Белорусский государственный университет

Практикум по курсу «Модели данных и СУБД»

учебное пособие

Для студентов университетов специальностей «Информатика», «Прикладная математика», «Актуарная математика» и «Экономическая кибернетика

Минск 2005

Авторы: Бондаренко С.П, Исаченко А.Н

Рецензенты: Образцов В.А., Разоренов Н.А.

Данный лабораторный практикум содержит пять лабораторных работ по языку SQL и по процедурному расширению языка SQL языку PL/SQL реляционной многопользовательской системы базы данных ORACLE.

Каждая лабораторная работа содержит необходимые теоретические сведения по рассматриваемой теме, демонстрационные примеры, задания для самостоятельной работы и контрольные вопросы.

Приведенные в конце пособия задания для индивидуального выполнения охватывают весь рассматриваемый учебный материал и позволяют закрепить пройденный курс.

Лабораторный практикум предназначен для студентов ВУЗов специальностей «Информатика», «Прикладная математика», «Актуарная математика» и «Экономическая кибернетика.

ВВЕДЕНИЕ

В практикуме излагаются основы работы с многопользовательской СУБД ORACLE – современной системой управления реляционными базами данных, поддерживающей работу в различных операционных средах и обеспечивающей реализацию самых современных технологий в области обработки данных. Появившись на рынке в 1979 году, постоянно развиваемая и совершенствуемая, система ORACLE в настоящее время является наиболее распространенным программным продуктом для решения задач, связанных с созданием баз данных и разработкой многофункциональных приложений, как в больших корпоративных системах, так и в сфере интернет-бизнеса. В пользу ее выбора говорят наличие средств, гарантирующих целостность и непротиворечивость данных; предоставление дополнительных механизмов обеспечения безопасности, которые не поддерживаются операционной системой и системой передачи данных; реализация многопользовательского доступа; поддержка объектно-реляционных баз данных; использование в качестве языка доступа к данным языка SQL; наличие процедурного языка программирования PL/SQL и различных инструментальных средств для разработки ORACLE-приложений.

Данный практикум предназначен для самостоятельного изучения языка запросов SQL, процедурного языка PL/SQL и приобретения необходимых практических навыков для работы с системой ORACLE. Он состоит из пяти лабораторных работ, которые содержат необходимые теоретические сведения по рассматриваемой теме и примеры, дополняющие изложенный теоретический материал. В конце каждой лабораторной работы имеются задания для самостоятельного выполнения, которые состоят из двух частей: заданий для практического исполнения и контрольных вопросов для повторения основ данной темы.

Первые две лабораторные работы охватывают материал, связанный с изучением языка SQL и получением необходимых практических навыков его использования. В первой рассматриваются аспекты работы с базами данных, связанные с созданием и модификацией объектов, входящих в схему пользователя (таблиц, представлений, последовательностей и индексов), обеспечением непротиворечивости, целостности и модификации данных, хранящихся в базе, управлением доступом к базе данных и ее объектам. Во второй излагаются вопросы формирования запросов на выборку информации из базы данных. Практические примеры иллюстрируют разнообразные возможности языка SQL. Предлагаемые задания для

самостоятельного выполнения снабжены ответами, что позволяет студенту самому контролировать процесс усвоения учебного материала. Последующие три работы обеспечивают возможность получения навыков в области разработки приложений для работы с базой данных на процедурном расширении языка SQL – языке PL/SQL. Рассмотрены основные операторы языка, использование курсоров, обработка исключительных ситуаций, создание триггеров, хранимых процедур и функций.

В конце практикума приведены индивидуальные задания, представляющие собой практические задачи из различных предметных областей, для выполнения которых необходимо использовать весь изложенный учебный материал. Демонстрационные примеры и задания являются оригинальными авторскими разработками.

Материал пособия апробирован на курсах переподготовки Специального факультета бизнеса и информационных технологий БГУ. Он также использовался при проведении занятий по курсу «Модели данных и СУБД» на факультете прикладной математики и информатики студентами специальностей «Прикладная математика», «Актуарная математика» и «Экономическая кибернетика».

Пособие предназначено для использования в учебном процессе факультета прикладной информатики и информатики при изучении курса «Модели данных и СУБД».

Лабораторная работа 1

СОЗДАНИЕ И РЕДАКТИРОВАНИЕ ТАБЛИЦ БАЗЫ ДАННЫХ

Цель работы. Знакомство с инструментальным средством SQL*Plus и основными элементами языка SQL. Создание таблиц базы данных. Задание ограничений целостности данных. Модификация информации и изменение структуры таблиц. Удаление таблиц из базы данных. Работа с демонстрационными примерами и самостоятельная работа.

ОСНОВНЫЕ СВЕДЕНИЯ И ДЕМОНСТРАЦИОННЫЕ ПРИМЕРЫ

Работа с инструментальным средством SQL*Plus

Инструментальное средство SQL*Plus представляет собой интерактивную среду доступа к базе данных ORACLE. С его помощью можно:

1) выполнять различные действия по обработке данных, хранящихся в базе ORACLE, используя операторы языка запроса SQL или программы, записанные на процедурном расширении языка SQL языке PL/SQL;

2) осуществлять администрирование базы данных.

Для запуска SQL*Plus необходимо выбрать следующую последовательность команд:

Пуск –> Программы –> Oracle for Windows NT –> SQL*Plus или SQL Plus 8.0

При запуске SQL*Plus на экране отображается диалоговое окно, в котором пользователю предлагается ввести свой идентификатор и пароль. Для работы в демонстрационном режиме пользователь должен ввести идентификатор SCOTT и пароль TIGER на верхнем или нижнем регистре. Завершить работу с SQL*Plus можно вводом команды EXIT.

В среде SQL*Plus можно выполнить оператор SQL, программный блок языка PL/SQL или команду SQL*Plus, реализующую служебную функцию. При этом следует иметь в виду, что оператор SQL должен быть завершен символом точка с запятой (;), блок PL/SQL должен завер-

шаться символом наклонная черта (/), команда SQL*Plus набирается без завершающего символа. Примером команды SQL*Plus является команда DESC[RIBE], предназначенная для просмотра информации об объекте базы данных. Если ввести команду DESC[RIBE] KNIGA, где KNIGA – имя таблицы, хранящейся в базе данных, то будет выведена информация о структуре этой таблицы.

Оператор SQL или блок PL/SQL могут занимать несколько строк. Выполненный оператор SQL или блок PL/SQL заносится в специальную область памяти, называемую буфером SQL, и может быть выполнен повторно либо командой R[UN], либо вводом символа наклонная черта (/). К хранимой в буфере SQL информации можно командой A[PPEND] text добавить text в конец строки, а командой CH[ANGE] /old/new заменить фрагмент текста old на фрагмент текста new. Для редактирования содержимого буфера SQL можно воспользоваться текстовым редактором Notepad, который операционной системой Windows используется по умолчанию. Для его вызова следует набрать команду ED[IT]. Содержимое буфера SQL можно записать на диск командой SAVE имя_файла и загрузить с диска командой GET имя_файла.

Любую последовательность операторов SQL, блоков PL/SQL и служебных команд SQL*Plus, расположенную на диске и называемую командным файлом или скриптом, можно выполнить одним из следующих двух способов:

START путь_файла\имя_файла

@путь_файла\имя_файла

Основы языка SQL

Язык SQL (Structured Query Language – язык структурированных запросов) является непроцедурным языком и ориентирован на операции с данными, представленными в виде логически взаимосвязанных совокупностей таблиц. Формулируя запросы на языке SQL можно создавать и модифицировать различные объекты базы данных и, оперируя группами строк, вставлять, выбирать, обновлять и удалять данные из таблиц. Кроме того, он позволяет управлять доступом к базе данных и ее объектам, и обеспечивать непротиворечивость и целостность данных, хранящихся в базе.

Алфавит языка. Включает следующие символы:

1) буквы: A..Z, a..z;
2) цифры: 0..9;
3) символы: + – * / ! @ $ # = < > ^ ' “ ( ) | _ ; , .

Идентификаторы и комментарии. Идентификаторы, длина которых может достигать 30 символов, обычно начинаются с буквы и могут включать в себя также цифры, символы $ и #, символ подчеркивания. Исключение составляют имена базы данных (до восьми символов). В некоторых версиях системы ORACLE допускается использование русских букв. Имя любого объекта может дополнительно иметь уточнитель – имя схемы: [схема.]имя_объекта. Схема представляет собой набор объектов разной структуры, принадлежащих конкретному пользователю, и идентифицируется его именем. Среди объектов схемы могут быть таблицы, представления (виртуальные таблицы), индексы, последовательности, триггеры, процедуры и функции.

Допускается использование однострочных и многострочных комментариев. Однострочные комментарии представляют собой следующую конструкцию:

```
-- текст комментария
```

Многострочные комментарии имеют следующий вид:

```
/*  текст комментария  */
```

Литералы. Символьные литералы определяются как тип CHAR и записываются в одинарных кавычках: 'test'. При необходимости присутствия одинарной кавычки внутри символьного литерала она удваивается.

Числовые литералы определяются как тип NUMBER и представляют собой целое или действительное значения со знаком или без знака, при этом действительные значения могут быть записаны в формате с десятичной точкой или в экспоненциальной форме.

Пустые значения. В языке SQL имеется специальное предопределенное значение NULL, которое расценивается как неопределенное значение. Оно не эквивалентно понятию пустая строка для символьных типов и не эквивалентно нулевому значению для числовых типов. Если в некотором столбце таблицы данные отсутствуют, говорят что его значение NULL. Столбец с данными любого типа может содержать значение NULL, если только он специально не описан как NOT NULL.

Псевдостолбцы. Это формируемые системой столбцы, имеющие стандартные имена. Их значения можно только просматривать и использовать, но корректировать (добавлять, удалять, изменять) нельзя.

К ним относятся: ROWID, ROWNUM, LEVEL, CURVAL, NEXTVAL.

Псевдостолбец ROWID содержит уникальные для всей базы данных физические адреса строк таблицы. Значение псевдостолбца ROWID определяется при вставке строки в таблицу и не изменяется, пока строка присутствует в таблице.

Псевдостолбец ROWNUM определяет порядковый номер строки, выбранной из таблицы при выполнении запроса. Он обычно используется для ограничения числа строк, выбираемых из таблицы.

Псевдостолбец LEVEL возвращает уровень вложенности данных, позволяя тем самым строить запросы для получения информации об иерархии данных.

Для работы с последовательностями генерируемых значений, используемых в качестве уникальных ключей, имеются псевдостолбцы:

имя_последовательности.CURRVAL – возвращает текущее значение из указанной последовательности генерируемых значений;

имя_последовательности.NEXTVAL – возвращает следующее значение из указанной последовательности генерируемых значений. Предварительно последовательность с именем имя_последовательности должна быть создана с помощью оператора CREATE SEQUENCE.

Типы данных. К наиболее часто используемым типам данных относятся символьные, числовые, тип DATE, двоичные и большие объекты.

Символьные типы данных представлены следующими типами:

CHAR(длина), VARCHAR2(длина) и LONG.

Тип данных CHAR представляет собой символьные строки фиксированной длины. Минимальная длина равна 1, максимальная – 2000 байт. Если значение, помещаемое в столбец данного типа, превосходит указанный размер, то выводится сообщение об ошибке; если длина помещаемого значения меньше указанной длины, то значение дополняется пробелами справа.

Тип данных VARCHAR2 представляет собой символьные строки переменной длины. Максимальный размер строки 4000 байт, минимальный – 1 байт. При помещении текста в столбец большего размера дополнение пробелами не производится.

Строки этих двух типов сравниваются по-разному. Строки типа CHAR – посимвольно с дополнением пробелами строки с меньшей длиной до размера строки с большей длиной. Строки VARCHAR2 – без дополнения пробелами до большей длины. Поэтому для двух в принципе одинаковых строк могут быть получены различные результаты при их сравнении.

Пример. Для двух строк 'AB' и 'AB ' (вторая строка содержит пробел, а первая нет) типа CHAR получим, что 'AB' = 'AB '. Для этих же строк типа VARCHAR2 результатом сравнения будет 'AB' < 'AB '.

Тип данных LONG представляет собой символьные данные переменной длины, величина которой может достигать 2 Гб. На использование переменных этого типа накладывается ряд ограничений.

Числовые типы представлены типом NUMBER, который позволяет определить три различных типа данных:

1) NUMBER;
2) NUMBER(p);
3) NUMBER(p, s);

В первом случае определяются действительные числа, диапазон которых от $1.0*10^{-130}$ до $1.0*10^{126}-1$, мантисса содержит 38 знаков. Во втором случае считается, что определяется диапазон целых чисел, где p – количество цифр в числе (от 1 до 38). В третьем случае описываются числа с фиксированной точкой; p – общее количество цифр (от 1 до 38), s – масштаб (от –84 до 127) – определяет количество цифр после запятой. Если $s > 0$, то число округляется до указанного числа знаков справа от десятичной точки, если $s < 0$, то число округляется до указанного числа знаков слева от десятичной точки.

Следует отметить, что язык SQL поддерживает типы данных стандарта ANSI SQL. Если такой тип данных (INTEGER, SMALLINT, DECIMAL, FLOAT и REAL и т. д.) встречаются при определении типа столбца таблицы, то имя типа сохраняется, но сами данные хранятся в виде, определяемом одним из типов базы данных Oracle.

Тип данных DATE представляет собой специальным образом организованный тип. Он хранит столетие, год, месяц, день, часы, минуты, секунды. Для выборки текущего дня и времени используется функция SYSDATE. Стандартный формат даты, автоматически преобразуемый во внутреннее представление, записывается символьной строкой следующего вида: 'DD–MON–YY', где DD – день месяца, MON – краткое название месяца, а YY – значение года.

Над переменными типа DATE можно выполнить арифметическое действие – вычитание. Результат операции определяет количество дней между этими двумя датами. К дате можно прибавить или отнять числовую константу, рассматриваемую как количество дней или часть дня.

Двоичные типы данных используются для хранения двоичных неструктурированных данных (звуковые файлы, файлы изображений и т. д.), обработка которых не поддерживается системой ORACLE. К ним относятся типы RAW(длина) и LONGRAW. Максимальный размер строки типа RAW 2000 байт, минимальный – 1 байт. Длина переменных типа LONGRAW может достигать 2 Гб.

К большим объектам (LOB-объектам) относятся CLOB, BLOB и BFILE. Они предназначены для хранения неструктурированных данных большого объема – до 4 Гб. Тип CLOB хранит данные символьного типа,

типы BLOB и BFILE используют для хранения двоичных данных. Столбцы типа LOB содержат не сами данные, а указатели на их местоположение.

Операторы SQL. Все операторы SQL делятся на ряд групп:

1) операторы языка описания данных – DDL;
2) операторы языка манипулирования данными – DML;
3) операторы управления транзакциями;
4) операторы управления сеансом;
5) операторы управления системой.

К операторам DDL относятся следующие операторы:

CREATE, ALTER, DROP, GRANT, REVOKE.

Оператор CREATE используется для создания объектов базы данных. К ним относятся:

TABLE – таблица базы данных;

VIEW – представление или вид, представляет собой виртуальную таблицу, которая строится на основе выбранной в результате выполнения запроса информации из одной или нескольких таблиц;

SEQUENCE – последовательность, последовательно выбираемые значения которой можно использовать для задания уникальных значений ключа;

INDEX – индекс, используемый для обеспечения более быстрого доступа к данным таблицы;

TRIGGER – хранимая в базе данных программная единица, запускаемая ORACLE автоматически при наступлении определенных событий;

FUNCTION – хранимая в базе данных программная единица, вызываемая пользователем или другими программными единицами для выполнения;

PROCEDURE – хранимая в базе данных программная единица, вызываемая пользователем или другими программными единицами для выполнения;

USER – имя пользователя, имеющего доступ к информации базы данных;

ROLE – совокупность определенных привилегий, обеспечивающих возможность создания, удаления и модификации объектов базы данных.

Оператор ALTER используется для изменения объектов базы данных. Применяется по отношению ко всем перечисленным выше объектам.

Оператор DROP применяется для удаления всех вышеперечисленных объектов из базы данных.

Оператор GRANT позволяет наделить роль или пользователя различного вида привилегиями или ролями.

Оператор REVOKE отменяет предоставленные пользователям или ролям привилегии и роли.

Операторы DML. К этой группе относятся операторы:

INSERT, DELETE, UPDATE, SELECT.

Первые три оператора позволяют осуществить соответственно вставку, удаление и модификацию строк таблиц. Оператор SELECT предназначен для построения запросов, в результате выполнения которых из таблиц выбирается вся необходимая информация. Запрос на выборку информации, включенный в запись некоторого другого оператора, образует подзапрос.

Операторы управления транзакциями. К этой группе относятся следующие операторы:

COMMIT – фиксация текущей транзакции;

ROLLBACK – откат текущей транзакции;

Транзакция представляет собой неделимую с точки зрения системы единицу работы, выполняемую системой. В случае успешного выполнения всех входящих в транзакцию действий ее результаты фиксируются (COMMIT). В противном случае состояние базы данных можно вернуть в исходное состояние, отменив транзакцию (ROLLBACK).

Операторы управления сеансом работы изменяют установки, задаваемые для сеанса работы в БД (ALTER SESSION, SET ROLE).

Операторы управления системой изменяют установки всей БД (ALTER SYSTEM).

Операции языка SQL. Операции языка SQL делятся на ряд групп.

Арифметические операции представлены двумя группами операций:

1) унарные +, –
2) бинарные +, –, *, /

Арифметические операции используются в выражениях для изменения знака операнда, сложения, вычитания, умножения и деления числовых величин. Унарные операции оперируют только с одним операндом, бинарные требуют при своем использовании два операнда.

Операции над строками. В этой группе операций имеется только одна операция – операция сцепления строк. Для ее обозначения используется комбинация двух символов – вертикальная черта (||).

Операции сравнения. Применяются в основном в операторах DML при построении простых условий проверки для сравнения значения одного выражения со значением другого выражения. Результатом сравне-

ния может быть либо TRUE, либо FALSE, либо UNKNOWN. Значение UNKNOWN может появиться в результате сравнения значений двух выражений, если одно из выражений или они оба имели значение NULL. Над значениями двух выражений X и Y могут быть выполнены следующие операции сравнения:

X = Y – проверка значений выражений X и Y на равенство; результат равен TRUE, если указанное соотношение выполняется;

X != Y, X<>Y, X ^= Y – проверка значений выражений X и Y на неравенство; результат равен TRUE, если указанное соотношение выполняется;

X < Y, X > Y, X >= Y, X <= Y – проверка значений выражений X и Y на соотношение «меньше, чем», «больше чем», «больше или равно», «меньше или равно»; результат равен TRUE, если указанное соотношение выполняется;

X [NOT] BETWEEN A AND B – проверка, [не] находится ли значение выражения X в указанном диапазоне, определяемом значениями выражений A и B; результат равен TRUE, если указанное соотношение выполняется;

X IN (список выражений | подзапрос) – проверка значения выражения X на равенство некоторому элементу из списка значений выражений или множества значений, возвращенных подзапросом; результат равен TRUE, если указанное соотношение выполняется хотя бы для одного элемента списка выражений или множества значений, возвращенных подзапросом;

X NOT IN (список выражений | подзапрос) – проверка значения выражения X на неравенство ни одному элементу из списка значений выражений или множества значений, возвращенных подзапросом; результат равен TRUE, если указанное соотношение выполняется для всех элементов списка выражений или множества значений, возвращенных подзапросом;

X LIKE Z – проверка значения выражения X на подобие, результат проверки равен TRUE, если X совпадает с шаблоном Z. Шаблон представляет собой символьную строку, внутри которой символ '%' используется для сопоставления с любой строкой из нуля или более символов, кроме NULL – строки, а символ '_' сопоставляется с любым одиночным символом;

X IS [NOT] NULL – проверка значения выражения X на [не] пустое значение NULL; результат равен TRUE, если указанное соотношение выполняется;

Операция сравнения с квантором ANY позволяет сравнивать проверяемое значение со всеми элементами из заданного списка значений выражений или множества значений, возвращенных подзапросом; результат проверки равен TRUE, если указанная операция сравнения (=, !=, >, <, >=, <=) выполняется хотя бы для одного элемента списка выражений или множества значений, возвращенных подзапросом;

Операция сравнения с квантором ALL позволяет сравнивать проверяемое значение со всеми элементами из заданного списка значений выражений или множества значений, возвращенных подзапросом; результат проверки равен TRUE, если указанная операция сравнения (=, !=, >, <, >=, <=) выполняется для всех элементов списка выражений или множества значений, возвращенных подзапросом;

Операция сравнения EXISTS проверяет результат выполнения подзапроса; результат проверки равен TRUE, если подзапрос возвращает не пустое множество значений.

Логические операции. Представлены стандартными логическими операциями: NOT, AND, OR, используемыми при построении сложных условий проверки, в которых простые условия объединяются в более сложное условие с помощью логических операций.

Логические операции выполняются в трехзначной логике, которая задается следующими таблицами истинности:

OR	True	False	Unknown
True	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

AND	True	False	Unknown
True	True	False	Unknown
False	False	False	False
Unknown	Unknown	False	Unknown

NOT	True	False	Unknown
	False	True	Unknown

Операции над множествами. Позволяют выполнить определенные действия над выбираемыми в результате выполнения одного или не-

скольких запросов группами строк. Естественно, что структуры этих строк должны совпадать по количеству, порядку расположения и типу данных входящих в них элементов. К ним относятся следующие операции:

UNION ALL – объединяет все строки, извлеченные одним или несколькими запросами, включая повторяющиеся;

UNION – объединяет все строки, извлеченные одним или несколькими запросами, с устранением дублирующих строк;

INTERSECT – объединяет только те строки, которые присутствуют в результатах выполнения каждого из запросов, с устранением дублирующих строк;

MINUS – объединяет все неповторяющиеся строки, извлеченные первым запросом, но не извлеченные вторым.

Другие операции. Класс других операций содержит две операции: операцию внешнего соединения (+) и специальную операцию PRIOR.

Операция внешнего соединения используется при выборе информации из нескольких таблиц в том случае, если из одной таблицы необходимо выбрать все строки, а из остальных таблиц только те строки, для которых выполняются определенные условия.

Операция PRIOR устанавливает взаимосвязь между родительскими и дочерними строками при построении иерархических запросов.

Функции SQL. Рассмотрим наиболее часто используемые группы функций языка SQL.

Числовые функции. Предназначены для вычисления степени числа, абсолютного значения, округления и усечения числа с заданной точностью, вычисления тригонометрических значений. Опишем некоторые числовые функции.

Функция ABS(n) возвращает абсолютное значение аргумента n, имеющего числовой тип.

Функция ROUND(n, [r]) осуществляет округление значения аргумента n, имеющего числовой тип, с точностью до количества указанных знаков r. При этом если значение r положительно, то округление производится до указанного количества знаков после запятой, если значение r отрицательно, то округление производится до указанного количества знаков до запятой. При r = 0 функция возвращает округленную целую часть аргумента n.

Функция MOD(m, n) возвращает остаток от деления целочисленного аргумента m на целочисленный аргумент n.

Функция POWER(m, n) возвращает аргумент m, имеющий числовой тип, возведенный в степень, заданную аргументом n, имеющим также числовой тип.

Функция SQRT(m) возвращает квадратный корень из аргумента n, имеющего числовой тип.

Символьные функции. Предназначены для работы со строками. Они могут возвращать либо строку, либо целое значение. Ниже приводится описание некоторых символьных функций.

Функция UPPER(str) возвращает строку str, все символы которой преобразованы в верхний регистр.

Функция LENGTH(str) возвращает длину строки str в символах.

Функция SUBSTR(str, n, m) выделяет из строки str подстроку длины n, начиная с символа в позиции m.

Функция LPAD(str, n, chr) возвращает строку str, дополненную слева указанным символом chr до указанной длины n.

Функция RPAD(str, n, chr) возвращает строку str, дополненную справа указанным символом chr до указанной длины n.

Функции работы с датами. Предназначены для работы с данными типа DATE. Ниже описываются некоторые функции этой группы.

Функция ADD_MONTHS(data, n) добавляет к указанной дате data или вычитает из нее n месяцев.

Функция MONTHS_BETWEEN(data1, data2) возвращает количество месяцев, находящихся между указанными датами data1 и data2.

Функция LAST_DAY(data) возвращает последний день месяца, указанного датой data.

Функции преобразования типа. В основном используются для преобразования данных символьного типа в числовой или в тип DATE и, наоборот, для преобразования данных числового типа или типа DATE в символьный тип. Преобразование осуществляется в соответствии с задаваемым форматом. Формат преобразования имеет вид символьной строки, где каждый символ или группа символов имеет определенное назначение.

Функция TO_CHAR(d1, [fmt]) преобразует значение d1 типа DATE в значение типа VARCHAR2 по формату fmt.

Функция TO_CHAR(n1, [fmt]) преобразует значение n1 типа NUMBER в значение типа VARCHAR2 по формату fmt.

Функция TO_DATE(char, [fmt]) преобразует значение char типа CHAR или VARCHAR2 в значение типа DATE по формату fmt.

Функция TO_NUMBER(char, [fmt]) преобразует значение char типа CHAR или VARCHAR2 в значение типа NUMBER по формату fmt.

В форматах для даты используются следующие группы символов:

DD – задает номер дня месяца в диапазоне от 1 до 31;

DAY – задает полное название дня недели;

MON – задает краткое название месяца;

MONTH – задает полное название месяца;

YY – задает две последние цифры номера календарного года;

YYYY – задает полный номер календарного года.

В форматах для чисел используются следующие символы:

цифра 9 – задает цифру;

символ точка (.) – задает десятичную точку;

цифра 0 – задает обязательный ноль;

буква s – задает обязательное наличие знака {+; –};

символ $ – задает знак доллара, проставляемый в начале числа.

Групповые функции. Выполняют операции над группами строк.

Функция COUNT({*}) – возвращает количество строк в группе.

Функция COUNT([DISTINCT] выражение) – возвращает количество строк в группе, игнорируя значение NULL.

Функция SUM([DISTINCT] выражение) – возвращает сумму значений указанного выражения для группы строк или списка значений, игнорируя значение NULL.

Функция AVG([DISTINCT] выражение) – возвращает среднее значение указанного выражения для группы строк или списка значений, игнорируя значение NULL.

Функция MIN([DISTINCT] выражение) – возвращает минимальное из значений указанного выражения для группы строк или списка значений, игнорируя значение NULL.

Функция MAX([DISTINCT] выражение) – возвращает максимальное из значений указанного выражения для группы строк или списка значений, игнорируя значение NULL.

Фраза DISTINCT предписывает групповым функциям рассматривать только различные значения выражения.

Другие функции. Функция NVL(выражение1, выражение2) обрабатывает пустое значение NULL. Если значение выражения1 равно NULL, то функция возвращает значение выражения2; если же значение выражения1 не равно NULL, то функция возвращает значение выражения1.

Создание таблиц, модификация информации и изменение структуры таблицы, удаление таблиц

Создание таблицы выполняется с помощью оператора CREATE, упрощенный синтаксис которого имеет следующий вид:

CREATE TABLE имя_таблицы
{({ имя_столбца тип_данных [DEFAULT значение]
[ограничения_столбца] | ограничение_таблицы}
[,{ имя_столбца тип_данных [DEFAULT значение]
[ограничения_столбца] | ограничение_таблицы}]...) |
AS подзапрос};

Таблица может быть создана либо стандартным образом через описание ее компонент, либо в результате выполнения некоторого подзапроса. Подзапрос – это обычный запрос на выборку информации, реализуемый оператором SELECT. При создании таблицы задаваемые имена таблиц и имена столбцов должны удовлетворять правилам, предписываемые идентификаторам. При этом естественно, что имена, присваиваемые таблицам, должны быть уникальными в схеме пользователя, а имена столбцов должны быть уникальными в рамках одной таблицы. Для каждого столбца указываются тип данных и, если необходимо, значение, вставляемое в столбец по умолчанию (DEFAULT значение). Важным элементом при создании таблиц является задание ограничений целостности данных, которые позволяют отслеживать правильность модификации имеющихся данных или вставляемых в таблицу новых данных. Ограничения целостности данных делятся на два типа: ограничения столбца и ограничения таблицы. Ограничения столбца позволяют определить условия, которым должны удовлетворять значения соответствующего столбца; ограничения целостности данных, накладываемые на таблицу, позволяют проверить правильность всех добавляемых или модифицируемых строк таблицы. Ограничение может быть именованным или безымянным. Удобнее использовать именованные ограничения, поскольку при выдаче информации, связанной с возникшим нарушением одного из ограничений, выдается и имя этого ограничения, что весьма полезно для исправления ошибок в дальнейшем. Задание ограничений на столбец или ограничений на таблицу осуществляется по следующему синтаксису:

[CONSTRAINT имя_ограничения] тип_ограничения

Имеются следующие типы ограничений, накладываемых на столбец:

PRIMARY KEY – это ограничение требует, чтобы вводимые в столбец значения были уникальными и отличными от пустых, поскольку они будут использоваться в качестве первичного ключа, однозначно иденти-

фицирующего запись; первичный ключ определяется для таблицы только единожды;

UNIQUE – это ограничение требует, чтобы вводимые в столбец значения в рамках одной таблицы были уникальными;

NOT NULL – это ограничение требует обязательного присутствия в столбце некоторого значения;

CHECK(выражение) – это ограничение позволяет подвергнуть определенной проверке вставляемое в столбец значение; если условия, наложенные на вставляемые значения, не выполняются, то значения в столбец не помещаются;

REFERENCES – это ограничение позволяет установить взаимосвязь значений данного столбца со значениями столбца другой таблицы. Взаимосвязь обеспечивается использованием следующей конструкции:

REFERENCES имя_таблицы[(имя_столбца)] [ON DELETE CASCADE]

При внесении значения в столбец создаваемой таблицы система будет автоматически проверять наличие аналогичного значения в указанном столбце указанной таблицы. При этом естественно, что для обеспечения однозначности устанавливаемой взаимосвязи все значения, находящиеся в указанном столбце, на которые производится ссылка, должны иметь ограничение UNIQUE или PRIMARY KEY. Если в качестве имени столбца указанной таблицы используется первичный ключ, то имя столбца можно не указывать. Таблица, на чей столбец ссылается другая таблица, называется главной, а таблица, ссылающаяся на нее, – подчиненной. Конструкции ON DELETE CASCADE указывает, что при удалении строк в главной таблице автоматически осуществляется удаление соответствующих строк и в подчиненной таблице.

Ограничение на таблицу во многом напоминают ограничения столбца, но при этом обычно задействуют, как правило, несколько столбцов. Например, можно задать ограничение PRIMARY KEY, указав список имен столбцов, тем самым, определив составной первичный ключ. При этом для столбцов, указываемых в списке, должны быть заданы ограничения UNIQUE и NOT NULL.

Используя следующую форму записи:

FOREIGN KEY (список_имен_столбцов)
REFERENCES имя_таблицы(список_имен_столбцов)
[ON DELETE CASCADE]

можно определить составной внешний ключ для таблицы. Естественно, что в случае составного внешнего ключа перечень столбцов в подчинен-

ной таблице и перечень столбцов в главной таблице должны совпадать по количеству, типу данных и порядку следования. Конструкция ON DELETE CASCADE позволяет обеспечить целостность и непротиворечивость данных при изменениях, которые затрагивают значения столбцов главной таблицы, являющихся внешним ключом по отношению к подчиненной таблице.

Если ограничение CHECK затрагивает значения нескольких столбцов, увязывая их в некоторое достаточно сложное условие, то такое ограничение также удобно определить как ограничение на таблицу.

Для генерации и вставки в столбец таблицы уникальных значений можно создать специально предназначенный для этих целей объект – последовательность. Создание последовательности выполняется с помощью оператора CREATE по следующему упрощенному синтаксису:

```
CREATE SEQUENCE имя_последовательности
[START WITH начальное_значение] [INCREMENT BY шаг];
```

В самом простейшем случае генерируется последовательность целых чисел от 1 до 10^{27} с шагом единица. Конструкция INCREMENT BY позволяет указать шаг изменения значений последовательности. Конструкция START WITH позволяет задать начальное значение генерируемой последовательности, которое при ее отсутствии устанавливается равным единице. Для вставки в столбец текущего значения последовательности нужно указать имя_последовательности.CURRVAL, а для вставки в столбец измененного по правилам формирования последовательности следующего значения используется имя_последовательности.NEXTVAL. Последовательности являются самостоятельными объектами базы данных, одна и та же последовательность может быть использована для задания уникальных значений столбцов нескольких таблиц, при удалении или модификации последовательности значения, созданные ею, сохраняются в таблицах базы данных.

Вставка строк в таблицу осуществляется с помощью оператора INSERT, который имеет следующий синтаксис:

```
INSERT INTO имя_таблицы [( список_столбцов)]
{VALUES (значение1 [, значение2] ...) | подзапрос};
```

Если список столбцов не указывается, то список значений в конструкции VALUES должен содержать значения для всех столбцов таблицы, причем порядок их следования должен однозначно соответствовать порядку их следования в строке. Использование подзапроса позволяет перенести строки из некоторой таблицы в создаваемую таблицу.

Удаление строк из таблицы осуществляется с помощью оператора DELETE, который имеет следующий синтаксис:

DELETE [FROM] имя_таблицы [WHERE условие];

При отсутствии ключевого слова WHERE из таблицы удаляются все строки, но сама таблица остается.

Модификация строк таблицы реализуется с помощью оператора UPDATE, форма записи которого приведена ниже. При отсутствии условия модифицируются все строки таблицы для указанного списка столбцов.

```
UPDATE имя_таблицы
SET {(имя_столбца1 [, имя_столбца2] ...) = (подзапрос) |
имя_столбца1=значение1, имя_столбца2={значение2 | (подзапрос)}}
[WHERE условие];
```

Для изменения структуры таблицы используется оператор ALTER TABLE, с помощью которого можно осуществить добавление одного или несколько новых столбцов в таблицу, изменение характеристик у одного или нескольких столбцов, добавление ограничения столбца или таблицы или удаление ограничений столбца или таблицы. Примеры его использования приведены в следующем пункте.

Удаление таблицы можно выполнить с помощью следующего оператора:

DROP TABLE имя_таблицы [CASCADE CONSTRAINTS];

При наличии конструкции CASCADE CONSTRAINTS вместе с удалением таблицы уничтожаются ограничения внешнего ключа в других таблицах.

Демонстрационные примеры

З а д а н и е «Книжный магазин».

Постоянно работающий магазин напрямую контактирует с издательствами и имеет постоянный штат продавцов, постоянных и случайных покупателей. Постоянные покупатели регулярно оставляют заявки на книги. Директор магазина должен иметь сведения:

о поступивших книгах: название книги, фамилия автора, цена, издательство, жанр;

о распределении книг среди продавцов: название книги, фамилия продавца, количество экземпляров, дата поступления;

о продаже книг: название книги, фамилия продавца, фамилия покупателя, дата продажи.

1. Необходимо создать таблицы: KNIGA, KNIGA_POSTAVKA, указав все необходимые ограничения целостности данных.

Перечень, названия и тип данных столбцов таблицы KNIGA:

Код книги	КОД_КНИГИ	Number(5)
Название книги	НАЗВАНИЕ	Varchar2(25)
Фамилия автора	ФАМИЛИЯ	Varchar2(20)
Цена книги	ЦЕНА	Number(7)
Издательство	ИЗДАТЕЛЬСТВО	Varchar2(15)
Жанр	ЖАНР	Varchar2(15)

Информация таблицы KNIGA:

Код кн.	Назв. книги	Фам. автора	Цена	Изд-во	Жанр
1	Дюна	Герберт	268	Аст	Фантастика
2	Ингвар и Ольха	Никитин	168	Аст	Роман
3	Гибель Богов	Перумов	345	Аст	Фантастика
4	Мифы	Асприн	266	Аст	Детектив
5	Казаки	Толстой	5568	Нова	Роман
6	Еретики Дюны	Герберт	368	Аст	Фантастика
7	Князь Рус	Никитин	1168	Аст	Роман
8	Страсть	Перумов	1385	Аст	Детектив
9	Мифотолкование	Асприн	2265	Нова	Детектив
10	Война и мир	Толстой	4588	Нова	Роман
11	Еретики Дюны	Герберт	2668	Нова	Фантастика
12	Владимир	Никитин	568	Аст	Детектив
13	Гибель Титана	Кристи	2345	Аст	Роман

Перечень, название и тип данных столбцов таблицы KNIGA_POSTAVKA:

Код операции	КОД_ОПЕРАЦИИ	Number(10)
Код книги	КОД_КНИГИ	Number(5)
Фамилия продавца	ПРОДАВЕЦ	Varchar2(20)
Количество единиц	КОЛ_ЕДИНИЦ	Number(4)

Дата поставки ДАТА_ПОСТУПЛЕНИЯ Date

Информация таблицы KNIGA_POSTAVKA:

Код опер.	Код книги	Фам. продавца	Кол-во ед.	Дата поставки
1	1	Иванов	5	20-01-2004
2	2	Иванов	5	20-01-2004
3	3	Иванов	5	10-02-2004
4	4	Иванов	5	10-02-2004
5	5	Иванов	5	10-02-2004
6	6	Петров	4	25-01-2004
7	7	Петров	4	25-01-2004
8	8	Петров	4	20-02-2004
9	9	Петров	4	20-02-2004
10	10	Петров	4	20-02-2004
11	11	Сидоров	3	25-01-2004
12	12	Сидоров	3	25-01-2004
13	13	Сидоров	3	25-01-2004
147	Сидоров	3	20-02-2004	
154	Сидоров	3	20-02-2004	

П о я с н е н и е. Создание таблицы KNIGA выполняется с помощью оператора CREATE TABLE. Столбец КОД_КНИГИ, содержащий уникальный код книги, является ключевым, и по отношению к его значениям устанавливается ограничение PRIMARY KEY. За значениями для этого столбца пользователь должен следить сам. Ограничение именуется как PK_KN. Поскольку столбец НАЗВАНИЕ не может иметь пустые значения, на него накладывается ограничение NOT NULL. На значения столбца ЦЕНА накладывается ограничение, связанное со стоимостью книги, она должна быть не менее 100 рублей. Ограничение получает имя CEN_KN. Если таблица KNIGA была уже создана ранее, то перед повторным созданием ее следует удалить следующим оператором:

```
DROP TABLE KNIGA;
```

Следующий оператор CREATE языка SQL создает таблицу KNIGA с необходимыми ограничениями целостности данных:

```
CREATE TABLE KNIGA (
КОД_КНИГИ NUMBER(5) CONSTRAINT PK_KN PRIMARY KEY,
```

НАЗВАНИЕ VARCHAR2(25) NOT NULL, АВТОР VARCHAR2(20),
ЦЕНА NUMBER(7) CONSTRAINT CEN_KN CHECK(ЦЕНА>100),
ИЗДАТЕЛЬСТВО VARCHAR2(15), ЖАНР VARCHAR2(15));

Вставка строк в таблицу KNIGA осуществляется следующей совокупностью операторов:

```
INSERT INTO KNIGA VALUES(1,'Дюна','Герберт',268,'Аст',
'Фантастика');
INSERT INTO KNIGA VALUES(2,'Ингвар и Ольха', 'Никитин', 168,
'Аст', 'Роман');
INSERT INTO KNIGA VALUES(3,'Гибель Богов','Перумов',345,'Аст',
'Фантастика');
INSERT INTO KNIGA
VALUES(4,'Мифы','Асприн',266,'Аст','Детектив');
INSERT INTO KNIGA
VALUES(5,'Казаки','Толстой',5568,'Нова','Роман');
INSERT INTO KNIGA VALUES(6,'Еретики Дюны', 'Герберт', 368,
'Аст', 'Фантастика');
INSERT INTO KNIGA VALUES(7,'Князь Рус','Никитин',1168,'Аст',
'Роман');
INSERT INTO KNIGA VALUES(8,'Страсть','Перумов',1385,'Аст',
'Детектив');
INSERT INTO KNIGA VALUES(9,'Мифотолкование', 'Асприн', 2265,
'Нова', 'Детектив');
INSERT INTO KNIGA VALUES(10,'Война и мир', 'Толстой', 4588,
'Нова', 'Роман');
INSERT INTO KNIGA VALUES(11,'Еретики Дюны', 'Герберт', 2668,
'Нова', 'Фантастика');
INSERT INTO KNIGA VALUES(12,'Владимир','Никитин',568,'Аст',
'Детектив');
INSERT INTO KNIGA VALUES(13,'Гибель Титана', 'Кристи', 2345,
'Аст', 'Роман');
```

Просмотр введенных значений можно выполнить с помощью следующего оператора SQL:

```
SELECT * FROM KNIGA;
```

З а м е ч а н и е. Пример строк, вызывающих нарушение ограничений целостности данных, определенных для таблицы KNIGA:

а) Вводится строка с повторение первичного ключа – 5

INSERT INTO KNIGA VALUES(5, 'Казаки', 'Толстой', 5568, 'Нова', 'Роман');

б) Вводится строка с пустым значением для столбца НАЗВАНИЕ –

INSERT INTO KNIGA (КОД_КНИГИ, АВТОР, ЦЕНА, ИЗДАТЕЛЬСТВО, ЖАНР) VALUES(1, 'Герберт', 68, 'Аст', 'Роман');

в) Вводится строка со значением для столбца ЦЕНА меньше 100 –

INSERT INTO KNIGA VALUES(1, 'Дюна', 'Герберт', 68, 'Аст', 'Фантастика');

П о я с н е н и е. Создание таблицы KNIGA_POSTAVKA выполняется с помощью оператора CREATE TABLE. Столбец КОД_ОПЕРАЦИИ, содержащий уникальный код операции, является ключевым и по отношению к его значениям устанавливается ограничение PRIMARY KEY. Это ограничение получает имя POST_PR. Для генерации уникальных значений этого столбца используется предварительно созданная с помощью оператора CREATE SEQUENCE последовательность KOD_OP:

CREATE SEQUENCE KOD_OP;

Поскольку столбец КОД_КНИГИ должен содержать только те значения, которые присутствуют в соответствующем столбце таблицы KNIGA, необходимо задать соответствующее ограничение на значения столбца КОД_КНИГИ. Это ограничение можно сделать как ограничением столбца, так и ограничением таблицы, задав ему имя POST_FK. Конструкция ON DELETE CASCADE обеспечит при удалении из таблицы KNIGA строк, содержащих значения внешнего ключа, автоматическое удаление строк и из таблицы KNIGA_POSTAVKA. Для столбца ДАТА_ПОСТУПЛЕНИЯ значением по умолчанию устанавливается текущая дата (SYSDATE). Если таблица KNIGA_POSTAVKA была уже создана ранее, то перед повторным созданием ее следует удалить следующим оператором:

DROP TABLE KNIGA_POSTAVKA;

Следующий оператор создает таблицу KNIGA_POSTAVKA с необходимыми ограничениями целостности данных:

```
CREATE TABLE KNIGA_POSTAVKA (
КОД_ОПЕРАЦИИ NUMBER(10)
CONSTRAINT POST_PR PRIMARY KEY,
КОД_КНИГИ NUMBER(5),
ПРОДАВЕЦ VARCHAR2(20),
```

```
КОЛ_ЕДИНИЦ  NUMBER(4),
ДАТА_ПОСТУПЛЕНИЯ  DATE  DEFAULT  SYSDATE,
CONSTRAINT  POST_FK  FOREIGN  KEY(КОД_КНИГИ)
REFERENCES  KNIGA(КОД_КНИГИ)  ON  DELETE  CASCADE);
```

Вставка строк в таблицу KNIGA_POSTAVKA осуществляется использованием следующей совокупности операторов INSERT:

```
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 1,
'Иванов', 5, '20-JAN-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 2,
'Иванов', 5, '20-JAN-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 3,
'Иванов', 5, '10-FEB-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 4,
'Иванов', 5, '10-FEB-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 5,
'Иванов', 5, '10-FEB-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 6,
'Петров', 4, '25-JAN-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 7,
'Петров', 4, '25-JAN-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 8,
'Петров', 4, '20-FEB-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 9,
'Петров', 4, '20-FEB-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL,
10, 'Петров', 4, '20-FEB-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL,
11, 'Сидоров', 3, '25-JAN-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL,
12, 'Сидоров', 3, '25-JAN-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL,
13, 'Сидоров', 3, '25-JAN-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 7,
'Сидоров', 3, '20-FEB-2004');
INSERT INTO KNIGA_POSTAVKA VALUES(KOD_OP.NEXTVAL, 4,
'Сидоров', 3, '20-FEB-2004');
```

Просмотр введенных значений можно выполнить с помощью следующего оператора SQL:

```
SELECT * FROM  KNIGA_POSTAVKA;
```

З а м е ч а н и е. Пример строк, вызывающих нарушение ограничений целостности данных для таблицы KNIGA_POSTAVKA:

а) Вводится строка с повторением значения первичного ключа – 12:

```
INSERT INTO KNIGA_POSTAVKA VALUES(12, 12, 'Сидоров', 3, '25-
JAN-2004');
```

б) Вводится строка, в которой указан отсутствующий в таблице KNIGA код книги – 15:

```
INSERT INTO KNIGA_POSTAVKA VALUES(16, 15, 'Сидоров', 3, '25-
JAN-2004');
```

2. Создать новую таблицу KNIGA1, структура которой идентична структуре таблицы KNIGA, и перенести в нее строки с информацией о книгах жанра «Фантастика» и «Роман» из таблицы KNIGA.

Следующий набор операторов SQL создает таблицу KNIGA1 и переносит в нее строки из таблицы KNIGA:

```
CREATE  TABLE  KNIGA1 (КОД_КНИГИ  NUMBER(5)
CONSTRAINT  PK_KN1  PRIMARY  KEY,
НАЗВАНИЕ  VARCHAR2(25)  CONSTRAINT  NN_KN1  NOT  NULL,
АВТОР  VARCHAR2(15),
ЦЕНА  NUMBER(7)  CONSTRAINT CEN_KN1 CHECK(ЦЕНА > 100),
ИЗДАТЕЛЬСТВО  VARCHAR2(15),
ЖАНР  VARCHAR2(15));

INSERT  INTO  KNIGA1  SELECT  КОД_КНИГИ,  НАЗВАНИЕ,
АВТОР, ЦЕНА, ИЗДАТЕЛЬСТВО, ЖАНР  FROM  KNIGA  WHERE
ЖАНР = 'Фантастика'  OR  ЖАНР = 'Роман';
```

Аналогичные действия выполняются следующим оператором:

```
CREATE  TABLE  KNIGA1 AS
SELECT  КОД_КНИГИ, НАЗВАНИЕ, АВТОР, ЦЕНА,
ИЗДАТЕЛЬСТВО, ЖАНР  FROM  KNIGA  WHERE
ЖАНР = 'Фантастика'  OR  ЖАНР = 'Роман';
```

Просмотр введенных значений можно выполнить с помощью следующего оператора SQL:

```
SELECT * FROM  KNIGA1;
```

3. Выполнить с помощью оператора UPDATE следующие изменения в таблице KNIGA1:

а) Заменить в таблице KNIGA1 в строке с фамилией автора «Кристи» в столбце АВТОР фамилию «Кристи» на «Агата Кристи» и в столбце ЖАНР жанр «Роман» на жанр «Детектив»;

б) Заменить в таблице KNIGA1 цену книги «Гибель Титана» на цену книги «Казаки».

Следующий набор операторов выполнит указанные действия:

```
UPDATE KNIGA1 SET ЖАНР='Детектив', АВТОР='Агата Кристи'
WHERE АВТОР='Кристи';

UPDATE KNIGA1 SET ЦЕНА=
(SELECT ЦЕНА FROM KNIGA1 WHERE НАЗВАНИЕ='Казаки')
WHERE НАЗВАНИЕ='Гибель Титана';
```

4. Удалить из таблицы KNIGA1 все строки, содержащие информацию о книгах в жанре «Роман». Используем следующий оператор:

```
DELETE KNIGA1 WHERE ЖАНР='Роман';
```

5. Добавить в таблицу KNIGA1 новый столбец ЦЕНА_В_У_Е с типом данных NUMBER(7,2) и пересчитать цену книг в условных единицах. Следующие операторы выполнят сначала добавление нового столбца в таблицу, а затем заполнят его соответствующими значениями:

```
ALTER TABLE KNIGA1 ADD ЦЕНА_В_У_Е NUMBER(7,2);
UPDATE KNIGA1 SET ЦЕНА_В_У_Е=ЦЕНА/2155;
```

З а м е ч а н и е. При добавлении нескольких новых столбцов в таблицу они заключаются в скобки:

```
ALTER TABLE KNIGA1 ADD (ГОД_ИЗДАНИЯ CHAR(4),
МЕСТО_ИЗДАНИЯ CHAR(15));
```

Для изменения характеристик столбца указывается ключевое слово MODIFY, имя столбца и новые характеристики:

```
ALTER TABLE KNIGA1 MODIFY (ГОД_ИЗДАНИЯ NUMBER(4),
МЕСТО_ИЗДАНИЯ CHAR(25));
```

Для добавления ограничения целостности данных необходимо указать ключевое слово CONSTRAINT, имя ограничения и само ограничение. При этом необходимо помнить, что добавляемое ограничение целостности данных не должно противоречить данным, находящимся в таблице.

```
ALTER TABLE KNIGA1 ADD CONSTRAINT ZZ CHECK(ЦЕНА>200);
```

Для временного отключения проверки ограничения целостности с именем ZZ необходимо использовать команду

ALTER TABLE KNIGA1 DISABLE CONSTRAINT ZZ;

Для включения проверки этого ограничения целостности можно использовать команду

ALTER TABLE KNIGA1 ENABLE CONSTRAINT ZZ;

Для удаления ограничения целостности с именем ZZ необходимо указать ключевые слова DROP и CONSTRAINT и имя удаляемого ограничения

ALTER TABLE KNIGA1 DROP CONSTRAINT ZZ;

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

Задания для самостоятельного выполнения

1. Создать таблицу KNIGA_PRODAGA для задания «Книжный магазин», указав все необходимые ограничения целостности данных, и заполнить ее указанной информацией. Учесть, что данные этой таблицы связаны с данными таблицы KNIGA, поэтому здесь обязательно должно быть указано ограничение, устанавливающее взаимосвязь значений соответствующих столбцов в обеих таблицах.

Перечень и тип данных столбцов таблицы KNIGA_PRODAGA:

Код операции	Number(10)
Код книги	Number(5)
Фамилия продавца	Varchar2(20)
Фамилия покупателя	Varchar2(20)
Дата продажи	Date

Информация таблицы KNIGA_PRODAGA:

Код опер.	Код книги	Фам. прод.	Фам. покуп.	Дата прод.
1	6	Петров	Бондарев	26-02-2004
2	12	Сидоров	Фоменко	26-02-2004
3	7	Петров	Ильюкевич	26-02-2004
4	12	Сидоров	Гончаров	20-03-2004
5	12	Сидоров	Никитенко	20-03-2004
6	8	Петров	Соболева	20-03-2004

7	12	Сидоров	Атаманова	26-02-2004
8	9	Петров	Бондарев	20-03-2004
9	1	Иванов	Соболева	26-02-2004
10	11	Сидоров	Никитенко	20-03-2004
11	13	Сидоров	Фоменко	26-02-2004
12	2	Иванов	Никитенко	20-03-2004
13	10	Петров	Ильюкевич	20-03-2004
14	3	Иванов	Никитенко	26-02-2004
15	4	Сидоров	Фоменко	20-03-2004
16	4	Иванов	Гончаров	20-03-2004
17	7	Сидоров	Никитенко	20-03-2004

---

2. Добавить в таблицу KNIGA1 записи о книгах в жанре «Детектив», выбрав их из таблицы KNIGA.

3. Используя оператор UPDATE, заменить в таблице KNIGA1:

а) жанр «Детектив» на жанр «Фантастика» в строках у писателя с фамилией «Асприн»;

б) жанр у книги «Владимир» на тот же жанр, который указан у книги «Князь Рус».

4. Удалить из таблицы KNIGA1 информацию издательства «Аст».

5. Добавить в таблицу KNIGA1 столбец, содержащий общее количество поступивших в магазин книг и заполнить его значением 100.

Вопросы для самоконтроля

1. Литералы и типы данных языка SQL.
2. Операции сравнения.
3. Логические операции. Трехзначная логика.
4. Операции над множествами.
5. Числовые функции.
6. Функции обработки символьных строк.
7. Функции для работы с датами.
8. Групповые функции.
9. Создание таблиц.
10. Ограничения целостности данных.
11. Модификация информации.
12. Изменение структуры таблиц.

Лабораторная работа 2

ВЫБОР ИНФОРМАЦИИ ИЗ БАЗЫ ДАННЫХ

Цель работы. Построение запросов различного вида для выбора информации из таблиц базы данных. Работа с демонстрационными примерами и самостоятельная работа.

ОСНОВНЫЕ СВЕДЕНИЯ И ДЕМОНСТРАЦИОННЫЕ ПРИМЕРЫ

Оператор SELECT. Построение простейших запросов для выбора информации из одной таблицы

Выбор информации из одной или нескольких таблиц осуществляется при помощи оператора SELECT, упрощенный синтаксис которого имеет следующий вид:

```
SELECT [DISTINCT] {* | {[имя_таблицы.] имя_столбца | выражение }
[псевдоним] [,{[имя_таблицы.] имя_столбца | выражение} [псевдоним]]...}
FROM {имя_таблицы | имя_представления | подзапрос} [псевдоним]
[,{имя_таблицы | имя_представления | подзапрос} [псевдоним]…]
[WHERE  условие]
[GROUP BY выражение1 [, выражение2... ]  [HAVING условие] ]
[{UNION | UNION ALL | INTERSECT | MINUS} SELECT оператор ]
[ORDER BY выражение1 [ASC | DESC]
[, выражение2 [ASC | DESC]] ...];
```

Вся извлеченная информация выводится в окно интерактивного редактора SQL*PLUS.

Построение списка выбора. В операторе SELECT при формировании списка выбора, состоящего из имен столбцов и выражений, для построения выражений могут использоваться имена столбцов таблиц и представлений, литералы, функции, соединяемые знаками арифметических действий. Если необходимо указать имена столбцов, которые имеют

одинаковые идентификаторы в двух разных таблицах, то к этим именам через точку необходимо приписать имя таблицы. При необходимости сложному выражению можно присвоить псевдоним, который будет использован в дальнейшем тексте оператора. Все имена таблиц, имена столбцов которых использовались для построения выражений в списке выбора, обязательно должны быть перечислены в тексте оператора после ключевого слова FROM.

Упорядочение строк. Строки, возвращаемые SELECT-запросом, могут быть упорядочены по возрастанию или убыванию значений определенных выражений. В качестве выражения может использоваться имя столбца. Для реализации этой процедуры используется конструкция ORDER BY, в которой указывается перечень, состоящий из одного или нескольких выражений, разделенных запятыми, по значениям которых и осуществляется упорядочение. По умолчанию извлекаемые строки упорядочиваются по возрастанию значений указанных в перечне выражений. Для задания другого варианта упорядочения строк после каждого выражения или группы выражений в перечне указывается либо ASC (по возрастанию) либо DESC (по убыванию значений), а перечисление этих групп осуществляется через запятую.

Условие выбора строк. В конструкции WHERE при построении условия, которому должны удовлетворять выбираемые строки, можно использовать весь имеющийся в языке SQL набор операций сравнения и логических операций. Для задания конкретного значения в условии можно воспользоваться переменной подстановки, которая позволяет ввести необходимое значение в момент выполнения запроса. Переменная подстановки определяется наличием символа & в начале ее имени.

Подзапросы или вложенные запросы применяются для возврата группы строк или множества значений, которые будут использованы родительским запросом. В зависимости от формы построения он может выполняться либо один раз для родительского запроса, либо один раз для каждой строки, извлеченной родительским запросом. В последнем случае такой подзапрос называется коррелированным подзапросом. Характерным признаком коррелированного подзапроса является наличие в его фразе WHERE ссылок на столбцы родительского запроса.

Объединение строк. В многокомпонентном запросе можно определенным образом объединить в единое целое группы строк, извлекаемые отдельно выполняемыми запросами. Для задания порядка объединения используется одно из ключевых слов конструкции UNION | UNION ALL | INTERSECT | MINUS.

Демонстрационные примеры

1. Выбрать из таблицы KNIGA всю информацию о книгах:

```
SELECT * FROM KNIGA;
```

2. Выбрать из таблицы KNIGA информацию о первых пяти книгах:

```
SELECT * FROM KNIGA WHERE ROWNUM < 6;
```

3. Выбрать из таблицы KNIGA информацию о книгах с указанием фамилии автора, названия и цены и упорядочить ее по возрастанию значений столбцов АВТОР, НАЗВАНИЕ и убыванию значений столбца ЦЕНА; фамилии авторов и названия вывести заглавными буквами:

```
SELECT UPPER(АВТОР), UPPER(НАЗВАНИЕ), ЦЕНА FROM
KNIGA ORDER BY АВТОР, НАЗВАНИЕ ASC, ЦЕНА DESC;
```

4. Выбрать из таблицы KNIGA информацию (фамилия автора, название) о книгах в жанре «Роман»;

```
SELECT АВТОР, НАЗВАНИЕ FROM KNIGA WHERE ЖАНР =
'Роман';
```

5. Выбрать из таблицы KNIGA информацию (фамилия автора, название, жанр) о книгах в жанре не «Роман»;

а)
```
SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA WHERE
ЖАНР != 'Роман';
```

б)
```
SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA WHERE
ЖАНР <> 'Роман';
```

в)
```
SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA
MINUS
SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA
WHERE ЖАНР = 'Роман';
```

6. Выбрать из таблицы KNIGA информацию (фамилия автора, название, цена) о книгах стоимостью больше 260 и меньше 1000;

```
SELECT АВТОР, НАЗВАНИЕ, ЦЕНА FROM KNIGA WHERE
ЦЕНА BETWEEN 260 AND 1000;
```

7. Выбрать из таблицы KNIGA информацию (фамилия автора, название, жанр) о книгах в жанрах «Роман» и «Детектив»:

а)
```
SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA
WHERE ЖАНР = 'Роман' OR ЖАНР = 'Детектив';
```

б) SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA
WHERE ЖАНР IN ('Роман', 'Детектив');

в) SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA
WHERE ЖАНР = 'Роман'
UNION
SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA
WHERE ЖАНР = 'Детектив';

8. Выбрать из таблицы KNIGA информацию (фамилия автора, название, жанр) о книгах жанра «Роман», «Детектив» издательства «Аст»:

а) SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA WHERE
(ЖАНР = 'Роман' OR ЖАНР = 'Детектив') AND
ИЗДАТЕЛЬСТВО = 'Аст';

б) SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA WHERE
ЖАНР IN ('Роман', 'Детектив') AND ИЗДАТЕЛЬСТВО = 'Аст';

9. Выбрать из таблицы KNIGA информацию (фамилия автора, название) о книгах, название которых начинается со слова «Гибель»:

а) SELECT АВТОР, НАЗВАНИЕ FROM KNIGA
WHERE НАЗВАНИЕ LIKE 'Гибель%';

б) SELECT АВТОР, НАЗВАНИЕ FROM KNIGA
WHERE SUBSTR(НАЗВАНИЕ, 1, 6) = 'Гибель';

10. Выбрать из таблицы KNIGA информацию (фамилия автора, название, жанр) о книгах, относящихся к указанному жанру. Необходимое значение задать, используя переменную подстановки:

SELECT АВТОР, НАЗВАНИЕ, ЖАНР FROM KNIGA
WHERE ЖАНР = '&GANR' ORDER BY АВТОР;

В ответ на запрос, выдаваемый системой, набрать одно из значений столбца ЖАНР (Роман, Фантастика, Детектив).

11. Выбрать из таблицы KNIGA информацию о книгах, имеющихся в других издательствах, того же жанра, что и в издательстве «Аст»:

а) SELECT * FROM KNIGA WHERE ЖАНР IN
(SELECT DISTINCT ЖАНР FROM KNIGA WHERE
ИЗДАТЕЛЬСТВО= 'Аст') AND ИЗДАТЕЛЬСТВО <> 'Аст';

б) SELECT * FROM KNIGA WHERE ЖАНР = ANY
SELECT DISTINCT ЖАНР FROM KNIGA WHERE
ИЗДАТЕЛЬСТВО = 'Аст') AND ИЗДАТЕЛЬСТВО <> 'Аст';

12. Выбрать из таблицы KNIGA список фамилий авторов, чьи книги имеются в каждом из издательств:

```
SELECT АВТОР FROM  KNIGA  WHERE  ИЗДАТЕЛЬСТВО = 'Аст'
INTERSECT
SELECT АВТОР FROM  KNIGA WHERE ИЗДАТЕЛЬСТВО = 'Нова';
```

Самостоятельное упражнение

1. Выбрать из таблицы KNIGA информацию (фамилия автора, название, издательство) о книгах издательства «Аст» и упорядочить ее по возрастанию значений столбцов АВТОР и НАЗВАНИЕ.

Ответ:

Автор	Название	Изд-во
Асприн	Мифы	Аст
Герберт	Дюна	Аст
Герберт	Еретики Дюны	Аст
Кристи	Гибель Титана	Аст
Никитин	Владимир	Аст
Никитин	Ингвар и Ольха	Аст
Никитин	Князь Рус	Аст
Перумов	Гибель Богов	Аст
Перумов	Страсть	Аст

2. Выбрать из таблицы KNIGA информацию (фамилия автора, название, цена) о книгах стоимостью больше 2000 и упорядочить ее по убыванию значений столбца ЦЕНА.

Ответ:

Автор	Название	Цена
Толстой	Казаки	5568
Толстой	Война и мир	4588
Герберт	Еретики Дюны	2668
Кристи	Гибель Титана	2345
Асприн	Мифотолкование	2265

3. Выбрать из таблицы KNIGA информацию (фамилия автора, название, издательство) о книгах авторов Герберт, Перумов, Асприн и упорядочить ее по значениям столбца АВТОР.

Ответ:

Автор	Название	Изд-во
Асприн	Мифы	Аст
Асприн	Мифотолкование	Нова
Герберт	Дюна	Аст
Герберт	Еретики Дюны	Аст
Герберт	Еретики Дюны	Нова
Перумов	Гибель Богов	Аст
Перумов	Страсть	Аст

4. Выбрать из таблицы KNIGA информацию (фамилия автора, название, жанр) о книгах всех авторов, кроме авторов Герберт и Никитин, и упорядочить ее по значениям столбца АВТОР.

Ответ:

Автор	Название	Жанр
Асприн	Мифы	Детектив
Асприн	Мифотолкование	Детектив
Толстой	Казаки	Роман
Толстой	Война и мир	Роман
Кристи	Гибель Титана	Роман
Перумов	Гибель Богов	Фантастика
Перумов	Страсть	Детектив

5. Выбрать из таблицы KNIGA информацию (фамилия автора, название, жанр) о книгах авторов Никитин и Толстой в жанре «Роман» и упорядочить ее по значениям столбца АВТОР.

Ответ:

Автор	Название	Жанр
Толстой	Казаки	Роман
Толстой	Война и мир	Роман
Никитин	Ингвар и Ольха	Роман
Никитин	Князь Рус	Роман

6. Выбрать из таблицы KNIGA информацию (фамилия автора, название) о книгах с названием, начинающимся на «Миф».

Ответ:

Автор	Название
Асприн	Мифы
Асприн	Мифотолкование

7. Выбрать из таблицы KNIGA информацию (фамилия автора, название, цена) о книгах указанного автора (использовать переменную подстановки и при запросе набрать – Никитин).

Ответ:

Автор	Название	Цена
Никитин	Ингвар и Ольха	168
Никитин	Князь Рус	1168
Никитин	Владимир	568

8. Выбрать из таблицы KNIGA всю информацию о книгах автора Перумов того же жанра, что и у автора Герберт.

Ответ:

Код кн.	Название	Автор	Цена	Изд-во	Жанр
3	Гибель Богов	Перумов	345	Аст	Фантастика

9. Выбрать из таблицы KNIGA список названий книг, которые имеются в каждом из издательств.

Ответ: Еретики Дюны.

Группирование строк

Строки, возвращаемые SELECT-запросом, могут быть объединены в группы на основе значений определенного выражения для каждой строки. Примером такого группирования может служить объединение в группы книг одного жанра, информация о которых имеется в таблице KNIGA. Так как в таблице присутствуют книги только трех жанров, то будут сформированы только три группы строк. Применив к каждой группе функцию SUM для столбца, содержащего значение цены, можно получить суммарную величину стоимости книг по каждому жанру. Для осуществления группирования используется конструкция GROUP BY оператора SELECT, в которой указывается перечень, состоящий из одного или нескольких выражений, разделенных запятыми, по значениям ко-

торых и осуществляется группирование. Если оператор SELECT содержит пункт GROUP BY, то список извлекаемых значений ограничен. Он может содержать константы, групповые функции, функцию SYSDATE и выражения, идентичные выражениям, указанным в пункте GROUP BY. На формирование результирующих строк могут быть наложены определенные условия. Для задания такого условия используется ключевое слово HAVING.

Демонстрационные примеры

1. Выбрать из таблицы KNIGA информацию о количестве различных жанров:

```
SELECT COUNT(DISTINCT ЖАНР) FROM KNIGA;
```

2. Выбрать из таблицы KNIGA информацию о количестве, суммарной стоимости и максимальной стоимости имеющихся книг:

```
SELECT COUNT(КОД_КНИГИ), SUM(ЦЕНА), MAX(ЦЕНА) FROM
KNIGA;
```

3. Выбрать из таблицы KNIGA по каждому издательству информацию о количестве и суммарной стоимости изданных им книг, сгруппировав ее по жанрам:

```
SELECT ИЗДАТЕЛЬСТВО, ЖАНР, COUNT(НАЗВАНИЕ),
SUM(ЦЕНА) FROM KNIGA GROUP BY ИЗДАТЕЛЬСТВО, ЖАНР;
```

4. Выбрать из таблицы KNIGA информацию о количестве и средней стоимости (округлив значение средней стоимости до двух знаков после запятой) книг в тех жанрах, где количество различных названий книг не менее 4:

```
SELECT ЖАНР, COUNT(DISTINCT НАЗВАНИЕ),
ROUND(AVG(ЦЕНА), 2) FROM KNIGA GROUP BY ЖАНР
HAVING COUNT(DISTINCT НАЗВАНИЕ) >= 4;
```

5. Выбрать из таблицы KNIGA информацию о минимальной стоимости книг в жанре «Роман»:

```
SELECT MIN(ЦЕНА) FROM KNIGA WHERE ЖАНР = 'Роман';
```

6. Выбрать из таблицы KNIGA информацию о самой дешевой книге в жанре «Роман»:

```
SELECT АВТОР, НАЗВАНИЕ, ЦЕНА FROM KNIGA
```

```
WHERE ЦЕНА = (SELECT MIN(ЦЕНА) FROM KNIGA
WHERE ЖАНР = 'Роман') AND ЖАНР = 'Роман';
```

7. Выбрать из таблицы KNIGA информацию (фамилия автора, название жанр, цена) о книгах, имеющих максимальную стоимость в своем жанре:

а)
```
SELECT АВТОР, НАЗВАНИЕ, ЖАНР, ЦЕНА FROM KNIGA
WHERE (ЦЕНА, ЖАНР) IN (SELECT MAX(ЦЕНА), ЖАНР
FROM KNIGA GROUP BY ЖАНР);
```

б)
```
SELECT АВТОР, НАЗВАНИЕ, KNIGA.ЖАНР, ЦЕНА FROM
KNIGA, (SELECT ЖАНР, MAX(ЦЕНА) МАКС FROM KNIGA
GROUP BY ЖАНР) P1
WHERE ЦЕНА = МАКС AND KNIGA.ЖАНР = P1.ЖАНР;
```

в)
```
SELECT АВТОР, НАЗВАНИЕ, ЖАНР, ЦЕНА FROM KNIGA P1
WHERE ЦЕНА = (SELECT MAX(ЦЕНА) FROM KNIGA
WHERE KNIGA.ЖАНР = P1.ЖАНР);
```

П о я с н е н и е. Третий запрос содержит коррелированный подзапрос. Поскольку в своем условии подзапрос содержит ссылку на столбец родительского запроса, поэтому он будет выполняться один раз для каждой строки, извлекаемой родительским запросом. В первом и во втором вариантах подзапрос не является коррелированным, он выполняется только один раз для родительского запроса.

8. Выбрать из таблицы KNIGA информацию о книгах, стоимостью больше средней стоимости книг:

```
SELECT АВТОР, НАЗВАНИЕ, ЦЕНА FROM KNIGA
WHERE ЦЕНА > (SELECT AVG(ЦЕНА) FROM KNIGA);
```

9. Выдать из таблицы KNIGA список жанров, по которым имеется наибольшее количество различных книг, с указанием количества книг:

```
SELECT ЖАНР, COUNT(DISTINCT НАЗВАНИЕ) FROM KNIGA
GROUP BY ЖАНР HAVING COUNT(DISTINCT НАЗВАНИЕ)=
(SELECT MAX(COUNT(DISTINCT НАЗВАНИЕ)) FROM KNIGA
GROUP BY ЖАНР);
```

10. Выбрать из таблицы KNIGA информацию о книгах (фамилия автора, название), относящихся к жанрам, по которым имеется наибольшее количество различных книг:

```
SELECT АВТОР, НАЗВАНИЕ FROM KNIGA WHERE ЖАНР IN
(SELECT ЖАНР FROM KNIGA GROUP BY ЖАНР HAVING
```

```
COUNT(DISTINCT НАЗВАНИЕ)=
(SELECT MAX(COUNT(DISTINCT НАЗВАНИЕ)) FROM KNIGA
GROUP BY ЖАНР));
```

Самостоятельное упражнение

1. Выбрать из таблицы KNIGA информацию о количестве издательств.

Ответ: 2.

2. Выбрать из таблицы KNIGA информацию о средней и минимальной стоимости имеющихся книг.

Ответ:

Avg(цена)	Min(цена)
1690	168

3. Выбрать из таблицы KNIGA информацию о количестве и минимальной стоимости книг, предлагаемых каждым издательством.

Ответ:

Изд-во	Количество	Min(цена)
Аст	9	168
Нова	4	2265

4. Выбрать из таблицы KNIGA информацию о максимальной и минимальной стоимости книг издательства «Аст» по тем жанрам, по которым минимальная стоимость книг больше 200.

Ответ:

Жанр	Max(цена)	Min(цена)
Детектив	1385	266
Фантастика	368	268

5. Выбрать из таблицы KNIGA по каждому автору информацию о количестве имеющихся его книг и их средней стоимости, сгруппировав ее по жанрам и округлив значение средней стоимости до двух знаков после запятой.

Ответ:

Автор	Жанр	Количество	Avg(цена)

Асприн	Детектив	2	1265.5
Герберт	Фантастика	3	1101.33
Толстой	Роман	2	5078
Кристи	Роман	1	2345
Никитин	Детектив	1	568
Никитин	Роман	2	668
Перумов	Детектив	1	1385
Перумов	Фантастика	1	345

6. Выбрать из таблицы KNIGA информацию (фамилия автора, название, цена) о самом дешевом фантастическом романе.

Ответ:

Автор	Название	Цена	Жанр
Герберт	Дюна	268	Фантастика

7. Выбрать из таблицы KNIGA информацию (фамилия автора, название, цена, жанр) о самой дорогой книге издательства «Аст».

Ответ:

Автор	Название	Цена	Жанр
Кристи	Гибель Титана	2345	Роман

8. Выбрать из таблицы KNIGA информацию (фамилия автора, название, цена) о книгах минимальной стоимости в каждом издательстве;

Ответ:

Автор	Название	Цена	Изд-во
Никитин	Ингвар и Ольха	168	Аст
Асприн	Мифотолкование	2265	Нова

9. Выбрать из таблицы KNIGA информацию (фамилия автора, название, жанр, цена) о книгах, стоимостью больше средней стоимости книг своего жанра.

Ответ:

Автор	Название	Жанр	Цена

Перумов	Страсть	Детектив	1385
Асприн	Мифотолкование	Детектив	2265
Толстой	Война и мир	Роман	4588
Толстой	Казаки	Роман	5568
Герберт	Еретики Дюны	Фантастика	2668

---

10. Выбрать из таблицы KNIGA список издательств, выпустивших наибольшее количество книг.

Ответ: АСТ.

Выбор информации из нескольких таблиц (соединение)

Соединение – это SELECT-запрос, который выбирает строки из двух или более таблиц. При этом запрос может извлекать любые столбцы из любой таблицы. Если хотя бы две из этих таблиц имеют одинаково названные столбцы, то имена таких столбцов должны уточняться именами таблиц, записываемых перед именами столбцов через точку. Большинство SELECT-запросов с соединениями содержат условия, в которых сравниваются значения столбцов из разных таблиц. Такие условия называются условиями соединения.

Эквисоединение. Это соединение с использованием в условии соединения операции равенства. Таким образом, эквисоединение извлекает строки с эквивалентными значениями в указанных столбцах.

Декартово произведение. Если в запросе не указано условие соединения, то строится декартово произведение таблиц, т. е. к каждой строке первой таблицы приписывается каждая строка второй таблицы.

Самосоединение. Соединяет таблицу саму с собой. При этом таблица появляется в списке FROM дважды и должна иметь дополнительное имя (псевдоним), чтобы можно было однозначно идентифицировать столбцы в условии соединения.

Внешнее соединение. Выдает все строки, которые удовлетворяют условию соединения, а также строки одной из таблиц, которые не удовлетворяют условию соединения. Чтобы записать запрос, который выполняет внешнее соединение таблиц A и B и выдает все строки из таблицы A, применим операцию внешнего соединения (+) ко всем столбцам из таблицы B в условиях соединения. Тогда для всех строк из таблицы A, для которых нет соответствующих строк в таблице B,

ORACLE предоставит строку, содержащую NULL во всех выражениях в списке столбцов, которые содержат столбцы из таблицы B.

Демонстрационные примеры

1. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA информацию о книгах, поставленных продавцам за период с 24.01.2004 по 12.02.2004, указав для выводимого значения даты специальный формат вывода:

```
SELECT ПРОДАВЕЦ, АВТОР, НАЗВАНИЕ,
TO_CHAR(ДАТА_ПОСТУПЛЕНИЯ, 'DD MONTH YYYY')
FROM KNIGA, KNIGA_POSTAVKA WHERE
KNIGA.КОД_КНИГИ = KNIGA_POSTAVKA.КОД_КНИГИ AND
ДАТА_ПОСТУПЛЕНИЯ BETWEEN '24-JAN-04' AND '12-FEB-04'
ORDER BY ПРОДАВЕЦ, АВТОР;
```

2. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA по указанному продавцу – перечень издательств и жанров имеющихся у него книг без повторения; для задания фамилии продавца использовать переменную подстановки:

```
SELECT DISTINCT ПРОДАВЕЦ, ИЗДАТЕЛЬСТВО, ЖАНР
FROM KNIGA, KNIGA_POSTAVKA
WHERE KNIGA_POSTAVKA.КОД_КНИГИ = KNIGA.КОД_КНИГИ
AND ПРОДАВЕЦ = '&PRODAVES'
ORDER BY ИЗДАТЕЛЬСТВО, ЖАНР;
```

3. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA список продавцов, у которых в наличии более 20 книг, указав данные об общем количестве и суммарной стоимости имеющихся у них книг:

```
SELECT ПРОДАВЕЦ, SUM(КОЛ_ЕДИНИЦ),
SUM(ЦЕНА*КОЛ_ЕДИНИЦ) FROM KNIGA, KNIGA_POSTAVKA
WHERE KNIGA_POSTAVKA.КОД_КНИГИ = KNIGA.КОД_КНИГИ
GROUP BY ПРОДАВЕЦ HAVING SUM(КОЛ_ЕДИНИЦ) >= 20;
```

4. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA по каждому издательству информацию об общем количестве и суммарной стоимости поставленных ими книг каждому продавцу:

```
SELECT ИЗДАТЕЛЬСТВО, ПРОДАВЕЦ, COUNT(КОЛ_ЕДИНИЦ),
SUM(ЦЕНА*КОЛ_ЕДИНИЦ) FROM KNIGA, KNIGA_POSTAVKA
WHERE KNIGA.КОД_КНИГИ = KNIGA_POSTAVKA.КОД_КНИГИ
GROUP BY ИЗДАТЕЛЬСТВО, ПРОДАВЕЦ;
```

5. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA по каждой книге, сведения о которой имеются в таблице KNIGA, информацию о количестве продавцов, которым она была поставлена:

```
SELECT KNIGA.КОД_КНИГИ, COUNT(ПРОДАВЕЦ) FROM
KNIGA, KNIGA_POSTAVKA WHERE
KNIGA_POSTAVKA.КОД_КНИГИ
(+)= KNIGA.КОД_КНИГИ GROUP BY KNIGA.КОД_КНИГИ;
```

6. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA по каждому продавцу информацию об отсутствующих у них книгах, общий перечень которых находится в таблице KNIGA:

```
SELECT ПРОДАВЕЦ, НАЗВАНИЕ, ЦЕНА FROM KNIGA,
KNIGA_POSTAVKA
MINUS
SELECT ПРОДАВЕЦ, НАЗВАНИЕ, ЦЕНА FROM KNIGA,
KNIGA_POSTAVKA WHERE KNIGA_POSTAVKA.КОД_КНИГИ =
KNIGA.КОД_КНИГИ;
```

7. Выбрать из таблицы KNIGA информацию (название, цена) о трех самых дешевых книгах; предполагается, что в перечне книг не более трех различных книг с минимальной ценой:

```
SELECT A.НАЗВАНИЕ, A.ЦЕНА FROM KNIGA A, KNIGA B
WHERE A.ЦЕНА >= B.ЦЕНА
GROUP BY A.НАЗВАНИЕ, A.ЦЕНА HAVING
COUNT(B.НАЗВАНИЕ) <= 3 ORDER BY A.НАЗВАНИЕ;
```

8. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA информацию (название и фамилию автора) о книгах, которые не были поставлены в магазин для продажи:

```
SELECT НАЗВАНИЕ, АВТОР FROM KNIGA WHERE
NOT EXISTS
(SELECT * FROM KNIGA_POSTAVKA WHERE
KNIGA_POSTAVKA.КОД_КНИГИ = KNIGA.КОД_КНИГИ);
```

Самостоятельное упражнение

1. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA информацию о книгах, поставленных в магазин, начиная с указанной даты (10.02.2004).

Ответ:

```
Автор          Название                Дата_поступления
-------------------------------------------------------------------
```

Перумов	Страсть	20-FEB-04
Асприн	Мифотолкование	20-FEB-04
Толстой	Война и мир	20-FEB-04
Никитин	Князь Рус	20-FEB-04
Асприн	Мифы	20-FEB-04

---

2. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA по каждому продавцу список имеющихся у него книг, упорядочив ее по издательствам и жанрам.

Ответ:

Продавец	Изд-во	Жанр	Автор	Название
Иванов	Аст	Детектив	Асприн	Мифы
Иванов	Аст	Роман	Никитин	Ингвар и Ольха
Иванов	Аст	Фантастика	Герберт	Дюна
Иванов	Аст	Фантастика	Перумов	Гибель Богов
Иванов	Нова	Роман	Толстой	Казаки
Петров	Аст	Детектив	Перумов	Страсть
Петров	Аст	Роман	Никитин	Князь Рус
Петров	Аст	Фантастика	Герберт	Еретики Дюны
Петров	Нова	Детектив	Асприн	Мифотолкование
Петров	Нова	Роман	Толстой	Война и мир
Сидоров	Аст	Детектив	Асприн	Мифы
Сидоров	Аст	Детектив	Никитин	Владимир
Сидоров	Аст	Роман	Кристи	Гибель Титана
Сидоров	Аст	Роман	Никитин	Князь Рус
Сидоров	Нова	Фантастика	Герберт	Еретики Дюны

---

3. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA фамилии продавцов, имеющих книги в жанре «Фантастика», указав названия этих книг и фамилии авторов.

Ответ:

Продавец	Автор	Название	Жанр
Иванов	Герберт	Дюна	Фантастика
Иванов	Перумов	Гибель Богов	Фантастика
Петров	Герберт	Еретики Дюны	Фантастика
Сидоров	Герберт	Еретики Дюны	Фантастика

---

4. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA по указанному продавцу информацию об общем количестве и суммарной стоимости поставленных ему книг. Использовать переменную подстановки и при запросе набрать – Сидоров.

Ответ:

Продавец	Sum(кол_единиц)	Sum(цена*кол_единиц)
Сидоров	15	21045

5. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA информацию о продавце, у которого присутствует максимальное количество различных книг в жанре «Фантастика».

Ответ:

Продавец	Количество
Иванов	2

6. Выбрать из таблиц KNIGA и KNIGA_POSTAVKA информацию о продавцах, у которых имеется максимальное количество различных книг в жанре «Фантастика», и список этих книг.

Ответ:

Продавец	Автор	Название	Цена	Жанр
Иванов	Герберт	Дюна	268	Фантастика
Иванов	Перумов	Гибель Богов	345	Фантастика

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

Задания для самостоятельного выполнения

Используя таблицы KNIGA и KNIGA_PRODAGA сформировать запросы для получения следующей информации.

1. Выдать по каждому из продавцов перечень фамилий покупателей без повторений.
2. Выдать сведения о количестве покупателей у каждого продавца.
3. Выдать фамилии продавцов, у которых количество покупателей больше трех.
4. Выдать по каждому продавцу список купленных у него книг;

5. Выдать информацию о максимальном количестве книг, купленных у одного продавца.
6. Выдать фамилию продавца, продавшего наибольшее количество книг.
7. Выдать по каждому автору – список названий проданных книг с указанием даты продажи, фамилий авторов и продавцов.
8. Выдать по указанному жанру, используя переменную подстановки (Роман, Фантастика) – список фамилий покупателей, купивших эти книги на указанную дату (26.02.2004, 20.03.2004).
9. Выдать по каждому из продавцов сведения об общем количестве проданных книг, общей и средней выручке, начиная с указанной даты. Использовать переменную подстановки и при запросе набрать – 26.02.2004.
10. Выдать по каждому издательству сведения о количестве, общей и средней стоимости проданных книг, сгруппировав их по жанрам.
11. Выдать по покупателю, купившему наибольшее количество книг, его фамилию и количество купленных книг.
12. Выдать по книге, купленной максимальное количество раз, название и количество проданных экземпляров.
13. Выдать список книг, не пользующихся спросом.

Вопросы для самоконтроля

1. Назначение оператора SELECT.
2. Формирование списка выбора.
3. Упорядочение выбираемых строк.
4. Построение условий отбора строк в запросе.
5. Объединение групп строк, извлеченных несколькими запросами.
6. Группирование выбираемых строк.
7. Виды соединений.
8. Использование подзапросов.
9. Коррелируемый подзапрос.

Лабораторная работа 3

ВВЕДЕНИЕ В ЯЗЫК PL/SQL

Цель работы. Знакомство с основными типами данных, основными операторами языка PL/SQL и структурой программы. Использование курсоров. Обработка исключительных ситуаций. Работа с демонстрационными примерами и самостоятельная работа.

ОСНОВНЫЕ СВЕДЕНИЯ И ДЕМОНСТРАЦИОННЫЕ ПРИМЕРЫ

Основы языка PL/SQL

PL/SQL – это процедурный, блочно-структурированный язык программирования, являющийся расширением языка SQL. Он представляет ряд возможностей, которые обеспечивают возможность создания больших, многофункциональных приложений для работы с базой данных.

Алфавит и лексемы языка. Алфавит языка включает следующий набор символов:

1) английские буквы верхнего и нижнего регистров A..Z, a..z;

2) арабские цифры 0..9;

3) символы + - * / < > = ; : . , ' ~ ! @ # $ % ^ & _ | ( ) { } [ ]

4) символы табуляции, пробелы и символы возврата каретки.

Лексемы, группы символов алфавита, делятся на идентификаторы, литералы, разделители и комментарии.

Идентификаторы имеют длину до тридцати символов и состоят из прописных и строчных букв, цифр и знака подчеркивания, причем первой должна быть буква. Допускается, но не рекомендуется использовать специальные символы, такие как #, $. Некоторые из идентификаторов в языке PL/SQL имеет специальное синтаксическое значение. Такие идентификаторы называются зарезервированными и не должны переопределяться.

Литералы – это явно заданное число, символ, строка или логическое значение, не представленное идентификатором. Литералы делятся на числовые, строковые и логические.

Числовые литералы бывают двух типов: целые и действительные. Целые литералы – это знаковые числа без десятичной точки (6 ; -14). Действительные литералы – знаковые целые или дробные числа с десятичной точкой (6.667 ; -12.0).

Строковые литералы – это последовательность символов, заключенных в одинарные кавычки (апострофы). Все строковые литералы, за исключением пустой строки ('') имеют тип CHAR. Если в строковом литерале необходимо указать одинарную кавычку, то при записи она просто удваивается.

Логические литералы – это предопределенные значения TRUE, FALSE и NULL. NULL указывает на неизвестное значение.

Разделитель – это совокупность одного или двух символов, которая имеет определенное значение в PL/SQL. Простые разделители содержат только один символ. К ним относятся знаки арифметических операций (+ , – , * , /), знаки операций отношения (= , > , <), признак конца выражения (;). Составные разделители содержат два символа. К ним относятся оператор присваивания (:=), оператор конкатенации (| |), операция возведения в степень (**), начало и конец метки (<< >>), оператор диапазона (..), операция отношения неравно (<>, !=, ~=, ^=), операция отношения меньше или равно (<=), операция отношения больше или равно (>=).

Комментарии содержат пояснительный текст и делятся на однострочные и многострочные. Однострочный комментарий представляет собой строку, начинающуюся с двух символов дефис (-). В многострочном комментарии текст заключается в специальные разделители: /* */.

Программа PL/SQL. Представляет собой набор блоков PL/SQL, рекурсивно вложенных друг в друга.

Структура блока:

```
[<<метка>>]
[DECLARE
   раздел объявлений]
BEGIN
   исполняемый раздел
[EXEPTION
   раздел обработки исключений]
END[<<метка>>];
```

Обязательным должен быть только исполняемый раздел, содержащий операторы языка. Существуют следующие типы блоков: анонимные, именованные, триггеры и подпрограммы (процедуры, функции, пакеты).

Типы данных и объявление переменных. Объявление переменных осуществляется в разделе объявлений, при этом помимо идентификатора переменной должен быть указан и ее тип. К основным типам данных языка PL/SQL относятся скалярные и составные типы. Среди составных типов наибольший интерес представляет тип RECORD (запись).

Объявление скалярных типов данных. Среди скалярных типов наиболее распространены числовые, символьные, тип DATE и логический (BOOLEAN) типы данных.

Числовые типы данных представлены в основном двумя типами: BINARY_INTEGER и NUMBER. Все другие числовые типы являются подмножествами двух этих типов.

Тип переменных BINARY_INTEGER используется для хранения целых знаковых чисел в двоичном виде. Диапазон этого типа от

– 2147483647 до 2147483647.

Тип переменных NUMBER используется для хранения чисел с фиксированной и плавающей точкой в диапазоне от 1E–130 до 10E125. Для объявления чисел с плавающей точкой можно просто указать NUMBER. Для объявления целого числа указывается NUMBER(точность), а для объявления числа с фиксированной точкой дополнительно указывается еще и масштаб, т. е. NUMBER(точность, масштаб). Точность представляет собой общее число знаков и не превосходит 38 десятичных знаков, масштаб указывает порядок округления и задается числом от –84 до 127. При положительном значении масштаба число округляется до указанного количества цифр, стоящих справа от запятой; при отрицательном значении число округляется до указанного количества цифр, стоящих слева от запятой. Например: число 123.456 при значении масштаба, равном 2, округляется до 123.46, а при значении, равном – 2, округляется до 100.

К символьным типам данных в основном относятся типы CHAR и VARCHAR2.

Тип CHAR(длина) используется для хранения последовательности символов фиксированной длины не более 32767 байт. Следует иметь в виду, что в языке SQL максимальное значение аналогичного типа равно 2000 байт, поэтому не все данные этого типа можно вставлять в столбцы таблицы с типом CHAR.

Тип VARCHAR2(длина) используется для хранения символьной последовательности переменной длины. Ограничение на длину составляет 32767 байт.

Тип *DATE* используется для хранения значений даты и времени. Значение даты и времени хранится во внутреннем двоичном формате и при помещении его в переменную символьного типа автоматически преобра-

зуется в строку, используя формат даты, установленный по умолчанию. Функция SYSDATE возвращает текущее значение даты и времени.

Тип *BOOLEAN* используется для хранения логических значений TRUE, FALSE, NULL. Над такими переменными можно выполнять только логические операции, причем в трехзначной логике.

Скалярные переменные объявляются явным и неявным образом.

Явное объявление переменной любого из скалярных типов осуществляется по следующему правилу:

имя_переменной [CONSTANT] тип [NOT NULL] [{:= |DEFAULT} значение];

При использовании ключевого слова CONSTANT переменной должно быть присвоено значение, которое впоследствии не может быть изменено. Если указано ключевое слово NOT NULL, то переменную необходимо проинициализировать и впоследствии она не может принимать значение NULL. Переменная инициализируется значением своего типа либо с помощью оператора присваивания, либо с помощью ключевого слова DEFAULT. Каждая переменная объявляется отдельно.

Неявное объявление переменной скалярного типа осуществляется с помощью специального атрибута %TYPE, который позволяет объявить переменную, тип которой соответствует либо типу другой переменной, либо типу столбца таблицы базы данных.

Пример.

```
A1      NUMBER;
A11     NUMBER:=15;
A12     NUMBER NOT NULL DEFAULT 15;
A2      A1%TYPE;
A3      KNIGA.ЦЕНА%TYPE;
```

Объявляя переменные, следует иметь в виду, что:

1) имена локальных переменных и формальных параметров имеют приоритет перед именами таблиц базы данных;

2) имена столбцов таблицы базы данных имеют приоритет перед именами локальных переменных и формальных параметров.

Объявление переменных типа RECORD (запись). Составной тип запись определяет структуру, содержащую любое число переменных – элементов любого типа, включая ранее определенные записи. Ссылка на отдельные элементы этой структуры осуществляется с помощью точечной нотации.

Тип записи должен быть определен до того, как будут объявлены переменные этого типа. Для явного определения нового составного типа данных используется следующий общий синтаксис:

```
TYPE тип_записи IS RECORD
(поле1 тип1 [NOT NULL] [{:= | DEFAULT } значение1],
 поле2 тип2 [NOT NULL] [{:= | DEFAULT } значение2], …);
```

Для явного объявления переменной-записи этого типа необходимо указать имя переменной и имя типа. Допускается неявное определение переменных типа запись, выполняемое с помощью атрибута %ROWTYPE, что позволяет определять переменные-записи, структура которых идентична структуре записи указанной таблицы или структуре ранее определенной переменной-записи.

Рассмотрим переменную-запись, объявляемую явно:

```
DECLARE
TYPE BOOK IS RECORD  --вводится тип записи – BOOK
(
AUTHOR  VARCHAR2(15),    --фамилия автора
NAME    VARCHAR2(25),    --название книги
PRICE   NUMBER(6)        --цена
);
BOOK_FAN   BOOK;         --явное объявление
BOOK_ROM   BOOK;         --явное объявление
BOOK_POEM  BOOK_FAN%ROWTYPE;  --неявное объявление
```

Присваивание значений элементам, входящим в запись, необходимо выполнять поэлементно, используя точечную нотацию:

```
BOOK_FAN.PRICE := 16000;
```

Для присвоения значений сразу всем полям записи или нескольким из них можно воспользоваться однострочным оператором SELECT либо оператором выборки очередной строки из открытого курсора FETCH:

Возможно присвоение значения одной переменной-записи другой при условии, что они одного типа:

```
BOOK_FAN := BOOK_ROM;
```

Однако следует иметь в виду, что переменная-запись, тип которой определен явно, и переменная-запись, тип которой определяется с помощью атрибута %ROWTYPE, всегда несовместимы.

Не допускается сравнение переменных-записей.

Ошибочной будет попытка передать в качестве значений в команде INSERT запись целиком. Значения записи должны передаваться поэлементно:

```
INSERT  INTO  TEMP VALUES (BOOK_FAN.AUTHOR,
BOOK_FAN.NAME, BOOK_FAN.PRICE);
```

Операторы PL/SQL. Представлены оператором присваивания, условным оператором и оператором цикла.

Оператор присваивания позволяет задать переменной некоторое значение и имеет следующий синтаксис:

```
переменная:=выражение;
```

Условный оператор IF, позволяющий проверить некоторый набор условий и выполнить соответствующие действия, имеет следующий синтаксис:

```
IF  логическое выражение1  THEN
        последовательность операторов1
ELSIF  логическое выражение2  THEN
        последовательность операторов2
ELSE
        последовательность операторов3
ENF IF;
```

Оператор цикла LOOP позволяет повторить выполнение заданной последовательности операторов необходимое количество раз. Существуют три формы записи оператора цикла LOOP.

В первом варианте условие завершения цикла находится внутри тела цикла и формируется с помощью ключевых слов EXIT и WHEN.

```
LOOP
   последовательность операторов
   [EXIT  [WHEN  условие]];
END LOOP;
```

Во втором варианте повторение операторов осуществляется до тех пор, пока остается истинным условие, указанное в начальной строке оператора LOOP. Дополнительно может быть сформировано еще одно условие выхода из цикла при помощи ключевого слова EXIT.

```
WHILE  условие  LOOP
   последовательность операторов
   [EXIT  [WHEN  условие]];
```

END LOOP;

В третьем варианте переменная цикла пробегает указанный диапазон значений от нижней границы до верхней, увеличивая каждый раз свое значение на единицу, после чего осуществляется выход из цикла.

```
FOR   переменная_цикла
IN [REVERSE]   нижняя_граница..верхняя_граница
LOOP
   последовательность операторов
END LOOP;
```

Переменная цикла определяется системой неявно как переменная типа BINARY_INTEGER и не требует объявления. Значениями границ могут быть переменные, константы, выражения. Вариант REVERSE означает, что значения просматриваются в обратном порядке от верхней границы к нижней.

Курсоры. Для выполнения команды SELECT система выделяет определенную область, куда помещает помимо служебной информации и сами выбранные строки (активный набор строк). В PL/SQL имеется специальная конструкция, которая позволяет задать имя этой области и осуществить доступ к хранящейся там информации. Такая конструкция называется курсором. Существуют курсоры двух типов: явные и неявные.

Явные курсоры. Явный курсор должен быть явно объявлен пользователем. Для его объявления используется следующий синтаксис:

CURSOR имя_курсора [(список_параметров)] IS SELECT…;

Оператор SELECT определяет набор извлекаемых строк. Список параметров может отсутствовать. Если же параметры используются, то они описываются следующим образом:

Имя_параметра тип [{:=| DEFAULT} значение];

Если указано значение параметра, то при открытии курсора этот параметр можно не указывать.

Управление явным курсором осуществляется в двух вариантах: явно и неявно.

При явном управлении курсором необходимо явно открыть курсор, явно выбрать строки из активного набора и явно закрыть курсор. При неявном управлении эти операции выполняются самой системой.

Явная форма управления курсорам. Для того чтобы явно открыть курсор, используется следующая конструкция:

OPEN имя_курсора [(список_параметров)];

В момент открытия курсора система выполняет указанный оператор SELECT с учетом передаваемых значений параметров, т. е. выбирает соответствующий набор строк, и указатель текущей записи устанавливается в этом наборе на первую строку. Однако строки программе не передаются. Чтобы получить строки одну за другой, используется оператор FETCH. Выборка строк допускается только в прямом направлении; продвижение по набору строк в обратном направлении невозможно. Синтаксис оператора FETCH:

FETCH имя_курсора INTO {имя_записи | список_столбцов};

При выполнении оператора FETCH выбирается очередная строка, а указатель текущей записи передвигается на следующую строку в наборе строк. Если были выбраны все строки, то при попытке нового считывания ошибка не возникает, но и строка не выбирается.

Для того чтобы закрыть курсор, необходимо выполнить следующий оператор:

CLOSE имя_курсора;

После закрытия курсора все попытки считывания информации приведут к ошибке. Для организации явного управления можно использовать курсорные атрибуты.

Курсорные атрибуты. Для организации явного управления курсором используются курсорные атрибуты. Курсорные атрибуты представляют собой функции, возвращающие определенное значение в зависимости от выполненных действий. К ним относятся:

%ISOPEN – возвращающий значение TRUE, если курсор открыт, и FALSE, если курсор закрыт;

%FOUND – возвращающий значение TRUE, если строка найдена, и FALSE, если строка не найдена;

%NOTFOUND – возвращающий значение TRUE, если строка не найдена, и FALSE, если строка найдена;

%ROWCOUNT – возвращающий числовое значение, показывающее количество выбранных строк в курсоре;

Неявная форма управления курсором. Для автоматической обработки курсора используется специальная форма записи оператора цикл FOR – циклы FOR с курсором.

Открытие, выборка и закрытие курсора в этом случае происходит автоматически. Возвращаемая переменная-строка определяется неявно и на нее нельзя ссылаться извне области видимости цикла. При неявной форме управления курсор может также иметь параметры.

Неявные курсоры. Они создаются и открываются системой при выполнении операторов INSERT, UPDATE и DELETE, а также при выполнении однострочного оператора SELECT. Неявный курсор называется SQL-курсором. Он имеет свои атрибуты, аналогичные атрибутам явного курсора:

SQL%FOUND, SQL%NOTFOUND, SQL%ROWCOUNT

Однако в однострочном операторе SELECT нельзя использовать атрибут SQL%NOTFOUND, потому что, если при его выполнении информация из таблицы не будет выбрана, то сгенерируется исключение NO_DATA_FOUND. Атрибут SQL%NOTFOUND обычно используется в операторах UPDATE и DELETE для проверки, успешно или нет, выполнены соответствующие операторы.

Модификация данных в курсоре. Конструкция курсор позволяет помимо оперирования наборами данных таблиц выполнять и модификацию информации таблиц. Для этого используются две дополнительных конструкции:

FOR UPDATE [OF столбец1, столбец2…] [NOWAIT]

указываемая в операторе SELECT, который определяет строки формируемого активного набора, и

WHERE CURRENT OF имя_курсора

которая указывается в операторах UPDATE или DELETE, выполняемых в процессе выборки и обработки строк из активного набора.

В обычной ситуации строки, выбираемые в активный набор, не блокируются, и любой другой пользователь может осуществить их модификацию в базе данных. Если после этого закрыть и вновь открыть курсор, то получаешь доступ к новым, модифицированным строкам.

При использовании варианта FOR UPDATE с перечнем столбцов блокируется доступ других пользователей к указанным столбцам, но и обновлять информацию можно только в этих столбцах. Если же используется FOR UPDATE без перечня столбцов, то блокируется вся таблица, и соответственно обновлять можно информацию, хранящуюся в любых столбцах таблицы.

Обработка исключительных ситуаций. В PL/SQL предусмотрены механизмы перехвата и обработки ошибок, возникающих при выполнении программы.

Генерация исключительной ситуации с выдачей соответствующего сообщения в рабочую среду в случае обнаружения ошибки может быть выполнена с помощью следующего оператора:

RAISE_APPLICATION_ERROR (errnum, errtext);
где errnum – код ошибки в диапазоне –20000 .. –20999;
errtext - символьная строка длиной до 512 байт.

Демонстрационные примеры

В данной лабораторной работе используется таблица KNIGA, созданная в лабораторной работе 1.

Создать программу, которая осуществляет повышение цен на книги жанра «Фантастика». При этом, при стоимости книги менее 2000, цена увеличивается на 20 %, а при стоимости больше или равной 2000 она увеличивается на 10 %.

П о я с н е н и е. Данная задача реализуется с помощью явно объявленного пользователем курсора. При этом, в первых двух вариантах показываются возможности использования обычного курсора и курсора с параметром. Приведенное решение демонстрирует два варианта обработки явно объявленного курсора: явную и неявную формы. В явной форме обработки по завершении просмотра строк активного набора осуществляется выход из цикла обработки. При этом используется курсорный атрибут %NOTFOUND. В неявной форме обработки курсора используется конструкция цикл FOR с курсором. Модифицированное значение цены записывается обратно в таблицу KNIGA с использованием конструкции WHERE CURRENT OF, при этом системе с помощью конструкции FOR UPDATE OF ЦЕНА указывается, что будет осуществляться обновление значений столбца ЦЕНА таблицы KNIGA. В программе неявным способом объявлены переменная типа запись ZAP и скалярная переменная NEW_CENA.

а) Использование обычного курсора:

```
DECLARE
   CURSOR  KUR IS              --явное объявление курсора KUR
      SELECT  КОД_КНИГИ,  ЦЕНА  FROM  KNIGA
      WHERE  ЖАНР='Фантастика'  FOR  UPDATE  OF  ЦЕНА;
   ZAP  KUR%ROWTYPE;       --объявление переменной-записи
   NEW_CENA  KNIGA.ЦЕНА%TYPE;      --объявление переменной
BEGIN
   OPEN  KUR;                                  --явное открытие курсора
```

```
    LOOP
      FETCH  KUR  INTO  ZAP;              --выборка текущей записи
      EXIT  WHEN  KUR%NOTFOUND;    --выход из цикла
      IF  ZAP.ЦЕНА < 2000  THEN          --изменение цены
        NEW_CENA := ZAP.ЦЕНА*1.2;
      ELSE
        NEW_CENA := ZAP.ЦЕНА*1.1;
      END  IF;
      UPDATE  KNIGA
        SET  ЦЕНА = NEW_CENA          --обновление цены
        WHERE CURRENT  OF  KUR;
  END  LOOP;
    CLOSE  KUR;                        --явное закрытие курсора
    COMMIT;                            --завершение транзакции
  END;
```

б) Использование курсора с параметром:

```
  DECLARE
    CURSOR  KUR  (GANR  KNIGA.ЖАНР%TYPE) IS  --курсор  имеет параметр
      SELECT  КОД_КНИГИ, ЦЕНА  FROM  KNIGA
      WHERE  ЖАНР=GANR  FOR  UPDATE  OF  ЦЕНА;
    ZAP  KUR%ROWTYPE;
    NEW_CENA  KNIGA.ЦЕНА%TYPE;
  BEGIN
    OPEN  KUR ('Фантастика');          -- значение параметра
      LOOP
        FETCH  KUR  INTO  ZAP;
        EXIT  WHEN  KUR%NOTFOUND;
        IF  ZAP.ЦЕНА < 2000  THEN
          NEW_CENA := ZAP.ЦЕНА*1.2;
        ELSE
          NEW_CENA := ZAP.ЦЕНА*1.1;
        END  IF;
        UPDATE  KNIGA  SET  ЦЕНА = NEW_CENA
          WHERE  CURRENT  OF  KUR;
      END  LOOP;
    CLOSE  KUR;
    COMMIT;
  END;
```

в) Использование цикла FOR с курсором:

```
DECLARE
    NEW_CENA  KNIGA.ЦЕНА%TYPE;
    CURSOR  KUR  IS
       SELECT  КОД_КНИГИ, ЦЕНА  FROM  KNIGA
       WHERE  ЖАНР='Фантастика'  FOR  UPDATE  OF  ЦЕНА;
BEGIN
   FOR  ZAP  IN  KUR  LOOP          --неявная обработка курсора
      IF  ZAP.ЦЕНА < 2000  THEN     --переменная ZAP неявно объявляется системой
         NEW_CENA := ZAP.ЦЕНА*1.2;
      ELSE
         NEW_CENA := ZAP.ЦЕНА*1.1;
      END  IF;
      UPDATE  KNIGA  SET  ЦЕНА = NEW_CENA
         WHERE  CURRENT   OF   KUR;
   END  LOOP;
   COMMIT;
END;
```

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

Задания для самостоятельного выполнения

1. Создать программу, с помощью которой издательство «Аст» осуществляет повышение цен на свои книги, информация о которых хранится в таблице KNIGA. Цена на каждую книгу увеличивается на 30 %, но при этом она не может превысить значение 5000. Реализовать данную программу тремя способами:

1) обычным курсором;
2) курсором с параметром;
3) циклом FOR с курсором.

2. Создать программу, которая рассчитывает и заносит во временную таблицу TEMP сведения о долях количества и суммарной стоимости книг по каждому жанру по отношению к общему количеству книг и общей суммарной стоимости всех книг. Величины указанных долей выразить в процентах. Реализовать данную программу двумя способами:

1) обычным курсором;
2) циклом FOR с курсором.

3. Создать программу, с помощью которой издательство «Нова» осуществляет пропорциональное повышение цен на свои книги, информация о которых хранится в таблице KNIGA. Цена на каждую книгу увеличивается на величину, которая пропорциональна доли стоимости книги по отношению к общей суммарной стоимости книг. Суммарная величина повышения стоимости всех книг равна 15000. Реализовать данную программу тремя способами:

1) обычным курсором;
2) курсором с параметром;
3) циклом FOR с курсором

Вопросы для самоконтроля

1. Лексемы и их классификация.
2. Структура программы в языке PL/SQL.
3. Основные типы скалярных переменных.
4. Тип данных запись.
5. Явное и неявное объявление переменных.
6. Операторы PL/SQL.
7. Явные курсоры и управление ими.
8. Неявные курсоры и их обработка.
9. Обработка исключительных ситуаций.

Лабораторная работа 4

ТРИГГЕРЫ БАЗЫ ДАННЫХ

Цель работы. Создание и отладка триггеров. Включение и выключение триггеров. Удаление триггеров из базы данных. Получение необходимой информации о триггерах. Работа с демонстрационными примерами и самостоятельная работа.

ОСНОВНЫЕ СВЕДЕНИЯ И ДЕМОНСТРАЦИОННЫЕ ПРИМЕРЫ

Триггеры базы данных

Триггер базы данных – это оформленный специальным образом именованный блок PL/SQL, хранящийся в базе данных. Каждый триггер связан с определенной таблицей и автоматически запускается ORACLE при выполнении одного из DML-операторов (INSERT, DELETE, UPDATE) или их совокупности над этой таблицей.

Назначение триггеров. Триггеры могут быть использованы:

1) для реализации сложных ограничений целостности данных, которые не могут быть осуществлены стандартным образом при создании таблицы;
2) предотвращения неверных транзакций;
3) выполнения процедур комплексной проверки прав доступа и секретности данных;
4) генерации некоторых выражений на основе значений, имеющихся в столбцах таблиц;

Создание и включение триггеров. Для создания и автоматического включения триггера применяется следующий общий синтаксис:

```
CREATE [OR REPLACE] TRIGGER имя_триггера
{BEFORE | AFTER}
{INSERT | DELETE | UPDATE [OF список_столбцов]}
ON имя_таблицы [FOR EACH ROW] [WHEN условие]
< PL/SQL_блок >
```

При наличии ключевых слов OR REPLACE триггер создается заново, если он уже существует.

Конструкция BEFORE | AFTER указывает на момент запуска триггера. Вариант BEFORE означает, что триггер будет запускаться перед выполнением активизирующего DML-оператора; вариант AFTER означает, что триггер будет запускаться после выполнения активизирующего DML-оператора.

Конструкция INSERT | DELETE | UPDATE [OF список_столбцов] указывает тип активизирующего триггер DML-оператора. Разрешается, используя логическую операцию OR, задать совокупность активизирующих операторов, например: INSERT OR DELETE. Если при использовании варианта UPDATE указан список столбцов, то триггер будет запускаться при модификации одного из указанных столбцов; если список столбцов отсутствует, то триггер будет запускаться при изменении любого из столбцов связанной с триггером таблицы.

Конструкция FOR EACH ROW указывает на характер воздействия триггера: строковый или операторный. Если конструкция FOR EACH ROW присутствует, то триггер является строковым; при отсутствии ее триггер является операторным. Операторный триггер запускается один раз до или после выполнения активизирующего триггер DML-оператора независимо от того, сколько строк в связанной с триггером таблице подвергается модификации. Строковый триггер запускается один раз для каждой из строк, которая подвергается модификации DML-оператором, активизирующим триггер.

С помощью ключевого слова WHEN можно задать дополнительное ограничение на строки связанной с триггером таблицы, при модификации которых может быть запущен триггер.

Конструкция PL/SQL_блок представляет блок PL/SQL, который ORACLE запускает при активизации триггера.

Классификация триггеров. В основном различают двенадцать типов триггеров. Тип триггера определяется сочетанием следующих трех параметров:

1) характером воздействия триггера на строки связанной с ним таблицы (строковый или операторный);
2) моментом запуска триггера: до (BEFORE) или после (AFTER) исполнения активизирующего триггер DML-оператора;
3) типом активизирующего триггер DML-оператора (INSERT, DELETE, UPDATE);

Порядок активизации триггеров. Если у таблицы имеется несколько типов триггеров, то они активизируются по следующей схеме:

1) выполняется операторный триггер BEFORE (если их несколько, то ничего о порядке их выполнения сказать нельзя);
2) выполняется строковый триггер BEFORE;
3) выполняется активизирующий триггер DML-оператор с последующей проверкой всех ограничений целостности данных;
4) выполняется строковый триггер AFTER с последующей проверкой всех ограничений целостности данных;
5) выполняется операторный триггер AFTER.

Триггерные предикаты. Если в триггере указывается совокупность активизирующих триггер DML-операторов (например, INSERT OR DELETE), то для распознавания того, какой конкретно из DML-операторов выполняется над связанной с триггером таблицей, используются триггерные предикаты: INSERTING, DELETING, UPDATING. Они представляют собой логические функции, возвращающие TRUE, если тип активизирующего оператора совпадает с типом предиката, и FALSE – в противном случае. Для задания одних и тех же действий в случае выполнения различных DML-операторов в условном операторе триггерные предикаты объединяются с помощью логических операций.

Псевдозаписи. Для строковых триггеров существуют специальные конструкции, которые позволяют при выполнении DML-операторов над строкой таблицы, обращаться как к старым значениям, которые находились в ней до модификации, так и к новым, которые появятся в строке после ее модификации. Эти конструкции называются псевдозаписями и обозначаются old и new. Структура этих псевдозаписей идентична структуре строки модифицируемой таблицы, но оперировать можно только отдельными полями псевдозаписи. Обращение к полям псевдозаписи происходит по следующей схеме: перед old или new ставится символ «:», далее через точку указывается название поля. Значения, которые принимают поля псевдозаписи при выполнении активизирующих DML-операторов, определяются следующим образом.

Оператор INSERT – псевдозапись :new эквивалентна вставляемой строке, а псевдозапись :old во всех полях имеет значение NULL.

Оператор DELETE – псевдозапись :old эквивалентна удаляемой строке, а псевдозапись :new во всех полях имеет значение NULL.

Оператор UPDATE – псевдозапись :new эквивалентна строке, полученной в результате модификации, а псевдозапись :old во всех полях имеет исходное значение строки.

Включение, выключение триггеров. Хранящийся в базе данных триггер можно временно отключить, не удаляя его из базы данных. Для этого используется следующая команда:

ALTER TRIGGER имя_триггера DISABLE;

Включить триггер через некоторый промежуток времени можно, используя команду

ALTER TRIGGER имя_триггера ENABLE;

Запретить или разрешить запуск всех триггеров, связанных с некоторой таблицей, можно с помощью команды

ALTER TABLE имя_таблицы {DISABLE | ENABLE} ALL TRIGGERS;

где вариант DISABLE используется для отключения, а вариант ENABLE – для включения всех триггеров данной таблицы.

Удаление триггеров из базы данных. Уничтожение триггера, т. е. удаление триггера из базы данных осуществляется с помощью следующей команды:

DROP TRIGGER имя_триггера;

Получение информации о триггерах. Триггеры хранятся в базе данных, поэтому информацию о них можно получить из представления словаря данных USER_TRIGGERS, например, следующей командой:

SELECT * FROM USER_TRIGGERS;

Демонстрационные примеры

В данной лабораторной работе используются таблицы KNIGA, KNIGA_POSTAVKA и KNIGA_PRODAGA, созданные в лабораторной работе 1.

1. Создать триггер, который перед вставкой очередной строки в таблицу KNIGA_POSTAVKA проверяет наличие указанного кода книги в таблице KNIGA. При отсутствии указанного кода книги в таблице KNIGA должно генерироваться исключение с выдачей соответствующего сообщения.

П о я с н е н и е. Добавление новых строк в таблицу KNIGA_POSTAVKA выполняется оператором INSERT. Поскольку триггер должен запускаться перед выполнением каждого оператора INSERT, следовательно, он должен быть строковым BEFORE-триггером. Для сохранения целостности данных необходимо проверить, имеются ли вносимые коды книг и в таблице KNIGA. Для этого с помощью однострочного оператора SELECT осуществляется выборка информации из таблицы KNIGA, где в условии выборки используется поле КОД_КНИГИ псевдозаписи :new.

Если количество строк с данным кодом книги в таблице KNIGA окажется равным нулю, будет сгенерировано исключение и выдано соответствующее сообщение.

Создание триггера TR1 выполняется вводом следующего оператора:

```
CREATE OR REPLACE TRIGGER TR1
   BEFORE INSERT ON KNIGA_POSTAVKA
   FOR EACH ROW
DECLARE
   KOL NUMBER(4);
BEGIN
   SELECT COUNT(*) INTO KOL FROM KNIGA
      WHERE КОД_КНИГИ = :NEW.КОД_КНИГИ;
   IF KOL = 0 THEN RAISE_APPLICATION_ERROR
      (−20212,'В таблице KNIGA нет информации о данной книге');
   END IF;
END TR1;
```

Действие триггера TR1 может быть проверено выполнением следующей совокупности операторов, осуществляющих вставку строк в таблицу KNIGA_POSTAVKA и тем самым вызывающих активизацию триггера TR2:

```
INSERT INTO KNIGA_POSTAVKA VALUES(21,15,’Иванов’,15,
’20-JAN-2004’);
INSERT INTO KNIGA_POSTAVKA VALUES(22,16,’Иванов’,5,
’20-JAN-2004’);
INSERT INTO KNIGA_POSTAVKA VALUES(23,17,’Иванов’,5,
’10-FEB-2004’);
```

Поскольку указанные коды книг (15, 16, 17) отсутствуют в таблице KNIGA, то во всех трех случаях будет сгенерировано исключение и выдано соответствующее сообщение.

2. Создать триггер, который запрещает вносить в таблицу KNIGA строки со значением поля ЦЕНА больше, чем 5000 рублей, а также осуществлять увеличение цены книг, информация о которых хранится в таблице KNIGA, более чем на 20 %. При нарушении данного требования должно генерироваться исключение с выдачей соответствующего сообщения.

П о я с н е н и е. Так как внесение новых строк в таблицу KNIGA осуществляется в результате выполнения оператора INSERT, а значение поля ЦЕНА в таблице KNIGA, содержащего цену книги, может быть из-

менено в результате выполнения оператора UPDATE, то в триггере указывается совокупность запускающих DML-операторов. Поскольку триггер должен запускаться перед выполнением каждого из указанных DML-операторов, следовательно, он является строковым BEFORE-триггером. Так как действия, выполняемые триггером, различны для каждого из запускающих DML-операторов, модифицирующих таблицу KNIGA, то для распознавания типа DML-оператора используются соответствующие триггерные предикаты INSERTING и UPDAITING. Вследствие того что при вставке новых строк проверке должно быть подвергнуто новое значение поля ЦЕНА, а при модификации значения поля ЦЕНА новое значение должно сравниваться со старым значением, необходимо использовать псевдозаписи :new и :old.

Создание триггера TR2 выполняется вводом следующего оператора:

```
CREATE OR REPLACE TRIGGER TR2
   BEFORE INSERT OR UPDATE OF ЦЕНА ON KNIGA
   FOR EACH ROW
BEGIN
   IF INSERTING THEN
      IF :NEW.ЦЕНА > 5000 THEN
         RAISE_APPLICATION_ERROR
         (-20102, 'В таблицу KNIGA нельзя вносить записи с ценой книги > 5000');
      END IF;
   END IF;
   IF UPDATING THEN
      IF :NEW.ЦЕНА > :OLD.ЦЕНА*1.2 THEN
         RAISE_APPLICATION_ERROR
         (-20103, 'В таблице KNIGA нельзя изменять цену книги более чем на 20 %');
      END IF;
   END IF;
END TR2;
```

Действие триггера TR2 может быть проверено выполнением следующих операторов, которые, осуществляя вставку строк в таблицу KNIGA и обновление строк в таблице KNIGA, тем самым вызывают его активизацию.

Операторы вставки строк в таблицу KNIGA, вызывающие активизацию триггера TR2:

```
INSERT INTO KNIGA VALUES( 21,  'Дюна', 'Герберт', 5268, 'Аст',
 'Фантастика');
INSERT INTO KNIGA VALUES( 22, 'Ингвар и Ольха', 'Никитин',
5168, 'Аст', 'Роман');
INSERT INTO KNIGA VALUES( 23, 'Гибель Богов', 'Перумов', 5345,
'Аст', 'Фантастика ');
INSERT INTO KNIGA VALUES( 24, 'Мифы', 'Асприн',5266, 'Аст',
 'Детектив');
```

Операторы обновления строк в таблице KNIGA, вызывающие активизацию триггера TR2:

```
UPDATE KNIGA SET ЦЕНА=6000 WHERE
ИЗДАТЕЛЬСТВО = 'Аст';
UPDATE KNIGA SET ЦЕНА=6000;
```

Поскольку эти операторы нарушают требования, предъявляемые к значению и модификации цены книг, то во всех случаях будет сгенерировано исключение и выдано соответствующее сообщение.

3. Создать триггер, который в созданную таблицу STAT, содержащую столбцы:

имя издательства – IZD,
количество книг в жанре «Роман» – KOL_ROM,
количество книг в жанре «Фантастика» – KOL_FAN,

при каждой модификации таблицы KNIGA формирует и заносит в соответствующие столбцы таблицы STAT суммарное количество книг по каждому из издательств в разрезе указанных тематик: «Роман» и «Фантастика».

П о я с н е н и е. Модификация таблицы KNIGA осуществляется выполнением следующих DML-операторов: INSERT, DELETE или оператора UPDATE, модифицирующего значения столбца ЖАНР в таблице KNIGA. Так как действия по формированию информации таблицы STAT выполняются после выполнения каждого из модифицирующих таблицу KNIGA операторов, то по типу это операторный AFTER-триггер. Поскольку действия, выполняемые триггером, одинаковы для всех типов активизирующих его операторов, то триггерные предикаты не используются. Перед созданием триггера должна быть создана таблица STAT.

Создание таблицы STAT может быть выполнено вводом следующей совокупности операторов:

```
DROP TABLE STAT;
CREATE TABLE STAT
```

```
( IZD  VARCHAR2(15),
  KOL_ROM  NUMBER(7),
  KOL_FAN  NUMBER(7)
);
```

Создание триггера TR3 выполняется вводом следующего оператора:

```
CREATE  OR  REPLACE  TRIGGER  TR3
    AFTER  INSERT  OR  DELETE  OR  UPDATE  OF  ЖАНР
    ON  KNIGA
DECLARE
   CURSOR  V1  IS  SELECT  ИЗДАТЕЛЬСТВО,
   COUNT(НАЗВАНИЕ)  KOL1
      FROM  KNIGA  WHERE  ЖАНР = 'Роман'
      GROUP  BY  ИЗДАТЕЛЬСТВО;
   CURSOR  V2  IS  SELECT  ИЗДАТЕЛЬСТВО,
   COUNT(НАЗВАНИЕ)  KOL2
      FROM  KNIGA  WHERE  ЖАНР = 'Фантастика'
      GROUP  BY  ИЗДАТЕЛЬСТВО;
BEGIN
   DELETE  FROM  STAT;
   FOR  Z1  IN  V1  LOOP
      INSERT  INTO  STAT  VALUES(Z1.ИЗДАТЕЛЬСТВО,
      Z1.KOL1, 0);
   END  LOOP;
   FOR  Z1  IN  V2  LOOP
      UPDATE  STAT  SET  KOL_FAN = Z1.KOL2
      WHERE  IZD = Z1.ИЗДАТЕЛЬСТВО;
      IF  SQL%NOTFOUND  THEN
         INSERT  INTO  STAT  VALUES(Z1.ИЗДАТЕЛЬСТВО, 0,
         Z1.KOL2);
      END IF;
   END LOOP;
END  TR3;
```

Действие триггера может быть проверено выполнением следующих операторов, которые, осуществляя вставку строк в таблицу KNIGA, удаление строк и обновление строк в таблице KNIGA, тем самым вызывают активизацию триггера TR3.

Операторы вставки строк в таблицу KNIGA, вызывающие активизацию триггера TR3:

```
INSERT INTO KNIGA VALUES( 46, 'Еретики Дюны', 'Герберт',368,
'Аст', 'Фантастика');
INSERT INTO KNIGA VALUES( 41, 'Еретики Дюны', 'Герберт’,
 2668, 'Нова', 'Фантастика ');
INSERT INTO KNIGA VALUES( 44, 'Казаки', 'Толстой',4568, 'Нова',
 'Роман ');
INSERT INTO KNIGA VALUES( 42, 'Ингвар и Ольха',
 'Никитин',168, 'Аст', 'Роман ');
```

Операторы удаления строк из таблицы KNIGA, вызывающие активизацию триггера TR3:

```
DELETE KNIGA WHERE АВТОР = 'Герберт';
DELETE KNIGA WHERE НАЗВАНИЕ = 'Еретики Дюны';
DELETE KNIGA WHERE НАЗВАНИЕ = 'Казаки';
```

Операторы модификации строк в таблице KNIGA, вызывающие активизацию триггера TR3:

```
UPDATE KNIGA SET ЖАНР='Фантастика' WHERE НАЗВАНИЕ =
 'Ингвар и Ольха';
```

Просмотр информации в таблице STAT можно выполнить следующим оператором:

```
SELECT * FROM STAT;
```

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

Задания для самостоятельного выполнения

1. Создать триггер, который перед вставкой очередной строки, содержащей сведения о продаже некоторой книги, в таблицу KNIGA_PRODAGA, проверяет наличие проданной книги у данного продавца по таблице KNIGA_POSTAVKA и отнимает от количества имеющихся у него книг единицу. При отсутствии необходимой информации в таблице KNIGA_POSTAVKA сгенерировать исключение и выдать диагностирующее сообщение.

2. Создать триггер, который при изменении фамилии продавца в таблице KNIGA_POSTAVKA меняет ее и в таблице KNIGA_PRODAGA. При отсутствии строки с изменяемой фамилией продавца в таблице KNIGA_PRODAGA сгенерировать исключение с выдачей соответствующего сообщения.

3. Создать триггер, который в созданной таблице STAT_PRODAVES со столбцами FIO, KOL_DETECTIV, KOL_FANTASTIC, KOL_ROMAN, содержащими сведения о проданных продавцом книгах, модифицирует значения в соответствующих столбцах таблицы STAT при добавлении строк в таблицу KNIGA_PRODAGA. В зависимости от жанра проданной продавцом книги значение увеличивается на единицу. Реализовать триггер в двух вариантах: строковом и операторном.

Вопросы для самоконтроля

1. Общая характеристика триггеров и их назначение.
2. Создание и включение триггера.
3. Ключевые слова BEFORE, AFTER.
4. Активизирующие DML-операторы.
4. Триггерные предикаты.
5. Строковые и операторные триггеры.
6. Классификация триггеров.
7. Порядок активизации триггеров.
8. Назначение атрибутов :new и :old.
9. Включение и выключение триггеров.
10. Изменение и уничтожение триггеров.
11. Получение информации о триггерах.
12. Использование триггеров.

Лабораторная работа 5

ХРАНИМЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ БАЗЫ ДАННЫХ

Цель работы. Создание и отладка хранимых процедур и функций. Перекомпиляция хранимых процедур и функций. Удаление хранимых процедур и функций из базы данных. Получение необходимой информации о хранимых процедурах и функциях. Работа с демонстрационными примерами и самостоятельная работа.

ОСНОВНЫЕ СВЕДЕНИЯ И ДЕМОНСТРАЦИОННЫЕ ПРИМЕРЫ

Хранимые процедуры и функции

Функции и процедуры (подпрограммы) представляют собой оформленные специальным образом именованные блоки PL/SQL, которые могут быть вызваны для выполнения и которым могут быть переданы параметры. Как правило, процедуры и функции реализуют определенное законченное действие над некоторым объектом базы данных.

Типы процедур и функций. Существуют два вида процедур и функций: локальные и хранимые. Локальные процедуры и функции могут использоваться только в тех блоках, где они определены. Хранимые процедуры и функции компилируются и хранятся в базе данных в скомпилированном виде. При необходимости они могут быть вызваны для выполнения анонимными и именованными блоками PL/SQL, процедурами и функциями обоих видов, триггерами, а также из интерактивной среды SQL* PLUS. Помимо этого хранимая функция может быть вызвана и в операторе SQL.

Создание хранимых процедур и функций. Для создания хранимой процедуры используется следующий общий синтаксис:

```
CREATE [OR REPLACE] PROCEDURE имя_процедуры
[(параметр1  [, параметр2,…..])] IS
   [раздел локальных объявлений]
```

```
BEGIN
   исполняемый раздел
[ EXCEPTION
   раздел обработки исключений]
END [имя процедуры];
```

Для создания хранимой функции используется следующий общий синтаксис:

```
CREATE [OR REPLACE] FUNCTION имя_функции
[(параметр1[,параметр2,…]] RETURN тип_данных IS
   [раздел локальных объявлений]
BEGIN
   исполняемый раздел
[ EXCEPTION
   раздел обработки исключений]
END [имя функции];
```

Хранимые процедуры и функции, вызываемые блоками PL/SQL, процедурами и функциями, триггерами, вызываются так же, как и локальные, заданием имени функции или процедуры с указанием списка фактических параметров. Если вызывается функция, то она должна быть частью выражения; если вызывается процедура, то она вызывается как отдельный оператор.

Для вызова из SQL* PLUS хранимой процедуры используется следующая форма записи:

```
EXECUTE имя_процедуры (список_фактических_параметров);
```

Поскольку функция из интерактивного редактора не может быть вызвана непосредственно, для ее вызова необходимо использовать блок PL/SQL, анонимный или именованный.

Параметры процедур и функций (подпрограмм). Для передачи информации в подпрограмму используются параметры. Переменные или выражения, перечисленные в списке параметров в спецификации подпрограммы, называются формальными параметрами, а перечисленные в списке параметров при вызове подпрограммы называются фактическими аргументами. При вызове подпрограммы фактические аргументы вычисляются и результирующие значения присваиваются формальным параметрам, причем производятся необходимые преобразования типов, поэтому формальные параметры и фактические аргументы должны иметь совместимые типы.

Список параметров представляет собой перечисление этих параметров через запятую. Каждый формальный параметр может быть описан следующим синтаксисом:

Имя_параметра [вид] тип [{:= | DEFAULT } значение];

Параметр *вид* определяет режим передачи параметра. Имеются три режима передачи параметров: IN (по умолчанию), OUT и IN OUT. Они используются для обозначения соответствия входных, выходных и модифицируемых параметров. Желательно не использовать режимы OUT и IN OUT при написании функций, чтобы избежать побочных эффектов.

Фактический аргумент, указываемый на месте IN-параметра, должен быть константой, литералом, проинициализированной переменной либо выражением, и в отличие от OUT- и IN OUT-параметров, IN-параметр может иметь значение по умолчанию. Если параметр передается с вариантом IN, то в подпрограмме ему нельзя присваивать значение.

На месте OUT- или IN OUT-параметра может быть указана только переменная. Как и переменные, OUT-параметры инициализируются NULL-значением, и тип OUT-параметра не может быть подтипом, определенным как NOT NULL. В противном случае генерируется исключение VALUE_ERROR.

Если при выполнении процедуры или функции возникают исключительные ситуации, то управление передается в вызывающий блок. Когда осуществляется нормальный выход из подпрограммы, то фактическим OUT- и IN OUT- аргументам присваиваются значения, а если возникают необработанные исключения, то значения не присваиваются.

Параметр *тип* определяет допустимый тип данных для параметра. В качестве типа параметра могут использоваться практически все основные типы данных языка. Однако если используются типы CHAR, VARCHAR2 или NUMBER, то нельзя указывать размерность для этих типов данных, а для типа NUMBER – точность и масштаб.

Порядок задания параметров. При вызове подпрограмм можно записывать список фактических аргументов, используя либо позиционную, либо именованную, либо смешанную нотации, либо используя передачу параметров по умолчанию:

1) позиционная нотация – это передача списка параметров простым перечислением, причем типы, количество и порядок следования параметров должны соответствовать объявленным раньше;
2) в именованной нотации стрелка => используется как оператор ассоциации, который связывает формальный параметр слева от стрелки с

фактическим аргументом справа от стрелки. При именованной нотации параметры могут указываться в любом порядке;

3) нотации могут смешиваться (смешанная нотация), но в этом случае позиционное указание параметров должно предшествовать именованному;
4) существует возможность передачи параметров по умолчанию. При этом формальным параметрам должны быть присвоены значения либо оператором присваивания, либо через ключевое слово DEFAULT, и они в списке фактических параметров должны быть записаны последними.

Подпрограммы и зависимости. Перекомпиляция подпрограмм. Хранимые функции и процедуры хранятся в скомпилированном виде в базе данных. При этом, как правило, их исполнение затрагивает некоторые объекты базы данных. Для обеспечения достоверности работы таких процедур или функций система постоянно отслеживает для каждой процедуры или функции состояние объектов, с которыми она связана. Если какой-то из связанных с ней объектов подвергается модификации с помощью оператора DDL, то процедура или функция объявляется системой недействительной или недостоверной (NO VALID). В этом случае процедуру или функцию, объявленную недостоверной, надо обязательно перекомпилировать.

Для того чтобы перекомпилировать хранимую процедуру используется команда:

ALTER PROCEDURE имя_процедуры COMPILE;

Команда ALTER FUNCTION перекомпилирует хранимую функцию:

ALTER FUNCTION имя_функции COMPILE;

Удаление подпрограмм из базы данных осуществляется следующей командой:

DROP {PROCEDURE | FUNCTION} имя_подпрограммы;

Получение информации о хранимых процедурах и функциях. Процедуры и функции хранятся в скомпилированном виде в базе данных, поэтому информацию о них можно получить из представления словаря данных USER_OBJECTS, например, следующей командой:

SELECT * FROM USER_OBJECTS;

Демонстрационные примеры

В данной лабораторной работе используются таблицы KNIGA, KNIGA_POSTAVKA и KNIGA_PRODAGA, созданные в лабораторной работе 1.

1. Создать хранимую процедуру, увеличивающую стоимость указанной книги в таблице KNIGA на 10 %. Передаваемый параметр: код книги.

Вариант 1.

```
CREATE OR REPLACE PROCEDURE UVEL(КОД_КНИГИ
KNIGA.ЦЕНА%TYPE) AS
   Q NUMBER(1) := 0;
BEGIN
   SELECT COUNT(*) INTO Q FROM KNIGA
      WHERE KNIGA.КОД_КНИГИ = UVEL.КОД_КНИГИ;
   IF Q<>0 THEN
      UPDATE KNIGA SET ЦЕНА = ЦЕНА + ЦЕНА*0.01
         WHERE KNIGA.КОД_КНИГИ = UVEL.КОД_КНИГИ;
   ELSE
      INSERT INTO KNIGA (КОД_КНИГИ) VALUES
      (UVEL.КОД_КНИГИ);
   END IF;
END UVEL;
```

Выполнение процедуры (вариант 1):

```
EXECUTE UVEL(1);
```

Вариант 2.

```
CREATE OR REPLACE PROCEDURE UVEL (КОД_КНИГИ
NUMBER) AS
BEGIN
   UPDATE KNIGA SET ЦЕНА = ЦЕНА*1.5
      WHERE KNIGA.КОД_КНИГИ = UVEL.КОД_КНИГИ;
      IF SQL%NOTFOUND THEN
         INSERT INTO KNIGA (КОД_КНИГИ) VALUES
          (UVEL.КОД_КНИГИ);
      END IF;
END UVEL;
```

Выполнение процедуры (вариант 2):

```
EXECUTE UVEL (1);
```

2. Создать хранимую процедуру, которая при поступлении книг к продавцу либо добавляет указанное количество к количеству уже имеющихся у продавца таких же книг, обновляя соответствующую запись в таблице KNIGA_POSTAVKA, либо вставляет новую запись в таблицу KNIGA_POSTAVKA, если у продавца книг такого наименования нет. Параметры: код книги, количество единиц, фамилия продавца.

Создание процедуры:

```
CREATE OR REPLACE PROCEDURE DOBAV (КОД_КНИГИ
NUMBER, КОЛ_ЕДИНИЦ NUMBER,
ПРОДАВЕЦ VARCHAR2) IS
   Q NUMBER(5) := 0;
BEGIN
   SELECT COUNT(*) INTO Q FROM KNIGA_POSTAVKA
     WHERE
     KNIGA_POSTAVKA.КОД_КНИГИ = DOBAV.КОД_КНИГИ
     AND KNIGA_POSTAVKA.ПРОДАВЕЦ = DOBAV.ПРОДАВЕЦ;
   IF Q <> 0 THEN
     UPDATE KNIGA_POSTAVKA SET
     КОЛ_ЕДИНИЦ = КОЛ_ЕДИНИЦ + DOBAV.КОЛ_ЕДИНИЦ
     WHERE
     KNIGA_POSTAVKA.КОД_КНИГИ = DOBAV.КОД_КНИГИ
     AND KNIGA_POSTAVKA.ПРОДАВЕЦ = DOBAV.ПРОДАВЕЦ;
   ELSE
     SELECT MAX(КОД_ОПЕРАЦИИ) INTO Q FROM
     KNIGA_POSTAVKA;
     Q := Q + 1;
      INSERT INTO KNIGA_POSTAVKA VALUES (Q,
      DOBAV.КОД_КНИГИ, DOBAV.ПРОДАВЕЦ,
      DOBAV.КОЛ_ЕДИНИЦ, SYSDATE);
   END IF;
END DOBAV;
```

Выполнение процедуры:

```
EXECUTE DOBAV(1, 10, 'Иванов');
EXECUTE DOBAV(7, 10, 'Иванов');
```

При первом вызове процедуры осуществляется добавление 10 книг к тем, что уже имеются у указанного продавца. При втором вызове в таблицу KNIGA_POSTAVKA вставляется новая запись.

3. Создать хранимую функцию, которая проверяет, входит ли для указанного продавца общее количество имеющихся у него книг в определенный диапазон, и если нет, то выдает соответствующее сообщение. Параметр: фамилия продавца. Минимальное и максимальное значения для проверки устанавливаются в теле функции с помощью переменных MIN_KOL и MAX_KOL.

Создание функции:

```
CREATE OR REPLACE FUNCTION PROV (ПРОДАВЕЦ
VARCHAR2)
   RETURN BOOLEAN IS
   MIN_KOL NUMBER(5) := 10;
   MAX_KOL NUMBER(5) := 100;
   Q NUMBER(5);
BEGIN
   SELECT SUM(КОЛ_ЕДИНИЦ) INTO Q FROM
   KNIGA_POSTAVKA WHERE
   KNIGA_POSTAVKA.ПРОДАВЕЦ = PROV.ПРОДАВЕЦ;
RETURN (Q > MIN_KOL AND Q < MAX_KOL);
END PROV;
```

Вызов функции PROV осуществляется с помощью тестирующей программы, приведенной ниже. Программа добавляет указанное количество (переменная DOBAV_КОЛ_ЕДИНИЦ) книг указанному продавцу (переменная DOBAV_ПРОДАВЕЦ), если общее количество имеющихся у него книг не превосходит 100.

```
DECLARE
DOBAV_ПРОДАВЕЦ VARCHAR2(20);
DOBAV_КОЛ_ЕДИНИЦ NUMBER(5);
DOBAV_КОД_КНИГИ NUMBER(5);
BEGIN
   DOBAV_ПРОДАВЕЦ := 'Иванов';
   DOBAV_КОЛИЕСТВО := 20;
   DOBAV_КОД_КНИГИ := 5;
   IF PROV(DOBAV_ПРОДАВЕЦ) THEN
      UPDATE KNIGA_POSTAVKA SET
      КОЛ_ЕДИНИЦ = КОЛ_ЕДИНИЦ + DOBAV_КОЛ_ЕДИНИЦ
      WHERE
       KNIGA_POSTAVKA.КОД_КНИГИ=DOBAV_КОД_КНИГИ
      AND KNIGA_POSTAVKA.ПРОДАВЕЦ=DOBAV_ПРОДАВЕЦ;
   END IF;
```

```
END;
```

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

Задания для самостоятельного выполнения

1. Создать хранимую процедуру, которая увеличивает наличие всех книг у указанного продавца на 10 единиц, но при этом общее количество не должно превышать 100 единиц.

2. Создать хранимую процедуру, которая при продаже книги уменьшает в таблице KNIGA_POSTAVKA количество единиц по данной книге на 1. Если значение равно 0, то запись удаляется. Если такой книги нет, то выдать сообщение об ошибке. Параметры: код книги, продавец.

3. Создать хранимую функцию, которая по указанному продавцу проверяет, выполнил ли он план по продаже книг. Параметры: продавец, план.

Вопросы для самоконтроля

1. Хранимые процедуры и функции и их назначение.
2. Создание хранимых функций и процедур.
3. Типы формальных параметров и их ограничения.
4. Направление параметров. Параметры по умолчанию.
5. Позиционная нотация.
6. Недостоверные процедуры. Перекомпиляция процедур.
7. Удаление хранимых процедур и функций.
8. Получение информации о хранимых процедурах и функциях.

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Задание 1. «Автопарк». Автопарк осуществляет обслуживание заказов на перевозку грузов, используя для этой цели свой парк автомашин и своих водителей. Водитель, выполнивший заказ, получает 20 % от стоимости перевозки.

Управление автопарком должно иметь сведения:

об автомашинах: номер машины, марка, пробег на момент приобретения, грузоподъемность;

о водителях: табельный номер, фамилия водителя, категория, стаж, адрес, год рождения;

о выполненных заказах: дата, фамилия водителя, номер машины, километраж, масса груза, стоимость перевозки.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по указанному водителю – перечень выполненных заказов за указанный период;
   - по указанной машине – общий пробег и общую массу перевезенных грузов;
   - по каждому водителю – общее количество поездок, общую массу перевезенных грузов, сумму заработанных денег;
   - по водителю, выполнившему наименьшее количество поездок, – все сведения и количество полученных денег;
   - по автомашине с наибольшим общим пробегом – все сведения.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу заказов, проверяет, не превышает ли масса груза грузоподъемности машины, и если это так, то запрещает помещать информацию в таблицу.
5. Создать хранимую процедуру, которая за указанный период определяет количество денег, начисленных каждому водителю за перевозки. В качестве параметра передать начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.
6. Создать хранимую функцию, которая за указанный период определяет количество денег, начисленных указанному водителю за перевозки.

В качестве параметра передать начальную дату периода, конечную дату периода и фамилию водителя.

Задание 2. «Рыболовная флотилия». Флотилия рыболовных траулеров осуществляет поиск косяков рыбы и его отлов. С этой целью каждый траулер отправляется в рейс на некоторое количество суток и посещает ряд мелководных участков моря, называемых банками. Каждая банка имеет свое название. Выловленная рыба классифицируется своим названием, количеством и качеством.

Управление флотилией должно иметь сведения:
о траулерах: название траулера, водоизмещение, дата постройки, члены команды (фамилия, должность, дата приема на работу, год рождения);
о результатах рейсов: название траулера, дата выхода в море, дата возвращения, название посещенной банки, название, качество и количество выловленной рыбы.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по указанному траулеру – перечень выполненных рейсов за указанный период с выдачей сведений об общем количестве выловленной рыбы;
   - по указанной банке – перечень и количество выловленной рыбы по видам рыбы;
   - по банке, на которой было выловлено максимальное количество рыбы низкого качества, – сведения о датах рейсов и траулерах, ее посещавших;
   - по траулеру, выловившему наибольшее количество рыбы, – сведения о капитане и посещенных траулером банках с указанием дат выхода и возвращения;
   - по членам команд – перечень людей, которые на указанную дату должны быть отправлены на пенсию.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу результатов рейса выполняет обновление информации в таблице статистики, где для каждого траулера хранится информация об общем количестве выловленной рыбы.
5. Создать хранимую процедуру, которая за указанный период осуществляет начисление премий членам экипажа за внеплановый отлов рыбы. В качестве параметра передать начальную дату периода, конечную дату

периода, плановое задание, среднюю стоимость килограмма рыбы. Результаты занести в специальную таблицу.

6. Создать хранимую функцию, которая за указанный период осуществляет начисление премии указанному члену экипажа.

Задание 3. «Воздушный извозчик». Отряд грузовых вертолетов осуществляет доставку грузов и людей в высокогорном районе. Каждый вертолет обслуживается экипажем из трех пилотов, постоянно закрепленных за ним. Летчики получают по 5 % от стоимости обычного рейса и 10 % от стоимости спецрейса.

Командир авиаотряда должен иметь сведения:

о вертолетах: номер вертолета, марка, дата изготовления, грузоподъемность, дата последнего капитального ремонта, летный ресурс времени до следующего капитального ремонта;

о членах экипажа: табельный номер, фамилия, должность, стаж, адрес, год рождения, номер вертолета;

о выполненных рейсах: дата, номер вертолета, код рейса, масса груза, количество перевезенных людей, длительность полета, стоимость рейса.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по каждому вертолету – общее количество часов, которые они налетали после капитального ремонта, и ресурс летного времени;
   - по каждому вертолету – перечень выполненных рейсов с указанием общей массы перевезенных грузов и количества человек за указанный период;
   - по всем вертолетам, выполнявшим спецрейсы, – общее количество рейсов, общая масса перевезенных грузов, общая сумма заработанных денег;
   - по вертолету, выполнившему максимальное количество рейсов, – все сведения об его экипаже и количестве заработанных денег;
   - по экипажу, заработавшему максимальное количество денег, – все сведения о выполненных им рейсах.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при внесении информации в таблицу рейсов, проверяет, не будет ли превышен ресурс летного времени для вертолета, и если это так, то запрещает вносить информацию в таблицу.
5. Создать хранимую процедуру, которая за указанный период определяет количество денег, начисленных экипажам авиаотряда за перевоз-

ки. В качестве параметра передать начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.

6. Создать хранимую функцию, которая за указанный период определяет количество денег, начисленных указанному летчику.

Задание 4. «Ипподром». Ипподром регулярно проводит состязания в беге лошадей. Лошади на соревнования заявляются владельцами, в состязании лошадь участвует вместе с жокеем, который в различных состязаниях может работать с различными лошадьми. Жокеи входят в штат ипподрома.

Управление ипподромом должно иметь сведения:
о лошадях: кличка лошади, возраст, стаж участия в соревнованиях, владелец, заплаченная цена;
о жокеях: фамилия жокея, стаж, год рождения, адрес;
о проведенных забегах: дата, номер забега, кличка лошади, фамилия жокея, занятое место.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по лошади, побеждавшей максимальное количество раз, – сведения о ней, датах соревнований и о жокеях;
   - по жокею, участвующему наибольшее количество раз в забегах, – сведения об этом жокее и об общем количестве забегов, в которых он участвовал;
   - по указанному жокею – сведения о датах забегов, участвовавших в них лошадях и занятых местах;
   - по указанному владельцу – список его лошадей с указанием дат забегов и занятых мест;
   - по всем забегам – все сведения о проводимых забегах и участвовавших в них лошадях за указанный период.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который запрещает помещать информацию в таблицу забегов при отсутствии соответствия кличек лошадей и фамилий жокеев с аналогичными данными в таблицах лошадей и жокеев.
5. Создать хранимую процедуру, которая осуществляет распределение призового фонда для тройки призеров последнего забега в соотношении 50 %, 30 %, 20 %. В качестве параметра передать величину призового фонда. Результаты занести в специальную таблицу.

6. Создать хранимую функцию, которая за указанный период выводит по указанному жокею сведения о забегах, где он участвовал.

Задание 5. «Музыкальный салон». Постоянно работающий музыкальный салон продает компакт-диски с записями определенных исполнителей, поступающие от различных компаний-производителей.

Управление салона интересуют сведения:

о компакт-дисках: код компакта, дата изготовления, компания-производитель, цена одного компакта;

об исполнителях музыкальных произведений: название музыкального произведения, автор, исполнитель, код компакта;

о поступлении и продаже компактов: дата операции, код операции (поступление или продажа), код компакта, количество экземпляров.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по всем компактам – сведения о количестве проданных и оставшихся компактов одного вида по убыванию разницы;
   - по указанному компакту – сведения о количестве и стоимости компактов, проданных за указанный период;
   - по компакту, купленному максимальное количество раз, – выдать все сведения о нем и музыкальных произведениях;
   - по наиболее популярному исполнителю – сведения о количестве проданных компактов с его произведениями;
   - по каждому автору – сведения о количестве проданных компактов с его записями и сумме полученных денег.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который запрещает помещать информацию о продаже компактов в таблицу, если суммарное количество проданных компактов превысит суммарное количество поступивших.
5. Создать хранимую процедуру, которая за указанный период определяет количество поступивших и проданных компактов по каждому виду. В качестве параметра передать начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.
6. Создать хранимую функцию, которая по заданному коду компакта выводит информацию о результатах его продажи за указанный период.

Задание 6. «Охотничье хозяйство». Система охотничьего хозяйства включает сеть пунктов приема шкурок пушных зверей (соболь, куница, песец, росомаха и т. д.), в каждом из которых работает один приемщик.

Пункты обслуживают свободных охотников, принимая от них шкурки зверей по разной цене в зависимости от вида и сорта меха. Впоследствии шкурки поставляются на различные меховые фабрики согласно заявке.

Управление охотничьим хозяйством должно иметь информацию:
об охотниках: фамилия охотника, адрес, год рождения;
о приеме шкурок: номер пункта, адрес пункта, фамилия приемщика, дата приема шкурок, фамилия охотника, вид меха, сорт меха, количество единиц меха, цена одной шкурки;
об отгрузке меха на фабрику согласно заявке: дата отгрузки, номер пункта, номер меховой фабрики, вид меха, сорт, количество единиц.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по фабрике, получившей пушнины на максимальную сумму, – сведения о пунктах и приемщиках;
   - по каждой меховой фабрике – общий перечень и количество полученной пушнины по видам и сортам;
   - по каждому пункту – общее количество принятой пушнины, в том числе первого сорта, сумма заработанных денег за указанный период;
   - по пункту, на котором было принято максимальное количество шкурок низкого качества, – все сведения о датах приема и охотниках;
   - по самому молодому охотнику – перечень и количество сданной им пушнины по видам и сортам.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который запрещает производить отгрузку меха на фабрику, если количество в заявке превышает наличие в пункте приема.
5. Создать хранимую процедуру, которая на указанную дату по заданному пункту вычисляет наличие имеющегося меха по видам и сортам и вносит эти сведения в специальную таблицу. В качестве параметра передать номер пункта и дату расчета.
6. Создать хранимую функцию, которая на указанную дату выводит всю информацию о наличии в пунктах меха заданного вида и сорта.

Задание 7. «Пушной аукцион». Управление звероводческими совхозами регулярно проводит пушные аукционы, куда звероводческие совхозы представляют свою продукцию. Пушнина выставляется по определенной цене, но в результате аукциона она может быть продана по более высокой или низкой цене.

Управление звероводческими совхозами должно иметь информацию:

*о звероферм*ах: номер звероферм*ы*, адрес, фамилия директора, телефон;
о выставленной на аукцион пушнине: номер звероферм*ы*, название меха, сорт, количество единиц, заявленная цена;
о результатах аукциона: номер звероферм*ы*, название меха, сорт, количество проданных единиц, продажная цена, категория покупателя (меховая фабрика, ателье, частное лицо).

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:

- по звероферме, чей мех получил самую высокую цену, – все сведения о ней;
- по каждой категории покупателей – общее количество купленных единиц, общая сумма уплаченных денег;
- по каждой звероферме – прибыль от продажи пушнины;
- по всем зверофермам, которые продали шкурки по цене выше средней аукционной цены, – все сведения о них;
- по звероферме, получившей максимальную прибыль, – все сведения о ней, количестве проданной пушнины и величине прибыли.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который запрещает внесение сведений в таблицу результатов аукциона, если вносимые сведения не соответствуют реальности.
5. Создать хранимую процедуру, которая определяет список звероферм с указанием сведений о них, которые получили прибыль от аукциона меньше, чем планировалось. В качестве параметра передать плановый процент прибыли.
6. Создать хранимую функцию, которая по указанной звероферме выдает величину прибыли, полученную на аукционе.

Задание 8. «Цветочная оранжерея». Цветочная оранжерея выращивает различные виды цветов и продает на заказ составленные из них композиции. Каждая композиция имеет свое название и может состоять как из цветов одного вида, так и из цветов разного вида. Заказ обычно выполняется в течение нескольких дней. При выполнении заказа в течение суток дополнительно взимается плата в размере 25 %. При выполнении заказа в течение двух суток дополнительно взимается плата в размере 15 %.

Дирекция оранжереи должна иметь информацию:
о цветах: название цветка, сорт, стоимость одного цветка;

о композициях: название композиции, название входящего в композицию цветка, сорт, количество единиц;
о выполнении заказов: дата принятия заказа, дата выполнения заказа, название композиции, количество единиц, покупатель.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по всем заказам – сумма полученных денег за указанный период;
   - по композиции, пользующейся максимальным спросом, – все сведения о ней;
   - по всем заказам – сведения о количестве выполненных заказов по срочности;
   - по всем заказам – сведения о количестве использованных цветов по видам и сортам за указанный период;
   - по всем заказам – сведения о количестве проданных композиций и сумме полученных денег по видам композиций.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который запрещает увеличивать цену на цветы, если стоимость композиции увеличивается более чем на 10 %.
5. Создать хранимую процедуру, которая осуществляет внесение информации в таблицу заказов, затем рассчитывает стоимость заказа и вносит сведения о нем в специальную таблицу. В качестве параметра передать все сведения о заказе.
6. Создать хранимую функцию, которая на указанную дату выводит всю информацию о полученных заказах.

Задание 9. «Парфюмерный базар». Постоянно работающий парфюмерный базар характеризуется участием в ней оптовых фирм-поставщиков и оптовых фирм-покупателей, при этом все сделки заключаются через специальных маклеров, имеющих доступ ко всем товарам.

Управление базаром должно иметь сведения:
о маклерах: фамилия маклера, адрес, год рождения;
о товаре: название товара, вид, цена единицы товара, оптовая фирма-поставщик, срок годности, количество поставленных единиц;
о заключенных сделках: дата сделки, название товара, вид, количество проданных единиц, фамилия маклера, оптовая фирма-покупатель.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:

- по каждому названию товара – сведения о проданном количестве и общей стоимости за указанный период;
- по каждому названию товара – перечень фирм-покупателей с указанием сведений о количестве единиц и стоимости купленного ими товара по каждой фирме-покупателю;
- по виду товара, пользующемуся наибольшим спросом, – сведения о количестве и стоимости проданного товара по каждой фирме-покупателю;
- по маклеру, совершившему максимальное количество сделок, – сведения о нем и фирмах-поставщиках;
- по каждой фирме-поставщику – список маклеров с указанием сведений о количестве и стоимости проданного ими товара по каждому маклеру.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.

4. Создать триггер, который по совершении очередной сделки обновляет по каждому маклеру сведения о количестве единиц проданного товара и сумме сделки в специальной таблице статистики.

5. Создать хранимую процедуру, которая на указанную дату выполняет обновление данных таблицы, содержащей сведения о товаре, вычитая из количества поставленных единиц количество проданных единиц до указанной даты, и удаляет соответствующие строки из таблицы заключенных сделок. В качестве параметра передать дату периода.

6. Создать хранимую функцию, которая на указанную дату выводит всю информацию о совершенных сделках.

Задание 10. «Аттракционы». Городская служба хозяйствования имеет в своем распоряжении несколько площадок аттракционов, функционирующих в парках города. Каждая площадка имеет свой круг сотрудников и набор аттракционов, цены на которые установлены в трех категориях: детские, льготные и взрослые. Каждый аттракцион имеет определенный срок службы, после чего он подлежит списанию. В конце рабочего дня использование аттракциона фиксируется датой, количеством проданных билетов по каждой из категорий.

Городская служба должна иметь сведения:
о площадках: номер площадки, адрес площадки, фамилия директора;
об аттракционах: номер аттракциона, название аттракциона, год приобретения, срок службы, номер площадки;
об использовании аттракционов: дата, номер площадки, номер аттракциона, количество проданных билетов по категориям.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:

- по указанной площадке – общее количество и сведения о количестве посетителей по возрастным категориям за указанный период;
- по каждому аттракциону – сведения о местоположении аттракциона и выручке, начиная с указанного периода;
- по каждому аттракциону – сведения об оставшемся времени его использования по возрастанию значений;
- по наиболее используемому аттракциону, – сведения о местоположении площадки, фамилии директора и количестве посетителей по возрастным категориям;
- по каждой площадке – количество полученных денег за указанный период.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который по всем площадкам ведет учет выручки за каждый день и, начиная с начала месяца.
5. Создать хранимую процедуру, которая на указанную дату рассчитывает процентное распределение выручки по возрастным категориям по всем площадкам. В качестве параметра передать расчетную дату. Результаты занести в специальную таблицу.
6. Создать хранимую функцию, которая на указанную дату выводит всю информацию о полученной прибыли.

Задание 11. «Автомастерские». Городская служба хозяйствования имеет в своем распоряжении несколько автомастерских, каждая из которых проводит обслуживание автомобилей определенных марок. При этом выполняются следующие виды работ:

- замена отдельных элементов кузова, подбор краски и покраска;
- замена ремней, регулировка клапанов, замена маслосъёмных колпачков;
- замена ведущих и ведомых шестерен;
- замена масла, замена фильтров.

Каждая автомастерская имеет свой штат работников.

Городская служба должна иметь сведения:

о мастерских: номер автомастерской, адрес, перечень марок ремонтируемых машин, список мастеров;

об отремонтированных машинах: госномер, марка, год выпуска, фамилия владельца, номер техпаспорта, адрес владельца;

о выполненных работах: номер мастерской, дата поступления, дата завершения ремонта, госномер, вид ремонта, стоимость ремонта, мастер.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:

- по указанной автомастерской – перечень выполненных работ каждым мастером за указанный период;
- по указанному мастеру – сведения о выполненных ремонтах и отремонтированных автомашинах;
- по каждой марке отремонтированных машин – сведения о номерах автомастерских, датах и видах ремонта, фамилиях мастеров;
- по каждой автомастерской – сведения об общем количестве проведенных ремонтов и общей выручке;
- по автомастерской с наибольшим количеством ремонтов – все сведения о проведенных ремонтах и отремонтированных машинах.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу выполненных работ обновляет хранящуюся в специальной таблице по всем мастерам информацию о количестве выполненных работ по видам работ.
5. Создать хранимую процедуру, которая за указанный период определяет для указанной автомастерской количество и перечень выполненных работ по видам. В качестве параметра передать номер автомастерской, начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.
6. Создать хранимую функцию, которая на указанную дату выводит информацию о количество выполненных работ для указанной мастерской.

Задание 12. «Зоопарк». Городская служба хозяйствования имеет в своем распоряжении зоопарк, функционирующий в черте города. Животные, находящиеся в зоопарке, привезены из различных стран и соответственно нуждаются в создании определенных условий. Рацион животных делится на несколько типов: мясной, зерновой, растительный, смешанный. На единицу массы животного приходится определенная норма соответствующего корма. В конце рабочего дня результаты посещения зоопарка фиксируются датой и количеством проданных билетов. Выручка от продажи билетов, которые делятся на три категории (детские, льготные, взрослые), частично покрывает расходы на приобретение

животных, кормление животных и их содержание. Однако городское хозяйство вынуждено осуществлять дотацию зоопарка, поэтому ему необходимо иметь ряд сведений:

о животных: номер клетки, название животного, кличка, возраст, вес, год приобретения, страна обитания, часть света, климат, тип рациона, стоимость животного;

о рационе: тип рациона, вид корма, норма расхода на единицу веса животного, стоимость единицы корма;

о посещении зоопарка: дата, число проданных билетов по категориям.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по зоопарку – общее количество посетителей и сумму выручки за указанный период;
   - по зоопарку – количество денег, необходимых на кормление животных за указанный период (день, месяц, год);
   - по всем животным – стоимость их содержания по убыванию;
   - по наиболее дорогому животному (стоимость животного плюс стоимость содержания за один месяц) – все сведения;
   - по зоопарку – сведения о количестве посетителей по возрастным категориям за указанный период.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу количества проданных билетов обновляет сведения о количестве посетителей за сутки и с начала месяца.
5. Создать хранимую процедуру, которая за указанный период определяет количество денег, необходимых на содержание указанного животного. В качестве параметра передать название животного, начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.
6. Создать хранимую функцию, которая на указанную дату выводит всю информацию о стоимости содержания указанного животного.

Задание 13. « Туристическое бюро». Бюро путешествий осуществляет обслуживание экскурсионных маршрутов, используя для этой цели свои автобусы и экипажи, состоящие из трех человек, которые строго закреплены за своим автобусом. Экипаж, выполнивший заказ на обслуживание, получает 20 %, от стоимости обслуживания. Стоимость билета до-

говорная и зависит от длины туристической трассы и числа участников экскурсии.

Бюро должно иметь сведения:

об экскурсионных маршрутах: название маршрута, начальный пункт, конечный пункт, протяженность пути;

об автобусах: номер автобуса, название, общая величина пробега;

о членах экипажа: фамилия, табельный номер, стаж, категория, адрес, год рождения, номер автобуса;

о выполненных рейсах: номер автобуса, дата отбытия, дата прибытия, название маршрута, количество перевезенных пассажиров, стоимость билета.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по указанному автобусу – перечень выполненных рейсов за указанный период;
   - по указанному автобусу – общее количество поездок, количество перевезенных пассажиров и полученных денег;
   - по каждому экипажу – количество начисленных денег за указанный период;
   - по наиболее дорогому маршруту – сведения об автобусах, экипажах и стоимости билетов;
   - по автобусу с наибольшим суммарным пробегом – сведения о количестве перевезенных пассажиров и величине пробега.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу экскурсий, проверяет соответствие вносимой информации той информации, которая хранится в основных таблицах.
5. Создать хранимую процедуру, которая за указанный период определяет количество денег, начисленных каждому экипажу за обслуживание экскурсий. В качестве параметра передать процент отчисления, начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.
6. Создать хранимую функцию, которая на указанную дату выводит информацию о количестве денег, начисленных указанному экипажу.

Задание 14. «Праздничная феерия». Магазин по продаже фруктов в предпраздничные дни формирует праздничные наборы из фруктов и орехов. Каждый набор имеет название. В наборы могут входить свежие

фрукты, сухофрукты, засахаренные фрукты и орехи: миндальные, фундук, кешью и арахис. Заказ обычно выполняется в течение нескольких дней. При выполнении заказа в течение суток дополнительно взимается плата в размере 20 %. При выполнении заказа в течение двух суток дополнительно взимается плата в размере 10 %. Имеющаяся система скидок позволяет уменьшить стоимость одного набора при покупке больших партий.

Директор магазина должен иметь информацию:

о фруктах и орехах: название фрукта или орехов, вид, страна поступления, стоимость одного килограмма;

о наборах: название набора, название ингредиента, вид, вес ингредиента;

о выполнении заказов: дата принятия заказа, дата выполнения заказа, название набора, количество единиц, заказчик;

о системе скидок: пороговое значение, процент снижения оплаты.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по всем заказам – общее количество, общую стоимость и размер скидки за указанный период;
   - по наборам, пользующимся максимальным спросом, – все сведения о них;
   - по всем заказам – сведения о количестве выполненных заказов по срочности;
   - по всем заказам – сведения о количестве использованных ингредиентов по каждому виду;
   - по всем заказам – сведения о количестве проданных наборов и сумме полученных денег по видам наборов.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который запрещает устанавливать размер скидки более 30 %.
5. Создать хранимую процедуру, которая осуществляет внесение информации в таблицу заказов, затем рассчитывает стоимость заказа, размер скидки и вносит эти сведения о нем в специальную таблицу. В качестве параметра передать все сведения о заказе.
6. Создать хранимую функцию, которая на указанную дату выводит информацию о количестве выполненных заказов.

Задание 15. « Морской круиз». Управление морского пароходства регулярно выделяет определенное количество теплоходов для выполне-

ния круизных рейсов. Каждый теплоход имеет свою команду. За выполнение рейса команда теплохода получает 5 % от стоимости рейса. Маршрут круиза содержит несколько пунктов остановки, но обычно везет одну и ту же группу туристов из начального пункта в конечный. Билеты делятся на три категории: первого, второго и третьего классов.

Управление интересуют сведения:

о маршрутах круизов: название круиза, количество дней, общая протяженность, количество остановок;

о теплоходах: название теплохода, регистрационный номер, дата выпуска, предельный срок эксплуатации, число членов команды, стоимость билетов по категориям;

о выполненных экскурсиях: название теплохода, название круиза, дата отбытия, количество проданных билетов по категориям.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по всем теплоходам – перечень теплоходов, которые возвращаются в указанный день;
   - по всем круизам – общее количество и стоимость проданных билетов по имеющимся категориям;
   - по каждому теплоходу – количество полученных денег за указанный период;
   - по наиболее популярному маршруту круиза – все сведения о нем и общее количество посетивших его туристов;
   - по теплоходу, выполнившему максимальное количество круизных рейсов – сведения о нем и числе перевезенных пассажиров.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу экскурсий, проверяет соответствие вносимой информации той информации, которая хранится в основных таблицах.
5. Создать хранимую процедуру, которая за указанный период определяет количество денег, начисленных каждому экипажу за выполнение круизных рейсов. В качестве параметра передать процент отчисления от общей суммы полученных денег, начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.
6. Создать хранимую функцию, которая на указанную дату выводит информацию о количестве денег, начисленных экипажу указанного теплохода.

Задание 16. « Реклама на ТВ». Фирмы рекламируют свои товары на ТВ в различных передачах и на различных каналах. Известна цена минуты рекламного времени в той или иной передаче. Для каждой фирмы определен общий процент скидки.

Генерального директора ТВ интересуют сведения:

о передачах: название передачи, номер канала, время выхода в эфир, стоимость единицы рекламного времени;

о рекламодателях: название фирмы, рекламируемый товар, процент скидки;

о выполненных рекламных паузах: дата, название передачи, название фирмы, рекламируемый товар, длительность рекламы (мин).

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по рекламируемым товарам – общее количество минут рекламного времени по каждой передаче;
   - по каждой фирме – перечень рекламируемых товаров с указанием общего количества рекламного времени и затраченных денег;
   - по каждой передаче – перечень фирм с указанием общего количества минут и общей стоимости рекламы по каждой фирме за указанный период;
   - по наиболее рекламируемому товару – перечень передач с указанием суммарного времени и стоимости рекламы по каждой передаче;
   - по передаче, пользующейся среди рекламодателей максимальной популярностью, – сведения о ней, общем количестве рекламного времени и сумме полученных денег.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу выполненных рекламных пауз проверяет соответствие вносимой информации той информации, которая хранится в основных таблицах.
5. Создать хранимую процедуру, которая на указанную дату подсчитывает процентное распределение полученной прибыли среди передач ТВ и заносит эти сведения в специальную таблицу. В качестве параметра передать расчетную дату.
6. Создать хранимую функцию, которая на указанную дату выводит информацию о полученной прибыли указанной передачей.

Задание 17. «Компьютерные игры». Постоянно работающий компьютерный салон продает компакт-диски с записями игр, поступающие от различных фирм.

Управление салона интересуют сведения:

о компакт-дисках: код компакт-диска, дата изготовления, фирма, цена одного компакта;

об играх: код компакт-диска, название игры, жанр игры, автор;

о поступлении и продаже компакт-дисков: дата операции, код операции (поступление или продажа), код компакт-диска, количество единиц.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по указанному компакт-диску – сведения о количестве и стоимости компактов, проданных за указанный период;
   - по компакт-диску, купленному минимальное количество раз – выдать все сведения о нем и имеющихся там играх;
   - по наиболее популярной игре – сведения о количестве проданных компакт-дисков с этой игрой;
   - по всем компакт-дискам – сведения о количестве проданных и оставшихся компакт-дисков одного вида по убыванию разницы;
   - по каждому из авторов – сведения о количестве проданных компакт-дисков с его играми и сумме полученных денег.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который запрещает помещать информацию о продаже компакт-дисков в таблицу операций, если суммарное количество проданных компакт-дисков превысит суммарное количество поступивших.
5. Создать хранимую процедуру, которая за указанный период определяет количество поступивших и проданных компакт-дисков по каждому виду компакта. В качестве параметра передать начальную дату периода и конечную дату периода. Результаты занести в специальную таблицу.

Задание 18. «Каменная сказка». Управление Ювелирторгом регулярно проводит ярмарки-продажи «Каменная сказка», где выставляются на продажу изделия ювелирных фирм. Изделия представляются по определенной цене, но в результате ярмарки они могут быть проданы по более высокой или низкой цене различным категориям покупателей.

Управление Ювелирторгом должно иметь информацию:

о ювелирных фирмах: название фирмы, адрес, фамилия директора, телефон;
о выставленных на очередную ярмарку изделиях: название фирмы, название изделия, сорт, количество единиц, заявленная цена;
о результатах деятельности ярмарки: название фирмы, название изделия, сорт, количество проданных единиц, продажная цена, категория покупателя (учреждение, оптовик, частное лицо).

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по каждой категории покупателей – общее количество купленных единиц, общая сумма уплаченных денег;
   - по каждой фирме – сведения о выставленных и проданных изделиях;
   - по всем фирмам, которые продали изделия по цене выше заявленной, – все сведения о них;
   - по фирме, чье изделие получило самую высокую цену, – все сведения о ней и об изделии;
   - по фирме, получившей максимальную прибыль, – все сведения о ней, количестве проданных изделий и величине прибыли.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который запрещает внесение сведений в таблицу результатов ярмарки, если вносимые сведения не соответствуют реальности, то есть не совпадают со сведениями, хранящимися в таблицах изделий и фирм.
5. Создать хранимую процедуру, формирующую список фирм, которые получили прибыль меньше от ярмарки, чем планировалось с указанием сведений о них. В качестве параметра передать плановый процент прибыли.

Задание 19. «Ювелирная фантазия». Постоянно работающий магазин «Ювелирная фантазия» продает ювелирные изделия, поступающие от ювелирных фабрик. В магазине имеется штат продавцов, реализующих поступающий товар. В качестве премиальных продавцы имеют 2 % от полученной выручки за продажу изделий.

Директор магазина должен иметь информацию:
о продавцах: фамилия продавца, оклад, год рождения, стаж работы, адрес, образование;

о товаре: дата поступления, название изделия, количество единиц, цена, фамилия продавца, фабрика-изготовитель;
о результатах дневной продажи: дата, название изделия, фамилия продавца, количество единиц.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:

- по указанному продавцу – перечень изделий, имевшихся у него в продаже в течение указанного периода;
- по указанной фабрике-изготовителю – список продавцов, имеющих их изделия для продажи;
- по каждому продавцу – сведения о количестве изделий, общей стоимости и сумме заработанных денег за указанный период;
- по фабрике-изготовителю, чьих изделий было продано на самую большую сумму, – все сведения о продавцах, которые торговали ими;
- по трем самым дорогим изделиям – фамилия продавца, дата поступления, цена, фабрика-изготовитель, дата продажи, если было продано.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу о результатах продажи за день обновляет хранящуюся в специальной таблице по всем фабрикам-изготовителям информацию о количестве поставленных и проданных изделий за сутки и с начала недели по каждой фабрике.
5. Создать хранимую процедуру, которая определяет список тех фабрик-изготовителей, с указанием сведений о них, которые получили прибыль от продажи изделий меньшую, чем планировалось за последний месяц. В качестве параметра передать величину прибыли.

Задание 20. «Кинопрокат». В городе имеется сеть кинотеатров, в которых управление кинопрокатом размещает фильмы для показа зрителям. Один и тот же фильм может идти в нескольких кинотеатрах, минимальный срок показа фильма – одна неделя.

Дирекция кинопроката должна иметь информацию:
о фильмах: название фильма, жанр, режиссер, исполнитель главной мужской роли, исполнительница главной женской роли, год выпуска, компания-производитель;

о кинотеатрах: название кинотеатра, класс кинотеатра (стерео, система Долби и т.д.), количество мест, средняя цена билета, адрес, фамилия директора;
о результатах деятельности кинотеатра за день: дата, название кинотеатра, название фильма, количество зрителей.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по фильму, идущему в кинотеатрах города наиболее длительный срок, – все сведения об этом фильме с указанием кинотеатра, где он демонстрируется в последнее время;
   - по каждому кинотеатру – суммарная и средняя выручка за указанный период;
   - по всем фильмам, идущим на текущей неделе, – все сведения о них по жанрам;
   - по всем фильмам – все сведения о фильмах, вошедшим в тройку фильмов с наилучшим кассовым сбором;
   - по каждому фильму – информация об общей выручке по каждому кинотеатру.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу о результатах деятельности кинотеатра за день обновляет хранящуюся в специальной таблице по всем кинотеатрам информацию о количестве полученных денег за сутки и с начала недели.
5. Создать хранимую процедуру, которая определяет список тех кинотеатров с указанием сведений о них, которые получили прибыль от проката фильмов меньшую, чем планировалось за последнюю неделю. В качестве параметра передать плановый процент прибыли.

Задание 21. «Производственно-коммерческая фирма». Производственно-коммерческая фирма осуществляет сборку компьютеров и продажу их потребителям как оптовым (школы, институты, учреждения), так и индивидуальным. При продаже действует система скидок: свыше 10 штук – 10 %, свыше 20 штук – 15 %, свыше 50 штук – 20 %. Цена компьютера устанавливается на 10 % выше, чем общая цена входящих в него комплектующих.

Дирекция фирмы должна иметь информацию:

о поступлении комплектующих единиц: номер комплектующей, название комплектующей, цена одной комплектующей, количество единиц, фирма-поставщик;
о компьютерах: номер модели, название модели, номер комплектующей, количество используемых единиц;
о продаже: дата, номер модели, количество проданных единиц, категория покупателя (учебное заведение, фирма, частное лицо и т. д.), покупатель.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:
   - по всем продажам – сведения о ежедневной выручке за указанный период;
   - о дне продажи с минимальной выручкой – сведения о проданных моделях с указанием их комплектующих;
   - о модели, пользующейся наибольшим спросом, – список фирм-поставщиков комплектующих;
   - по всем моделям – все сведения о моделях, вошедших в тройку компьютеров с наилучшим спросом;
   - по всем моделям – информация об общей выручке и количестве проданных компьютеров за указанный период.
3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу о результатах продаж компьютеров обновляет хранящуюся в специальной таблице по всем моделям информацию о количестве проданных компьютеров за текущий месяц по каждой модели.
5. Создать хранимую процедуру, которая на указанную дату обновляет количество комплектующих в соответствии с выполненными продажами и при достижении для некоторых из деталей порогового значения наличия выдает соответствующее сообщение. В качестве параметра передать дату расчета и пороговое значение.

Задание 22. «Фильмотека». Постоянно работающая фильмотека характеризуется участием в ней оптовых фирм поставщиков и регулярного штата продавцов, а также наличием контингента постоянных и временных покупателей.

Управление фильмотекой должно иметь сведения:
о кассетах с фильмами: название кассеты, тематика, режиссер, цена, год выпуска, киностудия;

о поступлении кассет: дата поступления, название кассеты, количество экземпляров, фамилия продавца, название фирмы-поставщика;
о продаже кассет: дата продажи, фамилия продавца, название кассеты, количество купленных экземпляров, фамилия покупателя.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:

- по каждой фирме поставщику – сведения о количестве и стоимости проданных кассет по тематике;
- по указанной тематике – списки фирм-поставщиков и продавцов;
- по указанному продавцу – сведения о количестве проданных кассет, общей и средней выручке за указанный период;
- по указанной тематике – перечень покупателей, купивших эти кассеты за указанный период;
- по кассете, купленной максимальное количество раз, – выдать все сведения.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу о результатах деятельности фильмотеки за день обновляет хранящуюся в специальной таблице по всем продавцам информацию о количестве проданных ими кассет за сутки и с начала недели.
5. Создать хранимую процедуру, которая определяет список тех продавцов с указанием сведений о них, которые получили прибыль от продажи кассет меньшую, чем планировалось за последнюю неделю. В качестве параметра передать плановый процент прибыли.

Задание 23. «Компьютерное кафе». Центр «Компьютер-Сервис» имеет в своем распоряжении сеть компьютерных кафе, в которых установлены компьютеры разных марок, используемые либо для игр, либо для выхода в Интернет. Каждое кафе имеет свой круг сотрудников и набор компьютеров, цены на которые установлены в двух категориях: для игры и для выхода в Интернет, цена устанавливается почасовая. Каждый компьютер имеет определенный срок службы, после чего он подлежит списанию. В конце рабочего дня использование компьютера фиксируется датой, количеством проданных билетов по каждой из категорий.

Центр «Компьютер-Сервис» должен иметь сведения:
о компьютерном кафе: название кафе, адрес кафе, фамилия директора, количество компьютеров;

о компьютерах: название компьютера, регистрационный номер компьютера, год приобретения, срок службы (в часах), стоимость одного часа времени использования по категориям (Интернет, игра), название кафе;
о результатах работы за день: дата, название кафе, регистрационный номер компьютера, количество проданных билетов по категориям, количество использованного времени по категориям (в часах).

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:

- по указанному названию кафе – общее количество посетителей и суммарное время использования компьютеров по категориям за указанный период;
- по каждому компьютеру – сведения о местоположении компьютера и общей выручке, начиная с указанного периода;
- по каждому компьютеру – сведения об оставшемся времени его использования по возрастанию значений;
- по наиболее посещаемому кафе, – сведения о местоположении кафе, фамилии директора и количестве посетителей за последний месяц;
- по трем кафе с наибольшим количеством посетителей – количество полученных денег за указанный период по каждому кафе и перечислением компьютеров, там установленных.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который по всем компьютерным кафе ведет учет выручки за каждый день, начиная с начала каждой очередной недели.
5. Создать хранимую процедуру, которая на указанную дату рассчитывает процентное распределение выручки по используемым категориям по всем компьютерным кафе. В качестве параметра передать расчетную дату. Результаты занести в специальную таблицу.

Задание 24. «Морская полиция». Отряд морской полиции осуществляет патрулирование участков морской акватории. Ежедневно патруль, состоящий из двух человек, на катере должен посетить несколько участков, пользующихся особым вниманием браконьеров, и задержать их, если они будут заниматься незаконной добычей рыбы.

Командиру отряда необходимо иметь сведения:
о патрульных: служебный номер, фамилия, должность, дата зачисления на работу, стаж, год рождения, номер катера;

о результатах патрулирования: дата патрулирования, номер катера, номер участка акватории, количество задержанных нарушителей.

Необходимо:

1. Создать таблицы БД с учетом ограничений целостности данных.
2. Используя оператор Select, выдать следующую информацию:

- по указанному катеру – перечень выполненных патрулирований за указанный период с указанием сведений об общем количестве задержанных браконьеров;
- по указанному участку акватории– перечень дат, номеров катеров, осуществлявших патрулирование, и количестве задержанных браконьеров;
- по участку, пользующемуся наибольшим вниманием браконьеров, – сведения о количестве задержанных там нарушителей и общем количестве патрулирований за указанный период;
- по экипажу, задержавшему наибольшее количество нарушителей, – сведения об экипаже и количестве выполненных патрулирований;
- по каждому катеру – количество выходов на дежурство с указанием количества проверенных участков.

3. Обеспечить с помощью операторов Insert, Update, Delete обновление информации в указанных таблицах.
4. Создать триггер, который при добавлении информации в таблицу результатов патрулирования выполняет обновление информации в таблице статистики, где для каждого члена отряда хранится информация о количестве выходов в море и количестве задержанных нарушителей.
5. Создать хранимую процедуру, которая за указанный период осуществляет начисление премий членам отряда за задержание нарушителей. В качестве параметра передать начальную дату периода, конечную дату периода, размер премии. Начисление премии произвести пропорционально количеству задержанных нарушителей. Результаты занести в специальную таблицу.

ЛИТЕРАТУРА

Баженова И. Ю. Oracle 8/8i. Уроки программирования. М.: Диалог-МИФИ, 2000.

Гроff Д. Р. Вайнберг П.Н. SQL: Полное руководство. Киев: BHV, 1999.

Oracle 8. Энциклопедия пользователя. М.: Диасофт, 1999.

Пэйдж В. Дж. Использование Oracle 8/8i. М.: Вильямс, 2000.

Смирнов С. Н. Работаем с Oracle. М.: Гелиос, 1998.

Урман Л. Oracle 8. Программирование на языке PL/SQL, М.: Лори, 1999.

СОДЕРЖАНИЕ