

Использование подзапросов и извлечение данных с помощью подзапросов

В КОНЦЕ ЗАНЯТИЯ ВЫ ИЗУЧИТЕ :

- Определять подзапросы
- Описывать типы проблем, для решения которых можно использовать подзапросы
- Перечислять типы подзапросов
- Создавать однострочные и многострочные подзапросы
- писать подзапросы, содержащие несколько столбцов
- использовать скалярные подзапросы в SQL
- решать задачи с помощью коррелированных подзапросов
- использовать операторы EXISTS и NOT EXISTS
- использовать предложение WITH

Темы

- Подзапрос: типы, синтаксис и указания
- Однострочные подзапросы:
 - групповые функции в подзапросе
 - предложение HAVING с подзапросами
- Многострочные подзапросы
 - использование оператора ALL или ANY
- Значения NULL в подзапросе

Синтаксис подзапроса

SELECT *select_list*
FROM *table*
WHERE *expr operator*

(SELECT *select_list*
FROM *table*);

- Подзапрос (внутренний запрос) выполняется *перед* основным (внешним) запросом.
- Результат подзапроса используется основным запросом.

Используем подзапрос

```
select
first_name,
last_name,
start_date
from employee
where start_date > (select start_date from employee where first_name||' ' ||last_name = 'Susan Barker');
```

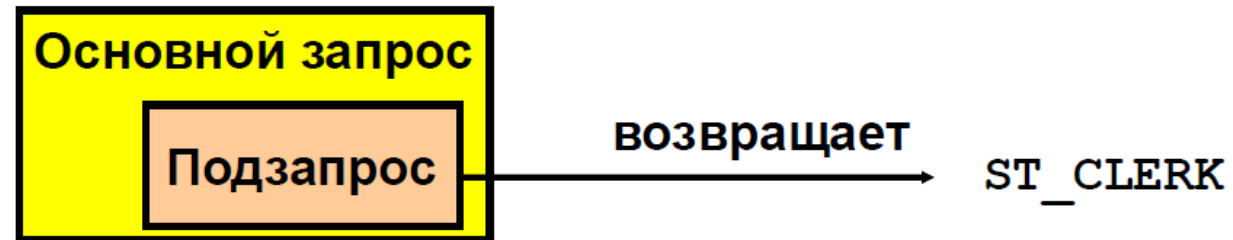
FIRST_NAME	LAST_NAME	START_DATE
John	Gooding	14-NOV-03
Helen	Fleming	17-MAR-04
Chris	Tucker	15-SEP-04
Sarah	Parker	02-DEC-02

Указания по использованию подзапросов

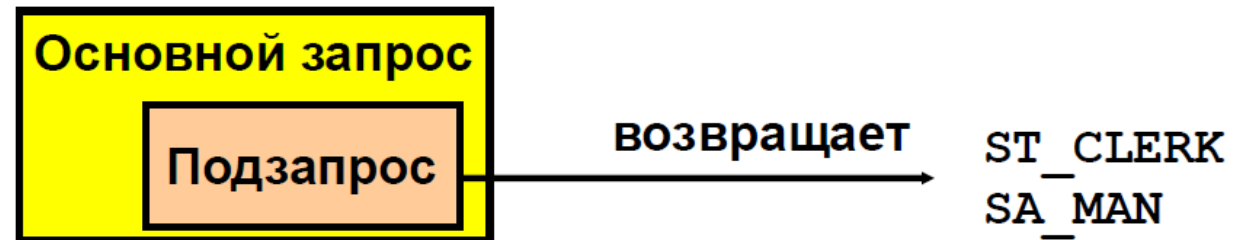
- Заключайте подзапросы в скобки.
- Для повышения наглядности кода размещайте подзапросы справа от условия сравнения (хотя в принципе запрос допустимо помещать с любой стороны от условия сравнения).
- Используйте с однострочными подзапросами однострочные операторы, а с многострочными подзапросами – многострочные операторы.

Типы подзапросов

- Однострочный подзапрос



- Многострочный подзапрос







Однострочные подзапросы

- Возвращают только одну строку
- Используют однострочные операторы сравнения

Оператор	Значение
=	Равно
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
<>	Не равно

Выполнение однострочных подзапросов

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id =  (SELECT job_id
FROM employees
WHERE last_name = 'Taylor')
AND salary >  (SELECT salary
FROM employees
WHERE last name = 'Taylor');
```


	 LAST_NAME	 JOB_ID	 SALARY
1	Abel	SA_REP	11000

Использование в подзапросах групповых функций

```
SELECT last_name, job_id, salary
FROM employees
WHERE salary =
```

2500

```
(SELECT MIN(salary)
FROM employees);
```



	LAST_NAME	JOB_ID	SALARY
1	Vargas	ST_CLERK	2500

Предложение HAVING в подзапросах

- Сервер Oracle выполняет подзапросы первыми.
- Сервер Oracle возвращает результаты в предложение HAVING основного запроса.

```
SELECT  department_id, MIN(salary)
FROM    employees
GROUP BY department_id
HAVING  MIN(salary) > (SELECT MIN(salary)
                       FROM    employees
                       WHERE    department_id = 50);
```

2500

	DEPARTMENT_ID	MIN(SALARY)
1	(null)	7000
2	90	17000
3	20	6000
...		
7	10	4400

Что неправильно в этой инструкции?

```
SELECT employee_id, last_name  
FROM employees  
WHERE salary =  
    (SELECT MIN(salary)  
     FROM employees  
     GROUP BY department_id);
```

Ошибка
Однострочный
оператор
с многострочным
подзапросом

Внутренний запрос не вернул ни одной строки

```
SELECT last_name, job_id
FROM employees
WHERE job_id =
    (SELECT job_id
     FROM employees
     WHERE last_name = 'Haas');
```

0 rows selected

Подзапрос не вернул ни одной строки,
поскольку нет сотрудника с фамилией «Haas».

Многострочные подзапросы

- Возвращают больше одной строки
- Используют многострочные операторы сравнения

Оператор	Значение
IN	Равен любому элементу из списка
ANY	Перед ним должны использоваться операторы =, !=, >, <, <=, >=. Сравнивает текущее значение с каждым значением, перечисленным в списке или возвращенным запросом. Получает значение FALSE, если запрос не вернул ни одной строки.
ALL	Перед ним должны использоваться операторы =, !=, >, <, <=, >=. Сравнивает текущее значение со всеми значениями, перечисленными в списке или возвращенными запросом. Получает значение TRUE, если запрос не вернул ни одной строки.

Использование оператора ANY в многострочных подзапросах

```
SELECT employee_id, last_name, job_id, salary
FROM   employees          9000, 6000, 4200
WHERE  salary < ANY
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	144	Vargas	ST_CLERK	2500
2	143	Matos	ST_CLERK	2600
3	142	Davies	ST_CLERK	3100
4	141	Rajs	ST_CLERK	3500
5	200	Whalen	AD_ASST	4400

...

9	206	Gietz	AC_ACCOUNT	8300
10	176	Taylor	SA_REP	8600

Использование оператора ALL в многострочных подзапросах

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary < ALL
      (SELECT salary
       FROM employees
       WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

9000, 6000, 4200

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	141	Rajs	ST_CLERK	3500
2	142	Davies	ST_CLERK	3100
3	143	Matos	ST_CLERK	2600
4	144	Vargas	ST_CLERK	2500

Значения NULL в подзапросе

```
SELECT emp.last_name  
FROM   employees emp  
WHERE  emp.employee_id NOT IN  
        (SELECT mgr.manager_id  
         FROM   employees mgr);
```

0 rows selected

Темы

- Написание подзапросов, содержащих несколько столбцов
- Использование скалярных подзапросов в SQL
- Решение задач с помощью коррелированных подзапросов
- Использование операторов EXISTS и NOT EXISTS
- Использование предложения WITH

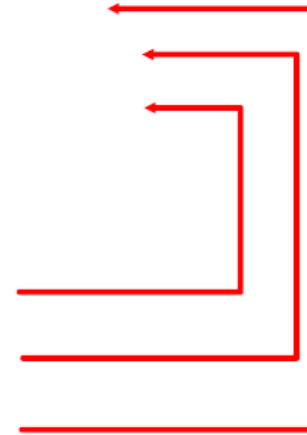
Подзапросы, содержащие несколько столбцов

Главный запрос

WHERE (MANAGER_ID, DEPARTMENT_ID) IN

Подзапрос

100	90
102	60
124	50



Каждая строка главного запроса сравнивается со значениями из подзапроса, содержащего несколько строк и несколько столбцов.

Сравнения столбцов

Сравнения нескольких столбцов с помощью подзапросов могут быть:

- непарными сравнениями;
- попарными сравнениями.

Подзапрос попарного сравнения

Выведите сведения о сотрудниках, руководимых тем же менеджером и работающих в том же отделе, что и сотрудники с именем John.

```
SELECT employee_id, manager_id, department_id
FROM   empl_demo
WHERE  (manager_id, department_id) IN
      (SELECT manager_id, department_id
       FROM empl_demo
       WHERE first_name = 'John')
AND first_name <> 'John';
```


Подзапрос непарного сравнения

Выведите сведения о сотрудниках, руководимых тем же менеджером, что и сотрудники с именем John, и работающих в том же отделе, что и сотрудники с именем John.

```
SELECT  employee_id, manager_id, department_id
FROM    empl_demo
WHERE    manager_id IN
        (SELECT manager_id
         FROM empl_demo
         WHERE first_name = 'John')
AND department_id IN
        (SELECT department_id
         FROM empl_demo
         WHERE first_name = 'John')
AND first_name <> 'John';
```

Примеры скалярных подзапросов

- Скалярные подзапросы в выражениях CASE:

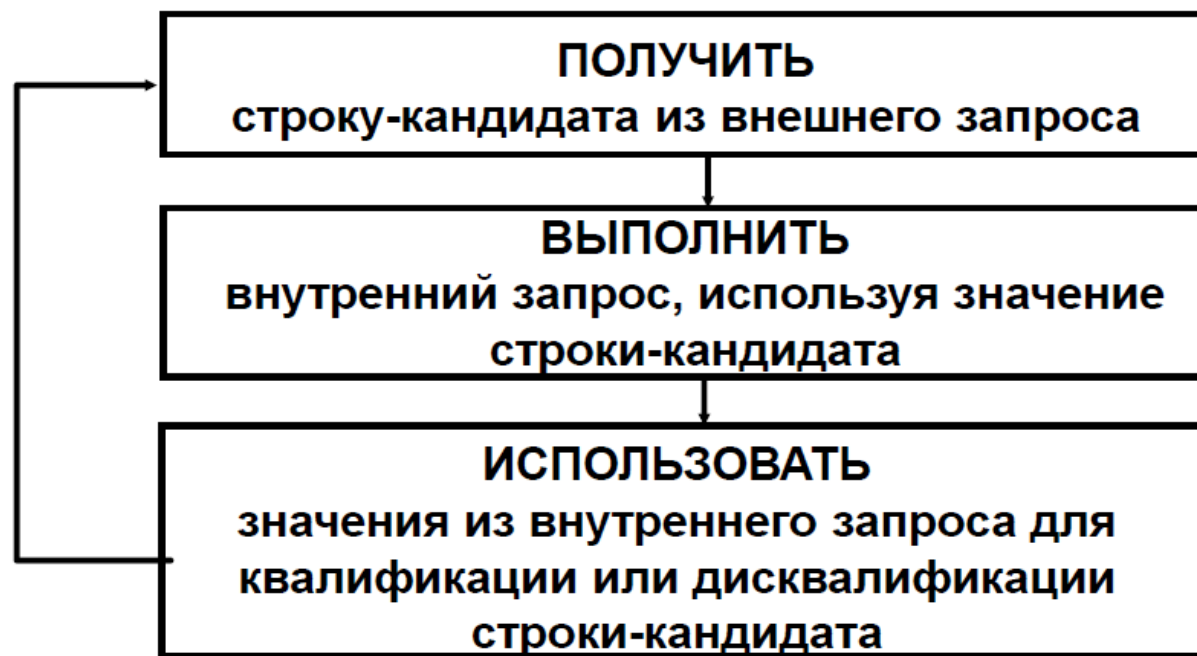
```
SELECT employee_id, last_name,  
       (CASE  
         WHEN department_id = 20  
           (SELECT department_id  
            FROM departments  
            WHERE location_id = 1800)  
         THEN 'Canada' ELSE 'USA' END) location  
FROM   employees;
```

- Скалярные подзапросы в предложении ORDER BY:

```
SELECT  employee_id, last_name  
FROM    employees e  
ORDER BY (SELECT department_name  
          FROM departments d  
          WHERE e.department_id = d.department_id);
```

Коррелированные подзапросы

Коррелированные подзапросы используются для строчной обработки. Каждый подзапрос выполняется однократно для каждой строки внешнего запроса.



Коррелированные подзапросы


Подзапрос ссылается на столбец из таблицы в родительском запросе.

```
SELECT column1, column2, ...
FROM   table1 outer
WHERE  column1 operator
      (SELECT column1, column2
        FROM   table2
        WHERE  expr1 =
              outer.expr2) ;
```

Использование коррелированных подзапросов

Найдите всех сотрудников с окладом больше среднего в своем отделе.

```
SELECT last_name, salary, department_id  
FROM   employees outer  
WHERE  salary >
```



```
(SELECT AVG(salary)  
FROM   employees  
WHERE  department_id =  
       outer.department_id);
```

Каждый раз, когда
обрабатывается
строка из внешнего
запроса, выполняется
внутренний запрос

Использование коррелированных подзапросов

Выведите сведения о сотрудниках, хотя бы дважды сменивших должность.

```
SELECT e.employee_id, last_name, e.job_id
FROM   employees e
WHERE  2 <= (SELECT COUNT (*)
              FROM   job_history
              WHERE  employee_id = e.employee_id) ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID
1	101	Kochhar	AD_VP
2	176	Taylor	SA_REP
3	200	Whalen	AD_ASST

Использование оператора EXISTS

- Оператор **EXISTS** проверяет существование строк в наборе результатов подзапроса.
- Если значение строки подзапроса найдено:
 - поиск не продолжается во внутреннем запросе;
 - условие помечается как **TRUE**.
- Если значение строки подзапроса не найдено:
 - условие помечается как **FALSE**;
 - поиск продолжает выполняться во внутреннем запросе.

Поиск сотрудников, у которых есть хотя бы один подчиненный

```
select  
emp_id, first_name, last_name  
from employee outer  
where exists (select 1 from employee a where a.superior_emp_id = outer.emp_id);
```

EMP_ID	FIRST_NAME	LAST_NAME	SUPERIOR_EMP_ID
1	Michael	Smith	-
3	Robert	Tyler	1
4	Susan	Hawthorne	3
6	Helen	Fleming	4
10	Paula	Roberts	4
13	John	Blake	4
16	Theresa	Markham	4

Поиск всех счетов, у которых отсутствует сберегательный счет

```
select
account_id, product_cd
from account outer
where NOT EXISTS (select 1
                  from product a
                  where a.product_cd = 'CHK'
                  and outer.product_cd = a.product_cd);
```

ACCOUNT_ID	PRODUCT_CD
2	SAV
3	CD
5	SAV
7	MM
9	SAV
10	MM
13	CD
14	CD
16	SAV

Предложение WITH

- Используя предложение WITH, можно применять один и тот же блок запроса в инструкции SELECT, когда она встречается более одного раза в сложном запросе.
- Предложение WITH позволяет извлекать результаты блока запроса и сохранять их во временной табличной области пользователя.
- Предложение WITH помогает ускорить работу.

Пример предложения WITH

```
with avg_bal as
(select
    avg(avail_balance) as bal
from account
),
chk_acc as
(select
*
from account
where product_cd = 'CHK'
)
select
account_id, avail_balance
from chk_acc
where avail_balance >= (select bal from avg_bal);
```

ACCOUNT_ID	AVAIL_BALANCE
20	23575.12
23	38552.05