

# Simple Tests for the V Storage Manager

Prof. Ghandeharizadeh  
585 Term Project

## Overview

This document describes simple tests to help with debugging of your implementation of the V storage manager. Below, we provide a conceptual description of each test. You are responsible for the implementation of these tests. The objective is to provide you with the highest degree of flexibility so that you can prototype V and implement the test routines in an incremental manner.

## Deliverables:

The main deliverable is a zip file containing your implementation of the V storage manager and the test suites. A link from the DEN web site enables you to upload your zip file.

Due date: Lunch time April 9<sup>th</sup>

## Test suites:

There are a total of 6 tests. The objective of each test is as follows:

Test 1: Evaluate the functionality to create data zones.

Test 2: Evaluate the functionality to insert records into both BDB-Disk and BDB-Mem.

Test 3: Analyze your caching algorithm and its behavior.

Test 4: Analyze the Delete functionality.

Test 5: Analyze the DeleteAllKeys functionality from both BDB-Disk and BDB-Mem.

Test 6: Analyze DeleteInAllDataZones method with BDB-Disk and BDB-Mem.

These tests are as follows.

## Test 1

Objective: Test the functionality to create data zones.

A. Initialize V with 512 MB of memory

B. In a for loop, repeat the following 5 times: //Note that create should return an error after the first iteration of this loop

B.1. Create a datazone named "alpha"

B.2. Create a datazone named "beta"

B.3. Create a datazone named "gamma"

B.4. Create a datazone named "omega"

C. Retrieve and print the name of the data zones

D. Verify the presence of datazones named alpha, beta, gamma, and omega

E. Shutdown V

F. Initialize V with 512 MB of memory

G. Let F equal GetNumberOfDataZones();

H. GetDataZoneNames using F;

I. Print the name of the data zones

- G. Verify the existence of datazones named alpha, beta, gamma, and omega
- K. Shutdown V

## Test 2

*Objective:* Test the functionality to insert records into both BDB-Disk and BDB-Mem.

*Assumption:* Test 1 is executed to create the datazones needed for Test 2.

- A. Initialize V with 512 MB of memory
- B. Let F equal `GetNumberOfDataZones()`;
- C. `GetDataZoneNames` using F;
- D. Print the name of the data zones
- E. Verify the presence of datazones named alpha, beta, gamma, and omega
  
- F. Insert 10,000 records of length 50 bytes into alpha. Key of each record is generated using a unique integer that varies from 0 to 9,999. Implementation hint: Use a for loop. The value is an array of 50 bytes with the first 4 bytes set to the length of the value, 50.
  
- G. Insert 5,000 records of length 100 bytes into beta, generating the keys in the same manner as those generated for alpha in Step F. The value is an array of 100 bytes with the first 4 bytes set to the length of the value, 100.
  
- H. Insert 1,000 records of length 1,000 bytes into gamma, generating the keys in the same manner as before. The value is an array of 1000 bytes with the first 4 bytes set to the length of the value, 1000.
  
- I. Insert 500 records of length 10,000 bytes into omega, generating the keys in the same manner as before. The value is an array of 10,000 bytes with the first bytes set to the length of the value, 10,000.
  
- J. In a for loop with id ranging from 1 to 10,000
  - J.1. Let `key = id % 500`,  
Note: `key` is a Vdt data type.
  - J.2. Get Value from alpha and verify an integer cast of its first four bytes produces the value 50,
  - J.3. Get Value from beta and verify an integer cast of its first four bytes produces the value 100
  - J.4. Get Value from gamma and verify an integer cast of its first four bytes produces the value 1,000
  - J.5. Get Value from omega and verify an integer cast of its first four bytes produces the value 10,000
  
- K. Shutdown V
  
- L. Initialize V with 512 MB of memory
- M. In a for loop with id ranging from 1 to 10,000

- M.1. Let  $\text{key} = \text{id} \% 500$ ,
- M.2. Get Value from alpha and verify an integer cast of its first four bytes produces the value 50
- M.3. Get Value from beta and verify an integer cast of its first four bytes produces the value 100
- M.4. Get Value from gamma and verify an integer cast of its first four bytes produces the value 1,000
- M.5. Get Value from omega and verify an integer cast of its first four bytes produces the value 10,000

N. Shutdown V

### Test 3

*Objective:* Analyze your caching algorithm and its behavior.

*Assumption:* Test 2 is run prior to Test 3 and the following datazones along with their corresponding number of records exists: alpha, beta, gamma, and omega.

- A. Initialize V with 1 MB of memory
- B. In a for loop with ids ranging from 1 to 10,000
  - B.1. Let  $\text{key} = \text{id} \% 500$ ,  
Note: key is a Vdt data type
  - B.2. Get Value from alpha and verify an integer cast of its first four bytes produces the value 50
  - B.3. Get Value from beta and verify an integer cast of its first four bytes produces the value 100
  - B.4. Get Value from gamma and verify an integer cast of its first four bytes produces the value 1,000
  - B.5. Get Value from omega and verify an integer cast of its first four bytes produces the value 10,000
- C. Print the number of unique objects in your cache.
- D. Verify if this number matches your design. For example, if your design packs small objects into the cache then many more objects of alpha should be cache resident as compared to the others.
- E. Shutdown V

### Test 4

*Objective:* Analyze the Delete functionality.

*Assumption 1:* Test 3 is run prior to Test 4 and the following datazones along with their corresponding number of records exists: alpha, beta, gamma, and omega.

*Assumption 2:* Existence of the following method to count the number of records for each database:

```

CountAndPrintRecords(int MaxValue, int ModValue)
{
    A. alphaCounter = betaCounter = gammaCounter = OmegaCounter = zero

    B. In a for loop with ids ranging from 1 to MaxValue
        B.1. Let key = id % ModValue
            Note: key is a Vdt data type
        B.2. Get Value from alpha and, if found, verify an integer cast of its first
four bytes produces the value 50
        B.3. If Value is found alphaCounter++
        B.4. Get Value from beta and, if found, verify an integer cast of its first
four bytes produces the value 100
        B.5. If Value is found betaCounter++
        B.6. Get Value from gamma and, if found, verify an integer cast of its
first four bytes produces the value 1,000
        B.7. If Value is found gammaCounter++
        B.8. Get Value from omega and, if found, verify an integer cast of its first
four bytes produces the value 10,000
        B.9. If Value is found omegaCounter++

    C. Print the values of alphaCounter, betaCounter, gammaCounter, and
omegaCounter
}

```

***Test 4 starts here***

- A. Initialize V with 512 MB of memory
- B. CountAndPrintRecords(10000, 500)
- C. In a for loop with ids ranging from 1 to 10,000
  - C.1. Let Key = id % 500
    - Note: Key should be a Vdt data instance.
  - C.2. Delete Key from alpha
  - C.3. Delete Key from beta
  - C.4. Delete Key from gamma
  - C.5. Delete Key from Omega
- D. CountAndPrintRecords(10000, 500)
- E. Shutdown V

**Test 5**

*Objective:* Analyze the DeleteAllKeys functionality from BDB-Disk.

*Assumption 1:* Test 3 is run prior to Test 5 and the following datazones along with their corresponding number of records exists: alpha, beta, gamma, and omega.

*Assumption 2:* Existence of CountAndPrintRecords method from Test 4.

- A. Initialize V with 512 MB of memory
- B. Invoke CountAndPrintRecords(10000, 10000)
- C. DeleteAllKeys from alpha datazone
- D. Invoke CountAndPrintRecords(10000, 10000)
- E. DeleteAllKeys from beta datazone
- F. Invoke CountAndPrintRecords(10000, 10000)
- G. DeleteAllKeys from gamma datazone
- H. Invoke CountAndPrintRecords(10000, 10000)
- I. DeleteAllKeys from omega datazone
- J. Invoke CountAndPrintRecords(10000, 10000)
- K. Shutdown V

## **Test 6**

*Objective:* Analyze DeleteInAllDataZones method with BDB-Disk and BDB-Mem

*Assumption 1:* Test 3 is run prior to Test 6 and the following datazones along with their corresponding number of records exists: alpha, beta, gamma, and omega.

*Assumption 2:* Existence of CountAndPrintRecords method from Test 4.

**Assumption 3:** Existence of GenVdtKeysArray

GenVdtValuesArray(MaxKeys, ModValue){

    A. For all ids ranging from 0 to MaxKeys

        A.1. Generate an array consisting of keys whose  $(id \% \text{ModValue} == 0)$

}

### ***Test 6 begins here***

- A. Initialize V with 512 MB of memory
- B. Let  $A = \text{GenVdtValuesArray}(10000, 50)$
- C. Invoke CountAndPrintRecords(10000, 10000)
- D. Invoke DeleteInAllDataZones with A and the number of elements in A
- E. Invoke CountAndPrintRecords(10000, 10000)
- F. Shutdown V