James Taylor

Assignment 1

University of Maryland University College

DATA650-9041 Spring 2021

Dr. Gortcheva

jtaylor380@student.umgc.edu

**Part 1 – Questions**

1) Discuss the key components of HBase and the function of each component.

Regions are what hold the data (similar to DataNodes in HDFS). It holds a table (or part of one) of data organize by row key and column. A Region Server manages the writing and reading commands to the one or more Regions under its responsibility. This is similar to the slave role in HDFS. The HMaster is conversely the master role, which is similar to the NameNode in HDFS. It is in charge of creating and deleting tables. It also assigns Regions to Region Servers and monitors the servers. Zookeeper helps track the health of the Region Servers in case on fails. The META Table keeps track of what row keys are assigned to what Region Servers.

2) Discuss the differences between row-oriented and column-oriented databases. Provide examples of each.

Row oriented databases store data by the row. For example, a business might store Customer ID, First Name, Last Name and address together. Column oriented databases store data by the column. Using the previous example, it would store all the customer ID’s in one place, then all the first names in another place, and the last names in another and so-on. The advantages of column oriented storage are data can be accessed faster when there is lots of it and it can handle semi or unstructured data unlike row oriented.

3) When would you use Hive instead of HBase?

Hive is best utilized for querying the data for analytical purposes. It allows the querying of data in a simpler manner. HBase is better for fast random access to the data (OLTP).

4) Discuss 7 HBase shell data manipulation commands and what they do.

The Count command returns the number of rows in a specified table. The Delete command deletes the value of a cell in a table. The Deleteall command deletes the cells in a specified row. The Get command returns the contents of a cell/row. The Put command makes a specific cell a specified value. The Scan command returans a table’s data. The Truncate command can recreate, drop or disable a specified table in the database.

5) How would you implement a query that joins multiple tables in HBase?

One could scan command an entire table and then put command that data in an existing table, or create a new table and scan the tables that are going to be combined and put in the new/empty table.

6) What is a namespace in HBase?

A grouping of tables. This helps with organizing where data exists.

7) What happens when you delete table cell(s) in HBase?

It deletes the value inside of the specified cell. The row and column remain intact.

8) Discuss the approaches for storing multimedia data, including videos and images, in HBase.

HBase stores all data in byte arrays so there is lots of flexibility to store sounds, video and images together.

**Part 2 - UMGC Academic Records Case Study**

**Introduction**

The problem being addressed is getting the old data into a new HBase database. This will improve search times and allow the database to house much more data. The database needs to handle things like taking new classes, changing grades, new instructors and other common administrative changes.

The old data will be transferred into the new database. Three example records will be used and some pseudocode to demonstrate how the new database will be populated.

**Problem Statement**

The data currently exists on the old database. It will be moved to a HBase database to improve cost, speed and size of the data storage. The data will have to be transfer into the new system, which will be demonstrated by example. The database will also be optimized by storing data that is commonly queried together. This enhances the write/read performance.

The new database needs to handle changes to data. Some examples are students changing their majors or specializations, changing classes or the instructors that lead them, inputting or updating grades, new students and instructors and other common tasks.

**Design**

The design of the database’s structure can be found in the appendix. The table is being viewed in a transposed position due to the amount of columns required, so what appears to be a row is actually a column of data and the blank columns is where a row of data would be populated. The column names here would likely be abbreviated so less data needs to be saved and read.

The column families (grey cells) and columns were optimized to minimize the data needed to be read during common query tasks. The first optimized strategy was designing columns to distinguish between students and faculty. There will likely be more queries for students than faculty due to the number of students and all the classes and grades they each will have. Having a system that does not have to load and search through data that is exclusive to instructors will save time, which is why the Instructor-ID and Employee-ID are separate from the name and username of the individual. A student will have the Person column-family populated but instructors will have the Person and Faulty column-families populated. If the query is not about an instructor the Faculty column-family doesn’t need to be loaded and read, saving time.

The Program column-family is separate because it will probably not be queried often. Students probably don’t change their program and specializations often, but when they do it will be a quick query. There is only room for two specializations for now, thought more can be added if analysis of the old data shows students have more or simply add another column as needed if a student requires it for their third specialization.

The Courses column-family contain the courses and its instructors. The instructors were stored with the courses because they would likely be queried together. The option of creating a column-family for instructors is a possibility (especially if queries without instructor names are common), but it becomes another column-family to worry about. Queries to find only instructor names don’t seem likely. Creating more columns for student’s courses (course, instructor and TA) can easily be added if they take more than seven class (only seven is shown because of the example records only require that many).

Grades received their own column-family because of the amount of GPA queries that will happen. These don’t require the course name or instructor name to make the calculation. It will also make registering and searching for classes faster since Grades won’t be needlessly read.

Graduation column-family is there because it is likely not to be queried often. Because of this we don’t want it being unnecessarily loaded and read.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Row-Key | | 1 | 2 | 3 |
| Person | Student Name | Yelena Bytenskaya | Linesh Dave | Jeff Martin |
| Username | ybytensk | ldave | jmartin |
| Faculty | Instructor-ID | 234567 | 567907 |  |
| EmplID | 123456 | 567890 | 987654 |
| Program | Program | Information Technology | Data Analytics | Information Technology |
| Specialization 1 | Database Systems |  | Database Systems |
| Specialization 2 |  |  | Project Management |
| Specialization 3 |  |  | Software Engineering |
| Courses | Course 1 | DBST651 | DATA610 | DBST651 |
| Instructor 1 | James Smith | Steve Knode | James Green |
| Instructor 1.1 | Jennifer Lopez |  |  |
| TA 1 |  |  | Yelena Bytenskaya |
| Course 2 | ITEC630 | DATA620 | DBST651 |
| Instructor 2 | Jennifer Lopez | Caroline Beam | James Green |
| TA 2 |  |  | Yelena Bytenskaya |
| Course 3 | DBST667 | DATA630 | ITEC610 |
| Instructor 3 | Catharine Murphy | Bati Firdu | Brandon Mooris |
| TA 3 |  |  |  |
| Course 4 |  | DATA630 | ITEC620 |
| Instructor 4 |  | Bati Firdu | Elena Gortcheva |
| TA 4 |  |  |  |
| Course 5 |  | DATA640 |  |
| Instructor 5 |  | Steve Knode |  |
| TA 5 |  |  |  |
| Course 6 |  | DATA650 |  |
| Instructor 6 |  | Elena Gortcheva |  |
| Instructor 6.1 |  | Ozan Ozcan |  |
| Ta 6 |  |  |  |
| Course 7 |  | DATA670 |  |
| Instructor 7 |  | Jon McKeeby |  |
| Instructor 7.1 |  | Steve Knode |  |
| TA 7 |  |  |  |
| Grades | Grade 1 | A | B | F |
| Grade 2 | B | A | B |
| Grade 3 | A | C | B |
| Grade 4 |  | A | A |
| Grade 5 |  | B |  |
| Grade 6 |  | A |  |
| Grade 7 |  | B |  |
| Grade 8 |  |  |  |
| Grade 9 |  |  |  |
| Grade 10 |  |  |  |
| Graduated | Graduated | Yes | Yes | No |

**Implementation Methods**

Implementing this new Hbase database requires the data to be read and inputted into the new on with HBase code. Some pseudocode will be provided on the three example records demonstrating how this could be done.

This database is able to handle common read/write queries. These would be registering for classes, changing professors, pulling transcript data, calculating GPA and updating grades. It is also capable of handling old students that graduated and became professors.

For example, if a student is currently taking a class they will have the Course and Instructor columns populated for the specific class and the designated grade column for that course blank. Once that class is completed the system will write to the blank Grade column on that row and populate it with the student’s grade. If the student takes the same class again, it will be treated as another new course. This means if the student has taken five classes and is retaking an old class (meaning it will be his sixth class they have taken), that class will populate the designated columns for the sixth class (Course 6, Instructor 6 and Grade 6). The database will not overwrite the old grade with the new one, it will populate a new column instead. For example, imagine if Record 1’s student retook ITEC630 and received an A grade. The database will not overwrite the Grade 2 column from B to A. Instead this class will be treated as a new one with its own instructor and grade. This allows for better tracking of classes and if the retaken class has a different instructor then the first time it was taken.

The number of columns is not a fixed value in this database. If a student currently has three current specializations but wants a third, another column could be created in the Program column-family as “Specialization 4” that would hold the new data point. The same would be true for a student’s 11th class.

Students that graduated from their program can be hired as instructor as well in this database. As seen in the first record (row-key 1), the student graduated and has an instructor ID and employee ID as well as the data for their classes and grades.

Classes can be taken in any order. The database will record the correct information as long as the data is populated correctly. This means the instructor for Course 1 is in the Instructor 1 column, and so on with the second and third class. Courses also can have multiple instructors. If the course has additional instructor (example: column instructor 1.1 in row-key 1), the database is able to handle it by creating another column. If a course has two additional instructors the database would create a new column instructor 1.2 for the given example. With all the different variations of classes all the student took and will take, it’s likely lots of additional instructor columns will be created (Instructor 1.1, 1.2, 2.1, 2.2, 3.1 and so on), but for the example records provided we only needed it for courses one, six and seven across the data.

There were problems with the data and some assumptions needed to be made. For record one, one DBST651 course exits and one grade exists but two instructors listed in data. Assumed the student took a DBST651 course with two instructors. For record two, DATA630 appears twice and has two different grades but only one instructor is assigned to a DATA630 class. Assumed the student took the class twice and the same instructor taught both. Also in record two, the course DATA650 appears once and has one grade but two instructors appear to have taught the student that class. Assumed the student took the class with two instructors. Also in record two, there was one DATA670 course but two instructors for it. Assumed the student took the class with two instructors. For record three, the course DBST651 appears twice but only one instructor/TA pair are listed. Assumed the student retook the class with the same instructor/TA.

**Conclusion**

Migrating to the HBase database will allow for larger, cheaper, faster and more fault resistant storage. The column families are optimized for common searches as to not load and read unneeded data.

One place worth investigating is to see if Hive is worth it. Hive is software that operates ontop of the HBase architecture that makes for more intuitive querying. I don’t think it is the best option however because the queries won’t be done directly into the database but rather a front-end system.