Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

УТВЕРЖДАЮ
Директор
_____ / С.А. Махновский
«09» февраля 2022 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ РАБОТ

по учебной дисциплине
ОПЦ.08 Основы проектирования баз данных

для обучающихся специальности

09.02.07 Информационные системы и программирование
Квалификация: Разработчик веб и мультимедийных приложений

Магнитогорск, 2022

ОДОБРЕНО

Предметно-цикловой комиссией «Информатики и вычислительной техники»
Председатель И.Г. Зорина
Протокол № 5 от 19.01.2022г.

Методической комиссией МпК
Протокол № 4 от 09.02.2022г

Разработчик:
преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» Многопрофильный колледж
Т.Б. Осолодкова

Методические указания по выполнению практических и лабораторных работ разработаны на основе рабочей программы учебной дисциплины «Основы проектирования баз данных».

Содержание практических и лабораторных работ ориентировано на подготовку обучающихся к освоению профессиональных модулей программы подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование и овладению профессиональными компетенциями.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки обучающихся составляют практические и лабораторные занятия.

Состав и содержание практических и лабораторных занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью практических занятий является формирование учебных практических умений (использовать современные case–средства для проектирования баз данных; создавать объекты баз данных в современных СУБД; применять стандартные методы для защиты объектов базы данных), необходимых в последующей учебной деятельности по профессиональному модулю ПМ.11 Разработка, администрирование и защита баз данных.

Ведущей дидактической целью лабораторных занятий является экспериментальное подтверждение и проверка существенных теоретических положений (законов, зависимостей).

В соответствии с рабочей программой учебной дисциплины «Основы проектирования баз данных» предусмотрено проведение практических и лабораторных занятий. В рамках практического занятия обучающиеся могут выполнять одну или несколько практических работ.

В результате их выполнения, обучающийся должен:

уметь:

- проектировать реляционную базу данных
- использовать язык запросов для программного извлечения сведений из баз данных,
- анализировать задачу и/или проблему и выделять её составные части;
- определять этапы решения задачи;
- выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- понимать тексты на базовые профессиональные темы;
- читать, понимать и находить необходимые технические данные и инструкции в руководствах в любом доступном формате.

Содержание практических и лабораторных занятий ориентировано на подготовку обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности и овладению ***профессиональными компетенциями:***

ПК 5.1. Собирать исходные данные для разработки проектной документации на информационную систему

ПК 5.4. Производить разработку модулей информационной системы в соответствии с техническим заданием

А также формированию ***общих компетенций***:

ОК 01 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02 Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности.

ОК 04 Эффективно взаимодействовать и работать в коллективе и команде.

ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках.

Выполнение обучающихся практических и лабораторных работ по учебной дисциплине «Основы проектирования баз данных» направлено на:

- *обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам учебной дисциплины;*
- *формирование умений применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;*

- *выработку при решении поставленных задач профессионально значимых качеств, таких как самостоятельность, ответственность, точность, творческая инициатива.*

Практические и лабораторные занятия проводятся в рамках соответствующей темы, после освоения дидактических единиц, которые обеспечивают наличие знаний, необходимых для ее выполнения.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тема 1.2 Взаимосвязи в моделях и реляционный подход к построению моделей
Практическое занятие №1 Проектирование реляционной базы данных.

Цель: получение практических навыков проектирования базы данных
Выполнив работу, Вы будете: ***уметь:***
- У1. Проектировать реляционную базу данных
- У01.2 анализировать задачу и/или проблему и выделять её составные части;
- У01.3 определять этапы решения задачи;
- У 01.4 Выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы

Материальное обеспечение:
Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение.
Задание:
В соответствии со своим вариантом проанализируйте предметную область решаемой задачи и разработайте логическую структуру соответствующей базы данных.
Порядок выполнения работы:
1. Выполнить анализ предметной области. Выполнить анализ данных.
2. Определить набор атрибутов для данной предметной области. 4. Определить набор таблиц.
3. При проектировании таблиц рекомендуется руководствоваться следующими основными принципами:
- каждая таблица должна содержать данные только на одну тему;
- данные не должны дублироваться.
4. Создать словарь имен.
5. Определить состав и типы полей.
6. Создать связи между таблицами.

Ход работы.
Краткие теоретические сведения:
Проектирование базы данных заключается в ее многоступенчатом описании с различной степенью детализации и формализации, в ходе которого производится уточнение и оптимизация структуры базы данных. Проектирование начинается с описания предметной области и задач информационной системы, идет к более абстрактному уровню логического описания данных и далее – к схеме физической (внутренней) модели базы данных. Трем основным уровням моделирования системы – **концептуальному, логическому и физическому** соответствуют три последовательных этапа детализации описания объектов базы данных и их взаимосвязей.

На **концептуальном уровне** проектирования производиться смысловое описание информации предметной области, определяются ее границы, производиться абстрагирование от несущественных деталей. В результате определяются моделируемые объекты, и их свойства, и связи. Выполняется структуризация знаний о предметной области, стандартизируется терминология. Затем строится концептуальная модель, описываемая на естественном языке. Для описания свойств и связей объектов применяют различные диаграммы.

На следующем шаге принимается решение о том, в какой конкретно СУБД будет реализована база данных. **Выбор СУБД** является сложной задачей и должен основывается на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид программного продукта и категория пользователей (или профессиональные программисты, или конечные пользователи, или то и другое).

Другими показателями, влияющими на выбор СУБД, являются:
- Удобство и простота использования;
- Качество средств разработки, защиты и контроля базы данных;

- Уровень коммуникационных средств (в случае применения ее в сетях);
- Фирма-разработчик;
- Стоимость.

Каждая конкретная СУБД работает с определенной моделью данных. Под моделью данных понимается способ их взаимосвязи: в виде иерархического дерева, сложной сетевой структуры или связанных таблиц. В настоящее время большинство СУБД использует табличную модель данных, называемую *реляционной*.

На **логическом уровне** производиться отображение данных концептуальной модели в логическую модель в рамках той структуры данных, которая поддерживается выбранной СУБД. Логическая модель не зависит от конкретной СУБД и может быть реализована на любой СУБД реляционного типа.

На **физическом уровне** производиться выбор рациональной структуры хранения данных и методов доступа к ним, которые обеспечивает выбранная СУБД. На этом уровне решаются вопросы эффективного выполнения запросов к БД, для чего строятся дополнительные структуры, например, индексы. В физической модели содержится информация обо всех объектах базы данных (таблицах, индексах, процедурах и др.) и используемых типах данных. Физическая модель *зависит* от конкретной СУБД. Одной и той же логической модели может соответствовать несколько разных физических моделей. Физическое проектирование является начальным этапом реализации базы данных.

Варианты заданий:

1. *БД «Зарплата».* Таблица «Должности»: код должности; должность. Таблица «Тарифы»: код должности; разряд; ставка, р./ч. Таблица «Работники»: отдел; код должности; разряд; табельный номер; Ф.И.О. Таблица «Табель»: год; месяц; табельный номер; количество отработанных часов; дата начисления зарплаты
2. *БД «Оптовая база».* Таблица «Товары»: код товара; товар; единица измерения; цена. Таблица «Склад»: дата поступления, код товара; количество. Таблица «Заявки»: номер заявки; организация; код товара; требуемое количество товара. Таблица «Отпуск товаров»: номер заявки; код товара; дата отпуска товара; количество отпущенного товара
3. *БД «Библиотека».* Таблица «Книги»: жанр; шифр книги; автор; название; год издания; количество экземпляров. Таблица «Читатели»: номер читательского билета; Ф.И.О.; адрес. Таблица «Выдачи»: дата выдачи; номер читательского билета; шифр книги; количество экземпляров; срок возврата; фактическая дата возврата
4. *БД «ГИБДД».* Таблица «Автомобили»: модель автомобиля; номер двигателя; номер кузова; серия и номер технического паспорта; государственный номер автомобиля. Таблица «Владельцы»: государственный номер автомобиля; Ф.И.О.; адрес, серия и номер водительского удостоверения. Таблица «Виды нарушений»: код нарушения; вид нарушения. Таблица «Нарушители»: дата нарушения; код нарушения; государственный номер автомобиля; размер штрафа, р.
5. *БД «Соревнования».* Таблица «Команда»: спортивная организация; шифр команды, название команды. Таблица «Участники»: шифр команды; номер участника; Ф.И.О. Таблица «Старт»: стартовый номер; номер участника; время старта; отметка о невыходе на старт. Таблица «Финиш»: порядковый номер финиширования; стартовый номер; время финиша; отметка о сходе с дистанции
6. *БД «Перевозки».* Таблица «Транспорт»: модель автомобиля; государственный номер автомобиля; удельный расход топлива, л/100 км. Таблица «Заявки»: номер заявки;
7. дата; пункт отправления; пункт назначения; груз; единица измерения; количество груза. Таблица «Доставка»: номер заявки; государственный номер автомобиля; дата отправления; дата возвращения; пройденное расстояние; расход топлива
8. *БД «Сессия».* Таблица «Кафедры и дисциплины»: номер кафедры; кафедра; шифр дисциплины; наименование дисциплины; вид аттестации (зачет или экзамен). Таблица «Преподаватели»: номер кафедры; табельный номер преподавателя; Ф.И.О. Таблица «Студенты»:

шифр студента; Ф.И.О. Таблица «Оценки»: дата; шифр дисциплины; табельный номер преподавателя; шифр студента; оценка

9. *БД «Телефонная компания».* Таблица «Тарифы»: код тарифа; вид тарифа; цена, р./мин. Таблица «Виды льгот»: код льготы; вид льготы; размер, %. Таблица «Абоненты»: лицевой счет; телефон; Ф.И.О.; адрес; код льготы. Таблица «Платежи»: лицевой счет; код тарифа; дата оплаты; сумма платежа; дата отключения за неуплату

Форма представления результата: Оформленная схема базы данных в тетради. **Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Проектирование структур баз данных

Практическое занятие №2. Нормализация реляционной базы данных.

Цель: получение практических навыков по нормализации базы данных

Выполнив работу, Вы будете: *уметь:*

У1. Проектировать реляционную базу данных

Уо 04.01 организовывать работу коллектива и команды,

Уо 04.03 эффективно работать в команде,

Уо 09.01 - понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы,

Уо 09.02 - участвовать в диалогах на знакомые общие и профессиональные темы.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение.

Задание:

В соответствии со своим вариантом задания, выданным на предыдущем занятии привести базу данных к 3НФ. Реализовать схему базы данных в среде MySQL Workbench.

Порядок выполнения работы:

Используя метод нормальных форм спроектировать базу данных. Процесс проектирования должен быть представлен в форме отчета, показан процесс формирования таблиц под выделенные сущности и их последовательная нормализация методом нормальных форм.

Ход работы.

Краткие теоретические сведения:

Нормализация баз данных

Нормальные формы – это рекомендации по проектированию баз данных. Рекомендуется нормализовать базу данных в некоторой степени потому, что этот процесс имеет ряд существенных преимуществ с точки зрения эффективности и удобства обращения с базой данных.

В нормализованной структуре базы данных можно производить сложные выборки данных относительно простыми SQL-запросами.

Целостность данных. Нормализованная база данных позволяет надежно хранить данные.

Нормализация предотвращает появление избыточности хранимых данных. Данные всегда хранятся только в одном месте, что делает легким процесс вставки, обновления и удаления данных. Есть исключение из этого правила. Ключи, сами по себе, хранятся в нескольких местах

потому, что они копируются как внешние ключи в другие таблицы.

Масштабируемость – это возможность системы справляться с будущим ростом. Для базы данных это значит, что она должна быть способна работать быстро, когда число пользователей и объемы данных возрастают. Масштабируемость – это очень важная характеристика любой модели базы данных и для РСУБД.

Нормализация баз данных заключается в приведении структуры хранения данных к нормальным формам (NF). Всего таких форм существует 8, но часто достаточным является соблюдение первых трех. Рассмотрим их более подробно на примере учебной базы данных. Примеры будут строится по принципу «что было бы, если было иначе, чем сейчас».

Первая нормальная форма

Основным правилом первой формы является необходимость неделимости значения в каждом поле (столбце) строки – **атомарность** значений.

Рассмотрим таблицы сотрудников и телефонных линий.

Сотрудники_Линии

	Имя столбца	Сжатый тип	Допускает значения NULL
🔑	Сотрудник	int	Нет
🔑	Линия	int	Нет
	Описание	varchar(255)	Да
	Приоритет	int	Нет

Чтобы избавиться от связывающей таблицы «Сотрудники_Линии», мы могли бы записать идентификаторы сотрудников для каждой линии в виде перечня в дополнительном столбце:

	id	Описание	Приоритет	Сотрудники
1	1	Поддержка держателей пластиковых карт	2	20, 21, 8
2	2	Поддержка потребительского и наличного кредитова...	2	16, 5, 22, 15
3	3	Поддержка автокредитования	2	21, 10
4	4	Поддержка ипотечного кредитования	2	16, 14, 15, 4, 8

Но подобная структура не является надежной. Представьте, что Вам необходимо поменять некоторым сотрудникам подключенные линии. Потребуется осуществить разбор составного поля, чтобы определить наличие id сотрудника в каждой записи линий, затем скорректировать перечень. Получается слишком сложный и долгий процесс для такой простой операции.

Организации структуры таблиц с применением дополнительной связывающей избавляет от подобных проблем.

Помимо атомарности к первой нормальной форме относятся следующие правила:

Строки таблиц не должны зависеть друг от друга, т.е. первая запись не должна влиять на вторую и наоборот, вторая на третью и т.д. Размещение записей в таблице не имеет никакого значения.

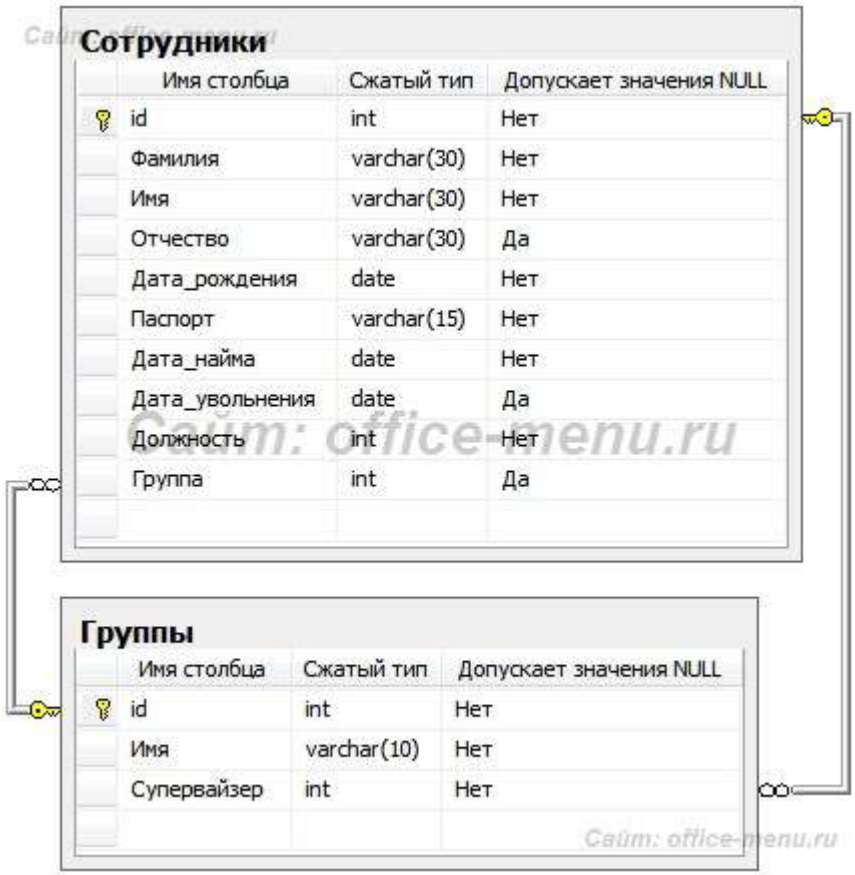
Аналогичная ситуация со столбцами записей. Их порядок не должен влиять на понимание информации.

Каждая строка должна быть уникальна, поэтому для нее определяется **первичный ключ**, состоящий из одного либо нескольких полей (**составной ключ**). Первичный ключ не может повторяться в пределах таблицы и служит идентификатором записи.

Вторая нормальная форма

Условием этой формы является отсутствие зависимости неключевых полей от части составного ключа.

Так как составной ключ в учебной базе наблюдается только в таблице «Сотрудники_Линии», то рассмотрим пример на ней.



На представленной диаграмме столбцы описания и приоритета зависят от столбца «Линия», входящего в составной ключ. Это значит, что для каждой линии, подключенной разным сотрудникам, потребуется повторно указывать описание и приоритетность. Подобная структура приводит к **избыточности данных.**

Также велика вероятность возникновения противоречивой информации. Изменяя приоритет или описание для линии, можно по ошибке оставить некоторые строки не обработанными. В таком случае, для одного и того же идентификатора линии значения зависимых полей будут различными.

Если соблюдены правила первой нормальной формы, то создание таблицы «Линии» и перенос в нее зависимых столбцов удовлетворяет второй нормальной форме.

Третья нормальная форма

3NF схожа по логике с 2NF, но с некоторым отличием. Если 2 форма ликвидирует зависимости неключевых полей от части ключа, то третья нормальная форма исключает зависимость неключевых полей от других неключевых полей.

На приведенном примере таблицы сотрудников видно, что столбец «Супервайзер» имеет зависимость от столбца «Группа», а это значит, что при изменении значения поля группы, потребуется изменить значение поля супервайзера.

Все риски, которые были рассмотрены для 2NF, так же относятся к 3NF и устраняются переносом зависимых полей в отдельную таблицу.

Денормализация базы данных

Теория нормальных форм не всегда применима на практике. Например, неатомарные значения не всегда являются «злом», а иногда наоборот. Связано это с необходимостью дополнительного объединения (следовательно, затрат производительности системы) при выполнении запросов, особенно когда производится обработка большого массива информации.

Форма представления результата:

представление отчета на образовательном портале (в соответствующем курсе). **Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Проектирование структур баз данных

Лабораторное занятие№1.

Преобразование реляционной базы данных в сущности и связи.

Цель: получение практических навыков по освоению операций настройки схемы базы данных.

Выполнив работу, Вы будете: *уметь:*

У1. Проектировать реляционную базу данных

Уо 04.01 организовывать работу коллектива и команды,

Уо 04.03 эффективно работать в команде,

Уо 09.01 - понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы,

Уо 09.02 - участвовать в диалогах на знакомые общие и профессиональные темы.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MySQL Workbench.

Задание:

Создать схему базы данных по заданному варианту.

Порядок выполнения работы:

1. Разработать логическую модель схемы по своему варианту. 2. Посмотреть и проанализировать физическую модель.
2. Сгенерировать скрипт создания таблиц.
3. Используя сгенерированный скрипт создать базу данных в выбранной СУБД.
4. Настроить схему базы данных.

Краткие теоретические сведения:

Этапы проектирования базы данных:

1. инфологическое проектирование;
2. определение требований к операционной обстановке, в которой будет функционировать информационная система;
3. выбор системы управления базой данных (СУБД) и других инструментальных программных средств;
4. логическое проектирование БД;

5. физическое проектирование БД.

Инфологическое проектирование.

Инфологическое проектирование – построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создается без ориентации на какую-либо конкретную СУБД и модель данных. Конкретный вид и содержание концептуальной модели базы данных определяется выбранными для этого формальным аппаратом. Обычно используют графические нотации, подобные ERдиаграммам. Чаще всего инфологическая (концептуальная) модель базы данных включает в себя: описание информационных объектов или понятий предметной области и связей между ними; описание ограничений целостности, то есть требований к допустимым значениям данных и к связям между ними. Пример инфологического проектирования показан на рисунке 1.

Entity-Relationship Diagrams

Имеется целый ряд методик создания информационно-логических моделей. Одна из наиболее популярных в настоящее время методик при разработке моделей использует ERD (Entity-Relationship Diagrams). В русскоязычной литературе эти диаграммы называют «объект – отношение» либо «сущность – связь». Модель ERD была предложена Питером Пин Шен Ченом в 1976 г. К настоящему времени разработано несколько ее разновидностей, но все они базируются на графических диаграммах, предложенных Ченом. Диаграммы конструируются из небольшого числа компонентов. Благодаря наглядности представления они широко используются в CASE-средствах (Computer Aided Software Engineering). Рассмотрим используемую терминологию и обозначения.

Сущность (Entity) – реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению.

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа (сущности).

Каждая сущность должна обладать некоторыми свойствами:

- иметь уникальное имя; причем к этому имени должна всегда применяться одна и та же интерпретация (определение сущности). И наоборот: одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются ею через связь;
- обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности.

Связь (Relationship) – поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Одна из участвующих в связи сущностей – независимая, называется родительской сущностью, другая – зависимая, называется дочерней или сущностью-потомком. Как правило, каждый экземпляр родительской сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров дочерней сущности. Каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущностипотомка может существовать только при существовании сущности-родителя. Связи дается имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи.

При построении ER-модели используются следующие принципы:

- сущности на диаграмме представляются прямоугольниками;
- каждый прямоугольник может иметь различные визуальные атрибуты; каждой сущности должно быть присвоено уникальное имя;
- имена сущностей необходимо задавать в единственном числе;
- связи на диаграмме представляются линиями, идущими от одной сущности (таблицы) к другой;
- каждой связи присваивается уникальное имя;
- связанные таблицы разделяют на родительские и дочерние;
- родительские таблицы отображаются прямоугольниками с прямыми углами, дочерние – со скругленными.

Логическое проектирование

Логическое проектирование – создание схемы базы данных на основе конкретной модели базы данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель – набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Основные понятия языка SQL

Лабораторное занятие№2

Создание базы данных. Создание и редактирование таблиц.

Цели:

- получение практических навыков по освоению операций создания таблиц;
- получение практических навыков по освоению операций подстановок.

Выполнив работу, Вы будете: *уметь:*

У2 Использовать язык запросов для программного извлечения сведений из баз данных,

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS Access, MySQL Workbench.

Задание 1:

Создать базу данных Туризм и таблицы базы данных в СУБД MS Access.

Порядок выполнения работы:

1. Создайте в своей рабочей папке папку с именем *База данных*.

2. Запустите ***MS Access***. Создайте в созданной папке новую базу данных с именем ***Туризм***.

Ход работы.

Краткие теоретические сведения:

Связи ***автоматические*** устанавливаются **Мастером подстановок.** Посмотреть, установить, отредактировать связи можно командой ***Работа с базой данных – Схема данных.***

Если связи устанавливаются первично, то откроется окно ***Таблицы,*** а если повторно, то окно ***Связи.*** Двойной щелчок на нужной таблице позволит перенести их в окно ***Связи.***

Для установки связи между таблицами ***вручную*** нужно перетянуть связываемое поле из главной таблицы и наложить его на соответствующее поле подчиненной таблицы.

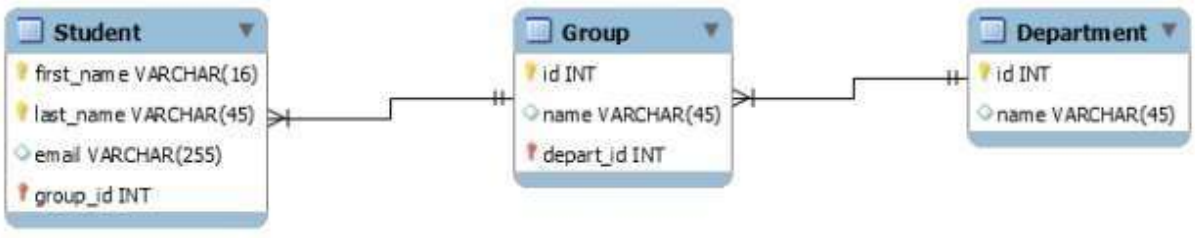


Рисунок 2 – Логическая модель

Удаление и изменение связей производится с помощью контекстного меню на линии связи, а также клавишей DEL.

В окне ***Схема данных*** двойной щелчок по линии связи позволит открыть окно **Связи.** В нем можно установить флажок у опции ***Целостность данных***, линия связи станет гораздо темнее и появятся значки «1» и «∞», означающие отношение «один» или «многие».

Если система определила тип связи (в нижней части диалогового окна) «один-к-одному» или «один-ко-многим», то можно поставить флажок «Поддерживать целостность данных».

Целостность данных – это набор правил, защищающих данные от случайных изменений или удалений с помощью механизма поддержки корректности связей между связанными таблицами.

Если связь определена и система взяла на себя поддержку целостности данных, то при просмотре главной таблицы (отношение «один») слева, рядом с полосой выделения появится колонка со знаками «+». Щелчок на «+» позволит открыть подчиненную таблицу (отношение «много» или «один»).

Создание таблиц с помощью ***Конструктора***

1. Создайте таблицу **Сотрудники** в режиме Конструктора. Наименования и типы полей представлены в приведенной таблице.

Название поля	Тип данных
Код клиента	Числовой
Название клиента	Текст
Контактное лицо	Текст
Признак группы	Логический
Телефон	Текст
Адрес	Текст

Название поля **Тип данных**

Код сотрудника	Числовой
ФИО	Текст
Должность	Текст
Дата найма	Дата/Время
Дата рождения	Дата/Время
Домашний телефон	Текст
Адрес	Текст
Размер оклада	Числовой

2. Для поля ***Домашний телефон*** задайте маску, набрав, например, следующий шаблон (999) 999-99-99.

3. Для поля ***Оклад*** задайте условие, что он больше 5000 р., но не больше 10000. Для этого в свойстве «Условие на значение» установите (>5000) AND (<10000). Предусмотрите выдачу сообщения при ошибке ввода данных.

4. Установите для ***Даты рождения*** и ***Даты найма*** маску ввода. Используйте краткий формат даты.

5. Создайте первичный ключ.

6. Перейдите в режим просмотра таблицы.

7. Спрячьте некоторые столбцы. Сделайте их опять видимыми командами **Контекстное меню — Скрыть/Показать столбцы.**

8. Зафиксируйте столбцы, содержащие фамилию и имя. Освободите столбцы. 11. Поменяйте тип шрифта и его начертание (**Формат — Шрифт**).

9. Закройте окно таблицы ***Сотрудники***, сохранив изменения.

10. Создайте новые таблицы **Клиенты** и **Страны**. В качестве первичного ключа задайте ***Код Клиента*** и ***КодТура***.

Использование ***Мастера подстановок***

Создайте в режиме **Конструктора** таблицу ***Договоры***, которая должна иметь следующие поля:

Поля *Код сотрудника, Код клиента, Код тура* являются полями подстановки. Для их задания используется Мастер подстановок:

- в Типе данных поля *Код сотрудника* раскрыть список типов и выбрать **Мастер подстановок;**
- указать, что столбец подстановки получает свои значения из таблицы ***Сотрудники***;

выбрать поля *Код сотрудника* и *Фамилия*;

- установить мышью подходящую ширину столбца;
- согласиться с предлагаемой подписью столбца подстановок *Фамилия*;
- сохранить таблицу с именем ***Договоры***.

Аналогично для подстановки *Кода клиента* и *Кода тура* вызывается **Мастер подстановок**. При этом для *Кода клиента* выбираем поля *Код клиента* и *Название клиента* из таблицы ***Клиенты***, а для *Кода тура* — поля *Код тура* и *Страна* из таблицы ***Страны***.

Просмотрите и проанализируйте уже установленные при работе с **Мастером подстановки** связи в окне ***Схема данных***.

В окне ***Схема данных*** двойным щелчком по линии связи откройте окно Связи и установите *Целостность данных*, *Каскадное обновление данных*, *Каскадное удаление данных*.

Введите в таблицы 10 разнообразных записей. В таблице ***Сотрудники*** *осуществите* ввод заведомо некорректных данных для проверки работоспособности условия на значение.

Просмотрите главную таблицу каждой связи (с помощью «+») и вызовите *подчиненную* таблицу для каждой записи

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Основные понятия языка SQL.

Лабораторное занятие№3

Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами.

Цель: получение практических навыков по созданию ключевых полей и индексов, установлению связей между таблицами

Выполнив работу, Вы будете: *уметь:*

У2 Использовать язык запросов для программного извлечения сведений из баз данных,

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS Access.

Задание 1:

По своему варианту базы данных в СУБД Access для каждой таблицы создайте первичные и внешние ключи, установите связи и укажите индексированные поля.

Порядок выполнения работы:

1. Используя рассмотренный ниже пример, установить для своей базы данных первичные ключи.
2. Установить необходимы индексированные поля.
3. Установить связи между таблицами на схеме данных.

Краткие теоретические сведения:

Индексы обеспечивают поиск и сортировку записей в Access. В индексе хранится местоположение записей на основе одного или нескольких полей, которые были выбраны для индексирования. После того как Access получает сведения о позиции в индексе, он может извлечь

данные путем перемещения непосредственно к нужной позиции. Благодаря этому использование индекса гораздо эффективнее просмотра всех записей для поиска необходимых данных.

Выбор полей для индексирования

Вы можете создавать индексы, основанные на одном или нескольких полях. В основном требуется индексировать поля, в которых часто осуществляется поиск, сортируемые поля и поля, объединенные с полями в других таблицах, что часто используется в запросах по нескольким таблицам. Индексы ускоряют поиск и выполнение запросов, однако они могут привести к снижению производительности при добавлении или обновлении данных. Каждый раз, когда вы добавляете или изменяете запись в таблице, содержащей один или несколько индексов, Access приходится обновлять индексы. Добавление записей с помощью запроса на добавление или с помощью импортирования записей также, скорее всего, будет происходить медленнее, если таблица-получатель содержит индексы.

Примечание: Первичный ключ таблицы индексируется автоматически.

Индексировать поля с типом данных "Объект OLE", "Вычисляемый" или "Вложение" невозможно. Индексировать другие поля следует в тех случаях, когда выполняются все указанные ниже условия.

Тип данных поля: короткий текст (текст в Access 2010), длинный текст (MEMO в Access 2010), число, Дата и время, счетчик, денежная единица, да/нет или гиперссылка.

Предполагается поиск значений в поле.

Предполагается сортировка значений в поле.

Предполагается сохранение большого числа различных значений в поле. Если поле содержит много одинаковых значений, то применение индекса может не дать значительного ускорения выполнения запросов.

Составные индексы

Если предполагается, что необходимо будет часто выполнять поиск или сортировку по нескольким полям, вы можете создать индекс для этого сочетания полей. Например, если в одном запросе часто задаются условия для полей "Поставщик" и "Наименование_продукта", имеет смысл создать для этих полей составной индекс.

При сортировке таблицы по составному индексу Access сначала выполняет сортировку по первому полю, заданному для индекса. Последовательность полей определяется при создании составного индекса. Если в первом поле содержатся записи с повторяющимися значениями, затем выполняется сортировка по второму полю, заданному для индекса, и т. д.

В составной индекс можно включить до 10 полей. Создание индекса

Перед созданием индекса необходимо решить, следует ли создать индекс для одного поля или составной индекс. Индекс для одного поля создается с помощью установки свойства **Индексированное поле.**

При создании уникального индекса невозможно ввести новое значение в определенном поле, если такое значение уже существует в том же поле другой записи. Access автоматически создает уникальный индекс для первичных ключей, однако может потребоваться запретить создание повторяющихся значений и в других полях. Например, вы можете создать уникальный индекс для поля, в котором содержатся серийные номера, чтобы двум продуктам нельзя было присвоить один и тот же серийный номер.

Создание индекса для одного поля

1. В области навигации щелкните правой кнопкой мыши имя таблицы, в которой необходимо создать индекс, и выберите в контекстном меню пункт **Конструктор.**
2. Щелкните пункт **Имя поля** для поля, которое следует индексировать.
3. В разделе **Свойства поля** откройте вкладку **Общие.**
4. В свойстве **Индексированное** выберите значение **Да (допускаются совпадения),** если следует разрешить повторяющиеся значения, или значение **Да (совпадения не допускаются)**, чтобы создать уникальный индекс.
5. Чтобы сохранить изменения, щелкните элемент **Сохранить** на **панели быстрого доступа** или нажмите клавиши CTRL+S.

Создание составного индекса

Чтобы создать составной индекс для таблицы, добавьте строку для каждого поля в индексе и укажите имя индекса только в первой строке. Все строки будут обрабатываться как часть одного индекса, пока не будет обнаружена строка с другим именем индекса. Чтобы вставить строку, щелкните правой кнопкой мыши место, куда вы хотите ее вставить, и выберите в контекстном меню команду **Вставить строки.**

1. В области навигации щелкните правой кнопкой мыши имя таблицы, в которой необходимо создать индекс, и выберите в контекстном меню пункт **Конструктор.**
2. На вкладке **Конструктор** в группе **Показать или скрыть** щелкните пункт **Индексы.**

Появится окно "Индексы". Измените размеры этого окна, чтобы отображались пустые строки и свойства индекса.

3. В первой пустой строке столбца **Индекс** введите имя индекса. Для индекса можно использовать либо имя одного из индексируемых полей, либо другое подходящее имя.
4. В столбце **Имя поля** щелкните стрелку, затем щелкните первое поле, которое следует использовать в индексе.
5. Следующую строку столбца **Индекс** оставьте пустой, затем в столбце **Имя поля** укажите второе индексируемое поле. Повторите этот шаг для всех полей, которые необходимо включить в индекс.
6. Чтобы изменить порядок сортировки значений полей, в столбце **Порядок сортировки** окна "Индексы" щелкните пункт **По возрастанию** или **По убыванию**. По умолчанию выполняется сортировка по возрастанию.
7. В разделе **Свойства индекса** окна **Индексы** укажите свойства индекса для строки в столбце **Имя индекса**, содержащем имя индекса.

Чтобы сохранить изменения, нажмите кнопку **Сохранить** на **панели быстрого доступа** или нажмите клавиши CTRL+S.

9. Закройте окно "Индексы".

Удаление индекса

Если индекс становится ненужным или приводит к значительному снижению производительности, его можно удалить. При этом удаляется только сам индекс, а не поля, на которых он основан.

1. В области навигации щелкните правой кнопкой мыши имя таблицы, для которой необходимо удалить индекс, и выберите в контекстном меню пункт **Конструктор**.
2. На вкладке **Конструктор** в группе **Показать или скрыть** щелкните пункт **Индексы**.

Появится окно "Индексы". Измените размеры этого окна, чтобы отображались пустые строки и свойства индекса.

3. В окне "Индексы" выделите строки, содержащие индекс, который следует удалить, и нажмите клавишу DELETE.
4. Чтобы сохранить изменения, нажмите кнопку **Сохранить** на **панели быстрого доступа** или нажмите клавиши CTRL+S.
5. Закройте окно **Индексы**

Просмотр или редактирование индексов

Чтобы оценить влияние индексов на производительность или убедиться, что необходимые

поля проиндексированы, просмотрите индексы в таблице.

1. В области навигации щелкните правой кнопкой мыши имя таблицы, индекс которой вы хотите изменить, и выберите в контекстном меню пункт **Конструктор**.
2. На вкладке **Конструктор** в группе **Показать или скрыть** щелкните пункт **Индексы**.

Появится окно "Индексы". Измените размеры этого окна, чтобы отображались пустые строки и свойства индекса.

3. Просмотрите или измените индексы и свойства индексов в соответствии со своими задачами.
4. Чтобы сохранить изменения, нажмите кнопку **Сохранить** на **панели быстрого доступа** или нажмите клавиши CTRL+S.
5. Закройте окно **Индексы** Автоматическое создание индексов

В некоторых случаях индексы создаются автоматически. Например, индексы создаются для любых полей, которые определяются пользователем в качестве первичного ключа таблицы.

Для автоматического создания индекса также можно использовать параметр **Автоиндекс при импорте и создании** в диалоговом окне **Параметры Access**. Access автоматически проиндексирует все поля, имена которых начинаются с указанных в поле **Автоиндекс при импорте и создании** знаков или заканчиваются ими, например **ID**, **ключ**, **код** или **число**. Чтобы просмотреть или изменить текущие параметры, сделайте следующее:

1. Выберите **Файл** > **Параметры**.
2. Щелкните **Конструкторы объектов**, а затем в разделе **Конструктор таблиц** добавьте, измените или удалите значения в поле **Автоиндекс при импорте и создании**. Для разделения значений используйте точку с запятой (**;**).

Примечание: Если имя поля начинается со значения, указанного в списке, или заканчивается им, поле будет автоматически проиндексировано.

3. Нажмите кнопку **ОК**.

Так как каждый индекс требует дополнительной обработки, производительность при добавлении или обновлении данных снижается. Поэтому рекомендуется изменить значения, указанные в поле **Автоиндекс при импорте и создании** или уменьшить их число, чтобы сократить количество создаваемых индексов.

Пример.

Установка ключевых полей.

Отдельные таблицы, содержащие информацию по определенной теме, необходимо связать в единую структуру базы данных. Для связывания таблиц следует задать **ключевые поля**.

Ключ состоит из одного или нескольких полей, значения которых однозначно определяют каждую запись в таблице. Наиболее подходящим в качестве ключевого поля является *Счетчик*, так как значения в данном поле являются уникальными (т. е. исключают повторы).

При создании таблиц в режиме конструктора ключевое поле устанавливается автоматически.

Откройте созданные Вами таблицы в **режиме Конструктор** и проверьте установленные ключевые поля:

1) в таблице *Сотрудники* ключевое поле *Код сотрудника*


сотрудники

Имя поля	Тип данных
Код сотрудника	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Отчество	Текстовый
Должность	Текстовый
Телефон	Текстовый
Адрес	Текстовый
Дата рождения	Дата/время
Заработная плата	Денежный
Фото	Поле объекта OLE
Электронная почта	Гиперссылка

2) в таблице *Клиенты* ключевое поле *Код клиента*

3) в таблице *Заказы* ключевое поле *Код заказа*

Если значение *Ключевых полей* не задалось автоматически, то задайте их вручную. Для этого откройте **таблицу Сотрудники** в **режиме Конструктора**.

Нажмите правой кнопкой мыши на поле **Код сотрудника** и в появившемся контекстном меню выберите команду **Ключевое поле**. Если в таблице необходимо установить несколько ключевых полей, то выделить их можно, удерживая клавишу Ctrl. Для **таблицы Клиенты** установите ключевое поле **Код клиента**, а для **таблицы Заказы - Код заказа.**

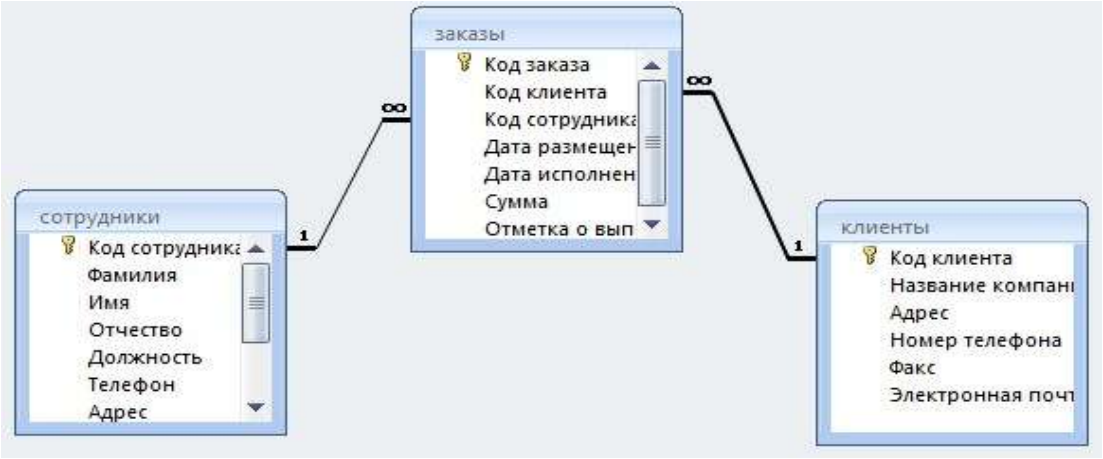
Создание связей между таблицами.

Существует несколько типов отношений между таблицами:

- *при отношении «один-к-одному»* каждой записи ключевого поля в первой таблице соответствует только одна запись в связанном поле другой таблицы, и наоборот. Отношения такого типа используются не очень часто. Иногда их можно использовать для разделения таблиц, содержащих много полей, для отделения части таблицы по соображениям безопасности;

- *при отношении «один-к-многим»* каждой записи в первой таблице соответствует несколько записей во второй, но запись во второй таблице не может иметь более одной связанной записи в первой таблице;

- *при отношении «многие-к-многим»* одной записи в первой таблице могут соответствовать несколько записей во второй таблице, а одной записи во второй таблице могут соответствовать несколько записей в первой.



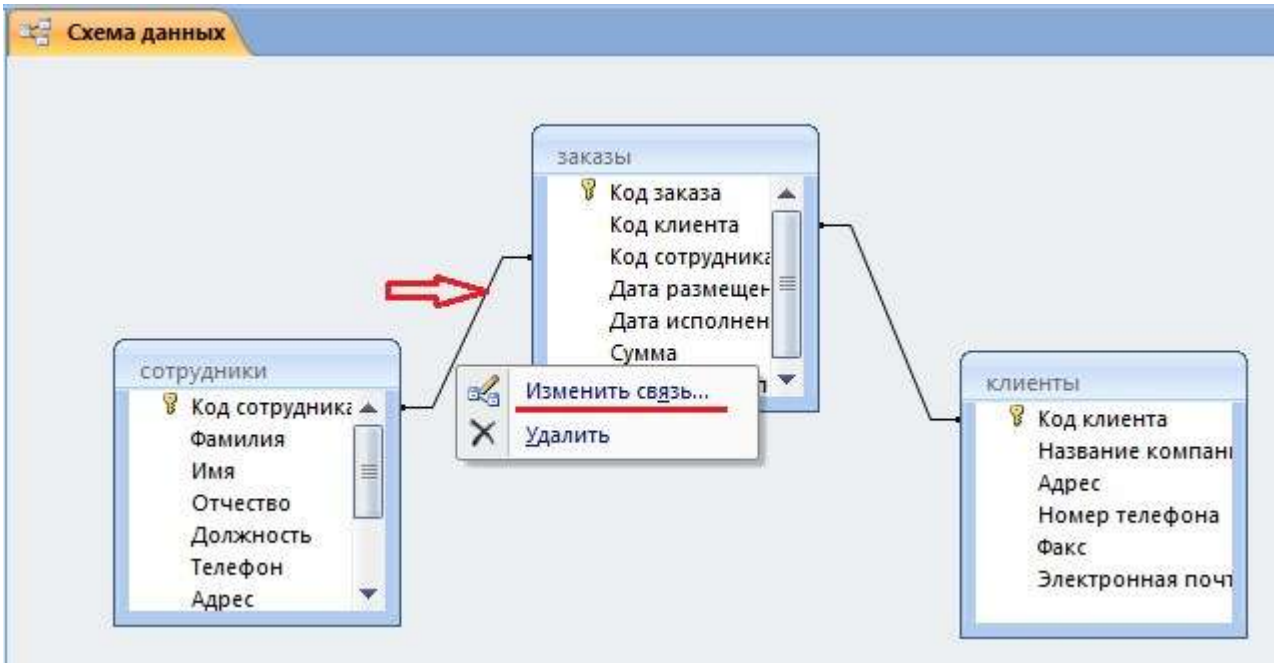
Закройте все открытые таблицы, так как создавать или изменять связи между открытыми таблицами нельзя.

Выполните команду вкладки Лента **Работа с базами данных** кнопка **Схема данных**

Если ранее никаких связей между таблицами базы не было, то при открытии окна **Схема данных** одновременно открывается окно **Добавление таблицы**, в котором выбираются нужные таблицы. Для добавления в схему данных новой таблицы необходимо щелкнуть правой кнопкой мыши на схеме данных и в контекстном меню выбрать пункт **Добавить таблицу.**

Если связи между таблицами уже были заданы, то откроется окно **Схема данных**, на котором будут отображены таблицы и связи между ними.

Отредактируйте связь между таблицами Сотрудники и Заказы, для этого щелкните правой



кнопкой мыши (ПКМ) на линию связи и в открышемся контекстном меню выберите команду **Изменить связь.**

Откроется диалоговое окно **Изменение связей**, в котором включите флажок **Обеспечение целостности данных**. Это позволит предотвратить случаи удаления записей из одной таблицы, при которых связанные с ними данные других таблиц останутся без связи. Обратите внимение на **тип отношений: один-ко-многим.**

Флажки **Каскадное обновление связанных полей** и **Каскадное удаление связанных записей** обеспечивают одновременное обновление или удаление данных во всех подчиненных таблицах при их изменении в главной таблице. Параметры связи можно изменить, нажав на кнопку Объединение. После установления всех необходимых параметров нажмите **кнопку ОК**

Аналогично измените связь между таблицами Клиенты и Заказы.

В результате должна получиться схема данных, представленная на рисунке

На схеме данных связи отображаются в виде соединительных линий со специальными значками около таблиц. Связь «один-к-многим» помечается «1» вблизи главной таблицы (имеющей первичный ключ) и «∞» вблизи подчиненной таблицы (имеющей внешний ключ). Связь «один-к-одному» помечается двумя «1» (оба поля таблиц имеют первичные ключи). Неопределенная связь не имеет никаких знаков. Если установлено объединение, то его направление отмечается стрелкой на конце соединительной линии (ни одно из объединенных полей не является ключевым и не имеет уникального индекса).

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Основные понятия языка SQL.

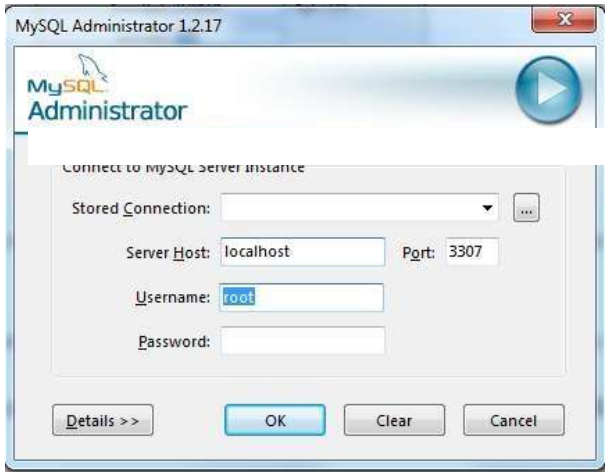
Лабораторное занятие№4

Резервное копирование и восстановление базы данных.

Цель: получение практических навыков по освоению операций создания резервной копии базы данных и восстановления базы данных из резервной копии на другом ПК.

Выполнив работу, Вы будете: ***уметь:***

У2 Использовать язык запросов для программного извлечения сведений из баз данных,



Материальное обеспечение:

Методические указания для выполнения практических работ, компьютер, программное обеспечение: СУБД MySQL

Порядок выполнения работы:

Для создания резервных копий базы данных в СУБД MySQL как одним из вариантов можно воспользоваться программой MySQL Administrator из пакета программ для работы с СУБД MySQL .

Чтобы создать резервную копию текущего состояния базы данных необходимо: Запустить MySQL Administrator

Выполнить подключение к СУБД MySQL от пользователя с полными правами (Рисунок 24).

Рисунок 1 - Подключение к СУБД MySQL

После того как подключение установлено появиться основной интерфейс программы MySQL Administrator, в котором необходимо выбрать раздел **Backup**

Далее следует нажать **New Project**

Затем необходимо выбрать базу данных, резервную копию которой необходимо сделать и нажать на кнопку со значком стрелки, направленной вправо

Для начала процесса резервного копирования выбранной базы данных в файл необходимо нажать кнопку **Execute Backup Now.**

Восстановление из резервной копии базы данных также происходит через приложение **MySQL Administrator**. Для восстановления необходимо выполнить следующие шаги:

1. Запустить MySQL Administrator
2. Выполнить подключение к СУБД MySQL от пользователя с полными правами. Выполнение подключения демонстрируются на рисунке 1.
3. После того как подключение установлено появиться основной интерфейс программы MySQL Administrator, в котором необходимо выбрать раздел Restore

4. Выполнить нажатие на кнопку **Open Backup File**
5. Выбрать файл резервной копии с расширением **.sql**
6. Нажать на кнопку **Start Restore**

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Основные понятия языка SQL.

Лабораторное занятие№5

Вставка данных. Импорт и экспорт данных.

Цель: получение практических навыков по освоению операций добавления записей в базу данных и навыков импортирования данных в базу данных

Выполнив работу, Вы будете: *уметь:*

У2 Использовать язык запросов для программного извлечения сведений из баз данных

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MySQL, MS Office

Задание 1:

В СУБД MySQL заполнить базу данных (свой вариант) значениями.

Порядок выполнения работы: Используя команду INSERT вставить данные в таблицы базы данных.

1. Создайте базу данных на сервере MySQL.
2. Сценарий SQL предоставлен так, чтобы создать большинство таблиц и вставки данных в них. Все, что нужно сделать, это импортировать сценарий SQL в вашу базу данных. database.sql.
3. Таблица Сотрудники (персонал, должности) не включены в этот сценарий SQL. Обратитесь к диаграмме базы данных (ERD) и словарю данных.
4. Создайте таблицы сотрудников (персонал, положение и расписаний) согласно спецификации.
5. Все данные сотрудников представлены в файле employee-import.
6. Эти данные не отформатированы для импортирования непосредственно в базу данных, необходимо отформатировать данные и загрузить их в таблицы, которые вы только что создали.
7. В поле " Full Name" в формате "Имя Фамилия" используются разные символы разделителя.
8. Убедитесь, что адреса электронной почты в правильном формате

Краткие теоретические сведения:

Для добавления одной или нескольких строк в таблицу используется команда:

INSERT [INTO] <Имя таблицы> [(<Список столбцов>)]

VALUES

(<Список значений 1>), (<Список значений 2>), ...
(<Список значений N>);

В команде INSERT используются следующие основные параметры. Имя таблицы, в которую добавляются строки.

• Список имен столбцов, для которых будут заданы значения. Если значения будут заданы для всех столбцов таблицы, то приводить список столбцов необязательно. Если столбец таблицы не включен в список, то в этом столбце при добавлении строки будет автоматически установлено значение по умолчанию.

• Значения, которые нужно добавить в таблицу. Значения могут указываться в одном из следующих форматов:

– набор значений для каждой добавляемой строки заключается в скобки. Набор значений внутри каждой пары скобок должен соответствовать указанному списку столбцов, а если список столбцов не указан, то упорядоченному списку всех столбцов, составляющих таблицу (список столбцов таблицы можно просмотреть с помощью команды DESCRIBE. Значения внутри набора, а также сами наборы отделяются друг от друга запятыми;

– символьные значения, а также значения даты и времени приводятся в одинарных кавычках. Для числовых значений кавычки необязательны. Десятичным разделителем для чисел с дробной частью служит точка. Время и даты вводятся, соответственно, в формате «YYYY-MM-DD» и «HH:MM:SS»;

– чтобы ввести в столбец неопределенное значение, то необходимо указать вместо значения ключевое слово NULL *без кавычек* (слово в кавычках рассматривается как обычная символьная строка);

– вместо значения можно указать ключевое слово DEFAULT *без кавычек,* тогда в столбец будет введено значение по умолчанию (если оно задано для этого столбца). Например, добавьте в таблицу Product сведения о продукции компании.

INSERT INTO Product (name_pr, description, price, qty) VALUES

(' Инфракрасный обогреватель', '3 режима нагрева: 400 Вт, 800 Вт, 1200 Вт',1445.00, 4), ('Гриль', 'Мощность 1440 Вт. Быстрый нагрев',4115.00, 6),

('Кофеварка', 'Цвет: черный. Мощность: 450 Вт',1710.00, 10), ('Чайник', 'Цвет: белый. Мощность: 2200 Вт. Объем: 2 л',1725.00, 14), ('Утюг', 'Цвет: фиолетовый. Мощность: 1400 вт',5300.00, 16);

Эта команда добавляет значения в столбцы name_pr *(наименование), description (описание), price (цена) и* qty (количество) *таблицы Product. При этом в столбец id (идентификатор) автоматически вносятся порядковые номера строк, поскольку этот столбец имеет тип данных SERIAL.*

Заполним таблицу **"Специальности"**. Для заполнения этой таблицы в обозревателе объектов щелкните правой кнопкой мыши по таблице **"Специальности"** и в появившемся меню выберите пункт **"Изменить первые 200 строк"**. В рабочей области "Microsoft SQL Server Management Studio" проявится окно заполнения таблиц. Заполните таблицу **"Специальности".**

Так как поле **"Код специальности"** является первичным полем связи и ключевым числовым счетчиком, то оно заполняется автоматически (заполнять его не нужно).

Закройте окно заполнения таблицы **"Специальность"** щелкнув по кнопке закрытия окна в верхнем правом углу, над таблицей.

После заполнения таблицы **"Специальности"** заполним таблицу **"Предметы"**. Откройте ее для заполнения как описано выше, и заполните.

Для заполнения дат в качестве разделителя можно использовать знак ".". Даты можно заполнять в формате "день.месяц.год".

Поле **"Код специальности"** является вторичным полем связи (для связи с таблицей **"Специальности"**). Следовательно, значения этого поля необходимо заполнять значениями поля **"Код специальности"** таблицы **"Специальности"**. В нашем случае это значения от 1 до 5. Если у Вас коды специальностей в таблице **"Специальности"** имеют другие значения, то внесите их в

таблицу «**Студенты».**

Поля с датами заполняются, как и в таблице **"Студенты"**. Поля **"Код предмета 1"**, **"Код предмета 2"** и **"Код предмета 3"** являются вторичными полями связи с таблицей **"Предметы".** Поэтому они должны быть заполнены значениями поля **"Код предмета из этой таблицы"**, то есть значениями от 1 до 5.

Закройте окно заполнения таблицы **"Оценки"**.

Задание3:

Выполнить импорт данных в базу данных.

Краткие теоретические сведения:

Если требуется добавить в таблицу большой массив данных, удобно использовать для этого команду загрузки данных из файла. Загрузка из файла выполняется программой MySQL значительно быстрее, чем вставка строк с помощью команды INSERT.

Например, чтобы загрузить данные в таблицу Customers, выполните следующие действия.

1. Запустите стандартную программу Windows Блокнот (Пуск → Все программы → Стандартные → Блокнот).

2. В окне программы Блокнот введите данные, используя для отделения значений друг от друга клавишу Tab, а для перехода на следующую строку – клавишу Enter.

Вместо отсутствующего значения необходимо при заполнении файла ввести символы «\N». Тогда в базу данных будет загружено неопределенное значение (NULL).

3. Для сохранения файла с данными нажмите комбинацию клавиш Ctrl+S. В стандартном окне Windows *Сохранить как* выберите папку, в которую нужно поместить файл (например, C: \data). Введите имя файла (например, Customers. txt) и нажмите кнопку Сохранить.

4. Для загрузки данных из созданного файла выполните команду LOAD DATA LOCAL INFILE 'C:/data/Customers.txt'

INTO TABLE Customers CHARACTER SET utf8;

Обратите внимание, что в пути к файлу необходимо использовать прямую косую черту, а не обратную.

Форма представления результата:

Отчет по выполненной лабораторной работе **Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Основные понятия языка SQL

Лабораторное занятие№6

Модификация данных. Использование условий при модификации данных. Цель: получение практических навыков модификации данных в базе.

Выполнив работу, Вы будете: ***уметь:***

У2 Использовать язык запросов для программного извлечения сведений из баз данных.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MySQL, MS Office

Задание:

Разработать модифицирующие запросы к базе данных Туризм.

Порядок выполнения работы:

1. Создайте обобщенную таблицу **Договоры по странам**

2. Создайте запросы по заданию.

Ход работы.

Краткие теоретические сведения:

Модификация базы данных с помощью запросов на изменение **Запрос на обновление**

Запрос этого типа используется при необходимости внесения изменений во множество записей базы данных, поэтому целесообразно сделать резервную копию таблицы.

Выполняется этот вид запроса в два этапа: сначала проверяется правильность отбора обновляемых записей с помощью запроса на выборку, затем он преобразуется в запрос на обновление и выполняется повторно.

При обновлении полей следует иметь в виду, что если при проектировании таблицы в свойствах поля было указано «условие на значение», то при обновлении этого поля условие может быть нарушено, чего не допустит MS access. Поэтому нужно: или изменить условие на значение, или удалить это условие в Конструкторе. **Запрос на добавление**

Периодически убирая в архивные таблицы «старые» записи, можно увеличить быстродействие основных частей и улучшить обзорность базы данных.

Кроме того, при необходимости добавить данные в таблицу базы данных из другой базы можно также использовать запросы на добавление.

Запрос на удаление

«Старые» или неиспользуемые записи таблиц можно удалить, но обязательно сначала произвести выборку и проверить ее. Целесообразно сделать копию.

Запрос на создание

1. Создайте обобщенную таблицу **Договоры по странам**, включив в нее следующие поля:

Из таблицы **Договоры**: ***Номер договора***; ***Название клиента*** Из таблицы **Страны**: ***Название страны***; ***Регион***.

Для этого:

- Создайте запрос на выборку этих данных, выполните его и проверьте результаты; Если результаты корректны, то поменяйте статус у запроса: **Запрос – Создание**
- **таблицы** – укажите новое имя таблицы **Договоры по странам;** Выполните запрос с новым статусом еще раз;
- Перейдите на вкладку **Таблицы** и убедитесь, что появилась новая таблица. Просмотрите ее.

Запрос на обновление

2. Увеличьте ***Размер оклада*** у менеджеров по продажам на 15%. Для этого:

- Составьте новый запрос на выборку, включив в него поля ***Фамилия***, **Должность** и *Размер оклада*;
- Проверьте составленный запрос;
- Видоизмените запрос, установив ему статус «*Обновление*» (**Запрос – Обновление**). В появившейся в бланке запроса строке «*Обновление*» для поля ***Размер оклада*** внесите с помощью **Построить** выражение [Размер оклада]*1,15;
- Выполните запрос, подтвердите обновление; сохраните запрос, дав ему имя и обратив внимание на появившийся значок у его имени; просмотрите результаты.

Запрос на добавление

3. Создайте путем копирования дубликат таблицы **Договоры** без данных, назвав ее **Договоры 2008 года**. Для этого в контекстном меню для таблицы Договоры выберите *Копировать*, затем выполните команду *Вставить*, в параметрах вставки укажите «*Только структуру*». Просмотрите таблицу **Договоры 2008 года** – она должна быть пустой и иметь такую же структуру, как и таблица **Договоры**.

4. Отберите в таблицу **Договоры 2008 года** записи обо всех договорах этого года. Для этого:

- Создайте запрос на выборку, включив в него все поля таблицы **Договоры** в любой последовательности, и критерий по дате, выполните его для проверки правильности;
- Измените статус запроса на «*Добавление*», в появившемся окне задайте имя таблицы для добавления **Договоры 2008 года**, обратите внимание на появление строки

«*Добавление*» в бланке запроса;

- Выполните запрос и подтвердите добавление; просмотрите результаты архивации и сохраните запрос, обратив внимание на значок у его имени.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Основные понятия языка SQL.

Лабораторное занятие№7 Удаление данных.

Цель: получение практических навыков удаления данных в базе.

Выполнив работу, Вы будете: *уметь:*

У2 Использовать язык запросов для программного извлечения сведений из баз данных.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MySQL, MS Offfice

Задание:

Разработать запросы на удаление к базе данных Туризм.

Порядок выполнения работы:

1. Выполнить задания в MySQL.
2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика№_Фамилия, с расширением doc, сохраненного в своей папке.

Ход работы.

1. Удалите из таблицы **Договоры** записи о договорах 2008 года, используя копию сохраненного запроса на добавление в таблицу **Договоры 2008 года**, изменив его статус на «*Удаление*».

2. Удалите все договоры из определенной страны.

Форма представления результата:

Отчет по выполненной лабораторной работе **Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.2 Организация SQL запросов

Лабораторное занятие№8

Организация SQL запросов на выборку данных.

Цель: получение практических навыков по освоению операций создания запросов с помощью языка SQL

Выполнив работу, Вы будете: ***уметь:***

У2. Использовать язык запросов для программного извлечения сведений из баз данных.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS Access, MySQL.

Задание :

Разработать запросы для базы данных Туризм с помощью конструктора в СУБД MS Access.

Порядок выполнения работы:

1. Выполнить задания в MySQL.
2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика1_Фамилия, с расширением doc, сохраненного в своей папке.

Краткие теоретические сведения:

Запрос является объектом базы данных. Он представляет собой сформулированную информационную потребность.

При работе с запросом можно выделить два этапа: формирование (проектирование) и выполнение. При выполнении запроса выбирается информация из всех таблиц базы данных в соответствии с критерием запроса.

В верхней части окна **Конструктора** размещаются нужные таблицы посредством команды ***Запрос – Добавить таблицу*** или та же команда в контекстном меню. В нижней части окна расположен бланк запроса, информация в него заносится путем перетаскивания нужных полей из таблиц в верхней части окна в строку «поле» или двойным щелчком мыши. При этом имя таблицы в бланке подставляется автоматически.

Наличие «галочки» в строке «Вывод на экран» означает присутствие данного поля в таблице результатов поиска. Критерии запроса устанавливаются в строке «Условия отбора» и последующих строках, связанных логическим оператором OR. Все критерии отбора, указанные в одной строке, объединяются оператором AND.

В качестве «Условия отбора» могут быть выражения (вычисляемое поле) даты, текст, которые либо вносятся вручную, либо инструментом, либо с помощью команды контекстное меню ***Построить.*** Константы типа Дата/Время заключаются в #.

Запросы бывают разных типов: *на выборку, на создание, на обновление, на добавление, на удаление, перекрестный, итоговый, параметрический* и др. По умолчанию формируется запрос на выборку. Тип запроса может быть преобразован в любой другой командой ***Запрос*.**

При создании критерия можно использовать инструмент ***Построить*** или такую же команду контекстного меню для категории «условие отбора».

Ход работы.

Запрос на выборку

1. Откройте базу данных ***Туризм*** и перейдите на вкладку **Создать - Запрос.**

2. В режиме **Конструктора** создайте и сохраните следующие запросы на выборку, определив нужные таблицы:

- список всех путешествий в определенную страну (например, Испанию);
- список всех регионов в конкретной стране (например, Англии). Сохраните запрос под именем «*Страна-Регион*»;
- все туры, проданные в 2008 году. Сохраните запрос с именем «*Туры 2008*»;
- список сотрудников, работающих с 1999 года и раньше. Сохраните запрос с именем «Ветераны». Добавьте в запрос строку «Сортировка» и установите сортировку по фамилиям.
- сотрудникам, которые родились в 1973 г., используя в качестве критерия выражение:

Between... and (**Построить — Операторы — Сравнения — Between**), а затем повторите запрос, построив выражение с помощью знаков «<» и «>»;

- сотрудникам, фамилии которых с «Г» по «Я»;
- сотрудникам, фамилии которых начинаются с «Н» по «Я» и с «А» по «В»; индивидуальным клиентам, фамилии которых имеют вторую букву «о»;
- пяти фамилиям сотрудников, которые начинаются с букв «А» или «В». постоянным клиентам, количество договоров с которыми больше 3.

Форма представления результата:

Отчет по выполненной лабораторной работе **Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.2 Организация SQL запросов

Лабораторное занятие№9

Организация SQL запросов на выборку данных. Операция конкатенации. Использование строковых функций.

Цель: получение практических навыков по освоению операции конкатенации и строковых функций в запросах на языке SQL

Выполнив работу, Вы будете: ***уметь:***

У2. Использовать язык запросов для программного извлечения сведений из баз данных.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS Access, MySQL.

Задание: Составить запросы для Учебной базы данных

Порядок выполнения работы:

1. Выполнить задания в MySQL.
2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика1_Фамилия, с расширением doc, сохраненного в своей папке.

Ход работы.

Краткие теоретические сведения:

Операция конкатенации «**&**» в СУБД Access позволяет соединять («склеивать») значения двух или более столбцов символьного типа или символьных констант в одну строку.

Пример:

```
SELECT SURNAME&'__'&NAME, STIPEND FROM STUDENT
WHERE KURS = 4 AND STIPEND> 0;
```

Функции преобразования символов в строке

LCASE(строка)-перевод в строчные символы **UCASE(строка)**-перевод в прописные символы **Пример:**

```
SELECT LCASE (SURNAME), UCASE (NAME) FROM STUDENT
WHERE KURS = 4 AND STIPEND> 0;
```

Строковые функции

LEN (строка) –определение длины строки тип возвращаемого значения INT

функция возвращает NULL, если строка имеет NULL значение ***LEFT (строка, длина)*** – выделение левой подстроки

RIGHT (строка, длина) – выделение правой подстроки ***MID (строка, начало, количество)*** – выделение подстроки ***LTRIM (строка)*** – удаляет пробелы в начале строки ***RTRIM (строка)*** – удаляет пробелы в конце строки

STR (число) – выполняет конвертирование числового значения в символьный формат ***INSTR (строка, подстрока [, <начало поиска> [, <номер вхождения>]])*** -

возвращает позицию найденной подстроки (тип возвращаемого значения- INT), где

<начало поиска> задает начальную позицию в строке для поиска<подстроки>. Если не задано, то по умолчанию применяется значение 1;

<номер вхождения> задает порядковый номер искомой подстроки. Если не задан, то по умолчанию применяется значение 1.

Значение выражений в <начале поиска> или <номера вхождения> должны иметь беззнаковый целый тип или приводиться к этому типу

Примеры:

```
SELECT SURNAME,LEN (SURNAME),KURS FROM STUDENT
WHERE KURS = 3;

SELECT SURNAME, Right(SURNAME,2) FROM STUDENT
WHERE KURS IN (2, 3, 4);
```

Задания:

1. Составьте запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы выходная таблица была представлена одним столбцом, содержащим последовательность, разделенных символом «;» значений всех столбцов этой таблицы; результат необходимо представить прописными символами (например, **ИВАН;ИВАНОВ;150;1;ОРЁЛ;3/12/82;10**).
2. Составьте запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы выходная таблица содержала один столбец в следующем виде: **Б.КУЗНЕЦОВ; место жительства – БРЯНСК; родился – 8.12.81.**
3. Составьте запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы столбец выходной таблицы был представлен в следующем виде: **Б;Кузнецов;место жительства-Брянск;родился-08.12.1981**
4. Составьте запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы выходная таблица содержала один столбец в следующем виде: **Борис Кузнецов родился в 1981 году.**
5. Выведите имена и фамилии студентов и объем получаемой ими стипендии, увеличенной в 10 раз.
6. Составьте запрос для таблицы STUDENT «Учебной базы данных» следующим образом: **БОРИС КУЗНЕЦОВ родился в 1981 году**, только для студентов 1, 2 и 4-го курсов.
7. Составьте запрос для таблицы UNIVERSITY «Учебной базы данных» таким образом, чтобы выходная таблица содержала один столбец в следующем виде: **Код – 10;ВГУ – г.ВОРОНЕЖ;Рейтинг=296.**

Форма представления результата:

Отчет по выполненной лабораторной работе **Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.2 Организация SQL запросов

Лабораторное занятие№10

Организация SQL запросов на выборку данных. Операции с данными.

Цель: получение практических навыков по освоению операций создания запросов с вычисляемыми полями с помощью языка SQL

Выполнив работу, Вы будете: ***уметь:***

У2. Использовать язык запросов для программного извлечения сведений из баз данных.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS Access, MySQL.

Задание 1:

Разработать запросы для базы данных Туризм с помощью конструктора в СУБД MS Access.

Порядок выполнения работы:

1. Выполнить задания в MySQL.
2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика1_Фамилия, с расширением doc, сохраненного в своей папке.

Краткие теоретические сведения:

3. Запрос является объектом базы данных. Он представляет собой сформулированную информационную потребность.
4. При работе с запросом можно выделить два этапа: формирование (проектирование) и выполнение. При выполнении запроса выбирается информация из всех таблиц базы данных в соответствии с критерием запроса.
5. В верхней части окна **Конструктора** размещаются нужные таблицы посредством команды ***Запрос – Добавить таблицу*** или та же команда в контекстном меню. В нижней части окна расположен бланк запроса, информация в него заносится путем перетаскивания нужных полей из таблиц в верхней части окна в строку «поле» или двойным щелчком мыши. При этом имя таблицы в бланке подставляется автоматически.
6. Наличие «галочки» в строке «Вывод на экран» означает присутствие данного поля в таблице результатов поиска. Критерии запроса устанавливаются в строке «Условия отбора» и последующих строках, связанных логическим оператором OR. Все критерии отбора, указанные в одной строке, объединяются оператором AND.
7. В качестве «Условия отбора» могут быть выражения (вычисляемое поле) даты, текст, которые либо вносятся вручную, либо инструментом, либо с помощью команды контекстное меню ***Построить.*** Константы типа Дата/Время заключаются в #.
8. Запросы бывают разных типов: *на выборку, на создание, на обновление, на добавление, на удаление, перекрестный, итоговый, параметрический* и др. По умолчанию формируется запрос на выборку. Тип запроса может быть преобразован в любой другой командой ***Запрос***.
9. При создании критерия можно использовать инструмент ***Построить*** или такую же команду контекстного меню для категории «условие отбора».
10. ***1. Вычисляемые поля в запросах***
11. С помощью запросов можно задать вычисления над данными и сделать вычисляемое поле новым полем в наборе данных. Для создания нового поля в пустой ячейке строки ***Поле*** в бланке запроса вводится формула:
12. **Имя поля: выражение**
13. Для построения выражений имеется специальное средство – **Построитель** выражений, вызываемый правой кнопкой мыши на поле или кнопкой ***Построить.***

14. В верхней части размещается область ввода. Нижняя содержит три списка для выбора имен полей и функций. В папке ***Функции*** размещаются встроенные функции, сгруппированные по категориям.

15. 2. ***Параметрические запросы***

16. Условия запроса могут быть включены непосредственно в бланк запроса, но для того чтобы сделать его более универсальным, можно вместо конкретного значения отбора включить в запрос параметр, т.е. создать параметрический запрос. Для этого в строку «условия отбора» вводится фраза в квадратных скобках, которая будет выводиться в качестве «подсказки» в процессе диалога, например, [введите фамилию]. Таких параметров может быть несколько, каждый для своего поля.

Задания:

Запросы с вычисляемыми полями

1. Создайте запрос для расчета ведомости заработной платы для сотрудников агентства, включив в нее следующие поля: ***Фамилия сотрудника, Размер оклада, Стаж, Надбавка, Налог, На руки.***

Для поля ***Стаж*** нужно использовать формулу, построенную с помощью кнопки **Построить**, в которой учитывается сегодняшняя дата и **Дата найма** на работу:

Стаж = (Date()-Сотрудники!ДатаНайма)\365

Для поля ***Надбавка*** нужно исходить из того, что она составляет 20% от ***Размера оклада***, если ***Стаж*** меньше 5 лет, и 30% — если стаж больше 5 лет:

IIf ([стаж]<5;0,2*[Сотрудники]![Размер оклада]; 0,3* [Сотрудники]! [Размер оклада])

Поле ***Налог*** рассчитывается как 13% от Размера оклада: [Сотрудники]![Размер оклада] *0,13

Поле На руки рассчитывается как: [Размер оклада]+[надбавка]—[налог].

В результате выполнения запроса будет получена новая ведомость.

2. Создайте запрос для определения стоимости путевок корпоративных клиентов, включив в него поля Клиент, Стоимость путевки

Стоимость путевки = Sum(договоры![Цена тура] *договоры![Число туристов])

Параметрические запросы

3. Сформируйте запрос для выборки всех туров по названию страны.

4. Создайте запрос для получения данных на сотрудников, работающих по турам в конкретную страну.

5. Создайте запрос по всем клиентам, оформившим договоры в определенную страну и регион.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату. Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.2 Организация SQL запросов

Лабораторное занятие№11 Вычисления в SQL запросах. Упорядочение выборки.

Цель: получение практических навыков по освоению вычислений в запросах и применение упорядочения выборки.

Выполнив работу, Вы будете: *уметь:*

У2. Использовать язык запросов для программного извлечения сведений из баз данных.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS Access, MySQL.

Задание 1:

Разработать запросы для Учебной базы данных

Порядок выполнения работы:

1. Выполнить задания в MySQL.
2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика1_Фамилия, с расширением doc, сохраненного в своей папке.

Ход работы.

Краткие теоретические сведения:

Оператор SELECT (выбрать) языка SQL является самым важным и самым часто используемым оператором. Он предназначен для выборки информации из таблиц базы

данных. Упрощенный синтаксис оператора SELECT выглядит следующим образом: SELECT [DISTINCT] <список атрибутов>

FROM <список таблиц> [WHERE <условие выборки>]

[ORDER BY <список атрибутов>] [GROUP BY <список атрибутов>] [HAVING <условие>]

[UNION <выражение с оператором SELECT>];

В квадратных скобках указаны элементы, которые могут отсутствовать в запросе. Ключевое слово SELECT сообщает базе данных, что данное предложение является

запросом на извлечение информации. После слова SELECT через запятую перечисляются наименования полей, содержимое которых запрашивается.

Обязательным ключевым словом в предложении-запросе SELECTявляется слово FROM (из). За ключевым словом FROM указывается список разделенных запятыми имен таблиц, из которых извлекается информация.

Например,

SELECT NAME, SURNAME FROM STUDENT;

Любой SQL-запрос должен заканчиваться символом «;».

Если необходимо вывести значения всех столбцов таблицы, то можно вместо перечисления их имен использовать символ «*»/

SELECT *

FROM STUDENT;

Для исключения из результата SELECT-запроса повторяющихся записей используется ключевое слово DISTINCT (отличный). Если запрос SELECT извлекает множество полей, то DISTINCT исключает дубликаты строк, в которых значения всех выбранных полей идентичны.

Использование в операторе SELECT предложения, определяемого ключевым словом WHERE (где), позволяет задавать выражение условия, принимающее значение истина или ложь для значений полей строк таблиц, к которым обращается оператор SELECT. Предложение WHERE определяет, какие строки указанных таблиц должны быть выбраны.

3.

Упорядочение выходных полей (ORDER BY) .

ORDER BY позволяет упорядочивать записи в соответствии со значениями одного или нескольких выбранных полей. При этом можно задать возрастающую (ASC) и убывающую (DESC) последовательность сортировки для каждого из столбцов.

По умолчанию принята возрастающая последовательность сортировки.

Задания:

1. Выбрать все данные из таблицы предметов обучения SUBJECT с упорядочением по наименованиям предметов.

```
SELECT * FROM SUBJECT
ORDER BY SUBJ_NAME;
```

2. Тот же список, но упорядоченный в обратном порядке.

```
SELECT * FROM SUBJECT
ORDER BY SUBJ_NAME DESC;
```

3. Упорядочить выводимый список предметов по семестрам, а внутри семестров – по наименованиям предметов.

```
SELECT * FROM SUBJECT
ORDER BY SEMESTER, SUBJ_NAME;
```

4. При упорядочении вместо наименований столбцов можно указывать их номера, имея, однако, в виду, что данном случае это номера столбцов указанные при определении выходных данных в запросе, а не номера столбцов в таблице. Полем с номером 1 является первое поле, указанное в предложении ORDER BY- независимо от его расположения в таблице.

```
SELECT SUBJ_ID, SEMESTER FROM SUBJECT
ORDER BY 2 DESC;
```

Если в поле, которое используется для упорядочения, существуют NULL- значения, то все они размещаются в конце или предшествуют всем остальным значениям этого поля.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено

Тема 2.2 Организация SQL запросов

Лабораторное занятие№12

Организация SQL запросов на выборку данных с использованием составных условий.

Цель: получение практических навыков по формированию запросов с составными условиями.

Выполнив работу, Вы будете: *уметь:*

У2. Использовать язык запросов для программного извлечения сведений из баз данных.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS Access, MySQL.

Задание 1:

Разработать запросы для Учебной базы данных

Порядок выполнения работы:

1. Выполнить задания в MySQL.
2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика1_Фамилия, с расширением doc, сохраненного в своей папке.

Краткие теоретические сведения:

Использование в операторе SELECT предложения, определяемого ключевым словом WHERE(где), позволяет задавать выражения условия (предикат), принимающее значение *истина* или *ложь* для значений полей строк таблиц, к которому обращается оператор SELECT.

Предложение WHERE определяет, какие строки *указанных таблиц должны быть выбраны.* В таблицу, являющуюся результатом запроса, включается только те строки, для которых условие(предикат), в предложении WHERE, принимает значение *истина.*

Операторы **IN** (равен любому из списка) **NOT IN** (не равен ни одному из списка) используется для сравнения проверяемого значения поля с заданным списком. Этот список значений указывается в скобках справа от оператора IN. Построенный с использованием IN предикат (условие) считается истинным, если значения поля, имя которого указано слева от IN, *совпадает* (подразумевается точное совпадение) с одним из значений, перечисленных в списке, указанном в скобках справа от IN.

Предикат, построенный с использованием NOT IN, считается истинным, если значение поля, имя которого указано слева от NOT IN, не совпадает ни с одним из значений, перечисленных в списке, указанном в скобках от NOT IN.

Оператор **BETWEEN** используется для проверки условия вхождения значения поля в заданный интервал, то есть вместо списка значений атрибута этот оператор задает границы его изменения.

Оператор **LIKE** применим только к символьным полям типа CHAR или VARCHAR. Этот оператор просматривает строковые значения полей с целью определения, входит ли заданная в операцию LIKE подстрока (образец поиска) в символьную строку-значение проверяемого поля. Можно применять шаблон:

Символ подчеркивания «**?**» указанный в шаблоне, определяет возможность наличия в указанном месте одного любого символа;

Символ «*****» допускает присутствие в указанном месте проверяемой строки последовательности любых символов произвольной длины.

Like _[A-K]*‘ – значения атрибутов символьного типа, начинающиеся с А по К

Like _[!A-K]*‘ - значения атрибутов символьного типа, начинающиеся на любые символы кроме символов с А по К (с Л по Я)

Like _[A,K]*‘ - значения атрибутов символьного типа, начинающиеся с А или К Оператор NULL используется для проверки содержимого поля на наличие в нем

пустого значения. **IS NULL** (является пустым), **IS NOT NULL**(является не пустым).

Задания:

1. Написать запрос, выполняющий выборку имен всех студентов с фамилией Петров, сведения о которых находятся в таблице STUDENT

```
SELECT NAME, SURNAME FROM STUDENT
WHERE SURNAME =‘Петров‘;
```

2. В задаваемых предложениях WHERE условиях могут использоваться операции сравнения, определяемые операторами =(равно), >(больше), <(меньше), >=(больше или равно), <=(меньше или равно), <>(не равно), а также логические операторы AND, OR и NOT. Запрос для получения имен и фамилий студентов, обучающихся на третьем курсе и получающих стипендию(размер стипендии больше нуля), будет выглядеть таким образом:

```
SELECT NAME, SURNAME FROM STUDENT
WHERE KURS = 3 AND STIPEND> 0;
```

3. . Получить из таблицы EXAM_MARKS сведения о студентах, имеющих экзаменационные оценки только 4 и 5.

```
SELECT *
FROM EXAM_MARKS WHERE MARK IN (4, 5);
```

4. Получить из таблицы EXAM_MARKS сведения о студентах, не имеющих ни одной экзаменационные оценки только, 4 и 5.

SELECT *
FROM EXAM_MARKS WHERE MARK NOT IN (4, 5);

5. Запрос на вывод записей о предметах, на изучение которых отводиться количество часов, находящееся в пределах между 30 и 40, имеет вид:

SELECT * FROM SUBJECT
WHERE HOUR BETWEEN 30 AND 40;

6. Написать запрос, выбирающий из таблицы STUDENT сведения о студентах, фамилии которых начинаются на букву «К». SELECT *
FROM STUDENT
WHERE SURNAME LIKE ―К*‖;

7. Вывести все данные студента с номером 265;

8. Вывести информацию о студентах с именем Вадим;

9. Выбрать фамилии студентов со 2-го и 3-го курсов получающих стипендию; 10. Вывести имена, фамилии и даты рождения студентов 5 курса;

10. Вывести информацию о студентах из Воронежа;

11. Выбрать фамилию, имя, курс, город студентов получающих стипендию от 100 до 200 рублей;

12. Вывести список идентификаторов университетов, исключая повторения.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.2 Организация SQL запросов

Лабораторное занятие№13

Использование агрегированных функций для организации SQL запросов на выборку данных.

Цель: получение практических навыков по формированию запросов с использованием агрегированных функций

Выполнив работу, Вы будете: ***уметь:***

У2. Использовать язык запросов для программного извлечения сведений из баз данных

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS Access, MySQL.

Задание 1:

Разработать итоговые запросы на языке SQL для учебной базы данных.

Порядок выполнения работы:

1. Выполнить задания в MySQL.

2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика1_Фамилия, с расширением doc, сохраненного в своей папке.

Ход работы.

Краткие теоретические сведения:

Агрегирующие функции позволяют получать из таблицы сводную (агрегированную)

информацию, выполняя операции над группой строк таблицы. Для задания в SELECT-запросе агрегирующих операций используются следующие ключевые слова:

- COUNT определяет количество строк или значений поля;
- SUM вычисляет арифметическую сумму всех выбранных значений данного поля; - AVG вычисляет среднее значение для всех выбранных значений данного поля;
- MAX вычисляет наибольшее из всех выбранных значений данного поля; - MIN вычисляет наименьшее из всех выбранных значений данного поля.

Предложение GROUP BY позволяет группировать записи в подмножества, определяемые значениями какого-либо поля, и применять агрегирующие функции уже не ко всем записям таблицы, а раздельно к каждой сформированной группе.

При необходимости часть сформированных с помощью GROUP BY групп может быть исключена с помощью предложения HAVING.

Предложение HAVING определяет критерий, по которому группы следует включать в выходные данные, по аналогии с предложением WHERE, которое осуществляет это для отдельных строк.

В условии, задаваемом предложением HAVING, указывают только поля или выражения, которые на выходе имеют единственное значение для каждой выводимой группы.

Задания:

1. Необходимо подсчитать общее количество преподавателей из Москвы.

```
SELECT COUNT (SURNAME) FROM LECTURER
WHERE CITY='Москва';
```

2. Для подсчета общего количества строк в таблице следует использовать функцию COUNT со звездочкой:

```
SELECT COUNT (*) FROM LECTURER WHERE CITY='Москва';
```

Если аргументом функции является столбец, содержащий пустое значение ***NULL,*** то COUNT вернет число строк, которые не содержат пустые значения и к которым применимо определенное в COUNT условие или группирование.

Поведение функции COUNT(*) не зависит от пустых значений. Она возвращает общее количество строк в таблице.

3. Подсчитать общее количество студентов в таблице «STUDENT».

```
SELECT COUNT (*) AS Количество FROM STUDENT;
```

4. Подсчитать количество студентов, для которых указаны даты рождения.

```
SELECT COUNT (BIRTHDAY) AS Количество FROM STUDENT;
```

5. Для определения среднего значения поля MARK(оценки) по всем записям таблицы EXAM_MARKS можно использовать запрос с функцией AVG следующего вида:

```
SELECT ROUND (AVG (MARK),2) AS Средняя_оценка FROM EXAM_MARKS;
```

6. Функция **AVG** подсчитывает сумму известных значений и делит ее на ***количество*** этих значений, а не на общее количество значений, среди которых могут быть ***NULL***- значения. Если столбец состоит только из пустых значений, то функция AVG также возвратит ***NULL.***

7. Вычислить общую сумму стипендии для студентов всех курсов из таблицы «STUDENT».

```
SELECT SUM(STIPEND) AS Стипендия FROM STUDENT;
```

8. Вывести максимальное значение стипендии студентов из таблицы «STUDENT».

```
SELECT MAX (STIPEND) AS Максимальная_стипендия FROM STUDENT;
```

9. Вывести университет с минимальным рейтингом.

```
SELECT MIN (RATING) AS Минимальный_рейтинг FROM UNIVERSITY;
```

10. Составить запрос для подсчѐта количества студентов, сдававших экзамен по предмету обучения с идентификатором, равным 22 по таблице EXAM MARKS.

11. Составить запрос, который выполняет выборку для каждого студента значения его идентификатора и минимальной из полученных им оценок по таблице EXAM MARKS, используя предложение GROUP BY.

12. Составить запрос, выполняющий вывод фамилии первого в алфавитном порядке (по фамилии) студента, фамилия которого начинается на букву «П» по таблице STUDENT.

13. Составить запрос, который выполняет вывод (для каждого предмета обучения) наименования предмета и максимального значения номера семестра, в котором этот предмет преподаѐтся по таблице SUBJECT.

14. Составить запрос для определения количества студентов, сдававших каждый экзамен. 14. Составить запрос для определения количества студентов, проживающих в Воронеже.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.2 Организация SQL запросов

Лабораторное занятие№14 Использование группировки данных.

Цель: получение практических навыков применения агрегированных функций с использованием группировки.

Выполнив работу, Вы будете: ***уметь:***

У2. Использовать язык запросов для программного извлечения сведений из баз данных

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания,
компьютер, программное обеспечение: MS Access, MySQL.

Задание 1:

Разработать итоговые запросы на языке SQL для учебной базы данных.

Краткие теоретические сведения:

Для формирования групп в запросе SELECT используется предложение GROUP BY(группировать по…).

В предложении **GROUP BY** должны быть указаны все выбираемые столбцы, приведенные после ключевого слова SELECT, кроме столбцов, указанных в качестве аргумента в агрегирующей ФУНКЦИИ.

При необходимости часть сформированных с помощью GROUP BY групп может быть исключена с помощью предложения **HAVING**

HAVING определяет критерий, по которому группы следует включать в выходные данные, по аналогии с предложением WHERE, которое осуществляет это для отдельных строк.

Итак, если в операторе SELECT задается одновременно предложение WHERE и HAVING,то порядок выполнения следующий:

1. Отбираются все записи, удовлетворяющие условию WHERE 2. Из отобранных записей формируются группы

3. Для каждой группы вычисляются значения групповых функций 4. Отбираются группы, удовлетворяющие условию HAVING

Порядок выполнения работы:

1. Вывести предметы, на которые отводится более 36 часов.

```
SELECT SUBJ_NAME, MAX (HOUR) FROM SUBJECT
GROUP BY SUBJ_NAME HAVING MAX (HOUR) >=36;
```

Найти максимальное значение оценки, полученной каждым студентом.

```
SELECT STUDENT_ID, MAX(MARK) AS Оценки FROM EXAM_MARKS
GROUP BY STUDENT_ID;
```

В предложении GROUP BY для группирования может быть использовано более одного столбца.

3. Найти максимальное значение оценки, полученной каждым студентом по каждому предмету.

```
SELECT STUDENT_ID ,SUBJ_ID, MAX(MARK) AS Оценки FROM EXAM_MARKS
GROUP BY STUDENT_ID ,SUBJ_ID;
```

В данном случае строки вначале группируются по значениям первого поля, указанного в SELECT , а внутри этих групп – в подгруппы по значениям второго поля.

4. Предложение ORDER BY может использоваться с GROUP BY для упорядочения групп записей. При этом оператор ORDER BY в запросе всегда должен стоять последним.

```
SELECT SUBJ_NAME, SEMESTER, MAX(HOUR)AS Часы FROM SUBJECT
GROUP BY SEMESTER, SUBJ_NAME ORDER BY SEMESTER;
```

Форма представления результата:

Отчет по выполненной лабораторной работе **Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.2 Организация SQL запросов

Лабораторное занятие№15

Использование условий при модификации данных.

Цель: получение практических навыков использования условий при модификации данных.

Выполнив работу, Вы будете: ***уметь:***

У2. Использовать язык запросов для программного извлечения сведений из баз данных

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS Access, MySQL.

Задание 1:

Разработать запросы на модификацию данных по базе данных Туризм.

Краткие теоретические сведения: Запрос на обновление

Запрос этого типа используется при необходимости внесения изменений во множество записей базы данных, поэтому целесообразно сделать резервную копию таблицы.

Выполняется этот вид запроса в два этапа: сначала проверяется правильность отбора обновляемых записей с помощью запроса на выборку, затем он преобразуется в запрос на

обновление и выполняется повторно.

При обновлении полей следует иметь в виду, что если при проектировании таблицы в свойствах поля было указано «условие на значение», то при обновлении этого поля условие может быть нарушено, чего не допустит MS access. Поэтому нужно: или изменить условие на значение, или удалить это условие в Конструкторе.

Запрос на добавление

Периодически убирая в архивные таблицы «старые» записи, можно увеличить быстродействие основных частей и улучшить обзорность базы данных.

Кроме того, при необходимости добавить данные в таблицу базы данных из другой базы можно также использовать запросы на добавление.

Запрос на удаление

«Старые» или неиспользуемые записи таблиц можно удалить, но обязательно сначала произвести выборку и проверить ее. Целесообразно сделать копию.

Создайте обобщенную таблицу **Договоры по странам**, включив в нее следующие поля:

Из таблицы **Договоры**: ***Номер договора***; ***Название клиента*** Из таблицы **Страны**: ***Название страны***; ***Регион***.

Для этого:

Создайте запрос на выборку этих данных, выполните его и проверьте результаты; Если результаты корректны, то поменяйте статус у запроса: **Запрос – Создание**

таблицы – укажите новое имя таблицы **Договоры по странам**; Выполните запрос с новым статусом еще раз;

Перейдите на вкладку **Таблицы** и убедитесь, что появилась новая таблица. Просмотрите ее.

Увеличьте ***Размер оклада*** у менеджеров по продажам на 15%. Для этого:

Составьте новый запрос на выборку, включив в него поля ***Фамилия,*** **Должность** и *Размер оклада*;

Проверьте составленный запрос;

Видоизмените запрос, установив ему статус «*Обновление*» (**Запрос – Обновление**). В появившейся в бланке запроса строке «*Обновление*» для поля ***Размер оклада*** внесите с помощью **Построить** выражение [Размер оклада]*1,15;

Выполните запрос, подтвердите обновление; сохраните запрос, дав ему имя и обратив внимание на появившийся значок у его имени; просмотрите результаты.

Создайте путем копирования дубликат таблицы **Договоры** без данных, назвав ее **Договоры 2008 года**. Для этого в контекстном меню для таблицы Договоры выберите *Копировать*, затем выполните команду *Вставить*, в параметрах вставки укажите «*Только структуру*». Просмотрите таблицу **Договоры 2008 года** – она должна быть пустой и иметь такую же структуру, как и таблица **Договоры.**

6. Отберите в таблицу **Договоры 2008 года** записи обо всех договорах этого года. Для этого:

- Создайте запрос на выборку, включив в него все поля таблицы **Договоры** в любой последовательности, и критерий по дате, выполните его для проверки правильности;
- Измените статус запроса на «*Добавление*», в появившемся окне задайте имя таблицы для добавления **Договоры 2008 года**, обратите внимание на появление строки «*Добавление*» в бланке запроса;
- Выполните запрос и подтвердите добавление; просмотрите результаты архивации и сохраните запрос, обратив внимание на значок у его имени.
- Удалите из таблицы **Договоры** записи о договорах 2008 года, используя копию сохраненного запроса на добавление в таблицу **Договоры 2008 года**, изменив его статус на «*Удаление*».

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна

или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.2 Организация SQL запросов

Лабораторное занятие№16

Организация SQL запросов на выборку данных с использованием подзапросов

Цель: получение практических навыков составления запросов с использованием подзапросов.

Выполнив работу, Вы будете: ***уметь:***

У2. Использовать язык запросов для программного извлечения сведений из баз данных

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: MS Access, MySQL.

Задание 1:

Составить запросы по Учебной базе данных

Краткие теоретические сведения:

SQL позволяет использовать одни запросы внутри других запросов, то есть вкладывать запросы друг в друга.

Как работает запрос SQL со связанным подзапросом?

- Выбирается строка из таблицы, имя которой указано во внешнем запросе.
- Выполняется подзапрос и полученное значение применяется для анализа этой строки в условии предложения **WHERE** внешнего запроса.
- По результату оценки этого условия принимается решение о включении или не включении строки в состав выходных данных.
- Процедура повторяется для следующей строки таблицы внешнего запроса.

Обратите внимание, что вышеуказанный запрос действителен только в том случае, если в результате выполнения подзапроса, указанного в скобках, возвращается одно значение. Если в результате подзапроса возвращается более одного значения, подзапрос будет недействительным.

Если в таблице STUDENT будет несколько записей со значениями поля SURNAME=‖Петров‖.

В некоторых случаях для гарантии получения единственного значения в результате выполнения подзапроса используется **DISTINCT**. Оператор **IN** так же широко применяется в подзапросах.

При использовании подзапросов во внутреннем запросе можно ссылаться на таблицу, имя которой указано в предложении **FROM** внешнего запроса. В этом случае связанный подзапрос выполняется один раз для каждой строки таблицы основного запроса.

В некоторых СУБД для выполнения этого запроса может потребоваться преобразование значения даты в символьный тип. SU и EX являются псевдонимами (алиасами), которые могут быть использованы в данном запросе вместо настоящих имен.

Исходя из того, что предложение **GROUP BY** позволяет группировать выводимые **SELECT**-запросом записи по некоторому полю, то предложение **HAVING** позволяет осуществлять при выводе фильтрацию этих групп.

Предикат предложения **HAVING** оценивается не для каждой строки результата, а для группы выходных записей, сформированной предложением **GROUP BY** внешнего запроса.

Оператор EXISTS генерирует значение **истина** или **ложь.**

Используя подзапросы в качестве аргумента, этот оператор оценивает результат выполнения подзапроса как true, если этот подзапрос генерирует выходные данные. В противном случае результат подзапроса будет false.

Оператор EXISTS не может принимать значение UNKNOWN(неизвестно).

При использовании связанных вложенных запросов предложение EXISTS анализирует каждую строку таблицы, на которую ссылается внешний запрос. Основной запрос извлекает строки из кандидатов, проверяющих условия. Для каждой строки-кандидата выполняется подзапрос.

Как только подзапрос находит строку, в которой значение столбца MARK удовлетворяет условию, он останавливает выполнение и возвращает true внешнему запросу, который затем анализирует его строку-кандидата.

Нельзя использовать функции агрегирования в подзапросе, указанном в инструкции EXISTS.

Оператор ALL, как правило, эффективно используется с неравенствами. В SQL выражение <> ALL реально означает *не равно ни одному* из результатов подзапроса.

Предположим, известна фамилия студента («Петров»), но неизвестно значение поля STUDENT_ID для него. Чтобы извлечь данные обо всех оценках этого студента можно использовать следующий запрос:

```
SELECT *
FROM EXAM_MARKS WHERE STUDENT_ID = (SELECT STUDENT_ID
FROM STUDENT WHERE SURNAME= _Петров‘);
```

Данные обо всех оценках студентов из Воронежа можно выбрать с помощью следующего запроса:

```
SELECT *
FROM EXAM_MARKS WHERE STUDENT_ID IN (SELECT STUDENT_ID FROM STUDENT
WHERE CITY = _Воронеж‘);
```

Выбрать сведения обо всех предметах обучения, по которым проводился экзамен 12 января 2000 г.

```
SELECT *
FROM SUBJECT AS SU WHERE '12.01.2000' IN
( SELECT EXAM_DATE FROM EXAM_MARKS AS EX
WHERE SU.SUBJ_ID = EX.SUBJ_ID );
```

Необходимо по данным из таблицы EXAM_MARKS определить сумму полученных студентами оценок, сгруппировав значения оценок по датам экзаменов и исключив те дни, когда число студентов, сдававших в течение дня экзамены, было равно1.

```
SELECT EXAM_DATE, SUM(MARK) AS Сумма_оценок FROM EXAM_MARKS AS A
GROUP BY EXAM_DATE HAVING 1=
(SELECT COUNT (MARK) FROM EXAM_MARKS AS B
WHERE A. EXAM_DATE = B. EXAM_DATE);
```

Подзапрос вычисляет количество строк с одной и той же датой, совпадающей с датой, для которой сформирована очередная группа основного запроса.

5. Извлечь из таблицы EXAM_MARKS данные о студентах, получивших хотя бы одну неудовлетворительную оценку.

```
SELECT DISTINCT STUDENT_ID FROM EXAM_MARKS AS A WHERE EXISTS
(SELECT *
FROM EXAM_MARKS AS B WHERE MARK < 3
AND B. STUDENT_ID =A. STUDENT_ID);
```

Выбрать сведения о студентах, проживающих в городе, где расположен университет, в котором они учатся.

```
SELECT *
FROM STUDENT AS S WHERE CITY=ANY (SELECT CITY
FROM UNIVERSITY AS U WHERE U.UNIV_ID=S.UNIV_ID);
```

Выбрать данные о названиях всех университетов с рейтингом более высоким, чем рейтинг любого университета Воронежа:

```
SELECT *
FROM UNIVERSITY WHERE RATING>ALL (SELECT RATING FROM UNIVERSITY
WHERE CITY= 'Воронеж');
```

В этом запросе вместо ALL можно использовать ANY(проанализировать, как в этом случае измениться смысл приведенного запроса):

```
SELECT *
FROM UNIVERSITY
WHERE NOT RATING>ANY (SELECT RATING
FROM UNIVERSITY WHERE CITY='Воронеж');
```

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.3 Работа с объектами базы данных на языке SQL

Лабораторное занятие№17

Выборка и модификация данных с использованием представлений.

Цель: получение практических навыков выборки и модификации данных с использованием представлений

Выполнив работу, Вы будете: ***уметь:***

У2. Использовать язык запросов для программного извлечения сведений из баз данных,

Уо 09.01 понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Уо 09.04 кратко обосновывать и объяснять свои действия (текущие и планируемые);

Уо 09.06 читать, понимать и находить необходимые технические данные и инструкции в руководствах в любом доступном формате.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер,

программное обеспечение: MS Access, MySQL.

Задание

Разработать представления для базы данных на языке SQL. **Краткие теоретические сведения:**

Представления, или *просмотры* (VIEW), представляют собой временные, производные (иначе - виртуальные) таблицы и являются объектами базы данных, информация в которых не хранится постоянно, как в базовых таблицах, а формируется динамически при обращении к ним. Обычные таблицы относятся к базовым, т.е. содержащим данные и постоянно находящимся на устройстве хранения информации. *Представление* не может существовать само по себе, а определяется только в терминах одной или нескольких таблиц. Применение *представлений* позволяет разработчику базы данных обеспечить каждому пользователю или группе пользователей наиболее подходящие способы работы с данными, что решает проблему простоты их использования и безопасности. Содержимое *представлений* выбирается из других таблиц с помощью выполнения запроса, причем при изменении значений в таблицах данные в *представлении* автоматически меняются. *Представление* - это фактически тот же запрос, который выполняется всякий раз при участии в какой-либо команде. Результат выполнения этого запроса в каждый момент времени становится содержанием *представления*. У пользователя создается впечатление, что он работает с настоящей, реально существующей таблицей.

У СУБД есть две возможности *реализации представлений*. Если его определение простое, то система формирует каждую запись *представления* по мере необходимости, постепенно считывая исходные данные из базовых таблиц. В случае сложного определения СУБД приходится сначала выполнить такую операцию, как материализация *представления*, т.е. сохранить информацию, из которой состоит *представление*, во временной таблице. Затем система приступает к выполнению пользовательской команды и формированию ее результатов, после чего временная таблица удаляется.

Представление - это предопределенный запрос, хранящийся в базе данных, который выглядит подобно обычной таблице и не требует для своего хранения дисковой памяти. Для хранения *представления* используется только оперативная память. В отличие от других объектов базы данных *представление* не занимает дисковой памяти за исключением памяти, необходимой для хранения определения самого *представления*.

Создания и изменения *представлений* в стандарте языка и реализации в MySQL совпадают и представлены следующей командой:

{ CREATE| ALTER} VIEW имя_просмотра [(имя_столбца [,...n])]

[WITH ENCRYPTION] AS SELECT_оператор [WITH CHECK OPTION]

Рассмотрим назначение основных параметров.

По умолчанию имена столбцов в *представлении* соответствуют именам столбцов в исходных таблицах. Явное указание имени столбца требуется для вычисляемых столбцов или при объединении нескольких таблиц, имеющих столбцы с одинаковыми именами. Имена столбцов перечисляются через запятую, в соответствии с порядком их следования в *представлении*.

Параметр WITH ENCRYPTION предписывает серверу шифровать SQL-код запроса, что гарантирует невозможность его несанкционированного просмотра и использования. Если при определении *представления* необходимо скрыть имена исходных таблиц и столбцов, а также алгоритм объединения данных, необходимо применить этот аргумент.

Параметр WITH CHECK OPTION предписывает серверу исполнять проверку изменений, производимых через *представление*, на соответствие критериям, определенным в операторе SELECT. Это означает, что не допускается выполнение изменений, которые приведут к исчезновению строки из *представления*. Такое случается, если для *представления* установлен горизонтальный фильтр и изменение данных приводит к несоответствию строки установленным фильтрам. Использование аргумента WITH CHECK OPTION гарантирует, что сделанные изменения будут отображены в *представлении*. Если пользователь пытается выполнить изменения, приводящие к исключению строки из *представления*, при заданном аргументе WITH CHECK OPTION сервер выдаст сообщение об ошибке и все изменения будут отклонены.

Порядок выполнения работы: выполнить запросы по заданным вариантам преподавателя.

Содержание запроса

1. Перевозки из Челябинска во Владивосток.
2. Перевозки в Магнитогорск грузов в количестве более 500 кг.
3. Перевозки во Владивосток, совершенные не позднее 01.05.2017.
4. Абитуриенты, окончившие школу с золотой медалью.
5. Абитуриенты, поступающие на специальность «Электроснабжение» и проживающие в Челябинске и Магнитогорске.
6. Абитуриенты, окончившие школу с золотой медалью и сдавшие экзамен по математике на оценку «5».
7. Должность и тарифная ставка работника Р. Л. Иванова.
8. Работники отдела «Проектирование» с тарифной ставкой от 180 до 250 р./ч.
9. Работники отдела «Проектирование», разряд которых выше 10-го. 1. Поступления товара «Сухое молоко».
10. Товары для организации «Восход» в количестве от 1000 до 5000 кг. 3. Товар «Сухое молоко», поступивший до 15.09.2017.
11. Книги, взятые на абонемент читателем Р.А. Петровым. 2. Книги, выданные в мае 2017 г. в количестве более 5 шт.
12. Книги жанра «Наука», выданные читателю П.Л. Цветкову. 1. Модель автомобиля владельца А.Г. Зайцева.
13. Нарушения, допущенные водителем Г.Д. Беловым осенью 2017г.
14. Нарушения, допущенные владельцами автомобилей модели «Тойота» до 02.08.2017.
15. Данные о старте и финише участника соревнований Р.А. Краснова. 2. Участники команды «Юниор» спортивной организации «Чемпион». 3. Участники команды «Юниор», не вышедшие на старт.
16. Перевозки груза из Омска в Тюмень.
17. Перевозки груза в количестве от 1 до 5 т в Екатеринбург.
18. Перевозки груза, отправленные в Москву не позднее 10.09.2017. 1. Успеваемость студентов по математике.
19. Студенты, получившие по физике оценку «4» или «5».
20. Успеваемость студента А. П. Иванова по математике и физике. 1. Состояние лицевого счета абонента В. Д. Федорова.
21. Должники, имеющие в 2017 г. льготу «Инвалидность».
22. Абоненты, отключенные за неуплату во втором квартале 2017г.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.3 Работа с объектами базы данных на языке SQL

Лабораторное занятие№18

Выполнение динамических запросов с использованием хранимых процедур.

Цель: получение практических навыков выполнения динамических запросов с использованием хранимых процедур.

Выполнив работу, Вы будете: ***уметь:***

У2. Использовать язык запросов для программного извлечения сведений из баз данных,

У2, 37, 38, 39

Уо 09.01 понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Уо 09.04 кратко обосновывать и объяснять свои действия (текущие и планируемые);

Уо 09.06 читать, понимать и находить необходимые технические данные и инструкции в руководствах в любом доступном формате.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение:, MySQL.

Задание

Разработать хранимые процедуры для базы данных на языке SQL.

Краткие теоретические сведения:

Хранимые процедуры представляют собой группы связанных между собой операторов SQL, применение которых делает работу программиста более легкой и гибкой, поскольку выполнить *хранимую процедуру* часто оказывается гораздо проще, чем последовательность отдельных операторов SQL. Хранимые процедуры представляют собой набор команд, состоящий из одного или нескольких операторов SQL или функций и сохраняемый в базе данных в откомпилированном виде. *Выполнение* в базе данных *хранимых процедур* вместо отдельных операторов SQL дает пользователю следующие преимущества:

- необходимые операторы уже содержатся в базе данных;
- все они прошли этап *синтаксического анализа* и находятся в исполняемом формате; перед *выполнением хранимой процедуры* MySQL генерирует для нее *план исполнения*, выполняет ее оптимизацию и компиляцию;
- *хранимые процедуры* поддерживают *модульное программирование*, так как позволяют разбивать большие задачи на самостоятельные, более мелкие и удобные в управлении части;
- *хранимые процедуры* могут вызывать другие *хранимые процедуры* и функции;
- *хранимые процедуры* могут быть вызваны из прикладных программ других типов;
- как правило, *хранимые процедуры* выполняются быстрее, чем последовательность отдельных операторов;
- *хранимые процедуры* проще использовать: они могут состоять из десятков и сотен команд, но для их запуска достаточно указать всего лишь имя нужной *хранимой процедуры*. Это позволяет уменьшить размер запроса, посылаемого от клиента на сервер, а значит, и нагрузку на сеть.

Создание новой и изменение имеющейся хранимой процедуры осуществляется с помощью следующей команды:

<определение_процедуры>::=

{CREATE | ALTER } PROC[EDURE] имя_процедуры [;номер]

[{@имя_параметра тип_данных } [VARYING] [=default][OUTPUT]][,...n]

[WITH { RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION }]

[FOR REPLICATION] AS

sql_оператор [...n]

Для *выполнения хранимой процедуры* используется команда: [[EXEC [UTE] имя_процедуры [;номер]

[[@имя_параметра=]{значение | @имя_переменной} [OUTPUT]|[DEFAULT]][,...n]

Если вызов *хранимой процедуры* не является единственной командой в пакете, то присутствие команды EXECUTE обязательно. Более того, эта команда требуется для вызова процедуры из тела другой процедуры или триггера.

Использование ключевого слова OUTPUT при вызове процедуры разрешается только для параметров, которые были объявлены при создании процедуры с ключевым словом OUTPUT.

Когда же при вызове процедуры для параметра указывается ключевое слово DEFAULT, то

будет использовано значение по умолчанию. Естественно, указанное слово DEFAULT разрешается только для тех параметров, для которых определено значение по умолчанию.

Из синтаксиса команды EXECUTE видно, что имена параметров могут быть опущены при вызове процедуры. Однако в этом случае пользователь должен указывать значения для параметров в том же порядке, в каком они перечислялись при создании процедуры. Присвоить параметру значение по умолчанию, просто пропустив его при перечислении нельзя. Если же требуется опустить параметры, для которых определено значение по умолчанию, достаточно явного указания имен параметров при вызове хранимой процедуры. Более того, таким способом можно перечислять параметры и их значения в произвольном порядке.

Порядок выполнения работы:

Варианты заданий по созданию хранимых процедур с вычисляемым полем:

Вариант	Имя таблицы	Вычисляемое поле
1	Рейсы	Прибыль за рейс, выполненный судном
2	Анкета	Возраст абитуриента на текущую дату
3	Табель	Зарплата работника
4	Отпуск товаров	Доход от продажи товара
5	Выдачи	Просрочено дней читателем
6	Нарушители	Размер штрафа в долларах
7	Финиш	Размер бонуса, определяемый как разница порядковых номеров на старте и финише
8	Доставки	Количество дней доставки груза
9	Сессия	Индивидуальный код студента, представляющий собой сумму шифра студента и шифра дисциплины
10	Платежи	Сумма оплаты с учетом льготы абонента

Варианты заданий по созданию хранимых процедур с групповыми операциями

Вариант	Имя таблицы	Итоговый показатель для расчета
1	Рейсы	Количество рейсов, выполненных каждым судном
2	Специальности	Количество анкет абитуриентов по каждой специальности
3	Работники	Количество отработанных часов каждым работником
4	Товары	Количество отпущенного товара по каждому наименованию
5	Книги	Количество книг, прочитанных каждым читателем
6	Нарушители	Количество нарушителей по каждому виду нарушений
7	Команда	Количество участников в каждой команде
8	Транспорт	Расстояние, пройденное каждым автомобилем
9	Дисциплина	Количество оценок «5» по каждой дисциплине
10	Абоненты	Количество абонентов по каждому виду льготы

Варианты заданий по созданию хранимых процедур с параметрами

Варианты заданий по созданию хранимых процедур с параметрами

Условие процедуры с параметром

Вариант

1. Рейсы, совершенные судном N
2. Абитуриенты, поступающие на специальность N
3. Работники отдела N
4. Организации, которые приобрели товар N
5. Книги, выданные читателю N

Условие процедуры с параметром

1. Владельцы автомобилей, допустившие нарушение N

2. Команда, в состав которой входит участник N
3. Перевозки в пункт назначения N
4. Успеваемость по всем дисциплинам студента N
5. Льготы, которые имеет абонент N

Форма представления результата:
Отчет по выполненной лабораторной работе

Критерии оценки:
Оценка «отлично» ставится, если задание выполнено верно.
Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.
Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.3 Работа с объектами базы данных на языке SQL

Лабораторное занятие№19

Агрегация данных с использованием триггеров.

Цель: получение практических навыков выполнения агрегации данных с использованием триггеров.
Выполнив работу, Вы будете: ***уметь:***
У2. Использовать язык запросов для программного извлечения сведений из баз данных,
Уо 09.01 понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;
Уо 09.04 кратко обосновывать и объяснять свои действия (текущие и планируемые);
Уо 09.06 читать, понимать и находить необходимые технические данные и инструкции в руководствах в любом доступном формате.

Материальное обеспечение:
Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение:, MySQL.

Задание
Модифицировать схему базы данных «Библиотека» таким образом, чтобы таблица subscribers хранила актуальную информацию о дате последнего визита читателя в библиотеку.
Порядок выполнения работы:

1. Выполнить задания в MySQL.

2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика1_Фамилия, с расширением doc, сохраненного в своей папке.

Ход работы.

Краткие теоретические сведения:
Триггер - это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено определенной модификацией данных в заданной таблице или столбце БД.

Триггер - это хранимая в базе данных процедура, автоматически вызываемая СУБД при возникновении соответствующих условий.

Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес логики.

Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан.

Все производимые им модификации данных рассматриваются как одна транзакция. В случае обнаружения ошибки или нарушения целостности данных происходит откат этой транзакции.

Момент запуска триггера определяется с помощью ключевых слов: BEFORE – триггер запускается до выполнения связанного с ним события, например, до добавления записи; AFTER – после события.

Синтаксис

BEFORE – триггер запускается до выполнения связанного с ним события; AFTER – триггер запускается после выполнения связанного с ним события. DELIMITER//

CREATE TRIGGER имя_триггера

BEFORE|AFTER триггерное_событие ON имя_таблицы [FOR EACH {ROW|STATEMENT}]

BEGIN тело_триггера; END;

// Где

FOR EACH ROW – выполняемые триггером действия задаются для каждой строки, охваченной данным событием.

FOR EACH STATEMENT – действия задаются только один раз для каждого события.

Порядок выполнения работы:

Ожидаемый результат

Таблица **subscribers** содержит дополнительное поле, хранящее актуальную информацию о дате последнего визита читателя в библиотеку:

S_id	**S_name**	**S_last_visit**
1.	**Иванов И.И.**	**2015-10-07**
2.	**Петров П.П.**	**Null**
3.	**Сидоров С.С.**	**2014-08-03**
4.	**Сидоров С.С.**	**2015-10-08**

Для решения этой задачи нужно будет выполнить три шага:

1. модифицировать таблицу **subscribers** (добавив туда поле для хранения даты последнего визита читателя);

2. проинициализировать значения последних визитов для всех читателей;

3. создать триггеры для поддержания этой информации в актуальном состоянии.

В MySQL триггеры не активируются каскадными операциями, потому изменения в таблице **subscriptions**, вызванные удалением книг, останутся «незаметными» для триггеров на этой таблице.

Модификация таблицы:

```
ALTER TABLE `subscribers`
ADD COLUMN `s_last_visit` DATE NULL DEFAULT NULL AFTER `s_name`; -- Инициализация данных:
UPDATE `subscribers`
LEFT JOIN (SELECT `sb_subscriber`,
MAX(`sb_start`) AS `last_visit`
FROM `subscriptions`
GROUP BY `sb_subscriber`) AS `prepared_data`
```

ON `s_id` = `sb_subscriber`
SET `s_last_visit` = `last_visit`

2. Поскольку значение даты последнего визита читателя зависит от информации, представленной в таблице **subscriptions** (конкретно — от значения поля **sb_start**), именно на этой таблице нам и придѐтся создавать триггеры. В принципе, во всех трѐх триггерах можно было использовать однотипное решение (**UPDATE** на основе **JOIN**), но для разнообразия в **INSERT**-триггере мы реализуем более простой вариант: проверим, оказалась ли дата добавляемой выдачи больше, чем сохранѐнная дата последнего визита и, если это так, обновим дату последнего визита.

3. В **UPDATE**- и **DELETE**-триггерах такое решение не годится: если в результате этих операций окажется, что читатель ни разу не был в библиотеке, значением даты его последнего визита должен стать **NULL**. Такой результат проще всего достигается с помощью **UPDATE** на основе **JOIN**.

Для операций **UPDATE** и **DELETE** критически важно использовать именно **AFTER**-триггеры, т.к. **BEFORE**-триггеры будут работать со «старыми» данными, что приведѐт к некорректному определению искомой даты.

Также обратите внимание, что в **UPDATE**-триггере мы должны обновлять дату последнего визита для двух потенциально разных читателей, т.к. при внесении изменения выдача может быть «передана» от одного читателя к другому (мы рассмотрим эту ситуацию в процессе проверки работоспособности полученного решения).

4. Удаление старых версий триггеров
-- (удобно в процессе разработки и отладки):
DROP TRIGGER `last_visit_on_subscriptions_ins`;
DROP TRIGGER `last_visit_on_subscriptions_upd`;
DROP TRIGGER `last_visit_on_subscriptions_del`;
-- Переключение разделителя завершения запроса,
-- т.к. сейчас запросом будет создание триггера,
-- внутри которого есть свои, классические запросы:
DELIMITER $$

5. Создание триггера, реагирующего на добавление выдачи книг: CREATE TRIGGER `last_visit_on_subscriptions_ins`
AFTER INSERT ON `subscriptions` FOR EACH ROW BEGIN
IF (SELECT IFNULL(`s_last_visit`, '1970-01-01')
FROM `subscribers`
WHERE `s_id` = NEW.`sb_subscriber`) < NEW.`sb_start` THEN
UPDATE `subscribers`
SET `s_last_visit` = NEW.`sb_start` WHERE `s_id` = NEW.`sb_subscriber`; END IF;
END; $$

6. Создание триггера, реагирующего на обновление выдачи книг: CREATE TRIGGER `last_visit_on_subscriptions_upd`
AFTER UPDATE ON `subscriptions` FOR EACH ROW BEGIN
UPDATE `subscribers`
LEFT JOIN (SELECT `sb_subscriber`, MAX(`sb_start`) AS `last_visit` FROM `subscriptions`
GROUP BY `sb_subscriber`) AS `prepared_data` ON `s_id` = `sb_subscriber`
SET `s_last_visit` = `last_visit`
WHERE `s_id` IN (OLD.`sb_subscriber`, NEW.`sb_subscriber`); END;
$$

7. Создание триггера, реагирующего на удаление выдачи книг: CREATE TRIGGER

`last_visit_on_subscriptions_del`
AFTER DELETE ON `subscriptions` FOR EACH ROW BEGIN
UPDATE `subscribers`
LEFT JOIN (SELECT `sb_subscriber`, MAX(`sb_start`) AS `last_visit` FROM `subscriptions`
GROUP BY `sb_subscriber`) AS `prepared_data` ON `s_id` = `sb_subscriber`
SET `s_last_visit` = `last_visit` WHERE `s_id` = OLD.`sb_subscriber`; END;
$$
-Восстановление разделителя завершения запросов: DELIMITER ;

8. Проверим работоспособность полученного решения. Будем изменять данные в таблице subscriptions и отслеживать изменения данных в таблице subscribers. Для начала добавим выдачу книги читателю с идентификатором 2 (ранее он никогда не был в библиотеке):

INSERT INTO
`subscriptions`
VALUES (200,2, 1,'2019-01-12', '2019-02-12', 'N')

S_id	S_name	S_last_visit
1.	**Иванов И.И.**	**2015-10-07**
2.	**Петров П.П.**	**2014-08-03**
3.	**Сидоров С.С.**	**2014-08-03**
4.	**Сидоров С.С.**	**2015-10-08**

Теперь эмулируем ситуацию «книга ошибочно записана на другого читателя» и изменим идентификатор читателя в только что добавленной выдаче с 2 на 1.

NULL-значение даты последнего визита Петрова П.П. корректно восстановилось: **UPDATE `subscriptions`**
SET `sb_subscriber` = 1 WHERE `sb_id` = 200
. Снова добавим выдачу книги Петрову П.П.: **INSERT INTO `subscriptions`**
VALUES (201, 2,1, '2020-01-12', '2020-02-12', 'N')

Изменим значение даты ранее откорректированной выдачи книги (которую мы переписали с Петрова П.П. на Иванова И.И.):

UPDATE `subscriptions`
SET `sb_start` = '2018-01-12'
WHERE `sb_id` = 200

Удалим эту выдачу

DELETE FROM `subscriptions` WHERE `sb_id` = 200

Удалим единственную выдачу книги Петрову П.П.:
DELETE FROM `subscriptions`
WHERE `sb_id` = 201

Будут ли аннулированы изменения данных, вызванные работой **BEFORE**-триггера, если операция, активировавшая этот триггер, не сможет завершиться успешно?

Изменим вид **INSERT**-триггера с **AFTER** на **BEFORE**: **DROP TRIGGER `last_visit_on_subscriptions_ins`; DELIMITER $$**
CREATE TRIGGER `last_visit_on_subscriptions_ins` BEFORE INSERT
ON `subscriptions` FOR EACH ROW
BEGIN
IF (SELECT IFNULL(`s_last_visit`, '1970-01-01') FROM `subscribers`
WHERE `s_id` = NEW.`sb_subscriber`) < NEW.`sb_start` THEN
UPDATE `subscribers`

```
SET `s_last_visit` = NEW.`sb_start` WHERE `s_id` = NEW.`sb_subscriber`; END IF;
END; $$
DELIMITER ;
```

15. Выполним последовательно добавление двух выдач книг с разными датами, но одинаковыми значениями первичных ключей:

```
INSERT INTO `subscriptions` VALUES (500,
2, 1,
'2020-01-12', '2020-02-12', 'N');
INSERT INTO `subscriptions` VALUES (500,
2, 1,
'2021-01-12', '2021-02-12', 'N');
```

Проверим данные в таблице **subscribers**:

Итак, изменения аннулируются, если операция не может быть завершена успешно. В противном случае значением даты последнего визита Петрова П.П. было бы 2021-01-12.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено

Тема 2.3 Работа с объектами базы данных на языке SQL

Лабораторное занятие№20

Контроль операций с данными с использованием триггеров.

Цель: получение практических навыков использования триггеров.

Выполнив работу, Вы будете:

уметь:

У2. Использовать язык запросов для программного извлечения сведений из баз данных,

Уо 09.01 понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Уо 09.04 кратко обосновывать и объяснять свои действия (текущие и планируемые);

Уо 09.06 читать, понимать и находить необходимые технические данные и инструкции в руководствах в любом доступном формате.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение:MySQL.

Задание

Создать хранимую функцию, автоматизирующую проверку условий задачи, т.е. возвращающую значение 1 (все условия выполнены) или -1, -2, -3 (если хотя бы одно условие нарушено, модуль числа соответствует номеру условия) в зависимости от того, выполняются ли следующие условия.

Порядок выполнения работы:

1. Выполнить задания в MySQL.

2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика1_Фамилия, с расширением doc, сохраненного в своей папке.

DELIMITER $$
CREATE FUNCTION CHECK_SUBSCRIPTION_DATES(sb_start DATE, sb_finish DATE, is_insert INT) RETURNS INT DETERMINISTIC BEGIN
DECLARE result INT DEFAULT 1;
У2, 37, 38, 39 **-- Блокировка выдач книг с датой выдачи в будущем IF (sb_start > CURDATE())**
THEN
SET result = -1; END IF;
-- Блокировка выдач книг с датой возврата в прошлом. IF ((sb_finish < CURDATE()) AND (is_insert = 1)) THEN
SET result = -2; END IF;
-- Блокировка выдач книг с датой возврата меньшей, чем дата выдачи. IF (sb_finish < sb_start)
THEN
SET result = -3; END IF;
RETURN result; END;
$$ DELIMITER ;
SELECT CHECK_SUBSCRIPTION_DATES('2025-01-01', '2026-01-01', 1); SELECT CHECK_SUBSCRIPTION_DATES('2025-01-01', '2026-01-01', 0); SELECT CHECK_SUBSCRIPTION_DATES('2005-01-01', '2006-01-01', 1); SELECT CHECK_SUBSCRIPTION_DATES('2005-01-01', '2006-01-01', 0); SELECT CHECK_SUBSCRIPTION_DATES('2005-01-01', '2004-01-01', 1); SELECT CHECK_SUBSCRIPTION_DATES('2005-01-01', '2004-01-01', 0);

Форма представления результата:
Отчет по выполненной лабораторной работе

Критерии оценки:
Оценка «отлично» ставится, если задание выполнено верно.
Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.
Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.3 Работа с объектами базы данных на языке SQL

Лабораторное занятие№21

Управление транзакциями в триггерах и хранимых процедурах.

Цель: получение практических навыков управления транзакциями в триггерах и хранимых процедурах.
Выполнив работу, Вы будете: ***уметь:***
У2. Использовать язык запросов для программного извлечения сведений из баз данных,
Уо 09.01 понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;
Уо 09.04 кратко обосновывать и объяснять свои действия (текущие и планируемые);
Уо 09.06 читать, понимать и находить необходимые технические данные и инструкции в руководствах в любом доступном формате.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение:MySQL.

Задание1.

Создать на таблице **books** триггер, определяющий уровень изолированности транзакции, в котором сейчас проходит операция вставки, и отменяющий операцию, если уровень изолированности транзакции отличен от **SERIALIZABLE**.

Порядок выполнения работы:

1. Выполнить задания в MySQL.
2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика1_Фамилия, с расширением doc, сохраненного в своей папке.

Ход работы:

1. Если операция вставки данных в таблицу **books** выполняется в транзакции с уровнем изолированности, отличным от **SERIALIZABLE**, триггер отменяет эту операцию и порождает исключительную ситуацию.

2. Для простоты (отсутствия необходимости вручную выполнять вставку) и единообразия (поддержки всеми тремя СУБД) используем **AFTER**-триггеры. Таким образом, представленные ниже решения будут отличаться только логикой определения уровня изолированности транзакций, т.к. в каждой СУБД соответствующий механизм реализован совершенно особенным, несовместимым с другими СУБД, образом.

```
DELIMITER $$
CREATE TRIGGER `books_ins_trans` AFTER INSERT
ON `books` FOR EACH ROW BEGIN
DECLARE isolation_level VARCHAR(50); SET isolation_level =
(
SELECT `VARIABLE_VALUE` FROM `information_schema`. `session_variables`
WHERE `VARIABLE_NAME` = 'tx_isolation'
);
IF (isolation_level != 'SERIALIZABLE') THEN
SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Please, switch your transaction to SERIALIZABLE isolation level and rerun this INSERT again.', MYSQL_ERRNO = 1001;
END IF; END;
$$ DELIMITER ;
```

3. Проверить работоспособность и корректность представленного решения можно выполнением следующего кода: первая попытка выполнить вставку закончится исключительной ситуацией, порождѐнной в триггере, а вторая попытка пройдѐт успешно.

```
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED; INSERT INTO `books`
(`b_name`, `b_year`, `b_quantity`)
VALUES ('И ещё одна книга', 1985,
2);
SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE; INSERT INTO `books` (`b_name`,
`b_year`, `b_quantity`)
VALUES ('И ещё одна книга', 1985,
2);
```

Задание2.

Создать хранимую процедуру, выполняющую подсчѐт количества записей в указанной таблице таким образом, чтобы запрос выполнялся максимально быстро (вне зависимости от

параллельно выполняемых запросов), даже если в итоге он вернѐт не совсем корректные данные.

Порядок выполнения работы:

1. Хранимая процедура должна выполнять подсчѐт записей в указанной таблице в транзакции с уровнем изолированности, обеспечивающим минимальную вероятность ожидания завершения конкурирующих транзакций или отдельных операций в них.

2. Идея решения данной задачи состоит в том, чтобы использовать такой уровень изолированности транзакций, который меньше всего подвержен влиянию со стороны блокировок, порождѐнных другими транзакциями.

3. В MySQL таким уровнем является **READ UNCOMMITTED**, в MS SQL Server — тоже **READ UNCOMMITTED** или **SNAPSHOT** (но **SNAPSHOT** может приводить к дополнительным расходам ресурсов), в Oracle чтение данных всегда происходит в независимом режиме, потому в этой СУБД можно использовать **READ COMMITTED** (тем более, что **READ UNCOMMITTED** в Oracle нет).

```
DELIMITER $$
CREATE PROCEDURE COUNT_ROWS(IN table_name VARCHAR(150), OUT rows_in_table INT)
BEGIN
SET SESSION TRANSACTION
ISOLATION LEVEL READ UNCOMMITTED; SET @count_query =
CONCAT('SELECT COUNT(1) INTO @rows_found FROM ', table_name);
PREPARE count_stmt FROM @count_query; EXECUTE count_stmt;
DEALLOCATE PREPARE count_stmt; SET rows_in_table := @rows_found; END;
$$ DELIMITER ;
```

4. Проверить работоспособность и корректность представленного решения можно выполнением следующего кода.

```
CALL COUNT_ROWS('subscriptions', @rows_in_table); SELECT @rows_in_table;
```

5. Проверить работоспособность и корректность представленного решения можно выполнением следующего кода.

```
DECLARE @res INT;
EXECUTE COUNT_ROWS 'subscriptions', @res OUTPUT; SELECT @res;
```

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.3 Работа с объектами базы данных на языке SQL

Лабораторное занятие№22

Управление пользователями. Запуск и остановка СУБД.

Цель: получение практических навыков выполнения агрегации данных с использованием триггеров.

У2, 37, 38, 39

Выполнив работу, Вы будете: ***уметь:***

У2. Использовать язык запросов для программного извлечения сведений из баз данных,

Уо 09.01 понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Уо 09.04 кратко обосновывать и объяснять свои действия (текущие и планируемые);

Уо 09.06 читать, понимать и находить необходимые технические данные и инструкции в руководствах в любом доступном формате.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение:MySQL.

Задание

Создать нового пользователя СУБД и предоставить ему полный набор прав на базу данных «Библиотека».

Порядок выполнения работы:

1. Выполнить задания в MySQL.
2. Вставьте скриншоты и описание в текстовый процессор MS Word. Представьте выполненную работу в виде файла с названием Практика1_Фамилия, с расширением doc, сохраненного в своей папке.

В MySQL решение данной задачи достигается двумя командами. В строке 1 мы создаѐм пользователя (конструкция **'%'** после **@** означает, что ему разрешѐн доступ с любых хостов), в строке 2 мы выдаѐм ему полные права на базу данных «Библиотека».

CREATE USER 'new_user'@'%' IDENTIFIED BY 'new_password'; GRANT ALL PRIVILEGES ON `library`.* TO 'new_user'@'%';

В MySQL необходимо использовать функцию PASSWORD для хеширования исходного значения пароля

SET PASSWORD FOR 'new_user'@'%' = PASSWORD('some_password')

Очевидно, что можно использовать отдельно части, отвечающие за остановку и запуск СУБД для достижения соответствующего результата.

rem Остановка net stop MySQL56 rem Запуск
net start MySQL56

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено.