**NoSQL Database Description**

Yilong Shu, Todd McCutchen, Wenhua Hu

*Department of Computer Science, the University of Alabama*

## **1. Database description**

In our database, we used the hashmap to store the data structure. All the inserted data should be in <fields, value> pair, and for each record row, the users can store as many fields as they want, and the value type is a String.

For the process of data storing in our database, we designed three spaces to store the different data that will be used in the query process. The three spaces are: workspace, searchspace, and datapage. Workspace is used to store the modified data, and the users can use the commit command to store all the data are in the workspace to disk. Meanwhile, the workspace also stores results fetched from datapage. Searchspace is used to store the filtered results picked form datapage fetches. Datapages are portions of the database from separate file partitions, and the users can only read the data in the datapage.

For each field you create an index for, there is a hashmap where the key is the value of the field, and the value in the hashmap is a TreeSet of record keys. Any quey on the database where the user provides matching criteria will first check the existing indexes. If matches are found in the indexes, then the database will have fewer record keys to fetch from the data pages, and will result in significantly less file I/O.

For the logging, any transaction in this database will be logged in the logs file. In our database, we designed five types logging transactions, there are insert, insertbykey, deletebykeys, deletebefieldsvalues, updatebykeys. All these transactions will have a record for the old-records and new-records.

For the version, all the committed files, including their idx and indexes will be copied into the versions files, it will be used for the rollback.

For the rollback, our database allows users to create checkpoint after each commit. Then the users can rollback to any checkpoint that have been created and the related version will be copied form the backup and be used for users as the current file.

Especially, in our database, we designed each page has at most 50 record. At the moment it is hard coded to a single variable and can be changed to any value before you recompile. If the inserted record key is larger than 50, it will inserted into the related page that can store the key record. When the users create a file, the related index for the file will be automatically created.

## **2. Grammar and query language**

In our database, we designed the following query language for the DML operations and aggregation functions.

1. **Log in function for the user authentication.** We have designed our three group person in the DBUserFile, so we can directly to log into the database with the previous designed user name and password. While other users want to use this database, they need to first register the user name and password, and then using the resisted username and password to log into the database and use it. Each users have three times to enter their password.
2. **Creat a datafile.** When the users log into the system, then they can create a file a store their data.
3. **Use datafile.** Before the users to do some DML operation to the created file, they firstly have to use this datafile. Otherwise, they will have a warning and let the users out the system (except create and describe).
4. **Describe.** It is used to describe all the created datafile and index in the files folder.
5. **Describe filename.** It will show the detail content information in the file. It includes filename, highestKey, visible, modifiable, owner, limitedAccess, record\_num and user.
6. **Scan.** It will show all the record in the used file.
7. **Insert.** It allows users to insert as many record as they want, based on the <field, value> pair. For each inserted record, it will automatically give a key number based on the insert order.
8. **Insertbykey.** It also allows users to insert record by the given key. If the given key already exist in the system, it will not allow the users to insert this record and ask them to change the key number.
9. **Deletebykeys.** It allows users to delete record based on the given key. They can delete many record at one time or just delete one record at one time.
10. **Deletebyfieldsvalues.** It allows users to delete record based on the given field values, they can give one pair of <field, value>, or many pairs.
11. **Updatebykeys.** It allows users to update one or many records for the given key record, and update the value of fields as the given pairs.
12. **Select \*.** It allows users to select all the record information based on the given <field> or <field, value> pair.
13. **Select field.** It allows users to select the specific fields based on the given <field, value> pair.
14. **Count \*.** It allowsusers to count the total record in the file.
15. **Count field.** It allows users to count the total record based on the given <field, value> pair.
16. **Sum field.** It allows users to get the sum of fields based on the given <field, value> pair.
17. **Avg field.** It allows users to get the average of fields based on the given <field, value> pair.
18. **Min field.** It allows users to get the minimum number of fields based on the given <field, value> pair.
19. **Max field.** It allows users to get the maximum number of fields based on the given <field, value> pair.
20. **DFU username.** Itallows users to delete the username in the file.
21. **Deleterecord filename.** It allows users to delete one created file.
22. **Createindex fieldname.** It allows users to create an index for a specific field. When the users search, the related the field will just copy from the disk to the pagespace, without copy all the pages into the pagespace, which will save the time and load of the computer.
23. **Deleteindex fieldname.** It allows users to delete the created index for the file.
24. **Commit.** It allows the updated records into the files. When the users log out the system and relog in, the updated records are still in the files and can be reused.
25. **checkpoint name.** It allows the users to create a checkpoint, and automatically create a version with the checkpoint name.
26. **rollback checkpointname.** It allows the users to get the record before the given checkpoint.

## **2. User Manual**

The tutorial will tell the users how to use the database. Before using the grammar and query language, the users have to build five folders in one location, named them as files, logs, versions, src, filesAttributes, users. The files is used to store the created and committed files and the related indexes. The logs is used to store the updated transctions. The versions is used to store the checkpoint version.The fileAttributes is used to store DBSumFile, which contains all the created file information. The users is used to store the DBUserFile, which contains all the user information. The users should also put the DBUserFile under the users folder before starting.

**Note: Our database has to be run in Java 7 environment. Before using each query language, the users should make sure that they have used the file (First to use filename). Meanwhile, the users need to use javac \*.java and java –classpath . DBmain to run our database under the file path they unzip the zip file.**

1. Log in. While calling the main method, the database will ask the users to enter their user name. If the user is the first time to use this database, they will need to first register a username and password.
2. help. While typing help, it will show all the query language that the users can be used.
3. create filename; (i.e., create t1;) When creating a file, the user will be added into the file user.
4. use filename; (i.e., use t1;). If the username is existing in the file, it will allow to use the file.
5. describe. It shows all the file, idx and indexes.
6. describe filename; (i.e., describe; ). It will show all the file information.
7. scan. It will show all the record in the file. **It must be used after use filename.**
8. insert field1=value1; field2=value2; (i.e., insert name=john; addr=AL;) it will insert one record with the <field, value> pair, and the system will automatically add a key increasingly based on the previous inserted record. Meanwhile, the users can insert as many as <field, value> pair they wanted, and any kinds of fields is allowed.
9. insertbykey key1; field1=value1; field2=value2; (i.e., insertbykey 6; name=John; age=34;) it will insert one record with the <field, value> pair in the given key, if the key already exists, it will not allow to insert this record and change the key number. Meanwhile, the users can insert as many as <field, value> pair they wanted, and any kinds of fields is allowed.
10. updatebykeys key1, key2; field1=value1; field2=value2; (i.e., updatebykeys 1; n=12;). It will update the given key record with the given field values. If the typed field name is not exist in this record, it will just append it as a new <field, value> pair for this key record.
11. deletebykeys key1, key2, key3; (i.e., deletebykeys 0, 5;) It will delete the record with the given key number. The users can delete one record or many record at one time by the key number.
12. deletebyfieldsvalues field1=value1; field2=value2; (i.e., deletebyfieldsvalues n=13;) it will delete the record with the given <field, value> pair.
13. searchbykey key1;(i.e., searchbykey 11;) it will show the record with the given key number. If the given key number does not exist in the file, it will show that the key “key1” does not exist in the database.
14. select \*; field1; field2; (i.e., select \*; name;) it will show all the row record information with the given field name;
15. select \*; field1=value1; field2=value2; (select \*; name=yilong; addr=GA;) it will show all the row record information with the given <field, value> pair.
16. select field1; field2; field3=value3; field4=value4; (i.e., select name; name=yilong;) it will show the given field record information, without the whole row information with the related <field, value> pair.
17. select field1, field2; field1; (i.e., select name, addr; name;) It will show all the given fields records that have the given field name.
18. count \*; it will show the total of record in the database.
19. count field1; field1=value1; field2=value2; (i.e., count name; grade=55;) It will show the total number of record about the given field1 name with the related <field, value> pair.
20. sum field1; field1=value1; field2=value2; (i.e., sum grade; name=wenhua;) It will show the sum of record about the given field1 name with the related <field, value> pair.
21. avg field1; field1=value1; field2=value2; (i.e., avg grade; name=wenhua;) It will show the average number of record about the given field1 name with the related <field, value> pair.
22. min field1; field1=value1; field2=value2; (i.e., sum grade; name=wenhua;) It will show the minimum number of record about the given field1 name with the related <field, value> pair.
23. max field1; field1=value1; field2=value2; (i.e., sum grade; name=wenhua;) It will show the maximum number of record about the given field1 name with the related <field, value> pair.
24. dfu username; (i.e., dfu Wenhua;) It will delete the user name in the file if the given user name exists in the file, if not, it will not delete it.
25. afu username; (i.e., afu yilong;) It will ass the user name in the file if the given user name not exists in the file, if not, it will not add the username. For the added username, there will be a “,” to separate with the existing username.
26. commit. While type commit, the updated file will be saved from the current workspace to the disk. While the users log out the database and then relog in, this record still can be reused.
27. checkpoint checkpointnumber; (i.e., checkpoint cpt1;) it will create a checkpoint and be used in the rollback. Meanwhile, a checkpointnumber named version will be created and copy all the files in the files folder to the versions folder, including the files, idx and indexes.
28. rollback checkpointnumber; (i.e., rollback cpt1;) it will return the record updated before the checkpointnumber. The version checkpointnumber will be copied from the backup and can be used.
29. createindex field1; (i.e., createindex name;) It will create an index for the given field, when searching or do some query functions for the database based on the field, it will just copy the field information form disk to memory, without all the row information.
30. deleteindex field1; (i.e., deleteindex name;) It will delete the created index for the given field.
31. deleterecord filename; (i.e., deleterecord t1;) It will delete the created file totally.