Оглавление

[**2. Разработка баз данных в СУБД MySQL** 1](#_Toc90983381)

[**2.1. Построение EDR-модели** 1](#_Toc90983382)

[**2.2.** **Заполнение базы данных** 2](#_Toc90983383)

[**Скрипт:** 2](#_Toc90983384)

[**2.3.** **Написание запросов по работе с данными** 3](#_Toc90983385)

[**2.3.1.** **Выборка и выборка с условие** 3](#_Toc90983386)

[**2.3.2.** **Агрегатные функции** 4](#_Toc90983387)

[**2.3.3.** **Группировка** 5](#_Toc90983388)

[**2.3.4.** **Подзапросы** 5](#_Toc90983389)

[**2.3.5.** **Соединение таблиц** 6](#_Toc90983390)

[**2.3.6.** **Запросы на добавление, удаление, изменение данных** 7](#_Toc90983391)

[**2.3.7.** **Встроенные функции** 7](#_Toc90983392)

[**3. Работа со встроенными объектами СУБД MySQL** 8](#_Toc90983393)

[**3.1.** **Работа с процедурами, функциями и представлениями** 8](#_Toc90983394)

[**3.1.1.** **Основные понятия** 8](#_Toc90983395)

[**3.2.** **Пользователи и его права.** 16](#_Toc90983396)

[**3.2.1.** **Практическая часть. Работа с пользователями и правами через написание команд** 16](#_Toc90983397)

[**3.3.Индексы** 21](#_Toc90983398)

[**3.4. Исключения** 22](#_Toc90983399)

[**Обработка ошибок MySQL в хранимых процедурах** 22](#_Toc90983400)

[**3.5 Триггеры** 25](#_Toc90983401)

[**Вывод:** 29](#_Toc90983402)

[**Приложения:** 29](#_Toc90983403)

[**Приложение1(Модель)** 29](#_Toc90983404)

[**Приложение2(Скрипт)** 29](#_Toc90983405)

## **2. Разработка баз данных в СУБД MySQL**

## **2.1. Построение EDR-модели**

**Вариант 36:**

Издательство – предприятие, занимающееся выпуском разнообразной печатной продукции. Издательство заключает договор с заказчиком (клиентом) на выполнение заказа. Заказчиком может выступать частное лицо или организация. Частное лицо может быть автором издания (или одним из авторов, если их несколько) или представителем автора. Организация для контактов с издательством также имеет своего представителя – контактную персону. Заказ может быть книгой, брошюрой, рекламным проспектом, буклетом, бюллетенем для голосования или каким-либо другим видом издательской продукции. Подготовленные издательством материалы заказчика печатаются в типографиях, где издательство размещает свои заказы. Информационная система должна позволять анализировать информацию о заказах, типографиях, авторах и т.п

**Модель:**

См.[Приложение1](#_Приложение1(Модель))

### **2.2. Заполнение базы данных**

|  |  |  |  |
| --- | --- | --- | --- |
| **Agreements** | | | |
| Поле | Тип | Атрибуты | Описание |
| id | int | Pk, NN, AI | Уникальный номер. |
| number | Varchar(80) | - | Номер договора |
| DataBegin | Date | - | Дата начала действия договора |
| DataOver | Date | - | Дата завершения договора |
| Payment | int | - | Гонорар |
| СontactPersons\_id | int | NN | Уникальный номер контактного лица |
| TypeOfOrder\_id | int | NN | Уникальный номер типа товара |

|  |  |  |  |
| --- | --- | --- | --- |
| **СontactPersons** | | | |
| Поле | Тип | Атрибуты | Описание |
| id | int | Pk, NN, AI | Уникальный номер. |
| SecondNameOrNameOrganization | Varchar(45) | - | Фамилия контакного лица или название фирмы |
| Spezialization\_id | int | NN | Уникальный номер специализации |

|  |  |  |  |
| --- | --- | --- | --- |
| **PrintingHouse** | | | |
| Поле | Тип | Атрибуты | Описание |
| id | int | Pk, NN, AI | Уникальный номер. |
| Name | Varchar(45) | - | Название типографии |
| adress | Varchar(45) | - | Адресс типографии |

|  |  |  |  |
| --- | --- | --- | --- |
| **PublisherContracts** | | | |
| Поле | Тип | Атрибуты | Описание |
| id | int | Pk, NN, AI | Уникальный номер. |
| Agreements\_id | int | NN | Уникальный номер договора |
| PrintingHouse\_id | int | NN | Уникальный номер типографии |
| CountcopiesProduct | int | - | Количество копий продукции требуемой к производству |
| OrderPrice | int | - | Цена за произдвоство |

|  |  |  |  |
| --- | --- | --- | --- |
| **TypeOforder** | | | |
| Поле | Тип | Атрибуты | Описание |
| id | int | Pk, NN, AI | Уникальный номер. |
| TypeOforder | Varchar(45) | - | Тип товара |

|  |  |  |  |
| --- | --- | --- | --- |
| **Spezialization** | | | |
| Поле | Тип | Атрибуты | Описание |
| id | int | Pk, NN, AI | Уникальный номер. |
| Proffesion | Varchar(45) | - | Тип профессии |

## **Скрипт:**

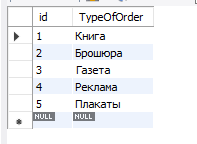
См.[Приложение2](#_Приложение2(Скрипт))

## **2.3. Написание запросов по работе с данными**

# **2.3.1. Выборка и выборка с условие**

1. выбрать данные из одной таблицы

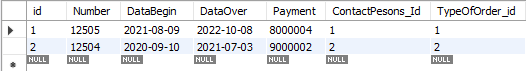
SELECT \* FROM publisher.typeoforder;



1. выбрать данные из одной таблицы по условию WHERE

select \* from agreements

where payment > 8000000;



1. выбрать данные с применением команды DISTINCT

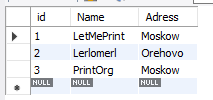
select distinct Adress from printinghouse;



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

select \* from printinghouse

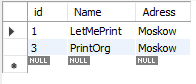
where Adress IN('Moskow','Orehovo');



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

select \* from printinghouse

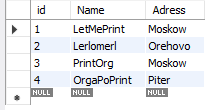
where adress like 'Mos%';



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

select \* from printinghouse

where adress is not null;



select \* from printinghouse

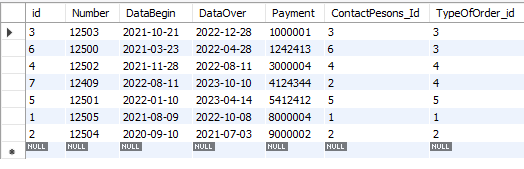
where adress is null;



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

select \* from agreements

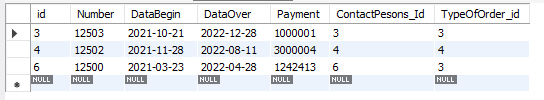
order by Payment;



1. выбрать данные с применением оператора BETWEEN

select \* from agreements

where payment between 1000001 and 3000004;



# **2.3.2. Агрегатные функции**

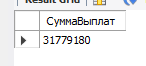
1. Avg

select avg(payment) as СреднееЗначениеВыплат from agreements;



1. Sum

select sum(payment) as СуммаВыплат from agreements;



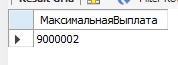
1. Min

select min(payment) as МинимальнаяВыплата from agreements;



1. Max

select max(payment) as МаксимальнаяВыплата from agreements;



1. Count

Select count(\*) as КолВоДоговоров from agreements;

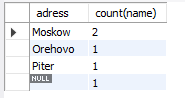


## **2.3.3. Группировка**

1. Напишите пример запроса с применением группировки GROUP BY

select adress, count(name) from printinghouse

group by adress;

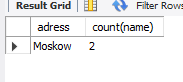


1. Напишите пример запроса с применением группировки GROUP BY с использованием условия HAVING

select adress, count(name) from printinghouse

group by adress

having count(name) > 1;

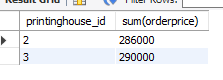


1. Напишите пример запроса с применением группировки GROUP BY с использованием условия HAVING

select printinghouse\_id, sum(orderprice) from publishercontracts

group by printinghouse\_id

having sum(orderprice) > 250000;



# **2.3.4. Подзапросы**

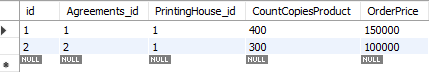
Напишите три пример использования подзапросов

1.

Select \* from publishercontracts

where printinghouse\_id=(select id from printinghouse

where name = "LetMePrint");

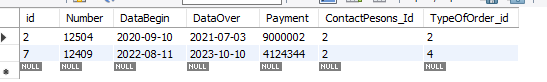


2.

Select \* from agreements

where ContactPesons\_id=(select id from contactpesons

where SecondNameOrNameOrganization = "Шаханов");

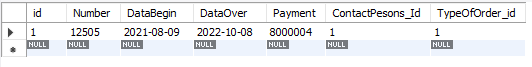


3.

Select \* from agreements

where ContactPesons\_id=(select id from contactpesons

where SecondNameOrNameOrganization = "Толстов");



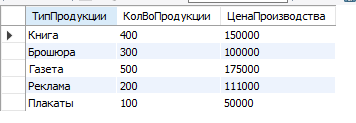
# **2.3.5. Соединение таблиц**

1. Напишите запрос с внутренним соединением таблиц (Inner Join)

select typeoforder as ТипПродукции,CountCopiesProduct as КолВоПродукции , orderprice as ЦенаПроизводства

from typeoforder inner join publishercontracts

on publishercontracts.id = typeoforder.id;

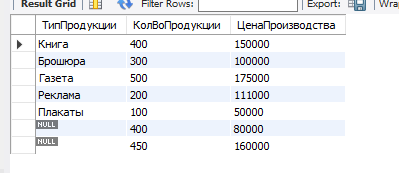


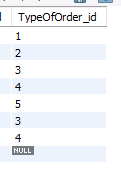
1. Напишите запрос с внешним соединением таблиц (Left | Right Join)

select typeoforder as ТипПродукции,CountCopiesProduct as КолВоПродукции , orderprice as ЦенаПроизводства

from typeoforder right OUTER JOIN publishercontracts

on publishercontracts.id = typeoforder.id;

  
Повторение данных запрос их считает за null



1. Напишите запрос объединения таблиц (Union)

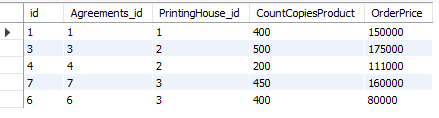
select \* from publishercontracts

where orderprice > 100000

UNION

select \* from publishercontracts

where CountCopiesProduct > 300;



# **2.3.6. Запросы на добавление, удаление, изменение данных**

1. Запрос на добавление

INSERT INTO typeoforder (TypeOfOrder) values('Беллютени')



1. Запрос на удаление

DELETE FROM typeoforder

WHERE TypeOfOrder='TestOrder';



1. Запрос на изменение

UPDATE typeoforder SET TypeOfOrder = 'Беллютенички' WHERE TypeOfOrder='Беллютени'



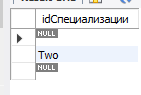
# **2.3.7. Встроенные функции**

1. Напишите запрос с использованием условия CASE

Select case specialization\_name when 'Автор' then 'Two'

end as idCпециализации

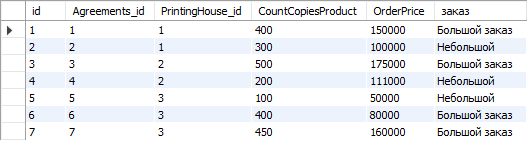
from specialization;



1. Напишите запрос с использованием условия IF

Select \*, if(CountCopiesProduct > 300, 'Большой заказ', 'Небольшой') as заказ

from publishercontracts;



## **3. Работа со встроенными объектами СУБД MySQL**

# **3.1. Работа с процедурами, функциями и представлениями**

## **3.1.1. 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 (как DDL, так и DML). Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.

**Функции** — выполняют произвольный список операторов SQL и возвращают результат последнего запроса в списке. В простом случае (не с множеством) будет возвращена первая строка результата последнего запроса. Если последний запрос вообще не вернёт строки, будет возвращено значение NULL.

**Представления** — особый объект, который содержит данные, полученные запросом SELECT из обычных таблиц. Это виртуальная таблица, к которой можно обратиться как к обычным таблицам и получить хранимые данные. Представление в SQL может содержать в себе как данные из одной единственной таблицы, так и из нескольких таблиц.

* + 1. **Практическая часть**

**Процедуры.**

1. Fn\_add\_contactpesons

Описание: добавляет строку в страницу contactpersons

Код процедуры:

CREATE DEFINER=`root`@`localhost` PROCEDURE `fn\_add\_contactpersons`()

BEGIN

insert into contactpesons values(7,"Лемфорт",2);

END

Вызов процедуры:

call publisher.pr\_add\_contactpersons();

Результат процедуры:



1. Pr\_Sum\_payment

Описание: выводит сумму выплат

Код процедуры:

CREATE DEFINER=`root`@`localhost` PROCEDURE `pr\_sum\_payment`()

BEGIN

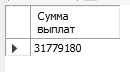
select sum(payment) as 'Cумма выплат' from agreements;

END

Вызов процедуры:

call publisher.pr\_sum\_payment();

Результат процедуры:



1. Напишите процедуру работы с диапазоном (границы диапазона задаются в качестве параметров)

Описание: выводит все строки с таблицы agreements, где значения оплаты от 100001 до 5412412.

Код процедуры:

CREATE DEFINER=`root`@`localhost` PROCEDURE `pr\_beetween`()

BEGIN

select \* from agreements

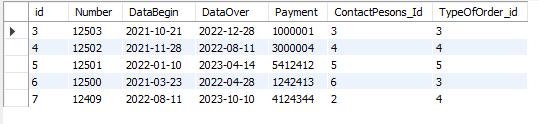
where payment between 1000001 and 5412412;

END

Вызов процедуры:

call publisher.pr\_beetween();

Результат процедуры:



1. Создайте процедуру, демонстрирующую работу с условными потоками в процедуре. (CASE… WHEN или IF… ELSE)

Описание: процедура с if, else

Код процедуры:

CREATE DEFINER=`root`@`localhost` PROCEDURE `pr\_if\_proceduer`()

BEGIN

SET @sum = (SELECT sum(payment) FROM agreements);

IF @sum< 5412412 THEN

SELECT "Мало" as "Траты";

ELSE

SELECT "Много" as "Траты";

end if;

END

Вызов процедуры:

call publisher.pr\_if\_proceduer();

Результат процедуры:



1. Напишите процедуру выводящую результат объединения (join) двух таблиц.

Описание: объединения двух таблицы, вызов информации о имени контактной персоны и ее специализации.

Код процедуры:

CREATE DEFINER=`root`@`localhost` PROCEDURE `pr\_procedure\_join`()

BEGIN

SELECT

contactpesons.SecondNameOrNameOrganization,

specialization.specialization\_name

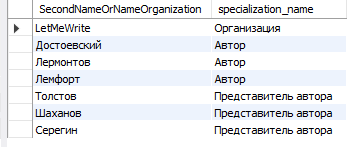
FROM contactpesons inner join specialization on specialization.id = contactpesons.Specialization\_id;

END

Вызов процедуры:

call publisher.pr\_procedure\_join();

Результат процедуры



**Функции.**

1. Напишите функцию выводящую количество записей по указанному критерию. (например, количество сотрудников на указанную букву)

Описание: выводит количество авторов

Код функции:

CREATE DEFINER=`root`@`localhost` FUNCTION `fn\_count\_zapies`() RETURNS int

READS SQL DATA

DETERMINISTIC

BEGIN

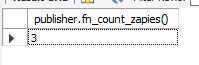
RETURN (SELECT count(id) as количествоавторов from contactpesons WHERE Specialization\_id LIKE 2);

END

Вызов функции:

select publisher.fn\_count\_zapies();

Результат:



1. Напишите функцию, которая использует в запросе к данным min или max.

Описание: максимальное значение выплаты

Код функции:

CREATE DEFINER=`root`@`localhost` FUNCTION `fn\_max`() RETURNS int

READS SQL DATA

DETERMINISTIC

BEGIN

RETURN (select max(payment) from agreements);

END

Вызов функции:

select publisher.fn\_max();

Результат:



1. Создайте функцию, которая работает с курсором, перебирает записи по средствам цикла (LOOP) и обновляет значения некоторых полей исходя из условия.

Описание: добавление 5 значений в таблицу typeoforder.

Код функции:

CREATE DEFINER=`root`@`localhost` FUNCTION `Func\_loop`(typeoforder\_1 varchar(45)) RETURNS varchar(45) CHARSET utf8mb3

READS SQL DATA

DETERMINISTIC

BEGIN

SET @counter =0;

read\_THEloop: LOOP

IF @counter >3 THEN

LEAVE read\_THEloop;

end IF;

SET @id = (SELECT max(id) FROM typeoforder)+1;

set @typeof =(typeoforder\_1);

INSERT into typeoforder values(@id,@typeof);

SET @counter= @counter + 1;

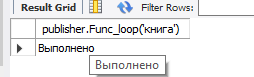
end LOOP;

RETURN 'Выполнено';

END

Вызов функции: select publisher.Func\_loop('книга');

Результат:



1. Напишите функцию удаления записей по указанному критерию. (Значение критерий вводится в виде входного параметра)

Описание: удаление записи из таблицы printinghouse

Код функции:

CREATE DEFINER=`root`@`localhost` FUNCTION `fn\_delete`(choise VARCHAR(45)) RETURNS varchar(45) CHARSET utf8mb3

READS SQL DATA

DETERMINISTIC

BEGIN

DELETE FROM printinghouse

WHERE name like choise;

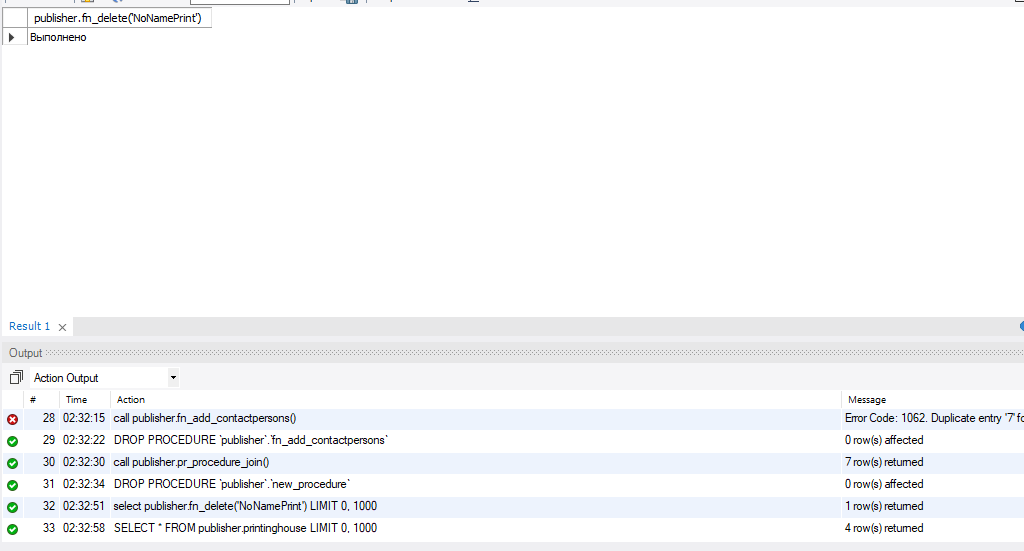
RETURN 'Выполнено';

END

Вызов функции:

select publisher.fn\_delete('NoNamePrint');

Результат:



1. Напишите функцию, которая пробует добавить новую запись в таблицу и если запись с такими значениями уже есть выдает соответствующее сообщение.

Описание: добавление контактного лица

Код функции:

CREATE DEFINER=`root`@`localhost` FUNCTION `fn\_add`(name\_person VARCHAR(45),Specialization\_id int) RETURNS varchar(45) CHARSET utf8mb3

READS SQL DATA

DETERMINISTIC

BEGIN

IF (SELECT count(SecondNameOrNameOrganization) from contactpesons where SecondNameOrNameOrganization = name\_person )>0

THEN return "Уже существует";

ELSE

SET @id = (SELECT max(id) FROM contactpesons)+1;

INSERT into contactpesons values(@id,name\_person,Specialization\_id);

return "Выполнено";

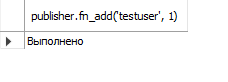
end IF;

END

Вызов функции:

select publisher.fn\_add('testuser', 1);

Результат:





**Представления.**

1. Создайте представление, которое объединяет данные из двух таблиц. На основе созданного представления сделайте выборку по условию where.

Описание: Обьединение двух таблиц

Код представления:

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `tolstovbdpublisher`.`view\_union` AS

SELECT DISTINCT

`tolstovbdpublisher`.`printinghouse`.`Name` AS `nameprinting`,

`tolstovbdpublisher`.`printinghouse`.`Adress` AS `adressprinting`,

`tolstovbdpublisher`.`publishercontracts`.`OrderPrice` AS `pricework`,

`tolstovbdpublisher`.`publishercontracts`.`Agreements\_id` AS `id\_dogovor`

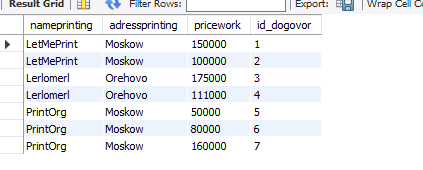
FROM

(`tolstovbdpublisher`.`publishercontracts`

JOIN `tolstovbdpublisher`.`printinghouse` ON ((`tolstovbdpublisher`.`publishercontracts`.`PrintingHouse\_id` = `tolstovbdpublisher`.`printinghouse`.`id`)))

Вызов представления: SELECT \* FROM tolstovbdpublisher.view\_union;

Результат:



1. Создайте представление, включающее агрегатную функцию.

Описание: агрегатная функция

Код представления:

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `tolstovbdpublisher`.`agregat\_view` AS

SELECT

COUNT(0) AS `count(\*)`

FROM

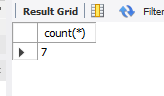
`tolstovbdpublisher`.`contactpesons`

WHERE

(`tolstovbdpublisher`.`contactpesons`.`SecondNameOrNameOrganization` = `tolstovbdpublisher`.`contactpesons`.`SecondNameOrNameOrganization`)

Вызов представления: SELECT \* FROM tolstovbdpublisher.agregat\_view;

Результат:



1. Создайте ещё **три** любые представления с выборкой данных. И покажите пример их использования.
2. View\_3

Код:

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `view\_3`

AS select distinct `agreements`.`Number` AS `НомерДоговора`

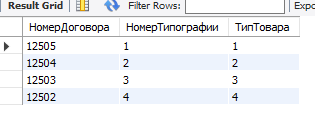
,`publishercontracts`.`PrintingHouse\_id` AS `НомерТипографии`

,`agreements`.`TypeOfOrder\_id` AS `ТипТовара`

from (`agreements` join `publishercontracts` on((`publishercontracts`.`PrintingHouse\_id` = `agreements`.`id`)))

Вызов представления: SELECT \* FROM publisher.view\_3;

Результат:



1. View\_4

Код:

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `view\_4`

AS select distinct

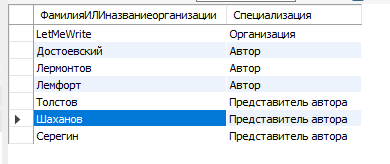
`contactpesons`.`SecondNameOrNameOrganization` AS `ФамилияИЛИназваниеорганизации`

,`specialization`.`specialization\_name` AS `Специализация`

from (`contactpesons` join `specialization` on((`specialization`.`id` = `contactpesons`.`Specialization\_id`)))

Вызов представления: SELECT \* FROM publisher.view\_4;

Результат:



1. View\_5

Код:

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `view\_5`

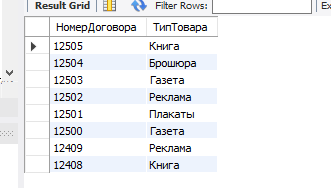
AS select distinct `agreements`.`Number` AS `НомерДоговора`

,`typeoforder`.`TypeOfOrder` AS `ТипТовара`

from (`agreements` join `typeoforder` on((`typeoforder`.`id` = `agreements`.`TypeOfOrder\_id`)))

Вызов представления: SELECT \* FROM publisher.view\_5;

Результат:



## **Пользователи и его права.**

**Пользователи и его права** нужны чтобы чужие БД не были доступны абсолютному каждому пользователю на сервере, существует система пользователей для этих баз данных. Сам доступ к какой-либо БД может быть назначен администратором (либо уполномоченным пользователем) другому пользователю, причем он может быть полным или в некоторой степени ограниченным. Более конкретно эта степень доступа выражается в привилегиях («правах» или «разрешениях»).

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

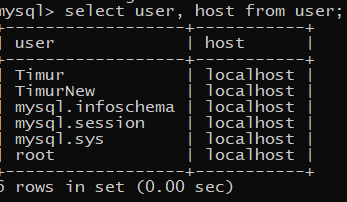
1. Создайте нового пользователя со своим именем латиницей (например: sergey).



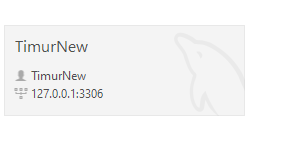
1. Присвойте пользователю все привилегии на вашу БД выбрав в ней одну любую таблицу.



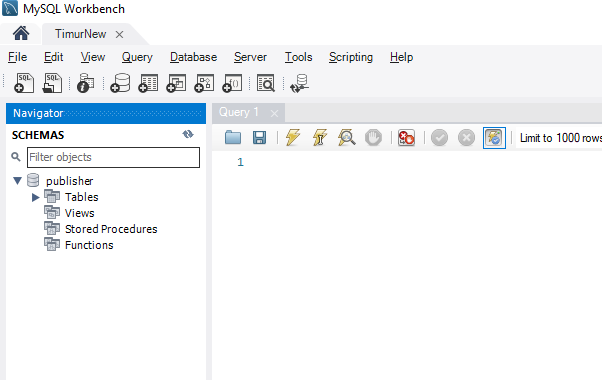
1. Выведите таблицу mysql.user и убедитесь, что созданный вами пользователь в ней есть.



1. Создайте новое подключение для созданного пользователя.



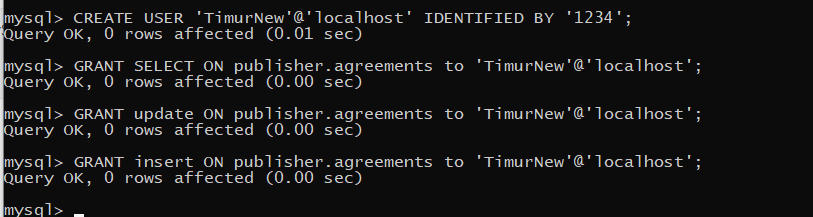
1. Подключитесь используя данное подключение.
2. Посмотрите, что вам доступна именно та БД и та таблица к торой вы назначили привилегии для вашего пользователя.



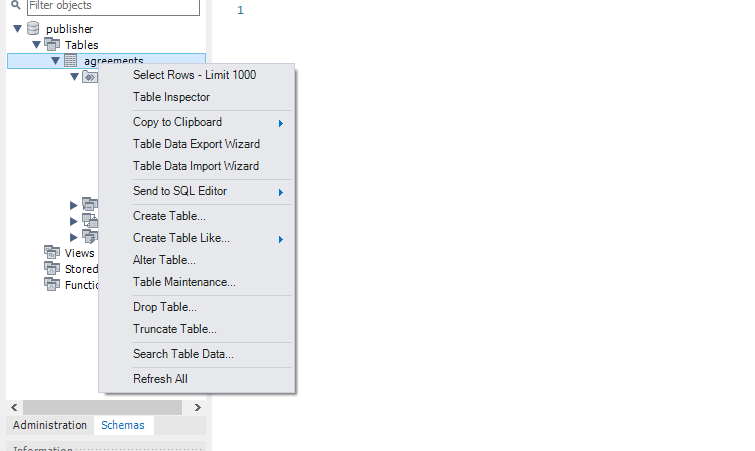
1. Удалите данного пользователя.



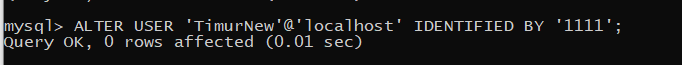
1. Создайте заново его-же, но уже с привилегиями: SELECT, UPDATE, INSERT.



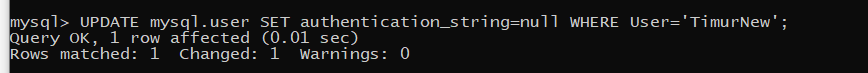
1. Убедитесь, что никакие другие действия с таблицей не разрешены.



1. Измените пароль своему пользователю на «1111»

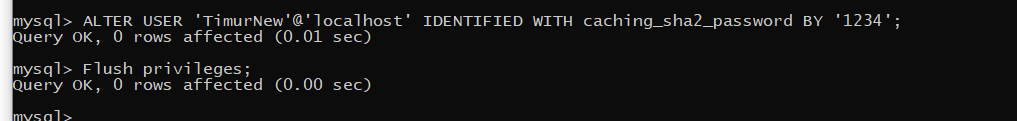


1. Перезайдите в СУБД с новым паролем.
2. Удалите пароль своему пользователю.



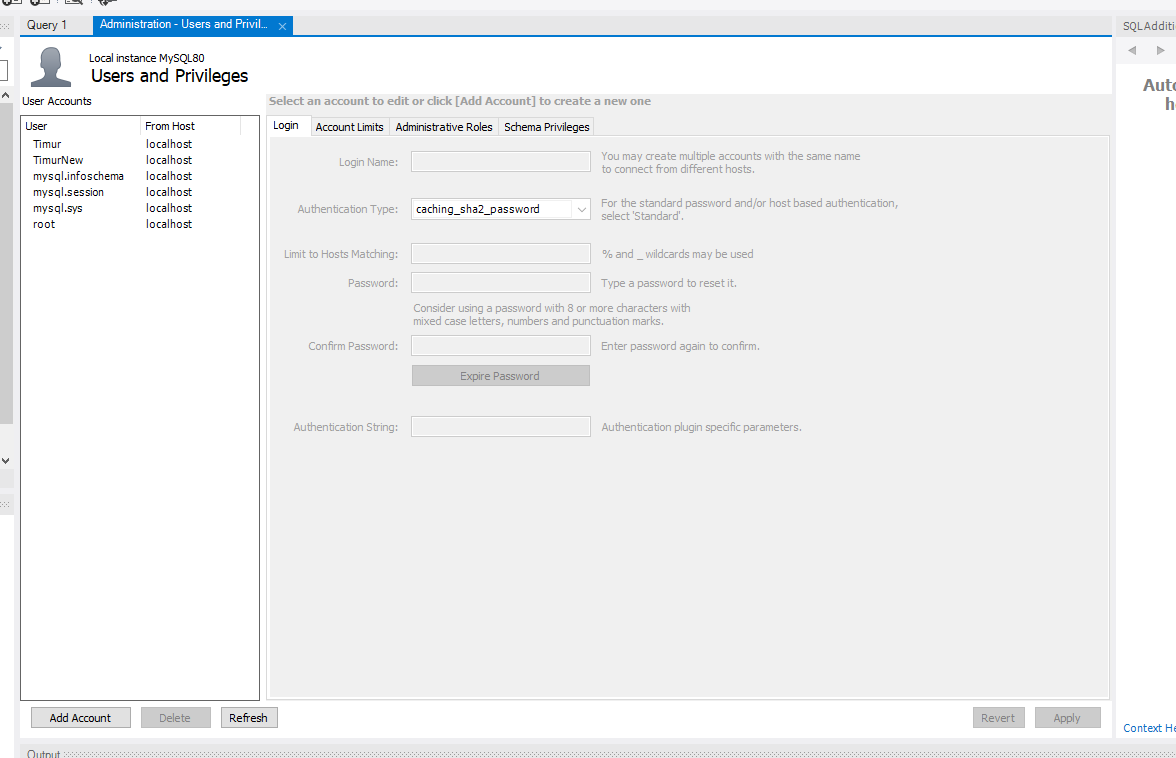


1. Перезайдите заново в СУБД (*уже не вводя пароль*)
2. Добавьте пароль своему пользователю «1234»

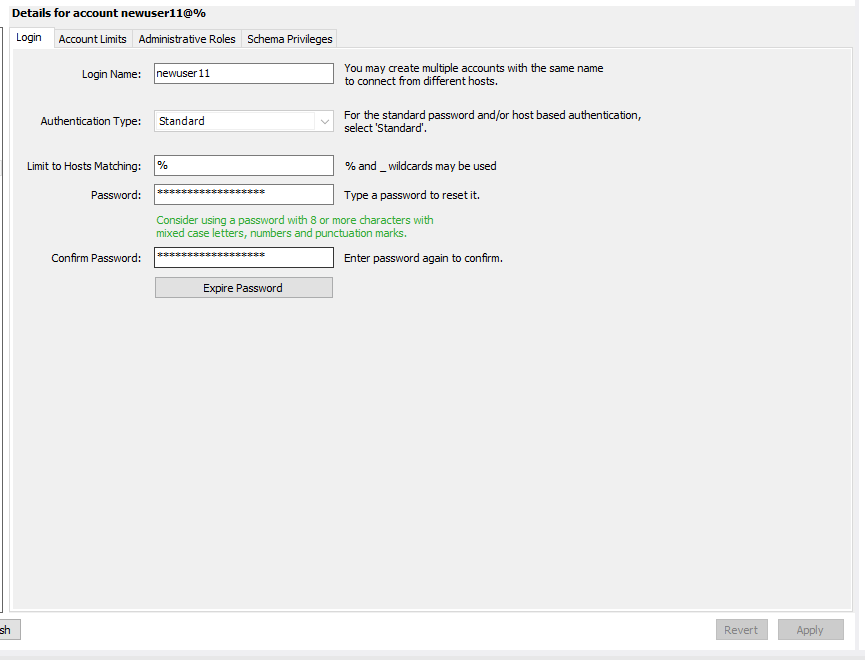


**Часть 2. Работа в окне Users and Privileges.**

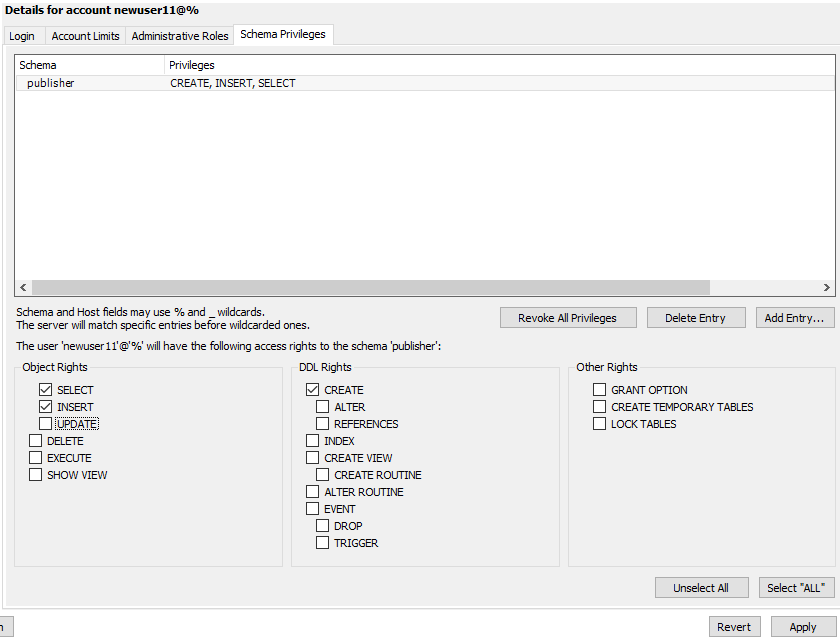
1. Подключитесь к СУБД с правами root.
2. Перейдите в окно «Users and Privileges»



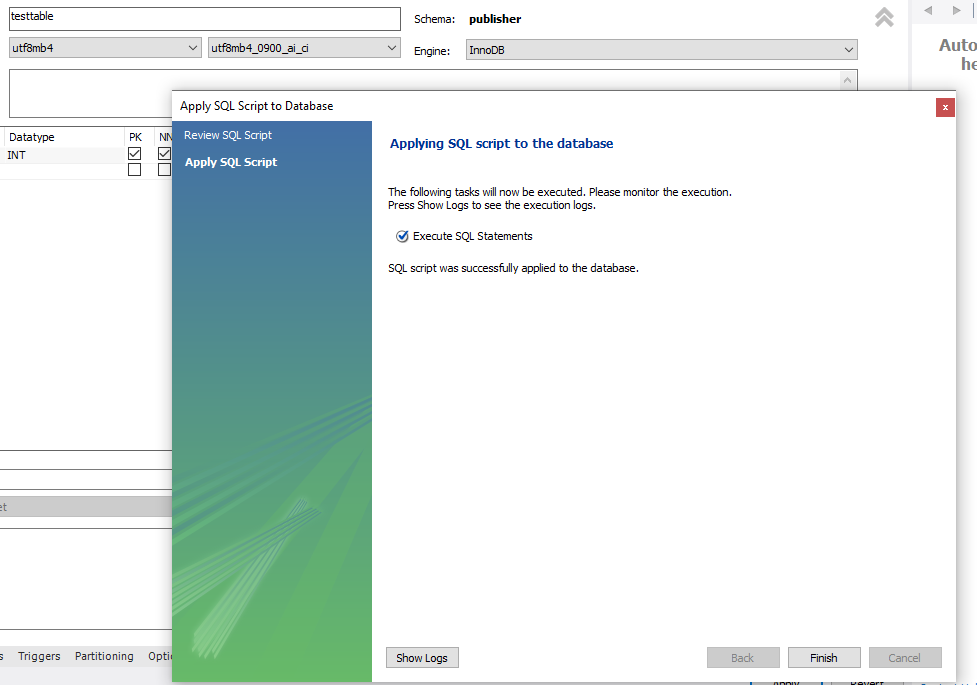
1. Создайте нового пользователя «Своем имя» и назначьте ему пароль.



1. Назначьте ему привилегии на выборку, добавление данных и добавление таблиц.



1. Создайте новое подключение с созданным пользователем. Убедитесь, что привилегии работают правильно. Создайте таблицу и попробуйте её удалить; вставьте описание ошибки с запретом на удаление таблицы.



Ошибка: Error Code: 1142. DROP command denied to user 'newuser11'@'localhost' for table 'testtable'

# **3.3.Индексы**

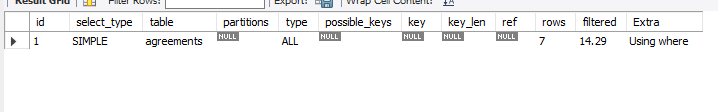
**Индексы** - это специальные таблицы, которые могут быть использованы поисковым двигателем базы данных (далее – БД), для ускорения получения данных.

* + 1. **Практическая часть**

1. **Индекс (Уникальный индекс)**

Код запроса: EXPLAIN SELECT \* FROM agreements WHERE payment = 8000004;

Выполнение запроса без индекса:

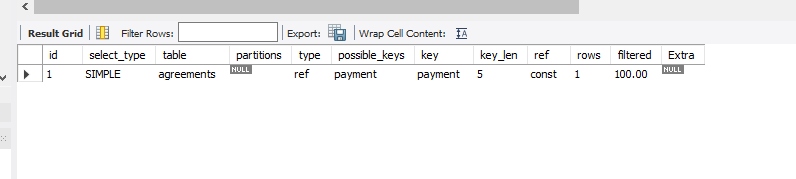


Rows: 7

Duration: 0.015 s

Создание индекса: CREATE INDEX payment ON agreements(payment);

Выполнение запроса с индексом:



Rows: 1

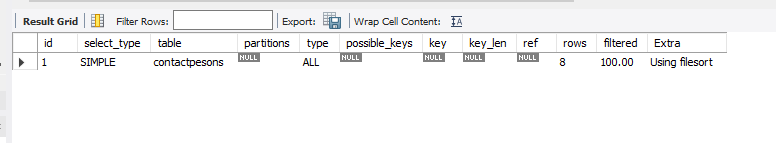
Duration: 0.000 s

**Вывод:** Запрос стал работать быстрее.

1. **Индекс(Сортировка по Фамилии/название организации)**

Код запроса: EXPLAIN SELECT \* FROM contactpesons order by SecondNameOrNameOrganization;

Выполнение запроса без индекса:

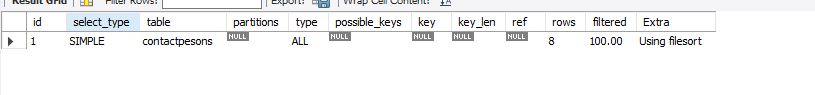


Rows: 8

Duration: 0.000 s

Создание индекса: CREATE INDEX SecondNameOrNameOrganization ON contactpesons(SecondNameOrNameOrganization);

Выполнение запроса с индексом:



Rows:8

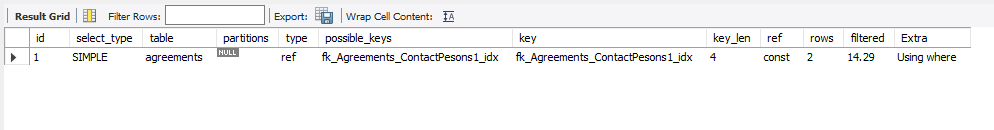
Duration:0.000 s

**Вывод:** Не влияет на скорость выполнения запроса

1. **Индекс(Составной)**

Код запроса: EXPLAIN SELECT \* FROM agreements WHERE number = '12409' AND ContactPesons\_Id = 2;

Выполнение запроса без индекса:

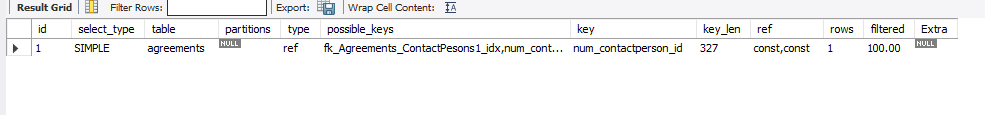


Duration: 0.000 s

Rows: 2

Создание индекса: create index num\_contactperson\_id on agreements(number, ContactPesons\_id)

Выполнение запроса с индексом:



Duration: 0.000 s

Rows: 1

**Вывод:** при выполнении запроса без индекса будет проверено 2 записи таблицы, после создания индекса, будет проверена одна запись. Не повлияло на скорость выполнение запроса.

# **3.4. Исключения**

**Исключения** – это ситуации, которые не должны возникать при нормальном выполнении программы. К числу исключений относятся: ошибки, генерируемые системой (например, нехватка памяти или повторяющееся значение индекса);

* + 1. **Практическая часть**

**Обработка ошибок MySQL в хранимых процедурах**

1.Напишите процедуру «create\_table\_<имя вашей таблицы>», которая создает таблицу «таблица из вашей БД», обработав ошибку номер 1050 (таблица уже существует)

КОД:

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

BEGIN

DECLARE EXIT HANDLER FOR 1050

SELECT ('Таблицы agreements уже сущетсвует!') as msg;

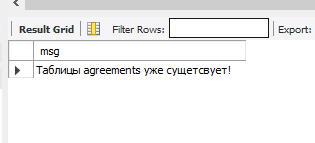
Create table agreements(

id int);

END

Вызов: call tolstovbdpublisher.create\_table();

Результат:



2.Напишите процедуру «found\_table\_<имя вашей таблицы>», которая выбирает данные из указанной таблицы. Если такой таблицы нет, то выводится сообщение об ошибке 1146 (Таблица не существует)

КОД:

CREATE DEFINER=`root`@`localhost` PROCEDURE `found\_table`(in tbl\_name varchar(45))

BEGIN

DECLARE EXIT HANDLER FOR 1146

SELECT concat('Таблицы "',tbl\_name,'" не сущетсвует!') as msg;

set @qr = concat('select \* from ',tbl\_name);

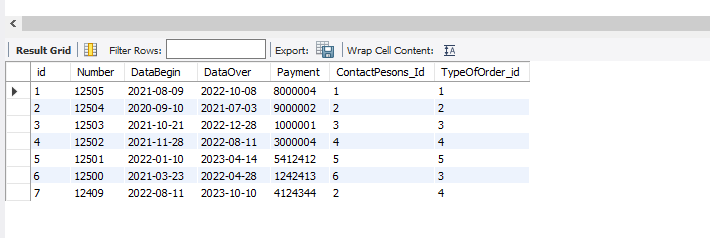
PREPARE reset\_stmt FROM @qr;

EXECUTE reset\_stmt;

END

Вызов: call tolstovbdpublisher.found\_table('agreements');

Результат:



3.Напишите процедуру «select\_column\_<имя вашей таблицы>», которая выводит данные указанного столбца таблицы «имя вашей таблицы», а в случае если такого столбца нет, выводит ошибку. (код ошибки 1054)

Код:

CREATE DEFINER=`root`@`localhost` PROCEDURE `agreements\_col`(in col\_name varchar(45))

BEGIN

-- declare qr varchar;

DECLARE EXIT HANDLER FOR 1054

SELECT concat('Column doesnt exists: "',col\_name,'" ') as msg;

set @qr = concat('select ', col\_name,' from agreements');

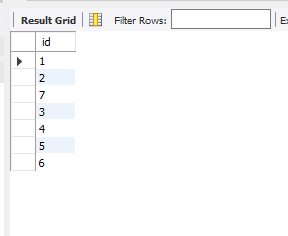
PREPARE reset\_stmt FROM @qr;

EXECUTE reset\_stmt;

END

Вызов: call tolstovbdpublisher.agreements\_col('id');

Результат:



4.Используя [таблицу ошибок sql](https://faqed.ru/web/databases/error-codes-mysql-server.htm) создайте ещё любые две процедуры с обработкой ошибок.

1. Процедура:

Код:

CREATE DEFINER=`root`@`localhost` PROCEDURE `Dublicate\_1062\_iskl`(IN id\_ag INT, IN id\_t INT, IN id1 INT)

BEGIN

DECLARE CONTINUE HANDLER FOR 1062

SELECT CONCAT('duplicate keys (',id1, ',' ,id\_ag,',',id\_t,') found') AS msg;

-- insert a new record into article\_tags

INSERT INTO agree(id1,id\_ag,id\_t)

VALUES(id1,id\_ag,id\_t);

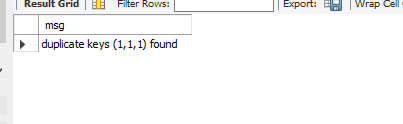
-- return tag count for the article

SELECT COUNT(\*) FROM agree;

END

Вызов: call publisher.Dublicate\_1062\_iskl(1, 1, 1);

Результат:



1. Процедура

Код:

CREATE DEFINER=`root`@`localhost` PROCEDURE `contactpesons\_col`(in col\_name varchar(45))

BEGIN

-- declare qr varchar;

DECLARE EXIT HANDLER FOR 1054

SELECT concat('Column doesnt exists: "',col\_name,'" ') as msg;

set @qr = concat('select ', col\_name,' from contactpesons');

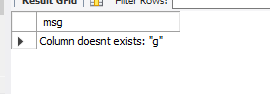
PREPARE reset\_stmt FROM @qr;

EXECUTE reset\_stmt;

END

Вызов: call publisher.contactpesons\_col('g');

Результат:



# **3.5 Триггеры**

**Триггеры** - представляют специальный тип хранимой процедуры, которая вызывается автоматически при выполнении определенного действия над таблицей или представлением, в частности, при добавлении, изменении или удалении данных, то есть при выполнении команд INSERT, UPDATE, DELETE.

* + 1. **Практическая часть**

1. **Insert** Triggers
2. BeforeInsert

Описание: проверяет цену производства(<180000) при добавлении записи в таблицу publishercontracts

Код:

CREATE TRIGGER checkOrderPrice

BEFORE INSERT

ON publishercontracts

FOR EACH ROW

IF NEW.OrderPrice >180000 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Цена производства не может быть больше 180000';

END IF//

DELIMITER ;

Проверка:

Insert into publishercontracts(id, agreements\_id, PrintingHouse\_id, CountCopiesProduct,OrderPrice) values('8','1','1',450, 185000)

Error Code: 1644. Цена производства не может быть больше 180000

1. AfterInsert

Описание: после добавление записи в таблицу agreements, в таблицу agreements\_status вносится id новой записи и сообщение о потверждении добавлении записи.

Код:

DELIMITER //

CREATE TRIGGER agreements\_status

AFTER INSERT

ON agreements

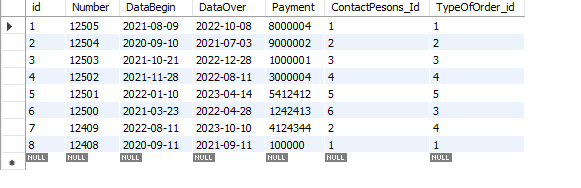
FOR EACH ROW

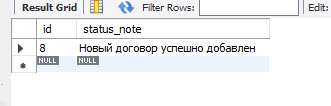
Insert into agreements\_status(id, status\_note) VALUES(NEW.id, 'Новый договор успешно добавлен')//

DELIMITER ;

Проверка:

insert into agreements(id,number,databegin,dataover,payment,contactpesons\_id,typeoforder\_id) values('8','12408','2020-09-11','2021-09-11','100000','1','1');





1. **Update** Triggers
2. BeforeUpdate

Описание: до обновление какой либо записи в таблице contactpersons триггер проверяет изменялось ли значение специализации.

Код:

CREATE TRIGGER spezialization\_update

BEFORE UPDATE

ON contactpesons

FOR EACH ROW

IF OLD.Specialization\_id ='1' or '2' or '3' THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Изменение специализации невозможно';

END IF //

DELIMITER ;

Проверка:

Update contactpesons set Specialization\_id='2' where id='1';

Error Code: 1644. Изменение специализации невозможно

1. AfterUpdate

Описание: после обновления в таблице agreements столбца payment в таблицу agreements\_log вносится следующая информация: id обновленной записи, прошлая сумма оплаты, новая суммы оплаты, кем была обновлена запись, когда была обновлена запись.

Код:

DELIMITER //

CREATE TRIGGER log\_agreements\_update

AFTER UPDATE

ON agreements

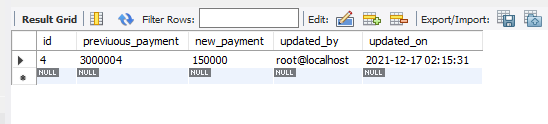
FOR EACH ROW

Insert into agreements\_log(id, previuous\_payment, new\_payment, updated\_by, updated\_on) VALUES (NEW.id,OLD.payment, NEW.payment,(SELECT USER()), NOW() )//

DELIMITER ;

Проверка:

update agreements set payment = '150000' where id = 4;



1. **Delete** Triggers
2. BeforeDelete

Описание: данный триггер не позволяет удалить контакную персону пока с ней заключен договор

Код:

DELIMITER //

CREATE TRIGGER beforedelete\_record

BEFORE DELETE

ON contactpesons

FOR EACH ROW

IF OLD.id in (select ContactPesons\_Id from agreements) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Невозможно удалить контакную персону пока с ней заключен договор';

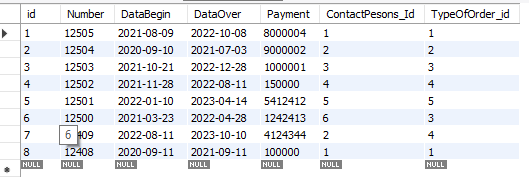
END IF//

DELIMITER ;

Проверка:

Delete from contactpesons where id='6';

Error Code: 1644. Невозможно удалить контакную персону пока с ней заключен договор



1. AfterDelete

Описание: данный триггер удаляет контакную персону после удаления с ней связанного(ых) договора(ов).

Код:

DELIMITER //

CREATE TRIGGER afterdelete\_agreements

AFTER DELETE

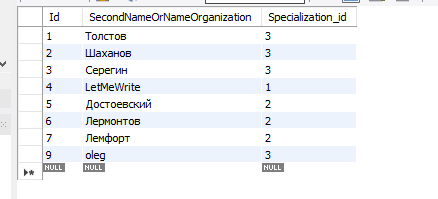
ON agreements

FOR EACH ROW

Delete from contactpesons where id=OLD.ContactPesons\_Id;//

DELIMITER ;

Проверка:



Delete from agreements where ContactPesons\_Id='9'

## **Вывод:**

Достигнутые цели:

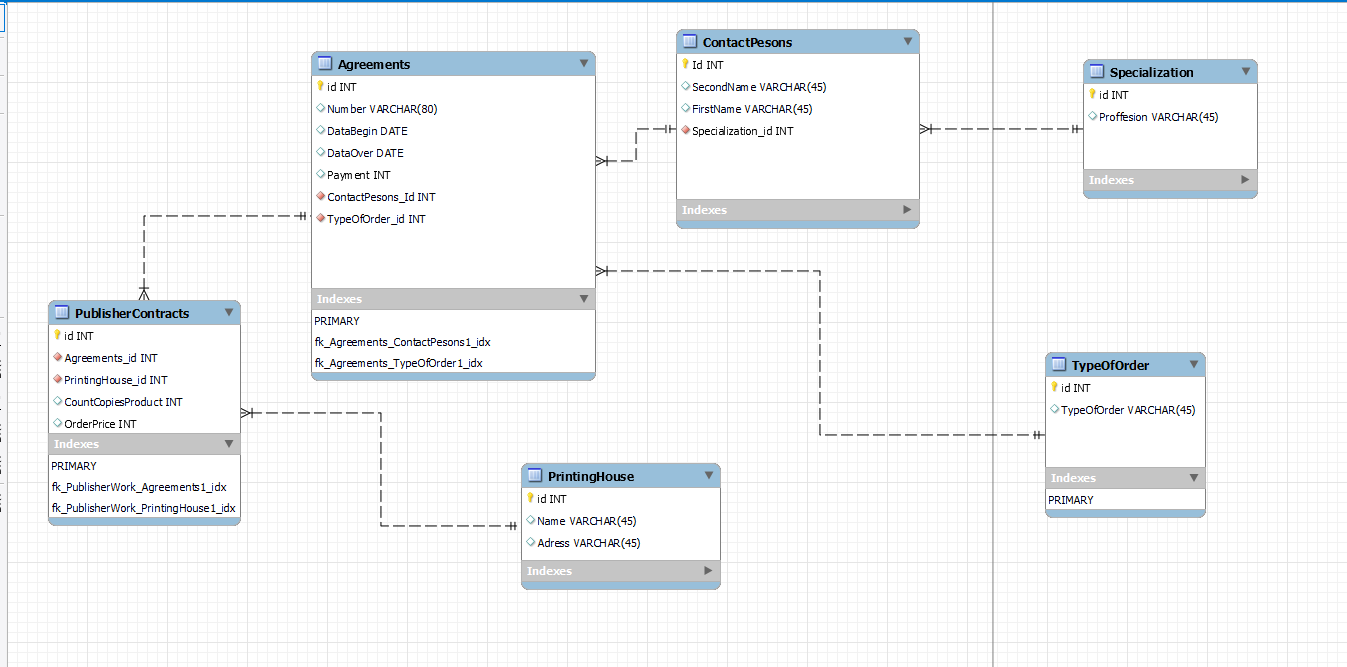
* Создание функционирующей базы данных
* Написание запросов для БД
* Создание процедур, функций, представлений для БД
* Создание индексов для БД
* Работа с исключениями
* Использование триггеров в БД

Работа велась со следующими объектами:

* MySQL – реляционная система управления базами данных.
* MySQL Workbench – инструмент для визуального проектирования баз данных.
* MySQL Command line client Unicode – консольный клиент mysql.

**Приложения:**

**Приложение1(Модель)**



## **Приложение2(Скрипт)**

MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema Publisher

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema Publisher

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `Publisher` ;

USE `Publisher` ;

-- -----------------------------------------------------

-- Table `Publisher`.`Specialization`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Publisher`.`Specialization` (

`id` INT NOT NULL AUTO\_INCREMENT,

`Proffesion` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Publisher`.`ContactPesons`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Publisher`.`ContactPesons` (

`Id` INT NOT NULL AUTO\_INCREMENT,

`SecondName` VARCHAR(45) NULL,

`FirstName` VARCHAR(45) NULL,

`Specialization\_id` INT NOT NULL,

PRIMARY KEY (`Id`),

INDEX `fk\_ContactPesons\_Specialization1\_idx` (`Specialization\_id` ASC) VISIBLE,

CONSTRAINT `fk\_ContactPesons\_Specialization1`

FOREIGN KEY (`Specialization\_id`)

REFERENCES `Publisher`.`Specialization` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Publisher`.`TypeOfOrder`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Publisher`.`TypeOfOrder` (

`id` INT NOT NULL AUTO\_INCREMENT,

`TypeOfOrder` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Publisher`.`Agreements`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Publisher`.`Agreements` (

`id` INT NOT NULL AUTO\_INCREMENT,

`Number` VARCHAR(80) NULL,

`DataBegin` DATE NULL,

`DataOver` DATE NULL,

`Payment` INT NULL,

`ContactPesons\_Id` INT NOT NULL,

`TypeOfOrder\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_Agreements\_ContactPesons1\_idx` (`ContactPesons\_Id` ASC) VISIBLE,

INDEX `fk\_Agreements\_TypeOfOrder1\_idx` (`TypeOfOrder\_id` ASC) VISIBLE,

CONSTRAINT `fk\_Agreements\_ContactPesons1`

FOREIGN KEY (`ContactPesons\_Id`)

REFERENCES `Publisher`.`ContactPesons` (`Id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Agreements\_TypeOfOrder1`

FOREIGN KEY (`TypeOfOrder\_id`)

REFERENCES `Publisher`.`TypeOfOrder` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Publisher`.`PrintingHouse`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Publisher`.`PrintingHouse` (

`id` INT NOT NULL AUTO\_INCREMENT,

`Name` VARCHAR(45) NULL,

`Adress` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Publisher`.`PublisherContracts`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Publisher`.`PublisherContracts` (

`id` INT NOT NULL AUTO\_INCREMENT,

`Agreements\_id` INT NOT NULL,

`PrintingHouse\_id` INT NOT NULL,

`CountCopiesProduct` INT NULL,

`OrderPrice` INT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_PublisherWork\_Agreements1\_idx` (`Agreements\_id` ASC) VISIBLE,

INDEX `fk\_PublisherWork\_PrintingHouse1\_idx` (`PrintingHouse\_id` ASC) VISIBLE,

CONSTRAINT `fk\_PublisherWork\_Agreements1`

FOREIGN KEY (`Agreements\_id`)

REFERENCES `Publisher`.`Agreements` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_PublisherWork\_PrintingHouse1`

FOREIGN KEY (`PrintingHouse\_id`)

REFERENCES `Publisher`.`PrintingHouse` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;