить отчет.

Варианты заданий

Все лабораторные работы выполняются по одной теме. Вариант задания (тема) выбирается один раз, в лабораторной работе № 3. Тема выбирается согласно номеру студента в списке группы.

Пример выполнения лабораторной работы

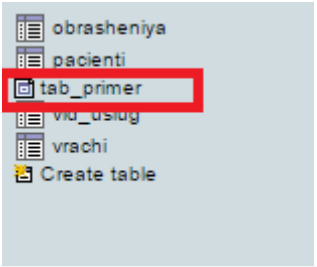
Есть таблица врачей, в которой 10 записей:

ID_Vrach	FIO	Specialnost	Kategoriya
1	Ivanovna M.P.	Pediatr	2
2	Porashenko I.V.	Hirurg	10
3	Leonova E.E.	Ortoped	5
4	Iosif K.I.	Okulist	3
5	Mishin M.M.	Laborant	6
6	Ivuhin B.I.	Pediatr	1
7	Umihina T.T.	Gastrointerolog	2
8	Tomina T.N.	Psiholog	1
9	Inova L.U.	Laborant	9
10	Muhina I.A.	Pediatr	2

У каждого врача своя категория, но у педиатров и психологов она неизменна. Что бы пользователям было удобнее наблюдать за изменениями категорий у других врачей создадим представление, где категория будет выше 2:

CREATE VIEW Tab_primer as select * from vrachi where `Kategoriya` > 2

В левом углу, где отображаются таблицы базы данных, появится новая, с другим значком. Это и есть наше представление:

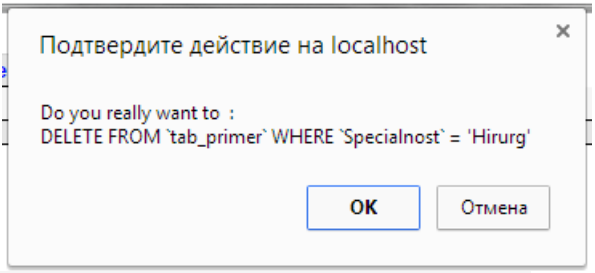


ID_Vrach	FIO	Specialnost	Kategoriya
2	Porashenko I.V.	Hirurg	10
3	Leonova E.E.	Ortoped	5
4	Iosif K.I.	Okulist	3
5	Mishin M.M.	Laborant	6
9	Inova L.U.	Laborant	9

Из больницы перевели отделение хирургии, поэтому хирургов можно удалить из БД:

DELETE FROM `tab_primer` WHERE `Specialnost` = 'Hirurg'

СУБД спросит об удалении, после чего запись будет стерта из БД:



ID_Vrach	FIO	Specialnost	Kategoriya
3	Leonova E.E.	Ortoped	5
4	Iosif K.I.	Okulist	3
5	Mishin M.M.	Laborant	6
9	Inova L.U.	Laborant	9

Внимание! Записи удаляются и модифицируются не в самом представлении, а в таблице – оригинале!

5. Требование к отчету

Структура отчёта зависит от типа задания и индивидуального стиля работы студента. Однако, все работы должны обладать некоторыми общими чертами и включать следующие основные элементы:

- Титульный лист.
- Лист рецензии.
- Содержание.
- Цель лабораторной работы.
- Ответы на контрольные вопросы.
- Введение.
- Теоретическая (практическая) часть. Материал согласно заданию.
- Заключение (выводы по работе).
- Список источников информации.
- Приложения.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЁТА:

- Текстовый материал должен быть подготовлен с использованием MS Word.
- Текст должен быть набран с учетом следующих параметров:
  1. шрифт Times New Roman;
  2. размер шрифта заголовков – 14, основного текста - 12;
  3. межстрочный интервал - 1;
  4. межбуквенный интервал - Normal;
  5. поля: левое – 2.5, правое – 2, верхнее и нижнее – 1.5;
  6. выравнивание – по ширине страницы;
  7. нумерация страниц – в нижем правом углу;
  8. в верхнем колонтитуле указать название работы, Ф.И.О. автора;
  9. образец оформления титульного листа приведен в приложении №1
- Использовать электронное оглавление.
- Рисунки должны быть пронумерованы и подписаны.
- Список источников должен содержать как список литературы, так и список ресурсов Интернет.

6. Контрольные вопросы

6.1 Что такое представление?

6.2 Зачем используется представление?

6.3 Можно ли создать представление от представления?

6.4 Когда можно модифицировать представление?

7. Рекомендуемая литература

7.1 Диго С. М. Базы данных: проектирование и использование [Text] : учебник для вузов / Диго С. М. - М. : Финансы и статистика, 2005. - 592 с. : ил. - 20 экз. - ISBN 5-279-02571-2 (18)

7.2 Андон Ф. Язык запросов SQL. Учебный курс [Text] / Андон Ф. - СПб. : Питер, 2006. - 416 с. : ил. - 30 экз. - ISBN 5-469-00394-9 : Б. ц. УДК 004.6004.65(075) (30)

7.3 Рудикова, Л. В. Базы данных. Разработка приложений для студента [Text] / Рудикова, Л. В. - СПб. : БХВ-Петербург, 2006. - 496 с. : ил. - 30 экз. - ISBN 5-94157-805-9 (30)

7.4 Карпова, И. П. Базы данных. Курс лекций и материалы для практических занятий [Текст] : учеб. пособие для вузов / И. П. Карпова . - СПб. : Питер, 2013. - 240 с.

7.5 Малыхина, М. П. Базы данных: основы, проектирование, использование [Text] : учеб. пособие для вузов / Малыхина, М. П. - СПб. : БХВ-Петербург, 2004. - 499 с. : ил. - 99 экз. - (Учебное пособие). - Библиогр.: с. 491-493 (28 назв.). - ISBN 5-94157-310-4 (98)

7.6 Мотев, А. А. Уроки MySQL [Text] : самоучитель / Мотев, А. А. - СПб. : БХВ-Петербург, 2006. - 208 с. : ил. - 20 экз. - (Самоучитель). - ISBN 5-94157-658-7(20)