**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОСТОВСКОЙ ОБЛАСТИ**

**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ РОСТОВСКОЙ ОБЛАСТИ**

**«РОСТОВСКИЙ-НА-ДОНУ КОЛЛЕДЖ СВЯЗИ И ИНФОРМАТИКИ»**

**ФОНД ОЦЕНОЧНЫХ СРЕДСТВ**

учебной дисциплины

**ОП.08 «Управление ИТ-проектами»**

программы подготовки специалистов среднего звена

для специальности

09.02.11 «Разработка и управление программным обеспечением»

(базовой подготовки)

г. Ростов-на-Дону

2025 г.

|  |  |
| --- | --- |
| **РАССМОТРЕНО**  На заседании цикловой комиссии  «Программирование»  Протокол № 1от 1.09. 2025 года  Председатель ЦК  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.С.Сулавко / | **УТВЕРЖДАЮ**  Заместитель директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / И.В. Подцатова /  « 02» сентября 2025 г. |

Фонд оценочных средств разработан на основе рабочей программы дисциплины ОП.08 «Управление ИТ-проектами», разработанной по специальности 09.02.11 «Разработка и управление программным обеспечением».

Организация-разработчик:

Государственное бюджетное профессиональное образовательное учреждение Ростовской области «Ростовский-на-Дону колледж информатики и связи»

Разработчик:

Кротенко Е.М. – преподаватель государственного бюджетного профессионального образовательного учреждения Ростовской области «Ростовский-на-Дону колледж информатики и связи»

Рецензент:

# **СОДЕРЖАНИЕ**

[1. Паспорт фонда оценочных средств 4](#_Toc52306563)

[2 Результаты освоения учебной дисциплины, подлежащие проверке 6](#_Toc52306564)

[3. Оценка освоения учебной дисциплины 9](#_Toc52306565)

[4.Контрольно-оценочные материалы для итоговой аттестации по учебной дисциплине 23](#_Toc52306567)

## Паспорт комплекта контрольно-оценочных средств

В результате освоения учебной дисциплины «Управление ИТ-проектами» обучающийся должен обладать предусмотренными ФГОС по специальности 09.02.11 «Разработка и управление программным обеспечением» со следующими умениями (У), знаниями (З), общими компетенциями (ОК) и профессиональными компетенциями (ПК):

**уметь:**

У1: проектировать реляционную базу данных;

У2: использовать язык запросов для программного извлечения сведений из баз данных;

У3: проектировать модели базы данных в современных case-средствах;

У4: реализовывать сложные запросы;

У5: настраивать привилегии доступа к данным;

У6: работать с Join

**знать:**

З1: основы теории баз данных;

З2: модели данных;

З3: особенности реляционной модели и проектирования баз данных;

З4: изобразительные средства, используемые в ER- моделировании;

З5: основы реляционной алгебры;

З6: принципы проектирования баз данных;

З7: обеспечение непротиворечивости и целостности данных;

З8: средства проектирования структур баз данных

З9: язык запросов SQL;

З10: принципы нормализации отношений;

З11: способы контроля доступа к данным и управления привилегиями.

**общие компетенции:**

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности.

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по правовой и финансовой грамотности в различных жизненных ситуациях.

ОК 4 Эффективно взаимодействовать и работать в коллективе и команде.

ОК 5 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

| **Дополнительные требования к результатам освоения ППССЗ** | |
| --- | --- |
| **Умения** | **Знания** |
| * работать с современными case-средствами проектирования баз данных; * реализовывать сложные запросы; * настраивать привилегии доступа к данным; * работать с популярными программами, такими как Jira (для разработки ПО), Wrike (для различных сфер, включая IT), Trello и Asana. | * двенадцать правил Кодда для определения концепции реляционной модели; * принципы нормализации отношений; * способы контроля доступа к данным и управления привилегиями. |

Формой аттестации в соответствии с рабочей программой по учебной дисциплине «Управление ИТ-проектами» является дифференцированный зачет.

## 2 Результаты освоения учебной дисциплины, подлежащие проверке

2.1 В результате аттестации по учебной дисциплине «Управление ИТ-проектами» осуществляется комплексная проверка следующих умений и знаний (смотрите таблицу 1):

Таблица 1

|  |  |  |
| --- | --- | --- |
| **Результаты обучения: умения, знания и общие компетенции** | **Показатели оценки результата** | **Форма контроля и оценивания** |
| **Уметь:** | |  |
| У1: проектировать реляционную базу данных | Создана концептуальная модель базы данных.  Создана логическая модель базы данных.  Создана Eмодель базы данных.  Реализовано администрирование базы данных на этапе проектирования. | Текущий контроль в форме:  – практическая проверка (защита практических работ).  Промежуточная аттестация в форме:  – дифференцированный зачет. |
| У2: использовать язык запросов для программного извлечения сведений из баз данных | Сформированы запросы к базе данных |
| У3: проектирование модели базы данных в современных case-средствах | Создана ER-модель базы данных c помощью CASE-средств |
| У4: реализовывать сложные запросы | Сформированы сложные запросы к базе данных |
| У5: настраивать привилегии доступа к данным | Установлены права доступа к данным в базе данных для различных пользователей |
| У6: Работать с Join | Применять Join при формировании запросов |
| **Знать:** | |  |
| З1: основы теории баз данных | Знание основных понятий и определений баз данных / реляционных баз данных | Текущий контроль в форме:  – устный опрос;   * письменная проверочная работа.   Промежуточная аттестация в форме:  – дифференцированный зачет.  . |
| З2: модели данных | Знание: моделей и структур информационных систем; логической и физической моделей данных; типов моделей данных (иерархические, сетевые, реляционные и постреляционные модели организации данных). |

|  |  |  |
| --- | --- | --- |
| **Результаты обучения: умения, знания и общие компетенции** | **Показатели оценки результата** | **Форма контроля и оценивания** |
| З3: особенности реляционной модели и проектирования баз данных | Знание: базовых понятий и классификаций СУБД; типов ключей и отношений; принципов. | Текущий контроль в форме:  – устный опрос;   * письменная проверочная работа; * практическая проверка (защита практических работ).   Промежуточная аттестация в форме:  – дифференцированный зачет. |
| З4: изобразительные средства, используемые в ER- моделировании | Знание: CASE-средств, CASE-систем и CASE-технологий; графическое представление моделей проектирования. |
| З5: основы реляционной алгебры | Знание: основ реляционной алгебры; традиционных и специальных операций над отношениями. |
| З6: принципы проектирования баз данных | Знание: типов ключей и отношений; принципов нормализации таблиц и отношений; этапов проектирования баз данных. |
| З7: обеспечение непротиворечивости и целостности данных | Знание: понятий целостности и непротиворечивости данных; примеры нарушения целостности и непротиворечивости данных; правил и ограничений. |
| З8: средства проектирования структур баз данных | Знание: этапы проектирования баз данных и их реализация |
| З9: язык запросов SQL | Знание: общие характеристики языка структурированных запросов SQL; структуры и типы данных SQL; стандарты языка SQL; команды определения данных и манипулирования данными. |
| З10: принципы нормализации отношений | Знание нормализации таблиц и отношений. |
| З11: способы контроля доступа к данным и управления привилегиями | Знание команд для установления права доступа к данным в базе данных. |

## 3.Оценка освоения учебной дисциплины

**3.1 Формы и методы оценивания**

Предметом оценки служат умения и знания, предусмотренные ФГОС по дисциплине «Управление ИТ-проектами».

Технология оценки знаний и умений по дисциплине соответствует действующему Положению о формах, периодичности и порядке проведения текущего контроля успеваемости и промежуточной аттестации обучающихся в ГБПОУ РО «РКСИ».

Контроль и оценка освоения учебной дисциплины по темам (разделам)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Элемент учебной дисциплины** | **Текущий контроль** | | **Промежуточная аттестация** | |
| **Форма контроля** | **Проверяемые**  **У, З, ОК, ПК** | **Форма контроля** | **Проверяемые**  **У, З, ОК, ПК** |
|  | | | *Дифференцированный зачет* | |
| **Тема 1.1**  Основные понятия баз данных | *– письменная проверочная работа* | *З1*  *ОК1,2,3,4,5,* |  | *З1*  *ОК1-2, 4-5, 9* |
| **Тема 1.2**  Модели данных | *– письменная проверочная работа* | *З1,2*  *ОК1,2,4,5,9* |  | *З1,2*  *ОК1-2, 4-5, 9* |
| **Тема 1.3**  Базовые понятия и классификация СУБД | *– письменная проверочная работа* | *З1,3*  *ОК1,2,4,5,9* |  | *З1,3*  *ОК1-2, 4-5, 9* |
| **Тема 1.4**  Проектирование баз данных | *– письменная проверочная работа;*  *– практическая проверка (защита практических работ №1-4).* | *З1,3,4,6,8,10*  *ОК1,2,4,5,9*  *У1,3*  ПК 11.1-11.6 |  | *З1,3,4,6,8,10*  *ОК1-2, 4-5, 9*  *У1,3*  ПК 11.1-11.6 |
| **Тема 1.5**  Целостность данных | *– письменная проверочная работа* | *З1,3,6,7,8*  *ОК1,2,4,5,9* |  | *З1,3,6,7,8*  *ОК1-2, 4-5, 9* |
| **Тема 1.6**  Реляционная алгебра | *– письменная проверочная работа* | *З1,3,5*  *ОК1,2,4,5,9* |  | *З1,3,5*  *ОК1-2, 4-5, 9* |
| **Тема 1.7**  Создание базы данных и работа с ней | *– письменная проверочная работа;*  *– практическая проверка (защита практических работ №5-6).* | *З1,3,9*  *ОК1,2,4,5,9*  *У2*  ПК 11.1-11.6 |  | *З1,3,9*  *ОК1-2, 4-5, 9*  *У2*  ПК 11.1-11.6 |
| **Тема 1.8**  Структурированный язык запросов SQL.  Реализация запросов к базе данных. | *– письменная проверочная работа;*  *– практическая проверка (защита практических работ №7-17).* | *З1,5,7,9,10,11*  *У2,4,5,6*  *ОК1,2,4,5,9*  ПК 11.1-11.6 |  | *З1,5,7,9,10,11*  *ОК1-2, 4-5, 9*  *У2,4,5,6*  ПК 11.1-11.6 |

**3.2Задания для оценки освоения учебной дисциплины**

**3.2.1 Задания для оценки знаний и умений в ходе проведения текущего контроля**

Задания для проведения *текущей письменной проверочной занятия* (письменных самостоятельных работ по теоретическому материалу по конкретной теме).

**Темы 1.1 – 1.3**

**Проверяемые** результаты обучения

З1: основы теории баз данных;

З2: модели данных;

З3: особенности реляционной модели и проектирования баз данных;

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности

ОК 4 Эффективно взаимодействовать и работать в коллективе и команде

ОК 5 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 9 Пользоваться профессиональной документацией на государственном и иностранном языках.

**Текст задания**

Ответьте на вопросы.

**Вариант 1**

1. Что такое база данных?
2. Что такое объект базы данных? Назовите виды и состояния объектов.
3. Что такое ключевое поле? Дайте определения существующим видам ключей.
4. Что подразумевается под целостностью данных?
5. Что такое нормализация отношений? Расскажите о трёх нормальных формах

**Вариант 2**

1. Что такое таблица базы данных?
2. Что содержит класс объектов базы данных?
3. В чём назначение индексов? Какие существуют типы индексов?
4. В чем назначение администрирования баз данных? (описать задачи и функции администрирования базы данных)
5. Технология обслуживания базы данных: обеспечение работоспособности и восстановление данных. Основные методы и средства защиты данных в базах данных.

**Вариант 3**

1. Что такое атрибут объекта базы данных?
2. Что такое СУБД?
3. Какие существуют типы отношений между таблицами? Дайте определение каждому отношению и опишите алгоритм его построения.
4. Методы и технологии защиты информации в базах данных.
5. Организация непротиворечивости и целостности данных.

**Вариант 4**

1. В чём назначение домена в базах данных?
2. CASE-средства для проектирования баз данных (определение, назначение, примеры).
3. Какие существуют типы моделей данных? Опишите их.
4. Способы контроля доступа к данным и управления привилегиями.
5. Дайте определение и назначение следующим понятиям: форма, отчёты и запросы в СУБД.

**Критерий оценки**

Каждый полный правильный ответ оценивается 1 баллом.

0-2 правильных ответа – «2»

3 правильных ответа – «3»

4 правильных ответа – «4»

5 правильных ответов – «5»

**Тема 1.4**

**Проверяемые** результаты обучения

З1: основы теории баз данных;

З3: особенности реляционной модели и проектирования баз данных;

З4: изобразительные средства, используемые в ER- моделировании;

З6: принципы проектирования баз данных;

З8: средства проектирования структур баз данных

З10: принципы нормализации отношений;

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности

ОК 4 Эффективно взаимодействовать и работать в коллективе и команде

ОК 5 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 9 Пользоваться профессиональной документацией на государственном и иностранном языках.

У1: проектировать реляционную базу данных;

У3: проектировать модели базы данных в современных case-средствах;

ПК 11.1 Осуществлять сбор, обработку и анализ информации для проектирования баз данных.

ПК 11.2 Проектировать базу данных на основе анализа предметной области.

ПК 11.3 Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.

ПК 11.4 Реализовывать базу данных в конкретной системе управления базами данных.

ПК 11.5 Администрировать базы данных.

ПК 11.6 Защищать информацию в базе данных с использованием технологии защиты информации.

**Текст задания по теоретической части**

Ответьте на вопросы.

**Вариант 1**

1. первичный ключ (определение, назначение);
2. суррогатный ключ (определение, назначение);
3. тип отношения «один к одному» (определение, алгоритм построения, как графически обозначается это отношение, какие ключи используются при построении);
4. тип отношения «многие ко многим» (определение, алгоритм построения, как графически обозначается это отношение, какие ключи используются при построении);
5. по каким ключам строятся связи в реляционной БД?

**Вариант 2**

1. внешний ключ (определение, назначение);
2. альтернативный (вторичный) ключ (определение, назначение);
3. тип отношения «один ко многим» (определение, алгоритм построения, как графически обозначается это отношение, какие ключи используются при построении);
4. индекс / индексация (определение, назначение);
5. сколько связей может быть у таблицы? От чего это зависит?

**Критерий оценки**

Каждый полный правильный ответ оценивается 1 баллом.

0-2 правильных ответа – «2»

3 правильных ответа – «3»

4 правильных ответа – «4»

5 правильных ответов – «5»

**Текст задания по практической части**

**Практическое занятие №1 Построение отношений**

Изучив теоретический материал темы 1.4 построить связи между таблицами в предоставленной базе данных.

Для этого:

* построить логические связи, т.е. определить на логическом уровне какие объекты и как будут между собой связаны согласной предметной области;
* определить в каждой таблице поля первичных ключей;
* добавить в необходимые таблицы внешние ключи для связей;
* связать таблицы между собой.

Работа выполняется по вариантам.

Пример раздаточного материала

**БД: Грузоперевозки**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Перевозка** | | Дата отправления  Адрес «откуда»  Адрес «куда»  Дата прибытия  Дата формирования заявки  Название груза  Вес груза  Характеристика груза  Общая стоимость поездки | | |  | | --- | | **Перевозчик** | | Название компании перевозчика  Адрес компании перевозчика  Телефон компании перевозчика  Платёжные реквизиты компании перевозчика |  |  | | --- | | **Автомобиль** | | Марка автомобиля  Модель автомобиля  ГОС номер автомобиля | |
| |  | | --- | | **Заказчик** | | Название заказчика  ФИО контактного лица заказчика  Номер телефона заказчика | | |  | | --- | | **Водитель** | | ФИО водителя | |
| |  | | --- | | **Категория перевозки** | | Категория перевозки (по городу, по городу в выходные дни, по области, по России и т.д.)  Стоимость поездки в зависимости от категории  Единица измерения перевозки (км., часы – зависит от категории перевозки) | | |

**Критерий оценки**

|  |  |  |  |
| --- | --- | --- | --- |
| **«5»** | **«4»** | **«3»** | **«2»** |
| В созданной ER–модель базы данных связи реализованы верно. | В созданной ER–модель базы данных не построена одна связь. | В созданной ER–модель базы данных не построены две связи. | В созданной ER–модель базы данных более двух не построенных связей. |

**Практическое занятие № 2 Нормализация таблиц и отношений**

Изучив теоретический материал темы 1.4 провести нормализацию таблиц, предоставленной базы данных.

Для этого:

* выявить аномалии в таблицах;
* устранить аномалии;
* привести таблицы к 3НФ;
* связать таблицы между собой;
* определить есть ли аномалии в связях;
* устранить аномалии связей.

Работа выполняется по вариантам.

Пример раздаточного материала смотрите выше.

**Критерий оценки**

|  |  |  |  |
| --- | --- | --- | --- |
| **«5»** | **«4»** | **«3»** | **«2»** |
| Созданная ER–модель базы данных не имеет аномалии избыточности данных и связей.  Связи реализованы верно. | Созданная ER–модель базы данных имеет правильные связи, но присутствует незначительная избыточность данных, не устранена одна аномалия. | Созданная ER–модель базы данных имеет правильные связи, но присутствует значительная избыточность данных, не устранено две аномалии. | Созданная ER–модель базы данных не соответствует поставленному заданию, не устранено более двух аномалий. |

**Практическое занятие № 3Создание ER-модели базы данных c помощью CASE-средств**

Изучив теоретический материал темы 1.4 необходимо из представленных данных создать физическую модель базы данных в онлайн сase–средстве dbdesigner.net.

Для этого:

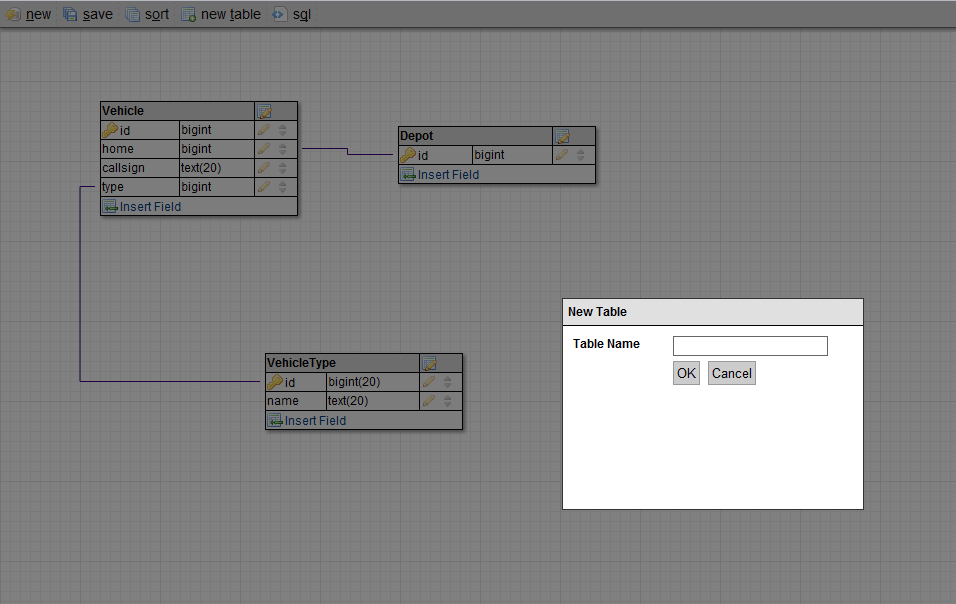
* загрузить браузер и открыть сайт dbdesigner.net;
* создать в сase–средстве базу данных и её таблицы, используя исходные данные из полученного раздаточного материала;
* выявить аномалии избыточности данных в таблицах;
* устранить аномалии;
* установить в таблицах необходимые ключи для реализации связей;
* связать таблицы между собой;
* определить есть ли аномалии в связях;
* если аномалии найдены – устранить их, если нет – показать работу преподавателю.

Работа выполняется по вариантам. Пример раздаточного материала смотрите выше.

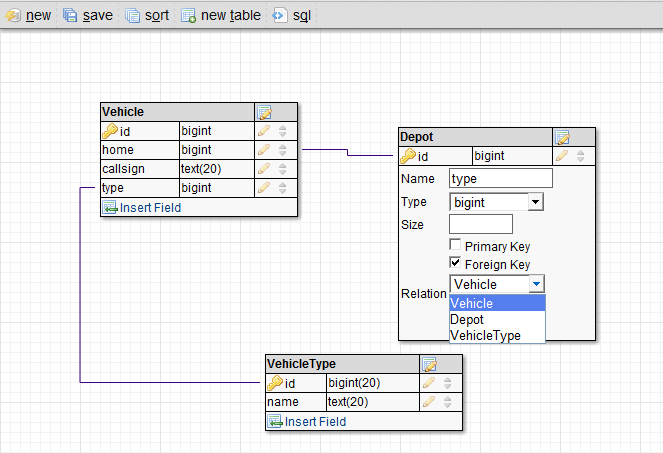
Пошаговая работа:

Создадим схему базы данных, используя генератор схем dbdesigner.net. Для начала нужно залогиниться.

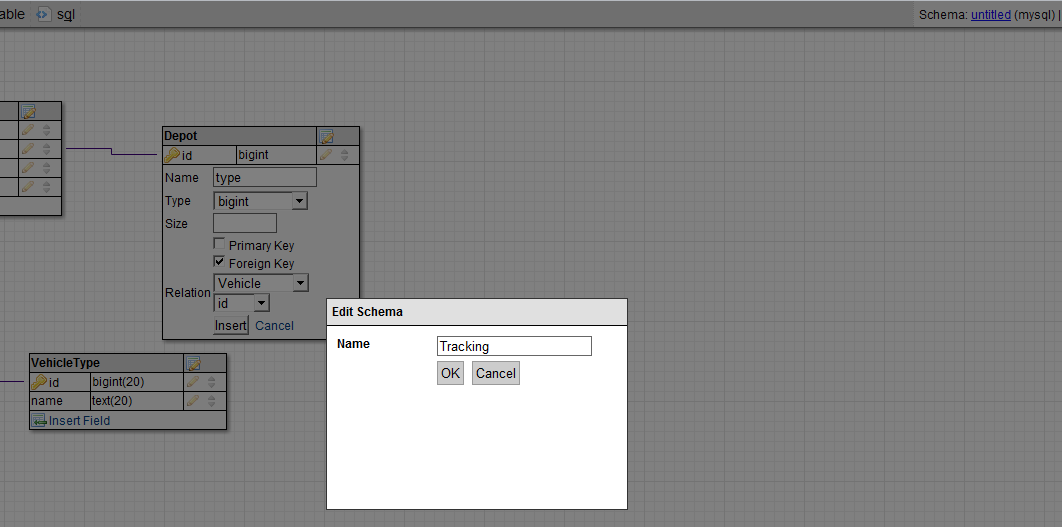
Затем задать имя базе данных. Затем создаем таблицу, нажав на соответствующую кнопку и задав ей имя.



Далее создаются поля в таблице. Поле может быть, как простого типа, так и ссылкой на другую таблицу, если у нас их уже несколько.



Когда все готово задается схеме имя и сохраняем ее.



**Критерий оценки**

|  |  |  |  |
| --- | --- | --- | --- |
| **«5»** | **«4»** | **«3»** | **«2»** |
| Созданная ER–модель базы данных не имеет аномалии избыточности данных и связей.  Связи реализованы верно.  ER–модель отражает исчерпывающую информацию о объектах базы данных. | Созданная ER–модель базы данных имеет правильные связи, но присутствует незначительная избыточность данных, не устранена одна аномалия.  ER–модель отражает исчерпывающую информацию о объектах базы данных. | Созданная ER–модель базы данных имеет правильные связи, но присутствует значительная избыточность данных, не устранено две аномалии.  ER–модель отражает исчерпывающую информацию о объектах базы данных. | Созданная ER–модель базы данных не соответствует поставленному заданию, не устранено более двух аномалий.  ER–модель отражает исчерпывающую информацию о объектах базы данных. |

**Практическое занятие № 4Разработка проекта базы данных**

Изучив теоретический материал темы 1.4 и выполнив практические занятия 1-3 необходимо из представленных данных разработать проект базы данных.

Для этого:

* сформировать концептуальную модель базы данных (из выданного индивидуального задания): определить объекты базы данных и их взаимосвязь;
* преобразовать концептуальную модель в логическую модель: преобразовать объекты в таблицы и определить типы отношений между ними;
* провести нормализацию таблиц;
* провести нормализацию связей;
* установить в таблицах ключи, необходимые для реализации связей;
* построить физические связи.

Работа выполняется по вариантам из раздаточного материала.

Критерии оценки соответствуют критериям из практического занятия №3.

**Темы 1.5 – 1.6**

**Проверяемые** результаты обучения

З1: основы теории баз данных;

З3: особенности реляционной модели и проектирования баз данных;

З6: принципы проектирования баз данных;

З7: обеспечение непротиворечивости и целостности данных;

З8: средства проектирования структур баз данных

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности

ОК 4 Эффективно взаимодействовать и работать в коллективе и команде

ОК 5 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 9 Пользоваться профессиональной документацией на государственном и иностранном языках.

**Текст задания по теоретической части**

Ответьте на вопросы.

**Вариант 1**

1. Дайте определение понятию целостность данных.
2. Опишите правила реализации целостности данных.
3. Что такое транзакции в БД?
4. Что подразумевает под собой ссылочная целостность на внешних ключах?
5. Реляционная алгебра (что это и для чего).
6. Операции реляционной алгебры (перечислить и описать, с разделением на традиционные и нетрадиционные (специальные) операции).

**Вариант 2**

1. Дайте определение понятию непротиворечивости данных;
2. Что подразумевается под ограничением целостности данных?
3. Классификация ограничений целостности данных (перечислить эти ограничения).
4. Что подразумевает под собой ссылочная целостность на триггерах?
5. Перечислите ограничения на операции в реляционной алгебре.
6. Операции над отношениями в реляционной алгебре дополненные Дейтом.

**Критерий оценки**

Каждый полный правильный ответ оценивается 1 баллом.

0-2 правильных ответа – «2»

3 правильных ответа – «3»

4 правильных ответа – «4»

5-6 правильных ответов – «5»

**Тема1.7 и практические занятия 5-6**

**Проверяемые** результаты обучения

З1: основы теории баз данных;

З3: особенности реляционной модели и проектирования баз данных;

З9: язык запросов SQL;

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности

ОК 4 Эффективно взаимодействовать и работать в коллективе и команде

ОК 5 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 9 Пользоваться профессиональной документацией на государственном и иностранном языках.

У2: использовать язык запросов для программного извлечения сведений из баз данных;

ПК 11.1 Осуществлять сбор, обработку и анализ информации для проектирования баз данных.

ПК 11.2 Проектировать базу данных на основе анализа предметной области.

ПК 11.3 Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.

ПК 11.4 Реализовывать базу данных в конкретной системе управления базами данных.

ПК 11.5 Администрировать базы данных.

ПК 11.6 Защищать информацию в базе данных с использованием технологии защиты информации.

**Текст задания по практической части**

**Практическое занятие № 5Создание базы данных в MySQL**

Краткие теоретические сведения

Подключение к серверу базы данных

mysql -u root -p

1) Создание баз данных

**create database <имябазыданных>**

create database Practika;

* 1. Создание базы данных с установкой кодировки (русский язык)

**create database <имябд>CHARACTER SET utf8 COLLATE utf8\_unicode\_ci**

create database st CHARACTER SET utf8 COLLATE utf8\_unicode\_ci

2) Просмотр имеющихся (созданных) баз данных

**show databases;**

show Practika;

1. Выбор базы данных для занятия с ней, т.е. для создания в ней таблиц

**use<имябазыданных>;**

usePractika;

4) Удаление баз данных **drop database <имя базы данных>;**

drop database Practika;

5) Созданиетаблиц

**createtable<имятаблицы>**

**(название\_столбца1 типам\_данных(длина\_поля) notnull,**

**название\_столбца2 типам\_данных(длина\_поля),**

**…**

**primarykey (имя\_поля\_первичного\_ключа)**

**);**

// создаётся таблица forums с первичным ключём id\_forum. Для определения первичного ключа используется оператор PRIMARY KEY

CREATE TABLE forums

(

id\_forum int(5) NOT NULL auto\_increment,

name char(38),

pos float(4,2),

hide int(3),

PRIMARY KEY (id\_forum)

);

// создаётсятаблица Customers спервичнымключём cnum ивнешнимключём snum. Для определения внешнего ключа используется оператор FOREIGN KEY. Оператор REFERENCES указывает имя таблицы, содержащей первичный ключ и имя самого первичного ключа. В данном случае столбцу внешнего ключа snum определили NOTNULL – это будет означать, что при заполнении таблицы Customers нельзя будет оставить не заполненным столбец snum. Если у столбца внешнего ключа не стоит ограничение NOTNULL, то значение внешнего ключа не обязательно и можно будет его оставить пустым при заполнении таблицы данными.

CREATE TABLE Customers

(  
 cnum integer NOT NULL,  
 cname char(10),  
 city char(10),  
 snum integer NOT NULL,

PRIMARY KEY (cnum),

FOREIGN KEY (snum) REFERENCES Salespeople(snum)

);

5.1) Установка значений по умолчанию

Когда вы вставляете строку в таблицу без указания значений в ней для каждого поля, SQL должен иметь *значение по умолчанию* для включения его в определенное поле, или же команда будет отклонена. Наиболее общим значением по умолчанию является **NULL**. Это – значение по умолчанию для любого столбца, которому не было дано ограничение NOT NULL или который имел другое назначение *по умолчанию*.

Значение **DEFAULT** (ПО УМОЛЧАНИЮ) указывается в команде CREATE TABLE тем же способом, что и ограничение столбца, хотя, с технической точки зрения, значение DEFAULT не ограничительного свойства — оно не ограничивает значения, которые вы можете вводить, а просто определяет, что может случиться, если вы не введете любое из них.

Предположим, что вы работаете в офисе Нью-Йорка, и подавляющее большинство ваших продавцов живут в Нью-Йорке. Вы можете указать Нью-Йорк в качестве значения поля city, по умолчанию, для вашей таблицы Продавцов:

CREATE TABLE Salespeople

(

snum integer NOT NULL UNIQUE,

sname char(10) NOT NULL UNIQUE,

city char(10) **DEFAULT** = 'New York',

comm decimal CHECK (comm < 1)

);

Конечно, вводить значение Нью-Йорк в таблицу каждый раз, когда назначается новый продавец, не такая уж необходимость, и можно просто пренебречь им (не вводя его) даже если оно должно иметь некоторое значение. *Значение по умолчанию* такого типа, более предпочтительно, чем, например, длинный конторский номер, указывающий на ваше собственное ведомство, в таблице Заказов.

Длинные числовые значения более расположены к ошибке, поэтому если подавляющее большинство (или все) ваших Заказов должны иметь ваш собственный конторский номер, желательно устанавливать для них значение *по умолчанию*.

5.2) установка уникального значения

**UNIQUE** – это ограничение устанавливает, что все значения данного столбца будут уникальны в пределах таблицы, и создает индекс. Можно применять к столбцам с поддержкой NULL, но так как NULL будет считаться уникальным значением, возможна только одна NULL-запись.

phoneVARCHAR(20) **UNIQUE**NOTNULL

5.3) установка проверки данных

**CHECK** – позволяет установить дополнительную проверку данных для столбца или набора столбцов. Это тоже CONSTRAINT, так как накладывает ограничение.

Примеры:

Ограничение даты рождения сотрудника –

birthday DATE NOT NULL **CHECK** (birthday > ‘1900-01-01’)

Ограничениепола –

pol char (10) NOT NULL **CHECK** (pol IN ('М', 'Ж'))

Ограничениетипатовара –

tip\_tovara VARCHAR (30) **CHECK** (VALUE IN ('Электроника', 'Одежда', 'Еда'));

При вставке данных, не прошедших проверку, будет сообщение об ошибке **Check constraint ‘staff\_chk\_1’ is violated**. Ситуация усложняется, когда установлено несколько **CHECK,** поэтому рекомендуется давать понятное имя.

Воспользуемся полной командой для создания **CHECK** и определим не только ограничение даты рождения, но и допустимые форматы телефона через регулярное выражение.

CREATE TABLE Staff

(

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(255) NOT NULL,

position VARCHAR(30),

birthday DATE NOT NULL,

has\_child BOOLEAN DEFAULT(0) NOT NULL,

phone VARCHAR(20) UNIQUE NOT NULL,

CONSTRAINT staff\_chk\_birthday CHECK (birthday > '1900-01-01'),

CONSTRAINT staff\_chk\_phone CHECK (phone REGEXP '[+]?[0-9]{1,3} ?\\(?[0-9]{3}\\)? ?[0-9]{2}[0-9 -]+[0-9]{2}')

);

Для добавления ограничений используем оператор **CONSTRAINT**, при этом, все названия уникальны, как и имена таблиц. Учитывая, что по умолчанию названия включают в себя и имя таблицы, рекомендуем придерживаться этого правила. Если используется **CONSTRAINT**, мы обязаны дать имя ограничению, которое вводим.

6) Просмотр имеющихся (созданных) таблиц в выбранной базе данных

**show tables;**

7) Просмотр структуры созданной таблицы

**describe<имятаблицы>;**

describeforums;

8) Вставка записей в таблицу

**INSERTINTO<имя\_таблицы>VALUES**

**(значение\_столбца1, значение\_столбца2,…),**

**…**

**(значение\_столбца1, значение\_столбца2,…);**

После оператора VALUES в скобках через запятую перечисляются значения соответствующих полей таблицы в соответствии с их типами.

INSERT INTO authors VALUES

(1, 'Maks', 123, 'maks@mail.ru ', 'www.softtime.ru', 'программист', 0, 0),

(2, 'Igor', 123, ['igor@mail.ru](mailto:'igor@mail.ru)', 'http://www.softtime.ru', 'Программист', 407, 0);

8.1) Вставка результатов запроса

Вы можете также использовать команду INSERT, чтобы получать или выбирать значения из одной таблицы и помещать их в другую, чтобы использовать их вместе с запросом. Чтобы сделать это, вы просто заменяете предложение VALUES (из предыдущего примера) на соответствующий запрос:

INSERT INTO Londonstaff  
SELECT \*  
FROM Salespeople  
WHERE city = 'London';

Здесь выбираются все значения, произведенные запросом — то есть, все строки из таблицы Продавцов со значениями city = "London" — и помещаются в таблицу называемую Londonstaff. Чтобы это работало, таблица Londonstaff должна отвечать следующим условиям:

\* Она должна уже быть создана командой CREATE TABLE.

\* Она должна иметь четыре столбца, которые совпадают с таблицей Продавцов в терминах типа данных; то есть первый, второй, и так далее, столбцы каждой таблицы должны иметь одинаковый тип данных (причем они не обязаны иметь одинаковых имен).

Общее правило то, что вставляемые столбцы таблицы, должны совпадать со столбцами, выводимыми подзапросом, в данном случае, для всей таблицы Продавцов.

Londonstaff — это теперь независимая таблица, которая получила некоторые значения из таблицы Продавцов (Salespeople). Если значения в таблице Продавцов будут вдруг изменены, это никак не отразится на таблице Londonstaff (хотя вы могли бы создать такой эффект с помощью Представления (VIEW), описанного в Главе 20).

Так как или запрос, или команда INSERT могут указывать столбцы по имени, вы можете, если захотите, переместить только выбранные столбцы, а также переупорядочить только те столбцы, которые вы выбрали.

Предположим, например, что вы решили сформировать новую таблицу с именем Daytotals, которая просто будет следить за общим количеством долларов сумм приобретений, упорядоченных на каждый день. Вы можете ввести эти данные независимо от таблицы Заказов, но сначала вы должны заполнить таблицу Daytotals информацией, ранее представленной в таблице Заказов.

Понимая, что таблица Заказов охватывает последний финансовый год, а не только несколько дней, как в нашем примере, вы можете видеть преимущество использования следующего условия INSERT в подсчете и вводе значений

INSERT INTO Daytotals (date, total)  
SELECT odate, SUM (amt)  
FROM Orders  
GROUP BY odate;

Обратите внимание что, как предложено ранее, имена столбцов таблицы Заказов и таблицы Daytotals — не должны быть одинаковыми. Кроме того, если дата приобретения и общее количество — это единственные столбцы в таблице, и они находятся в данном Заказе, их имена могут быть исключены из вывода из-за их очевидной простоты.

8.2) Использование подзапросов с INSERT

INSERT — это самый простой случай. Вы уже видели, как вставлять результаты запроса в таблицу. Вы можете использовать подзапросы внутри любого запроса, который генерирует значения для команды INSERT тем же самым способом, которым вы делали это для других запросов — т.е. внутри предиката или предложения HAVING.

Предположим, что мы имеем таблицу с именем SJpeople, столбцы которой совпадают со столбцами нашей таблицы Продавцов. Вы уже видели, как заполнять таблицу подобно этой, заказчиками в городе, например, в San Jose:

INSERT INTO SJpeople  
SELECT \*  
FROM Salespeople  
WHERE city = 'San Jose';

Теперь мы можем использовать подзапрос, чтобы добавить к таблице SJpeople всех продавцов, которые имеют заказчиков в San Jose, независимо от того, находятся ли там продавцы или нет:

INSERT INTO Sjpeople  
SELECT \*  
FROM Salespeople  
WHERE snum = ANY (SELECT snum  
 FROM Customers  
 WHERE city = 'San Jose');

***Оба запроса в этой команде функционируют так же, как если бы они не являлись частью выражения INSERT. Подзапрос находит все строки для заказчиков в San Jose и формирует набор значений snum. Внешний запрос выбирает строки из таблицы Salespeople, где эти значения snum найдены. В этом примере, строки для продавцов Rifkin и Serres, которые назначены заказчикам в San Jose — Liu и Cisneros, будут вставлены в таблицу SJpeople.***

**Индивидуальное задание**

Задание: создайте базу данных из раздаточного материала

Предусмотреть требования и ограничения к таблицам.

Продавцы:

* все поля кроме «comm» обязательные;
* нельзя ввести комиссионные ниже 0.10 и выше 0.20.

Заказчики:

* все поля кроме «snum» обязательные;
* по умолчанию в столбце «city» должно быть значение «SanJose».

Заказы:

* все поля кроме «snum» и «cnum» обязательные.

**Практическое занятие № 6Модификация базы данных и структуры таблиц в MySQL**

Краткие теоретические сведения

Изменение структуры таблицы

**ALTERTABLE<имя\_таблицы><значение>;**

|  |  |
| --- | --- |
| **Синтаксис** | **Описание команды** |
| **ALTERTABLE**<имя\_таблицы>  **ADD** <имя столбца> <тип столбца>  **FIRST** <имя столбца, перед которым добавлять новый столбец>  **AFTER**<имя столбца, после которого добавлять новый столбец> | Добавление нового столбца  **FIRST** добавляет новый столбец перед указанным столбцом.  **AFTER** добавляет новый столбец после указанного столбца.  Если место добавления не указано, по умолчанию столбец добавляется в конец таблицы. |
| **ALTER TABLE** <имя\_таблицы>  **ADD INDEX** [index\_name] (index\_col\_name,...) | Добавлениеиндекса index\_name длястолбца index\_col\_name. Если имя индекса index\_name не указывается, ему присваивается имя совпадающее с именем столбца index\_col\_name. |
| **ALTERTABLE**<имя\_таблицы>  **ADD PRIMARY KEY** <имя столбца, которому устанавливается первичный ключ>  **ADDFOREIGNKEY**<имя столбца, которому устанавливается первичный ключ>**REFERENCES**<имя таблицы, на которую ссылаетесь, т.е. главную><имя столбца, на который ссылаетесь> | Делает указанный столбец первичным ключом таблицы.  Делает указанный столбец внешним ключом таблицы. |
| **ALTERTABLE**<имя\_таблицы>  **CHANGE**<старое\_имя\_столбца>  <новое \_имя\_столбца> <тип\_данных\_столбца>  **Пример:**  **ALTERTABLE**<имя\_таблицы>  **CHANGE** old\_col\_name new\_col\_name type | Изменение имени столбца с обязательным указанием типа данных для нового столбца!!!  Пример:  Изменение столбца с именем old\_col\_name на столбец с именем new\_col\_name и типом type. |
| **ALTERTABLE**<имя\_таблицы>  **DROP**<имя\_столбца>  **Пример:**  **ALTERTABLE**<имя\_таблицы>  **DROP** col\_name | Удаление столбца  Пример:  Удаление столбца с именем col\_name. |
| **ALTER TABLE** <имя\_таблицы>  **DROP PRIMARY KEY** | Удаление первичного ключа таблицы. |
| **ALTERTABLE**<имя\_таблицы>  **DROP INDEX**<имя\_индексного\_столбца>  **Пример:**  **ALTERTABLE**<имя\_таблицы>  **DROP INDEX** index\_name | Удаление индексного столбца  Пример:  Удаление индекса index\_name. |

1) Добавление нового столбца

Добавим в таблицу forums новый столбец test, поместив его после столбца name.

ALTER TABLE forums ADD test int(10) AFTER name;

2) Удаление столбца

Теперь удалим столбец new\_test:

ALTER TABLE forums DROP new\_test;

3) Удаление таблицы

drop table <имя таблицы>;

4) Переименование таблицы

ALTER TABLE t1 RENAME t2;

5) Переименование столбца и изменение его типа данных

ALTER TABLE forums CHANGE test test int(15);

6) Удаление строк из таблиц

DELETE FROM Salespeople;

Обычно, вам нужно удалить только некоторые определенные строки из таблицы. Чтобы определить какие строки будут удалены, вы используете предикат, так же как вы это делали для запросов. Например, чтобы удалить продавца Axelrod из таблицы, вы можете ввести

DELETE FROM Salespeople  
WHERE snum = 1003;

Конечно, вы можете также использовать DELETE с предикатом, который бы выбирал группу строк, как показано в этом примере:

DELETE FROM Salespeople  
WHERE city = 'London';

7) Изменение содержимого таблицы

**UPDATE <имя\_таблицы> SET <имя\_столбца> = <новое\_значение>**

**WHERE <имя\_столбца> = <старое\_значение>;**

Следующий пример изменит название города в записях пользователей с "*Ryazan*" на "*Рязань*":

UPDATE users SET city='Рязань' WHERE city='Ryazan';

7.1) Команда UPDATE для многих столбцов

Однако вы не должны ограничивать себя модифицированием единственного столбца с помощью команды UPDATE. Предложение SET может назначать любое число столбцов, отделяемых запятыми. Все указанные назначения могут быть сделаны для любой табличной строки, но только для одной в каждый момент времени. Предположим, что продавец Motika ушел на пенсию, и мы хотим переназначить его номер новому продавцу:

UPDATE Salespeople  
SET sname = 'Gibson', city = 'Boston', comm = .10  
WHERE snum = 1004;

Эта команда передаст новому продавцу Gibson, всех текущих заказчиков бывшего продавца Motika и Заказы, в том виде, в котором они были скомпонованы для Motika с помощью поля snum.

**Вы не можете, однако, модифицировать сразу много таблиц в одной команде, частично потому, что вы не можете использовать префиксы таблицы со столбцами, измененными предложением SET. Другими словами, вы не можете сказать — "SET Salespeople.sname = Gibson" в команде UPDATE, вы можете сказать только так — "SET sname = Gibson".**

**Задание:**

1. Законспектировать все команды по данной практической работе.
2. Используя команды SQL выполните следующее задание:

* переименовать таблицу Продавцы, к текущему имени добавьте цифру 1;
* в таблице Продавцы переименуйте столбцы «sname» и «city» – в конец текущего имени добавьте цифру 1;
* в таблицу Продавцы добавить столбец **Naprav после столбца Sname** (направление деятельности продавца). Столбцу указать тип данных и размер поля;
* внести **данные** в столбец **Naprav** (например: продукты, молочная продукция, алкогольная продукция, табачные изделия, колбасная продукция, мясная продукция, соки-воды);
* в таблицу Заказы добавить столбец **Dostavka** (стоимость доставки, числовое значение) **после столбца Amt**. Столбцу указать тип данных и размер поля;
* внести **данные** в столбец **Dostavka** (цифры, означающие стоимость доставки);
* в таблицу Заказы добавить столбец **Oplata** (вид оплаты заказа) **после столбца Dostavka**. Столбцу указать тип данных и размер поля;
* внести **данные** в столбец **Oplata** (например: наличный, безналичный);
* в таблицу Заказчики добавить столбец **pol**(пол заказчика) **после столбца cname**. Столбцу указать тип данных и размер поля;
* внести **данные** в столбец **pol** («м» или «ж»).
* в таблице Заказчики изменить London на Paris;
* в таблице Продавцы изменить всю комиссию, которая **больше** 0,12 на 0,16;
* удалите строку из таблицы Заказы с номером заказа 3011.

1. Покажите работу преподавателю.
2. Вернитепрежниеназваниятаблиц и столбцов.

**Проверочное задание на проверку полученных знаний**

По вопросам 1-5 написать полный формат команды. По вопросам 6-10 написать результат занятия, после выполнения SQLкоманды.

**Вариант 1**

1. Создание баз данных
2. Просмотр имеющихся (созданных) таблиц в выбранной базе данных
3. Вставка записей в таблицу
4. Добавление нового столбца
5. Удаление таблицы
6. SELECT \* FROM Student
7. ALTER TABLE Marka RENAME MarkaTS
8. DELETE FROM cust WHERE city='London'
9. ALTERTABLE gruppa DROP PRIMARY KEY
10. UPDATE ts SET gos='o333oo' WHERE id\_ts=15

**Вариант 2**

1. Просмотр имеющихся (созданных) баз данных
2. Просмотр структуры созданной таблицы
3. Добавление первичного ключа к уже имеющейся таблице
4. Удаление столбца
5. Переименование таблицы
6. ALTER TABLE forums CHANGE Gruppa GruppaIT char (20)
7. UPDATE Zakaz SET oplata='Карта' WHERE oplata='Безнал'
8. ALTER TABLE forums DROP new\_test
9. ALTERTABLEpredmet ADDFOREIGNKEY (id\_pr) REFERENCESprepod(id\_pr)
10. describe forums

**Вариант 3**

1. Создание таблиц
2. Удаление баз данных
3. Добавление внешнего ключа к уже имеющейся таблице
4. Переименование столбца и изменение его типа данных
5. Изменение содержимого таблицы
6. ALTER TABLE forums ADD test int(10) AFTER name
7. drop table Student
8. ALTERTABLEprepod ADD PRIMARY KEY id\_pr
9. UPDATE bilet SET cena=1500 WHERE cena=1000
10. ALTER TABLE forums CHANGE test test int(15)

**Критерий оценки**

Каждый полный правильный ответ оценивается в 0,5 баллов.

0-4 правильных ответа – «2»

5-6 правильных ответа – «3»

7-8 правильных ответа – «4»

9-10 правильных ответов – «5»

**Тема 1.8 и практические занятия 7-8**

**Проверяемые** результаты обучения

З1: основы теории баз данных;

З5: основы реляционной алгебры;

З7: обеспечение непротиворечивости и целостности данных;

З9: язык запросов SQL;

З10: принципы нормализации отношений;

З11: способы контроля доступа к данным и управления привилегиями.

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности

ОК 4 Эффективно взаимодействовать и работать в коллективе и команде

ОК 5 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 9 Пользоваться профессиональной документацией на государственном и иностранном языках.

У2: использовать язык запросов для программного извлечения сведений из баз данных;

У4: реализовывать сложные запросы;

У5: настраивать привилегии доступа к данным;

У6: работать с Join

ПК 11.1 Осуществлять сбор, обработку и анализ информации для проектирования баз данных.

ПК 11.2 Проектировать базу данных на основе анализа предметной области.

ПК 11.3 Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.

ПК 11.4 Реализовывать базу данных в конкретной системе управления базами данных.

ПК 11.5 Администрировать базы данных.

ПК 11.6 Защищать информацию в базе данных с использованием технологии защиты информации.

**Текст задания по практической части**

**Практическое занятие № 7 Простая выборка данных c использованием условных операторов**

**Краткие теоретические сведения**

Выборка данных осуществляется с помощью оператора **SELECT.** Формат команды приведён ниже.

**SELECT** [**DISTINCT**] список\_выбираемых\_полей

**FROM**список\_таблиц

[**WHERE**условие\_выборки ]

[**GROUPBY**условие\_группировки ]

[**ORDERBY**условие\_упорядочивания ]

Пример выборки столбцов snum, sname, city, comm из таблицы Salespeople.

SELECT snum, sname, city, comm FROM Salespeople;

Для удаления избыточных данных команда **SELECT**использует оператор **DISTINCT**

SELECT DISTINCT snum FROM Orders;

В выборке данных могут применяться условные операторы. Например:

SELECT \* FROM Customers WHERE rating = 100;

Операторы Буля и как они работают:

1. **AND** (логическое И) берет два Буля (в форме A AND B) как аргументы и оценивает их по отношению к истине, верны ли они оба;
2. **OR** (логическое ИЛИ) берет два Буля (в форме A OR B) как аргументы и оценивает на правильность, верен ли один из них;
3. **NOT**(**!**) (логическое НЕ) берет одиночный Булев (в форме NOT A) как аргументы и заменяет его значение с неверного на верное или верное на неверное.

SELECT \* FROMCustomersWHEREcity = 'SanJose' ANDrating> 200;

SELECT \* FROMCustomersWHEREcity = 'SanJose' ORrating> 200;

SELECT \* FROM Customers WHERE city = 'San Jose' OR NOT rating > 200;

Уловныеоператоры IN, BETWEEN (REGEXP), LIKE:

1. оператор **IN** определяет набор значений, в который данное значение может или не может быть включено. В соответствии с нашей учебной базой данных, на которой вы обучаетесь по настоящее время, если вы хотите найти всех продавцов, которые размещены в Barcelona или в London, вы должны использовать следующий запрос :

SELECT \* FROM SalespeopleWHERE city = 'Barcelona' OR city = 'London';

Имеется и более простой способ получить ту же информацию:

SELECT \* FROM Salespeople WHERE city IN ('Barcelona', 'London');

1. оператор **BETWEEN** похож на оператор IN. В отличие от определения по номерам из набора, как это делает IN, BETWEEN определяет диапазон, значения которого должны уменьшаться, что делает предикат верным. Вы должны ввести ключевое слово BETWEEN с начальным значением, ключевое **AND** и конечное значение. В отличие от IN, BETWEEN чувствителен к порядку, и первое значение в предложении должно быть первым в алфавитном или числовом порядке. Обратите внимание, что, в отличие от английского языка, SQL не говорит "*значение находится между (BETWEEN) значением и значением*”, а просто "*значение BETWEEN значение AND значение*”. Следующий пример будет извлекать из таблицы Продавцов всех продавцов с комиссионными между .10 и .12:

SELECT \* FROM Salespeople WHERE comm BETWEEN .10 AND .12;

Обратите внимание, что данный оператор **не включает** в результат вывода нижнюю границу символьного выражения (таков уж его алгоритм занятия).

В SQLимеется оператор **REGEXP**, который может сделать то, чего не делает **BETWEEN.**

1. оператор **LIKE** применим только к полям типа CHAR или VARCHAR, с которыми он используется, чтобы находить подстроки. То есть, он ищет поле символа, чтобы видеть, совпадает ли с условием часть его строки. В качестве условия он использует групповые символы (wildcards) — специальные символы которые могут соответствовать чему-нибудь. Имеются два типа групповых символов, используемых с LIKE:

* символ подчеркивания (**\_**) замещает любой одиночный символ. Например, **'b\_t'** будет соответствовать словам **'bat'** или **'bit'**, но не будет соответствовать 'brat';
* знак процента (**%**) замещает последовательность любого числа символов (включая символы нуля). Например **'%p%t'** будет соответствовать словам **'put', 'posit',** или **'opt',** но не **'spite**'. Давайте найдем всех заказчиков, чьи имена начинаются с G :

SELECT \* FROM Customers WHERE cname LIKE 'G%';

**Индивидуальное задание:**

1. Изучите основные операторы и перепишите их назначение в тетрадь.
2. Используя команды SQL выполните задания.

**Часть 1:**

1. выведите города Заказчиков без повторений;
2. выведите имеющиеся рейтинги Заказчиков без повторений;
3. выведите имеющиеся комиссионные Продавцов без повторений;
4. выведите все поля из таблицы Продавцы при условии, что продавец из Лондона;
5. выведите все поля из таблицы Продавцы при условии, что комиссия продавца больше 0.12%;
6. выведите все поля из таблицы Заказчики при условии, что заказчик из Рима;
7. выведите все поля из таблицы Заказчики при условии, что рейтинг заказчика 100 и 300;
8. выведите все поля из таблицы Заказы при условии, что заказ меньше 1000;
9. выведите все поля из таблицы Заказчики при условии, что заказчик из Рима с рейтингом 200;
10. выведите все поля из таблицы Заказчики при условии, что заказчик или из Берлина, или с рейтингом 300.

**Часть 2:**

1. выберите все поля из таблицы Заказчики, в которых заказчиков обслуживали продавцы 1001, 1002 и 1007;
2. выберите все поля из таблицы Заказы, в которых заказы оформляли заказчики 2001, 2004 и 2008;
3. выберите из таблицы Продавцы все записи, где номер продавца от 1001 до 1005;
4. выберите из таблицы Продавцы все записи, у которых коэффициент продавца от 10% до 13%;
5. выберите из таблицы Заказчики все записи где имя заказчика от буквы А до G;
6. выберите из таблицы Заказчики все записи где заказчики проживают в городах от буквы А до L;
7. выберите из таблицы Заказы все заказы, которые были оформлены с 10.03.2021 по 10.05.2021;
8. выведите все данные о заказчиках, имя которых начинается с буквы 'С';
9. выведите все данные о заказчиках, имя которых начинается с буквы 'G';
10. выведите все данные о заказчиках, имя которых начинается с буквы 'G' и заканчивается 's'.

**Практическое занятие № 8 Агрегатные функции, форматирование вывода запросов в SQL**

**Краткие теоретические сведения**

Агрегатные (групповые) функции производят одиночное значение для всей группы таблицы. Имеется список этих функций:

|  |  |
| --- | --- |
| Функция | Назначение |
| **COUNT** | считает число значений в данном столбце, или число строк в таблице.  Когда она считает значения столбца, она используется с DISTINCT, чтобы производить счет чисел различных значений в данном поле  Чтобы подсчитать общее число строк в таблице, используйте функцию COUNT созвездочкойвместоимениполя  SELECT COUNT(\*) FROM Customers |
| **SUM** | производит арифметическую сумму всех выбранных значений данного поля |
| **AVG** | производит усреднение всех выбранных значений данного поля |
| **MIN** | производит наименьшее из всех выбранных значений данного поля |
| **MAX** | производит наибольшее из всех выбранных значений данного поля |
| **ROUND** | Функция округления результата занятия агрегатной функции  **round (агрегатная функция(столбец\_таблицы), n)**  n – число, до которого нужно округлять дробную часть |

Все эти функции возвращают единственное значение. При этом функции **COUNT, MIN** и **MAX** применимы к любым типам данных, в то время как **SUM** и **AVG** используются только для числовых полей. Разница между функцией **COUNT(\*)** и **COUNT(<имя поля>)** состоит в том, что вторая при подсчете не учитывает NULL-значения.

SELECT SUM(amt) FROM Orders;

**Переименование столбцов в запросе**

В результате выборки данных с использованием агрегатных функций, столбец с результатом имеет имя такое же как и сама агрегатная функция, которая была использщована в этом запросе. Функция AS позволяет переименовать столбец в заголовке которого было название с агрегатной функцией. Иными словами AS даёт псевдоним столбцу. Предыдущий пример изменим с использованием оператора AS.

SELECT SUM(amt) AS Суммаприобретений FROM Orders;

**Аргумент ALL**

Агрегатные функции могут также (в большинстве реализаций) использовать аргумент **ALL**, который помещается перед именем поля, подобно DISTINCT, но означает противоположное — включать дубликаты.

Следующая команда подсчитает (**COUNT**) число не NULL значений в поле rating в таблице Заказчиков (включая повторения):

SELECT COUNT(ALL rating) FROM Customers;

**Агрегаты, построенные на скалярном выражении**

Возможно использовать агрегатные функции с аргументами, которые состоят из скалярных выражений, включающих одно или более полей. Предположим, что таблица Заказов имеет еще один столбец, который хранит предыдущий неуплаченный баланс (поле blnc) для каждого заказчика. Вы должны найти этот текущий баланс, добавлением суммы приобретений к предыдущему балансу. Вы можете найти наибольший неуплаченный баланс следующим образом:

SELECT MAX(blnc + amt) FROM Orders;

**Скалярное выражение на основе выбираемых полей**

SQL позволяет вам помещать скалярные выражения и константы среди выбранных полей. Эти выражения могут дополнять или замещать поля в предложениях SELECT, и могут включать в себя одно или более выбранных полей.

SELECT snum, sname, city, comm \* 100 FROM Salespeople;

=============== SQL Execution Log ====  
| SELECT snum, sname, city, comm \* 100   
| FROM Salespeople;   
| =======================================  
| snum sname city   
| ------ --------- ----------- ---------   
| 1001 Peel London 12.000000   
| 1002 Serres San Jose 13.000000   
| 1004 Motika London 11.000000   
| 1007 Rifkin Barcelona 15.000000   
| 1003 Axelrod New York 10.000000   
 =========================================

Рисунок – Помещение выражения в вашем запросе

**Столбцы вывода**

Столбцы вывода – это столбцы данных созданные запросом способом, иным чем просто извлечение их из таблицы. Вы создаете их всякий раз, когда вы используете агрегатные функции, константы, или выражения в предложении SELECT запроса.

**Помещение текста в вашем выводе запроса**

Вы можете вставлять константы в предложение SELECT запроса, включая и текст. Однако символьные константы, в отличие от числовых констант, не могут использоваться в выражениях.

Тем ни менее, возможность вставлять текст в вывод ваших запросов очень удобная штука.

SELECT snum, sname, city, ' % ', comm \* 100 FROM Salespeople;

=============== SQL Execution Log =======  
| SELECT snum, sname, city, '%', comm \* 100   
| FROM Salespeople;   
| =====================================  
| snum sname city   
| ------ -------- ----------- ---- ---------   
| 1001 Peel London % 12.000000   
| 1002 Serres San Jose % 13.000000   
| 1004 Motika London % 11.000000   
| 1007 Rifkin Barcelona % 15.000000   
| 1003 Axelrod New York % 10.000000   
 ==========================================

Рисунок – Вставка символов в ваш вывод

SELECT ' For ', odate, ', there are ',COUNT(DISTINCT onum), 'orders.'

FROM Orders GROUP BY odate;

=============== SQL Execution Log ==============  
| SELECT 'For', odate, ', ' there are ' ,   
| COUNT (DISTINCT onum), ' orders '   
| FROM Orders   
| GROUP BY odate;   
| =============================================== |  
| odate   
| ------ ---------- --------- ------ -------  
| For 10/03/1990 , there are 5 orders.  
| For 10/04/1990 , there are 2 orders.   
| For 10/05/1990 , there are 1 orders.  
| For 10/06/1990 , there are 2 orders.   
=================================================

Рисунок – Комбинация текста, значений поля, и агрегатов

**Индивидуальное задание**

**Теоретическая часть:**

1. Перечислите и расшифруйте агрегатные функции.
2. Чем отличаются функции COUNT(\*) и COUNT(<имя поля>)?
3. Какая функция позволяет переименовать столбец-результат?
4. В чём назначение аргумента **ALL**?
5. Как можно использовать скалярные выражения в запросах?
6. Как строиться скалярное выражение на основе выбираемых полей?
7. Что подразумевают собой столбцы выводы?
8. Каким образом происходит помещение текста в вашем выводе запроса?

**Практическая часть:**

С помощью команд выполните следующие задания:

1. Покажите сколько продавцов в таблице Продавцы? Столбец с результатом назовите «Количество продавцов»;
2. Покажите из скольки городов продавцы? Столбец с результатом назовите «Города продавцов»;
3. Выведите сколько строк в таблице Заказчики? Столбец с результатом назовите «Количество строк»;
4. Выведите общий рейтинг заказчиков? Столбец с результатом назовите «Общий рейтинг заказчиков»;
5. Выведите на какую сумму сделаны все заказы? Столбец с результатом назовите «Сумма заказов»;
6. Выведите средний рейтинг заказчиков? Столбец с результатом назовите «Средний рейтинг заказчиков»;
7. Выведите среднюю стоимость заказов? Столбец с результатом назовите «Средняя стоимость заказа»;
8. Выведите минимальный рейтинг заказчиков? Столбец с результатом назовите «Минимальный рейтинг заказчика»;
9. Выведите максимальную комиссию продавцов? Столбец с результатом назовите «Максимальная комиссия продавца»;
10. Посчитайте общую стоимость заказа с учётом доставки. Столбец с результатом назовите «Общая стоимость заказа»;
11. Выведите из таблицы Заказчики поля: Имя, Город и Рейтинг, уменьшенный в 50 раз;
12. Выведите из таблицы Продавцы поля: Имя, Город, синус комиссионных, увеличенный в 10 раз;
13. Выведите строковую константу, поясняющую поле 'cnum' из таблицы Заказчики, Номер заказчика, Имя заказчика и Город;
14. Выведите строковые константы, поясняющие поля 'snum' и 'comm' из таблицы Продавцы, Номер, Имя продавца, Город, Комиссионные.

**Проверочное задание на проверку полученных знаний**

Напишите SQLкоманду на поставленный вопрос.

**Вариант 1**

1 В какие даты были сделаны заказы

2 Из какого города заказчик *Liu*

3 Кто из заказчиков проживает в *Berlin* и *Rome*

4 Кто из заказчиков с номерами 2002, 2004, 2006 имеет имя на «С»

5 Сколько заказов сделали каждый из продавцов с номерами 1001 и 1007

**Вариант 2**

1 В каких городах проживают заказчики

2 Из какого города продавец *Peel*

3 Кто из продавцов проживает в *London*с комиссией больше 11%

4 Какие заказы были сделаны летом на сумму больше 1000 руб.

5 Какая средняя стоимость заказов за весну

**Вариант 3**

1 Какие вариации комиссий продавцов используются в таблице «Продавцы»

2 Какие заказы сделаны после 10 марта 2021 года

3 Какие заказы оформлены 10 марта и 10 июня

4 Кто из заказчиков проживает в городах на «R»и «S»

5 Какой минимальный рейтинг заказчиков из города *SanJose*

**Вариант 4**

1 Какие вариации рейтингов заказчиков используются в таблице «Заказчики»

2 Кто из заказчиков проживает в *Paris*

3 Кто из заказчиков проживает в *SanJose*с рейтингом больше 150

4 Кто из продавцов проживает в *London(Paris)* с именем на «М» (Мотика)

5 Покажите самый дорогой заказ, сделанный 10 марта

**Вариант 5**

1 В каких городах проживают заказчики

2 Кто из продавцов имеет комиссионные ниже 15%

3 Какие заказы были оформлены 10 апреля с суммой больше 1000 руб.

4 Кто из заказчиков проживает в *Rome* с именем на «Р» (Перейра)

5 Какой общий рейтинг заказчиков из *Rome*

**Критерий оценки**

Каждый полный правильный ответ оценивается в1 балл.

0-2 правильных ответа – «2»

3 правильных ответа – «3»

4 правильных ответа – «4»

5 правильных ответов – «5»

**Тема 1.8 и Практическое занятие 9**

**Проверяемые** результаты обучения по знаниям **З1,5,7,9,10,11** по общим компетенциям **ОК1,2,4,5,9**, по умениям **У2,4,5,6** и по профессиональным компетенциям **ПК 11.1-11.6**

З1: основы теории баз данных;

З5: основы реляционной алгебры;

З7: обеспечение непротиворечивости и целостности данных;

З9: язык запросов SQL;

З10: принципы нормализации отношений;

З11: способы контроля доступа к данным и управления привилегиями.

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности

ОК 4 Эффективно взаимодействовать и работать в коллективе и команде

ОК 5 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 9 Пользоваться профессиональной документацией на государственном и иностранном языках.

У2: использовать язык запросов для программного извлечения сведений из баз данных;

У4: реализовывать сложные запросы;

У5: настраивать привилегии доступа к данным;

У6: работать с Join

ПК 11.1 Осуществлять сбор, обработку и анализ информации для проектирования баз данных.

ПК 11.2 Проектировать базу данных на основе анализа предметной области.

ПК 11.3 Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.

ПК 11.4 Реализовывать базу данных в конкретной системе управления базами данных.

ПК 11.5 Администрировать базы данных.

ПК 11.6 Защищать информацию в базе данных с использованием технологии защиты информации.

**Практическое занятие № 9 Группировка и упорядочивание данных в SQL**

**Часть 1 – Группировка**

Предложение **GROUP BY** используется для группирования результата одного или нескольких столбцов.

Например, вы хотите найти наибольшую сумму приобретений, полученную каждым продавцом. Вы можете сделать раздельный запрос для каждого из них, выбрав MAX (amt) из таблицы Заказов для каждого значения поля snum. GROUP BY, однако, позволит Вам поместить их все в одну команду:

SELECT snum, MAX(amt) FROM Orders GROUP BY snum;

Вывод для этого запроса показывается ниже

=============== SQLExecutionLog ==============  
| SELECTsnum, MAX (amt) |  
| FROMOrders |  
| GROUPBYsnum; |  
| =============================================== |  
| snum |  
| ------ -------- |  
| 1001 767.19 |  
| 1002 1713.23 |  
| 1003 75.75 |  
| 1014 1309.95 |  
| 1007 1098.16 |  
 =================================================

Вы можете также использовать GROUP BY с многочисленными полями. Совершенствуя вышеупомянутый пример далее, предположим, что вы хотите увидеть наибольшую сумму приобретений получаемую каждым продавцом каждый день. Чтобы сделать это, вы должны сгруппировать таблицу Заказов по продавцам и датам, и применить функцию MAX к каждой такой группе, подобно этому:

SELECTsnum, odate, MAX (amt)

FROM Orders GROUP BY snum, odate;

Вывод для этого запроса показывается ниже

=============== SQLExecutionLog ==============  
| SELECTsnum, odate, MAX (amt) |  
| FROMOrders |  
| GROUPBYsnum, odate; |  
| =============================================== |  
| snumodate |  
| ------ ---------- -------- |  
| 1001 10/03/1990 767.19 |  
| 1001 10/05/1990 4723.00 |  
| 1001 10/06/1990 9891.88 |  
| 1002 10/03/1990 5160.45 |  
| 1002 10/04/1990 75.75 |  
| 1002 10/06/1990 1309.95 |  
| 1003 10/04/1990 1713.23 |  
| 1014 10/03/1990 1900.10 |  
| 1007 10/03/1990 1098.16 |  
 =================================================

**HAVING**

Если предложение **WHERE** определяет предикат **для фильтрации строк**, то предложение **HAVING** применяется **после группировки** для определения аналогичного предиката, фильтрующего группы по значениям агрегатных функций. Это предложение необходимо для проверки значений, которые получены с помощью **агрегатной функции** не из отдельных строк источника записей, определенного в предложении **FROM**, а из **групп таких строк**. Поэтому такая проверка не может содержаться в предложении **WHERE**.

Заметим, что в предложении HAVING нельзя использовать **псевдоним**, используемый для именования значений **агрегатной функции**. Дело в том, что **предложение SELECT**, формирующее выходной набор запроса, выполняется предпоследним перед **предложением ORDER BY**.

Предположим, что вы хотели бы увидеть только максимальную сумму приобретений, значение которой выше $3000.00. Но исходя из выше написанного Вы не сможете сделать что-нибудь подобно следующему:

SELECT snum, odate, MAX(amt)  
FROM Oreders  
WHERE MAX(amt) > 3000.00  
GROUP BY snum, odate;

Это будет отклонением от строгой интерпретации ANSI. Чтобы увидеть максимальную стоимость приобретений свыше $3000.00, вы можете использовать предложение HAVING. Предложение HAVING определяет критерии, используемые чтобы удалять определенные группы из вывода, точно также как предложение WHERE делает это для индивидуальных строк. Правильной командой будет следующая:

SELECT snum, odate, MAX(amt)  
FROM Orders  
GROUP BY snum, odate  
HAVING MAX(amt) > 3000.00;

Вывод для этого запроса показывается ниже.

=============== SQL Execution Log =======  
| SELECT snum, odate, MAX (amt)   
| FROM Orders   
| GROUP BY snum, odate   
| HAVING MAX (amt) > 3000.00;   
| =====================================  
| snum odate   
| ------ ---------- --------   
| 1001 10/05/1990 4723.00   
| 1001 10/06/1990 9891.88   
| 1002 10/03/1990 5160.45   
 =======================================

Аргументы в предложении HAVING должны иметь одно значение на группу вывода. Следующая команда будет запрещена:

SELECT snum, MAX(amt)  
FROM Orders  
GROUP BY snum  
HAVING odate = 10/03/1988;

Поле оdate не может быть вызвано предложением HAVING, потому что оно может иметь (и действительно имеет) больше чем одно значение на группу вывода. Чтобы избегать такой ситуации, предложение HAVING должно ссылаться только на агрегаты и поля выбранные GROUP BY. Имеется правильный способ сделать вышеупомянутый запрос:

SELECT snum, MAX(amt)  
FROM Orders  
WHERE odate = 10/03/1990  
GROUP BY snum;

Следующий пример показывает наибольшие Заказы для Serres и Rifkin:

SELECT snum, MAX(amt)  
FROM Orders  
GROUP BY snum  
HAVING snum IN (1002,1007);

=============== SQL Execution Log =======  
| SELECT snum, MAX (amt)   
| FROM Orders   
| GROUP BY snum   
| HAVING snum IN (1002, 1007);   
| =====================================  
| snum   
| ------ --------   
| 1002 5160.45   
| 1007 1098.16   
 ======================================

Рисунок – Использование HAVING с полями GROUPBY

**Часть 2 – Упорядочивание**

Таблицы — это неупорядоченные наборы данных, и данные, которые выходят из них, не обязательно появляются в какой-то определенной последовательности. SQL использует команду **ORDER BY**, чтобы позволять вам упорядочивать ваш вывод. Эта команда упорядочивает вывод запроса согласно значениям, в том или ином количестве выбранных столбцов. Многочисленные столбцы упорядочиваются один внутри другого, также как с GROUP BY, и вы можете определять возрастание (ASC) или убывание (DESC) для каждого столбца. По умолчанию установлено — возрастание.

SELECT \* FROM Orders ORDER BY cnum DESC;

=============== SQLExecutionLog ==============  
| SELECT \*   
| FROMOrders  
| ORDERBYcnumDESC;   
| =============================================== |  
| onumamtodatecnumsnum  
| ------ -------- ---------- ----- -----   
| 3001 18.69 10/03/1990 2008 1007   
| 3006 1098.16 10/03/1990 2008 1007   
| 3002 1900.10 10/03/1990 2007 1004   
| 3008 4723.00 10/05/1990 2006 1001   
| 3011 9891.88 10/06/1990 2006 1001   
| 3007 75.75 10/04/1990 2004 1002   
| 3010 1309.95 10/06/1990 2004 1002   
| 3005 5160.45 10/03/1990 2003 1002   
| 3009 1713.23 10/04/1990 2002 1003   
| 3003 767.19 10/03/1990 2001 1001   
 =================================================

**Упорядочение по нескольким столбцам**

Мы можем также упорядочивать таблицу по другому столбцу.

SELECT \* FROM Orders ORDER BY cnum DESC, amt DESC;

=============== SQL Execution Log ===========  
| SELECT \*   
| FROM Orders   
| ORDER BY cnum DESC, amt DESC;   
| ===========================================  
| onum amt odate cnum snum   
| ------ -------- ---------- ----- -----   
| 3006 1098.16 10/03/1990 2008 1007   
| 3001 18.69 10/03/1990 2008 1007   
| 3002 1900.10 10/03/1990 2007 1004   
| 3011 9891.88 10/06/1990 2006 1001   
| 3008 4723.00 10/05/1990 2006 1001   
| 3010 1309.95 10/06/1990 2004 1002   
| 3007 75.75 10/04/1990 2004 1002   
| 3005 5160.45 10/03/1990 2003 1002   
| 3009 1713.23 10/04/1990 2002 1003   
| 3003 767.19 10/03/1990 2001 1001   
 =============================================

Вы можете использовать ORDER BY таким же способом сразу с любым числом столбцов. Обратите внимание что, во всех случаях, столбцы, которые упорядочиваются должны быть указаны в выборе SELECT.

**Упорядочение агрегатных групп**

ORDER BY может использоваться с GROUP BY для упорядочения групп.

SELECT snum, odate, MAX (amt)

FROM Orders GROUP BY snum, odate

ORDER BY snum;

=============== SQL Execution Log ==============  
| SELECT snum, odate, MAX (amt)   
| FROM Orders   
| GROUP BY snum, odate   
| ORDER BY snum ;   
| =============================================  
| snum odate amt   
| ----- ---------- --------   
| 1001 10/06/1990 767.19   
| 1001 10/05/1990 4723.00   
| 1001 10/05/1990 9891.88   
| 1002 10/06/1990 5160.45   
| 1002 10/04/1990 75.75   
| 1002 10/03/1990 1309.95   
| 1003 10/04/1990 1713.23   
| 1004 10/03/1990 1900.10   
| 1007 10/03/1990 1098.16   
 =================================================

**Упорядочение вывода по номеру столбца**

Вместо имен столбцов вы можете использовать их порядковые номера для указания поля, используемого в упорядочении вывода. Эти номера могут ссылаться не на порядок столбцов в таблице, а на их порядок в выводе.

SELECT sname, comm FROM Salespeople GROUP BY 2 DESC;

=============== SQL Execution Log ==========  
| (SELECT sname, comm   
| FROM Salespeople   
| ORDER BY 2 DESC;   
| =======================================  
| sname comm   
| -------- --------   
| Peel 0.17   
| Serres 0.13   
| Rifkin 0.15   
 ==========================================

Одна из основных целей этой возможности ORDER BY — дать вам возможность использовать GROUP BY со столбцами вывода также, как и со столбцами таблицы. Столбцы, производимые агрегатной функцией, константы или выражения в предложении SELECT запроса, абсолютно пригодны для использования с GROUP BY, если они ссылаются к ним с помощью номера.

SELECT snum, COUNT (DISTINCT onum)

FROM Orders GROUP BY snum

ORDER BY 2 DESC;

=============== SQL Execution Log ============  
| SELECT snum, odate, MAX (amt) |  
| FROM Orders |  
| GROUP BY snum |  
| ORDER BY 2 DESC; |  
| ============================================  
| snum |  
| ----- ---------- |  
| 1001 3 |  
| 1002 3 |  
| 1007 2 |  
| 1003 1 |  
| 1004 1 |  
 ==============================================

В этом случае, вы должны использовать номер столбца, так как столбец вывода не имеет имени; и вы не должны использовать саму агрегатную функцию.

Упорядочение с помощью оператора NULL

Если имеются пустые значения (NULL) в поле, которое вы используете для упорядочивания вашего вывода, они могут или следовать, или предшествовать каждому другому значению в поле.

**Индивидуальное задание**

**Теоретическая часть:**

1. Каково назначение GROUP BY?
2. Для чего необходимо предложение HAVING?
3. С помощью какого оператора производится упорядочивание данных?
4. Что означают операторы ASC и DESC?
5. Каким образом можно производить упорядочивание по нескольким столбцам?
6. Каким образом можно производить упорядочивание групп?
7. Как производить упорядочение вывода по номеру столбца?

**Практическая часть 1:**

1. Напишите запрос, который сосчитал бы число различных, отличных от NULL значений поля city в таблице Заказчиков.
2. Напишите запрос, который выбрал бы наименьшую сумму заказа для каждого заказчика.
3. Напишите запрос, который выбрал бы наименьшую сумму заказа каждого продавца за каждый день.
4. Напишите запрос, который показывает наименьшую стоимость заказа для заказчиков Cisneros, Grass и Clemens.
5. Напишите запрос, который выведет имена заказчиков в алфавитном порядке, но только тех, чьи имена начинаются с буквы G.
6. Напишите запрос, который выведет высшийрейтинг в каждом городе.
7. Напишите запрос, который сосчитал бы число Продавцов, регистрирующих свои Заказы, за каждый день (если продавец имел более одного Заказа в данный день, он должен учитываться только один раз).
8. Напишите запрос, который сосчитал бы число Заказов за каждый день для каждого Продавца.

**Практическая часть2:**

1. Выведите названия всех заказчиков относительно рейтинга, т. е. по каждому рейтингу должны быть выведены те заказчики, которые к нему относятся.
2. Выведите заказчиков, которых обслуживает конкретный продавец, т. е. по каждому продавцу должен быть выведен список заказчиков, с которыми он работает.
3. Вывести из таблицы Продавцы поля Номер, Имя, Город упорядоченные по убыванию по полю 'sname'.
4. Вывести из таблицы Заказчики поля Имя, Город, Рейтинг упорядоченные по возрастанию по полям 'rating' и 'cname'.
5. Вывести из таблицы Продавцы поля Имя, Город, сгруппированные по полю Город, и упорядоченные по полю Имя.
6. Вывести из таблицы Заказчики поля Имя, Город, Рейтинг, сгруппированные по 2-му столбцу.
7. Вывести из таблицы Заказчики поля Город и количество рейтингов каждого города упорядоченных по возрастанию.

**Проверочное задание на проверку полученных знаний**

Напишите SQLкоманду на поставленный вопрос.

**1 Вариант**

1. Показать самые большие комиссионные продавцов в каждом городе. Результат упорядочите по названию города.
2. Какой общий рейтинг заказчиков в каждом городе? Результат упорядочите по названию города.
3. Покажите сумму заказов за каждую дату. Результат упорядочите по дате.
4. Какая средняя стоимость заказа по каждой дате? Результат упорядочите по дате.
5. Покажите среднюю стоимость заказов каждого продавца за каждый день

**2 Вариант**

1. Сколько заказчиков в каждом городе? Результат упорядочите по названию города.
2. Сколько заказчиков обслужил каждый продавец? Результат упорядочите по номеру продавца.
3. Покажите самый дорогой заказ за каждую дату. Результат упорядочите по дате.
4. Покажите самую дорогую покупку, оформленную каждый продавцом.
5. Покажите количество заказчиков за каждый день

**Критерий оценки**

Каждый полный правильный ответ оценивается в1 балл.

0-2 правильных ответа – «2»

3 правильных ответа – «3»

4 правильных ответа – «4»

5 правильных ответов – «5»

**Тема 1.8 и практические занятия 10-11**

**Проверяемые** результаты обучения

З1: основы теории баз данных;

З5: основы реляционной алгебры;

З7: обеспечение непротиворечивости и целостности данных;

З9: язык запросов SQL;

З10: принципы нормализации отношений;

З11: способы контроля доступа к данным и управления привилегиями.

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности

ОК 4 Эффективно взаимодействовать и работать в коллективе и команде

ОК 5 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 9 Пользоваться профессиональной документацией на государственном и иностранном языках.

У2: использовать язык запросов для программного извлечения сведений из баз данных;

У4: реализовывать сложные запросы;

У5: настраивать привилегии доступа к данным;

У6: работать с Join

ПК 11.1 Осуществлять сбор, обработку и анализ информации для проектирования баз данных.

ПК 11.2 Проектировать базу данных на основе анализа предметной области.

ПК 11.3 Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.

ПК 11.4 Реализовывать базу данных в конкретной системе управления базами данных.

ПК 11.5 Администрировать базы данных.

ПК 11.6 Защищать информацию в базе данных с использованием технологии защиты информации.

**Практическое занятие № 10 Запрос из нескольких таблиц в MySQL**

**Теоретические сведения**

Одна из наиболее важных особенностей запросов SQL — это их способность определять связи между различными таблицами и выводить информацию из них в терминах этих связей, всю внутри одной команды.

Этот вид операции называется — *объединением*, которое является одним из видов операций в реляционных базах данных.

**Имена таблиц и столбцов**

Полное имя столбца таблицы фактически состоит из имени таблицы, сопровождаемого точкой и затем именем столбца <имя таблицы>.<имя столбца>. Имеются несколько примеров имен

Salespeople.snum  
Salespeople.city  
Orders.odate

**Создание объединения**

Предположим, что вы хотите поставить в соответствии вашему продавцу ваших заказчиков в городе, в котором они живут, поэтому вы увидите все комбинации продавцов и заказчиков для этого города. Вы будете должны брать каждого продавца и искать в таблице Заказчиков всех заказчиков того же самого города. Вымоглибысделатьэто, введяследующуюкоманду:

SELECT Customers.cname, Salespeople.sname, Salespeople.city  
FROM Salespeople, Customers  
WHERE Salespeople.city = Customers.city;

=============== SQL Execution Log ============  
| SELECT Customers.cname, Salespeople.sname,   
| Salespeople.city   
| FROM Salespeople, Customers   
| WHERE Salespeople.city = Customers.city   
| ============================================= |  
| cname cname city   
| ------- -------- ----   
| Hoffman Peel London   
| Clemens Peel London   
| Hoffman Motika London   
| Clemens Motika London   
| Liu Serres San Jose   
| Cisneros Serres San Jose   
 ===============================================

**Объединение таблиц через справочную целостность**

Эта особенность часто используется просто для эксплуатации связей, встроенных в базу данных. В предыдущем примере, мы установили связь между двумя таблицами в объединении. Это прекрасно. Но эти таблицы уже были соединены через поле snum. Эта связь называется *состоянием справочной целостности*. Используя объединение, можно извлекать данные в терминах этой связи. Например, чтобы показать имена всех заказчиков, соответствующих продавцам, которые их обслуживают, мы будем использовать такой запрос:

SELECT Customers.cname, Salespeople.sname  
FROM Customers, Salespeople  
WHERE Salespeople.snum = Customers.snum;

=============== SQL Execution Log ============  
| SELECT Customers.cname, Salespeople.sname,   
| FROM Salespeople, Customers   
| WHERE Salespeople.snum = Customers.snum   
| =============================================   
| cname sname   
| ------- -------- |  
| Hoffman Peel   
| Giovanni Axelrod   
| Liu Serres   
| Grass Serres   
| Clemens Peel   
| Cisneros Rifkin   
| Pereira Motika   
 =============================================

**Объединения таблиц по равенству значений в столбцах и другие виды объединений**

Объединения, которые используют предикаты, основанные на равенствах, называются — *объединениями по равенству*. Все наши примеры относились именно к этой категории, потому что все условия в предложениях WHERE базировались на математических выражениях, использующих знак равно (=). Строки **'city='London'** и **'Salespeople.snum=Orders.snum'** — примеры таких типов равенств, найденных в предикатах.

*Объединения по равенству —* это вероятно наиболее общий вид объединения, но имеются и другие. Вы можете, фактически, использовать любой из реляционных операторов в объединении. Здесьпоказанпримердругоговидаобъединения:

SELECT sname, cname  
FROM Salespeople, Customers  
WHERE sname < cname AND rating < 200;

=============== SQL Execution Log ===========  
| SELECT sname, cname   
| FROM Salespeople, Customers   
| WHERE sname < cname   
| AND rating < 200;   
| =========================================  
| sname cname   
| -------- -------   
| Peel Pereira   
| Motika Pereira   
| Axelrod Hoffman   
| Axelrod Clemens   
| Axelrod Pereira   
 ==========================================

## Объединение более двух таблиц

Вы можете также создавать запросы, объединяющие более двух таблиц. Предположим, что мы хотим найти все Заказы заказчиков не находящихся в тех городах где находятся их продавцы. Для этого необходимо связать все три наши типовые таблицы:

SELECTonum, cname, Orders.cnum, Orders.snum  
FROMSalespeople, Customers,Orders  
WHERECustomers.city<>Salespeople.city  
ANDOrders.cnum = Customers.cnum  
ANDOrders.snum = Salespeople.snum;

=============== SQL Execution Log ============  
| SELECT onum, cname, Orders.cnum, Orders.snum   
| FROM Salespeople, Customers, Orders   
| WHERE Customers.city <> Salespeople.city   
| AND Orders.cnum = Customers.cnum   
| AND Orders.snum = Salespeople.snum;   
| ==============================================  
| onum cname cnum snum   
| ------ ------- ----- -----   
| 3001 Cisneros 2008 1007   
| 3002 Pereira 2007 1004   
| 3006 Cisneros 2008 1007   
| 3009 Giovanni 2002 1003   
| 3007 Grass 2004 1002   
| 3010 Grass 2004 1002   
 =================================================

**Индивидуальное задание**

**Теоретическая часть:**

1. Что подразумевает операция объединения таблиц?
2. Из чего формируется полное имя столбца?
3. Каким образом создаётся объединение?
4. Что означает объединение по равенству?

**Практическая часть**

1. Вывести фамилию продавца, фамилию покупателя и сумму покупки между 700 и 2000,.
2. Вывести фамилию продавцаи сумму его покупки без комиссии.
3. Вывести фамилию продавцов и соответствующих им фамилии покупателей.
4. Вывести фамилию продавцов и фамилию покупателей, которые из одного города (поле Название города так же вывести).
5. Вывести поле дата, где комиссия продавца меньше 0.12 (поле Комиссия так же вывести).
6. Вывести поле Сумма заказа покупателя, у которого рейтинг больше 100 (поле Рейтинг так же вывести).
7. Вывести имя заказчика, у которого заказ совершен 10.03.2021 (поле Дата так же вывести).
8. Вывести имя покупателя, сумму его заказа и комиссионные продавца за 10.04.2021 (поле Дата так же вывести).
9. Вывести имя и рейтинг заказчиков, у которых сумма заказа больше 3000 и покупка совершена 10.03.2021 (поля Сумма заказа и Дата так же вывести).
10. Вывести номера заказов, продавцы и заказчики которых живут в Сан Джосе (поля ФИО заказчика, ФИО продавца и Название города так же вывести).

Покажите работу преподавателю.

**Практическое занятие № 11 Вложенные запросы**

**Краткие теоретические сведения**

Рассмотрим работу подзапроса. Предположим, что мы знаем имя продавца: Motika, но не знаем значение его поля snum, и хотим извлечь все заказы из таблицы Заказы. Запроси его результат смотрите ниже.

SELECT \*

FROMOrders

WHEREsnum = (SELECTsnum

FROMSalespeople

WHEREsname = 'Motika');

=============== SQLExecutionLog ==============  
| SELECT \*   
| FROMOrders  
| WHEREsnum =   
| (SELECTsnum  
| FROMSalespeople  
| WHEREsname = 'Motika');   
|=================================================  
| onumamtodatecnumsnum  
| ----- ------- ---------- ----- -----   
| 3002 1900.10 10/03/1990 2007 1004   
 =================================================

При использовании подзапросов в предикатах, основанных на уравнениях или неравенствах, вы должны убедиться, что использовали подзапрос, который будет выдавать одну и только одну строку вывода.

Если вы используете подзапрос, который не выводит никаких значений вообще, команда выполнится, но основной запрос не выведет никаких значений.

**DISTINCT с подзапросами**

Вы можете, в некоторых случаях, использовать DISTINCT, чтобы вынудить подзапрос генерировать одиночное значение. Предположим, что мы хотим найти все заказы кредитований для тех продавцов которые обслуживают Hoffmanа (cnum = 2001).

SELECT \*

FROM Orders

WHERE snum = ( SELECT DISTINCT snum

FROM Orders

WHERE cnum = 2001 );

=============== SQL Execution Log ==============  
| SELECT \* |  
| FROM Orders |  
| WHERE snum = |  
| (SELECT DISTINCT snum |  
| FROM Orders |  
| Where cnum = 2001); |  
| =============================================== |  
| onum amt odate cnum snum |  
| ----- --------- --------- ------ ------- |  
| 3003 767.19 10/03/1990 2001 1001 |  
| 3008 4723.00 10/05/1990 2006 1001 |  
| 3011 9891.88 10/06/1990 2006 1001 |  
 =================================================

Подзапрос установил, что значение поля snum совпало с Hoffman — 1001, и затем основной запрос выделил все порядки с этим значением snum из таблицы Порядков (не разбирая, относятся они к Hoffman или нет). Так как каждый заказчик назначен одному и только этому продавцу, мы знаем, что каждая строка в таблице Порядков сданным значением cnum должна иметь такое же значение snum. Однако, так как там может быть любое число таких строк, подзапрос мог бы вывести много (хотя и идентичных) значений snum для данного поля cnum. Аргумент DISTINCT предотвращает это.

**Предикаты с подзапросами являются необратимыми**

Вы должны обратить внимание, что предикаты, включающие подзапросы, используют выражение <скалярная форма><оператор><подзапрос>, а не <подзапрос><оператор><скалярное выражение> или, <подзапрос><оператор><подзапрос>.

Другими словами, вы не должны записывать предыдущий пример так:

SELECT \*

FROM Orders

WHERE ( SELECT DISTINCT snum

FROM Orders

WHEREcnum = 2001 ) = snum;

**Индивидуальное задание**

Теоретическая часть:

1. В чём преимущество подзапросов?
2. Какого назначение оператора DISTINCT в подзапросах?

**Практическая часть:**

1. Вывести из таблицы Заказы все значения полей у заказчика Hoffman.
2. Вывести заказы продавца из New York.
3. Вывести заказчиков, которые оформляли заказу продавца Motika.
4. Вывести продавцов, которые работали с заказчиком Liu.
5. Вывести покупки заказчика с рейтингом 100 и городом Rome.

Покажите работу преподавателю.

**Проверочное задание на проверку полученных знаний**

Напишите SQLкоманду на поставленный вопрос.

**1 Вариант**

1. Покажите все заказы (номер заказа и его стоимость), которые оформил продавец *Peеl*(имя Peel так же выведите)
2. Покажите заказчиков (номер и имя), с которыми работали продавцы (номер и имя) с комиссионными 0.10 и 0.15 (0.16) (комиссионные так же выведите)
3. Покажите общую стоимость заказов заказчиков с рейтингом 200
4. Вывести имена всех продавцов и среднюю стоимость их заказов
5. Покажите имя продавца и имя заказчика, чьи заказы сделаны в марте на сумму свыше 2000р.

**2 Вариант**

1. Покажите все заказы, которые оформил заказчик *Hoffman* (вывести имя продавца, сумму его заказа и имя заказчика)
2. Покажите имена продавцов, которые работали с заказчиками чьи имена на букву «С»
3. Покажите максимальную стоимость заказа продавца *Peel* (имя продавца так же выведите)
4. Выведите имена тех заказчиков, которые оформляли заказы летом.
5. Покажите общую стоимость заказов в каждом городе продавцов (город тоже вывести)

**3 Вариант**

1. Покажите все заказы (номер заказа и его стоимость), которые оформил продавец с комиссией 0.10 (комиссию тоже вывести)
2. Покажите заказчиков (номер заказчика и его имя), с которыми работали продавцы либо из *Barcelona* либо с комиссией 0.13 (город и комиссионные так же вывести)
3. Покажите среднюю стоимость заказов для каждого заказчика (имя так же вывести) из городов *Rome*и *Berlin* (название города так же вывести)
4. Вывести имена всех заказчиков и максимальную стоимость их заказов
5. Покажите заказчика (имя заказчика), который работал с продавцом из *London* в июне месяце (город и дату так же вывести)

**4 Вариант**

1. Покажите все заказы (номер заказа), которые оформилзаказчикс рейтингом 300
2. Покажите продавцов (номер продавца и его имя), которые работали с заказчиками из *Rome* с рейтингом меньше 200 (город и рейтинг так же вывести)
3. Покажите заказ (номер заказа) с минимальной суммой приобретения, который оформил продавец с комиссионными 0.12 (комиссионные так же вывести)
4. Покажите имена тех продавцов, которые оформляли заказы (номер заказа) на сумму от 1000р. до 5000р. Летом (сумму заказа и даты та же вывести).
5. Покажите продавца (имя), который работал с заказчиком с рейтингом больше 200, весной (рейтинг и даты так же вывести)

**Критерий оценки**

Каждый полный правильный ответ оценивается в1 балл.

0-2 правильных ответа – «2»

3 правильных ответа – «3»

4 правильных ответа – «4»

5 правильных ответов – «5»

**Тема 1.8 и практические занятия 12-13-14-15**

**Проверяемые** результаты обучения по знаниям **З1,5,7,9,10,11** по общим компетенциям **ОК1,2,4,5,9**, по умениям **У2,4,5,6** и по профессиональным компетенциям **ПК 11.1-11.6**

З1: основы теории баз данных;

З5: основы реляционной алгебры;

З7: обеспечение непротиворечивости и целостности данных;

З9: язык запросов SQL;

З10: принципы нормализации отношений;

З11: способы контроля доступа к данным и управления привилегиями.

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 2 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности

ОК 4 Эффективно взаимодействовать и работать в коллективе и команде

ОК 5 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 9 Пользоваться профессиональной документацией на государственном и иностранном языках.

У2: использовать язык запросов для программного извлечения сведений из баз данных;

У4: реализовывать сложные запросы;

У5: настраивать привилегии доступа к данным;

У6: работать с Join

ПК 11.1 Осуществлять сбор, обработку и анализ информации для проектирования баз данных.

ПК 11.2 Проектировать базу данных на основе анализа предметной области.

ПК 11.3 Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.

ПК 11.4 Реализовывать базу данных в конкретной системе управления базами данных.

ПК 11.5 Администрировать базы данных.

ПК 11.6 Защищать информацию в базе данных с использованием технологии защиты информации.

**Практическое занятие № 12 Использование агрегатных функций в подзапросах**

**Краткие теоретические сведения**

Вы хотите увидеть все Заказы, имеющие сумму приобретений выше средней на 4-е Октября

SELECT \*  
FROM Orders  
WHERE amt > (SELECT AVG (amt)  
 FROM Orders  
 WHERE odate = 10/04/1990);

=============== SQL Execution Log ==============  
| SELECT \* |  
| FROM Orders |  
| WHERE amt > |  
| (SELECT AVG (amt) |  
| FROM Orders |  
| WHERE odate = 01/04/1990); |  
| =============================================== |  
| onum amt odate cnum snum |  
| ----- -------- ---------- ----- ----- |  
| 3002 1900.10 10/03/1990 2007 1004 |  
| 3005 5160,45 10/03/1990 2003 1002 |  
| 3006 1098.19 10/03/1990 2008 1007 |  
| 3009 1713.23 10/04/1990 2002 1003 |  
| 3008 4723.00 10/05/1990 2006 1001 |  
| 3010 1309.95 10/06/1990 2004 1002 |  
| 3011 9891.88 10/06/1990 2006 1001 |  
=================================================

Средняя сумма приобретений на 10Апреля — 1788.98 (1713.23 + 75.75) делится пополам, что в целом равняется = 894.49. Все строки со значением в поле amt выше этого — являются выбранными.

Имейте в виду, что сгруппированные агрегатные функции, которые являются агрегатными функциями, определенными в терминах предложения GROUP BY, могут производить многочисленные значения. Они, следовательно, не позволительны в подзапросах такого характера. Даже если GROUP BY и HAVING используются таким способом, что только одна группа выводится с помощью подзапроса, команда будет отклонена в принципе. Вы должны использовать одиночную агрегатную функцию с предложением WHERE, что устранит нежелательные группы. Например, следующий запрос, который должен найти среднее значение комиссионных продавца в Лондоне

SELECT AVG (comm)  
FROM Salespeople  
GROUP BY city  
HAVING city = 'London;

не может использоваться в подзапросе! Во всяком случае, это не лучший способ формировать запрос. Другимспособомможетбыть

SELECTAVG (comm)  
FROMSalespeople  
WHEREcity = 'London';

**Использование подзапросов, которые выдают много строк с помощью оператора IN**

Вы можете использовать подзапросы, которые производят любое число строк, если вы используете специальный оператор IN (операторы BETWEEN, LIKE, и IS NULL не могут использоваться с подзапросами). Например, мы можем найти все атрибуты таблицы Заказов для продавца в Лондоне

SELECT \*  
FROM Orders  
WHERE snum IN (SELECT snum  
 FROM Salespeople  
WHERE city = 'London');

=============== SQL Execution Log ==============  
| SELECT \* |  
| FROM Orders |  
| WHERE snum IN |  
| (SELECT snum |  
| FROM Salespeople |  
| WHERE city = 'London'); |  
| =============================================== |  
| onum amt odate cnum snum |  
| ----- -------- ---------- ----- ------ |  
| 3003 767.19 10/03/1990 2001 1001 |  
| 3002 1900.10 10/03/1990 2007 1004 |  
| 3006 1098.19 10/03/1990 2008 1007 |  
| 3008 4723.00 10/05/1990 2006 1001 |  
| 3011 9891.88 10/06/1990 2006 1001 |  
 =================================================

**Подзапросы выбирают одиночные столбцы**

Смысл всех подзапросов, обсужденных в этой главе тот, что все они выбирают одиночный столбец. Это обязательно, поскольку выбранный вывод сравнивается с одиночным значением. Подтверждением этому то, что SELECT \* не может использоваться в подзапросе.

**Использование выражений в подзапросах**

Вы можете использовать выражение, основанное на столбце, а не просто сам столбец, в предложении SELECT подзапроса. Это может быть выполнено или с помощью реляционных операторов, или с IN. Например, следующий запрос использует реляционный оператор = (вывод показывается в рисунке):

SELECT \*  
FROM Customers  
WHERE cnum = (SELECT snum + 1000  
 FROM Salespeople  
 WHERE sname = 'Serres');

=============== SQL Execution Log ============  
| SELECT \* |  
| FROM Customers |  
| WHERE cnum = |  
| (SELECT snum + 1000 |  
| WHERE Salespeople |  
| WHERE sname = 'Serres' |  
| ============================================= |  
| cnum cname city rating snum |  
| ----- -------- ---- ------ ----- |  
| 2002 Giovanni Rome 200 1003 |  
 ===============================================

**Подзапросы в предложении HAVING**

Вы можете также использовать подзапросы внутри предложения HAVING. Эти подзапросы могут использовать свои собственные агрегатные функции, если они не производят многочисленных значений или использовать GROUP BY или HAVING. Следующийзапросявляетсяэтомупримером (выводпоказываетсяврисунке):

SELECT rating, COUNT (DISTINCT cnum)  
FROM Customers  
GROUP BY rating  
HAVING rating > (SELECT AVG (rating)  
 FROM Customers  
 WHERE city = 'San Jose');

=============== SQL Execution Log ============  
| SELECT rating,count (DISTINCT cnum) |  
| FROM Customers |  
| GROUP BY rating |  
| HAVING rating > |  
| (SELECT AVG (rating) |  
| FROM Customers |  
| WHERE city = 'San Jose' |  
|===============================================|  
| rating |  
| -------- -------- |  
| 300 2 |  
 ===============================================

Эта команда подсчитывает заказчиков с оценками выше среднего в San Jose. Так как имеются другие оценки отличные от 300, они должны быть выведены с числом номеров заказчиков, которые имели эту оценку.

**Индивидуальное задание**

**Теоретическая часть**

1. Для чего используют агрегатные функции в подзапросах?

**Практическая часть**

1. Напишите запрос, который вывел бы имена Заказчиков, их рейтинг и среднюю стоимость заказа.
2. Напишите запрос, который вывел бы имена Заказчиков из Рима, рейтинг которых ниже среднего.
3. Напишите запрос, который выводит комиссионные (comm) всех Продавцов, обслуживавших Заказчиков в Париже.
4. Написать запрос, который подсчитывает количество Заказов с суммой приобретения ниже среднего, совершенных 10 марта 2021.
5. Написать запрос, который выводит Продавца, уникальный номер которого на 1000 меньше уникального номера Заказчика, проживающего в Берлине.
6. Напишите запрос, который бы выбрал общую сумму всех приобретений в Заказах для каждого продавца, у которого эта общая сумма больше, чем сумма наибольшего Заказа в таблице.

Покажите работу преподавателю.

**Практическое занятие № 13 Использование предложения UNION**

**Краткие теоретические сведения**

В практических выше рассматривались различные способы, которыми запросы могут помещаться один внутрь другого. Имеется другой способ объединения многочисленных запросов — т.е. формирование их в объединение. В этой главе вы научитесь использованию предложения UNION в SQL. **UNION** отличается от подзапросов тем что в нем ни один из двух (или больше) запросов не управляются другим запросом. Все запросы выполняются независимо друг от друга, а уже вывод их – объединяется.

**Объединение нескольких запросов в один**

Предложение UNION объединяет вывод двух или более SQL запросов в единый набор строк и столбцов. Например, чтобы получить всех продавцов и заказчиков, размещенных в Лондоне, и вывести их как единое целое, вы могли бы ввести (результат вывода смотрите на рисунке 1):

SELECT snum, sname  
FROM Salespeople  
WHERE city = 'London'  
UNION  
SELECT cnum, cname  
FROM Customers  
WHERE city = 'London';

=============== SQL Execution Log ============  
| |  
| SELECT snum, sname |  
| FROM Salespeople |  
| WHERE city = 'London' |  
| UNION |  
| SELECT cnum, cname |  
| FROM Customers |  
| WHERE city = 'London'; |  
| ============================================= |  
| |  
| ----- -------- |  
| 1001 Peel |  
| 1004 Motika |  
| 2001 Hoffman |  
| 2006 Clemens |  
| |  
 ===============================================

Как вы можете видеть, столбцы, выбранные двумя командами, выведены так, как если бы она была одна. Заголовки столбца исключены, потому что ни один из столбцов, выведенных объединением, не был извлечен непосредственно из только одной таблицы. Следовательно, все эти столбцы вывода не имеют никаких имен. Кроме того, обратите внимание, что только последний запрос заканчивается точкой с запятой. Отсутствие точки с запятой дает понять SQL, что имеется еще одно или более запросов.

**UNION и устранение дубликатов**

UNION будет автоматически исключать дубликаты строк из вывода. Это нечто несвойственное для SQL, так как одиночные запросы обычно содержат DISTINCT, чтобы устранять дубликаты. Например, запрос, чей вывод показывается на рисунке 2,

SELECTsnum, city  
FROMCustomers;

имеет двойную комбинацию значений (snum=1001, city='London'), потому что мы не указали, чтобы SQL устранил дубликаты. Однако, если мы используем UNION в комбинации этого запроса с ему подобным в таблице Продавцов, то эта избыточная комбинация будет устранена. Рисунок 3 показывает вывод следующего запроса.

SELECT snum, city  
FROM Customers  
UNION  
SELECT snum, city  
FROM Salespeople;

=============== SQL Execution Log ============  
| SELECT snum, city |  
| FROM Customers; |  
| ============================================= |  
| snum city |  
| ----- -------- |  
| 1001 London |  
| 1003 Rome |  
| 1002 San Jose |  
| 1002 Berlin |  
| 1001 London |  
| 1004 Rome |  
| 1007 San Jose |  
 ===============================================

=============== SQLExecutionLog ============  
| FROMCustomers |  
| UNION |  
| SELECTsnum, city |  
| FROMSalespeople; |  
| ============================================= |  
| |  
| ----- -------- |  
| 1001 London |  
| 1002 Berlin |  
| 1007 SanJose |  
| 1007 NewYork |  
| 1003 Rome |  
| 1001 London |  
| 1003 Rome |  
| 1002 Barcelona |  
| 1007 SanJose |  
 ===============================================

Вы можете получить нечто похожее в некоторых программах SQL, используя UNION ALL вместо просто оператора UNION, наподобие этого:

SELECT snum, city  
FROM Customers  
UNION ALL  
SELECT snum, city  
FROM Salespeople;

**Использование строк и выражений с UNION**

Иногда, вы можете вставлять константы и выражения в предложения SELECT, используемые с UNION. Это не следует строго указаниям ANSI, но это полезная и необычно используемая возможность. Константы и выражения, которые вы используете, должны встречать совместимые стандарты, которые мы выделяли ранее. Эта свойство полезно, например, чтобы устанавливать комментарии, указывающие, какой запрос вывел данную строку.

Предположим, что вы должны сделать отчет о том, какие продавцы производят наибольшие и наименьшие Заказы по датам. Мы можем объединить два запроса, вставив туда текст, чтобы различать вывод для каждого из них.

SELECT a.snum, sname, onum, 'Самый высокий на', odate  
FROM Salespeople a, Orders b  
WHERE a.snum = b.snum AND b.amt = (SELECT MAX (amt)  
 FROM Orders c  
 WHERE c.odate = b.odate)  
UNION  
SELECT a.snum, sname, onum, 'Самый низкий на ', odate  
FROM Salespeople a, Orders b  
WHERE a.snum = b.snum AND b.amt = (SELECT MIN (amt)  
 FROM Orders c  
 WHERE c.odate = b.odate);

=============== SQL Execution Log ============  
| AND b.amt = |  
| (SELECT min (amt) |  
| FROM Orders c |  
| WHERE c.odate = b.odate); |  
| ============================================= |  
| |  
| ----- ------- ------ ---------- ----------- |  
| 1001 Peel 3008 Highest on 10/05/1990 |  
| 1001 Peel 3008 Lowest on 10/05/1990 |  
| 1001 Peel 3011 Highest on 10/06/1990 |  
| 1002 Serres 3005 Highest on 10/03/1990 |  
| 1002 Serres 3007 Lowest on 10/04/1990 |  
| 1002 Serres 3010 Lowest on 10/06/1990 |  
| 1003 Axelrod 3009 Highest on 10/04/1990 |  
| 1007 Rifkin 3001 Lowest on 10/03/1990 |  
 ===============================================  
  
**Использование UNION с ORDER BY**

Вы можете использовать предложение ORDER BY чтобы упорядочить вывод из объединения, точно так же как это делается в индивидуальных запросах. Давайте пересмотрим наш последний пример чтобы упорядочить имена с помощью их Заказовых номеров. Это может внести противоречие, такое как повторение имени Peel в последней команде, как вы сможете увидеть из вывода, показанного в рисунке ниже.

SELECT a.snum, sname, onum, 'Highest on', odate  
FROM Salespeople a, Orders b  
WHERE a.snum = b.snum AND b.amt = (SELECT MAX (amt)  
 FROM Orders c  
 WHERE c.odate = b.odate)  
UNION  
SELECT a.snum, sname, onum, 'Lowest on', odate  
FROM Salespeople a, Orders b  
WHERE a.snum = b.snum AND b.amt = (SELECT MIN (amt)  
 FROM Orders c  
 WHERE c.odate = b.odate)  
ORDER BY 3;

=============== SQL Execution Log ============  
| (SELECT min (amt) |  
| FROM Orders c |  
| WHERE c.odate = b.odate) |  
| ORDER BY 3; |  
| ============================================= |  
| |  
| ----- ------- ------ ---------- ----------- |  
| 1007 Rifkin 3001 Lowest on 10/03/1990 |  
| 1002 Serres 3005 Highest on 10/03/1990 |  
| 1002 Serres 3007 Lowest on 10/04/1990 |  
| 1001 Peel 3008 Highest on 10/05/1990 |  
| 1001 Peel 3008 Lowest on 10/05/1990 |  
| 1003 Axelrod 3009 Highest on 10/04/1990 |  
| 1002 Serres 3010 Lowest on 10/06/1990 |  
| 1001 Peel 3011 Highest on 10/06/1990 |  
 ===============================================

Пока ORDER BY используется по умолчанию, мы не должны его указывать. Мы можем упорядочить наш вывод с помощью нескольких полей, одно внутри другого и указать ASC или DESC для каждого, точно также как мы делали это для одиночных запросов. Заметьте, что номер 3 в предложении ORDER BY указывает, какой столбец из предложения SELECT будет упорядочен. Так как столбцы объединения — это столбцы вывода, они не имеют имен, и, следовательно, должны определяться по номеру.

**Внешнее объединение**

Операция, которая бывает часто полезна — это объединение из двух запросов, в котором второй запрос выбирает строки, исключенные первым. Наиболее часто вы будете делать это так, чтобы не исключать строки, которые не удовлетворили предикату при объединении таблиц. Это называется *внешним объединением*.

Предположим, что некоторые из ваших заказчиков еще не были назначены к продавцам. Вы можете захотеть увидеть имена и города всех ваших заказчиков, с именами их продавцов, не учитывая тех, кто еще не был назначен. Вы можете достичь этого, формируя объединение из двух запросов, один из которых выполняет объединение, а другой выбирает заказчиков с пустыми (NULL) значениями поля snum. Этот последний запрос должен вставлять пробелы в поля, соответствующие полю sname в первом запросе.

Как и раньше, вы можете вставлять текстовые строки в ваш вывод, чтобы идентифицировать запрос, который вывел данную строку. Использование этой методики во внешнем объединении дает возможность использовать предикаты для классификации, а не для исключения. Мы использовали пример нахождения продавцов с заказчиками, размещенными в их городах и раньше. Однако вместо просто выбора только этих строк, вы, возможно, захотите, чтобы ваш вывод перечислял всех продавцов, и указывал тех, кто не имел заказчиков в их городах, и кто имел. Следующий запрос, чей вывод показывается в Рисунке 6, выполнит это:

SELECT Salespeople.snum, sname, cname, comm  
FROM Salespeople, Customers  
WHERE Salespeople.city = Customers.city  
UNION  
SELECT snum, sname, 'NO MATCH ', comm  
FROM Salespeople  
WHERE NOT city = ANY (SELECT city  
 FROM Customers)  
ORDER BY 2 DESC;

=============== SQL Execution Log ============  
| FROM Salespeople |  
| WHERE NOT city = ANY (SELECT city |  
| FROM Customers) |  
| ORDER BY 2 DESC; |  
| ============================================= |  
| |  
| ----- ------- --------- ------------ |  
| 1002 Serres Cisneros 0.1300 |  
| 1002 Serres Liu 0.1300 |  
| 1007 Rifkin NO MATCH 0.1500 |  
| 1001 Peel Clemens 0.1200 |  
| 1001 Peel Hoffman 0.1200 |  
| 1004 Motika Clemens 0.1100 |  
| 1004 Motika Hoffman 0.1100 |  
| 1003 Axelrod NO MATCH 0.1000 |  
 ===============================================

Строка 'NO MATCH' была дополнена пробелами, чтобы получить совпадение поля cname по длине (это не обязательно во всех реализациях SQL). Второй запрос выбирает даже те строки, которые исключил первый. Вы можете также добавить комментарий или выражение к вашему запросу, в виде дополнительного поля. Если вы сделаете это, вы будете должны добавить некоторый дополнительный комментарий или выражение в той же самой позиции среди выбранных полей, для каждого запроса в операции объединения. Совместимость UNION предотвращает вас от добавления дополнительного поля для первого запроса, но не для второго. Имеется запрос, который добавляет строки к выбранным полям, и указывает, совпадает ли данный продавец с его заказчиком в его городе:

SELECT a.snum, sname, a.city, 'MATCHED '  
FROM Salespeople a, Customers b  
WHERE a.city = b.city  
UNION  
SELECT snum, sname, city, 'NO MATCH'  
FROM Salespeople  
WHERE NOT city = ANY (SELECT city  
 FROM Customers)  
ORDER BY 2 DESC;

=============== SQL Execution Log ============  
| WHERE a.city = b.city |  
| union |  
| SELECT snum,sname,city, 'NO MATCH' |  
| FROM Salespeople |  
| WHERE NOT city = (SELECT city |  
| FROM Customers) |  
| ORDER BY 2 DESC; |  
| ============================================= |  
| |  
| ----- ------- ------------ --------- |  
| 1002 Serres San Jose MATCHED |  
| 1007 Rifkin Barselona NO MATCH |  
| 1001 Peel London MATCHED |  
| 1004 Motika London MATCHED |  
| 1003 Axelrod New York NO MATCH |  
 ===============================================

**Индивидуальное задание:**

**Теоретическая часть**

1. В чём назначение UNION?
2. В чём отличие устранение дубликатов с использованием UNION и DISTINCT?
3. Для чего используется UNION с ORDER BY?
4. Что такое внешнее объединение и в чём его назначение?

**Практическая часть**

1. Создайте объединение из двух запросов, которое показало бы имена, города, и оценки всех заказчиков. Те из них, которые имеют поле rating=200 и более, должны, кроме того, иметь слова "Высокий Рейтинг", а остальные должны иметь слова "Низкий Рейтинг".
2. Напишите команду, которая бы вывела имена и номера каждого продавца и каждого заказчика, которые имеют больше одного заказа. Результат представьте в алфавитном порядке.
3. Сформируйте объединение из трех запросов. Первый выбирает поля snum всех продавцов в San Jose; второй, поля cnum всех заказчиков в San Jose; и третий поля onum всех Заказов на 10 Марта. Сохраните дубликаты между последними двумя запросами, но устраните любую избыточность вывода между каждым из них и самым первым. Примечание: в данных типовых таблицах не содержится никакой избыточности. Это только пример.

Покажите работу преподавателю.

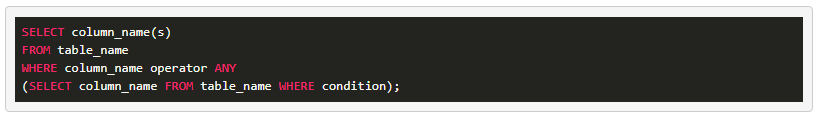
**Практическое занятие № 14 Операторы ANY и ALL в подзапросах**

Операторы ANY и ALL используются с подзапросами, когда подзапрос выдаёт больше одного значения (именно в таких случаях не всегда справляется оператор IN). В отличии от оператора IN, ANY и ALL могут использовать не только оператор « = », но и операторы « >», «<» и т.д.

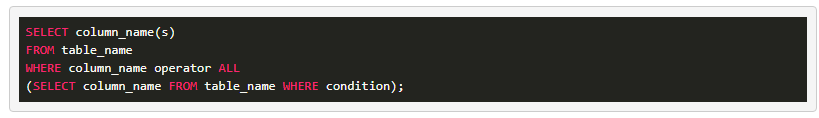
Оператор ANY возвращает true, если какое-либо из значений подзапроса соответствует условию.

Оператор ALL возвращает true, если все значения подзапроса удовлетворяют условию.

**Синтаксис ANY**

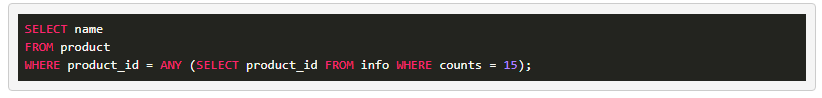
****

**Синтаксис ALL**

****

**Пример ANY**

Вывести имена товаров, соответствующие ЛЮБЫМ записям в таблице info, с количеством = 15:

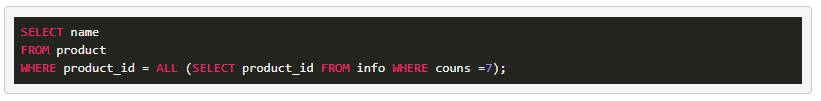


Вывести имена товаров, соответствующие ЛЮБЫМ записям в таблице info, с количеством> 15:

****

**Пример ALL**

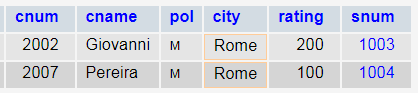
Вывести имена товаров, соответствующие ВСЕМ записям в таблице info с количество = 7:



Пример сравнения результатов одного и того же за проса с операторами ALL и ANY.

Вывести все данные о Заказчиках, у которых рейтинг выше, чем рейтинг у заказчиков из Rome.

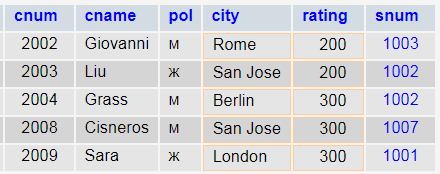
У заказчиков из Rome рейтинг равен 100 и 200.



ANY выдаст и тех заказчиков, у которых рейтинг больше 100, и тех, у кого рейтинг больше 200, т.е. или 200 или 300.

SELECT\*FROMCust

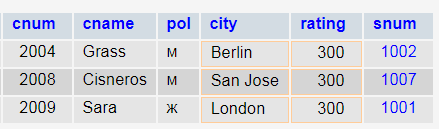
WHERErating>ANY(SELECT ratingFROMCustWHEREcity='Rome')



ALL имеет более жёсткие условия, он выдаст только тех заказчиков, у которых рейтинг 300. Поусловию 200 > 100, но 200 < 300. Таким образом он выдаст только те, кто именно больше самого максимального из предложенных.

SELECT\* FROMCust

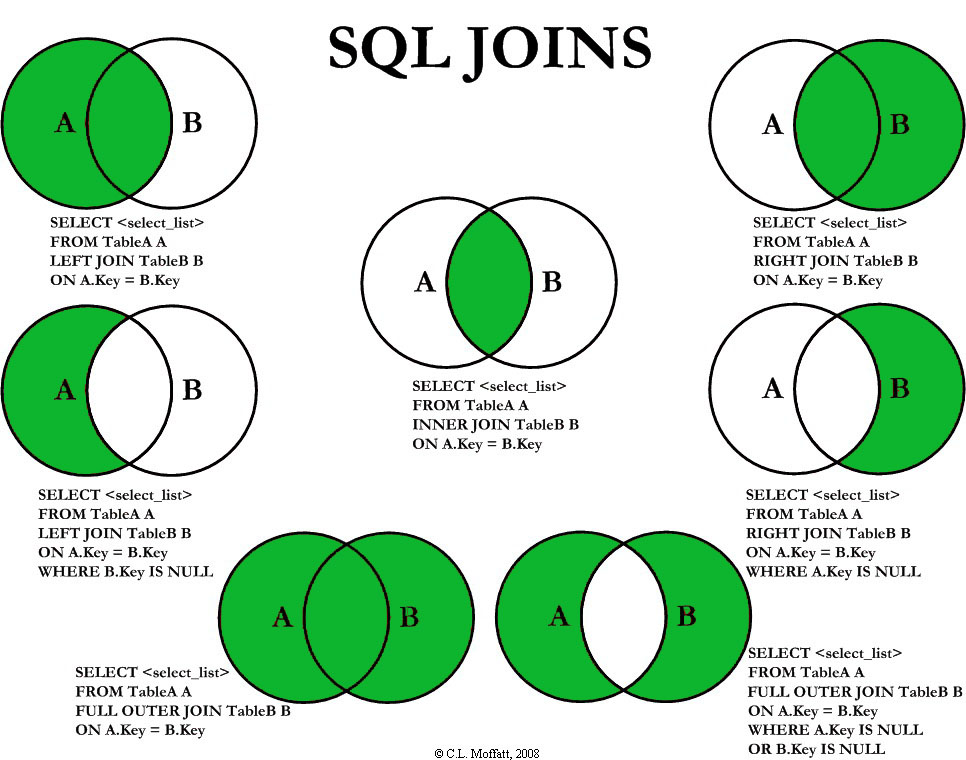
WHERErating> ALL (SELECT ratingFROMCustWHEREcity='Rome')



**Индивидуальное задание:**

1. Показать тех Продавцов и их города, которые проживают в тех же городах, что и Заказчики.
2. Показать продавцов, у которых комиссионные выше чем у продавцов из Лондона.
3. Покажите заказы, которые дешевле стоимости заказов за 2021-04-10. (запрос должен показать только те значения, которые ниже amt самого дешёвого за эту дату)
4. Покажите номер продавца, у которого самая высокая суммарная стоимость заказов (суммарную стоимость заказа так же вывести).
5. Покажите имя и номер продавца, у которого самая высокая суммарная стоимость заказов (суммарную стоимость заказа так же вывести).

**Практическое занятие №15 Работа с JOIN в MySQL**



|  |  |
| --- | --- |
| **Запросы без JOIN** | **Запросы с JOIN** |
| Раунд 1. Вывести все заказы с информацией по их заказчикам | |
| [SELECT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html#_blank) \* FROM orders,customers WHERE orders.cnum = customers.cnum | [SELECT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html#_blank) \* FROM orders INNER JOIN customers ON orders.cnum = customers.cnum  ЛИБО ЕЩЁ ПРОЩЕ  [SELECT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html#_blank) \* FROM orders INNER JOIN customers USING (cnum) |
| 1 : 1  (никакой особой разницы) | |
| Раунд 2. Вывести номер, имя продавца и имя заказчика для каждого заказа, у которого сумма больше 2000 | |
| [SELECT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html#_blank) onum,sname,cname FROM orders,customers,salespeople WHERE orders.snum = salespeople.snum [AND](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#_blank) orders.cnum = customers.cnum [AND](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#_blank) amt > 2000 | [SELECT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html#_blank) onum,sname,cname FROM orders INNER JOIN salespeople USING (snum) INNER JOIN customers USING (cnum) WHERE amt > 2000 |
| 2 : 2  (небольшая разница, просто без JOIN удобнее добавить таблицы, но неудобнее их связать) | |
| Раунд 3. Вывести имя **каждого** заказчика и сумму его заказов | |
| [SELECT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html#_blank) cname, ([SELECT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html#_blank)[SUM](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#_blank)(amt) FROM orders WHERE orders.cnum = customers.cnum) as 'CustomerTotal' FROM customers | [SELECT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html#_blank) cname, [SUM](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#_blank)(amt) as 'CustomerTotal' FROM customers [LEFT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/string-functions.html#_blank) OUTER JOIN orders USING (cnum) GROUP BY cnum |
| 2:3  (без JOIN придется делать дополнительные подзапросы для каждого запроса, затратно) | |
| Раунд 4. Вывести таблицу с соотношениями продавцов и заказчиков, также включающую продавцов и заказчиков, не имеющих соотношения | |
| SELECT sname, cname FROM salespeople,customers WHERE salespeople.snum = customers.snum UNION SELECT sname, NULL as 'cname' FROM salespeople WHERE snum NOT IN (SELECT snum FROM customers) UNION SELECT NULL as 'sname', cname FROM customers WHERE snum IS NULL | \*неподдерживаетсяв MySQL\*  SELECT sname,cname FROM salespeople FULL OUTER JOIN customers USING (snum) |
| 2 : 4  (нормально достигается только через JOIN, но в MySQL не поддерживается) | |
| Итоги: JOINы победили | |

**Задание для выполнения практической занятия необходимо взять из практического занятия № 10.**

**Проверочное задание на проверку полученных знаний**

Напишите SQLкоманду на поставленный вопрос.

1. **Вариант**
2. Покажите продавцов (номер и имя), которые оформили больше двух заказов (кол-во заказов так же вывести)
3. Покажите заказчиков (номер и имя), которые оформили больше одного заказа в каждом месяце (кол-во заказов и дату так же вывести)
4. Покажите город заказчика, у которого самый большой суммарный рейтинг (вывести название города и его максимальный суммарный рейтинг)
5. Покажите заказы (номер и суму) заказчиков, которые проживают в городе с максимальным суммарным рейтингом.
6. Покажите имя заказчика и стоимость его заказа, которые проживают в городе с максимальным суммарным рейтингом

**2Вариант**

1. Покажите продавца, у которого самая большая суммарная стоимость заказа (вывести имя продавца и общую сумму его покупок)
2. Покажите имена продавцов, которые работали с заказчиками чьи имена на букву «С»
3. Покажите максимальную стоимость заказа продавца *Peel* (имя продавца так же выведите)
4. Выведите имена тех заказчиков, которые оформляли заказы летом.
5. Покажите общую стоимость заказов в каждом городе продавцов при условии, что эта общая стоимость выше средней стоимости всех заказов (город тоже вывести)

**Критерий оценки**

Каждый полный правильный ответ оценивается в1 балл.

0-2 правильных ответа – «2»

3 правильных ответа – «3»

4 правильных ответа – «4»

5 правильных ответов – «5»

**Тема 1.8 и практические занятия 16-17**

**Практическое занятие № 16 Введение в представления**

Представление**(VIEW**) – объект данных, который не содержит никаких данных его владельца. Это – тип таблицы, чье содержание выбирается из других таблиц с помощью выполнения запроса. Поскольку значения в этих таблицах меняются, то автоматически, их значения могут быть показаны представлением.

Типы таблиц, с которыми вы имели дело до сих пор, назывались –*базовыми таблицами*. Это – таблицы, которые содержат данные. Однако имеется другой вид таблиц – представления. *Представления*– это таблицы, чье содержание выбирается или получается из других таблиц. Они работают в запросах и операторах DML точно также, как и основные таблицы, но не содержат никаких собственных данных.

*Представление*– это фактически запрос, который выполняется всякий раз, когда представление становится темой команды. Вывод запроса при этом в каждый момент становится содержанием представления.

**Команда CREATE VIEW**

Вы создаете представление командой **CREATE VIEW.** Она состоит из слов **CREATE VIEW** (создать представление), *имени представления*, которое нужно создать, слова **AS** (*как*), и далее *запроса*, как в следующем примере:

CREATE VIEW Londonstaff

AS SELECT \*

FROM Salespeople WHERE city = 'London';

Теперь Вы имеете представление, называемое **Londonstaff**. Вы можете использовать это представление точно так же, как и любую другую таблицу. Она может быть запрошена, модифицирована, вставлена в, удалена из, и соединена с, другими таблицами и представлениями. Давайте сделаем запрос такого представления (вывод показан в Рисунке 1):

Select \*FROMLondonstaff;

=============== SQL Execution Log ========  
| SELECT \* |  
| FROM Londonstaff; |  
| =======================================|  
| snum sname city comm |  
| ------ ---------- ----------- ------- |  
| 1001 Peel London 0.1200 |  
| 1004 Motika London 0.1100 |  
 =======================================

Рисунок 1 – Представление Londonstaff

Когда вы приказываете SQL выбрать (**SELECT**) все строки (**\***) из представления, он выполняет запрос, содержащийся в определении Londonstaff, и возвращает все из его вывода.

Имея предикат в запросе представления, можно вывести только те строки из представления, которые будут удовлетворять этому предикату. Преимущество использования представления, по сравнению с использованием основной таблицы в том, что представление будет модифицировано автоматически всякий раз, когда таблица, лежащая в его основе, изменяется. Содержание представления не фиксировано, и переназначается каждый раз, когда вы ссылаетесь на представление в команде. Если вы добавите завтра другого, живущего в Лондоне продавца, он автоматически появится в представлении.

Представления значительно расширяют управление вашими данными. Это превосходный способ дать публичный доступ к некоторой, но не всей информации в таблице. Если вы хотите, чтобы ваш продавец был показан в таблице Продавцов, но при этом не были показаны комиссии других продавцов, вы могли бы создать представление с использованием следующего оператора (вывод показан в Рисунке 2)

CREATE VIEW Salesown

AS SELECT snum, sname, city

FROM Salespeople:

=============== SQL Execution Log ============  
| SELECT \* |  
| FROM Salesown; |  
| ==============================================|  
| snum sname city |  
| ------ ---------- ----------- |  
| 1001 Peel London |  
| 1002 Serres San Jose |  
| 1004 Motika London |  
| 1007 Rifkin Barcelona |  
| 1003 Axelrod New York |  
 ===============================================

Рисунок 2 – Представление Salesown

Другими словами, это представление – такое же, как для таблицы Продавцов, за исключением того, что поле comm не упоминалось в запросе, и, следовательно, не было включено в представление.

**Модифицирование представлений**

Представление может теперь изменяться командами модификации DML, но модификация не будет воздействовать на само представление. Команды будут на самом деле перенаправлены к базовой таблице:

UPDATESalesown

SETcity = 'PaloAlto'

WHEREsnum = 1004;

Его действие идентично выполнению той же команды в таблице Продавцов. Однако, если значение комиссионных продавца будет обработано командой UPDATE

UPDATESalesown

SETcomm = 0.20

WHEREsnum = 1004;

она будет отвергнута, так как поле comm отсутствует в представлении Salesown. Это важное замечание, показывающее, что не все представления могут быть модифицированы.

**Именование столбцов**

В нашем примере, поля наших представлений имеют свои имена, полученные прямо из имен полей основной таблицы. Это удобно. Однако, иногда вам нужно снабжать ваши столбцы новыми именами:

* когда некоторые столбцы являются выводимыми и поэтому не имеющими имен;
* когда два или более столбцов в объединении имеют те же имена, что в их базовой таблице.

Имена, которые могут стать именами полей, **даются в круглых скобках после имени таблиц**. Они не будут запрошены, если совпадают с именами полей запрашиваемой таблицы. Тип данных и размер этих полей будут отличаются от типа данных и размера запрашиваемых полей, которые "передаются" в них. Обычно вы не указываете новых имен полей, но, если вы все-таки сделали это, вы должны делать это для каждого поля в представлении.

**Групповые представления**

*Групповые представления*– это представления, наподобие запроса **Ratingcount** в предыдущем примере, который содержит предложение GROUP BY, или который основывается на других групповых представлениях.

Групповые представления могут стать превосходным способом обрабатывать полученную информацию непрерывно. Предположим, что каждый день вы должны следить за порядком номеров заказчиков, номерами продавцов, принимающих Заказы, номерами Заказов, средним от Заказов, и общей суммой приобретений в Заказах.

Вместо того, чтобы конструировать каждый раз сложный запрос, вы можете просто создать следующее представление:

CREATE VIEW Totalforday

AS SELECT odate, COUNT(DISTINCT cnum), COUNT(DISTINCT snum),

COUNT(onum), AVG(amt), SUM(amt)

FROM Orders

GROUP BY odate;

Теперь вы сможете увидеть всю эту информацию с помощью простого запроса:

SELECT \*FROMTotalforday;

**Представления и объединения**

Представления не требуют, чтобы их вывод осуществлялся из одной базовой таблицы. Так как почти любой допустимый запрос SQL может быть использован в представлении, он может выводить информацию из любого числа базовых таблиц, или из других представлений.

Мы можем, например, создать представление, которое показывало бы Заказы продавца и заказчика по имени:

CREATE VIEW Nameorders

AS SELECT onum, amt, a.snum, sname, cname

FROM Orders a, Customers b, Salespeople c

WHERE a.cnum = b.cnum AND a.snum = c.snum;

Теперь вы можете выбрать все Заказы заказчика или продавца, или можете увидеть эту информацию для любого Заказа.

Например, чтобы увидеть все Заказы продавца Rifkin, вы должны ввести следующий запрос (вывод показан в 3 Рисунке):

SELECT \*

FROM Nameorders

WHERE sname = 'Rifkin';

=============== SQL Execution Log ==============  
| SELECT \* |  
| FROM Nameorders |  
| WHERE sname = 'Rifkin'; |  
| =============================================== |  
| onum amt snum sname cname |  
| ------ -------- ----- ------- ------- |  
| 3001 18.69 1007 Rifkin Cisneros |  
| 3006 1098.16 1007 Rifkin Cisneros |  
 ================================================

Рисунок 3 – Заказы Rifkin показаные в Nameorders

Вы можете также объединять представления с другими таблицами, или базовыми таблицами или представлениями, поэтому вы можете увидеть все Заказы Axelrod и значения его комиссионных в каждом Заказе:

SELECT a.sname, cname, amt comm

FROM Nameorders a, Salespeople b

WHERE a.sname = 'Axelrod' AND b.snum = a.snum;

=============== SQLExecutionLog ==============  
| SELECTa.sname, cname, amt \* comm |  
| FROMNameordersa, Salespeopleb |  
| WHEREa.sname = 'Axelrod'ANDb.snum = a.snum; |  
| =============================================== |  
| onumamtsnumsnamecname |  
| ------ -------- ----- ------- ------- |  
| 3001 18.69 1007 RifkinCisneros |  
| 3006 1098.16 1007 RifkinCisneros |  
 ================================================

Рисунок 4 – Объединение основной таблицы с представлением

В предикате, мы могли бы написать – "WHERE a.sname = ’Axelrod' AND b.sname = ’Axelrod’", но предикат, который мы использовали здесь, более общеупотребительный. Кроме того, поле snum – это первичный ключ таблицы Продавцов и, следовательно, должен по определению быть уникальным.

Если бы там, например, было два Axelrod, вариант с именем будет объединять вместе их данные. Более предпочтительный вариант – использовать поле snum, чтобы хранить его отдельно.

**Представления и подзапросы**

Представления могут также использовать и подзапросы, включая соотнесенные подзапросы. Предположим, ваша компания предусматривает премию для тех продавцов, которые имеют заказчика с самой высокой суммой Заказа для любой указанной даты. Вы можете проследить эту информацию с помощью представления:

CREATEVIEWElitesalesforce

ASSELECTb.odate, a.snum, a.sname,

FROMSalespeoplea, Ordersb

WHEREa.snum = b.snumANDb.amt = (SELECTMAX (amt)FROMOrdersc

WHEREc.odate = b.odate);

Если, с другой стороны, премия будет назначаться только продавцу, который имел самую высокую сумму Заказа за последние десять лет, вам необходимо будет проследить их в другом представлении, основанном на первом:

CREATE VIEW Bonus

AS SELECT DISTINCT snum, sname

FROM Elitesalesforce a

WHERE 10 <= (SELECT COUNT (\*)

FROM Elitesalestorce b

WHERE a.snum = b.snum);

Извлечение из этой таблицы продавца, который будет получать премию, выполняется простым запросом:

SELECT \* FROMBonus;

Теперь мы видим истинную мощность SQL. Извлечение той же полученной информации программами RPG или COBOL будет более длительной процедурой. В SQL это только запрос из двух комплексных команд, сохраненных как представление совместно с простым запросом.

**Удаление представлений**

Синтаксис удаления представления из базы данных подобен синтаксису удаления базовых таблиц:

**DROPVIEW<viewname>**

В этом нет необходимости, однако, сначала надо удалить все содержание, как это делается с базовой таблицей, потому что содержание представления не является созданным и сохраняется в течение определенной команды. Базовая таблица, из которой представление выводится, не эффективна, когда представление удалено.

Помните, вы должны являться владельцем представления, чтобы иметь возможность удалить его.

**Индивидуальное задание:**

**Теоретическая часть**

1. Что такое представление? В чём его назначение и приимущество?
2. Формат команды создания представления.
3. Как происходит модифицирование представления?
4. Как происходит изменение столбцов в представлении?
5. В чём назначение групповых представлений?
6. В чём назначение представлениея и объединения?
7. В чём назначение представлениея и подзапросов?
8. Формат команды удаления представления.

**Практическая часть**

1. Создайте представление, которое показываетвсе заказы, которые были сделаны с марта по апрель месяц.
2. Создайте представление, которое показываетимя продавца, имя заказчика, среднее по заказам каждогопродавца и каждого заказчика, а также самый дорогой заказ каждого продавца и каждого заказчика.
3. Создайте представление, которое показываетимя продавца, его среднюю и общую стоимость заказа. Предполагается, что все имена уникальны.
4. Создайте представление, которое показывает все заказы у тех продавцов, которые живут в Лондоне.
5. Создайте представление, которое покажет всех продавцов, которые делали заказы в апреле.

**Практическое занятие № 17 Пользователи и привилегии**

В этой главе, вы обучитесь работе с привилегиями пользователями.

SQL используется обычно в средах, которые требуют распознавания пользователей и различия между различными пользователями систем. Вообще говоря, администраторы баз данных, сами создают пользователей и дают им привилегии. С другой стороны, пользователи, которые создают таблицы, сами имеют права на управление этими таблицами.

*Привилегии* — это то, что определяет, может ли указанный пользователь выполнить данную команду. Имеется несколько типов привилегий, соответствующих нескольким типам операций. Привилегии даются и отменяются двумя командами SQL: —**GRANT** (*ДОПУСК*) и **REVOKE** (*ОТМЕНА*). Эта Практическое занятие покажет вам, как эти команды используются.

**Предоставление привилегий**

Каждый пользователь в SQL базе данных имеет набор привилегий.

Это то, что пользователю разрешается делать (возможно это файл регистрации, который может рассматриваться как минимальная привилегия). Эти привилегии могут изменяться со временем — новые добавляться, старые удаляться. Некоторые из этих привилегий определены в ANSI SQL, но имеются и дополнительные привилегии, которые являются также необходимыми. SQL привилегии, как определено ANSI, не достаточны в большинстве ситуаций реальной жизни. С другой стороны, типы привилегий, которые необходимы, могут видоизменяться с видом системы, которую вы используете, относительно которой ANSI не может дать никаких рекомендаций. Привилегии, которые не являются частью стандарта SQL, могут использовать похожий синтаксис и не полностью совпадающий со стандартом.

Привилегии объекта связаны одновременно и с пользователями, и с таблицами. То есть, привилегия дается определенному пользователю в указанной таблице, или базовой таблице или представлении. Вы должны помнить, что пользователь, создавший таблицу (любого вида), является владельцем этой таблицы. Это означает, что пользователь имеет все привилегии в этой таблице и может передавать привилегии другим пользователям в этой таблице.

Привилегии, которые можно назначить пользователю

|  |  |
| --- | --- |
| SELECT | Пользователь с этой привилегией может выполнять запросы в таблице. |
| INSERT | Пользователь с этой привилегией может выполнять команду INSERT в таблице. |
| UPDATE | Пользователь с этой привилегией может выполнять команду UPDATE на таблице. Вы можете ограничить эту привилегию для определенных столбцов таблицы. |
| DELETE | Пользователь с этой привилегией может выполнять команду DELETE в таблице. |
| REFERENCES | Пользователь с этой привилегией может определить внешний ключ, который использует один или более столбцов этой таблицы, как родительский ключ. Вы можете ограничить эту привилегию для определенных столбцов. |

**Практическая часть:**

**Часть 1 Создание пользователя**

Создадим пользователя с именем newuser:

CREATEUSER 'newuser'@'localhost' IDENTIFIEDBY '12345';

К сожалению, на данном этапе пользователь "newuser" не имеет прав делать что-либо с базами данных.

Таким образом, первое, что нам необходимо сделать, это предоставить пользователю доступ к информации, которая ему потребуется:

GRANT ALL PRIVILEGES ON \* . \* TO 'newuser'@'localhost';

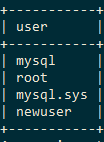
Звездочки в этой команде задают базу и таблицу, соответственно, к которым у пользователя будет доступ. Конкретно эта команда позволяет пользователю читать, редактировать, выполнять любые действия над всеми базами данных и таблицами.

Поле завершения настройки прав доступа новых пользователей, убедитесь, что вы обновили все права доступа:

FLUSH PRIVILEGES;

Проверим, если ли пользователь в системе:

SELECT `user` FROM `mysql` . `user`;



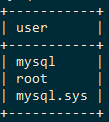
В списке пользователей появился newuser.

Теперь удалим пользователя из MySQL:

DROP USER 'newuser'@'localhost';

Ещё раз проверим список пользователей и убедимся, что пользователь удалён:

SELECT `user` FROM `mysql` . `user`;



**Часть 2 Работа с привилегиями**

Создадим пользователя:

CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';

Так как мы уже знаем, что пользователь на данном этапе не имеет прав, то дадим пользователю права на выборку из базы gruber:

GRANT SELECT ON gruber.\* TO 'newuser'@'localhost';

FLUSH PRIVILEGES;

Посмотрим список прав для ‘newuser’

SHOW GRANTS FOR 'newuser'@'localhost';

Если там есть строка –

GRANT SELECT ON `gruber`.\*TO 'newuser'@'localhost'

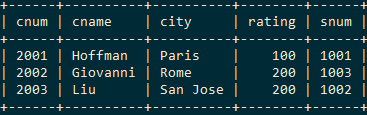
– то всё сделано правильно

Теперь зайдем под новым пользователем и посмотрим можем ли мы делать выборку из баз данных:

mysql -u newuser –p

use gruber;

SELECT \* FROM customers LIMIT 3;



Теперь изменим стоку в таблице:

UPDATE customers SET cname="HOST" WHERE cnum='2001';

MySQLвыдастошибку: UPDATE command denied to user 'newuser'@'localhost' for table 'customers'

Это вполне нормально, потому, что пользователь newuser может делать только выборку из таблицы.

Усложним задачу. Дадим ‘newuser’ право изменять данные в таблице orders базы данных gruber. Для этого нужно вернуться обратно под root:

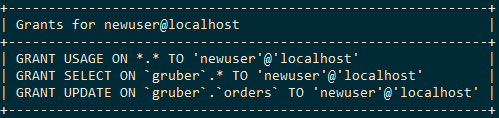
mysql -uroot –p

GRANTUPDATEONgruber.ordersTO 'newuser'@'localhost';

FLUSHPRIVILEGES;

Посмотрим, что получилось:

SHOW GRANTS FOR 'newuser'@'localhost';



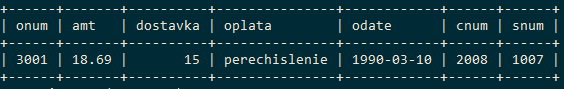
Возвращаемся под пользователя ‘newuser’:

mysql -unewuser –p

usegruber;

Попробуем сделать SELECT для таблицы orders:

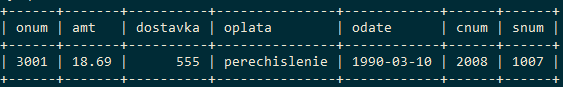
SELECT \* FROM orders LIMIT 1;



SELECT выдаёт результат. Теперь проработаемUPDATE:

UPDATE orders SET dostavka='555' WHERE onum='3001';

SELECT \* FROM orders WHERE onum='3001';



Всё работает. Поле dostavka изменилось с 15 до 555.

Сделаем изменения в таблицу customers:

UPDATE customers SET city='hacked' AND rating='-500' AND cname='NONAME';

MySQL выдаст ошибку:

ERROR 1142 (42000): UPDATE command denied to user 'newuser'@'localhost' for table 'customers'.

Если вы получили эту ошибку, то значит все хорошо.

**Часть 3 Удаление прав пользователям**

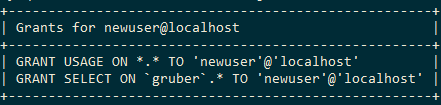
Удалим право наUPDATEтаблице orders;

REVOKEUPDATEONgruber.ordersFROM 'newuser'@'localhost';

FLUSHPRIVILEGES;

Посмотрим права пользователя:

SHOW GRANTS FOR 'newuser'@'localhost';



Теперь зайдем и проверим права:

mysql -u newuser–p

use gruber;

UPDATE orders SET dostavka='15' WHERE onum='3001';

Получим ошибку:

ERROR 1142 (42000): UPDATE command denied to user 'newuser'@'localhost' for table 'orders'

Это значит, что ‘newuser’ не имеет прав на редактирование и мы правильно удалили права

Удалите самостоятельно пользователя ‘newuser’.

Поздравляем! Вы научились создавать пользователей и давать права.

**Индивидуальное задание:**

1. Создать пользователя под именем ‘gruber\_user’ и паролем ‘12345’
2. Дать права на выборку из таблицы orders и salespeople.
3. Сделать выборку из таблицы salespeople
4. Дать право на вставку значений в таблицу orders
5. Вставить строку в таблицу orders
6. Удалить право делать выборку из таблицы orders
7. Дать пользователю право на обновление таблицы customers
8. Убрать право на обновление таблицы customers;
9. Удалить пользователя ‘gruber\_user’

**4 Контрольно-оценочные материалы для промежуточной аттестации по учебной дисциплине**

Предметом оценки являются умения, знания, практический опыт и дополнительные профессиональные компетенции. Контроль и оценка осуществляются в форме дифференцированный зачета.

К сдаче дифференцированный зачета допускаются студенты имеющие зачёты по всем практическим работам и положительные оценки за письменные проверочные занятия.

Оценка освоения дисциплины предусматривает использование пятибалльной системы оценивания.

Оценка при проведении дифференцированный зачета ставится при условии:

* выполнения полного объёма практических работ;
* при наличии оценок за письменные-самостоятельные занятия по каждой пройденной теме;
* либо, в случае не имения письменных работ по каждой пройденной теме, при ответе на пять вопросов, вынесенных на дифференцированный зачет.

**4.1 Пример задания для дифференцированного зачета**

**Билет 2Грузоперевозки**

**Исходные данные:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Перевозка**(8 записей) | | Дата отправления  Адрес «откуда»  Адрес «куда»  Дата прибытия  Дата формирования заявки  Название груза  Общая стоимость поездки  Категория перевозки («Город», «Город выходные», «Область», «Россия»)  Стоимость поездки в зависимости от категории  Единица измерения перевозки (км., часы – зависит от категории перевозки) | | |  | | --- | | **Перевозчик**(1 запись) | | Название компании перевозчика  Адрес компании перевозчика  Телефон компании перевозчика  Платёжные реквизиты компании перевозчика |  |  | | --- | | **Водитель**(3 записи) | | ФИО водителя | | Телефон |  |  | | --- | | **Заказчик**(5 записей) | | Название заказчика  ФИО контактного лица заказчика  Номер телефона заказчика |  |  | | --- | | **Автомобиль**(3 записи) | | Марка автомобиля  Модель автомобиля  ГОС номер автомобиля | |

**Задание:**

1. Создать базу данных
2. Провести нормализацию таблиц
3. Создать таблицы
4. Определить «ключи» в таблицах исходя из логической модели базы данных
5. Реализовать связи между таблицами
6. Заполнить таблицы
7. Реализовать запросы к базе данных и проверить правильность их выполнения
   1. Показать ФИО заказчика, который осуществлял поезду из категории «Область» (название категории так же вывести)
   2. Показать сколько поездок осуществил каждый водитель (ФИО водителя так же вывести)
   3. Показать заказчика, который заказывал такси больше всех (ФИО заказчика так же показать)
   4. Показать водителя, который принёс большую прибыль компании (ФИО водителя так же вывести)
   5. Показать заказчика «А», которого перевозил автомобиль «Б» (ФИО заказчика и ГОС номер автомобиля так же вывести)

**Регламент** проведения зачета по ОП.08

1. Все начинают выполнение занятия одновременно.
2. Опоздавшим студентам время на выполнение занятия НЕ добавляется.
3. Телефоны сдаются при входе в аудиторию.
4. На ПК не будет доступа в интернет, в локальную сеть колледжа и практическим работам по дисциплине.
5. Для реализации задания можно использовать любое ПО, отвечающее требованиям дисциплины. Можно работать на своём ноутбуке.
6. На реализацию задания отводится 120 минут.
7. Выйти из аудитории можно, но время на выполнение занятия НЕ добавляется.
8. По истечению времени все студенты покидают аудиторию. После проверки задания студенты возвращаются для ознакомления с оценкой.
9. Для проверки выполненного задания все результаты запросов необходимо сохранять в представления, а сами Select’ы в текстовый документ, который должен быть открытым и сохранён на рабочем столе.
10. Работа оценивается по критериям. Студент имеет право присутствовать при проверке занятия. Преподаватель, при проверке занятия, имеет право пригласить студента, если в этом будет необходимость.
11. Результат оглашается после проверки всех работ или сразу, если студент присутствовал при проверке.

**Критерии оценки**

|  |  |
| --- | --- |
| **Этап задания** | **Балл** |
| Создание базы данных **со всеми** таблицами | 1 – все таблицы  0 – не полная БД |
| Нормализация таблиц и устранение аномалий | 1 – нормализация проведена, аномалий нет  0,5 – есть незначительные ошибки (не более двух)  0 – нормализация не проведена или ошибок более двух |
| Определение «ключей» в таблицах | 1 – ключи установлены правильно и позволяют обеспечить целостность базы данных  0,5 – ключи определены не везде, но не более двух ошибок (это не относиться к «первичным ключам» – таблицы должны соответствовать первой нормальной форме)  0 – частично отсутствуют первичные ключи или более двух ошибок во «внешних ключах». |
| Реализованы связи между таблицами | 1 – связи, для реализации целостности данных базы данных, установлены все.  0,5 – отсутствует одна связь  0 – отсутствует более одной связи |
| Таблицы заполнены по заданию | 1 – таблицы заполнены по заданию  0 – таблицы не заполнены или заполнены частично |
| Запрос реализован правильно | каждый правильно реализованный запрос 2 балла  Запрос является правильным, если выданный результат соответствует с заданием |
| Максимальное кол-во баллов | 15 |

Оценка «5» – 15 – 13 баллов

Оценка «4» – 12 – 11 баллов

Оценка «3» – 10 – 8 баллов

Оценка «2» – 7 и менее 7 баллов

Лист согласования

**Дополнения и изменения к комплекту ФОС на учебный год**

Дополнения и изменения к комплекту ФОС на \_\_\_\_\_\_\_\_\_\_ учебный год по дисциплине Наименование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

В комплект ФОС внесены следующие изменения:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дополнения и изменения в комплекте ФОС обсуждены на заседании ЦК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Протокол № \_\_\_\_\_\_ от «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_г.

Председатель ЦК\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_