ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
Магнитогорский государственный технический университет им. Г.И. Носова

А.Б. Белявский, Л.Г. Егорова

Базы данных. Операторы выборки данных

Утверждено Редакционно-издательским советом университета
в качестве учебного пособия

Магнитогорск
2009

УДК 004.655.3 (075.8)

Рецензенты:

Декан Магнитогорского филиала
Московского психолого-социального института,
кандидат педагогических наук
А.Ю. Воробьева

Заведующий кафедрой вычислительной техники и
систем управления ЦПК «Персонал»
А.Н. Бурыкин

Белявский А.Б., Егорова Л.Г.
Базы данных. Операторы выборки данных: учеб. пособие. – Магнитогорск: ГОУ ВПО «МГТУ», 2009. – 98 с.

В пособии рассматриваются механизмы выборки данных языка SQL – оператора SELECT, а также механизмы выборки данных из нескольких таблиц, определяются и иллюстрируются на примерах запросов предикаты и встроенные в язык SQL функции. Пособие содержит большое количество примеров, реализованных в среде ORACLE Database Express Edition.

Пособие рекомендовано для студентов высших учебных заведений, обучающихся по направлению 230100 «Информатика и вычислительная техника», а также для пользователей баз данных, преподавателей и научных сотрудников, сферой деятельности которых является технология хранения и использования баз данных.

УДК 004.655.3 (075.8)

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Эффективное управление сложными организационными системами требует использования современных средств автоматизации. Успешное решение задач возможно только тогда, когда для каждого контингента пользователей автоматизированных систем обработки информации создана отвечающая их потребностям среда обработки данных. Основная цель всякой системы обработки информации – полное и своевременное удовлетворение информационных потребностей пользователей. Пользователь формулирует свои потребности на каком-либо доступном ему языке. Существующие технические средства в состоянии воспринять только язык детальных инструкций, поэтому необходима некоторая компонента системы обработки информации, которая обеспечит взаимодействие с пользователем на приемлемом языковом уровне. Помимо этого необходимы средства, обеспечивающие для разработчика системы приемлемый языковой уровень описания технологических процессов обработки данных.

В процессе эволюции программного обеспечения автоматизированных систем обработки информации возникло четкое понимание возможности и целесообразности разработки и реализации комплексов программ, которые позволяют описывать характерные для обработки данных операции на языке высокого уровня. Такие комплексы получили название системы управления базами данных (СУБД).

Можно выделить две черты, характерные для современных автоматизированных систем: разнообразие задач, решаемых различными пользователями на общей базе данных, и постоянное улучшение аппаратных средств, предназначенных для хранения и обработки данных. Поэтому необходимым условием существования СУБД является реализация принципа логической и физической независимости представления данных. Логической независимостью данных называют возможность изменения логической структуры данных без изменения существующих прикладных программ и технологии обработки данных. Наиболее типичной является ситуация увеличения или уменьшения числа обрабатываемых характеристик какого-либо информационного объекта. Физической независимостью данных называют возможность изменения физической организации данных без перестройки прикладных программ и логической структуры данных.

На сегодняшний день широкое распространение получили системы обработки данных на базе локальных вычислительных систем, представляющих собой группы компьютеров, соединенных высокоскоростными линиями связи. Для таких систем характерной чертой является то, что процессы обработки информации частично выполняются в месте ее получения, но наиболее ресурсоемкие процессы обработки информации происходят на сервере. Одной из таких систем является СУБД Oracle.

В данном пособии представлено описание языка SQL для СУБД Oracle с рядом рабочих примеров, которые демонстрируют основные концепции этого языка.

1. SQL – ЯЗЫК МАНИПУЛИРОВАНИЯ ДАННЫМИ В РЕЛЯЦИОННОЙ БАЗЕ ДАННЫХ

1.1. SQL и его история

Единственным средством общения и администраторов баз данных, и проектировщиков, и разработчиков, и пользователей с реляционной базой данных является структурированный язык запрос SQL (Structured Query Language). SQL есть полнофункциональный язык манипулирования данными в реляционных базах данных. В настоящее время он является общепризнанным, стандартным интерфейсом для реляционных баз данных, таких как Oracle, Informix, Sybase, DB/2, MS SQL Server и ряда других (стандарты ANSI и ISO). SQL – непроцедурный язык, который предназначен для обработки множеств, состоящих из строк и колонок таблиц реляционной базы данных. Хотя существуют его расширения, допускающие процедурную обработку. Проектировщики баз данных используют SQL для создания всех физических объектов реляционной базы данных.

Теоретические основы SQL были заложены Коддом, положившим начало развитию теории реляционных БД. Первая практическая реализация была выполнена в исследовательских лабораториях фирмы IBM Chamberlin D.D. и Royce R.F. Промышленное применение SQL было впервые реализовано в СУБД Ingres. Одной из первых промышленных реляционных СУБД является Oracle. По сути дела, реляционная СУБД - это программное обеспечение, которое управляет работой реляционной базы данных.

Первый международный стандарт языка SQL был принят в 1989 г. (SQL-89). В конце 1992 г. был принят новый международный стандарт SQL-92. В настоящее время большинство производителей реляционных СУБД используют его в качестве базового. Однако работы по стандартизации языка SQL далеки от завершения и уже разработан проект стандарта SQL-99, который вводит в обиход языка понятие объекта и разрешает на него ссылаться в операторах SQL. В исходном варианте SQL не было команд управления потоком данных, они появились в недавно принятом стандарте ISO/IEC 9075-5: 1996 дополнительной части SQL.

Каждой конкретной СУБД соответствует своя собственная реализация SQL, в целом поддерживающая определенный стандарт, но имеющая свои особенности. Эти реализации называются диалектами. Так, стандарт ISO/IEC 9075-5 предусматривает объекты, называемые постоянно хранимыми модулями или PSM-модулями (Persistent Stored Modules). В СУБД Oracle расширение PL/SQL является аналогом указанного выше расширения стандарта.

1.2. Описание основных операторов SQL

SQL состоит из набора команд манипулирования данными в реляционной базе данных, которые позволяют создавать объекты реляционной базы данных, модифицировать данные в таблицах (вставлять, удалять, исправлять), изменять схемы отношений базы данных, выполнять вычисления над данными, делать выборки из базы данных, поддерживать безопасность и целостность данных.

Весь набор команд SQL можно разбить на следующие группы:

- команды определения данных (DDL - Data Defininion Language);
- команды манипулирования данными (DML - Data Manipulation Language);
- команды выборки данных (DQL - Data Query Language);
- команды управления транзакциями;
- команды управления данными.

При выполнении каждая команда SQL проходит четыре фазы обработки:

- фаза синтаксического разбора, которая включает проверку синтаксиса команды, проверку имен таблиц и колонок в базе данных, а также подготовку исходных данных для оптимизатора;
- фаза оптимизации, которая включает подстановку действительных имен таблиц и колонок базы данных в представление, идентификацию возможных вариантов выполнения команды, определение стоимости выполнения каждого варианта, выбор наилучшего варианта на основе внутренней статистики;
- фаза генерации исполняемого кода, которая включает построение выполняемого кода команды;
- фаза выполнения команды, которая включает выполнение кода команды.

В настоящее время оптимизатор является составной частью любой промышленной реализации SQL. Работа оптимизатора основана на сборе статистики о выполняемых командах и выполнении эквивалентных алгебраических преобразований с отношениями базы данных. Такая статистика сохраняется в системном каталоге базы данных. Системный каталог является словарем данных для каждой базы данных и содержит информацию о таблицах, представлениях, индексах, колонках, пользователях и их привилегиях доступа. Каждая база данных имеет свой системный каталог, который представляет совокупность предопределенных таблиц базы данных.

1.3. Учебный пример

Отношения, описывающие классический учебный пример Oracle Corp., представлены таблицами: отделы (DEPT), сотрудники (EMP), категория оплаты (SALGRADE). Иерархическая структура подчиненности сотрудников представлена на рис. 1.1.

1. Таблица DEPT (ОТДЕЛЫ)

Предприятие состоит из отделов, которые имеют номера (DEPTNO), названия (DNAME). Каждый отдел расположен в определенном городе (LOC). Номер отдела (DEPTNO) уникальным образом идентифицирует запись об отделе, то есть является первичным ключом DEPT. В таблицу DEPT включены данные, представленные на рис. 1.2

2. Таблица EMP (СОТРУДНИКИ)

Каждый сотрудник (служащих) имеет уникальный номер - код (EMPNO), который выбран в качестве первичного ключа таблицы - EMP. ENAME – имя сотрудника, JOB – его должность, HIREDATE – дата приема сотрудника на работу, SAL – месячная оплата труда сотрудника в условных единицах, COMM – годовая сумма комиссионных выплат (годовой премии).

DEPTNO – номер отдела, в котором работает сотрудник. Должен выбираться из номеров отделов, присутствующих в таблице DEPT. То есть DEPTNO в таблице EMP является внешним ключом (ссылается на значение первичного ключа таблицы DEPT).

MGR – код руководителя, которому подчинен сотрудник. Должен выбираться из номеров сотрудников, присутствующих в этой же таблице. То есть MGR в таблице EMP является внешним ключом, ссылающимся на значения первичного ключа таблица EMP.

В таблицу EMP включены данные представленные на рис. 1.3.

3. Таблица SALGRADE (КАТЕГОРИИ ОПЛАТЫ)

Заработная плата сотрудников соответствует какой-то одной категории (GRADE). Каждой категории определен интервал уровня заработной платы от и до (LOSAL и HISAL).

Таблица SALGRADE содержит данные представленные на рис. 1.4.

4. Таблица DUAL

Эта псевдо-таблица ORACLE, доступная для выборки всем пользователям. Она состоит из одного столбца DUMMY типа VARCHAR2 и одной строки (рис. 1.5).

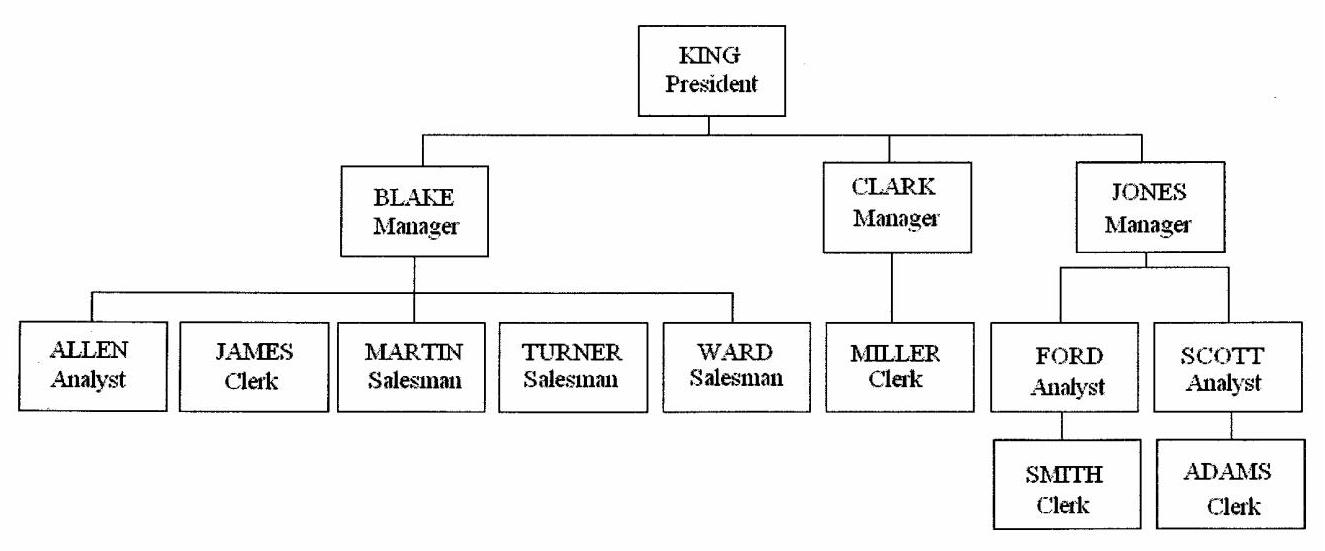

Рис.1.1. Иерархическая структура компании

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Рис. 1.2. Содержимое таблицы DEPT

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	17.11.81	5000	-	10
7698	BLAKE	MANAGER	7839	01.05.81	2850	-	30
7782	CLARK	MANAGER	7839	09.06.81	2450	-	10
7566	JONES	MANAGER	7839	02.04.81	2975	-	20
7654	MARTIN	SALESMAN	7698	28.08.81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20.02.81	1600	300	30
7844	TURNER	SALESMAN	7698	08.09.81	1500	0	30
7900	JAMES	CLERK	7698	03.12.81	950	-	30
7521	WARD	SALESMAN	7698	22.02.81	1250	500	30
7902	FORD	ANALYST	7566	03.12.81	3000	-	20
736	SMITH	CLERK	7902	17.12.80	800	-	20
7788	SCOTT	ANALYST	7566	09.12.82	3000	-	20
7876	ADAMS	CLERK	7788	12.12.83	1100	-	20
7934	MILLER	CLERK	7782	23.12.82	1300	-	10

Рис. 1.3. Содержимое таблицы EMP

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

Рис. 1.4. Содержимое таблицы SALGRADE

DUMMY
X

1 rows returned

Рис. 1.5. Содержимое таблицы DUAL

2. ВЫБОРКА ИНФОРМАЦИИ ИЗ БАЗЫ ДАННЫХ С ПОМОЩЬЮ ОПЕРАТОРА SELECT

2.1. Базовая структура оператора SELECT

Оператор SELECT – один из наиболее важных и самых распространенных операторов SQL. Он позволяет производить выборки данных из таблиц и преобразовывать к нужному виду полученные результаты. Оператор SELECT способен выполнять действия, эквивалентные операторам реляционной алгебры, причем в пределах единственной выполняемой команды. При его помощи можно реализовать сложные и громоздкие условия отбора данных из различных таблиц.

Оператор SELECT – средство, которое полностью абстрагировано от вопросов представления данных, что помогает сконцентрировать внимание на проблемах доступа к данным. Примеры его использования наглядно демонстрируют один из основополагающих принципов больших (промышленных) СУБД: средства хранения данных и доступа к ним отделены от средств представления данных. Операции над данными производятся в масштабе наборов данных, а не отдельных записей.

Оператор SELECT используется для формулирования и выполнения запросов пользователей к данным, хранящимся в базе, то есть производят выборку из базы данных, затребованной пользователем информации.

Она состоит из следующих предложений:

SELECT [DISTINCT] {*, *столбец* [псевдоним], выражение, литерал}
FROM {*таблица* | (*подзапрос*)}
[WHERE *условия выборки для отдельных строк*]
[GROUP BY *столбцы*]
[HAVING *критерий выборки для групп*]
[ORDER BY {*столбец* [ASC | DESC], ...}]

Фраза FROM определяет одну или несколько таблиц или подзапросов, используемых для извлечения данных.

В простейшем случае оператор SELECT состоит из:

- предложения SELECT с указанием столбцов, которые должны быть в таблице результатов;
- предложения FROM, задающие имена таблиц, к которым делается запрос.

Кроме имени таблицы здесь возможно задание подзапроса, результат выполнения которого будет рассматриваться как таблица для выборки данных.

Оператор SELECT определяет поля (столбцы), которые будут входить в результат выполнения запроса. В списке они разделяются запятыми и приводятся в такой очередности, в какой должны быть представлены в результате запроса. Если используется имя поля, содержащее пробелы или разделители, его следует заключить в квадратные скобки. Символом * можно выбрать все поля, а вместо имени поля применить выражение, его заменяющее.

2.2. Выборка столбцов таблицы

Выборка из таблицы всех столбцов
SELECT *список всех столбцов*
FROM *таблица*
или
SELECT {* | *таблица* .* }
FROM *таблица*
Примечание: В последнем случае столбцы выбираются в том же порядке, как они были заданы при создании таблицы.

Пример 2.1. Показать все содержимое таблицы EMP.

SELECT * FROM EMP
Результат выполнения примера 2.1 показан на рис. 2.1.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	17.11.81	5000	-	10
7698	BLAKE	MANAGER	7839	01.05.81	2850	-	30
7782	CLARK	MANAGER	7839	09.06.81	2450	-	10
7566	JONES	MANAGER	7839	02.04.81	2975	-	20
7654	MARTIN	SALESMAN	7698	28.08.81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20.02.81	1600	300	30
7844	TURNER	SALESMAN	7698	08.09.81	1500	0	30
7900	JAMES	CLERK	7698	03.12.81	950	-	30
7521	WARD	SALESMAN	7698	22.02.81	1250	500	30
7902	FORD	ANALYST	7566	03.12.81	3000	-	20
736	SMITH	CLERK	7902	17.12.80	800	-	20
7788	SCOTT	ANALYST	7566	09.12.82	3000	-	20
7876	ADAMS	CLERK	7788	12.12.83	1100	-	20
7934	MILLER	CLERK	7782	23.12.82	1300	-	10

Рис. 2.1. Результат выполнения запроса примера 2.1

Список выбираемых столбцов

SELECT { *столбец* [[as] [псевдоним], ... }
FROM {*таблица1* [, *таблица2* [, *таблица3*] ... }

Порядок, в котором столбцы перечислены в списке, определяет порядок их выборки.

Пример 2.2. Выбрать всех сотрудников с номерами отделов, в которых они работают и кодами их руководителей.

SELECT deptno, ename, mgr FROM emp

Результат выполнения примера 2.2 показан на рис. 2.2.

DEPTNO	ENAME	MGR
10	KING	-
30	BLAKE	7839
10	CLARK	7839
20	JONES	7839
30	MARTIN	7698
30	ALLEN	7698
30	TURNER	7698
30	JAMES	7698
30	WARD	7698
20	FORD	7566
20	SMITH	7902
20	SCOTT	7566
20	ADAMS	7788
10	MILLER	7782

Рис. 2.2. Результат выполнения запроса примера 2.2

Другие объекты выборки

В список выборки команды SELECT могут быть также включены другие объекты:

- арифметические операции {+, -, *, /};
- псевдонимы имен столбцов (например, столбец sal можно заменить Col_1 или "Зарплата");
- конкатенированные столбцы (оператор конкатенации – две вертикальные черты – ||);
- литералы (например фраза 'работает в отделе', взятая в ‘ ').

При выборке часто необходимо или удобно присвоить столбцам или выражениям альтернативные имена–псевдонимы (aliases).

Настоящее имя столбца отделяется от псевдонима пробелом и необязательным словом AS. Псевдонимы подчиняются стандартным правилам именования объектов, однако не относятся ни к какому пространству имен, поскольку существуют лишь во время выполнения запроса, в котором они заданы.

Литералы представляют собой строку, заключенную в одиночные кавычки (апострофы). Если литерал должен содержать одиночную кавычку, то она повторяется дважды.

Пример 2.3. Выбрать имена служащих, их годовой доход и премию. При расчете годового дохода использовать арифметическую операцию – умножение.

SELECT ename, sal*12, comm FROM emp

Результат выполнения запроса показан на рис. 2.3.

ENAME	SAL*12	COMM
KING	60000	-
BLAKE	34200	-
CLARK	29400	-
JONES	35700	-
MARTIN	15000	1400
ALLEN	19200	300
TURNER	18000	0
JAMES	11400	-
WARD	15000	500
FORD	36000	-
SMITH	9600	-
SCOTT	36000	-
ADAMS	13200	-
MILLER	15600	-

Рис. 2.3. Результат выполнения запроса примера 2.3

Пример 2.4. Выбрать имена служащих, их годовой доход и премию, используя арифметическую операцию – умножение и псевдонимы для столбцов "Годовая зарплата" и "Премия".

SELECT ename, sal*12 "Годовая зарплата", comm "Премия" FROM emp

Результат выполнения запроса показан на рис. 2.4.

ENAME	Годовая зарплата	Премия
KING	60000	-
BLAKE	34200	-
CLARK	29400	-
JONES	35700	-
MARTIN	15000	1400
ALLEN	19200	300
TURNER	18000	0
JAMES	11400	-
WARD	15000	500
FORD	36000	-
SMITH	9600	-
SCOTT	36000	-
ADAMS	13200	-
MILLER	15600	-

Рис. 2.4. Результат выполнения запроса примера 2.4

Пример 2.5. Выбрать всех служащих и их номера, используя оператор конкатенации. Результат оформить в виде одного столбца.

```
SELECT empno||ename "Служащие" FROM emp
```

Результат выполнения запроса показан на рис. 2.5.

Служащие
7839 KING
7698 BLAKE
7782 CLARK
7566 JONES
7654 MARTIN
7499 ALLEN
7844 TURNER
7900 JAMES
7521 WARD
7902 FORD
736 SMITH
7788 SCOTT
7876 ADAMS
7934 MILLER

Рис. 2.5. Результат выполнения запроса примера 2.5

Пример 2.6. Выбрать всех служащих, их номера и номера отделов, в которых они работают. При написании запроса использовать оператор конкатенации и литералы (‘ – ‘, ‘работает в отделе’).

SELECT empno||'-'||ename “Служащий”, 'работает в отделе', deptno FROM emp

Результат выполнения запроса показан на рис. 2.6.

Служащий	'работает в отделе'	DEPTNO
7839 - KING	работает в отделе	10
7698 - BLAKE	работает в отделе	30
7782 - CLARK	работает в отделе	10
7566 - JONES	работает в отделе	20
7654 - MARTIN	работает в отделе	30
7499 - ALLEN	работает в отделе	30
7844 - TURNER	работает в отделе	30
7900 - JAMES	работает в отделе	30
7521 - WARD	работает в отделе	30
7902 - FORD	работает в отделе	20
736 - SMITH	работает в отделе	20
7788 - SCOTT	работает в отделе	20
7876 - ADAMS	работает в отделе	20
7934 - MILLER	работает в отделе	10

Рис. 2.6. Результат выполнения запроса примера 2.6

2.3. Обработка пустых значений

В том случае, если некоторое значение не присвоено, не доступно или не определено, оно получает значение NULL (оно не эквивалентно значению 0 (ноль)).

Если хотя бы один из элементов выражения имеет NULL-значение, то и результат этого выражения неопределён. Чтобы получить результат необходимо преобразовать NULL-значение. Для этого используется функция NVL.

Пример 2.7. Выбрать имена (ename) и годовой доход всех служащих. Годовой доход определяется как сумма годового заработка (sal*12) и премии (comm). Так как премия выплачивается не всем служащим в запросе (рис. 1.3), то целесообразно использовать функцию NVL.

SELECT ename "ИМЯ", sal*12+nvl(comm,0) "ГОДОВОЙ ДОХОД" from emp

Результат выполнения запроса показан на рис. 2.7.

ИМЯ	ГОДОВОЙ ДОХОД
KING	60000
BLAKE	34200
CLARK	29400
JONES	35700
MARTIN	16400
ALLEN	19500
TURNER	18000
JAMES	11400
WARD	15500
FORD	36000
SMITH	9600
SCOTT	36000
ADAMS	13200
MILLER	15600

Рис. 2.7. Результат выполнения запроса примера 2.7

В данном примере предписывается интерпретировать NULL-премиальные как отсутствие премиальных (т.е. равных 0).

2.4. Упорядочение строк результата запроса

При выдаче результата запроса порядок строк в нем неопределен и может меняться от запроса к запросу. Для сортировки строк используется предложение ORDER BY, которое завершает команду SELECT.

По умолчанию сортировка выполняется в порядке возрастания значений полей (ASC). Для изменения порядка сортировки используется квалификатор DESC в предложении ORDER BY. Для сортировки по нескольким столбцам столбцы в предложении ORDER BY задаются через запятую. При сортировке пустые значения считаются больше любого непустого значения. Для того чтобы не повторять во фразе ORDER BY сложные выражения используют псевдонимы. Результаты выполнения запросов с применением предложения ORDER BY представлены на рис. 2.8–2.10.

Пример 2.8. Выбрать имена (ename) и годовой доход всех служащих (включая премию). Отсортировать результат по именам служащих в алфавитном порядке.

SELECT ename, sal*12+nvl(comm,0) FROM emp ORDER BY ENAME

ENAME	SAL*12+NVL(COMM,0)
ADAMS	13200
ALLEN	19500
BLAKE	34200
CLARK	29400
FORD	36000
JAMES	11400
JONES	35700
KING	60000
MARTIN	16400
MILLER	15600
SCOTT	36000
SMITH	9600
TURNER	18000
WARD	15500

Рис. 2.8. Результат выполнения запроса примера 2.8

Пример 2.9. Выбрать имена (ename) и годовой доход всех служащих (включая премию). Отсортировать результат по именам служащих в обратном порядке.

SELECT ename, sal*12+nvl(comm,0) FROM emp ORDER BY ename DESC

ENAME	SAL*12+NVL(COMM,0)
WARD	15500
TURNER	18000
SMITH	9600
SCOTT	36000
MILLER	15600
MARTIN	16400
KING	60000
JONES	35700
JAMES	11400
FORD	36000
CLARK	29400
BLAKE	34200
ALLEN	19500
ADAMS	13200

Рис. 2.9. Результат выполнения запроса примера 2.9

Пример 2.10. Выбрать номера отделов (deptno), должности (job) служащих этих отделов и месячную зарплату (sal). Отсортировать результат по возрастанию номеров отделов и в обратном порядке по месячной зарплате.

SELECT deptno, job, sal FROM emp ORDER BY deptno, sal DESC

DEPTNO	JOB	SAL
10	PRESIDENT	5000
10	MANAGER	2450
10	CLERK	1300
20	ANALYST	3000
20	ANALYST	3000
20	MANAGER	2975
20	CLERK	1100
20	CLERK	800
30	MANAGER	2850
30	SALESMAN	1600
30	SALESMAN	1500
30	SALESMAN	1250
30	SALESMAN	1250
30	CLERK	950

Рис. 2.10. Результат выполнения запроса примера 2.10

Без предложения ORDER BY строки в таблице результатов выдаются в порядке их выборки из базы данных. Этот порядок может не отражать физического расположения строк в базе данных и может меняться от запроса к запросу.

2.5. Предотвращение выборки повторяющихся строк

По запросу оператор SELECT выдает все строки, соответствующие запросу, включая одинаковые. Чтобы исключить одинаковые строки из результирующей таблицы задается квалификатор DISTINCT. SELECT DISTINCT со списком выбираемых столбцов реализует операцию проекции отношения – выбираются заданные атрибуты и исключаются повторения.

Пример 2.11. Получить перечень должностей в каждом отделе и отсортировать результат по возрастанию номеров отделов. В 20 и 30 отделе присутствуют служащие с одинаковыми должностями (см. рис. 1.3). Для того чтобы исключить повторение должностей в этих отделах в запросе необходимо использовать квалификатор DISTINCT.

SELECT DISTINCT deptno, job FROM emp order by deptno
Результат выполнения запроса показан на рис. 2.11.

DEPTNO	JOB
10	PRESIDENT
10	CLERK
10	MANAGER
20	MANAGER
20	ANALYST
20	CLERK
30	MANAGER
30	SALESMAN
30	CLERK

Рис. 2.11. Результат выполнения запроса примера 2.11

2.6. Выборка строк из таблицы по заданным условиям

Для задания условий, которым должны соответствовать значения столбцов требуемых строк таблицы, используется фраза WHERE <условия_выборки>. Фраза WHERE определяет условие, которому должны удовлетворять все строки, используемые для формирования результирующего набора.

```
SELECT [DISTINCT] список_столбцов
FROM {таблица | подзапрос}
WHERE условия_выборки
[ORDER BY список_сортировки]
```

Во фразе WHERE в виде логического выражения задается критерий выборки строк из таблиц базы данных. Если она задана, то всегда следует за предложением FROM.

В случае задания фразы WHERE выбираются только те строки, для которых значение заданного критерия – ИСТИНА (TRUE), а отвергаются те строки, для которых критерий имеет значение – ЛОЖЬ (FALSE) или НЕОПРЕДЕЛЕН (UNKNOWN).

При выполнении выборки данных в критерии отбора задаваемым ключевым словом WHERE используются специальные операции IN, BETWEEN, LIKE, IS NULL и предикаты с квантором. Определение условий выборки с помощью этих операции представлено в табл. 2.1.

Таблица 2.1

Определение условий выборки во фразе WHERE

Оператор	Смысл оператора	Пример
=	Равно	SELECT * FROM emp WHERE sal = 1500
!= , ^= или <>	Не равно	SELECT * FROM emp WHERE sal != 1500
>	Больше	SELECT * FROM emp WHERE sal > 1500
>=	Больше или равно	SELECT * FROM emp WHERE sal >= 1500
<	Меньше	SELECT * FROM emp WHERE sal < 1500
<=	Меньше или равно	SELECT * FROM emp WHERE sal <= 1500
[NOT] BETWEEN x AND y	[не] больше или равно **x И** меньше или равно **y**	SELECT * FROM emp WHERE sal BETWEEN 1000 AND 2000
IN (список)	Равно любому элементу списка. Эквивалентно ANY	SELECT * FROM emp WHERE job IN (‘CLERK’, ‘ANALYST’) SELECT * FROM emp WHERE sal IN (SELECT sal FROM emp WHERE deptno = 30)
NOT IN (список)	Не равно ни одному элементу списка.Эквивалентно «!=ALL ». Если хоть один из элементов списка NULL, то результатом будет FALSE	SELECT * FROM emp WHERE job NOT IN (‘CLERK’, ANALYST’) SELECT * FROM emp WHERE sal NOT IN (SELECT sal FROM emp WHERE deptno = 30)
ANY SOME	Сравнение с каждым элементом списка или подзапроса. Должно следовать за = , != , < , <= , > , >=. Принимает значение FALSE, если подзапрос не выбирает ни одной строки	SELECT * FROM emp WHERE job = ANY (‘CLERK’, ‘ANALYST’) SELECT * FROM emp WHERE sal = ANY (SELECT sal FROM emp WHERE deptno = 30)

Окончание табл. 2.1

Оператор	Смысл оператора	Пример
ALL	Сравнение с каждым элементом списка или подзапроса. Должно следовать за = , != , < , <= , > , >=. Принимает значение TRUE, если подзапрос не выбирает ни одной строки	SELECT * FROM emp WHERE sal >= ALL (1400, 3000)
IS [NOT] NULL	Проверка на NULL. Это единственные операторы, которые должны использоваться при проверке значения на NULL	SELECT ename, deptno FROM emp WHERE comm IS NULL
x [NOT] LIKE y [ESCAPE z]	TRUE. Если x удовлетворяет шаблону y. В шаблоне y символ «%» соответствует любой строке из нуля или более символов, а символ «_» – одному символу. z – любой символ, кроме «%» и «_». Символы «%» и «_» будут трактоваться как литералы, если следуют за символом, заданным в ESCAPE	SELECT * FROM emp WHERE ename LIKE ‘%S’ SELECT * FROM emp WHERE ename LIKE 'A_C%/%' ESCAPE '/'

Символьные строки и даты в предложении WHERE должны быть взяты в одинарные кавычки. Большинство компонентов SQL – операторов не чувствительны к регистру. Это означает, что могут использоваться любые буквы – как строчные, так и прописные. Одним важным исключением из этого правила являются символьные

данные, которые должны вводиться точно так же, как были введены соответствующие им значения, хранящиеся в базе данных.

Пример 2.12. Выбрать всех служащих, работающих клерками (CLERK).

SELECT ename, empno, job FROM emp WHERE job='CLERK'

Результат выполнения запроса показан на рис. 2.12.

ENAME	EMPNO	JOB
JAMES	7900	CLERK
SMITH	736	CLERK
ADAMS	7876	CLERK
MILLER	7934	CLERK

Рис. 2.12. Результат выполнения запроса примера 2.12

Предикат IN определяет множество, вхождение в которое определяет истинность предиката.

Пример 2.13. Выбрать служащих, номер руководителя которых 7902 или 7566.

SELECT ename, sal, mgr FROM emp WHERE mgr IN (7902, 7566)

Результат выполнения запроса показан на рис. 2.13.

ENAME	SAL	MGR
FORD	3000	7566
SMITH	800	7902
SCOTT	3000	7566

Рис. 2.13. Результат выполнения запроса примера 2.13

Предикат BETWEEN используется для поиска значения внутри некоторого интервала, определяемого своими минимальным и максимальным значениями. При этом указанные значения включаются в условие поиска. Ключевое слово BETWEEN указывается перед начальным значением, затем идет ключевое слово AND и завершает конструкцию конечное значение. Для предиката BETWEEN порядок следования начального и конечного значения важен.

Пример 2.14. Выбрать служащих, оклад которых больше или равен 1000 и меньше или равен 2000.

SELECT ename,sal FROM emp WHERE sal BETWEEN 1000 AND 2000

Результат выполнения запроса показан на рис. 2.14.

ENAME	SAL
MARTIN	1250
ALLEN	1600
TURNER	1500
WARD	1250
ADAMS	1100
MILLER	1300

Рис. 2.14. Результат выполнения запроса примера 2.14

Для задания сложного критерия поиска простые логические выражения объединяются в сложные операции AND и OR.

Пример 2.15. Выбрать служащих, оклад которых больше или равен 1000 и меньше или равен 2000 и работающих клерками.

SELECT ename, sal FROM emp WHERE sal BETWEEN 1000 AND 2000 AND job ='CLERK'

Результат выполнения запроса показан на рис. 2.15.

ENAME	SAL
ADAMS	1100
MILLER	1300

Рис. 2.15. Результат выполнения запроса примера 2.15

Предикат LIKE применим только к полям типа CHAR, VARCHAR и VARCHAR2, т.е. к текстовым полям. С помощью предиката LIKE можно выполнять сравнение выражения с заданным шаблоном, в котором допускается использование символов-заменителей. Для задания поискового критерия используются два специальных символа:

– % (процент) – заменяет любую последовательность символов, включая строку нулевой длины (‘’);

– _ (символ подчеркивания) заменяет точно один любой символ в заданной позиции.

Предикат LIKE принимает истинное значение при вхождении определенной подстроки в строку в соответствии строки заданному шаблону.

Пример 2.16. Найдите всех служащих, имена которых содержат комбинации символов TH или LL.

SELECT ename FROM emp WHERE ename LIKE '%TH%' OR ename LIKE '%LL%'

Результат выполнения запроса показан на рис. 2.16.

ENAME
ALLEN
SMITH
MILLER

Рис. 2.16. Результат выполнения запроса примера 2.16

Для обработки неопределенных значений в языке SQL используется специальный предикат IS NOT NULL. Предикат IS [NOT] NULL всегда принимает значения TRUE и FALSE. При этом значение x IS NULL равно TRUE тогда и только тогда, когда значение x не определено. Значение предиката x IS NOT NULL соответствует значению NOT (x IS NULL).

Пример 2.17. Покажите всех служащих, имеющих руководителя.

SELECT ename, job, sal FROM emp WHERE mgr IS NOT NULL

Результат выполнения запроса показан на рис. 2.17.

ENAME	JOB	SAL
BLAKE	MANAGER	2850
CLARK	MANAGER	2450
JONES	MANAGER	2975
MARTIN	SALESMAN	1250
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
JAMES	CLERK	950
WARD	SALESMAN	1250
FORD	ANALYST	3000
SMITH	CLERK	800
SCOTT	ANALYST	3000
ADAMS	CLERK	1100
MILLER	CLERK	1300

Рис. 2.17. Результат выполнения запроса примера 2.17

Если NULL – значение задается в операторе сравнения, то это должен быть либо оператор IS NULL, либо оператор IS NOT

NULL. При сравнении с NULL – значениями любые другие операторы дают неизвестный (UNKNOWN) результат, а значит строки не попадут в выборку.

Если в логическом выражении присутствуют AND и OR, то сначала выполняется операция AND, а затем OR. Для изменения порядка выполнения логических операций используются круглые скобки.

В выражении логические операторы выполняются после всех остальных в порядке их приоритета от высшего к низшему.

Приоритет логических операторов:

1. Оператор отрицания логического выражения – NOT (высший приоритет).
2. Оператор AND.
3. Оператор OR (низший приоритет).

Предикат ALL несет стандартную нагрузку квантора всеобщности, а предикаты ANY или SOME соответствуют стандартно понимаемому квантору существования. Предикаты ANY и SOME в Oracle несут одинаковую смысловую нагрузку и полностью взаимозаменяемы. В любом запросе вместо предиката ANY можно использовать SOME и наоборот – результат будет одинаков. Предикаты ANY и SOME принимают значение FALSE, если запрос не выбирает ни одной строки. Предикат ALL принимает значение TRUE, если запрос не выбирает ни одной строки.

Пример 2.18. Покажите всех служащих, работающих клерками и аналитиками.

```
SELECT * FROM emp WHERE job = ANY  ('CLERK', 'ANALYST')
```

Результат выполнения запроса показан на рис. 2.18.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7900	JAMES	CLERK	7698	03.12.81	950	-	30
7902	FORD	ANALYST	7566	03.12.81	3000	-	20
736	SMITH	CLERK	7902	17.12.80	800	-	20
7788	SCOTT	ANALYST	7566	09.12.82	3000	-	20
7876	ADAMS	CLERK	7788	12.12.83	1100	-	20
7934	MILLER	CLERK	7782	23.12.82	1300	-	10

Рис. 2.18. Результат выполнения запроса примера 2.18

3. SQL–ФУНКЦИИ ORACLE

Набор встроенных в язык SQL–функций позволяет выполнить многие типовые операции обработки данных вызовом соответствующей функции. SQL–функции бывают двух типов:

- однострочные (single row) или скалярные функции;
- групповые (group) или агрегатные функции.

Эти два вида функций отличаются тем, что обрабатывают данные разного числа строк. Однострочные возвращают один результат для каждой строки, выбранной по запросу из таблицы или представления. Групповые же функции возвращают одну строку результата для группы строк, выбранных по запросу.

Все встроенные функции можно использовать в SQL–выражениях (как в списке после ключевого слова SELECT, так и в других конструкциях). Очевидно, что, имея привилегии на выборку данных из любой таблицы, можно вычислить значение встроенной функции из любого набора параметров (указав ее в перечне списка SELECT). Однако имеется еще один способ. Каждая база данных Oracle имеет специальную общедоступную таблицу с именем dual. Эта таблица находится в схеме пользователя SYS и содержит один столбец dummy и одну строку (см. рис. 1.5). Для нее существует общий синоним. Один из наиболее распространенных вариантов использования таблицы dual – вычисление результатов SQL–функций с помощью запросов к ней.

Функции подобны оператору, обрабатывающему элементы данных и возвращаемому результат. SQL–функции отличаются от оператора форматом своего представления и задания аргументов. Формат функций позволяет использовать один, два, три и более аргументов или возвращать результат не получая ни одного аргумента:

FUNCTION [(argument1 [, argument2 […]])].

При передаче функции аргумента, отличного от типа, требуемого функцией, неявно выполняется необходимое преобразование типов данных.

3.1. Однострочные функции

3.1.1. Числовые функции

Однострочные числовые функции обрабатывают числовые аргументы и возвращают числовые значения. Большинство этих функций возвращает значение с точностью 38 десятичных знаков.

Тригонометрические функции COS, COSH, EXP, LN, LOG, SIN, SINH, SQRT, TAN и ATAN возвращают значения с точностью 36 десятичных цифр, функции ACOS, ASIN, ATAN и ATAN2 – с точностью 30 десятичных цифр. Список числовых функций приведен в табл. 3.1.

Таблица 3.1

Числовые функции

Функция	Возвращаемое значение
ROUND(n)	n, округленное до m десятичных знаков после точки. По умолчанию m = 0
TRUNC(n,[m])	n, усеченное до m десятичных цифр после точки. По умолчанию m = 0
CEIL(n)	Наименьшее целое, большее или равное n
FLOOR(n)	Наибольшее целое, меньшее или равное n
POWER(n)	m, возведенное в степень n
EXP(n)	e, возведенное в степень n
SQRT(n)	Квадратный корень из n или NULL, если n < 0
SIGN(n)	-1, если n < 0 или 0, если n = 0 и 1, при n > 0
ABS(n)	Абсолютное значение n
MOD(m, n)	Остаток от деления m на n
LN(n)	Натуральный логарифм n, где n > 0
LOG(m, n)	Логарифм по основанию m числа n
SIN(n)	Синус угла n (угол задается в радианах)
SINH(n)	Гиперболический синус n
TAN(n)	Тангенс угла n (угол задается в радианах)
TANH(n)	Гиперболический тангенс n
COS(n)	Косинус угла n (угол задается в радианах)
COSH(n)	Гиперболический косинус n

3.1.2. Символьные функции

Однострочные символьные функции обрабатывают символьные и числовые аргументы и могут возвращать как символьные, так и числовые значения. Если иное не оговорено особо, то эти функции возвращают значения типа VARCHAR2, максимальная длина результата ограничена до 2000 байт. Если функция возвращает данные типа CHAR, то длина результата ограничена 255 байт. При превышении указанных ограничений результирующая строка автоматически укорачивается до максимально допустимой длины, при этом не выдается никакого сообщения об ошибке.

Функции над отдельными строками обладают свойством вложенности. Количество уровней вложенности практически не ограничено.

Список символьных функций приведен в табл. 3.2.

Таблица 3.2

Символьные функции

Функция	Возвращаемое значение
LOWER(*столбец/значение*)	Строка, все символы в которой строчные
UPPER(*столбец/значение*)	Строка, все символы в которой прописные
INITCAP(*столбец/значение*)	Строка, в которой начальная буква каждого слова прописная
CONCAT(*строка1,строка2*)	Сцепленные строки *строка1* и *строка2*. Эквивалентна *строка1* || *строка2*
LPAD(*столбец/значение*,n [,'*строка*'])	Строка, дополненная слева до длины n цепочками символов *строка* (по умолчанию *пробелами*), n – общая длина строки
RPAD(*столбец/значение*,n [,'*строка*'])	Строка, дополненная справа до длины n цепочками символов *строка* (по умолчанию *пробелами*), n – общая длина строки
SUBSTR(*столбец/значение*, m,n)	Подстрока, начинающаяся с m-го символа и имеющая длину n символов (если n опущено, до конца строки)
INSTR(*строка1,строка2* [,n[,m]])	Позиция m-го вхождения *строка2* в *строка1*, поиск начать с n-го символа в строке *строка1*. По умолчанию n=1, m=1. Возвращаемая позиция вычисляется относительно первого символа, даже если n>1
LTRIM(*столбец/значение [,'символы'])*	Строка, из которой удалены начальные символы, входящие в *символы*. По умолчанию *символы* – один пробел
RTRIM*(столбец/значение [,'символы'])*	Строка, из которой удалены конечные символы, входящие в *символы*. По умолчанию *символы* – один пробел
SOUNDEX(столбец/значение)	Строка, представляющая фонетический эквивалент аргумента (для англ. языка)

Окончание табл. 3.2

Функция	Возвращаемое значение
TRANSLATE*(столбец/значение, строка1, строка2)*	Строка, преобразованная из набора символов *строка1* в набор символов *строка2* (для корректной работы функции *строка2* не должна представляться пустой строкой)
REPLACE(*столбец/значение*, *строка1*[,*строка2*])	Строка, в которой каждое вхождение цепочки *строка1* заменено цепочкой *строка2*. По умолчанию *строка2* задает пустую строку, что приводит к удалению всех вхождений *строка1*
LENGTH(*столбец/значение*)	Длина строки в символах

3.1.3. Функции работы с календарными датами

Функции для работы с календарными датами оперируют с данными типа DATE. Все они возвращают значение типа DATE. Исключением является функция MONTHS_BETWEEN, которая возвращает числовое значение. Список функций приведен в табл. 3.3.

Таблица 3.3

Функции для работы с календарными датами

Функция	Возвращаемое значение
MONTHS_BETWEEN (*дата1*, *дата2*)	Число месяцев, между *дата1* и *дата2*
ADD_MONTHS (*дата*, n)	Дата *дата* плюс n месяцев
LAST_DAY(*дата*)	Дата последнего дня месяца, содержащего дату *дата*
NEXT_DAY(*дата*, *значение*)	Дата ближайшего дня недели, имеющего имя или номер *значение*, которая равна или позже *дата*
ROUND(*дата*, *формат*')	*дата*, округленная в соответствии с *формат*
TRUNC(*дата*, *строка*)	*дата*, усеченная в соответствии с *формат*
SYSDATE	Текущие дата и время. Пример: Select SYSDATE from SYS.DUAL

Формат календарных дат, принимаемых по умолчанию, определяется либо явно значением параметра NLS_DATE_FORMAT, либо неявно значением NLS_TERRITORY. Он может быть определен также командой ALTER SESSION. Суммарная длина шаблона преобразования календарных дат не может превышать 22 символов.

В шаблонах, используемых для вывода данных, нельзя использовать дважды одни и те же элементы. Кроме того, запрещено смешивать элементы, представляющие одну и ту же информацию. Например, недопустимо использование в одной строке шаблона элементов ‘SYYY’ и ‘BC’. Шаблоны преобразования календарных дат приведены в табл. 3.4.

Таблица 3.4

Шаблоны преобразования календарных дат

Маска	Значение
– (минус) / (слэш) , (запятая) . (точка) ; (точка с запятой) : (двоеточие) “текст”	Знаки пунктуации и текст, заключенный в двойные кавычки, воспроизводимый в результате без изменений
Век	
A.D. или AD	Индикатор дат нашей эры с точками и без точек
B.C. или BC	Индикатор дат до нашей эры с точками и без точек
CC или SCC	Век. Префикс ‘S’ предписывает отмечать даты до нашей эры знаком ‘-’ (минус)
Год	
IYY IY I	Последние 3, 2, 1 цифры года по стандарту ISO
IYYY	4 цифры года по стандарту ISO
RR	Две последние цифры года (для других веков)
Y,YYY	Год с запятой в данной позиции
SYEAR или YEAR	Год прописью (по-английски). Префикс ‘S’ предписывает отмечать даты до нашей эры знаком ‘-’ (минус)

Продолжение табл. 3.4

Маска	Значение
SYYYY или YYYY	4 цифры года. Префикс ‘S’ предписывает отмечать даты до нашей эры знаком ‘-’ (минус)
YYY YY Y	Последние 3, 2, 1 цифры года.
Квартал	
Q	Квартал года (1–4; январь-март = 1)
Месяц	
MM	Месяц года (1–12; январь = 1)
MONTH	Название месяца, дополненное пробелами до 9 символов.
MON	Трехсимвольная аббревиатура названия месяца.
RM	Месяц года римскими цифрами (I–XII; январь = I).
Неделя	
IW	Неделя года по стандарту ISO (1–52 или 1–53).
WW	Неделя года (1–53), где неделя 1 начинается в первый день года и оканчивается на седьмой день года.
W	Неделя месяца (1–5), где неделя 1 начинается в первый день месяца и оканчивается на седьмой день
День	
D	Номер дня недели (1–7)
DD	Номер дня месяца (1–31)
DDD	Номер дня года (1–366)
DAY	Название дня недели
DY	Трехсимвольная аббревиатура названия дня недели
J	Номер дня по Юлианскому календарю – число дней с 1 января 4712 г. до н.э. Число, заданное для преобразования в дату по этому формату, должно быть целым
A.M. или AM	Индикатор времени до полудня с точками или без точек

Окончание табл. 3.4

Маска	Значение
Время	
P.M. или PM	Индикатор времени после полудня с точками или без точек
HH HH12	Часы дня от 1 до 12
HH24	Часы дня от 0 до 23
MI	Минута часа (1–59)
SS	Секунда (1–59)
SSSSS	Количество секунд, прошедших с начала суток, т.е. после полуночи (1–86399)

Язык, на котором выводятся данные, соответствующие элементам MONTH, MON, DAY, DY, BC, B.C., AD, AM, A.M., RM, и R.M., определяются либо явно параметром NLS_DATE_LANGUAGE или командой ALTER SESSION, либо неявно значением NLS_LANGUAGE.

В соответствии со стандартом год, используемый для определения порядкового номера недели, может отличаться от календарного года. Например, 1 января 1988 года относится к 53 неделе 1987 года. Согласно ISO номер недели года определяется следующим образом:

- если 1 января выпадает на пятницу, субботу или воскресенье, то эта неделя, включая 1 января, есть последняя неделя прошлого года, поскольку большинство ее дней относится к прошлому году;
- если 1 января выпадает на понедельник, вторник, среду или четверг, то эта неделя является первой неделей нового года, поскольку большинство ее дней относится к новому году.

Таким образом, для получения данных, соответствующих стандартам ISO, следует использовать элементы IW, I. IY, IYY, IYYY.

Элемент RR, хотя он и подобен элементу YY, обеспечивает дополнительную гибкость при задании и хранении дат других веков.

Если в функции TO_DATE используется элемент YY, то возвращаемое ей значение всегда относится к текущему веку. При использовании элемента RR век даты, возвращаемый функцией, зависит от значения двух цифр текущего года и двух цифр года, указанных при обращении к TO_DATE (табл. 3.5).

Таблица 3.5

Формат дат RR

		Если две последние цифры заданного года	
Если две последние цифры текущего года		0–49	50–99
	0–49	Результат – год текущего столетия	Результат – год предыдущего столетия
	50–99	Результат – год следующего столетия	Результат – год текущего столетия

Вывод прописными или заглавными буквами слов или римских цифр задается регистром соответствующего элемента шаблона. Например, указав ‘DAY’, получим ‘ВТОРНИК’, ‘Day’ – ‘Вторник’, ‘day’ – ‘вторник’. Для преобразования дат используются суффиксы преобразования дат (табл. 3.6).

Таблица 3.6

Суффиксы шаблонов преобразования дат

Суффикс	Назначение	Пример элемента	Пример результата
TH	Порядковый номер	DDTH	4TH
SP	Число прописью	DDSP	FOUR
SPTH или THSP	Порядковый номер прописью	DDSPTH	FOURTH

При использовании данных суффиксов возвращаемое значение всегда представлено для английского языка. Данные суффиксы применимы только при выводе данных и не могут использоваться при вводе.

3.1.4. Функции преобразования типов данных

Функции выполняют преобразование одного типа данных к другому. Список функций преобразования типов данных приведен в табл. 3.7.

Таблица 3.7

Функции преобразования типов данных

Функция	Возвращаемое значение
TO_CHAR(*число/дата* [,'*формат*'])	Преобразование *число/дата* из типа NUMBER или DATE в тип CHAR, согласно формату специфицированному *формат*
TO_NUMBER(*строка* [,'*формат*'[,'*язык*']])	*строка*, которая содержала представление числа в необязательном формате *формат* в значение типа NUMBER. *язык* задает язык, который используется для числовых разделителей и символов валюты
TO_DATE('*строка*' [,'*формат*'[,'*язык*']])	*строка*, преобразованная из значения CHAR, имеющего формат *формат*, в значение типа DATE. Если *формат* опущен, то *строка* должна иметь умалчиваемый формат даты. *язык* задает язык, используемый для названий дней и месяцев

Для преобразования чисел используются шаблоны, представленные в табл. 3.8.

Таблица 3.8

Шаблоны преобразования чисел

Элемент	Пример маски	Пример	Описание
9	9999	1234	Задает количество возвращаемых цифр. Отрицательные числа дополняются слева знаком '-', положительные пробелом. Ведущие нули подавляются, за исключением нуля, представляющего целую часть дробного числа
0	0999 9990	012 120	Вернуть ведущие нули. Вернуть хвостовые нули
$	$9999	$1234	Вернуть число, дополненное слева знаком доллара
B	B9999	123	Вернуть пробелы для целой части числа, в случае если эта целая часть равна нулю (независимо от того есть или нет в шаблоне формат '0')

Окончание табл. 3.8

Элемент	Пример маски	Пример	Описание
MI	9999MI	1234-	Отрицательное число дополнить справа ‘-’ (минусом), а положительное число пробелом
S	S9999 9999S	+1234 1234+	Отрицательное число дополнить слева ‘-’ (минусом), положительное – ‘+’ (плюсом). Отрицательное число дополнить справа ‘-’ (минусом), положительное – ‘+’ (плюсом)
PR	99999PR	<1234>	Вернуть отрицательное число в <угловых скобках>, положительное – дополнить ведущим и хвостовым пробелом
D	99D99	12.34	Десятичный разделитель (например ‘.’) в данной позиции
G	9G999	1,234	Десятичный разделитель (например ‘,’) в данной позиции
C	C999		Обозначение национальной валюты по стандарту ISO в данной позиции
L	999L		Территориальное обозначение национальной валюты в данной позиции
,	99,999	12,345	Вернуть запятую в данной позиции.
.	999.99	123.45	Вернуть десятичную точку в данной позиции
V	999V99		Вернуть число, умноженное на 10 в степени n (если потребуется округлить его). Степень n равна количеству девяток после ‘V’ (в данном примере n = 2)
EEEE	9.99EEEE	1.234E+02	Вернуть число с плавающей точкой
RN rn		XXVI Xxvi	Вернуть число, записанное римскими цифрами в верхнем регистре. Вернуть число, записанное римскими цифрами в нижнем регистре. Допустим для целых чисел от 1 до 3999
FM		90.9	Вернуть число без ведущих и хвостовых пробелов

Если шаблон не содержит элементов MI, S или PR, то для отрицательных чисел слева автоматически добавляется знак '-' (минус), а для положительных – пробел.

В шаблоне может быть только один десятичный разделитель (D) или (.) и несколько разделителей десятичных разрядов (G) или (.).

Элементы MI и PR могут быть заданы только в последней позиции шаблона. Элемент S допускается как в первой, так и в последней позиции.

Список функции для обработки данных любого типа представлен в табл. 3.9.

Таблица 3.9

Функции для обработки данных любого типа

Функция	Возвращаемое значение
DECODE(*столбец/ выражение*, *значение 1, результат 1* [,*значение2, результат 2*,[...]][,*deflt*])	Если значение *столбец/выражение* равно одному из *значение**, возвращает соответствующее значение *результат**, в противном случае возвращает значение *deflt*. Функция должна иметь минимум 4 параметра, *столбец/выражение* может быть любого типа, *значение** в списке должны быть того же типа, что и *столбец/ выражение*
NVL(*столбец/выражение*, *результат*)	*Результат* если *столбец/выражение*, иначе *столбец/выражение*
GREATEST(*столбец*1/ *результат 1* [,*столбец 2*/*результат 2* [,...]])	Возвращает *столбец*/результат** с наибольшим значением
LEAST(*столбец*1/*результат 1* [,*столбец2*/ *результат2* [,...]])	Возвращает *столбец*/результат** с наименьшим значением
VSIZE(*столбец/значение*)	Число байт во внутреннем представлении *столбец/значение*

Подробное описание функций приведено в прил. 1, 2.

3.2. Групповые функции и фраза GROUP BY

Групповые функции названы так потому, что они возвращают результат, полученный при обработке групп строк, выбираемых из таблицы или представления, а не каждой строки, как это делали однострочные функции.

У групповых функций есть следующие опции:

- DISTINCT – предписывает групповой функции использовать для вычисления результата только различающиеся значения заданного выражения.
- ALL – предписывает групповой функции использовать для вычисления результата все, в том числе и повторяющиеся, значения заданного выражения. Если опция не задана, то по умолчанию принимается ALL.

Например, если задана DISTINCT, то средняя величина для 1,1,1 и 3, будет равна 2, если же указать ALL, то результатом будет 1,5.

Все групповые функции, кроме записи функции COUNT с аргументом (*), игнорируют NULL – значения. Для заданий способа интерпретации NULL – значений в вычислениях можно использовать функцию NVL.

Групповая функция возвращает NULL, если запрос, в котором она используется:

- не возвращает ни одной строки;
- для всех возвращаемых строк значение выражения, заданного в аргументе, есть NULL.

Исключение из этого правила – COUNT, у которой в качестве аргумента указан символ * (COUNT(*)) всегда возвращающая определенное значение, то есть 0 или положительное число.

Полный список групповых функций приведен в табл. 3.10.

Таблица 3.10

Групповые функции

Функция	Возвращаемое значение
AVG([DISTINCT | ALL] n)	Среднее значение группы полей в столбце, без учета NULL–значений
COUNT([DISTINCT | ALL] *выражение*)	Количество строк в группе, в которых *выражение* имеет не пустое значение
COUNT(*)	Общее количество строк в группе
MAX([DISTINCT | ALL] *выражение*)	Максимальное значение выражения в группе
MIN([DISTINCT | ALL] *выражение*)	Минимальное значение выражения в группе
STDDEV([DISTINCT | ALL] n)	Среднеквадратичное отклонение в группе
SUM([DISTINCT | ALL] n)	Сумма значений по группе, игнорируя NULL–значения
VARIANCE([DISTINCT | ALL] n)	Дисперсия в группе

Рассмотрим групповые функции.

Функция COUNT([DISTINCT | ALL] *выражение*) - подсчет количества всех значений столбцов, за исключением значения NULL и с учетом указания фраз ALL или DISTINCT. Пример применения функции COUNT представлен на рис. 3.1.

Пример 3.1. Подсчитать количество отделов в компании.

```
SELECT COUNT (ALL DEPTNO) "КОЛ-ВО ОТДЕЛОВ" FROM DEPT
```

КОЛ-ВО ОТДЕЛОВ
4

Рис. 3.1. Результат выполнения запроса примера 3.1

Функция COUNT (*) – подсчет количества всех значений столбцов в группе. Пример применения функции COUNT (*) представлен на рис. 3.2.

Пример 3.2. Подсчитать количество менеджеров в компании.

```
SELECT COUNT(*) MANAGERS FROM EMP WHERE JOB = 'MANAGER'
```

MANAGERS
3

Рис. 3.2. Результат выполнения запроса примера 3.2

Функция AVG([DISTINCT | ALL] n) – определение среднего значения, без учета NULL – значений. Пример применения функции AVG представлен на рис. 3.3.

Пример 3.3. Определить среднюю величину премии в компании.

```
SELECT AVG(COMM) "СРЕДНЯЯ ПРЕМИЯ"  FROM EMP
```

СРЕДНЯЯ ПРЕМИЯ
550

Рис. 3.3. Результат выполнения запроса примера 3.3

Функция SUM([DISTINCT | ALL] n) – подсчет суммы всех значений группы. Если при этом получаемое значение выходит за пределы суммируемого типа данных, то инициируется ошибка выполнения SQL-оператора. Пример применения функции SUM представлен на рис. 3.4.

Пример 3.4. Вычислить сумму окладов всех сотрудников.

```
SELECT SUM(SAL) "СУММАРНАЯ" FROM EMP
```

СУММАРНАЯ
29025

Рис. 3.4. Результат выполнения запроса примера 3.4

Функция MIN([DISTINCT | ALL] *выражение*) – определение минимального значения из группы.

Функция MAX([DISTINCT | ALL] *выражение*) – определение максимального значения из группы.

Пример применения функций MIN и MAX представлен на рис. 3.5.

Пример 3.5. Вычислить минимальную и максимальную заработную плату в компании.

```
SELECT MAX(SAL), MIN(SAL) FROM EMP
```

MAX(SAL)	MIN(SAL)
5000	800

Рис. 3.5. Результат выполнения запроса примера 3.5

Фраза GROUP BY оператора SELECT применяется для определения группы строк, над которыми выполняются функции агрегирования, и используется для разбиения таблицы на более мелкие группы по значению полей разбиения.

Синтаксис оператора SELECT в этом случае будет следующий:

```
SELECT список_столбцов
FROM таблица
WHERE условие_выбора_строк
GROUP BY столбцы_группировка
HAVING условие_выбора_групп
ORDER BY столбцы (порядок сортировки)
```

Если в операторе SELECT указана фраза GROUP BY, то все имена столбцов, указываемые в списке для определения создаваемого результирующего набора, должны быть указаны с функциями агрегирования, за исключением столбцов, по которым выполняется группировка, поскольку для каждой группы строк в результирующий набор будет включена только одна строка, содержащая значения, полученные функциями агрегирования над данной группой строк.

Пример 3.6. Вычислить среднюю зарплату для каждой должности.

SELECT JOB, AVG (SAL) FROM EMP GROUP BY JOB

Результат выполнения примера 3.6 показан на рис. 3.6.

JOB	AVG(SAL)
ANALYST	3000
CLERK	1037.5
MANAGER	2758.3333
PRESIDENT	5000
SALESMAN	1400

Рис. 3.6. Результат выполнения запроса примера 3.6

Предложение GROUP BY можно использовать для получения результатов по группам внутри групп.

При подстановке столбцов таблицы в предложение SELECT, содержащее групповые функции, необходимо соблюдать следующее правило: в списке выборки предложения SELECT можно ставить только те столбцы, по которым производится разбиение на группы, то есть заданные в предложении GROUP BY.

Фраза HAVING оператора SELECT определяет предикат аналогично фразе WHERE, но применяемый к строкам, полученным в результате выполнения групповых функций.

Пример 3.7. Определить должности, по которым минимальная заработная плата менее 1500.

SELECT JOB, MIN(SAL) FROM EMP GROUP BY JOB HAVING MIN(SAL) <= 1500

Результат выполнения примера 3.7 показан на рис. 3.7.

JOB	MIN(SAL)
SALESMAN	1250
CLERK	800

Рис. 3.7. Результат выполнения запроса примера 3.7

В отличие от предложения HAVING, предложение WHERE не может использоваться для задания ограничений на результаты вычисления по группам.

4. ВЫБОРКА ДАННЫХ ИЗ НЕСКОЛЬКИХ ТАБЛИЦ

4.1. Реляционная алгебра

Реляционная алгебра – это теоретический язык операций, которые на основе одного или нескольких отношений позволяют создавать другое отношение без изменения самих исходных отношений. Это позволяет создавать вложенные выражения реляционной алгебры точно так же, как создаются вложенные арифметические выражения. Реляционная алгебра является языком последовательного использования отношений, в которых все кортежи, возможно взятые даже из разных отношений, обрабатываются одной командой, без организации циклов.

Отношение – это множество кортежей длины k для некоторого фиксированного k, называемого арностью отношения. Иногда представляется удобным присвоить имена компонентам кортежей называемым атрибутами, хотя в других случаях целесообразно считать компоненты неименованными и обращаться к ним по номерам. При определении реляционной алгебры предполагается, что столбцы не обязательно должны быть именованными и порядок в кортежах существен. При обращении с отношениями как с базой данных предполагается, что все они конечны. Требования конечности отношений порождает некоторые трудности в определении реляционной алгебры и исчисления. Например, алгебраическая операция дополнения недопустима, поскольку R – обозначает обычно бесконечное отношение – множество всех кортежей, не принадлежащих R. Какого-либо способа перечислить отношение – R не существует, даже если язык запросов допускает такое выражение.

Операндами реляционной алгебры являются постоянные и (или) переменные отношения, обозначающие отношения фиксированной арности. Для определения реляционной алгебры используются пять основных операций:

1. Объединение. Объединение отношений R и S, обозначаемое как $R \cup S$, представляет собой множество кортежей, которые принадлежат R и S либо им обоим. Оператор объединения применяется только к отношениям одной и той же арности. Поэтому все кортежи в результате имеют одинаковое число компонентов.

2. Разность. Разностью отношений R и S, обозначаемой как $R–S$, называется множество кортежей, принадлежащих R, но не принадлежащих S. Здесь требуется, чтобы R и S имели одну и ту же арность.

3. Декартово произведение. Пусть R и S – отношения арности k_1 и k_2 соответственно. Тогда декартовым произведением $R \times S$ отношений R и S называется множество кортежей длины k_1

+ k_2, первые k_1 компонентов которых образуют кортежи, принадлежащие R, а последние k_2 – кортежи, принадлежащие S.

4. Проекция. Существо этой операции заключается в том, что берется отношение R, удаляются некоторые из его компонентов и (или) переупорядочиваются оставшиеся компоненты. Пусть R – отношение арности k. Обозначим через $\pi_{i1,i2,\ldots im}(R)$, где i_j – являются различными целыми в диапазоне от 1 до k, проекцию R на компоненты i_i, i_2, …, i_m, т. е множество $a_1a_2 \ldots a_m$ таких, что существует некоторый принадлежащий R кортеж $b_1b_2 \ldots b_m$ длины k, удовлетворяющий условию: $a_j=b_{ij}$ для j = 1, 2, …, m. Например, для построения $\pi_{3,1}(R)$ нужно из каждого кортежа t, принадлежащего R, сформировать кортеж длины 2 из третьего и первого его компонентов в указанном порядке. Если у R имеются атрибуты, с помощью которых помечены его столбцы, то можно поставить имена этих атрибутов вместо номеров компонентов и использовать те же имена в отношении, полученном в результате проекции. Например, для отношения R со схемой $R(A, B, C, D)$ проекция $\pi_{C,A}(R)$ представляет собой то же самое, что и $\pi_{3,1}(R)$. В результирующем отношении атрибут C именует его первый столбец, а атрибут A – второй.

5. Селекция. Пусть F – формула, образованная:

а) операндами, являющимися константами или номерами компонентов;

б) арифметическими операторами сравнения

$$<, =, >, \leq, \neq, \geq;$$

в) логическими операторами $\wedge$ (и), $\vee$ (или) и $\neg$ (нет).

В этом случае $\sigma_F(R)$ есть множество кортежей t, принадлежащих R, таких, что при подстановке i-го компонента t вместо любого вхождения номера i в формулу F для всех i она станет истинной. Например, $\sigma_{2>3}(R)$ обозначает множество кортежей, принадлежащих R, второй компонент которых больше третьего компонента. В то же время $\sigma_{1=Smith \vee 1=Jones}(R)$ есть множество кортежей, принадлежащих R, первый компонент которых имеет значение Smith или Jones. Как и в проекции, формула в селекции может ссылаться на столбцы по именам, а не по номерам, если столбцы отношения именованы. Константы в формулах должны быть заключены в кавычки. Это позволяет отличать их от номеров или имен столбцов.

На рис. 4.1 приведены два отношения R и S, на рис. 4.2 – отношения $R \cup S$ и $R - S$ соответственно. Заметим, что объединение

и разность можно выполнять, даже если столбцы двух отношений имеют различные имена, поскольку у этих отношений одно и тоже число компонентов. При этом не существует очевидных имен столбцов результирующего отношения.

A	B	C
a	b	c
d	a	f
c	b	d

Отношение R

D	E	F
b	g	a
d	a	f

Отношение S

Рис. 4.1. Отношения *R* и *S*

a	b	c
d	a	f
c	b	d
b	g	a

$R \cup S$

a	b	c
c	b	d

$R - S$

Рис. 4.2. Операции реляционной алгебры $R \cup S$ и $R - S$

На рис. 4.3 показано $R \times S$. Поскольку множества атрибутов *R* и *S* не пересекаются, имена столбцов можно перенести в $R \times S$. Если же для *R* и *S* имя столбца является общим, например *G*, мы могли бы различать эти два столбца, называя их *R.G* и *S.G*. На рис. 4.4 показана проекция $\pi_{A,C}(R)$ и селекция $\sigma_{B=b}(R)$.

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

Рис. 4.3. Операция реляционной алгебры $R \times S$

A	C
a	c
d	f
c	d

$\pi_{A,C}(R)$

A	B	C
a	b	c
c	b	d

$\sigma_{B=b}(R)$

Рис. 4.4. Операции реляционной алгебры $\pi_{A,C}(R)$ и $\sigma_{B=b}(R)$

4.2. Соединения

Важнейшей особенностью запросов SQL является их способность определять связи между несколькими таблицами и выводить информацию из них в терминах этих связей. Такие операции называются соединением, которое является одним из видов операций в реляционных базах данных. Используя объединения, происходит непосредственное связывание информации с любым номером таблицы, и, таким образом, создаются связи между сравнимыми частями данных.

При многотабличном запросе таблицы, представленные в виде списка в предложении FROM, отделяются друг от друга запятыми. Предикат запроса может ссылаться к любому столбцу любой связанной таблицы и, следовательно, может использоваться для связи между ними. Обычно предикат сравнивает значения в столбцах различных таблиц, чтобы определить, удовлетворяет ли WHERE установленному условию. До этого имена таблиц в запросах опускались, потому данные выбирались из одной таблицы. В одной команде можно задавать различные условия соединения.

Допускается также создавать запросы, объединяющие более двух таблиц. Таким образом, при объединении таблиц, очевидно, хорошо просматривается логика связи между данными и можно больше не ограничиваться просмотром одной таблицы в каждый момент времени. Кроме того, объединение позволяет делать сложные сравнения между любыми полями любого числа таблиц и использовать полученные результаты для того, чтобы решать - какую информацию хотелось бы видеть.

В многотабличном запросе можно объединять данные таблиц, однако интересно и то, что та же самая методика может использоваться для объединения вместе двух копий одиночной таблицы. Для объединения таблицы с собой можно сделать каждую строку таблицы одновременно и комбинацией ее с собой, и комбинацией с каждой другой строкой таблицы, а затем оценить каждую комбинацию в терминах предиката. Это позволяет легко создавать определенные виды связей между различными элементами внутри одиночной таблицы. Например, допускается изобразить объединение таблицы с собой, как объединение двух копий одной и той же таблицы, причем она на самом деле не копируется, но SQL выполняет команду так, как если бы это было сделано.

Использование команды для объединения таблицы с собой аналогично тому приему, который используется для объединения нескольких таблиц. Когда объединяется таблица с собой, все повторяемые имена столбца заполняются префиксами имени табли-

цы. Чтобы ссылаться к этим столбцам внутри запроса, необходимо иметь два различных имени для этой таблицы. Это можно сделать с помощью определения временных имен, называемых псевдонимами, которые определяются в предложении FROM запроса. Синтаксис в этом случае следующий: после имени таблицы оставляют пробел, а затем должен следовать псевдоним для нее.

Применительно к SQL существуют два основных типа соединения – соединение по равенству (эквисоединение) и соединение по неравенству. В условиях соединения по равенству используются операторы проверки на равенство (=). В условиях соединения по неравенству используются операторы сравнения, отличные от проверки на равенство.

Чтобы отличить одноименные поля одной таблицы от полей других, к наименованию поля приписывают через точку имя таблицы (префикс).

Пример 4.1. Вывести имена, должности сотрудников и названия отделов в которых они работают.

SELECT ENAME, JOB, DNAME FROM EMP, DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO;

Результат выполнения примера 4.1 показан на рис. 4.5.

ENAME	JOB	DNAME
KING	PRESIDENT	ACCOUNTING
BLAKE	MANAGER	SALES
CLARK	MANAGER	ACCOUNTING
JONES	MANAGER	RESEARCH
MARTIN	SALESMAN	SALES
ALLEN	SALESMAN	SALES
TURNER	SALESMAN	SALES
JAMES	CLERK	SALES
WARD	SALESMAN	SALES
FORD	ANALYST	RESEARCH
SMITH	CLERK	RESEARCH
SCOTT	ANALYST	RESEARCH
ADAMS	CLERK	RESEARCH
MILLER	CLERK	ACCOUNTING

Рис. 4.5. Результат выполнения запроса примера 4.1

Для сокращения наименований таблиц в предложении **FROM** можно указывать псевдонимы имен таблиц, действующих в

данной команде. В примере 4.2 псевдоним для таблицы EMP определен как E, а для таблицы DEPT как D.

Пример 4.2. Вывести имена и должности сотрудников, а также номера и названия отделов, в которых они работают. Результат отсортировать по номерам отделов.

SELECT D.DEPTNO, ENAME, JOB, DNAME FROM EMP E, DEPT D WHERE D.DEPTNO=E.DEPTNO ORDER BY D.DEPTNO DEPTNO ENAME JOB DNAME;

Результат выполнения примера 4.2 показан на рис. 4.6.

DEPTNO	ENAME	JOB	DNAME
10	KING	PRESIDENT	ACCOUNTING
10	CLARK	MANAGER	ACCOUNTING
10	MILLER	CLERK	ACCOUNTING
20	ADAMS	CLERK	RESEARCH
20	SCOTT	ANALYST	RESEARCH
20	SMITH	CLERK	RESEARCH
20	FORD	ANALYST	RESEARCH
20	JONES	MANAGER	RESEARCH
30	WARD	SALESMAN	SALES
30	JAMES	CLERK	SALES
30	ALLEN	SALESMAN	SALES
30	MARTIN	SALESMAN	SALES
30	BLAKE	MANAGER	SALES
30	TURNER	SALESMAN	SALES

Рис. 4.6. Результат выполнения запроса примера 4.2

Когда условия соединения отсутствуют, то результат соединения – декартово произведение таблиц. Декартово произведение определяет новое отношение, которое является результатом конкатенации каждого кортежа из одного отношения с каждым кортежем из другого отношения. В таком виде полученное отношение содержит больше информации, чем необходимо.

Пример 4.3. Вывести имена и должности сотрудников, а также номера и названия отделов, в которых они работают. Результат отсортировать по номерам отделов.

SELECT D.DEPTNO, ENAME, JOB, DNAME FROM EMP E, DEPT D ORDER BY D.DEPTNO;

Результат выполнения примера 4.3 показан на рис. 4.7.

DEPTNO	ENAME	JOB	DNAME
10	KING	PRESIDENT	ACCOUNTING
10	BLAKE	MANAGER	ACCOUNTING
10	CLARK	MANAGER	ACCOUNTING
10	JONES	MANAGER	ACCOUNTING
10	MARTIN	SALESMAN	ACCOUNTING
10	ALLEN	SALESMAN	ACCOUNTING
10	TURNER	SALESMAN	ACCOUNTING
10	JAMES	CLERK	ACCOUNTING
10	WARD	SALESMAN	ACCOUNTING
10	FORD	ANALYST	ACCOUNTING
10	SMITH	CLERK	ACCOUNTING
10	SCOTT	ANALYST	ACCOUNTING
10	ADAMS	CLERK	ACCOUNTING
10	MILLER	CLERK	ACCOUNTING
20	KING	PRESIDENT	RESEARCH
20	BLAKE	MANAGER	RESEARCH
20	CLARK	MANAGER	RESEARCH
20	JONES	MANAGER	RESEARCH
20	MARTIN	SALESMAN	RESEARCH
20	ALLEN	SALESMAN	RESEARCH
20	TURNER	SALESMAN	RESEARCH
20	JAMES	CLERK	RESEARCH
20	WARD	SALESMAN	RESEARCH
20	FORD	ANALYST	RESEARCH
20	SMITH	CLERK	RESEARCH
20	SCOTT	ANALYST	RESEARCH
20	ADAMS	CLERK	RESEARCH
20	MILLER	CLERK	RESEARCH
30	KING	PRESIDENT	SALES
-	-	-	-
30	MILLER	CLERK	SALES
40	KING	PRESIDENT	OPERATIONS
-	-	-	-
40	MILLER	CLERK	OPERATIONS

56 rows returned

Рис. 4.7. Результат выполнения запроса примера 4.3

Связь между таблицами EMP (служащие) и SALGRADE (категория оплаты) может производиться по неравенству, так как в таблице EMP нет ни одного столбца, соответствующего по значению столбцу таблицы SALGRADE, т.е для соединения необходимо использовать оператор сравнения, отличный от “=”. Можно соединить таблицу саму с собой.

Пример 4.4. Определите к какой категории относится уровень зарплаты каждого сотрудника компании.

```
SELECT E.ENAME "ИМЯ", E.SAL "ОКЛАД", S.GRADE "КАТЕГОРИЯ" FROM EMP E, SALGRADE S  WHERE E.SAL BETWEEN S.LOSAL AND S.HISAL;
```

Результат выполнения примера 4.4 показан на рис. 4.8.

ИМЯ	ОКЛАД	КАТЕГОРИЯ
SMITH	800	1
JAMES	950	1
ADAMS	1100	1
WARD	1250	2
MARTIN	1250	2
MILLER	1300	2
TURNER	1500	3
ALLEN	1600	3
CLARK	2450	4
BLAKE	2850	4
JONES	2975	4
FORD	3000	4
SCOTT	3000	4
KING	5000	5

Рис. 4.8. Результат выполнения запроса примера 4.4

Используя псевдонимы, можно соединить таблицу саму с собой, давая разные псевдонимы различным экземплярам. В приведенном ниже примере два экземпляра таблицы EMP имеют псевдонимы E и M, соответствующие смыслу “Сотрудник” и “Руководитель”.

Пример 4.5 Получить список руководителей и подчиненных им сотрудников.

```
SELECT E.EMPNO, E.ENAME, M.EMPNO MGRNO, M.ENAME MGR_NAME FROM EMP E, EMP M WHERE E.MGR = M.EMPNO;
```

Результат выполнения примера 4.5 показан на рис. 4.9.

EMPNO	ENAME	MGRNO	MGR_NAME
7698	BLAKE	7839	KING
7782	CLARK	7839	KING
7566	JONES	7839	KING
7654	MARTIN	7698	BLAKE
7499	ALLEN	7698	BLAKE
7844	TURNER	7698	BLAKE
7900	JAMES	7698	BLAKE
7521	WARD	7698	BLAKE
7902	FORD	7566	JONES
736	SMITH	7902	FORD
7788	SCOTT	7566	JONES
7876	ADAMS	7788	SCOTT
7934	MILLER	7782	CLARK

Рис. 4.9. Результат выполнения запроса примера 4.5

4.3. Внешние соединения

Если строка в одной из таблиц не удовлетворяет условию соединения, то она не появится в результате запроса, так же, как это происходит и для других условий поиска. Например, в отделе 40 не работает ни одного сотрудника, поэтому в результат соединения таблиц DEPT и EMP по условию EMP.DEPTNO=DEPT.DEPTNO не попадет ни одной строки для отдела 40. Для таких случаев используется внешнее соединение.

Внешнее соединение, задается оператором (+) (символ плюс, заключенный в круглые скобки) и позволяет выбрать строки одной таблицы, для которых в другой таблице нет строк, соответствующих условию соединения.

Оператор (+) помещается на той стороне соединяющего условия, которая соответствует таблице с отсутствующими данными, в которой необходимо выполнить добавление фиктивных строк. Он предписывает в случае отсутствия строк, удовлетворяющих условию соединения, возвращать NULL для всех выражений списка выборки, которая содержит имена столбцов таблицы с отсутствующими данными.

Пример 4.6. Получите список сотрудников компании, номера и названия отделов, в которых они работают.

```
SELECT E.ENAME, D.DEPTNO, D.DNAME FROM EMP E,
DEPT D WHERE E.DEPTNO(+)=D.DEPTNO ORDER BY D.DEPTNO;
```

Результат выполнения примера 4.6 показан на рис. 4.10.

ENAME	DEPTNO	DNAME
KING	10	ACCOUNTING
CLARK	10	ACCOUNTING
MILLER	10	ACCOUNTING
ADAMS	20	RESEARCH
SCOTT	20	RESEARCH
SMITH	20	RESEARCH
FORD	20	RESEARCH
JONES	20	RESEARCH
WARD	30	SALES
JAMES	30	SALES
ALLEN	30	SALES
MARTIN	30	SALES
BLAKE	30	SALES
TURNER	30	SALES
-	40	OPERATIONS

Рис. 4.10. Результат выполнения запроса примера 4.6

В выборке присутствует отдел под номером 40, хотя в нем нет ни одного сотрудника.

Ограничения:

- Нельзя соединить в одном операторе SELECT одну таблицу с двумя другими.
- Условие внешнего соединения не может включать операции IN и OR.

4.4. Операции над множествами

Для реализации операций над множествами объединение, пересечение и разности служат операторы UNION, INTERSECT и MINUS.

Объединение отношений возможно только в том случае, если они совместимы по объединению. Для объединения отношений используется оператор **UNION**. В зависимости от наличия или отсутствия в записи оператора опции ***ALL*** дубликаты либо исключаются, либо допускаются.

Пример 4.7. Выбрать должности, которые есть в отделах 10 и 30. *Дубликаты* ***не исключаются.***

```
SELECT JOB FROM EMP WHERE DEPTNO=10
UNION ALL
SELECT JOB FROM EMP WHERE DEPTNO=30 ORDER BY JOB
```

Результат выполнения примера 4.7 показан на рис. 4.11.

JOB
CLERK
CLERK
MANAGER
MANAGER
PRESIDENT
SALESMAN
SALESMAN
SALESMAN
SALESMAN

Рис. 4.11. Результат выполнения запроса примера 4.7

Пример 4.8. Выбрать должности, которые есть в отделах 10 и 30. *Дубликаты* ***исключаются.***

```
SELECT JOB FROM EMP WHERE DEPTNO=10
UNION
SELECT JOB FROM EMP WHERE DEPTNO=30;
```

Результат выполнения примера 4.8 показан на рис. 4.12.

JOB
CLERK
MANAGER
PRESIDENT
SALESMAN

Рис. 4.12. Результат выполнения запроса примера 4.8

Операция пересечения определяет отношение, которое содержит кортежи, присутствующие в пересекаемых отношениях. Пересекаемые отношения должны быть совместимы по объединению. Для пересечения отношений используется оператор **INTERSECT**. *Дубликаты исключаются.*

Пример 4.9. Выбрать должности, которые имеются как в отделе 10, так и в отделе 30.

```
SELECT JOB FROM EMP WHERE DEPTNO=10
INTERSECT
SELECT JOB FROM EMP WHERE DEPTNO=30;
```

Результат выполнения примера 4.9 показан на рис. 4.13.

JOB
CLERK
MANAGER

Рис. 4.13. Результат выполнения запроса примера 4.9

Разность двух отношений состоит из кортежей, которые имеются в одном отношении, но отсутствуют в другом отношении. Эти отношения должны быть совместимы по объединению. Для операции разности используется оператор **MINUS**. *Дубликаты исключаются.*

Пример 4.10. Выбрать должности сотрудников отдела 10, которых нет в отделе 30.

```
SELECT JOB FROM EMP WHERE DEPTNO=10
MINUS
SELECT JOB FROM EMP WHERE DEPTNO=30;
```

Результат выполнения примера 4.9 показан на рис. 4.14.

JOB
PRESIDENT

Рис. 4.14. Результат выполнения запроса примера 4.10

Если переставить запросы местами, то результат будет другой.

Предложение ORDER BY может находиться только в самом конце запроса, содержащего несколько команд с операторами над множествами и относится только к конечному результату. При записи предложения ORDER BY запрещено использовать имена столбцов, вместо имен задаются порядковые номера столбцов сортировки.

Пример 4.11. Демонстрирует использование предложения ORDER BY.

```
SELECT JOB, SAL, ENAME FROM EMP WHERE DEPTNO=10
UNION SELECT JOB, SAL, ENAME FROM EMP WHERE
DEPTNO=30 ORDER BY 3;
```

Результат выполнения примера 4.11 показан на рис. 4.15.

JOB	SAL	ENAME
SALESMAN	1600	ALLEN
MANAGER	2850	BLAKE
MANAGER	2450	CLARK
CLERK	950	JAMES
PRESIDENT	5000	KING
SALESMAN	1250	MARTIN
CLERK	1300	MILLER
SALESMAN	1500	TURNER
SALESMAN	1250	WARD

Рис. 4.15. Результат выполнения запроса примера 4.11

При использовании операторов над множествами необходимо выполнять следующие правила:

- Команды SELECT должны выбирать одинаковое количество столбцов.
- Соответствующие столбцы должны иметь одинаковый тип и смысл.
- Одинаковые строки автоматически исключаются (кроме UNION ALL).
- В результирующей таблице столбцы именуются по первой SQL команде.
- Предложение ORDER BY может стоять только в конце запроса и относится к окончательному результату запроса.
- Ссылка на столбцы в предложении ORDER BY – только по их порядковым номерам (нумерация с 1).
- Операторы над множествами могут задаваться в подзапросах.
- Очередность выполнения команд SELECT – сверху вниз.
- Для изменения порядка выполнения операторов над множествами используются круглые скобки.

4.5. Подзапросы

Подзапрос представляет собой инструмент создания временной таблицы, содержимое которой извлекается и обрабатывается внешним оператором. Подзапрос – это команда SELECT, помещенная в другую команду SELECT и заключенная в скобки, для получения промежуточных результатов.

Синтаксис подзапроса во фразе WHERE будет следующий:

SELECT список_столбцов

FROM таблица

```
WHERE столбец
(SELECT  список_столбцов
FROM таблица
WHERE условие)
ORDER BY столбцы_сортировки
```

Пример 4.12. Найти служащих, получающих самую маленькую заработную плату в компании.

```
SELECT ENAME AS "ИМЯ", SAL AS "ОКЛАД" FROM EMP
WHERE SAL=(SELECT MIN(SAL) FROM EMP);
```

Результат выполнения примера 4.12 показан на рис. 4.16.

ИМЯ	ОКЛАД
SMITH	800

Рис. 4.16. Результат выполнения запроса примера 4.12

Первой выполняется внутренняя команда SELECT (блок подзапроса), затем выполняется главный блок, использующий полученный в подзапросе результат. Если подзапросов несколько, то сначала выполняются все подзапросы, а затем их результаты подставляются в главный запрос.

Пример 4.13. Выбрать сотрудников отдела продаж (‘SALES’), занимающих ту же должность, что и служащий по фамилии SMITH.

```
SELECT EMPNO, ENAME, JOB, SAL  FROM EMP
WHERE JOB = (SELECT JOB FROM EMP WHERE ENAME =
'SMITH')  AND DEPTNO = (SELECT DEPTNO FROM DEPT WHERE
DNAME ='SALES')
```

Результат выполнения примера 4.13 показан на рис. 4.17.

EMPNO	ENAME	JOB	SAL
7900	JAMES	CLERK	950

Рис. 4.17. Результат выполнения запроса примера 4.13

Пример 4.14. Найти служащих, получающих зарплату минимальную в своем отделе.

```
SELECT DEPTNO, ENAME, SAL FROM EMP
WHERE (SAL, DEPTNO) IN (SELECT MIN (SAL), DEPTNO
FROM EMP GROUP BY DEPTNO);
```

Результат выполнения примера 4.14 показан на рис. 4.18.

DEPTNO	ENAME	SAL
20	SMITH	800
30	JAMES	950
10	MILLER	1300

Рис. 4.18. Результат выполнения запроса примера 4.14

Операторы ANY и ALL могут применяться для задания условий, если подзапрос возвращает множество значений. Они задаются в предложениях WHERE или HAVING вместе с логическими операторами (=, <>, >, <, >=, <=).

ANY (синоним SOME) сравнивает значение левой части оператора сравнения с каждым значением, возвращаемым подзапросом. Результат сравнения есть ИСТИНА, если хотя бы одно значение из выбранных в подзапросе удовлетворяет условию сравнения.

Пример 4.15. Получить список сотрудников отдела 30, получающих зарплату больше минимальной в 20 отделе.

```
SELECT ENAME, SAL, JOB, DEPTNO FROM EMP
WHERE DEPTNO = 30 AND SAL > ANY (SELECT DISTINCT SAL FROM EMP WHERE DEPTNO = 20) ORDER BY SAL, ENAME;
```

Результат выполнения примера 4.15 показан на рис. 4.19.

ENAME	SAL	JOB	DEPTNO
JAMES	950	CLERK	30
MARTIN	1250	SALESMAN	30
WARD	1250	SALESMAN	30
TURNER	1500	SALESMAN	30
ALLEN	1600	SALESMAN	30
BLAKE	2850	MANAGER	30

Рис. 4.19. Результат выполнения запроса примера 4.15

Оператор ALL сравнивает значение со всеми значениями, возвращаемыми подзапросом. Результат сравнения есть ИСТИНА, если все значения, выбранные в подзапросе, удовлетворяют условию сравнения.

Пример 4.16. Получить список сотрудников, получающих зарплату больше любого, работающего в отделе 30.

```
SELECT ENAME, SAL, JOB, DEPTNO FROM EMP
WHERE SAL > ALL (SELECT DISTINCT SAL FROM EMP WHERE DEPTNO = 30) ORDER BY SAL,ENAME;
```

Результат выполнения примера 4.16 показан на рис. 4.20.

ENAME	SAL	JOB	DEPTNO
JONES	2975	MANAGER	20
FORD	3000	ANALYST	20
SCOTT	3000	ANALYST	20
KING	5000	PRESIDENT	10

Рис. 4.20. Результат выполнения запроса примера 4.16

Подзапросы могут использоваться в предложении HAVING. Пример 4.16 демонстрирует использование предложения HAVING.

Пример 4.17. Выбрать те отделы, где число сотрудников больше трех и средняя зарплата больше средней зарплаты отдела 30.

SELECT DEPTNO, AVG (SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT(EMPNO) > 3 AND AVG (SAL) >(SELECT AVG (SAL) FROM EMP WHERE DEPTNO = 30);

Результат выполнения примера 4.17 показан на рис. 4.21.

DEPTNO	AVG(SAL)
20	2175

Рис. 4.21. Результат выполнения запроса примера 4.17

4.6. Коррелированные подзапросы

Коррелированные подзапросы – это вложенные подзапросы, выполняющиеся для каждой «строки–кандидата» из главного запроса, для выполнения которых требуется информация из строк главного запроса.

Последовательность выполнения коррелированного подзапроса:

- внешним запросом выбирается «строка–кандидат»;
- выполняется внутренний запрос, используя полученное значение из «строки–кандидата»;
- результат выполнения внутреннего запроса возвращается во внешний запрос для проверки соответствия «строки–кандидата» заданному критерию;
- процедура повторяется для всех строк из внешнего запроса.

Пример 4.18 демонстрирует использование коррелированного подзапроса.

Пример 4.18. Получить список сотрудников, получающих зарплату больше средней по отделу, в котором они работают.

```
SELECT EMPNO, ENAME, SAL, DEPTNO FROM EMP X
WHERE SAL > (SELECT AVG (SAL) FROM EMP WHERE DEPTNO = X.DEPTNO)
ORDER BY DEPTNO;
```

Результат выполнения примера 4.18 показан на рис. 4.22.

EMPNO	ENAME	SAL	DEPTNO
7839	KING	5000	10
7566	JONES	2975	20
7902	FORD	3000	20
7788	SCOTT	3000	20
7698	BLAKE	2850	30
7499	ALLEN	1600	30

Рис. 4.22. Результат выполнения запроса примера 4.18

Это коррелированный подзапрос, так как имя столбца из внешнего запроса (X. DEPTNO) стоит в предложении WHERE внутреннего запроса. Изначально внешний запрос «не знает», для какого отдела нужно вычислить среднюю зарплату. Это определяется только при выборе очередной «строки–кандидата». То есть внутренний подзапрос в данном примере будет выполнен столько раз, сколько строк в таблице EMP. Поэтому выполнение таких запросов для больших таблиц может занять довольно много времени.

Другая очень важная и удобная форма запроса с коррелированным подзапросом – использование оператора существования EXISTS, который проверяет, найдена ли по подзапросу хотя бы одна строка. При этом от внутреннего запроса не требуется возвращение какого-либо значения, важен сам факт, что-то найдено. Поэтому в списке выборки Вы можете написать, например, 1, 'ПРИВЕТ', *и т.п.

Пример 4.19. Получить список сотрудников, имеющих хотя бы одного подчиненного.

```
SELECT EMPNO, ENAME, JOB, DEPTNO FROM EMP X
WHERE  EXISTS (SELECT 1 FROM EMP WHERE EMP.MGR = X.EMPNO)
ORDER BY DEPTNO;
```

Результат выполнения примера 4.19 показан на рис. 4.23.

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20
7902	FORD	ANALYST	20
7788	SCOTT	ANALYST	20
7698	BLAKE	MANAGER	30

Рис. 4.23. Результат выполнения запроса примера 4.19

Использование коррелированного подзапроса с операторами EXISTS / NOT EXISTS часто оказывается наиболее эффективным путем выполнения некоторых запросов и применяется для переформулирования выражений, в которых есть IN, NOT IN, ANY (SOME) и ALL.

Это определяется использованием индексов, количеством строк, возвращаемых подзапросом, размером таблиц и необходимостью создания временных таблиц во внешней памяти для хранения промежуточных результатов. Так как временные таблицы в racle не имеют индексов, то это может привести к снижению производительности в запросах, использующих операторы IN, ANY, ALL.

Кроме того, использование оператора NOT EXISTS вместо оператора NOT IN может быть более корректным в случае, когда подзапрос находит NULL-значения.

Пример 4.20. Получить список сотрудников, не имеющих подчиненных.

```
SELECT EMPNO,  ENAME, JOB, DEPTNO
FROM EMP
WHERE  EMPNO NOT  IN  (SELECT MGR  FROM  EMP);
```

По данному запросу не будет найдено ни одной строки, так как в столбце MGR подзапросом выбрано в том числе и одно NULL-значение. Запрос с NOT EXISTS, напротив, выберет 8 строк.

Правила задания подзапросов:

1. Внутренний подзапрос берется в круглые скобки и должен стоять в правой части оператора сравнения внешнего запроса.
2. Подзапрос не может содержать предложения ORDER BY.
3. Предложение ORDER BY ставится последним в основном запросе.
4. Имена столбцов в предложении SELECT внутреннего запроса должны стоять в той же последовательности, что и имена столбцов в левой части оператора сравнения внешнего запроса. Типы столбцов также должны попарно соответствовать друг другу.

5. Подзапросы всегда выполняются от внутренних к внешним, если они не коррелируют с друг другом.

6. При задании критериев поиска могут использоваться логические операторы, SQL – операторы, а также операторы ANY (SOME) и ALL.

Подзапросы могут:

• возвращать одну или более строк;

• возвращать один или более столбцов;

• использовать группы или групповые функции;

• задаваться в сложных критериях поиска внешних запросов с использованием предикатов AND и OR;

• соединять таблицы;

• обращаться к таблице, отличной от той, к которой обращается внешний запрос;

• стоять в командах SELECT, UPDATE, DELETE, INSERT, CREATE TABLE;

• коррелировать с внешним запросом;

• использовать SET операторы над множествами. Допускается до 255 уровней вложенности подзапросов.

Использование подзапросов во фразе FROM

Во фразе FROM оператора SELECT могут быть заданы подзапросы. Пример 4.21 демонстрирует использование таких подзапросов.

Пример 4.21

```
SELECT deptno "Отдел",
ROUND(100*by_dept.dept_emp/total.total_emp,0) "% Служащих", ROUND(100*BY_DEPT.DEPT_SAL/TOTAL.TOTAL_SAL,0) "% Зарплаты"
 FROM (SELECT COUNT(empno)as dept_emp, SUM(sal) as dept_sal, deptno FROM emp
 GROUP BY deptno) by_dept, (SELECT COUNT(empno) as total_emp, SUM(sal) as total_sal FROM emp) total
```

Результат представлен на рис. 4.24.

Отдел	% Служащих	% Зарплаты
10	21	30
20	36	37
30	43	32

Рис. 4.24. Результат выполнения запроса примера 4.21

Подзапросам могут быть присвоены псевдонимы, которые указываются следом за скобкой, закрывающей подзапрос. В отличие от псевдонимов столбцов использование AS для подзапроса во фразе FROM недопустимо.

Для закрепления материала, изученного в данном учебном пособии, предложены лабораторные работы (прил. 3).

Библиографический список

1. Кузин А.В., Левонисова С.В. Базы данных: учебное пособие. – М.: Akademia, 2005. – 320 с.
2. Диго С.М. Базы данных: проектирование и использование: учебник. – М.: Финансы и статистика, 2005. – 592 с.
3. Кузнецов С.Д. Основы баз данных: курс лекций. – М.: Интернет – Университет Информационных Технологий, 2005. – 484 с.
4. Дейт К. Дж. Введение в системы баз данных. – 8-е изд.: пер с англ. – М.; СПб.: Изд. дом «Вильямс», 2006. – 1328 с.
5. Данилов О.И. Язык запросов SQL. Самоучитель. – М.: Изд. дом «Питер», 2006. – 227 с.
6. Конолли Т., Бег К. Базы данных. Проектирование, реализация, сопровождение. Теория и практика: пер с англ. – М.; СПб.: Изд. дом «Вильямс», 2003. – 1436 с.
7. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Изд. дом «Бином», 2002. – 348 с.
8. Ульман Дж. Основы систем баз данных: пер. с англ. – М.: Финансы и статистика, 1983. – 334 с.
9. Полякова Л.Н. Основы SQL: учебное пособие. – М.: Интернет – Университет Информационных Технологий, 2007. – 223 с.
10. Смирнов С.Н., Задворьев И.С. Работаем с Oracle: учебное пособие. – М.: Гелиос АРВ, 2002. – 496 с.

Приложение 1

Создание таблиц для учебного примера

```
CREATE TABLE DEPT (
DEPTNO      NUMBER(2) NOT NULL,
DNAME       CHAR(14),
LOC         CHAR(13),
CONSTRAINT DEPT_PRIMARY_KEY PRIMARY KEY (DEPTNO));


CREATE TABLE EMP (
EMPNO  NUMBER(4) NOT NULL,
ENAME  CHAR(10),
JOB CHAR(9),
MGR  NUMBER(4) CONSTRAINT EMP_SELF_KEY REFERENCES
EMP  (EMPNO),
HIREDATE  DATE,
SAL  NUMBER(7,2),
COMM  NUMBER(7,2),
DEPTNO  NUMBER(2) NOT NULL,
CONSTRAINT EMP_FOREIGN_KEY FOREIGN KEY (DEPTNO)
REFERENCES DEPT (DEPTNO),
CONSTRAINT EMP_PRIMARY_KEY PRIMARY KEY (EMPNO));

CREATE TABLE SALGRADE (
GRADE  NUMBER,
LOSAL NUMBER,
HISAL NUMBER);
```

Приложение 2

Числовые функции

Функция EXP(*числовой_аргумент*) возвращает число e (основание натуральных логарифмов) в степени параметра *числовой_аргумент*. Пример применения функции EXP представлен на рис. П2.1.

Пример 1

SELECT EXP (4) as “e в 4-й степени” FROM DUAL

e в 4-й степени
54,59815

Рис. П2.1. Результат выполнения запроса примера 1

Функция LN(*числовой_аргумент*) возвращает натуральный логарифм положительного параметра *числовой_аргумент*. Пример применения функции LN представлен на рис. П2.2.

Пример 2

SELECT LN (95) as “Натуральный логарифм” FROM DUAL

Натуральный логарифм
4,55387689

Рис. П2.2. Результат выполнения запроса примера 2

Функция LOG (*основание, числовой аргумент*) возвращает логарифм по основанию, заданному параметром *основание*, параметра *числовой_аргумент*. Параметр *основание* может быть любым положительным числом, за исключением 1, а параметр *числовой_аргумент* должен быть положительным числом. Пример применения функции LOG представлен на рис. П2.3.

Пример 3

SELECT LOG (10,100) as “ЛОГАРИФМ 100 ПО ОСНОВАНИЮ 10” FROM DUAL

ЛОГАРИФМ 100 ПО ОСНОВАНИЮ 10
2

Рис. П2.3. Результат выполнения запроса примера 3

Функция POWER (*основание, числовой_аргумент*) возвращает значение параметра *основание* в степени параметра *числовой_аргумент*. Если параметр *основание* отрицательный, то параметр *числовой_аргумент* должен быть целым. Пример применения функции POWER представлен на рис. П2.4.

Пример 4

SELECT POWER (3,2) as “3 в квадрате” FROM DUAL

3 в квадрате
9

Рис. П2.4. Результат выполнения запроса примера 4

Функция SQRT (*числовой аргумент*) возвращает значение квадратного корня параметра *числовой аргумент*. Если параметр *числовой аргумент* отрицательный, то возвращается сообщение об ошибке. Пример применения функции SQRT представлен на рис. П2.5.

Пример 5

SELECT SQRT (84) as “Квадратный корень из 84” FROM DUAL

Квадратный корень из 84
9,1651514

Рис. П2.5. Результат выполнения запроса примера 5

Функции SIN(*числовой_аргумент*), COS(*числовой_аргумент*), TAN(числовой_аргумент) возвращают соответственно синус, косинус и тангенс параметра *числовой_аргумент*. Параметр *числовой_аргумент* предполагается заданным в радианах. Пример применения функций SIN и COS представлен на рис. П2.6, П2.7.

Пример 6

SELECT SIN (30*3.14159265359/180) as "СИНУС 30 ГРАДУСОВ" FROM DUAL

СИНУС 30 ГРАДУСОВ
,500000000000029843457312725584897995956

Рис. П2.6. Результат выполнения запроса примера 6

Пример 7

SELECT COS (180*3,14159265359/180) as “КОСИНУС 30 ГРАДУСОВ” FROM DUAL

КОСИНУС 180 ГРАД.
-,99999999999999999999999997862483333497

Рис. П2.7. Результат выполнения запроса примера 7

Функции ASIN(*числовой_аргумент*), ACOS(*числовой_аргумент*), возвращают соответственно арксинус и арккосинус параметра *числовой_аргумент*. Параметр *числовой_аргумент* предполагается находящимся в диапазоне от -1 до 1. Возвращаемое значение после применения функции ACOS находится в диапазоне от 0 до π. Возвращаемое значение после применения функции ASIN находится в диапазоне от –π/2 до π/2. При вычислении функции с параметром *числовой_аргумент* вне указанного диапазона выдается сообщение об ошибке. Функция ATAN(*числовой_аргумент*) возвращает соответственно арктангенс параметра *числовой_аргумент*. Возвращаемое значение после применения функции ATAN находится в диапазоне от –π/2 до π/2. Примеры применения функций ASIN, ACOS, ATAN представлены на рис. П2.8–П2.10.

Пример 8

SELECT ASIN (.3) as “АРКСИНУС” FROM DUAL

АРКСИНУС
,304692654

Рис. П2.8. Результат выполнения запроса примера 8

Пример 9

SELECT ACOS (.3) as "АРККОСИНУС" FROM DUAL

АРККОСИНУС
1,26610367277

Рис. П2.9. Результат выполнения запроса примера 9

Пример 10

SELECT ATAN (.3) as “АРКТАНГЕНС” FROM DUAL

АРКТАНГЕНС
,2914567944

Рис. П2.10. Результат выполнения запроса примера 10

Функции SINH(*числовой_аргумент*), COSH(*числовой_аргумент*), TANH(*числовой_аргумент*) возвращают соответственно гиперболический синус, гиперболический косинус и гиперболический тангенс параметра *числовой_аргумент*. Примеры применения функций SINH, COSH, TANH представлены на рис. П2.11–П2.13.

Пример 11

SELECT SINH (1) as “ГИПЕРБОЛ. СИНУС 1” FROM DUAL

ГИПЕРБОЛ. СИНУС 1
1,175201193643

Рис. П2.11. Результат выполнения запроса примера 11

Пример 12

SELECT COSH (0) as "ГИПЕРБ. КОСИНУС 0" FROM DUAL

ГИПЕРБ. КОСИНУС 0
1

Рис. П2.12. Результат выполнения запроса примера 12

Пример 13

SELECT TANH (0.57) as “ГИПЕР. ТАНГЕНС 0.57” FROM DUAL

ГИПЕР.ТАНГЕНС 0.57
,515359278007

Рис. П2.13. Результат выполнения запроса примера 13

Функция ROUND (*числовой_аргумент*[, *позиция*]) округляет значение параметра *числовой_аргумент* с точностью, определяемой параметром *позиция*. Параметр *позиция* определяет число десятичных знаков после запятой. Если параметр *позиция* отрицательный, то аргумент округляется до целых чисел соответствующего масштаба (для значения параметра –1 до десятков, – 2 – до сотен и т.д.) Значение параметра *позиция* по умолчанию – 0. Пример применения функции ROUND представлен на рис. П2.14.

Пример 14

SELECT ROUND (456.789,1) AS "ДО 1 ЗНАКА", ROUND (456.789,-2) AS "ДО СОТЕН" FROM DUAL;

ДО 1 ЗНАКА	ДО СОТЕН
456,8	500

Рис. П2.14. Результат выполнения запроса примера 14

Функция TRUNC (*числовой_аргумент*[, *позиция*]) отсекает (отбрасывает без округления) значение параметра *числовой_аргумент* с точностью, определяемой параметром *позиция.* Параметр *позиция* определяет число десятичных знаков после запятой. Если параметр *позиция* отрицательный, то аргумент округляется до целых чисел соответствующего масштаба (для значения

параметра -1 до десятков, от -2 до сотен и т.д.). Значение параметра *позиция* по умолчанию – 0. Пример применения функции TRUNC представлен на рис. П2.15.

Пример 15

SELECT TRUNC (456.789,1) AS "ДО 1 ЗНАКА", TRUNC (456.789,-2) AS "ДО СОТЕН" FROM DUAL;

ДО 1 ЗНАКА	ДО СОТЕН
456,7	400

Рис. П2.15. Результат выполнения запроса примера 15

Функция FLOOR(*числовой_аргумент*) возвращает наибольшее целое, меньшее или равное значению параметра *числовой_аргумент*. Пример применения функции FLOOR представлен на рис. П2.16.

Пример 16

SELECT FLOOR (15.7) AS “FLOOR” FROM DUAL;

FLOOR
15

Рис. П2.16. Результат выполнения запроса примера 16

Функция CEIL(*числовой_аргумент*) возвращает наименьшее целое, большее или равное значению параметра *числовой_аргумент.* Пример применения функции CEIL представлен на рис. П2.17.

Пример 17

SELECT CEIL (15.7) AS “ Целое” FROM DUAL;

Целое
16

Рис. П2.17. Результат выполнения запроса примера 17

Функции FLOOR и CEIL аналогичны функциям ROUND и TRUNC.

Функция ABS(*числовой_аргумент*) – модуль параметра *числовой_аргумент.* Пример применения функции ABS представлен на рис. П2.18.

Пример 18

SELECT ABS (-15) AS "Абсолютное значение." FROM DUAL;

Абсолютное значение
15

Рис. П2.18. Результат выполнения запроса примера 18

Функция SIGN(*числовой_аргумент*) возвращает -1, если параметр *числовой_аргумент* < 0, возвращает 0, если параметр *числовой_аргумент* = 0, и возвращает 1, если параметр *числовой_аргумент* > 0. Пример применения функции SIGN представлен на рис. П2.19.

Пример 19

SELECT SIGN (-20) AS "Отрицательное число", IGN (20) AS "Положительное число" FROM DUAL;

Отрицательное число	Положительное число
-1	1

Рис. П2.19. Результат выполнения запроса примера 19

Функция MOD(*числовой_аргумент, основание*) возвращает остаток от деления параметра *числовой_аргумент* на значение, определяемое параметром *основание*. Использование отрицательных значений параметра основание не рекомендуется, поскольку результат не соответствует принятому определению модуля числа. Пример применения функции MOD представлен на рис. П2.20.

Пример 20

SELECT MOD(11,4) AS "Остаток" FROM DUAL;

Остаток
3

Рис. П2.20. Результат выполнения запроса примера 20

Символьные функции

Функция LOWER(*строка*) преобразует каждую букву параметра *строка* в строчную. Пример применения функции представлен на рис. П2.21.

Пример 21. Выбрать всех служащих с наименованием их должностей. Результат вывести в виде одного столбца. Должность выводить строчными буквами в круглых скобках.

SELECT ENAME||'('|| LOWER (JOB) ||')' Employee FROM EMP;

EMPLOYEE
KING(president)
BLAKE(manager)
CLARK(manager)
JONES(manager)
MARTIN(salesman)
ALLEN(salesman)
TURNER(salesman)
JAMES(clerk)
WARD(salesman)
FORD(analyst)
SMITH(clerk)
SCOTT(analyst)
ADAMS(clerk)
MILLER(clerk)

Рис. П2.21. Результат выполнения запроса примера 21

Функция UPPER(*строка*) преобразует каждую букву параметра *строка* в прописную. Пример применения функции представлен на рис. П2.22.

Пример 22. Выбрать всех служащих с наименованием их должностей. Результат вывести в виде одного столбца. Должность выводить прописными буквами в круглых скобках.

```
SELECT ENAME||'('|| UPPER (JOB) ||')' Employee FROM EMP;
```

EMPLOYEE
KING(PRESIDENT)
BLAKE(MANAGER)
CLARK(MANAGER)
JONES(MANAGER)
MARTIN(SALESMAN)
ALLEN(SALESMAN)
TURNER(SALESMAN)
JAMES(CLERK)
WARD(SALESMAN)
FORD(ANALYST)
SMITH(CLERK)
SCOTT(ANALYST)
ADAMS(CLERK)

Рис. П2.22. Результат выполнения запроса примера 22

Функция INITCAP(*строка*) преобразует каждую первую букву слов параметра *строка* в прописную, а все последующие в строчные. Пример применения функции представлен на рис. П2.23.

Пример 23. Выбрать всех служащих с наименованием их должностей. Результат вывести в виде одного столбца. Должность выводить с прописной первой буквой в круглых скобках.

```
SELECT ENAME||'('|| INITCAP(JOB) ||')' Employee FROM EMP;
```

EMPLOYEE
KING(President)
BLAKE(Manager)
CLARK(Manager)
JONES(Manager)
MARTIN(Salesman)
ALLEN(Salesman)
TURNER(Salesman)
JAMES(Clerk)
WARD(Salesman)
FORD(Analyst)
SMITH(Clerk)
SCOTT(Analyst)
ADAMS(Clerk)
MILLER(Clerk)

Рис. П2.23. Результат выполнения запроса примера 23

Функция RPAD(*строка_1,число_символов* [, *символы_наполнители*]) возвращает значение параметра *строка_1,* дополненное справа до числа символов, которое задано параметром *число_символов*, символом-наполнителем, заданным параметром *символы_наполнители*. По умолчанию символом-наполнителем является пробел.

Функция LPAD(*строка_1,число_символов* [, *символы_наполнители*]) возвращает значение параметра *строка_1,* дополненное слева до числа символов, которое задано параметром *число_символов*, символом-наполнителем, заданным параметром *символы_наполнители*. По умолчанию символом-наполнителем является пробел. Примеры применения функций LPAD и RPAD представлены на рис. П2.24, П2.25.

Пример 24. Вывести имена служащих, работающих в компании. Дополнить результат справа до десяти символов. В качестве символа-наполнителя использовать точку.

```
SELECT RPAD(ENAME,10,'.') "Имя" FROM EMP;
```

Имя
KING.....
BLAKE.....
CLARK.....
JONES.....
MARTIN....
ALLEN.....
TURNER....
JAMES.....
WARD.....
FORD.....
SMITH.....
SCOTT.....
ADAMS.....
MILLER....

Рис. П2.24. Результат выполнения запроса примера 24

Пример 25. Вывести имена служащих, работающих в компании. Дополнить результат слева до десяти символов. В качестве символа-наполнителя использовать точку.

```
SELECT LPAD(JOB,10,'.') "Должность" FROM EMP;
```

Должность
.PRESIDENT
...MANAGER
...MANAGER
...MANAGER
..SALESMAN
..SALESMAN
..SALESMAN
.....CLERK
..SALESMAN
...ANALYST
.....CLERK
...ANALYST
.....CLERK
.....CLERK

Рис. П2.25. Результат выполнения запроса примера 25

Функция LTRIM (*строка_1*,[,*строка_шаблон*]) возвращает усеченное слева значение параметра *строка_1*. Из строки параметра *строка_1* символы удаляются слева до тех пор, пока удаляемый символ входит во множество символов параметра *строка_шаблон*. По умолчанию *строка_шаблон* состоит из символа пробела. Пример применения функции LTRIM представлен на рис. П2.26.

Пример 26. Вывести слово job используя функцию LTRIM.
SELECT LTRIM('job', 'j')"Должность" FROM dual;

Должность
ob

Рис. П2.26. Результат выполнения запроса примера 26

Функция RTRIM (*строка_1*,[,*строка_шаблон*]) возвращает усеченное справа значение параметра *строка_1*. Из строки параметра *строка_1* символы удаляются справа до тех пор, пока удаляемый символ входит во множество символов параметра *строка_шаблон*. По умолчанию *строка_шаблон* состоит из символа пробела. Пример применения функции RTRIM представлен на рис. П.2.27.

Пример 27. Вывести слово job, используя функцию RTRIM.
SELECT RTRIM('job', 'b')"Должность" FROM dual;

Должность
jo

Рис. П2.27. Результат выполнения запроса примера 27

Функция TRANSLATE *(строка_1, символы_поиска, символы_замены*) возвращает значение параметра *строка_1*, для которой выполнено следующее преобразование. Все вхождения параметра *символ_поиска* замещены значением параметра *символ_замены*. Если в строке *символы_поиска* содержится больше символов, чем в строке *символы_замены*, то символы, которым нет соответствия, замещаются на пустой символ (то есть исключаются из результирующей строки). Параметр *символы_замены* не может быть пустым. Функция TRANSLATE может применяться, в частности, для обработки текстов, подготовленных с использованием различных раскладок клавиатур. Пример применения функции TRANSLATE представлен на рис. П2.28.

Пример 28. Вывести названия отделов. В названии отделов заменить символ “S” на символ “2”.

SELECT DNAME, TRANSLATE(DNAME,'S','2') DNAM FROM DEPT;

DNAME	DNAM
ACCOUNTING	ACCOUNTING
RESEARCH	RE2EARCH
SALES	2ALE2
OPERATIONS	OPERATION2

Рис. П2.28. Результат выполнения запроса примера 28

Функция REPLACE (*строка_1, строка_поиска[,строка_замещения*]) возвращает значение параметра *строка_1,* для которой выполнено следующее преобразование. Все вхождения параметра *строка_поиска* замещены значением параметра *строка_замещения*. Если параметр *строка_замещения* не задан, то все вхождения параметра *строка_поиска* удаляются. Пример применения функции REPLACE представлен на рис. П2.29.

Пример 29. Вывести имена служащих, работающих в 30 отделе и их должность. При выводе должности заменить должность SALESMAN на SALESPERSON. Должность выводить с прописной первой буквой.

SELECT ENAME, DEPTNO, INITCAP (REPLACE (JOB, 'SALESMAN', 'SALESPERSON')) "Должность" FROM EMP WHERE DEPTNO = 30;

ENAME	DEPTNO	Должность
BLAKE	30	Manager
MARTIN	30	Salesperson
ALLEN	30	Salesperson
TURNER	30	Salesperson
JAMES	30	Clerk
WARD	30	Salesperson

Рис. П2.29. Результат выполнения запроса примера 29

Функция SUBSTR (*строка_1*, *позиция* [,*длина_подстроки*]) возвращает подстроку параметра *строка_1*, начиная с позиции, заданной параметром *позиция*, и длиной, заданной параметром *длина_подстроки*. Если параметр *длина_подстроки* не задан, то

возвращается подстрока до конца строки, заданной параметром *строка_1*. Пример применения функции SUBSTR представлен на рис. П2.30.

Пример 30

```
SELECT SUBSTR ('ABCDEFG', 3.1,4) AS "+ПОЗИЦИЯ",
       SUBSTR ('ABCDEFG', -5, 4) AS "-ПОЗИЦИЯ",
FROM DUAL;
```

+ПОЗИЦИЯ	-ПОЗИЦИЯ
CDEF	CDEF

Рис. П2.30. Результат выполнения запроса примера 30

Функция INSTR (*строка_1, строка_поиска* [,*позиция_начала_поиска* [, *число_вхождений*]]) возвращает позицию вхождения строки, задаваемой параметром *строка_поиска* в строку, задаваемую параметром строка_1. Позиция начала поиска задается необязательным числовым параметром *позиция_начала_поиска*, а необязательный параметр *число_вхождений* задает требуемое число вхождений строки поиска в основную строку. Значения по умолчанию для необязательных параметров – 1. Если нет вхождения строки поиска в основную строку, функция возвращает значение 0. Пример применения функции INSTR представлен на рис.П2.31.

Пример 31. Вывести позицию буквы R в слове CLERK.

```
SELECT INSTR('CLERK','R') AS "С начала" FROM DUAL;
```

С начала
4

Рис.П2.31. Результат выполнения запроса примера 31

Функция LENGTH (строка) возвращает длину строки, заданной в качестве аргумента. Пример применения функции LENGTH представлен на рис. П2.32.

Пример 32. Вывести названия отделов и количество символов в названиях отделов, подсчитанное с помощью функции LENGTH.

```
SELECT DNAME, LENGTH(DNAME) L FROM DEPT;
```

DNAME	L
ACCOUNTING	10
RESEARCH	8
SALES	5
OPERATIONS	10

Рис. П2.32. Результат выполнения запроса примера 32

Функции работы с календарными датами

Функция SYSDATE является системной переменной, которая возвращает дату и время, определяемые средствами операционной системы сервера базы данных. Пример применения функции SYSDATE представлен на рис. П2.33.

Пример 33. Вывести текущую дату.

```
SELECT SYSDATE "СЕГОДНЯ" FROM DUAL;
```

СЕГОДНЯ
10.04.08

Рис. П2.33. Результат выполнения запроса примера 33

Функция ROUND (*дата*[*, шаблон*]) округляет значение параметра *дата* по шаблону, определяемому параметром шаблон. Если параметр *шаблон* опущен, то аргумент *дата* округляется до дней (время устанавливается за полночь). Пример применения функции ROUND представлен на рис. П2.34.

Пример 34. Вывести текущую дату и текущую дату, округленную с помощью функции ROUND.

```
SELECT SYSDATE, ROUND (SYSDATE) "ROUND" FROM DUAL;
```

SYSDATE	ROUND
30.05.08	31.05.08

Рис. П2.34. Результат выполнения запроса примера 34

Функция TRUNC (*дата*[*, формат*]) усекает значение параметра *дата* по шаблону, определяемому параметром *формат*. Если параметр *формат* опущен, то аргумент *дата* усекается до ближайшего дня (время устанавливается за полночь). Пример применения функции TRUNC представлен на рис. П2.35.

Пример 35. Вывести текущую дату с использованием функции TRUNC. Дату вывести в формате ‘10-04-2008’, а время с точностью до часов и минут.

SELECT TO_CHAR (TRUNC(SYSDATE, 'HH'),'DD-MM-YYYY HH24:MI')"СЕЙЧАС" FROM DUAL;

СЕЙЧАС
10-04-2008 12:00

Рис. П2.35. Результат выполнения запроса примера 35

Функция NEXT _DAY (*дата, название_дня*) возвращает дату дня, который является первым днем, более поздним, чем текущая дата с названием, совпадающим с указанным параметром *название_дня* или его номером. Пример применения функции NEXT _DAY представлен на рис. П2.36.

Пример 36. Вывести текущую дату и дату следующего дня с названием – пятница.

SELECT SYSDATE "СЕГОДНЯ", NEXT_DAY (SYSDATE, 'ПЯТНИЦА') "ПЯТНИЦА" FROM DUAL;

СЕГОДНЯ	ПЯТНИЦА
10.04.08	11.04.08

Рис. П2.36. Результат выполнения запроса примера 36

Функция LAST_ DAY(*дата, название_дня*) возвращает дату дня, который является последним днем месяца, более поздним, чем текущая дата с названием, совпадающим с указанным параметром *название_дня*. Пример применения функции LAST_ DAY представлен на рис. П2.37.

Пример 37. Вывести текущую дату и дату последнего дня мясяца.

SELECT SYSDATE "СЕГОДНЯ", LAST_DAY (SYSDATE) "ПОСЛЕДНИЙ ДЕНЬ МЕСЯЦА" FROM DUAL;

СЕГОДНЯ	ПОСЛЕДНИЙ ДЕНЬ МЕСЯЦА
10.04.08	30.04.08

Рис. П2.37. Результат выполнения запроса примера 37

Функция ADD_MONTHS (*дата, количество_месяцев*) возвращает дату плюс *количество_месяцев*. Пример применения функции ADD_MONTHS представлен на рис. П2.38.

Пример 38. Вывести текущую дату и дату через месяц.

SELECT SYSDATE "СЕГОДНЯ", ADD_MONTHS(SYSDATE, 1) "ДАТА ЧЕРЕЗ МЕСЯЦ" FROM DUAL;

СЕГОДНЯ	ДАТА ЧЕРЕЗ МЕСЯЦ
10.04.08	10.05.08

Рис. П2.38. Результат выполнения запроса примера 38

Функция MONTHS_BETWEEN (*дата_1*, *дата_2*) возвращает количество месяцев *между дата_1 и дата_2.* Пример применения функции MONTHS_BETWEEN представлен на рис. П2.39.

Пример 39. Вывести текущую дату, дату '10.01.2008' и количество месяцев между этими датами.

SELECT SYSDATE "СЕГОДНЯ", '10.01.2008', MONTHS_BETWEEN (SYSDATE,'10.01.2008')"КОЛ-ВО МЕСЯЦЕВ" FROM DUAL;

СЕГОДНЯ	'10.01.2008'	КОЛ-ВО МЕСЯЦЕВ
10.04.08	10.01.2008	3

Рис. П2.39. Результат выполнения запроса примера 39

Функции преобразования типов данных

Функция TO_CHAR (*аргумент*[, *формат*]) возвращает результат преобразования значения параметра *аргумент* типа NUMBER или DATE в символьную строку. Для чисел если параметр *формат* опущен, *аргумент* преобразовывается в строку с длиной, достаточной для хранения всех значащих цифр. Пример применения функции TO_CHAR представлен на рис. П2.40.

Пример 40. Вывести имена сотрудников из 20 отдела и название месяца в котором они поступили на работу.

SELECT ENAME, TO_CHAR(HIREDATE, 'Month')DATE_HIRED FROM EMP WHERE DEPTNO = 20;

ENAME	DATE_HIRED
JONES	Апрель
FORD	Декабрь
SMITH	Декабрь
SCOTT	Декабрь
ADAMS	Декабрь

Рис. П2.40. Результат выполнения запроса примера 40

Функция TO_DATE (*символьный_аргумент*[, *формат*]) возвращает результат преобразования значения параметра *символьный_аргумент* символьного типа в тип DATE. Если параметр формат опущен, символьный аргумент должен соответствовать формату даты, принятому в системе по умолчанию. Пример применения функции TO_ DATE представлен на рис. П2.41.

Пример 41. Преобразовать дату в формате '01.04.2008' с применением функции TO_DATE.

```
SELECT TO_DATE ('01.04.2008') "ФУНКЦИЯ TO_DATE"
FROM DUAL;
```

ФУНКЦИЯ TO_DATE
01.04.08

Рис. П2.41. Результат выполнения запроса примера 41

Функция TO_NUMBER (*символьный_аргумент*) возвращает результат преобразования значения параметра *символьный_аргумент* символьного типа в значение типа NUMBER. Параметр *символьный_аргумент* может представлять числа любой допустимой Oracle нотации. Пример применения функции TO_ NUMBER представлен на рис. П2.42.

Пример 42. Получить результат арифметической операции умножения двух чисел, одно из которых задано как символ (‘30’).

```
SELECT TO_NUMBER ('30')*5 "ФУНКЦИЯ TO_NUMBER"
FROM DUAL;
```

ФУНКЦИЯ TO_NUMBER
150

Рис. П2.43. Результат выполнения запроса примера 42

Функция NVL (*аргумент_1, аргумент_2*) возвращает *аргумент_2*, если *аргумент_1* имеет неопределенное значение (NULL), в противном случае возвращает *аргумент_1*. Тип данных возвращаемого значения определяется типом данных параметра *аргумент_1*.

Функция DECODE (*выражение*, *аргумент_1*, *результат_1* [*аргумент_2, результат_2,* …][*значение_по_умолчанию*]) возвращает значение параметра *результат_x*, если параметр выражение совпадает с параметром *аргумент_x*, где x принимает значение 1,2, … . При обращении к функции DECODE необходимо

задать как минимум 4 аргумента. Если совпадения ни с одним параметром *аргумент_x* не обнаружено, то возвращается параметр *значение_по_умолчанию*. Если *значение_по_умолчанию* не задано, то возвращается неопределенное значение (NULL). Наряду с декодированием функцию DECODE используют для написания различных логических запросов, например, в зависимости от значений полей в записи одной таблицы можно выбирать, по какому правилу соединять эту запись с записями другой таблицы. Пример применения функции DECODE представлен на рис. П2.43.

Пример 43. Вывести имена служащих, их должности и кодировку должностей (если должность служащего 'CLERK' то выводить 'WORKER'; если должность 'MANAGER', то выводить 'BOSS'; во всех остальных случаях 'UNDEF').

```
SELECT ENAME, JOB, DECODE (JOB, 'CLERK', 'WORKER',
'MANAGER', 'BOSS', 'UNDEF') "DECODE" FROM EMP
```

ENAME	JOB	DECODE
KING	PRESIDENT	UNDEF
BLAKE	MANAGER	BOSS
CLARK	MANAGER	BOSS
JONES	MANAGER	BOSS
MARTIN	SALESMAN	UNDEF
ALLEN	SALESMAN	UNDEF
TURNER	SALESMAN	UNDEF
JAMES	CLERK	WORKER
WARD	SALESMAN	UNDEF
FORD	ANALYST	UNDEF
SMITH	CLERK	WORKER
SCOTT	ANALYST	UNDEF
ADAMS	CLERK	WORKER
MILLER	CLERK	WORKER

Рис. П2.43. Результат выполнения запроса примера 43

Функция GREATEST (*аргумент_1, аргумент_2*, ...) возвращает наибольшее значение из списка параметров *аргумент_x.* При этом используются обычные правила сравнения для различных типов. Пример применения функции GREATEST представлен на рис. П2.44.

Пример 44. Вывести имена служащих, их месячный доход и годовую премию. В последнем столбце с помощью функции GREATEST вывести наибольшее значение месячного дохода или премии.

SELECT ENAME, SAL, COMM, GREATEST(SAL, COMM) FROM EMP WHERE COMM IS NOT NULL

ENAME	SAL	COMM	GREATEST(SAL,COMM)
MARTIN	1250	1400	1400
ALLEN	1600	300	1600
TURNER	1500	0	1500
WARD	1250	500	1250

Рис. П2.44. Результат выполнения запроса примера 44

Функция LEAST (*аргумент_1, аргумент_2*, ...) возвращает наименьшее значение из списка параметров *аргумент_x.* Пример применения функции LEAST представлен на рис. П2.45.

Пример 45. Вывести имена служащих, их месячный доход и годовую премию. В последнем столбце с помощью функции LEAST вывести наименьшее значение месячного дохода или премии.

SELECT ENAME, SAL, COMM, LEAST(SAL, COMM) FROM EMP WHERE COMM IS NOT NULL

ENAME	SAL	COMM	LEAST(SAL,COMM)
MARTIN	1250	1400	1250
ALLEN	1600	300	300
TURNER	1500	0	0
WARD	1250	500	500

Рис. П2.45. Результат выполнения запроса примера 45

Приложение 3

Лабораторная работа №1
ВВЕДЕНИЕ В SQL

1. Выберите всю информацию из таблицы SALGRADE.

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

2. Выберите всю информацию из таблицы EMP.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	17.11.81	5000	-	10
7698	BLAKE	MANAGER	7839	01.05.81	2850	-	30
7782	CLARK	MANAGER	7839	09.06.81	2450	-	10
7566	JONES	MANAGER	7839	02.04.81	2975	-	20
7654	MARTIN	SALESMAN	7698	28.08.81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20.02.81	1600	300	30
7844	TURNER	SALESMAN	7698	08.09.81	1500	0	30
7900	JAMES	CLERK	7698	03.12.81	950	-	30
7521	WARD	SALESMAN	7698	22.02.81	1250	500	30
7902	FORD	ANALYST	7566	03.12.81	3000	-	20
736	SMITH	CLERK	7902	17.12.80	800	-	20
7788	SCOTT	ANALYST	7566	09.12.82	3000	-	20
7876	ADAMS	CLERK	7788	12.12.83	1100	-	20
7934	MILLER	CLERK	7782	23.12.82	1300	-	10

3. Найдите всех служащих с зарплатой в диапазоне от $1000 до $2000.

ENAME	DEPTNO	SAL
MARTIN	30	1250
ALLEN	30	1600
TURNER	30	1500
WARD	30	1250
ADAMS	20	1100
MILLER	10	1300

4. Покажите на экране номера отделов и их наименование в алфавитном порядке.

DEPTNO	DNAME
10	ACCOUNTING
40	OPERATIONS
20	RESEARCH
30	SALES

5. Покажите все категории должностей из таблицы EMP.

JOB
SALESMAN
CLERK
PRESIDENT
MANAGER
ANALYST

6. Выберите все данные о сотрудниках 10 и 20 отделов в алфавитном порядке по их именам.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	12.12.83	1100	-	20
7782	CLARK	MANAGER	7839	09.06.81	2450	-	10
7902	FORD	ANALYST	7566	03.12.81	3000	-	20
7566	JONES	MANAGER	7839	02.04.81	2975	-	20
7839	KING	PRESIDENT	-	17.11.81	5000	-	10
7934	MILLER	CLERK	7782	23.12.82	1300	-	10
7788	SCOTT	ANALYST	7566	09.12.82	3000	-	20
736	SMITH	CLERK	7902	17.12.80	800	-	20

7. Найдите всех служащих, имена которых содержат комбинации символов TH или LL.

ENAME
ALLEN
SMITH
MILLER

8. Покажите значения полей ENAME и JOB для всех клерков в отделе 20.

ENAME	JOB
SMITH	CLERK
ADAMS	CLERK

9. Покажите следующие столбцы по служащим, имеющим руководителя.

ENAME	JOB	SAL
BLAKE	MANAGER	2850
CLARK	MANAGER	2450
JONES	MANAGER	2975
MARTIN	SALESMAN	1250
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
JAMES	CLERK	950
WARD	SALESMAN	1250
FORD	ANALYST	3000
SMITH	CLERK	800
SCOTT	ANALYST	3000
ADAMS	CLERK	1100
MILLER	CLERK	1300

10. Получите имена и годовой доход всех служащих.

ENAME	Годовой доход
KING	60000
BLAKE	34200
CLARK	29400
JONES	35700
MARTIN	16400
ALLEN	19500
TURNER	18000
JAMES	11400
WARD	15500
FORD	36000
SMITH	9600
SCOTT	36000
ADAMS	13200
MILLER	15600

11. Покажите следующую информацию о всех служащих, зачисленных на работу в 1981 году.

ENAME	DEPTNO	HIREDATE
KING	10	17.11.81
BLAKE	30	01.05.81
CLARK	10	09.06.81
JONES	20	02.04.81
MARTIN	30	28.08.81
ALLEN	30	20.02.81
TURNER	30	08.09.81
JAMES	30	03.12.81
WARD	30	22.02.81
FORD	20	03.12.81

12. Покажите имя, годовую зарплату и премию для всех продавцов (должность SALESMAN), у которых месячная зарплата (поле SAL) превосходит премию (поле COMM). Отсортируйте строки по полю SAL в порядке убывания. Если несколько служащих получают одинаковую зарплату, то в пределах строк с одинаковой зарплатой упорядочите их по именам сотрудников (поле ENAME).

ENAME	SAL	COMM
ALLEN	1600	300
TURNER	1500	0
WARD	1250	500

13. Получите таблицу результатов, как показано ниже.

Who, what and when
KING занимает должность PRESIDENT в отделе 10 принят 17.11.81
BLAKE занимает должность MANAGER в отделе 30 принят 01.05.81
CLARK занимает должность MANAGER в отделе 10 принят 09.06.81
JONES занимает должность MANAGER в отделе 20 принят 02.04.81
MARTIN занимает должность SALESMAN в отделе 30 принят 28.08.81
ALLEN занимает должность SALESMAN в отделе 30 принят 20.02.81
TURNER занимает должность SALESMAN в отделе 30 принят 08.09.81
JAMES занимает должность CLERK в отделе 30 принят 03.12.81
WARD занимает должность SALESMAN в отделе 30 принят 22.02.81
FORD занимает должность ANALYST в отделе 20 принят 03.12.81
SMITH занимает должность CLERK в отделе 20 принят 17.12.80
SCOTT занимает должность ANALYST в отделе 20 принят 09.12.82
ADAMS занимает должность CLERK в отделе 20 принят 12.12.83
MILLER занимает должность CLERK в отделе 10 принят 23.12.82

Лабораторная работа №2
ФУНКЦИИ

1. Выберите имена всех служащих и их зарплату, увеличенную на 15% и округленную до целых чисел.

DEPTNO	ENAME	PRT_SAL
10	KING	5750
30	BLAKE	3278
10	CLARK	2818
20	JONES	3421
30	MARTIN	1438
30	ALLEN	1840
30	TURNER	1725
30	JAMES	1093
30	WARD	1438
20	FORD	3450
20	SMITH	920
20	SCOTT	3450
20	ADAMS	1265
10	MILLER	1495

2. Сформируйте следующую таблицу результатов.

Имя_Должность
KING.....PRESIDENT
BLAKE.....MANAGER
CLARK.....MANAGER
JONES.....MANAGER
MARTIN.....SALESMAN
ALLEN.....SALESMAN
TURNER.....SALESMAN
JAMES.....CLERK
WARD.....SALESMAN
FORD.....ANALYST
SMITH.....CLERK
SCOTT.....ANALYST
ADAMS.....CLERK
MILLER.....CLERK

3. Сформируйте следующую таблицу результатов.

EMPLOYEE
KING (President)
BLAKE (Manager)
CLARK (Manager)
JONES (Manager)
MARTIN (Salesman)
ALLEN (Salesman)
TURNER (Salesman)
JAMES (Clerk)
WARD (Salesman)
FORD (Analyst)
SMITH (Clerk)
SCOTT (Analyst)
ADAMS (Clerk)
MILLER (Clerk)

4. Сформируйте следующую таблицу результатов.

ENAME	DEPTNO	Должность
BLAKE	30	Manager
MARTIN	30	Salesperson
ALLEN	30	Salesperson
TURNER	30	Salesperson
JAMES	30	Clerk
WARD	30	Salesperson

5. Выберите имена и даты зачисления на работу всех служащих 20 отдела. Замените наименование столбца с датами на DATE_HIRED. Если этого не сделать, то размер столбца по умолчанию будет равен 80 символам. На столбец с именем DATE_HIRED задана форматирующая команда, сокращающая его длину до 30 символов.

ENAME	DATE_HIRED
JONES	Апрель, Second,1981
FORD	Декабрь, Third,1981
SMITH	Декабрь, Seventeenth,1980
SCOTT	Декабрь, Ninth,1982
ADAMS	Декабрь, Twelfth,1983

6. Выдайте по каждому служащему его имя, дату зачисления на работу и дату годовой аттестации (Review Date). Дата аттестации наступает ровно через год после зачисления служащего на работу. Упорядочите строки по возрастанию дат аттестации.

ENAME	HIREDATE	REVIEW
SMITH	17.12.80	17.12.81
ALLEN	20.02.81	20.02.82
WARD	22.02.81	22.02.82
JONES	02.04.81	02.04.82
BLAKE	01.05.81	01.05.82
CLARK	09.06.81	09.06.82
MARTIN	28.08.81	28.08.82
TURNER	08.09.81	08.09.82
KING	17.11.81	17.11.82
FORD	03.12.81	03.12.82
JAMES	03.12.81	03.12.82
SCOTT	09.12.82	09.12.83
MILLER	23.12.82	23.12.83
ADAMS	12.12.83	12.12.84

7. Напечатайте список служащих, включающий их имя и оклад. Если оклад ниже $1500, то замените его на сообщение “Below 1500”. Значение $1500 замените на “On Target”.

ENAME	SALARY
ADAMS	BELOW 1500
ALLEN	1600
BLAKE	2850
CLARK	2450
FORD	3000
JAMES	BELOW 1500
JONES	2975
KING	5000
MARTIN	BELOW 1500
MILLER	BELOW 1500
SCOTT	3000
SMITH	BELOW 1500
TURNER	On Target
WARD	BELOW 1500

8. Напишите запрос, который выдает день недели по любой дате, указанной в запросе, формат представления даты DD.MM.YY.

DAY
ПОНЕДЕЛЬНИК

9. Определите сколько раз символ ‘S’ встречается в названиях отделов.

DNAME	L	MS
ACCOUNTING	10	0
RESEARCH	8	1
SALES	5	2
OPERATIONS	10	1

10. Напишите команду для вычисления количества времени в годах и месяцах, которое любой из служащих проработал в компании. Служащий должен задаваться по имени в запросе.

ENAME	Стаж работы
KING	26 YEARS 2 MONTHS

11. Служащему, зачисленному на работу до 15 числа любого месяца, платят первую зарплату в последнюю пятницу (Friday) этого месяца. Зачисленные на работу после 15 числа получают первую зарплату в последнюю пятницу следующего месяца. Выведите список имен служащих, дат зачисления на работу и дат первой выплаты. Отсортируйте данные по дате зачисления на работу.

ENAME	HIREDATE	PAYDAY
SMITH	17.12.80	30.01.81
ALLEN	20.02.81	27.03.81
WARD	22.02.81	27.03.81
JONES	02.04.81	24.04.81
BLAKE	01.05.81	29.05.81
CLARK	09.06.81	26.06.81
MARTIN	28.08.81	25.09.81
TURNER	08.09.81	25.09.81
KING	17.11.81	25.12.81
FORD	03.12.81	25.12.81
JAMES	03.12.81	25.12.81
SCOTT	09.12.82	31.12.82
MILLER	23.12.82	28.01.83
ADAMS	12.12.83	30.12.83

Лабораторная работа №3

ГРУППОВЫЕ ФУНКЦИИ

1. Найдите минимальную зарплату среди всех служащих

MINIMUM
800

2. Определите минимальную, максимальную и среднюю зарплату в компании.

MAX(SAL)	MIN(SAL)	AVG(SAL)
5000	800	2073

3. Вычислите минимальную и максимальную зарплату для каждой должности.

JOB	MAXIMUM	MINIMUM
SALESMAN	1600	1250
CLERK	1300	800
PRESIDENT	5000	5000
MANAGER	2975	2450
ANALYST	3000	3000

4. Определите, сколько менеджеров работает в компании, не выдавая по ним никаких данных.

MANAGERS
3

5. Вычислите среднюю зарплату и средний годовой доход для каждой из должностей. Не забудьте учесть премию.

JOB	AVGSAL	AVGCOMP
ANALYST	3000	36000
CLERK	1038	12450
MANAGER	2758	33100
PRESIDENT	5000	60000
SALESMAN	1400	17350

6. Определите разницу между наибольшим и наименьшим окладами в компании.

Разность
4200

7. Найдите отделы, в которых работает более трех служащих.

DEPTNO	COUNT(*)
30	6
20	5

8. Проверьте, действительно ли коды служащих (столбец EMPNO) в таблице EMP уникальны.

9. Выберите наименее оплачиваемых служащих, работающих на каждого из менеджеров. Исключите из таблицы результатов все группы, в которых минимальная зарплата меньше $1000. Упорядочите результаты по значению поля «Минимальная зарплата» в порядке возрастания.

MGR	MIN(SAL)
7788	1100
7782	1300
7839	2450
7566	3000
-	5000

Лабораторная работа №4

ПРОСТЫЕ СОЕДИНЕНИЯ

1. Найдите имена всех служащих и отделов, в которых они работают. Отсортируйте результаты по номерам отделов.

ENAME	DNAME
KING	ACCOUNTING
BLAKE	SALES
CLARK	ACCOUNTING
JONES	RESEARCH
MARTIN	SALES
ALLEN	SALES
TURNER	SALES
JAMES	SALES
WARD	SALES
FORD	RESEARCH
SMITH	RESEARCH
SCOTT	RESEARCH
ADAMS	RESEARCH
MILLER	ACCOUNTING

2. Выберите имена всех служащих, номера и наименования отделов, в которых они работают.

ENAME	DEPTNO	DNAME
KING	10	ACCOUNTING
BLAKE	30	SALES
CLARK	10	ACCOUNTING
JONES	20	RESEARCH
MARTIN	30	SALES
ALLEN	30	SALES
TURNER	30	SALES
JAMES	30	SALES
WARD	30	SALES
FORD	20	RESEARCH
SMITH	20	RESEARCH
SCOTT	20	RESEARCH
ADAMS	20	RESEARCH
MILLER	10	ACCOUNTING

3. Получите следующую информацию для всех сотрудников, у которых зарплата превышает $1500.

ENAME	LOCATION	DNAME
KING	NEW YORK	ACCOUNTING
BLAKE	CHICAGO	SALES
CLARK	NEW YORK	ACCOUNTING
JONES	DALLAS	RESEARCH
ALLEN	CHICAGO	SALES
FORD	DALLAS	RESEARCH
SCOTT	DALLAS	RESEARCH

4. Сформируйте таблицу, отражающую градацию служащих по уровню их зарплаты.

ENAME	JOB	SAL	GRADE
SMITH	CLERK	800	1
JAMES	CLERK	950	1
ADAMS	CLERK	1100	1
WARD	SALESMAN	1250	2
MARTIN	SALESMAN	1250	2
MILLER	CLERK	1300	2
TURNER	SALESMAN	1500	3
ALLEN	SALESMAN	1600	3
CLARK	MANAGER	2450	4
BLAKE	MANAGER	2850	4
JONES	MANAGER	2975	4
FORD	ANALYST	3000	4
SCOTT	ANALYST	3000	4
KING	PRESIDENT	5000	5

5. Сформируйте список служащих третьей категории оплаты.

ENAME	JOB	SAL	GRADE
ALLEN	SALESMAN	1600	3
TURNER	SALESMAN	1500	3

6. Найдите всех, кто работает в Далласе.

ENAME	LOCATION
JONES	DALLAS
FORD	DALLAS
SMITH	DALLAS
SCOTT	DALLAS
ADAMS	DALLAS

7. Выберите имя, должность, оклад, категорию оклада, наименование отдела для всех сотрудников компании, кроме клерков.

ENAME	JOB	GRADE	DNAME
KING	PRESIDENT	5	ACCOUNTING
FORD	ANALYST	4	RESEARCH
SCOTT	ANALYST	4	RESEARCH
JONES	MANAGER	4	RESEARCH
BLAKE	MANAGER	4	SALES
CLARK	MANAGER	4	ACCOUNTING
ALLEN	SALESMAN	3	SALES
TURNER	SALESMAN	3	SALES
WARD	SALESMAN	2	SALES
MARTIN	SALESMAN	2	SALES

8. Сформируйте следующую информацию по служащим, получающим ровно $36000 в год, а также по всем клеркам.

ENAME	JOB	TOTAL_SAL	DEPTNO	DNAME	GRADE
FORD	ANALYST	36000	20	RESEARCH	4
SCOTT	ANALYST	36000	20	RESEARCH	4
JAMES	CLERK	11400	30	SALES	1
ADAMS	CLERK	13200	20	RESEARCH	1
SMITH	CLERK	9600	20	RESEARCH	1
MILLER	CLERK	15600	10	ACCOUNTING	2

Лабораторная работа №5

ДРУГИЕ ВИДЫ СОЕДИНЕНИЙ И ОПЕРАЦИИ НАД МНОЖЕСТВАМИ

1. Выберите номера имен всех служащих, а также номера и имена их менеджеров. Используйте простое соединение.

EMPNO	ENAME	MGRNO	MGR_NAME
7698	BLAKE	7839	KING
7782	CLARK	7839	KING
7566	JONES	7839	KING
7654	MARTIN	7698	BLAKE
7499	ALLEN	7698	BLAKE
7844	TURNER	7698	BLAKE
7900	JAMES	7698	BLAKE
7521	WARD	7698	BLAKE
7902	FORD	7566	JONES
736	SMITH	7902	FORD
7788	SCOTT	7566	JONES
7876	ADAMS	7788	SCOTT
7934	MILLER	7782	CLARK

2. Модифицируйте запрос так, чтобы в таблице результатов появился KING, не имеющий менеджера.

EMPNO	ENAME	MGRNO	MGR_NAME
7839	KING	-	-
7698	BLAKE	7839	KING
7782	CLARK	7839	KING
7566	JONES	7839	KING
7654	MARTIN	7698	BLAKE
7499	ALLEN	7698	BLAKE
7844	TURNER	7698	BLAKE
7900	JAMES	7698	BLAKE
7521	WARD	7698	BLAKE
7902	FORD	7566	JONES
736	SMITH	7902	FORD
7788	SCOTT	7566	JONES
7876	ADAMS	7788	SCOTT
7934	MILLER	7782	CLARK

3. Найдите должность, на которую назначили служащих во второй половине 1980 года и в тот же период 1981 года.

JOB
CLERK

4. Определите всех служащих, зачисленных в компанию раньше своих менеджеров.

EMPLOYEE	HIREDATE	MANAGER	HIREDATE
BLAKE	01.05.81	KING	17.11.81
CLARK	09.06.81	KING	17.11.81
JONES	02.04.81	KING	17.11.81
ALLEN	20.02.81	BLAKE	01.05.81
WARD	22.02.81	BLAKE	01.05.81
SMITH	17.12.80	FORD	03.12.81

5. Найдите отдел, в котором нет служащих. Выполните это упражнение двумя способами.

DEPTNO	DNAME
40	OPERATIONS

Лабораторная работа 6

ПОДЗАПРОСЫ

1. Найдите служащих, получающих больше всех по своей должности. Упорядочите данные по убыванию значения зарплаты.

ENAME	JOB	SAL
KING	PRESIDENT	5000
FORD	ANALYST	3000
SCOTT	ANALYST	3000
JONES	MANAGER	2975
ALLEN	SALESMAN	1600
MILLER	CLERK	1300

2. Найдите служащих, получающих меньше всех по своей должности. Упорядочите данные по возрастанию значения зарплаты.

JOB	ENAME	SAL
CLERK	SMITH	800
SALESMAN	WARD	1250
SALESMAN	MARTIN	1250
MANAGER	CLARK	2450
ANALYST	FORD	3000
ANALYST	SCOTT	3000
PRESIDENT	KING	5000

3. Определите, кто из служащих в каждом из отделов был зачислен на работу последним по времени. Результаты упорядочите по дате зачисления.

DEPTNO	ENAME	HIREDATE
30	JAMES	03.12.81
10	MILLER	23.12.82
20	ADAMS	12.12.83

4. Сформируйте следующую информацию по служащим, получающим больше средней зарплаты в их отделе. Отсортируйте данные по номерам отделов.

ENAME	ЗАРПЛАТА	DEPTNO
KING	5000	10
CLARK	2450	10
SCOTT	3000	20
JONES	2975	20
FORD	3000	20
BLAKE	2850	30

5. Найдите все отделы, не имеющие служащих, используя подзапрос.

DEPTNO	DNAME
40	OPERATIONS

6. Получите показанные ниже данные для отдела, в котором суммарный годовой доход сотрудников максимален.

DEPTNO	КОМПЕНСАЦИЯ
20	130500

7. Получите список сотрудников, имеющих хотя бы более одного подчиненного.

EMPNO	ENAME	JOB	DEPTNO
7782	CLARK	MANAGER	10
7839	KING	PRESIDENT	10
7902	FORD	ANALYST	20
7788	SCOTT	ANALYST	20
7566	JONES	MANAGER	20
7698	BLAKE	MANAGER	30

8. Выбрать те отделы, где число сотрудников больше трех и средняя зарплата больше средней зарплаты отдела 30.

DEPTNO	AVG(SAL)
20	2175

9. Кто входит в тройку самых высокооплачиваемых сотрудников компании? Найдите их имена и оклады.

ENAME	SAL
KING	5000
FORD	3000
SCOTT	3000

10. В каком году в компанию было зачислено наибольшее количество человек? Выдайте этот год и количество зачисленных служащих.

ГОД	КОЛИЧЕСТВО
1981	10

11. Сформируйте следующую информацию по служащим, получающим больше средней зарплаты в их отделе. Отсортируйте данные по номерам отделов.

ENAME	SAL	DEPTNO	СРЕДНЯЯ
ALLEN	1600	30	1567
BLAKE	2850	30	1567
FORD	3000	20	2175
JONES	2975	20	2175
SCOTT	3000	20	2175
KING	5000	10	2917

12. Получите список сотрудников, не имеющих подчиненных.

EMPNO	ENAME	JOB	DEPTNO
7934	MILLER	CLERK	10
7876	ADAMS	CLERK	20
736	SMITH	CLERK	20
7900	JAMES	CLERK	30
7844	TURNER	SALESMAN	30
7521	WARD	SALESMAN	30
7499	ALLEN	SALESMAN	30
7654	MARTIN	SALESMAN	30

13. Напишите запрос, печатающий звездочку против служащего, зачисленного на работу последним по времени. Укажите столбцы ENAME и HIREDATE таблицы EMP, а также столбец MAXDATE, в который будет помещена эта звездочка.

ENAME	HIREDATE	MAXDATE
ADAMS	12.12.83	*
ALLEN	20.02.81	
BLAKE	01.05.81	
CLARK	09.06.81	
FORD	03.12.81	
JAMES	03.12.81	
JONES	02.04.81	
KING	17.11.81	
MARTIN	28.08.81	
MILLER	23.12.82	
SCOTT	09.12.82	
SMITH	17.12.80	
TURNER	08.09.81	
WARD	22.02.81	

14. Получите список сотрудников отдела 30, получающих зарплату больше минимальной в 20 отделе.

ENAME	SAL	JOB	DEPTNO
JAMES	950	CLERK	30
MARTIN	1250	SALESMAN	30
WARD	1250	SALESMAN	30
TURNER	1500	SALESMAN	30
ALLEN	1600	SALESMAN	30
BLAKE	2850	MANAGER	30