**Практическое занятие №7**

**Группировка данных и агрегатные функции**

**Цель занятия**. Научиться, выполнять группировку данных и вычислять агрегатные функции в операторе запроса в БД PostgreSQL.

Группировка результатов выборки из БД - это мощная функция SQL, которая позволяет вычислять ключевые статистические данные для группы записей (например, средние, максимальные и минимальные значения в группе записей).

Обычная выборка может вернуть только строки. С помощью функции группирования можно получать группы записей и вычислять показатели для каждой группы. В данной работе вы узнаете, как группировка делает SQL-запросы намного более мощными и разнообразными.

Но вначале рассмотрим агрегатные функции.

**Агрегатные функции**

Агрегатные функции (в стандарте SQL они называются функциями над множествами) используются для обобщения отдельных значений данных. Агрегатные функции вычисляют некоторое интегральное значение для заданного множества строк. Таким множеством строк может быть или вся таблица или группа строк, если агрегатная функция применяется к сгруппированной таблице.

При вычислении агрегатной функции по некоторой колонке не учитываются неопределенные значения (NULL). Если в агрегатной функции используется ключевое слово DISTINCT, то из процесса вычислений исключаются значения-дубликаты. Если агрегатная функция применяется без ключевого слова DISTINCT, то вычисление производится над всеми значениями.

В стандарте SQL определены пять агрегатных функций:

1) COUNT - функция подсчета количества значений. Применяется в двух форматах:

*COUNT(\*)* подсчитывает число строк в заданном множестве. Учитываются все строки - в том числе одинаковые и содержащие значение null.

*COUNT([DISTINCT] колонка****)*** подсчитывает число не Null-значений в заданном столбце (с DISTINCT – количество разных значений).

2) MAX(выражение) - определяет максимальное значение.

3) MIN(выражение) - определяет минимальное значение.

4) SUM(выражение) - вычисляет сумму значений.

5) AVG(выражение) - вычисляет среднее значение.

Последние 2 функции используются только для числовых значений.

Аргументом агрегатной функции может быть как отельная колонка, так и выражение

Например, SUM(KOL\*CENA)

*Пример 1.* Вывести общий объем всех поставок. Функция вычисляется по всей таблице:

SELECT SUM(kol) from kart

Результат запроса:



*Пример 2*. Определить количество строк, в которых заполнена колонка кода продукции

SELECT COUNT(prod\_kod) from kart

Результат запроса:



Пример 3. Определить количество разных наименований продукции, проданной покупателю с кодом 1

SELECT COUNT (DISTINCT prod\_kod)

FROM book b, kart k

WHERE b.book\_id=k.book\_id and pokup\_kod=1

Результат запроса:



***Упражнение 1.***

1) Определить максимальный объем продажи.

2) Определить среднюю цену продажи.

3) Определить количество покупателей в январе 2021 г.

**Раздел GROUP BY**

Позволяет сгруппировать данные по одной или нескольким колонкам и вычислить для каждой группы агрегатную функцию

**GROUP BY <список колонок>**

Результатом будет "***сгруппированная таблица***".

Важно отметить, что в список выборки оператора SELECT, содержащего раздел GROUP BY, можно включать *только* агрегатные функции и колонки, *которые входят в раздел группировки*.

*Пример 4*. 1)Получить суммарные продажи по продукции для покупателя с кодом 1

SELECT prod\_kod,sum(kol)

FROM book b, kart k

WHERE b.book\_id=k.book\_id

and pokup\_kod=1

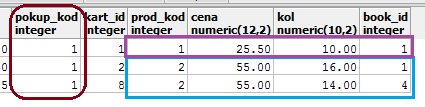
GROUP BY prod\_kod

Алгоритм выполнения запроса:

1) Будет выполнено прямое произведение таблиц book и kart.

2) К полученным строкам будет применен раздел WHERE, условия которого выполняют естественное соединение таблиц и отбирают строки (продажи) для покупателя с кодом 1.

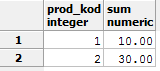
3) Полученные строки будут сгруппированы так, чтобы в каждую группу попадут строки с одинаковыми значениями в колонке prod\_kod.



4) В каждой группе будет просуммированы значения в колонке kol.

5) От каждой группы в результирующую таблицу будет включена одна строка.

Результат запроса:



*Пример 5.* Получить суммарные продажи в рублях по покупателям

SELECT sp.pokup\_name, sum(kol\*cena) as stoim

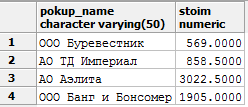
FROM spokup sp, book b, kart k

WHERE sp.pokup\_kod =b.pokup\_kod

and b.book\_id=k.book\_id

GROUP BY sp.pokup\_name

Результат запроса:



***Упражнение 2.***

1) Определить суммарные объемы продажи по каждой продукции

2) Определить количество разных наименований продукции по каждому покупателю.

**Раздел HAVING**

Условия отбора можно применять не только к строкам таблиц, но и к группам. Раздел HAVING содержит условное выражение, вычисляемое для каждой группы, определенной в разделе GROUP BY. Это условное выражение должно содержать агрегатные функции, вычисляемые для каждой группы.

Рассмотрим пример. Выбрать продукцию, для которой максимальный объем поставки больше 10.

Первый вариант запроса, который напрашивается будет таким:

SELECT prod\_kod FROM kart WHERE MAX(kol)>1000

Но данный оператор запроса не верен, т.к. условие WHERE вычисляется для каждой строки BOOK, т.е. до того как будет вычислена агрегатная функция, которая применима только к группе строк.

Для отбора групп вместо раздела WHERE используется раздел HAVING. Этот раздел задает условие отбора для сгруппированной таблицы.

**HAVING <условие отбора>**

В условии отбора раздела HAVING используют те же операции (предикаты), что и в условии отбора раздела WHERE. Но в условии отбора раздела HAVING колонки можно использовать в качестве аргумента агрегатных функций, вычисляющих в данном случае некоторое агрегатное значение для всей группы строк.

Результатом выполнения раздела HAVING будут те строки сгруппированной таблицы, для которых условия отбора дает true.

*Замечание.* В одном запросе могут встретиться как условия отбора строк в разделе WHERE, так и условия отбора групп в разделе HAVING. Разница в том, что условие раздела WHERE применяется к выражениям, которые можно вычислить по каждой строке, а в разделе HAVING - к агрегатным функция по группам строк.

Правильная запись оператора запроса для приведенного выше примера:

SELECT prod\_kod FROM kart

GROUP BY prod\_kod

HAVING MAX(KOL)>10

*Пример 6*. Получить суммарные поставки по видам продукции, для которых общий объем поставки превышает 20

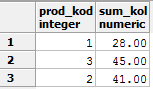
SELECT PROD\_KOD,SUM(KOL) as SUM\_KOL

FROM kart

GROUP BY prod\_kod

HAVING sum(kol)> 20

Результат запроса:



*Пример 7*.Получить наименования продукции, которая продавалась более одного раза

SELECT prod\_name, count(\*)

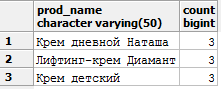
FROM sprod sp, kart k

WHERE k.prod\_kod =sp.prod\_kod

GROUP BY sp.prod\_name

HAVING count(k.prod\_kod) > 1

Результат запроса:



***Упражнение 3.***

1) Вывести покупателей, у которых общий объем поставок в стоимостном выражении превышает 1000

2) Вывести продукцию, у которой минимальная поставка превышает 10

**Аналитические функции**

Аналитические функции позволяют получить различные итоги и подитоги по набору строк, отобранных оператором SELECT.

Для рассмотрения аналитических функций добавим в БД таблицу.

Таблица вакансий (vacancies) по специальностям с указанием городов и стран содержит колонки:

vac\_id - идентификатор вакансии

country - страна

city – город

level - специальность

vacancy - количество вакансий

Код создания таблицы

CREATE TABLE prod.vacancies

(

vac\_id bigserial NOT NULL,

country character varying(20) NOT NULL,

city character varying(20) NOT NULL,

level character varying(20) NOT NULL,

vacancy integer NOT NULL,

CONSTRAINT vacancies\_pk PRIMARY KEY (vac\_id)

)

Добавим данные в таблицу:

insert into vacancies(country,city,level,vacancy)

values('Россия','Москва','программист',40),

('Россия','Санкт-Петербург','программист',25),

('Россия','Казань','программист',28),

('Россия','Томск','программист',18),

('Россия','Москва','аналитик',30),

('Россия','Санкт-Петербург','аналитик',15),

('Россия','Томск','аналитик',12),

('Россия','Москва','дизайнер',22),

('Россия','Санкт-Петербург','дизайнер',10),

('Россия','Казань','дизайнер',11),

('Белоруссия','Минск','программист',16),

('Белоруссия','Витебск','программист',9),

('Белоруссия','Брест','программист',8),

('Белоруссия','Минск','аналитик',11),

('Белоруссия','Брест','аналитик',6),

('Белоруссия','Минск','дизайнер',10),

('Белоруссия','Витебск','дизайнер',4)

В PostgreSQL имеются 3 аналитические функции:

* ROLLUP
* CUBE
* GROUPING SETS

Аналитическая функция **ROLLUP** наиболее проста в понимании, т.к. она просто формирует подитоги и общий итог по набору данных, отобранному оператором SELECT.

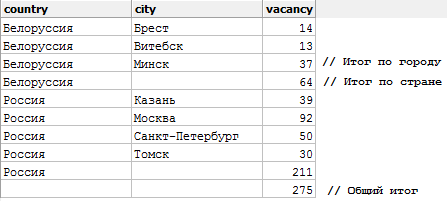
*Пример 8.* Сгруппировать данные по городам (city) и странам (country) и определить количество вакансий для каждого города, промежуточные итоги для каждой страны и общий итог по количеству вакансий.

SELECT country, city, sum(vacans) vacancy

FROM vacancies

GROUP BY ROLLUP (country, city);

Результат

****

Аналитическая функция **CUBE** вычисляет все возможные комбинации итогов по указанным колонкам группирования.

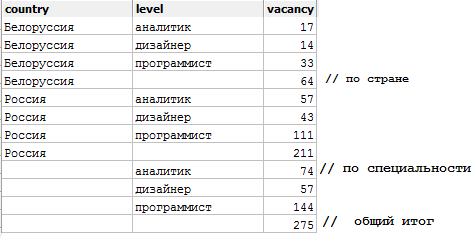
Пример 9. Сгруппировать данные вакансий по городам, специальностям и вывести итоги для комбинации страна – специальность, каждой страны (country), каждой специальности (level).

SELECT country, level, sum(vacans) vacancy

FROM vacancies

GROUP BY CUBE (country, level)

Результат



И, наконец, аналитическая функция **GROUPING SETS** является своеобразным родителем для ROLLUP и CUBE. Он позволяет задать вручную комбинации группирования и вывести по ним подитоги и общий итог.

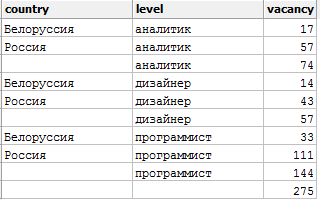
*Пример 10*. Необходимо сгруппировать данные и получить подитоги по стране+специальности, по стране, по специальности и общий итог:

SELECT country, level, sum(vacans) vacancy

FROM vacancies

GROUP BY GROUPING SETS ((country, level),(level),())

Результат:



***Упражнение 4*.** Получить итоги по городу, специальности и для пар город-специальность.

**Задание**

1. Выполните упражнения
2. Составьте похожие запросы для вашей БД