№ 4559

004
Р 131

РАБОТА С БАЗАМИ ДАННЫХ

Методические указания

НОВОСИБИРСК
2016

Министерство образования и науки Российской Федерации
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

004 № 4559
Р 131

РАБОТА С БАЗАМИ ДАННЫХ

Методические указания к лабораторным работам для студентов IV курса факультета ФПМИ (направления 01.03.02, 02.03.03)

НОВОСИБИРСК
2016

УДК 004.65(076.5)
Р 131

Составители:

канд. техн. наук, доцент *В.М. Стасышин*,
вед. инженер ЦИУ *Т.Л. Стасышина*

Рецензент канд. техн. наук, доцент ТПИ *В.Г. Кобылянский*

Работа подготовлена на кафедре теоретической
и прикладной информатики

ВВЕДЕНИЕ

Задачей курса «Базы данных» является не изучение особенностей той или иной СУБД, а освоение технологий работы с базами данных [1]. В качестве используемой СУБД в лабораторных работах используется система управления базами данных PostgreSQL, хотя в равной степени это могла быть любая из существующих СУБД, функциональные возможности которой не ниже, чем в PostgreSQL.

Лабораторный практикум по курсу «Базы данных» состоит из двух частей.

Первые три лабораторные работы выполняются в интерактивном режиме и знакомят студентов с основными операциями по работе с базами данных (создание базы данных и таблиц, занесение данных, выполнение простейших операций над данными, формирование запросов на языке SQL, передача полномочий на пользование базой данных, работа с внешней базой данных). При этом предполагается, что студенты знакомы с основами языка SQL, например, в объеме пособия «Язык структурных запросов SQL» [2].

Во второй части практикума студенты знакомятся с различными технологиями работы с базами данных.

Четвертая и пятая лабораторные работы посвящены изучению технологии встроенного SQL (встраивание конструкций языка SQL в программу на языке Си, курсоры, динамический SQL).

В шестой лабораторной работе студенты знакомятся со стандартом ODBC для доступа к базам данных.

Две последующие лабораторные работы второй части методического указания посвящены изучению технологий работы с базами данных в среде WWW с использованием языка PHP и ADO.NET.

В девятой лабораторной работ студенты изучают компоненты ADO (Active Data Objects) при создании приложения средствами C++ Builder.

Последняя лабораторная работа второй части методического указания посвящена изучению средств серверного программирования – триггеров и хранимых процедур.

Каждая лабораторная работа содержит необходимые сведения по набору языковых и инструментальных средств, требуемых для выполнения лабораторной работы, порядок ее выполнения, набор вариантов (при необходимости) и перечень вопросов для самопроверки и контроля.

В третьей и четвертой частях приведены методические указания к выполнению проверочной работы по теме «Проектирование информационных систем и баз данных» и курсовой работы по дисциплине «Технологии баз данных» для студентов направления 02.03.03 «Математическое обеспечение и администрирование информационных систем».

ЧАСТЬ I

ОСНОВЫ РАБОТЫ С БАЗАМИ ДАННЫХ

ЛАБОРАТОРНАЯ РАБОТА 1

СОЗДАНИЕ И МОДИФИКАЦИЯ БАЗ ДАННЫХ И ТАБЛИЦ

ЦЕЛЬ РАБОТЫ

Создать схему базы данных, ознакомиться с возможностями веб-приложения phpPgAdmin, создать с его помощью набор таблиц и заполнить таблицы данными для последующей работы.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

1. Ознакомиться с интерфейсом и возможностями программы phpPgAdmin.

2. Изучить набор команд языка SQL, связанный с созданием базы данных, созданием, модификацией структуры таблиц и их удалением, вставкой, модификацией и удалением записей таблиц:

create database – создание базы данных;
database – выбор существующей базы данных;
close database – закрытие текущей базы данных;
drop database – удаление базы данных;
create table – создание таблицы базы данных;
alter table – модификация структуры таблицы базы данных;
drop table – удаление таблицы базы данных;
insert – добавление одной или нескольких строк в таблицу;
delete – удаление одной или нескольких строк из таблицы;
update – модификация одной или нескольких строк таблицы.

3. Из командной строки в существующей базе данных **students** командой

new_schema [*имя схемы*] [*имя базы данных*]

создать схему базы данных с произвольным именем (имя базы данных **students**).

4. Используя программу *phpPgAdmin*, в созданной схеме создать четыре таблицы. При создании таблиц предусмотреть выполнение следующих условий:

– поля номер поставщика, номер детали, номер изделия во всех таблицах имеет символьный тип и длину 6;

– поля рейтинг, вес и количество имеют целочисленный тип;

– поля фамилия, город (поставщика, детали или изделия), название (детали или изделия) имеют символьный тип и длину 20;

– ни для одного поля не предусматривается использование индексов;

– для всех полей допускаются значения NULL и значения-дубликаты, кроме полей номер поставщика из таблицы S, номер детали из таблицы P, номер изделия из таблицы J.

Таблицы S и P создать средствами системы меню программы *phpPgAdmin*, таблицы J и SPJ создать, написав и выполнив соответствующие запросы для создания таблиц (команда *Create table*).

5. Записать и выполнить совокупность запросов для занесения нижеприведенных данных в созданные таблицы:

insert into имя_таблицы [(*поле* [,*поле*]...)]
values (*константа* [,*константа*]...)

Таблица поставщиков (S)

Номер поставщика	Фамилия	Рейтинг	Город
S1	Смит	20	Лондон
S2	Джонс	10	Париж
S3	Блейк	30	Париж
S4	Кларк	20	Лондон
S5	Адамс	30	Афины

Таблица деталей (P)

Номер детали	Название	Цвет	Вес	Город
P1	Гайка	Красный	12	Лондон
P2	Болт	Зеленый	17	Париж
P3	Винт	Голубой	17	Рим
P4	Винт	Красный	14	Лондон
P5	Кулачок	Голубой	12	Париж
P6	Блюм	Красный	19	Лондон

Таблица изделий (J)

Номер изделия	Название	Город
J1	Жесткий диск	Париж
J2	Перфоратор	Рим
J3	Считыватель	Афины
J4	Принтер	Афины
J5	Флоппи-диск	Лондон
J6	Терминал	Осло
J7	Лента	Лондон

Таблица поставок (SPJ)

Номер поставщика	Номер детали	Номер изделия	Количество
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500

S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

Убедиться в успешности выполненных действий. При необходимости исправить ошибки.

6. Проверить результат заполнения таблиц, написав и выполнив простейший запрос

*select * from имя_таблицы*

При наличии ошибок выполнить корректировку, исправив либо удалив ошибочные строки таблиц

delete имя_таблицы where предикат

update имя_таблицы

set поле=выражение [,*поле=выражение*]... *where предикат*

Указанный предикат должен однозначно специфицировать удаляемые либо модифицируемые строки посредством задания соответствующих условий, которым должны удовлетворять отдельные поля строки.

Если посредством значений полей это сделать невозможно, можно прибегнуть к использованию значений скрытого oid-столбца, представляющего собой, по сути, внутренний номер строки. Для этого необходимо предварительно получить значения oid-столбца для занесенных строк

select oid, ** from имя_таблицы*,

а затем требуемые значения использовать при формировании условий в операторах удаления либо модификации.

7. Средствами системы меню программы *phpPgAdmin* выполнить модификацию структуры таблицы SPJ, добавив поле с датой поставки. Убедиться в успешности выполненных действий. При необходимости исправить ошибки.

Занести произвольные даты поставки, использую меню программы *phpPgAdmin*.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие типы данных допустимы при создании таблицы?
2. Как создать таблицу средствами меню программы phpPgAdmin?
3. Как создать таблицу средствами языка SQL?
4. Как разделяются операторы SQL в случае нескольких операторов в запросе?
5. Каким образом выполнить простейшие операции вставки, модификации, удаления строк данных в таблице средствами SQL?
6. Каким образом выполнить простейшие операции вставки, модификации, удаления строк данных в таблице средствами меню программы *phpPgAdmin*?
7. Каким образом выполнить просмотр таблицы?
8. Что такое rowid-столбец (oid-столбец)?
9. Как получить информацию о структуре таблицы в рамках программы *phpPgAdmin*?
10. Какие ограничения на столбцы таблицы можно задать?
11. Как задать ограничения на столбцы таблицы?

ЛАБОРАТОРНАЯ РАБОТА 2

ЗАПРОСЫ К БАЗЕ ДАННЫХ

ЦЕЛЬ РАБОТЫ

Используя данные таблиц, созданных подготовленной в первой лабораторной работе, подготовить и реализовать серию запросов, связанных с выборкой информации и модификацией данных таблиц.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

1. Изучить набор команд языка SQL, связанный с созданием запросов, добавлением, модификацией и удалением строк таблицы:

select – осуществление запроса по выборке информации из таблиц базы данных;

insert – добавление одной или нескольких строк в таблицу;

delete – удаление одной или нескольких строк из таблицы;

update – модификация одной или нескольких строк таблицы;

2. Изучить состав, правила и порядок использования ключевых разделов (секций) оператора select:

select – описание состава данных, которые следует выбрать по запросу (обязательная секция);

from – описание таблицы, из которой следует выбирать данные (обязательная секция);

join – описание дополнительной таблицы, из которой следует выбирать данные, и условия ее соединения с ранее описанными таблицами;

where – описание условий отбора строк (записей);

group by – описание критерия формирования групп строк (группой называется множество строк, имеющих одинаковые значения в указанных столбцах, для каждой группы возвращается одна строка результата);

having – наложение одного или более условий на группу;

order by – сортировка результата выполнения запроса по одному или нескольким столбцам;

into temp – создание временной таблицы, в которую будет осуществлен вывод результатов соответствующего запроса.

Порядок следования секций в команде *select* должен соответствовать приведенной выше последовательности.

3. Подготовить и выполнить средствами программы *phpPgAdmin* четыре запроса по выборке информации из таблиц базы данных для решения приведенных ниже задач.

4. Подготовить и выполнить средствами программы *phpPgAdmin* два запроса по модификации информации (вставка, удаление, замещение) из таблиц базы данных для решения приведенных ниже задач. При этом в тех заданиях, где речь идет о создании таблиц, предполагается формирование постоянной таблицы базы данных.

5. Защитить лабораторную работу, ответив на контрольные вопросы и выполнив проверочные задания.

ВАРИАНТЫ ЗАДАНИЙ ПО ВЫБОРКЕ ИНФОРМАЦИИ

Вариант 1

1. Выбрать поставщиков, поставляющих детали, поставляемые поставщиками, проживающими в Лондоне.

2. Вывести без повторений пары городов таких, что была поставка детали из первого города для изделия во втором городе. Упорядочить список по алфавиту.

3. Получить список городов, в которые выполнили поставки ТОЛЬКО поставщики, поставлявшие голубые детали.

4. Вывести полный список городов и для каждого города найти суммарное количество деталей красного цвета, которые были в него поставлены. Города в списке должны быть ВСЕ. Список должен быть упорядочен по алфавиту.

Вариант 2

1. Выбрать поставщиков, поставляющих детали, поставляемые для изделий из Осло.

2. Найти изделия такие, что детали для изделия поставлялись из того же города, где находится поставщик, выполнивший поставку. Вывести полную информацию об изделиях: номер, название, город.

3. Получить список поставщиков, которые поставляли ТОЛЬКО детали, в названии которых присутствует буква ‘к’.

4. Вывести полный список поставщиков и для каждого поставщика найти суммарное количество деталей с весом больше 17, которые были им поставлены. Поставщики в списке должны быть ВСЕ. Список должен быть упорядочен по номеру поставщика.

Вариант 3

1. Выбрать изделия, для которых поставщик с рейтингом 20 поставлял детали, поставлявшиеся для изделия J2.

2. Найти изделия, для которых детали с весом от 17 до 19 поставлялись поставщиком с рейтингом больше 20. Вывести полную информацию об изделиях: номер, название, город.

3. Получить список поставщиков, выполнивших поставки ТОЛЬКО для изделий с красными деталями.

4. Вывести полный список изделий и для каждого изделия определить, из скольких разных городов для него поставлялись детали с весом 12. Изделия в списке должны быть ВСЕ. Список должен быть упорядочен по номеру изделия.

Вариант 4

1. Выбрать изделия, для которых поставляли детали поставщики, поставлявшие зеленые детали.

2. Найти поставки такие, что поставщик, изделие и деталь размещены в одном и том же городе. Вывести номер поставщика, номер изделия, номер детали и город, где размещены изделие, поставщик и деталь.

3. Получить список деталей, поставленных ТОЛЬКО первым по алфавиту поставщиком.

4. Вывести полный список деталей и для каждой детали определить, сколько поставщиков с рейтингом меньше 30 поставляли эту деталь. Детали в списке должны быть ВСЕ. Список должен быть упорядочен по номеру детали.

Вариант 5

1. Выбрать детали, поставлявшиеся для изделия J7 поставщиками, поставлявшими детали из списка изделия J2.

2. Найти изделия, для которых детали зеленого цвета поставлялись поставщиком, который проживает в городе с буквой «а» в названии. Вывести полную информацию об изделиях: номер, название, город.

3. Получить список деталей, которые поставлялись ТОЛЬКО в города, где проживают поставщики.

4. Вывести полный список деталей и для каждой детали определить, во сколько разных городов поставлялась эта деталь поставщиками с рейтингом меньше 20. Детали в списке должны быть ВСЕ. Список должен быть упорядочен по номеру детали.

Вариант 6

1. Выбрать детали, поставлявшиеся для изделий, для которых поставщик S3 поставлял детали из Рима.

2. Найти поставщиков, которые поставляли детали с весом не меньше 17 для изделий из Рима. Вывести номер поставщика, фамилию, рейтинг и город, где он проживает.

3. Получить список деталей, которые поставлялись ТОЛЬКО поставщиками, выполнившие поставки для изделия J7.

4. Вывести полный список городов и для каждого города найти общее число поставок, выполненных из этого города. Города в списке должны быть ВСЕ. Список должен быть упорядочен по алфавиту.

Вариант 7

1. Выбрать поставщиков, поставляющих голубые детали, поставляемые для изделий с деталью P6.

2. Найти поставки такие, что поставщик, изделие и деталь размещены в разных городах. Вывести номер поставщика, номер изделия, номер детали и города, где размещены изделие, поставщик и деталь.

3. Получить список поставщиков, выполнивших поставки ТОЛЬКО для изделий с длиной названия больше 8.

4. Вывести полный список поставщиков и для каждого поставщика найти количество поставок из городов, где производятся детали голубого цвета. Поставщики в списке должны быть ВСЕ. Список должен быть упорядочен по номеру поставщика.

ВАРИАНТЫ ЗАДАНИЙ ПО МОДИФИКАЦИИ ИНФОРМАЦИИ

Вариант 1

1. Построить таблицу с упорядоченным списком городов, в которых размещается только один поставщик, или производится только одна деталь, или собирается только одно изделие.

2. Увеличить рейтинг всех поставщиков, имеющих в настоящее время рейтинг меньше, чем у поставщика S5, на величину, равную умноженному на 2 числу выполненных поставщиком поставок.

Вариант 2

1. Построить таблицу с упорядоченным списком городов таких, что в городе есть и поставщик, и деталь, и изделие.

2. Для всех поставщиков, имеющих в настоящее время наименьший рейтинг, установить рейтинг равным числу изделий, для которых поставщик выполнил поставки, умноженному на 5.

Вариант 3

1. Построить таблицу с упорядоченным списком городов таких, что в городе производится какая-либо деталь и собирается какое-либо изделие, но не размещается ни один поставщик.

2. Каждое изделие из Афин перевести в город, из которого для изделия сделано наибольшее число поставок. Если таких городов больше одного, перевести в последний по алфавиту из этих городов.

Вариант 4

1. Построить таблицу с упорядоченным списком городов таких, что в городе размещается 2 поставщика и собирается 2 изделия, но не производится ни одна деталь.

2. Всех поставщиков, имеющих в настоящее время наибольший рейтинг, перевести в город, откуда поставщик сделал наибольшее число поставок. Если таких городов больше одного, перевести в первый по алфавиту из этих городов.

Вариант 5

1. Построить таблицу с упорядоченным списком городов таких, что в городе размещается более двух объектов (объектами считаются поставщик, деталь, изделие).

2. Каждую деталь, производимую не в Лондоне, перевести в город, куда сделано наименьшее число поставок детали. Если таких городов больше одного, перевести в первый по алфавиту из этих городов.

Вариант 6

1. Построить таблицу с упорядоченным списком городов таких, что в городе производится какая-либо деталь или собирается какое-либо изделие, но не размещается ни один поставщик.

2. Каждое изделие с длиной названия > 10 перевести в город, в котором проживает первый по алфавиту поставщик деталей для этого изделия.

Вариант 7

1. Построить таблицу с упорядоченным списком городов таких, что в городе собирается какое-либо изделие, но не производится ни одна деталь и не размещается ни один поставщик.

2. Каждую деталь с весом < 17 перевести в город, куда поставлено наибольшее суммарное количество детали. Если таких городов больше одного, перевести в первый по алфавиту из этих городов.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое коррелированный запрос? Чем отличается коррелированный запрос от некоррелированного?
2. Какие существуют ограничения на формирование коррелированного запроса?
3. Каким образом сохранить результаты запроса в таблице?
4. Какими средствами SQL реализуются следующие операции реляционной алгебры: ограничение, декартово произведение, выбор, пересечение, объединение, разность, соединение?
5. Что такое внешнее соединение?
6. Назовите основные агрегатные функции; в чем их отличие от строковых функций?
7. Что помимо имен столбцов может стоять в целевом списке?
8. Перечислите разделы (секции) оператора select. В каких разделах может использоваться подзапрос, а в каких – нет?
9. Что такое внутреннее соединение?
10. Как реализован в SQL квантор существования?
11. Как реализован в SQL квантор всеобщности?
12. В каких случаях вместо фразы *in* можно использовать операцию сравнения?
13. Какие существуют средства группирования в SQL? Как они используются?

ЛАБОРАТОРНАЯ РАБОТА 3

ПОЛНОМОЧИЯ НА ИСПОЛЬЗОВАНИЕ СХЕМЫ БАЗЫ ДАННЫХ. РАБОТА С ВНЕШНИМИ СХЕМАМИ БАЗЫ ДАННЫХ

ЦЕЛЬ РАБОТЫ

Ознакомиться со средствами предоставления полномочий на использование схем и баз данных и таблиц и основами работы с внешними базами данных.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

После создания базы данных (схемы базы данных) пользователь (программа) является исключительным собственником созданной базы данных. Это означает, что ни один другой пользователь (программа) не имеет доступа ни к одной из таблиц схемы базы данных, включая их просмотр, если владелец базы данных не предоставил соответствующих прав.

Предоставление прав реализуется оператором *Grant*.

Оператор Grant в форме

Grant usage on schema schema-name to
{public/<список users>/Group group-name }

предоставляет права указанным категориям на доступ к объектам схемы *schema-name* базы данных.

Назначение опций следующее:

public – пользователи группы *public*;

<список users> – перечень пользователей, например, sb01, sb02 и т. д.

Group group-name – пользователи группы *group-name*.

Оператор Grant в форме

Grant {all/insert/delete/select/update}
on {имя_таблицы/view/synonym}
to {public/<список users>}

предоставляет права на уровне отдельной таблицы.

Назначение опций следующее:

insert, *delete*, *select*, *update* – права на выполнение указанной операции с таблицей;

alle – права на выполнение всех операций;

имя таблицы, *view*, *synonym* – идентификация таблицы, представлений, синонимов.

Отнятие прав реализуется оператором Revoke.

Оператор Revoke в форме

Revoke usage on schema schema-name from
{public/<список users>/Group group-name }

отбирает права у указанной категории на доступ к объектам схемы *schema-name* базы данных.

Оператор Revoke в форме

Revoke {*all/insert/delete/select/update*}
on {*имя_таблицы/view/synonym*}
from {*public/<список users>*}

отнимает права на уровне отдельной таблицы.

Та схема, имя которой зафиксировано в окне редактора SQL программы *phpPgAdmin* (*Schema search path*), является текущей. Любая другая схема базы данных называется внешней. Для ссылки на таблицу во внешней схеме базы данных (при условии, что вам даны такие полномочия владельцем) необходимо указать имя этой схемы базы данных как часть имени таблицы, например, *schema.table*, где *schema* – имя схемы базы данных, *table* – имя таблицы.

Возможна и более общая форма записи:

database.schema.table

где *database* – имя базы данных. Однако в качестве имени базы данных может выступать только текущая база данных (та, к которой осуществлен connect). **СУБД PostgreSQL не поддерживает работу с внешними базами данных.**

Замечание. Ряд СУБД (например, Informix, DB2) поддерживают работу с внешними базами данных, в том числе распределенными. Текущей базой данных является база данных, к которой осуществлен *connect*. Любая другая база данных называется внешней. Для ссылки на таблицу во внешней базе данных

необходимо указать имя этой базы данных как часть имени таблицы, например, *salesdb*:*contracts*, где *salesdb* – имя внешней базы данных, *contracts* – имя таблицы. К имени базы данных можно добавить имя сервера, т. е. сетевой машины, где запущен еще один сервер баз данных баз данных, и таким образом в случае распределенной базы данных обращение к таблице *contracts* базы данных *salesdb*, размещенной на сервере *central*, будет выглядеть следующим образом: *salesdb@central*:*contracts*.

ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Занести в таблицу поставщиков S строки с фамилиями членов бригады.

2. Занести произвольным образом в таблицу поставок SPJ несколько строк (три-пять строк) о поставках, связанных с занесенными фамилиями.

3. Выполнить два запроса к схеме данных согласно номеру вашего варианта. При выполнении запроса данные должны выбираться из таблиц вашей собственной схемы данных.

4. Повторить задание п. 3 с той разницей, что сведения о номенклатуре деталей и изделий (таблицы P и J) должны браться из собственной схемы данных, а сведения о поставщиках и поставках (таблицы S и SPJ) должны браться из схемы данных соседней бригады. Предварительно необходимо узнать имя этой схемы данных. Убедитесь в невозможности выполнить задания.

5. Обеспечить, чтобы владелец используемой вами внешней схемы данных предоставил вам полномочия на просмотр используемых вами таблиц в его схеме данных, дав соответственно ему такие же полномочия для выполнения аналогичных действий.

6. Повторить задание п. 4. Сравнить результаты с результатами, полученными в п. 3.

7. Сделать попытку изменить информацию о поставщиках-владельцах схемы данных (город, рейтинг и т. д.) в таблице S внешней схемы данных. Убедиться в невозможности выполнить задание.

8. Обеспечить, чтобы владелец внешней используемой вами схемы данных предоставил вам полномочия на модификацию данных из используемых вами таблиц в его схемы данных, дав соответственно ему такие же полномочия для выполнения аналогичных действий.

9. Повторить задание п. 7. Проверьте успешность выполнения действий.

10. Дождавшись, когда владелец внешней схемы данных закончит выполнять п. 9, сделать попытку удалить из таблицы S используемой вами внешней схемы данных поставщиков с именами, принадлежащими владельцам схемы данных, и связанные с ними поставки из таблицы SPJ. Убедиться в невозможности выполнить задание.

11. Обеспечить, чтобы владелец используемой вами внешней схемы данных предоставил вам полномочия на удаление из используемых вами таблиц в его схеме данных, дав соответственно ему такие же полномочия для выполнения аналогичных действий.

12. Повторить задание п. 10. Проверьте успешность выполнения действий.

13. Отнять предоставленные вами права на пользование вашей схемой данных.

ВАРИАНТЫ ЗАДАНИЙ

Вариант 1

1. Выдать информацию о каждом изделии в формате:

Изделие № <*номер изделия*> – <*наименование изделия*>

где

- <*номер изделия*> – только цифра,
- <*наименование изделия*> – все буквы маленькие.

Например, строка для изделия J1 должна выглядеть так:

Изделие № 1 – жесткий диск.

2. Получить номера изделий, для которых поставлялась КАЖДАЯ деталь, поставлявшаяся поставщиками с рейтингом не выше 20.

Вариант 2

1. Выдать информацию о каждом изделии в формате:

Город <*город изделия*>: изделие <*номер изделия*> (<*длина названия*>)

где

- < город изделия > – все буквы большие,

- <*номер изделия*> – только цифра,
- <*длина названия*> – цифра.

Например, строка для изделия J1 должна выглядеть так:

Город ПАРИЖ: изделие 1(12).

2. Получить номера изделий, для которых выполнил поставки КАЖДЫЙ поставщик, поставлявший деталь с весом > 17.

Вариант 3

1. Выдать информацию о каждой детали в формате:

Деталь <*номер детали*> <*б1*> – <*б2*> / <*цвет*>

где

- <*номер детали*> – только цифра,
- <*б1*> – первая буква названия,
- <*б2*> – последняя буква названия,
- <*цвет*> – все буквы большие.

Например, строка для детали P1 должна выглядеть так:

Деталь 1 Г-а / КРАСНЫЙ

2. Получить номера деталей, которые поставлялись для КАЖДОГО изделия, для которого поставлялись красные детали.

Вариант 4

1. Выдать информацию о каждой детали в формате:

Деталь <*номер детали*> вес в килограммах = <*вес детали*>

где

- <*номер детали*> – только цифра,
- <*вес детали*> – вес в килограммах с точностью 2 знака после запятой.

Например, строка для детали P1 должна выглядеть так:

Деталь 1 вес в килограммах = 5.45.

2. Получить номера поставщиков, поставлявших детали для КАЖДОГО изделия, имеющего поставки с объемом от 500 до 700.

Вариант 5

1. Выдать информацию о каждой детали в формате:

Город <*город детали*>: <*название детали* > (<*длина названия*>)

где

- <*город детали*> – все буквы большие,
- <*название детали*> – только первые три буквы,
- <*длина названия*>> – цифра.

Например, строка для детали P1 должна выглядеть так:

Город ЛОНДОН: Гай (5).

2. Получить номера поставщиков, поставлявших КАЖДУЮ деталь из списка изделия J5.

Вариант 6

1. Выдать информацию о каждом поставщике в формате:

Город <*город поставщика*>: <*имя поставщика* > (<*рейтинг*>)

где

- <*город поставщика*> – все буквы маленькие,
- <*имя поставщика*> – все буквы большие,
- <*рейтинг*> – цифра

Например, строка для поставщика S1 должна выглядеть так:

Город лондон: СМИТ (20).

2. Получить номера деталей, которые поставлял КАЖДЫЙ поставщик, выполнивший поставки из Рима.

Вариант 7

1. Вывести объем каждой поставки в формате:

Поставка детали <*номер детали* > от <*дата*>
в объеме <*количество*> штук

где

- <*номер детали*> – только цифра,
- <*дата*> – дд-мм-гггг
- <*количество*> – цифра

Например, строка для поставщика S1 должна выглядеть так:

Поставка детали 1 от 01-01-2011 в объеме 200 штук.

2. Получить номера деталей, которые поставлялись для КАЖДОГО изделия, для которого поставлял детали поставщик S4.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Кто является владельцем базы данных?
2. Кто является владельцем схемы данных?
3. Какими правами обладают другие пользователи по отношению к вашей схеме данных?
4. Какими правами обладает администратор базы данных по отношению к вашей схеме данных?
5. Каким образом предоставляются права на пользование схемой данных и отдельными ее таблицами?
6. Каким образом изымаются права на пользование схемой данных и отдельными ее таблицами?
7. Что такое внешняя база данных?
8. Что такое внешняя схема данных?
9. Как идентифицируется таблица внешней схемы данных?
10. Как идентифицируется таблица внешней распределенной базы данных?

ЧАСТЬ II

ТЕХНОЛОГИИ РАБОТЫ С БАЗАМИ ДАННЫХ

ЛАБОРАТОРНАЯ РАБОТА 4

РАБОТА С БАЗОЙ ДАННЫХ СРЕДСТВАМИ ВСТРОЕННОГО SQL

ЦЕЛЬ РАБОТЫ

Приобрести практические навыки работы со встроенным SQL средствами ESQL/C, включая использование курсоров и средств динамического SQL.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

ESQL/C – инструмент разработки приложений над базами данных на языке Си с возможностью использования средств SQL. При создании ESQL/C-программы пользователь разрабатывает С-программу, включает в нее специальные заголовочные файлы и SQL-описания, реализующие работу с базой данных. Препроцессор ESQL/C преобразует SQL-описания в обращения к библиотечным функциям, которые взаимодействуют с сервером базы данных, и дает на выходе С-код. Далее полученный С-код компилируется и линкуется. Исходя из сказанного пользователь, желающий разрабатывать ESQL/C-программы, должен обладать навыками работы с языком Си и уметь пользоваться средствами SQL. При этом необходимо учитывать особенности той операционной системы, в рамках которой создается клиентское приложение. В данной и последующей лабораторных работах предполагается разработка несложных ESQL/C-программ. В лабораторных работах используется база

данных, содержащая четыре таблицы, созданные в первой лабораторной работе:

- таблица поставщиков (S);
- таблица деталей (P);
- таблица изделий (J);
- таблица поставок (SPJ).

Если приведенная схема базы данных отсутствует (или любая таблица из нее), то необходимо восстановить базу данных, пользуясь интерактивными средствами программы *phpPgAdmin* либо выполнив необходимый запрос на языке.

Для разработки ESQL/C-программ необходимо изучить [1]:

- общие правила подготовки программ и использования программных средств ESQL/C;
- аппарат определения и использование главных переменных;
- средства встраивания SQL-описаний в С-программы;
- структуру области связи *SQLCA* и средства обработки ошибок SQL-запросов;
- назначение и структуру заголовочных файлов.

Настоящая лабораторная работа связана с написанием таких программ на ESQL/C, которые после обработки SQL-запроса возвращают единственную строку, а также выполняют простейшие операции по модификации базы данных без использования аппарата курсора.

Исходный файл с программой на ESQL/C должен иметь расширение .ec (например, source.ec).

Вызов транслятора со встроенного ESQL/C выполняется командой pgcci. Ниже приведен упрощенный вариант синтаксиса команды pgcci:

pgcci <имя файла без расширения>

Входные данных для SQL-описаний передаются через так называемые главные переменные, через них же возвращаются результаты запроса, которые отображаются на экране терминала для контроля выполнения. При объявлении главных переменных в языке ESQL/C оператору описания переменных предшествует знак $ либо объявление производится внутри блока:

exec SQL begin declare section
. . . .
exec SQL end declare section

При использовании главных переменных в программе на ESQL/C внутри SQL-описаний им также предшествует знак $, вне SQL-описаний главные переменные используются обычным образом. SQL-описаниям (операторам языка SQL) в программе на языке ESQL/C также предшествует знак $.

Сервер баз данных возвращает код результата и, возможно, другую информацию в структуру данных, называемую областью связи SQL (*SQL Communication Area – SQLCA*). Структура и назначение отдельных полей SQLCA приведены в *Приложении 1*. Структура *SQLCA* описана в заголовочном файле sqlca.h, который автоматически подключается к программе на ESQL/C. Среди других заголовочных файлов отметим:

datatime.h – описывает структуру для типа данных datetime;

decimal.h – описывает структуру для типа данных decimal;

locator.h – описывает структуру для blobs-данных;

varchar.h – описывает структуру для типа данных varchar;

sqlhdr.h – описывает прототипы функций библиотеки ESQL/C;

sqltype.h, *sqltypes.h* – структуры для работы с динамическими главными переменными.

Для работы с объектами схемы базы данных необходимо указать:

- базу данных;
- схему базы данных.

База данных задается оператором в программе на языке ESQL/C

Connect to <имя базы данных>@<имя сервера> USER <логин> using <Пароль>

- имя базы данных, используемое в лабораторных работах 1–10, – *Students*;
- имя сервера сообщает преподаватель;
- имя логина указывается в двойных кавычках;
- пароль указывается в двойных кавычках или посредством главной переменной, подлежащей вводу на этапе выполнения программы.

Для указания схемы базы данных возможны несколько вариантов. Один из них состоит в указании перед запуском загрузочного файла с программой на языке ESQL/C схемы базы данных командой из командной строки:

Set_search_path <имя-схемы> <имя-базы данных>

Обработка многострочного запроса осуществляется с помощью специального объекта данных, называемого курсором. Множество строк, возвращаемое предложением *Select*, называется активным множеством. В процессе использования курсор может задаваться в двух режимах: последовательном и скроллирующем. Последовательный курсор позволяет просматривать активное множество только в последовательном порядке, а также используется при модификации и удалении строк из активного множества.

Скроллирующий курсор позволяет просматривать строки из активного множества в произвольном порядке.

Основные операции при работе с курсором:

- объявление курсора, выполняемое оператором Declare;
- открытие курсора, выполняемое оператором Open;
- выборка по курсору очередной строки запроса в главные переменные, выполняемая оператором Fetch;
- закрытие курсора, выполняемое оператором Close.

С помощью динамического SQL программа-клиент выполняет программное формирование оператора SQL для его последующего исполнения, делая это в три этапа:

- программа собирает текст оператора SQL в виде символьной строки, хранящейся в программной переменной; в общем случае это может быть не один, а несколько операторов SQL, разделенных точкой с запятой;
- программа выполняет оператор Prepare, который обращается к серверу баз данных для анализа текста оператора и подготовки его к выполнению;
- программа использует оператор Execute или средства работы с курсором для выполнения подготовленного оператора.

ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Изучить общие правила подготовки программ и использования программных средств ESQL/C, правила использования главных переменных, средства встраивания SQL-описаний в С-программы, структуру *SQLCA*, назначение и структуру заголовочных файлов.

2. Разработать и отладить ESQL/C-программу, реализующую задачу 1 из соответствующего варианта заданий, результатом которой является единственная строка.

3. Разработать и отладить ESQL/C-программу, реализующую задачу 2 из соответствующего варианта заданий и связанную с модификацией базы данных.

4. Изучить синтаксис и правила использования операторов *Declare*, *Open*, *Fetch*, *Close*, а также особенности работы с курсором.

5. Разработать и отладить набор ESQL/C-программ, решающих задачи 3–5 из соответствующего варианта заданий с использованием аппарата курсоров (последовательного и скроллирующего). Результатом работы программ является набор строк, которые подлежат выводу на экран с соответствующими пояснительными заголовками.

ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОЙ ПРОГРАММЕ

Разрабатываемые ESQL/C-программы должны удовлетворять следующим требованиям:

- обеспечивать необходимую обработку ошибок;
- все действия в отношении базы данных должны выполняться в рамках транзакций (операторы SQL *Begin work*, *Commit work*, *Rollback work*);
- должен быть предусмотрен вывод сообщений обо всех шагах выполнения программы, в том числе и о возможных ошибках;
- программа должна быть достаточно документирована.

ВАРИАНТЫ ЗАДАНИЙ

Вариант 1

1. Выдать число поставок, выполненных для изделий с деталями зеленого цвета.

2. Поменять местами города, где размещены изделия с самым коротким и самым длинным названием, т. е. изделия с самым коротким названием перевести в город, где размещено изделие с самым длинным названием, и наоборот, изделия с самым длинным названием перевести в город, где размещено изделие с самым коротким названием. Если городов несколько, брать первый по алфавиту из этих городов.

3. Найти детали, имеющие поставки, вес которых меньше среднего веса поставок этой детали для изделий из Лондона.

4. Выбрать поставщиков, не поставляющих ни одной из деталей, поставляемых поставщиками, находящимися в Лондоне.

5. Выдать полную информацию о поставщиках, выполнивших поставки ТОЛЬКО с объемом от 200 до 500 деталей.

Вариант 2

1. Выдать число изделий, для которых детали с весом больше 12 поставлял первый по алфавиту поставщик.

2. Поменять местами фамилии первого и последнего по алфавиту поставщика, т. е. первому по алфавиту поставщику установить фамилию последнего по алфавиту поставщика и наоборот.

3. Найти изделия, для которых выполнены поставки, вес которых более чем в 4 раза превышает минимальный вес поставки для изделия. Вывести номер изделия, вес поставки, минимальный вес поставки для изделия.

4. Выбрать поставщиков, не поставивших ни одной из деталей, имеющих наименьший вес.

5. Выдать полную информацию о поставщиках, поставляющих ТОЛЬКО красные детали и только для изделия с длиной названия не меньше 7.

Вариант 3

1. Выдать число деталей, поставлявшихся для изделий, у которых есть поставки с весом от 5000 до 6000.

2. Поменять местами вес деталей из Рима и из Парижа, т. е. деталям из Рима установить вес детали из Парижа, а деталям из Парижа установить вес детали из Рима. Если деталей несколько, брать наименьший вес.

3. Найти детали, имеющие поставки, объем которых не превышает половину максимального объема поставки этой детали поставщиком из Парижа.

4. Выбрать поставщиков, не поставивших ни одной из деталей, поставляемых для изделий из Парижа.

5. Выдать полную информацию о деталях, которые поставлялись ТОЛЬКО поставщиками, проживающими в Афинах.

Вариант 4

1. Выдать число деталей, которые поставлялись поставщиками, имеющими поставки с объемом от 600 до 700 деталей.

2. Поменять местами цвета самой тяжелой и самой легкой детали, т. е. деталям с наибольшим весом установить цвет детали с минималь-

ным весом, а деталям с минимальным весом установить цвет детали с наибольшим весом. Если цветов несколько, брать первый по алфавиту из этих цветов.

3. Найти поставщиков, имеющих поставки, вес которых составляет менее четверти наибольшего веса поставки этого поставщика. Вывести номер поставщика, вес поставки, четверть наибольшего веса поставки поставщика.

4. Выбрать изделия, для которых не поставлялось ни одной из деталей, поставляемых поставщиком S4.

5. Выдать полную информацию о деталях, которые поставлялись ТОЛЬКО поставщиками с максимальным рейтингом.

Вариант 5

1. Выдать число деталей, которые поставлялись поставщиками, выполнявшими поставки в Париж.

2. Поменять местами цвета деталей из Рима и из Лондона, т. е. деталям из Рима установить цвет детали из Лондона, а деталям из Лондона установить цвет детали из Рима. Если цветов несколько, брать первый по алфавиту из этих цветов.

3. Найти поставщиков, имеющих поставки, объем которых не менее чем в 3 раза превышает средний объем поставки этого поставщика для изделий из Рима. Вывести номер поставщика, объем поставки, средний объем поставки поставщика для изделий из Рима.

4. Выбрать изделия, для которых не делал поставок ни один из поставщиков, поставлявших красные детали.

5. Выдать полную информацию об изделиях, для которых поставлялись ТОЛЬКО детали с весом больше 12.

Вариант 6

1. Выдать число поставщиков, поставлявших детали для изделий, собираемых в городе, где производят красные детали.

2. Поменять местами названия первого и последнего по алфавиту изделия, т. е. первому по алфавиту изделию установить название последнего по алфавиту изделия и наоборот.

3. Найти поставщиков, имеющих поставки, объем которых меньше объема наименьшей поставки красных деталей, сделанной этим поставщиком. Вывести номер поставщика, объем поставки, минимальный объем поставки красных деталей поставщиком.

4. Выбрать детали, не поставлявшиеся ни одним поставщиком, поставлявшим детали для изделия J3.

5. Выдать полную информацию об изделиях, у которых есть поставки ТОЛЬКО с весом от 1000 до 7000.

Вариант 7

1. Выдать число цветов деталей, поставлявшихся поставщиками, выполнявшими поставки для изделий из Парижа.

2. Поменять местами названия деталей, стоящих первой и последней в списке, упорядоченном по весу и названию.

3. Найти изделия, имеющие поставки, объем которых более чем в 2 раза превышает средний объем поставки для изделия. Вывести номер изделия, объем поставки, средний объем поставки для изделия.

4. Выбрать изделия, для которых не поставлялась ни одна деталь, имеющая наибольший вес

5. Выдать полную информацию об изделиях, для которых поставлялись ТОЛЬКО детали из последнего по алфавиту города.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое главные переменные? Как они определяются и используются в программах на языке ESQL/C?

2. Каковы правила использования SQL-описаний в программах на ESQL/C?

3. Как обрабатываются NULL-значения в программах на языке ESQL/C?

4. Каково назначение заголовочных файлов?

5. Что такое курсор? В чем отличие последовательного и скроллирующего курсоров по описанию и по использованию?

6. Каковы назначение и синтаксис операторов *Declar*e, *Open*, *Fetch*, *Close*?

7. По какой из команд сервер выделяет память под курсор?

8. По какой из команд сервер начинает поиск строк запроса?

9. Чем заканчивается работа оператора *Open и Fetch*?

10. Каковы назначение и синтаксис операторов *Prepare*, *Execute*?

ЛАБОРАТОРНАЯ РАБОТА 5

ДИНАМИЧЕСКИЙ SQL

ЦЕЛЬ РАБОТЫ

Приобрести практические навыки работы со средствами динамического SQL при написании программ на ESQL/C.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

С помощью динамического SQL программа-клиент выполняет программное формирование оператора SQL для его последующего исполнения, делая это в три этапа:

- программа собирает текст оператора SQL в виде символьной строки, хранящейся в программной переменной; в общем случае это может быть не один, а несколько операторов SQL, разделенных точкой с запятой;
- программа выполняет оператор *Prepare*, который обращается к серверу баз данных для анализа текста оператора и подготовки его к выполнению;
- программа использует оператор *Execute* или средства работы с курсором для выполнения подготовленного оператора.

Динамический оператор SQL по форме напоминает любой другой оператор SQL, записанный в программе, с тем ограничением, что он не может содержать имена главных переменных. Поэтому в динамический оператор *Select* нельзя включать спецификатор *Into*; в любом месте, где главная переменная могла бы появиться в выражении, ей соответствует знак вопроса.

Оператор *Prepare* создает структуру данных, имеющую имя и отображающую строку символов с текстом оператора SQL.

Подготовленный по оператору *Prepare* динамический оператор (группу операторов) можно многократно выполнять. С помощью оператора *Execute* выполняются операторы, отличные от операторов *Select*, а также операторы *Select*, которые возвращают в качестве результата одну строку. Если оператор *Select* возвращает более одной строки, динамический оператор *Select* выполняется не с помощью оператора *Execute*,

а подключается к курсору и в дальнейшем используется обычным образом с помощью курсорных средств. В обоих случаях при выполнении динамического оператора с помощью спецификатора *Using* ему передаются главные переменные, участвующие в выражениях и принимающие возвращаемые значения и, по сути, играющие роль фактических параметров. Как ограничение, следует отметить, что знак вопроса нельзя использовать вместо идентификаторов SQL, таких как имя базы данных, таблицы или столбца: эти идентификаторы должны указываться в тексте оператора при его подготовке, возможно, как полученные из пользовательского ввода.

ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Изучить синтаксис и правила использования операторов *Prepare*, *Execute*, а также особенности работы с курсором при выполнении динамического оператора SQL.

2. Разработать и отладить набор ESQL/C-программ, решающих задачи из соответствующего варианта заданий. Результатом работы программ является одна или несколько строк, которые подлежат выводу на экран с соответствующими пояснительными заголовками.

ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОЙ ПРОГРАММЕ

Разрабатываемые ESQL/C-программы должны удовлетворять следующим требованиям:

- обеспечивать необходимую обработку ошибок;
- использовать аппарат транзакций;
- все используемые в программах операторы SQL, включая операторы, реализующие аппарат транзакций, должны быть динамически подготовлены;
- необходимые параметры, определяющие условия задачи, вводятся с клавиатуры и передаются в строку с текстом динамического оператора SQL;
- все параметры, специфицирующие выполняемые действия, должны передаваться через главные переменные; такими параметрами в условиях задач являются название города, название детали, номер поставщика и т. д.;

- должен быть предусмотрен вывод сообщений обо всех шагах выполнения программы, в том числе и о возможных ошибках;
- программа должна быть достаточно документирована.

ВАРИАНТЫ ЗАДАНИЙ

Вариант 1

1. Получить наибольший объем поставки для каждого изделия и найти их среднее.

2. Для указанного поставщика S* найти средний объем его поставок для каждого из изделий (для которых он поставлял детали). Вывести номер изделия, название изделия, город изделия, средний объем поставок для изделия.

3. Ввести номер изделия J*. Найти цвета деталей, поставлявшихся для изделия J*, и определить, какой процент поставки деталей каждого цвета составляют от общего числа поставок для изделия. Вывести цвет детали, число поставок деталей этого цвета, общее число поставок для изделия J*, процент.

Вариант 2

- Получить минимальный вес поставки для каждого изделия и найти их сумму.
- Для указанного поставщика S* найти число поставок каждой детали, им поставлявшейся. Вывести номер детали, город детали, название детали, число поставок детали.
- Ввести номер изделия J*. Найти поставщиков, поставлявших детали для изделия J*, и определить, какой процент составляет объем поставок каждого поставщика от общего количества деталей, поставленных для изделия. Вывести номер поставщика, объем поставок этого поставщика, общий объем поставок для изделия J*, процент.

Вариант 3

1. Получить число поставок для каждого поставщика и найти их среднее.

2. Для каждого изделия из указанного города найти суммарный объем поставок по каждой детали, для него поставлявшейся. Вывести номер изделия, название изделия, номер детали, название детали, цвет детали, суммарный объем поставок детали для изделия.

3. Ввести номер детали P*. Найти города, в которые поставлялась деталь P*, и определить, какой процент составляют поставки в каждый город от общего числа поставок детали P*. Вывести город, число поставок деталей в этот город, общее число поставок детали P*, процент.

Вариант 4

1. Получить для каждой детали наименьший объем поставки и найти их сумму.

2. Для указанного изделия найти суммарный объем поставок каждой детали, поставленной для него. Вывести номер детали, цвет детали, вес детали, суммарный объем поставок детали для изделия.

3. Ввести номер изделия J*. Найти детали, поставлявшиеся для изделия J*, и определить, какой процент составляет суммарный вес поставок каждой детали от общего веса деталей, поставленных для изделия. Вывести номер детали, суммарный вес поставок этой детали, общий вес деталей для изделия J*, процент.

Вариант 5

1. Получить для каждого поставщика максимальный вес поставки и найти их среднее.

2. Для каждого поставщика из указанного города (Парижа) найти максимальный вес поставки по каждой детали, им поставлявшейся. Вывести номер поставщика, имя поставщика, номер детали, вес детали, максимальный вес поставки детали поставщиком.

3. Ввести номер поставщика S*. Найти города, из которых поставлял детали поставщик S*, и определить, какой процент поставки из каждого города составляют от общего числа поставок поставщика S*. Вывести город, число поставок из этого города, общее число поставок поставщика S*, процент.

Вариант 6

1. Получить для каждого поставщика средний вес поставки и найти их сумму.

2. Для указанной детали (P3) найти средний вес поставки для каждого изделия, для которого деталь поставлялась. Вывести номер изделия, название изделия, город изделия, средний вес поставки.

3. Ввести номер детали P*. Найти изделия, для которых поставлялась деталь P* , и определить, какой процент составляет объем поста-

вок для каждого изделия от общего объема поставок детали P*. Вывести номер изделия, объем поставок для этого изделия, общий объем поставок детали P*, процент.

Вариант 7

1. Получить для каждой детали суммарный объем поставок и найти их среднее.

2. Для каждой детали из указанного города (Лондона) найти объем ее наименьшей поставки для каждого изделия, для которого деталь поставлялась. Вывести номер детали, город детали, номер изделия, название изделия, объем наименьшей поставки детали для изделия.

3. Ввести номер поставщика S*. Найти цвета деталей, которые поставлял поставщик S*, и определить, какой процент поставки деталей каждого цвета составляют от общего числа поставок поставщика S*. Вывести цвет детали, число поставок деталей этого цвета, общее число поставок поставщика S*, процент.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каковы назначение и синтаксис оператора *Prepare*?
2. Каковы назначение и синтаксис оператора *Execute*?
3. Каковы особенности использования динамических операторов SQL?
4. Что такое динамические главные переменные? Каково их назначение?
5. С какими операторами связано использование динамических главных переменных?
6. Каково назначение оператора *Execute Immediate*?

ЛАБОРАТОРНАЯ РАБОТА 6

РАБОТА С БАЗОЙ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ СРЕДСТВ ODBC

ЦЕЛЬ РАБОТЫ

Ознакомиться со структурой программ, использующих средства ODBC, базовыми функциями ODBC, реализовать и выполнить с использованием простейших функций ODBC запросы, связанные с модификацией таблиц и выборки данных из результирующего множества.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

1. Ознакомиться со структурой программы ODBC.
2. Изучить функции выполнения подготовительных операций в ODBC-программе.
3. Ознакомиться со средствами обработки ошибок в ODBC-программе.
4. Изучить функции непосредственного и подготавливаемого выполнения SQL-операторов, передачи параметров.
5. Изучить базовые функции выборки данных *sqlbindcol*, *sqlfetch*, *sqlgetdata*.
6. Ознакомиться с алгоритмами извлечения данных из результирующего множества с использованием средств ODBC.
7. Настроить среду выполнения, разработать и отладить ODBC-программу.

1. Структура ODBC-программы и функции инициализации

Общая структура ODBC-программы представлена на рис. 1.

Идентификатор окружения каждого приложения ODBC создается функцией *SQLAllocEnv* и освобождается в конце приложения с помощью функции *SQLFreeEnv*. Тип идентификатора окружения – *HENV*. Синтаксис функций *SQLAllocEnv* и *SQLFreeEnv* приведен ниже.

RETCODE SQLAllocEnv (*env*);

HENV env; – указатель области хранения в памяти идентификатора окружения.

Рис. 1. Последовательность действий при создании ODBC-программы

RETCODE SQLFreeEnv (*env*);

HENV env; – имя идентификатора окружения, который должен быть освобожден.

Идентификатор соединения представляет собой соединение между источником данных и прикладной программой. Для каждого источника данных, с которым приложение предполагает соединиться, должен быть назначен идентификатор соединения функцией *SQLAllocConnect* и освобожден функцией *SQLFreeConnect*. Приложение может соединиться с источником данных, используя функцию *SQLConnect*, и разъединиться, используя функцию *SQLDisconnect*. Тип идентификатора соединения – *HDBC*. Синтаксис функций *SQLAllocConnect*, *SQLFreeConnect*, *SQLConnect* и *SQLDisconnect* приведен ниже.

RETCODE SQLAllocConnect (*env*, *dbc*);

HENV env; – указатель на идентификатор окружения прикладной программы.

HDBC dbc; – указатель области хранения памяти для идентификатора соединения.

RETCODE SQLFreeConnect (*dbc*);

HDBC dbc; – указатель области памяти для освобождаемого идентификатора соединения.

RETCODE SQLConnect (*dbc*, *szDSN*, *sbDSN*, *szUID*, *sbUID*, *szAuthStr*, *cbAuthStr*);

HDBC dbc; – идентификатор соединения;

UCHAR szDSN; – строка с именем источника данных, с которым прикладная программа собирается соединиться;

SWORD sbDSN; – длина строки источника данных; если это имя имеет нулевое окончание, то этот параметр можно установить в *SQL_NTS*, который является константой ODBC и используется вместо длины параметра, если параметр содержит строку с нулевым окончанием;

UCHAR szUID; – имя пользователя;

SWORD sbUID; – длина имени пользователя или *SQL_NTS*;

UCHAR szAuthStr; – пароль пользователя;

SWORD cbAuthStr; – длина пароля.

RETCODE SQLDisconnect (*dbc*);

HDBC dbc; – идентификатор доступа для отсоединения.

Идентификатор оператора аналогичен идентификатору окружения или соединения, за исключением того, что он ссылается на SQL-оператор. Идентификатор соединения может быть связан несколькими идентификаторами операторов, но каждый идентификатор оператора связан только со своим идентификатором соединения. Чтобы назначить идентификатор оператора, приложение вызывает функцию *SQLAllocStmt*, а для освобождения – функцию *SQLFreeStmt*. Тип идентификатора оператора – *HSTMT*. Синтаксис функций *SQLAllocStmt* и *SQLFreeStm* приведен ниже:

RETCODE SQLAllocStmt (*dbc*, *stmt*);

HDBC dbc; – идентификатор соединения;

HSTMT stmt; – указатель области хранения в памяти для идентификатора оператора;

RETCODE SQLFreeStmt (*stmt*, *fOption*);

HSTMT stmt; – идентификатор оператора;

UWORD fOption; – одна из следующих опций:

SQL_CLOSE – закрывает курсор, связанный с *hstmt* (если он был определен), и отбрасывает все ожидаемые результаты. Прикладная программа может вновь открыть этот курсор позднее, вновь выполнить оператор *Select* теми же самыми или другими значениями параметров. Если курсор не открыт, то эта опция не повлияет на программу.

SQL_DROP – освобождает идентификатор оператора *hstmt*, освобождает все ресурсы, связанные с ним, закрывает курсор, если он открыт, и отбрасывает все ожидаемые строки. Эта опция завершает все обращения к идентификатору *hstmt*. Идентификатор оператора *hstmt* обязательно должен быть переназначен для повторного использования. Эта опция освобождает все ресурсы, которые были определены с помощью функции *SQLFreeStmt*.

SQL_UNBIND – освобождает все буферы столбцов, которые повторно используются функцией *SQLBindCol* для данного идентификатора оператора.

SQL_RESET_PARAMS – освобождает все буферы параметров, которые были установлены функцией *SQLBindCol* для данного идентификатора оператора.

2. Средства отслеживания ошибок

Для отслеживания ошибок в ODBC используется функция *SQLError*, которая возвращает сообщение об ошибке при неудачном завершении какой-либо функции ODBC.

Каждая ODBC-функция возвращает *RETCODE*, который принимает одно из следующих значений:

SQL_SUCCESS	Операция выполнена без ошибки.
SQL_SUCCESS_WITH_INFO	Функция завершена, но при ее вызове SQLError указывает на ошибку. В большинстве случаев это возникает, когда значение, которое должно быть возвращено, большего размера, чем это предусмотрено буфером прикладной программы.
SQL_ERROR	Функция не была завершена из-за возникшей ошибки. При вызове функции *SQLError* можно будет получить больше информации о сложившейся ситуации.
SQL_INVALID_HANDLE	Неправильно определен идентификатор окружения, соединения или оператора. Это часто случается, когда идентификатор используется после того, как он был освобожден, или если был определен нулевой указатель.
SQL_NO_DATA_FOUND	Больше нет подходящей информации. Фактически это не является ошибкой. Чаще всего такой статус возникает при использовании курсора, когда больше нет строк для его продвижения.
SQL_NEED_DATA	Необходимы данные для параметра.

Синтаксис функции *SQLError* приведен ниже:

RETCODE SQLError (*henv*, *hdbc*, *hsmt*, *szSqlState*, *pfNativeError*, *szErrorMsg*, *cbErrorMsgMax*, *cbErrorMsg*)

HENV henv; – идентификатор окружения;
HDBC hdbc; – идентификатор соединения;
HSTMT hsmt – идентификатор оператора;

UCHAR szSqlState; – указание возвратить *SQLSTATE* в качестве строки завершения;

SDWORD pfNativeError; – параметр, возвращающий ошибку, возникшую в СУБД, а также ее собственный код; если соответствующего собственного кода ошибки не существует, то возвращается нуль;

UCHAR szErrorMsg; – указатель на буфер, куда будет возвращен текст ошибки (строка с нулевым окончанием);

SWORD cbErrorMsgMa; – максимальный размер описанного выше буфера, который должен быть не более *SQL_MAX_MESSAGE_LENGTH-1*;

SWORD cbErrorMsg; – число байт, скопированных в буфер.

3. Непосредственное и подготавливаемое выполнение операторов SQL

Непосредственное выполнение используется в тех случаях, когда:

- SQL-операторы, которые должны быть выполнены, будут выполняться только один раз;
- не требуется информации о результирующем множестве до выполнения оператора.

SQLExecDirect представляет собой самый быстрый способ запустить SQL-оператор при одноразовом выполнении. Синтаксис функции *SQLExecDirect* приведен ниже:

RETCODE SQLExecDirect (*hstmt*, *szSqlStr*, *cbSqlStr*);
HSTMT hstmt; – идентификатор оператора;
UCHAR szSqlStr; – строка с SQL-оператором;
SDWORD cbSqlStr; – длина строки *szSqlStr*.

Подготавливаемое выполнение предпочтительнее использовать, когда необходима информация о результирующем множестве до выполнения оператора или когда требуется выполнить SQL-оператор более одного раза. Для этого необходимы две функции: *SQLPrepare* и *SQLExecute*.

Функция *SQLPrepare* подготавливает SQL-строку для выполнения:

RETCODE SQLPrepare (*hstmt*, *szSqlStr*, *cbSqlStr*);
HSTMT hstmt; – идентификатор оператора;
UCHAR szSqlStr; – строка с SQL-оператором;
SDWORD cbSqlStr; – длина строки szSqlStr.

Функция *SQLExecute* выполняет подготовленный оператор:

RETCODE SQLExecute(*hstmt*);
HSTMT hstmt; – идентификатор оператора.

Функции *SQLPrepare* и *SQLExecDirect* отличаются тем, что при вызове *SQLPrepare* оператор SQL в действительности не выполняется, вместо этого определяется путь доступа к данным источника данных. Использование подготавливаемого выполнения удобно для операторов, которые будут выполняться более одного раза. Так как путь доступа к данным уже определен, выполнение может осуществляться несколько быстрее, чем при использовании *SQLExecDirect*. Кроме того, каждый вызов *SQLExecute* передает базе данных только идентификатор для планирования обращения, а не весь SQL-оператор.

4. Использование параметров при выполнении

Параметры используются при непосредственном и подготавливаемом выполнении. Маркеры параметров определяются в SQL-операторах с помощью знаков «?». Например,

SELECT n_post FROM s WHERE town=?

или

INSERT INTO p(*name*, *town*) *VALUES* (*?,?*).

Для того чтобы связать буфер с маркерами параметров, прикладная программа должна вызвать функцию *SQLBindParameter*:

RETCODE SQLBindParameter (*hsmt*, *ipar*, *fParamType*, *fCType*, *fSqlType*, *cbColDef*, *ibScale*, *rgbValue*, *cbValueMax*, *pcValue*);

HSTMT hsmt; – идентификатор оператора;

UWORD ipar; – номер параметра для связи; в операторе SQL параметры нумеруются слева направо, начиная с 1; например, для следующего SQL-оператора используются три параметрических маркера: *INSERT INTO j* (*n_izd*, *name*, *town*) *VALUES* (*?,?,?*); чтобы связать эти параметры, функция *SQLBindParameter* вызывается три раза со значением *ipar*, установленным в 1, 2 и 3 соответственно;

SWORD fParamType; – тип параметра для связи, принимающий одно из трех значений: *SQL_PARAM_INPUT*, *SQL_PARAM_INPUT_OUTPUT* или *SQL_PARAM_OUTPUT*; первое значение используется для процедур с параметрами ввода, второе значение маркирует параметр

ввода/вывода в процедуре, третье значение маркирует значение возврата или параметр вывода в процедуре;

SWORD fCType; – С-тип данных для параметра, из которого необходимо конвертировать данные;

SWORD fSqlType; – ODBC-тип данных для параметра, в который конвертируются данные и который должен совпадать с SQL-типом столбца, соответствующего этому параметрическому маркеру;

UDWORD cbColDef; – точность столбца или выражения соответствующего маркера параметра;

SWORD ibScale; – размер столбца или выражения соответствующего маркера параметра;

PTR rgbValue; – указатель буфера для данных параметра, который при вызове функций *SQLExecute* или *SQLExecuteDirect* содержит действительные значения параметра;

SDWORD cbValueMax; – максимальная длина буфера *rgbValue*;

SDWORD pcbValue; – указатель буфера для длины параметра.

Ниже приведены основные значения С- и SQL-типов параметров.

С-тип		SQL-тип	
SQL_C_BINARY	SQL_C_FLOAT	SQL_BINARY	SQL_DOUBLE
SQL_C_BIT	SQL_C_TIME	SQL_BIT	SQL_FLOAT
SQL_C_CHAR	SQL_C_DEFAULT	SQL_CHAR	SQL_INTEGER
SQL_C_DATE	SQL_C_SLONG	SQL_DATE	SQL_REAL
SQL_C_DOUBLE	SQL_C_SSHORT	SQL_DECIMAL	SQL_SMALLINT
		SQL_TIME	SQL_VARCHAR

5. Функции выборки данных

В ODBC существуют две функции базового уровня для выборки результатов – *SQLBindCol* и *SQLFetch*. Функция *SQLBindCol* определяет область хранения данных результирующего множества, функция *SQLFetch* осуществляет выборку данных в области хранения.

Каждый столбец, который требуется выбрать, связывается с помощью отдельного вызова функции *SQLBindCol*. Функция *SQLBindCol* назначает область хранения в памяти и тип данных для столбца результирующего множества. Она определяет:

- буфер хранения для получения содержимого столбца данных в результирующем множестве;

- длину указанного буфера хранения;
- область памяти для хранения длины столбца выборки;
- преобразование типа данных.

Алгоритм программы, использующей функции *SQLFetch* и *SQLBindCol* для возвращения данных из результирующего множества, предполагает выполнение следующих шагов.

1. Вызывается функция *SQLBindCol* один раз для каждого столбца, который должен быть возвращен из результирующего множества.

2. Вызывается функция *SQLFetch* для перемещения курсора на следующую строку и возврата данных из связанных столбцов.

3. Повторяется шаг 2 до тех пор, пока функция *SQLFetch* не возвратит значение *SQL_NO_DATA_FOUND*. Это указывает на то, что был достигнут конец результирующего множества. Если результирующее множество является пустым, значение *SQL_NO_DATA_FOUND* будет возвращено при первом вызове *SQLFetch*.

Синтаксис функции *SQLBindCol* приведен ниже:

RETCODE SQLBindCol (*hstmt*, *icol*, *fcType*, *rgbValue*, *cbValueMax*, *pcbValue*);

HSTMT stmt; – идентификатор оператора;

UWORD icol; – идентификатор окружения;

SWORD fCtype; – С-тип данных результирующего множества;

PTR rgbValue; – указатель области хранения данных; если *rgbValue* является нулевым указателем, то драйвер «отвязывает» столбец; для отвязывания всех столбцов прикладная программа вызывает *SQLFreeStmt* с опцией *SQL_UNBIND*;

SDWORD cbValueMax; – максимальная длина буфера *rgbValue*; для символьных данных *rgbValue* должен также включать в себя место для байта нулевого окончания;

SDWORD pcbValue; – число байт, которое может возвращаться в *rgbValue* при вызове *SQLFetch*.

Как только все требуемые столбцы связаны, вызывается функция *SQLFetch* для выборки данных в определенной области хранения. Синтаксис функции *SQLFetch* приведен ниже:

RETCODE SQLFetch (*hstmt*);

HSTMT stmt; – идентификатор оператора.

Функция *SQLGetData* позволяет выполнить выборку данных из столбцов, для которых область хранения не была заранее подготовлена с помощью функции *SQLBindCol*. Функция *SQLGetData* вызывается

после вызова функции *SQLFetch* для выборки данных из текущей строки.

Алгоритм программы, использующей функции *SQLFetch* и *SQLGetData* для извлечения данных из каждой строки результирующего множества, предполагает выполнение следующих шагов.

1. Вызывается функция *SQLFetch* для продвижения на следующую строку.

2. Вызывается функция *SQLGetData* для каждого столбца, который должен быть возвращен из результирующего множества.

3. Повторяются шаги 1 и 2 до тех пор, пока функция *SQLFetc*h не возвратит значение *SQL_NO_DATA_FOUND*. Это указывает на то, что был достигнут конец результирующего множества. Если результирующее множество является пустым, то значение *SQL_NO_DATA_FOUND* будет возвращено при первом вызове *SQLFetch*.

Синтаксис функции *SQLGetData* приведен ниже:

RETCODE SQLGetData (*hstmt*, *icol*, *fcType*, *rgbValue*, *cbValueMax*, *pcbValue*);

HSTMT stmt; – идентификатор оператора;

UWORD icol; – номер столбца результирующего множества, упорядоченный слева направо, начиная с 1;

SWORD fCtype; – С-тип данных результирующего множества;

PTR rgbValue; – указатель области хранения данных. Если *rgbValue* является нулевым указателем, то драйвер «отвязывает» столбец. Для отвязывания всех столбцов прикладная программа вызывает функцию *SQLFreeStmt* с опцией *SQL_UNBIND*;

SDWORD cbValueMax; – максимальная длина буфера *rgbValue*; для символьных данных *rgbValue* должен также включать в себя место для байта нулевого окончания;

SDWORD FAR pcbValue*; – число байт, которое может возвращаться в *rgbValue* при вызове *SQLGetData*.

Замечание. Прикладная программа может использовать *SQLBindCol* для некоторых столбцов, а *SQLGetData* – для других столбцов в пределах той же самой строки.

6. Настройка доступа к источнику данных

Настройка доступа к источнику данных включает:

- редактирование файла .odbc.ini.;
- определение переменной ODBCINI;

- установку переменных окружения СУБД;
- включение необходимых заголовочных файлов.

Для настройки источника данных необходимо проверить, находится ли системный текстовый файл *.odbc.ini* в домашней директории пользователя, и отредактировать его, настроив на требуемые источники данных.

В файле *.odbc.ini* должен быть под некоторым идентификатором описан требуемый источник данных, имя которого используется функцией *SQLConnect*, и далее должен присутствовать раздел с данным именем, в котором содержатся атрибуты, описывающие источник данных. Файл *.odbc.ini* должен содержать имя источника данных, имя сервера баз данных, имя базы данных и другие атрибуты.

В переменной окружения *ODBCINI* необходимо указать полное имя системного файла *.odbc.ini* из домашней директории. Сделать это можно, либо введя с консоли соответствующую команду, либо поместив эту команду в файл загрузки *.login*. Форма записи команды зависит от используемой программы Shell-интерпретатора.

Замечание. Системные требования зависят от той СУБД, с которой работает пользователь. Например, при работе с СУБД Informix это обеспечивается переменными окружения, выставленными в файле *.cshrc* в домашней директории пользователя.

Для получения доступа до ODBC-функций в программе должен быть описан заголовочный файл *sqlext.h*. В файле *sqltypes.h* находятся описания типов данных, используемых в ODBC.

Как и любой исходный файл, написанный на языке Си, файл с программой, вызывающей ODBC-функции, должен иметь расширение .c. При компиляции следует подключить необходимые библиотеки, используемые в вызываемых функциях. Поскольку в этом случае командная строка принимает достаточно сложный вид, целесообразно на уровне системного администратора оформить командный файл, в котором подключаются необходимые библиотеки.

ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Убедиться в наличии и заполненности базы данных поставщиков, деталей, изделий, поставок.

2. Разработать ODBC-программу для решения задачи 1 из соответствующего варианта с помощью функций непосредственного выполнения.

3. Разработать ODBC-программу для решения задачи 2 из соответствующего варианта с помощью функций подготавливаемого выполнения.

4. После выполнения задач 1–2 привести базу данных в исходное состояние.

5. Разработать ODBC-программу для решения задачи 3 из соответствующего варианта с помощью функций *SQLBindCol*, *SQLFetch*.

6. Разработать ODBC-программу для решения задачи 3 из соответствующего варианта с помощью функций *SQLFetch*, *SQLGetData*.

ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОЙ ПРОГРАММЕ

Разрабатываемая программа должна удовлетворять следующим требованиям:

- все используемые функции ODBC должны анализироваться на корректность кода возврата;
- в программе должен быть предусмотрен вывод сообщений обо всех шагах ее выполнения, в том числе и о возможных ошибках;
- при выполнении запросов должно быть предусмотрено использование параметров;
- параметры варианта задания должны быть введены в ходе выполнения программы и переданы в SQL-запрос;
- при выполнении программы должна контролироваться целостность базы данных;
- программа должна быть достаточно документирована.

ВАРИАНТЫ ЗАДАНИЙ

Вариант 1

1. Удалить поставки заданной детали, сделанные указанным поставщиком для изделия с наибольшим числом поставок.

2. Увеличить рейтинг поставщика, выполнившего наибольшую поставку заданной детали, на указанную величину.

3. Для каждого поставщика вывести его номер и признак наличия у него поставок указанной детали (1 – поставки были, 0 – не было). Поставщики в списке должны быть ВСЕ. Список должен быть упорядочен по номеру поставщика.

Вариант 2

1. Удалить того поставщика из заданного города, который сделал наименьшее число поставок.

2. Изменить цвет самой тяжелой детали, поставленной для указанного изделия, на заданное значение.

3. Для каждого поставщика вывести его номер и объем поставок, сделанный поставщиком для указанного изделия (или 0, если поставок не было). Поставщики в списке должны быть ВСЕ. Список должен быть упорядочен по номеру поставщика.

Вариант 3

1. Вставить поставщика с заданным именем, а также городом и рейтингом, как у поставщика с наименьшим числом поставок.

2. Удалить самую легкую из деталей, поставлявшихся для изделия из указанного города.

3. Для каждого поставщика вывести его номер и число сделанных им поставок с объемом больше указанного значения (или 0, если таких поставок не было). Поставщики в списке должны быть ВСЕ. Список должен быть упорядочен по номеру поставщика.

Вариант 4

1. Удалить поставщика, выполнившего меньше всего поставок указанной детали.

2. Изменить вес детали, для которой заданный поставщик сделал наибольшее число поставок, на указанное значение.

3. Для каждого изделия вывести его номер и признак наличия поставок для этого изделия указанным поставщиком (‘да’, если поставки

были, ‘нет’ – не было). Изделия в списке должны быть ВСЕ. Список должен быть упорядочен по номеру изделия.

Вариант 5

1. Удалить изделие из заданного города, имеющее наименьшее число поставщиков.

2. В таблице поставок для заданных номеров детали и изделия изменить номер поставщика на указанное значение.

3. Для каждого изделия вывести его номер и объем максимальной поставки для этого изделия указанной детали (или 0, если поставок не было). Изделия в списке должны быть ВСЕ. Список должен быть упорядочен по номеру изделия.

Вариант 6

1. Увеличить рейтинг поставщика, выполнившего наибольший суммарный объем поставок для заданного изделия, на указанную величину.

2. Вставить деталь с заданным названием и остальными параметрами, такими же, как у детали с наименьшим числом поставок.

3. Для каждого изделия вывести его номер и число поставок для этого изделия с объемом в диапазоне между двумя указанными значениями (или 0, если таких поставок не было). Изделия в списке должны быть ВСЕ. Список должен быть упорядочен по номеру изделия.

Вариант 7

1. Изменить название самой тяжелой детали из города, где проживает указанный поставщик, на заданное значение.

2. Удалить все поставки поставщика, имеющего указанное число изделий, для которых он поставлял детали.

3. Для каждой детали вывести ее номер и признак наличия поставок для указанного изделия (1 – поставки были, 0 – не было). Детали в списке должны быть ВСЕ. Список должен быть упорядочен по номеру детали.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какова структура ODBC-программы? Перечислите ее основные компоненты.
2. С помощью каких средств ODBC можно отследить наличие ошибки?
3. В каких случаях непосредственное выполнение операторов наиболее эффективно?
4. Когда используется подготавливаемое выполнение?
5. Как описываются маркеры параметров и какая для этого предусмотрена функция? Каким образом можно связать несколько параметров?
6. С помощью какого параметра можно освободить буферы всех столбцов?
7. Как описать доступ к необходимой базе данных?
8. С помощью какой функции описывается соединение с необходимым источником данных? Каковы ее параметры?
9. Какие существуют базовые функции выборки данных? Охарактеризуйте каждую из них.
10. Как «отвязать» отдельный столбец результирующего множества?
11. Каким образом можно «отвязать» все столбцы результирующего множества?
12. Что возвращает функция *SQLFetch*, если в результирующем множестве больше не осталось данных?
13. Как определить область хранения данных?
14. Каков алгоритм выборки данных с помощью курсора?
15. Как работает функция *SQLFetch*?
16. Можно ли использовать *SQLFetch* для продвижения курсора в обратном направлении?

ЛАБОРАТОРНАЯ РАБОТА 7

ИСПОЛЬЗОВАНИЕ ЯЗЫКА PHP ДЛЯ ДОСТУПА К БАЗАМ ДАННЫХ

ЦЕЛЬ РАБОТЫ

Ознакомиться с базовыми конструкциями языка PHP с целью написания с их использованием простейших PHP-программ доступа к базам данных.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

1. Изучить основы языка разметки гипертекста HTML.
2. Изучить необходимые конструкции HTML-формы.
3. Ознакомиться с синтаксисом языка PHP.
4. Изучить особенности передачи значений переменных HTML-формы в переменные PHP.
5. Ознакомиться с набором функций для общения с СУБД postgresql.
6. С использованием средств языка PHP разработать и отладить программу доступа к базе данных.

Язык PHP – это действующий на стороне сервера встраиваемый в HTML язык, имеющий синтаксис, близкий к языку Си. Язык PHP дает возможность вставлять в файлы HTML инструкции языка PHP для создания динамического содержания. Эти инструкции обрабатывает препроцессор-интерпретатор PHP и заменяет их тем содержимым, которое производит этот код. PHP-программа может целиком состоять из конструкций языка PHP, а может быть смесью конструкций языков PHP и HTML. Стандартное расширение файла с PHP-программой – *php*. Схема обработки PHP-программы приведена на рис. 2.

Одним из распространенных применений PHP является работа с базами данных. Для целого ряда баз данных PHP имеет собственную поддержку, а другие доступны через ODBC-функции PHP. При вызове PHP-программы URL-адрес должен содержать номер порта, через который работает PHP:

html://fpm.ami.nstu.ru:81/~pmxxyy/t1.php

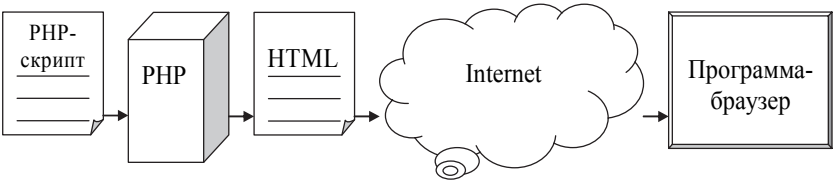


Рис. 2. Схема обработки PHP-программы

К особенностям языка PHP относятся:

- возможность встраивать конструкции языка PHP в HTML-документ;
- возможность включать в PHP-программу файлы;
- наличие достаточного набора встроенных функций;
- возможность определять собственные переменные, строки, массивы, объекты;
- наличие необходимого набора управляющих структур;
- возможность вводить собственные функции.

Одна из наиболее удобных особенностей PHP – это способность автоматически передавать значения переменных из HTML-форм в переменные PHP. Язык PHP автоматически генерирует набор переменных, имена которых совпадают с именами объектов в HTML-форме и содержащих значения данных объектов. В результате отпадает необходимость в выполнении рутинного преобразования, связанного с разбором последовательности

имя=значение&имя1=значение1&...&имяN=значениеN.

Для связи с любой из СУБД язык PHP в своем наборе имеет ряд функций, которые очень похожи между собой и имеют одинаковую логику работы и аналогичные параметры.

В приведенной ниже таблице представлен минимальный набор функций, необходимых для написания PHP-программ, общающихся с СУБД PostgeSQL.

ФУНКЦИИ ВЗАИМОСВЯЗИ С СУБД PostgeSQL

1. **resource Pg_connect (строка_соединения)**	Создание соединения с сервером PostgreSQL
Входные параметры	Строка, содержащая параметры соединения (адрес сервера, порт, имя базы данных, имя пользователя, пароль и пр.)

Продолжение таблицы

	Возвращаемое значение	Пример: $dbconn = pg_connect («host=fpm2 port=5432 dbname=students user=pm2101 password=pass»); идентификатор соединения, если соединение прошло успешно, и FALSE в противном случае
2.	**resource Pg_query (id_соединения, строка_запроса)**	Выполнение запроса к базе
	Входные параметры	id_соединения – идентификатор соединения строка_запроса – строка SQL-запроса
	Возвращаемое значение	Результат в виде ресурса или FALSE
3.	**array Pg_fetch_row(resource результат_запроса, int номер_строки)** **array Pg_fetch_assoc (resource результат_запроса, int номер_строки)**	Получить строку запроса как нумерованный массив Получить строку запроса как ассоциативный массив
	Входные параметры	результат_запроса – идентификатор результата, возвращенный функцией Pg_query() (только для запросов типа select); номер_строки – целое число
	Возвращаемое значение	Строка таблицы базы данных, возвращаемая как массив
4.	**int Pg_affected_rows (resource result_id)**	Получение числа столбцов, обработанных запросом
	Входные параметры	result_id – результат, возвращенный функцией Pg_query()
	Возвращаемое значение	Возвращается число столбцов, обработанных запросом, ассоциированных с result_id. Для вставок, обновлений и удалений – это реальное количество обработанных столбцов. Для выборок – ожидаемое количество
5.	**bool pg_free_result (resource result_id)**	Освобождение ресурсов запроса

Окончание таблицы

Входные параметры	result_id – результат, возвращенный функцией Pg_query()
Возвращаемое значение	Освобождает ресурсы, занятые запросом с идентификатором результата result_id. Возвращает 0 в случае ошибки.
6. **bool pg_close (resource link_id)**	Закрытие соединения с PostgreSQL
Входные параметры	link_id – идентификатор соединения;
Возвращаемое значение	1 в случае успешного закрытия соединения с PostgreSQL и 0 в случае ошибки

Общая схема PHP-программы, выполняющей взаимодействие с базой данных, содержит следующие этапы:

- подключиться к серверу баз данных и зарегистрироваться;
- выбрать базу данных, которая будет использоваться;
- отправить запрос SQL на сервер и получить данные;
- отключиться от сервера баз данных.

При этом остаются актуальными все замечания, сделанные в предыдущей лабораторной работе относительно установки переменных окружения и обеспечения мер безопасности при работе с базой данных.

ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Убедиться в наличии и заполненности таблиц поставщиков, деталей, изделий, поставок.

2. Разработать и отладить HTML-формы для ввода данных пользователя согласно варианту задания (можно модифицировать HTML-формы из предыдущей лабораторной работы).

3. Разработать и отладить PHP-программы для обработки данных HTML-форм и доступа к базе данных.

4. После выполнения лабораторной работы привести базу данных в исходное состояние.

ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОЙ ПРОГРАММЕ

Разрабатываемые программы должны удовлетворять следующим требованиям:

- программное приложение должно содержать HTML-документ с формой для ввода данных и PHP-программу, вызываемую по окончании работы с HTML-формой;
- ввод параметров задания в HTML-форме может быть осуществлен либо путем ввода значений в текстовом виде, либо посредством выбора значений из предлагаемого списка;
- программа должна быть написана в предположении, что любой пользователь без ограничений может иметь доступ к данным;
- в программе должен быть предусмотрен вывод сообщений обо всех шагах ее выполнения, в том числе и о возможных ошибках;
- программа должна быть достаточно документирована.

ВАРИАНТЫ ЗАДАНИЙ

Вариант 1

1. Получить информацию о деталях, поставки которых были осуществлены для указанного изделия.
2. Вставить поставщика с заданными параметрами.

Вариант 2

1. Получить информацию о поставщиках, которые осуществляли поставки деталей из заданного города в указанный период.
2. Изменить цвет самой тяжелой детали на указанный.

Вариант 3

1. Получить информацию о поставщиках, поставивших детали для изделий из указанного города.
2. Увеличить рейтинг поставщика, выполнившего наибольшую поставку заданной детали, на указанную величину.

Вариант 4

1. Получить информацию обо всех деталях, поставляемых для указанного изделия более чем одним поставщиком.
2. В таблице поставок изменить номер поставщика при заданном номере детали и изделия.

Вариант 5

1. Получить информацию о поставщиках, поставивших указанную деталь в заданный период.
2. Удалить поставщика, выполнившего меньше всего поставок в указанный город.

Вариант 6

1. Получить информацию о деталях, поставки которых были осуществлены для указанного изделия всеми поставщиками.
2. Увеличить рейтинг поставщика, выполнившего наибольший суммарный объем поставок, на указанную величину.

Вариант 7

1. Получить информацию об изделиях, для которых была поставлена указанная деталь.
2. Изменить название и город детали с максимальным весом на указанные значения.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каким образом вставить конструкции PHP в HTML-документ?
2. Каким образом обеспечить, чтобы встречающиеся в строке переменные были заменены их значениями?
3. Каковы правила определения функций в языке PHP?
4. Каковы особенности передачи значений переменных из HTML-формы в переменные PHP?
5. Каким образом осуществляется взаимодействие с базами данных в языке PHP?
6. Каковы основные элементы HTML-формы?
7. Каковы основные свойства HTML-формы?
8. В каком виде данные, введенные в форме, передаются PHP-модулю?
9. Какова общая схема работы PHP-программы, обрабатывающей данные посредством HTML-формы?
10. В чем разница методов GET и POST?
11. В чем заключается технология «cookies»?

ЛАБОРАТОРНАЯ РАБОТА 8

ДОСТУП К БАЗАМ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ADO.NET

Цель работы

Ознакомиться с основными методами создания простейших web-приложений с использованием технологии ADO.NET.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

1. Ознакомиться с принципами разработки ADO.NET-приложений в интегрированной среде разработки Visual Studio.
2. Изучить основные свойства, методы и события классов, используемых для работы с базами данных.
3. Ознакомиться с набором компонент для проектирования интерфейса ADO.NET-приложения.
4. С использованием технологии ADO.NET разработать и отладить Web-приложение, работающее с базой данных.

ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Убедиться в наличии в базе данных всех необходимых таблиц и достаточности данных в них. В лабораторной работе используется схема базы данных, содержащая 10 таблиц:

- таблица поставщиков (S);
- таблица деталей (P);
- таблица изделий (J);
- таблица поставок (SPJ);
- таблица норм (Q),
- таблица издержек (E)
- таблица производства (W),
- таблица цен (V),
- таблица заказчиков (C),
- таблица заказов (R).

2. Если приведенная схема базы данных отсутствует (или любая таблица из нее), то необходимо восстановить базу данных, выполнив необходимый запрос на языке SQL в рамках программы *phpPgAdmin*. В приложении 2 приведены скрипты запросов для создания таблиц и заполнения их данными. При выполнении работы эти запросы могут быть дополнены или изменены.

3. Разработать и протестировать динамическую web-страницу для выполнения в соответствии с вариантом задания запросов на выборку и модификацию данных и отображения их результатов.

ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОЙ ПРОГРАММЕ

Разрабатываемое web-приложение должно удовлетворять следующим требованиям:

- содержать форму для ввода параметров запросов и отображения результатов выполнения запросов в соответствии с заданием, а также обработчик (на Visial C#) для доступа к базе данных и выполнения запросов;
- ввод параметров задания на форме может быть осуществлен либо путем ввода значений в текстовом виде, либо посредством выбора значений из предлагаемого списка (в случае, когда список может быть сформирован из БД);
- программа должна быть написана в предположении, что любой пользователь без ограничений может иметь доступ к данным;
- в программе должен быть предусмотрен вывод сообщений обо всех шагах ее выполнения, в том числе и о возможных ошибках;
- программа должна быть достаточно документирована.

ВАРИАНТЫ ЗАДАНИЙ

Вариант 1

1. Получить информацию о размере издержек на изготовление указанного изделия на заданную дату.

2. В таблицу издержек добавить новую запись с заданными значениями номера изделия, размера издержек и даты начала действия.

Вариант 2

1. Получить информацию о деталях, которых в настоящий момент не хватает для изготовления заданного количества указанного изделия.

2. Увеличить на заданное число количество в последней поставке каждой детали для указанного изделия.

Вариант 3

1. Получить информацию о рекомендованной цене на указанное изделие на заданную дату.
2. Для изделий, в состав которых входит заданная деталь, сдвинуть на месяц назад дату начала действия последней рекомендованной цены.

Вариант 4

1. Получить информацию о выручке поставщиков, сделавших больше всего поставок для указанного изделия.
2. Для указанного поставщика удвоить количество деталей в поставках за заданный период.

Вариант 5

1. Получить информацию о последней цене деталей, которые были поставлены для указанного изделия.
2. Вставить заказ с указанными параметрами.

Вариант 6

1. Получить информацию о поставщиках, поставивших детали для изделий из указанного города.
2. Увеличить рейтинг поставщика, выполнившего наибольшую поставку некоторой детали, на указанную величину.

Вариант 7

1. Получить информацию о поставщиках, которые осуществляли поставки деталей из заданного города в указанный период.
2. Вставить поставщика с заданными параметрами

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какова общая схема доступа к базе данных посредством технологии ADO.NET?
2. Какова общая схема работы web-сервера при использовании ADO.NET?
3. Назначение, основные свойства и методы объекта DbConnection.

4. Назначение, основные свойства и методы объекта DbCommand.

5. Каков механизм передачи данных, введенных в форме, модулю-обработчику?

6. Какими средствами реализуется механизм транзакций при работе с базами данных с использованием ADO.NET?

7. Как обеспечить обработку ошибок при работе с базами данных в ADO.NET?

ЛАБОРАТОРНАЯ РАБОТА 9

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ Activex Data Objects (ADO) ДЛЯ ДОСТУПА К БАЗАМ ДАННЫХ

ЦЕЛЬ РАБОТЫ

Ознакомиться с компонентами ADO среды разработки C++Builder с целью их использования при создании Windows-приложения для работы с базами данных.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

1. Ознакомиться с принципами объектно-ориентированного визуального программирования и технологией разработки Windows-приложений в среде C++Builder.

2. Изучить основные свойства, методы и события компонентов, необходимых для разработки простейшего приложения.

3. Ознакомиться с набором компонентов для работы с базами данных по технологии ADO.

4. Ознакомиться с понятием исключения и методами обработки исключений.

5. С использованием компонентов ADO разработать и отладить Windows-приложение, работающее с базой данных.

ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Убедиться в наличии в базе данных всех необходимых таблиц и достаточности данных в них. В лабораторной работе используется схема базы данных, содержащая 10 таблиц:

- таблица поставщиков (S);
- таблица деталей (P);
- таблица изделий (J);
- таблица поставок (SPJ);
- таблица норм (Q),
- таблица издержек(E)
- таблица производства(W),
- таблица цен(V),
- таблица заказчиков(C),
- таблица заказов(R).

Если приведенная схема базы данных отсутствует (или любая таблица из нее), то необходимо восстановить базу данных, выполнив необходимый запрос на языке SQL в рамках программы *phpPgAdmin*. Скрипты запросов для создания таблиц и заполнения их данными приведены в приложении 2.

2. Разработать и протестировать Windows-приложение для работы с базой данных, позволяющее просматривать выборки данных в соответствии с первым и вторым запросом задания, а также модифицировать данные в соответствии с третьим запросом задания.

3. После выполнения лабораторной работы привести базу данных в исходное состояние.

ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОМУ ПРИЛОЖЕНИЮ

Разрабатываемое приложение должно удовлетворять следующим требованиям.

- Приложение должно включать в себя три формы:
  - модуль данных, содержащий все необходимые компоненты для работы с базой данных;
  - форму для просмотра выборок данных;
  - форму для выполнения запроса модификации данных.
- Соединение с базой данных должно выполняться через компонент ADOConnection. Для выборки данных использовать ADOQuery.

- Модификация данных должна выполняться либо серверной функцией, вызываемой из приложения через ADOStoredProc, либо запросом через ADOQuery.
- Просмотр выборок должен осуществляться через компоненты DBGrid. Просмотр должен быть согласованным (выборка по второму запросу должна выполняться для текущей строки выборки по первому запросу). Строки выборок должны быть отсортированы по указанным в задании столбцам. Названия столбцов должны быть на русском языке. Строки выборок, удовлетворяющие указанным в задании условиям, должны быть выделены (цветом фона и/или цветом/стилем шрифта).
- Визуализация формы для выполнения запроса на модификацию может вызываться либо нажатием кнопки, либо через контекстное меню.
- Ввод параметров для модификации данных может быть осуществлен либо путем ввода значений в текстовом виде, либо посредством выбора значений из предлагаемого списка.
- После модификации данных должно появляться сообщение о том, насколько успешно прошла модификация.
- В случае возникновения ошибки должно выдаваться соответствующие сообщение.

ВАРИАНТЫ ЗАДАНИЙ

Вариант 1

1. Для каждого изделия за каждый год получить:
   - число поставщиков;
   - общее количество поставленных деталей;
   - на какую сумму выполнены поставки.

Упорядочить по изделию и году. Выделить строки, где только один поставщик.

2. Для указанных изделия и года по каждой детали вывести:
   - количество поставленных деталей;
   - на какую сумму выполнены поставки;
   - среднюю цену детали.
3. Добавить новую поставку.

Вариант 2

1. Для каждого поставщика за каждый год получить:
   - сумму, на которую он выполнил поставки;
   - общее число поставок;
   - число изделий.

Упорядочить по поставщику и году. Выделить строки, где сумма меньше 100.

2. Для указанных поставщика и года по каждой детали вывести:
   - количество поставленных деталей;
   - на какую сумму выполнены поставки;
   - процент выручки по этой детали от общей суммы за год.

3. Изменить количество деталей в поставке.

Вариант 3

1. Для каждого изделия на конец каждого года получить:
   - размер максимальной поставки;
   - сумму, на которую выполнены поставки для изделия;
   - процент этой суммы от общей суммы по всем изделиям за год.

Упорядочить по году и проценту. Выделить строки, где процент не меньше 50.

2. Для указанных изделия и года по каждой поставке вывести:
   - сумму поставки;
   - разницу между ценой детали в поставке и средней ценой детали за год.

3. Изменить цену детали в поставке.

Вариант 4

1. Для каждой детали получить:
   - сколько этой детали есть в наличии;
   - сколько этой детали нужно для выполнения текущих заказов (не отправленных).

Упорядочить по номеру детали. Выделить строки, где деталей не хватает для выполнения заказов.

2. Для указанной детали получить перечень текущих заказов, где требуется эта деталь, и для каждого заказа вывести:
   - сведения о заказе (дата, номер, заказчик, изделие, количество);
   - сколько деталей требуется;

- сколько есть в наличии деталей для заказанного изделия.

3. Добавить новый заказ.

Вариант 5

1. Для каждого изделия и за каждый гол получить:
   - количество собранных изделий;
   - количество проданных изделий;
   - на какую сумму.

Упорядочить по году и количеству проданных изделий. Выделить строки, где количество собранных изделий превышает количество проданных как минимум на 5.

2. Для указанных изделия и года вывести в хронологическом порядке:
   - сведения о производстве (дата, количество (с плюсом), издержки на производство);
   - сведения о продажах (дата, количество (с минусом), стоимость заказа);

3. Изменить количество изделий в партии.

Вариант 6

1. Для каждого заказчика за каждый год получить:
   - сумму, на которую он закупил изделия;
   - максимальный процент скидки;
   - минимальный процент скидки.

Упорядочить по году и заказчику. Выделить строки, где максимальный и минимальный проценты скидки совпадают.

2. Для указанных заказчика и года вывести все заказы в хронологическом порядке и для каждого получить:
   - сумму сделки;
   - рекомендованную цену изделия;
   - процент скидки.

3. Изменить цену изделия в заказе.

Вариант 7

1. Для каждого изделия за каждый год получить:
   - среднюю себестоимость изделия;
   - количество собранных изделий;

- среднюю цену продажи;
- количество проданных изделий.

Упорядочить по году и изделию. Выделить строки, где средняя себестоимость меньше средней цены более чем в полтора раза.

Замечание. Себестоимость партии изделий рассчитывается по формуле: сумма по деталям (норма * цена детали на дату выпуска) плюс издержки на дату выпуска.

2. Для указанных изделия и года вывести все заказы в хронологическом порядке и для каждого получить:
   - сведения о заказе;
   - среднюю себестоимость изделия;
   - рекомендованную цену изделия.
3. Изменить размер издержек.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какова общая схема работы с базами данных в C++Builder?
2. Назначение, основные свойства, методы и события компонента ADOConnection.
3. Назначение, основные свойства, методы и события компонента ADO ADOQuery.
4. Назначение, основные свойства, методы и события компонента ADOStoredProc.
5. Назначение, основные свойства, методы и события компонента DBGrid.
6. Какими средствами реализуется механизм транзакций при работе с базами данных?
7. Как обеспечить обработку ошибок при работе с базами данных в C++Builder?

ЛАБОРАТОРНАЯ РАБОТА 10

РЕАЛИЗАЦИЯ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ ДАННЫХ ПРИ ПОМОЩИ ТРИГГЕРОВ

ЦЕЛЬ РАБОТЫ

Научиться писать триггеры, обеспечивающие выполнение ограничений целостности данных в таблицах БД PostgreSQL.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

1. Ознакомиться с технологией разработки триггеров в PostgreSQL.
2. Изучить синтаксис команды определения триггера и команды определения триггерной функции.
3. Ознакомиться с основными правилами разработки триггеров, обеспечивающих целостность данных.
4. Ознакомиться с инструментами разработки триггеров в PhpPg Admin.
5. Разработать и протестировать триггер.

ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Убедиться в наличии в базе данных всех необходимых таблиц и достаточности данных в них. В лабораторной работе используется схема базы данных, содержащая 10 таблиц:

- таблица поставщиков (S);
- таблица деталей (P);
- таблица изделий (J);
- таблица поставок (SPJ);
- таблица норм (Q),
- таблица издержек (E)
- таблица производства (W),
- таблица цен (V),
- таблица заказчиков (C),
- таблица заказов (R).

2. Если приведенная схема базы данных отсутствует (или любая таблица из нее), то необходимо восстановить базу данных, выполнив необходимый запрос на языке SQL в рамках программы *PhpPgAdmin*. Скрипты запросов для создания таблиц и заполнения их данными – в приложении 2.

3. Получить задание у преподавателя. Для ограничения, указанного в задании, определить, какие таблицы затрагивает ограничение, сформулировать ограничение в терминах базы данных и определить, при каких операциях ограничение может быть нарушено.

4. Разработать требуемое количество триггерных функций и триггеров.

5. Протестировать разработанные триггеры.

6. После выполнения лабораторной работы привести базу данных в исходное состояние.

ТРЕБОВАНИЯ К РАБОТЕ

Разрабатываемый триггер должен удовлетворять следующим требованиям:

- обрабатывать все возможные варианты внесения некорректных данных;
- выдавать адекватные, максимально развернутые сообщения.

При тестировании нужно проверить все операции, при которых может быть нарушено ограничение как на корректных данных, так и на некорректных.

ЗАДАНИЯ

1. Цена продажи не может быть меньше, чем себестоимость (стоимость деталей + издержки).

2. Цена продажи не может быть меньше, чем (рекомендованная цена – скидка клиента).

3. Рекомендованная цена не может быть меньше, чем себестоимость (стоимость деталей + издержки) *1,5.

4. Для каждого изделия должны поставляться только детали, для которых установлена норма расхода.

5. На момент производства партии изделий в наличии должно быть нужное количество деталей.

6. Цена продажи не может быть больше, чем рекомендованная цена *2.

7. На момент отправки заказчику изделий в наличии должно быть их достаточное количество.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое ограничения целостности? Какими средствами они обеспечиваются?
2. Что такое триггер? Синтаксис определения триггеров в PostgreSQL.
3. Что такое триггерная функция? Синтаксис триггерной функции.
4. Какие специальные переменные доступны триггерным функциям?
5. Как сформировать сообщение об ошибке в триггерной функции?

ЧАСТЬ III

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ПРОВЕРОЧНОЙ РАБОТЫ ПО ТЕМЕ «ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ И БАЗ ДАННЫХ»

В ходе выполнения проверочной работы по теме «Проектирование информационных систем и баз данных» студенты должны продемонстрировать полученные в курсе лекций знания в области:

- проектирования информационных систем;
- инфологического и логического проектирования баз данных.

Для выбранного варианта с некоторой предметной областью из учебного пособия [3] студенты должны выполнить и представить преподавателю описание комплекса работ по проектированию информационной системы и базы данных. Выбранный вариант предметной области студент сообщает преподавателю, ведущему лабораторные работы.

Проверочная работа выполняется студентом индивидуально и является **одной из составных частей, определяющих итоговую оценку знаний студента на экзамене.** Выполняется проверочная работа в ходе семестра параллельно с выполнением лабораторных работ и представляется на проверку в электронном виде не позднее чем за пять дней до экзамена.

ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ПРОВЕРОЧНОЙ РАБОТЫ

Структурно отчет проверочной работы по теме «Проектирование информационных систем и баз данных» состоит из трех частей:

- проектирование информационной системы;
- инфологическое проектирование базы данных;
- логическое проектирование базы данных.

I. Проектирование информационной системы

Для успешной реализации информационной системы объект проектирования должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные модели бизнес-процессов. Опыт проектирования информационных систем показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов.

При проектировании информационной системы необходимо провести анализ целей этой системы и выявить требования к ней отдельных пользователей. В реальной жизни информация для построения модели информационной системы берется на основе проведения всестороннего обследования организации с учетом существующих или внедряющихся бизнес-процессов.

При выполнении проверочной работы вариант описания предметной области проектируемой системы выбирается из учебного пособия [3]. Необходимо проанализировать предметную область, уточнив и дополнив ее, руководствуясь собственным опытом, консультациями и другими источниками.

Для целей проектирования информационной системы могут быть использованы следующие виды моделей [3]:

- методология функционального моделирования работ SADT (Structured Analysis and Design Technique);
- диаграммы потоков данных DFD (Data Flow Diagrams);
- методология объектного проектирования на языке UML (UML-диаграммы).

В ходе проектирования информационной системы необходимо выполнить следующее.

1. Структурно разбить предметную области на отдельные подразделения (отделы, службы, подсистемы и пр.) согласно выполняемым ими функциям. Определить состав подразделений (подсистем) информационной системы.

2. Определить задачи и функции системы в целом и функции каждого подразделения (подсистемы).

3. Построить диаграммы работ, диаграммы потоков данных, UML-диаграммы для всей информационной системы в целом и для отдельных бизнес-процессов, отражающие логику и взаимоотношения подразделений (подсистем).

4. Оформить раздел отчета проверочной работы «Проектирование информационной системы», включив в него:

- уточненное описание предметной области проектируемой системы;
- структуру автоматизируемой организации и структуру системы;
- задачи и функции системы в целом и функции каждого подразделения (подсистемы);
- описание информационной системы в целом и ее отдельных бизнес-процессов посредством диаграмм работ, диаграммы потоков данных, UML-диаграмм.

II. Инфологическое проектирование базы данных

Цель инфологического этапа проектирования состоит в получении семантических (концептуальных) моделей, отражающих предметную область и информационные потребности пользователей. В качестве инструмента для построения семантических моделей данных на этапе инфологического проектирования является неформальная модель «Сущность-связь» (ER-модель – Entity-Relationship) [3].

В ходе инфологического проектирования необходимо следующее.

1. Выделить необходимый набор сущностей, отражающих предметную область и обеспечивающий удобное выполнение всех задач, операций и функций системы (см. этап «проектирование системы») и представленных в задании запросов на выборку данных. Определить сущности вида подтип/супертип, где это необходимо. Выполнить классификацию сущностей, разделив их на стержневые, ассоциативные, характеристические, обозначающие.

2. Определить необходимый набор атрибутов каждой сущности, выделив идентифицирующие атрибуты каждой сущности. Выполнить классификацию атрибутов каждой сущности (описательные, указывающие, вспомогательные).

4. Определить связи между сущностями. Проанализировав структуру связей, исключить избыточные. Определить множественность и условность связей. Дать формулировку связей с точки зрения каждой участвующей сущности.

5. Формализовать связи. Формализация связи выполняется размещением вспомогательных атрибутов в соответствующих сущностях модели.

6. Построить ER-диаграмму модели базы данных.

7. Оформить раздел отчета «Инфологическое проектирование модели базы данных», включив в него:

- описание сущностей с полным перечнем атрибутов; классификацию сущностей и атрибутов;
- ER-диаграмму с описанием связей между сущностями; при этом все связи должны быть описаны и включать:
  - идентификатор связи;
  - формулировку имен связи с точки зрения каждой участвующей сущности;
  - вид связи (множественность, условность);
  - описание, какими атрибутами связь была формализована.

III. Логическое проектирование базы данных

Цель логического этапа проектирования – организация данных, выделенных на этапе инфологического проектирования в форму, принятую в выбранной СУБД. Задачей логического этапа проектирования является отображение объектов предметной области в объекты используемой модели данных, чтобы это отображение не противоречило семантике предметной области и было по возможности наилучшим (эффективным, удобным и т. д.). С точки зрения выбранной СУБД задача логического проектирования реляционной базы данных состоит в обоснованном принятии решений о том:

- из каких отношений должна состоять база данных;
- какие атрибуты должны быть у этих отношений;
- какие ограничения должны быть наложены на атрибуты и отношения базы данных, чтобы обеспечить ее целостность.

Требования к выбранному набору отношений и составу их атрибутов должны удовлетворять следующим условиям:

- отношения должны отличаться минимальной избыточностью атрибутов;
- выбранные для отношения первичные ключи должны быть минимальными;
- между атрибутами не должно быть нежелательных функциональных зависимостей;
- выбор отношений и атрибутов должен обеспечивать минимальное дублирование данных;
- не должно быть трудностей при выполнении операций включения, удаления и модификации данных;

- перестройка набора отношений при введении новых типов должна быть минимальной.

Удовлетворение отмеченных требований обеспечивается аппаратом нормализации отношений. Нормализация отношений – это пошаговый обратимый процесс композиции или декомпозиции исходных отношений в отношения, обладающие лучшими свойствами при включении, изменении и удалении данных, назначение им ключей по определенным правилам нормализации и выявление всех возможных функциональных зависимостей [3].

В ходе логического проектирования необходимо:

1) выполнить процедуру построения реляционной модели данных из ER-модели, построив необходимый набор отношений. Определить состав атрибутов каждого отношения;

2) определить первичные и внешние ключи отношений;

3) Выполнить шаги по нормализации полученных отношений, приведя модель к третьей нормальной форме;

4) задать необходимые декларативные ограничения целостности исходя из специфики предметной области;

5) средствами имеющейся СУБД записать скрипты на создание базы данных, ее таблиц, задав необходимые ограничения целостности;

6) оформить раздел отчета «Логическое проектирование модели базы данных», включив в него полные скрипты на создание базы данных, ее таблиц, необходимых ключей и полного набора ограничений целостности. Выполнение скрипта должно приводить к корректному созданию базы данных в рамках СУБД, используемой при выполнении лабораторных работ. Скрипты должны быть достаточно документированы.

ЧАСТЬ IV

ТЕХНОЛОГИИ ТИРАЖИРОВАНИЯ ДАННЫХ. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ТЕХНОЛОГИИ БАЗ ДАННЫХ»

ЦЕЛЬ РАБОТЫ

Изучить технологии тиражирования данных.

СОДЕРЖАНИЕ РАБОТЫ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЕЕ ВЫПОЛНЕНИЮ

1. Ознакомиться с технологиями тиражирования данных.
2. Изучить схемы репликации данных.
3. Для выбранного варианта спроектировать и программно реализовать модель репликации данных.
4. Оформить отчет и продемонстрировать работу разработанной модели репликации данных.

Основной недостаток физического распределения данных – жесткие требования к скорости и надежности каналов связи. Если база данных распределена по нескольким территориально удаленным узлам, объединенным относительно медленными и ненадежными каналами связи, а число одновременно работающих пользователей составляет сотни и выше, то вероятность того, что распределенная транзакция будет зафиксирована в обозримом временном интервале, становится чрезвычайно малой.

В действительности далеко не во всех задачах требуется обеспечение идентичности базы данных в различных узлах в любое время. Достаточно поддерживать тождественность данных лишь в определенные критичные моменты времени. Следовательно, можно накапливать изменения в данных в одном узле и периодически копировать эти изменения на другие узлы.

Тиражирование (репликация) данных (Data Replication - DR) – это в общем случае асинхронный перенос изменений объектов исходной базы данных (source database) в базы данных, принадлежащие другим узлам распределенной системы (target database).

Допущения при выполнении курсовой работы

1. В реальной жизни технология тиражирования данных используется для переноса данных, размещенных в базах данных, расположенных в различных узлах распределенной системы. При выполнении курсовой работы предполагается перенос данных между различными таблицами одной схемы базы данных. Поскольку целью курсовой работы является изучение схем репликации данных, а не технические аспекты работы распределенной СУБД и вопросы эффективности выполнения запросов, данное допущение не принципиально.

2. В схеме базы данных, с которой ведется работа, создаются несколько таблиц одинаковой структуры и единого содержимого, которые имитируют таблицы, находящиеся в различных базах данных («Таблица_в_БД1», «Таблица_в_БД2» и т. д.).

3. Для схем репликации 1 и 4 согласованный распределенный набор данных, идентичность которого необходимо обеспечить в процессе работы в source database и target database, – строки таблицы, для схем 2 и 3 – вся таблица.

4. Упомянутые выше таблицы могут быть любой структуры, но при этом должны содержать два поля:

- поле даты / времени для хранения времени вставки / обновления / удаления строки;
- символьное поле, идентифицирующее выполненную операцию (вставка/ обновление/удаление) и источник изменений.

Проектируемое программное обеспечение состоит из трех программ.

1. Программа инициализации данных (ИД).

2. Программа имитации работы системы (ИРС).

3. Программы репликатора данных (РД).

1. Программа инициализации данных генерирует и записывает данные (10…15 строк) во все таблицы («Таблица_в_БД1», «Таблица_в_БД2» и т. д.), участвующие в реализуемой схеме модели репликации. Модель данных определяется выбранным вариантом задания (см. табл. 1).

При этом:

- в поле даты/времени заносится текущее время инициализации данных;
- в поле идентификации операции заносится значение «начальная вставка».

Содержимое таблицы, идентичное во всех условных базах данных, запоминается в журнале содержимого таблиц (файл данных).

В качестве программы ИД может быть SQL-скрипт, запущенный через phpPgAdmin.

2. Программа имитации работы системы с определенной дискретностью (интервал в несколько секунд) моделирует процесс работы информационной системы, выполняя следующие действия.

- Случайным образом выбирает одну из условных баз данных («БД1», «БД2» и т. д.).
- Случайным образом выбирает одну из операций (вставка / обновление / удаление).
- Если выполняется операция вставки, то в выбранную базу данных вставляется строка, в которой:
  - в поле даты/времени заносится текущее время вставки;
  - в поле идентификации операции заносится значение «вставка в БДi», где i = 1,2… .
- В журнале изменений (файл данных) запоминается:
  - время вставки;
  - база данных, в которую выполняется вставка;
  - вставленная строка.
- Если выполняется операция обновления, то в выбранной базе данных обновляется строка с минимальным oid, в которой:
  - в поле даты/времени заносится текущее время обновления;
  - в поле идентификации операции заносится значение «обновление в БДi», где i = 1,2… .
- В журнале изменений запоминается:
  - время обновления;
  - база данных, в которую выполняется обновление;
  - старое и новое состояние обновляемой строки.
- Если выполняется операция удаления, то в выбранной базе данных удаляется строка с максимальным oid.
- Перед удалением в журнале изменений запоминается:
  - время удаления;

– база данных, из которую выполняется удаление;
– удаляемая строка.

Все действия по вставке/обновлению/удалению строк выполняются в форме транзакций.

3. Программа репликатора данных работает в соответствии с моделью репликации, определенной условиями (схема репликации, условие запуска репликатора, способ разрешения коллизий), заданными в табл. 1. После выполнения цикла репликации (переноса данных и обеспечения согласованного состояния таблиц) программа РД фиксирует в журнале содержимого таблиц:

- текущее время;
- содержимое всех таблиц условных баз данных («БД1», «БД2» и т. д.).

Порядок выполнения действия:

1) запустить программу ИД;
2) запустить программу ИРС;
3) параллельно с работой программы ИРС запустить программу РД;
4) остановить работу программ ИРС и РД;
5) проанализировать по журналу изменений и журналу содержимого правильность работы схемы репликации.

Замечание 1. Для обеспечения согласованной работы параллельно выполняющихся программ ИРС и РД перед выполнением программой репликатора цикла репликации необходимо заблокировать доступ к таблицам для программы ИРС, например,

lock table Таблица_в_БД1, *Таблица_в_БД1* [,...] *in exclusive*

< действия по репликации данных >

commit

Замечание 2. Формат записи в журнал изменений и журнал содержимого таблиц определяется студентом, однако содержимое этих журналов должно позволять легко проанализировать правильность работы схемы репликации.

Замечание 3. Алгоритм репликации может строиться как на сравнении записей таблиц (репликация по состоянию), так и на основании состояния журнала изменений (репликация изменений).

Таблица 1

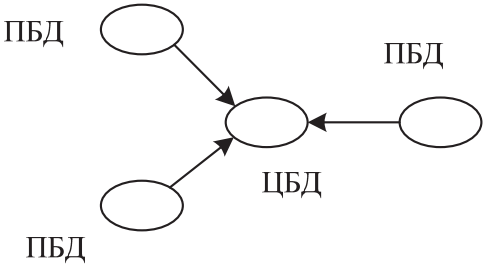
Варианты заданий

Номер варианта	Схема репликации	Запуск репликатора	Разрешение коллизий
1	1	1	1
2	2	1	1
3	3	1	1
4	4	1	1
5	1	2	1
6	2	2	1
7	3	2	1
8	4	2	1
9	1	1	1
10	2	1	2
11	3	1	2
12	4	1	2
13	1	2	2
14	2	2	2
15	3	2	2
16	4	2	2
17	1	1	2
18	2	1	2
19	3	1	3
20	4	1	3
21	1	2	3
22	2	2	3
23	3	2	3
24	4	2	3

ОБОЗНАЧЕНИЕ В ТАБЛИЦЕ

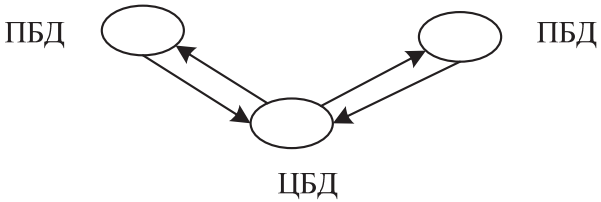
Схема репликации

1. Однонаправленное тиражирование «Центр-филиалы». Изменение (вставка, модификация, удаление), выполненное в одной из периферийных баз данных (ПБД), тиражируется в центральную базу данных (ЦБД).



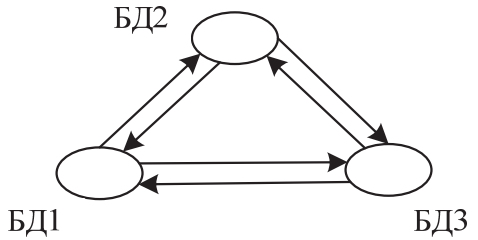
Приведенная схема репликации может соответствовать следующей бизнес-модели. Периферийные базы данных находятся в различных регионах, и в каждой из них накапливаются данные о работе с клиентами своего региона. Поскольку клиенты могут перемещаться между регионами, данные об одном и том же клиенте могут появиться в двух ПБД. Появление новых данных реплицируется в центральную базу данных, где данные накапливаются и осуществляется OLAP-анализ данных. ЦБД не выполняет изменений данных.

2. Многонаправленное тиражирование «Центр-филиалы». Изменение (вставка, модификация, удаление), выполненное в одной из ПБД, тиражируется в ЦБД, после чего изменения реплицируются во все другие ПБД.

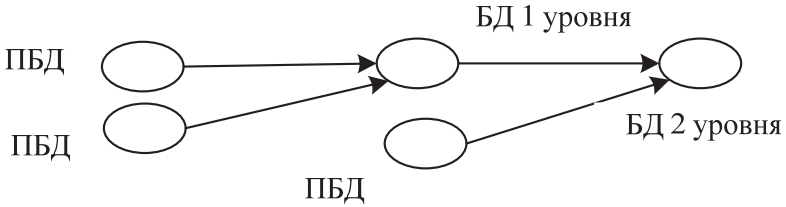


Приведенная схема репликации может соответствовать следующей бизнес-модели. Периферийные базы данных находятся в разных филиалах. С одним и тем же клиентом сотрудники могут работать и заносить и менять данные и в центральном и в периферийном офисе, но один клиент не может обратиться в два различных филиала, работающих с различными ПБД.

3. Многонаправленное тиражирование. Все базы данных функционально равноправны. Изменение (вставка, модификация, удаление), выполненное в одной из баз данных, тиражируется во все другие базы данных распределенной системы.



4. Каскадное однонаправленное тиражирование. Изменение (вставка, модификация, удаление), выполненное в одной из ПБД, тиражируется в базу данных следующего уровня, как показано на рисунке.



Приведенная схема репликации может соответствовать следующей бизнес-модели. Периферийные базы данных нижнего уровня – ПБД филиала местного уровня, где собираются сведения о клиентах. При этом один и тот же клиент может обращаться в различные местные филиалы. Следующий уровень – офис с БД уровня региона, верхний уровень – БД центрального офиса. Все изменения выполняются только в ПБД и затем транслируются на следующий уровень.

Запуск репликатора

1. Через определенный интервал времени в секундах, задаваемый при запуске программы РД.
2. После выполнения указанного числа транзакций, задаваемого при запуске программы РД.

Разрешение коллизий

1. В пользу более раннего обновления.
2. В зависимости от приоритета базы данных.
3. В пользу более позднего обновления.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

ОБЛАСТЬ СВЯЗИ SQL

```
struct sqlca_s
{
        long sqlcode;                   /* код завершения:  */
        char sqlerrm[72];           /* параметры сообщений об ошибке */
        char sqlerrp[8];                /* для внутреннего пользования */
        long sqlerrd[6];
                /* 0 – ожидаемое количество возвращаемых строк */
                /* 1 – значение поля serial после insert или ISAM-код ошибки */
                /* 2 – количество обработанных строк */
                /* 3 – оценочное число обращений к диску*/
                /* 4 – смещение ошибки в SQL-описании*/
                /* 5 – rowid строки после вставки, удаления, корректировки*/
        struct sqlcaw_s
                char sqlwarn0;   /* = W в случае любого sqlwarn[1–7] */
                char sqlwarn1;   /* = W в случае любых усечений*/
                char sqlwarn2;   /* = W в случае возвращения пустого значения */
                char sqlwarn3;   /* = W, если список Select не совпадает со
                                    списком полей*/
                char sqlwarn4;   /* = W, если нет where в операторах
                                            Insert или Delete */
                char sqlwarn5;   /* = W, если не ANSI-описание*/
                char sqlwarn6;   /* = W зарезервировано */
                char sqlwarn7;   /* = W зарезервировано */
        } sqlwarn;
};
extern struct sqlca_s sqlca;
extern long SQLCODE;
#define SQLNOTFOUND 100;
```

СОЗДАНИЕ ТАБЛИЦ

---------- ТАБЛИЦА J – Изделия (Job)

```
CREATE TABLE j (
n_izd character(6) NOT NULL,
name character(20),
town character(20));

COMMENT ON TABLE j IS 'Изделия';
COMMENT ON COLUMN j.n_izd IS 'номер изделия';
COMMENT ON COLUMN j.name IS 'название изделия';
COMMENT ON COLUMN j.town IS 'город изделия';

ALTER TABLE ONLY j ADD CONSTRAINT j_pkey PRIMARY KEY (n_izd);
```

---------- ТАБЛИЦА P – Детали (Piece)

```
CREATE TABLE p (
n_det character(6) NOT NULL,
name character(20),
cvet character(20),
ves integer,
town character(20));

COMMENT ON TABLE p IS 'Детали';
COMMENT ON COLUMN p.n_det IS 'номер детали';
COMMENT ON COLUMN p.name IS 'название детали';
COMMENT ON COLUMN p.cvet IS 'цвет детали';
COMMENT ON COLUMN p.ves IS 'вес детали';
COMMENT ON COLUMN p.town IS 'город детали';

ALTER TABLE ONLY p ADD CONSTRAINT p_pkey PRIMARY KEY (n_det);
```

---------- ТАБЛИЦА S – Поставщики (Suppliers)

```
CREATE TABLE s (
n_post character(6) NOT NULL,
name character(20),
reiting integer,
town character(20));

COMMENT ON TABLE s IS 'Поставщики';
COMMENT ON COLUMN s.n_post IS 'номер поставщика';
COMMENT ON COLUMN s.name IS 'имя поставщика';
COMMENT ON COLUMN s.reiting IS 'ранг поставщика';
COMMENT ON COLUMN s.town IS 'город поставщика';

ALTER TABLE ONLY s ADD CONSTRAINT s_pkey PRIMARY KEY (n_post);
```

---------- ТАБЛИЦА SPJ1 – Поставки (Supply)

```
CREATE TABLE spj1 (
n_spj character(6) NOT NULL,
n_post character(6) NOT NULL,
n_det character(6) NOT NULL,
n_izd character(6) NOT NULL,
kol integer,
date_post date NOT NULL,
cost integer NOT NULL
CONSTRAINT poscost CHECK ((cost > 0)),
CONSTRAINT poskol CHECK ((kol > 0)));

COMMENT ON TABLE spj1 IS 'Поставки';
COMMENT ON COLUMN spj1.n_spj IS 'номер поставки';
COMMENT ON COLUMN spj1.n_post IS 'номер поставщика';
COMMENT ON COLUMN spj1.n_det IS 'номер детали';
COMMENT ON COLUMN spj1.n_izd IS 'номер изделия';
COMMENT ON COLUMN spj1.kol IS 'количество деталей';
COMMENT ON COLUMN spj1.date_post IS 'дата поставки';
COMMENT ON COLUMN spj1.cost IS 'цена за одну деталь';
```

```
ALTER TABLE ONLY spj1 ADD CONSTRAINT spj1_n_post_key UNIQUE (n_post,n_det,n_izd,date_post);
ALTER TABLE ONLY spj1 ADD CONSTRAINT spj1_pkey PRIMARY KEY (n_spj);
```

---------- ТАБЛИЦА C – Заказчики (Client)

```
CREATE TABLE c (
n_cl character(6) NOT NULL,
name character(20),
town character(10),
discount integer);

COMMENT ON TABLE c IS 'Заказчики';
COMMENT ON COLUMN c.n_cl IS 'номер заказчика';
COMMENT ON COLUMN c.name IS 'имя заказчика ';
COMMENT ON COLUMN c.town IS 'город заказчика';
COMMENT ON COLUMN c.discount IS 'скидка';

ALTER TABLE ONLY c ADD CONSTRAINT c_pkey PRIMARY KEY (n_cl);
```

---------- ТАБЛИЦА Q – норма расхода деталей на одно изделие

```
CREATE TABLE q (
n_q character(6) NOT NULL,
n_izd character(6) NOT NULL,
n_det character(6) NOT NULL,
kol integer NOT NULL);

COMMENT ON TABLE q IS 'Норма расхода деталей на одно изделие';
COMMENT ON COLUMN q.n_q IS 'номер записи';
COMMENT ON COLUMN q.n_izd IS 'номер изделия';
COMMENT ON COLUMN q.n_det IS 'номер детали';
COMMENT ON COLUMN q.kol IS 'количество деталей для одного изделия';

ALTER TABLE ONLY q ADD CONSTRAINT q_n_izd_key UNIQUE (n_izd,n_det);
ALTER TABLE ONLY q ADD CONSTRAINT q_pkey PRIMARY KEY (n_q);
```

---------- ТАБЛИЦА W – Выпуск изделий (Working)

```
CREATE TABLE w (
n_part character(6) NOT NULL,
n_izd character(6) NOT NULL,
date_part date,
kol integer NOT NULL);

COMMENT ON TABLE w IS 'Выпуск изделий';
COMMENT ON COLUMN w.n_part IS 'номер партии';
COMMENT ON COLUMN w.n_izd IS 'номер изделия';
COMMENT ON COLUMN w.date_part IS 'дата изготовления партии';
COMMENT ON COLUMN w.kol IS 'количество изделий';

ALTER TABLE ONLY w ADD CONSTRAINT w_pkey PRIMARY KEY (n_part);
```

---------- ТАБЛИЦА E – Издержки на производство

```
CREATE TABLE e (
n_exp character(6) NOT NULL,
n_izd character(6) NOT NULL,
date_begin date NOT NULL,
cost integer NOT NULL);

COMMENT ON TABLE e IS 'Издержки на производство';
COMMENT ON COLUMN e.n_exp IS 'номер записи';
COMMENT ON COLUMN e.n_izd IS 'номер изделия';
COMMENT ON COLUMN e.date_begin IS 'действует с даты';
COMMENT ON COLUMN e.cost IS 'размер издержек на одно изделие';

ALTER TABLE ONLY e ADD CONSTRAINT e_n_izd_key UNIQUE (n_izd,date_begin);
ALTER TABLE ONLY e ADD CONSTRAINT e_pkey PRIMARY KEY (n_exp);
```

---------- ТАБЛИЦА V – Рекомендованная цена

```
CREATE TABLE v (
n_v character(6) NOT NULL,
```

```
n_izd character(6) NOT NULL,
date_begin date NOT NULL,
cost integer NOT NULL);
COMMENT ON TABLE v IS 'Рекомендованная цена';
COMMENT ON COLUMN v.n_v IS 'номер цены';
COMMENT ON COLUMN v.n_izd IS 'номер изделия';
COMMENT ON COLUMN v.date_begin IS 'действует с даты';
COMMENT ON COLUMN v.cost IS 'размер';

ALTER TABLE ONLY v ADD CONSTRAINT n_n_izd_key UNIQUE (n_izd,date_begin);
ALTER TABLE ONLY v ADD CONSTRAINT n_pkey PRIMARY KEY (n_v);
```

---------- ТАБЛИЦА R – Заказы

```
CREATE TABLE r (
n_real character(10) NOT NULL,
n_izd character(6) NOT NULL,
n_cl character(6) NOT NULL,
date_order date NOT NULL,
date_pay date,
date_ship date,
kol integer NOT NULL,
cost integer NOT NULL,
CONSTRAINT «posRcost» CHECK ((cost > 0)),
CONSTRAINT «posRkol» CHECK ((kol > 0)));

COMMENT ON TABLE r IS 'Заказы';
COMMENT ON COLUMN r.n_real IS 'номер заказа';
COMMENT ON COLUMN r.n_izd IS 'номер изделия';
COMMENT ON COLUMN r.n_cl IS 'номер покупателя';
COMMENT ON COLUMN r.date_order IS 'дата заказа';
COMMENT ON COLUMN r.date_pay IS 'дата оплаты';
COMMENT ON COLUMN r.date_ship IS 'дата отправки заказа';
COMMENT ON COLUMN r.kol IS 'количество изделий';
COMMENT ON COLUMN r.cost IS 'отпускная цена изделия';

ALTER TABLE ONLY r ADD CONSTRAINT r_pkey PRIMARY KEY (n_real);
```

ВСТАВКА ДАННЫХ

---------- ТАБЛИЦА J – Изделия (Job)

```
INSERT INTO j VALUES
('J1','Жесткий диск','Париж'),
('J2','Перфоратор','Рим'),
('J3','Считыватель','Афины'),
('J4','Принтер','Афины'),
('J5','Флоппи-диск','Лондон'),
('J6','Терминал','Осло'),
('J7','Лента','Лондон');
```

---------- ТАБЛИЦА P – Детали (Piece)

```
INSERT INTO p VALUES
('P1','Гайка','Красный',12,'Лондон'),
('P2','Болт','Зеленый',17,'Париж '),
('P3','Винт','Голубой',17,'Рим'),
('P4','Винт','Красный',14,'Лондон'),
('P5','Кулачок','Голубой',12,'Париж '),
('P6','Блюм','Красный',19,'Лондон');
```

---------- ТАБЛИЦА S – Поставщики (Suppliers)

```
INSERT INTO s VALUES
('S1','Смит ',20,'Лондон'),
('S2','Джонс',10,'Париж '),
('S3','Блейк',30,'Париж '),
('S4','Кларк',20,'Лондон'),
('S5','Адамс',30,'Афины ');
```

---------- ТАБЛИЦА SPJ1 – Поставки (Supply)

```
insert into SPJ1 (N_SPJ,N_POST,N_DET,N_IZD,KOL,DATE_POST,COST)
values ('N1','S2','P3','J4',500,to_date('06-02-2011','dd-mm-yyyy'),15),
('N2','S1','P1','J1',200,to_date('09-02-2011','dd-mm-yyyy'),7),
('N3','S2','P3','J5',600,to_date('10-02-2011','dd-mm-yyyy'),15),
('N4','S2','P3','J6',400,to_date('18-02-2011','dd-mm-yyyy'),16),
('N5','S2','P3','J1',400,to_date('21-02-2011','dd-mm-yyyy'),15),
('N6','S2','P3','J2',200,to_date('28-02-2011','dd-mm-yyyy'),15),
```

('N7','S2','P3','J7',800,to_date('28-02-2011','dd-mm-yyyy'),16),
('N8','S2','P3','J3',200,to_date('28-02-2011','dd-mm-yyyy'),15),
('N9','S2','P5','J2',100,to_date('02-03-2011','dd-mm-yyyy'),9),
('N10','S3','P4','J2',500,to_date('03-03-2011','dd-mm-yyyy'),4),
('N11','S4','P6','J3',300,to_date('10-03-2011','dd-mm-yyyy'),13),
('N12','S1','P1','J4',700,to_date('16-03-2011','dd-mm-yyyy'),7),
('N13','S4','P6','J7',300,to_date('18-03-2011','dd-mm-yyyy'),13),
('N14','S5','P2','J2',200,to_date('26-03-2011','dd-mm-yyyy'),6),
('N15','S5','P5','J5',500,to_date('04-04-2011','dd-mm-yyyy'),9),
('N16','S5','P5','J7',100,to_date('12-04-2011','dd-mm-yyyy'),10),
('N17','S5','P6','J2',200,to_date('15-04-2011','dd-mm-yyyy'),14),
('N18','S5','P4','J4',800,to_date('04-05-2011','dd-mm-yyyy'),5),
('N19','S5','P5','J4',400,to_date('10-05-2011','dd-mm-yyyy'),10),
('N20','S5','P6','J4',500,to_date('16-05-2011','dd-mm-yyyy'),14),
('N21','S3','P3','J1',100,to_date('19-05-2011','dd-mm-yyyy'),16),
('N22','S3','P1','J1',300,to_date('21-05-2011','dd-mm-yyyy'),16),
('N23','S1','P1','J1',200,to_date('22-05-2011','dd-mm-yyyy'),7),
('N24','S1','P1','J4',700,to_date('26-05-2011','dd-mm-yyyy'),8),
('N25','S5','P2','J4',100,to_date('31-05-2011','dd-mm-yyyy'),6),
('N26','S2','P3','J1',400,to_date('02-06-2011','dd-mm-yyyy'),16),
('N27','S2','P3','J2',200,to_date('06-06-2011','dd-mm-yyyy'),16),
('N28','S2','P3','J4',500,to_date('17-06-2011','dd-mm-yyyy'),16),
('N29','S2','P3','J5',750,to_date('25-06-2011','dd-mm-yyyy'),16),
('N30','S5','P3','J3',200,to_date('01-07-2011','dd-mm-yyyy'),16),
('N31','S2','P5','J2',300,to_date('06-07-2011','dd-mm-yyyy'),9),
('N32','S3','P5','J2',500,to_date('08-07-2011','dd-mm-yyyy'),4),
('N33','S5','P2','J2',200,to_date('19-07-2011','dd-mm-yyyy'),6),
('N34','S3','P4','J2',200,to_date('25-07-2011','dd-mm-yyyy'),5),
('N35','S5','P2','J4',100,to_date('26-07-2011','dd-mm-yyyy'),6),
('N36','S5','P5','J5',600,to_date('27-07-2011','dd-mm-yyyy'),10),
('N37','S5','P1','J4',100,to_date('07-08-2011','dd-mm-yyyy'),9),
('N38','S5','P3','J4',200,to_date('09-08-2011','dd-mm-yyyy'),14),
('N39','S5','P4','J4',800,to_date('14-08-2011','dd-mm-yyyy'),5),
('N40','S5','P5','J4',400,to_date('15-08-2011','dd-mm-yyyy'),11),
('N41','S5','P6','J4',500,to_date('16-08-2011','dd-mm-yyyy'),13),
('N42','S2','P3','J7',1000,to_date('04-10-2011','dd-mm-yyyy'),17),

('N43','S2','P5','J7',50,to_date('04-10-2011','dd-mm-yyyy'),10),
('N44','S4','P6','J7',150,to_date('04-10-2011','dd-mm-yyyy'),12),
('N45','S2','P3','J3',450,to_date('10-01-2012','dd-mm-yyyy'),18),
('N46','S5','P1','J4',100,to_date('20-01-2012','dd-mm-yyyy'),8),
('N47','S4','P6','J7',700,to_date('17-02-2012','dd-mm-yyyy'),13),
('N48','S5','P4','J2',200,to_date('05-03-2012','dd-mm-yyyy'),6),
('N49','S2','P3','J6',700,to_date('02-04-2012','dd-mm-yyyy'),17),
('N50','S5','P5','J7',400,to_date('04-04-2012','dd-mm-yyyy'),11),
('N51','S5','P3','J4',200,to_date('29-04-2012','dd-mm-yyyy'),14),
('N52','S1','P2','J4',200,to_date('10-05-2012','dd-mm-yyyy'),8),
('N53','S4','P6','J3',600,to_date('12-07-2012','dd-mm-yyyy'),13);

---------- ТАБЛИЦА C – Заказчики (Client)
INSERT INTO c VALUES
('C1','Купер','Лондон ',7),
('C2','Росси','Рим ',3),
('C3','Эспозито','Рим ',0),
('C4','Перакис ','Афины',5),
('C5','Свен ','Осло',10),
('C6','Фишер','Берлин ',3);

---------- ТАБЛИЦА Q – Норма расхода деталей на одно изделие
INSERT INTO q (N_IZD,N_DET,KOL,N_Q) VALUES
('J4','P5',2,'Q1'),
('J4','P2',1,'Q2'),
('J4','P6',2,'Q3'),
('J5','P3',4,'Q4'),
('J5','P5',3,'Q5'),
('J6','P3',6,'Q6'),
('J7','P3',5,'Q7'),
('J7','P6',3,'Q9'),
('J7','P5',1,'Q8'),
('J1','P1',3,'Q10'),
('J1','P3',4,'Q11'),
('J2','P2',5,'Q12'),
('J2','P3',6,'Q13'),

('J2','P4',9,'Q14'),
('J2','P5',4,'Q15'),
('J2','P6',2,'Q16'),
('J3','P3',4,'Q17'),
('J3','P6',3,'Q18'),
('J4','P1',6,'Q19'),
('J4','P3',1,'Q20'),
('J4','P4',3,'Q21');

---------- ТАБЛИЦА W – Выпуск изделий (Working)

insert into W (N_PART,N_IZD,DATE_PART,KOL)
values ('W1','J1',to_date('02-03-2011','dd-mm-yyyy'),40),
('W2','J6',to_date('07-03-2011','dd-mm-yyyy'),35),
('W3','J1',to_date('10-03-2011','dd-mm-yyyy'),20),
('W4','J6',to_date('15-03-2011','dd-mm-yyyy'),30),
('W5','J3',to_date('09-04-2011','dd-mm-yyyy'),25),
('W6','J3',to_date('17-04-2011','dd-mm-yyyy'),20),
('W7','J7',to_date('19-04-2011','dd-mm-yyyy'),75),
('W8','J2',to_date('25-04-2011','dd-mm-yyyy'),20),
('W9','J7',to_date('28-04-2011','dd-mm-yyyy'),25),
('W10','J5',to_date('13-05-2011','dd-mm-yyyy'),40),
('W11','J1',to_date('15-06-2011','dd-mm-yyyy'),30),
('W12','J4',to_date('26-06-2011','dd-mm-yyyy'),40),
('W13','J1',to_date('03-07-2011','dd-mm-yyyy'),60),
('W14','J3',to_date('10-07-2011','dd-mm-yyyy'),40),
('W15','J2',to_date('15-07-2011','dd-mm-yyyy'),10),
('W16','J5',to_date('23-07-2011','dd-mm-yyyy'),75),
('W17','J4',to_date('03-08-2011','dd-mm-yyyy'),60),
('W18','J5',to_date('21-08-2011','dd-mm-yyyy'),50),
('W19','J4',to_date('16-09-2011','dd-mm-yyyy'),100),
('W20','J7',to_date('31-10-2011','dd-mm-yyyy'),50),
('W21','J3',to_date('17-01-2012','dd-mm-yyyy'),10),
('W22','J5',to_date('24-01-2012','dd-mm-yyyy'),130),
('W23','J1',to_date('21-02-2012','dd-mm-yyyy'),25),
('W24','J5',to_date('19-03-2012','dd-mm-yyyy'),40),

('W25','J6',to_date('12-04-2012','dd-mm-yyyy'),90),
('W26','J7',to_date('27-04-2012','dd-mm-yyyy'),200),
('W27','J2',to_date('06-05-2012','dd-mm-yyyy'),30),
('W28','J4',to_date('18-05-2012','dd-mm-yyyy'),60),
('W29','J3',to_date('01-08-2012','dd-mm-yyyy'),100),
('W30','J1',to_date('20-08-2012','dd-mm-yyyy'),50),
('W31','J6',to_date('25-08-2012','dd-mm-yyyy'),25);

---------- ТАБЛИЦА E – Издержки на производство
INSERT INTO e
VALUES ('E1','J1',to_date('2011-01-01','YYYY-MM-DD'),80),
('E2','J1',to_date('2011-05-01','YYYY-MM-DD'),85),
('E3','J2',to_date('2011-05-01','YYYY-MM-DD'),220),
('E4','J3',to_date('2011-03-01','YYYY-MM-DD'),100),
('E5','J4',to_date('2011-06-01','YYYY-MM-DD'),130),
('E6','J4',to_date('2011-09-01','YYYY-MM-DD'),140),
('E7','J5',to_date('2011-05-01','YYYY-MM-DD'),85),
('E8','J5',to_date('2011-07-01','YYYY-MM-DD'),90),
('E9','J5',to_date('2011-08-01','YYYY-MM-DD'),95),
('E10','J6',to_date('2011-03-01','YYYY-MM-DD'),95),
('E11','J6',to_date('2011-07-01','YYYY-MM-DD'),105),
('E12','J7',to_date('2011-04-01','YYYY-MM-DD'),130),
('E13','J7',to_date('2011-08-01','YYYY-MM-DD'),135);

---------- ТАБЛИЦА V – Рекомендованная цена
insert into V (N_V,N_IZD,DATE_BEGIN,COST)
values ('V1','J1',to_date('01-01-2011','dd-mm-yyyy'),280),
('V2','J2',to_date('01-01-2011','dd-mm-yyyy'),420),
('V3','J3',to_date('01-01-2011','dd-mm-yyyy'),300),
('V4','J4',to_date('01-01-2011','dd-mm-yyyy'),330),
('V5','J5',to_date('01-01-2011','dd-mm-yyyy'),290),
('V6','J6',to_date('01-01-2011','dd-mm-yyyy'),295),
('V7','J7',to_date('01-01-2011','dd-mm-yyyy'),330),
('V8','J1',to_date('01-05-2011','dd-mm-yyyy'),300),
('V9','J2',to_date('01-05-2011','dd-mm-yyyy'),440),
('V10','J3',to_date('01-05-2011','dd-mm-yyyy'),295),

('V11','J4',to_date('01-05-2011','dd-mm-yyyy'),440),
('V12','J5',to_date('01-05-2011','dd-mm-yyyy'),285),
('V13','J6',to_date('01-05-2011','dd-mm-yyyy'),305),
('V14','J7',to_date('01-05-2011','dd-mm-yyyy'),350),
('V15','J1',to_date('01-11-2011','dd-mm-yyyy'),360),
('V16','J2',to_date('01-11-2011','dd-mm-yyyy'),500),
('V17','J3',to_date('01-11-2011','dd-mm-yyyy'),350),
('V18','J4',to_date('01-11-2011','dd-mm-yyyy'),510),
('V19','J5',to_date('01-11-2011','dd-mm-yyyy'),340),
('V20','J6',to_date('01-11-2011','dd-mm-yyyy'),360),
('V21','J7',to_date('01-11-2011','dd-mm-yyyy'),420);

---------- ТАБЛИЦА R –Заказы
insert into R (N_REAL,N_IZD,N_CL,DATE_ORDER,DATE_PAY,DATE_SHIP, KOL, COST)
values ('R1','J1','C1',to_date('10-02-2011','dd-mm-yyyy'),to_date('13-02-2011','dd-mm-yyyy'),to_date('03-03-2011','dd-mm-yyyy'),15,275),
('R2','J6','C2',to_date('01-03-2011','dd-mm-yyyy'),to_date('06-03-2011','dd-mm-yyyy'),to_date('13-03-2011','dd-mm-yyyy'),30,300),
('R3','J1','C3',to_date('05-03-2011','dd-mm-yyyy'),to_date('07-03-2011','dd-mm-yyyy'),to_date('10-03-2011','dd-mm-yyyy'),30,280),
('R4','J3','C2',to_date('03-04-2011','dd-mm-yyyy'),to_date('07-04-2011','dd-mm-yyyy'),to_date('09-04-2011','dd-mm-yyyy'),25,300),
('R5','J2','C5',to_date('22-04-2011','dd-mm-yyyy'),to_date('24-04-2011','dd-mm-yyyy'),to_date('26-04-2011','dd-mm-yyyy'),15,400),
('R6','J7','C5',to_date('08-05-2011','dd-mm-yyyy'),to_date('11-05-2011','dd-mm-yyyy'),to_date('13-05-2011','dd-mm-yyyy'),70,325),
('R7','J5','C2',to_date('15-05-2011','dd-mm-yyyy'),to_date('19-05-2011','dd-mm-yyyy'),to_date('22-05-2011','dd-mm-yyyy'),40,290),
('R8','J3','C4',to_date('20-05-2011','dd-mm-yyyy'),to_date('23-05-2011','dd-mm-yyyy'),to_date('24-05-2011','dd-mm-yyyy'),15,290),
('R9','J1','C6',to_date('09-06-2011','dd-mm-yyyy'),to_date('15-06-2011','dd-mm-yyyy'),to_date('20-06-2011','dd-mm-yyyy'),45,285),
('R10','J6','C2',to_date('15-06-2011','dd-mm-yyyy'),to_date('16-06-2011','dd-mm-yyyy'),to_date('16-06-2011','dd-mm-yyyy'),20,300),
('R11','J4','C3',to_date('19-06-2011','dd-mm-yyyy'),to_date('27-06-2011','dd-mm-yyyy'),to_date('27-06-2011','dd-mm-yyyy'),30,445),

('R12','J1','C5',to_date('01-07-2011','dd-mm-yyyy'),to_date('03-07-2011','dd-mm-yyyy'),to_date('05-07-2011','dd-mm-yyyy'),50,277),
('R13','J5','C1',to_date('01-09-2011','dd-mm-yyyy'),to_date('04-09-2011','dd-mm-yyyy'),to_date('04-09-2011','dd-mm-yyyy'),100,285),
('R14','J4','C4',to_date('20-09-2011','dd-mm-yyyy'),to_date('21-09-2011','dd-mm-yyyy'),to_date('21-09-2011','dd-mm-yyyy'),120,420),
('R15','J3','C2',to_date('20-10-2011','dd-mm-yyyy'),to_date('23-10-2011','dd-mm-yyyy'),to_date('24-10-2011','dd-mm-yyyy'),25,295),
('R16','J7','C6',to_date('01-11-2011','dd-mm-yyyy'),to_date('03-11-2011','dd-mm-yyyy'),to_date('06-11-2011','dd-mm-yyyy'),80,415),
('R17','J1','C1',to_date('11-01-2012','dd-mm-yyyy'),to_date('13-01-2012','dd-mm-yyyy'),to_date('15-01-2012','dd-mm-yyyy'),10,280),
('R18','J6','C2',to_date('15-04-2012','dd-mm-yyyy'),to_date('22-04-2012','dd-mm-yyyy'),to_date('23-04-2012','dd-mm-yyyy'),60,350),
('R19','J4','C6',to_date('13-05-2012','dd-mm-yyyy'),to_date('15-05-2012','dd-mm-yyyy'),to_date('20-05-2012','dd-mm-yyyy'),50,500),
('R20','J2','C6',to_date('01-06-2012','dd-mm-yyyy'),to_date('05-06-2012','dd-mm-yyyy'),null,30,490),
('R21','J7','C2',to_date('20-06-2012','dd-mm-yyyy'),to_date('21-06-2012','dd-mm-yyyy'),to_date('25-06-2012','dd-mm-yyyy'),150,415),
('R22','J3','C4',to_date('10-08-2012','dd-mm-yyyy'),to_date('12-08-2012','dd-mm-yyyy'),to_date('13-08-2012','dd-mm-yyyy'),60,330),
('R23','J5','C3',to_date('15-08-2012','dd-mm-yyyy'),to_date('19-08-2012','dd-mm-yyyy'),to_date('21-08-2012','dd-mm-yyyy'),120,350),
('R24','J4','C3',to_date('20-08-2012','dd-mm-yyyy'),null,null,70,520),
('R25','J3','C2',to_date('20-08-2012','dd-mm-yyyy'),to_date('22-08-2012','dd-mm-yyyy'),to_date('26-08-2012','dd-mm-yyyy'),45,340),
('R26','J1','C3',to_date('25-08-2012','dd-mm-yyyy'),to_date('27-08-2012','dd-mm-yyyy'),null,50,290),
('R27','J6','C2',to_date('15-09-2012','dd-mm-yyyy'),to_date('17-09-2012','dd-mm-yyyy'),null,70,350);

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Стасышин В.М.*, *Стасышина Т.Л.* Технологии доступа к базам данных: учеб. пособие. – Электронная версия, 2014. – 176 с.

2. *Стасышин В.М.* Язык структурных запросов SQL: учеб. пособие. – Новосибирск: Изд-во НГТУ, 1996. – 33 с.

3. *Стасышин В.М.* Проектирование информационных систем и баз данных. – Новосибирск: Изд-во НГТУ, 2012. – 100 с.

ОГЛАВЛЕНИЕ

РАБОТА С БАЗАМИ ДАННЫХ

Методические указания

Редактор *И.Л. Кескевич*
Выпускающий редактор *И.П. Брованова*
Корректор *И.Е. Семенова*
Компьютерная верстка *Л.А. Веселовская*

Налоговая льгота – Общероссийский классификатор продукции
Издание соответствует коду 95 3000 ОК 005-93 (ОКП)

Подписано в печать 08.02.2016. Формат 60 × 84 1/16. Бумага офсетная. Тираж 100 экз.
Уч.-изд. л. 5,58. Печ. л. 6,0. Изд. № 193/15. Заказ № Цена договорная

Отпечатано в типографии
Новосибирского государственного технического университета
630073, г. Новосибирск, пр. К. Маркса, 20