Работа с СУБД SQL

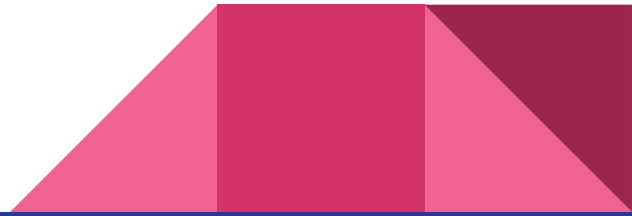
Объединения таблиц. Подзапросы.

Проверить, идет ли запись

Меня хорошо видно && слышно?

Проверка

- настройка микрофона и аудио
- проверка работы чата



Внешние ключи

Ограничение Foreign Key (внешний ключ)

Таблицы БД могут быть связаны отношением *основная – подчиненная* (master – detail). Например, *Группа – Студент* или *Улица – Дом*.

В реляционных БД такая связь реализуется как ассоциативная, помещением в подчиненную таблицу (detail) первичного ключа или ключа со свойством *UNIQUE* основной таблицы.

Например, пусть имеются таблицы

Streets(Street_ID int identity(1,1) primary key, StreetName varchar(22))

Houses(House_ID int identity(1,1) primary key, Street_ID int, HouseNumber char(5))

Чтобы указать принадлежность дома улице, в таблицу HOUSES помещается идентификатор улицы *Street_ID*.

Чтобы указать, что поле *Houses.Street_ID* является ссылкой на запись из таблицы *Streets*, используется ограничение *Foreign Key*.

Для приведенного примера оператор создания таблицы Houses с таким ограничением, имеет вид:

```
CREATE TABLE Houses( House_ID int identity(1,1) primary key, Street_ID int not null,
    HouseNumber char(12) not null,
    constraint FK_StreetsHouses foreign key (Street_ID) references Streets(Street_ID))
```

Если поле *foreign key* является единственным, то ограничение может быть установлено на уровне поля:

```
CREATE TABLE Houses(
  House_ID int identity(1,1) primary key,
  Street_ID int not null foreign key references Streets(Street_ID),
  HouseNumber char(12) not null
);
```

Полный синтаксис ограничения foreign key

CONSTRAINT <имя ограничения> FOREIGN KEY [(<имя поля> [, <имя поля>] …)]
REFERENCES <имя master таблицы>

[(<имя поля> [, <имя поля>]…)]

[ON DELETE { CASCADE | NO ACTION | SET NULL | SET DEFAULT}]

[ON UPDATE { CASCADE | NO ACTION | SET NULL | SET DEFAULT}]

Полный синтаксис ограничения foreign key

Фразы *ON DELETE* и *ON UPDATE* указывают серверу, что делать при удалении записи в таблице *master* или что делать при изменении ключа, на который ссылается *foreign key*. Возможные действия:

- NO_ACTION – никаких действий

- CASCADE – при удалении записи будут удалены все подчинённые записи, а при модификации записи – поля foreign key подчинённой таблицы (detail) будут изменены также как были изменены поля основной таблицы (master).

- SET NULL – поля foreign key detail – таблицы примут неопределённые значения

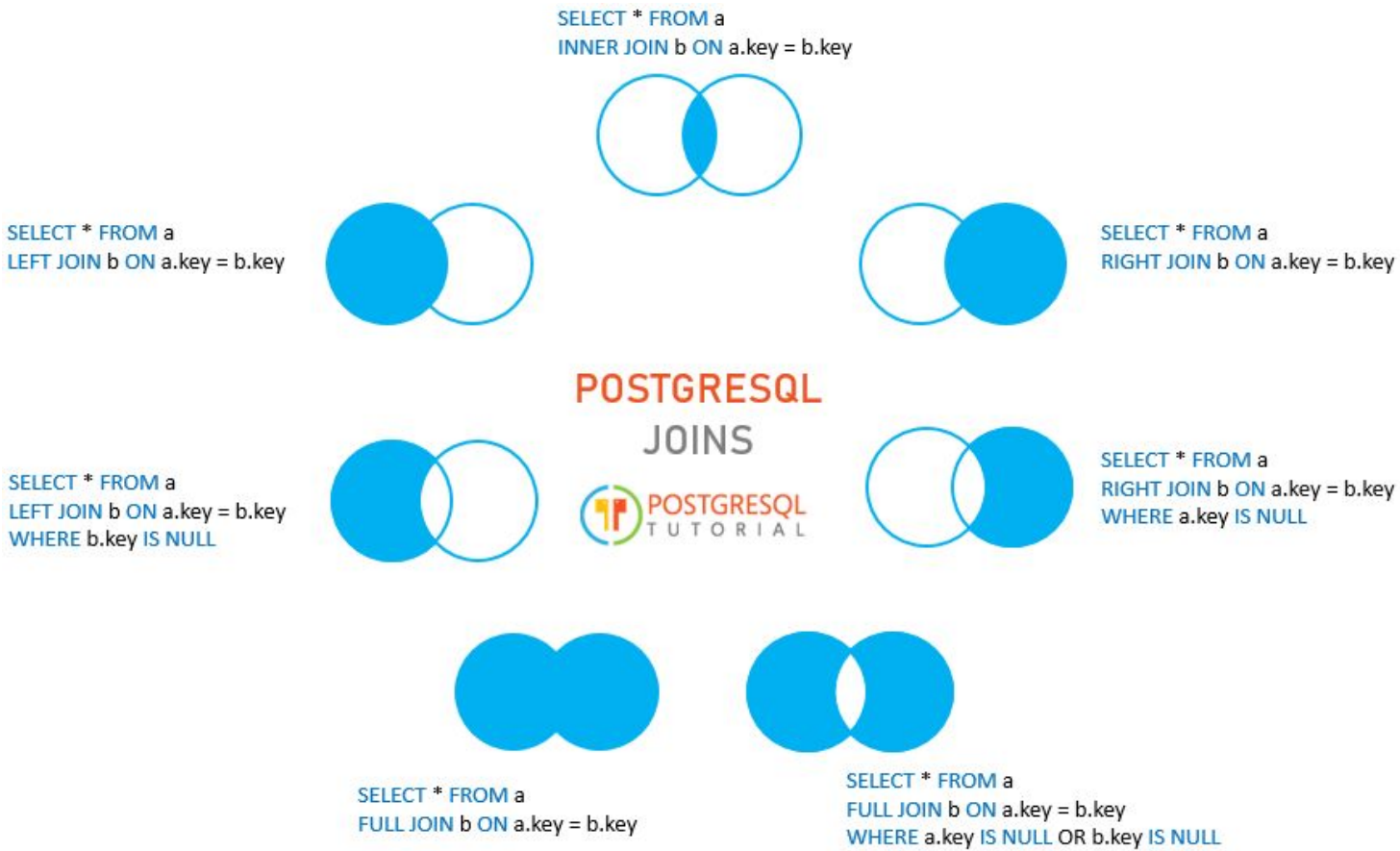
- SET DEFAULT - поля foreign key detail – таблицы примут значения по умолчанию

Нарушение ограничения foreign key возникает в двух случаях:

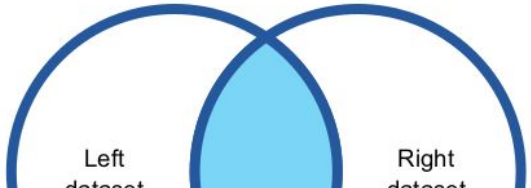
- при попытке создать подчиненную запись, для которой отсутствует запись в основной таблице, например, при попытке записи в таблицу *Houses* идентификатора *Street_ID* несуществующей улицы.
  - Исключением является ситуация, когда поле(я) внешнего ключа допускают неопределенное значение и в них помещается *null*.
- при попытке удаления записи из таблицы *master*, для которой существуют подчиненные, например, при попытке удаления улицы, на которой имеются дома.

Объединения таблиц

JOIN в PostgreSQL



Inner Join




```
+---+-------+
| id|  value |
+---+-------+
|  1 |Value 1|
|  2 |Value 2|
|  3 |Value 3|
|  4 |Value 4|
+---+-------+
```

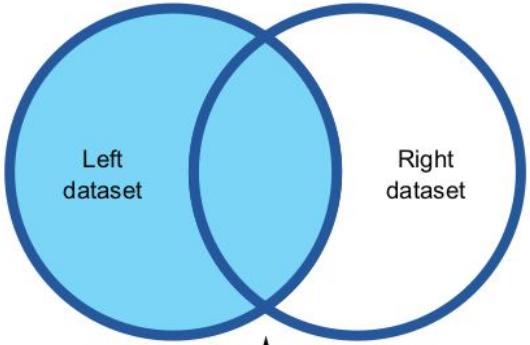
INNER JOIN

```
+---+-------+---+---------+
| id|  value | id|   value  |
+---+-------+---+---------+
|  3 |Value 3|  3 |Value 3  |
|  4 |Value 4|  4 |Value 4  |
|  4 |Value 4|  4 |Value 4_1|
+---+-------+---+---------+
```

```
+---+---------+
| id|   value  |
+---+---------+
|  3 |Value 3  |
|  4 |Value 4  |
|  4 |Value 4_1|
|  5 |Value 5  |
|  6 |Value 6  |
+---+---------+
```

https://www.db-fiddle.com/f/fUfQvv8QaXG5wdMWr9zzbA/5

Left [OUTER] Join




```
+---+-------+
| id| value |
+---+-------+
|  1|Value 1|
|  2|Value 2|
|  3|Value 3|
|  4|Value 4|
+---+-------+
```

LEFT JOIN
…
LEFT_OUTER JOIN

```
+---+-------+----+---------+
| id| value | id |  value  |
+---+-------+----+---------+
|  1|Value 1|null|null     |
|  2|Value 2|null|null     |
|  3|Value 3|  3 |Value 3  |
|  4|Value 4|  4 |Value 4_1|
|  4|Value 4|  4 |Value 4  |
+---+-------+----+---------+
```

```
+---+---------+
| id|  value  |
+---+---------+
|  3|Value 3  |
|  4|Value 4  |
|  4|Value 4_1|
|  5|Value 5  |
|  6|Value 6  |
+---+---------+
```

https://www.db-fiddle.com/f/wHak6mbZeymqcSSgFzPEuP/3

Right [OUTER] Join

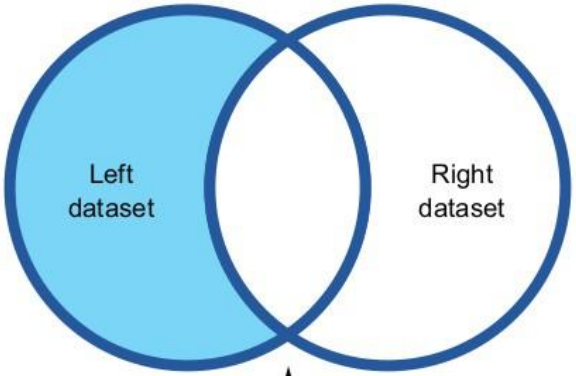

```
+---+-------+
| id| value |
+---+-------+
|  1|Value 1|
|  2|Value 2|
|  3|Value 3|
|  4|Value 4|
+---+-------+
```

RIGHT JOIN
…
RIGHT_OUTER JOIN

```
+----+-------+---+---------+
| id | value | id|  value  |
+----+-------+---+---------+
|null|null   |  5|Value 5  |
|null|null   |  6|Value 6  |
|   3|Value 3|  3|Value 3  |
|   4|Value 4|  4|Value 4  |
|   4|Value 4|  4|Value 4_1|
+----+-------+---+---------+
```

```
+---+---------+
| id|  value  |
+---+---------+
|  3|Value 3  |
|  4|Value 4  |
|  4|Value 4_1|
|  5|Value 5  |
|  6|Value 6  |
+---+---------+
```

Left-Semi Join



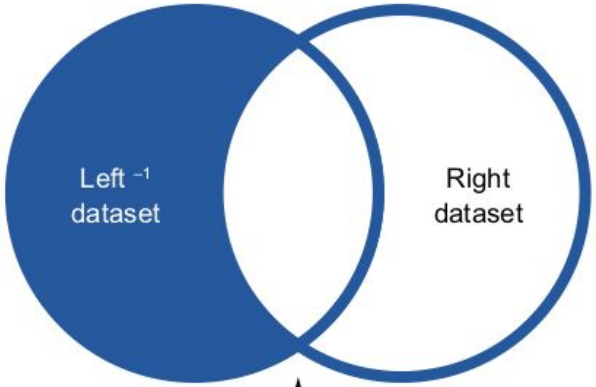

```
+---+-------+
| id| value |
+---+-------+
|  1|Value 1|
|  2|Value 2|
|  3|Value 3|
|  4|Value 4|
+---+-------+
```

LEFT_SEMI JOIN

```
+---+-------+
| id| value |
+---+-------+
|  3|Value 3|
|  4|Value 4|
+---+-------+
```

```
+---+---------+
| id|   value |
+---+---------+
|  3|Value 3  |
|  4|Value 4  |
|  4|Value 4_1|
|  5|Value 5  |
|  6|Value 6  |
+---+---------+
```

Left-Anti Join




```
+---+-------+
| id| value |
+---+-------+
|  1|Value 1|
|  2|Value 2|
|  3|Value 3|
|  4|Value 4|
+---+-------+
```

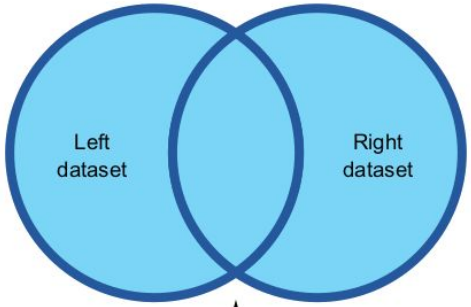
LEFT_ANTI JOIN

```
+---+-------+
| id| value |
+---+-------+
|  1|Value 1|
|  2|Value 2|
+---+-------+
```

```
+---+---------+
| id|   value |
+---+---------+
|  3|Value 3  |
|  4|Value 4  |
|  4|Value 4_1|
|  5|Value 5  |
|  6|Value 6  |
+---+---------+
```

https://www.db-fiddle.com/f/6dPvL1tA3WLc98u7pHz9ZQ/1

FULL [Outer] Join



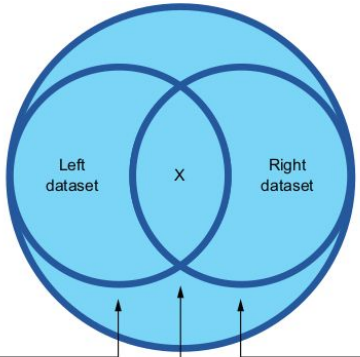

```
+---+-------+
| id| value |
+---+-------+
|  1|Value 1|
|  2|Value 2|
|  3|Value 3|
|  4|Value 4|
+---+-------+
```

```
+---+---------+
| id|  value  |
+---+---------+
|  3|Value 3  |
|  4|Value 4  |
|  4|Value 4_1|
|  5|Value 5  |
|  6|Value 6  |
+---+---------+
```

OUTER JOIN
…
FULL JOIN
…
FULL_OUTER JOIN

```
+----+-------+----+---------+
| id | value | id |  value  |
+----+-------+----+---------+
|null|null   |  5 |Value 5  |
|null|null   |  6 |Value 6  |
|  1 |Value 1|null|null     |
|  2 |Value 2|null|null     |
|  3 |Value 3|  3 |Value 3  |
|  4 |Value 4|  4 |Value 4  |
|  4 |Value 4|  4 |Value 4_1|
+----+-------+----+---------+
```

Cross-Join



id	value
1	Value 1
2	Value 2
3	Value 3
4	Value 4

CROSS JOIN

id	value	id	value
1	Value 1	4	Value 4_1
1	Value 1	3	Value 3
1	Value 1	5	Value 5
1	Value 1	4	Value 4
1	Value 1	6	Value 6
2	Value 2	3	Value 3
2	Value 2	4	Value 4_1
2	Value 2	6	Value 6
2	Value 2	4	Value 4
2	Value 2	5	Value 5
3	Value 3	4	Value 4
3	Value 3	4	Value 4_1
3	Value 3	5	Value 5
3	Value 3	6	Value 6
3	Value 3	3	Value 3
4	Value 4	3	Value 3
4	Value 4	4	Value 4
4	Value 4	4	Value 4_1
4	Value 4	6	Value 6
4	Value 4	5	Value 5

id	value
3	Value 3
4	Value 4
4	Value 4_1
5	Value 5
6	Value 6

Lateral Join

SELECT <target list>
FROM <table>
JOIN LATERAL
(<subquery using table.column>) as foo;

Джойн с зависимым подзапросом.
Соединение с подзапросом с использованием поля из предыдущей таблицы
https://www.db-fiddle.com/f/u7jr2CtYVQ4djmvFEgfAkj/2

Доп материал:
https://www.heap.io/blog/postgresqls-powerful-new-join-type-lateral

Множества

Множества

Под множеством в Постгресе понимается результат запроса - совокупность результирующих строк.

Между такими множествами мы можем проводить различные типы операций, а именно:

- ❖ объединение (UNION)
- ❖ пересечение (INTERSECT)
- ❖ вычитание (EXCEPT)

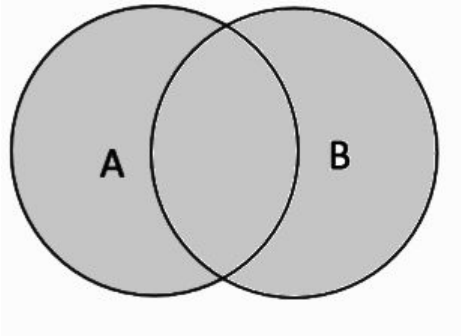
Ограничений на количество таких последовательных операций над множествами нет.

Также есть возможность оставлять дубликаты или убирать их, для этого есть специальное слово ALL. При его отсутствии при завершении очередной операции будут убраны дубликаты (DISTINCT).

https://www.postgresql.org/docs/current/queries-union.html

Множества. Объединение

SELECT <target list>
FROM <table>
UNION [ALL]
SELECT <target list>
FROM <table2>



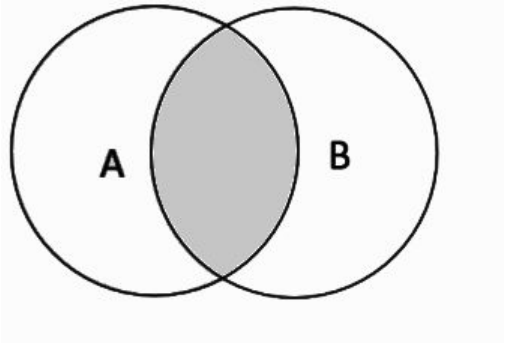
UNION

- Объединяет результаты двух запросов
- На выходе будут возвращены строки из всех множеств
- Можно исключать дубликаты (убрав ключевое слово ALL)

https://www.db-fiddle.com/f/hZQYzYduxdtoUymrxb6gDM/3

Множества. Пересечение

```
SELECT <target list>
FROM <table>
INTERSECT [ALL]
SELECT <target list>
FROM <table2>
```



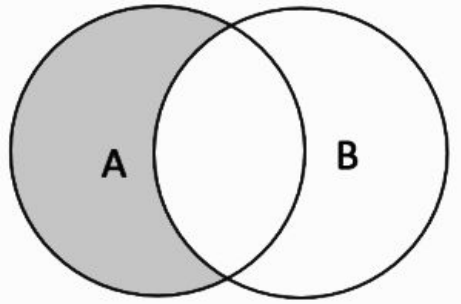
INTERSECT

- Пересекает результаты двух запросов
- На выходе будут только те строки, которые полностью совпали
- Можно исключать дубликаты (убрав ключевое слово ALL)

https://www.db-fiddle.com/f/niD6i5yCBfbKiVFfNQwyqk/2

Множества. Исключение

```
SELECT <target list>
FROM <table>
EXCEPT [ALL]
SELECT <target list>
FROM <table2>
```



EXCEPT

- Исключает результаты второго запроса из первого
- На выходе будут только те строки, которые есть в первом множестве и нет во втором
- Можно исключать дубликаты (убрав ключевое слово ALL)

https://www.db-fiddle.com/f/oCVpfRa8K9LikbxpWbsoVr/1

Множества

Основные моменты, на которые стоит обратить внимание:

- очередность операций, как и комбинации с и без ALL не ограничена
- количество и тип полей последующих множеств должен совпадать
- можно задать имена полям
- сортировку осуществлятляем только в самый последний момент, иначе порядок будет негарантирован
- ALL убирает дубликаты, но не гарантирует сортировку

Подзапросы

Подзапрос в выборке колонок

```
1  SELECT ProductName,
2         Company,
3         Price,
4          (SELECT AVG(Price) FROM Products AS SubProds
5           WHERE SubProds.Company=Prods.Company)  AS AvgPrice
6  FROM Products AS Prods
7  WHERE Price >
8      (SELECT AVG(Price) FROM Products AS SubProds
9       WHERE SubProds.Company=Prods.Company);
```

Data Output | Explain | Messages | Query History

	productname character varying (30)	company character varying (20)	price numeric	avgprice numeric
1	iPhone X	Apple	66000	53000.000000000000
2	Galaxy S9	Samsung	56000	51000.000000000000

Подзапрос в условии

```
Select    select_list
From      table
Where     expr operator
                ( Select    select_list
                  From      table );
```

```sql
SELECT *
FROM Products
WHERE Price > (SELECT AVG(Price) FROM Products);
```

Data Output | Explain | Messages | Query History

	id integer	productname character varying (30)	company character varying (20)	productcount integer	price numeric
1	1	iPhone X	Apple	2	66000
2	2	iPhone 8	Apple	2	51000
3	4	Galaxy S9	Samsung	2	56000

Пример

Оператор SELECT во фразе FROM:

Получить список накладных на поступление товаров от организаций, расположенных на улице 'Свободы' с 1 января 2009г. по 30 апреля 2014г.

```
SELECT Nakl.*, o.OrgName, Address
FROM Nakl,     (SELECT * FROM Org) AS o
WHERE Nakl.Org_ID=o.Org_ID
  and Address like '%Свободы%'
  and Nakl.InOut='+'
  and (Dat BETWEEN '20090101' and '20140430')
ORDER BY o.OrgName,Nakl.Dat;
```

Пример

Пример иллюстрирует соединение таблицы с самой собой.

Получить список пар организаций, имеющих одинаковый адрес:

```
SELECT t1.OrgName AS Org1, t2.OrgName AS Org2
FROM Org t1, Org t2
WHERE t1.Address=t2.Address
  and t1.Org_ID<t2.Org_ID;
```

условие t1.Org_ID<t2.Org_ID добавлено, чтобы в результирующем множестве записей не появлялись пары вида (A,A) (A,B),(B,A).

Пример

Оператор select может появиться и во фразе WHERE.

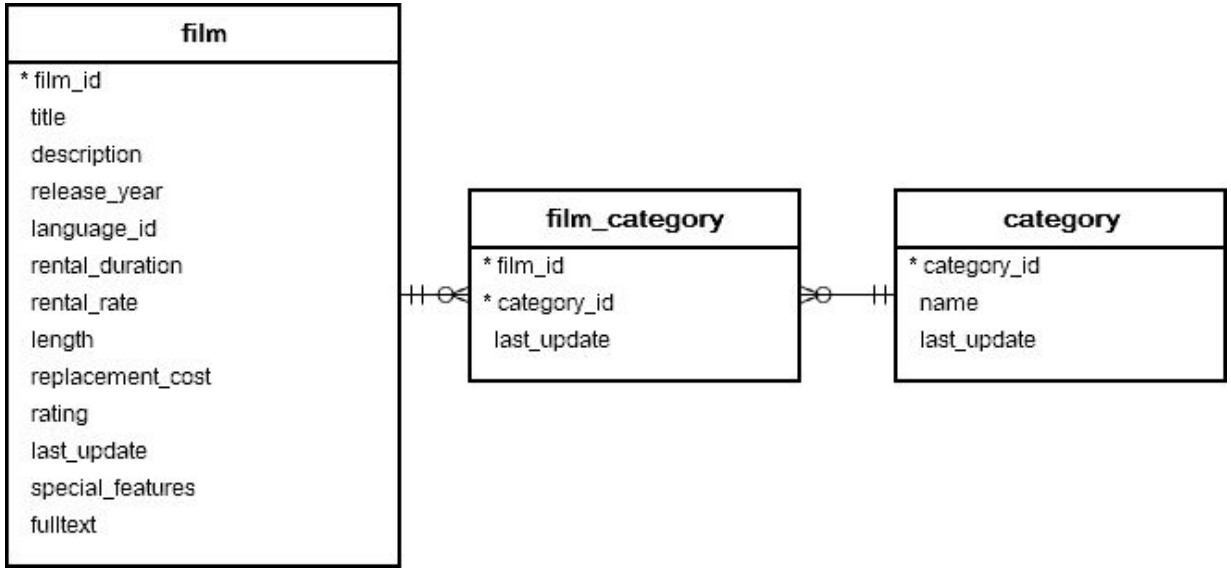
Получить данные о последней накладной для каждой организации.

```
select o.OrgName, n1.*
from Org o, Nakl n1
where o.Org_ID=n1.Org_ID
  and n1.Dat=
      (select max(Dat)
       from Nakl n2
       where n2.Org_ID=o.Org_ID)
```

Практика

- Получить информацию о фильме с самой низкой rental rate (ставкой проката)

  * Использовать функции MIN() и MAX() для поиска самых коротких и самых длинных фильмов по категориям



Практика (*)

Таблица emails содержит электронные письма, отправленные с адреса zach@g.com и полученные на него.

Задача: написать запрос, чтобы получить время отклика на каждое письмо (id), отправленное на zach@g.com. Не включать письма на другие адреса. Предположим, что у каждого треда уникальная тема.

Имейте в виду, что в треде может быть несколько писем туда и обратно между zach@g.com и другими адресатами.

ДЗ №2

https://github.com/DWH-course-examples/SQL-postgres/blob/main/homework/task2.md

СПАСИБО ЗА ВНИМАНИЕ!

#аис
#учисьваис