Основы SQL: SELECT

Базы данных

Юдинцев В. В.

КАФЕДРА МАТЕМАТИЧЕСКИХ МЕТОДОВ В ЭКОНОМИКЕ

4 апреля 2023 г.

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Содержание

SELECT

SELECT

Оператор запроса в языке SQL, возвращающий набор данных (выборку) из базы данных.

```
SELECT
  <список столбцов>
FROM
  <список таблиц>
[WHERE <условие выбора строк>]
[GROUP BY <условие группировки>]
[HAVING <условие выбора групп>]
[ORDER BY <условие сортировки>]
```

SELECT

Все столбцы и строки из таблицы products

```
SELECT * FROM products
```

Все строки и выбранные столбцы из таблицы products

```
SELECT product_id, product_name FROM products
```

Выбрать все столбцы из таблицы products для product_id = 152

```
SELECT * FROM products WHERE product_id = 152
```

SELECT

Дополнительные условия

```
SELECT
  product_id, product_name
FROM
  products
WHERE
  product_id = 152 AND manufacturer_id = 12
```

Лоические операции с условиями

- AND
- OR
- NOT

Реляционные операторы

=	Равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
!=	Не равно

Пример

Исходная таблица

```
|store_id|store_name|
|--------|----------|
|       1|Московский|
|       2|Чкаловский|
|       3|Петровский|
|       4|Алабинский|
|       5|Филиал_А  |
```

Все строки с идентификатором store_id больше 2 и меньше 5

```sql
select * from stores where store_id > 2 AND store_id < 5;
```

Результат

```
|store_id|store_name|
|--------|----------|
|       3|Петровский|
|       4|Алабинский|
```

BETWEEN

- Для выбора значений из интервала можно использовать оператор **BETWEEN**
- Все строки таблицы с идентификатором store_id в интервале от 2 до 5

```
select * from stores where store_id BETWEEN 2 AND 4;
```

Результат

```
|store_id|store_name|
|--------|----------|
|       2|Чкаловский|
|       3|Петровский|
|       4|Алабинский|
```

NOT

- Для отрицания условия используется оператор NOT
- Все строки таблицы с идентификатором store_id не в интервале от 2 до 5

```
select * from stores where store_id NOT BETWEEN 2 AND 4;
```

Результат

```
|store_id|store_name|
|--------|----------|
|       1|Московский|
|       5|Филиал_А  |
```

DISTINCT

Директива **DISTINCT** используется с **SELECT** для выбора только отличающихся значений.
Например, таблица студентов содержит столбец с датой рождения:

id	name	birthdate
1	Поповский	2000-10-20
2	Аничков	2001-12-05
3	Вениаминов	2001-07-12
4	Чеботарев	2001-02-10

Необходимо вывести список годов рождения студентов без повторений, т.е. результатом должен быть список 2000, 2001.

DISTINCT

При использовании простого SELECT и функции YEAR для извлечения номера года из даты результатом будет список годов рождения с повторениями

```
select YEAR(birthdate) as birthyear from student;
```

```
birthyear|
---------|
     2000|
     2001|
     2001|
     2001|
```

Чтобы исключить повторения необходимо использовать директиву **DISTINCT**

```
select DISTINCT YEAR(birthdate) as birthyear from student;
```

```
birthyear|
---------|
     2000|
     2001|
```

Функции для работы с датами

https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html

- MONTH()
- YEAR()
- DAY()
- NOW()
- WEEKDAY()
- QUARTER()
- DATE_ADD()
- DATEDIFF()
- ...

Сравнение строк

Для сравнения строк используются операторы

- LIKE
- NOT LIKE
- STRCMP
- MATCH – полнотекстовый поиск для больших объёмов текста

Оператор LIKE

Таблица stores

store_id	store_name
1	Московский
2	Чкаловский
3	Петровский
4	Алабинский

Выбор строк с именами, содержащими сочетание 'вск';

```
select * from stores where store_name LIKE '%вск%';
```

store_id	store_name
1	Московский
2	Чкаловский
3	Петровский

Шаблоны LIKE

- `LIKE '%вск%'`
  % – любое количество символов (или отсутствие символов):
  'Чкаловский', 'вскряк!', 'Ижевск', 'вск'
- `LIKE '_ом'`
  _ – только один любой символ: 'сом', 'дом', но не 'облом'
- Чтобы использовать эти символы в строках для сравнения именно как символы, а не как шаблоны их нужно экранировать символом \
  `LIKE '%\_ом'`

Пример LIKE

Выбор из таблицы

```
|store_id|store_name|
|--------|----------|
|       1|Московский|
|       2|Чкаловский|
|       3|Петровский|
|       4|Алабинский|
```

строк с именами, заканчивающимися на ’нский’:

```
select * from stores where store_name LIKE('%нский');
```

```
|store_id|store_name|
|--------|----------|
|       4|Алабинский|
```

Оператор NOT LIKE

Таблица stores

```
|store_id|store_name|
|--------|----------|
|       1|Московский|
|       2|Чкаловский|
|       3|Петровский|
|       4|Алабинский|
```

Выбор строк с именами, НЕ содержащими сочетание ’вск’;

```
select * from stores where store_name NOT LIKE '%вск%';
```

```
|store_id|store_name|
|--------|----------|
|       4|Алабинский|
```

STRCMP

Результат функции `STRCMP(str1,str2)`

-1	str1 меньше str2
0	str1 и str2 равны
+1	str1 больше str2

Примеры STRCMP

Строка ТекстБ больше строки ТекстА

```
select STRCMP('ТекстА','ТекстБ');

|STRCMP('ТекстА','ТекстБ')|
|-------------------------|
|                       -1|
```

Строка ТекстА больше строки Текст

```
select STRCMP('ТекстА','Текст');

|STRCMP('ТекстА', 'Текст')|
|-------------------------|
|                        1|
```

STRCMP и регистр букв

Строки сравниваются без учёта регистра

```
select STRCMP('ТЕКСТ','Текст');

|STRCMP('ТЕКСТ', 'Текст')|
|------------------------|
|                       0|
```

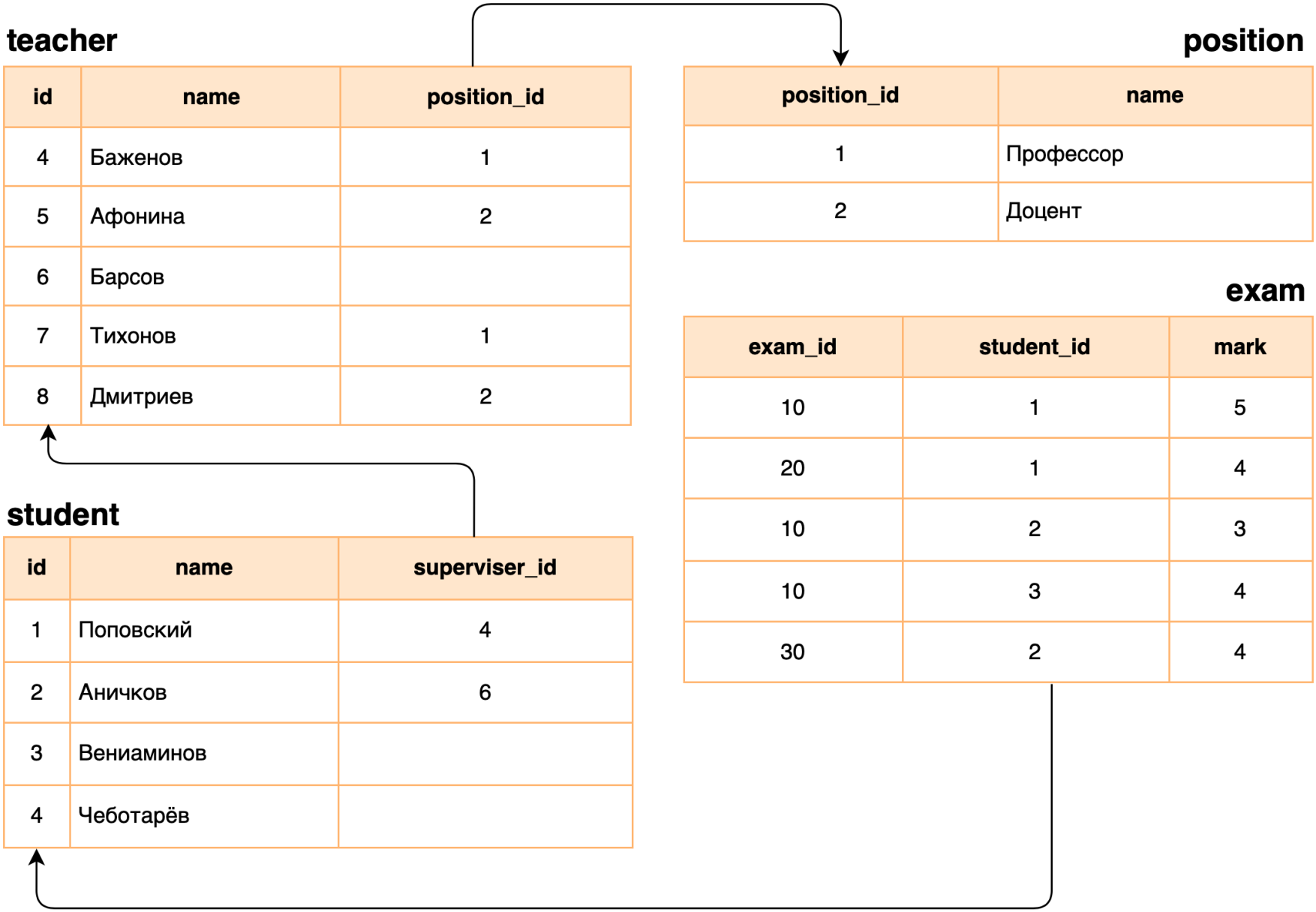
```
select * from viewer where FNAME LIKE 'a%';

|id|FNAME   |LNAME  |MNAME      |gender|birthdate  |
|--|--------|-------|-----------|------|-----------|
| 1|Андрей  |Петров|Викторович|M     |1981-11-21|
```

Это поведение зависит от типа поля в котором хранятся строки.

Соединение таблиц

Пример базы данных



teacher

id	name	position_id
4	Баженов	1
5	Афонина	2
6	Барсов	
7	Тихонов	1
8	Дмитриев	2

position

position_id	name
1	Профессор
2	Доцент

exam

exam_id	student_id	mark
10	1	5
20	1	4
10	2	3
10	3	4
30	2	4

student

id	name	superviser_id
1	Поповский	4
2	Аничков	6
3	Вениаминов	
4	Чеботарёв	

UNION

Объединение (UNION) таблиц

```
SELECT id, name from student
UNION
SELECT id, name from teacher;
```

Преподаватели и студенты

```
 id|name       |
---|-----------|
  1|Поповский  |
  2|Аничков    |
  3|Вениаминов|
  4|Чеботарев  |
  4|Баженов    |
  5|Афонин     |
  6|Барсов     |
```

Количество столбцов должно быть одинаково. Имена столбцов определяются первой таблицей в запросе.

INNER JOIN

- Оператор **INNER JOIN** формирует таблицу из записей двух или нескольких таблиц
- Каждая строка из первой (левой) таблицы, сопоставляется с каждой строкой из второй (правой) таблицы
- Если условие **ON** для строки истинно, то строка попадают в результирующую таблицу

INNER JOIN

Список студентов, у которых есть руководители

```
SELECT
  student.name, teacher.name as superviser_name
FROM
  student
INNER JOIN teacher ON teacher.id = student.superviser_id
```

Результат

```
name      |superviser_name|
----------|---------------|
Поповский|Баженов         |
Аничков   |Барсов          |
```

INNER JOIN

При отсутствии условия **ON** список будет большой (все сочетания строк)

```
name         superviser_name
----------   ----------------
Поповский    Баженов
Аничков      Баженов
Вениаминов   Баженов
Чеботарев    Баженов
Поповский    Афонин
Аничков      Афонин
Вениаминов   Афонин
Чеботарев    Афонин
Поповский    Барсов
Аничков      Барсов
...
Вениаминов   Тихонов
Чеботарев    Тихонов
Поповский    Дмитриев
Аничков      Дмитриев
Вениаминов   Дмитриев
Чеботарев    Дмитриев
```

Запрос к двум табицам

Список студентов, у которых есть руководители, можно получить запросом к двум таблицам (FROM) с условием WHERE:

```
SELECT
  student.name, teacher.name as superviser_name
FROM
  student, tecaher
WHERE
  teacher.id = student.superviser_id
```

Результат

```
name      |superviser_name|
----------|---------------|
Поповский|Баженов         |
Аничков   |Барсов          |
```

Вариант с INNER JOIN предпочтительней

Запрос к трём таблицам

Студенты, у которые сдали хотя бы один экзамен и есть руководитель

```
SELECT
  student.name, exam.exam_id, exam.mark, teacher.name
FROM
  student
INNER JOIN exam ON exam.student_id = student.id
INNER JOIN teacher ON teacher.id = student.superviser_id
```

name	exam_id	mark	name
Поповский	10	5	Баженов
Поповский	20	4	Баженов
Аничков	10	3	Барсов
Аничков	30	4	Барсов

Запрос к трём таблицам

Тот же результат без использования **INNER JOIN**

```
SELECT
  student.name, exam.exam_id, exam.mark, teacher.name
FROM
  student, exam, teacher
WHERE
  exam.student_id = student.id
AND
  teacher.id = student.superviser_id
```

name	exam_id	mark	name
Поповский	10	5	Баженов
Поповский	20	4	Баженов
Аничков	10	3	Барсов
Аничков	30	4	Барсов

Псевдонимы имён таблиц

В директиве FROM после имени таблицы может быть задан псевдоним таблицы. Например, можно дать таблицам короткие имена:

```
SELECT
  s.name, e.exam_id, e.mark, t.name as teacher_name
FROM
  student s, exam e, teacher t
WHERE
  e.student_id = s.id
AND
  t.id = s.superviser_id;
```

Показанные здесь псевдонимы, сотоящие из одной буквы, усложняют чтение запроса и приведены только для примера.

LEFT JOIN

```
SELECT
  a, b
FROM
  A
LEFT JOIN B
  ON A.f = B.f
```

- Строки из A (левая таблица), которым соответсвуют строки в таблице B
- Строки из A и строки из таблицы B, заполненные значениями NULL, если строкам из таблицы A не соответсвуют знаечния из таблицы B (условие ON не выпоняется).

LEFT JOIN

Левая таблица – teacher, правая – position

```
SELECT
 id, teacher.name, position.name as position_name
FROM
 teacher
LEFT JOIN position ON
 teacher.position_id = position.position_id
```

Для Барсова отсутствует запись в таблице position (правая таблица запроса), поэтому position_name для этой записи равно пустое (NULL)

```
id|name     |position_name|
--|---------|-------------|
 4|Баженов  |Профессор    |
 7|Тихонов  |Профессор    |
 5|Афонин   |Доцент       |
 8|Дмитриев|Доцент       |
 6|Барсов   |             |
```

LEFT JOIN

Если поменять порядок таблиц

```
SELECT
  id, teacher.name, position.name as position_name
FROM
  position
LEFT JOIN
  teacher ON teacher.position_id = position.position_id
```

то результатом будет список только тех преподавателей, у которых есть запись о должности:

```
id|name     |position_name|
--|---------|-------------|
 4|Баженов  |Профессор    |
 7|Тихонов  |Профессор    |
 5|Афонин   |Доцент       |
 8|Дмитриев|Доцент       |
```

RIGHT JOIN

Тот же результат получится, если в исходном запросе использовать RIGHT JOIN

```
SELECT
 id, teacher.name, position.name as position_name
FROM
 teacher
RIGHT JOIN position ON
 teacher.position_id = position.position_id
```

Результатом будет список только тех преподавателей, у которых есть запись о должности:

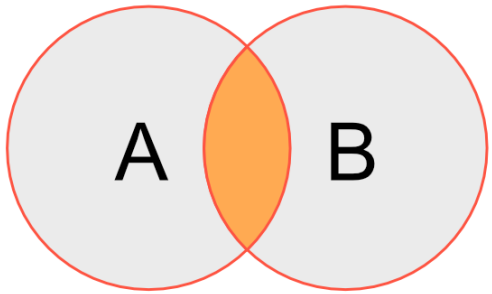
```
id|name     |position_name|
--|---------|-------------|
 4|Баженов  |Профессор    |
 7|Тихонов  |Профессор    |
 5|Афонин   |Доцент       |
 8|Дмитриев|Доцент       |
```

Соединения и операции с множествами

Пересечение

A.id	B.id
1	3
2	4
3	5
4	7
5	8

```
SELECT A.id, B.id
FROM A
INNER JOIN B ON A.id = B.id
```



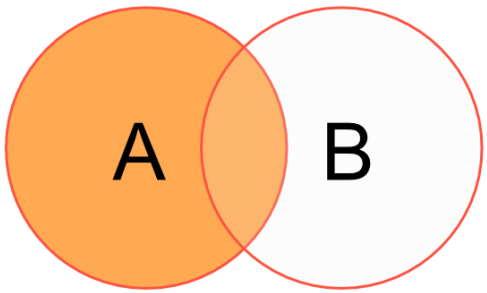
A.id	B.id
3	3
4	4
5	5

Соединения и операции с множествами

Пересечение

A.id	B.id
1	3
2	4
3	5
4	7
5	8

```
SELECT A.id, B.id
FROM A
LEFT JOIN B ON A.id = B.id
```

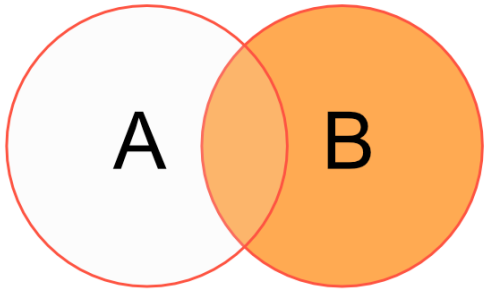


A.id	B.id
1	NULL
2	NULL
3	3
4	4
5	5

Соединения и операции с множествами

A.id	B.id
1	3
2	4
3	5
4	7
5	8

```
SELECT A.id, B.id
FROM A
RIGHT JOIN B ON A.id = B.id
```



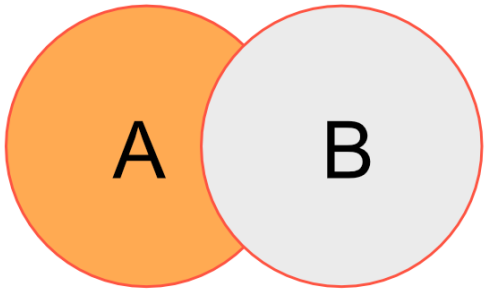
A.id	B.id
3	3
4	4
5	5
NULL	7
NULL	8

Соединения и операции с множествами

Разность

A.id	B.id
1	3
2	4
3	5
4	7
5	8

```
SELECT A.id, B.id
FROM A
LEFT JOIN B ON A.id = B.id
WHERE B.id IS NULL
```



A.id	B.id
1	NULL
2	NULL

Еще один пример

Информационная система склада

Магазин 1

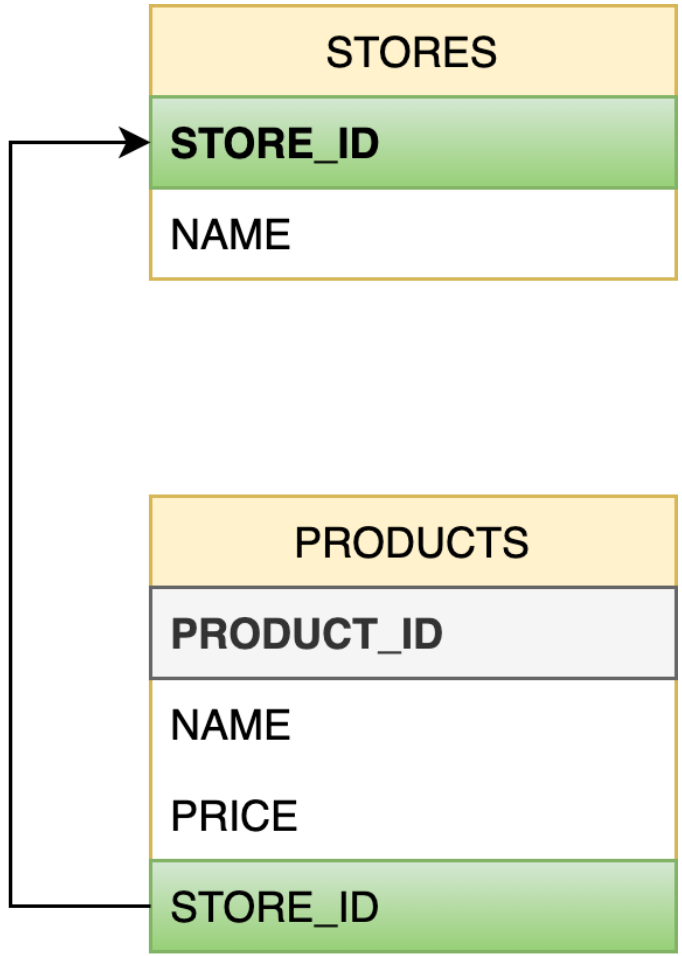
Товар	Цена
Чайник	900
...	...

Магазин 2

Товар	Цена
...	...
...	...

Магазин 3

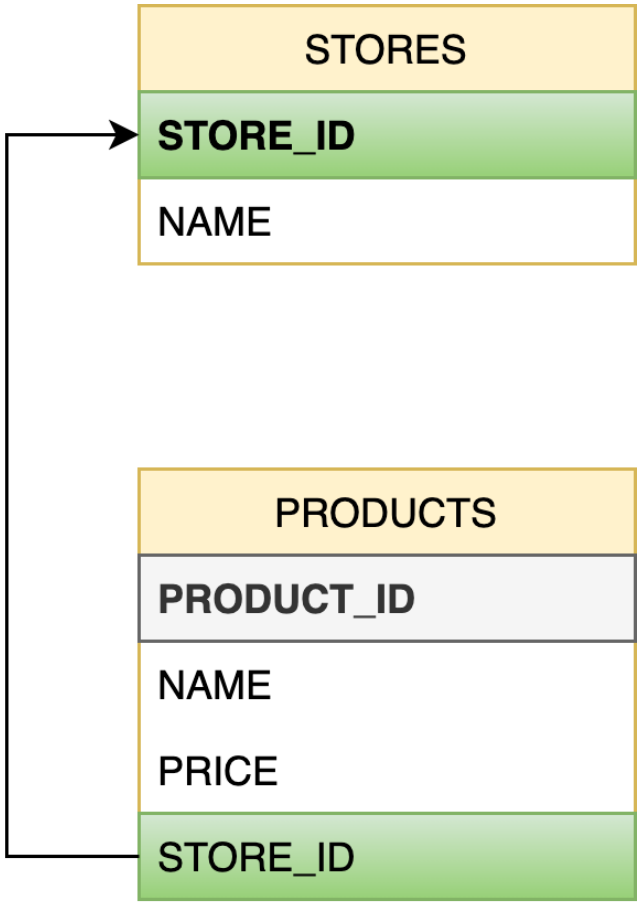
Товар	Цена
...	...
...	...



SQL DDL

```
CREATE TABLE STORES
(
  STORE_ID SERIAL PRIMARY KEY,
  NAME VARCHAR(100)
);

CREATE TABLE PRODUCTS
(
  PRODUCT_ID SERIAL PRIMARY KEY,
  NAME VARCHAR(100),
  PRICE NUMERIC(8,2),
  STORE_ID BIGINT UNSIGNED,
  FOREIGN KEY (STORE_ID)
↪   REFERENCES STORES(STORE_ID)
);
```



Заполняем данными?

```
INSERT INTO STORES (NAME) VALUES
('Космопорт'),
('Аврора'),
('Империя');

INSERT INTO PRODUCTS (STORE_ID, NAME, PRICE) VALUES
('Космопорт','Чайник',800.50),
('Космопорт','Телевизор',15640),
('Аврора','Принтер',10000),
('Империя','Ноутбук',50000);
```

Заполняем данными

```
INSERT INTO STORES (NAME) VALUES
('Космопорт'),
('Аврора'),
('Империя');

INSERT INTO PRODUCTS (STORE_ID, NAME, PRICE) VALUES
(
 (SELECT STORE_ID FROM STORES WHERE NAME='Космопорт'),
 'Чайник',800.50),
(
 (SELECT STORE_ID FROM STORES WHERE NAME='Космопорт'),
 'Телевизор',15640
),
(
 (SELECT STORE_ID FROM STORES WHERE NAME='Аврора'),
 'Принтер',10000
),
( (SELECT STORE_ID FROM STORES WHERE NAME='Империя'),
 'Ноутбук',50000
);
```

Товары в филиале

```
SELECT
  STORES.name,
  PRODUCTS.name,
  PRODUCTS.price
FROM
  PRODUCTS LEFT JOIN STORES ON PRODUCTS.STORE_ID = STORES.STORE_ID
WHERE
  STORES.NAME = 'Космопорт';
```

```
name      |name      |price   |
----------+----------+--------+
Космопорт|Чайник    |  800.50|
Космопорт|Телевизор|15640.00|
```

Товары, которых нет в филиале

```
SELECT
  STORES.name,
  PRODUCTS.name,
  PRODUCTS.price
FROM
  PRODUCTS LEFT JOIN STORES ON PRODUCTS.STORE_ID = STORES.STORE_ID
WHERE
  NOT STORES.NAME = 'Аврора';
```

```
 name      |name      |price   |
 ----------+----------+--------+
 Космопорт|Чайник    |  800.50|
 Космопорт|Телевизор|15640.00|
 Империя   |Ноутбук   |50000.00|
```

GROUP BY, HAVING и агрегатные функции

Пример

К таблицам (учителя, студенты, должности) добавим таблицу экзамены, которая содержит

- Идентификатор экзамена – **exam_id**
- Ссылку на студента, сдавшего экзамен – **student_id**
- Оценку – **mark**

Пример

teacher

id	name	position_id
4	Баженов	1
5	Афонина	2
6	Барсов	
7	Тихонов	1
8	Дмитриев	2

position

position_id	name
1	Профессор
2	Доцент

exam

exam_id	student_id	mark
10	1	5
20	1	4
10	2	3
10	3	4
30	2	4

student

id	name	superviser_id
1	Поповский	4
2	Аничков	6
3	Вениаминов	
4	Чеботарёв	

Студенты с оценками

Выбор из таблицы только тех, студентов, которые сдали хотя бы один экзамен:

```
SELECT
  student.name, exam.exam_id, exam.mark
FROM
  student
LEFT JOIN exam ON
  student.id = exam.student_id
WHERE
  exam.exam_id IS not null;
```

Результат выполнения запроса

name	exam_id	mark
Поповский	10	5
Поповский	20	4
Аничков	10	3
Вениаминов	10	4
Аничков	30	4

Агрегатные функции

- В первом столбце результата несколько раз повторяются имена студентов, которые сдали больше одного экзамена
- Можно сгруппировать эти повторяющиеся значения имени, но тогда к остальным полям нужно применить некоторые агрегирующие функции, чтобы превратить несколько занчений в одно, например найти среднее значение полученных оценок, максимальное значение, минимальное.
- Для таких запросов используется оператор GROUP BY и агрегатные функции.

name	exam_id	mark
Поповский	10	5
Поповский	20	4
Аничков	10	3
Вениаминов	10	4
Аничков	30	4

GROUP BY

- **GROUP BY** используется для объединения результатов выборки по одному или нескольким столбцам
- **GROUP BY** используется совместно с агрегатными функциями

Агрегатные функции

Агрегатной функцией в языке SQL называется функция, возвращающая какое-либо одно значение по набору значений столбца:

- COUNT
- SUM
- AVG
- MAX
- MIN

AVG()

Студенты, которые сдали хотя бы один экзамен с вычислением среднего балла полученных оценок:

```
SELECT
  student.name, AVG(exam.mark) as avg_mark
FROM
  student
LEFT JOIN exam ON
  student.id = exam.student_id
WHERE
  exam.exam_id IS not null
GROUP BY
  student.name;
```

Результат выполнения запроса

```
name       |avg_mark|
-----------|--------|
Аничков    |  3.5000|
Вениаминов|  4.0000|
Поповский  |  4.5000|
```

MIN() и MAX()

Те же студенты с выводом минимальной и максимальной оценки:

```
SELECT
  student.name,
  MIN(exam.mark) as min_mark, MAX(exam.mark) as max_mark
FROM
  student
LEFT JOIN exam ON student.id = exam.student_id
WHERE
  exam.exam_id IS not null
GROUP BY
  student.name;
```

Результат выполнения запроса

```
name       |min_mark|max_mark|
-----------|--------|--------|
Аничков    |       3|       4|
Вениаминов|       4|       4|
Поповский  |       4|       5|
```

SUM()

Суммарное количество баллов за все экзамены у студента. Из запроса исключаем условие WHERE, чтобы увидеть и тех студентов, которые ещё ничего не сдали

```
SELECT
  student.name,
  SUM(exam.mark) as sum_marks
FROM
  student
LEFT JOIN exam ON
  student.id = exam.student_id
GROUP BY
  student.name;
```

Результат выполнения запроса

```
name       |sum_marks|
-----------|---------|
Аничков    |        7|
Вениаминов |        4|
Поповский  |        9|
Чеботарев  |         |
```

COUNT()

При помощи функции COUNT можно определить количество экзаменов, которое сдал каждый студент.

```
SELECT
  student.name, COUNT(exam.mark) as n_exams
FROM
  student
LEFT JOIN exam ON
  student.id = exam.student_id
GROUP BY
  student.name;
```

Результат выполнения запроса

```
name       |n_exams|
-----------|-------|
Аничков    |      2|
Вениаминов |      1|
Поповский  |      2|
Чеботарев  |      0|
```

Сортировка

Результат можно отсортировать по заданному столбцу при помощи оператора ORDER BY

```
SELECT
  student.name, COUNT(exam.mark) as n_exams
FROM
  student
LEFT JOIN exam ON
  student.id = exam.student_id
GROUP BY
  student.name
ORDER BY n_exams;
```

По умолчанию сортировка выполняется по возрастанию указанного поля:

```
name       |n_exams|
-----------|-------|
Чеботарев  |      0|
Вениаминов |      1|
Поповский  |      2|
Аничков    |      2|
```

Сортировка

ORDER BY имя_поля DESC используется для сортировки по убыванию:

```
SELECT
  student.name, COUNT(exam.mark) as n_exams
FROM
  student
LEFT JOIN exam ON
  student.id = exam.student_id
GROUP BY
  student.name
ORDER BY n_exams DESC;
```

По умолчанию сортировка выполняется по возрастанию указанного поля:

```
name       |n_exams|
-----------|-------|
Поповский  |      2|
Аничков    |      2|
Вениаминов |      1|
Чеботарев  |      0|
```

Сортировка по нескольким полям

Сортировка может быть выполнена по нескольким столбцам (полям):

```
SELECT
  student.name, COUNT(exam.mark) as n_exams
FROM
  student
LEFT JOIN exam ON
  student.id = exam.student_id
GROUP BY
  student.name
ORDER BY n_exams DESC, student.name ASC;
```

По количеству сданных экзаменов, а затем по имени:

```
name       |n_exams|
-----------|-------|
Аничков    |      2|
Поповский  |      2|
Вениаминов |      1|
Чеботарев  |      0|
```

HAVING

Выражение **HAVING** позволяет отобрать по заданному условию результаты агрегатных функций. Студенты с суммой баллов больше 6:

```
SELECT
  student.name,
  SUM(exam.mark) as sum_marks
FROM
  student
LEFT JOIN exam ON
  student.id = exam.student_id
GROUP BY
  student.name
HAVING sum_marks>6;
```

Результат выполнения запроса

```
name      |sum_marks|
----------|---------|
Аничков   |        7|
Поповский |        9|
```

Ограничение вывода

LIMIT

- Для ограничения количество получаемых в результате запроса строк используется директива LIMIT в конце запроса. Первые десять строк:

```
SELECT * FROM users ORDER BY id LIMIT 10
```

- Десять строк с пропуском первых 5

```
SELECT * FROM users ORDER BY id LIMIT 5, 10
```

- Десять строк с пропуском первых 5

```
SELECT * FROM users ORDER BY id LIMIT 10 OFFSET 5
```

LIMIT

Первые два студента с максимальным средним баллом:

```
SELECT
  student.name, AVG(exam.mark) as avg_mark
FROM
  student
LEFT JOIN exam ON
  student.id = exam.student_id
WHERE
  exam.exam_id IS not null
GROUP BY
  student.name
ORDER BY avg_mark DESC
LIMIT 2;
```

Результат выполнения запроса:

```
name       |avg_mark|
-----------|--------|
Поповский  |  4.5000|
Вениаминов|  4.0000|
```

Подзапросы

Подзапросы

- SQL позволяет создавать вложенные запросы
- Вложенные запросы используются для вставки значения в таблицу, которое можно найти по другому значению атрибута
- Вложенные запросы позволяют вычислить значение, которое затем используется в объемлющем (верхнем) запросе

```
SELECT
 name as student_name
FROM
 student
WHERE
 superviser_id = (SELECT id FROM teacher WHERE name = 'Барсов');
```

Подзапросы

Имя студента, руководителем выпускной работы которого является преподаватель Барсов.
Подзапрос вычисляет идентификатор (id) преподавателя по его имени и полученное значение используется запросом верхнего уровня для поиска студента в таблице student с полем superviser_id, равным идентификатору Барсова:

```
SELECT
 name as student_name
FROM
 student
WHERE
 superviser_id = (SELECT id FROM teacher WHERE name = 'Барсов');
```

Список использованных источников

- Хомоненко А. Д., Цыганков В. М., Мальцев М. Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А. Д. Хомоненко. – 6-е изд., доп. - СПб.: КОРОНА-Век, 2009. – 736 с.
- Осипов Д. Л. Технологии проектирования баз данных. – М.: ДМК Пресс, 2019.
- SQL Учебник

  `https://schoolsw3.com/sql/index.php`
- SQL Tutorial

  `https://www.tutorialspoint.com/sql/index.htm`

https://classmech.ru/pages/databases/