2

Ограничение и сортировка выходных данных

Рассматриваемые вопросы

- Ограничение количества строк, возвращаемых запросом.
- Сортировка возвращаемых строк.
- Использование переменных подстановки для ограничения и сортировки выходных данных во время выполнения запросов

Ограничение количества выбираемых строк путем отбора

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20
4	205	Higgins	AC_MGR	110
5	206	Gietz	AC_ACCOUNT	110

...

“retrieve all employees in department 90”

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

Ограничение количества выбираемых строк

- Количество возвращаемых строк можно ограничить с помощью предложения WHERE:

```
SELECT *|{[DISTINCT] столбец|выражение
[псевдоним],...}
FROM   таблица
[WHERE условие(я)];
```

- Предложение WHERE следует за предложением FROM.

Использование предложения WHERE

```
SELECT employee_id, last_name, job_id,
department_id
FROM   employees
WHERE  department_id = 90 ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

Символьные строки и даты

- Символьные строки и даты заключаются в апострофы.
- В символьных значениях различаются регистры символов, а в датах – форматы.
- Формат дат по умолчанию – DD-MON-RR (число-месяц-год).

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen' ;
```

```
SELECT last_name
FROM   employees
WHERE  hire_date = '17-FEB-96' ;
```

Условия сравнения

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
`BETWEEN ...AND...`	Between two values (inclusive)
`IN(set)`	Match any of a list of values
`LIKE`	Match a character pattern
`IS NULL`	Is a null value

Использование условий сравнения

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000 ;
```

	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

Использование условия BETWEEN

Условие BETWEEN используется для вывода строк на основе диапазона значений

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500 ;
```

Нижняя граница

Верхняя граница

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

Использование условия IN

Условие принадлежности IN используется для проверки на вхождение значений в список.

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	201	Hartstein	13000	100
2	101	Kochhar	17000	100
3	102	De Haan	17000	100
4	124	Mourgos	5800	100
5	149	Zlotkey	10500	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

Использование условия LIKE

- Условие LIKE используется для поиска символьных значений по шаблону с метасимволами.
- Условия поиска могут включать цифровые и символьные литералы:
  - % обозначает ноль или много символов;
  - _ обозначает один символ.

```
SELECT first_name
FROM   employees
WHERE  first_name LIKE 'S%' ;
```

Использование условия LIKE

- Метасимволы можно комбинировать:

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%' ;
```

	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

- Для поиска % или _ можно использовать идентификатор ESCAPE .

Использование условия NULL

С помощью оператора IS NULL производится проверка на неопределенные значения.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```

	LAST_NAME	MANAGER_ID
1	King	(null)

Логические условия

Оператор	Значение
AND	Возвращает результат ИСТИННО, если выполняются *оба* условия.
OR	Возвращает результат ИСТИННО, если выполняется *любое из* условий.
NOT	Возвращает результат ИСТИННО, если следующее условие не выполняется.

Использование оператора AND

Оператор AND (“И“) требует выполнения обоих условий.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
AND    job_id LIKE '%MAN%' ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	201	Hartstein	MK_MAN	13000
2	149	Zlotkey	SA_MAN	10500

Использование оператора OR

Оператор `OR` (“ИЛИ“) требует выполнения любого из условий.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%' ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	201	Hartstein	MK_MAN	13000
2	205	Higgins	AC_MGR	12000
3	100	King	AD_PRES	24000
4	101	Kochhar	AD_VP	17000
5	102	De Haan	AD_VP	17000
6	124	Mourgos	ST_MAN	5800
7	149	Zlotkey	SA_MAN	10500
8	174	Abel	SA_REP	11000

Использование оператора NOT

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

Приоритеты операторов

Оператор	Значение
1	Арифметические операторы
2	Оператор конкатенации
3	Операторы сравнения
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Не равно
7	Логическое условие NOT
8	Логическое условие AND
9	Логическое условие OR

Изменить стандартную последовательность вычислений можно с помощью круглых скобок.

Приоритеты операторов

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR     job_id = 'AD_PRES'
AND    salary > 15000;
```

1

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000
2	Abel	SA_REP	11000
3	Taylor	SA_REP	8600
4	Grant	SA_REP	7000

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  (job_id = 'SA_REP'
OR     job_id = 'AD_PRES')
AND    salary > 15000;
```

2

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000

Использование предложения ORDER BY

- Предложение ORDER BY используется для сортировки строк:
  - ASC: сортировка по возрастанию (используется по умолчанию)
  - DESC: сортировка по убыванию

В команде SELECT предложение ORDER BY указывается последним.

```
SELECT   last_name, job_id, department_id,
hire_date
FROM     employees
ORDER BY hire_date ;
```

	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES	90	17-JUN-87
2	Whalen	AD_ASST	10	17-SEP-87
3	Kochhar	AD_VP	90	21-SEP-89
4	Hunold	IT_PROG	60	03-JAN-90
5	Ernst	IT_PROG	60	21-MAY-91
6	De Haan	AD_VP	90	13-JAN-93

...

Сортировка

- Сортировка в порядке убывания:

```
SELECT   last_name, job_id, department_id,
hire_date
FROM     employees
ORDER BY hire_date DESC ;
```

1

- Сортировка по псевдониму столбца:

```
SELECT employee_id, last_name, salary*12 annsal
FROM   employees
ORDER BY annsal ;
```

2

Сортировка

- Сортировка с использованием порядкового номера столбца в списке SELECT:

```
SELECT   last_name, job_id, department_id,
hire_date
FROM     employees
ORDER BY 3;
```

3

- Сортировка по нескольким столбцам:

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

4

Предложение SQL Row Limiting

- Предложение `row_limiting_clause` позволяет ограничить количество строк, возвращаемых запросом
- Запросы с сортировкой и ограничением выборки первыми N строками известны как `Top-N` запросы или `Top-N` анализ.
- Вы можете указать количество извлекаемых строк или процент строк с помощью ключевых слов предложения `FETCH FIRST`.
- Можно использовать ключевое слово `OFFSET` для того, чтобы не извлекать(пропустить) первые N строк в выборке.
- Ключевое слово `WITH TIES` позволяет включить в выборку дополнительные строки, соответствующие значению ключа сортировки последней строки `Top-N` запроса

Использование предложения SQL Row Limiting

Предложение `row_limiting_clause` в команде `SELECT` должно следовать после предложения `ORDER BY`.

Синтаксис:

```
subquery::=
{ query_block
  | subquery { UNION [ALL] | INTERSECT | MINUS }
subquery
[ { UNION [ALL] | INTERSECT | MINUS } subquery ]...
  | ( subquery )
{
[ order_by_clause ]
[OFFSET offset { ROW | ROWS }]
[FETCH { FIRST | NEXT } [{ row_count | percent PERCENT
}] { ROW | ROWS }
  { ONLY | WITH TIES }]
```

Пример SQL Row Limiting

```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
FETCH FIRST 5 ROWS ONLY;
```

Script Output | Query Result

SQL | All Rows Fetched: 5

	EMPLOYEE_ID	FIRST_NAME
1	100	Steven
2	101	Neena
3	102	Lex
4	103	Alexander
5	104	Bruce

```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```

	EMPLOYEE_ID	FIRST_NAME
1	107	Diana
2	124	Kevin
3	141	Trenna
4	142	Curtis
5	143	Randall

Переменные подстановки

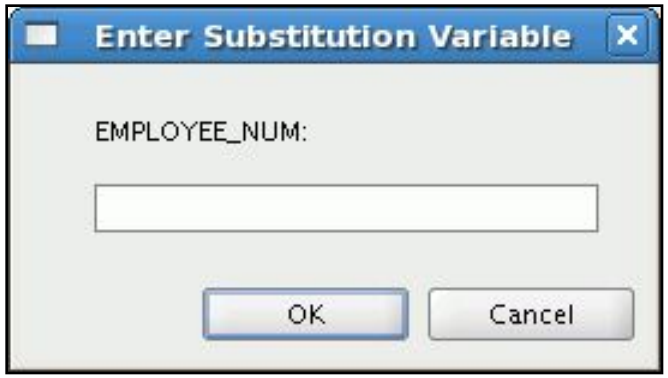
Переменные подстановки

- Использование переменных подстановки *i*SQL*Plus для временного хранения значений:
  - одиночный амперсанд (&) и двойной амперсанд (&&);
- Переменные подстановки могут замещать или дополнять:
  - Условие WHERE
  - Предложение ORDER BY
  - Выражение столбца
  - Имя таблицы
  - Целую команду SELECT

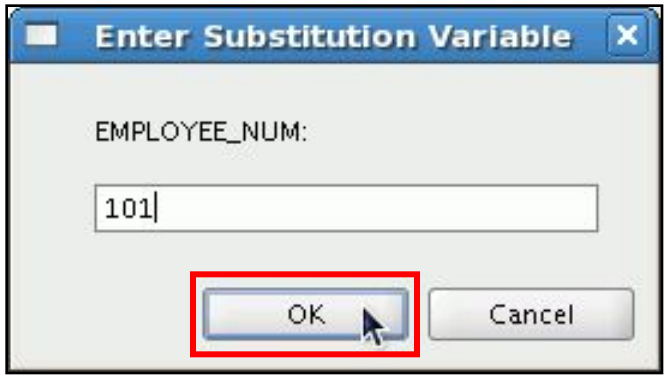
Использование переменной подстановки с одним амперсандом (&)

Переменная с одним амперсандом (&) позволяет запросить значение у пользователя

```
SELECT employee_id, last_name, salary,
department_id
FROM   employees
WHERE  employee_id = &employee_num ;
```



Использование переменной подстановки с одним амперсандом (&)

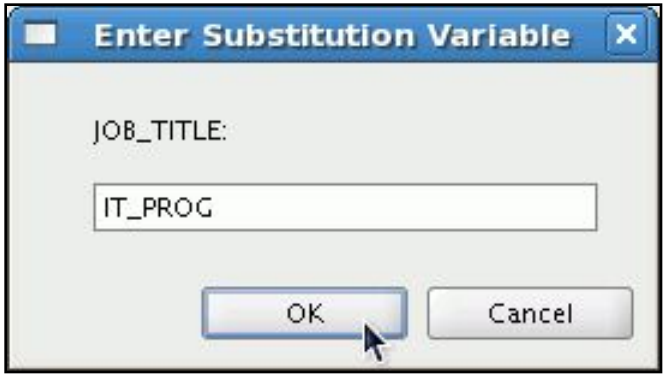


	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	101	Kochhar	17000	90

Символьные значения и даты в переменных подстановки

Даты и символьные значения заключаются в апострофы

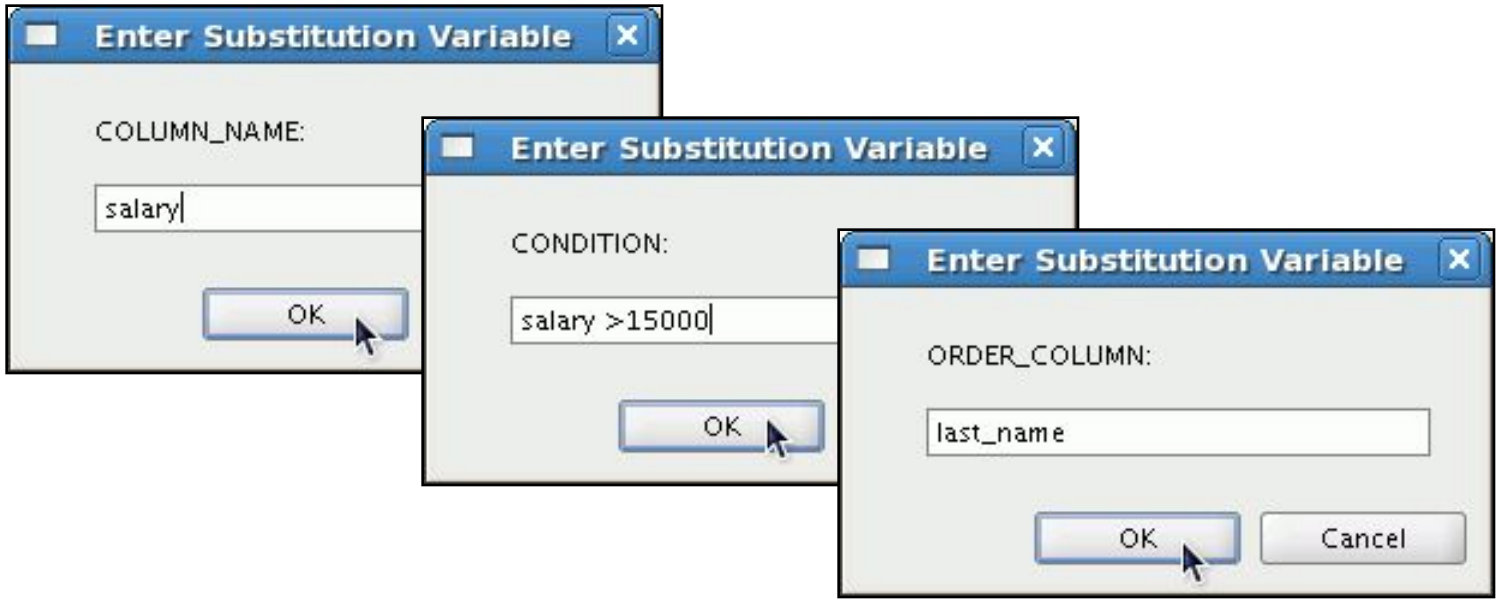
```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title' ;
```



	LAST_NAME	DEPARTMENT_ID	SALARY*12
1	Hunold	60	108000
2	Ernst	60	72000
3	Lorentz	60	50400

Задание имен столбцов, выражений и текста

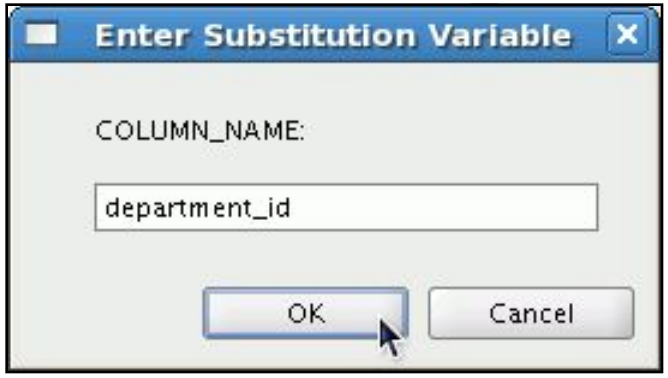
```
SELECT employee_id, last_name, job_id,&column_name
FROM   employees
WHERE  &condition
ORDER BY &order_column ;
```



Использование переменной подстановки с двумя амперсандами (&&)

Переменная подстановки с двумя амперсандами (&&) позволяет многократно использовать значение переменной, не запрашивая его повторно у пользователя.

```
SELECT   employee_id, last_name, job_id,
&&column_name
FROM     employees
ORDER BY &column_name ;
```



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20

...

Использование команды DEFINE

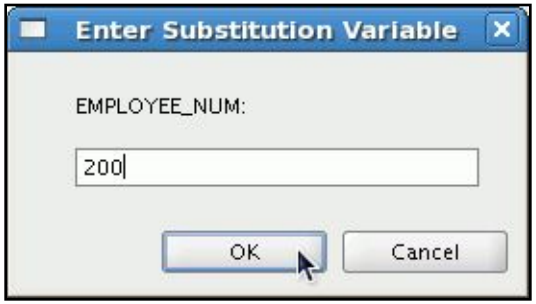
- Используйте команду SQL*Plus DEFINE, чтобы создать и присвоить значение переменной.
- Используйте команду SQL*Plus UNDEFINE, чтобы удалить переменную.

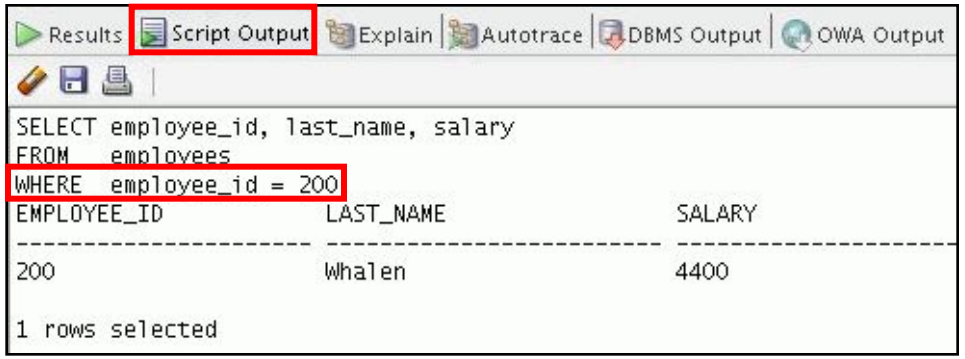
```
DEFINE employee_num = 200

SELECT employee_id, last_name, salary,
department_id
FROM   employees
WHERE  employee_id = &employee_num ;

UNDEFINE employee_num
```

Использование команды VERIFY

Если задан режим SET VERIFY ON, выводится текст команды до и после замены переменных подстановки значениями.

```
SET VERIFY ON
SELECT employee_id, last_name, salary
FROM   employees
WHERE  employee_id = &employee_num;
```





Тест

Какие операторы для предложения `WHERE` написаны верно?

1. `>=`
2. `IS NULL`
3. `!=`
4. `IS LIKE`
5. `IN BETWEEN`
6. `<>`

Итоги

- Использование предложения WHERE для ограничения количества выводимых строк
  - Условия сравнения
  - Условия BETWEEN, IN, LIKE и NULL
  - Логические операторы AND, OR и NOT
- Использование предложения ORDER BY для сортировки выходных результатов:

```
SELECT   *|{[DISTINCT] столбец|выражение [псевдоним],...}
FROM     таблица
[WHERE   условие(я)]
[ORDER BY  {столбец, выражение, псевдоним} [ASC[ASC|DESC]]
;
```

- Использование переменных подстановки с одним и двумя амперсандами для ограничения и ортировки выходных данных во время выполнения

Обзор практического занятия 2

- Выборка данных и изменение последовательности вывода строк
- Ограничение количества возвращаемых строк с помощью предложения WHERE
- Сортировка строк с помощью предложения ORDER BY
- Использование переменных подстановки для создания более гибких команд SELECT языка SQL