SCHOOL OF BUSINESS

DEPARTMENT OF MANAGEMENT SCIENCE & TECHNOLOGY

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ACADEMIC YEAR OF 2021 – 2022

ADVANCES TOPICS IN DATA ENGINEERING

ASSIGNMENT I

KONSTANTINOS NINAS

f2822108

SUPERVISING INSTRUCTOR

THANASIS VERGOULIS

Table of Contents

Task 1 – Among Hive, Impala, and Drill, which is the one the implements more precisely the concept of data virtualization? Elaborate.

Both Drill and Impala are SQL-on-Hadoop query engines, inspired by Google's Dremel. Drill is capable of querying more data sources and file formats than Impala, even nested ones, such as JSON and Parquet files, which is a task that Impala is not capable of. Impala is capable to work only on top of the Hive metastore. Hive is a data warehouse software used to access large distributed datasets built on Hadoop/MapReduce, while Drill can process SQL queries on the whole HDFS. Additionally, Drill can link multiple data sources on-the-fly, such as other databases (e.g., MongoDB) or cloud storage databases (e.g., Amazon S3). Unlike Impala and Hive, which require a relational schema, Drill does not require any schema or any type of specification for the data to start the query execution process. Drill is capable to discover the schema of the records while it processes them. There is also no need to for Drill to keep centralized metadata. In other words, Drill does not tables/views to operate or a database administration group. The same is not true for Impala, which needs at least one table/view to be declared before querying.

Data virtualization is a technique that enables an application to retrieve, manipulate and provide summaries of the data without requiring any information about the data, such as their physical location and their format. Given the above descriptions of the capabilities of each software and the definition of the data virtualization concept, we can infer that the most suitable software to implement a data virtualization project is Drill.

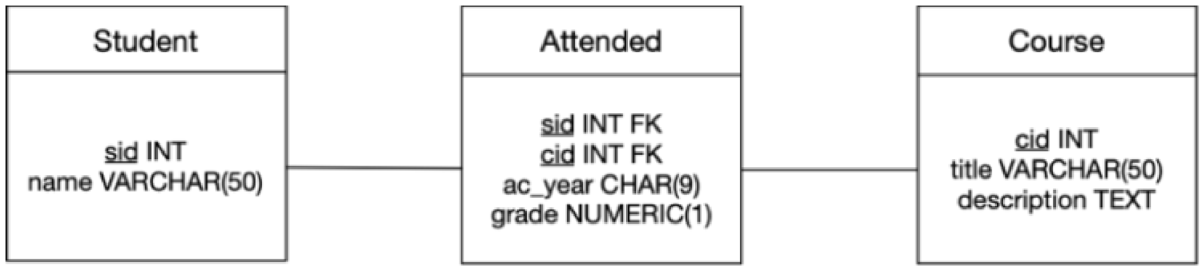
Task 2 – Suggest for a tool that will allow for simple queries on multiple data sources for a bookstore client.

We started working for a large bookstore company. Our client has a large data center containing data in various formats. More specifically, all client data (e.g., personal information, orders) are stored in a Mongo DB database, e-books are stored on HDFS, and social media metadata (likes, ratings, reviews) are stored in a Hive database. Our client would like to simplify the queries used by various User Interface elements.

We would suggest the use of Apache Drill to simplify the queries that will simultaneously process data from all the different data sources. In specific, Drill will be able to access social media metadata in the Hive metastore through Thrift. In addition, Drill can access easily the e-books stored on HDFS and the client data stored in the MongoDB. The setup needed for Drill to access the data stored on HDFS and MongoDB is quite fast and easy. As a result, we would suggest for the installation of Apache Drill on the cluster of the data center of our client in order to properly deal with this task.

Task 3 – Create an Impala Database given a specific schema.

We have another client that already has an Impala Database. They want us to create a new Impala Database according to the schema presented in plot 1.



Plot 1. Impala Database Schema

The first step is to create the database and the required tables. We will name the database as “MSc_BA”, given the theme of the tables. The commands that are needed for the above tasks are:

1. --To create the database named MSc_BA. We use if not exists so that no action is taken if a database with that name already exists
   ```
   CREATE DATABASE IF NOT EXISTS MSc_BA;
   ```
2. --We declare the database in which we want the tables to be created in
   ```
   USE MSc_BA;
   --we create the table “Student” with the columns sid (student id) and name
   CREATE TABLE Student(sid INT
   , name VARCHAR(50));

   --we create the table “Course” with the columns cid (course id), title and description of the course
   CREATE TABLE Course (cid INT
   , title VARCHAR(50)
   , description STRING);

   --we create the table “Attended” with the columns sid (student id), cid (course id), ac_year
   (academic year) and grade
   CREATE TABLE Attended (sid INT
   , cid INT
   , ac_year CHAR(9)
   , grade DECIMAL(3,1)
   ,CONSTRAINT fk1 FOREIGN KEY (sid) REFERENCES Student(sid) DISABLE
   ```

```
NOVALIDATE
, CONSTRAINT fk2 FOREIGN KEY (cid) REFERENCES Course(cid) DISABLE
NOVALIDATE);
```

Following, we will provide an example of an insert statement for the student table using my own credentials.

3.
```
--we use the insert statement to insert manually a record in the table student
INSERT INTO Student (sid, name)
VALUES (1, Konstantinos Ninas);
```

Next, we are asked to provide a query that should retrieve the names of all students that attended the course “Artificial Intelligence” during the academic year 2021-2022.

4.
```
SELECT Student.name as Student_Name
FROM Student JOIN Attended JOIN Course
WHERE (Student.sid = Attended.sid AND Attended.cid = Course.cid AND
Course.title=’Artificial Intelligence’ AND Attended.ac_year=’2021-2022’);
```

Finally, we are asked to provide a query that retrieves the titles and the average grades of all the courses for which the average grade of the students that attended them is lower than 6.

5.
```
SELECT Course.title as Course_Title,
avg(Attended.grade) as Average_Grade
FROM Course JOIN Attended
WHERE (Course.cid = Attended.cid)
GROUP BY Course.title
HAVING avg(Attended.grade)<6;
```

Task 4 – Identify a query with bad performance and try to improve it.

In the final task, we are assigned to identify the performance of each query, and to find the one with the worst performance. To identify the performance of each query we will use the profile command. The profile command displays the low-level information about the query. This command is used for diagnosis and performance tuning of a query. An example of such query would be:

1.
```
SELECT Student.name as Student_Name
FROM Student JOIN Attended JOIN Course
WHERE (Student.sid = Attended.sid AND Attended.cid = Course.cid AND
Course.title=’Artificial Intelligence’ AND Attended.ac_year=’2021-2022’);
profile;
```

Another command that can be used to identify the performance of a query is the command “summary”. The summary command within the impala-shell interpreter gives an easy-to-digest overview of the timings for the different phases of execution for a query. An example of the use of the summary command would be:

2.
   ```
   SELECT Student.name as Student_Name
   FROM Student JOIN Attended JOIN Course
   WHERE (Student.sid = Attended.sid AND Attended.cid = Course.cid AND
   Course.title=’Artificial Intelligence’ AND Attended.ac_year=’2021-2022’);
   summary;
   ```

It is noticed that the query with the greatest delay was the one that required the names of all students that attended the course “Artificial Intelligence” during the academic year 2021-2022. The delay occurred due to the number of joins in the query. To solve that issue we would suggest to completely drop the table “Student” and add a column to the table “Attended” that would include the name of each student. With this approach, we would need less joins in the query and thus, we would reduce the delay in the query execution. The commands needed to take the above actions can be observed below:

3.
   ```
   --To add the student name (sname) column to the Attended table
   ALTER TABLE Attended ADD COLUMNS (sname VARCHAR(50));
   ```

4.
   ```
   --To delete the table Student
   drop table Student;
   ```

5.
   ```
   –Then the query would be the following
   SELECT Attended.sname as Student_Name,
   FROM Attended JOIN Course
   WHERE (Attended.cid = Course.cid AND
   Course.title=’Artificial Intelligence’ AND Attended.ac_year=’2021-2022’);
   ```

On the contrary, if the removal of any table is not desired, we could use the “STRAIGHT_JOIN” command. If an Impala join query is inefficient because of outdated statistics or unexpected data distribution, you can keep Impala from reordering the joined tables by using the STRAIGHT_JOIN command. The STRAIGHT_JOIN keyword turns off the reordering of join clauses that Impala does internally and produces a plan that relies on the join clauses being ordered optimally in the query text. An example of the use of the above command is the following:

6.
   ```
   SELECT STRAIGHT_JOIN Student.name as Student_Name
   FROM Student JOIN Attended JOIN Course
   WHERE (Student.sid = Attended.sid AND Attended.cid = Course.cid AND
   Course.title=’Artificial Intelligence’ AND Attended.ac_year=’2021-2022’);
   ```

Additionally, we would suggest that the data type of the column ac_year (academic year) in the “Attended” table be updated to VARCHAR(9) or STRING. We recommend this update because, currently CHAR type currently does not have the Impala Codegen support, while both VARCHAR and STRING types do, and the performance gain of Codegen outweighs the benefits of fixed width CHAR. To perform the above update, we would execute a query such as the following:

7.
   ```
   ALTER TABLE Attended CHANGE ac_year ac_year VARCHAR(9);
   ```