

Denormalized Raw Data Architecture in Hadoop

Mrs. Sheetal S. Patil, Vaishnav Singh, Sushmita Kumari, Shubham Jain
Bharati Vidyapeeth (Deemed To Be) University, College Of Engineering, Pune

Abstract - The Concept of Denormalization of raw data management systems. is one of the challenging concepts implemented by data Professionals and companies. The idea of this concept was created from business field instead of academic field. Still very few companies have implemented this concept in order to get most value from it. There are many challenges faced by organization on implementing this. We are implementing this concept using Hadoop distributed file system. With the help of this concept we can store a large amount of data and perform analytics at a much faster rate, which cannot be done in relational database.

Keywords: Hadoop, HDFS, Map Reduce

I. INTRODUCTION

We are facing a lot of challenges as data evolves into greater diversity (more types of data, more sources of data). Data produced by organization is manipulated by data mart to gain more finer insight in finer granularity. So denormalization of data is necessary. Big data technologies are also called as destructive technology as they have revolutionized the way of doing things. Initially we are going to transfer data from relational database management system environment to HDFS environment for this we are using sqoop tool. This work is done in landing zone and then cleaning and deduplication of data is done in standardization zone. Once the cleaning of data is done we also encrypt the private data of the user in the standardization zone. The most recent regions in information modernization is the expansion of information meres to both greenfield and previous information biological communities. The concept of denormalization of raw data is as of now underway in numerous multiplatform information distribution center conditions, progressed examination applications, and the mixture information biological systems encompassing client relationship the executives and deals compel computerization. TDWI feels that this concept is setting down deep roots, and a lot more associations will receive it for a developing rundown of utilization cases in big business investigation and activities. The motivation behind this report is to quicken clients' comprehension of information lakes and their new accepted procedures, use cases, techniques for sorting out the activities, and information stages and instruments that are related with it (regardless of whether from sellers or open source).

The Denormalized raw data is a unified storehouse that enables you to store all your organized and unstructured

information at any scale. You can store your information as seems to be, without having to initially structure the information, and run distinctive sorts of investigation—from dashboards and perceptions to huge information preparing, ongoing examination, and AI to direct better choices.

II. METHODOLOGY

As Denormalized raw data is generally a new idea, there are just a couple of scholarly writings which are focused to Data Lake. Denormalized raw Data is characterized as "a procedure empowered by an enormous information storehouse dependent on minimal effort advancements that improves the catch, refinement, recorded, and investigation of crude information inside a venture." It may contain crude, unstructured or multi-organized information where most piece of these information may have unrecognized incentive for the association.

The essential thought of this concept is straightforward, all information transmitted by the association will be put away in a solitary information structure called raw data. Information will be put away in the lake in their unique organization. Complex preprocessing and change of stacking information into information stockrooms will be dispensed with. The forthright expenses of information ingestion can likewise be diminished. When information is set in the HDFS, it's accessible for investigation by everybody in the association. Denormalization of raw data particularly from the perspective of business area rather than research network.

- All information is stacked from source frameworks.
- No information is dismissed.
- Data are put away at the leaf level in an untransformed or about untransformed state.

III. PROPOSED SYSTEM

- **Basic Architecture**

Data Sources

The data companies examine through commercial intelligence comes from a various type of data sources. The most common of these are:

- Databases
- Flat Files
- Web Services
- Other sources such as RSS feeds

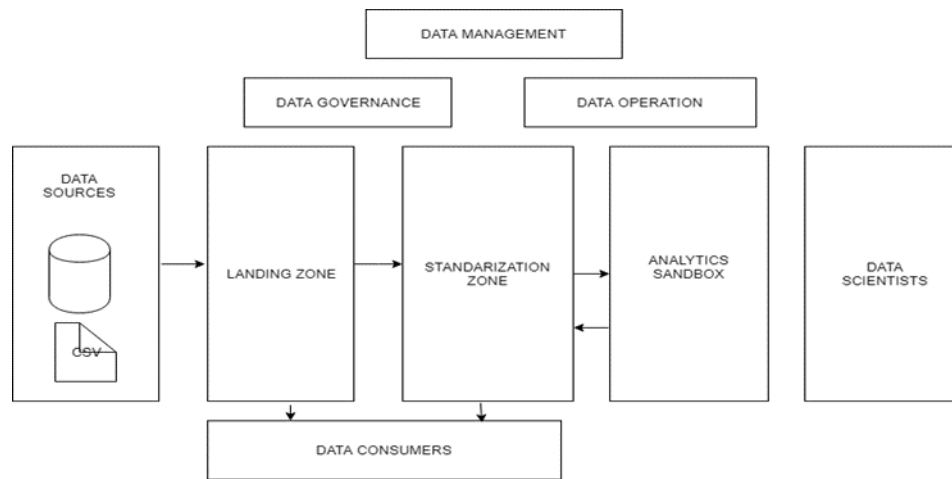


Fig 1:Data Lake Architecture

Landing Zone

Enormous information ingestion is tied in with moving information - particularly unstructured information from where it is begun, into a framework where it tends to be put away and dissected, for example, Hadoop[1].

Information ingestion might be persistent or offbeat, constant or bunched or both (lambda design) contingent on the attributes of the source and the goal. In numerous situations, the source and the goal might not have similar information timing, arrangement or convention and will require some sort of change or transformation to be usable by the goal framework.

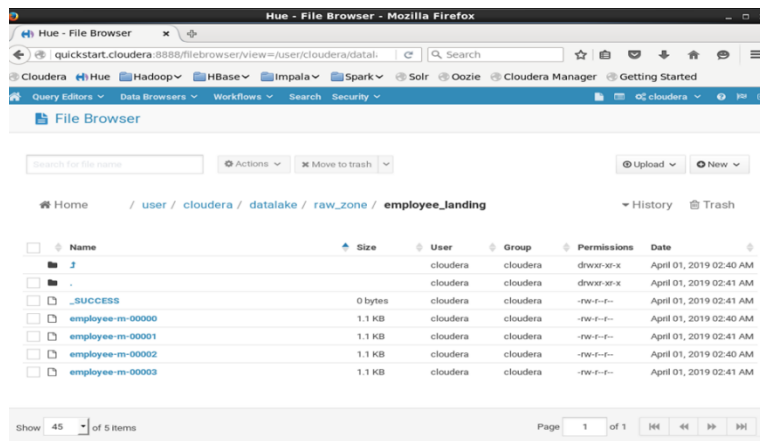
As the quantity of IoT gadgets develops, both volume and difference of information sources are extending quickly, sources which presently should be suited, and frequently continuously. However separating the information with the end goal that it tends to be utilized by the goal framework is a critical test as far as time and assets. Making information ingestion as proficient as conceivable helps center assets around enormous information gushing and examination, as opposed to the ordinary endeavors of information arrangement and change[8].

Command For Sqoop

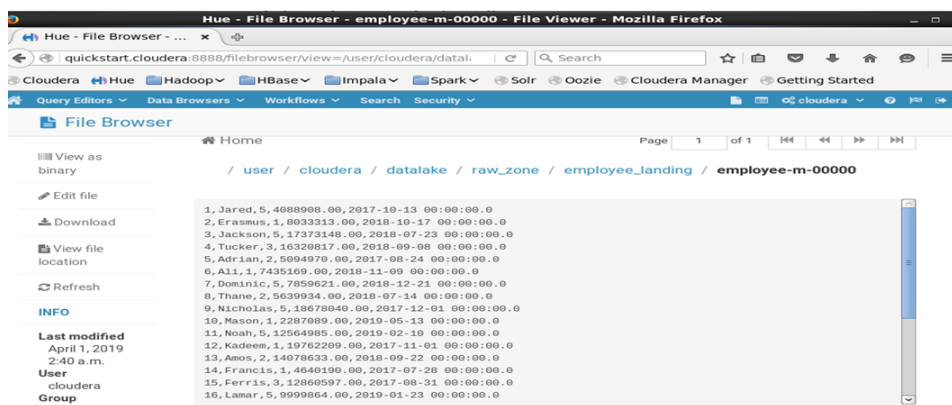
```

cloudera@quickstart:~$ sqoop import \
> -D mapreduce.output.basename=dept \
> --connect jdbc:mysql://localhost/Employee \
> --username root \
> --password cloudera \
> --table dept_datalake \
> -m 1 \
> --target-dir /user/cloudera/datalake/raw_zone/dept_landing/
Warning: /usr/lib/sqoop/.accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
19/04/01 02:45:58 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.10.0
19/04/01 02:45:58 WARN tool.BaseSqoopTool: Setting your password on the command-
line is insecure. Consider using -P instead.
19/04/01 02:45:58 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
19/04/01 02:45:58 INFO tool.CodeGenTool: Beginning code generation
19/04/01 02:45:59 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `dept_datalake` AS t LIMIT 1
19/04/01 02:45:59 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `dept_datalake` AS t LIMIT 1
19/04/01 02:45:59 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/ha
doop-mapreduce
Note: /tmp/sqoop-cloudera/compile/b063224110bf98501127a4014b4e454b/dept_datalake
.java uses or overrides a deprecated API.
    
```

File Stored in HDFS



Data Representation in HDFS



Standardization Zone

- Data De-Duplication.
- Data Security.

These are the two different tasks to be performed in standardization zone.

Data De-Duplication: -

Whenever some data is updated in the RDBMS and then we ingest it in HDFS it cause data de-duplication which can cause error in your final analysis.so we have to remove the older record and keep the updated record[2].

For Example Let’s assume an employee who has been promoted as a manager ,so in this case the database is updated and there are two different records of the same employee so we have to remove the older record and keep the updated record.

It cannot be done by using a simple WHERE clause as the data is in huge amount so scanning each element will take much time so we will be using the RANK function to solve this problem.

```

INSERT OVERWRITE TABLE datalake_raw_tables.employee_raw_${YYYYMMDD}
SELECT
    id
    ,
    first_name
    ,
    dept_id
    ,
    sal
    ,
    datetime_updated
    ,
    year
    ,
    month
    ,
    day
    
```

```

FROM
(SELECT *, RANK() over (PARTITION BY id
ORDER BY datetime_updated DESC, year desc, month desc,day DESC) as rank
FROM datalake_raw_tables.employee_raw
WHERE ((year = ${YYYY} AND month = ${MM} AND day <= ${DD}) OR (year = ${YYYY} AND month = ${MM} AND day <= ${DD}))
WHERE ranked_raw.rank=1;
    
```

Hive Query For De-Duplication

Data Security

Data Security is necessary as the data should be accessible to authorized person only. So we try to encrypt the private data using AES encryption algorithm[5].

For using AES encryption we use the following steps

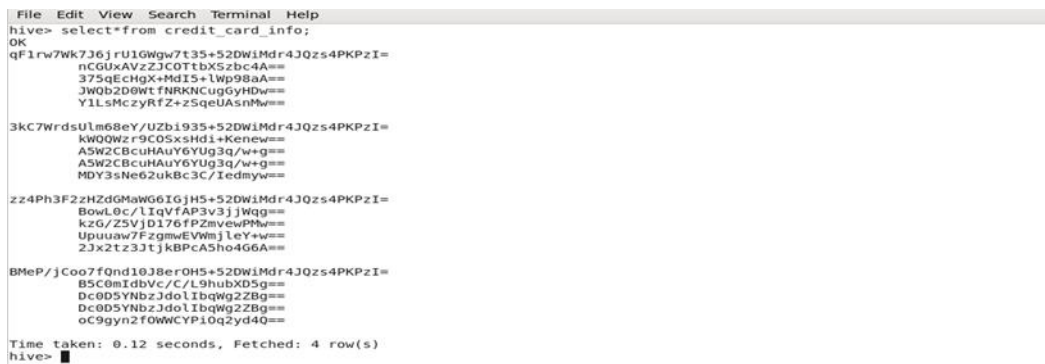
- Write two programs of AES Encryption and Decryption in JAVA.

- Export these two programs as JAR files.
- Once the jar files are created , add the jar files in Hive environment.
- Then create two temporary functions as Encrypt and Decrypt which can be used along with the hive query for encryption and decryption purpose.

Without Encryption

```
hive> select*from credit_card_info_stg;
OK
3453245436578975      1234      50000.0 35000.0 2017-03-22
3453227896578975      1243      500000.0      500000.0      2017-03-12
3453227833268975      1324      15000.0 12000.0 2017-03-09
3453227897332975      4312      60000.0 60000.0 2017-02-16
Time taken: 0.169 seconds, Fetched: 4 row(s)
hive>
```

After Encryption



```
File Edit View Search Terminal Help
hive> select*from credit_card_info;
OK
qF1rw7Wk7J6jrU1Gwgw7t35+52DwIMdr4JQzs4PKPzI=
nCGUxAVzZJCOTtbX5zbc4A==
375qEcHgX+MdI5+Lwp9BaA==
JWQb2DQWtFNKKNcug5yHd==
Y1Ls4CzyRfZ+z5qouA5nMw==
3kC7WrdsUlm68ey/Uzb1935+52DwIMdr4JQzs4PKPzI=
kWO0Wzr9C0SxsHd1+Kcne==
A5W2CBcuHAUy6YUg3q/w+g==
A5W2CBcuHAUy6YUg3q/w+g==
MDY3sNe62ukBc3C/Iedmyw==
zz4Ph3F2zHZdGMaWG6IGjH5+52DwIMdr4JQzs4PKPzI=
BowL0c/LIqVfAP3v3jJWqg==
kzG/Z5VjD176fPZmvevPMw==
Upuuaw7FzgmwEVWmJLeY+w==
2Jx2tz3JtjKBPcA5ho46GA==
BMeP/jCoo7fQnd10J8erOH5+52DwIMdr4JQzs4PKPzI=
B5C0mIdbVc/C/L9hubXD5g==
Dc0D5YNbzJdol1BqWg2ZBq==
Dc0D5YNbzJdol1BqWg2ZBq==
oC9gyn2f0WwCYP1Oq2yd40==
Time taken: 0.12 seconds, Fetched: 4 row(s)
hive>
```

Analytic Sandbox

Once the deduplication and security task is performed on the datasets we have to join all the different datasets to create one single denormalized table. Performing join operation is a costly operation as it takes more time as compared to other operations so we perform join

at a single instance and create a single denormalized table for all the different datasets[3].

Once the denormalized table is created we can perform analytics in a faster way as compared to performing joins again and again.

Query For Denormalization

```
INSERT INTO TABLE datalake_raw_tables.employee_raw_${YYYYMMDD}
SELECT prev_snapshot.*
FROM
(SELECT *
FROM datalake_raw_tables.employee_raw_${PREV_YYYYMMDD}) prev_snapshot
LEFT OUTER JOIN datalake_temp_tables.employee_raw_${YYYYMMDD}_delta
today_delta
ON prev_snapshot.id = today_delta.id
WHERE today_delta.id IS NULL
;
```

System Tools

Apache Sqoop

Apache Sqoop proficiently exchanges mass information between Apache Hadoop and organized datastores, for example, social databases. Sqoop offloads certain errands, (for

example, ETL handling) from the EDW to Hadoop for effective execution at a much lower cost. Sqoop can likewise be utilized to remove information from Hadoop and fare it into outer organized datastores. Sqoop works with social databases, for example, Teradata, Netezza, Oracle, MySQL, Postgres, and HSQLDB[6].

Apache Hive

Apache Hive is an information distribution center programming venture based over Apache Hadoop for giving information inquiry and analysis. Hive gives a SQL-like interface to question information put away in different databases and document frameworks that coordinate with Hadoop. Conventional SQL inquiries must be actualized in the MapReduce[3] Java API to execute SQL applications and questions over conveyed information. Hive gives the important SQL deliberation to coordinate SQL-like questions (HiveQL) into the basic Java without the need to actualize inquiries in the low-level Java API. Since most information warehousing applications work with SQL-based questioning dialects, Hive helps transportability of SQL-based applications to Hadoop[7].

IV. ALGORITHM

- Fetch user data and store it in relational database tables.
- With the help of sqoop fetch the data from relational database and store it in HDFS. Make sure that sqoop password is not reflected use sqoop password alias technique.
- Create staging tables and move the data into these staging tables with the help of HIVE.
- Perform Deduplication task on the given tables.
- Encrypt the data which is private.
- Perform denormalization on the given tables using Join operation.

V. LIMITATIONS

- Not feasible for working on small data sets. It's better to use RDBMS for small data sets.
- Automation of the project is a complex task as the requirements may change frequently.
- Cannot work on unstructured data.
- Cannot use a GUI for the task can be done only on command line interface.
- Only raw data can be used processed data is not preferred.

VI. CHALLENGES

- Updating tables with incremental load is a challenging task in this project for overcoming this problem we are creating external hive tables.
- Securing data is another challenging task we are using AES algorithm for the encryption of data.

VII. CONCLUSION

Enormous information is an unprecedented innovation. New kinds of examination that weren't feasible on information distribution centers are currently broad. Early information lakes were trumpeted as victories dependent on their ease and deftness. However, as more standard use cases developed, associations found that regardless they required the administration and administration controls that commanded in the information stockroom period. The information lake has turned into a center ground between information ware- houses and "information swamps" in offering frameworks that are as yet spry and adaptable, however have the shields and inspecting highlights that are vital for business-basic information. Incorporated information lake the executives arrangements like the Zaloni Data Platform (ZDP) are currently conveying the essential controls without making enormous information as moderate and resolute as its forerunner arrangements. Use cases are developing even in delicate businesses like human services, money related administrations, and retail. Endeavors are additionally looking forward. They see that to be genuinely important, the information lake can't be a storehouse; rather, it must be one of a few stages in a cautiously considered start to finish present day undertaking information design. Similarly as you should consider metadata from an undertaking wide viewpoint, you should almost certainly coordinate your information lake with outside devices that are a piece of your endeavor wide information see. At exactly that point will you have the capacity to construct an information lake that is open, extensible, and simple to coordinate into your different business-basic stages.

VIII. REFERENCES

- [1]. S.Vikram Phaneendra & E.Madhusudhan Reddy, "Big Data- solutions for RDBMS problems- A survey" In 12th IEEE/IFIP Network Operations & Management Symposium (NOMS 2010) (Osaka, Japan, Apr 19{23 2013).
- [2]. Kiran kumara Reddi & DnvsI Indira "Different Technique to Transfer Big Data : survey" IEEE Transactions on 52(8) (Aug.2013) 2348 { 2355}.
- [3]. Jimmy Lin "MapReduce Is Good Enough?" The control project. IEEE Computer 32 (2013).
- [4]. Umasri.M.L, Shyamalagowri.D ,Suresh Kumar.S "Mining Big Data Current status and forecast to the future" Volume 4, Issue 1, January 2014 ISSN: 2277 128X

- [5]. Albert Bifet “Mining Big Data In Real Time” Informatica 37 (2013) 15–20 DEC 2012.
- [6]. Bernice Purcell “The emergence of “big data” technology and analytics” Journal of Technology Research 2013.
- [7]. Sameer Agarwal†, Barzan MozafariX, Aurojit Panda†, Henry Milner†, Samuel MaddenX, Ion Stoica “BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data” Copyright © 2013i ACM 978-1-4503-1994 2/13/04.
- [8]. Yingyi Bu _ Bill Howe _ Magdalena Balazinska _ Michael D. Ernst “The HaLoop Approach to Large-Scale Iterative Data Analysis” VLDB 2010 paper “HaLoop: Efficient Iterative Data Processing on Large Clusters.
- [9]. Shadi Ibrahim★ _ Hai Jin _ Lu Lu “Handling Partitioning Skew in MapReduce using LEEN” ACM 51 (2008) 107–113.
- [10]. Kenn Slagter · Ching-Hsien Hsu “An improved partitioning mechanism for optimizing massive data analysis using MapReduce” Published online: 11 April 2013.