

Продолжаем учиться

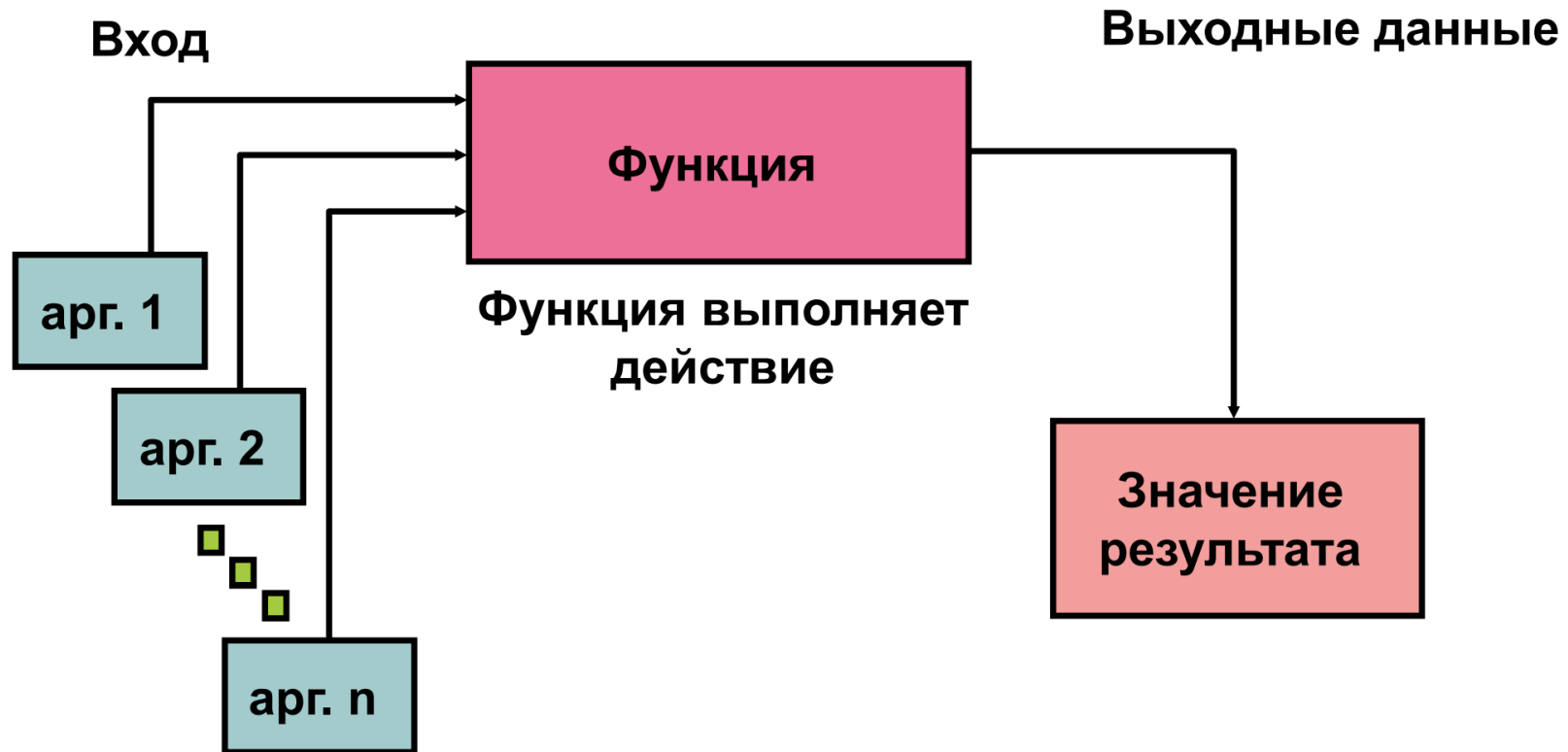
В КОНЦЕ ЗАНЯТИЯ ВЫ ИЗУЧИТЕ :

- Выполнение вычислительных операций над данными с использованием функций
- Изменение отдельных элементов данных с использованием функций
- Изменение форматов отображения дат при помощи функций
- Преобразование типов данных столбцов при помощи функций
- Использование функций NVL
- Использование логики IF-THEN-ELSE и других условных выражений в инструкции SELECT
- Использование групповых функций COUNT, MAX, MIN, SUM и AVG
- Создание запросов, использующих предложение GROUP BY
- Создание запросов, использующих предложение HAVING

Тема 1

- Однострочные функции SQL
- Символьные функции
- Числовые функции
- Работа с датами
- Функции для работы с датами

Функции SQL

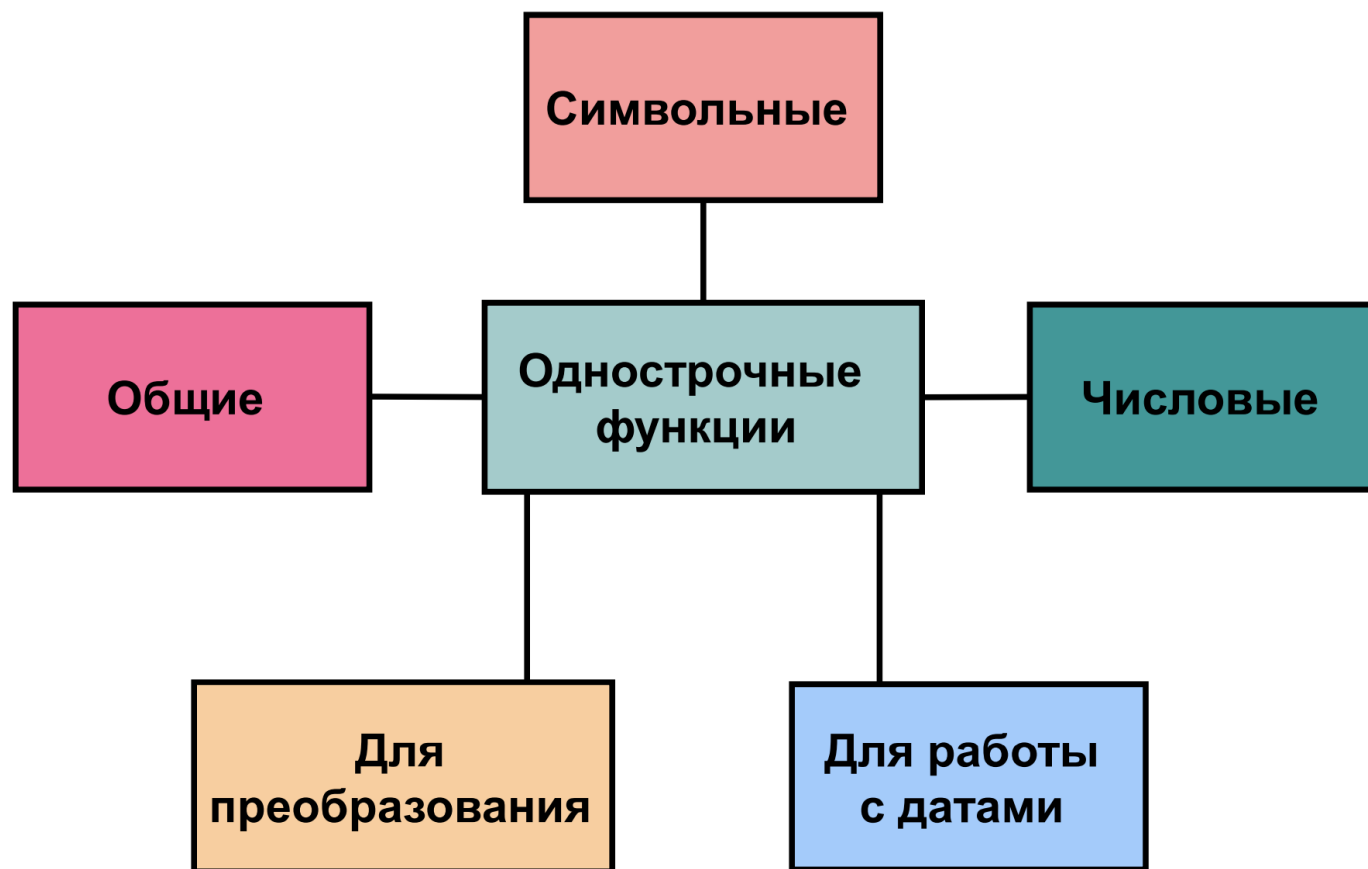


Однострочные функции

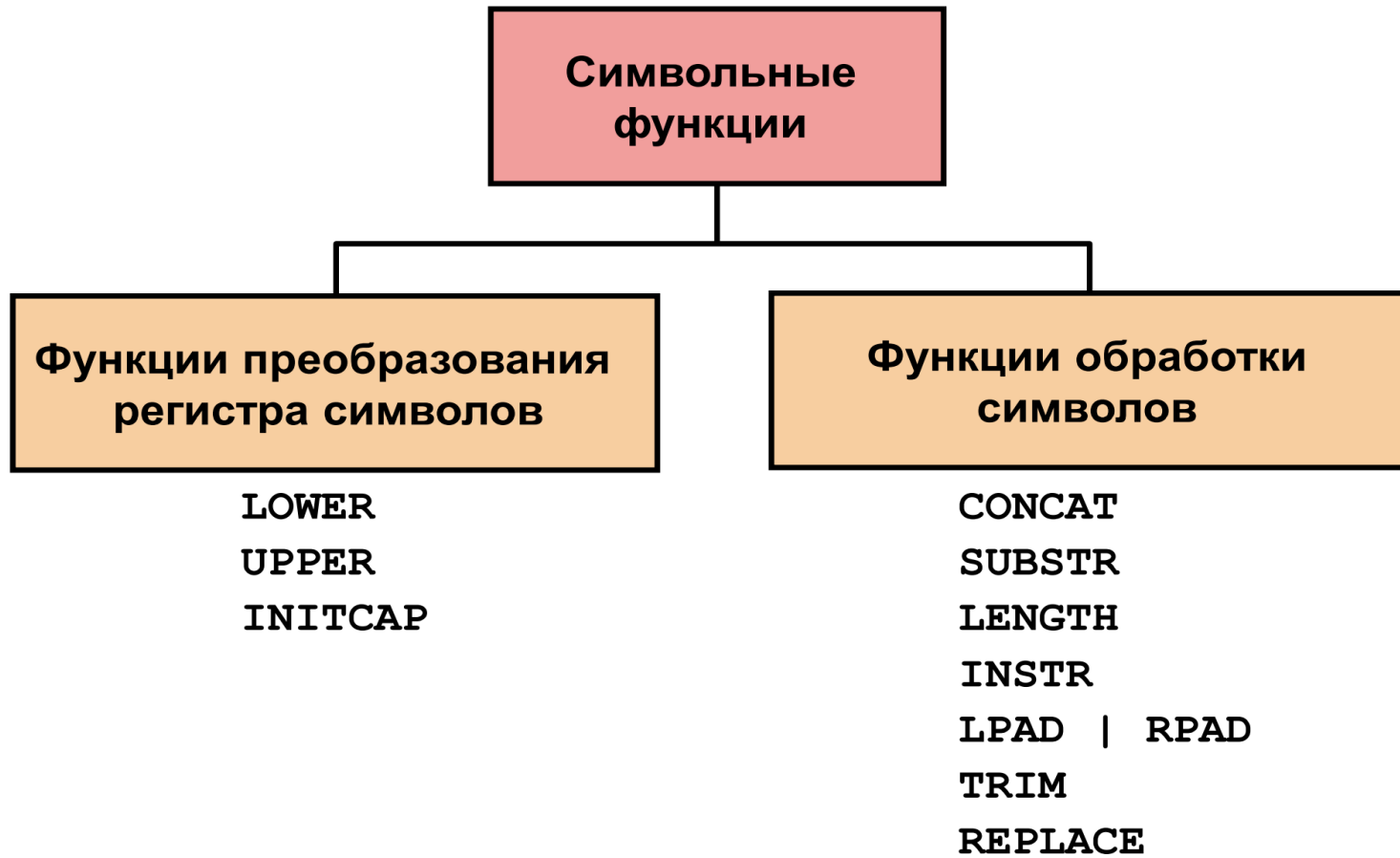
Однострочные функции:

- Манипулируют элементами данных
- Используют аргументы и возвращают одно значение
- Выполняют операции с каждой возвращаемой строкой
- Возвращают один результат по каждой строке
- Могут изменять тип данных
- Допускают вложение
- Поддерживают столбцы и выражения в качестве аргументов

Однострочные функции



Символьные функции



Функции преобразования регистра символов

Эти функции изменяют регистр символов в символьных строках:

```
select  
LOWER('HELLO WORLD!') as lower_, --перевод в нижний регистр  
UPPER('hello world!') as upper_, --перевод в верхний регистр  
INITCAP('hello world!') as initcap_ --возводит в заглавную  
from dual;
```

LOWER_	UPPER_	INITCAP_
hello world!	HELLO WORLD!	Hello World!

Функции манипулирования символами

Для работы с символьными строками используются следующие функции:

```
select
CONCAT('Hello ', 'world!') as concat_, --соединяет
SUBSTR('hello world!',1,5) as substr_, --обрезает до символа
LENGTH('hello world!') as length_, --количество
INSTR('Hello World!','W') as instr_, --ищет вхождение символа
LPAD('Hello',10,'*') as lpad_, --дополняет слева
RPAD('Hello',10,'*') as rpad_, --дополняет справа
REPLACE('Hello yap','yap','world!') as replace_, --заменяет
TRIM('      Hello World!      ') as trim_ --удаляет дубли справа и слева
from dual;
```

CONCAT_	SUBSTR_	LENGTH_	INSTR_	LPAD_	RPAD_	REPLACE_	TRIM_
Hello world!	hello	12	7	*****Hello	Hello*****	Hello world!	Hello World!

Числовые функции

- ROUND: округляет значение до указанного разряда
- TRUNC: сокращает значение до указанного разряда
- MOD: возвращает остаток от деления

```
select  
ROUND(100.1212120, 2) as round_,  
TRUNC(100.1212212, 2) as trunc,  
MOD(1600, 300) as mod_  
from dual;
```

ROUND_	TRUNC	MOD_
100.12	100.12	100

Работа с датами

- В базе данных Oracle даты хранятся во внутреннем числовом формате, включающем век, год, месяц, день, часы, минуты и секунды.
- По умолчанию даты выводятся в формате, заданном параметрами NLS(National Language Support – поддерживает местные языки)

```
select  
account_id, open_date  
from account  
where open_date = '15-JAN-00';
```

ACCOUNT_ID	OPEN_DATE
1	15-JAN-00
2	15-JAN-00
9	15-JAN-00

Использование функции SYSDATE

Функция SYSDATE возвращает текущую дату и время

```
select  
sysdate as curr_date,  
trunc(sysdate) as trunc_date,  
to_char(sysdate, 'dd.mm.yyyy') as char_date,  
to_date(sysdate, 'dd.mm.yyyy') as date_date  
from dual;
```

CURR_DATE	TRUNC_DATE	CHAR_DATE	DATE_DATE
08-MAR-21	08-MAR-21	08.03.2021	08-MAR-21

Использование арифметических операторов при работе с датами

```
select  
first_name, last_name,  
(sysdate - start_date) / 7 as week_,  
(sysdate - start_date) / 31 as month_,  
(sysdate - start_date) / 365 as year_  
from employee;
```

FIRST_NAME	LAST_NAME	WEEK_	MONTH_	YEAR_
Michael	Smith	1028.487979497354497354497354497354	232.239221176821983273596176821983273596	19.72442700405885337392186707255200405885

Функции манипулирования датами

MONTHS_BETWEEN	Число месяцев между двумя датами
ADD_MONTHS	Прибавление календарных месяцев к дате
NEXT_DAY	Следующий день после указанной даты
LAST_DAY	Последний день месяца
ROUND	Округление даты
TRUNC	Сокращение даты

Использование функций работы с датами

MONTHS_BETWEEN ('01-SEP-95', '11-JAN-94')	19.6774194
ADD_MONTHS ('31-JAN-96', 1)	'29-FEB-96'
NEXT_DAY ('01-SEP-95', 'FRIDAY')	'08-SEP-95'
LAST_DAY ('01-FEB-95')	'28-FEB-95'

Применение функций ROUND и TRUNC к датам

Предположим, что SYSDATE = '25-JUL-03':

Функция	Результат
ROUND (SYSDATE, 'MONTH')	01-AUG-03
ROUND (SYSDATE, 'YEAR')	01-JAN-04
TRUNC (SYSDATE, 'MONTH')	01-JUL-03
TRUNC (SYSDATE, 'YEAR')	01-JAN-03

Тема 2

- Функции TO_CHAR, TO_DATE и TO_NUMBER
- Вложенные функции
- Функции общего назначения:
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- Условные выражения:
 - CASE
 - DECODE

Использование функции TO_CHAR с датами

TO_CHAR (дата, 'модель_формата')

```
select  
to_char(sysdate, 'dd.mm.yyyy') as char_date_1,  
to_char(sysdate, 'dd-mm-yyyy') as char_date_2,  
to_char(sysdate, 'dd/mm/yyyy') as char_date_3,  
to_char(sysdate, 'yyyy') as char_date_4,  
to_char(sysdate, 'mm') as char_date_5,  
to_char(sysdate, 'dd') as char_date_6,  
to_char(sysdate, 'ww') as char_date_7,  
to_char(sysdate, 'yyyy-mm-dd') as char_date_8  
from dual;
```

CHAR_DATE_1	CHAR_DATE_2	CHAR_DATE_3	CHAR_DATE_4	CHAR_DATE_5	CHAR_DATE_6	CHAR_DATE_7	CHAR_DATE_8
08.03.2021	08-03-2021	08/03/2021	2021	03	08	10	2021-03-08

Использование функции TO_CHAR с числами

TO_CHAR (число, 'модель_формата')

```
select  
avail_balance,  
TO_CHAR(avail_balance, '$999999.00') as t_$,  
TO_CHAR(avail_balance, 'L999999.00') as t_L,  
TO_CHAR(avail_balance, '0999999.00') as t_0  
from account;
```

AVAIL_BALANCE	T_\$	T_L	T_MI
1057.75	\$1057.75	\$1057.75	0001057.75
500	\$500.00	\$500.00	0000500.00

Элемент	Описание	Пример	Результат
9	Числовая позиция (число 9-ок определяют ширину вывода),	999999	1234
0	Вывод начальных нулей	099999	001234
\$	Знак доллара	\$999999	\$1234
L	Символ местной валюты	L999999	FF1234
D	Возвращает символ разделения дробной части в указанной позиции. Значением по умолчанию является точка (.).	99D99	99.99
.	Определение позиции десятичной точки	999999.99	1234.00
G	Возвращает разделитель группы в указанной позиции. Можно определить несколько разделителей групп в модели числового формата.	9,999	9G999
,	Определение позиции запятой	999,999	1,234
MI	Знаки "минуса" справа (отрицательные величины)	999999MI	1234-
PR	Заключение отрицательных чисел в скобки	999999PR	<1234>
EEEE	Экспоненциальное представление (формат должен определять четыре E),	99.999EEEE	1.234E+03
U	Возвращает в указанной позиции "Евро" (или другую) двойную валюту	U9999	€1234
V	Умножает на 10 n раз (n = число 9-ок после V)	9999V99	123400
S	Возвращает отрицательное или положительное значение	S9999	-1234 или +1234
B	Выводит на экран нулевые значения как пробел, а не 0	B9999.99	1234.00

Использование функций TO_CHAR и TO_DATE с форматом даты RR

Чтобы найти сотрудников, нанятых на работу ранее 2005 года, используйте формат даты RR

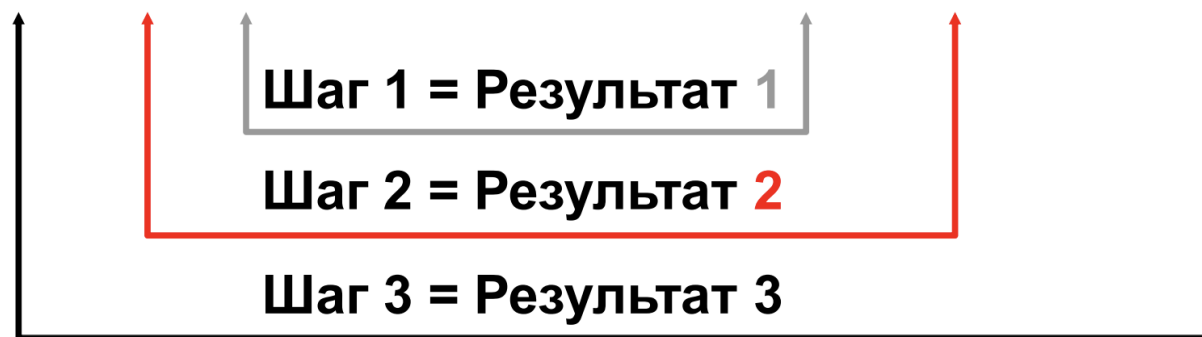
```
select  
first_name, last_name, start_date, to_char(start_date, 'dd.mm.yyyy') as char_date  
from employee  
where start_date < to_date('01-JAN-05', 'dd--mon-rr');
```

FIRST_NAME	LAST_NAME	START_DATE	CHAR_DATE
Michael	Smith	22-JUN-01	22.06.2001
Susan	Barker	12-SEP-02	12.09.2002

Вложенные функции

- Уровень вложенности однострочных функций не ограничен.
- Вложенные функции выполняются в направлении от нижнего уровня к верхнему.

F3 (F2 (F1 (col, arg1) , arg2) , arg3)



Вложенные функции

```
select  
city,  
instr(replace(substr(initcap(lower(city)),1,4),'n','a'),'W') as city_2  
from customer;
```

Функции общего назначения

Перечисленные ниже функции работают с любыми типами данных, в том числе с неопределенными значениями NULL:

- NVL (expr1, expr2)
- NVL2 (expr1, expr2, expr3)
- NULLIF (expr1, expr2)
- COALESCE (expr1, expr2, ..., exprn)

Функция NVL преобразует значение NULL в текущее значение:

- Типы данных, которые можно использовать, – это даты, строки и числа.
- Типы данных должны соответствовать:

```
select  
txn_id,  
nvl(15,txn_id) as nvl_txn_id,  
execution_branch_id,  
NVL(execution_branch_id,'1') as nvl_execution_branch_id  
from acc_transaction  
where execution_branch_id is null;
```

TXN_ID	NVL_TXN_ID	EXECUTION_BRANCH_ID	NVL_EXECUTION_BRANCH_ID
25	15	-	1
26	15	-	1
27	15	-	1

Использование функции NVL2

```
select  
first_name,  
superior_emp_id,  
nvl2(superior_emp_id, 'EMP_ID', 'ISNULL') as nvl2  
from employee  
where emp_id in ('1', '2');
```

FIRST_NAME	SUPERIOR_EMP_ID	NVL2
Michael	-	ISNULL
Susan	1	EMP_ID

Использование функции NULLIF

```
NULLIF( expr1, expr2 )
```

- Функция NULLIF возвращает NULL, если expr1 и expr2 равны.
- Функция NULLIF возвращает expr1, если expr1 и expr2 не равны.

```
select  
first_name,  
length(first_name) as length_1,  
last_name,  
length(last_name) as length_2,  
NULLIF(length(first_name), length(last_name)) as result  
from employee  
where first_name in ('Cindy', 'Frank', 'Theresa', 'Beth');
```

FIRST_NAME	LENGTH_1	LAST_NAME	LENGTH_2	RESULT
Cindy	5	Mason	5	-
Frank	5	Portman	7	5
Theresa	7	Markham	7	-
Beth	4	Fowler	6	4

Использование функции COALESCE

По сравнению с функцией NVL у функции COALESCE есть преимущество – она может принимать несколько альтернативных значений.

Если первое выражение не равно NULL, функция COALESCE вернет его; в противном случае функция COALESCE будет применена к оставшимся выражениям.

Использование функции COALESCE

```
select  
first_name,  
COALESCE(to_char(superior_emp_id), to_char(first_name), to_char('last_name') 'All_NO') as result  
from employee;
```

FIRST_NAME	RESULT
Michael	Michael
Susan	1
Robert	1

Условные выражения

Позволяют использовать в инструкциях SQL логику
IF-THEN-ELSE (ЕСЛИ...ТО...ИНАЧЕ)

Применяются два метода:

- выражение CASE
- функция DECODE

Выражение CASE

Облегчает реализацию условных запросов, выполняя функции операторов IF-THEN-ELSE:

```
CASE expr WHEN comparison_expr1 THEN return_expr1  
      [WHEN comparison_expr2 THEN return_expr2  
      WHEN comparison_exprn THEN return_exprn  
      ELSE else_expr]  
END
```

Использование выражения CASE

```
select
first_name,
title,
CASE title WHEN 'President' THEN 100000
          WHEN 'Treasurer' THEN 1000000
          WHEN 'Operations Manager' THEN 25000
          WHEN 'Loan Manage' THEN 56000
ELSE 30000 END salary
from employee;
```

```
select
first_name,
title,
CASE WHEN title = 'President' and first_name = 'Michael' THEN 100000
      WHEN title = 'Treasurer' and first_name = 'Robert' THEN 1000000
      WHEN title = 'Operations Manager' and first_name = 'Susan' THEN 25000
      WHEN title = 'Loan Manage' and first_name = 'John' THEN 56000
ELSE 30000 END salary
from employee;
```

FIRST_NAME	TITLE	SALARY
Michael	President	100000
Susan	Vice President	30000
Robert	Treasurer	1000000
Susan	Operations Manager	25000

Функция DECODE

Облегчает реализацию условных запросов, выполняя функции выражения CASE или операторов IF-THEN-ELSE:

```
DECODE(col|expression, search1, result1  
      [, search2, result2, ..., ]  
      [, default])
```


Использование функции DECODE

```
select
first_name,
title,
DECODE(title, 'President', 100000,
        'Treasurer', 1000000,
        'Operations Manager', 25000,
        'Loan Manage', 56000,
        56000
        ) as salary
from employee;
```

FIRST_NAME	TITLE	SALARY
Michael	President	100000
Susan	Vice President	56000
Robert	Treasurer	1000000

Агрегация данных и групповые функции

Тема 3

- Групповые функции:
 - Типы и синтаксис
 - Использование AVG, SUM, MIN, MAX, COUNT
 - Использование в групповых функциях ключевого слова DISTINCT
 - Значения NULL в групповых функциях
- Группирование строк:
 - Предложение GROUP BY
 - Предложение HAVING
- Вложенные групповые функции

Что такое групповые функции?

Групповые функции оперируют с наборами строк и выдают по одному результату на группу.

EMPLOYEES

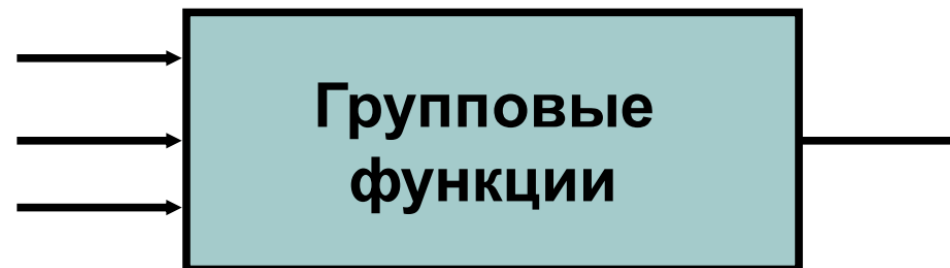
	DEPARTMENT_ID	SALARY
1	90	24000
2	90	17000
3	90	17000
4	60	9000
5	60	6000
6	60	4200
7	50	5800
8	50	3500
9	50	3100
10	50	2600
...		
18	20	6000
19	110	12000
20	110	8300

Максимальный
оклад в таблице
EMPLOYEES

MAX(SALARY)
24000

Типы групповых функций

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE



Использование функций AVG, SUM, MIN, MAX

Функции AVG и SUM можно использовать для числовых данных.

```
select  
round(avg(avail_balance),2) as avg_bal,  
sum(avail_balance) as sum_bal,  
min(avail_balance) as min_bal,  
max(avail_balance) as max_bal  
from account  
where open_emp_id = 10;
```

AVG_BAL	SUM_BAL	MIN_BAL	MAX_BAL
3051.62	42722.64	200	9345.55

Использование функций MIN и MAX

Функции MIN и MAX можно использовать с числовыми и символьными типами данных, а также с датами.

```
select  
max(open_date) as max_date,  
min(open_date) as min_date  
from account  
where open_emp_id = 10;
```

MAX_DATE	MIN_DATE
30-JUN-04	15-JAN-00

Использование функции COUNT

COUNT(*) возвращает число строк в таблице:

```
select  
count(emp_id) as c_emp_id  
from employee;
```

C_EMP_ID
18

COUNT(expr) возвращает число строк, для которых значение expr не равно NULL:

```
select  
count(superior_emp_id) as c_semp_id  
from employee;
```

C_SEMP_ID
17

Использование ключевого слова DISTINCT

- COUNT(DISTINCT expr) возвращает число уникальных и не равных NULL значений expr.
- Пример: чтобы показать число различных отделов, представленных в таблице EMPLOYEE:

```
select  
count(distinct title) as u_title,  
count(title) as c_title  
from employee;
```

U_TITLE	C_TITLE
7	18

Групповые функции и значения Null

Групповые функции игнорируют значения NULL в столбцах:

1 **SELECT** **AVG (commission_pct)**
FROM employees

AVG(COMMISSION_PCT)	
1	0.2125

Функция NVL заставляет групповые функции учитывать значения NULL:

2 **SELECT** **AVG (NVL (commission_pct, 0))**
FROM employees

Создание групп данных: синтаксис предложения GROUP BY

```
SELECT      столбец, групповая_функция (столбец)  
FROM        таблица  
[WHERE      условие]  
[GROUP BY   выражение_группировки]  
[ORDER BY   столбец] ;
```

Строки таблицы можно объединить в более компактные группы при помощи предложения GROUP BY.

Использование предложения GROUP BY

Все столбцы из списка SELECT, не используемые групповыми функциями, должны быть перечислены в предложении GROUP BY.

```
select  
product_cd,  
sum(avail_balance) as sum_bal  
from account  
group by product_cd;
```

PRODUCT_CD	SUM_BAL
MM	34090.28
SAV	3711.52
BUS	18691.1
SBL	100000
CHK	146016.02
CD	39000

Использование предложения GROUP BY

Столбец, перечисленный в предложении GROUP BY, не обязательно должен присутствовать в списке SELECT.

```
select  
sum(avail_balance) as sum_bal  
from account  
group by product_cd;
```

SUM_BAL
34090.28
3711.52
18691.1
100000
146016.02
39000

Использование предложения GROUP BY с несколькими столбцами

```
select
open_emp_id,
product_cd,
sum(avail_balance) as sum_bal
from account
group by open_emp_id, product_cd
order by open_emp_id;
```

OPEN_EMP_ID	PRODUCT_CD	SUM_BAL
1	CD	23000
1	CHK	1564.32
1	MM	29665.28
1	SAV	1535.54
10	BUS	18691.1
10	CD	16000
10	CHK	6631.54
10	SAV	1400

Недопустимые запросы при использовании групповых функций

Любой столбец или выражение из списка `SELECT`, которые не являются агрегатной функцией, должны быть перечислены в предложении `GROUP BY`:

```
SELECT department_id, COUNT(last_name)
FROM employees
```

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

Чтобы для каждого идентификатора отдела `department_id` подсчитать число фамилий, необходимо добавить предложение `GROUP BY`.

```
SELECT department_id, job_id, COUNT(last_name)
FROM employees
GROUP BY department_id ;
```

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

Либо добавьте в `GROUP BY` столбец `job_id`, либо удалите столбец `job_id` из списка `SELECT`.

Недопустимые запросы при использовании групповых функций

- Для ограничения групп нельзя использовать предложение WHERE.
- Для ограничения групп следует использовать предложение HAVING.
- Групповые функции в предложении WHERE использовать нельзя.

```
select
product_cd,
avg(avail_balance) as sum_bal
from account
where avg(avail_balance) > 0
group by product_cd
order by open_emp_id;
```

Ограничение групповых результатов при помощи предложения HAVING

Когда используется предложение HAVING, сервер Oracle ограничивает группы следующим образом:

- 1.Строки объединяются в группы.
- 2.Применяется групповая функция.
- 3.Отображаются группы, соответствующие предложению HAVING

```
SELECT      столбец, групповая_функция
FROM        таблица
[WHERE      условие]
[GROUP BY   выражение_group_by]
[HAVING     условие_группирования]
[ORDER BY   столбец] ;
```


Использование предложения HAVING

```
select  
product_cd,  
max(avail_balance) as max_bal  
from account  
group by product_cd  
having max(avail_balance) > 10000;
```

PRODUCT_CD	MAX_BAL
SBL	50000
CHK	38552.05

Вложенные групповые функции

```
select  
max(avg(avail_balance)) as max_bal  
from account  
group by product_cd;
```

MAX_BAL
50000