

Южно-Уральский
государственный
университет

Национальный
исследовательский
университет

КРОК

SQL

PostgreSQL

(лекция 2)

КРОК

Челябинск, ул. Карла Маркса, д. 38

Смирнов Анатолий
Технический менеджер

Кузнецов Сергей
Старший инженер-разработчик

Фоменко Алексей
Младший инженер-разработчик



Ограничения целостности задаются при создании таблицы следующим образом:

```
create table ИМЯ_ТАБЛИЦЫ (  
    ИМЯ_СТОЛБЦА_1    ТИП_ДАННЫХ    ОГРАНИЧЕНИЕ_1    ОГРАНИЧЕНИЕ_2...,  
    ИМЯ_СТОЛБЦА_2    ТИП_ДАННЫХ    ОГРАНИЧЕНИЕ_1    ОГРАНИЧЕНИЕ_2...,  
    ...  
    ИМЯ_СТОЛБЦА_N    ТИП_ДАННЫХ    ОГРАНИЧЕНИЕ_1    ОГРАНИЧЕНИЕ_2...  
);
```



Первичным ключом (primary key) или идентификатором называют ограничение для однозначной идентификации записи в таблице.

- Первичные ключи уникальны: в таблице не может быть двух строк с одинаковыми первичными ключами
- Первичные ключи не могут принимать значение null

При нарушении приведенных правил, СУБД выдаст ошибку.

```
create table students (  
    id          serial primary key,  
    firstname   varchar(50),  
    surname     varchar(100),  
    lastname    varchar(100)  
);
```



Наполним таблицу students данными. При этом в списке столбцов команды insert не будем указывать столбец id

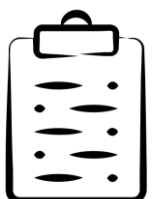
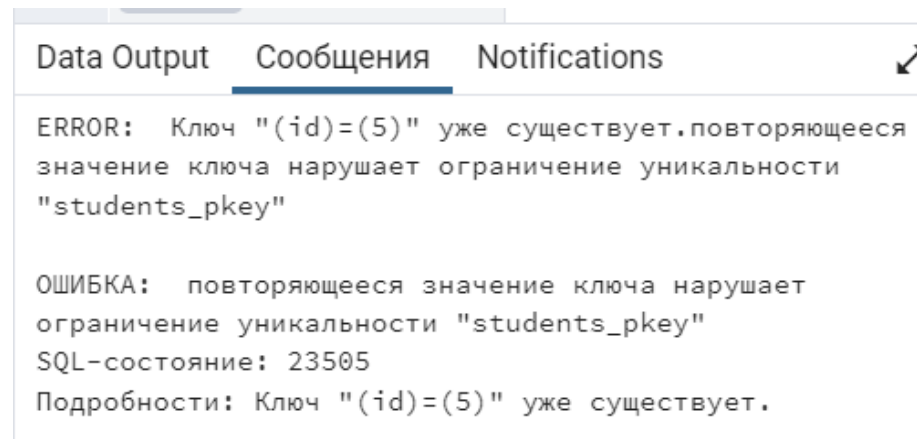
```
insert into students (surname, firstname, lastname)
values ('Петров', 'Иван', 'Иванович'),
       ('Петров', 'Иван', 'Иванович'),
       ('Алексеева', 'Дарья', 'Михайловна'),
       ('Иванова', 'Марина', 'Дмитриевна'),
       ('Лебедев', 'Сергей', 'Дмитриевич'),
       ('Фёдорова', 'Анна', 'Владимировна'),
       ('Иванова', 'Кристина', 'Дмитриевна')
```

id [PK] integer	firstname character varying (50)	surname character varying (100)	lastname character varying (100)
1	Иван	Петров	Иванович
2	Иван	Петров	Иванович
3	Дарья	Алексеева	Михайловна
4	Марина	Иванова	Дмитриевна
5	Сергей	Лебедев	Дмитриевич
6	Анна	Фёдорова	Владимировна
7	Кристина	Иванова	Дмитриевна



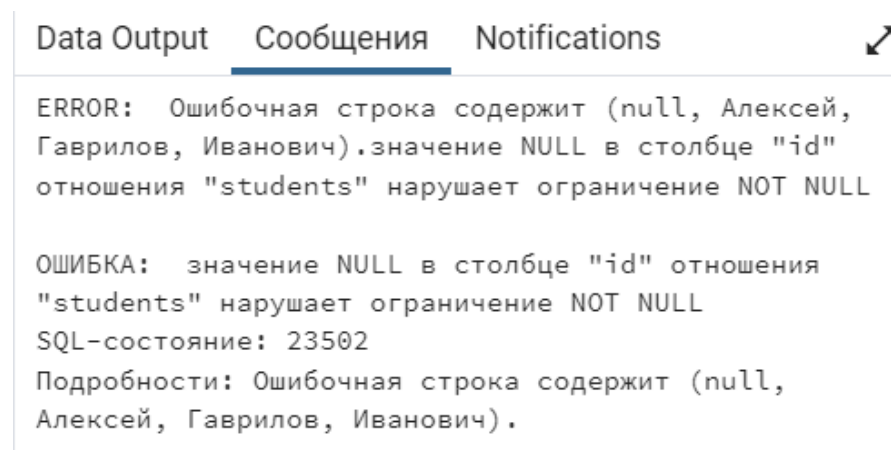
Попробуем добавить нового студента с id равным 5. Данный id уже присутствует в таблице.

```
insert into students
(id, surname, firstname, lastname)
values
(5, 'Гаврилов', 'Алексей', 'Иванович')
```



Теперь попробуем добавить значение null

```
insert into students
(id, surname, firstname, lastname)
values
(null, 'Гаврилов', 'Алексей',
'Иванович')
```





Данное ограничение не позволяет добавлять пустые null значения в указанный столбец.

```
create table students (  
    id          serial primary key,  
    firstname  varchar(50) not null,  
    surname    varchar(100) not null,  
    lastname   varchar(100)  
);
```

```
alter table students  
alter column lastname set not null;  
  
alter table students  
alter column lastname drop not null;
```

```
insert into students (surname, firstname, lastname)  
values (null, 'Алексей', 'Иванович')
```

Data Output Сообщения Notifications

ERROR: Ошибочная строка содержит (1, Алексей, null, Иванович).значение NULL в столбце "surname" отношения "students" нарушает ограничение NOT NULL

ОШИБКА: значение NULL в столбце "surname" отношения "students" нарушает ограничение NOT NULL

SQL-состояние: 23502

Подробности: Ошибочная строка содержит (1, Алексей, null, Иванович).





Иногда значение столбца или нескольких столбцов должно быть уникальным.

```
create table students (  
    id          serial          primary key,  
    firstname   varchar(50)     not null,  
    surname     varchar(100)    not null,  
    lastname    varchar(100),  
    ser_passport varchar(4)      not null,  
    num_passport varchar(6)      not null,  
    unique(ser_passport, num_passport)  
);
```

students

General Столбцы Дополнительно Ограничения Параметры Безопасность SQL

Первичный ключ Внешний ключ Проверка Уникальность Исключить

Имя		Столбцы
 	students_ser_passport_num_passport_key	ser_passport,num_passport



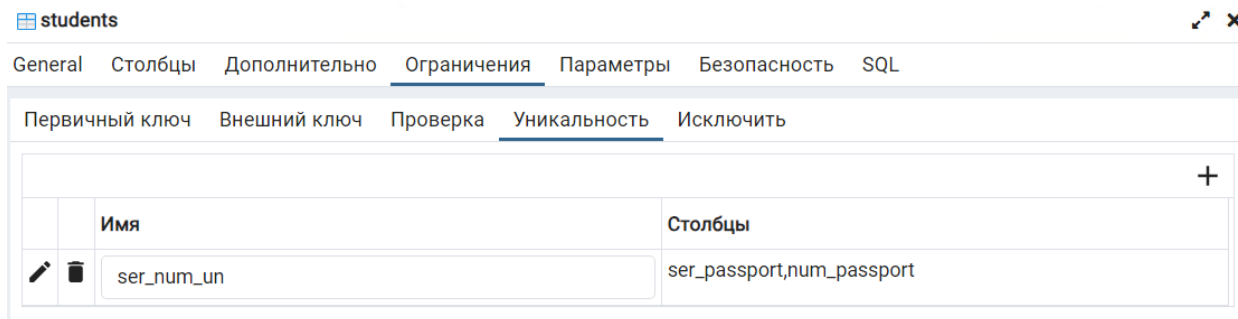
Можно создать с явным именем

```
create table students (  
    id          serial          primary key,  
    firstname   varchar(50)     not null,  
    surname     varchar(100)    not null,  
    lastname    varchar(100),  
    ser_passport varchar(4)      not null,  
    num_passport varchar(6)      not null,  
    constraint ser_num_un unique (ser_passport, num_passport)  
);
```



Либо добавить позже после create table:

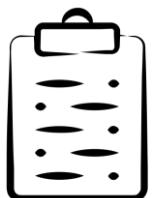
```
alter table students  
add constraint ser_num_un unique (ser_passport, num_passport);
```





Добавим в таблицу запись с одинаковой серией, ошибки нет:

```
insert into students (surname, firstname, lastname, ser_passport, num_passport)
values ('Романов', 'Сергей', 'Иванович', '1111', '556677'),
       ('Романов', 'Сергей', 'Иванович', '1111', '123456');
```



Попробуем добавить запись с уже существующей серией и номером паспорта, будет ошибка:

```
insert into students (surname, firstname, lastname, ser_passport, num_passport)
values ('Алексеев', 'Артём', 'Олегович', '1111', '556677')
```

Data Output Сообщения Notifications



ERROR: Ключ "(ser_passport, num_passport)=(1111, 556677)" уже существует.повторяющееся значение ключа нарушает ограничение уникальности "students_ser_passport_num_passport_key"

ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности
"students_ser_passport_num_passport_key"

SQL-состояние: 23505

Подробности: Ключ "(ser_passport, num_passport)=(1111, 556677)" уже существует.

Ограничения проверки. Checks



```
create table students (  
    id          serial          primary key,  
    firstname   varchar(50)     not null check(substr(firstname, 1, 1) =  
                                         upper(substr(firstname, 1, 1))),  
    surname     varchar(100)    not null,  
    lastname    varchar(100),  
    ser_passport varchar(4)      not null check(length(ser_passport) = 4),  
    num_passport varchar(6)      not null constraint c_num_passport  
                                         check(length(num_passport) = 6),  
    unique(ser_passport, num_passport)  
);
```

students			
General		Столбцы	
Дополнительно		Ограничения	
Параметры		Безопасность	
SQL			
Первичный ключ		Внешний ключ	
Проверка		Уникальность	
Исключить			
		+	
	Имя	Проверка	
	c_num_passport	length(num_passport::text) = 6	
	students_firstname_check	substr(firstname::text, 1, 1) = upper(substr(firstname::text,	
	students_ser_passport_check	length(ser_passport::text) = 4	



Добавление при помощи alter

```
create table students (  
    id          serial          primary key,  
    firstname   varchar(50)    not null,  
    surname     varchar(100)   not null,  
    lastname    varchar(100),  
    ser_passport varchar(4)     not null,  
    num_passport varchar(6)     not null,  
    unique(ser_passport, num_passport)  
);
```

```
alter table students add constraint ch_firstname_letter  
check(substr(firstname, 1, 1) = upper(substr(firstname, 1, 1)));
```

```
alter table students add constraint ch_ser_passport  
check(length(ser_passport) = 4);
```

```
alter table students add check(length(num_passport) = 6);
```

Первичный ключ		Внешний ключ	Проверка	Уникальность	>
					+
		Имя	Проверка		
		ch_firstname_letter	substr(firstname::text, 1, 1) = up		
		ch_ser_passport	length(ser_passport::text) = 4		
		students_num_passport_check	length(num_passport::text) = 6		



Попробуем добавить запись, у которой три цифры в серии:

```
insert into students (surname, firstname, lastname, ser_passport, num_passport)
values ('Краснов', 'Олег', 'Петрович', '123', '445566')
```

Data Output Сообщения Notifications

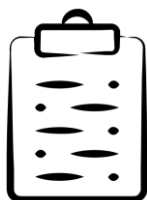


ERROR: Ошибочная строка содержит (1, Олег, Краснов, Петрович, 123, 445566).новая строка в отношении "students" нарушает ограничение-проверку "students_ser_passport_check"

ОШИБКА: новая строка в отношении "students" нарушает ограничение-проверку "students_ser_passport_check"

SQL-состояние: 23514

Подробности: Ошибочная строка содержит (1, Олег, Краснов, Петрович, 123, 445566).



Запись, у которой firstname с маленькой буквы:

```
insert into students (surname, firstname, lastname, ser_passport, num_passport)
values ('Краснов', 'олег', 'Петрович', '1234', '445566')
```

Data Output Сообщения Notifications



ERROR: Ошибочная строка содержит (4, олег, Краснов, Петрович, 1234, 445566).новая строка в отношении "students" нарушает ограничение-проверку "students_firstname_check"

ОШИБКА: новая строка в отношении "students" нарушает ограничение-проверку "students_firstname_check"

SQL-состояние: 23514

Подробности: Ошибочная строка содержит (4, олег, Краснов, Петрович, 1234, 445566).



Проверка на заполнение совокупности полей

```
create table students (  
    id          serial          primary key,  
    firstname   varchar(50)    not null,  
    surname     varchar(100)   not null,  
    lastname    varchar(100),  
    ser_passport varchar(4),  
    num_passport varchar(6),  
    unique(ser_passport, num_passport)  
);  
  
alter table students add  
check((ser_passport is null and num_passport is null) or  
      (ser_passport is not null and num_passport is not null));
```

id [PK] integer	firstname character varying (50)	surname character varying (100)	lastname character varying (100)	ser_passport character varying (4)	num_passport character varying (6)
1	олег	Краснов	Петрович	1234	445566
3	олег	Краснов	Петрович2	[null]	[null]
4	олег	Краснов	Петрович3	[null]	[null]
5	олег	Краснов	Петрович4	1234	11111



Если одно из полей null, будет ошибка

Запрос История запросов

```
1 insert into students (surname, firstname, lastname, ser_passport, num_passport)
2 values ('Краснов', 'олег', 'Петрович3', '1234', null)
```

Data Output Сообщения Notifications

ERROR: Ошибочная строка содержит (6, олег, Краснов, Петрович3, 1234, null).новая строка в отношении "students" нарушает ограничение-проверку "students_check"

ОШИБКА: новая строка в отношении "students" нарушает ограничение-проверку "students_check"

SQL-состояние: 23514

Подробности: Ошибочная строка содержит (6, олег, Краснов, Петрович3, 1234, null).

Запрос История запросов

```
1 insert into students (surname, firstname, lastname, ser_passport, num_passport)
2 values ('Краснов', 'олег', 'Петрович3', null, '111111')
```

Data Output Сообщения Notifications

ERROR: Ошибочная строка содержит (7, олег, Краснов, Петрович3, null, 111111).новая строка в отношении "students" нарушает ограничение-проверку "students_check"

ОШИБКА: новая строка в отношении "students" нарушает ограничение-проверку "students_check"

SQL-состояние: 23514

Подробности: Ошибочная строка содержит (7, олег, Краснов, Петрович3, null, 111111).

Задача: добавить город к таблице users



users

name	city
Иван	Челябинск
Анна	Екатеринбург
Алексей	Екатеринбург
Софья	Челябинск
Михаил	Тюмень
Илья	Екатеринбург
Мария	Екатеринбург

Неоптимально, город повторяется



cities

id	name
1	Челябинск
2	Екатеринбург
3	Тюмень

users

name	city
Иван	1
Анна	2
Алексей	2
Софья	1
Михаил	3
Илья	2
Мария	2

users.city – внешний ключ на cities.id



Чтобы наложить ограничение внешнего ключа для определенного столбца, необходимо написать ключевое слово **references**.

Если не делать никаких дополнительных действий в таблице `users`, то в столбец `city` можно будет добавить идентификаторы, которых не будет в таблице `cities`.

```
create table cities (  
    id    serial    primary key,  
    name varchar(100) not null  
);  
  
create table users (  
    id    serial    primary key,  
    name varchar(100) not null,  
    city integer    references cities (id)  
);
```




Заполнение данными

```
insert into cities (name)
values ('Челябинск'),
       ('Екатеринбург'),
       ('Тюмень');
```

```
insert into users (name, city)
values ('Иван', 1),
       ('Анна', 2),
       ('Алексей', 2),
       ('Софья', 1),
       ('Михаил', 2),
       ('Илья', 2),
       ('Мария', null);
```



Записи в cities с id = 4 не существует, будет ошибка

```
insert into users (name, city)
values ('Иван', 4);
```

Data Output	Сообщения	Notifications
ERROR: Ключ (city)=(4) отсутствует в таблице "cities".INSERT или UPDATE в таблице "users" нарушает ограничение внешнего ключа "users_city_fkey"		
ОШИБКА: INSERT или UPDATE в таблице "users" нарушает ограничение внешнего ключа "users_city_fkey"		
SQL-состояние: 23503		
Подробности: Ключ (city)=(4) отсутствует в таблице "cities".		



Чтобы соединять таблицы, в языке SQL предусмотрены специальные логические операторы соединения таблиц:

- join
- left join
- right join
- full join
- cross join



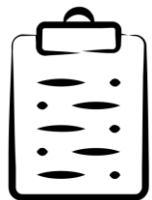
Используются почти в 99,9% случаев



Общий синтаксис, используется оператор on:

```
select *  
  from table1 t1  
    <оператор соединения> table1 t2 с on УСЛОВИЕ(-я)
```

Соединение inner join (join)



Порядок следования таблиц для внутреннего соединения не имеет значения. В выборке будут присутствовать только те строки, которые удовлетворяют условию соединения.

id	name
[PK] integer	character varying (10)
1	Челябинск
2	Екатеринбург
3	Тюмень

id	name	city
[PK] integer	character varying (100)	integer
1	Иван	1
2	Анна	2
3	Алексей	2
4	Софья	1
5	Михаил	2
6	Илья	2
7	Мария	[null]

```
select *  
  from users u  
 join cities c on c.id = u.city
```

id	name	city	id	name
integer	character varying (100)	integer	integer	character varying (100)
1	Иван	1	1	Челябинск
2	Анна	2	2	Екатеринбург
3	Алексей	2	2	Екатеринбург
4	Софья	1	1	Челябинск
5	Михаил	2	2	Екатеринбург
6	Илья	2	2	Екатеринбург

Соединение left outer join (left join)



- Сначала включаются все строки, которые удовлетворяют внутреннему соединению таблиц;
- В результирующую выборку добавляются строки из левой таблицы, которые не удовлетворяют внутреннему соединению.

id	name
[PK] integer	character varying (10)
1	Челябинск
2	Екатеринбург
3	Тюмень

id	name	city
[PK] integer	character varying (100)	integer
1	Иван	1
2	Анна	2
3	Алексей	2
4	Софья	1
5	Михаил	2
6	Илья	2
7	Мария	[null]

```
select *  
from users u  
left join cities c on c.id = u.city
```

id	name	city	id	name
integer	character varying (100)	integer	integer	character varying (100)
1	Иван	1	1	Челябинск
2	Анна	2	2	Екатеринбург
3	Алексей	2	2	Екатеринбург
4	Софья	1	1	Челябинск
5	Михаил	2	2	Екатеринбург
6	Илья	2	2	Екатеринбург
7	Мария	[null]	[null]	[null]

Соединение right outer join (right join)



- Сначала включаются все строки, которые удовлетворяют внутреннему соединению таблиц;
- В результирующую выборку добавляются строки из правой таблицы, которые не удовлетворяют внутреннему соединению.

id	name
[PK] integer	character varying (10)
1	Челябинск
2	Екатеринбург
3	Тюмень

id	name	city
[PK] integer	character varying (100)	integer
1	Иван	1
2	Анна	2
3	Алексей	2
4	Софья	1
5	Михаил	2
6	Илья	2
7	Мария	[null]

```
select *  
  from users u  
 right join cities c on c.id = u.city
```

id	name	city	id	name
integer	character varying (100)	integer	integer	character varying (100)
1	Иван	1	1	Челябинск
2	Анна	2	2	Екатеринбург
3	Алексей	2	2	Екатеринбург
4	Софья	1	1	Челябинск
5	Михаил	2	2	Екатеринбург
6	Илья	2	2	Екатеринбург
[null]	[null]	[null]	3	Тюмень

Соединение full outer join (full join)



- Сначала включаются все строки, которые удовлетворяют внутреннему соединению таблиц;
- В результирующую выборку добавляются строки из правой и из левой таблиц, которые не удовлетворяют внутреннему соединению.

id	name
[PK] integer	character varying (10)
1	Челябинск
2	Екатеринбург
3	Тюмень

id	name	city
[PK] integer	character varying (100)	integer
1	Иван	1
2	Анна	2
3	Алексей	2
4	Софья	1
5	Михаил	2
6	Илья	2
7	Мария	[null]

```
select *  
  from users u  
 full join cities c on c.id = u.city
```

id	name	city	id	name
integer	character varying (100)	integer	integer	character varying (100)
1	Иван	1	1	Челябинск
2	Анна	2	2	Екатеринбург
3	Алексей	2	2	Екатеринбург
4	Софья	1	1	Челябинск
5	Михаил	2	2	Екатеринбург
6	Илья	2	2	Екатеринбург
7	Мария	[null]	[null]	[null]
[null]	[null]	[null]	3	Тюмень

Соединение full outer join (full join)

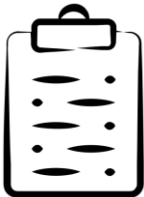
id	name
[PK] integer	character varying (10)
1	Челябинск
2	Екатеринбург
3	Тюмень

id	name	city
[PK] integer	character varying (100)	integer
1	Иван	1
2	Анна	2
3	Алексей	2
4	Софья	1
5	Михаил	2
6	Илья	2
7	Мария	[null]

```
select *  
  from users u  
 full join cities c on 1 = 0
```

id	name	city	id	name
integer	character varying (100)	integer	integer	character varying (100)
[null]	[null]	[null]	1	Челябинск
[null]	[null]	[null]	2	Екатеринбург
[null]	[null]	[null]	3	Тюмень
1	Иван	1	[null]	[null]
2	Анна	2	[null]	[null]
3	Алексей	2	[null]	[null]
4	Софья	1	[null]	[null]
5	Михаил	2	[null]	[null]
6	Илья	2	[null]	[null]
7	Мария	[null]	[null]	[null]

Перекрестное соединение cross join



- Данный тип соединения не зависит от расположения таблиц. В результирующую выборку попадают все столбцы из обеих таблиц. Строки соединяются по следующему алгоритму: каждая строка первой таблицы соединяется со всеми строками второй таблицы.

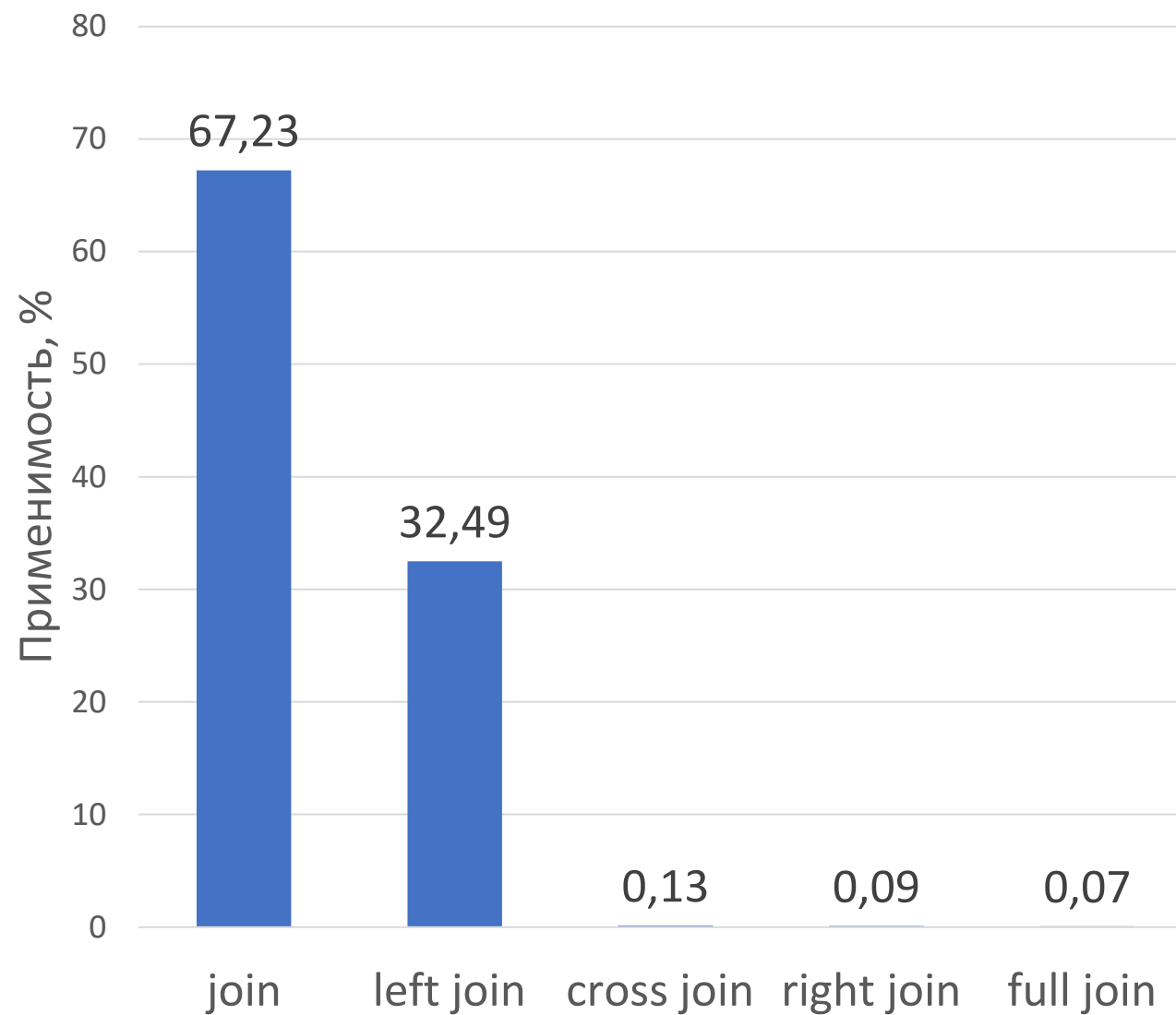
id	name
[PK] integer	character varying (10)
1	Челябинск
2	Екатеринбург
3	Тюмень

id	name	city
[PK] integer	character varying (100)	integer
1	Иван	1
2	Анна	2
3	Алексей	2
4	Софья	1
5	Михаил	2
6	Илья	2
7	Мария	[null]

```
select *
  from users u
 cross join cities c
```

	id	name	city	id	name
	integer	character varying (100)	integer	integer	character varying (100)
13	5	Михаил	2	1	Челябинск
14	5	Михаил	2	2	Екатеринбург
15	5	Михаил	2	3	Тюмень
16	6	Илья	2	1	Челябинск
17	6	Илья	2	2	Екатеринбург
18	6	Илья	2	3	Тюмень
19	7	Мария	[null]	1	Челябинск
20	7	Мария	[null]	2	Екатеринбург
21	7	Мария	[null]	3	Тюмень

Соединение	Сколько раз встречается в проекте МИС «БАРС», шт.	
join	41443	99,7%
left join	20026	
cross join	81	
right join	55	
full join	41	





Помимо фильтрации данных одной из распространенных операций в языке SQL, является группировка данных. Для группировки данных используется оператор `group by`.

```
select c.name,  
       count(u.id) cnt  
from cities c  
join users u on u.city = c.id  
group by c.name
```

name character varying (100) 🔒	cnt bigint 🔒
Челябинск	2
Екатеринбург	4

```
select c.name,  
       count(u.id) cnt  
from cities c  
left join users u on u.city = c.id  
group by c.name
```

name character varying (100) 🔒	cnt bigint 🔒
Тюмень	0
Челябинск	2
Екатеринбург	4

```
create table flights (  
    id          serial primary key,  
    date_when   date    not null,  
    user_id     integer not null references users(id),  
    status      integer not null  
);
```

```
insert into flights (date_when, user_id, status)  
values ('2022-01-01', 1, 0),  
      ('2023-02-25', 1, 0),  
      ('2022-04-04', 1, 1),  
      ('2022-10-04', 2, 1),  
      ('2023-12-05', 2, 2),  
      ('2023-09-15', 3, 2),  
      ('2022-10-07', 3, 0),  
      ('2022-08-11', 3, 2),  
      ('2023-07-22', 3, 2);
```

Формат даты по умолчанию
yyyy-mm-dd

Пример 1

id [PK] integer	date_when date	user_id integer	status integer
1	2022-01-01	1	0
2	2023-02-25	1	0
3	2022-04-04	1	1
4	2022-10-04	2	1
5	2023-12-05	2	2
6	2023-09-15	3	2
7	2022-10-07	3	0
8	2022-08-11	3	2
9	2023-07-22	3	2

id [PK] integer	name character varying	city integer
1	Иван	1
2	Анна	2
3	Алексей	2
4	Софья	1
5	Михаил	2
6	Илья	2
7	Мария	[null]

```
select t1.name,  
       t2.date_when,  
       t2.status  
from users t1  
left join flights t2 on t2.user_id = t1.id
```

name character varying (100)	date_when date	status integer
Иван	2022-01-01	0
Иван	2023-02-25	0
Иван	2022-04-04	1
Анна	2022-10-04	1
Анна	2023-12-05	2
Алексей	2023-09-15	2
Алексей	2022-10-07	0
Алексей	2022-08-11	2
Алексей	2023-07-22	2
Михаил	[null]	[null]
Илья	[null]	[null]
Софья	[null]	[null]
Мария	[null]	[null]

Пример 1. Продолжение

id [PK] integer	date_when date	user_id integer	status integer
1	2022-01-01	1	0
2	2023-02-25	1	0
3	2022-04-04	1	1
4	2022-10-04	2	1
5	2023-12-05	2	2
6	2023-09-15	3	2
7	2022-10-07	3	0
8	2022-08-11	3	2
9	2023-07-22	3	2

```
select t1.name,  
       t2.date_when,  
       t2.status  
from users t1  
left join flights t2 on t2.user_id = t1.id  
where t2.status = 0
```

?

id [PK] integer	name character varyi	city integer
1	Иван	1
2	Анна	2
3	Алексей	2
4	Софья	1
5	Михаил	2
6	Илья	2
7	Мария	[null]

```
select t1.name,  
       t2.date_when,  
       t2.status  
from users t1  
left join flights t2 on t2.user_id = t1.id  
and t2.status = 0
```

?

Пример 1. Продолжение

```
select t1.name,  
       t2.date_when,  
       t2.status  
from users t1  
left join flights t2 on t2.user_id = t1.id  
where t2.status = 0
```

name character varying (100)	date_when date	status integer
Иван	2022-01-01	0
Иван	2023-02-25	0
Алексей	2022-10-07	0

```
select t1.name,  
       t2.date_when,  
       t2.status  
from users t1  
left join flights t2 on t2.user_id = t1.id  
and t2.status = 0
```

name character varying (100)	date_when date	status integer
Иван	2023-02-25	0
Иван	2022-01-01	0
Анна	[null]	[null]
Алексей	2022-10-07	0
Софья	[null]	[null]
Михаил	[null]	[null]
Илья	[null]	[null]
Мария	[null]	[null]

Пример 2

Вывести кол-во записей в flights с группировкой по статусам.

id [PK] integer	date_when date	user_id integer	status integer
1	2022-01-01	1	0
2	2023-02-25	1	0
3	2022-04-04	1	1
4	2022-10-04	2	1
5	2023-12-05	2	2
6	2023-09-15	3	2
7	2022-10-07	3	0
8	2022-08-11	3	2
9	2023-07-22	3	2

```
select 'Кол-во статусов ' || f.status status,  
       count(f.id) cnt  
from flights f  
group by f.status
```



status text	cnt bigint
Кол-во статусов 0	3
Кол-во статусов 2	4
Кол-во статусов 1	2

Пример 3

Вывести имя пользователя, название города и дату полета по связям `users.city = cities.id`, `flights.user_id = users.id`. Если данных по связи нет, выводить null

```
select u.name user_name,  
       c.name city_name,  
       f.date_when  
from users u  
left join cities c on c.id = u.city  
left join flights f on f.user_id = u.id  
order by u.name, f.date_when
```

user_name character varying (100)	city_name character varying (100)	date_when date
Алексей	Екатеринбург	2022-08-11
Алексей	Екатеринбург	2022-10-07
Алексей	Екатеринбург	2023-07-22
Алексей	Екатеринбург	2023-09-15
Анна	Екатеринбург	2022-10-04
Анна	Екатеринбург	2023-12-05
Иван	Челябинск	2022-01-01
Иван	Челябинск	2022-04-04
Иван	Челябинск	2023-02-25
Илья	Екатеринбург	[null]
Мария	[null]	[null]
Михаил	Екатеринбург	[null]
Софья	Челябинск	[null]

Пример 4

Вывести, сколько пользователей живет в каждом городе

id	name
[PK] integer	character varying (100)
1	Челябинск
2	Екатеринбург
3	Тюмень

```
select c.name city_name,  
       count(u.id) || ' чел.' cnt  
from cities c  
left join users u on u.city = c.id  
group by c.name
```

id	name	city
[PK] integer	character varying (100)	integer
1	Иван	1
2	Анна	2
3	Алексей	2
4	Софья	1
5	Михаил	2
6	Илья	2
7	Мария	[null]



city_name	cnt
character varying (100)	text
Тюмень	0 чел.
Челябинск	2 чел.
Екатеринбург	4 чел.

Пример 5

Вывести список всех записей из users и дату минимального полета по связи `flights.user_id = users.id`.

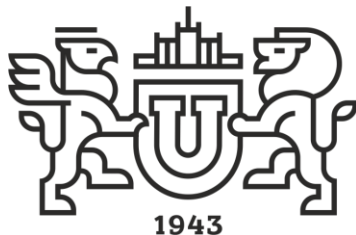
id [PK] integer	date_when date	user_id integer	status integer
1	2022-01-01	1	0
2	2023-02-25	1	0
3	2022-04-04	1	1
4	2022-10-04	2	1
5	2023-12-05	2	2
6	2023-09-15	3	2
7	2022-10-07	3	0
8	2022-08-11	3	2
9	2023-07-22	3	2

id [PK] integer	name character varying	city integer
1	Иван	1
2	Анна	2
3	Алексей	2
4	Софья	1
5	Михаил	2
6	Илья	2
7	Мария	[null]



```
select *  
  from (select u.name user_name,  
              min(f.date_when) date_min  
        from users u  
       left join flights f on f.user_id = u.id  
      group by u.name  
     ) t  
 order by t.date_min
```

user_name character varying (100)	date_min date
Иван	2022-01-01
Алексей	2022-08-11
Анна	2022-10-04
Илья	[null]
Мария	[null]
Михаил	[null]
Софья	[null]



Южно-Уральский
государственный
университет

Национальный
исследовательский
университет

КРОК

Спасибо за внимание!



КРОК

Челябинск, ул. Карла Маркса, д. 38