4

Использование функций преобразования и условных выражений

Цели

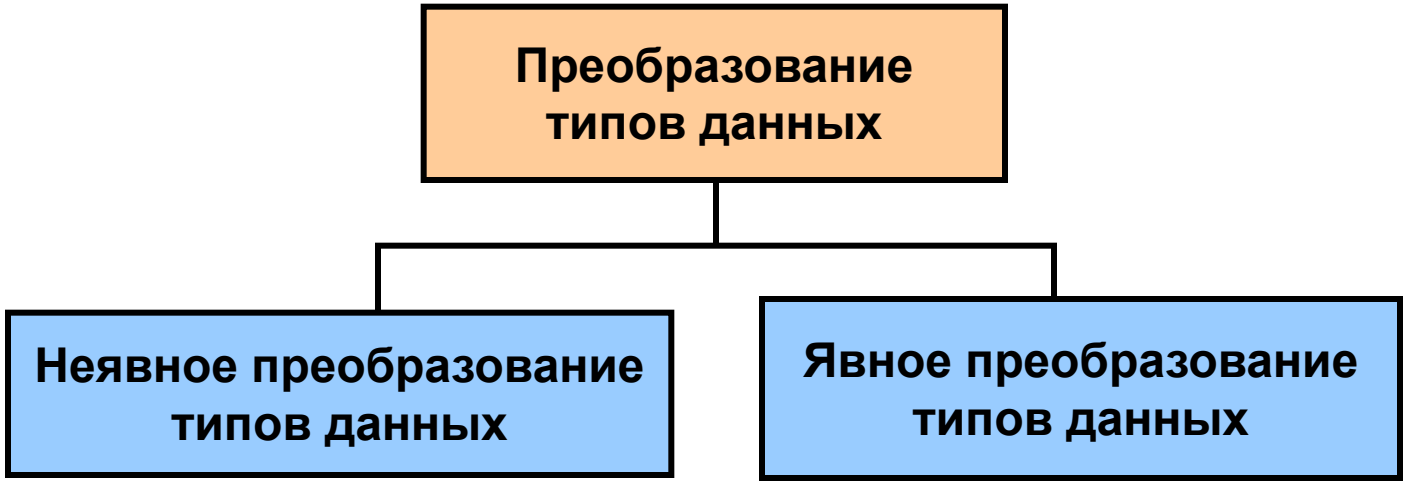
Изучив материал этого занятия, вы сможете:

- Описывать различные типы функций преобразования, доступных в SQL
- Использовать функции преобразования `TO_CHAR`, `TO_NUMBER` и `TO_DATE`
- Применять условные выражения в инструкции `SELECT`

План занятия

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

Функции преобразования



Неявное преобразование типов данных

В выражениях сервер Oracle может автоматически преобразовывать данные следующих типов:

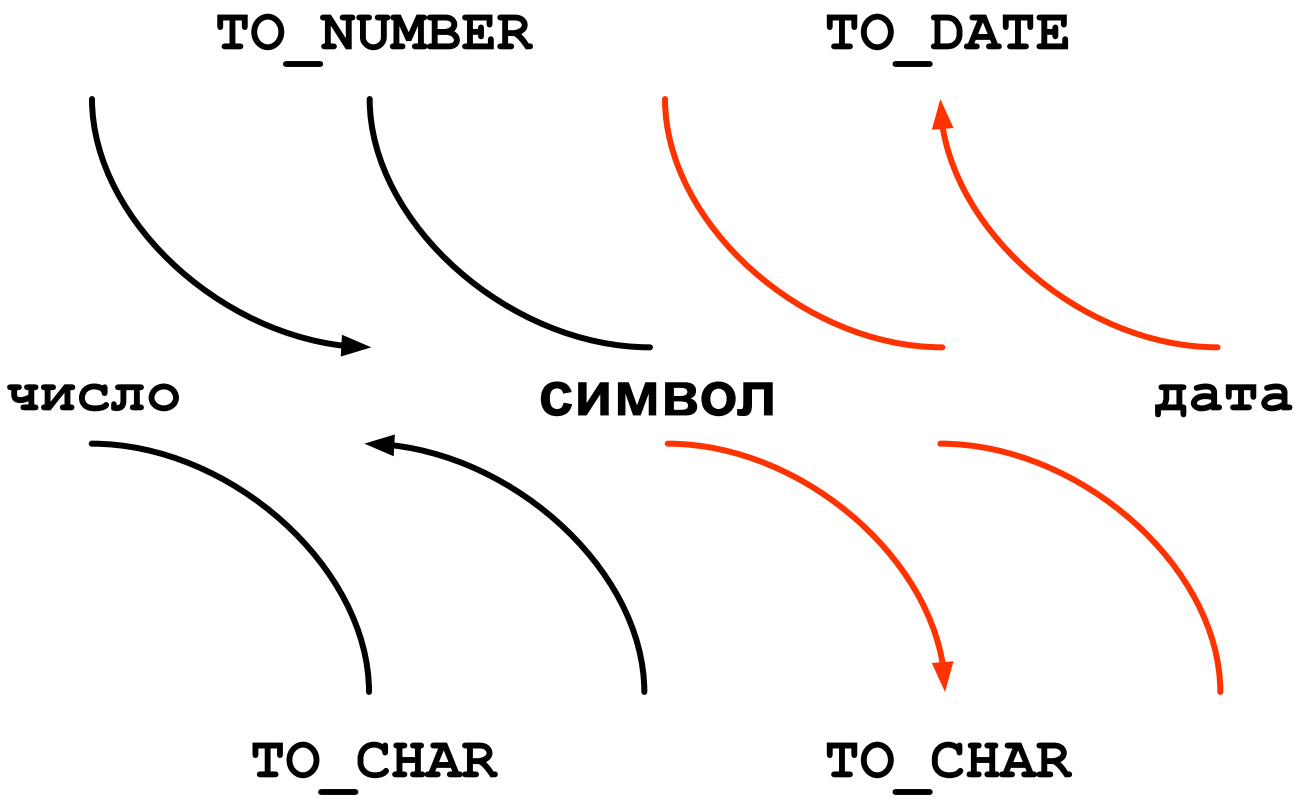
Из	В
VARCHAR2 или CHAR	NUMBER
VARCHAR2 или CHAR	DATE

Неявное преобразование типов данных

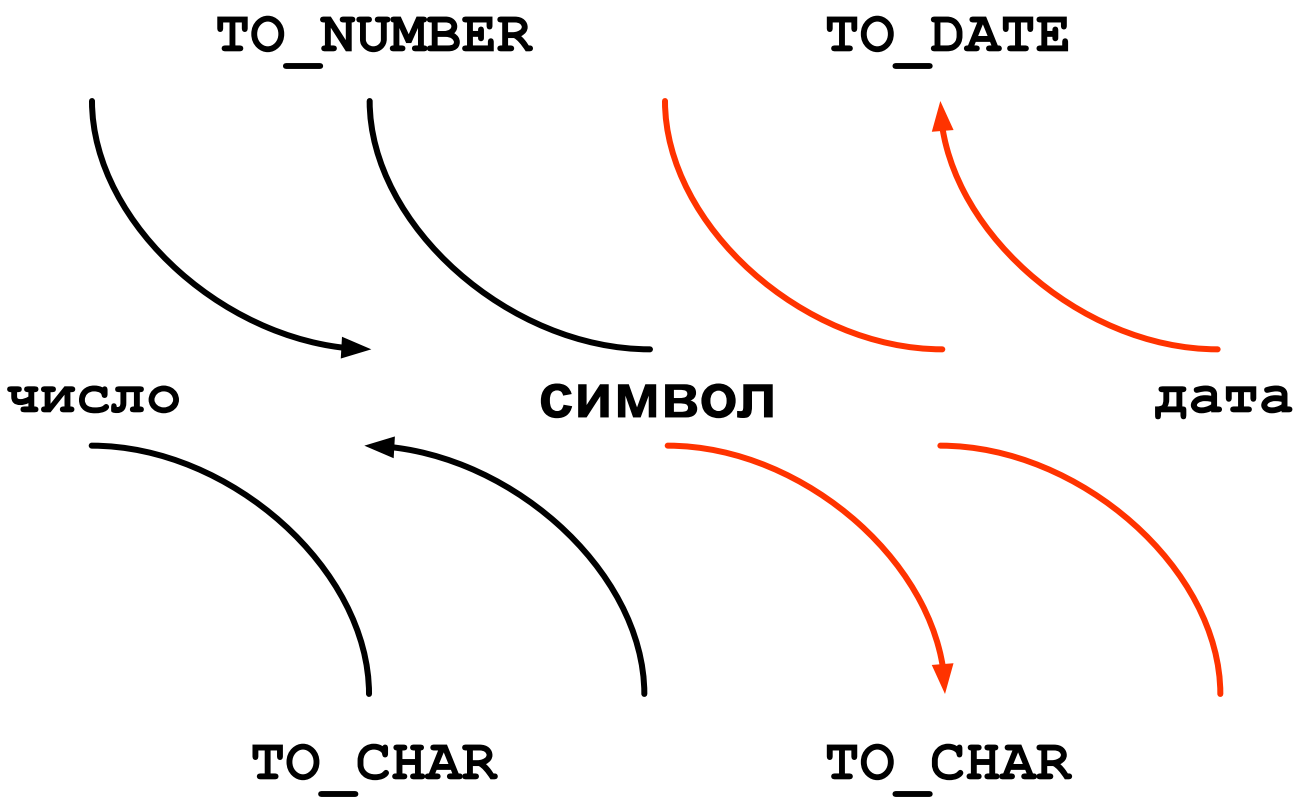
При оценке выражений сервер Oracle может автоматически преобразовывать данные следующих типов:

Из	В
NUMBER	VARCHAR2 или CHAR
DATE	VARCHAR2 или CHAR

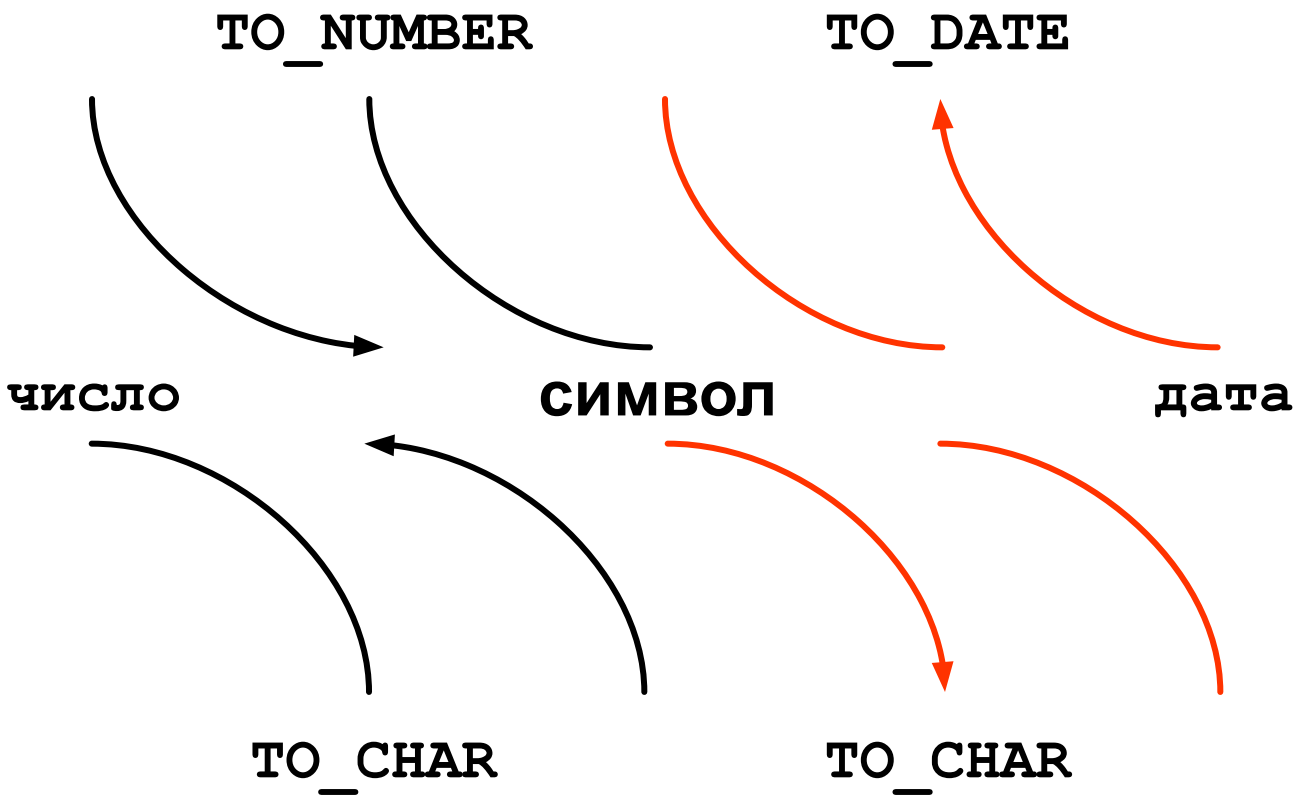
Явное преобразование типов данных



Явное преобразование типов данных



Явное преобразование типов данных



План занятия

- Неявное и явное преобразование типов данных
- 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')
```

Модель формата:

- Должна быть заключена в одиночные кавычки
- Чувствительна к регистру
- Может содержать любой допустимый элемент форматирования даты
- Располагает элементом `fm` для удаления заполняющих пробелов или начальных нулей
- Отделяется от значения даты при помощи запятой

Элементы модели формата даты

Элемент	Результат
YYYY	Полный порядковый номер года
YEAR	Год, прописью (на английском языке)
MM	Двузначный номер месяца
MONTH	Полное наименование месяца
MON	Трехбуквенное сокращение для месяца
DY	Трехбуквенное сокращение для дня недели
DAY	Полное наименование дня недели
DD	Номер дня месяца

Элементы модели формата даты

- Элементы времени форматируют ту часть даты, которая определяет время:

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

- При добавлении строк символов заключайте их в двойные кавычки:

`DD "of" MONTH`	`12 of OCTOBER`

- Суффиксы чисел для представления их прописью:

`ddspth`	`fourteenth`

Использование функции TO_CHAR с датами

```
SELECT last_name,
       TO_CHAR(hire_date, 'fmDD Month YYYY')
       AS HIREDATE
FROM   employees;
```

	LAST_NAME	HIREDATE
1	King	17 June 1987
2	Kochhar	21 September 1989
3	De Haan	13 January 1993
4	Hunold	3 January 1990
5	Ernst	21 May 1991
6	Lorentz	7 February 1999
7	Mourgos	16 November 1999
8	Rajs	17 October 1995
9	Davies	29 January 1997
10	Matos	15 March 1998

...

19	Higgins	7 June 1994
20	Gietz	7 June 1994

Использование функции TO_CHAR с числами

```
TO_CHAR(число, 'модель_формата')
```

Вот некоторые элементы форматирования, которые можно использовать с функцией TO_CHAR для отображения чисел в виде строк символов:

Элемент	Результат
9	Представление числа
0	Принудительное отображение нуля
$	Размещение плавающего символа
L	Использование плавающего локального символа валюты
.	Печать десятичной точки
,	Печать запятой как разделителя групп разрядов

Использование функции TO_CHAR с числами

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```

	SALARY
1	$6,000.00

Использование функций TO_NUMBER и TO_DATE

- Преобразование строки символов в числовой формат при помощи функции TO_NUMBER:

```
TO_NUMBER(строка[, 'модель_формата'])
```

- Преобразование строки символов в формат даты при помощи функции TO_DATE:

```
TO_DATE(строка[, 'модель_формата'])
```

- Для этих функций предусмотрен модификатор fx. Этот модификатор определяет точное соответствие символьного аргумента и модели формата даты функции TO_DATE.

Использование функций TO_CHAR и TO_DATE с форматом даты RR

Чтобы найти сотрудников, нанятых на работу ранее 1990 года, используйте формат даты RR, который даст одинаковые результаты независимо от того, выполнялась ли команда в 1999 году или сейчас:

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM  employees
WHERE hire_date < TO_DATE('01-Jan-90','DD-Mon-RR');
```

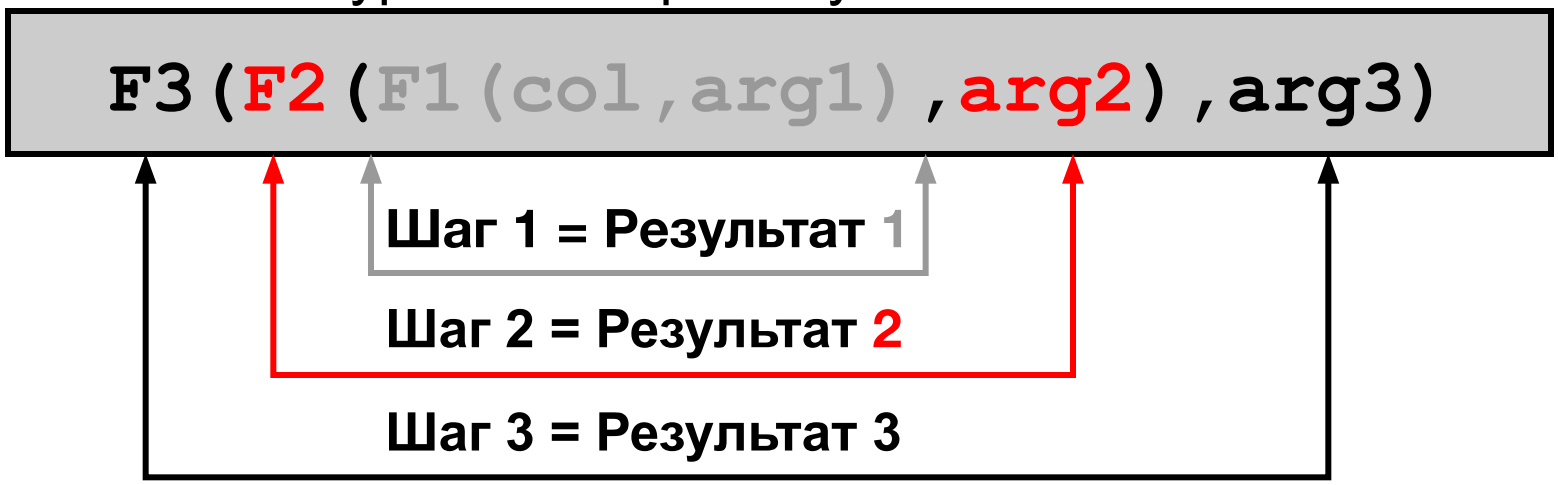
	LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1	King	17-Jun-1987
2	Kochhar	21-Sep-1989
3	Whalen	17-Sep-1987

План занятия

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

Вложенные функции

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



Вложенные функции

```
SELECT last_name,
  UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))
FROM   employees
WHERE  department_id = 60;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US

План занятия

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

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:
  - `NVL(commission_pct,0)`
  - `NVL(hire_date,'01-JAN-97')`
  - `NVL(job_id,'Пока без должности')`

Использование функции NVL

```
SELECT last_name, salary, NVL(commission_pct, 0),
    (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

1

2

	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	King	24000	0	288000
2	Kochhar	17000	0	204000
3	De Haan	17000	0	204000
4	Hunold	9000	0	108000
5	Ernst	6000	0	72000
6	Lorentz	4200	0	50400
7	Mourgos	5800	0	69600
8	Rajs	3500	0	42000
9	Davies	3100	0	37200
10	Matos	2600	0	31200
11	Vargas	2500	0	30000
12	Zlotkey	10500	0.2	151200

...

1

2

Использование функции NVL2

```
SELECT last_name,  salary, commission_pct,
       NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

1

2

	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM

1

2

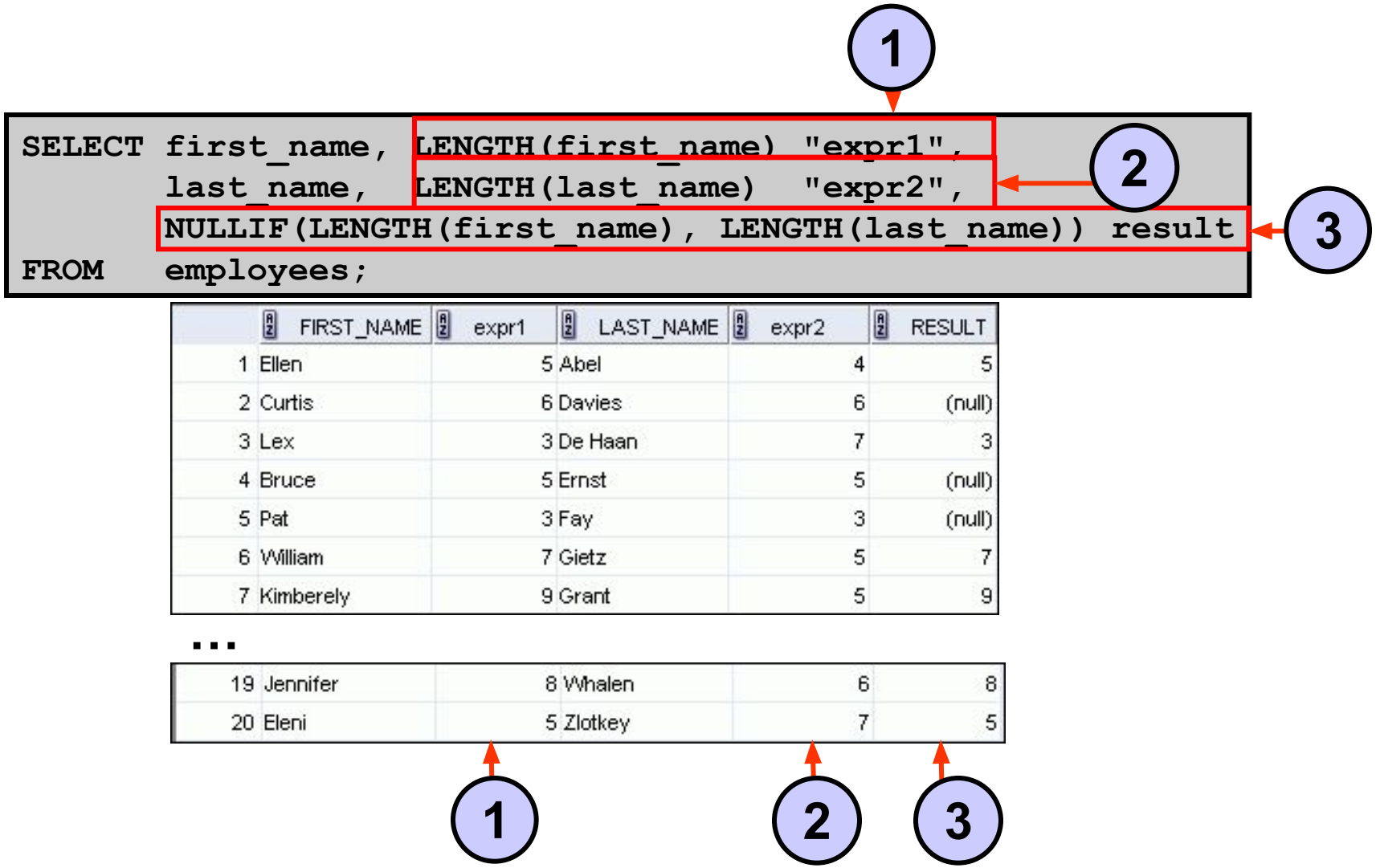
Использование функции NULLIF

```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name,  LENGTH(last_name)  "expr2",
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM   employees;
```

	FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
1	Ellen	5	Abel	4	5
2	Curtis	6	Davies	6	(null)
3	Lex	3	De Haan	7	3
4	Bruce	5	Ernst	5	(null)
5	Pat	3	Fay	3	(null)
6	William	7	Gietz	5	7
7	Kimberely	9	Grant	5	9

...

19	Jennifer	8	Whalen	6	8
20	Eleni	5	Zlotkey	7	5



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 last_name, employee_id,
COALESCE(TO_CHAR(commission_pct),TO_CHAR(manager_id)
,
    'No commission and no manager')
FROM employees;
```

	LAST_NAME	EMPLOYEE_ID	COALESCE(TO_CHAR(COMI
1	King	100	No commission and no manager
2	Kochhar	101	100
3	De Haan	102	100
4	Hunold	103	102
5	Ernst	104	103
6	Lorentz	107	103
7	Mourgos	124	100
8	Rajs	141	124

...

12	Zlotkey	149	.2
13	Abel	174	.3
14	Taylor	176	.2
15	Grant	178	.15
16	Whalen	200	101

...

План занятия

- Неявное и явное преобразование типов данных
- Функции `TO_CHAR`, `TO_DATE` и `TO_NUMBER`
- Вложенные функции
- Функции общего назначения:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Условные выражения:
  - `CASE`
  - `DECODE`

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

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

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE       salary END     "REVISED_SALARY"
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
5	Ernst	IT_PROG	6000	6600
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
9	Davies	ST_CLERK	3100	3565
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
...				

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 last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
...				

Использование функции DECODE

Отобразить ставку налога, применимую к каждому из сотрудников отдела 80:

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                          0, 0.00,
                          1, 0.09,
                          2, 0.20,
                          3, 0.30,
                          4, 0.40,
                          5, 0.42,
                          6, 0.44,
                             0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

Заключение

На этом занятии были изучены следующие темы:

- 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`

Упражнение 4: обзор

Упражнение охватывает следующие темы:

- Создание запросов, использующих функции `TO_CHAR`, `TO_DATE` и другие функции обработки дат
- Создание запросов, использующих условные выражения, такие как `DECODE` и `CASE`