МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Нижегородский государственный университет им. Н.И. Лобачевского**

**Е.А.Неймарк**

**Методические указания для выполнения лабораторных работ по курсу «Базы Данных»**

Учебно-методическое пособие

Рекомендовано методической комиссией ИИТММ  
для студентов ННГУ, обучающихся по направлению подготовки   
090403 «Прикладная информатика»

Нижний Новгород

2017

#### УДК 004.6

#### Н-45

**Н-45 Неймарк Е.А** Методические указания для выполнения лабораторных работ по курсу «Базы Данных». Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. –  68с.

Рецензент: д. т. н., профессор Федосенко Ю.С.

Данное учебно-методическое пособие представляет собой дидактический материал для выполнения лабораторных работ и создания отчета по ним. В пособии представлен вариант задания аналогичный заданиям для выполнения лабораторных работ. Приведено подробное описание методики выполнения каждой лабораторной работы и представлен вид отчета, являющегося результатом выполнения работы.

В результате выполнения лабораторных работ, студенты приобретают навыки анализа предметной области, построения концептуальной модели, даталогического и физического проектирования баз данных. А также учатся решать задачи, возникающие при работе с базой данных, среди них: создание таблиц, выполнение запросов, анализ данных для построения таблиц типа View, написание процедур и функций, а также учатся реализовывать бизнес-логику предметной области при помощи триггеров.

Пособие предназначено для студентов ИИТММ направления подготовки «Прикладная информатика», изучающих курс «Базы данных».

Ответственный за выпуск:

зам. председателя методической комиссии ИИТММ ННГУ,

к.т.н., доцент **В.М. Сморкалова**

#### УДК 004.6

**© Нижегородский государственный**

**университет им. Н.И. Лобачевского, 2017**

**© Неймарк Е.А.**

Введение

В данной учебно-методической разработке представлен вариант задания для выполнения лабораторных работ и подробное описание его выполнения для каждой из лабораторных работ.

Целями выполнения лабораторных работ по курсу «Базы данных» являются получение навыков анализа предметной области, построения концептуальной модели, даталогического и физического проектирования баз данных. А также навыков решения практических вопросов при работе с базой данных, а именно: построение и выполнение запросов, анализ данных для построения таблиц типа View, написание процедур и функций, и реализации бизнес-логики при помощи триггеров.

При выполнении заданий студенты применяют и закрепляют знания, полученные на лекционных и практических занятиях по курсу Базы Данных.

В разработке представлено задание для выполнения лабораторных работ. Для каждой лабораторной работы подробно описано выполнение каждого из заданий работы, а также структура и вид отчета по этой лабораторной работе.

В первой лабораторной работе приводится пример анализа предметной области, а также построения инфологической, даталогической и физической модели базы данных.

Во второй лабораторной работе закрепляется навык построения и выполнения различных запросов.

Третья лабораторная работа способствует приобретению навыков создания и работы с View, Trigger, а также навыков по написанию и использованию сохраняемых процедур и функций.

По каждой лабораторной работе студент должен представить отчет о выполнении. Структура отчета и вариант оформления отчета по поставленному заданию также представлены в методической разработке.

Описание задания.

***Вариант. Пекарня***

**Описание предметной области.**

Пекарня производит выпечку и продает ее к клиентам, согласно предварительным заказам. Необходимо составить БД для учета заказов и проданной продукции. Каждый продукт имеет вес и цену. Кроме того, должен вестись учет клиентов, для каждого клиента хранится его имя, адрес и номер телефона. В заказе указываются имя заказчика, вид заказанной продукции, дата приема заказа и дата исполнения заказа.

**Таблицы.**

Продукция (Название, Вес, Цена)

Заказчик (ФИО, Телефон, Адрес)

Заказ (Клиент, Продукция, Дата приема заказа, дата исполнения заказа)

**Развитие постановки задачи.**

В ходе работы выяснилось, что часто клиенты заказывают несколько видов продукции одновременно и(или) покупают несколько штук продукции одного вида. Это нужно оформлять в одном заказе. Кроме того, пекарня открыла несколько филиалов, теперь в заказе нужно указывать какой филиал принял и исполнил данный заказ. Ассортимент в каждом из филиалов один.

**Запросы**

1. Вывести какое количество каждого вида продукции было заказано в каждом из филиалов за последний месяц.
2. Вывести суммарное количество продукции, проданной каждым филиалом по месяцам за текущий год.
3. Вывести суммарное количество заказов, выполненное каждым филиалом по месяцам за текущий год.
4. Вывести те филиалы, которые выполнили максимальное количество заказов в каждом месяце за последние полгода.
5. Для каждого филиала вывести общее число заказов, средний чек заказа, минимальную сумму заказа за каждый месяц.

**View**

1. Для каждого заказчика вести историю заказов в следующем виде: имя клиента, дата первого заказа, дата последнего заказа, виды заказанной продукции и общее их количество через запятую в формате: «продукция1 – количество, продукция2 – количество…», общая сумма заказов.
2. Для каждого филиала вывести общее число заказов, средний чек заказа, минимальную сумму заказа за каждый месяц.

**Триггер**

В таблице Заказчик добавить поле сумма заказов, при добавлении нового заказа для данного заказчика, обновлять это поле с учетом нового заказа.

Задания для лабораторной работы №1

**Структура отчета**

1. **Начальная постановка задачи**
2. **Инфологическая модель**

Выделить сущности, связи (с кратностями).

Представить диаграмму сущность-связь

1. **Даталогическая модель.**
2. **Определить атрибуты и их тип**
3. **Определить первичный ключ.**
4. **Определить внешний ключ (если он есть)**
5. **Определить обязательные для заполнения поля.**
6. **Определить значение по умолчанию для некоторых полей.**
7. **Представить даталогическую модель в виде схемы**
8. **Физическая модель**
   1. **Создание таблиц**

Предоставить команды создания таблиц (копии экрана).

* 1. **Структура таблиц**

Вывести список таблиц в БД.

Вывести структуру таблиц.

* 1. **Добавление данных**

Предоставить команды ввода данных в таблицы   
(копии экрана).

* 1. **Содержимое таблиц**

Вывести полное содержимое каждой из таблиц

1. **Развитие постановки задачи**

**Повторить пункты 1-3**

1. **Анализ выполнения условий нормализации.**

Какой НФ соответствует БД, обосновать.

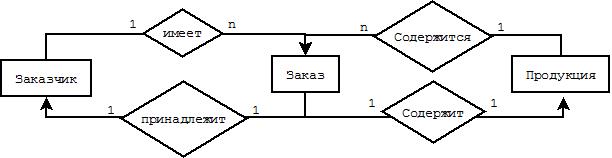
При создании физической модели предоставлять команды создания новых таблиц и изменения существующих. Выводить структуру и содержимое для новых и измененных таблиц.

Выполнение лабораторной работы №1

1. **Проанализировать предметную область (варианты предметных областей прилагаются).**

На данном этапе нам нужно построить инфологическую и даталогическую модели предметной области. Основные сущности уже выделены: это Заказчик, Заказ и Продукция. Каждый заказ принадлежит только одному заказчику и имеет только один вид продукции. При этом у каждого заказчика может быть множество заказов, каждый вид продукции может содержаться в нескольких заказах.

Отобразим графически сущности и их отношения, это инфологическая модель (Рис. 1).



*Рис. 1. Диаграмма Сущность-Связь.*

Теперь надо провести дальнейший анализ с целью построения **даталогической** модели. Для этого зададим все атрибуты таблицы и их типы. Определим первичные ключи и внешние ключи для каждой из таблиц.

* 1. **Определить первичный ключ.**

Определим в качестве первичных ключей ID\_заказчика, ID\_заказа, ID\_продукта в таблицах Заказчик, Заказ и Продукт соответственно.

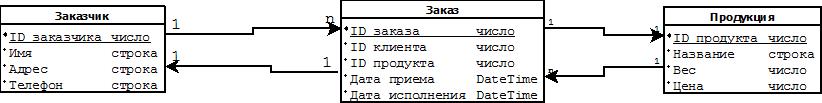
* 1. **Определить обязательные для заполнения поля.**

Все поля требуют обязательного заполнения.

* 1. **Определить значение по умолчанию для некоторых полей.**

Для полей Дата приема и Дата исполнения по умолчанию будем проставлять текущую дату и время.

Полученная даталогическая модель представлена на *Рис. 2*



*Рис. 2. Даталогическая модель*

1. **Произвести физическое моделирование в рамках реляционной модели данных. Создать таблицы согласно представленному описанию, таблицы обязательно должны содержать перечисленные поля.**
2. **Для каждой создаваемой таблицы:**
   1. **Определить первичный ключ.**
   2. **Определить внешний ключ (если он есть).**
   3. **Определить обязательные для заполнения поля.**
   4. **Определить типы полей в зависимости от требований предметной области.**
   5. **Определить значение по умолчанию для некоторых полей.**
   6. **Ввести данные в таблицы.**
   7. **При вводе выяснить, что дает наложение условий на значения полей.**

Строим физическую модель данных, основываясь на построенной даталогической модели.

Первичные ключи в каждой таблице уже определены. Внешние ключи будут только в таблице Заказ. Один внешний ключ к таблице Заказчик, другой к таблице Продукция.

В даталогической модели мы определили типы полей, в физической модели мы прописываем те типы, которые доступны в рамках применяемого языка (MySQL).

Поскольку для Даты приема и Даты исполнения по умолчанию ставится текущая дата и время, то выберем ее реализацию в типе TIMESTAMP.

Строим таблицы.

**CREATE** TABLE **Client (**

ID\_client INT NOT NULL AUTO\_INCREMENT,

Name CHAR(20) NOT NULL,

Address VARCHAR(30) NOT NULL,

Phone CHAR(15) NOT NULL,

PRIMARY KEY (ID\_client));

CREATE TABLE Order (

ID\_order INT(10) NOT NULL AUTO\_INCREMENT,

ID\_client INT(10) NOT NULL,

ID\_prod INT(10) NOT NULL,

order\_date TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

exec\_date TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (ID\_order));

CREATE TABLE Product (

ID\_prod INT(10) NOT NULL AUTO\_INCREMENT,

Name CHAR(25) NOT NULL,

Weight INT(4) UNSIGNED NOT NULL,

Price INT(4) UNSIGNED NOT NULL,

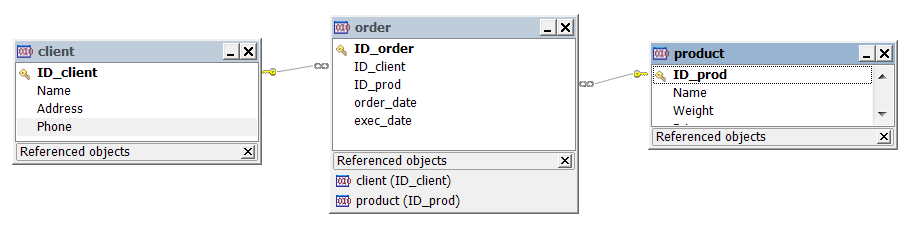
PRIMARY KEY (ID\_prod));

Добавляем внешние ключи к таблице Order**.**

ALTER TABLE Order

ADD CONSTRAINT fk\_client FOREIGN KEY (ID\_client) REFERENCES client (ID\_client) ON UPDATE RESTRICT ON DELETE RESTRICT,

ADD CONSTRAINT fk\_prod FOREIGN KEY (ID\_prod) REFERENCES product (ID\_prod) ON UPDATE RESTRICT ON DELETE RESTRICT;

****

*Рис. 3. Физическая модель*

Добавляем данные.

В этой команде используется классический вид оператора, когда все имена полей прописаны, добавляемые кортежи идут в скобках, через запятую.

INSERT INTO client

(ID\_client,Name ,Address ,Phone)

VALUES

(1, 'Шашкин А.К.', 'Паровозный пер. 10,12', '1235790'),

(2, 'Малклкова А.Е.', 'Ленина 24,36', '987234452'),

(3, 'Шаповаленко М.П.', 'Григорьева, 34,2', '987245');

Если мы при вставке передаем значения всех полей, то их имена можно не прописывать в команде.

INSERT INTO client VALUES

(4, 'Кузнецова А.Н.', 'Покровская, 7,15', '656234'),

(5, 'Станков А.Р.', 'Выборгская, 35/7,12', '765234'),

(6, 'Бояринова К.У.', 'Мостовая 21,2', '4353468');

Если поле указано как AUTO\_INCREMENT (что справедливо для поля ID\_prod в этой таблице), то его значение можно не прописывать в команде INSERT, тогда оно будет сгенерировано автоматически.

Если при вставке в кортеже пропущены значения каких-либо полей, то имена всех вставляемых полей необходимо прописать.

INSERT INTO product ( Name ,Weight ,Price)

VALUES

('Пирог с Капустой', 1000, 400),

('Кулебяка', 500, 750),

('Пирог с луком и яйцом', 1000, 700),

('Пирог с капустой', 2000, 700),

('Пирог с вишней', 500, 400),

('Пирог с мясом', 1000, 1200),

('Пирог с малиной', 500, 450),

('Рыбник', 500, 600),

('Ватрушка', 500, 400);

Варианты заполнения таблицы с полями по умолчанию.

В таблице order у нас поля order\_date и exec\_date имеют значения по умолчанию (по умолчанию туда заносится время заполнения строки), кроме того, поле ID\_order указано как AUTO\_INCREMENT, следовательно, при вставке его также можно опустить.

INSERT INTO order (ID\_client ,ID\_prod )

VALUES (4, 5),(6,9), (3,7),(2,1);

Порядок полей можно менять, но при этом должно быть строгое соответствие между перечислением имен полей и таким же содержимым в добавляемых кортежах.

INSERT INTO order (exec\_date, ID\_client ,ID\_prod )

VALUES ('14.05.2015', 3,9);

Важно отметить, что значение, которое вставилось в поле exec\_date, «20.05.2014 15:00:00», поскольку формат TIMESTAMP «YYMMDDHH». Для достижения желаемого результата надо вставлять в следующем виде.

INSERT INTO order (exec\_date, ID\_client ,ID\_prod )

VALUES ('15.05.14', 3,9);

Следующее добавление вызывает ошибку, поскольку у нас нет клиента с номером 7. Это проверка на работу внешних ключей.

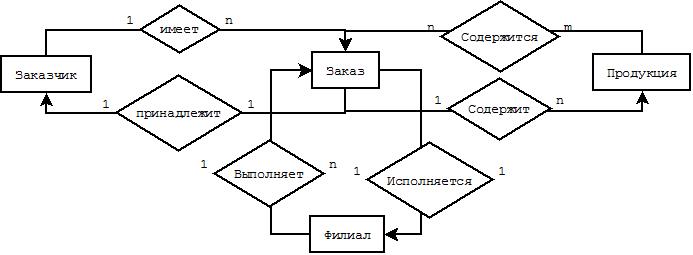
INSERT INTO order (ID\_client, ID\_prod ) VALUES (7, 5);

1. **Проанализировать предметную область для описанного развития постановки задачи. Какие изменения необходимо внести в существующую модель.**

В новой постановке в одном заказе может содержаться несколько видов продукции, следовательно, измениться кратность связи Заказ-Продукт. Появилась новая сущность Филиал.

1. **Выделить сущности, проследить связи между сущностями (инфологическая модель).**

Строим инфологическую модель. В построенной ранее модели, изменяем кратность связей Продукция – Заказ и Заказ – Продукция. Добавляем новую сущность Филиал. Поскольку ассортимент в каждом из филиалов одинаковый, то связи у Филиала будут только с Заказом.

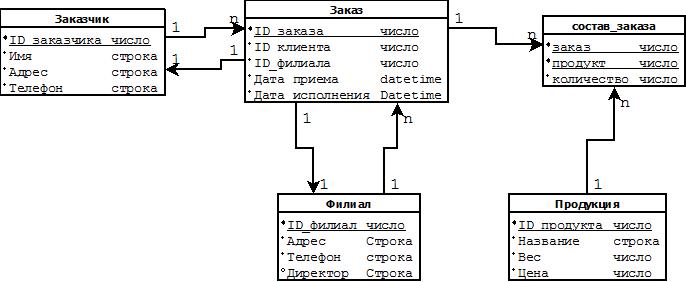
****

*Рис. 4. Инфологическая модель развития постановки задачи*

1. **Проанализировать какие атрибуты есть у сущностей, построить даталогическую модель, внести соответствующие изменения в структуру таблиц (в физическую модель), заполнить их данными.**

Новую даталогическую диаграмму строим на основании исходной. В ней изменяем кратность связей Продукция – Заказ и Заказ – Продукция. Добавим в Заказ новое поле ID\_филиала.

Поскольку в нашей инфологической модели имеется отношение многие ко многим, следовательно, в даталогической модели и физической реализации в рамках реляционной модели, необходимо добавить еще одну таблицу, отражающую связь, между заказом и продукцией. При этом из таблицы Заказ убираем поле, указывающее вид продукта, и переносим его в новую таблицу. Также в этой таблице будем отражать какое количество продукции каждого вида заказано. Новая таблица будет иметь внешние ключи к таблицам Заказ и Продукция.



*Рис. 5. Даталогическая модель развития постановки задачи*

Дополнительно определяем атрибуты и первичный ключ для новой сущности Филиал. Поскольку в представленной постановке задачи ничего не говориться о данных, которые должны быть предоставлены для филиала, предположим, что для каждого филиала будем указывать его телефон и адрес, а также фамилию директора, если таковой имеется. Установим, что поля Директор и Телефон не обязательны к заполнению, поле Директор может содержать пустое значение, поле Телефон по умолчанию заполняется номером справочной (один номер для всех филиалов).

Кроме того, в таблице Заказ добавится один внешний ключ к таблице Филиал. А также удалится поле и внешний ключ, отражающий связь между заказом и продукцией.

Построим физическую модель (таблицы) и заполним их новыми данными.

Строим две новые таблицы.

CREATE TABLE point (

Id\_point int(10) NOT NULL AUTO\_INCREMENT,

Address char(25) CHARACTER SET latin1 NOT NULL,

Phone char(15) CHARACTER SET latin1 DEFAULT '80023456',

Director varchar(20) DEFAULT NULL,

PRIMARY KEY (`Id\_point`)

);

CREATE TABLE order-product (

Order\_ID int(10) NOT NULL,

Product\_ID int(10) NOT NULL,

Amount int(4) unsigned NOT NULL,

PRIMARY KEY (Order\_ID,Product\_ID)

) ;

Добавим внешние ключи. Имена внешних ключей можно не указывать, тогда система даст им свои имена.

ALTER TABLE order-product

ADD FOREIGN KEY (Order\_ID) REFERENCES order

(ID\_order) ON UPDATE RESTRICT ON DELETE RESTRICT,

ADD FOREIGN KEY (Product\_ID) REFERENCES product (ID\_prod) ON UPDATE RESTRICT ON DELETE RESTRICT;

Поскольку изначально атрибута Заказ.ID\_филиала у нас не было, то это поле надо создавать с возможностью значения NULL.

ALTER TABLE order

ADD ID\_point INT(10) AFTER ID\_prod;

ALTER TABLE order

ADD FOREIGN KEY (ID\_point) REFERENCES point (Id\_point) ON UPDATE RESTRICT ON DELETE RESTRICT;

После внесения изменения в структуру таблицы и до ее заполнения новыми данными необходимо заполнить это поле, например, значением 1 (будем считать, что первым в таблицу внесен тот филиал, который исполнял раньше все заказы).

Но сначала, надо внести данные в таблицу Филиал.

INSERT INTO point(Address, Director)

VALUES ('Ковалевской 15', 'Федорова Е.А.');

Теперь устанавливаем значения в столбце.

UPDATE order SET ID\_point = 1;

И устанавливаем значение NOT NULL.

ALTER TABLE order

CHANGE ID\_point ID\_point INT(10) NOT NULL;

Заполняем значения в таблице order-product согласно старым данным из таблицы Заказ вручную.

INSERT INTO order-product

(Order\_ID, Product\_ID, Amount) VALUES

(1, 5, 1), (2, 9, 1), (3, 7, 1), (4, 1, 1), (5, 9, 1),

(6, 9, 1);

Или при помощи выборки.

INSERT INTO order-product

(Order\_ID, Product\_ID, Amount)

select ID\_order, ID\_product, 1 from order;

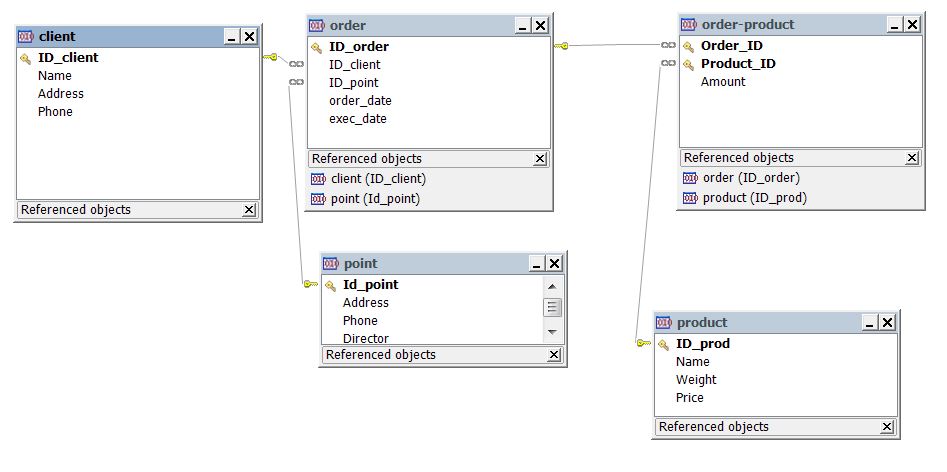
Удаляем внешний ключ и поле для внешнего ключа из таблицы Заказ.

ALTER TABLE `order`

DROP FOREIGN KEY fk\_prod,

DROP ID\_prod;

Получили структуру таблиц, представленную на *Рис. 6*

****

*Рис. 6. Физическая модель развития постановки задачи*

Теперь заполним таблицы.

Вносим новые филиалы.

INSERT INTO point (Address, Phone) VALUES

('Привалова 94', '214456');

INSERT INTO point (Address, Director) VALUES

('Морозова 2', 'Зинякова К.П.');

Добавим новые заказы для новых филиалов.

INSERT INTO `order`

( ID\_client, ID\_point, exec\_date) VALUES

( 1, 2, '29.10.2015 17:07:06'),

( 5, 3, '12.10.2015 17:26:58'),

( 6, 2, '22.10.2015 17:27:30');

Добавим данные в таблицу order-product.

INSERT INTO `order-product`

(Order\_ID, Product\_ID, Amount) VALUES

(7, 5, 1), (7, 7, 2),(7, 9, 1),

(8, 1, 1), (8, 3, 1), (8, 4, 3), (8, 7, 2),

(9, 1, 1), (9, 3, 2), (9, 6, 2);

1. **Анализ выполнения условий нормализации. Какой нормальной форме соответствует построенная база данных.**

Для того, чтобы проверить, соответствует ли наша БД условиям нормализации, вспомним правила, которые должны выполняться для каждой из нормальных форм.

1НФ :

1. Каждый набор данных (кортеж) имеет первичный ключ;
2. В кортеже каждый атрибут атомарен;
3. В кортеже нет повторяющихся однородных данных;

2НФ :

1. В таблице нет зависимости от части ключа;
2. В таблице нет одинаковых наборов значений, относящихся к разным кортежам (это отношение многие ко многим);
3. Для таких наборов созданы отдельные таблицы;
4. Связь между таблицами осуществлена через внешний ключ.

3НФ :

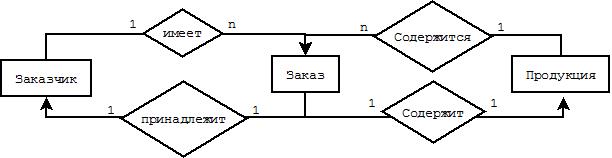
1. Нет полей, зависящих не от первичного ключа, т.е. каждый неключевой столбец независим друг от друга;
2. Если есть поля, зависящие от других атрибутов кортежа, то эти связи или надо вынести в отдельную таблицу, или установить другие зависимости.

Построенная БД соответствует 3НФ, поскольку:

1. Данные все атомарны.
2. Повторяющихся однородных данных в кортежах нет.
3. Каждому кортежу соответствует первичный ключ.
4. Во всех таблицах (кроме Заказ-Продукция) введены уникальные первичные ключи, что исключает зависимость неключевых столбцов от части ключа.
5. Множественная связь Заказ-Продукция выделена в отдельную таблицу. Связь между таблицами Заказ, Заказ-Продукция и Продукция осуществляется через внешние ключи.
6. В таблицах нет полей, не зависящих от первичного ключа.

Отчет по лабораторной работе №1

1. **Начальная постановка задачи**
2. **Инфологическая модель**
   1. Основные сущности: Заказчик, Заказ и Продукция.
   2. Связи: Заказчик(1)-имеет-Заказ(n), Продукция(1)-содержится-Заказ(n), Заказ(1) -принадлежит-Заказчик(1), Заказ(1) -содержит-Продукция(1),
   3. Диаграмма сущность-связь :



1. **Даталогическая модель.**
2. **Определить первичный ключ.**

Заказчик - ID\_заказчика, Заказ - ID\_заказа, Продукт - ID\_продукта.

1. **Определить внешний ключ (если он есть)**

Заказ.ID\_заказчика - Заказчик .ID\_заказчика;

Заказ. ID\_продукта - Продукт .ID\_продукта.

1. **Определить обязательные для заполнения поля.**

Все поля требуют обязательного заполнения.

1. **Определить значение по умолчанию для некоторых полей.**

Дата приема текущая дата и время

Дата исполнения текущая дата и время.

n

n

|  |  |
| --- | --- |
| Заказ | |
| **ID\_заказа** | число |
| ID\_заказчика | число |
| ID\_продукта | число |
| Дата приема | DateTime |
| Дата исполнения | DateTime |

1

1

1

|  |  |
| --- | --- |
| Продукт | |
| **ID\_продукта** | число |
| Название | строка |
| Вес | число |
| Цена | число |

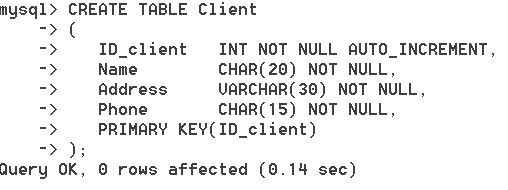
|  |  |
| --- | --- |
| Заказчик | |
| **ID\_заказчика** | число |
| Имя | строка |
| Адрес | строка |
| Телефон | строка |

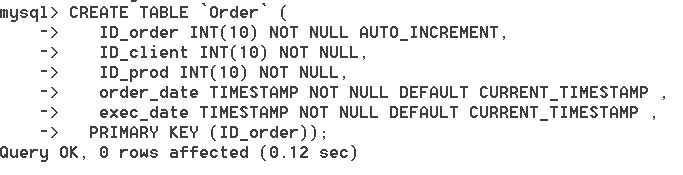
1

1

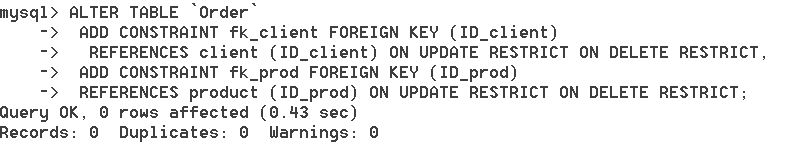
1

1. **Физическая модель**
2. **Создание таблиц**

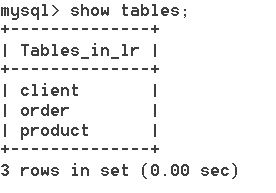
****

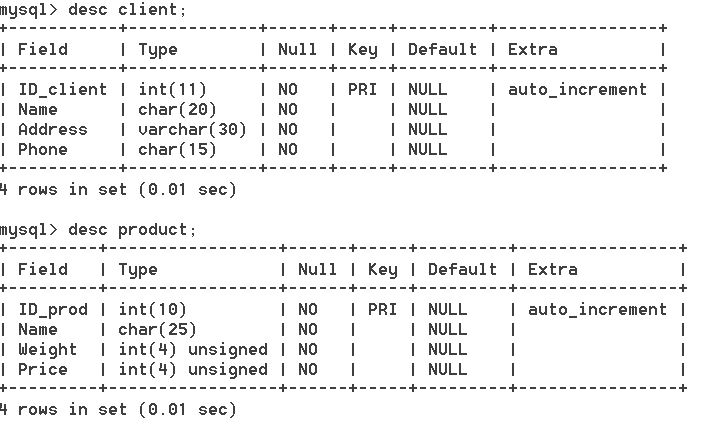
****

****

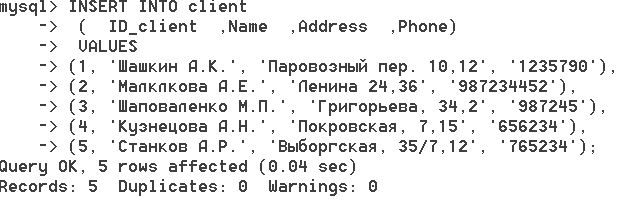
****

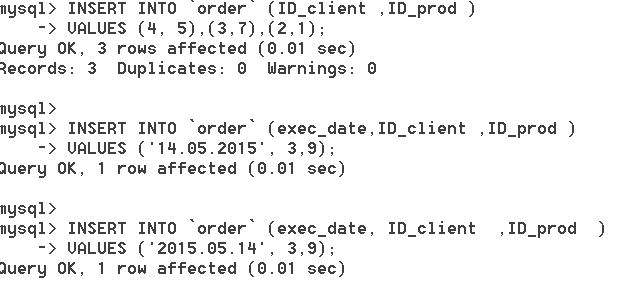
1. **Структура таблиц.**





1. **Заполнение таблиц.**

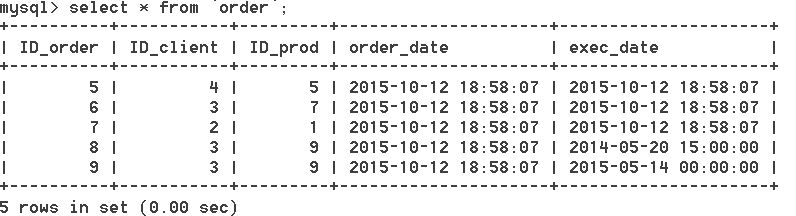


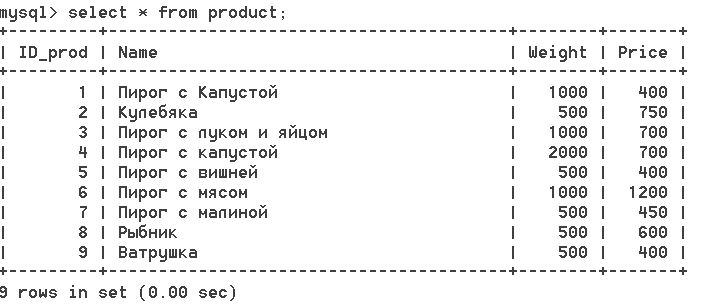


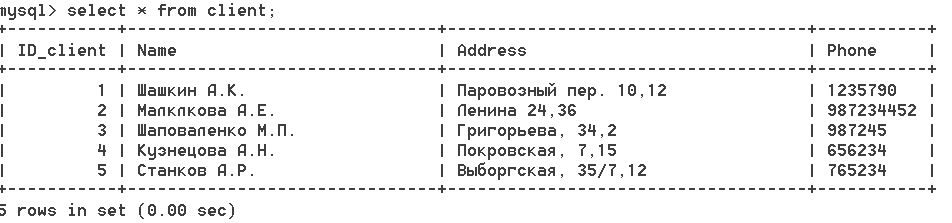
Ошибка при нарушении внешнего ключа.

G:\РАБОТА\Курсы\БД\ЛАБОРАТОРНЫЕ\для примера выполнения ЛР\insert orderErr.JPG

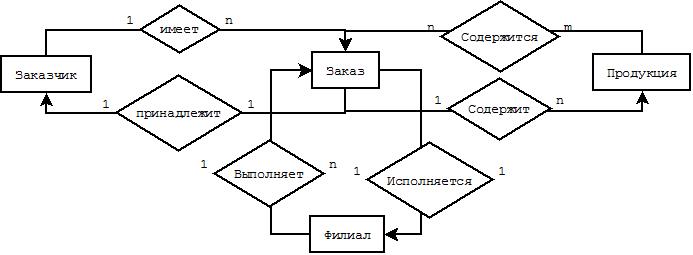
1. **Содержимое таблиц:**







1. **Начальная постановка задачи**
2. **Инфологическая модель**
   1. Основные сущности: Заказчик, Заказ ,Продукция, Филиал.
   2. Связи: Заказчик(1)-имеет-Заказ(n), Продукция(m)-содержится-Заказ(n), Заказ(1) -принадлежит-Заказчик(1), Заказ(1) -содержит-Продукция(n),
   3. Диаграмма сущность-связь

****

1. **Даталогическая модель.**
2. **Определить первичный ключ.**

Заказчик - ID\_заказчика, Заказ - ID\_заказа, Продукт - ID\_продукта.

1. **Определить внешний ключ (если он есть)**

Заказ.ID\_заказчика - Заказчик .ID\_заказчика;

Заказ. ID\_продукта - Продукт .ID\_продукта.

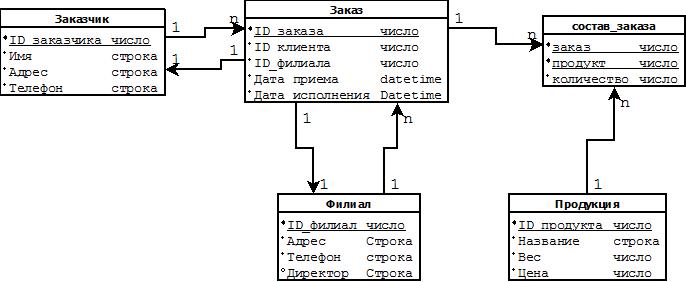
Заказ. ID\_филиала – Филиал. ID\_филиала

1. **Определить обязательные для заполнения поля.**

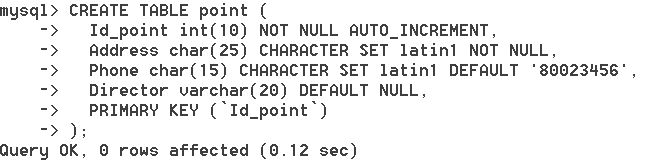
В таблице Филиал все поля, кроме Директор, обязательны для  
 заполнения.

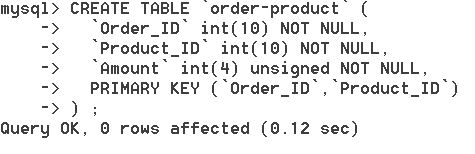
1. **Определить значение по умолчанию для некоторых полей.**

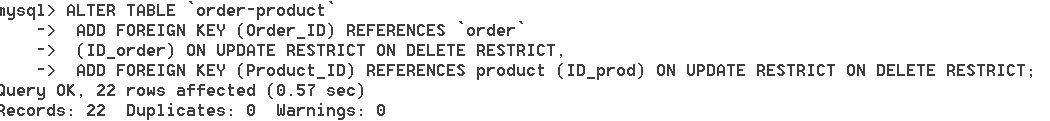
Филиал.Телефон “80023456”

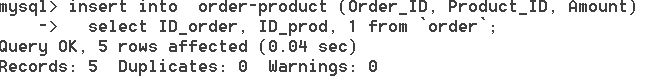


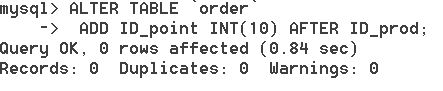
1. **Физическая модель**
   1. **Создание таблиц**

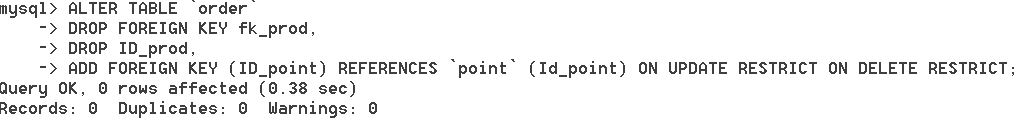


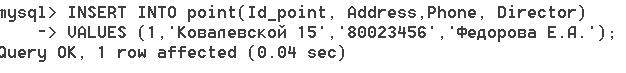


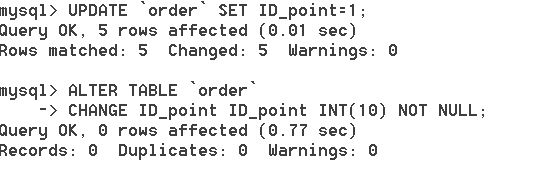




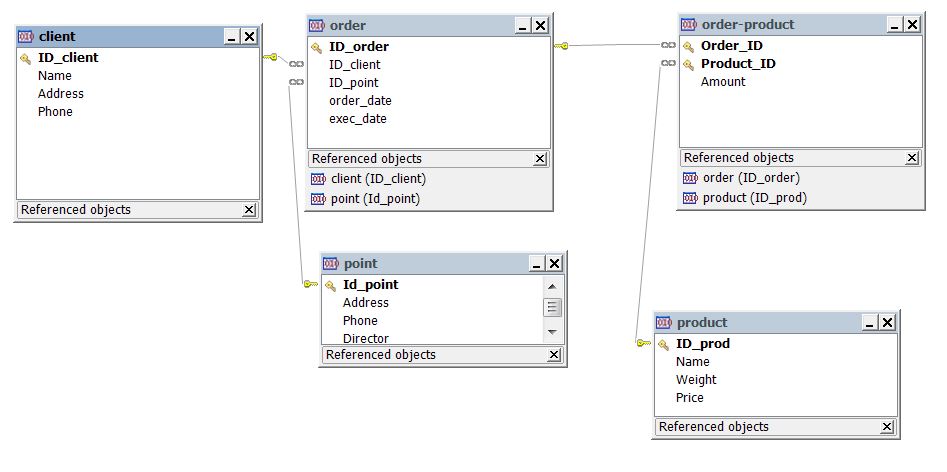


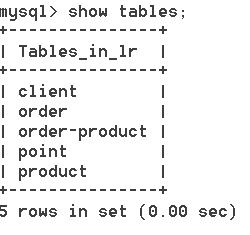


****

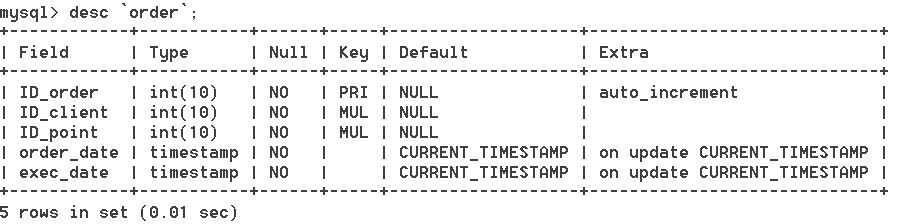
****

* 1. **Структура таблиц**

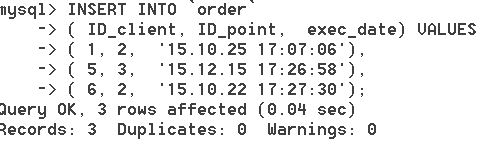
****

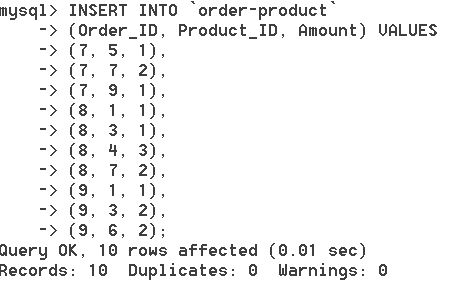


****

****

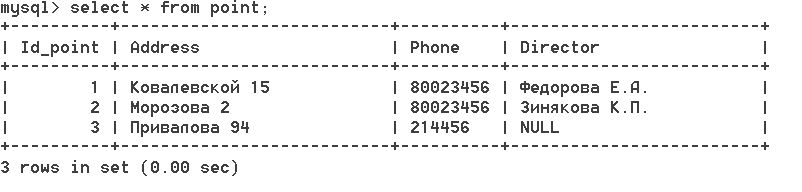
* 1. **Добавление данных**

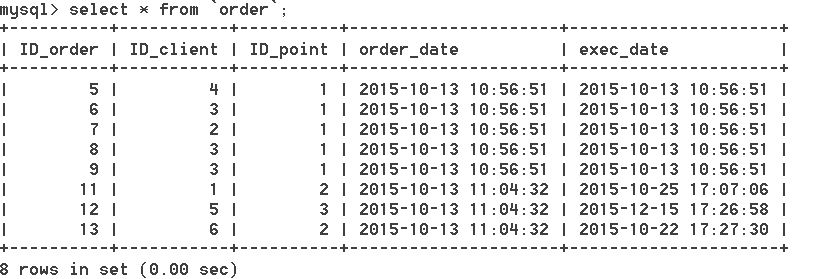


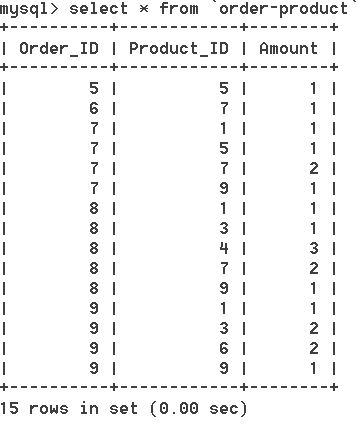


**G:\РАБОТА\Курсы\БД\ЛАБОРАТОРНЫЕ\для примера выполнения ЛР\insert point1.JPGG:\РАБОТА\Курсы\БД\ЛАБОРАТОРНЫЕ\для примера выполнения ЛР\insert point2.JPG**

* 1. **Содержимое таблиц**

****





1. **Анализ выполнения условий нормализации**

Построенная БД соответствует 3НФ, поскольку:

1. Данные все атомарны.
2. Повторяющихся однородных данных в кортежах нет.
3. Каждому кортежу соответствует первичный ключ.
4. Во всех таблицах (кроме Заказ-Продукция) введены уникальные первичные ключи, что исключает зависимость неключевых столбцов от части ключа.
5. Множественная связь Заказ-Продукция выделена в отдельную таблицу. Связь между таблицами Заказ, Заказ-Продукция и Продукция осуществляется через внешние ключи.
6. В таблицах нет полей, не зависящих от первичного ключа.

Задания для лабораторной работы №2

**Структура отчета**

1. Простой выбор данных (select, \*, from, where, order by, вычислимые поля, distinct).
   1. Создать простой запрос на выборку из одной таблицы. Включить несколько полей таблицы. Выбрать несколько полей, по которым сортируется вывод, по одному из полей отсортировать в порядке возрастания, по другому – в порядке убывания.
   2. Выбрать из таблицы строки, содержащие значение NULL.
   3. Выбрать из таблицы неповторяющиеся значения.
   4. Использовать в запросе оператор Like.
   5. Выполнить один запрос с использованием условного оператора ИЛИ и один с использованием условного оператора И.
   6. Выполнить запрос с использованием функций форматирования строк.
   7. Выполнить запрос, в котором одно из полей является результатом арифметической функции над каким-либо полем таблицы.
   8. Выполнить отсортированный по вычислимому полю запрос, для вычислимого поля использовать альтернативное имя.
2. Объединение таблиц в запросе.
   1. Выполнить запрос на объединение связанных таблиц.
   2. Выполнить запрос из нескольких таблиц, использовать в запросе условные операторы И и ИЛИ.
   3. Выполнить запрос, показывающий избыточные данные в одной из связанных по внешнему ключу таблиц.
   4. Объединить в запросе данные разных типов из двух таблиц, не используя соединение таблиц.
3. Выбор данных с помощью группирующих запросов с условием (group by, having, min(), max(), sum(), count(), …).
   1. Выполнить группирующий запрос, содержащий несколько полей с группировкой по одному полю.
   2. Выполнить группирующий запрос с группировкой по нескольким полям.
   3. Выполнить группирующий запрос с соединением таблиц.
   4. Выполнить группирующий запрос с условным оператором, в котором сначала выполняется условный оператор, а затем производится выборка.
   5. Выполнить группирующий запрос с условным оператором, в котором сначала производится выборка, а затем выполняется условный оператор.
   6. Выполнить группирующий запрос, в котором одно из полей получено при помощи функции group\_concаt.
4. Выбор данных с помощью подзапросов.
   1. Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора In.
   2. Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора Any.
   3. Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора All.
   4. Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора Exists.
5. Выполнить запрос, описанный в задании.

**Структура отчета полностью повторяет структуру вопросов. Для каждого из вопросов задания представляются экранные копии команд и результатов их выполнения.**

Выполнение лабораторной работы №2

1. **Простой выбор данных (select, \*, from, where, order by, вычислимые поля, distinct).**
   1. **Создать простой запрос на выборку из одной таблицы. Включить несколько полей таблицы. Выбрать несколько полей, по которым сортируется вывод, по одному из полей отсортировать в порядке возрастания, по другому – в порядке убывания.**

Сделаем выборку полей Название, Вес и Цена из таблицы Продукция. Отсортируем по полю Вес в порядке возрастания, по полю Цена в порядке убывания.

SELECT Name, Weight, Price

FROM product

ORDER BY Price DESC, Weight ASC;

* 1. **Выбрать из таблицы строки, содержащие значение NULL.**

В созданной БД только поле Директор из таблицы Филиал не обязательно к заполнению и не имеет значения по умолчанию, следовательно, в запросе будем выводить именно это поле.

SELECT Address, Phone

FROM `point`

WHERE Director IS NULL;

* 1. **Выбрать из таблицы неповторяющиеся значения.**

Сначала надо выбрать таблицу, в которой есть повторяющиеся значения, затем сделать выборку без повторов. В нашей БД в таблице Продукция есть повторяющиеся названия. Выведем данные без повторений.

SELECT DISTINCT Name FROM product;

* 1. **Использовать в запросе оператор Like.**

В этом запросе надо использовать выбор поля, имеющего строковое значение. Выберем все строки из таблицы Продукция, в которых в названии есть подстрока “<пробел>с<пробел>”.

SELECT Name, Price, Weight

FROM product

WHERE Name LIKE "% с %";

* 1. **Выполнить один запрос с использованием условного оператора И и один с использованием условного оператора ИЛИ.**

В первом запросе выберем все строки из таблицы Продукция, в которых Цена меньше 1000 и Вес больше 500.

SELECT Name, Price, Weight

FROM product

WHERE Weight > 500 AND Price < 1000;

Во втором запросе выберем все строки из таблицы Продукция, в которых Цена меньше 1000 или Вес меньше 1000.

SELECT Name, Price, Weight

FROM product

WHERE Weight < 1000 OR Price < 1000;

* 1. **Выполнить запрос с использованием функций форматирования строк.**

При форматировании можно использовать как строковые поля, так и любые другие. В данном случае используем функцию concat(), в которой соединим строковые и числовые поля.

SELECT concat(Name, " адрес ", Address, ", телефон ",Phone)

FROM client;

SELECT concat(Name, " цена ",Price, "руб, вес ",Weight, "гр")

FROM product;

* 1. **Выполнить запрос, в котором одно из полей является результатом арифметической функции над каким-либо полем таблицы.**

Вычислим стоимость 100 грамм продукции.

SELECT Name, (Price / Weight) \* 100 FROM product;

* 1. **Выполнить отсортированный по вычислимому полю запрос, для вычислимого поля использовать альтернативное имя.**

Используем предыдущий запрос и изменим его согласно требованиям. Дадим вычислимому полю альтернативное имя – Порция и отсортируем по нему.

SELECT Name, (Price / Weight) \* 100 AS Portion

FROM product

ORDER BY Portion;

1. **Объединение таблиц в запросе.**
   1. **Выполнить запрос на объединение связанных таблиц.**

В этом запросе объединим таблицы Заказчик и Заказ, выведем имена клиентов и даты приема и исполнения заказов. Для правильного выполнения запроса, в условии выборки необходимо указывать поля, по которым происходит соединение таблиц.

SELECT client.Name, `order`.order\_date, `order`.exec\_date

FROM client, `order`

WHERE `order`.ID\_client = client.ID\_client;

* 1. **Выполнить запрос из нескольких таблиц, использовать в запросе условные операторы И, ИЛИ.**

Используем предыдущий запрос в качестве базового, добавим к нему условие приема заказа после 13 октября 2015г или фамилия клиента должна начинаться на «Ш».

SELECT client.Name, `order`.order\_date, `order`.exec\_date

FROM client, `order`

WHERE `order`.ID\_client = client.ID\_client

AND (`order`.order\_date > "15.10.13" OR client.Name LIKE "Ш%");

* 1. **Выполнить запрос, показывающий избыточные данные в одной из связанных по внешнему ключу таблиц.**

Данный запрос поможет выявить данные в родительской таблице (на нее указывает внешний ключ), которые не входят в дочернюю (в которой есть внешний ключ).

Например, какие из клиентов занесены в базу, но не сделали ни одного заказа. Добавим предварительно избыточные данные в таблицу Заказчик.

SELECT name

FROM client LEFT JOIN `order` USING (ID\_client)

WHERE ID\_order IS NULL;

Или какие из видов продукции не были ни разу заказаны.

SELECT name

FROM Product LEFT JOIN `order-product` ON (ID\_prod = product\_ID)

WHERE product\_ID IS NULL;

* 1. **Объединить в запросе данные разных типов из двух таблиц, не используя соединение таблиц**

Используя оператор UNION, можно объединять в запросе разнородные данные, при этом соединения таблиц не происходит.

Выведем Имя, Адрес и Телефон из таблицы Заказчик и Имя, Вес и Цену из таблицы Продукция, из каждой таблицы выведем по 5 строк.

SELECT client.Name, Address, Phone

FROM client

LIMIT 5

UNION

SELECT product.Name, Weight, Price

FROM product

LIMIT 10;

1. **Выбор данных с помощью группирующих запросов с условием (group by, having, min(), max(), sum(), count(), …).**

Группировка производится для вычисления при помощи агрегатных функций или для использования других функций на группировках (например, group\_concat). Использование агрегатных функций без группировки приводит к ошибкам.

* 1. **Выполнить группирующий запрос, содержащий несколько полей с группировкой по одному полю.**

Выведем среднюю цену продукции одного наименования.

SELECT name, avg(Price)

FROM product

GROUP BY name;

* 1. **Выполнить группирующий запрос с группировкой по нескольким полям.**

Выведем количество заказов, сделанных каждым из клиентов в каждом офисе.

SELECT ID\_client, ID\_point, count(\*)

FROM `order`

GROUP BY ID\_client, ID\_point;

* 1. **Выполнить группирующий запрос с соединением таблиц.**

Для каждого клиента и офиса продаж выведем заказы с минимальной датой приема заказа и максимальной датой исполнения заказа, выводя при этом имя клиента.

SELECT name,

ID\_point,

max(exec\_date),

min(order\_date)

FROM `order`, client

WHERE `order`.ID\_client = client.ID\_client

GROUP BY name, ID\_point;

* 1. **Выполнить группирующий запрос с условным оператором, в котором сначала выполняется условный оператор, а затем производится выборка.**

Используем предыдущий запрос в качестве базового, выберем только те строки, у которых Дата выполнения позже 13 октября 2015.

SELECT name,

ID\_point,

max(exec\_date),

min(order\_date)

FROM `order` JOIN client USING (ID\_client)

WHERE `order`.exec\_date > "2015.10.13"

GROUP BY name, ID\_point;

* 1. **Выполнить группирующий запрос с условным оператором, в котором сначала производится выборка, а затем выполняется условный оператор.**

Здесь требуется сделать выборку из уже полученных в результате выполнения запроса данных, следовательно, надо использовать оператор HAVING. Возьмем запрос из п.3.3 в качестве базового и выберем только строки, в которых max(exec\_date) позже 13 октября 2015.

SELECT name,

ID\_point,

max(exec\_date),

min(order\_date)

FROM `order` JOIN client USING (ID\_client)

GROUP BY name, ID\_point

HAVING max(exec\_date) > "2015.10.13";

* 1. **Выполнить группирующий запрос, в котором одно из полей получено при помощи функции group\_concаt.**

Для каждого клиента выведем содержание всех его заказов.

SELECT client.Name,

group\_concat(product.Name, " ", `order-product`.Amount, ";")

FROM client, `order`,

`order-product`, product

WHERE client.ID\_client = `order`.ID\_client

AND `order`.ID\_order = `order-product`.Order\_ID

AND `order-product`.Product\_ID = product.ID\_prod

GROUP BY `order`.ID\_client;

1. **Выбор данных с помощью подзапросов.**
   1. **Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора IN.**

Выведем всех клиентов, которые делали заказы во втором филиале.

SELECT Name

FROM client

WHERE ID\_client IN (SELECT ID\_client

FROM `order`

WHERE `order`.ID\_point = 2);

* 1. **Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора Any.**

Тот же результат можно получить при использовании оператора ANY.

SELECT Name

FROM client

WHERE ID\_client = ANY (SELECT ID\_client

FROM `order`

WHERE `order`.ID\_point = 2);

* 1. **Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора All.**

Выберем тех заказчиков, которые не делали заказы, результат должен быть аналогичен п.2.3.

SELECT Name

FROM client

WHERE ID\_client <> ALL (SELECT ID\_client FROM `order`);

* 1. **Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора Exists.**

Покажем еще один способ выбора заказчиков без заказов.

SELECT Name

FROM client

WHERE NOT EXISTS

(SELECT 1

FROM `order`

WHERE `order`.ID\_client = client.ID\_client);

1. **Выполнить запрос, описанный в задании.**
2. Вывести какое количество каждого вида продукции было заказано в каждом из филиалов за последний месяц.

Данный запрос можно выполнить двумя способами:

Выведем результаты запроса для текущего месяца.

SELECT name, sum(`order-product`.Amount)

FROM product, `order-product`, `order`

WHERE `order-product`.Order\_ID = `order`.ID\_order

AND product.ID\_prod = `order-product`.Product\_ID

AND (month(now()) - month(`order`.order\_date)) < 1

GROUP BY product.Name;

А теперь выведем заказы за последние 30 дней. Результаты выполнения запросов различны.

SELECT name, sum(`order-product`.Amount)

FROM product, `order-product`, `order`

WHERE `order-product`.Order\_ID = `order`.ID\_order

AND product.ID\_prod = `order-product`.Product\_ID

AND (to\_days(now()) - to\_days(`order`.order\_date)) < 30

GROUP BY product.Name;

1. Вывести суммарное количество продукции, проданной каждым филиалом по месяцам за текущий год.

SELECT address, month(`order`.order\_date) AS month,

sum(`order-product`.Amount) AS amount

FROM `order-product`,`order`,`point`

WHERE `order-product`.Order\_ID = `order`.ID\_order

AND `order`.ID\_point = `point`.Id\_point

AND year(now()) = year(`order`.order\_date)

GROUP BY month, `order`.id\_point

ORDER BY 1, 2;

1. Вывести суммарное количество заказов, выполненное каждым филиалом по месяцам за текущий год.

SELECT count(ID\_Order) AS amount, address, month(`order`.order\_date) AS month

FROM `order`, `point`

WHERE `order`.ID\_point = `point`.Id\_point

AND year(now()) = year(`order`.order\_date)

GROUP BY `order`.ID\_point, 3;

1. Вывести те филиалы, которые выполнили максимальное количество заказов в каждом месяце за последние полгода.

Данный запрос можно выполнить, используя дополнительную таблицу (лучше, если это будет VIEW, поскольку данные в нем автоматически обновляются, а в таблице нет)

Сначала создадим таблицу-представление, в которой будем хранить количество заказов, выполненное каждым филиалом по месяцам за последние полгода.

CREATE VIEW sum\_orders

AS

SELECT count(ID\_Order) AS amount,

id\_point,

month(`order`.order\_date) AS month

FROM `order`

WHERE `order`.order\_date > date\_sub(now() , INTERVAL 6 month)

GROUP BY 2, 3;

Теперь выполним запрос, требуемый в задании:

SELECT amount, id\_point, month

FROM sum\_orders a

WHERE amount = (SELECT max(amount)

FROM sum\_orders b

WHERE a.month = b.month

GROUP BY b.month)

AND month(now()) - month < 6

ORDER BY 3;

1. Для каждого филиала вывести общее число заказов, средний чек заказа, минимальную сумму заказа за каждый месяц.

Данный запрос можно выполнить как созданием промежуточного представления, так и при помощи выборки из результатов выборки, что мы и сделаем.

Внутренний запрос приносит такие данные о заказе как: его дата, сумма и номер филиала. Для удобства во внутреннем запросе мы используем алиасы столбцов. Кроме того, мы должны использовать алиас для всех результатов внутренней выборки, так как они рассматриваются как «таблица» в части FROM.

SELECT point,

month(ex\_date),

year(ex\_date),

min(summa),

avg(summa)

FROM (SELECT `order`.exec\_date AS ex\_date,

`order`.ID\_point AS point,

product.Price \* `order-product`.Amount AS summa

FROM `order-product`, `order`, product

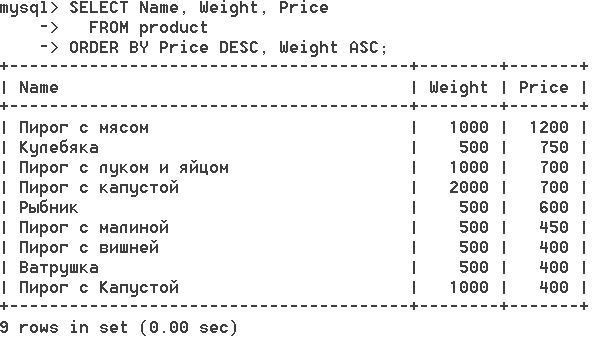
WHERE `order-product`.Order\_ID = `order`.ID\_order

AND `order-product`.Product\_ID = product.ID\_prod) a

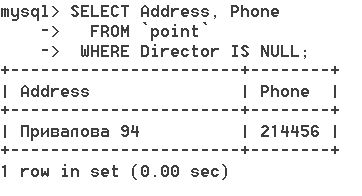
GROUP BY 1, 2, 3

Отчет по лабораторной работе №2

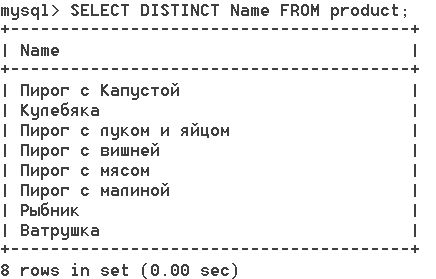
1. **Простой выбор данных (select, \*, from, where, order by, вычислимые поля, distinct).**
   1. **Создать простой запрос на выборку из одной таблицы. Включить несколько полей таблицы. Выбрать несколько полей, по которым сортируется вывод, по одному из полей отсортировать в порядке возрастания, по другому – в порядке убывания.**



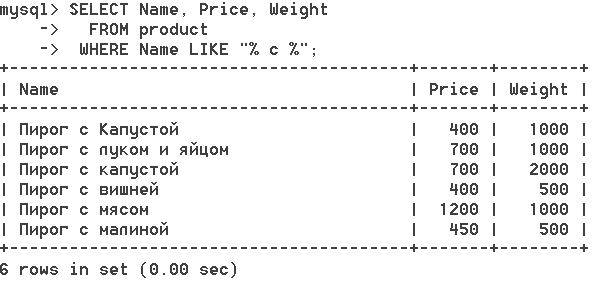
* 1. **Выбрать из таблицы строки, содержащие значение NULL.**



* 1. **Выбрать из таблицы неповторяющиеся значения.**

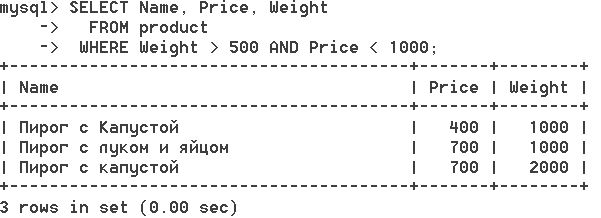


* 1. **Использовать в запросе оператор Like.**



* 1. **Выполнить один запрос с использованием условного оператора ИЛИ и один с использованием условного оператора И.**

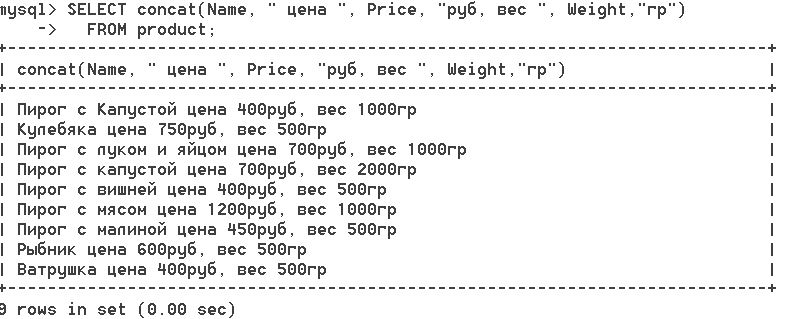
Использование оператора И

****

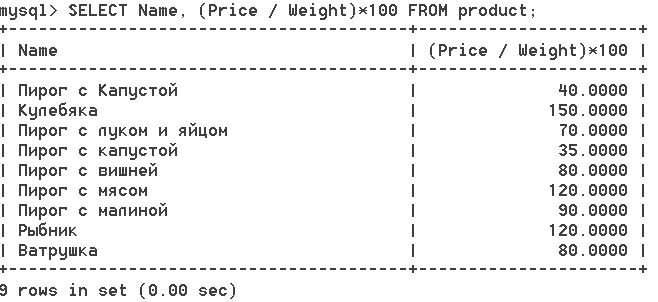
Использование оператора ИЛИ

****

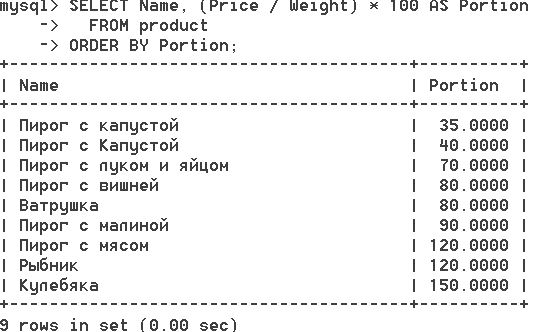
* 1. **Выполнить запрос с использованием функций форматирования строк.**



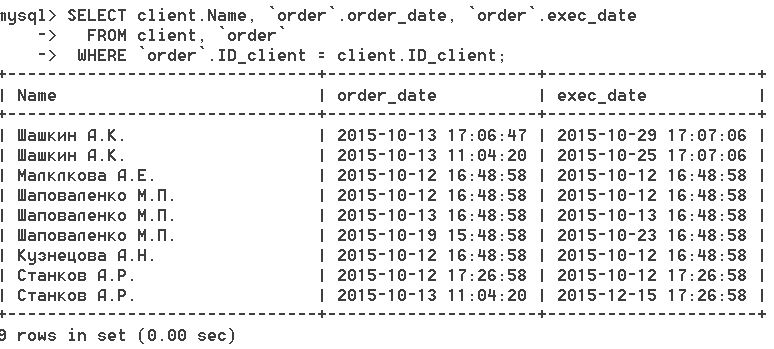
* 1. **Выполнить запрос, в котором одно из полей является результатом арифметической функции над каким-либо полем таблицы.**



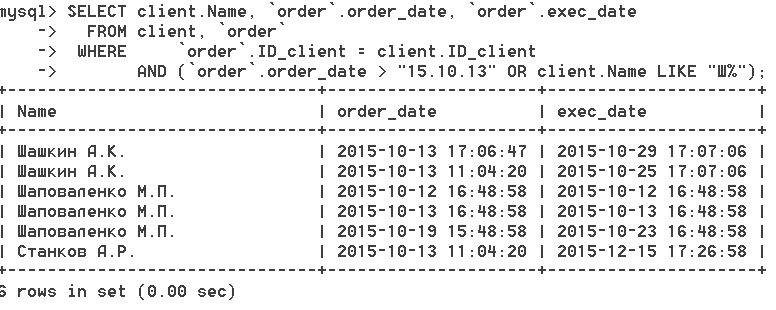
* 1. **Выполнить отсортированный по вычислимому полю запрос, для вычислимого поля использовать альтернативное имя.**



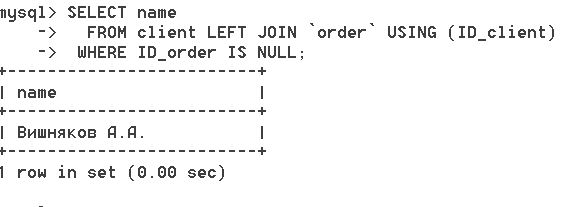
1. **Объединение таблиц в запросе.**
   1. **Выполнить запрос на объединение связанных таблиц.**



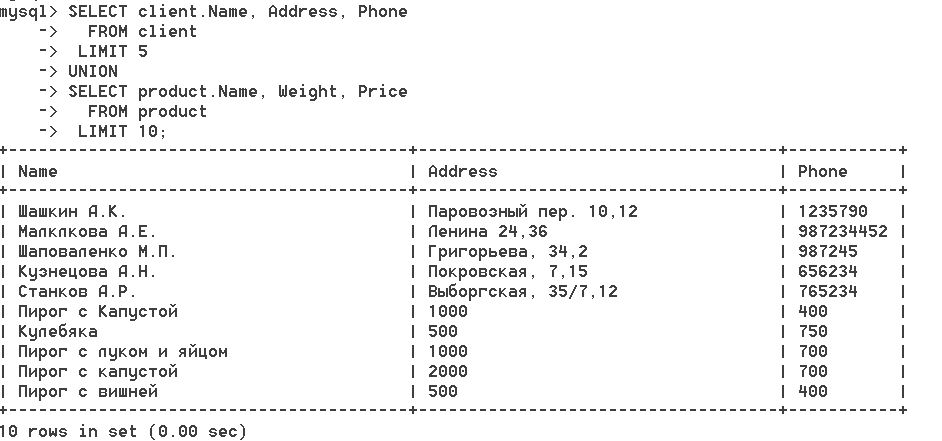
* 1. **Выполнить запрос из нескольких таблиц, использовать в запросе условные операторы И и ИЛИ.**



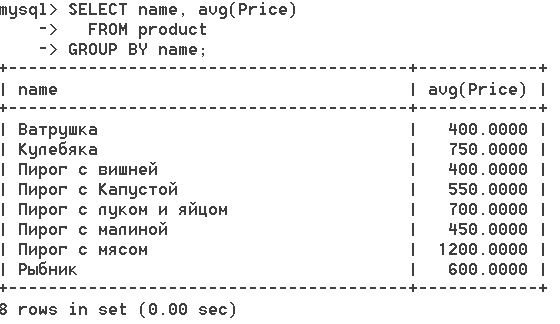
* 1. **Выполнить запрос, показывающий избыточные данные в одной из связанных по внешнему ключу таблиц.**



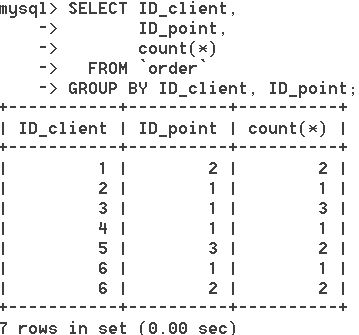
* 1. **Объединить в запросе данные из двух таблиц, не используя соединение таблиц.**



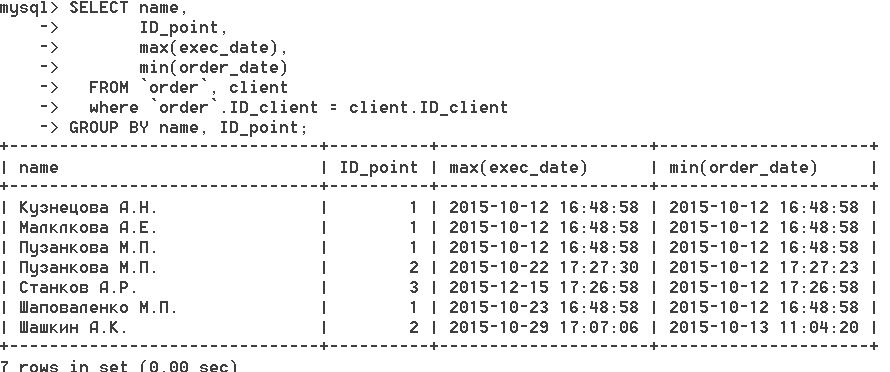
1. **Выбор данных с помощью группирующих запросов с условием (group by, having, min(), max(), sum(), count(), …).**
   1. **Выполнить группирующий запрос, содержащий несколько полей с группировкой по одному полю.**



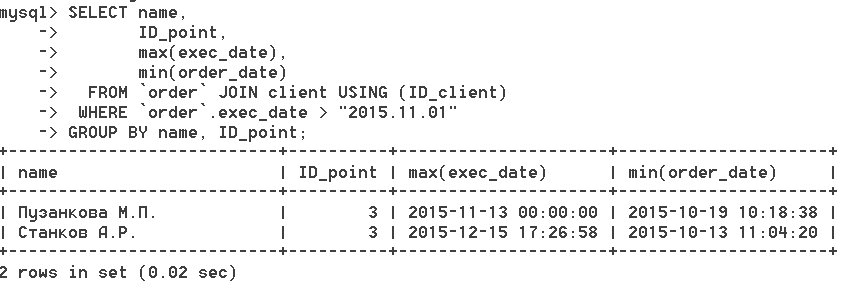
* 1. **Выполнить группирующий запрос с группировкой по нескольким полям.**



* 1. **Выполнить группирующий запрос с соединением таблиц.**



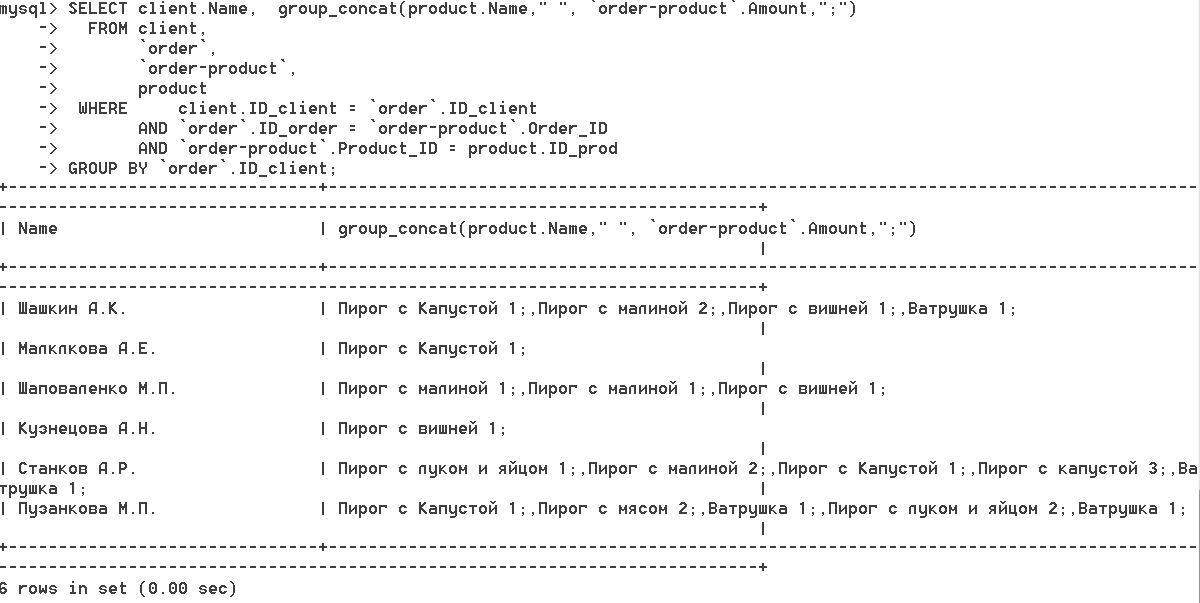
* 1. **Выполнить группирующий запрос с условным оператором, в котором сначала выполняется условный оператор, а затем производится выборка.**



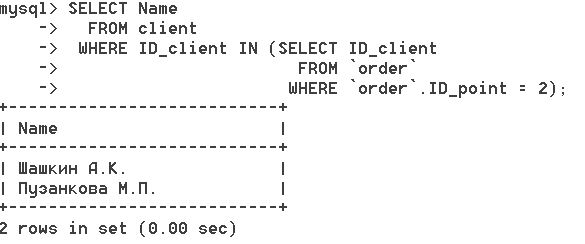
* 1. **Выполнить группирующий запрос с условным оператором, в котором сначала производится выборка, а затем выполняется условный оператор.**



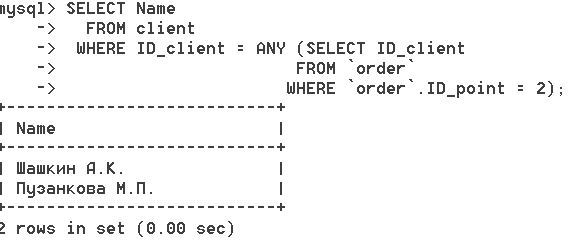
* 1. **Выполнить группирующий запрос, в котором одно из полей получено при помощи функции group\_concаt.**



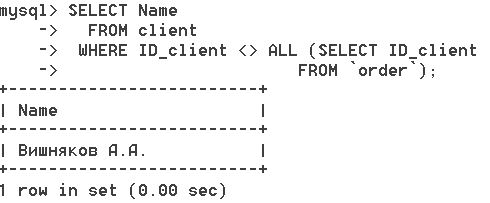
1. **Выбор данных с помощью подзапросов.**
   1. **Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора In.**



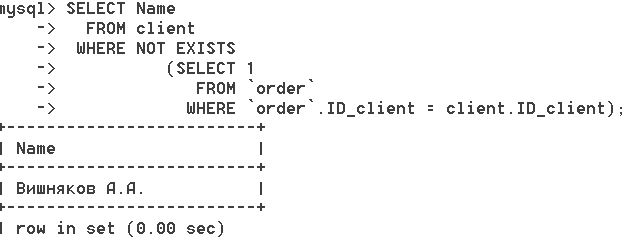
* 1. **Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора Any.**



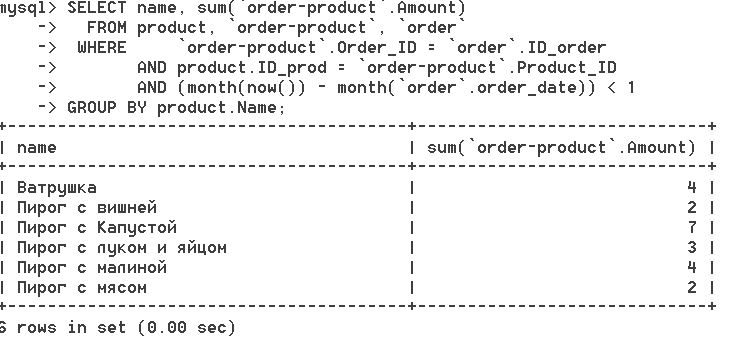
* 1. **Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора All.**



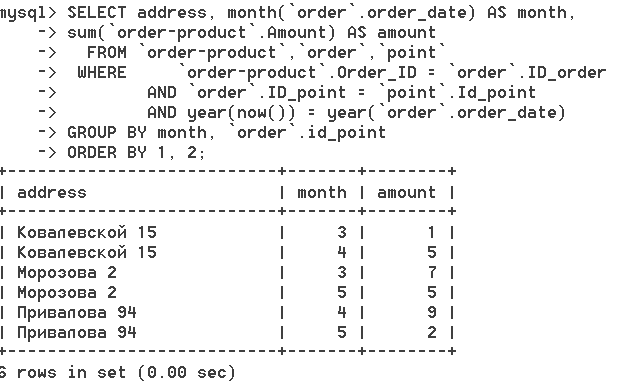
* 1. **Выполнить запрос, в котором производится условный выбор из подзапроса с использованием оператора Exists.**



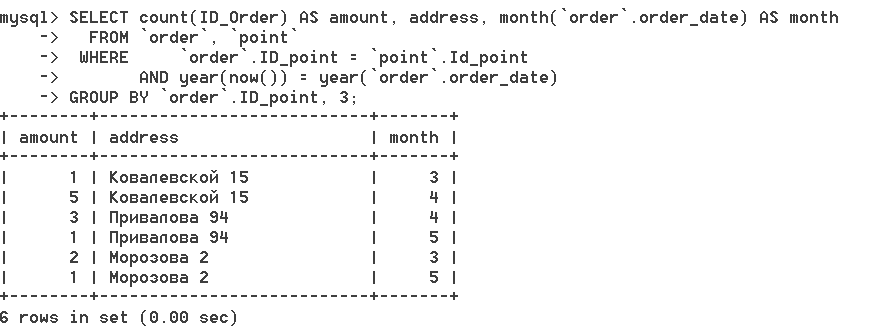
1. **Выполнить запрос, описанный в задании.**
2. Вывести какое количество каждого вида продукции было заказано в каждом из филиалов за последний месяц.



1. Вывести суммарное количество продукции, проданной каждым филиалом по месяцам за текущий год.

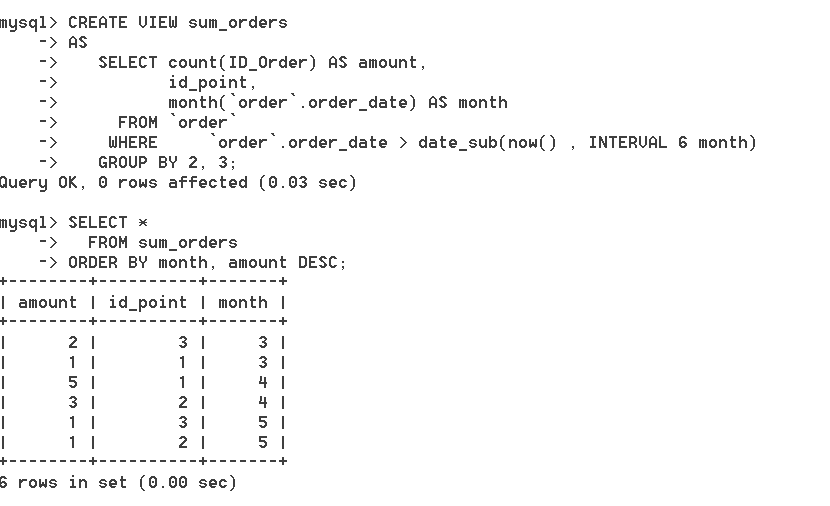


1. Вывести суммарное количество заказов, выполненное каждым филиалом по месяцам за текущий год.

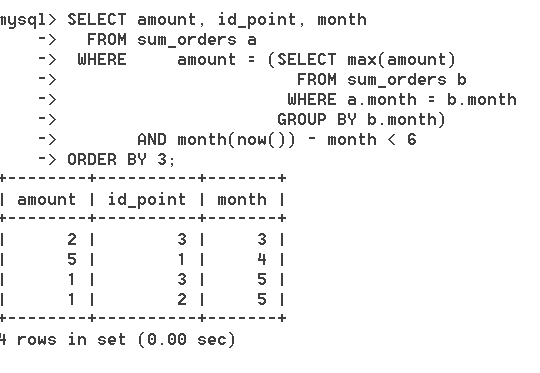


1. Вывести те филиалы, которые выполнили максимальное количество заказов в каждом месяце за последние полгода.

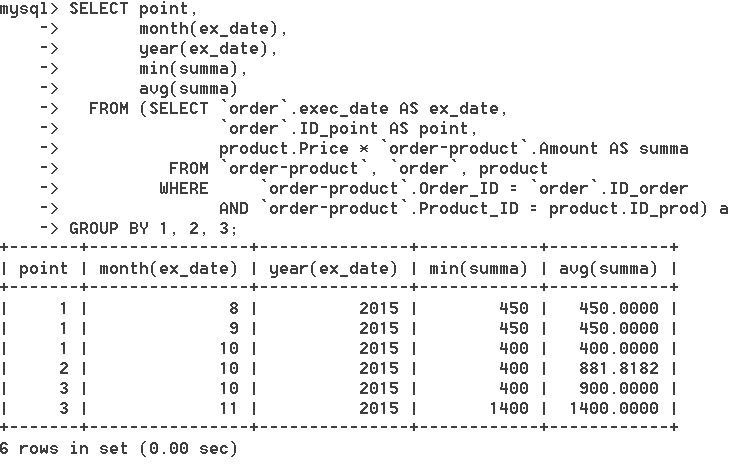
Создаем промежуточную таблицу:



Выполняем искомый запрос:



1. Для каждого филиала вывести общее число заказов, средний чек заказа, минимальную сумму заказа за каждый месяц.



Задания для лабораторной работы №3

**Структура отчета**

1. **Создать простую таблицу типа View с соединением таблиц**

Описать что это представление позволяет видеть.

1. **Создать таблицу типа View, содержащую вычислимые поля и поля с форматированием данных.**

Описать что это представление позволяет видеть.

1. **Создать таблицу типа View, согласно требованиям, представленным в задании.**

Привести текст задания. Если для его выполнения требуются промежуточные действия, представить их в отчете.

1. **Создать Trigger, выполняющий требования бизнес-логики, представленной в задании.**

Привести текст задания. Сделать изменения в таблицах, если требуется, отразить это в отчете.

1. **Написать процедуру с входными и выходными параметрами (минимум по одному), в теле каждой использовать один условный выбор и (или) один цикл.**
2. **Написать функцию с входными параметрами, в теле каждой использовать один условный выбор и (или) один цикл.**
3. **Привести примеры вызова процедуры и функции из пп. 5,6.**

Структура отчета полностью повторяет структуру вопросов. Для каждого из вопросов задания представляются экранные копии команд и результатов их выполнения.

В пп.5,6 в процедуре и функции ОБЯЗАТЕЛЬНО минимум один раз использовать цикл и минимум один раз условный выбор.

Процедура и функция должны иметь практический смысл, приложимый к предметной области.

Выполнение лабораторной работы №3

1. **Создать простую таблицу типа View с соединением таблиц.**

Создадим представление, в котором покажем какое количество продано каждого вида продукции.

Поскольку часть столбцов являются результатами функций, то для удобства обращения к таблице, при создании представления будем использовать алиасы для имен столбцов.

CREATE VIEW products\_bought

AS

SELECT product.Name, sum(Amount) AS Amount

FROM `order-product`, `order`, product

WHERE lr.`order-product`.Order\_ID = lr.`order`.ID\_order

AND lr.`order-product`.Product\_ID = lr.product.ID\_prod

GROUP BY name;

1. **Создать таблицу типа View, содержащую вычислимые поля и поля с форматированием данных.**

Создадим представление, в котором выведем перечень заказанной продукции за каждый месяц и общую сумму заказов.

В названии столбцов также используем алиасы. В функции group\_counat мы использовали команду dictinct, чтобы выводить каждый вид продукции только один раз, а не столько раз, сколько он был заказан.

CREATE VIEW products\_by\_month

AS

SELECT month(`order`.order\_date) AS month,

GROUP\_CONCAT(DISTINCT product.Name) AS Assortment,

sum(product.Price \* `order-product`.Amount) AS Amount

FROM `order-product`, `order`, product

WHERE lr.`order-product`.Order\_ID = lr.`order`.ID\_order

AND lr.`order-product`.Product\_ID = lr.product.ID\_prod

GROUP BY 1;

1. **Создать таблицу типа View, согласно требованиям, представленным в задании.**
2. Для каждого заказа вывести его дату, сумму и перечисление купленной продукции.
3. Для каждого заказчика вести историю заказов в следующем виде: имя клиента, дата первого заказа, дата последнего заказа, виды заказанной продукции и общее их количество через запятую в формате: «продукция1 – количество, продукция2 – количество…», общая сумма заказов.

Создать такое представление в одном запросе невозможно. Поэтому для выполнения задания создадим промежуточное представление, в котором для каждого клиента будет перечислена вся заказанная им продукция и будут посчитаны количество каждого вида продукции.

CREATE VIEW products\_by\_clients

AS

SELECT client.ID\_client,

client.name AS Client,

`order-product`.Product\_ID,

product.Name AS Product,

sum(`order-product`.Amount) AS Amt,

sum(product.Price \* `order-product`.Amount) AS Total

FROM client,

`order`,

`order-product`,

product

WHERE lr.`order-product`.Order\_ID = lr.`order`.ID\_order

AND client.ID\_client = `order`.ID\_client

AND lr.`order-product`.Product\_ID = lr.product.ID\_prod

GROUP BY client.ID\_client, `order-product`.Product\_ID;

Теперь создадим представление, требуемое в задании, необходимые данные получим из только что созданного View products\_by\_clients и таблицы Заказ.

CREATE VIEW clients\_orders

AS

SELECT Client,

min(order\_date) AS First\_order,

max(order\_date) AS Last\_order,

GROUP\_CONCAT(DISTINCT Product, " - ", B.Amt) AS Assortment,

sum(Total) AS Total

FROM `order` A, products\_by\_clients B

WHERE A.ID\_client = B.ID\_client

GROUP BY B.ID\_client;

1. Создать представление, в котором для каждого филиала вывести общее число заказов, средний чек заказа, минимальную сумму заказа за каждый месяц.

Несмотря на то, что в предыдущем задании аналогичное задание было выполнено за один запрос, сделать при помощи того же запроса представление по правилам языка невозможно. Поэтому будем также выполнять задание в два шага.

Создадим представление, в котором будут представлены нужные данные о заказе:

CREATE VIEW orders

AS

SELECT order\_id,

`order`.exec\_date AS ex\_date,

`order`.ID\_point AS point,

product.Price \* `order-product`.Amount AS summa

FROM `order-product`, `order`, product

WHERE `order-product`.Order\_ID = `order`.ID\_order

AND `order-product`.Product\_ID = product.ID\_prod;

1. **Создать Trigger, выполняющий требования бизнес-логики, представленной в задании.**

В таблице Заказчик добавить поле сумма заказов, при добавлении нового заказа для данного заказчика, обновлять это поле с учетом нового заказа.

Внесем необходимые изменения в таблицу Заказчик.

ALTER TABLE client

ADD Total INT DEFAULT '0' AFTER Phone;

Для того, чтобы написать тело триггера, процедуры или функции, необходимо временно изменить делимитер языка MySql c «;» на любой другой символ. Это делается при помощи команды

DELIMETR <new delimetr>

В данном примере мы использовали в качестве делимитера двойной знак доллара, при этом делимитер в теле триггера остается классическим: «;».

Создадим триггер.

CREATE TRIGGER `update\_total`

AFTER INSERT

ON `order-product`

FOR EACH ROW

BEGIN

declare order\_total int(10);

declare clientID int(10);

SELECT product.Price \* `order-product`.Amount into order\_total

FROM `order-product`, product

WHERE `order-product`.Product\_ID = product.ID\_prod

AND `order-product`.Order\_ID = new.Order\_ID

AND `order-product`.Product\_ID = new.Product\_ID;

SELECT ID\_client into clientID

FROM `order` a

WHERE a.ID\_order = new.Order\_ID;

UPDATE client

SET Total = Total + order\_total

WHERE ID\_client = clientID;

END;

Можно записать тело по-другому, без объявления переменных, в одну команду.

CREATE TRIGGER `update\_total`

AFTER INSERT

ON `order-product`

FOR EACH ROW

BEGIN

UPDATE client

SET Total =

Total

+ (SELECT product.Price \* `order-product`.Amount

FROM `order-product`, product

WHERE `order-product`.Product\_ID = product.ID\_prod

AND `order-product`.Order\_ID = new.Order\_ID

AND `order-product`.Product\_ID = new.Product\_ID)

WHERE ID\_client = (SELECT ID\_client

FROM `order` a

WHERE a.ID\_order = new.Order\_ID);

END;

Теперь проиллюстрируем работу созданного триггера. Покажем вид таблицы Заказчик до изменений.

SELECT \*

FROM client

WHERE ID\_client = 6;

Добавим сначала данные в таблицу Заказ.

INSERT INTO `order`(ID\_client, ID\_point, exec\_date)

VALUES (6, 3, "15-11-13");

Выведем полученный номер заказа.

SELECT id\_order

FROM `order`

ORDER BY 1 DESC

LIMIT 1;

Теперь добавим данные в таблицу связи.

INSERT INTO `order-product` (Order\_ID, Product\_ID, Amount)

VALUES (15, 3, 2);

И выведем изменения в таблице Заказчик, которые иллюстрируют результат работы триггера.

SELECT \*

FROM client

WHERE ID\_client = 6;

1. **Написать процедуру с входными и выходными параметрами (минимум по одному), в теле каждой использовать один условный выбор и (или) один цикл.**

По условиям задания, в функции и в процедуре нужно использовать цикл и условный оператор. В процедуре используем условный оператор IF.

Создадим процедуру, которая будет получать на вход ID Продукта и планируемое число продаж для этого продукта. Если план выполнен, то процедура возвращает “YES”, если продано меньше единиц продукта, то “NO”.

Перед написанием процедуры не забудем поменять делимитер.

CREATE PROCEDURE product\_orders(IN productID int,

IN plan int,

OUT ismade char(3))

BEGIN

DECLARE productcount int;

SELECT sum(`order-product`.Amount)

INTO productcount

FROM `order`, `order-product`

WHERE `order`.ID\_order = `order-product`.Order\_ID

AND `order-product`.Product\_ID = productID;

IF (productcount >= plan)

THEN

SET ismade = "YES";

ELSE

SET ismade = "NO";

END IF;

END;

1. **Написать функцию с входными параметрами, в теле каждой использовать один условный выбор и (или) один цикл.**

По условиям задания, в функции и в процедуре нужно использовать цикл и условный оператор. Условный оператор мы уже использовали, следовательно, в функции мы должны использовать цикл.

Создадим функцию, которая выводит номер филиала, в котором заказчик имеет наибольшее число заказов. Поскольку в процедуре был использован только условный оператор, то в функции необходимо использовать цикл. Не забудем опять про делимитер.

CREATE FUNCTION maxOrders(clientID int)

RETURNS int

BEGIN

DECLARE pointID, currentPoint, maxPoint int;

DECLARE currentOrders, maxOrders int;

SET currentPoint = 1;

SET maxOrders = 0;

SELECT max(ID\_point) INTO maxPoint FROM `point`;

WHILE currentPoint <= maxPoint

DO

SELECT count(\*)

INTO currentOrders

FROM `order`, client

WHERE `order`.ID\_client = client.ID\_client

AND `order`.ID\_client = clientID

AND ID\_point = currentPoint;

IF (currentOrders > maxOrders)

THEN

SET maxOrders = currentOrders;

SET pointID = currentPoint;

END IF;

SET currentPoint = currentPoint + 1;

END WHILE;

RETURN pointID;

END;

1. **Привести примеры вызова процедуры и функции из пп. 7,8.**

**Пример вызова процедуры.**

Сначала установим значения для входных параметров.

SET @productID = 7;

SET @plan = 4;

Теперь вызовем процедуру.

CALL product\_orders(@productID, @plan, @ismade);

И получим значение выходного параметра процедуры, т.е. результат ее выполнения.

SELECT @ismade;

Процедуру можно вызвать и без установки параметров

CALL product\_orders(7, 7, @ismade);

**Пример вызова функции.**

Установим входной параметр.

SET @pointID=2;

Вызовем функцию.

SELECT maxOrders(@pointID);

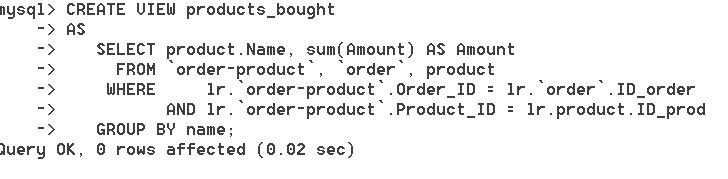
А теперь вызовем функцию без установки параметров.

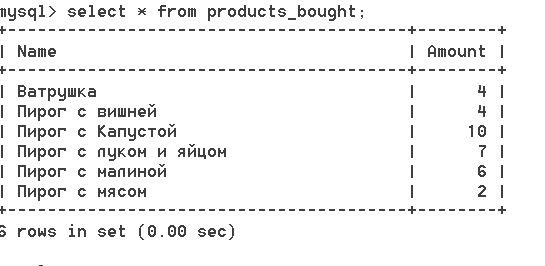
SELECT maxOrders(2);

SELECT maxOrders(5);

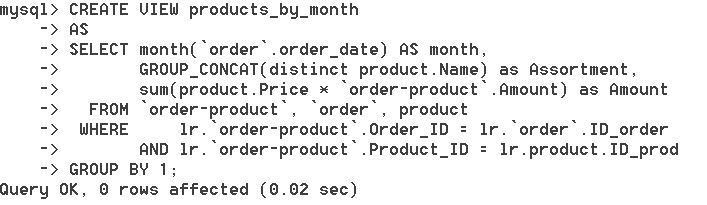
Отчет по лабораторной работе №3

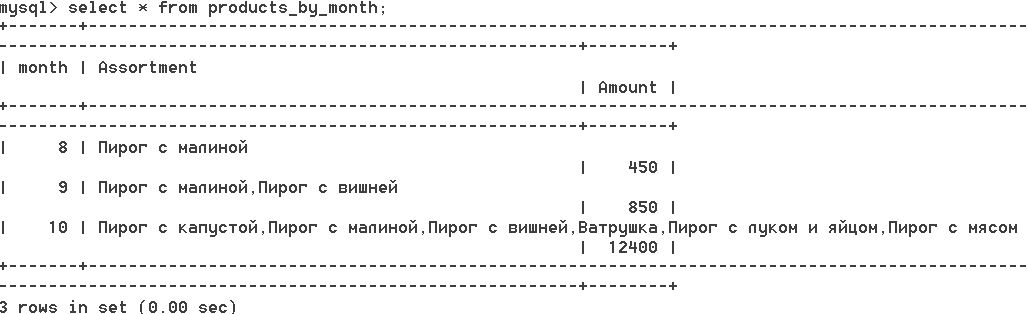
1. **Создать простую таблицу типа View с соединением таблиц**

****

****

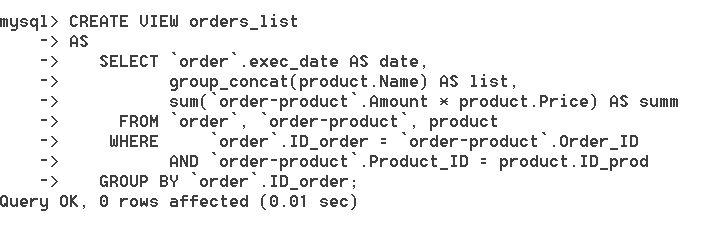
1. **Создать таблицу типа View, содержащую вычислимые поля и поля с форматированием данных**



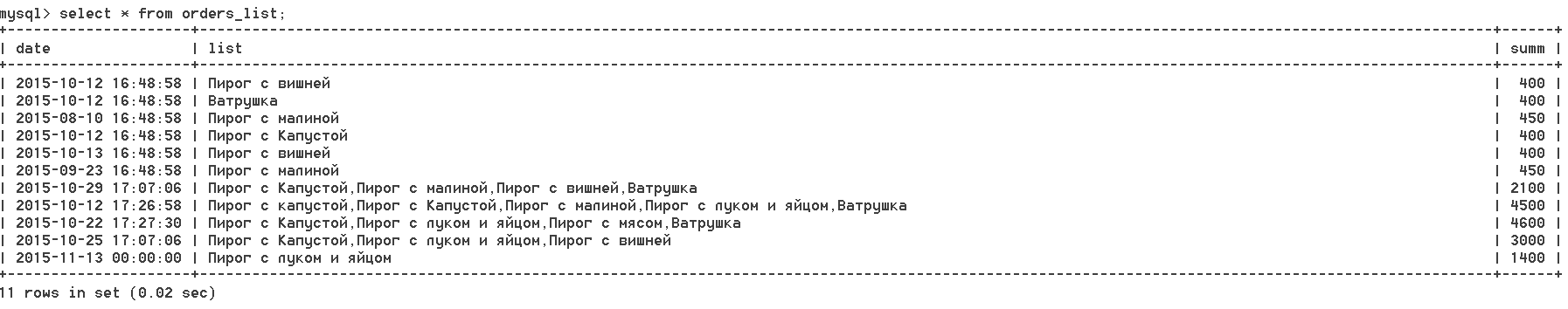


1. **Создать таблицу типа View, согласно требованиям, представленным в задании.**
2. Для каждого заказа вывести его дату, сумму и перечисление купленной продукции.

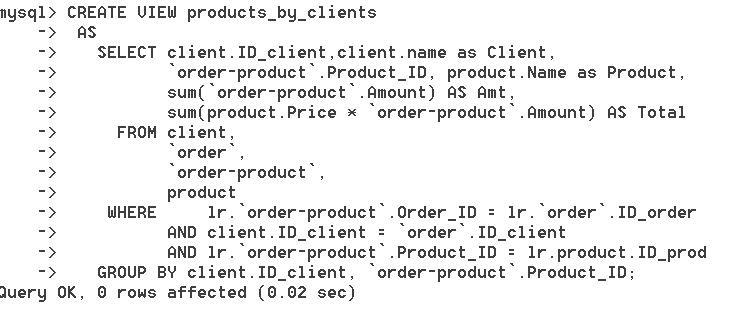
**Требуемое представление**



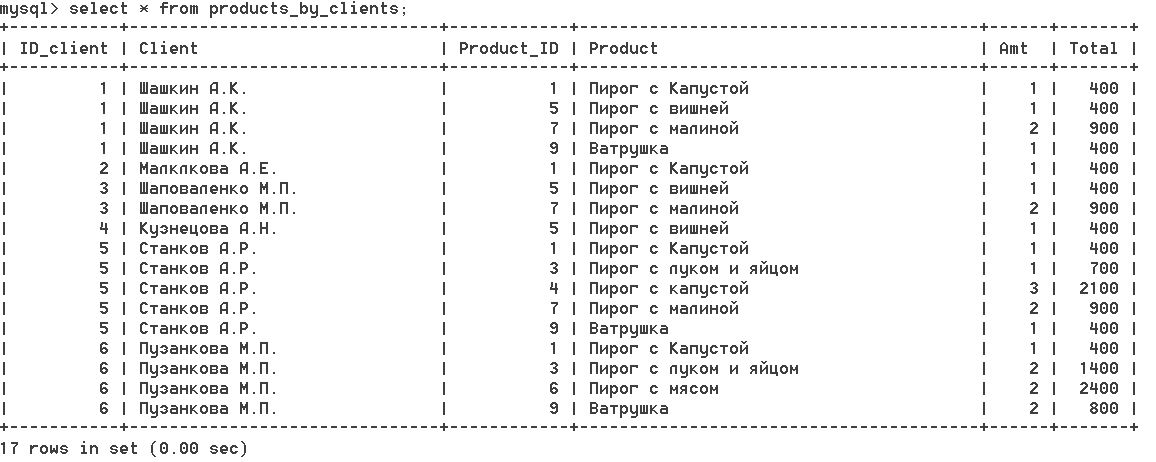
Его содержимое:



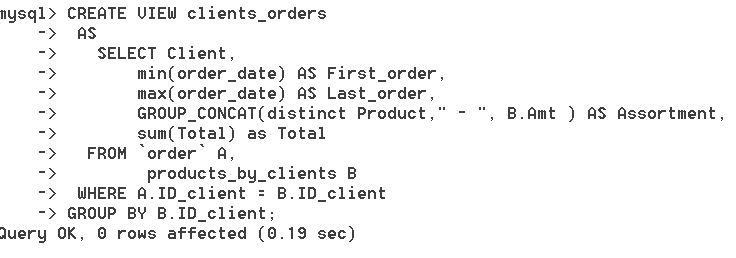
1. Для каждого заказчика вести историю заказов в следующем виде: имя клиента, дата первого заказа, дата последнего заказа, виды заказанной продукции и общее их количество через запятую в формате: «продукция1 – количество, продукция2 – количество…», общая сумма заказов.

****

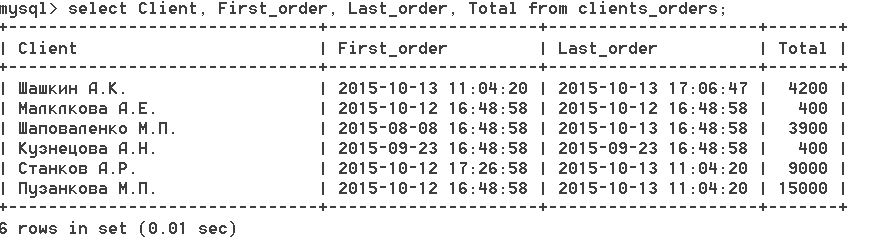
Его содержимое

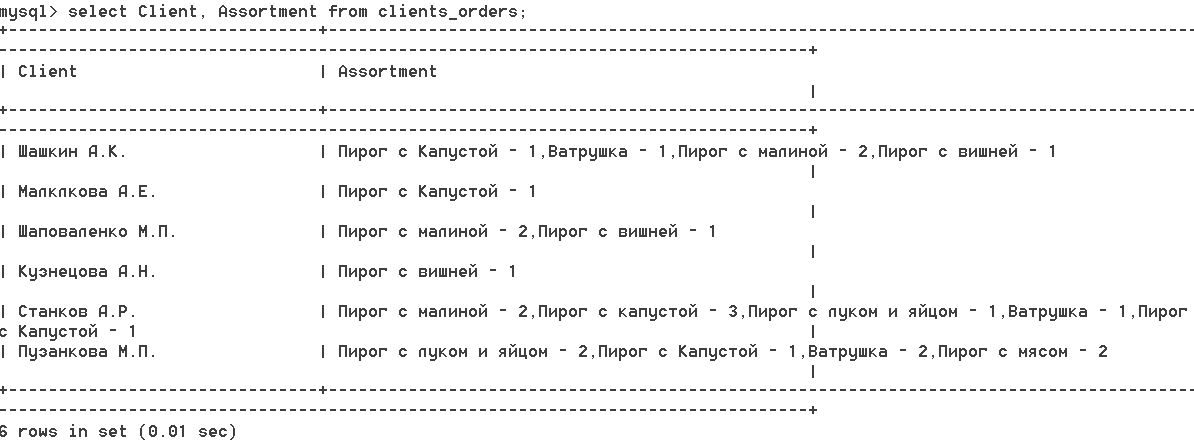
****

**Требуемое представление**

****

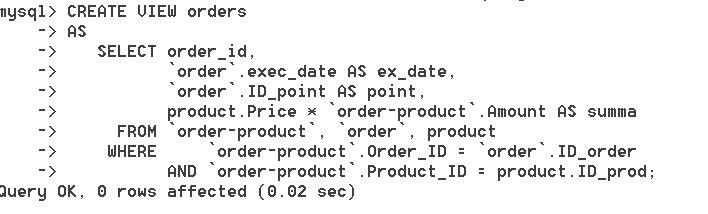
Выведем данные из созданного представления. Вывод сделан по частям, поскольку столбцы очень широкие.

****

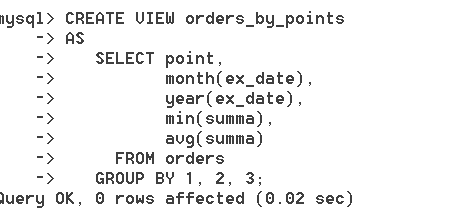
****

1. Создать представление, в котором для каждого филиала вывести общее число заказов, средний чек заказа, минимальную сумму заказа за каждый месяц.

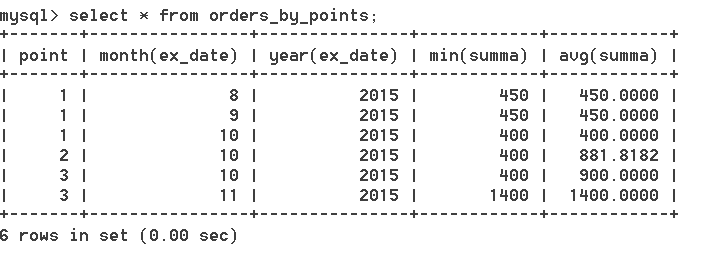
Создаем вспомогательное представление.



**Требуемое представление**

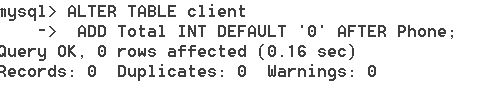


Выведем данные из созданного представления.



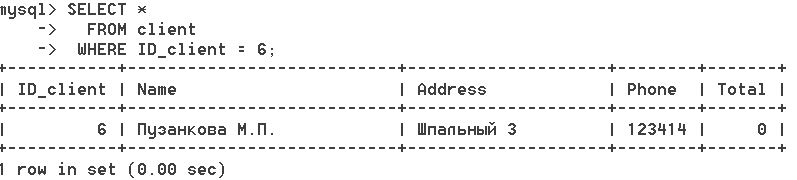
1. **Создать Trigger, выполняющий требования бизнес-логики, представленной в задании.**

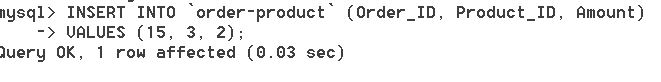
Вносим изменения в таблицу Client.

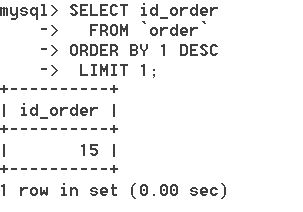


****

Таблица Client до работы триггера

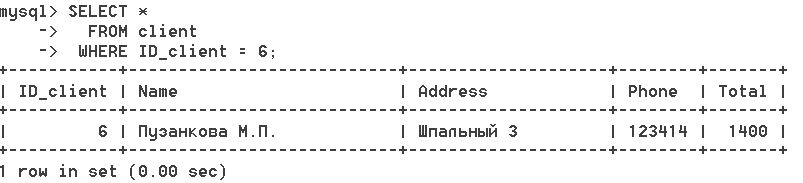
****

****

****

**H:\РАБОТА\Курсы\БД\ЛАБОРАТОРНЫЕ\для примера выполнения ЛР\экраны ЛР3\4.4.JPG**

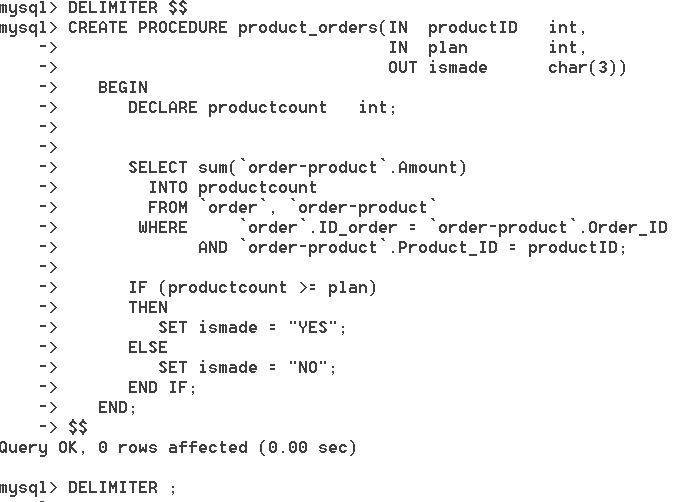
Таблица Client после работы триггера

****

1. **Написать процедуру с входными и выходными параметрами (минимум по одному), в теле каждой использовать один условный выбор и (или) один цикл.**

(По условиям задания, в функции и в процедуре нужно использовать цикл и условный оператор. В процедуре используем условный оператор IF.)

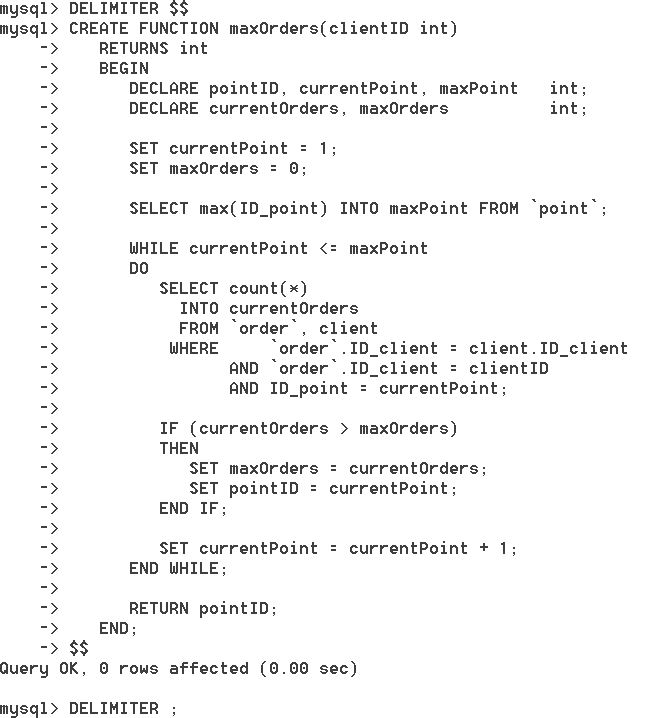
Создадим процедуру, которая будет получать на вход ID Продукта и планируемое число продаж для этого продукта. Если план выполнен, то процедура возвращает “YES”, если продано меньше единиц продукта, то “NO”.



1. **Написать функцию с входными параметрами, в теле каждой использовать один условный выбор и (или) один цикл.**

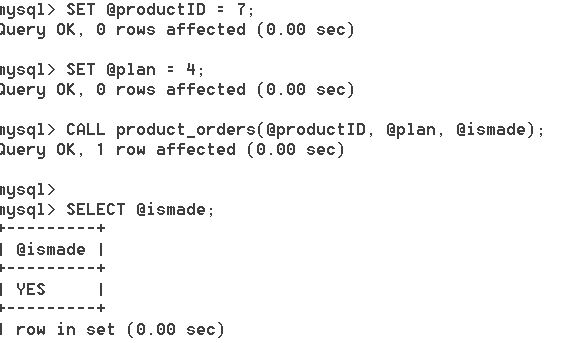
(По условиям задания, в функции и в процедуре нужно использовать цикл и условный оператор. Условный оператор мы уже использовали, следовательно, в функции мы должны использовать цикл.)

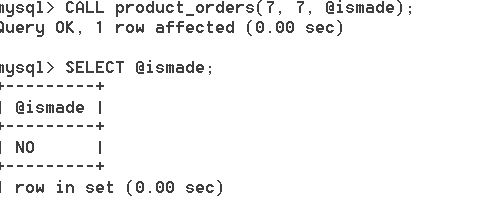
Создадим функцию, которая выводит номер филиала, в котором заказчик имеет наибольшее число заказов.



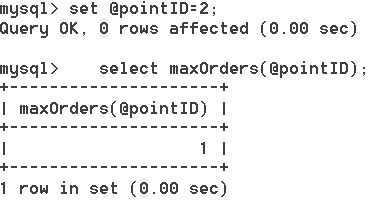
1. **Привести примеры вызова процедуры и функции из пп. 5,6.**

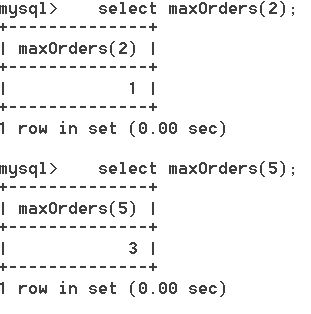
Пример вызова процедуры.





Пример вызова функции





Оглавление

[Введение 3](#_Toc452502199)

[Описание задания. 4](#_Toc452502200)

[Задания для лабораторной работы №1 5](#_Toc452502201)

[Выполнение лабораторной работы №1 6](#_Toc452502202)

[Отчет по лабораторной работе №1 16](#_Toc452502203)

[Задания для лабораторной работы №2 25](#_Toc452502204)

[Выполнение лабораторной работы №2 27](#_Toc452502205)

[Отчет по лабораторной работе №2 36](#_Toc452502206)

[Задания для лабораторной работы №3 49](#_Toc452502207)

[Выполнение лабораторной работы №3 50](#_Toc452502208)

[Отчет по лабораторной работе №3 57](#_Toc452502209)

Елена Александровна **Неймарк**

**Методические указания**

**для выполнения лабораторных работ**

**по курсу «Базы Данных»**

***Учебно-методическое пособие***

Федеральное государственное автономное образовательное учреждение высшего образования «Нижегородский государственный университет им. Н.И. Лобачевского».

603950, Нижний Новгород, пр. Гагарина, 23.