# Database Management Systems Project 2: SimpleDB

# Team 7

# Akond Rahman (aarahman)

# Saurav Shekhar(sshekha3)

# Mithilesh Jaywant Gawande (mgawand)

# Anuja Chandrashekhar Yawalkar(ayawalk)

Contents

[Java classes that are modified 2](#_Toc417746802)

[Changes made in the page.java class 2](#_Toc417746803)

[Graph and Analysis 6](#_Toc417746804)

[Use BasicBufferMgr with linear search strategy. 6](#_Toc417746805)

[Run 1000 random select statements from the client. Note down the time taken. 6](#_Toc417746806)

[Repeat this 3 times and note down the average value. 6](#_Toc417746807)

[Use BasicBufferMgr with hashed search strategy. 6](#_Toc417746808)

[Repeat b and c. 6](#_Toc417746809)

[Plot the graph. 6](#_Toc417746810)

[Testing task 2: 6](#_Toc417746811)

# Java classes that are modified

* CHANGES IN THE Buffer manager .java class

1. HANDLING NEW DATA TYPES

Findmajor.java ( select stmt dept name – run 1000 times run in for loop)

## Changes made in the page.java class

|  |  |
| --- | --- |
| * The following String was added to take the date input in string format and then store it in the form of Date Object.      |  | | --- | | String theFormat="yyyyMMdd";  SimpleDateFormat dateFormatObj = new SimpleDateFormat(theFormat); | |

* The following functions were added to set Date and get Date .

|  |
| --- |
|  |
|  |
| public synchronized void setDate(int offset, Date val) |
| { |
| contents.position(offset); |
| String dateStr = convertDateToStr(val) ; |
| byte[] byteval = dateStr.getBytes(); |
| contents.putInt(byteval.length); |
| contents.put(byteval); |
| } |
|  |
| public synchronized Date getDate(int offset) |
| { |
| contents.position(offset); |
| int len = contents.getInt(); |
| byte[] byteval = new byte[len]; |
| contents.get(byteval); |
| String dateStringToSend = new String(byteval); |
| Date returnDateParam = convertStringToDate(dateStringToSend); |
| return returnDateParam ; |
| } |

* The following functions were added to set Byte and get byte:-

|  |
| --- |
| { |
| contents.position(offset); |
|  |
| byte[] byteval = convertShrotValToByteArray(shortParam); |
| contents.putInt(byteval.length); |
| contents.put(byteval); |
| } |
|  |
| public synchronized short getShort(int offset) |
| { |
| Short shortValToret ; |
| contents.position(offset); |
| int len = contents.getInt(); |
| byte[] byteval = new byte[len]; |
| contents.get(byteval); |
| shortValToret = convertByteArrayToShort(byteval); |
| return shortValToret ; |
| }   * The following classes were added to handle Boolean Data Types:- |
| public synchronized void setBoolean(int offset, boolean boolParam) |
| { |
| String strToStore=""; |
| contents.position(offset); |
| if(boolParam) |
| { |
| strToStore = "true" ; |
| } |
| else |
| { |
| strToStore = "false" ; |
| } |
| byte[] byteval = strToStore.getBytes(); |
| contents.putInt(byteval.length); |
| contents.put(byteval); |
| } |
|  |
| public synchronized boolean getBoolean(int offset) |
| { |
| boolean boolToret; |
| contents.position(offset); |
| int len = contents.getInt(); |
| byte[] byteval = new byte[len]; |
| contents.get(byteval); |
| String strToUse= new String(byteval); |
| if(strToUse.equals("true")) |
| { |
| boolToret = true ; |
| } |
| else |
| { |
| boolToret = false; |
| } |
|  |
| return boolToret ; |
|  |
| } |
|  |
| * The following methods were added to handle Short Data Type:- |
| public synchronized void setShort(int offset, short shortParam) |
| { |
| contents.position(offset); |
|  |
| byte[] byteval = convertShrotValToByteArray(shortParam); |
| contents.putInt(byteval.length); |
| contents.put(byteval); |
| } |
|  |
| public synchronized short getShort(int offset) |
| { |
| Short shortValToret ; |
| contents.position(offset); |
| int len = contents.getInt(); |
| byte[] byteval = new byte[len]; |
| contents.get(byteval); |
| shortValToret = convertByteArrayToShort(byteval); |
| return shortValToret ; |
| } |
| The following methods were added to convert the Short Data Type to Byte Data Type:- |
| private byte[] convertShrotValToByteArray(short value) { |
|  |
| int byteLen = 2 ; |
| byte[] bytesToret = new byte[byteLen]; |
| ByteBuffer bufferInUse = ByteBuffer.allocate(bytesToret.length); |
| bufferInUse.putShort(value); |
| return bufferInUse.array(); |
| } |
|  |
| * The following methods were added to convert the Byte array to the Short Array. |
| private short convertByteArrayToShort(byte[] array) |
| { |
| ByteBuffer bufferToRet = ByteBuffer.wrap(array); |
| return bufferToRet.getShort(); |
| } |
|  |
| * The following methods were added to convert the Date to string data type:- |
| private String convertDateToStr(Date dateParam) |
| { |
|  |
| String strToret ="" ; |
| strToret = dateFormatObj.format(dateParam); |
| return strToret ; |
| } |
| * The following methods were added to convert the Date to string data type:- |
| private Date convertStringToDate(String strParam) |
| { |
| Date dateObjToRet = new Date(); |
|  |
| try |
| { |
| dateObjToRet = dateFormatObj.parse(strParam); |
|  |
| } |
| catch (ParseException e) |
| { |
| System.err.println("Error log " + e.getMessage()); |
| e.printStackTrace(); |
| } |
| return dateObjToRet ; |
|  |
| } |
|  |
| } |

# Graph and Analysis

- Draw a graph between buffer sizes and time taken to run 1000 random select statements.

## Use BasicBufferMgr with linear search strategy.

## Run 1000 random select statements from the client. Note down the time taken.

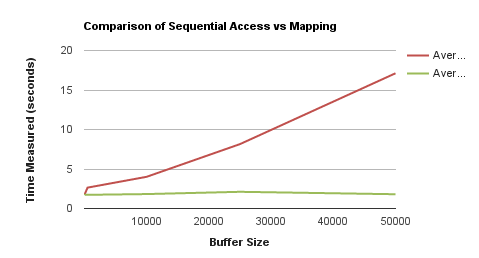
## Repeat this 3 times and note down the average value.

## Use BasicBufferMgr with hashed search strategy.

## Repeat b and c.

## Plot the graph.

|  |  |  |
| --- | --- | --- |
| Buffer Sizes (Bytes) | Time in Seconds (Normal) | Time in Seconds (Normal) |
| 8 | 1.721 | 1.716 |
| 16 | 1.715 | 1.704 |
| 50 | 1.716 | 1.762 |
| 500 | 1.888 | 1.767 |
| 10000 | 2.656 | 2.596 |
| 25000 | 3.962 | 3.994 |
| 50000 | 8.078 | 8.161 |
| 100000 | 16.932 | 17.66 |
|  |  |  |
| Time in Seconds (HashMap) | Time in Seconds (HashMap) | Time in Seconds (HashMap) |
| 1.705 | 1.679 | 1.711 |
| 1.708 | 1.699 | 1.729 |
| 1.712 | 1.834 | 1.65 |
| 1.733 | 1.762 | 1.758 |
| 1.728 | 1.745 | 1.723 |
| 1.819 | 1.81 | 1.852 |
| 2.097 | 2.145 | 2.097 |
| 1.739 | 1.908 | 1.77 |
|  |  |  |
|  | Average Time (Sequential) | Average Time (HashMap) |
| 8 | 1.717666667 | 1.698333333 |
| 16 | 1.714 | 1.712 |
| 50 | 1.733666667 | 1.732 |
| 500 | 1.818666667 | 1.751 |
| 10000 | 2.642 | 1.732 |
| 25000 | 4.012 | 1.827 |
| 50000 | 8.15 | 2.113 |
| 100000 | 17.11666667 | 1.805666667 |



- buffer sizes : 8,16,50,500,10000,25000,50000,100000.

- Analysis of graph

Populate db tables with random values to perform the analysis.

# Testing task 2:

The following code was used to pass values and test the getter and setter methods.

First we created a new file called MyFileTester. We included the following methods to send data of various data types to the page file at the desired memory locations and retrieved it

public class MyFileTester { public static void main(String[] args) {….}

Then we create a Database testingDB and initialized the offsets using the following code.

|  |
| --- |
| String dirToCreateDB="testingDB"; SimpleDB.init(dirToCreateDB); int intOffset=99;  int strOffset=20;  int dateOffset=1;  int byteAOffset =2 ;  int boolOffset = 3 ;  int shortOffset = 4 ; |

We then created New pages to send data from this class to page class and retrieve back.

|  |
| --- |
| Page tempPage1 = new Page();  Block tempBlock = new Block("MyTempBlock", 6); |

* Testing the string data handling:-

String strName = new String("Anuja Chandrashekhar Yawalkar");

tempPage1.setString(strOffset, strName);

String stringToRet = tempPage1.getString(strOffset);

System.out.println("String to return ... " + stringToRet );

* Testing the Int Data Handling:-

tempPage1.setInt(intOffset, 99);

int n = tempPage1.getInt(intOffset);

System.out.println("Value of n=" + n);

tempPage1.setInt( n+1, 100);

tempPage1.write(tempBlock);

int newN = tempPage1.getInt(n+1);

System.out.println("Value @ 'n+1'=" + newN);

* Testing the Date Data Handling:-

String dateStr="19870819";

String theFormat="yyyyMMdd";

DateFormat formatter = new SimpleDateFormat(theFormat);

Date testDateObj = new Date();

try {testDateObj = (Date) formatter.parse(dateStr); }

catch (ParseException e)

{e.printStackTrace();}

tempPage1.setDate(dateOffset, testDateObj);

System.out.println("The date was: " + tempPage1.getDate(dateOffset));

//byte[] byteArrayExample = new byte[]{3,2,5,4,1};

byte [] byteArrayExample = "Akond".getBytes();

String strToRet="" ;

tempPage1.setBytArray(byteAOffset, byteArrayExample);

strToRet = new String(tempPage1.getByteArray(byteAOffset));

System.out.println("Byte array cotents as string=" + strToRet );

* Testing the Boolean Data Handling:-

boolean boolToInsert = true ;

System.out.println("bools=" + boolToInsert);

tempPage1.setBoolean(boolOffset, boolToInsert);

System.out.println("boolean returned = " + tempPage1.getBoolean(boolOffset));

* Testing the shory Data Handling

short shortValToRet = 21474 ;

tempPage1.setShort(shortOffset, shortValToRet);

System.out.println("short value to return = " + tempPage1.getShort(shortOffset));}}

* When we put the initial buffer oddest for data type as 397 or above we get the following Exceptions:-

Exception in thread "main" java.nio.BufferOverflowException

at java.nio.Buffer.nextPutIndex(Unknown Source)

at java.nio.DirectByteBuffer.putInt(Unknown Source)

at simpledb.file.Page.setInt(Page.java:146)

at simpledb.file.MyFileTester.main(MyFileTester.java:54)