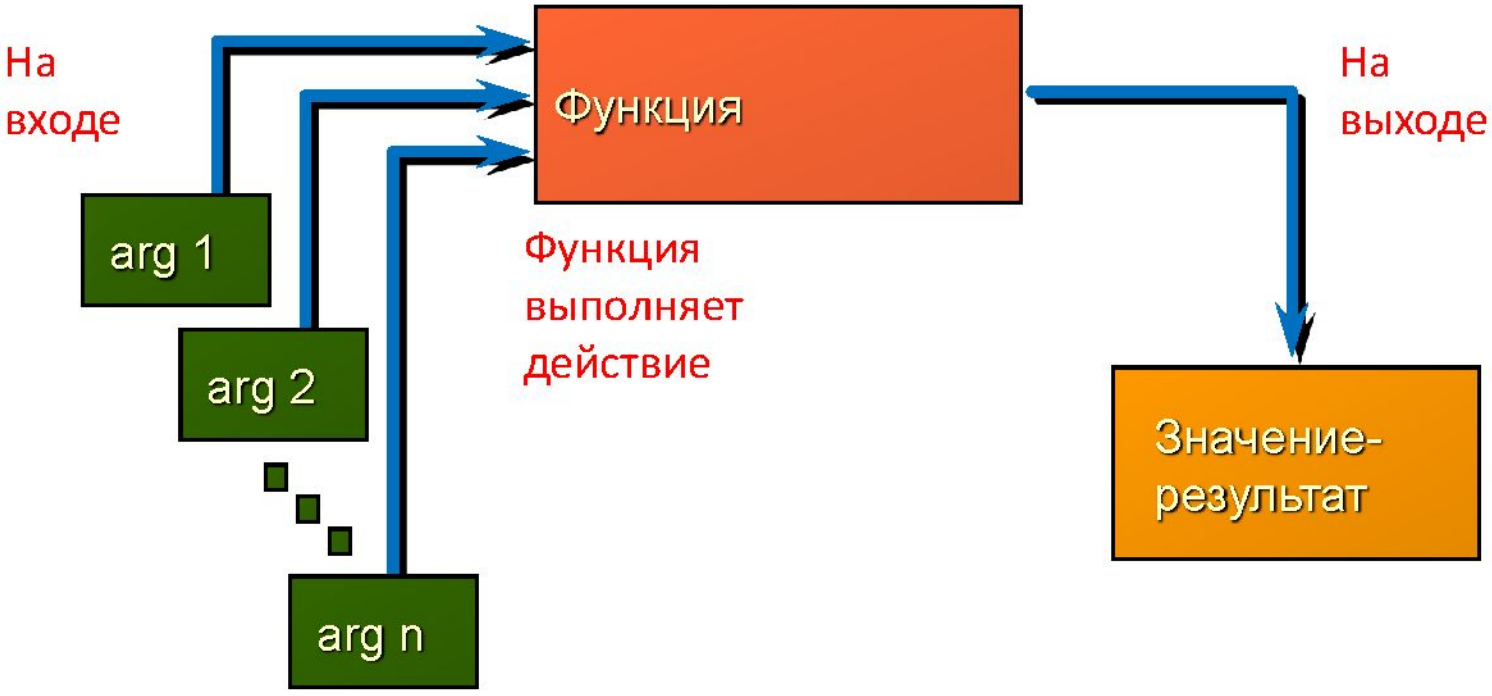
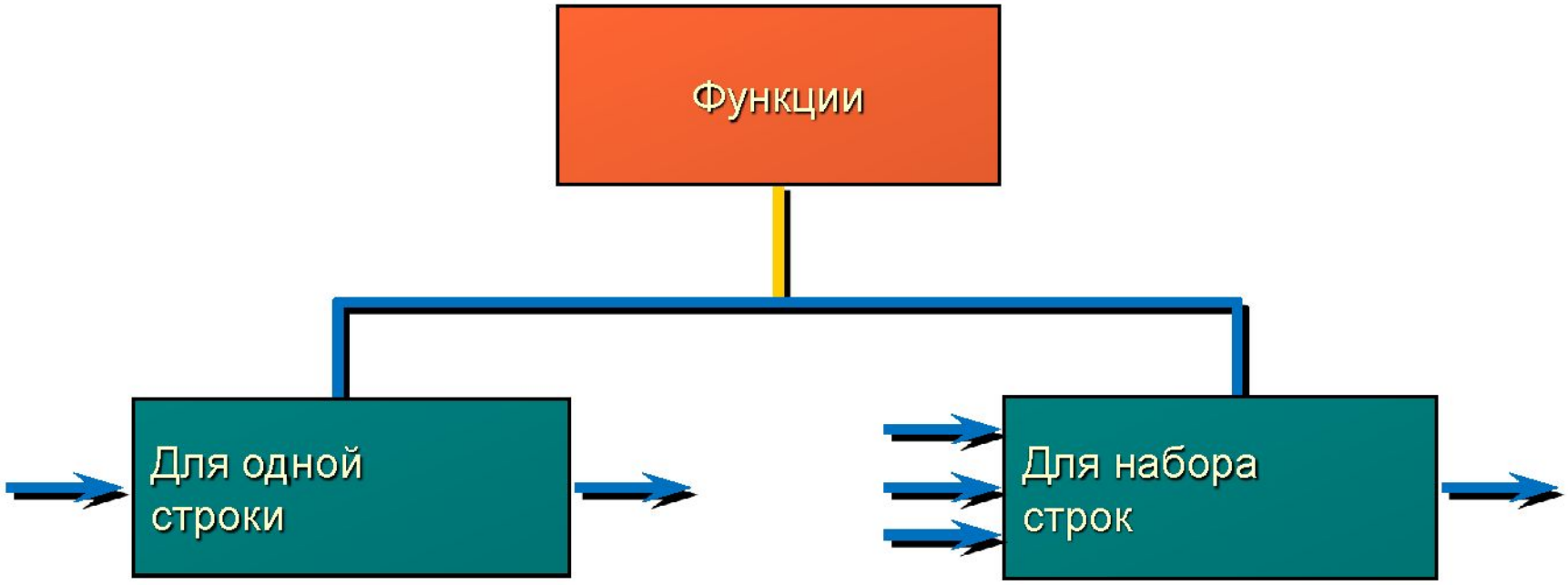
Курс QA. Скалярные и агрегатные функции в SQL

NetCracker®

Функции SQL



Два вида функций SQL

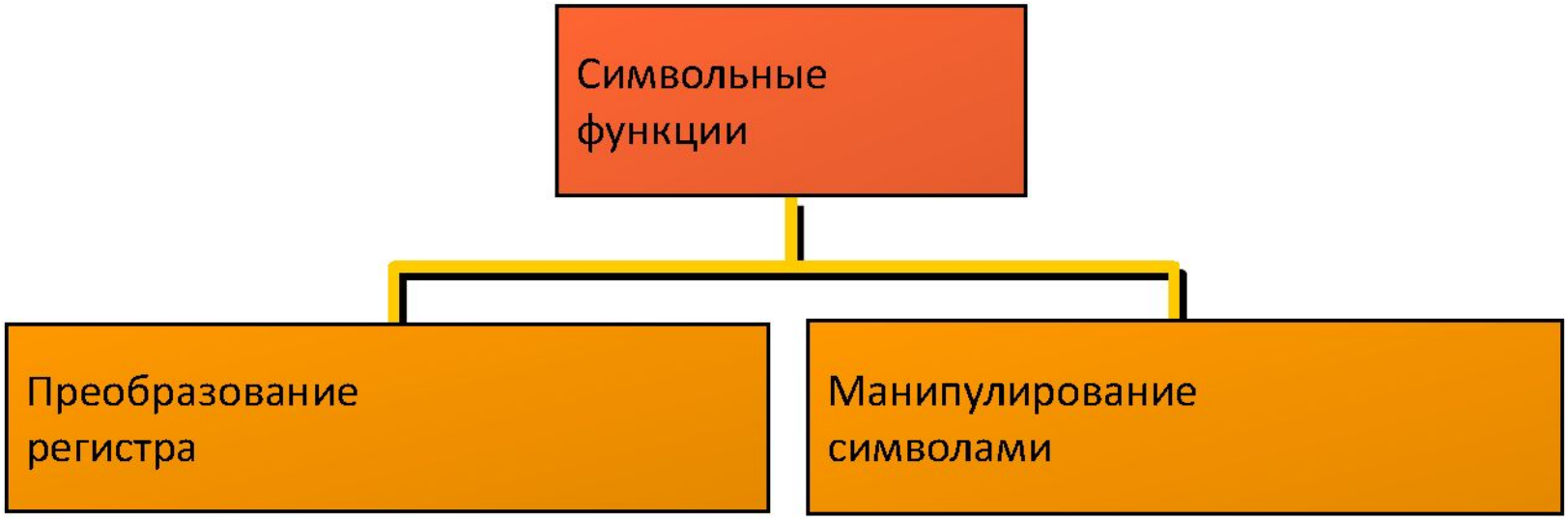


Функции для одной строки

- Манипулируют данными
- Допускают аргументы и возвращают одно значение
- Действуют на каждую поданную строку
- Возвращают одно результирующее значения для одной строки
- Могут изменять тип данных
- Могут быть вложенными

```
function_name (column|expression, [arg1, arg2,...])
```

Символьные функции



Функции преобразования регистра

Функция	Результат
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL course')	Sql Course

Использование функций преобразования регистра

Вывод номера служащего, имя и номер департамента для служащего Blake.

```
SQL> SELECT   empno, ename, deptno
  2  FROM     emp
  3  WHERE    ename = 'blake';
no rows selected
```

```
SQL> SELECT   empno, ename, deptno
  2  FROM     emp
  3  WHERE    LOWER(ename) = 'blake';
```

```
     EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7698 BLAKE              30
```

Функции манипулирования символами

Манипулируют символьными строками

Функция	Результат
CONCAT('Good', 'String')	GoodString
SUBSTR('String',1,3)	Str
LENGTH('String')	6
INSTR('String', 'r')	3
LPAD(sal,10,'*')	******5000

Использование функций манипулирования символами

```
SQL> SELECT ename, CONCAT (ename, job), LENGTH(ename),
  2    INSTR(ename, 'A')
  3 FROM    emp
  4 WHERE   SUBSTR(job,1,5) = 'SALES';
```

ENAME	CONCAT(ENAME,JOB)	LENGTH(ENAME)	INSTR(ENAME,'A')
MARTIN	MARTINSALESMAN	6	2
ALLEN	ALLENSALESMAN	5	1
TURNER	TURNERSALESMAN	6	0
WARD	WARDSALESMAN	4	2

Числовые функции

- ROUND: Округляет значение до указанного вида

  ROUND(45.926, 2)  45.93

- TRUNC: Округляет в меньшую сторону до указанного вида

  TRUNC(45.926, 2)  45.92

- MOD: Возвращает остаток от деления

  MOD(1600, 300)  100

Использование функции ROUND

```
SQL> SELECT ROUND(45.923,2), ROUND(45.923,0),
  2           ROUND(45.923,-1)
  3  FROM    DUAL;
```

ROUND(45.923,2)	ROUND(45.923,0)	ROUND(45.923,-1)
45.92	46	50

Использование функции TRUNC

```
SQL> SELECT TRUNC(45.923,2), TRUNC(45.923),
  2         TRUNC(45.923,-1)
  3  FROM   DUAL;
```

```
TRUNC(45.923,2) TRUNC(45.923) TRUNC(45.923,-1)
--------------- -------------- ---------------
          45.92             45              40
```

Использование функции MOD

Вычисление остатка от деления зарплаты на комиссионные для всех работников, работающих продавцами.

```
SQL> SELECT   ename, sal, comm, MOD(sal, comm)
  2  FROM     emp
  3  WHERE    job = 'SALESMAN';
```

ENAME	SAL	COMM	MOD(SAL,COMM)
MARTIN	1250	1400	1250
ALLEN	1600	300	100
TURNER	1500	0	1500
WARD	1500	50.5	35.5

Работа с датами

- Oracle хранит даты во внутреннем числовом формате: век, год, месяц, день, час, минута, секунда.
- Формат даты по умолчанию имеет вид DD-MM-YY.
- Функция SYSDATE возвращает дату и время.

Арифметика с датами

- Добавление или вычитания числа к/из даты приводит к получению даты.
- Вычитание двух дат друг из друга имеет результатом число дней между этими датами.
- Для добавления часов к дате разделите их количество на 24.

Использование арифметических операторов при работе с датами

```
SQL> SELECT ename, (SYSDATE-hiredate)/7 WEEKS
  2  FROM   emp
  3  WHERE  deptno = 10;
```

```
ENAME          WEEKS
---------- ---------
KING       830.93709
CLARK      853.93709
MILLER     821.36566
```

Функции для дат

Функция	Описание
MONTHS_BETWEEN	Количество месяцев между двумя датами
ADD_MONTHS	Добавление календарных месяцев к дате
NEXT_DAY	Следующий день относительно даты
LAST_DAY	Последний день месяца
ROUND	Округляет дату
TRUNC	Округляет дату в меньшую сторону

Использование функций для дат

MONTHS_BETWEEN ('01.09.2012','13.01.2013')

-4,3870968

ADD_MONTHS ('13.01.2013',6)

'13.07.13'

NEXT_DAY ('13.01.2013','понедельник')

'14.01.13'

LAST_DAY('13.01.2013')

'31.01.13'

Использование функций для дат

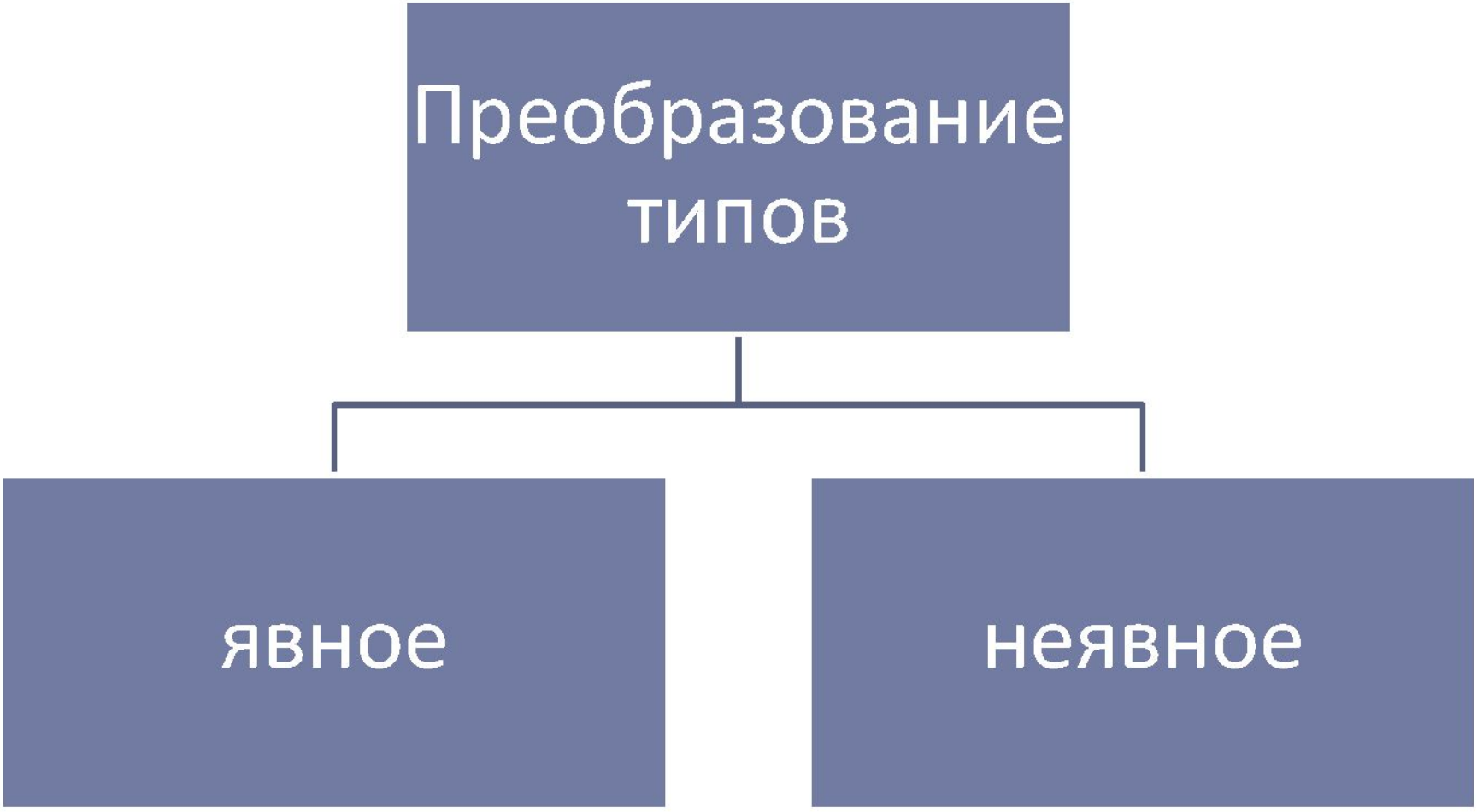
ROUND('25.07.12','MONTH') → 01.08.12

ROUND('25.07.12','YEAR') → 01.01.13

TRUNC('25.07.12','MONTH') → 01.07.12

TRUNC('25.07.12','YEAR') → 01.01.12

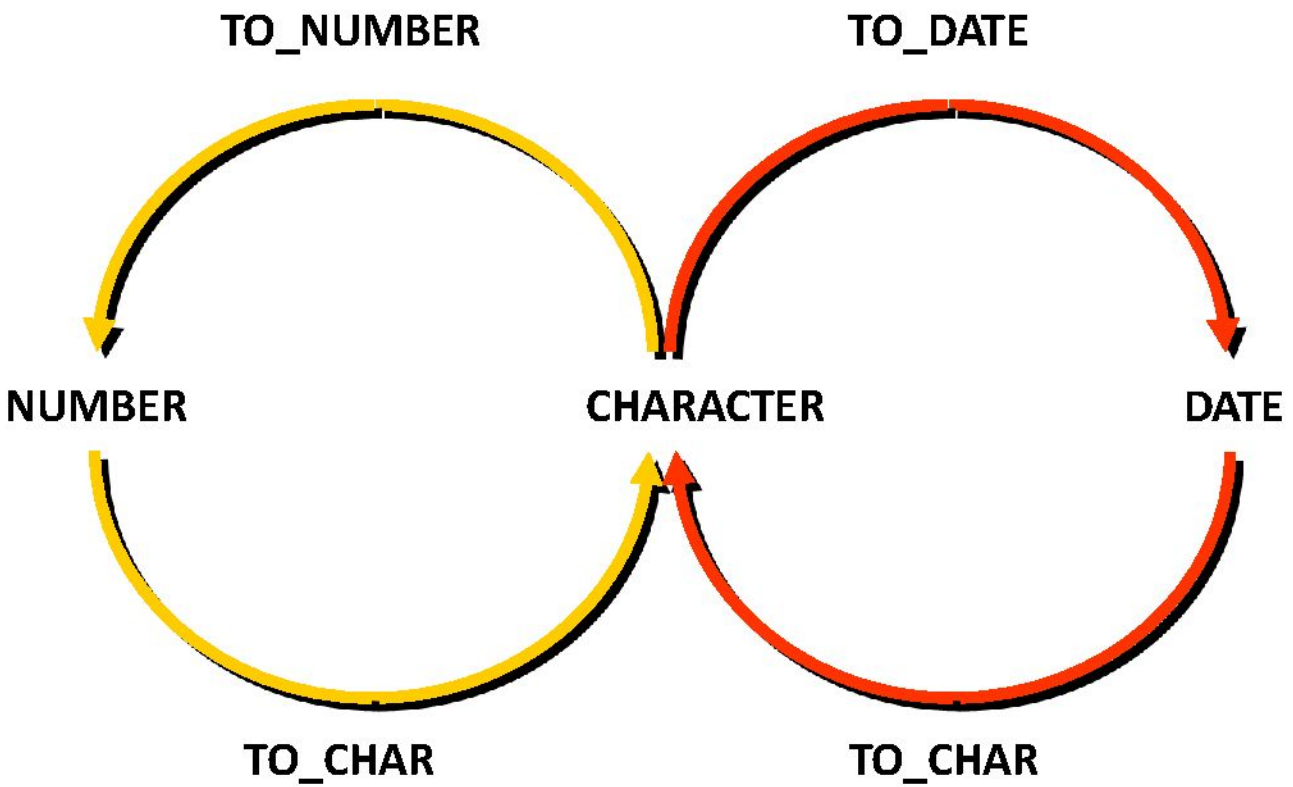
Функции преобразования



Неявное приведение типов

	CHAR	VARCHAR2	NCHAR	NVARCHAR2	DATE	DATETIME/ INTERVAL	NUMBER	BINARY_FLOAT	BINARY_DOUBLE	LONG	RAW	ROWID	CLOB	BLOB	NCLOB
CHAR	--	X	X	X	X	X	X	X	X	X	X	--	X	X	X
VARCHAR2	X	--	X	X	X	X	X	X	X	X	X	X	X	--	X
NCHAR	X	X	--	X	X	X	X	X	X	X	X	X	X	--	X
NVARCHAR2	X	X	X	--	X	X	X	X	X	X	X	X	X	--	X
DATE	X	X	X	X	--	--	--	--	--	--	--	--	--	--	--
DATETIME/ INTERVAL	X	X	X	X	--	--	--	--	--	X	--	--	--	--	--
NUMBER	X	X	X	X	--	--	--	X	X	--	--	--	--	--	--
BINARY_ FLOAT	X	X	X	X	--	--	X	--	X	--	--	--	--	--	--
BINARY_ DOUBLE	X	X	X	X	--	--	X	X	--	--	--	--	--	--	--
LONG	X	X	X	X	--	X[1]	--	--	--	--	X	--	X	--	X
RAW	X	X	X	X	--	--	--	--	--	X	--	--	--	X	--
ROWID	--	X	X	X	--	--	--	--	--	--	--	--	--	--	--
CLOB	X	X	X	X	--	--	--	--	--	X	--	--	--	--	X
BLOB	--	--	--	--	--	--	--	--	--	--	X	--	--	--	--
NCLOB	X	X	X	X	--	--	--	--	--	X	--	--	X	--	--

Явное приведение типов



Функция TO_CHAR для дат

```
TO_CHAR(date, 'fmt')
```

Модель формата:

- Заключается в одинарные кавычки и чувствительна к регистру
- Может включать корректные элементы форматирования
- Элемент fm служит для удаления лишних пробелов и предваряющих нулей
- Отделен от даты запятой

Элементы формата даты

YYYY	Год, записанный цифрами
YEAR	Строчная запись года
MM	Номер месяца (2 цифры)
MONTH	Полное имя месяца
DY	Сокращение названия дня недели (3 символа)
DAY	День недели

Элементы формата даты

- Вывод времени дня.

HH24:MI:SS AM	15:45:32 PM

- Строки можно добавить, заключив их в кавычки.

DD "of" MONTH	12 of OCTOBER

- Суффиксы числительных.

ddspth	fourteenth

Использование функции TO_CHAR для дат

```
SQL> SELECT ename,
  2         TO_CHAR(hiredate, 'fmDD Month YYYY') HIREDATE
  3  FROM    emp;
```

```
ENAME      HIREDATE
---------- -----------------
KING       17 November 1981
BLAKE      1 May 1981
CLARK      9 June 1981
JONES      2 April 1981
MARTIN     28 September 1981
ALLEN      20 February 1981
...
14 rows selected.
```

Функция TO_CHAR для чисел

```
TO_CHAR(number, 'fmt')
```

Используйте форматы функции TO_CHAR для получения символьного представления чисел.

9	Представляет цифру
0	Выводит ноль
$	Знак доллара
L	Местный знак валюты
.	Десятичная точка
,	Разделитель тысяч

Использование функции TO_CHAR с числами

```
SQL> SELECT   TO_CHAR(sal,'$99,999') SALARY
  2  FROM     emp
  3  WHERE    ename = 'SCOTT';
```

```
  SALARY
--------
  $3,000
```

Функции TO_NUMBER и TO_DATE

Преобразование символьной строки к числу производится с помощью функции

TO_NUMBER

```
TO_NUMBER(char)
```

TO_DATE

```
TO_DATE(char[, 'fmt'])
```

Формат даты RR

Текущий год	Указанная дата	ФорматRR	ФорматYY
2013	27.10.95	95	1995
2055	27.10.95	95	2095
2013	27.10.17	17	2017
2055	27.10.17	17	2117

		Указанный двумя цифрами год	
		0-49	50-99
Две цифры текущего года	0-49	Возвращается дата текущего века	Возвращается дата предыдущего века
	50-99	Возвращается дата следующего века	Возвращается дата текущего века

Работа с NULL

- Для работы с NULL-значениями существуют функции:
  - `NVL (expr1, expr2)`
  - `NVL2 (expr1, expr2, expr3)`
  - `NULLIF (expr1, expr2)`
  - `COALESCE (expr1, expr2, ..., exprn)`

Функция NVL

Преобразует null в реальное значение

- Может использоваться для дат, символьных и числовых данных.
- Типы данных должны совпадать
  - NVL(comm,0)
  - NVL(hiredate,'01.01.97')
  - NVL(job,'No Job Yet')

Использование функции NVL

```
SQL> SELECT ename, sal, comm, (sal*12)+NVL(comm,0)
  2  FROM   emp;
```

```
ENAME             SAL       COMM (SAL*12)+NVL(COMM,0)
---------- --------- --------- --------------------
KING             5000                         60000
BLAKE            2850                         34200
CLARK            2450                         29400
JONES            2975                         35700
MARTIN           1250      1400                16400
ALLEN            1600       300                19500
...
14 rows selected.
```

Функция NVL2

```
SELECT ename,  salary, comm,
       NVL2(comm,
            'SAL+COMM', 'SAL') income
FROM   emp WHERE deptno IN (50, 80);
```


Функция NULLIF

```
SELECT ename,
       NULLIF(empno, mng) result
FROM   emp;
```


Функция COALESCE

- В отличии от NVL функция COALESCE может принимать множество альтернативных значений.
- Функция COALESCE возвращает первый аргумент, не равный null.

Использование COALESCE

```
SELECT ename, emplno,
COALESCE(TO_CHAR(comm),TO_CHAR(mng),
         'No commission and no manager')
FROM emp;
```


Агрегатные функции (Group Functions)

- Агрегатные функции принимают в аргументом множество строк и возвращает одну строку результата для каждой группы

Основные агрегирующие функции

Функция	Назначение	Допустимые аргументы
AVG	Среднее значение	Числа
COUNT	Количество	Любой тип
MAX	Максимальное значение	Числа, даты, строки
MIN	Минимальное значений	Числа, даты, строки
STDDEV	Среднеквадратическое отклонение	Числа
SUM	Сумма	Числа
VARIANCE	Дисперсия	Числа

Синтаксис

```
SELECT      group_function(column), ...
FROM        table
[WHERE      condition]
[ORDER BY   column]
```

- Пример:

```
SELECT AVG(sal), MAX(sal),
       MIN(sal), SUM(sal)
FROM   emp
WHERE  job LIKE '%SAL%'
```


Использование функции COUNT

- COUNT(*) возвращает количество строк в результате запроса:

1

```
SELECT  COUNT(*)
FROM    emp
WHERE   deptno = 30
```

6

- COUNT(*expr*) возвращает количество не пустых результатов:

2

```
SELECT  COUNT(comm)
FROM    emp
WHERE   deptno = 30
```

4

EMPNO	ENAME	COMM	DEPTNO
7839	KING		10
7698	BLAKE		30
7782	CLARK		10
7566	JONES		20
7654	MARTIN	1400	30
7499	ALLEN	300	30
7844	TURNER	0	30
7900	JAMES		30
7521	WARD	500	30

Использование DISTINCT и COUNT

- COUNT(DISTINCT expr) возвращает количество непустых неповторяющиеся *expr*.
- Сколько отделов в которых есть сотрудники?

```
SELECT
   COUNT(DISTINCT deptno)
FROM    emp
```

3

EMPNO	ENAME	COMM	DEPTNO
7839	KING		10
7698	BLAKE		30
7782	CLARK		10
7566	JONES		20
7654	MARTIN	1400	30
7499	ALLEN	300	30
7844	TURNER	0	30
7900	JAMES		30
7521	WARD	500	30

Пустые значения (Null) и агрегирующие ф-ции

- Агрегирующие функции пропускают Null значения:

1

```
SELECT AVG(comm)
FROM   emp
```

550

EMPNO	ENAME	COMM	DEPTNO
7839	KING		10
7698	BLAKE		30
7782	CLARK		10
7566	JONES		20
7654	MARTIN	1400	30
7499	ALLEN	300	30
7844	TURNER	0	30
7900	JAMES		30
7521	WARD	500	30

- Если вы их хотите включить – используйте функцию NVL :

2

```
SELECT AVG(NVL(comm, 0))
FROM   emp
```

244,44

Группировка информации

EMPNO	ENAME	SAL	DEPTNO
7839	KING	5000	10
7698	BLAKE	2850	30
7782	CLARK	1500	10
7566	JONES	2975	20
7654	MARTIN	1250	30
7499	ALLEN	1600	30
7844	TURNER	1500	30

DEPTNO	avg(SAL)
10	3520
20	2975
30	1800

Синтаксис GROUP BY

- Можно разделить строки на группы используя предложение GROUP BY.

```
SELECT      column, group_function(column)
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[ORDER BY   column]
```

Имена столбцов через запятую.
Псевдонимы столбцов не допускаются
Псевдонимы таблиц допускаются

Использование GROUP BY

- **Все** столбцы в списке SELECT для которых не применена агрегирующая функция должны быть указаны в предложении GROUP BY .

```
SELECT    deptno, AVG(sal)
FROM      emp
GROUP BY deptno
```

DEPTNO	avg(SAL)
10	3520
20	2975
30	1800

Использование GROUP BY

- Колонки указанные в GROUP BY могут не упоминаться в списке SELECT.

```
SELECT    AVG(sal)
FROM      emp
GROUP BY deptno
```


Группировка по нескольким столбцам

Средняя зарплата по должностям работников в разных отделах

EMP

ENAME	JOB	SAL	DEPTNO
BLAKE	MANAGER	2850	30
CLARK	MANAGER	1500	10
JONES	MANAGER	2975	20
MARTIN	SALESMAN	1250	30
ALLEN	SALESMAN	1600	30
TURNER	SALESMAN	1500	30
JAMES	SALESMAN	950	10

DEPNO	JOB	avg(SAL)
30	MANAGER	2850
10	MANAGER	1500
20	MANAGER	2975
30	SALESMAN	1450
10	SALESMAN	950

Группировка по нескольким столбцам

- Запрос покажет среднюю и максимальную зарплату по должностям в разных отделах:

```
SELECT   deptno, job, AVG(sal)
FROM     emp
GROUP BY deptno, job
ORDER BY job
```


Ошибки при составлении запросов

- Все колонки из Select, к которым не применяются агрегирующие функции должны быть указаны в `GROUP BY` :

```
SELECT deptno, COUNT(ename)
FROM   emp
```

ORA-00937: not a single-group group function
00937. 00000 - "not a single-group group function"

Нужно добавить `GROUP BY deptno`.

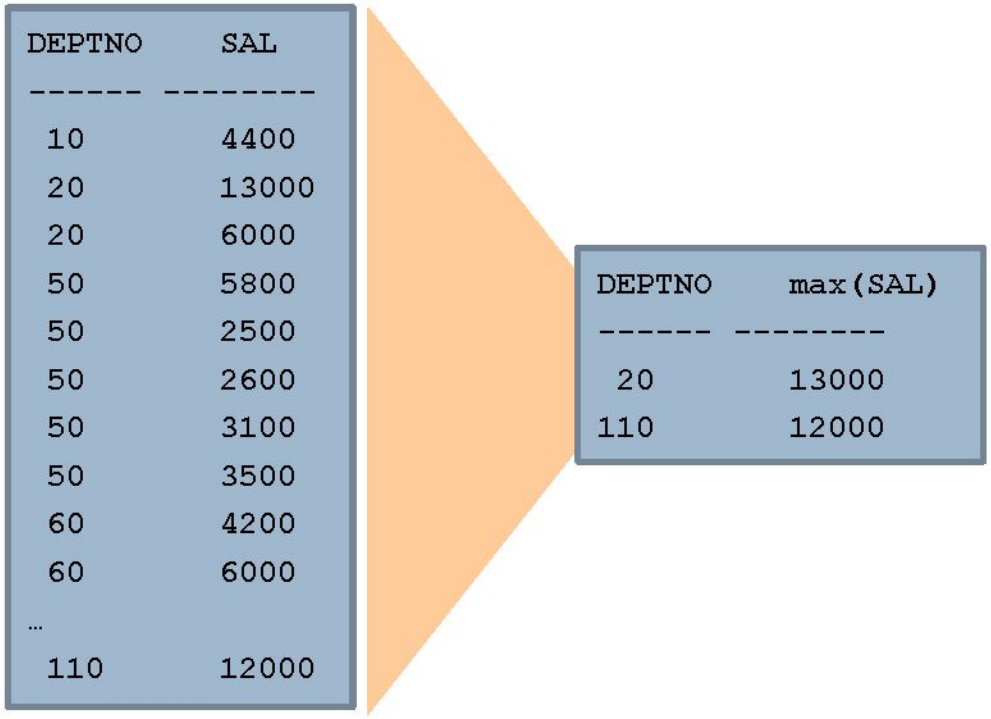
```
SELECT deptno, job, COUNT(ename)
FROM   emp
GROUP BY deptno
```

ORA-00979: not a GROUP BY expression
00979. 00000 - "not a GROUP BY expression"

Нужно добавить колонку `job` в `GROUP BY` или убрать ее из `SELECT`.

Ограничения на результаты группировки

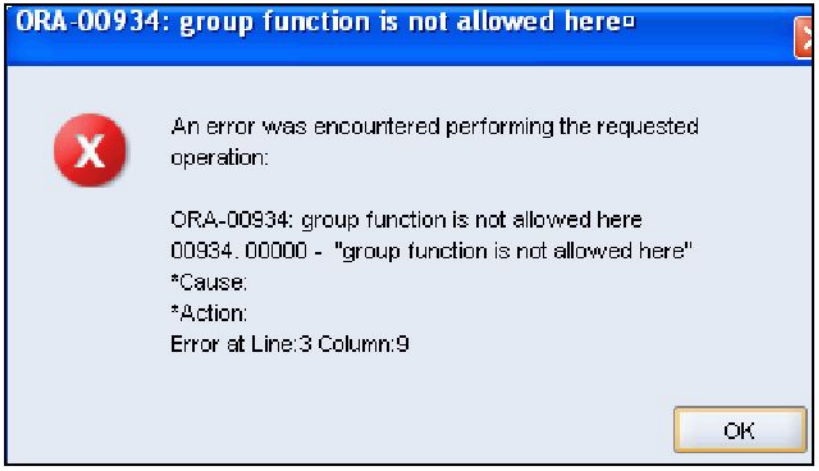
- Вывести максимальную зарплату по отделам, где она превышает 8000



Ошибки при составлении запросов

- Нельзя использовать агрегирующие функции в `WHERE`
- Для этого есть предложение `HAVING`.

```
SELECT   deptno, AVG(sal)
FROM     emp
WHERE    AVG(sal) > 8000
GROUP BY deptno
```



Ограничение с использованием Having

- При использовании HAVING сервер БД выполняет действия в следующем порядке:
  1. Строки группируются.
  2. Применяется агрегирующая функция.
  3. Возвращаются группы, которые удовлетворяют условиям в HAVING .

```
SELECT     column, group_function
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column]
```


Использование HAVING

- Вывести максимальную зарплату по отделам, где она превышает 8000

```
SELECT    deptno, MAX(sal)
FROM      emp
GROUP BY deptno
HAVING    MAX(sal)>8000
```

```
DEPTNO     max(SAL)
------ --------
 20        13000
110        12000
```


Использование HAVING и WHERE

```
SELECT   deptno, SUM(sal)
FROM     emp
WHERE    deptno != 20
GROUP BY deptno
HAVING   SUM(sal) > 10000
ORDER BY SUM(sal)
```

DEPTNO	sum(SAL)
30	12000
10	18000

Использование HAVING и WHERE

```
SELECT   deptno, SUM(sal)
FROM     emp
WHERE    deptno != 20 AND job != 'PRESIDENT'
GROUP BY deptno
HAVING   SUM(sal) > 10000
ORDER BY SUM(sal)
```

DEPTNO	sum(SAL)
30	18000
10	12000

Вложенные агрегирующие функции

- Отображение максимальной средней по отделам зарплаты:

```
SELECT   MAX(AVG(sal))
FROM     emp
GROUP BY deptno
```


Вопрос

- Какие из утверждений о `GROUP BY` справедливы?

1. Нельзя использовать псевдонимы (alias) атрибутов в предложении `GROUP BY`.
2. Все столбцы указанные `GROUP BY` должны быть указаны в `SELECT`.
3. Условие `WHERE` применяется к строкам до группировки.
4. Условие `HAVING` применяется к строкам после группировки.
5. Предложение `GROUP BY` задает порядок сортировки строк.

Итоги

- Сегодня мы рассмотрели:
  - Агрегирующие функции COUNT, MAX, MIN, SUM, и AVG
  - Как записывать запросы с предложением GROUP BY
  - Как записывать запросы с предложением HAVING

```
SELECT     column, group_function
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column]
```