3 Функции Oracle SQL

3.1 Для чего нужны функции. Однострочные функции

Функции являются очень мощным средством SQL и используются в следующих целях:

- Вычисления с данными.
- Изменение отдельных единиц данных.
- Управление выводом групп строк.
- Изменение формата данных в столбцах.
- Преобразование типов данных в столбцах
- Функции SQL принимают один или несколько аргументов и возвращают одно или несколько значений.

Отметим, что в Oracle предусмотрены два типа функций: встроенные функции Oracle и пользовательские функции, которые пользователю необходимо создавать как объекты в базе данных. В этом разделе речь пойдет только про встроенные функции Oracle. Пользовательские функции будут рассматриваться в разделе, посвященном PL/SQL.
Встроенные функции Oracle разделяются на несколько категорий:

- однострочные функции (single-row);
- агрегатные функции, например, MAX(), MIN(), AVG() и т.п.
- аналитические функции, например, RANK(), NTILE(), LAG() и т.п.
- объектные функции (REF(), MAKE_REF(), VALUE());
- моделирующие функции (CV(), PREVIOUS(), PRESENTV()).

В этом разделе будут рассмотрены только однострочные функции. Их отличительной особенностью является то, что они работают только с одной строкой и возвращают по одному результату на строку.

Однострочные функции манипулируют элементами данных.

- Принимают аргументы и возвращают одно значение.
- Работают с каждой строкой, возвращаемой запросом.
- Возвращают один результат на строку.
- Изменяют тип данных.
- Могут быть вложенными.

Синтаксис:

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

Однострочные функции используются для работы с элементами данных. Они принимают один или несколько аргументов и возвращают по одному значению для каждой строки, выдаваемой запросом.
Аргументом может быть:

- Константа, заданная пользователем
- Значение переменной
- Имя столбца
- Выражение

3.2 Список функций

В этом разделе для справки приведен список однострочных функций с примерами их применения.

Арифметические функции

ABS(n)	Абсолютное значение числа n	ABS(-5)=5
CEIL(n)	Наименьшее целое, большее или равное числу n.	CEIL(5.1)=6
FLOOR(n)	Наибольшее целое, меньшее или равное числа n	FLOOR(5.9)=5
MOD(n, m)	Остаток от деления m на n, или m, если n=0	MOD(5,4)=1
ROUND(n [,m])	Число n, округленное до m десятичных знаков	ROUND(5.2611,1)=5.3
SIGN(n)	Знак числа n. Если n<0 => -1, n=0 => 0, n>0 => 1	SIGN(-5)=-1
Символьные функции		
CHR(n)	Символ из набора символов, установленного для базы данных, в соответствии с двоичным эквивалентом числа n	CHR(10)= код клавиши ‘enter’
CONCAT(char, char2)	Конкатенация строк char1 и char2	CONCAT(‘AB’,’CD’) =’ABCD’
INITCAP(char)	Строка, в которой первые буквы слов переведены в верхний регистр., остальные – в нижний	INITCAP ('abc cde') =’Abc Cde’
LOWER(char)	Переводит строку в нижний регистр	LOWER(’ABCDE’)= ‘abcde’
LPAD(char1, n [,char2]) RPAD(char1, n [,char2])	Выравнивает строку вправо(влево) и выводит ее длиной n. Если остается свободное место, оно заполняется строкой char2 с повторениями.	LPAD(‘ABCD’,8,’*+’) =’*+*+ABCD’
LTRIM(char [,set]) RTRIM(char [,set])	Удаляются левые(правые) пробелы или символы из последовательности set	LTRIM(‘aaaabc’, ’a’)=’bc’
REPLACE(char, search_string [,replacement_string])	Все вхождения в строке char подстроки search_string заменяет на replacement_string либо удаляляются, если replacement_string опущена	REPLACE(‘abcbdb’,’b’,’1’)=’a1c1d1’
SUBSTR(char, m [,n])	Подстрока строки char начиная с символа m длиной n. Если m отрицательно, то позиция отсчитывается от конца строки	SUBSTR(‘ABCDEF’,2,3)=’BCD’
SUBSTRB(char, m [,n])	Аналогична предыдущей, но аргументы указывают позицию и длину в байтах	
UPPER(char)	Переводит строку в верхний регистр	UPPER(‘abc’)=’ABC’
ASCII(char)	Десятичное значение первого байта аргумента char	ABCII(‘A’)=65
INSTR(char1, char2 [,n [,m]])	Номер позиции подстроки char2 в строке char1, начиная с позиции n и вхождения m. Если n<0, то позиция отсчитывается с конца. По умолчанию n и m =1.	INSTR(‘ABCDBCBC’, ’BC’, 3, 2)=7
LENGTH(char)	Длина строки в символах	LENGTH(‘ABC’)=3
LENGTHB(char)	Длина строки в байтах	
Функции даты/времени		
ADD_MONTHS(d, n)	Добавляет или вычитает n месяцев к дате	ADD_MONTHS(sysdate,4)=’02.02.2002’
LAST_DAY(d)	Последний день месяца, указанного датой	LAST_DAY(sysdate)= ’31.13.2001’
MONTHS_BETWEEN (d1, d2)	Количество месяцев между двумя месяцами. Если d1<d2, то возвращается отрицательное значение	MONTHS_BETWEEN(sysdate, ADD_MONTHS(sysdate,5))=-5
NEW_TIME(d, z1, z2)	Дата и время в регионе, указанном аргументом z2, для даты d в регионе z1.	

NEXT_DAY(d, char)	Дата рабочего дня недели ближайшего после указанной даты	NEXT_DAY('28-DEC-2001', 'Tuesday')='01-JAN-2002'
SYSDATE	Текущая дата	
Функции преобразования типа		
CHARTORAWID(char)	Преобразование значения типа char в значение типа rowid	
ROWIDTOCHAR(rowid)	Преобразование значения типа rowid в значение типа char.	
TO_CHAR(d [,fmt [,'nlsparams']])	Преобразует дату d типа DATE в значение типа VARCHAR2 в формате, определенном форматом даты fmt. Параметр nlsparams определяет язык, на котором возвращается дата	
TO_CHAR(n [,fmt [,'nlsparams']])	Преобразует значение n типа NUMBER в значение типа VARCHAR2, используя необязательный аргумент формата числа fmt.	
TO_DATE(char [,fmt [,'nlsparams']])	Преобразует значение char типа CHAR или VARCHAR2 в значение типа DATE. Аргумент fmt должен указывать формат даты, содержащийся в char, в том случае, если это формат не по умолчанию	
TO_NUMBER(char [,fmt [,'nlsparams']])	Преобразует значение char типа CHAR или VARCHAR2 в значение типа NUMBER	
Функции идентификации пользователя		
UID	Значение, идентифицирующее текущего пользователя	select uid, user from dual
USER	Текущий пользователь, как значение типа varchar2	
Функции агрегирования		
AVG([DISTINCT|ALL] n)	Среднее значение для n	AVG(summa)
COUNT({*|[DISTINCT|ALL] expr})	Количество строк в запросе	COUNT(*)
MAX([DISTINCT|ALL] expr)	Максимальное значение выражения	MAX(summa)
MIN([DISTINCT|ALL] expr)	Минимальное значение выражения	MIN(summa)
SUM([DISTINCT|ALL] expr)	Сумма всех значений n в запросе	SUM(summa)
Функции для LOB-объектов		
EMPTY_BLOB() EMPTY_CLOB()	Возвращает пустой локатор LOB-объекта, который используется для инициализации LOB-переменной или в операторах INSERT или UPDATE для инициализации столбца таблицы или атрибута значением EMPTY	
BFILENAME('directory','filename')	Возвращает локатор для объекта типа BFILE, который связан с двоичным файлом операционной системы	
Функции для работы со ссылками на объекты		
DEREF(e)	Ссылка на объект	
MAKE_REF(table, key [,key…])	Создает REF для строки объектного вида, используя в качестве первичного ключа аргумент	

	key	

3.3 Символьные функции

3.3.1 Что такое символьные функции

Однострочные символьные функции принимают на входе символьные данные, а возвращают символьное или числовое значение. Символьные функции делятся на:

- Функции преобразования регистра символов
- Функции манипулирования символами

ФУНКЦИИ	**НАЗНАЧЕНИЕ**
LOWER(column | expression)	Преобразует алфавитные символы в нижний регистр.
UPPER (column | expression)	Преобразует алфавитные символы в верхний регистр.
INITCAP (column | expression)	Преобразует символьные значения: первая буква каждого слова становится заглавной, а остальные • строчными.
CONCAT(column1 | expression, column2 | expression)	Присоединяет первое символьное значение ко второму, Эквивалентно оператору конкатенации (||).
SUBSTR (column | expression, m [, n])	Возвращает n символов символьного значения, начиная с символа m. Если m отрицательно, отсчет начинается с конца символьного значения. Если n отсутствует, возвращаются все символы до конца строки.
LENGTH(column | expression)	Возвращает количество символов в значении.
INSTR(column | expression, m)	Возвращает номер позиции указанного символа.
LPAD (column | expression, n, ‘string’)	Дополняет символьное значение, выровненное справа, заданными символами 'string' до длины n.

3.3.2 Функции преобразования регистра символов

Преобразование регистра для символьных строк:

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

Три функции преобразования регистра символов - это LOWER, UPPER и INITCAP.

- LOWER: Преобразует строку символов верхнего регистра или обоих регистров в символы нижнего регистра.
- UPPER: Преобразует строку символов нижнего регистра или обоих регистров в символы верхнего регистра.

- INITCAP: Преобразует первую букву каждого слова в заглавную, а остальные буквы - в строчные.

Пример

```
SELECT  'Должность сотрудника '|| INITCAP (fio) ||' - '|| LOWER(dol) AS
"Должность" FROM inspector;
```

```
Должность
-----------------------------------------------------------------
Должность сотрудника Лютикова З.С. - инспектор
Должность сотрудника Азарченко Л.М. - инспектор
Должность сотрудника Чабановская В.С. - нач.отдела
Должность сотрудника Кудрявцев А.А. - инспектор
Должность сотрудника Савилов Ю.Н. - нач.отдела
Должность сотрудника Смирнова Т.Н. - инспектор
Должность сотрудника  - инспектор
Должность сотрудника Чайка Н.Ф. - нач.отдела
```

*Пример.*Вывод номера служащего, фамилии и номера отдела для служащего по фамилии Савилов Ю.Н.

```
SQL>  SELECT    kodinsp, fio, otdel
 2    FROM      inspector
 3    WHERE     fio = 'савилов ю.н.';
no rows selected.
```

```
SQL>  SELECT    kodinsp, fio, otdel
 2    FROM      inspector
 3    WHERE     Lower(fio) = 'савилов
               ю.н.';
```

```
KODI  FIO                  OT
------- -------------------- -----
5611    Савилов Ю.Н.         56
```

Предложение WHERE в первой команде SQL задает фамилию служащего -"савилов ю.н.". Т.к. все данные в таблице inspector хранятся в разных символах , совпадения с фамилией " савилов ю.н " не будет, вследствие чего ни одна строка не будет выбрана.
Предложение WHERE во второй команде SQL указывает, что фамилия служащего из таблицы inspector должна быть преобразована в строчные буквы и только после этого сравниваться с фамилией. Т.к. обе фамилии теперь представлены в символах нижнего регистра, совпадение будет обнаружено и будет выбрана одна строка.
Следующее предложение WHERE даст такой же результат:

```
… WHERE fio = 'Савилов Ю.Н.'
```

В выходных данных фамилия выглядит так, как хранится в базе данных. Чтобы вывести фамилию с заглавной первой буквой, используйте функцию INITCAP в списке выбора команды SELECT.

```
SELECT       kodinsp, initcap(fio), otdel

FROM         inspector

WHERE        Lower(fio) = 'савилов ю.н.';
```

3.3.3 Функции манипулирования символьными строками

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

CONCAT, SUBSTR, LENGTH, INSTR и LPAD - это пять функций манипулирования символами, которые обсуждаются на этом занятии.

- CONCAT: соединяет значения. Для функции CONCAT можно использовать не более двух параметров.
- SUBSTR: возвращает подстроку заданной длины.
- LENGTH: возвращает количество символов в виде числового значения.
- INSTR: возвращает номер позиции указанного символа.
- LPAD: дополняет символьное значение, выровненное справа, до заданной длины.

Примечание: функция манипулирования символами RPAD дополняет до нужной длины символьное значение, выровненное слева.

Пример

```
SQL>  SELECT     fio, CONCAT(fio, dol), LEHGTH(fio),
  2              INSTR(fio, 'a')
  3   FROM       inspector
  4   WHERE      substr(dol, 1,4) = 'Инсп';
```

FIO	CONCAT(FIO,DOL)	LENGTH(FIO)	INSTR(FIO,'A')
Лютикова З.С.	Лютикова З.С.Инспектор	13	8
Азарченко Л.М.	Азарченко Л.М.Инспектор	14	3
Кудрявцев А.А.	Кудрявцев А.А.Инспектор	14	0
Смирнова Т.Н.	Смирнова Т.Н.Инспектор	13	8

В этом примере для всех служащих следующие данные: фамилия и должность служащего, соединенные в одно целое, длина фамилии и номер позиции буквы а в фамилии.

Пример.

Измените команду SQL так, чтобы получить данные по служащим, чьи фамилии заканчиваются буквой Н.

```
SQL>  SELECT     fio, CONCAT(fio, dol), LENGTH(fio),
  2              INSTR(fio, 'a')
  3   FROM       inspector
  4   WHERE      substr(fio, -2,1) = 'Н';
```

FIO	CONCAT(FIO,DOL)	LEN	INS
Савилов Ю.Н.	Савилов Ю.Н.Нач.отдела	12	2
Смирнова Т.Н.	Смирнова Т.Н.Инспектор	13	8

Лабораторная работа 3.1 Работа со строковыми функциями

Задание 1:

Вам потребовалось создать для каждого сотрудника идентификатор, который должен выглядеть как 3 первые символа имени плюс два первых символа фамилии. Все символы этого идентификатора должны быть представлены в верхнем регистре.
Напишите запрос, который возвращал бы из таблицы hr.employees информацию об имени и фамилии сотрудника, а также идентификатор сотрудника в соответствии с поставленными условиями. Результат выполнения запроса должен быть таким, как представлено на рис. 3.1-1.

	Имя	Фамилия	Идентификатор
1	Ellen	Abel	ELLAB
2	Sundar	Ande	SUNAN
3	Mozhe	Atkinson	MOZAT
4	David	Austin	DAVAU
5	Hermann	Baer	HERBA
6	Shelli	Baida	SHEBA
7	Amit	Banda	AMIBA
8	Elizabeth	Bates	ELIBA

3.4 Числовые функции

- ROUND: Округляет значение до заданной точности
  ROUND(45.926, 2) ————> 45.93
- TRUNC: Усекает значение до заданного количества десятичных знаков
  TRUNC(45.926, 2) ————> 45.92
- MOD: Возвращает остаток от деления
  MOD (1600, 300) ————> 100

Числовые функции принимают на входе числовые данные и возвращают числовые значения. Некоторые из числовых функций описаны в этом разделе.

Функция	**Назначение**
ROUND(column | expression, n)	Округляет столбец, выражение или значение до n десятичных разрядов. Если n отрицательно, округляются разряды слева от десятичной точки.
TRUNC(column | expression, n)	Усекает столбец, выражение или значение до n десятичных разрядов, а если n опущено, то до целого. Если n отрицательно, усекаются до нуля разряды слева от десятичной точки.
MOD (m, n)	Возвращает остаток от деления m на n.

3.4.1 Функция ROUND

Пример

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

ROUND(4 5. 92 3,2)	ROUND(4 5. 92 3,0)	ROUND(4 5. 92 3, -1)
45.92	46	50

- Функция ROUND округляет столбец, выражение или значение до n десятичных разрядов. Если второй аргумент равен нулю или отсутствует, значение округляется до нуля десятичных разрядов. Если второй аргумент равен 2, значение округляется до двух десятичных разрядов. Если второй аргумент равен -2, значение округляется вверх до сотен (до целого числа с двумя нулями).
- Функция ROUND может использоваться и с функциями даты. Примеры приведены позже в этом занятии.

3.4.2 Функция 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, -l)
45.92	45	40

Функция TRUNC усекает столбец, выражение или значение до n десятичных разрядов. Аргументы функции TRUNC аналогичны аргументам функции ROUND. Если второй аргумент равен нулю или отсутствует, значение усекается до нуля десятичных разрядов. Если второй аргумент равен 2, значение усекается до двух десятичных разрядов. Если второй аргумент равен –2, значение усекается до сотен (до целого числа с двумя нулями). Функция TRUNC , как и функция ROUND, может использоваться с функциями даты.

3.4.4 Функция MOD

Вычисление остатка от деления оклада на комиссионные для всех служащих, работающих в должности продавца (salesman):
Функция MOD вычисляет остаток от деления value1 на value2.

3.5 Работа с датами

- Oracle хранит данные во внутреннем цифровом формате.
- - Век, год, месяц, число, часы, минуты, секунды
- По умолчанию дата выдается в формате
- DD-MON-YY (число - месяц-год)
- Функция SYSDATE возвращает текущие дату и время
- Для применение функции SYSDATE() используется фиктивная функция DUAL().

3.5.1 Формат дат Oracle

Oracle хранит даты во внутреннем числовом формате, представляющем столетие, год, месяц, число, часы, минуты и секунды.
По умолчанию выходной и входной форматы любой даты - DD-MON-YY(число-месяц-год).
Действительные даты Oracle лежат в диапазоне от 1 января 4712 до н.э. до 31 декабря 9999 н.э.

3.5.2 Функция SYSDATE()

SYSDATE - это функция, возвращающая текущие дату и время. SYSDATE можно использовать, как любое другое название столбца. Например, можно получить текущую дату путем выборки SYSDATE из таблицы. Обычно SYSDATE выбирается из фиктивной таблицы DUAL.
Владельцем таблицы DUAL является пользователь SYS. Обращаться к ней могут все пользователи. Таблица содержит один столбец (DUMMY) и одну строку со значением X. Таблица DUAL полезна, если какое-то значение необходимо получить только один раз - например, константу, псевдостолбец или выражение, которые не вычисляются по таблице с пользовательскими данными.
Пример.
Получите текущую дату с помощью таблицы DUAL.

```
SQL> SELECT    SYSDATE
 2   FROM      DUAL;
```

3.5.3 Арифметические операции с датами

- Результатом прибавления числа к дате и вычитания числа из даты является дата.
- Результатом вычитания одной даты из другой является количество дней, разделяющих эти даты.
- Прибавление часов к дате производится путем деления количества часов на 24.
- Т.к. в базе данных даты хранятся в виде чисел, с ними можно выполнять такие арифметические операции, как сложение и вычитание. Прибавлять и вычитать можно как числовые константы, так и даты.

Возможны следующие операции:

Операция	**Результат**	**Описание**
дата+число	Дата	Добавляет количество дней к дате
дата-число	Дата	Вычитает количество дней из даты
дата - дата	Количество дней	Вычитает одну дату из другой
дата+число/24	Дата	Прибавляет часы к дате

3.5.4 Использование арифметических операторов с датами

Пример

```
SQL>    SELECT        fio, (SYSDATE-begin_date)/7 WEEKS
  2     FROM          inspector
  3     WHERE         otdel = '56';
```

```
FIO                 WEEKS
------------------- ----------------------
Кудрявцев А.А.      360.571200396825
Савилов Ю.Н.        272.714057539683
Смирнова Т.Н.       246.142628968254
```

Пример сверху показывает вывод фамилий и количества отработанных недель всех служащих отдела 56. Из текущей даты вычитается дата найма, а затем для вычисления количества недель результат делится на 7.
Примечание: SYSDATE() - это функция SQL, возвращающая текущие дату и время, поэтому вы можете получить не такие результаты, как в примере.

3.5.5 Функции для работы с датами

ФУНКЦИЯ	**ОПИСАНИЕ**
MONTHS_BETWEEN	Число месяцев, разделяющих две даты
ADD_MONTHS	Добавление календарных месяцев месяцев к дате
NEXT_DAY	Ближайшая дата, когда наступит заданный день недели
LAST_DAY	Последняя дата текущего месяца
ROUND	Округление до целых суток
TRUNC	Отсечение части даты, обозначающей время

Эти функции работают с датами Oracle. Все функции для работы с датами возвращают значение типа DATE за исключением функции MONTHS_BETWEEN, возвращающей числовое значение.

- MONTHS_BETWEEN (date1, date2): вычисляет количество месяцев между date1 и date2. Результат может быть положительным или отрицательным. Если date1 позже date2, результат положителен; если date1 предшествует dare2, результат отрицателен. Дробная часть результата представляет часть месяца.
- ADD_MONTHS (date, n): прибавляет n календарных месяцев к date, n должно быть целым и может быть отрицательным.
- NEXT_DAY (date, ‘char’): возвращает дату после date, когда наступит заданный день недели (‘char’). ‘char’ может быть числом, представляющим день недели, или строкой символов.
- LAST_DAY (date): возвращает последнюю дату месяца, содержащего date.
- ROUND(date [,’fmt’]): возвращает дату date, округленную до единицы заданной моделью формата fmt. Если модель fmt отсутствует, date округляется до ближайшей даты.
- TRUNC(date [,’fmt’]): возвращает дату date, в которой время усечено до единицы, заданной моделью формата fmt. Если модель fmt отсутствует, date усекается до ближайшего дня.

Это только некоторые из имеющихся функций. Модели формата обсуждаются далее на этом занятии. Примерами моделей формата являются месяц (MONTH) или год (YEAR).

3.5.6 Примеры функций для работы с датами

- MONTHS_BETWEEN ('01-SEP-95','11-JAN.94') → 19.6774194
- ADD_MONTHS (‘11-JAN-94’, 6) → ‘11-JUL-94'
- NEXT_DAY ('01-SEP-95’,’FRIDAY') → ’08-SEP-95’
- LAST_DAY ('01-SEP-95') → '30-SEP-95'

Для всех служащих, проработавших менее 200 месяцев, выводится номер служащего, дата найма, количество отработанных месяцев, дата аттестации после 6 месяцев работы, дата первой пятницы после даты найма и последний день месяца, когда служащий был нанят на работу.

```
SQL>  SELECT     fio, begin_date,
  2              MONTHS_BETWEEN(SYSDATE, begin_date) TENURE,
  3              add MONTHS(begin_date, 6) REVIEW,
  4              NEXT_DAY(begin_date, 'FRIDAY'), LAST_DAY(begin_date)
  5   FROM       Inspector
  6   WHERE      MONTHS_BETWEEN (SYSDATE, begin_date) < 200;
```

FIO	BEGIN_DATE	TENURE	REVIEW	NEXT_DAY	LAST_DAY
Лютикова З.С.	11.11.1994	98.5484117383513	11.05.1995	18.11.1994	30.11.1994
Азарченко Л.М.	25.11.1995	86.0967988351254	25.05.1996	01.12.1995	30.11.1995
Чабановская В.С.	30.07.1995	89.9355085125448	30.01.1996	04.08.1995	31.07.1995
Кудрявцев А.А.	01.03.1996	82.8709923835125	01.09.1996	08.03.1996	31.03.1996
Савилов Ю.Н.	06.11.1997	62.7097020609319	06.05.1998	07.11.1997	30.11.1997
Смирнова Т.Н.	11.05.1998	56.5484117383513	11.11.1998	15.05.1998	31.05.1998
Чайка Н.Ф.	15.08.1994	101.419379480287	15.02.1995	19.08.1994	31.08.1994

Примеры функций для работы с датами

- ROUND(‘25-JUL-95’,'MONTH') → 01-AUG-95
- ROUND(‘25-JUL-95',’YEAR’) → 01-JAN-96
- TRUNC(‘25-JUL-95','MONTH’) → 01-JUL-95
- TRUNC(’25-JUL-95',’YEAR’) → 01-JAN-95

Функции ROUND и TRUNC могут использоваться для числовых значений и дат. Если они используются с датами, даты округляются или усекаются в соответствии с заданной моделью формата. Следовательно, можно округлять даты до ближайшего года или месяца.

Пример.
Сравните даты найма всех служащих, нанятых в 1997 г. Выведите на экран номер каждого служащего, дату и месяц найма с помощью функций ROUND и TRUNC.

```
SQL>  SELECT   fio, begin_date,
  2           ROUND(begin_date, 'MONTH'), TRUNC(begin_date,
              'MONTH')
  3   FROM     inspector
  4   WHERE    Begin_date like '%97';
```

FIO BEGIN_DATE ROUND(BEGIN_DATE,'MONTH') TRUNC(BEGIN_DATE,'MONTH')

Савилов Ю.Н.06.11.1997 01.11.1997 01.11.1997

Лабораторная работа 3.2 Применение функций для работы с датой/временем и арифметических функций

Задание:

Напишите запрос, который бы возвращал информацию об именах и фамилиях сотрудников из таблицы hr.employees, а также об дате приема каждого сотрудника на работу и количестве полных месяцев, которое каждый сотрудник отработал по настоящее время (настоящее время определяется по часам вашего компьютера).

Результат запроса должен выглядеть так, как представлено на рис. 3.2-1.

	Имя	Фамилия	Оклад	Дата приема на работу	Проработано месяцев
1	Donald	OConnell	2600	21-JUN-99	104
2	Douglas	Grant	2600	13-JAN-00	98
3	Jennifer	Whalen	4400	17-SEP-87	245
4	Michael	Hartstein	13000	17-FEB-96	144
5	Pat	Fay	6000	17-AUG-97	126
6	Susan	Mavris	6500	07-JUN-94	165
7	Hermann	Baer	10000	07-JUN-94	165

Примечание:

Значение в столбце "Проработано месяцев" у вас может быть другим, поскольку другим будет время на часах компьютера.

3.6 Функции преобразования

Помимо типов данных Oracle, столбцам таблиц в базе данных Oracle8-9i можно назначать типы данных ANSI, DB2 и SQL/DS. Но внутри системы сервер Oracle преобразует эти типы данных в типы данных Oracle8i.

В некоторых случаях сервер Oracle допускает данные какого-то типа там, где он ожидает данные другого типа. Это допускается, если сервер Oracle может автоматически привести данные к определенному типу. Такое преобразование типов данных может производиться неявно сервером Oracle или явно пользователем.

Неявные преобразования типов данных производятся в соответствии с правилами, изложенными ниже.

Явное преобразование типов данных производится с помощью функций преобразования. Функции преобразования преобразуют один тип данных в другой. Обычно эти функции следуют общепринятому правилу datatype TO datatype. Первый тип данных является входным, а второй - выходным.

Хотя неявное преобразование типов данных возможно, для упрощения чтения команд SQL рекомендуется делать это явно.

3.6.1 Неявное преобразование типов данных

Для операций присваивания Oracle может автоматически выполнять следующие преобразования:

Исходный формат	**Новый формат**
VARCHAR2 или CHAR	NUMBER
VARCHAR2 или CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

Для операций присваивания сервер Oracle может автоматически выполнить следующие преобразования:

- VARCHAR2 или CHAR в NUMBER
- VARCHAR2 или CHAR в DATE
- NUMBER в VARCHAR2
- DATE вVARCHAR2

Присваивание значения будет успешным, если сервер Oracle сможет привести тип данных значения, расположенного справа от оператора присваивания, к типу переменной.
При вычислении выражений Oracle может автоматически выполнять следующие преобразования:

Исходный формат	**Новый**
VARCHAR2 или CHAR	NUMBER
VARCHAR2 или CHAR	DATE

При вычислении выражений сервер Oracle может автоматически выполнить следующие преобразования:

- VARCHAR2 или CHAR в NUMBER
- VARCHAR2 или CHAR в DATE

Обычно сервер Oracle автоматически выполняет преобразования в выражениях, если необходимое преобразование не охвачено правилом преобразования типов данных для операций присваивания.
Преобразования из CHAR в NUMBER успешны только в случае, если символьная строка представляет действительное число. Преобразования CHAR в DATE успешны только в случае, если символьная строка имеет формат даты по умолчанию DD-MON-YY (число-месяц-год).

3.6.2 Явное преобразование типов данных

Для преобразования значения из одного типа данных в другой SQL предлагает три функции.

Функция	**Назначение**
TO_CHAR(number|date, [‘fmt ‘])	Преобразует число или дату в строку символов VARCHAR2 с моделью формата .fmt
TO_NUMBER	Преобразует символьную строку, содержащую цифры, в число

TO_DATE	Преобразует символьную строку с датой в значение даты согласно указанному fmt. (Если элемент fmt опущен, используется формат DD-MON-YY.)

Примечание: это только часть имеющихся функций преобразования. Более подробную информацию см. в руководстве Oracle Server SQL Reference.

3.6.3 Функция TO_CHAR с датами

TO_CHAR(date, 'fmt')

Модель формата:

- Должна быть заключена в апострофы. Различает символы верхнего и нижнего регистров.
- Может включать любые разрешенные элементы формата даты.
- Использует элемент fm для удаления конечных пробелов и ведущих нулей.
- Отделяется от значения даты запятой.

3.6.4 Вывод данных в заданном формате

До сих пор для вывода всех дат Oracle использовался стандартный формат DD-MON-YY (число-месяц-год). Функция TO_CHAR позволяет преобразовать дату из этого стандартного формата в формат, заданный пользователем.

Отметим некоторые моменты, связанные с применением функции TO_CHAR:

- Модель формата различает символы верхнего и нижнего регистров и должна быть заключена в апострофы.
- Модель формата может включать любой действительный элемент формата даты. Дата обязательно отделяется от модели формата запятой.
- Названия дней и месяцев на выводе автоматически заполняются до нужной длины пробелами.
- Для удаления вставленных пробелов и ведущих нулей используйте элемент fm режима заполнения (fill mode).
- Изменить ширину выходного символьного столбца можно с помощью команды COLUMN SQL*Plus.
- Ширина столбца по умолчанию - 80 символов.

3.6.5 Элементы формата даты

YYYY	**Полный год цифрами**
YEAR	Год прописью
MM	Двузначное цифровое обозначение месяца
MONTH	Полное название месяца
DY	Трехзначное алфавитное сокращенное название дня недели
DAY	Полное название дня недели

3.6.6 Примеры элементов формата даты

Элемент	**Описание**
SCC или CC	Век: если задано S, дате до нашей эры предшествует "-"
Годы в датах YYYY или	Год: если задано S, дате до нашей эры предшествует "-"

SYYYY	
YYY или YY или Y	3, 2 или 1 цифра года
Y, YY	Год с запятой в указанной позиции
IYYY, IYY, IY, I	4, 3, 2 или 1 цифра года по стандарту ISO
SYEAR или YEAR	Год прописью: если задано S, дате до нашей эры предшествует “-“
BC или AD	Индикатор даты до новой эры/новой эры
B.C. или A.D.	Индикатор даты до новой эры/новой эры с точками
Q	Квартал года
MM	Месяц двумя цифрами
MONTH	Название месяца, наполненное пробелами до 9 символов
MON	Трехбуквенное сокращенное название месяца
RM	Месяц римскими цифрами
WW или W	Неделя года или месяца
DDD или DD или D	День года, месяца или недели
DAY	Название дня, дополненное пробелами до 9 символов
DY	Сокращенное название дня из трех символов
J	Юлианская дата, количество дней с 31 декабря 4713 г. до н. э.

3.6.7 Элементы модели формата даты

- Элементы, которые задают формат части даты, обозначающей время.

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

- Символьные строки добавляются в кавычках.

DD "of “ MONTH	12 of OCTOBER

- Числовые суффиксы используются для вывода числительных прописью.

ddspth	fourteenth

3.6.8 Форматы времени

Для вывода времени и литералов, а также для преобразования цифровых дат в даты прописью пользуйтесь следующими форматами:

Элемент	**Описание**
AM или PM	Индикатор “до полудня/после полудня"
A.M. или P.M.	Индикатор "до полудня/после полудня” - обозначение с точками
HH или HH12 или HH24	Время суток в 12- или 24-часовом диапазоне
MI	Минуты (0-59)
SS	Секунды (0-59)
SSSSS	Секунды после полуночи (0-86399)

Другие форматы:

Элемент	**Описание**
/ . ,	Знаки пунктуации воспроизводятся в результате
"of the"	Строка, указанная в кавычках, воспроизводится в результате

3.6.9 Использование суффиксов для вывода чисел

Элемент	Описание
TH	Порядковый номер (напр., DDTH для 4TH)
SP	Число прописью (напр., DDSP для FOUR)
SPTH или THSP	Порядковое число прописью (напр., DDSPTH для FOURTH)

3.6.10 Функция TO_CHAR с датами

Пример

```
SQL>    SELECT     fio,
  2                TO_CHAR(begin_date, 'fmDD Month YYYY') HIREDATE
  3     FROM       inspector;
```

```
FIO                          HIREDATE
---------------------------- ------------------------
Лютикова З.С.                11 November 1994
Азарченко Л.М.               25 November 1995
Чабановская В.С.             30 July 1995
Кудрявцев А.А.               1 March 1996
Савилов Ю.Н.                 6 November 1997
Смирнова Т.Н.                11 May 1998

Чайка Н.Ф.                   15 August 1994
```

Команда SQL в приведенном примере выводит фамилии и даты найма всех служащих.

Пример.

Измените пример для вывода дат в формате "Seventh of February 1981 08:00:00 AM".

```
SQL>    SELECT   fio,
    2            TO_CHAR(begin_date, 'fmDdspth "of" Month YYYY fmHH: MI: SS AM')
    3            Begin_date
    4   FROM     inspector;
```

```
FIO                          BEGIN_DATE
---------------------------- ------------------------------------------------------
Лютикова З.С.                Eleventh of November 1994 12: 00: 00 AM
Азарченко Л.М.               Twenty-Fifth of November 1995 12: 00: 00 AM
Чабановская В.С.             Thirtieth of July 1995 12: 00: 00 AM
Кудрявцев А.А.               First of March 1996 12: 00: 00 AM
Савилов Ю.Н.                 Sixth of November 1997 12: 00: 00 AM
Смирнова Т.Н.                Eleventh of May 1998 12: 00: 00 AM

Чайка Н.Ф.                   Fifteenth of August 1994 12: 00: 00 AM
```

Обратите внимание на то, что название месяца соответствует заданной модели Формата (INITCAP).

Лабораторная работа 3.3 Изменение выводимого формата даты

Задание:

Необходимо представить информацию из таблицы сервера Oracle в соответствии с принятым на предприятии стандартом вывода данных, который выглядит как год/месяц/число, например, 2008/02/20. Напишите запрос, который бы выводил из таблицы hr.employees информацию об имени. фамилии и дате приема на работу сотрудников в соответствии с описанным форматом. Вначале должны выводиться те, кто был принят работу позже. Результат выполнения запроса должен выглядеть так, как представлено на рис. Лаб. 3.3-1.

	Имя	Фамилия	Дата приема на работу
1	Amit	Banda	2000/04/21
2	Sundita	Kumar	2000/04/21
3	Sundar	Ande	2000/03/24
4	Steven	Markle	2000/03/08
5	David	Lee	2000/02/23
6	Hazel	Philtanker	2000/02/06
7	Girard	Geoni	2000/02/03
8	Eleni	Zlotkey	2000/01/29

Рис. Лаб. 3.3-1

3.6.11 Функция TO_CHAR с числами

TO_CHAR(number, ‘fmt’);

Форматы, используемые с функцией TO_CHAR, для вывода символьного значения в виде числа:

9	Цифра
0	Вывод нуля
$	Плавающий знак доллара
L	Плавающий символ местной валюты
.	Вывод десятичной точки
,	Вывод разделителя троек цифр

Функция TO_CHAR преобразует данные типа NUMBER в данные типа VARCHAR2. Это особенно полезно при конкатенации.
Элементы числового формата
Преобразуя число в данные типа VARCHAR2, можно пользоваться следующими элементами.

Элемент	**Описание**	**Пример**	**Результат**
9	Цифровой разряд (количество девяток определяет ширину поля вывода)	999999	1234
0	Вывод ведущих нулей	099999	001234
$	Плавающий символ доллара	$999999	$I234
L	Плавающий символ местной валюты	L999999	FFI234
.	Десятичная точка в указанной позиции	999999.99	1234.00

,	Запятая в указанной позиции	999,999	1,234
Ml	Знак “минус” справа (отрицательные значения)	999999MI	1234-
PR	Отрицательные символы в скобках	999999PR	<I234>
EEEE	Научное обозначение (обязательны четыре E)	99.999EEEE	1234E+03
V	Умножить на 10 n раз (n= кол-во девяток после V)	9999V99 ggggygg	123400
B	Вывод нулевых значений пробелами вместо нулей	B9999.99	1234.00

3.6.12 Пример функции TO_CHAR с числами:

```
SELECT to_char(oklad,'$9999.99') oklad   from  oklad WHERE kodinsp = '5205';
```

OKLAD

$1850.00

Замечания:

- Если количество цифровых разрядов числа превышает количество разрядов, предусмотренное моделью формата, сервер Oracle выводит вместо всего числа строку символов фунта (#).
- Сервер Oracle округляет хранимое десятичное значение до количества десятичных разрядов, заданное в модели формата.

Лабораторная работа 3.4 Изменение формата выводимых чисел

Задание:

Напишите запрос, который бы выводил информацию о имени, фамилии и зарплате сотрудников из таблицы hr.employees в формате, представленном на рис. 3.4-1:

	Имя	Фамилия	Оклад
1	Steven	King	$24000.15
2	Neena	Kochhar	$17000.00
3	Lex	De Haan	$17000.00
4	John	Russell	$14000.00
5	Karen	Partners	$13500.00
6	Michael	Hartstein	$13000.25
7	Shelley	Higgins	$12000.00

Рис. 3.4-1.

При этом данные должны быть отсортированы таким образом, чтобы первыми выводились строки для сотрудников с наибольшей зарплатой.

Примечание.

Некоторые значения зарплаты на рис. 3.4-1 были изменены, поэтому они могут не совпадать с вашими значениями.

3.6.13 Функции TO_NUMBER и TO_DATE

- Преобразование строки символов в числовой формат с помощью функции TO_NUMBER :
  TO_NUMBER (char)
- Преобразование строки символов в формат даты с помощью функции TO_DATE :
  TO_DATE (char [, ‘fmt’])

Иногда требуется преобразовать символьную строку в число или дату. Для этого используются функции TO_NUMBER и TO_DATE. Выбор модели формата будет основан на предыдущих демонстрациях использования элементов формата.
Пример
Вывод фамилий и даты найма всех служащих, принятых на работу 22 февраля 1981 г.

```
SQL>  SELECT     fio, begin_date
  2   FROM       inspector
  3   WHERE      Begin_date = TO_DATE(‘ November 25, 1995’, 'Month dd, YYYY’);
```

FIO BEGIN_DATE
--- ---
Азарченко Л.М. 25.11.1995

3.6.14 Формат даты RR

Текущий год	**Заданная дата**	**Формат RR**	**Формат YY**
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

	Год, заданный двузначным числом		
		0-49	50-99
Если две последних цифры текущего года равны:	0-49	Возвращаемая дата относится к текущему столетию.	Возвращаемая дата относится к столетию перед текущим.
	50-99	Возвращаемая дата относится к столетию после текущего.	Возвращаемая дата относится к текущему столетию.

3.6.15 Элемент RR в формате даты

Элемент RR аналогичен элементу YY, но позволяет задавать разные столетия. Элемент RR можно использовать вместо YY, чтобы столетие в возвращаемом значении варьировалось в зависимости от заданного двузначного года и двух последних цифр текущего года. Поведение элемента RR суммируется в таблице.

Текущий год	**Заданная дата**	**Интерпретированная дата (RR)**	**Интерпретированная дата (YY)**
1994	27-OCT-95	1995	1995
1994	27-OCT-I7	2017	1917
2001	27-OCT-17	2017	2017

3.6.16 Функция NVL

Преобразует неопределенное значение в действительное

Используемые типы данных - DATE, символьные (CHARACTER) и числовые (NUMBER). Типы данных должны совпадать

- NVL(comm, 0)
- NVL(hiredate,'01-JAN-97')
- NVL(job, ‘No Job Yet')

Функция NVL используется для преобразования неопределенного значения (NULL) в действительное.

Синтаксис

NVL(expr1, expr2*)*

где:

expr1 исходное значение или выражение, которое может содержать неопределенное значение.

expr2 конечное значение для преобразования неопределенного значения.

Функцию NVL можно использовать для преобразования данных любого типа, но тип данных возвращаемого значения всегда такой, как у *expr1*.

3.6.17 Преобразования NVL для различных типов данных

Тип данных	**Пример преобразования**
NUMBER	NVL(*number_column*, 9)
DATE	NVL(*date_column*, ’01-JAN-95’)
CHAR или VARCHAR2	NVL(*character_column*, ‘Unavailable’)

3.6.18 Использование функции NVL:

Пример

```
SQL>    SELECT     Kodinsp,nvl(fio,’He определен’)
  2     FROM       inspector;
```

KODI NVL(FIO,'НЕОПР
------- --
5205 Лютикова З.С.
5214 Азарченко Л.М.
5218 Чабановская В.С.
5605 Кудрявцев А.А.
5611 Савилов Ю.Н.
5612 Смирнова Т.Н.
5700 Не определен
5712 Чайка Н.Ф.

Если какой-либо столбец в выражении содержит неопределенное значение, результатом также будет неопределенное значение. Необходимо преобразовать неопределенное значение в число прежде, чем применять арифметический оператор.

Лабораторная работа 3.5 Применение функции NVL

Задание:

Напишите запрос, который выводит информацию об имени и фамилии сотрудников из таблицы hr.employees. а также ставку комиссии (столбец COMMISSION_PCT для сотрудника. При этом для тех сотрудников, для которых комиссия не определена, нужно вывести значение 0. Результат выполнения запроса должен быть таким, как представлено на рис. 3.5-1.

	Имя	Фамилия	Ставка комиссии
51	Trenna	Rajs	0
52	Curtis	Davies	0
53	Randall	Matos	0
54	Peter	Vargas	0
55	John	Russell	0.4
56	Karen	Partners	0.3
57	Alberto	Errazuriz	0.3
58	Gerald	Cambrault	0.3
59	Eleni	Zlotkey	0.2
60	Peter	Tucker	0.3

Рис. 3.5-1 (показаны значения начиная со строки 51).

3.7 Функция DECODE

Упрощает условные запросы, выполняя работу команды CASE или IF-THEN-ELSE
Синтаксис

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

- Функция DECODE действует подобно IF-THEN-ELSE в различных языках. Функция DECODE расшифровывает выражение (expression) после сравнения его с каждым искомым значением (search). Если выражение равно искомому значению, функция возвращает результат (result).
- Если выражение не совпадает ни с одним из искомых значений, а значение по умолчанию не задано, функция возвращает неопределенное значение.

Использование функции DECODE :
То же самое можно сделать с помощью команды IF-THEN-ELSE

Лабораторная работа 3.6 Применение функции DECODE

Задание:

Напишите запрос, который возвращает информацию об имени, фамилии и должности сотрудников (столбец JOB_ID) на основе таблицы hr.employees. При этом:

- если в столбце JOB_ID для сотрудников находится значение SA_REP, то должно выводиться "Торговый представитель";
- если в столбце JOB_ID для сотрудников находится значение SA_MAN, то должно выводиться "Менеджер по продажам";
- если в этом столбце находится любое другое значение, то должно выводиться "Другое".

Результат запроса должен быть таким, как представлено на рис. 3.6-1

	Имя	Фамилия	Должность
52	Curtis	Davies	Другое
53	Randall	Matos	Другое
54	Peter	Vargas	Другое
55	John	Russell	Менеджер по продажам
56	Karen	Partners	Менеджер по продажам
57	Alberto	Errazuriz	Менеджер по продажам
58	Gerald	Cambrault	Менеджер по продажам
59	Eleni	Zlotkey	Менеджер по продажам
60	Peter	Tucker	Торговый представитель
61	David	Bernstein	Торговый представитель
62	Peter	Hall	Торговый представитель

Рис. 3.6-1

3.8 Вложение функций

- Однострочные функции могут быть вложены на любую глубину.
- Вложенные функции вычисляются от самого глубокого уровня к внешнему.

Однострочные функции могут быть вложены на любую глубину. Вложенные функции вычисляются от самой внутренней к самой внешней. Примеры демонстрируют гибкость этих функций.

Пример 1.

```
SQL>   SELECT      fio,
  2                NVL(to_char(TO_number(mgr)) , 'No Manager')
  3    FROM        inspector
  4    WHERE       mgr IS NULL;
```

```
FIO                          NVL(TO_CHAR(TO_NUMBER(MGR)),'
---------------------------- ----------------------------------------------------
Чабановская В.С.     No Manager
Савилов Ю.Н.                 No Manager
Чайка Н.Ф.           No Manager
```

В этом примере на экран выводится фамилия инспектора, который не имеет начальника.
Оценка команды SQL включает два шага:
1. Вычисление внутренней функции для преобразования числа в символьную строку.
  - Result1 = TO_CHAR(mgr)
2. Вычисление внешней функции для замены неопределенного значения текстовой строкой.
- NVL(Result1, ‘No Manager')

Т. к. псевдоним не задан, все выражение становится заголовком столбца.

Пример 2:

Вывод даты первой пятницы через 6 месяцев с даты найма. Формат даты –“Friday, March 12th, 1982". Результат упорядочен по датам найма.

```
SQL>  SELECT      TO_CHAR(NEXT_DAY(ADD_MONTHS

  2               (begin_date, 6),'FRIDAY'),

  3               'fmDay, Month ddth, YYYY')

  4               "Next 6 Month Review"

  5   FROM        inspector

  6   ORDER BY    Begin_date;
```

Next 6 Month Review

Friday, February 17th, 1995
Friday, May 12th, 1995
Friday, February 2nd, 1996
Friday, May 31st, 1996
Friday, September 6th, 1996
Friday, May 8th, 1998
Friday, November 13th, 1998

5. Агрегирование данных с помощью групповых функций

5.1 Что такое групповые функции?

Групповые функции работают с множеством строк и возвращают один результат на группу.
В отличие от однострочных функций групповые функции обрабатывают множества строк для получения единого результата по группе. Таким множеством строк может быть целая таблица или таблица, разбитая на группы.

5.2 Типы групповых функций

Каждая функция принимает аргумент. В следующей таблице показаны варианты синтаксиса.

Функция	**Описание**
AVG([DISTINCT|ALL] n)	Среднее значение n без учета неопределенных значений
COUNT({*|[DISTINCT|ALL] expr})	Количество строк, где результатом вычисления выражения является любое определенное значение. Если используется “*” - счетчик всех выбранных строк, включая дубликаты и строки с неопределенными значениями
MAX([DISTINCT|ALL] expr)	Максимальное значение expr без учета неопределенных значений
MIN([DISTINCT|ALL] expr)	Минимальное значение expr без учета неопределенных значений
STDDEV ([DISTINCT|ALL] n)	Стандартное отклонение n без учета неопределенных значений

SUM([DISTINCT|ALL] n)	Суммирование значений n без учета неопределенных значений
VARIANCE ([DISTINCT|ALL] n)	Дисперсия n без учета неопределенных значений

5.3 Использование групповых функций

Пример

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

5.4 Указания по использованию групповых функций

- Если используется слово DISTINCT, дубликаты при вычислении функции не учитываются. Если используется слово ALL, рассматриваются все значения, включая дубликаты. Слово ALL указывать не обязательно, т.к. оно используется по умолчанию.
- Допустимые типы данных для аргументов: CHAR, VARCHAR2, NUMBER или DATE, если задано выражение (expr).
- Все групповые функции, кроме COUNT(*) игнорируют неопределенные значения. Для замены неопределенных значений определенными используется функция NVL.

5.5 Использование функций AVG и SUM

Функции AVG и SUM применяются к столбцам с числовыми данными

```
SQL>      SELECT          AVG(oklad), MAX(oklad),
  2                       MIN(oklad), SUM(oklad)
  3       FROM            Oklad o,inspector i
  4       WHERE           i.kodinsp=o.kodinsp and  dol LIKE
                          'Инсп%';
```

Функции AVG, SUM, MIN и MAX применяются к столбцам, в которых можно хранить цифровые данные. В примере на слайде вычисляются средний, самый высокий, самый низкий оклад и сумма окладов всех продавцов.

5.6 Использование функций MIN и MAX

Функции MAX и MIN применяются к данным любого типа

```
SQL>  SELECT   MIN(begin_date), MAX(begin_date)
  2   FROM     inspector;
```

Функции MAX и MIN применяются к данным любого типа. Пример показывает вычисление самого давно работающего и самого недавно работающего служащего.
Следующий пример показывает первого и последнего служащего в алфавитном списке служащих.

```
SQL>  SELECT     MIN(fio), MAX(fio)
  2   FROM       inspector;
```

Функции AVG, SUM, VARIANCE и STDDEV применимы только к цифровым данным.

5.7 Использование функции COUNT

COUNT(*) возвращает количество строк в таблице

```
SQL>  SELECT     COUNT(*)
 2    FROM       inspector
 3    WHERE      otdel = 52;
```

Имеется два формата функции COUNT:

- COUNT(*)
- COUNT(expr)

COUNT (*) возвращает количество строк в таблице, включая строки- дубликаты и строки с неопределенными значениями.
Функция COUNT(expr) возвращает количество строк с определенными значениями в столбце, заданном выражением (expr).
В приведенном выше примере подсчитывается количество служащих в отделе 52.

Пример

COUNT(expr) возвращает количество строк с определенными значениями (не NULL)

```
SQL>    SELECT     COUNT(fio)
  2     FROM       inspector

  3     WHERE      otdel = 57;
```

```
COUNT(FIO)
----------
1
```

В вышеприведенном примере вычисляется количество реальных служащих отдела 57, которые введены с фамилией . Результат - 1. т.к. другой служащий отдела – фиктивный инспектор, в поле fio стоит null.
Пример 1.
Вывод количества отделов в таблице inspector.

```
SQL>  SELECT   COUNT(DISTINCT(otdel))

  2   FROM     inspector;
```

```
COUNT(DISTINCT(OTDEL))
----------
3
```

5.8 Групповые функции и неопределенные значения

Все групповые функции, кроме COUNT(*), игнорируют неопределенные значения в столбце. В примере среднее вычисляется только по строкам, где столбец COMM содержит действительное значение. Среднее вычисляется, как частное от деления общей суммы комиссионных, выплаченных всем служащим, на количество служащих, получающих комиссионные.

Пример синтаксиса

```
SQL>  SELECT        AVG(oklad)
```

```
  2    FROM          oklad;
```

5.9 Использование функции NVL с групповыми функциями

Функция NVL заставляет групповые функции включать неопределенные значения

```
SELECT      COUNT(nvl(fio,'Не
            определен'))
FROM        inspector

WHERE       otdel = 57;
```

COUNT(NVL(FIO))
2

Функция NVL заставляет групповые функции включать неопределенные значения. В примере среднее вычисляется по всем строкам, включая строки с неопределенным значением в столбце COMM. Среднее в этом случае - это частное от деления общей суммы комиссионных, выплаченных служащим, на общее количество служащих в компании.

Лабораторная работа 5.1 Применение агрегатных функций

Задание:

Напишите запрос, который бы вернул информацию о максимальной, минимальной и средней заработной плате из таблицы employees в схеме hr (информация о заработной плате находится в столбце salary). Результаты выполнения запроса должны выглядеть так, как представлено на рис. 5.1-1.

	Максимальная зарплата	Минимальная зарплата	Средняя зарплата
1	24000	2100	6461.68

Рис. 5.1-1

5.10 Создание групп данных

5.10.1 Группы данных

До сих нор все групповые функции обрабатывали таблицу как одну большую группу информации. Но иногда требуется разделить таблицу на более мелкие группы. Сделать это можно с помощью предложения GROUP BY.

Предложение GROUP BY

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

Предложение GROUP BY разбивает строки таблицы на группы.
С помощью предложения GROUP BY можно разделить строки таблицы на группы. Затем можно использовать групповые функции для получения сводной информации по каждой группе.
Синтаксис:

group_by_expression	задает столбцы, по значениям которых группируются строки.

5.10.2 Указания по использованию GROUP BY

- Если групповая функция задана в предложении SELECT, получить одновременно индивидуальный результат можно только в случае, если отдельный столбец указан в предложении GROUP BY. Если список столбцов отсутствует, выдается сообщение об ошибке.
- Предложение WHERE позволяет исключать строки до их группирования.
- Список столбцов в предложении GROUP BY обязателен.
- Использование псевдонимов столбцов в предложении GROUP BY недопустимо.
- По умолчанию строки сортируются в порядке возрастания значений столбцов, заданных в предложении GROUP BY. Изменить этот порядок можно с помощью предложения ORDER BY.

5.10. 3 Использование предложения GROUP BY

Все столбцы, которые входят в список SELECT и к которым не применяются групповые функции, должны быть указаны в предложении GROUP BY.

```
SQL>    SELECT          otdel, AVG(sal)

  2     FROM            Oklad o,inspector I

 3      WHERE           i.kodinsp=o.kodinsp

  4     GROUP BY        otdel;
```

Если используется предложение GROUP BY, все столбцы из списка SELECT, к которым не применяются групповые функции, должны быть включены в предложение GROUP BY. В приведенном примере выдаются номера отделов и средний оклад по каждому отделу.
Команда SELECT с предложением GROUP BY делает следующее:

- Предложение SELECT задает выбираемые столбцы:
    - Столбец номеров отделов из таблицы oklad
    - Среднее всех окладов в группе, заданной с помощью предложения GROUP BY
- Предложение FROM задает таблицы, к которым должна обратиться база данных (в данном случае – таблицы INSPECTOR и OKLAD).
- Предложение WHERE задает критерии отбора строк.
- Предложение GROUP BY указывает, как должны быть сгруппированы строки. Строки группируются по отделам, чтобы функция AVG, применяемая к столбцу окладов, вычислила средний оклад по каждому отделу.

Пример. Столбец, указанный в GROUP BY, может отсутствовать в списке SELECT.

```
SQL>    SELECT           AVG(sal)

  2     FROM            Oklad o,inspector I

 3      WHERE           i.kodinsp=o.kodinsp

  4     GROUP BY        otdel;
```

Столбец, заданный в предложении GROUP BY, может отсутствовать в предложении SELECT. В примере средние оклады по отделам выводятся без соответствующих номеров отделов. Но без номеров отделов результат трудно понять.
Можно использовать групповую функцию в предложении ORDER BY.

```
SQL>    SELECT          otdel, AVG(sal)
```

```
  2      FROM            Oklad o,inspector I
 3       WHERE           i.kodinsp=o.kodinsp
  4      GROUP BY        otdel;
         ORDER BY        AVG(sal)
```

Лабораторная работа 5.2 Применение выражения Group By

Задание:

Напишите запрос, который бы возвращал информацию о максимальной, минимальной и средней заработной плате для каждой должности в таблице employees в схеме hr. Информация о заработной плате находится в столбце salary, а информация о должности — в таблице job_id. Результат выполнения запроса должен выглядеть так, как представлено на рис. 5.2-1

	Должность	Максимальная зарплата	Минимальная зарплата	Средняя зарплата
1	AC_MGR	12000	12000	12000.00
2	AC_ACCOUNT	8300	8300	8300.00
3	IT_PROG	9000	4200	5760.00
4	ST_MAN	8200	5800	7280.00
5	AD_ASST	4400	4400	4400.00
6	PU_MAN	11000	11000	11000.00
7	SH_CLERK	4200	2500	3215.00
8	AD_VP	17000	17000	17000.00
9	FI_ACCOUNT	9000	6900	7920.00
10	MK_MAN	13000	13000	13000.00

Рис. 5.2-1

5.10.4 Группировка по нескольким столбцам

Вложенные группы

Иногда требуются результаты по группам внутри групп. Отчет на рисунке показывает среднюю сумму заработной платы, выплаченной по каждой должности в каждом отделе. *Пример.*Строки таблицы OKLAD сначала группируются по номерам отделов, а затем внутри этих групп - по должностям.

```
SQL>     SELECT           Otdel,dol,AVG(sal)
  2      FROM            Oklad o,inspector I
 3       WHERE           i.kodinsp=o.kodinsp
  4      GROUP BY        Otdel,dol
```

Чтобы получить результаты по группам и подгруппам, следует указать несколько столбцов в предложении GROUP BY. Последовательностью столбцов в предложении GROUP BY можно задать порядок сортировки строк по умолчанию. Команда SELECT в примере, содержащая предложение GROUP BY, делает следующее:

- Предложение SELECT задает выбираемые столбцы:
    - Номера отделов из таблицы OKLAD
    - Должности из таблицы INSPECTOR

- Среднее всех окладов в группе, заданной с помощью предложения GROUP BY

- Предложение FROM задает таблицы, к которым должна обратиться база данных в
- Предложение GROUP BY указывает, как должны быть сгруппированы строки:
  - Сначала по номерам отделов.
  - Затем по должностям внутри отделов.

Таким образом, функция AVG применяется к столбцу окладов по каждой должности внутри каждого отдела.

5.10.5 Недействительные запросы с групповыми функциями

Если в одной и той же команде SELECT используется смесь отдельных элементов и групповых функций (COUNT), предложение GROUP BY со списком отдельных элементов обязательно. Если предложение GROUP BY отсутствует, выдается сообщение об ошибке "not a single-group group function" ("групповая функция для более, чем одной группы"), и столбец, вызвавший ошибку, помечается звездочкой (*). Для устранения ошибки добавьте предложение GROUP BY.
Любой столбец или выражение из списка SELECT, к которым не применяется групповая функция, обязательно указывается в предложении GROUP BY.
Предложения WHERE для исключения групп не используется. Такая команда SELECT вызовет ошибку, т.к. для вывода средних окладов только тех отделов, где средний оклад превышает 2000 долларов, используется предложение WHERE.
Для исключения групп следует использовать предложение HAVING.

```
SQL>  SELECT      otdel, AVG(oklad)
  2   FROM        Inspector I,oklad o
      WHERE       i.kodinsp=o.kodinsp
  3   GROUP BY    otdel
  4   HAVING      AVG(sal) > 2000;
```

5.10.6 Исключение групп

Подобно тому, как предложение WHERE используется для исключения строк, предложение HAVING используется для исключения групп. Чтобы выяснить средний оклад по каждому отделу, но вывести результат только по тем отделам, где он превышает 2000 , необходимо сделать следующее:

1. Найти средний оклад по каждому отделу путем группировки строк по номерам отделов.
2. Исключить из отчета те группы, где средний оклад не превышает 2000 р.

5.10.7 Предложение HAVING

Для исключения групп пользуйтесь предложением HAVING

- Строки группируются.
- Применяется групповая функция.
- Выводятся группы, удовлетворяющие условию в предложении HAVING.

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

С помощью предложения HAVING из выходных данных исключаются некоторые группы. Таким образом, исключение групп производится по агрегированной информации.

Синтаксис:

group_condition	вывод ограничивается группами строк, удовлетворяющими заданному условию.

Сервер Oracle обрабатывает предложение HAVING следующим образом:

- Строки группируются.
- К группе применяется групповая функция.
- Выводятся группы, удовлетворяющие критериям в предложении HAVING.

Предложение HAVING может предшествовать предложению GROUP BY, но логичнее сделать предложение GROUP BY первым. Образование групп и вычисление групповых функций происходят до того, как к группам из списка SELECT применяется предложение HAVING.

5.10.8 Вложенные групповые функции

Пример: Вывод максимального среднего оклада.

```
SQL>    SELECT      MAX(AVG(oklad))

        FROM        oklad o, INSPECTOR I

        WHERE       I.KODINSP=O.KODINSP

  2     GROUP BY    OTDEL
```

Групповые функции могут быть вложены на любую глубину. В этом примере вычисляется максимальный средний оклад.

Лабораторная работа 5.3 Дополнительные возможности GROUP BY

Задание:

Напишите запрос, в котором должна выводиться информация о максимальной, минимальной и средней заработной плате на основе информации из столбца salary таблицы hr.employees. При этом данная информация:

- вначале должна быть сгруппирована по должностям (столбец job_id таблицы employees);
- затем должна быть сгруппирована по отделам (столбец department_name таблицы departments);
- должны выводиться только те группы для должностей, средняя заработная плата для которых выше или равна 7000;
- информация должна быть отсортирована вначале по отделам, а затем по должностям.

Результаты выполнения запроса должны выглядеть так, как представлено на рис. Лаб. 5.3-1.

	Отдел	Должность	Максимальная зарплата	Минимальная зарплата	Средняя зарплата
1	Accounting	AC_ACCOUNT	8300	8300	8300.00
2	Accounting	AC_MGR	12000	12000	12000.00
3	Executive	AD_PRES	24000	24000	24000.00
4	Executive	AD_VP	17000	17000	17000.00
5	Finance	FI_ACCOUNT	9000	6900	7920.00
6	Finance	FI_MGR	12000	12000	12000.00
7	Marketing	MK_MAN	13000	13000	13000.00
8	Public Relations	PR_REP	10000	10000	10000.00
9	Purchasing	PU_MAN	11000	11000	11000.00
10	Sales	SA_MAN	14000	10500	12200.00
11	Sales	SA_REP	11500	6100	8396.55
12	Shipping	ST_MAN	8200	5800	7280.00

Рис. Лаб. 5.3-1

5.10.9 ИТОГИ

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

Последовательность оценки предложений:

- WHERE
- GROUP BY
- HAVING

В SQL имеется семь групповых функций:

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

Предложение GROUP BY позволяет создавать подгруппы. Исключать группы можно с помощью предложения HAVING. В команде предложения HAVING и GROUP BY должны следовать за предложением WHERE. Предложение ORDER BY должно быть последним.

Сервер Oracle обрабатывает предложения в следующем порядке:

- Если имеется предложение WHERE, сервер выявляет строки- кандидаты.
- Сервер выявляет группы, заданные предложением GROUP BY.
- Предложение HAVING исключает из выходных данных группы, не удовлетворяющие его критериям.