МІНІСТЕРСТВО ОСВІТИ УКРАЇНИ

КИЇВСЬКИЙ УНІВЕРСІТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет кібернетики

Кафедра теоретичної кібернетики

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

БАКЛАВРА

На тему : Швидкість обробки запитів

на SQL-серверах

Виконавець: студент 4 курсу

Дима Павло Костянтинович

Науковий керівник : ассистент Коваль Ю.В.

Випускна кваліфікаційна робота бакалавра

Рекомендована до захисту в ДЕК

Протокол № 10, від “25” травня 2000 р.

Зав.кафедри проф. Белов Ю.А

м Київ - 2000

ЗМІСТ

1.Вступ………..………………………………………………………….2

2.Швидкість роботи SQL сервера…………………………………….5

2.1 Технологія DAO…….………………………………………………6

2.2 Технологія ODBC…..………………………………………………6

2.3 JSCRIPT……………………………………………………………..7

3. Результати тестування..…………………………………………….17

Висновок………………………………………………………………..21

Список використаної літератури…………………………………….22

**Частина 1 : ВСТУП**

У світі сучасних інформаційних технологій особливо виділяє-ться всесвітня мережа Internet. Опорним елементом в цій мережі є сервера баз данних. Саме на сервері зберігається інформація до якої можуть отримати доступ користувачі. При роботі користувача з базою данних в Internet, він посилає запит до бази даннихю цей запит сформовано за правилами мови SQL (Structured Query Language) – мова структурованих запитів.

В мережах використовуються такі SQL–сервера як Oracle, Microsoft SQL Server, Sybase, dbase, Informix, Progress та інші. Як приклад SQL – сервера візьмемо Microsoft SQL Server 7.0 .

Система MS SQL Server - це багаторівнева система обміну данними від клієнта до сервера, дозволяє створювати і підтримувати бази данних.

На відіну від бази данних орієнтованих на великі комп”ютери та міні-комп”ютери, до серверної бази данних користувачі отримують доступ зі своїх комп”ютерів, а не з допомогою терміналів. При цьому механізми роботи SQL-сервера допомагає розв”язувати проблеми, що виникають у результаті звернень до бази данних багатьох комп”ютерів, кожен з яких може незалежно звертатись до різних частин бази данних, що зберігається на сервері.

Зв”язок системи клієнт/сервер з комп”ютерами-клієнтами забезпечу-ється через засоби передачі данних і базової операційної системи. Для цього система клієнт/сервер MS SQL Server 7.0 використовує вбудовані компоненти ОС Windows NT.

Серверна СУБД MS SQL Server 7.0 добавляє до мережевих компонент окремі елементи сервіса, такі як OLE-DB відповідає за зв”язок та упровадження об”єктів баз даних і ODBC - відкритий інтерфейс баз даних. Ці компоненти забезпечують тісний зв”язок необхідного програмного забезпечення.

SQL Server 7.0 має багато інструментів, призначених для розробки зовнішнього інтерфейсу бази даних. Окрім мови SQL для зберігання, вибірки данних можна викоритовувати додатки [приложения] з графічним інтерфейсом користувача GUI. Це дуже спрощує розробку баз даних, таблиць і зв”язків між ними. Зручно використовувати SQL Server разом з програмами для роботи в Internet.

Центральним компонентом системи SQL Server є реляційна база даних та її структура. В реляційній базі даних інформація представляється в вигляді рядків та стовбців. Рядки не впорядковані, крім випадку, коли для таблиці створюється кластерний індекс. Сама структура операторів запитів, з допомогою яких проходять виборка даних з рядків таблиці, не залежить від того впорядковані рядки чи ні.

**Частина 2 : ШВИДКІСТЬ РОБОТИ SQL SERVER 7.0**

В процесі роззвитку SQL Server при розробці клієнтських додатків використовувалось декілька підходів. Ядром системи завжди був процессор обробки запитів, який працював на сервері. Запити писались на мові SQL вводились і опрацьовувались через утіліти типу Query Analyzer.

При написанні програмних додатків для взаємодії з SQL Server частіше за усе використовують :

1. DB-Library
2. ODBC і DAO

DB-Library являє собою специфічний для SQL Server інтерфейс прикладних програм . Він є менш зручний ніж ODBC, для переносу програмного додатку в іншу серверну систему але з точки зору працездатності він реалізовує самий швидкий доступ до інформації.

Це забезпечується не тільки тим, що він надає оптимізований інтерфейс API, але використовує в програмах особисту мову системи SQL Server. ODBC (як і SQL-OLE ) надають такий же рівень сервісу але створюють додатковий рівень абстракції між додатком і та процессором запитів SQL Server .

В DB-Lib не підтримується зв”язок таблиць або автоматичного оновлення значень. Це можна розглядати як позитив оскільки гарантується повний контроль над взаємодією програмного додатку з сервером і оптимізацією зв”язків. Додатково програмісту додається повний контроль над виявденням і обробкою помилок, пов”язаних з транзакціями.

2.1  ***ТЕХНОЛОГІЯ DAO***

(Data Access Object – об”єкти доступу до данних) приззначена для викоритання об”єктів, методів, властивотей, значно полегшшують роботу програмного додатку з базою даних. Для обміну інформацією з SQL сервером в об”єктах DAO викоритову-ються рівні дотупу Jet і ODBC, одна вони утворюють ще один рівень абстракції між додатками і викликами функцій ODBC, які використовуються при обробці запитів.

Технологія DAO для роботи з базами даних, таблицями, видами передбачає викоритання коллекцій об”єктів. Наприклад для створення нової таблиці легше викликати метод Add відповідного об”єкта, між використовувати стандартний підхід технології ODBC.

При роботі з базою даних можна використати стандартний підхід для доступу до майже будь-якого сховища даних, що підтриимується засобоми ODBC.

Але треба відмітити, що технологія DAO поступово витісняється більш прогресивною технологією ADO(ActiveX Data Object – об”єкти данх ActiveX ) .

#### 2.2 ТЕХНОЛОГІЯ ODBC

З метою можливого встановлення зв”язку з різними СУБД фірмою Microsoft була розроблена технологія ODBC . Вона забезпечує створення додаткового рівня абстракції між програмним додатком і СУБД. Даний рівень дозволяє створити одну команду Select і використовувати її для виборки даних з будь-яких підтримуємих типів баз даних, враховуючи ті випадки, коли СУБД взагалі не підтримує мову SQL.

Служби ODBC являють собою окремий рівень доступу до файлів баз даних. ODBC бере на себе відповідальністьза отримання від до-датків запитів на вибірку інформації і переведення їх на мову, що використовує ядро база даних, для обробки запиту. Гарним прикладом використання ODBC технології є СУБД Access фірми Microsoft. Головна різниця між ODBC і BD-Lib полягає у тому, що ODBC вимагає побудови і передачі йому лише стандартних SQL команд, а BD-Lib вимагає особливого синтаксису при побудові у буфері команд, що напрравляються безпосередньо у серверне середовище.

Головним недоліком ODBC є необхідність в трансляції запитів. Проведення цієї додаткової роботи відповідно відображається на швидкості доступу до даних. При роботі з ODBC можна отримати суттєвий виграш у швидкості обробки даних, якщо в системі клі-єнт/сервер саме на сервері системи SQL Server буде оброблятись запит, а не на клієнтському комп”ютері. Також приріст у швидкості роботи ODBC дає використання встроєнних процедур.

#### 2.3 JSCRIPT

Тепер перейдемо до процесів, що пов”язані з вимірюванням швидкості системи SQL Server 7.0 . У данній роботі прпонується написання програмного додатку на встроєнному shell OS Windows NT - Java Script . Сам програмний додаток це програма написана для інтерпритуюча мова. Файл з розширенням . js буде виконавчим для OS Windows NT. Спочатку напишемо Script для створення бази даних :

// ADO Flags

var adLockOptimistic = 3;

var adOpenStatic = 3;

var SQL\_DataBase = "DSN=OTSt;UID=sa;PWD=;DATABASE=OTSt"

//var SQL\_DataBase = "DSN=OTStt;UID=;PWD="

var oConn;

var oRs;

function db\_Connect(){

// Create ADO Connection Object. Use IISSDK OBDC Souce with

// default sa account and no password

oConn = new ActiveXObject("ADODB.Connection");

oConn.Open(SQL\_DataBase);

oConn.CommandTimeout=600;

}

function db\_Record(SQL){

var oRs;

// Create ADO Recordset Component, and associate it with ADO

connection

oRs = new ActiveXObject("ADODB.Recordset");

oRs.ActiveConnection = oConn;

// Get empty recordset

oRs.Source = SQL;

oRs.CursorType = adOpenStatic; // use a cursor

other than Forward Only

oRs.LockType = adLockOptimistic; // use a locktype

permitting insertions

oRs.Open();

return oRs;

}

function db\_Close(){

oConn.Close();

}

function db\_Requery(Rec,SQL){

Rec.Close();

Rec.Source = SQL;

Rec.Open();

}

db\_Connect();

try {

oRs=db\_Record("CREATE TABLE dbo.DBTEST ( String char (30) NULL, Number int NOT NULL DEFAULT (0), Random int NOT NULL DEFAULT (0))");

} catch (e) {

}

oRs=db\_Record("SELECT \* FROM DBTEST WHERE 0=1");

for(i=0;i<100000;i++){

oRs.Addnew();

oRs("String").value="User"+i;

oRs("Number").value=i;

oRs("Random").value=Math.round(100\*Math.random());

oRs.Update();

if(i%10000==0)

WScript.echo("Now in base present "+i+" records");

}

WScript.echo("Completed");

База даних буде мати вид :

|  |  |  |
| --- | --- | --- |
| String | Number | Random |
| User1 | 1 | 34 |
| User2 | 2 | 12 |
| User3 | 3 | 75 |
| . . . . | . . . . | . . . . |

У наведеному Script функція Connect відповідає за регістрацію в SQL Server і утворення бази даних DBTest .

DSN – ім”я зв”язку, визначається при утворенні джерела даних ODBC.

UID – ідентифікатор користувача, використовується при регістрації на сервері, у SQL Server ідентифікатор користувача завжди має починатись з “ sа ” .

PWD – або “ password ” кодове слово доступу , необхідний для регістрації на сервері. В даному випадку не використовуємо.

Функція Record проводить запис у всі рядки бази даних DBTest створені значення , у стовбчик Random – створені генератором випадкових чисел значення від 0 до 99 .

Кількість рядків у таблиці буде 10 000, створення займає 3,5 хвилин. Стовбчик Number буде ключем.

Напишемо ще один Script , для тестування швидкості роботи SQL Server . Щоб визначети час відповіді на запит як умога точніше будемо посилати на сервер не один запит , а декілька десятків однакових запитів, обрахувавши середній час відповіді отримаємо точний час обробки запиту і знаходження сервером результуючої множини елементів. Але треба враховувати такий випадок - якщо для підвищення точності послати декілька сотень однакових запитів, то SQL Server почне хешування таблиці для оптимізації своєї роботи

і результати роботи будуть не точні, оскільки при звичайному запиті хешування таблиці не робиться.

// ADO Flags

var adLockOptimistic = 3;

var adOpenStatic = 3;

var SQL\_DataBase = "DSN=OTSt;UID=sa;PWD=;DATABASE=OTSt"

//var SQL\_DataBase = "DSN=OTStt;UID=;PWD="

var oConn;

var oRs;

function db\_Connect(){

// Create ADO Connection Object. Use IISSDK OBDC Souce with

// default sa account and no password

oConn = new ActiveXObject("ADODB.Connection");

oConn.Open(SQL\_DataBase);

oConn.CommandTimeout=600;

}

function db\_Record(SQL){

var oRs;

// Create ADO Recordset Component, and associate it with ADO connection

oRs = new ActiveXObject("ADODB.Recordset");

oRs.ActiveConnection = oConn;

// Get empty recordset

oRs.Source = SQL;

oRs.CursorType = adOpenStatic; // use a cursor other than Forward Only

oRs.LockType = adLockOptimistic; // use a locktype permitting insertions

oRs.Open();

return oRs;

}

function db\_Close(){

oConn.Close();

}

function db\_Requery(Rec,SQL){

Rec.Close();

Rec.Source = SQL;

Rec.Open();

}

function Test(TestNumber, Query, CountTests){

Max=0;

Min=1000000;

Delta=0;

Ave=0;

WScript.echo("Query #"+TestNumber);

for(i=0;i<CountTests;i++){

StartTime=new Date();

db\_Requery(oRs,Query);

EndTime=new Date();

Delta=(EndTime-StartTime);

WScript.echo(" Probe #"+i+" Result="+Delta+" ms");

Ave=(Ave\*i+Delta)/(i+1);

}

WScript.echo(" Result="+Ave+" ms");

}

db\_Connect();

oRs=db\_Record("SELECT \* FROM DBTEST WHERE 1=0");

Test(1, " SELECT \* FROM DBTEST " , 50 );

Test(2 , " SELECT \* FROM DBTEST ORDER BY String",50);

Test( 3 ," SELECT \* FROM DBTEST ORDER BY Number ",50);

Test (4,"SELECT Sum(Number) FROM DBTEST GROUP BY String",50);

Test(5 , " SELECT Sum(Number) FROM DBTEST GROUP BY

Random",50);

Test( 6 , " SELECT \* FROM DBTEST WHERE Number = 99999 " , 50);

Test(7 ," SELECT \* FROM DBTEST WHERE String = 'User99999'",50);

Test(8,"SELECT \* FROM DBTEST WHERE Number in (SELECT

Number FROM DBTEST ) " , 50 );

Test(9 ,"SELECT \* FROM DBTEST WHERE Number in (SELECT

Number FROM DBTEST WHERE String < ' User50000 ' ) ",50);

Test(10 , " SELECT \* FROM DBTEST WHERE Number in (SELECT

Number FROM DBTEST WHERE String='User50000')", 50);

Test(11, " SELECT \* FROM DBTEST WHERE Number in (SELECT

Number FROM DBTEST WHERE String > ' User10000 ' ) " , 50);

Test(12 , "INSERT INTO DBTEST (String,Number)

VALUES('User30',30) " , 50 );

Test(13 , "DELETE FROM DBTEST (String,Number)

VALUES('User30',30) " , 50);

Test(14 ," DELETE FROM DBTEST " , 1);

db\_Close()

Головна функція Test у циклі 50 разів посилає запит на сервер, обраховує різницю між початком обробки запиту і часом закінчення обробки :

StartTime=new Date();

« обробка запиту »

EndTime=new Date();

Delta=(EndTime - StartTime);

У останньому script маємо 14 тестів для тестування різних можли-востей SQL Server. Розглянемо їх докладніше .

Тести 12 і 13 характеризують швидкість вставки і вилучення елементів у базі даних.

Тест 14 знищує усю таблицю.

Запити на вибірку інформації :

Тест 1:" SELECT \* FROM DBTEST " - виділити всі елементи ;

Тест 2: " SELECT \* FROM DBTEST ORDER BY String" -відсортувати усі елементи за значенням у стовбчику “String”;

Тест 3:" SELECT \* FROM DBTEST ORDER BY Number"- відсортува-ти усі елементи за значенням у стовбчику “ Number ”;

Тест 4:"SELECT Sum(Number) FROM DBTEST GROUP BY String"

- скласти усі елементи стовбчика “ Number ” за значенням у стовбчи-ку “String”;

Тест 5:" SELECT Sum(Number) FROM DBTEST GROUP BY

Random" - скласти усі елементи стовбчика “ Number ” за значенням у стовбчику “ Random ”;

Тест 6:" SELECT \* FROM DBTEST WHERE Number = 99999 " – визначити усі елементи, у яких значення у стовбчику “ Number ” = 99999;

Тест 7:" SELECT \* FROM DBTEST WHERE String = 'User99999'" - визначити усі елементи, у яких значення у стовбчику “String “ = 'User99999' ;

Тест 8 : "SELECT \* FROM DBTEST WHERE Number in (SELECT Number FROM DBTEST ) " - визначити усі елементи з множини чисел ;

Тест 9 :"SELECT \* FROM DBTEST WHERE Number in (SELECT

Number FROM DBTEST WHERE String < ' User50000 ' ) " - визначити усі елементи з множини елементів, у яких значення у стовбчику “String “ < ' User50000 ';

Тест 10 : " SELECT \* FROM DBTEST WHERE Number in (SELECT Number FROM DBTEST WHERE String='User50000')" - визначити усі елементи з множини елементів, у яких значення у стовбчику “String “ = ' User50000';

Тест 11 : " SELECT \* FROM DBTEST WHERE Number in (SELECT Number FROM DBTEST WHERE String > ' User10000 ' ) "- визначити усі елементи з множини елементів, у яких значення у стовбчику “String “ > ' User10000' ;

**Частина** 3 **: *РЕЗУЛЬТАТИ ТЕСТУВАННЯ***

Для наведення прикладу універсальності при використанні написан-ного script при вимірюванні швидкодії, виміряємо швидкість обробки запитів у системі SQL Server 7.0 та Microsoft Access 97.

Щоб script сприймався системою Access без суперечень треба змінити параметри ідентифікації :

//var SQL\_DataBase = "DSN=OTSt;UID=sa;PWD=;DATABASE=OTSt"

var SQL\_DataBase = "DSN=OTStt;UID=;PWD="

Отримані результати приведемо у таблиці :

|  |  |  |
| --- | --- | --- |
| Тест № | SQL Server 7.0 | Access 97 |
| 1 | 2,673 | 1,81 |
| 2 | 7,781 | 5,603 |
| 3 | 7,356 | 5,211 |
| 4 | 8,239 | 7,54 |
| 5 | 0,903 | 0,494 |
| 6 | 0,345 | 0,231 |
| 7 | 0,390 | 0,287 |
| 8 | 11,207 | 7,11 |
| 9 | 3,843 | 2,53 |
| 10 | 0,655 | 0,507 |
| 11 | 11,469 | 9,816 |
| 12 | 0,3 | 0,285 |
| 13 | 0,31 | 0,22 |
| 14 | 55,62 | 34,324 |

Як видно з результатів проведенного вимірювання запити на локальній базі даних Access опрацьовуються значно швидше ніж на сервері.

Аналізуючи запити 2 і 3, 4 і 5, 6 і 7 бачимо, що обробка елементів різних типів у однакових запитах має різний час.

Так наприклад сортувати усіх елементів за значенням у стовбчику “String” опрацьовується повільніше ніж у стовбчику " Number"- причина у розміру елементів цих типів.

Теж саме при обробці запитів 4 і 5 при сумуванні елементів стовбчика " Number" за елементами стовбчиків “ String ” і ” Random”.

Стовбчик “String ” зберігає значення типу String і стовбчик ”Random”

зберігає значення типу Integer, елементи типів мають різний розмір і обробляються з різною швидкістю.

Розглядаючи швидкість обробки множин елментів зазначимо, що чии більше елементів обробляє сервер у базі даних тим більше часу на це йде. Так наприклад запит 10 на пошук у базі даних елемента типу String - 'User50000' ззаймає набагато менше часу ніж пошук елементів, що менше 'User50000'. Таких елементів приблизно 50 тисяч і їх пошук займає досить довгий час. Пошук елементів, що більше 'User10000' займає у сервера ще більший час.

Запити 12,13 визначають час занесення до бази нового рядка і вилучення рядка з бази даних.

Проводячи наведені дослідження за роботою сервера ми отримуємо повну характеристику його роботи.

Використаємо наведені вище програмні додатки і проведемо виміри на інших SQL-серверах. Порівняємо швидкості роботи SQL Server 7.0 і Oracle 8 :

|  |  |  |
| --- | --- | --- |
| Тест № | SQL Server 7.0 | Oracle 8 |
| 1 | 2,673 | 2,71 |
| 2 | 7,781 | 8,06 |
| 3 | 7,356 | 8,09 |
| 4 | 8,239 | 8,75 |
| 5 | 0,903 | 1,041 |
| 6 | 0,345 | 0,4 |
| 7 | 0,390 | 0,432 |
| 8 | 11,207 | 11,917 |
| 9 | 3,843 | 4,05 |
| 10 | 0,655 | 0,78 |
| 11 | 11,469 | 12,021 |
| 12 | 0,3 | 0,295 |
| 13 | 0,31 | 0,291 |
| 14 | 55,62 | 43,134 |

Як видно з результатів SQL Server 7.0 працює швидше за Oracle 8 при обробці вибірок, а при вставці і вилученні елементів у Oracle краші результати. Це можна пояснити різною будовою ядер обробки запитів.

Проведемо такіж заміри і для SQL сервера Informix.

|  |  |  |
| --- | --- | --- |
| Тест № | SQL Server 7.0 | Informix |
| 1 | 2,673 | 2,11 |
| 2 | 7,781 | 6,932 |
| 3 | 7,356 | 7,138 |
| 4 | 8,239 | 7,934 |
| 5 | 0,903 | 0,856 |
| 6 | 0,345 | 0,456 |
| 7 | 0,390 | 0,471 |
| 8 | 11,207 | 13,04 |
| 9 | 3,843 | 4,41 |
| 10 | 0,655 | 0,748 |
| 11 | 11,469 | 12,814 |
| 12 | 0,3 | 0,315 |
| 13 | 0,31 | 0,31 |
| 14 | 55,62 | 51,48 |

По результам видно, що SQL Server 7.0 працює повільніше за Informix при обробці простих запитів, а при пошуку і порівнянні елементів працює швидше. На запитах по вставці і вилученні елементів у Informix і SQL Server 7.0 майже однакові результати.

Таким чином ми досягли мети данної роботи - побудували програмні додатки, якими змогли визначити характеристики працездатності різних серверних баз даних.

#### ВИСНОВОК

В результаті проведенної роботи було розглянуто різні фактори, що впливають на швидкість роботи системи SQL Server 7.0.

Були написані дві програми, що дають змогу вимірювати швид-кість обробки запитів багаторівневою серверною системою Microsoft SQL Server 7.0 і тим самим дають можливість досліджувати ефективність настройки параметрів системи, дають змогу оцінити оптимальність побудови бази даних, оптимальність використання індексів, правил, обмежень, ефективність роботи з транзакціями.

Були розглянуті сильні і слабкі боки технологій роботи з програмними додатками - DB-Library та ODBC з DAO.

Як приклад роботи програмних додатків отримані дані про швидкодію систем баз даних SQL Server 7.0 та Microsoft Access 97.

Порівняні результати застосування програмних додатків у системах SQL-серверів таких як Oracle та Informix з швидкістю роботи Microsoft SQL Server 7.0.

Використовуючи написані script-и програмісти і розробники систем баз даних типу клієнт/сервер можуть визначати характеристики створених ними систем з ціллю аналізу і удосконалення.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ :

1. “Використання Microsoft SQL Server 7.0” Стефан Вінкоп

вид. Москва 1999 р.

1. “ Teach Yourself Oracle 8 In 21 Days” Cary N. Prague
2. “ Секреты Access 97” Уильям Амо изд.Диалектика 1997р.
3. “ Технологія Java” Джо Вебер вид. “BHV” 1996 р.
4. Журнал “PC World” № 3,7,8 за 97 р., № 1,9 за 98 р.
5. www.techrepublic.com