union & join и Агрегация

UNION/UNION ALL

Оператор union позволяет вертикально объединить данные из двух выборок.

```
-- шаблон
select
  <column1_name>,
  <column2_name>,
  <column3_name>
from table_1
Union
select
  <column1_name>,
  <column2_name>,
  <column3_name>
from table_2
```

Есть ряд особенностей при работе с UNION/UNION ALL:

1. Значения объединяются позиционно, не по названию поля
2. Типы данных должны быть совместимы
3. Кол-во полей должно совпадать
4. Названия полей и типы данных применяется из первой таблицы
5. Union убирает дубли, Union all подтягивает все данные

Обращаю ваше внимание, что Union в вопросе отсеивания дублей работает подобно distinct. Для определения дублей используется сортировка, а это достаточно тяжелый процесс (статейка про расчет сложности алгоритма).

Практическое задание

```
-- создать две таблицы и заполнить данными

create table goods(
  id integer primary key,
```

```
  title varchar(128),
  quantity integer check(quantity between 0 and 10)
);

insert into goods (id, title, quantity) values(1, 'велосипед', 4);
insert into goods (id, title, quantity) values(2, 'лыжи',  5);
insert into goods (id, title, quantity) values(3, 'коньки',  7);
insert into goods (id, title, quantity) values(4, 'скейт', 2);

create table goods_1(
  id integer primary key,
  title varchar(128),
  price integer,
  quantity integer check(price between 0 and 10)
);

insert into goods_1 (id, title, price, quantity) values(1, 'велосипед', 12000, 4);
insert into goods_1 (id, title, price, quantity) values(2, 'лыжи', 10000, 5);
insert into goods_1 (id, title, price, quantity) values(3, 'коньки', 6000, 7);
insert into goods_1 (id, title, price, quantity) values(4, 'скейт', 10000, 2);

-- объединить данные из goods и goods_1 (без дублей)

select
  id,
  title,
  quantity
from goods
union
select
  id,
  title,
  quantity
from goods_1

-- объединить данные из goods и goods_1 (все записи)

select
  id,
  title,
  quantity
from goods
union all
select
  id,
  title,
  quantity
from goods_1

-- объединить данные из goods и goods_1, указав price, где это возможно

select
  id,
  title,
  quantity,
  null as price
from goods
union all
```

```
select
  id,
  title,
  quantity,
  price
from goods_1
```

На данном занятии был разобран Join. Разные виды join операторов и их отличия.

Задача join

Join позволяет объединять данные из таблиц "Горизонтально". То есть указав правило, по которым мы хотите объединять строки, join их объединяет. Давайте рассмотрим синтаксис join на примере (в данном примере мы будем рассматривать inner join, другие типы мы разберем далее.

Представим, что у нас есть таблица **Names** с идентификатором (**ID**) и именем (**name**) пользователя. И таблица **Ages** с идентификатором (**ID**) и возрастом (**age**) пользователя.

Как вы могли заметить, в обоих таблицах присутствует поле ID которое однозначно определяет пользователя, по нему мы и будем производить join.

```
select
  t1.id,
  t1.name,
  t2.age
from names t1 -- t1 это алиас таблицы
inner join ages t2 -- t2 это алиас таблицы
on t1.id = t2.id -- условие join
```

В данном запросе мы получили выборку с id, name, age пользователей.

Обратите внимание, что в запросе мы используем алиасы, это позволяет нам сильно сократить запрос и сделать его более читаемым.

Условие join мы указываем после ключевого слова on.

Стоит разобраться, как у нас работает сам процесс соединения строк. В самом начале сопоставляются все строки из левой таблицы со всеми строками из

правой таблицы и в результирующую выборку попадут только те, которые удовлетворяют запросу.

Из этого следует, что если одной записи из из условной таблицы A соответствуют 2 записи из таблицы B, то в результирующей выборке будет 2 записи по этим строкам.

Существует несколько типов join, давайте их разберем.

Inner join

При inner join в результирующую выборку попадут только те записи, которые нашли себе пару по условию join (указанное после ключевого слова on).

Все остальные записи в результирующую выборку не попадут.

Left join

При left join в результирующую выборку попадут все записи из левой таблицы (название которой идет после from) и только те записи из правой, которые удовлетворили условию join.

Все остальные записи в результирующую выборку не попадут, а те записи из первой таблицы, которые не нашли пару из второй будут иметь значение NULL в полях второй таблицы.

> Это два наиболее часто используемых типа join

Right join

При right join в результирующую выборку попадут все записи из правой таблицы (название которой идет после join) и только те записи из левой, которые удовлетворили условию join.

Все остальные записи в результирующую выборку не попадут, а те записи из второй таблицы, которые не нашли пару из первой будут иметь значение NULL в полях первой таблицы.

> Обратите внимание, что left и right join очень похожи и взаимозаменяемы. Распространена практика использовать только left join.

Full join

При full join в результирующую выборку попадут все записи из правой и левой таблицы.

В тех записях, которые не удовлетворяют условию, в полях иной таблиц стоят null.

Full join похож на left и right join вместе.

Cross join

При cross join данные объединяются по принципу все со всеми. Так, если сделать cross join с таблицей в 3 строки и таблицей в 6 строк, то результат будет состоять из 18 строк.

Self join

Данный вид join позволяет соединять таблицу саму с собой. Наиболее частый пример, это таблицы таблица с сотрудниками, где так же указано поле boss_id с идентификатором босса, который так же является сотрудником.

Такой вид join позволяет хранить в таблице древовидную структуру.

▼ **Задание (на правило join)**

Какое максимальное и минимальное кол-во записей можно достичь используя JOIN

```
-- запрос
select
    t1.id
from table_1 t1
Inner join table_2 t2
on t1.id = t2.id
```

таблицы

Table_1

Aa ID
Untitled
???

Aa ID
???
???
???

Copy of Table_2

Aa ID
Untitled
???
???
???
???

Ответ

В случае, если все поля в двух таблицах будут отличаться, то под условие не подойдет ни одна пара записей, в таком случае inner join вернут пустую выборку.

В случае, если все записи будут одинаковыми, то будет cross join и кол-во записей будет 16.

▼ **Задание (практика)**

```
/*
Используя данные из схемы HR выведите имя и фамилию сотрудника
и название его департамента (hr.departments).
*/

select
    t1.FIRST_NAME,
    t1.LAST_NAME,
    t2.department_name
from hr.employees t1
left join hr.departments t2
on t1.department_id = t2.department_id;


/*
Выведите названия департаментов, где есть сотрудники с зп больше 15000
*/

select
```

```
    t1.FIRST_NAME,
    t1.LAST_NAME,
    t2.department_name
from hr.employees t1
left join hr.departments t2
on t1.department_id = t2.department_id
where t1.salary > 15000

/*
Напишите запрос, который покажет имя и фамилию сотрудников,
которые получают зп больше своего менеджера
*/

select
    t1.FIRST_NAME,
    t1.LAST_NAME
from hr.employees t1
inner join hr.employees t2
on t1.MANAGER_ID = t2.employee_id
and t1.salary > t2.salary
```

Агрегация

Агрегация позволяет произвести расчет значений полей одной из существующих функций. Агрегация может быть по всей таблице или по группа (используя group by).

Виды агрегаций

1. count() - кол-во не null строк
2. max() - максимальное значение
3. min() - минимальное значение
4. avg() - среднее значение
5. sum() - сумма значений

```
-- шаблон
select
  max(<column_name>),
  min(<column_name>),
  avg(<column_name>),
```

```
  sum(<column_name>)
from <table_name>
```

Важно указать, что при агрегации в select могут быть только те поля, которые либо присутствуют в группировке, либо обработаны агрегационной функцией

```
-- шаблон
select
  max(<column_name>),
  <column1_name> -- ОШИБКА!!!
from <table_name>
```

Задание

```
-- Найдите максимальную зарплату сотрудников из таблицы HR.EMPLOYEES

select
  max(salary) as max_salary
from HR.EMPLOYEES
```

Подзапросы

Подзапросы не являются неотъемлемой частью агрегации и могут использоваться без нее, однако они помогают реализовывать сложную логику в паре с агрегацией.

Подзапрос может использоваться для формирования источника данных в запросе.

```
-- шаблон

select
  <column_name>
from (
  select *
  from <table1_name>
) t1
```

Обратите внимание что в таких случаях указывать алиас у источника данных необходимо

Так же результат работы подзапроса может использоваться в условии

```
-- шаблон

select
  <column_name>
from <table_name>
where <column1_name> = (
  select
    max(<column_name>)
  from <table1_name>
)
```

Задание

```
-- Найдите имена и фамилии сотрудников с максимальной зарплатой HR.EMPLOYEES

select
    FIRST_NAME,
    LAST_NAME
from hr.employees
where salary = (select max(salary) from hr.employees )
```

Группировка

Группировка позволяет сформировать отдельные группы из строк и считать агрегацию в них. Группы формируются по различным значениям указанного поля или полей.

Давайте разберем шаблон

```
select
<column1_name>,
  max(<column_name>)
from <table_name>
group by <column1_name>
```

В данном запросе группы формируются по уникальным значениям column1_name и в них идет рассчет максимального значения column_name.

Задание

```
-- Найдите имена и фамилии сотрудников с максимальной зарплатой
-- в каждом департаменте HR.EMPLOYEES

select
    t1.FIRST_NAME,
    t2.LAST_NAME
from hr.employees t1
inner join (
    select
        department_id,
        max(salary) as max_salary
    from hr.employees
    group by department_id
) t2
on t1.department_id = t2.department_id
and t1.salary = t2.max_salary
```

Агрегация позволяет произвести расчет значений полей одной из существующих функций. Агрегация может быть по всей таблице или по группа (используя group by).