Министерство образования и науки Российской Федерации

САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

О. Ю. Сабинин, Н. В. Андреева

БАЗЫ ДАННЫХ.
ПРАКТИКУМ
ЧАСТЬ 2. ORACLE SQL

Учебное пособие

Санкт-Петербург
Издательство Политехнического университета
2013

Сабинин О. Ю., **Базы данных. Практикум. Часть 2. Oracle SQL** : учеб. пособие / О. Ю. Сабинин, Н. В. Андреева. – СПб. : Изд-во Политехн. ун-та, 2013. – 136 с.

Включены задачи на создание запросов к базам данных с использованием языка SQL реализации Oracle. Данное учебное пособие предназначено для тех, кто уже знаком с основами языка SQL, его основной целью является предоставление материала для закрепления и углубления практических навыков создания эффективных SQL-запросов с учётом особенностей СУБД Oracle 11g. Решение задач позволит студентам более детально изучить различные возможности языка Oracle SQL: выборку данных из нескольких таблиц, реализацию условной логики, использование подзапросов, организацию иерархии и рекурсии, обращение к данным как к многомерным массивам и др. Также приведены задачи на использование разнообразных видов функций: однострочных, агрегатных, аналитических (оконных и ранжирования).

Учебное пособие предназначено для самостоятельной работы студентов, обучающихся по IT-специальностям, и может использоваться для подготовки к олимпиадам.

Печатается по решению редакционно-издательского совета
Санкт-Петербургского государственного политехнического университета

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Сборник содержит задачи, основанные на содержании лекций второй части курса «Базы данных» (Oracle SQL). Использование данных задач предполагается в дополнение к проведению обычных практических занятий – для развития и закрепления навыков использования изученных возможностей языка. Некоторые задачи сборника были сформулированы по материалам различных олимпиад, предполагающих использование диалекта Oracle SQL, а также обсуждений на форумах, размещённых на таких сайтах как sql.ru (http://www.sql.ru/forum/actualtopics.aspx?bid=3) и dba-village.com (http://www.dba-village.com/village/dvp_forum.main).

Все задачи сгруппированы в несколько разделов — в соответствии с изучаемой тематикой, общая справочная информация приведена в приложении. При решении задач любого раздела разрешается пользоваться только теми возможностями языка и материалами приложения, которые явно указаны в тематике текущего раздела (в частности, если задач на иерархию ещё не было, запрещается использовать раздел CONNECT BY даже для генерации последовательности значений).

Слова «произвольный», «заданный» (произвольная строка, заданная дата) в условии задачи означают, что значение параметра, о котором идёт речь должно вводиться пользователем.

Для решения задач в основном используются таблицы базовой схемы Human Resources (HR) СУБД Oracle. Задача 7.11 предполагает работу с таблицами базовой схемы Order Entry (OE). Задачи 4.2 и 4.3 требуют участия преподавателя или установки СУБД Oracle 11g release 2 Express Edition (облегчённая бесплатная версия СУБД Oracle), установочные файлы которой

можно скачать из раздела Downloads сайта Oracle.com (http://www.oracle.com/technetwork/products/express-edition/downloads/index.html).

Используемая терминология:

Команды SQL являются самостоятельными логическими единицами языка и отделяются друг от друга разделителем в виде точки с запятой.

Примеры команд SQL: SELECT, MERGE, ALTER, DROP, GRANT.

Команды состоят из одной или более отдельных логических частей, называемых **разделами**.

Примеры разделов: SELECT, FROM, GROUP BY, WITH, MODEL.

Каждый раздел начинается с одноимённого **ключевого слова**, а также может содержать другие ключевые слова.

Примеры ключевых слов: UPDATE, PRIOR, TO, NULLS FIRST.

РАЗДЕЛ 1. ЗАДАЧИ НА ИСПОЛЬЗОВАНИЕ РАЗЛИЧНЫХ ВИДОВ ОДНОСТРОЧНЫХ ФУНКЦИЙ, УСЛОВНЫХ ВЫРАЖЕНИЙ И ГРУППИРОВКУ

Допускается использование справочной информации из разделов 1-11 Приложения.

1.1. Определить сумму цифр в произвольной символьной строке.

Пример результата:

Строка	Сумма цифр
Women now have a life expectancy of 80.1 years, 5.3 more than men.	17

1.2. В 1845 году в США была установлена традиция, согласно которой выборы президента проводятся во вторник после первого понедельника ноября в год, делящийся на 4 без остатка. Определить, дату ближайших к заданной дате президентских выборов в США.

Примеры результата:

Заданная дата	Дата выборов
25.06.2012	12.11.2012
23.02.2003	08.11.2004

1.3. В произвольной символьной строке убрать все лидирующие и конечные пробелы, а между словами оставить только по одному пробелу.

Пример результата:

Исходная строка	Результат
····This·····is·the·····solution····[1]	This·is·the·solution

[1] Здесь и далее символ «·» используется для обозначения одного пробела.

1.4. Определить дату начала ближайшего к заданной дате Уимблдонского турнира, который начинается за шесть недель до первого понедельника августа. Если заданная дата совпадает с датой начала турнира в этом году, вывести её.

Примеры результата:

Заданная дата	Дата начала турнира
25.06.2012	25.06.2012
26.06.2012	24.06.2013
27.12.2010	20.06.2011

1.5. Определить временной интервал между заданной и текущей датами. Результат вывести в виде: ## лет[1] ## мес ## дн, где # обозначает цифру, лидирующие нули и отсутствующие значения не выводить.

Примеры результата:

Заданная дата	Текущая дата	Временной интервал
25.06.2012	30.09.2012	3 мес 5 дн
04.07.2015	30.09.2012	2 года 9 мес 4 дн

1.6. Определить, сколько раз 29 февраля встречается между двумя заданными датами.

Примеры результата:

Заданная дата 1	Заданная дата 2	Результат
12.12.2001	14.02.2012	2
12.12.2001	29.02.2012	3
12.02.1500	02.03.1504	2
12.02.1700	03.02.1704	1

[1] В зависимости от конкретного числа вывести подходящее окончание: лет, или год, или года.

1.7. В названии отдела вывести только второе слово, если название состоит из двух или более слов. Иначе вывести первое слово.

Пример результата:

DEPARTMENT_NAME	RESULT
Accounting	Accounting
…	…
Control And Credit	And
…	…
Government Sales	Sales
…	…

1.8. Создать таблицу Test, состоящую из двух столбцов ID и Text. Столбец ID — ключевой, а столбец Text содержит символьную информацию, включающую цифры и символы латинского алфавита, при этом либо буквенная, либо цифровая части могут отсутствовать. Требуется создать запрос, который выведет информацию из второго столбца в отсортированном виде. Правила сортировки следующие:

- выражения, начинающиеся с буквы, вывести первыми;
- символьные выражения, начинающиеся с чисел, должны быть отсортированы в порядке увеличения чисел, а в случае наличия выражений с одинаковыми числами — по алфавиту той части выражении, которая следует за числом, считая число менее приоритетным;
- в случае наличия выражений, начинающихся с одинакового числа, некоторые из которых не содержат буквенной части, первыми вывести числа.

Например, для таблицы:

ID	Text
1	12six
2	t7w6o
3	76three
4	123four
5	five
6	76one
7	123

Результат должен быть такой:

Text
five
t7w6o
12six
76one
76three
123
123four

1.9. Найти все (вещественные и/или комплексные) корни квадратного уравнения $ax^2 + bx + c = 0$ для заданных коэффициентов a, b и c. Результат округлить до двух цифр после запятой.

Примеры результата:

a	b	c	Корни квадр. уравнения
2	7	6	Корень 1: -1.50; корень 2: -2.00
1.5	14	8	Корень 1: -0.61; корень 2: -8.72
3	9	9	Корень 1: -1,5+0.87* i; корень 2: -1,5-0.87*i
0	7	4	Не является квадратным уравнением

1.10. Создать запрос для определения произведения количеств сотрудников в отделах.

Пример результата:

Intersection
3304800

1.11. Написать команду, которая позволит выводить фамилию сотрудника через запятую столько раз, сколько букв в его фамилии.

Пример результата:

LAST_NAME
OConnell, OConnell, OConnell, OConnell, OConnell, OConnell, OConnell, OConnell
Grant, Grant, Grant, Grant, Grant
…

РАЗДЕЛ 2. ЗАДАЧИ НА СОЕДИНЕНИЕ ТАБЛИЦ, ИСПОЛЬЗОВАНИЕ ПОДЗАПРОСОВ И ОПЕРАТОРОВ SET

Допускается использование справочной информации из разделов 1-13 Приложения.

2.1. Вывести фамилии сотрудников, их оклад, идентификатор отдела и плотный[1] ранг окладов в пределах отдела, в котором работает сотрудник. Ранг 1 должен быть у максимальной зарплаты.

Пример результата:

DEPARTMENT_ID	LAST_NAME	SALARY	SALARY_RANC
10	Whalen	4400	1
20	Hartstein	13000	1
20	Fay	6000	2
…	…	…	…
50	Chung	3800	10
50	Dilly	3600	11
50	Ladwig	3600	11
50	Rajs	3500	12
…	…	…	…

2.2. Вывести фамилии сотрудников, их оклад, идентификатор отдела и неплотный[2] ранг окладов в пределах

[1] При присвоении рангов учитываются только уникальные значения окладов, поэтому в последовательности значений рангов нет промежутков (у сотрудника Rajs ранг оклада 12, несмотря на то, что его оклад — на 13-м месте в общем списке отсортированных по возрастанию окладов отдела 50).

[2] Ранги присваиваются согласно положению каждого уникального значения оклада в общем списке окладов, поэтому в последовательности значений рангов могут содержаться «дырки» (у сотрудника Landry ранг оклада 14, так как его оклад — на 13-м месте в общем списке отсортированных по возрастанию окладов отдела 50, а

отдела, в котором работает сотрудник. Ранг 1 должен быть у максимальной зарплаты.

Пример результата:

DEPARTMENT_ID	LAST_NAME	SALARY	SALARY_RANC
10	Whalen	4400	1
20	Hartstein	13000	1
20	Fay	6000	2
…	…	…	…
50	Chung	3800	10
50	Dilly	3600	11
50	Ladwig	3600	11
50	Rajs	3500	13
…	…	…	…

2.3. Вывести фамилии тех сотрудников, чей оклад выше среднего в отделе, в котором они работают. В результат вывести:

1) идентификатор отдела, в котором работает сотрудник;

2) средний оклад по отделу, округлённый до целого числа;

3) фамилию сотрудника;

4) оклад сотрудника.

Сведения должны быть отсортированы по возрастанию:

1) по идентификатору отдела, к которому приписан сотрудник;

2) по окладу, установленному сотруднику

оклада с рангом 12 нет совсем, так как на 2-м и 3-м месте — одинаковые оклады, которым присвоен ранг 11).

Пример результата:

DEPARTMENT_ID	AVG_SAL	LAST_NAME	SALARY
20	9500	Hartstein	13000
30	4150	Raphaely	11000
50	3476	Rajs	3500
50	3476	Ladwig	3600
…	…	…	…
100	8601	Faviet	9000
100	8601	Greenberg	12008
110	10154	Higgins	12008

2.4. Вывести фамилии сотрудников, начальники которых работают в другой стране. В результат вывести:

1) идентификатор сотрудника;

2) фамилию сотрудника;

3) название страны, где расположен офис сотрудника;

4) идентификатор непосредственного руководителя сотрудника;

5) фамилию непосредственного руководителя сотрудника;

6) название страны, где расположен офис непосредственного руководителя сотрудника.

Пример результата:

EMP_ID	EMP_NAME	EMP_COUNTRY	MAN_ID	MAN_NAME	MAN_COUNTRY
201	Hartstein	Canada	100	King	United States of America
204	Baer	Germany	101	Kochhar	United States of America
148	Cambrault	United Kingdom	100	King	United States of America
…	…	…	…	…	…

2.5. В таблицу записана информация, об удачных и неудачных попытках подключения к базе данных (Пользователь, Время, Удачно\Неудачно). Требуется получить список пользователей, которые совершили подряд три неудачные попытки подключения наряду с зафиксированным временем третьей неудачной попытки. После трех подряд неудачных попыток отсчет попыток начинается сначала.

Например, для таблицы:

Пользователь	**Время**	**Статус**
A	20.11.11 17:58:00	Неудачно
B	20.11.11 18:00:05	Удачно
C	20.11.11 18:10:03	Неудачно
A	20.11.11 18:12:20	Неудачно
B	20.11.11 18:18:00	Неудачно
B	20.11.11 18:20:01	Удачно
C	20.11.11 18:25:42	Неудачно
A	20.11.11 18:30:12	Неудачно
A	20.11.11 18:32:24	Неудачно
A	20.11.11 18:35:00	Удачно
B	20.11.11 18:41:30	Удачно
C	20.11.11 18:42:08	Неудачно
C	20.11.11 18:48:00	Удачно
A	20.11.11 18:52:00	Неудачно
A	20.11.11 18:53:13	Неудачно
B	20.11.11 18:54:30	Неудачно
A	20.11.11 18:55:19	Неудачно
A	20.11.11 18:55:58	Удачно

Результат должен быть такой:

Пользователь	**Время**
A	20.11.11 18:30:12
C	20.11.11 18:42:08
A	20.11.11 18:55:19

2.6. Создать запрос для определения списка городов, в которых расположены департаменты, суммарная заработная плата в которых выше средней суммарной заработной платы в департаментах этого города. Если к отделу не приписано ни одного сотрудника, считать суммарную заработную плату в этом отделе равной нулю и учитывать её при подсчёте среднего значения по городу. В результат вывести:

1) название города;

2) название департамента;

3) среднюю суммарную зарплату в городе, округлённую до двух знаков после запятой;

4) суммарную зарплату в департаменте.

Пример результата:

CITY	DEPARTMENT_ NAME	AVG _CITY_SUM_SAL	SUM_DEPT_ SAL
Seattle	Purchasing	7580,95	24900
Seattle	Accounting	7580,95	20300
Seattle	Finance	7580,95	51600
Seattle	Executive	7580,95	58000

2.7. Выбрать сотрудников компании, оклады которых наиболее близки к среднему окладу по подразделению, к которому они приписаны. Требуется вывести:

1) идентификатор сотрудника;

2) фамилию сотрудника;

3) идентификатор должности, которую занимает сотрудник;

4) идентификатор подразделения, к которому приписан сотрудник;

5) оклад, установленный сотруднику;

6) средний оклад по подразделению, к которому приписан сотрудник (округлить до целых).

Сведения должны быть отсортированы по возрастанию:

1) по идентификатору подразделения, к которому приписан сотрудник;

2) по окладу, установленному сотруднику;

3) по фамилии сотрудника.

Пример результата:

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID	SALARY	AVG_SAL
200	Whalen	AD_ASST	10	4400	4400
202	Fay	MK_REP	20	6000	9500
201	Hartstein	MK_MAN	20	13000	9500
…	…	…	…	…	…

2.8. Проверить столбцы First_name, Last_name, Salary таблицы Employees на уникальность значений и вывести все строки таблицы, в которых хотя бы в одном столбце встречается значение, которое в этом столбце не уникально (встречается несколько раз).

Пример результата:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	E_MAIL	…	SALARY	…
198	Donald	OConnell	DOCONNEL	…	2600	…
199	Douglas	Grant	DGRANT	…	2600	…
200	Jennifer	Whalen	JWHALEN	…	4400	…
…	…	…	…	…	…	…

2.9. Создать запрос для вывода фамилий, последних должностей и дат приема на работу сотрудников, информация о работе которых в некоторые временные интервалы отсутствует.

Пример результата:

LAST_NAME	JOB_ID	HIRE_DATE
Raphaely	PU_MAN	07.12.94
Kaufling	ST_MAN	01.05.95
Whalen	AD_ASST	17.09.87

2.10. Дана произвольная строка, состоящая из чисел разделённых заданными одинаковыми арифметическими операторами (только «+» или только «-»). Требуется вычислить математическое выражение, записанное таким образом.

Пример результата

Строка	Результат
12+9+45+61	127

2.11. Дана таблица из двух столбцов: 1 — строка, 2 — число. Требуется написать запрос, в результате которого каждая строка таблицы выдавалась бы столько раз, сколько определено для неё во втором столбце.

Например, для таблицы:

Строка	Число
apple	2
banana	1
melon	4

Результат должен быть такой:

Строка
apple
apple
banana
melon
melon
melon
melon

2.12. Написать запрос, выдающий сведения о нарастающей сумме окладов в пределах отдела.

Требуется вывести:

1) идентификатор отдела;

2) идентификатор сотрудника;

3) фамилию сотрудника;

4) оклад сотрудника;

5) накопленную сумму окладов.

Результат отсортировать по столбцам 1 и 4, перечисленным выше.

Пример результата:

DEPARTMENT_ID	EMPLOYEE_ID	LAST_NAME	SALARY	SUM_SAL
10	200	Whalen	4400	4400
20	202	Fay	6000	6000
20	201	Hartstein	13000	19000
…	…	…	…	…
50	132	Olson	2100	2100
50	128	Markle	2200	4300
50	136	Philtanker	2200	6500
…	…	…	…	…

2.13. Написать команду для удаления одной из двух записей, которые отличаются лишь значениями в двух столбцах. При этом записи должны удовлетворять условиям:

R1.Col1 = R2.Col2 и R2.Col1 = R1.Col2

Например, для таблицы:

Col1	Col2	Col3	Col4
one	two	three	four
two	one	three	four
two	four	one	three
one	two	four	five

Результат должен быть такой:

Col1	Col2	Col3	Col4
one	two	three	four
two	four	one	three
one	two	four	five

2.14. Дана таблица из трёх столбцов, в каждой ячейке которой может содержаться любое количество любых символов. Необходимо вывести содержимое таблицы, выполнив следующее условие: записи, которые могут быть получены из других записей перестановкой значений в столбцах, должны выводиться только один раз.

Например, для таблицы:

Col1	Col2	Col3
ab	a	a
a	b	b
a	a	b
a	de	tu
de	tu	a
a	tu	de
m	l	3n
l	3n	m

Один из вариантов результата такой:

Col1	Col2	Col3
ab	a	a
a	b	b
a	a	b
de	tu	a
l	3n	m

2.15. Из таблицы Employees необходимо выбрать такие пары окладов, суммы которых также содержатся в этой таблице. Также необходимо вывести идентификаторы сотрудников, с окладами, удовлетворяющими условию задачи.

Результат представить в виде:

Оклады	**Сотрудники**
3000,14000->17000	187,145->101
3000,14000->17000	187,145->102
3000,14000->17000	197,145->101
3000,14000->17000	197,145->102
…	…

РАЗДЕЛ 3. РАБОТА С ПРЕДСТАВЛЕНИЯМИ СЛОВАРЯ ДАННЫХ

Допускается использование справочной информации из разделов 1-13 Приложения.

3.1. Для всех связей между таблицами схемы вывести:

1) имя главной таблицы;

2) имя подчиненной таблицы;

3) имя первого столбца первичного ключа;

4) имя второго столбца первичного ключа;

5) общее число столбцов первичного ключа;

6) имя первого столбца вторичного ключа;

7) имя второго столбца вторичного ключа.

Пример результата:

SUPP	CUST	PK_COL1	PK_COL2
DEPARTMENTS	JOB_HISTORY	DEPARTMENT_ID	-
LOCATIONS	DEPARTMENTS	LOCATION_ID	-
…	…	…	…

(продолжение таблицы)

PK_COL_CNT	FK_COL1	FK_COL2
1	DEPARTMENT_ID	-
1	LOCATION_ID	-
…	…	…

3.2. Для каждой таблицы схемы вывести:

1) имя таблицы;

2) имя первого столбца первичного ключа;

3) имя второго столбца первичного ключа;

4) общее число столбцов первичного ключа;

5) имя первой (по алфавиту) подчиненной таблицы;

6) имя второй (по алфавиту) подчиненной таблицы;

7) общее количество подчиненных таблиц у рассматриваемой таблицы;

8) имя первого столбца со значением по умолчанию;

9) первое значение по умолчанию.

Пример результата:

SUPP	PK_COL1	PK_COL2	PK_COL_CNT
DEPARTMENTS	DEPARTMENT_ID	-	1
LOCATIONS	LOCATION_ID	-	1
…	…	…	…

(продолжение таблицы)

CUST1	CUST2	CUST_ CNT	DEF_COL1	DEF1
EMPLOYEES	JOB_HISTORY	2	-	-
DEPARTMENTS	-	1	-	-
…	…	…	…	…

3.3. Для каждой таблицы схемы вывести:

1) имя таблицы;

2) имя первого столбца первого (по алфавиту) ограничения уникальности;

3) имя второго столбца первого ограничения уникальности;

4) общее число столбцов в первом ограничении уникальности;

5) имя первого столбца второго (по алфавиту) ограничения уникальности;

6) имя второго столбца второго ограничения уникальности;

7) общее число столбцов во втором ограничении уникальности;

8) общее число ограничений уникальности.

Пример результата:

TABLE_NAME	U_CONS1_COL1	U_CONS1_COL2	U_CONS1_COL_CNT	U_CONS2_COL1
EMPLOYEES	EMAIL	-	1	LAST_NAME
…	…	…	…	…

(продолжение таблицы)

U_CONS2_COL2	U_CONS2_COL_CNT	U_CONS_CNT
FIRST_NAME	2	2
…	…	…

3.4. Для каждой таблицы схемы вывести:

1) имя таблицы;

2) имя первого (по алфавиту) ограничения Check;

3) текст первого ограничения Check;

4) количество столбцов в первом ограничении Check;

5) имя второго (по алфавиту) ограничения Check;

6) текст второго ограничения Check;

7) количество столбцов во втором ограничении Check;

8) общее количество ограничений Check.

Пример результата:

TABLE_NAME	C1_NAME	C1_TEXT	C1_COL_CNT
EMPLOYEES	EMP_SALARY_MIN	salary > 0	1
COUNTRIES	SYS_C0011603	"COUNTRY_ID" IS NOT NULL	1
…	…		…

(продолжение таблицы)

C2_NAME	C2_TEXT	C2_COL_CNT	C_CNT
SYS_C0011599	"LAST_NAME" IS NOT NULL	1	5
-	-	0	1
…		…	…

3.5. Для таблиц схемы, имеющих индексы вывести:

1) имя таблицы;

2) имя первого (по алфавиту) неуникального индекса;

3) количество столбцов первого неуникального индекса;

4) имя первого (по алфавиту) уникального индекса;

5) количество столбцов первого уникального индекса;

6) общее число неуникальных индексов;

7) общее число уникальных индексов.

Пример результата:

TABLE_NAME	NUI1_NAME	NUI1_COL_CNT
EMPLOYEES	EMP_DEPARTMENT_IX	1
DEPARTMENTS	-	0
…	…	…

(продолжение таблицы)

UI1_NAME	UI1_COL_CNT	NUI_CNT	UI_CNT
EMP_EMAIL_UK	1	4	2
DEPT_ID_PK	1	0	1
…	…	…	…

3.6. Для всех таблиц схемы вывести:

1) имя таблицы;

2) имя первого (по алфавиту) ограничения Check;

3) имена столбцов, на которые действует это ограничение;

4) могут ли эти столбцы содержать пустые значения;

5) общее количество ограничений Check для таблицы.

Пример результата:

TABLE_NAME	C1_NAME	C1_COL
COUNTRIES	SYS_C0011603	COUNTRY_ID
DEPARTMENTS	SYS_C0011604	DEPARTMENT_NAME
…	…	…

(продолжение таблицы)

C1_COL_NULLABLE	C_CNT
N	1
N	1
…	…

3.7. Отобразить все связи по внешнему ключу между таблицами схемы наряду с количеством пользователей, имеющих привилегии на удаление строк из главной таблицы, и правило удаления строк из дочерней таблицы при удалении строк главной.

Результат должен содержать 4 столбца:

1) название дочерней таблицы, соединённое через точку с названием столбца внешнего ключа дочерней таблицы;

2) название главной таблицы, соединённое через точку с названием столбца первичного ключа главной таблицы;

3) количество пользователей, имеющих привилегии на удаление строк из главной таблицы;

4) правило удаления строк.

Пример результата:

CUST.FK	SUPP.PK	US_DEL_CNT	DEL_RULE
COUNTRIES.REGION_ID	REGIONS.REGION_ID	0	NO ACTION
…	…	…	…

3.8. Одной командой SELECT вывести сведения обо всех столбцах таблиц текущей схемы, которые используются во внешних ключах.

В результат вывести пять столбцов:

1) имя ссылочного ограничения целостности (внешнего ключа);

2) имя таблицы, которой принадлежит данное ссылочное ограничение целостности (внешний ключ);

3) имя столбца таблицы, который входит во внешний ключ (foreign key);

4) имя таблицы, на которую ссылается данный внешний ключ;

5) имя столбца таблицы, на которую ссылается данный внешний ключ и которому соответствует столбец, указанный в п.3.

Результат отсортировать по возрастанию по столбцам 2, 1 и 3, перечисленным выше.

Пример результата:

FK_NAME	FK_TABLE	FK_COL
COUNTR_REG_FK	COUNTRIES	REGION_ID
…	…	…
DEPT2_FK	DEPT2	ID
DEPT2_FK	DEPT2	NAME
…	…	…

(продолжение таблицы)

PK_TABLE	PK_COL
REGIONS	REGION_ID
…	…
DEPT3	DEPARTMENT_ID
DEPT3	DEPARTMENT_NAME
…	…

РАЗДЕЛ 4. ЗАДАЧИ НА ИСПОЛЬЗОВАНИЕ ДОПОЛНИТЕЛЬНЫХ ВОЗМОЖНОСТЕЙ ГРУППИРОВКИ. РАБОТА С ВНЕШНИМИ ТАБЛИЦАМИ

Допускается использование справочной информации из разделов 1-15 Приложения.

4.1. Написать запрос, выдающий отчёт о суммарных выплатах сотрудникам, непосредственно подчиняющихся заданному руководителю по идентификаторам должностей (поле Job_id). Непосредственное подчинение предполагает подчинение на первом уровне. *Номер каждого руководителя должен встречаться в отчете лишь дважды*.

Пример результата:

Номер руководителя	Должность	Кол-во сотрудников	Выплаты	Вид выплаты
100	AD_VP	2	34000	Зарплата сотрудников в должности AD_VP
	MK_MAN	1	13000	Зарплата сотрудников в должности MK_MAN
	PU_MAN	1	11000	Зарплата сотрудников в должности PU_MAN
	SA_MAN	5	79650	Зарплата сотрудников в должности SA_MAN

	ST_MAN	5	36400.	Зарплата сотрудников в должности ST_MAN
100		14	174050	Суммарная зарплата у руководителя 100
101	AC_MGR	1	12000	Зарплата сотрудников в должности AC_MGR
	FI_MGR	1	12000	Зарплата сотрудников в должности FI_MGR
	HR_REP	1	6500	Зарплата сотрудников в должности HR_REP
	PR_REP	1	10000	Зарплата сотрудников в должности PR_REP
	AD_ASST	1	4400	Зарплата сотрудников в должности AD_ASST
101		5	44900	Суммарная зарплата у руководителя 101
		34	301250	Общий итог

4.2.[1] Создать директорию Init для следующего пути: C:\oracleexe\app\oracle\product\11.2.0\server\config\scripts\.[2]

Создать внешнюю таблицу для чтения данных из текстового файла init.ora, расположенного в данном каталоге. Таблица должна содержать два столбца: Имя параметра и Значение.

Создать запрос к созданной внешней таблице, который позволит найти общее количество файлов управления (Control_Files).

Пример результата:

Ctl_files_cnt
1

4.3.[3] Создать директорию Alert для следующего пути: C:\oraclexe\app\oracle\diag\rdbms\xe\xe\trace\2.[4]

Создать внешнюю таблицу для чтения данных из текстового файла alert_xe.log, расположенного в данном каталоге.

Создать запрос к созданной внешней таблице, позволяющий получать информацию об ошибках Oracle за определенный день месяца.

---

[1] Для самостоятельной работы с данной задачей предполагается установка Oracle 11g r2 XE. Если возможности использовать Oracle 11g r2 XE нет, то имя используемой директории согласовать с преподавателем.

[2] Указан путь по умолчанию

[3] Для самостоятельной работы с данной задачей предполагается установка Oracle 11g r2 XE. Если возможности использовать Oracle 11g r2 XE нет, то имя используемой директории согласовать с преподавателем.

[4] Указан путь по умолчанию.

Пример результата:

Date	Error_text
25.11.2010 19:24:12	ORA-07445: обнаружено прерывание: core dump [psdghi()+976] [ACCESS_VIOLATION] [ADDR:0x0] [PC:0x4EC59C0] [UNABLE_TO_READ] []
25.11.2010 19:26:27	ORA-07445: обнаружено прерывание: core dump [psdghi()+976] [ACCESS_VIOLATION] [ADDR:0x0] [PC:0x4EC59C0] [UNABLE_TO_READ] []
25.11.2010 19:29:57	ORA-07445: обнаружено прерывание: core dump [psdghi()+976] [ACCESS_VIOLATION] [ADDR:0x0] [PC:0x4EC59C0] [UNABLE_TO_READ] []
…	…
25.11.2010 19:51:25	ORA-07445: обнаружено прерывание: core dump [psdghi()+976] [ACCESS_VIOLATION] [ADDR:0x0] [PC:0x4EC59C0] [UNABLE_TO_READ] []

4.4. Вывести все даты за 2013 год и соответствующие дни недели без использования иерархических запросов и Model.

Пример результата:

Date	Day
01.01.2011	Суббота
02.01.2011	Воскресенье
03.01.2011	Понедельник
…	…
29.12.2011	Четверг
30.12.2011	Пятница
31.12.2011	Суббота

4.5. Создать[1] директорию (объект базы данных Directory) для каталога файловой системы, содержащего текстовый файл с информацией в виде:

[1] В случае использования университетского сервера баз данных по вопросу создания директории обратиться к преподавателю.

Номер·····ФИО·····Дата·····Оценка····Дисциплина
1·····Петров·С.С.·····12-05-2008·····4·····Математика
2·····Сомов·Л.Л.·····4-05-2008·····5·····Физика
3·····Амосов·Д.Г.·····3-05-2008·····4·····Физика

Создать внешнюю таблицу для чтения данных из этого файла.

4.6. Показать в одном отчете для каждого отдела: его номер, наименование, количество работающих сотрудников, средний оклад вместе со следующими данными по каждому сотруднику — фамилия, оклад и должность.

Пример результата:

Номер отдела	**Название отдела**	**Кол-во сотрудников**	**Средний оклад**	**Фамилия**
30	Purchasing	6	4150	
				Raphaely
				Khoo
				Baida
				Tobias
				Himuro
				Colmenares
40	HumanResources	1	6500	
				Mavris
50	Shipping	45	3475,56	
				Weiss
				Fripp
				Kaufling
…	…	…	…	…

(продолжение таблицы)

Оклад	**Должность**
11000	PurchasingManager
3100	PurchasingClerk
2900	PurchasingClerk
2800	PurchasingClerk
2600	PurchasingClerk
2500	PurchasingClerk
6500	HumanResourcesRepresentative
8000	StockManager
8200	StockManager
7900	StockManager
…	……

4.7. Сформировать отчёт, содержащий номер отдела, название отдела, имена и фамилии сотрудников и общее кол-во сотрудников в каждом отделе. Сортировка — по названию отдела и имени и фамилии сотрудника. Также в отчёте должен присутствовать столбец со сквозной нумерацией сотрудников, при этом строки, содержащие общее кол-во сотрудников (в т. ч. по отделам) нумероваться не должны.

Пример результата:

Номер п/п	**Идентификатор отдела**	**Название отдела**	**Сотрудник**
1	110	Accounting	Shelley Higgins
2	110	Accounting	William Gietz
	Кол-во сотрудников в отделе	Accounting	2
3	10	Administration	Clark Kent

Номер п/п	Идентификатор отдела	Название отдела	Сотрудник
4	10	Administration	Jennifer Whalen
	Кол-во сотрудников в отделе	Administration	2
…	…	…	…
110	50	Shipping	Vance Jones
111	50	Shipping	Winston Taylor
	Кол-во сотрудников в отделе	Shipping	45
	Общее кол-во сотрудников в отделах		111

4.8. Сформировать отчёт, содержащий номер отдела, название отдела, имена и фамилии сотрудников, а также их оклады, отсортированные по возрастанию. Отчёт должен иметь следующий вид:

FIRST_NAME	LAST_NAME	SALARY
Отдел №	10	«Administration»
Jennifer	Whalen	4400
Clark	Kent	1000
Отдел №	20	«Marketing»
Max	Smart	1000
Pat	Fay	6000
Michael	Harstein	13000
…	…	…
Отдел №	110	«Accounting»
William	Gietz	8300
Shelley	Higgins	12000

Сотрудников, не приписанных к конкретному отделу, не выводить.

РАЗДЕЛ 5. ЗАДАЧИ НА ИСПОЛЬЗОВАНИЕ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ И ВОЗМОЖНОСТЕЙ ИЕРАРХИИ

Допускается использование справочной информации из разделов 1-18 Приложения.

5.1. Для двух заданных сотрудников найти их ближайшего общего начальника.

Пример результата:

EMP_1	EMP_2	MAN_ID	MAN_NAME
174	178	149	Zlotkey

5.2. Создать регулярное выражение для проверки сложности пароля. Пароль не должен содержать три последовательные буквы латинского алфавита.

Пример корректного пароля: O9gh$drW3

Пример некорректного пароля: O9gh$frW3 (f, g, h — три последовательные буквы латинского алфавита)

5.3. Определить список последовательностей подчиненности от сотрудников, не имеющих менеджера (менеджеров высшего уровня), до сотрудников, не имеющих подчиненных. Результат представить в виде:

MAN_LIST
King->Kochhar->Greenberg->Faviet (не имеет подчинённых)
…

5.4. Дана таблица из двух столбцов ACTION и CODE, в каждом из которых хранятся списки чисел, разделённых пробелами. Создать запрос для разделения данных.

Например, для таблицы:

ACTION	CODE
1000 1100 900	841000 841100 841111
700 500 400	923400 923411

Результат должен быть:

N_LIST	N_POS	ACTION	CODE
1	0	1000 1100 900	841000 841100 841111
	1	1000	841000
	2	1100	841100
	3	900	841111
2	0	700 500 400	923400 923411
	1	700	923400
	2	500	923411
	3	400	

В результате выборки приняты обозначения: N_LIST — порядковый номер списка в исходной таблице, N_POS — порядковый номер числа в списке.

5.5. Дана таблица со следующей структурой:

Id	Linked_id	Part
0	-1	Оглавление
1	0	Глава 1
2	1	Часть 1
3	1	Часть 2
4	0	Глава 2
5	4	Часть 1
6	4	Часть 2

(количество глав и частей произвольное). Создать запрос для вывода оглавления в виде:

Оглавление

1 Глава 1

1.1 Часть 1

1.2 Часть 2

2 Глава 2

2.1 Часть 1

2.2 Часть 2

5.6. Для каждого отдела из таблицы Departments отобразить в виде одной строки с запятой в качестве разделителя фамилии сотрудников, работающих в нем.

Пример результата:

DEPARTMENT_ID	EMP_LIST
10	Kent,Whalen
20	Fay,Hartstein,Smart
…	…
280	(null)

5.7. В произвольной строке, состоящей из символьных элементов, разделенных запятыми, отсортировать элементы по алфавиту.

Пример результата:

Исходная строка	Результат
abc,cde,ef,gh,mn,test,ss, df,fw,ewe,wwe	abc,cde,df,ef,ewe, fw,gh,mn,ss,test,wwe

5.8. Создать запрос для вывода всех дат, отсутствующих в некоторой последовательности дат. Граница последовательности дат должна определяться некоторым (в данном случае максимальным) значением даты.

Например, для таблицы:

DATE_LIST
16-11-2008
18-11-2008
19-11-2008
23-11-2008

Результат должен быть:

DATE_MISS
17-11-2008
20-11-2008
21-11-2008
22-11-2008

5.9. Имеется таблица с некоторым набором чисел. Для любого заданного пользователем числа необходимо запросом определить, в какой диапазон чисел из таблицы оно попало, и округлить до ближайшей границы этого диапазона.

Например, если дана таблица

ID
10000
30000
45000
55000
70000

Результат должен быть:

Заданное число	Результат
3300	Вне заданных диапазонов
47800	45000

5.10. Предположим, что на билетах для проезда в городском транспорте указывают десятичный номер с заданным чётным количеством разрядов n и буквенно-цифровую серию билетной катушки. Нумерация билетов каждой катушки начинается с 000001.

Существует примета, что билет, у которого сумма $^{n}/_{2}$ первых цифр номера равна сумме $^{n}/_{2}$ последних цифр — это «Счастливый билет». Одной командой SELECT вывести количество «счастливых билетов» в каждой серии (одно число). Решение проверить для n = {2, 4, 6}.

Пример результата:

N	CNT
6	55251

5.11. Определить ближайший к заданной дате год, когда 29 февраля придется на воскресенье.

Пример результата:

Заданная дата	Результат
01.10.2012	2020

5.12. Из заданных наборов символов исключить те наборы символов, которые отличаются только порядком.

Например, для таблицы:

LIST
rtyew
trwye
rtgbfdsa
tgrbafsd

Результат должен быть:

RESULT
rtyew
rtgbfdsa

5.13. Создать запрос для определения сумм окладов сотрудников от сотрудников, не имеющих менеджера (менеджеров высшего уровня), до сотрудников, не имеющих подчиненных.

Пример результата:

MAN_LIST	SUM_SAL
King->Kochhar->Greenberg->Faviet	62008
…	…
King->Hartstein->Fay	43000

5.14. Создать запрос для определения сумм окладов двух заданных сотрудников, находящихся в явном или неявном подчинении. Если заданы два сотрудника, не подчиняющиеся друг другу, то выводить фамилии и оклады каждого из них.

Например,

1) если заданы сотрудники с идентификаторами 105 и 121 (не подчиняются друг другу), то результат должен быть:

EMP_ID	SALARY
105, 121	4800, 8200

2) если заданы сотрудники с идентификаторами 105 и 102 (105-й сотрудник подчиняется 103-му, а 103-й — 102-му), то результат должен быть:

EMP_ID	SALARY
102, 105	21800

РАЗДЕЛ 6. ЗАДАЧИ НА ИСПОЛЬЗОВАНИЕ АНАЛИТИЧЕСКИХ ФУНКЦИЙ И РАЗДЕЛА MODEL

Допускается использование справочной информации из разделов 1-19 Приложения.

6.1. Для произвольной строки, состоящей из чисел, разделенных заданным разделителем, получить строку, отображающую эти числа в обратном порядке.

Пример результата:

Исходная строка	Результат
0|0|1|2|1|2|10|22|34|15|0|105|66|73	73|66|105|0|15|34|22|10|2|1|2|1|0|0

6.2. Для двух произвольных списков определить список тех элементов, которые не присутствуют одновременно в двух списках.

Например, если введены списки:

a, df, u,123,www,r

и

r,df,123,u, pp

Результат должен быть:

Результат
a, www, pp

6.3. Для произвольного сотрудника из таблицы Employees определить общее количество его подчиненных (всех уровней).

Пример результата

EMP_ID	SUB_CNT
100	106
102	5

6.4. По заданной таблице родственных отношений определить для каждого человека количество родственников (родных/двоюродных братьев и сестёр).

Например, для таблицы родственных отношений:

ID	Имя	Пол	ID_родителя
1	Иван	м	2
2	Петр	м	3
3	Екатерина	ж	
4	Мария	ж	2
5	Анастасия	ж	3
6	Павел	м	5

Результат должен быть:

ID	Имя	Кол_во родных братьев	Кол_во родных сестер	Кол_во двоюр. братьев	Кол_во двоюр. сестер
1	Иван	0	1	1	0
2	Петр	0	1	0	0
3	Екатерина	0	0	0	0
4	Мария	1	0	1	0
5	Анастасия	1	0	0	0
6	Павел	0	0	1	1

6.5. Для всех таблиц схемы определить списки имен вторичных ключей без индексов (через запятую).

Пример результата:

TABLE_NAME	FK_LIST
COUNTRIES	COUNTR_REG_FK
DEPARTMENTS	DEPT_LOC_FK,DEPT_MGR_FK
…	…
JOB_HISTORY	JHIST_DEPT_FK,JHIST_EMP_FK,JHIST_JOB_FK

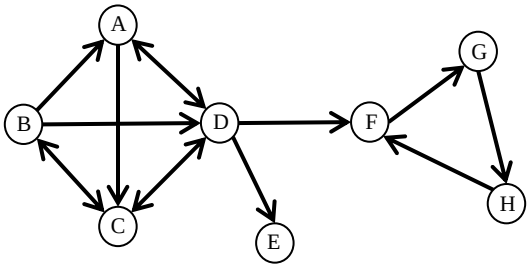
6.6. Связи между вершинами графа заданы символьным выражением, состоящим из двух частей, разделённых символом «->». Каждая часть выражения может содержать произвольное количество вершин графа, перечисленных через запятую. При этом предполагается, что каждая вершина из правой части

связана направленным отношением со всеми вершинами, указанными в левой части. Например, запись a,b->c,d означает: a->c, a->d, b->c, b->d.

Создать запрос для определения всех имеющихся циклов.

Например, для таблицы:

RELATION
a,b -> c,d
b,d -> a
c -> b,d
d -> a,c,e,f
f -> g
g -> h
d,h -> f



Результат должен быть:

CYCLE
a->c->b->a
b->d->c->b
c->d->a->c
a->d->c->b->a
f->g->h->f
a->c->b->d->a

6.7. Определить сумму цифр в произвольной символьной строке. Если результат больше 10, то процесс продолжить — снова найти сумму цифр и так далее — пока не получится одна цифра.

Пример результата:

Исходная строка	Результат
t29h8n3m7l88g799999999999927	7

Пояснение: сумма всех цифр, входящих в исходную строку — 169; 1 + 6 + 9 = 16; 1 + 6 = 7

6.8. Имеется таблица с числовым столбцом первичного ключа. Для каждого из значений в этом столбце требуется получить в виде горки: первая часть строки — перечисление всех чисел, меньших заданного в порядке возрастания, вторая — в порядке убывания, в середине — само число.

Например, для таблицы:

ID
3
11
1
8
9
6
23

Результат должен быть:

ID	Result
1	1
3	1,3,1
6	1,3,6,3,1
8	1,3,6,8,6,3,1
9	1,3,6,8,9,8,6,3,1
11	1,3,6,8,9,11,9,8,6,3,1

РАЗДЕЛ 7. ЗАДАЧИ ПРОИЗВОЛЬНОЙ ТЕМАТИКИ

Допускается использование любой справочной информации из Приложения, если её применение не ограничено явно условием задачи.

7.1. Проверить наличие циклов в таблице подчиненностей. Вывести циклические зависимости в строчку в виде Номер1.Имя1->Номер2.Имя2->…Номер1.Имя1, начиная с первого по алфавиту имени.

Например, для таблицы подчинённостей:

Номер	Имя	Номер_начальника
1	Алексей	2
2	Пётр	3
3	Павел	4
4	Иван	2
5	Кристина	3
6	Андрей	5

Результат должен быть:

CYCLE
4.Иван->3.Павел->2.Пётр->4.Иван

7.2. Создать запрос для вывода информации о всех сотрудниках в виде:

Фамилия	King	Cochar	De Haan	…	…
Оклад	24000	17000	10000	…	…
Общее кол-во подчинённых	8	5	3	…	…

7.3. Вывести информацию о таблицах схемы в виде:

Имя таблицы	Столбцы первичного ключа	Столбцы с ограничением уникальности	Список подчиненных таблиц со столбцами вторичных ключей
Table 1	Col1, Col2	Col3, Col4	Table2 (Col5, Col6), Table3 (Col7,Col8)

Пример результата:

Имя таблицы	**Столбцы первичного ключа**	**Столбцы с ограничением уникальности**	**Список подчиненных таблиц со столбцами вторичных ключей**
DEPT3	DEPARTMENT_ID, DEPARTMENT_NAME	-	DEPT2(ID,NAME)
EMPLOYEES	EMPLOYEE_ID	EMAIL	DEPARTMENTS(MANAGER_ID), EMPLOYEES(MANAGER_ID), JOB_HISTORY(EMPLOYEE_ID)
EMPLOYEES	EMPLOYEE_ID	FIRST_NAME, LAST_NAME	DEPARTMENTS(MANAGER_ID), EMPLOYEES(MANAGER_ID), JOB_HISTORY(EMPLOYEE_ID)
…	…	…	…

7.4. Дана таблица, содержащая информацию о структуре некоторого иерархического дерева. Создать запрос для определения всех путей от всех начальных вершин до всех конечных вершин по направленному дереву (от родительской вершины к дочерней). В пределах одного пути необходимо перемножить значения поля Количество и просуммировать результаты по всем путям от начальной вершины до конечной вершины.

Например, для таблицы:

Номер дочерней вершины	Номер родительской вершины	Количество
2	1	2
5	1	2
3	2	1
4	2	2
4	6	2
6	5	3
4	3	2

Результат должен быть:

Начальная вершина	Конечная вершина	Количество путей	Сумма произведений
1	3	1	2
1	4	3	20
2	4	2	4
…	…	…	…

Пояснение на примере второй строки результата: от вершины 1 до вершины 4 имеется три пути — 1, 2, 4; 1, 2, 3, 4 и 1, 5, 6, 4. Соответствующие произведения: 2 * 2 = 4; 2 * 1 * 2 = 4; 2 * 3 * 2 = 12. Сумма — 4 + 4 + 12 = 20.

7.5. Имеется таблица со списком объектов, характеризующихся номером, объемом и весом. Создать запрос для определения списка различных объектов, для которых:

• суммарный объем будет ближе всех к предельной величине объёма при максимальном весе, но не будет превышать её;

• суммарный вес не будет превышать максимальный вес;

• если при одинаковом суммарном объёме получаются различные значения суммарного веса, решение должно быть принято в пользу меньшего.

Список не должен состоять из единственного объекта с параметрами, соответствующими заданным максимальному весу и объёму.

Задачу решить с использованием раздела MODEL.

Например, для таблицы:

Номер	Объем	Вес
1	77	2
2	99	1
3	19	3
4	91	1
5	6	3
6	6	1
7	3	1
8	92	6

Результат должен быть:

Максимальный вес	Предельный объём	Список объектов
6	92	1, 6, 7

Пояснение: максимальному значению веса в таблице (6) соответствует объём 92. Ближе всех к данному значению объёма будет сумма 77 + 6 + 3 = 86, которой соответствует два списка объектов: 1, 5, 7 и 1, 6, 7. Т.к. значение суммарного веса для списка 1, 6, 7 меньше, то необходимо выбрать его.

7.6. Создать таблицу следующего вида с ключевым столбцом Номер:

Номер	Количество
1	2000
3	2500
6	3000
10	1000

Создать запрос для получения регрессионной оценки количества при пропущенных значениях в столбце Номер.

Результат представить в виде:

Номер	Количество	Расчетные значения
1	2000	2000
2		2250
3	2500	2500
4		2666.66
5		2833.33
6	3000	3000
7		2500
8		2000
9		1500
10	1000	1000

7.7. Создать запрос, который позволит получать все комбинации сумм для чисел, содержащихся в числовом столбце таблицы. Таблица может содержать любое число чисел.

а) задачу решить без использования раздела Model

б) задачу решить с использованием раздела Model

Например, для таблицы:

ID	A
1	7
2	12
3	31

Результат должен быть:

ID	Слагаемые	Сумма
1	7+12	19
2	7+31	38
3	12+31	43
4	7+12+31	50

7.8. Написать запрос к таблице EMPLOYEES, который вернёт оклад сотрудника с заданным номером, а в случае отсутствия сотрудника с указанным номером — вернёт 0 (ноль).

Пример результата:

ID	Salary
105	4800
208	0

7.9. Для всех таблиц схемы определить списки имен вторичных ключей без индексов (через запятую).

TABLE_NAME	FK_WITHOUT_INDEX
COUNTRIES	COUNTR_REG_FK
DEPARTMENTS	DEPT_LOC_FK,DEPT_MGR_FK
…	…

7.10. Определить, сколько раз каждая из цифр от 0 до 9 встречается в столбце Phone_number таблицы Employees.

Пример результата:

0	1	2	3	4	5	6	7	8	9
15	12	23	45	24	33	45	12	30	15

Второй вариант условия — определить цифры, которые максимальное количество раз встречаются в столбце Phone_number таблицы Employees.

Пример результата:

MAX_CNT	*NUM*
212	*1*
212	*4*

7.11. Одной командой SELECT выбрать из таблицы CUSTOMERS базовой схемы OE сведения о покупателях из той страны, в которой не расположено ни одного отдела компании. Страной покупателя считать ту, что указана в адресе покупателя. Сведения о расположении отделов компании взять из таблиц схемы HR.

В результат вывести:

1) идентификатор покупателя;

2) фамилию покупателя;

3) номера телефонов покупателя через запятую (без указания типа данных);

4) страну, указанную в адресе покупателя.

Результат отсортировать по идентификатору покупателя по возрастанию.

Пример результата:

CUSTOMER_ID	CUST_LAST_NAME	PHONE_NUMBERS	COUNTRY_ID
308	Dunaway	+86 811 012 4093	CN
309	Bates	+86 123 012 4095	CN
323	Falk	+91 80 012 4123, +91 80 083 4833	IN
…	…	…	…

7.12. Вычислить число Пи с точностью 40 знаков после запятой. Для расчета использовать быстро сходящуюся последовательность:

$$\pi = \sum_{k=0}^{\infty} \left[\frac{1}{16^{k}} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right) \right]$$

Пример результата:

Pi
3,14159265358979323846264338327950288419 71

7.13. Задана произвольная символьная строка, состоящая из двух частей, разделенных символами «=>». В левой и правой части выражения содержатся символьные строки, разделенные запятыми.

Требуется создать запрос, который будет выводить все возможные пары комбинаций из левой и правой частей.

Пример результата для строки a, fgf,yy=>uu,gh:

PATH
a=>uu
a=>gh
fgf=>uu
fgf=>gh
yy=>uu
yy=>gh

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Генник Дж. SQL. Карманный справочник : [пер. с англ.] / Дж. Генник. — СПб. : Питер, 2004. — 224 с.

2. Мишра С. Секреты Oracle SQL : [пер. с англ.] / С. Мишра, А. Бьюли. — СПб. : Символ-Плюс, 2003. — 368с.

3. Молинаро Э. SQL: сборник рецептов : [пер. с англ.] / Э. Молинаро. — СПб. : Символ-Плюс, 2009. — 672 с.

4. Прайс Д. Oracle Database 10g. Операторы SQL и программы PL/SQL : [пер. с англ.]. / Д. Прайс. — М. : Лори, 2007. — 566 с.

5. Пржиялковский В. В. Введение в Oracle SQL : учеб. пособие / В. В. Пржиялковский — М. : Национальный открытый Университет “ИНТУИТ”: БИНОМ, Лаборатория знаний, 2011. — 319 с.

6. Селко Дж. Стиль программирования Джо Селко на SQL : [пер. с англ.] / Дж. Селко. — СПб. : Русская Редакция, Питер, 2006. — 196с.

7. Селко Дж. SQL для профессионалов. Программирование : [пер. с англ.] / Дж. Селко. — М. : Лори, 2009. — 442 с.

8. Сичкаренко В. А. SQL-99. Руководство разработчика баз данных / В. А. Сичкаренко. — СПб. : ДиаСофтЮП, 2002. — 816 с.

9. Gennick J. Oracle Regular Expressions. Pocket Reference / J. Gennick, P. Linsley. — O’Relly Media, 2003. — 64 p.

10. Price J. Oracle Database 11g SQL / J. Price. — Oracle Press, McGraw-Hill, 2008. — 656 p.

11. Mishra S. Mastering Oracle SQL / S. Mishra, A. Beaulieu. — 2nd Edition. — O'Reilly Media, 2004. — 496 p.

12. Oracle® Database SQL Language Reference, 11g Release 2 (11.2). — Oracle Press, 2012.

13. Oracle® Database Utilities 11g Release 2 (11.2). — Oracle Press, 2012.

14. Oracle® Database Data Warehousing Guide 11g Release 2 (11.2) — Oracle Press, 2011.

15. Oracle® Database Reference 11g Release 2 (11.2). — Oracle Press, 2012.

16. Хомич Л. Диалект Oracle SQL: Model в примерах / Л. Хомич // Веб-Аналитик.ИНФО. — 2010. — № 9 (6).

ПРИЛОЖЕНИЕ. СПРАВОЧНАЯ ИНФОРМАЦИЯ

1. ПРИОРИТЕТ РАЗДЕЛОВ КОМАНДЫ SELECT

Общий порядок записи разделов команды SELECT показан в таблице ниже[1]:

№ п/п	Оператор или группа операторов
1	WITH
2	SELECT
3	DISTINCT
4	FROM
5	PIVOT/UNPIVOT
6	WHERE
7	CONNECT BY
8	GROUP BY
9	HAVING
10	MODEL
11	ORDER BY

Обычный логический порядок обработки разделов команды SELECT сервером Oracle представлен в следующей таблице:

Приоритет	Раздел / ключевое слово	Примечание
1	WITH	
2	FROM	Если указано несколько таблиц, производится их соединение в соответствии с определёнными условиями.
3	PIVOT/UNPIVOT	
4	CONNECT BY	Использование в случае внешнего

[1] Подробнее смотри в документации [12]

<table>
<tr><th>Приоритет</th><th>Раздел / ключевое слово</th><th>Примечание</th></tr>
<tr><td></td><td></td><td>соединения нескольких таблиц возможно при условии, что для соединения используется синтаксис ANSI SQL — с применением ключевого слова JOIN.</td></tr>
<tr><td>5</td><td>WHERE</td><td>Значение rownum присваивается строкам выборки на данном этапе — т.е. после осуществления фильтрации, если она была задана. При этом следует помнить, что Oracle не гарантирует вывод строк в каком-либо определённом порядке без использования явной сортировки с помощью ORDER BY, которая производится в самом конце.
Таким образом, запрос
SELECT rownum rn, employee_id
FROM employees
ORDER BY employee_id;
может вернуть, например, такой результат:
<table>
<tr><th>RN</th><th>EMPLOYEE_ID</th></tr>
<tr><td>3</td><td>101</td></tr>
<tr><td>1</td><td>102</td></tr>
<tr><td>2</td><td>103</td></tr>
<tr><td>4</td><td>104</td></tr>
</table>
</td></tr>
<tr><td>6</td><td>GROUP BY</td><td></td></tr>
<tr><td>7</td><td>HAVING</td><td></td></tr>
<tr><td>8</td><td>MODEL</td><td></td></tr>
</table>

Приоритет	Раздел / ключевое слово	Примечание
9	ORDER BY	ASC — по возрастанию (используется по умолчанию) или DESC — по убыванию. Может использоваться в сочетании с NULLS FIRST — значения NULL отобразить первыми или NULLS LAST — значения NULL отобразить последними. Без явного указания места NULL-значений при сортировке по возрастанию они выводятся последними, при сортировке по убыванию — первыми.
10	DISTINCT	При использовании этого ключевого слова в разделе ORDER BY разрешается ссылаться только на столбцы из списка SELECT
11	SELECT	Вывод итоговых результатов

Сначала производится выборка и соединение (при необходимости) строк из таблиц, указанных в разделе FROM. Затем осуществляется транспонирование строк в столбцы (или столбцов в строки) — в случае наличия раздела PIVOT (или UNPIVOT). После этого происходит отбор строк для составления дерева иерархии — в случае наличия раздела CONNECT BY. Далее производится отбор строк по условию, указанному в разделе WHERE и присваивание отобранным строкам значения rownum (при необходимости). На следующем шаге отобранные строки объединяются в группы по критерию, указанному в разделе GROUP BY, после чего может быть осуществлена фильтрация строк на основе значений агрегатных функций для

полученных групп — в соответствии с условием, указанным в разделе HAVING.

Одним из последних обрабатывается раздел MODEL. В последнюю очередь производится сортировка полученного множества строк по условию, указанному в разделе ORDER BY, удаление дубликатов, если задано ключевое слово DISTINCT и, наконец, вывод результатов — в соответствии с последовательностью столбцов и выражений указанных в разделе SELECT.

При наличии в команде SELECT раздела WITH, запросы в нём содержащиеся, обрабатываются прежде всего. Oracle рассматривает результаты таких запросов как внутренние представления или временные таблицы.

2. ПРИОРИТЕТ ОПЕРАТОРОВ И УСЛОВИЙ

Порядок обработки операторов и условий SQL представлен в следующей таблице

Приоритет	**Оператор или группа операторов**	**Примечание**
1	Арифметические операторы	/, * +, -
2	Оператор конкатенации	‖
3	Условия сравнения группы 1	=,!=, <, >, <=, >=
4	Условия сравнения группы 2	IS [NOT] NULL, LIKE [NOT] BETWEEN, [NOT] IN, EXISTS
5	Логическое условие NOT	
6	Логическое условие AND	
7	Логическое условие OR	

Правила приоритета можно переопределять с помощью скобок. В отсутствии скобок операторы и условия с одинаковым приоритетом в пределах одного выражения обрабатываются слева направо (данный порядок не гарантируется в случае обработки нескольких одинаковых операторов AND или OR подряд).

3. ИСПОЛЬЗОВАНИЕ ПОДСТАНОВОЧНЫХ ПЕРЕМЕННЫХ

Подстановочные переменные позволяют сохранять значение временных переменных, включаемых в команду SQL.

Переменную можно создать заранее и присвоить ей значение — с помощью команды DEFINE. В этом случае её значение сохраняется на весь период текущей сессии. Для сброса значения переменной используется команда UNDEFINE.

Имена переменных в командах SQL записывают с префиксом в виде одного или двух амперсандов (&)[1]. Если переменная не существует (ей не было присвоено значение с помощью команды DEFINE), SQL*Plus или SQL Developer запрашивают значение у пользователя:

- в случае записи с одним амперсандом — каждый раз, когда переменная с данным именем встретится при обработке данного или последующих запросов, так как при такой форме записи значение переменной аннулируется сразу после использования

[1] Имена строковых переменных и дат вместе с префиксом должны быть заключены в апострофы, если не предполагается сделать это во время ввода значения переменной пользователем.

• в случае записи с двумя амперсандами[1] — только первый раз. Так же, как и при использовании команды DEFINE, введённое значение сохраняется и используется в течение текущей сессии каждый раз при использовании переменной с таким именем — в т. ч. во всех последующих запросах. Для сброса значения переменной используется команда UNDEFINE.

Примеры:

```
DEFINE emp_id=102;
SELECT last_name
FROM employees
WHERE employee_id=&emp_id;
```

```
old:SELECT last_name
FROM employees
WHERE employee_id=&emp_id
new:SELECT last_name
FROM employees
WHERE employee_id=102
```

LAST_NAME
De Haan

```
UNDEFINE dt;
SELECT ROUND(MONTHS_BETWEEN(TO_DATE('&&dt',
'dd.mm.yyyy'), SYSDATE)) period, '&dt' var
FROM dual;
```

```
old:SELECT ROUND(MONTHS_BETWEEN(TO_DATE('&&dt',
'dd.mm.yyyy'), SYSDATE)) period, '&dt' var
FROM dual
```

[1] Имя переменной с префиксом в виде двух амперсандов достаточно записать только один раз, далее используется имя переменной с префиксом в виде одного амперсанда.

new:SELECT
ROUND(MONTHS_BETWEEN(TO_DATE('22.10.1984',
'dd.mm.yyyy'), SYSDATE)) period, '22.10.1984' var
FROM dual

PERIOD	VAR
-336	22.10.1984

Далее приводятся основные сведения о различных видах функций и некоторые примеры. Более подробное описание и примеры для каждой из функций можно уточнить в документации [12].

4. ОДНОСТРОЧНЫЕ ФУНКЦИИ: ОБЩИЕ

Синтаксис и описание общих однострочных функций приводятся в следующей таблице:

Функция	Описание
Функции для работы со значениями NULL	
NVL(expr1, expr2)	Если expr1 не NULL, будет возвращено его значение; если expr1 — NULL, будет возвращено значение expr2. Типы данных аргументов должны соответствовать.
NVL2(expr1, expr2, expr3)	Если expr1 не NULL, будет возвращено значение expr2; если expr1 — NULL, будет возвращено значение expr3. Тип данных expr3 должен соответствовать типу данных expr2.
LNNVL(condition)	Даёт возможность оценить условие condition, в котором один или оба операнда могут быть NULL-значениями. Возвращает TRUE, если результат сравнения по заданному

Функция	Описание
	условию condition — FALSE или UNKNOWN. Возвращает FALSE, если результат сравнения — TRUE. Может использоваться для включения в результат строк, содержащих значение NULL в оцениваемом столбце. Например, запрос: *SELECT DISTINCT department_id* *FROM employees* *WHERE department_id <= 30;* вернёт только те строки, в которых значения в столбце department_id удовлетворяют условию из раздела WHERE, а запрос: *SELECT DISTINCT department_id* *FROM employees* *WHERE LNNVL(department_id > 30);* вернёт также и те строки, в которых значение department_id — NULL. Также данная функция используется ORACLE во внутренних целях — для оценки условий NOT IN и NOT EXISTS.
NULLIF(expr1, expr2)	Сравнивает два выражения и возвращает NULL, если они равны. В противном случае будет возвращено первое из выражений. В качестве аргумента expr1 нельзя задать NULL.
COALESCE(expr1, expr2 [, expr3, …])	Возвращает первое не-NULL выражение из списка. Если значения всех аргументов — NULL функция вернёт NULL.

Функция	Описание
Функции сравнения	
LEAST(expr1, expr2 [, expr3, …])	Возвращает наименьшее выражение из списка.
GREATEST(expr1, expr2 [, expr3, …])	Возвращает наибольшее выражение из списка.

5. ОДНОСТРОЧНЫЕ ФУНКЦИИ: ЧИСЛОВЫЕ

Синтаксис и описание числовых однострочных функций приводятся в следующей таблице:

Функция	Описание	Примеры
ABS(n)	Возвращает абсолютное значение (модуль) числа.	ABS(-5.4) = ABS(5.4) = 5.4
CEIL(n)	Возвращает наименьшее целое число, большее или равное n.	CEIL(7.2) = 8 CEIL(-7.2) = -7
FLOOR(n)	Возвращает наибольшее целое число, меньшее или равное n.	FLOOR(7.2) = 7 FLOOR(-7.2) = -8
MOD(n1, n2)	Возвращает остаток от деления n1 на n2; n1 и n2 могут содержать дробные части.	MOD(11000, 5000) = 1000 MOD(2000, 5000) = 2000 MOD(12.5, 1.5) = 0.5
POWER(n1, n2)	Возвращает значение n1 в степени n2, n1 может быть отрицательным.	POWER(3, 2)
ROUND(n1 [, n2])	Округляет значение столбца или выражения до n2 десятичных разрядов, n2 может быть отрицательным. По умолчанию n2 = 0.	ROUND(45.452, 2) = 45.45 ROUND(45.452, 1) = 45.5 ROUND(3292.3, -2) = 3300

Функция	Описание	Примеры
TRUNC(n1 [, n2])	Сокращает значение столбца или выражения до n2 десятичных разрядов, n2 может быть отрицательным. По умолчанию n2 = 0.	TRUNC(45.452, 2) = 45.45 TRUNC(45.452, 1) = 45.4 TRUNC(3292.3, -2) = 3200

6. ОДНОСТРОЧНЫЕ ФУНКЦИИ: СИМВОЛЬНЫЕ

Синтаксис и описание символьных однострочных функций приводятся в следующей таблице:

Функция	Описание	Пример
Функции, возвращающие символьное значение		
CONCAT(char1, char2)	Возвращает результат конкатенации символьных строк char1 и char2. Эквивалент оператора конкатенации ‖.	CONCAT('abc', 34) = abc34
INITCAP(char)	Преобразует первую букву каждого слова в прописную, а остальные — в строчные. Разделителем слов считается пробельный символ или не буквенно-цифровой символ.	INITCAP('one day') = One Day INITCAP('one_day') = One_Day
LOWER(char)	Преобразует буквенные символы в строчные буквы.	LOWER('FLOWER') = flower

Функция	Описание	Пример
UPPER(char)	Преобразует буквенные символы в прописные буквы.	UPPER(‘flower’) = FLOWER
TRIM([{ { LEADING | TRAILING | BOTH } [trim_character] | trim_character } FROM] trim_source)	Удаляет из символьной строки заданный символ trim_character. При использовании ключевого слова LEADING удаляются все вхождения символа в начале строки, TRAILING — в конце, BOTH — и в начале, и в конце (этот вариант используется по умолчанию). Если trim_character или trim_source — NULL, то функция вернёт NULL. По умолчанию значение trim_character — пробел.	TRIM(TRAILING ‘.’ FROM ‘…Such a wonderful day…’) = …Such a wonderful day TRIM (‘‘ FROM ‘abc’) = NULL
LTRIM(char [, set])	Удаляет из символьной строки char все значения из набора set. Сканирование строки начинается с первого символа и	LTRIM(‘----one’, ‘-’) = one LTRIM(‘one’, ‘two’) = ne

Функция	Описание	Пример
	заканчивается по достижению первого символа, не входящего в набор set. По умолчанию значение set — пробел.	
RTRIM(char [, set])	Удаляет из символьной строки char все значения из набора set. Сканирование строки начинается с последнего символа и заканчивается по достижению первого символа, не входящего в набор set. По умолчанию значение set — пробел.	RTRIM('lalaley', 'la') = lalaley RTRIM('lalaley', 'ley') = lala
SUBSTR(char, position [, substring_length])	Возвращает символы строки char в количестве substring_length, начиная с позиции position. Если значение position отрицательно, начальная позиция отсчитывается с конца строки, если значение	SUBSTR('All you need…', 5) = you need… SUBSTR('All you need…', -7, 4) = need

Функция	Описание	Пример
	position равно 0, то оно рассматривается как 1. Если substring_length опущено, возвращаются все символы до конца строки.	
REPLACE(char, search_string [, replacement_string])	Символьная строка char просматривается в поисках заданной символьной строки search_string, каждое найденное вхождение которой заменяется указанной строкой подстановки replacement_string. Если значение replacement_string опущено или NULL, все вхождения search_string будут удалены. Если search_string — NULL, будет возвращена исходная строка char.	REPLACE('Jack and Jue', 'J', 'Bl') = Black and Blue REPLACE('ab02cd023', '02') = abcd3
TRANSLATE(expr, from_string, to_string)	Возвращает выражение expr, в котором каждый символ из набора	TRANSLATE('AaBbCc', 'bca', '123') = A3B1C2 TRANSLATE('one_2two_5three784', 'x0123456789', 'x')

Функция	Описание	Пример
	from_string заменён на соотвествующих (находящийся на такой же позиции) символ из набора to_string. Если какой-либо символ встречается в from_string несколько раз, то он будет заменён на первое соответствующее вхождение в to_string. Аргумент from_string может содержать большее количество символов, чем to_string, в этом случае символы из набора from_string, для которых нет соответствия в to_string будут удалены из исходной строки expr. Однако нельзя использовать в качестве to_string пустую строку для удаления всех символов набора from_string из исходного выражения	= one_two_three

Функция	Описание	Пример
	expr, так как сервером Oracle пустая строка трактуется как NULL и функция вернёт значение NULL. Если необходимо удалить из исходной строки определённый набор символов, дополните его другим символом, который будет использоваться как значение аргумента to_string (например, TRANSLATE(expr, 'x0123456789', 'x' — для удаления всех цифр).	
LPAD(expr1, n [, expr2])	Возвращается выражение expr1, дополненное слева до длины n символами, определяемыми заданной строкой expr2. Expr1 не может быть NULL или пустой строкой. По умолчанию expr2 — пробел.	LPAD('level', 7, '*') = **level
RPAD(expr1, n [, expr2])	Возвращается выражение expr1,	RPAD('level', 7, '*') = level**

Функция	Описание	Пример
	дополненное справа до длины n символами, определяемыми заданной строкой expr2. Expr1 не может быть NULL или пустой строкой. По умолчанию expr2 — пробел.	
Функции, возвращающие числовое значение		
INSTR(string, substring [, position [, occurrence]])	Возвращает номер позиции заданной строки substring. Можно указать начальную позицию для поиска (position) и номер вхождения (occurence). По умолчанию position и occurence считаются равными 1, т. е. поиск начинается с начала строки и возвращается первое вхождение substring. Если значение position отрицательно, позиция начала поиска отсчитывается с конца строки (и	INSTR('Blablabla', 'bla', 1, 2) = 4 INSTR('Blablabla', 'bla', -2, 1) = 7

Функция	Описание	Пример
	поиск первого символа строки substring идёт справа налево). Значение position не может быть равно 0. Значение occurence должно быть положительно.	
LENGTH(char)	Возвращает количество символов в символьной строке.	LENGTH('Hello world') = 11

7. ОДНОСТРОЧНЫЕ ФУНКЦИИ: ФУНКЦИИ ДЛЯ РАБОТЫ С ДАТОЙ И ВРЕМЕНЕМ

Синтаксис и описание функций для работы с датой и временем приводятся в следующей таблице:

Функция	Описание	Примеры
SYSDATE	Возвращает текущую дату и время операционной системы сервера базы данных	
ADD_MONTHS(date, n)	Прибавляет n календарных месяцев к дате date. Значение n должно быть целым и может быть отрицательным	ADD_MONTHS('31.10.2012', 1) = 30.11.2012 ADD_MONTHS('31.10.2012', -13) = 31.09.2011

Функция	Описание	Примеры
	(в случае вычитания месяцев).	
MONTHS_BETWEEN(date 1, date2)	Определяет число месяцев между датами date1 и date2. Результат может быть как положительным, так и отрицательным (если date1 раньше date2). Дробная часть результата представляет соответствующую часть месяца.	MONTHS_BETWEEN (’31.10.2012’, ’30.09.2012’) = 1 MONTHS_BETWEEN (SYSDATE, ’30.11.2012’) = -1,8541483…
LAST_DAY(date)	Возвращает дату последнего дня того месяца, к которому относится дата date.	LAST_DAY('02.02.2012') = 29.02.2012
NEXT_DAY(date, char)	Возвращает дату следующего указанного дня недели char после даты date. Значение char может числовым представлением дня или	NEXT_DAY('30.12.2012', 3) = 02.01.2013 NEXT_DAY(SYSDATE, ‘Понедельник’) = 08.10.2012

Функция	Описание	Примеры
	символьной строкой (аббревиатурой или полным названием дня недели). Порядковый номер дня недели и язык определяются параметрами текущего сеанса (например, при установленном регионе Израиль вторым днём недели будет считаться понедельник).	
ROUND(date [, fmt])	Возвращает дату date, округлённую до единицы, заданной моделью формата fmt. Если модель формата опущена, округление происходит до ближайшего дня.	ROUND('18.10.2012', 'MONTH') = 01.11.2012 ROUND('18.10.2012', 'YEAR') = 01.01.2013
TRUNC(date [, fmt])	Возвращает дату date, в которой	TRUNC('18.10.2012', 'MONTH') = 01.10.2012

Функция	Описание	Примеры
	значение времени сокращено путём усечения до единицы, заданной моделью формата fmt. Если модель формата опущена, сокращение происходит до начала дня.	TRUNC('18.10.2012', 'YEAR') = 01.01.2012

8. ОДНОСТРОЧНЫЕ ФУНКЦИИ: ЯВНОЕ ПРЕОБРАЗОВАНИЕ ТИПОВ ДАННЫХ

Синтаксис и описание функций преобразования приводятся в следующей таблице:

Функция	Описание
TO_CHAR({ datetime | number } [, fmt [, 'nlsparam']])	Преобразует значение даты datetime или числа number в строку символов VARCHAR2 в соответствии с необязательной маской формата fmt. При преобразовании числа параметр nlsparam используется для указания следующих символов: • разделитель целой и дробной части числа • разделитель групп • локальный символ валюты • международный символ валюты

Функция	Описание
	При преобразовании даты параметр nlsparam используется для указания языка представления возвращаемых имён месяцев и дней и их аббревиатур. Для всех опущенных параметров по умолчанию используются параметры текущего сеанса.
TO_NUMBER(char [, fmt [, 'nlsparam']])	Преобразует состоящую из цифр строку символов char в число, используя необязательный параметр маски формата fmt. Назначение параметра nlsparam то же, что и в функции TO_CHAR для преобразования числа. Для всех опущенных параметров по умолчанию используются параметры текущего сеанса.
TO_DATE(char [, fmt [, 'nlsparam']])	Преобразует строку символов char, представляющую дату, в значение даты в соответствии с необязательной маской формата fmt. Назначение параметра nlsparam то же, что и в функции TO_CHAR для преобразования даты. Для всех опущенных параметров по умолчанию используются параметры текущего сеанса.

9. МАСКИ (МОДЕЛИ) ФОРМАТА

Маска формата должна быть заключена в одиночные кавычки. Элементы маски формата, предназначенные для вывода текста, чувствительны к регистру:

- если первый символ элемента введён в нижнем регистре, все символы соответствующего результата будут выведены в нижнем регистре;
- если первые два символа элемента введены в верхнем регистре, все символы соответствующего результата будут выведены в верхнем регистре;
- если первый символ элемента введён в верхнем регистре, а второй — в нижем, то первый символ соответствующего результата будет выведен в верхнем регистре, а остальные — в нижнем.

Описание элементов масок формата приводятся в следующей таблице:

Элемент маски	**Описание**	**Пример маски**	**Результат**
Элементы маски формата функций TO_NUMBER и TO_CHAR для чисел			
9	Позиции цифр (количество девяток определяет ширину области отображения)	999999	1234
0	В начале: отображение одного и более начальных нулей. В конце: отображение отсутствующей целой части в виде нуля.	099999 09.99 90.99	001234 00.12 0.12
,	Запятая в указанной позиции (разделение групп разрядов). Можно использовать несколько запятых в одной маске.	999,999	1,234
G	Возвращает символ разделителя групп в указанной позиции (зависит от значения параметра NLS_NUMERIC_CHARACTE	9G999	1,234

Элемент маски	Описание	Пример маски	Результат
	R). Можно указать несколько разделителей групп в одной маске.		
.	Десятичная точка в указанной позиции (отображение дробной части). Разрешено использовать только одну десятичную точку в одной маске.	999999.99	1234.00
D	Возвращает символ десятичного разделителя в указанной позиции (зависит от значения параметра NLS_NUMERIC_CHARACTER). Разрешено использовать только один символ десятичного разделителя в одной маске.	999999D99	1234.00
EEEE	Отображение числа в научной нотации	99.999EEEE	1.234E+03
$	Отображение символа доллара перед числом.	$999999	$1234
L	Отображение локального символа валюты в указанной позиции.	L999999 999999L	р.1234 1234р.
MI	Отображение знака минуса для отрицательных чисел или пробела для положительных— справа от числа. Данный элемент можно использовать только в конце маски формата.	9999MI	1234-
U	Отображение в указанной	9999U	1234€

Элемент маски	Описание	Пример маски	Результат
	позиции символа евро или другого, заданного значением параметра NLS_DUAL_CURRENCY.		
V	Возведение в степень n по основанию 10 (значение n равно числу цифр 9 после V)	9999V99	123400
S	Отображение соответствующего знака для положительного или отрицательного числа (на первой или последней позиции).	S9999 9999S	+1234 1234+
Элементы маски формата функции TO_DATE или TO_CHAR для дат и времени			
/ , . ; :	Отображение знаков пунктуации	to_char (sysdate, 'dd:mm; yyyy')	22:10; 2012
"text"	Отображение текста (всего, что находится между двойными кавычками)	to_char (sysdate, "“The” Day)	The Monday
YYYY или SYYYY	Год. Для дат до н.э. используется префикс -	to_char(sysdate, YYYY)	2012
Y YY YYY	Последние одна, две или три цифры года.	to_char(sysdate, 'Y') to_char(sysdate, 'YY') to_char(sysdate, 'YYY')	2 12 012
YEAR или SYEAR	Год прописью. Для дат до н.э. используется префикс -	to_char(sysdate, 'Year')	Twenty Twelve

Элемент маски	Описание	Пример маски	Результат
Q	Квартал года.	to_char(sysdate, 'Q')	4
MM	Номер месяца в виде двузначного числа	to_char(sysdate, 'mm')	11
MONTH	Название месяца, дополненное пробелами до общей длины в 9 символов	to_char(sysdate, 'MONTH')	NOVEMBER·
MON	Трёхбуквенное сокращение названия месяца	to_char(sysdate, 'Mon')	Nov
RM	Номер месяца римскими цифрами.	to_char(sysdate, 'RM')	XI
WW или W	Порядковый номер недели в году или месяце	to_char(sysdate, 'WW') to_char(sysdate, 'W')	45 1
DDD, DD или D	Порядковый номер дня в году, месяце или неделе.	to_char(sysdate, 'DDD') to_char(sysdate, 'DD') to_char(sysdate, 'D')	312 07 3
DAY	Название дня, дополненное пробелами до общей длины в 9 (для английского языка) или 11 (для русского языка) символов.	to_char(sysdate, 'Day')	Monday···
DY	Трёхбуквенное сокращение названия дня	to_char(sysdate, 'Dy')	Wed
J	День по Юлианскому календарю = целое число дней с 1 января 4712 года до н.э.	to_char(sysdate, 'J')	2456239
HH или HH12	Порядковый номер часа в дне в интервале 1-12	to_char(sysdate, 'HH')	04

Элемент маски	Описание	Пример маски	Результат
HH24	Порядковый номер часа в дне в интервале 0-23	to_char(sysdate, 'HH24')	16
MI	Количество минут (0-59)	to_char(sysdate, 'MI')	47
SS	Количество секунд (0-59)	to_char(sysdate, 'SS')	55
SSSSS	Количество секунд, истёкших с полуночи (0-86399)	to_char(sysdate, 'SSSSS')	60608
AM или PM A.M. или P.M.	Индикатор полудня (обозначения эквивалентны, т.е. использование любого из них даст корректный результат в зависимости от времени суток)	to_char(sysdate, 'hh:mi am')	04:47 pm
TH	Отображение суффикса порядкового номера	to_char(sysdate, 'DDth')	07TH
SP	Отображение числа прописью	to_char(sysdate, 'DDsp')	SEVEN
SPTH или THSP	Отображение порядкового числа прописью	to_char(sysdate, 'DDspth')	SEVENTH
fm	Fill Mode. Удаляет лидирующие или последние пробелы.	to_char(sysdate, 'fmDay') to_char(sysdate, 'Day')	Monday
fx	Format eXact. Определяет необходимость точного соответствия символьного аргумента модели формата даты функции TO_DATE: • знаки пунктуации и заключённый в кавычки текст символьного аргумента должны точно (за	to_date('4.5.1999', 'fxdd.mm.yyyy')	ORA-01862: числовое значение не соответствует длине позиции формата

Элемент маски	Описание	Пример маски	Результат
	исключением регистра) совпадать с моделью формата; • символьный аргумент не может содержать лишних пробелов; • числовые элементы символьного аргумента должны содержать ровно то же количество цифр, что и соответствующий элемент модели формата. В случае отсутствия точного соответствия возвращает ошибку.		

10. УСЛОВНЫЕ ВЫРАЖЕНИЯ[1]

Синтаксис и описание условных выражений приводятся в следующей таблице:

Условное выражение	Описание
Синтаксис простого выражения CASE expr WHEN comparison_expr1 THEN return_expr1 [WHEN comparison_expr2 THEN return_expr2	Простое выражение: выполняется поиск первой пары WHEN … THEN, для которой expr равно comparision_expr, и возвращается return_expr. Если ни одна из пар не отвечает условиям и существует выражение ELSE, будет возвращено else_expr, иначе будет возвращено NULL. Выражение поиска: выполняется поиск первой пары WHEN … THEN, для которой верно условие condition, и возвращается return_expr.

[1] позволяют реализовать логику IF-THEN-ELSE

Условное выражение	Описание
… WHEN comparison_exprn THEN return_exprn ELSE else_expr] END Синтаксис выражения поиска CASE WHEN condition1 THEN return_expr1 [WHEN condition2 THEN return_expr2 … WHEN conditionn THEN return_exprn ELSE else_expr] END	Если ни одно из условий не является верным, будет возвращено else_expr, иначе будет возвращено NULL. В разделах WHEN, THEN и ELSE могут использоваться подзапросы.
DECODE(column | expression, search1, result1 [, search2, result2 , …] [, default])	Анализирует expression, сравнивая его с каждым значением search и возвращая result при совпадении. Если значение expression не соответствует ни одному search и стандартное значение (default) опущено, то будет возвращён NULL. В качестве значений search, result и default могут использоваться подзапросы. В качестве значения search можно использовать NULL.

11. АГРЕГАТНЫЕ ФУНКЦИИ

Синтаксис и описание агрегатных функций приводятся в следующей таблице:

Функция	Описание
AVG([DISTINCT | ALL] expr)	Среднее значение expr без учёта NULL-значений. По умолчанию рассматриваются все (ALL) не-NULL значения expr. Если указано ключевое слово DISTINCT, рассматриваются только уникальные не-NULL значения.
COUNT({ * | [DISTINCT | ALL] expr })	Количество строк expr. По умолчанию учитываются все (ALL) строки, не содержащие NULL-значения. Если указано ключевое слово DISTINCT, учитываются только строки, содержащие уникальные не-NULL значения. Если указана звёздочка (*), считаются все строки таблицы, включая те, что содержат дубликаты и NULL-значения. Функция COUNT никогда не возвращает NULL.
MAX([DISTINCT | ALL] expr)	Максимальное значение expr, без учёта NULL-значений. По умолчанию рассматриваются все (ALL) не-NULL значения expr. Если указано ключевое слово DISTINCT, рассматриваются только уникальные не-NULL значения.
MIN([DISTINCT | ALL] expr)	Минимальное значение expr, без учёта NULL-значений. По умолчанию рассматриваются все (ALL) не-NULL

Функция	Описание
	значения expr. Если указано ключевое слово DISTINCT, рассматриваются только уникальные не-NULL значения.
STDDEV([DISTINCT | ALL] expr)	Среднеквадратичное отклонение expr без учёта NULL-значений. По умолчанию рассматриваются все (ALL) не-NULL значения expr. Если указано ключевое слово DISTINCT, рассматриваются только уникальные не-NULL значения.
VARIANCE([DISTINCT | ALL] expr)	Дисперсия expr без учёта NULL-значений (если количество строк expr равно 1, функция вернёт 0). По умолчанию рассматриваются все (ALL) не-NULL значения expr. Если указано ключевое слово DISTINCT, рассматриваются только уникальные не-NULL значения.
SUM([DISTINCT | ALL] expr)	Сумма expr без учёта NULL-значений. По умолчанию рассматриваются все (ALL) не-NULL значения expr. Если указано ключевое слово DISTINCT, рассматриваются только уникальные не-NULL значения.

12. ТИПЫ СОЕДИНЕНИЙ

Синтаксис и описание различных типов соединения таблиц приводятся в следующей таблице:

Тип соединения	Описание
Внутренние соединения	Возвращаются только строки, удовлетворяющие условию соединения

Тип соединения	Описание
table1 NATURAL JOIN table2	Соединение по равенству значений всех столбцов с одинаковыми именами в соединяемых таблицах.
table1 [INNER] JOIN table2 USING (table.column1 [, table.column2, …])	Соединение по равенству значений тех столбцов соединяемых таблиц, которые указаны после ключевого слова USING.
[INNER] JOIN… ON	Соединение по равенству или неравенству столбцов с любыми именами в соответствии с условием, указанным после ключевого слова ON
Внешние соединения	Из одной таблицы возвращаются только строки, удовлетворяющие условию соединения, а из другой — все (с учётом ограничения WHERE), вне зависимости от того, удовлетворяют они условию соединения или нет.
RIGHT [OUTER] JOIN	Возвращаются все строки из правой таблицы
LEFT [OUTER] JOIN	Возвращаются все строки из левой таблицы
FULL [OUTER] JOIN	Возвращаются все строки обеих таблиц
Перекрёстное соединение	
CROSS JOIN	Декартово (прямое, картезианское, перекрёстное) произведение соединяемых таблиц — будут возвращены все возможные комбинации строк (количество таких комбинаций равно произведению количеств строк соединяемых таблиц). Используется по умолчанию, если условие соединения опущено.

13. ОПЕРАТОРЫ РАБОТЫ С МНОЖЕСТВАМИ ЗАПИСЕЙ (SET-ОПЕРАТОРЫ)

SET-операторы используются для объединения результатов нескольких команд SELECT в один набор. Запросы, содержащие такие операторы называют составными.

Все операторы работы с множествами записей имеют одинаковый приоритет и выполняются сервером Oracle в порядке появления в составном запросе. Порядок обработки также можно задать явно — с помощью скобок.

Условия использования SET-операторов:

- число выражений во всех разделах SELECT составного запроса должно быть одинаковым (совпадение имён выражений не требуется)
- типы данных выражений второго и последующих разделов SELECT составного запроса должны соответствовать типам данных соответствующих выражений первого раздела SELECT
- раздел ORDER BY можно использовать только в самом конце составного запроса
- если используется оператор INTERSECT, необходимо явно задавать порядок обработки запросов.

Виды и описание операторов работы с множествами записей приводятся в следующей таблице:

Оператор	Описание
UNION ALL	Объединение нескольких наборов строк. Результирующий набор может содержать дубликаты.
UNION	Объединение нескольких наборов строк без учёта дубликатов. По умолчанию строки результирующего набора сортируются по возрастанию значений столбцов в разделе SELECT.

Оператор	Описание
MINUS	Все строки первого набора, не совпадающие со строками из второго набора. Результирующий набор строк не содержит дубликатов.
INTERSECT	Строки, содержащиеся и в первом, и во втором наборе. Результирующий набор строк не содержит дубликатов.

14. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ ГРУППИРОВКИ

Добавление в раздел GROUP BY ключевых слов CUBE, ROLLUP или GROUPING SETS позволяет получать детальные отчёты по агрегируемым данным.

- ROLLUP группирует строки выборки по первым n, n-1, n-2, … 0 столбцам, указанным в разделе GROUP BY и вычисляет значения агрегатных функций для каждой из полученных групп, включая общий итог для всей таблицы.
- ROLLUP создаёт группы на основе всех возможных сочетаний столбцов, указанных в разделе GROUP BY, и вычисляет значения агрегатных функций для каждой из полученных групп, а также общий итог для всей таблицы. Общее количество возможных сочетаний, таким образом, составляет 2^n;
- GROUPING SETS позволяет явным образом указать столбцы для создания групп — группировка будет производиться по каждому столбцу, указанному в разделе GROUP BY.

В случае применения любого из вышеуказанных ключевых слов (ROLLUP, CUBE, GROUPING SETS) с помощью скобок можно указать, что несколько столбцов должны быть обработаны как один. Например:

- ROLLUP (a, b, c) эквивалентно GROUPING SETS ((a, b, c), (a, b), (c), ());

• CUBE (a, b, c) эквивалентно GROUPING SETS ((a, b, c), (a, b), (a, c), (b, c), (a), (b), (c), ()).

Пример использования ключевого слова ROLLUP для получения числа служащих в каждом отделе наряду с промежуточным выводом числа служащих по каждой должности в отделе и общего числа служащих в компании:

```
SELECT  department_id, job_id, count(*) cnt
FROM employees
GROUP  BY ROLLUP  (department_id, job_id);
```

Результат:

DEPARTMENT_ID	JOB_ID	CNT
(null)	SA_REP	1
(null)	(null)	**1**
10	AD_ASST	1
10	(null)	**1**
20	MK_MAN	1
20	MK_REP	1
20	(null)	**2**
…	…	…
110	AC_MGR	1
110	AC_ACCOUNT	5
110	(null)	**6**
(null)	(null)	107

В столбце CNT обычным шрифтом показаны промежуточные итоги, полученные в результате группировки по столбцам department_id и job_id, жирным — промежуточные итоги, полученные в результате группировки по столбцу department_id и подчёркнут общий итог — значение функции count(*) для всей таблицы employees.

Из примера видно, что если какой-либо из столбцов раздела GROUP BY не участвовал в группировке, то в

рассматриваемой строке результирующей выборки в нём появляется NULL-значение. Для того, чтобы отличать NULL-значения, образовавшиеся таким образом (например, в первых двух столбцах последней строки) от изначально присутствовавших в таблице (например, в первом столбце первых двух строк, где выведены промежуточные итоги для работника, не приписанного ни к одному отделу) используют функцию GROUPING(expr), которая возвращает 0, если по выражению (столбцу) expr производилась группировка, т.е. NULL-значение присутствовало в таблице изначально и 1 — если группировка не производилась, а NULL-значение было добавлено в процессе вычисления значений агрегатных функций.

Например, запрос:

```
SELECT department_id, job_id, count(*) cnt,
grouping(department_id) gr_dept, grouping(job_id) gr_job
FROM employees
GROUP BY ROLLUP (department_id, job_id);
```

даст следующий результат:

DEPARTMENT_ID	JOB_ID	CNT	GR_DEPT	GR_JOB
(null)	SA_REP	1	0	0
(null)	(null)	**1**	0	1
10	AD_ASST	1	0	0
10	(null)	**1**	0	1
20	MK_MAN	1	0	0
20	MK_REP	1	0	0
20	(null)	**2**	0	1
…	…	…		
110	AC_MGR	1	0	0
110	AC_ACCOUNT	5	0	0
110	(null)	**6**	0	1
(null)	(null)	107	1	1

Подробнее — см. документацию ORACLE [14, раздел 21: SQL for Aggregation in Data Warehouses].

15. ВНЕШНИЕ ТАБЛИЦЫ

Внешняя таблица представляет собой доступную только для чтения таблицу, метаданные которой хранятся в базе данных, а данные — за её пределами. Определение внешней таблицы может использоваться для осуществления любых SQL-запросов к внешним данным без необходимости их предварительной загрузки в базу.

Для создания внешних таблиц и доступа к ним необходимо создать в схеме объект DIRECTORY[1], который связать с каталогом файловой системы сервера Oracle, содержащим данные для внешней таблицы, а также выдать права на чтение данных из этого каталога. Например:

```
CREATE OR REPLACE DIRECTORY ext_dir AS 'D:\\external';
GRANT READ ON DIRECTORY ext_dir TO user;
```

Если необходимо создать журналы с дополнительной информацией (log-файл и bad-файл) или структуру внешней таблицы предполагается использовать для выгрузки данных из базы, то потребуются также права на запись данных в каталог:

```
GRANT WRITE ON DIRECTORY ext_dir TO user;
```

Конструкция для создания внешней таблицы имеет следующий синтаксис:

```
CREATE TABLE table_name
(col_name datatype[, …])
ORGANIZATION EXTERNAL
([TYPE access_driver_type]
```

[1] Для создания объектов DIRECTORY необходимо иметь системные привилегии CREATE ANY DIRECTORY.

```
DEFAULT DIRECTORY directory_name
ACCESS PARAMETRES
(…)
LOCATION ('location_specifier')
)
[REJECT LIMIT {integer | UNLIMITED}];
```

Для непосредственной загрузки/выгрузки данных используются драйвера доступа[1] двух основных типов: ORACLE_LOADER и ORACLE_DATAPUMP. Первый используется по умолчанию, а второй — когда необходимо осуществить выгрузку данных из базы данных в файл.

В разделе DEFAULT DIRECTORY можно указать один или несколько объектов DIRECTORY, которые соответствуют каталогам в файловой системе, где могут находиться внешние источники данных.

В необязательном разделе ACESS PARAMETRES указываются значения параметров для интерпретации внешних данных (набор параметров зависит от используемого драйвера доступа).

Например, при использовании драйвера ORACLE_LOADER можно указать следующие параметры:

- RECORDS DELIMITED BY NEWLINE — записи разделяются символом новой строки;
- FIELDS TERMINATED BY ',' LRTRIM — все поля заканчиваются запятой (для последнего поля записи это условие не является обязательным), пробелы в начале и конце текста обрезаются.

Подробное описание этих и других параметров дано в документации Oracle [13, часть III External Tables].

[1] Драйвер доступа — это API-интерфейс, который интерпретирует внешние данные для базы данных.

В разделе LOCATION указывают один или более внешних источников данных, которые, как правило, являются файлами.

Раздел REJECT LIMIT позволяет указать максимальное количество ошибок, которые могут быть допущены при попытке интерпретации внешних данных, прежде чем будет возвращена ошибка Oracle и запрос будет отменён. По умолчанию допустимое количество ошибок равно 0.

Некоторые ограничения:

- для внешней таблицы нельзя определить ограничения или индексы;
- выполнение DML-команд (UPDATE, INSERT, или DELETE) невозможно;
- внешняя таблица не может содержать столбцы объектного типа, а также типа varray или long. Но можно поместить массивы или данные типа long в столбцы внешней таблицы, тип данных которых определён как LOB.

16. ИЕРАРХИЧЕСКИЕ ЗАПРОСЫ

Для обработки иерархически связанных данных используют конструкцию, синтаксис которой показан ниже:

{ CONNECT BY [NOCYCLE] condition [AND condition]... [START WITH condition]

| START WITH condition CONNECT BY [NOCYCLE] condition [AND condition]...

}

Выборка формируется следующим образом:

1) определяется строка (строки) являющаяся корнем дерева иерархии — в соответствии с условием, заданным с помощью ключевых слов START WITH. По умолчанию корневыми считаются все строки рассматриваемой таблицы.

Корневая строка (строки) будет являться строкой первого уровня иерархии;

2) для каждой корневой строки определяются дочерние строки — в соответствии с условием, заданным в разделе CONNECT BY. Все дочерние строки, выбранные на данном этапе, будут являться строками второго уровня иерархии;

3) каждая из полученных строк второго уровня теперь рассматривается как корневая и для неё определяются свои дочерние строки — в соответствии с условием, заданным в разделе CONNECT BY. Дочерние строки, выбранные на данном этапе, будут являться строками третьего уровня иерархии;

и т. д.

Обязательное условие в разделе CONNECT BY, определяющее отношение между родительскими и дочерними строками в иерархии, задаётся с помощью ключевого слова PRIOR, относящегося к родительской строке.

Например, CONNECT BY PRIOR employee_id = manager_id означает, что дочерними строками (строками следующего уровня иерархии) будут все те, для которых значение manager_id совпадёт со значением employee_id рассматриваемой родительской строки (строки текущего уровня иерархии).

Вложенные запросы в разделе CONNECT BY запрещены.

CONNECT BY может также содержать дополнительные условия-фильтры — для удаления определённых ветвей иерархического дерева (родителя со всеми его потомками).

Например, запрос[1]

[1] Псевдостолбец level, используемый в запросе, определяет уровень или ранг рассматриваемой строки в иерархии.

```
SELECT employee_id, LPAD(last_name,
LENGTH(last_name)+level-1, '_') last_name
FROM employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id;
```

Даст следующий результат:

EMPLOYEE_ID	LAST_NAME
100	King
101	_Kochhar
…	…
204	__Baer
205	__Higgins
206	____Gietz
…	…

Для удаления сотрудника Higgins и всех его подчинённых необходимо использовать видоизменить рассматриваемый запрос следующим образом:

```
SELECT employee_id, LPAD(last_name,
LENGTH(last_name)+level-1, '_') last_name
FROM employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
AND employee_id!= 205;
```

Результат:

EMPLOYEE_ID	LAST_NAME
100	King
101	_Kochhar
…	…
204	__Baer
…	…

Если запрос содержит раздел WHERE, то после построения дерева иерархии исключаются все строки таблицы,

которые не удовлетворяют условию в этом разделе, но их дочерние строки выводятся.

Например, для удаления из выборки только сотрудника Higgins, запрос, рассматриваемый выше необходимо переписать так:

```
SELECT employee_id, LPAD(last_name, LENGTH(last_name)+level-1, '_') last_name
FROM employees
WHERE employee_id!= 205
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id;
```

Результат:

EMPLOYEE_ID	LAST_NAME
100	King
101	_Kochhar
…	…
204	__Baer
206	**____Gietz**
…	…

При попытке отсортировать данные выборки по какому-либо столбцу, полученная иерархия нарушится, т.к. уровень конкретной строки не будет учитываться при сортировке. Для того чтобы отсортировать строки с учётом уровня — без нарушения иерархии — используют ключевое слово SIBLINGS. Например, ORDER SIBLINGS BY last_name — для сортировки выборки по фамилии сотрудника в пределах каждого уровня иерархии;

Если условие в CONNECT BY приводит к петле (зацикливанию), Oracle возвращает ошибку. Для того, чтобы этого избежать, необходимо использовать ключевое слово NOCYCLE в разделе CONNECT BY. Например, CONNECT BY

NOCYCLE PRIOR employee_id = manager_id. В этом случае для строки, являющейся причиной зацикливания, построение иерархии будет закончено, т.е. не будет производиться дальнейший поиск её дочерних строк.

В иерархических запросах используют следующие псевдостолбцы:

- connect_by_iscycle — для отметки строк, являющихся причиной образования циклов.
- connect_by_isleaf — для отметки строк, не имеющих потомков;
- level — для отображения уровня или ранга текущей строки в иерархии;

Также используется оператор connect_by_root(column) — для отображения для каждой строки выборки значения столбца column корневой строки иерархии.

17. РЕКУРСИВНЫЕ ЗАПРОСЫ НА ОСНОВЕ РАЗДЕЛА WITH

В Oracle 11g для обращения к иерархически связанным данным используются также рекурсивные запросы, реализованные на основе раздела WITH.

Так как каждому запросу из раздела WITH присваивается собственное имя, оно может быть использовано для обращения запроса к самому себе. После имени рекурсивного запроса в скобках указывают псевдонимы столбцов, количество которых должно совпадать с количеством столбцов в опорном и рекурсивном блоках запроса (см. далее).

Для организации рекурсии запрос в разделе WITH должен содержать два блока:

1) опорный (anchor member) — раздел SELECT, использующийся для формирования исходного множества строк. Может включать операторы работы со множествами (UNION, UNION ALL, MINUS, INTERSECT).

2) рекурсивный (recursive member) — раздел SELECT, использующийся для формирования добавочного множества строк на основе исходного. Полученное добавочное множество рассматривается как новое исходное множество строк и используется для получения нового добавочного множества и т. д.

Рекурсивный блок запроса НЕ может содержать:

- ключевое слово DISTINCT;
- раздел GROUP BY;
- раздел MODEL;
- агрегатные функции (но аналитические функции использовать можно);
- подзапросы, обращающиеся к имени рекурсивного запроса;
- внешние соединения, содержащие имя рекурсивного запроса в правой части.

Объединение исходного и добавочного множеств происходит с помощью оператора UNION ALL.

Для ограничения количества выводимых строк (количества повторений рекурсивного запроса) используется раздел WHERE.

Для определения порядка строк выборки, полученной в результате накопления строк исходного и всех добавочных множеств, используют раздел SEARCH (указывается за скобками, в которые заключены блоки рекурсивного запроса), синтаксис которого показан ниже:

```
{ SEARCH
{ DEPTH FIRST BY c_alias [, c_alias]...
[ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
| BREADTH FIRST BY c_alias [, c_alias]...
[ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
}
SET ordering_column
}
```

- ключевое слово BREADTH FIRST BY (обход дерева иерархии в ширину) используется, когда необходимо поместить все строки одного уровня перед строками, являющимися их потомками (сортировка по номеру уровня);
- ключевое слово DEPTH FIRST BY (обход дерева в глубину) используется, когда необходимо поместить все строки-потомки конкретной строки перед выводом строк одного с ней уровня (собственно, представление дерева иерархии);
- упорядочивание строк происходит по столбцу, указанному после ключевого слова BY;
- псевдонимы столбцов, указанные в разделе SEARCH должны входить в список псевдонимов столбцов рекурсивного запроса;
- в конце раздела SEARCH после обязательного ключевого слова SET указывается дополнительное (вымышленное) имя столбца, который будет включен в число столбцов результата рекурсивного запроса и в котором автоматически будут проставлены числовые значения, соответствующие порядку строк выборки.

Для отслеживания и обработки циклов используют раздел CYCLE (указывается за скобками, в которые заключены блоки рекурсивного запроса), синтаксис которого показан ниже:

```
{CYCLE  c_alias [, c_alias]...
SET cycle_mark_c_alias  TO cycle_value
DEFAULT  no_cycle_value
}
```

- псевдонимы столбцов c_alias используются для определения цикла. Они должны входить в список псевдонимов столбцов рекурсивного запроса.
- cycle_value и no_cycle_value должны быть единичными символами;
- при обнаружении цикла столбцу cycle_mark_c_alias строки, являющейся причиной зацикливания, присваивается значение cycle_value. Построение рекурсии для этой строки заканчивается, т.е. дальнейшее формирование добавочного множества строк на её основе не производится.
- если не было найдено ни одного цикла, то столбцу cycle_mark_c_alias присваивается значение по умолчанию (no_cycle_value).
- столбец cycle_mark_c_alias изначально не входит в список псевдонимов столбцов рекурсивного запроса — он добавляется автоматически.

Пример использования рекурсивного запроса для расчёта последовательности Фибоначчи:

```
WITH
numbers (n, prev) AS (
SELECT  0 n, 1 prev from dual
UNION  ALL
SELECT  n + prev, n from numbers
WHERE  n + prev < 50
```

```
)
SELECT n from numbers;
```

Результат:

N
0
1
1
2
3
5
8
13
21
34

Здесь:

- имя рекурсивного запроса — numbers
- n и prev — псевдонимы столбцов выборки
- select 0 n, 1 prev from dual — опорный блок запроса
- select n + prev, n from numbers where n + prev < 50 — рекурсивный блок запроса.

Пример рекурсивного запроса с использованием раздела SEARCH — для отображения дерева иерархии сотрудников:

```
WITH
hierarchy (emp, lname, lvl) AS (
SELECT employee_id emp, last_name lname, 1 lvl from employees
WHERE manager_id IS NULL
UNION ALL
SELECT employee_id, last_name, lvl + 1
FROM employees e JOIN hierarchy
ON (emp = manager_id)
)
SEARCH DEPTH FIRST BY emp SET ord
```

SELECT emp, LPAD (lname, LENGTH (lname) + lvl * 2 - 2, '_') name, lvl from hierarchy;

Результат:

EMP	NAME	LVL
100	King	1
101	__Kochhar	2
108	____Greenberg	3
109	______Faviet	4
110	______Chen	4
111	______Sciarra	4
112	______Urman	4
113	______Popp	4
200	____Whalen	3
203	____Mavris	3
…	…	…

18. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

Регулярные выражения определяют маски, которые состоят из метасимволов (являющихся операторами или их частями) и символьных литералов.

Синтаксис и описание метасимволов и операторов, использующихся в регулярных выражениях, приводятся в следующей таблице:

Оператор / Метасимвол	Описание	Пример
\d	Цифровой символ. Эквивалент POSIX-выражения [[:digit:]].	^\(\d{3}\) \d{3}-\d{4}$ соответствует: (650) 555-0100, не соответствует: 650-555-0100
\D	Нецифровой символ. Эквивалент POSIX-	\w\d\D соответствуют: b2b и b2_

Оператор / Метасимвол	Описание	Пример
	выражения [^[:digit:]].	не соответствует: b22
\w	Словесный символ (буква, цифра или символ подчёркивания). Эквивалент POSIX-выражения [[:alnum:]_].	\w+@\w+(\.\w+)+ соответствует: jdoe@company.co.uk не соответствует: jdoe@company
\W	Несловесный символ. Эквивалент POSIX-выражения [^[:alnum:]_].	\w+\W\s\w+ соответствует: to: bill не соответствует: to bill
\s	Пробел. Эквивалент POSIX-выражения [[:space:]].	\(\w\s\w\s\) соответствует: (a·b·) не соответствуют: (ab) или (a,b.)
\S	Символ, не являющийся пробелом. Эквивалент POSIX-выражения [^[:space:]].	\(\w\S\w\S\) соответствуют: (abde) и (a,b.) не соответствует: (a·b·d·e)
\A	Указатель на начало текста в однострочном или многострочном формате. Не является эквивалентом POSIX-оператора ^!	\AL в строке Line1\n[1]Line2\n приведённому выражению соответствует только первая L.
\Z	Указатель на конец	\s\Z

[1] \n — символ новой строки

Оператор / Метасимвол	Описание	Пример
	текста или символ новой строки (в однострочном или многострочном формате). Не является эквивалентом POSIX-оператора $!	в строке L·i·n·e·\n приведённому выражению соответствует последний символ пробела.
\z	Указатель на конец строки в однострочном или многострочном формате (т.е. символ новой строки включается в рассмотрение как часть строки). Не является эквивалентом POSIX-оператора $!	\s\z в строке L·i·n·e·\n приведённому выражению соответствует символ новой строки.
^	По умолчанию соответствует началу строки. В многострочном формате соответствует началу каждой строки в тексте.	^def подстрока def соответствует приведённому выражению в строке defghi, но не в строке abcdef.
$	По умолчанию соответствует концу строки. В многострочном формате соответствует	def$ подстрока def соответствует приведённому выражению в строке abcdef, но не в строке defghi.

Оператор / Метасимвол	Описание	Пример
	концу каждой строки в тексте.	
+?	Одно или более вхождений предшествующего подвыражения («ленивая»[1] квантификация).	\w+?x\w в строке abxcxd приведённому выражению соответствует abxc («жадная» квантификация: выражению \w+x\w соответствует abxcxd).
*?	Ноль или более вхождений предшествующего подвыражению («ленивая» квантификация). Всегда соответствует пустой строке.	\w*?x\w в строке xaxbxc приведённому выражению соответствует xa («жадная» квантификация: выражению \w*x\w соответствует xaxbxc).
??	Не более одного (ноль или одно) вхождения предшествующего подвыражения («ленивая» квантификация). Всегда соответствует	a??aa в строке aaaa приведённому выражению соответствует aa («жадная» квантификация: выражению a?aa соответствует aaa).

[1] При «ленивой» (nongreedy) квантификации всегда ищется максимально короткое из возможных подвыражений (например, указание «один или более символов» будет трактоваться как «один символ»), при «жадной» (greedy) — максимально длинное. Для превращения «ленивого» оператора в «жадный» удалите модификатор (?).

Оператор / Метасимвол	Описание	Пример
	пустой строке.	
{m}?	Ровно m вхождений предшествующего подвыражения («ленивая» квантификация).	(a|aa){2}? в строке aaaaa приведённому выражению соответствует aa («жадная» квантификация: выражению (a|aa){2} соответствует aaaa) b{2}? или b{2} в строке bbbb обоим выражениям соответствует bb.
{m,}?	По крайней мере m вхождений предшествующего подвыражения («ленивая» квантификация).	a{2,}? в строке aaaaa приведённому выражению соответствует aa («жадная» квантификация: выражению a{2,} соответствует вся строка aaaaa).
{m,n}?	По крайней мере, m, но не больше чем n вхождений предшествующего подвыражения («ленивая» квантификация). Выражение {0,n}? всегда соответствует пустой строке.	a{2,4}? в строке aaaaa приведённому выражению соответствует aa («жадная» квантификация: выражению a{2,4} соответствует aaaa).

Оператор / Метасимвол	Описание	Пример
|	Разделяет альтернативные варианты.	a|b соответствует: a или b.
( )	Подвыражение, заключённое в скобки рассматривается как одна логическая единица. Такое подвыражение может быть просто символьной строкой или содержать операторы. На подвыражение можно ссылаться с помощью обратной ссылки \n.	(abc)?def соответствуют: abcdef или def не соответствуют: abcdefg или xdef.
\n	Обратная ссылка — соответствует n-ному предшествующему подвыражению. N может принимать значения от 1 до 9. Подсчёт производится слева направо, начиная с открывающей скобки каждого предшествующего подвыражения.	(abc|def)xy\1 соответствуют: abcxyabc или defxydef не соответствуют: abcxydef или abcxy. ^(.*)\1$ соответствует строка, состоящая из двух одинаковых подстрок.
\	Экранирование. Символ или метасимвол,	abc\+def соответствует: abc+def не соответствуют: abcdef

Оператор / Метасимвол	Описание	Пример
	следующий за данным, рассматривается как литерал.	или abccdef.

Синтаксис и описание предопределённых символьных классов POSIX приводятся в таблице ниже:

Класс символов	Описание
.	Любой символ
[:alpha:]	Буквы
[:lower:]	Буквы в нижнем регистре
[:upper:]	Буквы в верхнем регистре
[:digit:]	Цифры
[:alnum:]	Буквы и цифры
[:space:]	Пробелы (непечатаемые символы), такие как перевод каретки, новая строка, вертикальная табуляция и подача страницы
[:punct:]	Знаки препинания
[:cntrl:]	Управляющие символы (непечатаемые)
[:print:]	Печатаемые символы

Синтаксис и описание условий и функций для работы с регулярными выражениями приводятся в следующей таблице:

Функция	Описание	Пример
REGEXP_LIKE(source_char, pattem [, match_param])	Возвращает подстроки, удовлетворяющие определённому шаблону pattern в соответствии с параметром match_param. Может использоваться в разделе WHERE запроса. Параметр match_param для этой и всех последующих	WHERE REGEXP_LIKE ((employees.first_name, '^Ste(v|ph)en$') Будут выбраны все сотрудники с именем Steven или Stephen.

Функция	Описание	Пример
	функций таблицы может принимать следующие значения: i — нечувствительность к регистру; c — чувствительность к регистру (по умолчанию определяется значением параметра NLS_SORT); n — при использовании шаблона сопоставления с любым символом (.) символ новой строки \n включается в рассмотрение (по умолчанию символ новой строки не рассматривается); m — исходная строка рассматривается как многострочный текст, в котором концом строки считается символ новой строки \n. x — игнорирование пробелов в маске (шаблоне).	
REGEXP_COUNT source_char, pattern [, position [, match_param]])	Возвращает количество вхождений подстроки, соответствующей рассматриваемому шаблону pattern, в строку source_char, начиная с позиции position(по	REGEXP_COUNT ('Albert Einstein', 'e', 7, 'c') Результатом будет 1 — количество букв е (но не E) в строке 'Albert Einstein',

Функция	Описание	Пример
	умолчанию — 1), в соответствии с параметром match_param.	начиная с 7 символа.
REGEXP_INSTR (source_char, pattern [, position [, occurrence [, return_opt [, match_param [, subexpr]]]]]	Возвращает позицию первого символа подстроки, соответствующей данному шаблону pattern, или первого за ней символа (в соответствии со значением параметра return_opt) в рассматриваемой строке source_char, начиная с позиции position (по умолчанию — 1) и в соответствии с параметром match_param. Рассматривается n-ное вхождение подстроки, где n задаётся параметром occurrence (по умолчанию — 1). Параметр return_opt определяет будет ли возвращена позиция первого символа подстроки, соответствующей рассматриваемому шаблону (return_opt = 0, используется по умолчанию) или позиция следующего за ней символа (return_opt = 1).	REGEXP_INSTR (employees.email, '\w+@\w+(\.\w+)+') Будет возвращена начальная позиция первого корректного адреса электронной почты в столбце email таблицы employees. Если результат больше 0, значит столбец содержит корректный адрес.

Функция	Описание	Пример
	Параметр subexpr (от 0 до 9) определяет, какое из подвыражений шаблона используется для поиска и сопоставления — в случае если шаблон состоит из нескольких подвыражений, заключённых в скобки. Значение этого параметра по умолчанию — 0 (т.е. рассматривается шаблон целиком).	
REGEXP_REPLACE(source_char, pattern [, replace_string [, position [, occurrence [, match_param]]]])	Возвращает строку source_char, в которой n-ное вхождение подстроки, соответствующей шаблону pattern, заменено на строку replace_string. N определяется значением параметра occurrence (по умолчанию — 0, т.е. учитываются все вхождения). Поиск начинается с позиции position (по умолчанию — 1).	REGEXP_REPLACE (countries.country_name, '(.)', '\1 ') После каждого символа в столбце country_name таблицы countries будет добавлен пробел.
REGEXP_SUBSTR(source_char, pattern [, position [, occurrence [, match_param [, subexpr]]]])	Функция похожа на REGEXP_INSTR, от которой отличается только тем, что вместо начальной позиции подстроки, соответствующей данному	REGEXP_SUBSTR('Oracle·2010', 'O·r·a·c·l·e', 1, 1, 'x') Результатом будет 'Oracle', т.к. при указании параметра

Функция	Описание	Пример
	шаблону pattern в рассматриваемой строке source_char возвращает саму подстроку.	x пробелы в шаблоне игнорируются.

19. АНАЛИТИЧЕСКИЕ ФУНКЦИИ

Синтаксис и описание основных аналитических функций приводятся в следующей таблице:

Функция	Описание
AVG([DISTINCT | ALL] expr) [OVER(analytic_clause)]	Используется для вычисления среднего значения выражения expr в пределах группы и окна. Для поиска среднего после удаления дублирующихся значений можно указывать ключевое слово DISTINCT.
CORR(expr1, expr2) [OVER (analytic_clause)]	Выдает коэффициент корреляции для пары выражений expr, возвращающих числовые значения. Это сокращение для выражения: COVAR_POP(expr1, expr2) / STDDEV_POP(expr1) * STDDEV_POP(expr2). В статистическом смысле, корреляция — это степень связи между переменными. Связь между переменными означает, что значение одной переменной можно в определенной степени предсказать по значению другой. Коэффициент корреляции представляет степень корреляции в виде числа в

Функция	Описание
	диапазоне от -1 (высокая обратная корреляция) до 1 (высокая корреляция). Значение 0 соответствует отсутствию корреляции.
COUNT({ * | [DISTINCT | ALL] expr }) [OVER (analytic_clause)]	Эта функция считает строки в группах. Если указать * или любую константу, кроме NULL, функция count будет считать все строки. Если указать выражение expr, функция count будет считать строки, для которых оно имеет не NULL-значение. Можно задавать модификатор DISTINCT, чтобы считать строки в группах после удаления дубликатов.
COVAR_POP(expr1, expr2) [OVER (analytic_clause)]	Возвращает ковариацию генеральной совокупности (population covariance) пары выражений expr с числовыми значениями.
COVAR_SAMP(expr1, expr2) [OVER (analytic_clause)]	Возвращает выборочную ковариацию (sample covariance) пары выражений expr с числовыми значениями.
CUME_DIST() OVER ([query_partition_clause] order_by_clause)	Вычисляет относительную позицию строки в группе. Функция CUME_DIST всегда возвращает число большее 0 и меньше или равное 1. Это число представляет "позицию" строки в группе из N арок. В группе из трех строк,

Функция	Описание
	например, возвращаются следующие значения кумулятивного распределения: 1/3, 2/3 и 3/3.
DENSE_RANK( ) OVER([query_partition_clause] order_by_clause)	Эта функция вычисляет относительный ранг каждой возвращаемой запросом строки по отношению к другим строкам, основываясь на значениях выражений в конструкции ORDER BY. Данные в группе сортируются в соответствии с конструкцией ORDER BY, а затем каждой строке поочередно присваивается числовой ранг, начиная с 1. Ранг увеличивается при каждом изменении значений выражений, входящих в конструкцию ORDER BY. Строки с одинаковыми значениями получают один и тот же ранг (при этом сравнении значения NULL считаются одинаковыми). Возвращаемый этой функцией "плотный" ранг дает ранговые значения без промежутков. Сравните с представленной далее функцией RANK.
FIRST_VALUE { (expr) [{RESPECT | IGNORE} NULLS] | (expr [{RESPECT | IGNORE} NULLS]) } OVER (analytic_clause)	Возвращает первое значение выражения expr в группе с учётом NULL-значений (respect nulls) или без (ignore nulls). По умолчанию NULL-значения учитываются.

Функция	Описание
LAG { (value_expr [, offset [, default]]) [{ RESPECT | IGNORE } NULLS] | (value_expr [{ RESPECT | IGNORE } NULLS] [, offset [, default]]) } OVER ([query_partition_clause] order_by_clause)	Функция LAG дает доступ к другим строкам результирующего множества, избавляя от необходимости выполнять самосоединения. Она позволяет работать с курсором как с массивом. Можно ссылаться на строки, предшествующие текущей строке в группе. О том, как обращаться к следующим строкам в группе, см. в описании функции LEAD. Смещение offset — это положительное целое число со стандартным значением 1 (предыдущая строка). Стандартное значение default возвращается, если индекс выходит за пределы окна (для первой строки группы будет возвращено стандартное значение).
LAST_VALUE { (expr) [{ RESPECT | IGNORE } NULLS] | (expr [{ RESPECT | IGNORE } NULLS]) OVER (analytic_clause)	Возвращает последнее значение выражения expr в группе с учётом NULL-значений (respect nulls) или без (ignore nulls). По умолчанию NULL-значения учитываются.
LEAD { (value_expr [, offset [, default]]) [{ RESPECT | IGNORE } NULLS] | (value_expr [{ RESPECT | IGNORE } NULLS] [, offset [, default]]) } OVER ([query_partition_clause]	Функция LEAD противоположна функции LAG. Если функция LAG дает доступ к предшествующим строкам группы, то функция LEAD позволяет обращаться к строкам, следующим за текущей. Смещение offset — это

Функция	Описание
order_by_clause)	положительное целое число со стандартным значением 1 (следующая строка). Стандартное значение default возвращается, если индекс выходит за пределы окна (для последней строки группы будет возвращено стандартное значение).
LISTAGG(measure_expr [, 'delimiter']) WITHIN GROUP (order_by_clause) [OVER query_partition_clause]	Объединяет значения поля measure_expr в одну строку с указанным разделителем 'delimiter' (по умолчанию — NULL). NULL-значения игнорируются. Order_by_clause задаёт порядок сортировки значений measure_expr перед конкатенацией. Результирующие данные могут быть разбиты на группы в соответствии с выражениями query_partition_clause.
MAX([DISTINCT | ALL] expr) [OVER (analytic_clause)]	Находит максимальное значение выражения expr в пределах окна в группе.
MIN([DISTINCT | ALL] expr) [OVER (analytic_clause)]	Находит минимальное значение выражения expr в пределах окна в группе.
NTH_VALUE (measure_expr, n) [FROM { FIRST | LAST }][{ RESPECT | IGNORE } NULLS] OVER (analytic_clause)	Расширяет возможности функций FIRST_VALUE and LAST_VALUE, возвращая любое, заданное порядковым номером n в группе, значение выражения measure_expr. Вычисление порядкового номера

Функция	Описание
	может производиться с первой (first) или последней (last) строки группы. NULL-значения могут учитываться (respect) или игнорироваться (ignore).
NTILE(expr) OVER ([query_partition_clause] order_by_clause)	Делит группу на фрагменты по значению выражения expr. Например, если выражение = 4, то каждой строке в группе присваивается число от 1 до 4 в соответствии с фрагментом, в которую она попадает. Если в группе 20 строк, первые 5 получают значение 1, следующие 5 значение 2 и т. д. Если количество строк в группе не делится на значение выражения без остатка, строки распределяются так, что ни в одном фрагменте количество строк не превосходит минимальное количество в других фрагментах более чем на 1, причем дополнительные строки будут в группах с меньшими номера фрагмента. Например, если снова выражение = 4, а количество строк = 21, в первом фрагменте будет 6 строк, во втором и последующих — 5.
PERCENT_RANK( ) OVER ([query_partition_clause] order_by_clause)	Аналогична функции CUME_DIST (кумулятивное распределение). Вычисляет ранг строки в группе

Функция	Описание
	минус 1, деленный на количество обрабатываемых строк минус 1. Эта функция всегда возвращает значения в диапазоне от 0 до 1 включительно.
PERCENTILE_CONT(expr) WITHIN GROUP (ORDER BY expr [DESC | ASC]) [OVER (query_partition_clause)]	Функция обратного распределения, которая предполагает использование непрерывной модели распределения. Принимает значение процентиля expr (число от 0 до 1) в качестве входного параметра и возвращает интерполированное значение, которое должно было бы давать это значение процентиля с учётом спецификации сортировки. NULL-значения игнорируются.
PERCENTILE_DISC(expr) WITHIN GROUP (ORDER BY expr [DESC | ASC]) [OVER (query_partition_clause)]	Функция обратного распределения, которая предполагает использование дискретной модели распределения. Принимает значение процентиля expr (число от 0 до 1) в качестве входного параметра и возвращает интерполированное значение, которое должно было бы давать это значение процентиля с учётом спецификации сортировки. NULL-значения игнорируются.
RANK( ) OVER ([query_partition_clause] order_by_clause)	Эта функция вычисляет относительный ранг каждой строки, возвращаемой запросом, на основе значений выражений, входящих в конструкцию ORDER BY. Данные в

Функция	Описание
	группе сортируются в соответствии с конструкцией ORDER BY, а затем каждой строке поочередно присваивается числовой ранг, начиная с 1. Строки с одинаковыми значениями выражений, входящих в конструкцию ORDER BY, получают одинаковый ранг, но если две строки получат одинаковый ранг, следующее значение ранга пропускается. Если две строки получили ранг 1, строки с рангом 2 не будет; следующая строка в группе получит ранг 3. В этом отличие от функции DENSE_RANK, которая не пропускает значений.
RATIO_TO_REPORT(expr) OVER ([query_partition_clause])	Эта функция вычисляет значение expr / (sum(expr)) по строкам группы. Это дает процент, который составляет значение текущей строки по отношению к сумме выражения expr.
REGR_xxxxxxx (expr1, expr2) [OVER (analytic_clause)]	Эти функции линейной регрессии применяют стандартную линейную регрессию по методу наименьших квадратов к паре выражений. Предлагается 9 различных функций регрессии: { REGR_SLOPE | REGR_INTERCEPT | REGR_COUNT

Функция	Описание
	| REGR_R2 | REGR_AVGX | REGR_AVGY | REGR_SXX | REGR_SYY | REGR_SXY }
ROW_NUMBER( ) OVER ([query_partition_clause] order_by_clause)	Возвращает смещение строки по отношению к началу упорядоченной группы. Может использоваться для последовательной нумерации строк, упорядоченных по определенным критериям.
STDDEV([DISTINCT | ALL] expr) [OVER (analytic_clause)]	Вычисляет стандартное (среднеквадратичное) отклонение (standard deviation) текущей строки по отношению к группе.
STDDEV_POP(expr) [OVER (analytic_clause)]	Эта функция вычисляет стандартное отклонение генеральной совокупности (population standard deviation) и возвращает квадратный корень из дисперсии генеральной совокупности (population variance). Она возвращает значение, совпадающее с квадратным корнем из результата функции VAR_POP.
STDDEV_SAMP(expr) [OVER (analytic_clause)]	Эта функция вычисляет накопленное стандартное отклонение выборки (cumulative sample standard deviation) и возвращает квадратный корень

Функция	Описание
	выборочной дисперсии (sample variance). Она возвращает значение, совпадающее с квадратным корнем из результата функции VAR_SAMP.
SUM([DISTINCT | ALL] expr) [OVER (analytic_clause)]	Вычисляет общую сумму значений выражения expr для группы.
VAR_POP(expr) [OVER (analytic_clause)]	Эта функция возвращает дисперсию генеральной совокупности для набора числовых значений (значения NULL игнорируются). Функция VAR_POP вычисляет значение: (SUM(expr* expr) — SUM(expr)*SUM(expr) / COUNT(expr)) / COUNT(expr)
VAR_SAMP(expr) [OVER (analytic_clause)]	Эта функция возвращает выборочную дисперсию для набора числовых значений (значения NULL игнорируются). Она вычисляет значение: (SUM(expr*expr) — SUM(expr)*SUM(expr) / COUNT(expr)) / (COUNT(expr) - 1)
VARIANCE([DISTINCT | ALL] expr) [OVER (analytic_clause)]	Возвращает дисперсию для выражения. Сервер Oracle вычисляет дисперсию как: - 0, если количество строк в группе =1; - VAR_SAMP, если количество строк в группе >1.
aggregate_function KEEP	Групповые функции

Функция	Описание
(DENSE_RANK FIRST ORDER BY expr [DESC | ASC] [NULLS { FIRST | LAST }] [, expr [DESC | ASC] [NULLS { FIRST | LAST }]]...) [OVER ([query_partition_clause])]	(aggregate_function), допустимые к использованию: MIN, MAX, SUM, AVG, COUNT, VARIANCE, STDDEV. Вычисления производятся над набором значений получившим ранг 1. Если заданный ранг получило только одно значение, вычисления производятся над множеством из одного элемента.
aggregate_function KEEP (DENSE_RANK LAST ORDER BY expr [DESC | ASC] [NULLS { FIRST | LAST }] [, expr [DESC | ASC] [NULLS { FIRST | LAST }]]...) [OVER ([query_partition_clause])]	Групповые функции (aggregate_function), допустимые к использованию: MIN, MAX, SUM, AVG, COUNT, VARIANCE, STDDEV Вычисления производятся над набором значений получившим максимальный ранг. Если заданный ранг получило только одно значение, вычисления производятся над множеством из одного элемента.

19. ИСПОЛЬЗОВАНИЕ РАЗДЕЛА PIVOT

Раздел PIVOT команды SELECT позволяет транспонировать таблицу, разместив результаты вычисления агрегатных функций для заданных групп не в столбец, а в строку. Поэтому, как правило, результирующая таблица содержит больше столбцов, чем строк.

Когда данный раздел ещё не был реализован, для получения схожего результата использовалась конструкция наподобие COUNT (CASE expr WHEN expr1 THEN 1).

Общий синтаксис раздела PIVOT:

```
PIVOT [ XML ]
  ( aggregate_function ( expr ) [[AS] alias ]
  [, aggregate_function ( expr ) [[AS] alias ] ]...
  FOR { column
          | ( column [, column]... )
            }
  IN ( { { { expr
              | ( expr [, expr]... )
                } [ [ AS] alias]
              }...
            | subquery
            | ANY [, ANY]...
           }
        )
  )
```

Сначала производится вычисление результатов агрегатных функций aggregate_function (expr), при этом группировка осуществляется неявно — без указания раздела GROUP BY — по всем столбцам выборки, не указанным в разделе PIVOT в совокупности со столбцами, указанными в разделе PIVOT после ключевого слова IN.

При построении результирующей выборки сначала выводятся все столбцы, не использующиеся для вычисления результатов агрегатных функций и не указанные после ключевого слова FOR, затем добавляются новые столбцы — с названиями, соответствующими значениям expr в столбце column, указанным после ключевого слова IN. Значениями этих дополнительных столбцов будут соответствующие результаты вычисления агрегатных функций.

Например, для того, чтобы узнать количество служащих, нанятых в каждом отделе в определённые годы можно выполнить такой запрос:

```
SELECT *
FROM
(SELECT department_id, extract(year from hire_date) year
FROM employees)
PIVOT
(COUNT(*) FOR year IN (2004, 2005, 2007, 2008))
ORDER BY 1;
```

Сначала в результате выполнения подзапроса в разделе FROM будет сформирована таблица со столбцами department_id и year, отображающая годами найма новых сотрудников для каждого отдела (для каждого сотрудника — своя строка со значением года). Затем будет подсчитано количество строк со значением года 2004 — для каждого департамента, а также будут получены аналогичные значения для остальных лет, указанных в списке после ключевого слова IN.

После этого будет сформирована результирующая выборка, состоящая из столбца department_id, а также дополнительных столбцов с названиями 2004, 2005, 2007 и 2008, в которых будет содержаться количество сотрудников, нанятых в соответствующем году — для каждого департамента из первого столбца:

DEPARTMENT_ID	2004	2005	2007	2008
10	0	0	0	0
20	1	1	0	0
30	0	2	1	0
…	…	…	…	…

20. ИСПОЛЬЗОВАНИЕ РАЗДЕЛА MODEL

Раздел MODEL команды SELECT позволяет представлять результаты выборки в виде многомерного массива и работать с ними как с книгой Excel, определяя правила вычислений с помощью ключевого слова RULES. С помощью данного раздела можно обрабатывать неограниченные объёмы данных.

Многомерный массив создаётся путём разбиения элементов выборки на три группы:

- PARTITION BY — определяет логические группы в пределах многомерного массива;
- DIMENSION BY — столбцы, указанные после этих ключевых слов используются для однозначного определения любой ячейки массива в пределах логической группы, т.е. необходимо использовать столбцы, гарантирующие уникальную идентификацию. В данном разделе могут быть указаны константы — для создания столбцов, отсутствующих в таблице;
- MEASURES — столбцы, указанные после данного ключевого слова, рассматриваются как ячейки многомерного массива и используются для сохранения результатов вычислений. Здесь также могут быть указаны константы или выражения для создания новых столбцов, не существующих в таблице на момент вычислений.

Продолжая сравнение MODEL с книгой Excel, можно определить условие, заданное с помощью ключевых слов PARTITION BY как критерий разбиения книги на листы, список столбцов после ключевых слов DIMENSION BY — как идентификаторы строк листа, а список столбцов после ключевого слова MEASURES — как идентификаторы столбцов листа.

Правила расчёта значений ячеек массива (столбцов из MEASURES) задаются с помощью ключевого слова RULES.

Каждое правило представляет собой выражение, в левой части которого указывается ссылка на ячейку или диапазон ячеек, а в правой — присваиваемое значение или алгоритм его вычисления. С помощью правил можно организовать циклические вычисления и рекурсию. Также набор правил может быть пустым.

Для обращения к ячейкам многомерного массива используются следующие виды ссылок:

• позиционные — каждая ячейка однозначно определяется через указание конкретных значений соответствующих столбцов из DIMENSION BY. Например, если требуется задать зарплату сотрудника из определённого отдела, то столбцы department_id и employee_id будут указаны в DIMENSION BY, а столбец salary — в MEASURES и ссылка будет выглядеть следующим образом: salary[10, 100] — зарплата сотого сотрудника из десятого отдела;

• символические — определяют диапазон ячеек. Например, salary[department_id <= 50, employee_id > 115] — зарплаты сотрудников с номерами, превышающими 115 и работающими в отделах с номерами не меньше 50.

Для определения любой ячейки используют метасимвол ANY. Например, salary [20, any] — зарплата любого (каждого) сотрудника из отдела 20.

Значение конкретной ячейки при использовании символических ссылок может быть получено с помощью функции CV, которую указывают в правой части правила для получения используемого в данный момент значения соответствующего столбца из левой части. Например, правило может выглядеть следующим образом: salary[department_id <= 50, employee_id > 115] = salary[cv(department_id), cv(employee_id)] + 1000, что означает отображение повышенной на 1000 зарплаты

для каждого сотрудника, имеющего номер больше 115 и работающего в отделе с номером не меньше 50.

Сокращённый синтаксис раздела MODEL:

```
MODEL [RETURN {ALL|UPDATED} ROWS]
    [PARTITION BY (<cols>)]
    DIMENSION BY (<cols>)
    MEASURES (<cols>)
    [IGNORE NAV | [KEEP NAV]
    [RULES] [UPDATE | UPSERT | UPSERT ALL]
    [AUTOMATIC ORDER | SEQUENTIAL ORDER]
    [ITERATE (<number>) [UNTIL <condition>]]
    (<rule>, <rule>,.., <rule>)
```

Семантика ключевых слов:

• RETURN {ALL|UPDATED} ROWS — позволяет указать, возвращать все строки или же только те, значения в которых были обновлены на основе заданных правил. По умолчанию возвращаются все строки;

• IGNORE NAV — определяет значение по умолчанию, которое будет использовано вместо NULL и пропущенных (не найденных в таблице) значений: 0 — для чисел, пустая строка — для символьных строк, 1 января 2001 года — для дат и NULL — для остальных типов данных;

• KEEP NAV — оставляет пропущенные значения и NULL без изменений. Используется по умолчанию;

• UPDATE — обновляет существующие значения ячеек. Если значение не существует, обновления не происходит;

• UPSERT — обновляет существующие значения ячеек. Если значение не существует, но в правилах задано соответствующее условие, оно будет вставлено. При этом для

вставки значений разрешается использовать только позиционные ссылки. Используется по умолчанию;

• UPSERT ALL — работает аналогично UPSERT, но также допускает вставку новых значений с использованием символических ссылок.

Опции UPDATE, UPSERT, UPSERT ALL, IGNORE NAV и KEEP NAV могут быть заданы на глобальном уровне (для всех последующих правил) — после ключевого слова RULES или на локальном — перед описанием отдельно взятого правила. Опции, указанные на локальном уровне переопределяют глобально заданное поведение.

• AUTOMATIC ORDER — правила вычисляются с учётом логической зависимости между ними, порядок вычисления может не совпадать с порядком написания;

• SEQUENTIAL ORDER — правила вычисляются в том порядке, в котором были записаны. Используется по умолчанию;

• ITERATE (<number>) и UNTIL <condition> — используются для организации циклов и могут быть использованы только при порядке вычислений заданном как SEQUENTIAL ORDER. Целое положительное число number определяет количество повторений, а необязательное условие condition — условие выхода из цикла до выполнения заданного числа повторений. В качестве счётчика используется системная переменная iteration_number с начальным значением 0. Для доступа к значению ячейки, полученному на предыдущем шаге, используется функция PREVIOUS.

Простой пример использования цикла:

```
SELECT x, s
FROM DUAL
MODEL
DIMENSION BY (1 AS x)
```

```
MEASURES (1024 AS s)
RULES ITERATE (4)
(s[1] = s[1]/2);
```

Данный запрос вернёт следующий результат:

X	S
1	64

Здесь значение в ячейке s при x=1 было четыре раза изменено и выведен конечный результат. Если необходимо видеть промежуточные результаты вычислений, то новые значения x можно задать с помощью переменной счётчика iteration_number:

```
SELECT x, s
FROM DUAL
MODEL
DIMENSION BY (1 AS x)
MEASURES (1024 AS s)
RULES ITERATE (4)
(s[iteration_number+2] = s[iteration_number+1]/2);
```

В этом случае результат будет следующий:

X	S
1	1024
2	512
3	256
4	128
5	64

Также для организации циклов может быть использована конструкция FOR. Если данное ключевое слово указывается в левой части выражения, определяющего правило, то ссылка на диапазон ячеек рассматривается как позиционная. Например, запрос

```
SELECT employee_id, salary
```

```
FROM employees
MODEL
DIMENSION BY (employee_id)
MEASURES (salary)
RULES UPSERT
(salary[FOR employee_id IN (90, 100, 110)] = 10000)
ORDER BY 1;
```

выведет всех сотрудников и их оклады, причём для сотрудников 100 и 110 установит оклад в 10000 и также добавит сотрудника 90 с окладом в 10000, т.к. задано ключевое слово UPSERT:

EMPLOYEE_ID	SALARY
90	**10000**
100	**10000**
101	17000
102	17000
…	…

Без использования ключевого слова FOR ссылка salary[employee_id IN (90, 100, 110)] рассматривается как символическая, т.к. определяет некоторый диапазон ячеек, поэтому использование опции UPSERT, не допускающей создания новых ячеек при работе с символическими ссылками в данном случае привело бы к тому же результату, как и использование опции UPDATE:

```
SELECT employee_id, salary
FROM employees
MODEL
DIMENSION BY (employee_id)
MEASURES (salary)
RULES UPSERT
(salary[employee_id IN (90, 100, 110)] = 10000)
ORDER BY 1;
```

EMPLOYEE_ID	SALARY
100	**10000**
101	17000
102	17000
103	9000
…	…

Для задания шага используют ключевые слова INCREMENT (для увеличения) DECREMENT (для уменьшения).

Синтаксис, используемый для задания диапазонов чисел и дат: FOR dimension FROM value1 TO value2 [INCREMENT | DECREMENT] value3. Здесь value1 — начальное значение, value2 — конечное значение, value3 — шаг, задаваемый с помощью положительного целого числа или значения интервального типа (для дат).

Пример использования FOR-цикла для числовых значений: sales['Bounce', FOR year FROM 2001 TO 2005 INCREMENT 1] = sales['Bounce', year=CV(year)-1] * 1.2

В случае рассмотрения диапазона строк используют другой синтаксис: FOR dimension LIKE string FROM value1 TO value2 [INCREMENT | DECREMENT] value3, где строка string может содержать только один символ %, который будет заменяться числовыми значениями из диапазона value1-value2 с шагом value3. Пример использования конструкции данного типа: sales[FOR product LIKE 'product-%' FROM 1 TO 3 INCREMENT 1, 2003] = sales[CV(product), 2002] * 1.2

Некоторые ограничения на использование раздела MODEL:

- список столбцов в разделах SELECT и ORDER BY должен содержать только те столбцы, что упоминаются в разделе MODEL;

• в разделе MODEL не допускается использовать псевдонимы столбцов из списка SELECT, но разрешено задавать их напрямую (например, MEASURES BY salary AS sal);

• один и тот же столбец нельзя одновременно указать и в DIMENSION BY, и в MEASURES;

• после ключевых слов DIMENSION BY и MEASURES нельзя указывать NULL или пустую строку (''). Для создания пустого столбца можно использовать функцию CAST, приводящую NULL к нужному типу.

Подробнее — см. документацию ORACLE [14, раздел 23: SQL for Modeling] и статью [16].

21. ПОЛЬЗОВАТЕЛЬСКИЕ ТИПЫ ДАННЫХ

Пользователь Oracle имеет возможность создавать собственные типы данных (user defined data types) на основе существующих — встроенных или ранее созданных. Созданные типы являются самостоятельными объектами схемы Oracle.

К пользовательским типам данных относятся, например, объектный тип (object type), массив переменной длины (varray) и вложенная таблица (nested table). Описание перечисленных типов данных приводится в таблице ниже.

Пользовательский тип данных	**Описание**
Object type	Объектный тип данных. Используется для описания сущностей реального мира (например, заказов, которые обрабатываются приложением). Объектный тип данных включает следующие компоненты: • уникальное в пределах схемы имя объектного типа (может совпадать с именем сущности

Пользовательский тип данных	Описание
	реального мира); • набор полей или атрибутов, типы данных которых могут быть встроенными или созданными пользователями ранее; • набор методов, которые представляют собой функции или процедуры, написанные на языке PL/SQL и хранимые в БД или написанные на написанные на одном из объектных языков программирования, таких как C или Java, и хранимые вне БД. Для доступа к определённому полю объектного типа используют точечный синтаксис: [<схема>.]<псевдоним таблицы>.<имя_объектного_типа_данных>.<имя_аттрибута>. Например, запрос SELECT count (distinct c.cust_address.postal_code) FROM customers c; выдаст количество различных индексов, встречающихся в адресах клиентов. Здесь customers — имя таблицы, содержащей столбец объектного типа данных cust_address, одним из атрибутов которого является postal_code.
Varray	Массив переменной длины — представляет собой упорядоченное множество элементов одного и того же типа данных, который может быть встроенным или созданным пользователями ранее. Каждый элемент массива имеет свой индекс,

Пользовательский тип данных	Описание
	который представляет собой номер позиции данного элемента внутри массива. Длиной массива считается количество элементов в нём. При создании массива переменной длины необходимо указать максимально возможное значение его длины. Для доступа к элементам массива используют функцию TABLE, позволяющую представить массив как виртуальную таблицу, которую затем можно соединить с основной таблицей для получения необходимого сочетания данных[1]. Функция TABLE имеет следующий синтаксис: TABLE (collection_expression). Например, запрос SELECT c.cust_first_name, v.column_value FROM customers c, TABLE(c.phone_numbers) v ORDER BY 1; выдаст имена клиентов наряду с их номерами телефонов. Здесь phone_numbers — название столбца таблицы customers с типом данных varray, column_value — псевдостолбец, использующийся для обращения к единственному столбцу получившейся в результате применения функции TABLE виртуальной таблицы и содержащему элементы массива.

[1] Для соединения в данном случае обязательно использование не-ANSI синтаксиса (предусматривающего написание имён соединяемых таблиц через запятую в разделе FROM) без указания условий соединения.

Пользовательский тип данных	Описание
Nested table	Вложенная таблица — используется для описания неупорядоченного набора элементов одного и того же типа данных, который может быть встроенным или созданным пользователями ранее. Если тип данных элементов (строк) вложенной таблицы — объектный с несколькими атрибутами, то такая вложенная таблица будет содержать несколько столбцов, соответствующих атрибутам этого объектного типа. В других случаях вложенная таблица будет содержать только один столбец. Для доступа к данным вложенной таблицы также используют функцию TABLE. Например, SELECT p.name_obj, n.num FROM people_reltab p, TABLE(p.phones_ntab) n; Здесь phones_ntab — название столбца таблицы people_reltab с типом данных nested_table, phones_ntab — название атрибута объектного типа, являющегося в данном случае столбцом вложенной таблицы.