Факультет компьютерных наук

Введение в SQL
Простые SQL запросы

Тип данных – атрибут, определяющий, какого рода данные могут храниться в объекте: целые числа, символы, данные денежного типа, метки времени и даты, двоичные строки и так далее.

Тип данных	Псевдоним	Описание
bigint	int8	Целое число из 8 байт
integer	Int, int4	Целое число из 4 байт
numeric(p,s)	decimal(p,s)	Вещественное число заданной точности
real	float4	Число с плавающей точкой
boolean	bool	Логическое значение true/false
character(n)	char(n)	Символьная строка фиксированной длины
character varying n	varchar(n)	Символьная строка переменной длины
date	date	Календарная дата
timestamp	timestamp	Дата и время
timestamp with timezone	timestamptz	Дата и время с часовым поясом

Первичные и внешние ключи служат для идентификации строк и связи между собой записей разных таблиц. Внешний ключ ссылается на первичный.

Первичный ключ	Внешний ключ
Уникален	Может содержать дубли
Не содержит NULL	Может содержать NULL
Может являться внешним ключом	Может являться первичным ключом
Может состоять из нескольких колонок	Состоит из одной колонки
Строго один ключ на таблицу	Возможно наличие нескольких внешних ключей

Состав простого запроса

SELECT - оператор вывода (обязательный оператор)

FROM - оператор выбора таблиц

WHERE - оператор фильтрации

ORDER BY - оператор сортировки

Порядок выполнения простого запроса

(3) SELECT - оператор вывода (обязательный оператор)

(1) FROM - оператор выбора таблиц

(2) WHERE - оператор фильтрации

(4) ORDER BY - оператор сортировки

FROM

Выбирать данные для запроса можно как из 1, так и из нескольких таблиц. Отрабатывается первым.

Пример забора данных из 1 таблицы:

```
SELECT *
FROM table
```

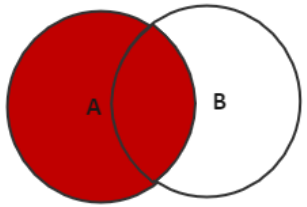
Однако выбор данных из нескольких таблиц уже не будет таким простым, так как появится вопрос - как совместить между собой данные из нескольких таблиц.

Для этого и существуют внешние ключи - они указывают на колонки, по содержанию совпадающие с первичными ключами других таблиц. С их участием делается JOIN.

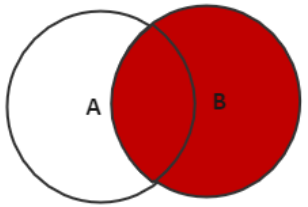
JOIN служит для соединения данных из нескольких таблиц. Отрабатывается во время чтения оператора FROM.

JOIN	Пример	Результат
CROSS JOIN	FROM table1 CROSS JOIN table2	Декартово произведение
INNER JOIN	FROM table1 INNER JOIN table2 ON table1.column = table2.column	Вывод только совпавших по ключу строк
LEFT JOIN	FROM table1 LEFT JOIN table2 ON table1.column = table2.column	Полный вывод записей левой таблицы и совпавших по ключу записей правой таблицы. В строках без совпадений выводится NULL.
RIGHT JOIN	FROM table1 RIGHT JOIN table2 ON table1.column = table2.column	Полный вывод записей правой таблицы и совпавших по ключу записей левой таблицы. В строках без совпадений выводится NULL.
FULL OUTER JOIN	FROM table1 FULL OUTER JOIN table2 ON table1.column = table2.column	Вывод всех записей правой и левой таблицы. В строках без совпадений выводится NULL.

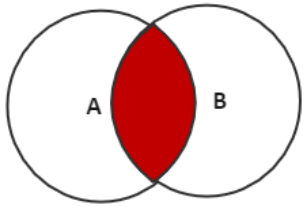
SQL JOINS



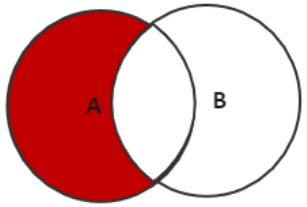

```
SELECT <selet_list>
FROM Table A A
LEFT JOIN TableB B
ON A.Key = B. Key
```



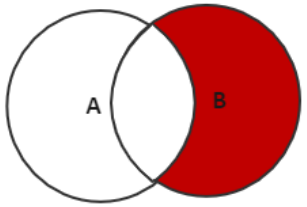

```
SELECT <select_list>
FROM Table A A
RIGHT JOIN TableB B
ON A.Key = B. Key
```



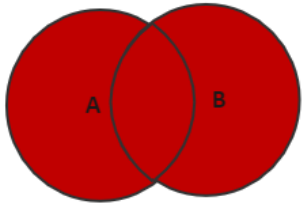

```
SELECT <select_list>
FROM Table A A
INNER JOIN TableB B
ON A.Key = B. Key
```



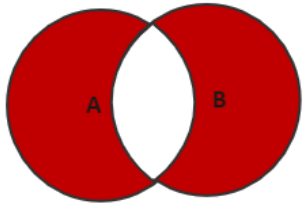

```
SELECT <select_list>
FROM Table A A
LEFT JOIN TableB B
ON A.Key = B. Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM Table A A
RIGHT JOIN TableB B
ON A.Key = B. Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM Table A A
FULL OUTER JOIN TableB B
ON A.Key = B. Key
```




```
SELECT <select_list>
FROM Table A A
FULL OUTER  JOIN TableB B
ON A.Key = B. Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

WHERE

Оператор WHERE позволяет задавать условия фильтрации строк из таблицы. Отрабатывается вторым.
Что можно использовать в WHERE?

- <> , < , > , <= , >= , =
- X between v1 and v2
- X in (1, 2, 3, ….)
- X in (table expression with one attribute)
- Is NULL
- Is NOT NULL

Пример:

```
SELECT *

FROM table

WHERE column between 1 and 5
```

SELECT

Оператор SELECT . Отрабатывается третьим.
Что можно использовать в SELECT?

- * - вывести все колонки
- Column1, column2, column3 - перечисление
- (23+43)/23%2, ‘a’, 4 - выражения и константы
- lower(column) - функции

Пример:

```
SELECT *, column2, coalesce(column3, column4, ‘t’) as column10
FROM table
WHERE column between 1 and 5
```

ORDER BY

Оператор ORDER BY позволяет отсортировать результат запроса. Отсортировать запрос можно двумя способами:

- ASC - ascending - от меньшего к большему
- DESC - descending - от большего к меньшему

Отрабатывается последним.

Пример:

```
SELECT *, column2, coalesce(column3, column4, ‘t’) as column10
FROM table
WHERE column between 1 and 5
ORDER BY column2 DESC
```

Полезные материалы

1) CTE - https://postgrespro.ru/docs/postgrespro/9.5/queries-with
2) Подзапросы - https://learn.microsoft.com/ru-ru/sql/relational-databases/performance/subqueries?view=sql-server-ver16
3) UNION / UNION ALL - https://metanit.com/sql/postgresql/6.5.php
4) LIKE / NOT LIKE - https://postgrespro.ru/docs/postgresql/9.6/functions-matching-
5) Функции даты и времени - https://postgrespro.ru/docs/postgrespro/10/functions-datetime
6) Строковые функции - https://postgrespro.ru/docs/postgrespro/9.5/functions-string

Факультет компьютерных наук

Сложные SQL запросы

Состав сложного запроса

SELECT - оператор вывода (обязательный оператор)

FROM - оператор выбора таблиц

WHERE - оператор фильтрации

GROUP BY - оператор группировки

HAVING - оператор фильтрации по значению агрегатов

ORDER BY - оператор сортировки

Порядок сложного запроса

(5) SELECT - оператор вывода (обязательный оператор)

(1) FROM - оператор выбора таблиц

(2) WHERE - оператор фильтрации

(3) GROUP BY - оператор группировки

(4) HAVING - оператор фильтрации по значению агрегатов

(6) ORDER BY - оператор сортировки

Функции агрегации и GROUP BY

Агрегатная функция выполняет вычисление над набором значений и возвращает одно значение. В табличной модели данных это значит, что функция берет ноль, одну или несколько строк и возвращает единственное значение для определенного набора разрезов - группировок, которые задаются в GROUP BY.

Функция	Физический смысл
SUM()	Возвращает сумму значений внутри группировки
COUNT()	Возвращает количество значений внутри группировки
MIN()	Возвращает минимальное значение внутри группировки
MAX()	Возвращает максимальной значение внутри группировки
AVG()	Возвращает средние значение внутри группировки

Функции агрегации и GROUP BY

SELECT department, gender, salary

FROM employees

department	gender	salary
Департамент аналитики	ж	1000
Департамент аналитики	м	2000
Департамент операций	м	1000
Департамент операций	м	2000
Департамент операций	м	1500
Департамент операций	ж	5000

Функции агрегации и GROUP BY

SELECT department, gender, AVG(salary), MIN(salary), MAX(salary), COUNT(salary), SUM(salary)

FROM employees

GROUP BY department, gender

department	gender	AVG	MIN	MAX	COUNT	SUM
Департамент аналитики	ж	1000	1000	1000	1	1000
Департамент аналитики	м	2000	2000	2000	1	2000
Департамент операций	м	1500	1000	2000	3	4500
Департамент операций	ж	5000	5000	5000	1	5000

Функции агрегации и GROUP BY

Важное правило: все, что перечислено в SELECT и не является выражением функции агрегации, должно быть перечислено в GROUP BY.

Пример:

Неверно

```
SELECT column1, column2, AVG(column3)
FROM table
GROUP BY column3
```

Верно

```
SELECT column1, column2, AVG(column3)
FROM table
GROUP BY column1, column2
```

HAVING

Этот оператор совершает фильтрацию по результату расчета агрегатной функции.

Например

```
SELECT column1, column2, AVG(column3)
FROM table
GROUP BY column1, column2
HAVING AVG(column3) > 3
```

Оконные функции

Оконная функция - функция, которая работает с выделенным набором строк (окном) и выполняет вычисление для этого набора строк в отдельном столбце.

Окно - это набор строк, указанный для оконной функции по одному из столбцов или группе столбцов таблицы.

Оконные функции схожи с агрегатными функциями, но на выходе они выдают изначальное количество строк, а также они нуждаются в уточнении окна вычисления, которое выглядит следующим образом:

FUNCTION(column) OVER (PARTITION BY … ORDER BY ...)

Оконные функции могут быть использованы в SELECT или ORDER BY.

Оконные функции

SELECT department, gender, salary

FROM employees

department	gender	salary
Департамент аналитики	ж	1000
Департамент аналитики	м	2000
Департамент операций	м	1000
Департамент операций	м	2000
Департамент операций	м	1500
Департамент операций	ж	5000

Оконные функции

```
SELECT
        department, gender, salary,
        AVG(salary) OVER (PARTITION BY department, gender),
        MIN(salary) OVER (PARTITION BY department, gender),
        MAX(salary ) OVER (PARTITION BY department, gender),
        COUNT(salary ) OVER (PARTITION BY department, gender),
        SUM(salary ) OVER (PARTITION BY department, gender)
FROM employees
```

Оконные функции

department	gender	salary	AVG	MIN	MAX	COUNT	SUM
Департамент аналитики	ж	1000	1000	1000	1000	1	1000
Департамент аналитики	м	2000	2000	2000	2000	1	2000
Департамент операций	м	1000	1500	1000	2000	3	4500
Департамент операций	м	2000	1500	1000	2000	3	4500
Департамент операций	м	1500	1500	1000	2000	3	4500
Департамент операций	ж	5000	5000	5000	5000	1	5000

Полезные материалы

1) Другие оконные функции - https://postgrespro.ru/docs/postgrespro/10/functions-window
2) Статья на habr по оконкам - https://habr.com/ru/articles/664000/
3) Статья по parquet - https://vc.ru/newtechaudit/531223-chto-takoe-parquet-i-zachem-on-prigoditsya


ВЫСШАЯ • ШКОЛА • ЭКОНОМИКИ •