Лекция 4: запросы и SQL

1. SQL

ИТМО ВТ

Запрос

Части запроса:

- **WHERE** — для определения, какие строки должны быть выбраны или включены в GROUP BY.
- **GROUP BY** — для объединения строк с общими значениями в элементы меньшего набора строк.
- **HAVING** — для определения, какие строки после GROUP BY должны быть выбраны.
- **ORDER BY** — сортировка результирующего набора данных.

SELECT поля FROM таблицы WHERE условия…

ИТМО ВТ

Логические операторы

- AND
- OR
- NOT

Состояния: true, false, NULL

AND и OR коммутативны: от перемены мест операндов результат не меняется

ИТМО ВТ

Операторы сравнения

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

- Операторы бинарные.
- Возвращают значения типа boolean.

Операторы и NULL

- Логические операторы: по таблицам истинности.
- Операторы сравнения: результат NULL, когда любой из операндов NULL.

ИТМО ВТ

Предикаты сравнения

a BETWEEN *x* AND *y* /* между */

a NOT BETWEEN *x* AND *y* /* не между */

a IS DISTINCT FROM *b* /* как <>, кроме NULL */

a IS NOT DISTINCT FROM *b* /*как =, кроме NULL */

выражение IS NULL /* является NULL */

выражение IS NOT NULL /* не является NULL */

ИТМО ВТ

Поиск по шаблону: LIKE

- *строка* LIKE *шаблон*
- *строка* NOT LIKE *шаблон*

Выражение LIKE возвращает true, если строка соответствует заданному шаблону.

```
'abc' LIKE '_b_'    /* true */
'abc' LIKE 'c'      /* false */
```

ILIKE — регистро-независимый поиск

ИТМО ВТ

Поиск по шаблону: SIMILAR TO

- *строка* SIMILAR TO *шаблон*
- *строка* NOT SIMILAR TO *шаблон*

Выражение SIMILAR TO возвращает true, если строка соответствует заданному шаблону.

Поддерживает больше видов регулярных выражений, чем LIKE

ИТМО ВТ

Регулярные выражения POSIX

- ~ /* Проверяет соответствие регулярному выражению с учётом регистра */
- ~* /* Проверяет соответствие регулярному выражению без учёта регистра */
- !~ /* Проверяет несоответствие регулярному выражению с учётом регистра */
- !~* /* Проверяет несоответствие регулярному выражению без учёта регистра */

ИТМО ВТ

Регулярные выражения POSIX

- Предоставляют более мощные средства поиска по шаблонам, чем операторы LIKE и SIMILAR TO

- Примеры:

```
'abcd' ~ 'abcd'        /* true */
'abcd' ~ '(b|d)'       /* true */
'abcd' ~ '^(b|c)'      /* false */
```

ИТМО ВТ

Условные выражения: CASE

if/then/else в других языках программирования:

CASE WHEN *условие* THEN *результат*

    [WHEN ...]

    [ELSE результат]

END

ИТМО ВТ

Условные выражения

1. CASE можно использовать везде, где допускаются выражения.

2. Каждое условие представляет собой выражение, возвращающее boolean.

3. Если не выполняются условия WHEN, значением CASE становится результат в ELSE.

4. Если в 3. предложение ELSE отсутствует, результатом будет NULL.

Условные выражения

```
SELECT Result FROM EXAM;
Result
------
 76
 95

SELECT Result,
       CASE WHEN Result>=91 THEN 'Отл.'
            WHEN Result>=75 THEN 'Хор.'
            WHEN Result>=60 THEN 'Удовл.'
            ELSE 'other result'
       END AS Description
    FROM EXAM;

 Result| Description
-------+-------------
    76  | Хор.
    95  | Отл.
```

ИТМО ВТ

Преобразования типов

- CAST (*expression* AS *type*) /* SQL-синтаксис */
- *expression*::*type* /* PostgreSQL */
- *typename* (*expression*) /* через функцию */

Пример:

```
SELECT CAST(42 AS float8);
/* через функцию float8(int4) */
```

```
CREATE CAST (source_type AS target_type)
    WITH FUNCTION function_name (argument_type [, ...])
```

ИТМО ВТ

IN

- *выражение* IN (*значение* [, …])

Результат «true», если значение *выражения* равняется одному из значений выражений в правой части:

(выражение = значение1) OR (выражение = значение2)

OR ...

2. Запросы с использованием нескольких таблиц

ИТМО ВТ

Декартово произведение таблиц

- Декартово произведение n таблиц — это таблица, содержащая все возможные строки s, такие, что s является сцеплением какой-либо строки из первой таблицы, строки из второй таблицы, ... и строки из n-й таблицы.
- Количество строк декартова произведения таблиц равно произведению количества строк соединяемых таблиц.

Декартово произведение таблиц

lt:

id	name
1	aaa
2	bbb
3	ccc

rt:

id	value
1	xxx
3	yyy
7	zzz

```
=> SELECT * FROM lt,rt;
```

id	name	id	value
1	aaa	1	xxx
1	aaa	3	yyy
1	aaa	7	zzz
2	bbb	1	xxx
2	bbb	3	yyy
2	bbb	7	zzz
3	ccc	1	xxx
3	ccc	3	yyy
3	ccc	7	zzz

Запросы с использованием нескольких таблиц

Вывести список всех студентов из России:

2 таблицы:

STUDENT(StudentID, Name, Surname, City)

CITIES(CityName, Country)

Соединение с помощью фразы WHERE

Вывести список всех студентов из России:

```
SELECT Name, City
FROM STUDENT, CITIES
WHERE CITIES.CityName = STUDENT.City AND
        CITIES.Country = 'Россия';
```

ИТМО ВТ

Запросы с использованием нескольких таблиц

1. СУБД формирует строки декартова произведения таблиц, перечисленных во фразе FROM.
2. СУБД проверяет, удовлетворяют ли данные в сформированной строке условиям из фразы WHERE.
3. Если данные в строке удовлетворяют условию из фразы WHERE, то СУБД включает в ответ на запрос те поля строки, которые соответствуют столбцам, перечисленным во фразе SELECT.

3. Соединения (Joins)

ИТМО ВТ

Соединение

В реляционной алгебре:

$R \bowtie_\theta S$ — **соединение:**

$$R \bowtie_\theta S = \sigma_\theta(R \times S)$$

ИТМО ВТ

Виды соединений

Тета-соединение:

$R \bowtie_{\theta} S$, где θ — предикат: R.attrx (лог. оп.) S.attry

Эквисоединение:

$R \bowtie_{\theta} S$, где в θ — лог. оп. =

Натуральное соединение:

$R \bowtie S$

ИТМО ВТ

Соединения

Начиная с SQL:1992 и для соединений таблиц принято использовать фразу JOIN.

- T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2 ON boolean_expression
- T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2 USING (join column list)
- T1 NATURAL { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2

ИТМО ВТ

Соединения (JOIN ON)

- В предложении с ON указываются выражения логического типа.
- Пара строк из таблиц 1 и 2 соответствуют друг другу, если выражение ON возвращает для них true.

```
SELECT Surname
FROM STUDENT
JOIN CITIES ON (CityName = City);
```

Позволяет сформировать **эквисоединение таблиц**

Соединения (JOIN USING)

- USING — сокращённая запись условия, когда с обеих сторон соединения столбцы имеют одинаковые имена. Например, запись соединения таблиц LT и RT с USING (**at1**, **at2**) формирует условие:

  ON **LT.at1** = **RT.at1** AND **LT.at2** = **RT.at2**

- При выводе JOIN USING исключаются избыточные столбцы.

```
SELECT Surname
FROM STUDENT
JOIN STUDENT_OLYMPIAD USING (StudentID);
```

ИТМО ВТ

Соединения (NATURAL JOIN)

NATURAL — упрощённая форма USING: образует список USING из всех имён столбцов, существующих в обеих входных таблицах.

Если столбцов с одинаковыми именами **нет**, NATURAL работает как CROSS JOIN.

```
SELECT Surname
FROM STUDENT
NATURAL JOIN STUDENT_OLYMPIAD;
```

ИТМО ВТ

Виды соединений

- INNER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN
- CROSS JOIN

INNER JOIN

Для каждой строки из исходной таблицы LT итоговая таблица включает строку для каждой строки из таблицы RT, если строка удовлетворяет условию соединения.

$$R \bowtie_{condition} S$$

Для каждой записи r в R:

Для каждой записи s в S:

Если condition == true, r+s - в результат.

INNER JOIN

lt:

```
 id  | name
-----+------
   1 | aaa
   2 | bbb
   3 | ccc
```

rt:

```
 id  | value
-----+-------
   1 | xxx
   3 | yyy
   7 | zzz
```

```
=> SELECT lt.id, name, value
   FROM lt INNER JOIN rt ON lt.id = rt.id;
lt.id| name | value
-----+------+-------
   1 |  aaa | xxx
   3 |  ccc | yyy
```

ИТМО ВТ

CROSS JOIN

Осуществляет полное перекрестное соединение двух таблиц.

R X S

Каждая строка первой таблицы соединяется со всеми строками второй таблицы, что создает результирующий набор — **декартово произведение таблиц**.

CROSS JOIN

lt:

id	name
1	aaa
2	bbb
3	ccc

rt:

id	value
1	xxx
3	yyy
7	zzz

```
=> SELECT * FROM lt CROSS JOIN rt;
```

id	name	id	value
1	aaa	1	xxx
1	aaa	3	yyy
1	aaa	7	zzz
2	bbb	1	xxx
2	bbb	3	yyy
2	bbb	7	zzz
3	ccc	1	xxx
3	ccc	3	yyy
3	ccc	7	zzz

ИТМО ВТ

LEFT OUTER JOIN

1. Сначала выполняется внутреннее соединение (INNER JOIN).
2. В результат добавляются все строки из таблицы LT, которым не соответствуют никакие строки из таблицы RT; вместо значений столбцов RT вставляются NULL.

В результирующей таблице всегда будет минимум одна строка для каждой строки из LT.

LEFT OUTER JOIN

lt:

id	name
1	aaa
2	bbb
3	ccc

rt:

id	value
1	xxx
3	yyy
7	zzz

```
=> SELECT lt.id, name, value FROM lt
   LEFT JOIN rt ON lt.id = rt.id;
 lt.id| name | value
------+------+-------
    1 | aaa  | xxx
    2 | bbb  |
    3 | ccc  | yyy
```

ИТМО ВТ

RIGHT OUTER JOIN

1. Сначала выполняется внутреннее соединение (INNER JOIN).
2. В результат добавляются все строки из таблицы RT, которым не соответствуют никакие строки из таблицы LT; вместо значений столбцов LT вставляются NULL.

Это соединение является обратным к левому (LEFT JOIN): в результирующей таблице всегда будет минимум одна строка для каждой строки из таблицы RT.

RIGHT OUTER JOIN

lt:

```
 id  | name
-----+------
   1 | aaa
   2 | bbb
   3 | ccc
```

rt:

```
 id  | value
-----+-------
   1 | xxx
   3 | yyy
   7 | zzz
```

```
=> SELECT name, rt.id, value FROM lt
   RIGHT JOIN rt ON lt.id = rt.id;
name | rt.id | value
-----+-------+-------
 aaa |   1   | xxx
 ccc |   3   | yyy
     |   7   | zzz
```

ИТМО ВТ

FULL OUTER JOIN

1. Сначала выполняется внутреннее соединение.
2. В результат добавляются все строки из таблицы LT, которым не соответствуют никакие строки из таблицы RT; вместо значений столбцов RT вставляются NULL.
3. В результат добавляются все строки из таблицы RT, которым не соответствуют никакие строки из таблицы LT; вместо значений столбцов LT вставляются NULL.

FULL OUTER JOIN

lt:

id	name
1	aaa
2	bbb
3	ccc

rt:

id	value
1	xxx
3	yyy
7	zzz

```
=> SELECT * FROM lt
FULL JOIN rt ON lt.id = rt.id;
 id  | name | id  | value
-----+------+-----+-------
   1 | aaa  |   1 | xxx
   2 | bbb  |     |
   3 | ccc  |   3 | yyy
     |      |   7 | zzz
```

ИТМО ВТ

Выполнение соединений (JOIN)

- Неэффективно: $R \bowtie_\theta S = \sigma_\theta(R \times S)$
- Реализации:
  - Nested Loop Join
  - Hash Join
  - Sort-merge Join

4. Вложенные подзапросы

ИТМО ВТ

Вложенный подзапрос

- Вложенный подзапрос (subquery) — предложение SELECT, которое заключено в круглые скобки и вложено в WHERE/HAVING часть другого SQL-предложения.
- Может содержать в своей WHERE (HAVING) - части другой вложенный подзапрос.
- Во внешнем и вложенном подзапросе может использоваться одна и та же таблица.

Простые вложенные подзапросы

ИТМО ВТ

- Простые вложенные подзапросы применяются для получения множества значений, которые проверяются в основном запросе:

```
SELECT Surname
FROM STUDENT
WHERE CityName IN (
        SELECT City FROM CITIES
        WHERE CITIES.Country = 'Россия'
);
```

Простые вложенные подзапросы

ИТМО ВТ

- Обрабатываются "снизу вверх": первым обрабатывается вложенный подзапрос самого нижнего уровня.

```
SELECT Surname
FROM STUDENT
WHERE CityName IN (
        SELECT City FROM CITIES
        WHERE CITIES.Country = 'Россия'
);
```

Получаем список городов в России, используем их для выполнения внешнего запроса

Простые вложенные подзапросы

ИТМО ВТ

- Обрабатываются "снизу вверх": первым обрабатывается вложенный подзапрос самого нижнего уровня.

```
SELECT Surname
FROM STUDENT
WHERE CityName IN (
        'Москва', 'Санкт-Петербург', ...
);
```

Простые вложенные подзапросы

ИТМО ВТ

То же можно получить с помощью соединения:

```
SELECT Surname
FROM STUDENT
WHERE CityName IN (
SELECT City FROM CITIES
WHERE
    CITIES.Country = 'Россия'
);
```

```
SELECT Surname
FROM STUDENT
JOIN CITIES ON CityName = City
WHERE CITIES.Country = 'Россия';
```

Коррелированные вложенные подзапросы

Вложенный подзапрос не может быть выполнен до обработки внешнего запроса:

```
SELECT Surname
FROM STUDENT
WHERE EXISTS (
        SELECT 1 FROM STUDENT_OLYMPIAD
        WHERE StID = STUDENT.StudentID
);
```

Коррелированные вложенные подзапросы

- Вложенный подзапрос зависит от значения **STUDENT.StudentID.**
- Это значение изменяется по мере того, как СУБД проверяет строки таблицы **STUDENT**.

```
SELECT Surname
FROM STUDENT
WHERE EXISTS (
        SELECT 1 FROM STUDENT_OLYMPIAD
        WHERE StID = STUDENT.StudentID
);
```

Коррелированные вложенные подзапросы

1. СУБД проверяет первую строку таблицы STUDENT (StudentID = 1). Значение STUDENT.StudentID в этот момент = 1. СУБД обрабатывает внутренний запрос, получая в результате 1, если такой олимпиадник есть:

```
SELECT Surname
FROM STUDENT
WHERE EXISTS (
        SELECT 1 FROM STUDENT_OLYMPIAD WHERE
        StID = STUDENT.StudentID
);
```

Коррелированные вложенные подзапросы

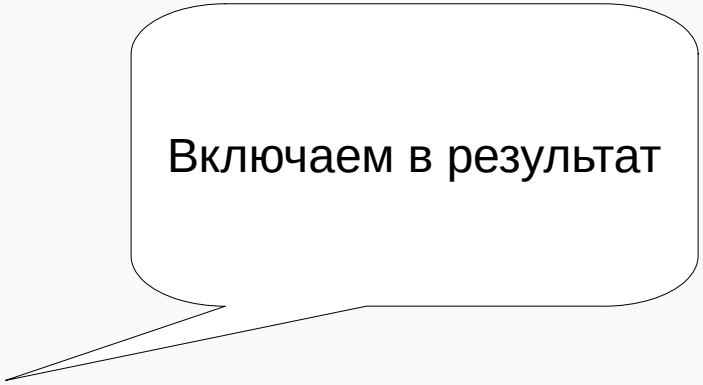
1. СУБД проверяет первую строку таблицы STUDENT (StudentID = 1). Значение STUDENT.StudentID в этот момент = 1. СУБД обрабатывает внутренний запрос, получая в результате 1, если такой олимпиадник есть:

```
SELECT Surname
FROM STUDENT
WHERE EXISTS (
        SELECT 1 FROM STUDENT_OLYMPIAD WHERE
        StID = 1
);
```

Коррелированные вложенные подзапросы

1. СУБД проверяет первую строку таблицы STUDENT (StudentID = 1). Значение STUDENT.StudentID в этот момент = 1. СУБД обрабатывает внутренний запрос, получая в результате 1, если такой олимпиадник есть:

```
SELECT Surname
FROM STUDENT
WHERE EXISTS (
        1
);
```



Коррелированные вложенные подзапросы

2. Система будет повторять обработку для каждой строки из таблицы STUDENT, пока не будут рассмотрены все строки таблицы STUDENT.

```
SELECT Surname
FROM STUDENT
WHERE EXISTS (
        SELECT 1 FROM STUDENT_OLYMPIAD
        WHERE StID = 2
);
```

Коррелированные вложенные подзапросы

- Обрабатываются СУБД в обратном порядке.
- Сначала выбирается первая строка таблицы, сформированной основным запросом. Из этой строки выбираются значения тех столбцов, которые используются во вложенном подзапросе.
- Если эти значения удовлетворяют условиям вложенного подзапроса, то выбранная строка включается в результат.

5. Операторы для работы со вложенными подзапросами

ИТМО ВТ

IN

выражение IN (*подзапрос*)

- *Подзапрос* должен возвращать ровно один столбец.
- Результат *выражения* сравнивается с каждым значением, возвращённым *подзапросом*.
- Результатом выражения IN будет «true», если значение *выражения* соответствует хотя бы одному значению, которое вернул подзапрос; «false» в противном случае *(включая случай, когда подзапрос не возвращает строк)*.

ИТМО ВТ

IN и NULL

выражение IN (*подзапрос*)

- Если *выражение*: NULL или

  соответствие в правой части не найдено, и есть хотя бы одно NULL-значение в результате *подзапроса*

  результат: NULL

ИТМО ВТ

EXISTS

EXISTS (*подзапрос*)

- EXISTS принимает оператор SELECT (*подзапрос)*
- Если *подзапрос* возвращает хотя бы одну строку, то результатом EXISTS будет «true», а если не возвращает ни одной — «false».
- Если *подзапрос* возвращает NULL, результат - «true»

ИТМО ВТ

EXISTS

Вывести фамилии студентов, попавших на комиссию?

STUDENT (StudentID, Surname, Name, GroupID)

EXAM (ExamID, StID, Result)

ИТМО ВТ

EXISTS

EXISTS (*подзапрос*)

```
SELECT Surname
FROM STUDENT
WHERE EXISTS (
        SELECT 1 FROM EXAM WHERE
          STUDENT.StudentID = EXAM.StID
          AND EXAM.Result < 60
);
```

ИТМО ВТ

ANY/SOME

выражение оператор ANY (*подзапрос*)

выражение оператор SOME (*подзапрос*)

- *Подзапрос* должен возвращать ровно один столбец
- Значение *выражения* сравнивается со значением в каждой строке результата *подзапроса* с помощью заданного *оператора*, который должен возвращать логическое значение.
- Результатом ANY будет «true», если хотя бы для одной строки условие истинно, и «false» в противном случае (в том числе, если подзапрос не возвращает строк).
- Оператор IN аналогичен = ANY (= SOME)

ИТМО ВТ

ANY, пример

```
SELECT *
    FROM STUDENT
        WHERE StudentId = ANY
            ( SELECT StID FROM EXAMS );
```

ИТМО ВТ

ANY/SOME и NULL

выражение оператор ANY (*подзапрос*)

выражение оператор SOME (*подзапрос*)

- Если соответствие в правой части не найдено, и
- Есть хотя бы одно NULL-значение после применения *оператора* к значению из результата *подзапроса*

  результат: NULL

ИТМО ВТ

ALL

выражение оператор ALL (*подзапрос*)

- *Подзапрос* должен возвращать ровно один столбец
- Значение *выражения* сравнивается со значением в каждой строке результата *подзапроса* с помощью *оператора*, который должен возвращать логическое значение.
- Результатом ALL будет «true», если условие истинно для всех строк (и когда *подзапрос* не возвращает строк), и «false», если находятся строки, для которых оно ложно.
- Результат будет NULL, если сравнение не возвращает «false» ни для одной из строк, но как минимум для одной результат сравнения при применении *оператора* NULL.

ИТМО ВТ

Литература

При подготовке презентации использовались материалы из:

- Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов, Издательство: BHV, 2009 г.
- Документация PostgreSQL.

  https://www.postgresql.org/about/licence/