

# MySQL

## Хранимые процедуры

1. Вывести все сведения о поставке (все поля таблицы *Purchases*), а также название книги (поле *Title\_book*) с максимальной общей стоимостью (использовать поля *Cost* и *Amount*).

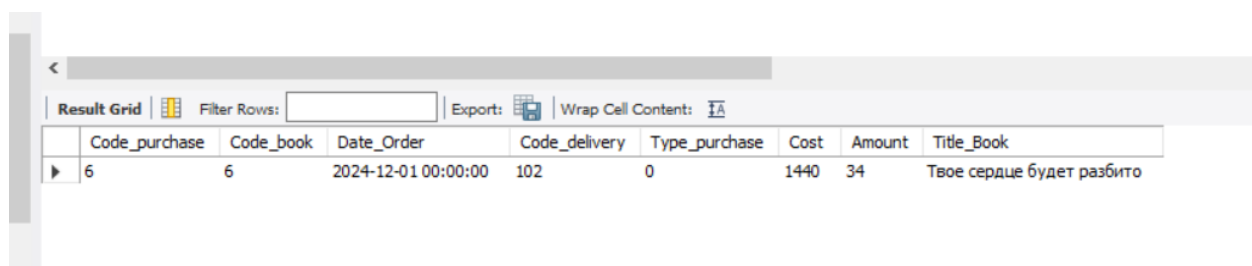
```
CREATE DEFINER='root'@'localhost' PROCEDURE `MaxCostBook_Purchases`()  
BEGIN  
select purchases.*, books.Title_Book  
from purchases  
join books on purchases.Code_book =books.Code_book  
order by(purchases.Cost * purchases.Amount)desc  
limit 1;  
END
```

### Объяснение:

Выбираем все данные из таблицы *purchases* и название книги из таблицы *books*, соединяем таблицы по полю *Code\_book* и сортируем результаты по убыванию общей стоимости покупки.

### Вызов процедуры:

```
call librarynew.MaxCostBook_Purchases();
```



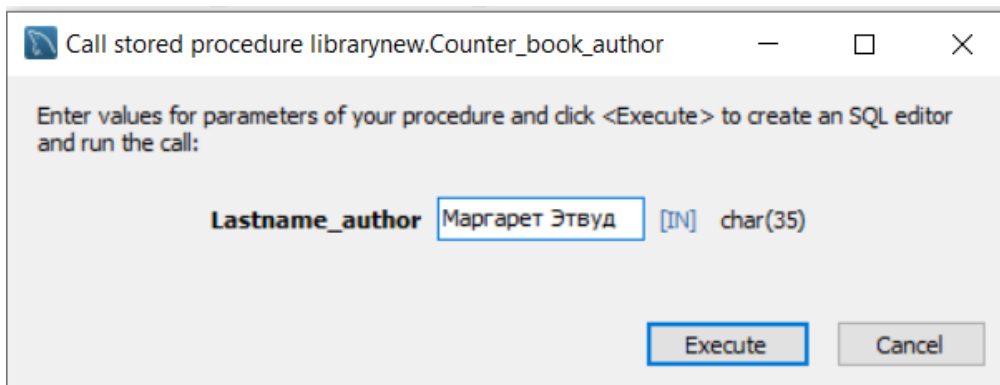
	Code_purchase	Code_book	Date_Order	Code_delivery	Type_purchase	Cost	Amount	Title_Book
▶	6	6	2024-12-01 00:00:00	102	0	1440	34	Твое сердце будет разбито

2. Сосчитать количество книг определенного автора (ФИО автора является входным параметром).

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Counter_book_author`(in Lastname_author  
char(35))  
  
BEGIN  
  
select count(books.Code_book) as Counter_book  
  
from books  
  
join authors on books.Code_author = authors.Code_author  
  
where authors.Name_author = Lastname_author;  
  
END
```

#### Объяснение:

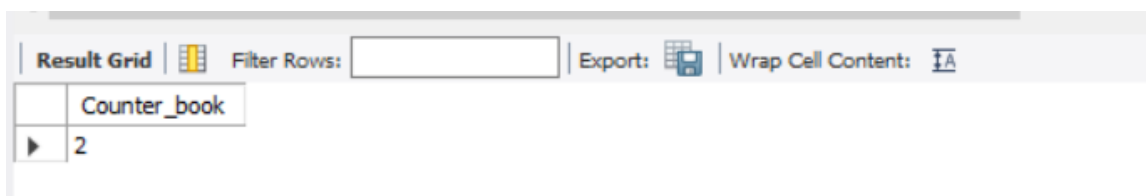
Принимаем параметр Lastname\_author



Считаем количество книг, соединяем таблицы books и authors, фильтровали по фамилии автора и возвратили результат Counter\_book.

#### Вызов процедуры:

```
call librarynew.Counter_book_author('Маргарет Этвуд');
```



Counter_book
2

3. Определить адрес определенного поставщика (Наименование поставщика является входным параметром, адрес поставщика – выходным параметром).

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Address_delivery`(in delivery_name  
char(35), out delivery_address char(35))
```

```
BEGIN
```

```
select Address into delivery_address
```

```
from deliveries
```

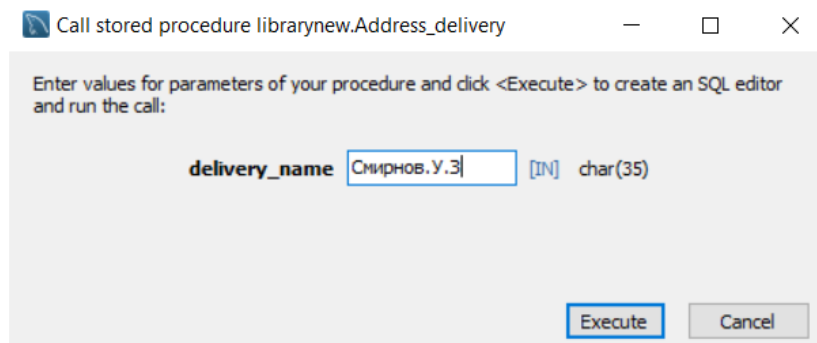
```
where Name_delivery = delivery_name
```

```
limit 1;
```

```
END
```

#### Объяснение:

Принимает входной параметр delivery\_name



Возвращает выходной параметр delivery\_address

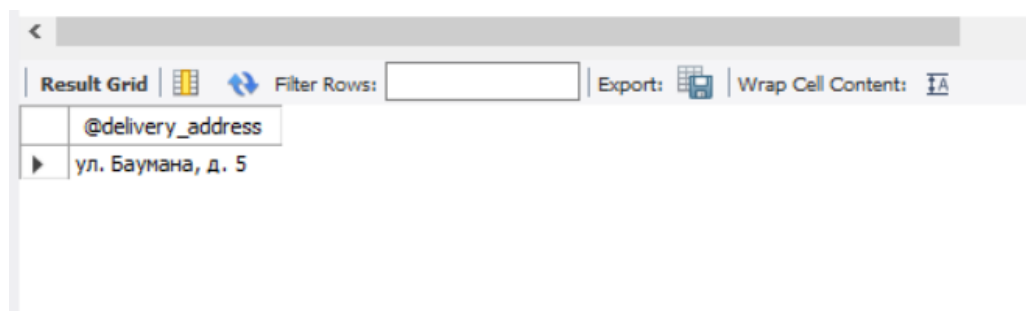
Ищем в таблице deliveries запись, где Name\_delivery совпадает с переданным названием, извлекаем адрес и сохраняем в переменной delivery\_address.

#### Вызов процедуры:

```
set @delivery_address = '0';
```

```
call librarynew.Address_delivery('Смирнов.У.3', @delivery_address);
```

```
select @delivery_address;
```



4. Выполните операцию вставки в таблицу Books. Код книги должен увеличиваться автоматически на единицу.

```
CREATE DEFINER='root'@'localhost' PROCEDURE `Insert_Book`(in p_Title_book char(35), in p_Code_author int, in p_Pages int, in p_Code_publish int)
```

```
BEGIN
```

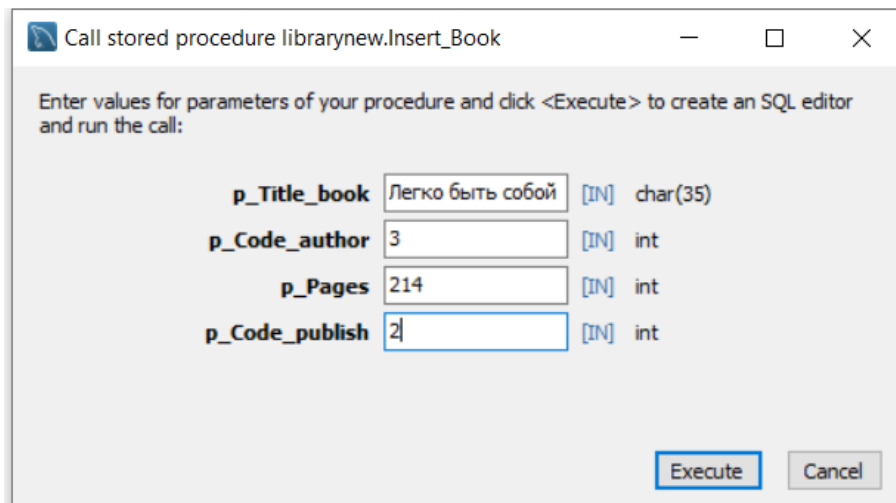
```
insert into books(Title_book, Code_author, Pages, Code_publish)
```

```
values(p_Title_book, p_Code_author, p_Pages, p_Code_publish);
```

```
END
```

Объяснение:

Принимаем 4 входных параметра:



Call stored procedure librarynew.Insert\_Book

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

p_Title_book	Легко быть собой	[IN]	char(35)
p_Code_author	3	[IN]	int
p_Pages	214	[IN]	int
p_Code_publish	2	[IN]	int

Execute Cancel

Вставляем новую запись в таблицу books с переданными значениями

Вызов процедуры:

```
call librarynew.Insert_Book('Легко быть собой', 3, 214, 2);
```

ID	BOOK	AUTHOR	PAGES	PUBLISH	PRICE	DISCOUNT
31	Легко быть собой	3	214	2	0.00	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

5. Определить поставки с минимальной и максимальной стоимостью книг. Отобразить список всех поставок. Если стоимость поставки – максимальная, то вывести сообщение «Максимальная стоимость», если стоимость – минимальная, то вывести сообщение «Минимальная стоимость», иначе – «Средняя стоимость».

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `MinMax_purchase`()
BEGIN
Declare max_cost int;
Declare min_cost int;
select max(Cost), min(Cost) into max_cost, min_cost
from purchases;
select Code_purchase, Code_book, Date_order, Code_delivery, Type_purchase, Cost, Amount,
case
when Cost = max_cost then 'Максимальная стоимость'
when Cost = min_cost then 'Минимальная стоимость'
else 'Средняя стоимость'
end as Cost_Status
from purchases;
END
```

#### Объяснение:

Объявляет две переменные:

- max\_cost
- min\_cost

Находим максимальную и минимальную стоимость, выбираем все поля из таблицы purchases, добавляем столбец Cost\_Status, который помечает каждую запись: "Максимальная стоимость", "Минимальная стоимость", "Средняя стоимость"

#### Вызов процедуры:

```
call librarynew.MinMax_purchase();
```

	Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount	Cost_Status
►	1	3	2006-10-01 00:00:00	103	1	500	28	Средняя стоимость
	2	2	2010-12-10 00:00:00	101	1	800	6	Средняя стоимость
	3	5	2004-04-16 00:00:00	105	0	450	36	Средняя стоимость
	4	1	2023-08-18 00:00:00	104	1	750	20	Средняя стоимость
	5	4	2020-02-03 10:04:00	103	0	860	24	Средняя стоимость
	6	6	2024-12-01 00:00:00	102	0	1440	34	Максимальная стоимость
	7	3	2024-11-09 00:00:00	103	1	1123	30	Средняя стоимость
	8	13	2025-02-26 01:15:08	103	1	100	5	Минимальная стоимость
	9	14	2025-02-26 01:15:08	103	1	200	15	Средняя стоимость

6. Определить количество записей в таблице поставщиков. Пока записей меньше 10, делать в цикле добавление записи в таблицу с автоматическим наращиванием значения ключевого поля, а вместо названия поставщика ставить значение 'не известен'.

```
CREATE DEFINER=`root` @`localhost` PROCEDURE `Counter_delivery`()
BEGIN
declare record_count int;
select count(*) into record_count
from deliveries;
while record_count < 10 do
    insert into deliveries(Name_delivery, Name_company, Address, Phone,INN)
    values('не известен','не известен','не известен', 0, 'не известен');
    set record_count= record_count + 1;
end while;
select 'В таблице 10 записей!' as Itog;
END
```

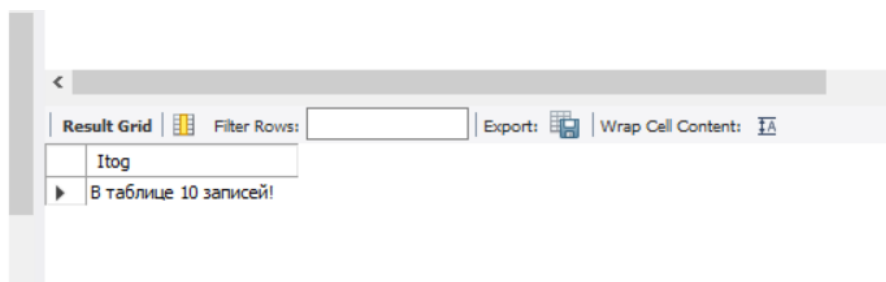
#### Объяснение:

Подсчитываем текущее количество записей в таблице deliveries и сохраняем результат в переменную record\_count, цикл while выполняется, пока количество записей меньше 10: (вставляет новую строку с предустановленными значениями, увеличивает счетчик record\_count на 1 после каждой вставки)

#### Вызов процедуры:

```
call librarynew.Counter_delivery();
```

	Code_delivery	Name_delivery	Name_company	Address	Phone	INN
▶	101	Иванов.И.Г.	ЗАО "Книжные вести"	ул. Ленина, д. 10	89001234567	7701234567
	102	Кузнецов.В.Д	ИП "МирЛитературы"	пр. Невский, д. 20	89491234567	7801234567
	103	Смирнов.У.З	ООО "МирКниг"	ул. Баумана, д. 5	89381234567	1601234567
	104	Петрова.Д.А	ОАО "Книжные мечты"	нет сведений	89161234567	6601234567
	105	Жуков.Ш.Р.	ООО "Торговая Сеть Книг"	ул. Красный проспект, д. 30	89271234567	5401234567
	106	не известен	не известен	не известен	0	не известен
	107	не известен	не известен	не известен	0	не известен
	108	не известен	не известен	не известен	0	не известен
	109	не известен	не известен	не известен	0	не известен
	110	не известен	не известен	не известен	0	не известен
*	NULL	NULL	NULL	NULL	NULL	NULL



# Триггеры

1. *Создайте триггер, запускаемый при занесении новой строки в таблицу авторы. Триггер должен увеличивать счетчик числа добавленных строк.*

```
CREATE DEFINER=`root`@`localhost` TRIGGER `authors_AFTER_INSERT` AFTER INSERT  
ON `authors` FOR EACH ROW BEGIN  
    set @row_count = @row_count + 1;  
END
```

Объяснение:

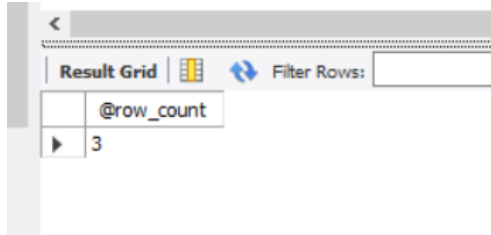
Триггер срабатывает после каждой вставки, увеличивает переменную @row\_count на 1 при добавлении новой записи.

Пример вставки:

```
set @row_count = 0;
```

```
INSERT INTO authors (Name_author, Birthday)  
VALUES ('Amanda Doe', '1980-01-01');
```

```
select @row_count;
```



Результат триггера:

	Code_author	Name_author	Birthday	Count_books
▶	1	Марк Твен	1835-11-20	2
	2	Стивен Кинг	1947-09-21	3
	3	Джейн Остин	1975-12-16	3
	4	Маргарет Этвуд	1939-11-18	2
	5	Анна Джейн	1988-01-14	2
	6	Достоевский, Ф.М.	1821-11-11	0
	7	John Doe	1980-01-01	6
	8	Amanda Doe	1980-01-01	0
	9	Amanda Doe	1980-01-01	0
	10	Amanda Doe	1980-01-01	0

2. *Добавьте в таблицу Авторы поле Количество книг (Count\_books) целого типа со значением по умолчанию 0. Создайте хранимую процедуру, которая подсчитывает количество книг по каждому автору и заносит в поле Count\_books эту информацию. Создайте триггер, запускаемый после внесения новой информации о книге.*

```
CREATE DEFINER=`root`@`localhost` TRIGGER `books_AFTER_INSERT` AFTER INSERT
ON `books` FOR EACH ROW BEGIN
    update authors
    set Count_books = Count_books + 1
    where Code_author = New.Code_author;
END
```

Хранимая процедура:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Author_count_books`()
BEGIN
    update authors a
    set a.Count_books = (
    select count(*)
    from books b
    where b.Code_author = a.Code_author);
END
```

Объяснение:

Этот триггер:

Увеличивает счетчик книг (Count\_books) на 1 у автора

Эта процедура:

Обновление счетчика книг (Count\_books) для всех авторов в таблице authors, для каждого автора подсчитываем количество книг в таблице books, где Code\_author совпадает с кодом автора.

Пример вставки:

```
ALTER TABLE authors
ADD COLUMN Count_books INT DEFAULT 0;
```

```
SET SQL_SAFE_UPDATES = 0;
```

-- Добавляем авторов

```
INSERT INTO authors (Code_author, Name_author, Birthday)
VALUES (18, 'John Doe', '1980-01-01');
```

-- Добавляем книги

```
INSERT INTO books (Title_book, Code_author, Pages, Code_publish)
VALUES ('Book 1', 18, 200, 1),
('Book 2', 18, 150, 1),
('Book 3', 18, 300, 1);
```

```
select * from authors;
```

Вызов хранимой процедуры:



CALL Author\_count\_books();

Результат триггера:

	Code_author	Name_author	Birthday	Count_books
	4	Маргарет Этвуд	1939-11-18	2
	5	Анна Джейн	1988-01-14	2
	6	Достоевский,Ф.М.	1821-11-11	0
	7	John Doe	1980-01-01	9
	8	Amanda Doe	1980-01-01	0
	9	Amanda Doe	1980-01-01	0
	10	Amanda Doe	1980-01-01	0
	11	Amanda Doe	1980-01-01	0
	12	Amanda Doe	1980-01-01	0
	13	Amanda Doe	1980-01-01	0
	14	Amanda Doe	1980-01-01	0
	15	Amanda Doe	1980-01-01	0
	16	Amanda Doe	1980-01-01	0
	17	Amanda Doe	1980-01-01	0
	18	John Doe	1980-01-01	3

3. *Создайте триггер, запускаемый при внесении информации о новых поставках. Выполните проверку о количестве добавляемой книги в таблице Книги. Если количество экземпляров книг в таблице меньше 10, то необходимо увеличить стоимость книг на 20 %.*

```
CREATE DEFINER='root'@'localhost' TRIGGER `purchases_AFTER_INSERT` AFTER
INSERT ON `purchases` FOR EACH ROW BEGIN
    declare book_count int;
    select amount into book_count
    from books
    where Code_book = New.Code_book;

    if book_count < 10 then
        update books
        set Cost = Cost * 1.20
        where Code_book = New.Code_book;
    end if;
END
```

Объяснение:

Объявляем переменную book\_count, запрашиваем текущее количество (amount) книги из таблицы books, если количество этой книги на складе меньше 10 (book\_count < 10), то: увеличиваем цену (Cost) этой книги на 20%.

Пример вставки:

```
ALTER TABLE books
```

```
ADD COLUMN Cost DECIMAL(10, 2) DEFAULT 0;
```

## ALTER TABLE books

```
ADD COLUMN Amount INT DEFAULT 0;
```

## -- Добавляем книги

```
INSERT INTO books (Code_book, Title_book, Code_author, Pages, Code_publish, Cost, Amount)
```

VALUES (13, 'Book 13', 1, 200, 1, 100.00, 5), -- Количество экземпляров книги 13 меньше 10

(14, 'Book 14', 2, 150, 1, 200.00, 15); -- Количество экземпляров книги 14 больше 10

## -- Добавляем поставки

```
INSERT INTO purchases (Code_purchase, Code_book, Date_Order, Code_delivery,
Type_purchase, Cost, Amount)
```

VALUES (8, 13, NOW(), 103, 1, 100.00, 5), -- Количество экземпляров книги 13 меньше 10  
(9, 14, NOW(), 103, 1, 200.00, 15); -- Количество экземпляров книги 14 больше 10

```
SELECT * FROM books;
```

Результат триггера:

Таблица книг:

	Code_book	Title_book	Code_author	Pages	Code_publish	Cost	Amount
	2	Приключения Тома Сойера	1	300	1	0.00	0
	3	Львица	4	400	3	0.00	0
	4	Сияние	2	500	5	0.00	0
	5	Служанка	4	350	3	0.00	0
	6	Твое сердце будет разбито	5	300	2	0.00	0
	7	Наука. Техника. Иновации	3	450	4	0.00	0
	8	Алиса в стране чудес	2	400	5	0.00	0
	9	Русалочка	5	345	5	0.00	0
	10	Book 1	7	200	1	0.00	0
	11	Book 2	7	150	1	0.00	0
	12	Book 3	7	300	1	0.00	0
	13	Book 13	1	200	1	120.00	5
	14	Book 14	2	150	1	200.00	15

### Таблица Поставщиков:

[illegible]

4. *Запретить вставлять новые строки в таблицу Поставщики, выводя при этом сообщение «Вставка строк запрещена».*

```
CREATE DEFINER=`root`@`localhost` TRIGGER `deliveries_AFTER_INSERT` AFTER  
INSERT ON `deliveries` FOR EACH ROW BEGIN
```

```
    signal sqlstate '45000'
```

```
    set message_text = 'Вставка строк запрещена';
```

```
END
```

Объяснение:

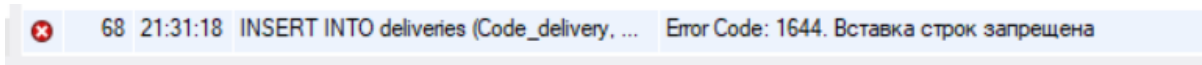
После попытки вставить строку срабатывает сигнал, о том, что ставка строк запрещена

Пример вставки:

```
INSERT INTO deliveries (Code_delivery, Name_delivery, Name_company, Address, Phone,  
INN)
```

```
VALUES (111, 'Supplier A', 'Company A', '123 Street', 12378901, '12389012345');
```

Результат триггера:



# Транзакции

1. Проверьте выполнение команд транзакции при добавлении новой информации об издательствах.

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

```
start transaction;
```

```
insert into publishing_house(Publish, City)
```

```
values('Издательство 1', 'Уфа');
```

```
savepoint after_first_insert;
```

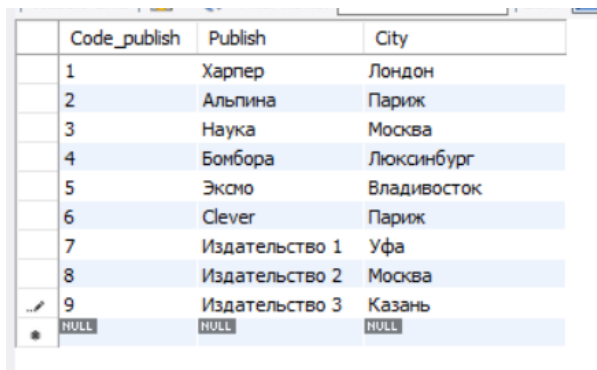
```
insert into publishing_house(Publish, City)
```

```
values('Издательство 2', 'Москва');
```

```
SELECT LAST_INSERT_ID();
```

```
insert into publishing_house(Publish, City)
```

```
values('Издательство 3', 'Казань');
```



	Code_publish	Publish	City
	1	Харпер	Лондон
	2	Альпина	Париж
	3	Наука	Москва
	4	Бомбора	Люксембург
	5	Эксмо	Владивосток
	6	Clever	Париж
	7	Издательство 1	Уфа
	8	Издательство 2	Москва
	9	Издательство 3	Казань
*	NULL	NULL	NULL

```
rollback to savepoint after_first_insert;
```



	Code_publish	Publish	City
	1	Харпер	Лондон
	2	Альпина	Париж
	3	Наука	Москва
	4	Бомбора	Люксембург
	5	Эксмо	Владивосток
	6	Clever	Париж
	7	Издательство 1	Уфа
*	NULL	NULL	NULL

```
select * from publishing_house
```

```
where Publish in ('Издательство 1', 'Издательство 2', 'Издательство 3');
```

	Code_publish	Publish	City
7		Издательство 1	Уфа
	NULL	NULL	NULL

commit;

SELECT \* FROM publishing\_house;

	Code_publish	Publish	City
1		Харпер	Лондон
2		Альпина	Париж
3		Наука	Москва
4		Бомбора	Люксембург
5		Эксмо	Владивосток
6		Clever	Париж
7		Издательство 1	Уфа
	NULL	NULL	NULL

Объяснение:

Установили уровень изоляции "чтение подтвержденных данных" и начали транзакцию. Добавили 3 издательства, после Издательства 1 создали точку сохранения after\_first\_insert, затем вставили еще 2 издательства, после чего откатили изменения до точки сохранения after\_first\_insert и в базе данных осталась лишь одна запись «Издательство 1», после сохранили изменения.

# Создание пользователей

## 1. Администратор – обладает всеми правами

```
CREATE USER 'library_admin'@'localhost' IDENTIFIED BY 'admin_password';  
GRANT ALL PRIVILEGES ON librarynew.* TO 'library_admin'@'localhost';
```

```
FLUSH PRIVILEGES;
```

```
SHOW GRANTS FOR 'library_admin'@'localhost';
```

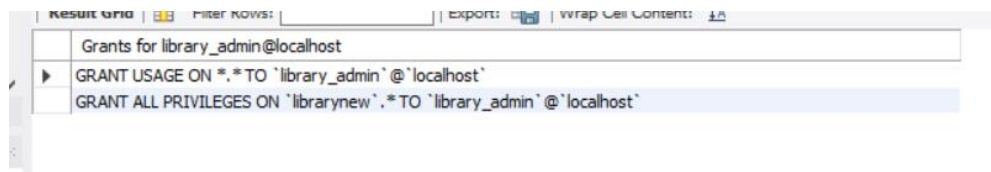
-- Удаление пользователя

```
DROP USER 'library_admin'@'localhost';
```

### Объяснение:

Создаем нового пользователя с именем library\_admin и устанавливаем пароль 'admin\_password' далее даем пользователю все права на все таблицы базы данных librarynew, и при изменении прав, после показываем все права пользователя.

### Результат:



## 2. Диспетчер – просматривает, заполняет и изменяет справочники: книги, авторы, издательства, поставщики.

```
CREATE USER 'dispatcher'@'localhost' IDENTIFIED BY 'dispatcher_password';
```

```
GRANT SELECT, INSERT, UPDATE ON librarynew.books TO 'dispatcher'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON librarynew.authors TO 'dispatcher'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON librarynew.publishing_house TO  
'dispatcher'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE ON librarynew.deliveries TO 'dispatcher'@'localhost';
```

```
FLUSH PRIVILEGES;
```

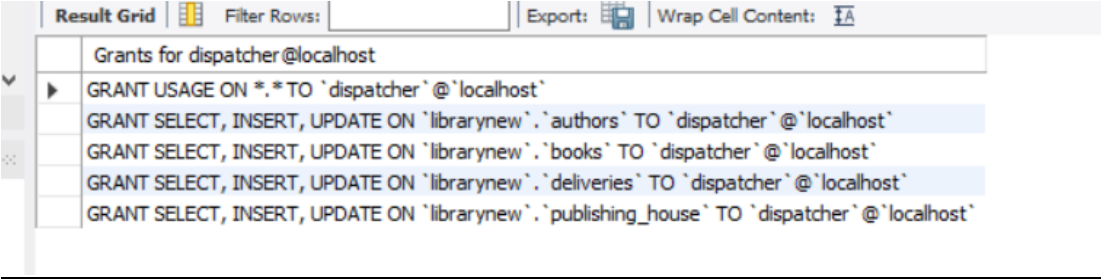
```
SHOW GRANTS FOR 'dispatcher'@'localhost';
```

-- Удаление пользователя  
DROP USER 'dispatcher'@'localhost';

Объяснение:

Создаем нового пользователя с именем 'dispatcher' и устанавливаем пароль 'dispatcher\_password', даём права только на 4 таблицы (books, authors, publishing\_house, deliveries), разрешаем только выборку (SELECT), добавление (INSERT) и обновление (UPDATE), применяем изменения прав, после показываем все назначенные права пользователя.

Результат:



Grants for dispatcher@localhost	
▶	GRANT USAGE ON *.* TO 'dispatcher'@'localhost'
	GRANT SELECT, INSERT, UPDATE ON 'librarynew'. 'authors' TO 'dispatcher'@'localhost'
	GRANT SELECT, INSERT, UPDATE ON 'librarynew'. 'books' TO 'dispatcher'@'localhost'
	GRANT SELECT, INSERT, UPDATE ON 'librarynew'. 'deliveries' TO 'dispatcher'@'localhost'
	GRANT SELECT, INSERT, UPDATE ON 'librarynew'. 'publishing_house' TO 'dispatcher'@'localhost'

*3. Менеджер по работе с поставщиками – просматривает и добавляет новую информацию в справочники, оформляет поставки.*

```
CREATE USER 'supply_manager'@'localhost' IDENTIFIED BY 'supply_manager_password';  
  
GRANT SELECT, INSERT ON librarynew.deliveries TO 'supply_manager'@'localhost';  
GRANT SELECT, INSERT ON librarynew.publishing_house TO 'supply_manager'@'localhost';  
GRANT SELECT, INSERT ON librarynew.purchases TO 'supply_manager'@'localhost';
```

```
FLUSH PRIVILEGES;
```

```
SHOW GRANTS FOR 'supply_manager'@'localhost';
```

-- Удаление пользователя  
DROP USER 'supply\_manager'@'localhost';

Объяснение:

Создаем нового пользователя с именем 'supply\_manager' и устанавливаем пароль 'supply\_manager\_password', даём права только на 3 таблицы (deliveries, publishing\_house, purchases), разрешаем только выборку (SELECT) и добавление (INSERT), применяем изменения прав, после показываем все назначенные права пользователя.

Результат:

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
Grants for supply_manager@localhost				
▶	GRANT USAGE ON *,* TO `supply_manager`@`localhost`			
⋮	GRANT SELECT, INSERT ON `librarynew`.`deliveries` TO `supply_manager`@`localhost`			
	GRANT SELECT, INSERT ON `librarynew`.`publishing_house` TO `supply_manager`@`localhost`			
	GRANT SELECT, INSERT ON `librarynew`.`purchases` TO `supply_manager`@`localhost`			

#### 4. Поставщики – просматривают только свои поставки

```
CREATE USER 'supplier1'@'localhost' IDENTIFIED BY 'supplier1_password';
CREATE USER 'supplier2'@'localhost' IDENTIFIED BY 'supplier2_password';

CREATE VIEW supplier1_deliveries AS
SELECT * FROM deliveries WHERE Name_company = 'ИП "МирЛитературы"';

CREATE VIEW supplier2_deliveries AS
SELECT * FROM deliveries WHERE Name_company = 'ООО "МирКниг"';

GRANT SELECT ON librarynew.deliveries TO 'supplier1'@'localhost';

GRANT SELECT ON librarynew.deliveries TO 'supplier2'@'localhost';
FLUSH PRIVILEGES;

SHOW GRANTS FOR 'supplier1'@'localhost';
SHOW GRANTS FOR 'supplier2'@'localhost';

SELECT * FROM supplier1_deliveries;
SELECT * FROM supplier2_deliveries;
```

-- Удаление

```
DROP USER 'supplier1'@'localhost';
DROP USER 'supplier2'@'localhost';
```

Объяснение:





Создаются два пользователя: supplier1 и supplier2, каждый со своим паролем. Для каждого поставщика создается отдельное представление (VIEW). Каждое представление показывает только доставки соответствующей компании. Оба пользователя получают права только на просмотр (SELECT) таблицы deliveries, применяем изменения прав, после показываем все назначенные права пользователя и просматриваем данные через созданные представления.

Результат:



	Code_delivery	Name_delivery	Name_company	Address	Phone	INN
▶	102	Кузнецов.В.Д	ИП "МирЛитературы"	пр. Невский, д. 20	89491234567	7801234567

Supplier2\_deliveries:

Result Grid			Filter Rows: <input type="text"/>	Export: 	Wrap Cell Content: 	
	Code_delivery	Name_delivery	Name_company	Address	Phone	INN
▶	103	Смирнов.У.З	ООО "МирКниг"	ул. Баумана, д. 5	89381234567	1601234567

# PostgreSQL

## Хранимые процедуры

1. *Вывести фамилии и имена студентов (поля Surname, Name из таблицы Students) с максимальным средним баллом за весь период обучения (условие по полю Estimate из таблицы Progress).*

```
CREATE OR REPLACE FUNCTION get_students_with_max_avg()
  RETURNS table(
    "Surname" character,
    "Name" character
  )
  LANGUAGE 'plpgsql'
  VOLATILE
  PARALLEL UNSAFE
  COST 100

AS $BODY$
begin
  return query
  SELECT "Students"."Surname", "Students"."Name"
  FROM "Students"
  JOIN (
    SELECT "Code_stud", AVG("Estimate") AS avg_estimate
    FROM "Progress"
    GROUP BY "Code_stud"
  ) AS student_avg ON "Students"."Code_stud" = student_avg."Code_stud"
  WHERE student_avg.avg_estimate = (
    SELECT MAX(avg_estimate)
    FROM (
      SELECT AVG("Estimate") AS avg_estimate
      FROM "Progress"
      GROUP BY "Code_stud"
    ) AS all_avg
  );
end;
$BODY$;
```

### Объяснение:

#### **Возвращаемые данные:**

Таблица с двумя колонками: "Surname" и "Name"

Соединяем таблицы "Students" и "Progress". Для каждого студента вычисляем средний балл (AVG) из таблицы "Progress". Находим максимальное значение среднего балла среди всех студентов. Возвращаем студентов, чей средний балл равен этому максимальному значению.

### Вывод результата:

```
select * from get_students_with_max_avg();
```

	Surname character	Name character
1	Федоров ...	Кирилл ...
2	Андреев ...	Тимур ...
3	Иванов ...	Иван ...

2. Определить средний балл определенного студента (ФИО студента является входным параметром).

```
CREATE OR REPLACE FUNCTION public.students_avg(IN student_surname character,IN student_name
character,IN student_lastname character)
RETURNS numeric
LANGUAGE 'plpgsql'
VOLATILE
PARALLEL UNSAFE
COST 100
```

```
AS $BODY$
declare avg_estimate numeric(3,2);
begin
    SELECT AVG("Progress"."Estimate") into avg_estimate
    FROM "Students"
    JOIN "Progress" ON "Students"."Code_stud" = "Progress"."Code_stud"
    WHERE "Students"."Surname" = "student_surname"
    AND "Students"."Name" = "student_name"
    AND "Students"."Lastname" = "student_lastname";
    return avg_estimate;
end;
$BODY$;
```

Объяснение:

Принимаем 3 входных параметра: student\_surname, student\_name, student\_lastname.

Объявляем переменную avg\_estimate для хранения результата, соединяем таблицы "Students" и "Progress", находим студента по полному ФИО и вычисляем средний балл его оценок (AVG). Возвращаем полученное значение среднего балла.

Вывод результата:

```
select * from students_avg(Сидорова, Елизавета, Сергеевна);
```

	students_avg numeric
1	4.00

3. *Определить специальность и номер курса определенного студента (ФИО студента является входным параметром, Название специальности и Номер курса – выходными параметрами).*

```
CREATE OR REPLACE FUNCTION public.students_resorces(IN student_surname character,IN student_name character,IN student_lastname character)
```

```
RETURNS table(  
    "Num_course" integer,  
    "Name_specialtiy" character  
)
```

```
LANGUAGE 'plpgsql'
```

```
VOLATILE
```

```
PARALLEL UNSAFE
```

```
COST 100
```

```
AS $BODY$
```

```
begin
```

```
    return query
```

```
    select "Groups"."Num_course", "Groups"."Name_speciality"
```

```
    from "Students"
```

```
    join "Groups" on "Students"."Code_group" = "Groups"."Code_group"
```

```
    WHERE "Students"."Surname" = student_surname
```

```
    AND "Students"."Name" = student_name
```

```
    AND "Students"."Lastname" = student_lastname;
```

```
end;
```

```
$BODY$;
```

Объяснение:

Принимаем 3 входных параметра: student\_surname, student\_name, student\_lastname.

Возвращает таблицу с двумя колонками: Num\_course, Name\_specialtiy.

Соединяем таблицы "Students" и "Groups" по коду группы, находим студента по полному ФИО и возвращаем его курс и специальность.

Вывод результата:

```
select * from students_resorces('Федоров','Кирилл','Романович')
```

	Num_course integer	Name_specialtiy character
1	1	Химическая технология

4. Выполните операцию вставки в таблицу *Students*. Код студента должен автоматически увеличиваться на единицу.

```
CREATE OR REPLACE PROCEDURE public.students_insert(IN stud_surname character, IN stud_name
character, IN stud_lastname character, IN stud_code_group integer, IN stud_birthday date, IN stud_phone
numeric)
```

```
LANGUAGE 'plpgsql'
```

```
AS $BODY$
```

```
BEGIN
```

```
INSERT INTO "Students" (
```

```
"Code_stud", "Surname", "Name", "Lastname",
```

```
"Code_group", "Birthday", "Phone"
```

```
)
```

```
VALUES (
```

```
(SELECT (COALESCE(MAX(CAST(regex_replace("Code_stud", '[^0-9]', 'g') AS integer)), 0) +
1)::varchar FROM "Students"),
```

```
stud_surname, stud_name, stud_lastname, stud_code_group, stud_birthday, stud_phone);
```

```
END;
```

```
$BODY$;
```

Объяснение:

Принимаем 6 входных параметров: student\_surname, student\_name, student\_lastname, stud\_code\_group, stud\_birthday, stud\_phone.

Извлекаем максимальный числовой код из существующих записей, увеличиваем значение на 1 для нового студента.

Вывод результата:

```
call students_insert ('Нафикова','Арина','Рафаэлевна', 101, '2007-07-21', 9170485817);
```

```
call students_insert ('Рябина','Карина','Алексеевна', 103, '2005-02-25', 9170485849);
```

	Code_stud [PK] character (10)	Surname character (25)	Name character (25)	Lastname character (25)	Code_group integer	Birthday date	Phone numeric	Avg_Estimate real
1	67892	Файрушина	Евгения	Рафаэлевна	101	2007-02-26	9170485817	[default]
2	67893	Рябина	Карина	Алексеевна	103	2005-02-25	9170485849	0
3	67894	Нафикова	Арина	Рафаэлевна	101	2007-07-21	9170485817	0
4	A12345	Сидорова	Елизавета	Сергеевна	101	2006-03-17	79171234567	4
5	B67890	Соколов	Артем	Артемович	103	2006-12-23	2345678902	4
6	C12345	Федоров	Кирилл	Романович	104	2001-03-25	5678901235	5

5. *Определить средний возраст всех студентов. Вывести список всех студентов. Если возраст студента больше среднего возраста, то вывести сообщение «Вы старше среднего возраста всех студентов», если возраст – меньше, то вывести сообщение «Ваш возраст меньше среднего возраста всех студентов», а иначе – «Ваш возраст равен среднему возрасту всех студентов».*

```
CREATE OR REPLACE FUNCTION public.avg_old_students()
```

```
RETURNS TABLE (
```

```
    "Surname" CHARACTER,
```

```
    "Name" CHARACTER,
```

```
    "Lastname" CHARACTER,
```

```
    "Student_age" INT,
```

```
    message_age TEXT
```

```
)
```

```
LANGUAGE plpgsql
```

```
AS $BODY$
```

```
DECLARE
```

```
    avg_old numeric;
```

```
BEGIN
```

```
    SELECT avg(extract(year FROM age("Birthday"))) INTO avg_old
```

```
    FROM "Students";
```

```
    RETURN QUERY
```

```
    SELECT
```

```
        "Students"."Surname",
```

```
        "Students"."Name",
```

```
        "Students"."Lastname",
```

```
        extract(year FROM age("Birthday"))::int AS "Student_age",
```

```
        CASE
```

```
            WHEN extract(year FROM age("Birthday")) > avg_old THEN 'Вы старше среднего возраста всех студентов'
```

```
            WHEN extract(year FROM age("Birthday")) < avg_old THEN 'Ваш возраст меньше среднего возраста всех студентов'
```

```
            ELSE 'Ваш возраст равен среднему возрасту всех студентов'
```

```
        END AS message_age
```

```
    FROM "Students";
```

```
END;
```

```
$BODY$;
```

### Объяснение:

Возвращаем данные: "Surname", "Name", "Lastname", "Student\_age", "message\_age".

Сначала вычислили средний возраст всех студентов и преобразовали даты рождения в возраст. Для каждого студента сравнивает его возраст с вычисленным средним значением и сформировали сообщения: 'Вы старше среднего возраста всех студентов', 'Ваш возраст меньше среднего возраста всех студентов', 'Ваш возраст равен среднему возрасту всех студентов'.

### Вывод результата:

```
select * from avg_old_students();
```

	Surname character	Name character	Lastname character	Student_age integer	message_age text
1	Васильев...	Валентин...	Сергеевн...	18	Ваш возраст меньше среднего возраста всех студентов
2	Иванов ...	Иван ...	Иванович...	25	Вы старше среднего возраста всех студентов
3	Никитина...	Елизавет...	Игоревна...	19	Ваш возраст меньше среднего возраста всех студентов
4	Андреев ...	Тимур ...	нет сведе...	23	Вы старше среднего возраста всех студентов
5	Васильев...	Екатерин...	Владими...	19	Ваш возраст меньше среднего возраста всех студентов
6	Макаров ...	Наталья ...	Алексеев...	25	Вы старше среднего возраста всех студентов
7	Рябина ...	Карина ...	Алексеев...	20	Ваш возраст меньше среднего возраста всех студентов
8	Нафиков...	Арина ...	Рафаэлев...	17	Ваш возраст меньше среднего возраста всех студентов
9	Файруши...	Евгения ...	Рафаэлев...	18	Ваш возраст меньше среднего возраста всех студентов
10	Иванов ...	Павел ...	Сергееви...	17	Ваш возраст меньше среднего возраста всех студентов
11	Смирнов ...	Лев ...	Николаев...	17	Ваш возраст меньше среднего возраста всех студентов
12	Орлова ...	Дарья ...	Матвеевн...	25	Вы старше среднего возраста всех студентов
13	Соколов ...	Артем ...	Артемови...	18	Ваш возраст меньше среднего возраста всех студентов
14	Федоров ...	Кирилл ...	Романов...	24	Вы старше среднего возраста всех студентов
15	Петров ...	Роман ...	Дмитрие...	21	Вы старше среднего возраста всех студентов
16	Сидорова...	Елизавет...	Сергеевн...	19	Ваш возраст меньше среднего возраста всех студентов

6. *Определить количество записей в таблице дисциплин. Пока записей меньше 10, делать в цикле добавление записи в таблицу с автоматическим наращиванием значения ключевого поля, а вместо названия дисциплины ставить значение 'не известно'.*

```
CREATE OR REPLACE PROCEDURE public.add_subjects()
LANGUAGE 'plpgsql'
AS $BODY$
declare
    subject_count INT;
    next_code_subject INT;
begin
    select count(*) into subject_count from "Subjects";
    while subject_count < 10 loop
        select coalesce(max("Code_subject"), 0) + 1 into next_code_subject
            from "Subjects";
        insert into "Subjects" ("Code_subject", "Name_subject", "Count_hours")
        values (next_code_subject, 'не известно', 0);
        subject_count := subject_count + 1;
    end loop;
    raise notice 'В таблице 10 записей!';
end;
$BODY$;
ALTER PROCEDURE public.add_subjects()
    OWNER TO postgres;
```

#### Объяснение:

Сначала подсчитываем количество существующих записей в таблице "Subjects".  
Прописываем условие, что пока количество записей меньше 10: генерируем следующий код предмета +1 и добавляем новую запись с параметрами: Code\_subject, Name\_subject, Count\_hours, увеличивая счетчик записей.

#### Вывод результата:

call add\_subjects();

Data Output   Сообщения   Notifications

ЗАМЕЧАНИЕ: В таблице 10 записей!  
CALL

Запрос завершён успешно, время выполнения: 45 msec.



	Code_subject [PK] integer	Name_subject character (25)	Count_hours integer
1	1	Программиро...	120
2	2	Математика ...	250
3	3	Физика ...	87
4	4	Химия ...	98
5	5	математическ...	64
6	6	Экономика ...	92
7	7	не известно ...	0
8	8	не известно ...	0
9	9	не известно ...	0
10	10	не известно ...	0

# Триггеры

1. *Создайте триггер, запускаемый при занесении новой строки в таблицу, преподаватели. Триггер должен увеличивать счетчик числа добавленных строк.*

```
CREATE OR REPLACE FUNCTION public.new_lectors()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    VOLATILE
    COST 100
AS $BODY$
DECLARE
    next_code_lector INT;
BEGIN
    SELECT COALESCE(MAX("Code_lector"), 0) + 1 INTO next_code_lector
    FROM "Lectors";
    NEW."Code_lector" := next_code_lector;
    RETURN NEW;
END;
$BODY$;
CREATE TRIGGER trg_before_insert_lectors
BEFORE INSERT ON "Lectors"
FOR EACH ROW
EXECUTE FUNCTION public.new_lectors();
```

## Вставка данных:

```
INSERT INTO "Lectors" ("Name_lector", "Science", "Post", "Date_")
VALUES ('Новикова Арина', 'Математика', 'Доцент', '2023-10-01');
UPDATE lectors_counter SET counter = counter + 1;
SELECT * FROM lectors_counter;
```

## Объяснение:

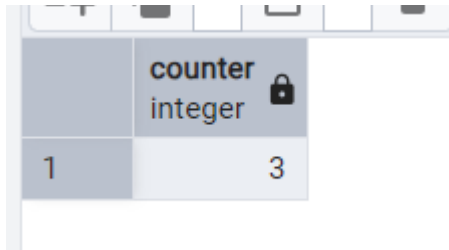
### Функция:

Находим максимальное значение Code\_lector в таблице\_используем (COALESCE для обработки случая с пустой таблицей (начинает с 1))\_увеличиваем значение на 1 для нового лектора и присваиваем сгенерированный код полю NEW.Code\_lector

### Триггер:

Срабатывает перед каждой вставкой в таблицу "Lectors". Для каждой новой строки вызываем функцию new\_lectors() и применяем возвращенное значение.

#### Вывод результата:



	counter integer	
1		3

*2. Добавьте в таблицу Студенты поле Средний балл (Avg\_Estimate) вещественного типа со значением по умолчанию 0. Создайте хранимую процедуру, которая подсчитывает средний балл для каждого студента и заносит в поле Avg\_Estimate эту информацию. Создайте триггер, запускаемый после внесения новой информации об оценках студента и автоматически обновляет информацию о среднем балле студента.*

```
ALTER TABLE "Students"
```

```
ADD COLUMN "Avg_Estimate" REAL DEFAULT 0;
```

```
-- Процедура
```

```
CREATE OR REPLACE FUNCTION public.students_avg_estimate()
```

```
RETURNS void
```

```
LANGUAGE 'plpgsql'
```

```
VOLATILE
```

```
PARALLEL UNSAFE
```

```
COST 100
```

```
AS $BODY$
```

```
BEGIN
```

```
UPDATE "Students" AS s
```

```
SET "Avg_Estimate" = (
```

```
SELECT AVG(p."Estimate")
```

```
FROM "Progress" AS p
```

```
WHERE p."Code_stud" = s."Code_stud"
```

```
);
```

```
END;
```

```
$BODY$;
```

```

select * from students_avg_estimate();

-- триггерная функция
CREATE OR REPLACE FUNCTION public.update_avg_estimate()

    RETURNS trigger

    LANGUAGE 'plpgsql'

    VOLATILE

    COST 100

AS $BODY$
begin
    PERFORM public.students_avg_estimate();
    RETURN NULL;
end;
$BODY$;

-- Триггер
CREATE TRIGGER after_estimate
after INSERT OR UPDATE OR DELETE ON "Progress"
FOR EACH ROW
EXECUTE FUNCTION public.update_avg_estimate();

SELECT "Code_stud", "Avg_Estimate"
FROM "Students"
WHERE "Code_stud" = 'A12345';

```

	Code_stud [PK] character (10) 	Avg_Estimate real 
1	A12345	4

```

select * from "Students";

```

	Code_stud [PK] character (10)	Surname character (25)	Name character (25)	Lastname character (25)	Code_group integer	Birthday date	Phone numeric	Avg_Estimate real
1	S002	Васильева ...	Валентина ...	Сергеевна ...	101	2006-09-10	89170486729	4
2	S001	Иванов ...	Иван ...	Иванович ...	101	2000-01-01	1234567890	5
3	D67890	Никитина ...	Елизавета ...	Игоревна ...	102	2005-07-30	8901234568	3
4	E12345	Андреев ...	Тимур ...	нет сведений ...	101	2002-04-18	9012345678	5
5	H67890	Васильева ...	Екатерина ...	Владимировн...	109	2005-07-05	6789012346	4
6	I12345	Макаров ...	Наталья ...	Алексеевна ...	105	2000-05-15	9012345679	3
7	67893	Рябина ...	Карина ...	Алексеевна ...	103	2005-02-25	9170485849	0
8	67894	Нафикова ...	Арина ...	Рафаэлевна ...	101	2007-07-21	9170485817	0
9	67892	Файрушина ...	Евгения ...	Рафаэлевна ...	101	2007-02-26	9170485817	[default]
10	R67891	Иванов ...	Павел ...	Сергеевич ...	[null]	2007-06-09	[null]	[default]
11	K12345	Смирнов ...	Лев ...	Николаевич ...	105	2007-06-15	2345678962	2
12	O12345	Орлова ...	Дарья ...	Матвеевна ...	102	1999-08-22	8901234567	[default]
13	B67890	Соколов ...	Артем ...	Артемович ...	103	2006-12-23	2345678902	4
14	C12345	Федоров ...	Кирилл ...	Романович ...	104	2001-03-25	5678901235	5
15	L67890	Петров ...	Роман ...	Дмитриевич ...	108	2004-04-18	2345788902	[default]
16	A12345	Сидорова ...	Елизавета ...	Сергеевна ...	101	2006-03-17	79171234567	4

UPDATE "Progress"

SET "Estimate" = 5

WHERE "Code\_progress" = 1;

16	A12345	Сидорова ...	Елизавета ...	Сергеевна ...	101	2006-03-17	79171234567	5
----	--------	--------------	---------------	---------------	-----	------------	-------------	---

### Объяснение:

**Процедура обновления** students\_avg\_estimate(): Вычисляем средний балл для каждого студента из таблицы "Progress" и обновляем поле "Avg\_Estimate" в таблице "Students".

**Триггерная функция** update\_avg\_estimate(): Вызываем процедуру students\_avg\_estimate() при изменениях в "Progress".

**Триггер** after\_estimate: Вызываем триггерную функцию update\_avg\_estimate()

*3. Создайте триггер, запускаемый при внесении информации о новых оценках. Выполните проверку наличия информации о добавляемом студенте в таблице Студенты. Если данная информация в таблице отсутствует, то необходимо запустить хранимую процедуру на вставку записи в таблицу Студенты (параметры можно задать произвольно).*

-- Функция

```
CREATE OR REPLACE FUNCTION public.insert_students(IN p_code_stud character,IN p_code_group
integer,IN p_surname character,IN p_name character,IN p_lastname character,IN p_birthday date,IN p_phone
numeric)
```

```
    RETURNS void
```

```
    LANGUAGE 'plpgsql'
```

```
    VOLATILE
```

```
    PARALLEL UNSAFE
```

```
    COST 100
```

```
AS $BODY$
```

```
begin
```

```
    if not exists(select 1
```

```
        from "Students"
```

```
        where "Code_stud" = p_code_stud)then
```

```
        insert into "Students" ("Code_stud", "Code_group", "Surname", "Name", "Lastname", "Birthday",
"Phone")
```

```
        values(p_code_stud, p_code_group, p_surname, p_name, p_lastname, p_birthday, p_phone);
```

```
        raise notice 'Студент % добавлен в таблицу Students', p_code_stud;
```

```
    else
```

```
        raise notice 'Студент % уже есть в таблице Students', p_code_stud;
```

```
    end if;
```

```
end;
```

```
$BODY$;
```

```
select * from insert_students('S001', 'Иванов', 'Иван', 'Иванович', 101, '2002-01-01',8971047828);
```

-- Триггерная функция

```
CREATE OR REPLACE FUNCTION public.check_students_insert()
```

```
    RETURNS trigger
```

```
    LANGUAGE 'plpgsql'
```

```
    VOLATILE
```

```
    COST 100
```

```
AS $BODY$
```

```
begin
```

```
    if not exists (select 1
```

```

        from "Students"
        where "Code_stud" = new."Code_stud")then
        perform public.insert_students(new."Code_stud",
        1, 'Фамилия', 'Имя', 'Отчество', '2006-09-10',89170486729);
    end if;
    return new;
end;
$BODY$;
--Триггер
create trigger insert_progress_before
before insert on "Progress"
for each row
execute function public.check_students_insert();
-- Проверка
-- Добавляем студента в таблицу Students
INSERT INTO "Students" ("Code_stud", "Surname", "Name", "Lastname", "Code_group", "Birthday", "Phone")
VALUES ('S001','Иванов', 'Иван', 'Иванович', 101, '2000-01-01', 1234567890);
-- Вставляем запись в таблицу Progress
INSERT INTO "Progress" ("Code_stud", "Code_subject", "Code_lector", "Date_exam", "Estimate",
"Code_progress")
VALUES ('S001', 1, 3489, '2023-10-01', 5, 10);
-- Вставляем запись в таблицу Progress для несуществующего студента
INSERT INTO "Progress" ("Code_stud", "Code_subject", "Code_lector", "Date_exam", "Estimate",
"Code_progress")
VALUES ('S002', 1, 3489, '2023-10-01', 4, 12);

```

#### Объяснение:

**Функция insert\_students():** Проверяем существование студента по коду и добавляем нового студента, если его нет.

**Триггерная функция check\_students\_insert():** Срабатывает перед вставкой в таблицу "Progress" и проверяем существование студента.

**Триггер insert\_progress\_before:** Вызываем проверку перед каждой вставкой оценок.

#### Вывод результата:

```

SELECT * FROM "Students";
SELECT * FROM "Progress";

```

	Code_stud [PK] character (10)	Surname character (25)	Name character (25)	Lastname character (25)	Code_group integer	Birthday date	Phone numeric	Avg_Estimate real
1	S003	Васильева ...	Валентина ...	Сергеевна ...	101	2006-09-10	89170486729	4
2	S002	Васильева ...	Валентина ...	Сергеевна ...	101	2006-09-10	89170486729	4
3	S001	Иванов ...	Иван ...	Иванович ...	101	2000-01-01	1234567890	5
4	D67890	Никитина ...	Елизавета ...	Игоревна ...	102	2005-07-30	8901234568	3
5	E12345	Андреев ...	Тимур ...	нет сведений ...	101	2002-04-18	9012345678	5

	Code_stud character (15)	Code_subject integer	Code_lector integer	Date_exam date	Estimate integer	Code_progress [PK] integer
1	I12345	5	5434	2023-06-23	3	9
2	H67890	4	3489	2023-06-22	4	8
3	E12345	1	1237	2023-06-19	5	5
4	D67890	4	3489	2023-06-18	3	4
5	C12345	3	2370	2023-06-17	5	3
6	B67890	2	1238	2023-06-16	4	2
7	K12345	1	2370	2023-06-25	2	11
8	S001	1	3489	2023-10-01	5	10
9	S002	1	3489	2023-10-01	4	12
10	A12345	1	7870	2023-06-15	5	1
11	S003	1	3489	2023-10-01	4	13

4. Запретить вставлять новые строки в таблицу Группы, выводя при этом сообщение «Вставка строк запрещена».

--Триггерная функция

```
CREATE OR REPLACE FUNCTION public.insert_groups()
```

```
RETURNS trigger
```

```
LANGUAGE 'plpgsql'
```

```
VOLATILE
```

```
COST 100
```

```
AS $BODY$
```

```
begin
```

```
    raise exception 'Вставка строк запрещена';
```

```
    return null;
```

```
end;
```

```
$BODY$;
```

```
-- Триггер
```

```
create trigger insert_groups_stroke
```

```
before insert on "Groups"
```

```
    for each row
```



```
execute function public.insert_groups();  
INSERT INTO "Groups" ("Code_group", "Name_group", "Num_course", "Name_speciality")  
VALUES (1, 'Группа Е', 1, 'Геодезия и картография');  
SELECT * FROM "Groups";
```

#### Объяснение:

При вызове генерируем исключение с сообщением "Вставка строк запрещена", прерывая операцию.

Вызываем функцию insert\_groups **перед** каждой попыткой вставки.

#### Вывод результата:

```
ERROR: Вставка строк запрещена  
CONTEXT: функция PL/pgSQL insert_groups(), строка 2, оператор RAISE  
  
ОШИБКА: Вставка строк запрещена  
SQL-состояние: P0001
```

# Транзакции

1. Проверьте выполнение команд транзакции при добавлении новой информации о преподавателях.

```
set transaction isolation level read committed;  
begin;
```

```
insert into "Lectors" ("Name_lector", "Science", "Post", "Date_")  
values ('Антипов Антон Иванович', 'Доктор наук', 'Доцент', '2023-10-01');
```

```
savepoint after_first_insert;
```

	Code_lector [PK] integer	Name_lector character (50)	Science character (30)	Post character (25)	Date_ date
3	2370	Лебедев Максим Макси...	Кандидат нау...	Старший преп...	2015-09-25
4	7870	Смирнова Елена Дмитр...	Доктор наук ...	Доцент ...	2012-02-18
5	5434	Сидорова Мария Серге...	Кандидат нау...	Доцент ...	2007-07-05
6	3489	Иванова Светлана Игор...	Доктор наук ...	Профессор ...	2007-11-20
7	3458	Макаров Даниил Данил...	Доктор наук ...	Старший преп...	2010-03-10
8	6769	Кузнецов Андрей Андре...	Кандидат нау...	Профессор ...	1999-08-15
9	1237	Козлова Анна Николае...	Доктор наук ...	Доцент ...	2004-09-01
10	7871	Петров Савелий Яковле...	к.т.н. ...	[null]	[null]
11	7872	Иванов Иван ...	Доктор наук ...	Доцент ...	2023-10-01
12	7873	Новикова Арина ...	Доктор наук ...	Доцент ...	2023-10-01
13	7874	Иванов Иван Иванович ...	Доктор наук ...	Доцент ...	2020-01-01
14	7875	Антипов Антон Иванов...	Доктор наук ...	Доцент ...	2023-10-01

```
insert into "Lectors" ("Name_lector", "Science", "Post", "Date_")  
values ('Сергеев Александр Петрович', 'Доктор наук', 'Профессор', '2023-10-02');
```

	Code_lector [PK] integer	Name_lector character (50)	Science character (30)	Post character (25)	Date_ date
3	2370	Лебедев Максим Макси...	Кандидат нау...	Старший преп...	2015-09-25
4	7870	Смирнова Елена Дмитр...	Доктор наук ...	Доцент ...	2012-02-18
5	5434	Сидорова Мария Серге...	Кандидат нау...	Доцент ...	2007-07-05
6	3489	Иванова Светлана Игор...	Доктор наук ...	Профессор ...	2007-11-20
7	3458	Макаров Даниил Данил...	Доктор наук ...	Старший преп...	2010-03-10
8	6769	Кузнецов Андрей Андре...	Кандидат нау...	Профессор ...	1999-08-15
9	1237	Козлова Анна Николае...	Доктор наук ...	Доцент ...	2004-09-01
10	7871	Петров Савелий Яковле...	к.т.н. ...	[null]	[null]
11	7872	Иванов Иван ...	Доктор наук ...	Доцент ...	2023-10-01
12	7873	Новикова Арина ...	Доктор наук ...	Доцент ...	2023-10-01
13	7874	Иванов Иван Иванович ...	Доктор наук ...	Доцент ...	2020-01-01
14	7875	Антипов Антон Иванов...	Доктор наук ...	Доцент ...	2023-10-01
15	7876	Сергеев Александр Пет...	Доктор наук ...	Профессор ...	2023-10-02

```
select * from "Lectors"  
where "Name_lector" in ('Антипов Антон Иванович', 'Сергеев Александр Петрович');
```

	Code_lector [PK] integer	Name_lector character (50)	Science character (30)	Post character (25)	Date_ date
1	7875	Антипов Антон Иванов...	Доктор наук ...	Доцент ...	2023-10-01
2	7876	Сергеев Александр Пет...	Доктор наук ...	Профессор ...	2023-10-02

rollback to savepoint after\_first\_insert;

select \* from "Lectors"

where "Name\_lector" in ('Антипов Антон Иванович', 'Сергеев Александр Петрович' );

	Code_lector [PK] integer	Name_lector character (50)	Science character (30)	Post character (25)	Date_ date
1	7875	Антипов Антон Иванов...	Доктор наук ...	Доцент ...	2023-10-01

commit;

select \* from "Lectors";

	Code_lector [PK] integer	Name_lector character (50)	Science character (30)	Post character (25)	Date_ date
3	2370	Лебедев Максим Макси...	Кандидат нау...	Старший преп...	2015-09-25
4	7870	Смирнова Елена Дмитр...	Доктор наук ...	Доцент ...	2012-02-18
5	5434	Сидорова Мария Серге...	Кандидат нау...	Доцент ...	2007-07-05
6	3489	Иванова Светлана Игор...	Доктор наук ...	Профессор ...	2007-11-20
7	3458	Макаров Даниил Данил...	Доктор наук ...	Старший преп...	2010-03-10
8	6769	Кузнецов Андрей Андре...	Кандидат нау...	Профессор ...	1999-08-15
9	1237	Козлова Анна Николае...	Доктор наук ...	Доцент ...	2004-09-01
10	7871	Петров Савелий Яковле...	к.т.н. ...	[null]	[null]
11	7872	Иванов Иван ...	Доктор наук ...	Доцент ...	2023-10-01
12	7873	Новикова Арина ...	Доктор наук ...	Доцент ...	2023-10-01
13	7874	Иванов Иван Иванович ...	Доктор наук ...	Доцент ...	2020-01-01
14	7875	Антипов Антон Иванов...	Доктор наук ...	Доцент ...	2023-10-01

### Объяснение:

Установили уровень изоляции "чтение подтвержденных данных" и начали транзакцию. Добавили 2-х преподавателей, после вставки 1 преподавателя создали точку сохранения after\_first\_insert, затем добавили 2-го преподавателя, вывели посмотреть именно добавленные строки, после чего откатили изменения до точки сохранения after\_first\_insert и вновь вывели только новые строки, в базе данных осталась лишь одна запись, 1-го преподавателя, после сохранили изменения.