ИННОВАЦИОННАЯ ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА

Проект 2: индивидуальная траектория обучения и качество образования

Цель: ориентированное на требования рынка образовательных услуг улучшение качества подготовки и переподготовки специалистов

Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования

Владимирский государственный университет

Кафедра вычислительной техники

Методические указания к лабораторным работам по курсу «Управление базами данных»

Составители:
И. Р. ДУБОВ
Ю. Г. АРШАНКИН
Ю. В. КОРОЛЕВ

Владимир 2008

УДК 004.658(076)
ББК 32.973-018.2я7
М54

Рецензент

Доктор технических наук, профессор
Владимирского государственного университета
Р.И. Макаров

Печатается по решению редакционного совета
Владимирского государственного университета

М54 Методические указания к лабораторным работам по курсу “Управление базами данных” / сост.: И. Р. Дубов, Ю. Г. Аршанкин, Ю. В. Королев; Владим. гос. ун-т. – Владимир : Изд-во Владим. гос. ун-та, 2008. – 24 с.

Содержат методические указания для выполнения четырех лабораторных работ по курсу «Управление базами данных». В работах рассматриваются эффективность индексации таблиц баз данных, механизмы транзакций и блокировок, принципы и приемы администрирования СУБД, а также программный интерфейс ODBC.

Предназначены для студентов специальности 230101 – вычислительные машины, комплексы, системы и сети дневной формы обучения.

Библиогр.: 3 назв.

УДК 004.658(076)
ББК 32.973-018.2я7

Введение

В настоящее время большое количество задач в различных прикладных областях решается с помощью программного обеспечения, использующего системы управления базами данных (СУБД) для хранения, доступа и обработки данных. СУБД позволяют значительно снизить сложность отдельных приложений и повысить степень их унификации. Это достигается тем, что на СУБД перекладывается ряд задач, общих для разных приложений, и при этом промышленные СУБД поддерживают определенные стандарты.

База данных (БД) содержит не только сами данные, но и их полное описание. Это позволяет единообразно реализовать доступ к данным из любых приложений даже в том случае, когда приложения создаются различными разработчиками программного обеспечения. СУБД обеспечивает непротиворечивое состояние данных при работе любых приложений.

Обработка одних и тех же данных может выполняться разными приложениями параллельно. Для предотвращения побочных эффектов в этой ситуации СУБД изолирует параллельно выполняемые транзакции, повышая надежность результатов вычислений. Кроме основных функций по хранению и обработке данных, СУБД обеспечивает безопасность доступа к данным. Это позволяет защитить данные от умышленного или случайного повреждения и ограничить доступ к конфиденциальной информации.

Курс «Управление базами данных» дает студентам основные понятия о способах управления базами данных и средствах их администрирования. В процессе выполнения предлагаемых лабораторных работ студенты должны глубже изучить теоретический материал и получить соответствующие практические навыки. Для успешного выполнения лабораторных работ студенты должны уже знать основные положения теории баз данных, реляционную теорию и язык структурированных запросов SQL.

В предлагаемом курсе лабораторных работ рассматриваются эффективность организации базы данных, параллельная обработка данных (механизм транзакций) и администрирование СУБД. Также предлагается исследовать программный интерфейс ODBC. Все лабораторные работы выполняются с использованием СУБД Microsoft SQL Server. Для выполнения работы, посвященной ODBC, дополнительно требуется составить программу на языке C++.

Лабораторная работа № 1
Исследование эффективности индексирования

Цель работы: изучить способы формирования индексов и исследовать эффективность индексирования.

Порядок выполнения работы

1. Запустить на выполнение Query Analyzer. Подсоединиться к SQL Server с именем sa и пустым паролем. Дальнейшие действия выполнять с помощью SQL-скриптов.
2. Создать базу данных.
3. Создать в базе данных новую таблицу. Структура таблицы должна соответствовать индивидуальному заданию из практикума по дисциплине “Базы данных”. На данном этапе не задавать ограничения для первичного ключа таблицы (за исключением not null ограничения).
4. При помощи адаптированного скрипта А внести в таблицу данные.
5. Исследовать время выполнения запросов на выборку по отдельным полям таблицы. Сопоставить время выполнения запросов, в которых условие определяется числовыми и символьными полями.
6. Создать некластерные индексы по двум различным полям. Провести исследование времени выполнения выборки. Одно из полей должно быть тем полем, которое предполагается использовать в качестве первичного ключа.
7. Удалить некластерный индекс для поля первичного ключа и ввести ограничение для первичного ключа.
8. Провести исследование времени выборки по первичному ключу (некластерному индексу).
9. Сопоставить результаты исследования, сделать выводы.

Рекомендации

1. Для создания базы данных использовать оператор

```
create database IndicesResearchDB
```

2. Изменение текущей базы данных (перед созданием первой таблицы)

```
use IndicesResearchDB
```

3. Создание таблицы

```
create table Product ( ProductId int not null
                     , Name nvarchar(50)
                     , Weight float
                     , ProductCategoryId int )
```

4. Вариант заполнения таблицы данными для исследования (скрипт А)

```
declare @startTime datetime
      , @endTime   datetime
      , @timeDiff  int

set @startTime = getdate()
set @endTime = @startTime
set @timeDiff = 1000 * 60 * 5 -- 5 минут

declare @productId         int
      , @name              nvarchar(50)
      , @productCategoryId int
      , @weight            float

set @productId = 0

while datediff(ms, @startTime, @endTime) < @timeDiff
begin
    set @productId         = @productId + 1
    set @name              = str( round( rand() *
1000, 0 ) )
    set @productCategoryId = round( rand() * 300, 0 )
    set @weight            = round( rand() * 2000, 0
)

    insert into Product (ProductId , Name , Weight ,
ProductCategoryId )
                 values (@productId, @name, @weight,
@productCategoryId)


    set @endTime = getdate()
end

select count(*) from Product
```

```
print str( datediff( ms, @startTime, @endTime) ) + '
ms'
```

5. Получение времени выборки

```
dbcc dropcleanbuffers -- clean buffers
declare @startTime dateTime
      , @endTime   dateTime
      , @count     int

select @count = count(*) from Product

SET @startTime = getdate()

declare @counter int
set @counter = 100

while (@counter > 0)
begin
    select * from Product
     where ProductId = round( rand() * @count, 0)

    set @counter = @counter - 1
end

set @endTime = getdate()
print str( datediff( ms, @startTime, @endTime)) + '
ms'
```

6. Создание некластерного индекса

```
create index IX_Product_ProductCategoryId ON Product
(ProductCategoryId)
```

7. Удаление некластерного индекса

```
drop index IX_Product_ProductCategoryId on Product
```

8. Создание кластерного индекса

```
ALTER TABLE dbo.Product ADD CONSTRAINT
    PK_Product_ProductId PRIMARY KEY CLUSTERED
    (
```

```
        ProductId
    )
```

9. Удаление кластерного индекса

alter table Product drop constraint PK_Product_ProductId

```
alter table Product drop constraint
PK_Product_ProductId
```

10. Для просмотра свойств таблицы (в том числе индексов) использовать в Enterprise Manager локальное меню для выделенной таблицы.

Содержание отчета

1. Тексты всех разработанных SQL скриптов.
2. Результаты сравнения времени доступа при отсутствии индекса, при наличии некластерного индекса и при наличии кластерного индекса.
3. Выводы по работе.

Контрольные вопросы

1. Физическое представление данных в БД.
2. Плотный индекс.
3. Разреженный индекс.
4. B-деревья.
5. Индексирование по атрибутам с неуникальными значениями.

Лабораторная работа № 2
Управление транзакциями и блокировки

Цель работы: изучить механизм формирования транзакций, уровни изоляции транзакций и взаимные блокировки.

Теоретические положения

В общем случае результат выполнения транзакции не должен зависеть от других выполняющихся одновременно транзакций. Но полная изоляция транзакций делает невозможной параллельную обработку данных. В большинстве случаев такая строгость не нужна, и поэтому были введены так называемые «уровни изоляции» (Isolation Level), которые определяют степень параллелизма выполнения транзакций. Чем ниже уровень изоляции, тем выше степень параллелизма и тем больше риск «неправильного» выполнения транзакции.

В стандарте ANSI SQL вводятся четыре уровня изоляции. И по названиям, и по поведению уровни изоляции в Microsoft SQL Server 7/2000 полностью соответствуют описанным в стандарте.

В стандарте уровни изоляции описываются при помощи «феноменов» – побочных эффектов низкой изолированности транзакций. Всего их четыре:

1) грязная запись (Dirty Write);
2) грязное чтение (Dirty Read);
3) неповторяющееся чтение (Repeatable Read);
4) фантомы (Phantoms).

Используются четыре уровня изоляции, которые устраняют вышеперечисленные феномены:

1. Read Uncommitted – самый низкий уровень изоляции. При этом уровне изоляции ликвидируется феномен «грязная запись». Но транзакция может читать «грязные» данные незафиксированных транзакций. «Грязные» они потому, что если незафиксированная транзакция будет откачена, то получится, что были прочитаны никогда не существовавшие данные.
2. Read Committed – этот уровень изоляции решает проблему грязного чтения, т.е. в транзакции с таким уровнем изоляции невозможно прочитать данные незафиксированных тран-

закций. Однако остается феномен неповторяющегося чтения. Суть этого феномена в том, что если первая транзакция один раз прочитала данные, а потом вторая их изменила и зафиксировалась, то повторное чтение тех же данных первой транзакцией вернет уже измененные данные. Отметим, что Microsoft SQL Server использует этот уровень изоляции по умолчанию.

3. Repeatable Read – этот уровень решает предыдущую проблему, но при этом возможно появление фантомов. Изменение однажды прочитанных первой транзакцией данных другими транзакциями (до фиксации первой) невозможно. Однако если первая транзакция сделала выборку по какому-то условию, а потом вторая транзакция добавила новые данные, этому условию удовлетворяющие, и зафиксировалась, то повторная выборка первой транзакцией по тому же условию вернет в том числе и добавленные данные – фантомы.
4. Serializable – при этом уровне изоляции никакие фантомы невозможны в принципе, равно как и другие феномены. Этот уровень изоляции ни на какие феномены не опирается, просто требуется, чтобы результат параллельного выполнения транзакций был таким же, как если бы они выполнялись последовательно.

Порядок выполнения работы

В данной лабораторной работе предлагается исследовать поведение СУБД MS SQL Server при различных уровнях изоляции. Для этого надо выполнить следующие пункты:

1. Создать таблицу в базе данных. Структура таблицы должна соответствовать индивидуальному заданию из практикума по дисциплине “Базы данных”.

```
create table SomeTable (x int, y int)
go
```

2. Вставить одну запись в таблицу.

```
insert into SomeTable (x, y) values (1, 1)
```

3. В одном окне SQL Query Analyzer написать скрипт обновления записи с явно объявленной транзакцией.

```
begin tran

update SomeTable
```

4. Выделить строки 1-6 и запустить скрипт на выполнение (F5). SQL Query Analyzer выполнит выделенные строки, т.е. мы получим незафиксированную транзакцию.

```
    set x = 2
where y = 1

commit
```

5. В другом окне SQL Query Analyzer попробовать выбрать данные из таблицы при read committed и read uncommitted уровнях изоляции.

```
set transaction isolation level read uncommitted
begin tran

select * from SomeTable

commit

set transaction isolation level read committed
begin tran

select * from SomeTable

commit
```

6. В первом окне выделить 7-ю стоку и запустить скрипт на выполнение. Повторить запросы из пункта 5. Объяснить полученные результаты.

7. Аналогично пунктам 3-6 показать проблемы при уровнях изоляции Read Committed, Repeatable Read и Serializable. Для переключения уровней изоляции использовать следующие скрипты:

```
set transaction isolation level repeatable read
set transaction isolation level serializable
```

8. Создать вторую таблицу в базе данных. Структура таблицы должна соответствовать индивидуальному заданию из практикума по дисциплине “Базы данных”.

```
create table AnotherTable (x int, y int)
go
```

9. Вставить в обе таблицы по две записи.

```
delete from SomeTable

insert into SomeTable (x, y) values (1, 1)
insert into SomeTable (x, y) values (2, 2)

insert into AnotherTable (x, y) values (1, 1)
insert into AnotherTable (x, y) values (2, 2)
```

10. В одном окне SQL Query Analyzer написать скрипт обновления записей в двух таблицах в контексте одной транзакции.

```
begin tran

update SomeTable
   set x = 1
 where y = 1

update AnotherTable
   set x = 2
 where y = 2

commit
```

11. Во втором окне SQL Query Analyzer написать скрипт обновления записей в двух таблицах, но в обратном порядке.

```
begin tran

update AnotherTable
   set x = 1
 where y = 1

update SomeTable
   set x = 2
 where y = 2

commit
```

12. В первом окне выделить строки 1-6 и запустить скрипт на выполнение, переключиться во второе окно, выделить строки 1-6, запустить скрипт на выполнение. Переключиться в первое окно, выделить оставшийся скрипт, выполнить его. Переключиться во второе окно, выделить оставшийся скрипт и выполнить его. Объяснить полученный результат.

13. В одном из окон заменить второе обновление данных на выборку из той же таблицы.

```
begin tran
```

```
update AnotherTable
   set x = 1
 where y = 1

select * from SomeTable

commit
```

14. Выполнить шаги, приведенные в пункте 12. Избавиться от взаимной блокировки.

Содержание отчета

1. Тексты всех разработанных SQL скриптов, включая скрипты создания баз данных и таблиц.
2. Описание и анализ содержимого таблиц для каждого шага выполнения работы.
3. Выводы по работе.

Контрольные вопросы

1. Атомарность, согласованность, изолированность и устойчивость транзакции.
2. Проблема потерянного обновления.
3. Проблема зависимости от нефиксированных результатов.
4. Проблема несогласованной обработки.
5. Вложенные транзакции.
6. Точка сохранения.
7. Блокировки, взаимные блокировки транзакций.
8. Двухфазный протокол блокировки транзакций.
9. Метод временных отметок.
10. Иерархия уровней детализации блокируемых объектов.
11. Средства восстановления баз данных.
12. Методы восстановления баз данных.

Лабораторная работа № 3
Система безопасности сервера баз данных

Цель работы: получить навыки в обеспечении безопасности сервера баз данных.

Порядок выполнения работы

Используя учетную запись SA, зарегистрировать себя в качестве администратора сервера. Для выполнения работы следует подготовить базу данных в соотвествии с вариантом индивидуального задания, в которой должно быть не менее четырех таблиц tab1, tab2, tab3, tab4.

Имеются следующие реальные пользователи: usr1, usr2, usr3, usr4, usr5. Обеспечить следующие возможности для работы с данными:

- usr1: Может создавать новые учетные записи. Не имеет доступа к базе данных.
- usr2: Не может выполнять действий по конфигурированию сервера. Может читать данные из tab1 и tab2, но не может читать данные из tab3 и tab4. Может создавать новые таблицы.
- usr3: Может только выполнять несистемные хранимые процедуры.
- usr4: Может только добавлять данные в tab3.
- usr1 и usr2 работают в одном отделе, работникам которого предоставлено разрешение на чтение, изменение и добавление данных в tab1 и tab2.
- usr3 и usr4 не имеют доступа к tab1 и tab2.

Разработать хранимую процедуру, которая добавляет данные в tab1, tab2, tab3, tab4. Сделать так, чтобы добавление с помощью этой хранимой процедуры мог выполнить любой пользователь базы данных (воспользоваться ролью приложения).

В процессе выполнения работы следует создать SQL-скрипты, которые обеспечат проверку правильности установки прав пользователей.

Имена пользователей и названия таблиц сделать соответствующими предметной области.

Содержание отчета

1. Тексты всех разработанных SQL скриптов, включая скрипты создания баз данных, таблиц, хранимых процедур.
2. Текст скриптов, с помощью которых проверялись варианты доступа к базе данных.
3. Выводы по работе.

Контрольные вопросы

1. Учетная запись Windows NT.
2. Учетная запись MS SQL Server.
3. Пользователь базы данных.
4. Роли сервера.
5. Фиксированные роли базы данных.
6. Пользовательские роли.
7. Роль приложения.
8. Права доступа к элементам базы данных.
9. Разрешение на выполнение команд SQL-Transact.
10. Неявное отклонение разрешений.

Лабораторная работа № 4
Изучение ODBC

Цель работы: получить навыки проектирования приложений, взаимодействующих с источниками данных через ODBC.

Порядок выполнения работы

1. Создать базу данных в MS SQL Server и в ней создать таблицу в соответствии со своим индивидуальным заданием. Таблица должна содержать не менее 3 полей различных типов: целое, вещественное, символьное. Вставить в таблицу 10 новых записей.

2. Зарегистрировать доступ к созданной таблице как источник данных ODBC. Для этого выбрать в панели управления службу регистрации источников данных (“Источники данных ODBC”). Создать системное имя источника, выбрав драйвер “SQL Server”. Для подключения выбрать тот сервер баз данных, на котором была создана база данных с тестовой таблицей. Указать действующее имя учетной записи и пароль. Указать созданную таблицу в качестве таблицы по умолчанию. Проверить правильность подключения.

3. Разработать приложение, осуществляющее доступ к созданному источнику данных через ODBC. Для этого создать проект в Visual Studio типа Win32 Console. В нем создать новый файл для головной функции. Поместить в него программу, приведенную в приложении. Изучить данную программу. Изменить программу таким образом, чтобы она выполняла получение данных из тестовой таблицы, созданной в п. 1. При этом должны быть реализованы два варианта доступа к данным: через имя источника данных ODBC и с помощью непосредственного обращения к драйверу MS SQL Server. Отладить программу так, чтобы она выполнялась без ошибок.

Содержание отчета

1. Тексты всех разработанных SQL скриптов, включая скрипты создания базы данных, таблицы, добавления новых записей в тестовую таблицу.
2. Текст программы.
3. Выводы по работе.

Контрольные вопросы

1. Структура ODBC. Менеджер драйверов, драйверы источников данных.
2. Дескрипторы объектов ODBC.
3. Алгоритм взаимодействия приложения с ODBC.
4. Преобразование типов данных в ODBC.
5. Связывание столбцов таблицы с переменными приложения.
6. Регистрация источников данных ODBC.

ПРИЛОЖЕНИЕ
Пример программы, реализующей взаимодействие с ODBC

```
#include <windows.h>
#include <odbcinst.h>
#include <sql.h>
#include <sqlext.h>
#include <ctype.h>
#include <string>
#include <iostream>

using namespace std;

SQLCHAR connectionStringDSN[256] =
"DSN=NorthWindConnection;UID=sa; PWD=;";
SQLCHAR connectionStringDriver[256] =
  "DRIVER={SQL Server};SERVER=IDUBOV;"
  "UID=sa;PWD=;DATABASE=Northwind;";

void errorMessage(const char* s)
{
    cout<< "*** Error ***  " << s << endl;
}

void Message(const char* s)
{
    cout << s << endl;
}

void DoSelect(SQLCHAR connectionString[256])
{
    HENV    hEnv;
    HDBC    hDbc;
    HSTMT   hStmt = SQL_NULL_HSTMT;
    RETCODE rc;
    SQLSMALLINT cbOutConStr = 0;

    // Получить соединение с СУБД
    SQLCHAR outConnectionString[256];
    SQLAllocEnv(&hEnv);
```

```
    SQLAllocConnect(hEnv, &hDbc);
    rc = SQLDriverConnect(hDbc, NULL,
        connectionString, SQL_NTS,
        outConnectionString,
sizeof(outConnectionString),
        &cbOutConStr, SQL_DRIVER_COMPLETE);
//SQL_DRIVER_NOPROMPT SQL_DRIVER_COMPLETE

    if(!(rc == SQL_SUCCESS) && !(rc ==
SQL_SUCCESS_WITH_INFO))
    {
        errorMessage("Error SQLDriverConnect");
        return;
    }

    SQLHSTMT hstmt;
    rc = SQLAllocStmt(hDbc, &hstmt);
    SQLCHAR command[256] = "SELECT EmployeeId,
Address FROM Employees";
    rc = SQLExecDirect(hstmt, command, SQL_NTS);

    SQLINTEGER EmployeeId;
    SQLCHAR Address[60];
    SQLINTEGER cbEmployeeId, cbAddress;

    if (rc == SQL_SUCCESS || rc ==
SQL_SUCCESS_WITH_INFO) {
        /* Bind columns 1, 2 */
        SQLBindCol(hstmt, 1, SQL_C_ULONG,
&EmployeeId, 0, &cbEmployeeId);
        SQLBindCol(hstmt, 2, SQL_C_CHAR, Address, 60,
&cbAddress);
        /* Fetch and print each row of data. On */
        /* an error, display a message and exit. */
        while (TRUE) {
            rc = SQLFetch(hstmt);
            if (rc == SQL_ERROR)
            {
                errorMessage("Error in the fetch");
            }
            if (rc == SQL_SUCCESS || rc ==
SQL_SUCCESS_WITH_INFO)
```

```
            {
                cout << EmployeeId << " " << Address
<< endl;
            } else {
                break;
            }
        }
        //    SQLFreeStmt(hstmt, SQL_DROP);
        SQLDisconnect(hDbc);
        SQLFreeConnect(hDbc);
        SQLFreeEnv(hEnv);
    }
    else
        errorMessage("Exec SQL error");
}


    void ListDSN()
    {

        const short SQL_MAX_DSN_LENGTH_ =
SQL_MAX_DSN_LENGTH;

        UCHAR szDSN[SQL_MAX_DSN_LENGTH+1];
        UCHAR szDescription[256];
        short wDSNLen;
        SQLSMALLINT wDesLen;
        int retCode;
        SQLHENV hEnv = NULL;
        string DSNName;
        string resultString;
        string Descr;

        SQLAllocEnv(&hEnv);

        retCode = SQLDataSources(hEnv,
SQL_FETCH_FIRST, szDSN, SQL_MAX_DSN_LENGTH_+1,
            &wDSNLen, szDescription, 256, &wDesLen);

        while(retCode == SQL_SUCCESS || retCode ==
SQL_SUCCESS_WITH_INFO)
        {
```

```
            DSNName = (string)((char *) szDSN);
            Descr = (string)((char *) szDescription);
            resultString += DSNName;
            resultString += "\n";

            retCode = SQLDataSources(hEnv,
SQL_FETCH_NEXT, szDSN,SQL_MAX_DSN_LENGTH_+1,
                &wDSNLen, szDescription, 256,
&wDesLen);
        }

        SQLFreeEnv(hEnv);
        Message(resultString.c_str());
    }

    void ShowMenu()
{
    int choice = 0;
    bool done = false;
    do
    {
        cout << "\n*** MENU ***" << endl;
        cout << "0 - output" << endl;
        cout << "1 - list all DNS" << endl;
        cout << "2 - select (using DNS)" << endl;
        cout << "3 - select (using driver directly)"
<< endl;
        cout << "Choice : ";
        cin >> choice;
        switch(choice)
        {
        default : done = true; break;
        case 1 : ListDSN(); break;
        case 2 : DoSelect(connectionStringDSN);
break;
        case 3 : DoSelect(connectionStringDriver);
break;
        }
    }while(!done);
}
```

```
int main(int count, char* cparams[])
{
    ShowMenu();
    return 0;
}
```

Библиографический список

1. Карпова, Т. Базы данных: модели, разработка, реализация / Т. Карпова. – СПб.: Питер, 2002. – 304 с. – ISBN 5-272-00278-4.

2. Коннолли, Т. Базы данных: проектирование, реализация, сопровождение / Т. Конноли, К. Бегг, А. Страчан. – 2-е изд. – М.: Вильямс, 2001. – 1120 с. – ISBN 5-8459-0109-X.

3. Шкарина, Л. Язык SQL / Л. Шкарина. – СПб.: Питер, 2001. – ISBN 5-318-00195-5.

Оглавление

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ЛАБОРАТОРНЫМ РАБОТАМ ПО КУРСУ
«УПРАВЛЕНИЕ БАЗАМИ ДАННЫХ»

Составители
Дубов Илья Ройдович
Аршанкин Юрий Георгиевич
Королев Юрий Валентинович

Ответственный за выпуск – зав. кафедрой профессор В.Н. Ланцов

Подписано в печать 25.03.08.
Формат 60x84/16. Усл. печ. л. Тираж 100 экз.
Заказ
Издательство
Владимирского государственного университета.
600000, Владимир, ул. Горького, 87.