# 1. Разработка баз данных в СУБД MySQL

## 1.1. Построение EDR-модели

Существует некоторая фирма, торгующая путёвками. Путевка выступает в качестве товара и как товар имеет определенные данные. На каждую путёвку есть связь с туристом. Путёвка может быть описана следующим образом(код путёвки,код страны,длительность,цена,дата отправления),туристы(код туриста,фио,дата рождения,паспорт),страна(код страны,страна),билет(код билета,код путёвки,код туриста)

Кроме того, на каждый автомобиль имеются технические данные. Фирма имеет своих клиентов – покупателей автомобилей, сведения о которых хранит в течение определенного времени. Деятельность фирмы может быть описана данными, сгруппированными следующим образом: товар (код товара, страна-изготовитель, марка автомобиля, модель, цвет, наличие на складе (да, нет), цена); технические данные (код товара, тип кузова, количество дверей, количество мест, тип двигателя, расположение двигателя, рабочий объем двигателя); клиенты (ФИО клиента, паспортные данные (серия, номер), домашний адрес, телефон); покупка (код товара, код клиента, дата, доставка (да, нет), вид оплаты (перечисление или наличные).

В процессе проектирования ERD модели БД были выделены следующий сущности:

Auto, AutoModel, AutoBrand, Country, Color, TechData, BodyType, EngineType, EngineLocation, Purchase, TypeOfPayment, Client.

Описание сущностей:

Route – хранит данные о путевках.

tourist – хранит данные о туристах.

country– хранит данные о странах.

Trip –хранит связь путёвки и туриста

|  |  |  |  |
| --- | --- | --- | --- |
| **tourist** | | | |
| Поле | Тип | Атрибуты | Описание |
| Idtourist | INT | PK NN AI | Уникальный номер туриста |
| fio | Varchar(45) | NN | Фамилия имя отчество |
| passport | Varchar(45) | NN | Номер и серия паспорта |

|  |  |  |  |
| --- | --- | --- | --- |
| **route** | | | |
| Поле | Тип | Атрибуты | Описание |
| Idroute | INT | PK NN AI | Уникальный номер путёвки |
| idcountry | INT | NN | Номер страны |
| durutaion | INT | NN | Номер и серия паспорта |
| cost | INT | NN | Цена путёвки |
| dateout | INT | NN | Дата вылета |

|  |  |  |  |
| --- | --- | --- | --- |
| **country** | | | |
| Поле | Тип | Атрибуты | Описание |
| Idcountry | INT | PK NN AI | Уникальный номер страны |
| country | INT | NN | Страна |

|  |  |  |  |
| --- | --- | --- | --- |
| **trip** | | | |
| Поле | Тип | Атрибуты | Описание |
| Idtrip | INT | PK NN AI | Уникальный номер путешествия |
| idroute | INT | NN | Номер страны |
| idtourist | INT | NN | Номер туриста |

**1.2 Заполнение базы данных**

Таблица route

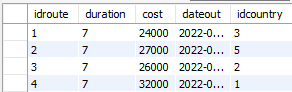


Таблица tourist

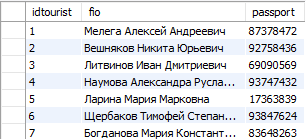


Таблица trip

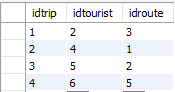
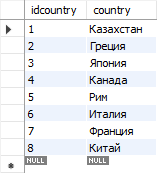


Таблица country

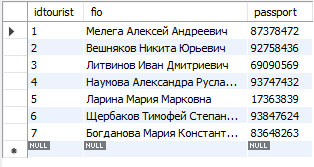


1.3 Написание запросов по работе с данными

### 1.3.1. Выборка и выборка с условием

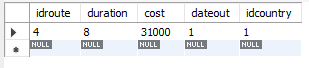
1. Выбрать данные из одной таблицы.

select \* from tourist



1. Выбрать данные из одной таблицы по условию WHERE

select \* from route where cost > 29000



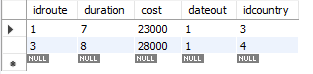
1. Выбрать данные с применением команды DISTINCT

select count(distinct fio) as 'Мелега Алексей Андреевич' from tourist



1. Выбрать данные из таблицы по условию WHERE с применением оператора вхождения в коллекцию – IN

select \* from route where cost in(23000,28000)



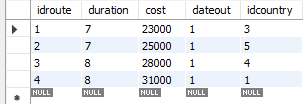
1. Выбрать данные из таблицы по условию WHERE с использованием оператора Like

select fio from tourist where fio like 'б%'



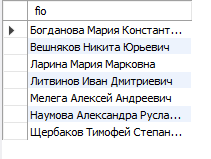
1. Выберите данные из таблицы с применением проверки данных на NULL (IS NULL, IS NOT NULL)

select \* from route where idcountry is not null



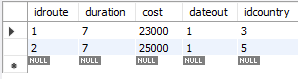
1. Выберите данные из таблицы с применением сортировки по выбранным полям

select fio from tourist order by fio



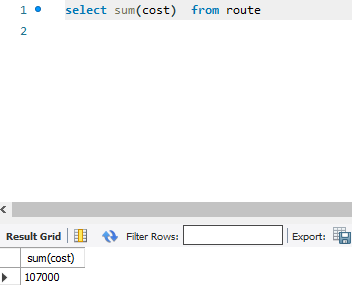
1. Выбрать данные с применением оператора BETWEEN

select \* from route where cost between 20000 and 25000

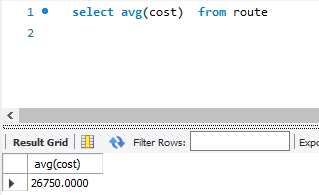


### 1.3.2. Агрегатные функции

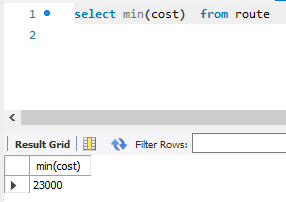
Sum



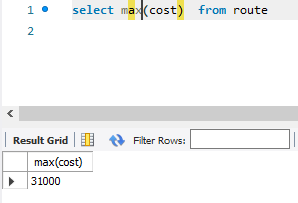
AVG



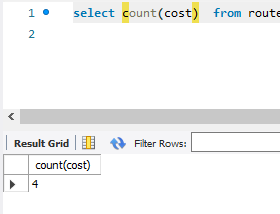
Min



Max



Count



### 1.3.3. Группировка

1. GROUP BY

SELECT sum(cost) as '' FROM route group by cost



1. GROUP BY с использованием условия HAVING

SELECT sum(cost) as 'Cумма' FROM route group by cost having sum(cost) < 28000

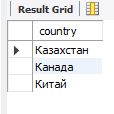


### 1.3.4. Подзапросы

Напишите 3 примера запросов с подзапросами

Первый подзапрос

select country from country where country like 'к%'



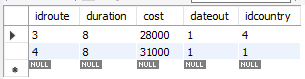
Второй подзапрос

select count(fio) from tourist



Третий подзапрос

select \* from route where duration > 7



### 1.3.5. Соединение таблиц

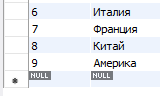
1. Inner Join
2. Left | Right Join
3. Union

### 1.3.6. Запросы на добавление, удаление, изменение данных

Напишите 3 запроса

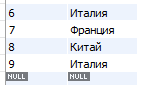
Запрос на добавления

insert into country values(9 ,'Америка')



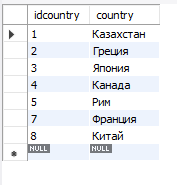
Запрос на изменение данных

update country set country='Италия' where country='Америка'



Запрос на удаление

delete from country where country='Италия';



### 1.3.7. Встроенные функции

1. Условие CASE

SELECT \*,

CASE

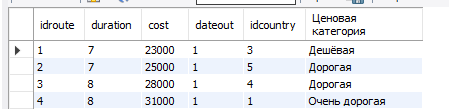
WHEN cost < 25000 THEN 'Дешёвая'

WHEN cost < 30000 THEN 'Дорогая'

else 'Очень дорогая'

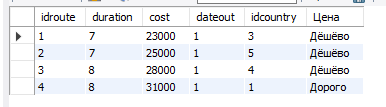
END as 'Ценовая категория'

FROM route



1. Условие IF

SELECT \*, IF(cost =31000,"Дорого","Дёшёво") AS 'Цена' FROM route



1. **Работа со встроенными объектами СУБД MySQL**

## 2.1 Работа с процедурами, функциями и представлениями

### 2.1.1. Основные понятия

Процедура - объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры очень похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных. Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.

Функция - представляет собой модуль, который возвращает значение командой RETURN (вместо аргументов OUT или IN OUT). В отличие от вызова процедуры, который представляет собой отдельный оператор, вызов функции всегда является частью исполняемого оператора, то есть включается в выражение или служит в качестве значения по умолчанию, присваиваемого переменной при объявлении.

Представление - это виртуальная таблица, основанная на результирующем наборе инструкции SQL. Представление содержит строки и столбцы, как и настоящая таблица. Поля в представлении - это поля из одной или нескольких реальных таблиц в базе данных. Вы можете добавить в представление инструкции SQL функций, WHERE и JOIN и представить данные так, как если бы они поступали из одной таблицы.

### 2.1.2. Практическая часть

Напишите 5 хранимых процедур

1)

2)

Напишите 5 хранимых функций

Представления

Напишите 5 представлений

## Пользователи и его права

Пользователь базы данных — это идентификатор имени входа при подключении к базе данных. Имя пользователя базы данных может совпадать с именем входа, но это не является обязательным требованием.

Права — это возможность выполнения конкретного типа SQL-оператора или доступа к объекту базы данных, принадлежащему другому пользователю.

### 2.2.1. Практическая часть. Работа с пользователями и правами через написание команд

1. Создайте нового пользователя со своим именем латиницей.

CREATE USER Artim@localhost IDENTIFIED BY '1234';

1. Присвойте пользователю все привилегии на вашу БД выбрав в ней одну любую таблицу.

grant all privileges on auto.client to 'maria'@'localhost';

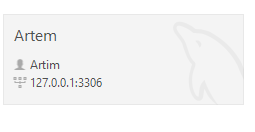
flush privileges;

1. Выведите таблицу mysql.user и убедитесь, что созданный вами пользователь в ней есть.

select \* from mysql.user



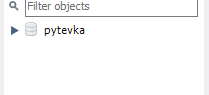
4.Создайте новое подключение для созданного пользователя.



5.Подключитесь используя данное подключение.



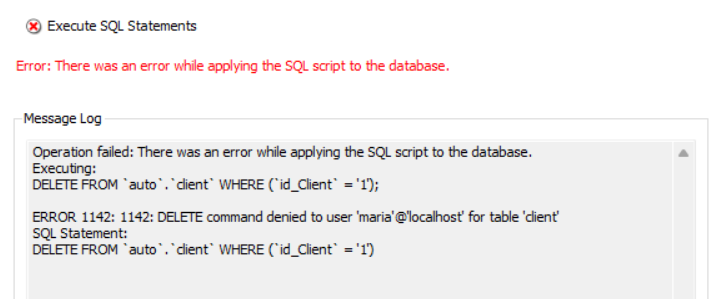
6.Посмотрите, что вам доступна именно та БД и та таблица, которой вы назначили привилегии для вашего пользователя.



7.Удалите данного пользователя.

drop user 'Artim'@'localhost';

8. Убедитесь, что никакие другие действия с таблицей не разрешены.



9. Измените пароль своему пользователю на «1111»

ALTER USER 'maria'@'localhost' IDENTIFIED BY '1111';

10.Перезайдите в СУБД с новым паролем.



1. 11. Удалите пароль своему пользователю.

## 2.3. Индексы

**Выбор индексов в MySQL**

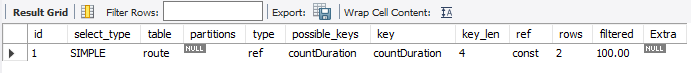
Выполняем запрос без индекса.

explain select \* from route where duration = 7;



Создаем индекс.

CREATE INDEX countDuration ON route(duration);

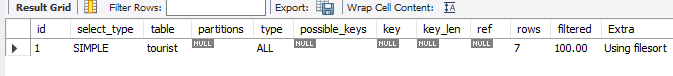
explain select \* from route where duration = 7; 

Колонка rows показывает число записей, которые пришлось прочитать базе данных для выполнения этого запроса.

**Вывод:** запрос после создания индекса «countDuration» работает быстрее.

**Сортировка.**

Выполним сортировку по фио.

explain select \* from tourist order by fio; 

Создадим индекс.

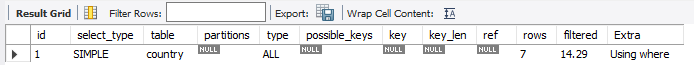
CREATE INDEX fio ON client(FIO);

explain select \* from tourist order by fio 

Вывод: создание индекса для поля «FIO» особо не повлияла на скорость выполнения запроса.

Создаю уникальный индекс для поля «coutry»

EXPLAIN SELECT \* FROM country WHERE country = 'Казахстан';



CREATE UNIQUE INDEX countr ON country(country);

EXPLAIN SELECT \* FROM country WHERE country = 'Казахстан';



**Вывод:** запрос выполняется быстрее с уникальным индексом.

**Составные индексы**

Запрос на выборку данных по адресу и номеру паспорта.

SELECT \* FROM tourist WHERE fio like '%Мелега Алексей Андреевич%' AND passport = '87378472'



explain select \* FROM tourist WHERE fio like '%Мелега Алексей Андреевич%' AND passport = '87378472'

Создаю индекс «adres\_pasportid»

CREATE INDEX fio\_pasport ON tourist(fio, passport);

explain select \* FROM tourist WHERE fio like '%Мелега Алексей Андреевич%' AND passport = '87378472' 

Удаляю индекс adres\_pasportid.

drop index `fio\_pasport` on tourist;

Создаю индекс «pasportid\_adres».

CREATE INDEX passport\_fio ON tourist(passport, fio);

explain select \* FROM tourist WHERE fio like '%Мелега Алексей Андреевич%' AND passport = '87378472';

**Вывод:** в первом случае (без индексов) будет проверено все 10 записей таблицы. После создания индекса «fio\_pasport» будет проверено 10 строк, а при индексе «passport\_fio» уже первая строка окажется нужной. Т.к. при упорядочивании по полю fio будете рассмотрены все варианты для Address like ' Мелега Алексей Андреевич ', а уже потом применено условие Passport = '87378472'. При индексе «passport\_fio» условие PassportID = '873409' будет найдено сразу же.

## 2.4. Исключения

### 2.4.1. Практическая часть

1. Напишите процедуру «create\_table\_route», которая создает таблицу «auto», обработав ошибку номер 1050 (таблица уже существует)

Скрипт:

CREATE DEFINER=`root`@`localhost` PROCEDURE `create\_table\_auto`()

BEGIN

declare exit handler for 1050

select 'Таблица auto уже существует' as msg;

create table auto(

id int primary key,

modelauto varchar(45));

END

Вызываем процедуру:



**2.5 Триггеры**

### 2.5.1 Практическая часть

Напишите несколько (5-6) триггеров для любой из таблиц вашей БД и продемонстрируйте их работу.