Database Consistency Verification for CMS metadata

August 2016

Author:
Shubham Gupta

Supervisor(s):
Valentin Kuznetsov
Katarzyna Maria Dziedziniewicz-wojcik
Nicolò Magini

CERN openlab Summer Student Report

15 years CERN openlab

Table of Contents

Abstract

The CMS experiment has two largely populated databases for CMS meta-data, the Data Book keeping System (DBS) and data location system (PhEDEx) which are populated by the production teams with information on the identity and contents of the CMS datasets.

Due to internal workflow organization some information may not be in sync between these databases, e.g. the number of files in each dataset etc. The goal of this project is to develop a cross-consistency tool to get periodic updates about database discrepancies.

Describing the Data

The CMS (Compact Muon Solenoid) Detector produces a huge amount of raw data every second. Through the application of various filters and multi-level triggers, the amount of data is reduced to around 600 megabytes (This is suspected to be an older figure and the actual number should be bigger) a second [2]. The amount of data despite the reduction by triggers is large.

Dealing with such amounts of data is difficult and a lot of information about this data is required in order to make meaningful use of it. This data about the data is known as metadata.

The two primary metadata datasets that formed the basis of this project were Data Book keeping Service (DBS) and Physics Experiment Data Export (PhEDEx).While DBS and PhEDEx both store similar data, their purposes are entirely different.

1. **Data Book keeping Service (DBS):**

   The Data Book keeping Service (DBS) provides a catalog of event metadata for Monte Carlo and recorded data of the CMS experiment. DBS contains record of what data exist, their process-oriented provenance information, including parentage relationships of files and datasets, their configurations of processing steps, as well as associations with run numbers and luminosity sections to find any particular subset of events within a dataset, on a large scale of about 200,000 datasets and more than 40 million files, which adds up to around 700 GB of metadata [1].

2. **Physics Experiment Data Export (PhEDEx):**

   The Physics Experiment Data Export (PhEDEx) project was started in 2004 to manage global data transfers for CMS over the Grid in a robust, reliable, and scalable way. PhEDEx is based on a high-availability Oracle database cluster hosted at CERN (Transfer Management Data Base, or TMDB) acting as a blackboard" for the system state (data replica location and current tasks). Users can request transfers of datasets or blocks of files through the interactive PhEDEx web site; a web data service is also available for integration with other CMS data management components [1].

Initial Steps and Testing

So during the initial days, we discussed different ways to tackle the problem of resolving inconsistencies. One of the ways was using the Oracle database which already had DBS and PhEDEx data available. Another approach discussed involved using Apache Spark as the PhEDEx data was also available through hdfs and DBS data could be obtained either by adding it to hdfs or through DBS APIs. We went for the first approach that is using oracle database both due to its simplicity and due to easier availability of the data.

After accesses to all the copies of the DBS and PhEDEx tables had been obtained, a number of test procedures were built to understand the size of the data stored and speed at which the whole table could be traversed and also to measure the time taken for other operations such as comparison of different columns.

Initial tests which used a limited set of fields gave encouraging results with maximum time for a rudimentary comparison script to run being around 5 to 7 minutes. So all blocks of DBS were being traversed and compared with PhEDEx in around 5-7 minutes.

These test procedures were later removed from the github repository as they had served their purpose, however access to them should be available through git version control if it is required. But due to their primitive nature, it would require the user to install them manually into a suitable schema with all the required grants.

The tables and Schemas

The DBS schema's copy that was being used for this project was named cms_dbs3_prod_part. It was a partial copy of the production DBS database. The tables of DBS that were being used were named Blocks, files and PhEDEx's partial copy was named cms_transfermgmt_part. The tables of PhEDEx that were the counterparts to blocks and files were T_DPS_BLOCK and T_DPS_FILE.

I was using the schema named cms_dbs_tranf_ver which had select grants to both of the above schemas and all the procedures were running through this schema and also the results were being stored on the tables that were created on this schema only. To store the results of the inconsistencies, three tables were created, these were inconsistent_blocks, inconsistent_files and invalid_dbs_blocks.

The inconsistent_blocks and inconsistent_files tables as is evident from their names, store details about inconsistent blocks and files respectively. The invalid_dbs_blocks table stores details about those blocks which contain at least one invalid file. The total number of files and the number of invalid files in a block are also stored in this table. Only blocks where all its files were found to be valid re not stored in this table. There is also an is_valid field in this table which has the value 1 if all the files in the block are invalid and it is zero otherwise.

The output

One of the first things done in this project was agreement on the format of the outputs. It was agreed to use Comma Separated Value (CSV) file format due to its simplicity as well as the possibility of use in other applications such as an input for web based visualization tools.

As discussed earlier, the procedures running in oracle were storing the results in a table. Rather than using yet another procedure for generating the CSV files, I used python to extract and produce the CSV files. This also had the advantage that creating a command line interface was easier and it also allowed me to create many custom options for the CSV output like including only certain types of blocks or files. Such degree of customization in the output CSV file would have been difficult to achieve using plsql procedures alone.

Format of CSV file

Before discussing the format of the CSV file it is important to understand the type of inconsistencies expected. There should be three broad cases of inconsistency which are expected to be extracted. They are:

1. A block/file is present in DBS but not in PhEDEx.
2. A block/file is not present in DBS but it is present in PhEDEx.
3. Both DBS and PhEDEx have the data blocks/files but parameters like size or number of files do not match.

Based on these types of inconsistencies, the format which was agreed upon was:

DBS_BLOCK_ID,DATASET_ID_DBS,BLOCK_NAME_DBS,FILE_COUNT_DBS,BLOCK_SIZE_DBS,OPEN_MISMATCH_DBS,PHEDX_BLOCK_ID,DATASET_ID_PHEDX,BLOCK_NAME_PHEDX,FILE_COUNT_PHEDX,BLOCK_SIZE_PHEDX,OPEN_MISMATCH_PHEDX

So CSV file would contain rows of inconsistency data formatted in this way. The first part on the CSV row is obtained from DBS and the second part from PhEDEx. According to this format, the rows for the three cases would look like.

1. This is the case where a Block is present in DBS but not in PhEDEx. As we can see that the right part of the row is empty and consists only of commas which are the value separators. An example of this case is provided below.

555717,13394,/Cosmics/Commissioning09-PromptReco-v1/RECO#b671cae3-5b60-498c-b689-9797038757eb,20,104660375913,0,,,,,,

2. This case is similar to the above case but instead of missing PhEDEx, DBS block is missing. So the values on the left consist only of the comma which again are the value separators. An example of this case is provided below.

,,,,,,6905414,890207,/DYJetsToLL_M-50_HT-800to1200_TuneCUETP8M1_13TeV-madgraphMLM-pythia8/RunIISummer15GS-MCRUN2_71_V1-v1/GEN-SIM#a4fbc1c4-38d1-11e6-a6d3-001e67abefa8,4,9493245405,1

3. In this third and final case, we see that the values for both DBS and PhEDEx blocks are present but there is a mismatch in one of the block parameters. An example of this case is also provided below.

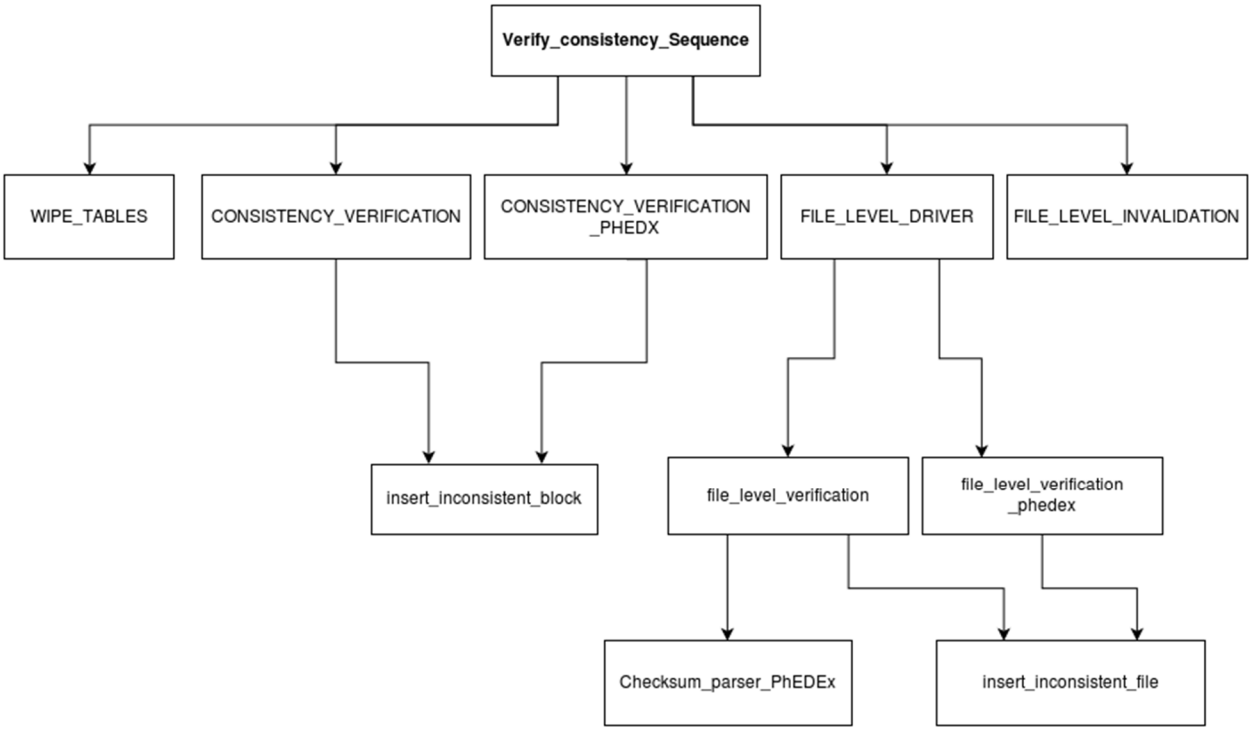
14668820,12528293,/SingleMuMinusFlatPt3To70_EtaPhiRestricted/RunIIFall15DR76-PU25nsData2015v1_76X_mcRun2_asymptotic_v12-v1/AODSIM#3fcf519c-9a81-11e5-8972-a0369f23d008,1,2147483647,0,5768578,796298,/SingleMuMinusFlatPt3To70_EtaPhiRestricted/RunIIFall15DR76-PU25nsData2015v1_76X_mcRun2_asymptotic_v12-v1/AODSIM#3fcf519c-9a81-11e5-8972-a0369f23d008,1,2709479936,0

Along with these formats, it was also agreed that we need to have a row indicating the column names and the next row to the column names giving the number of rows in the document. Adding these extra rows is also handled by the python files that are responsible for creating the CSV files in the required format.

This format description is also provided in the direction.md [3] file of the github repository.

Description of the procedures used in Consistency verification

The diagram below depicts the procedures and their dependency on other procedures. The leaf nodes in this pseudo-graph can be considered as independent procedures and the parent nodes are the procedures those procedure which depend upon the leaf node procedures.



1. Verify_Consistency_sequence

The *Verify_consistency_sequence* is the main procedure which runs a sequence of other procedures. It effectively is the procedure that orchestrates the whole *consistency verification* process. It starts with wiping old data from the tables (wipe_tables procedure) and then it runs *consistency_verification_2* followed by *consistency_verification_phedx* and then it calls the *file_level_driver* and finally the *file_level_invalidation*

procedure. This is the procedure that is expected to be run as a DBMS Job periodically.the sequence in which the procedures are executed by verify_consistency_sequence is significant since each of the procedure depends upon the successful execution of its preceding procedures to run.

2. Consistency_verification_2

This procedure is the block level consistency verification procedure, it scans the BLOCKS table in DBS and T_DPS_BLOCK table in PhEDEx to find inconsistent blocks by comparing parameters such as block size, the number of files in the block (file count) and also the open/close status of the block. Blocks where these parameters don't match or cases where there is a DBS block for which a PhEDEx block and vice versa is not found are then stored in the inconsitent_blocks table using the *insert_inconsistent_block_procedure*.

3. Consistency_verification_phedex

Finds all DBS blocks from the inconsistent_blocks tables which have no equivalent PhEDEx counterpart and then scans all of their files for the validity status of the files. If all files are found to be invalid then this is stored in the invalid_dbs_blocks. It also stores the field IS_BLOCK_INVALID as 1. If not all files are invalid, the IS_BLOCK_INVALID is stored as zero along with the number of invalid and total files. Also if the number of invalid files is found to be zero then details of such a block are not recorded in the invalid_dbs_blocks table.

4. File_level_driver

This procedure manages consistency verification at the file level. It just finds blocks which are present in both PhEDEx and DBS but have a mismatch in a parameter such as file count or block size and then applies file_level_verification and file_level_verification_phedx procedures for each of the blocks. The results of these procedures are stored in the inconsistent_files table by the insert_inconsistent_file procedure.

5. File_level_invalidation

This procedure uses the inconsistent_blocks table to find all the blocks which have no PhEDEx counterpart and check the validity status for all the contained files. The results of this operation are stored in the invalid_dbs_blocks table. The fields of this table also store the total number of files in a block and how many of them are inconsistent as well as is_invalid field which stores 1 if all files stored are invalid.If all the files of a block are` valid then no record of that block is stored, on the other hand, blocks having a part of their files invalid are stored in the invalid_dbs_blocks but the is_block_invalid field for them is assigned as 0.

6. Wipe_tables

The wipe tables procedure is used to clear the contents of the inconsistentc_blocks, inconsistent_files and invalid_dbs_blocks tables before other procedures contained in the verify_consistency_sequence can be run. Earlier this procedure used the delete from method but this was later replaced with the faster truncate method.

7. Insert_inconsistent_block

This procedure's function is to insert a record into the inconsistent_blocks table.The reason behind creating this procedure rather than using a direct insert statement was to make it to easier and shorter to use insertion and also as capabilities like producing the time stamp and converting it into unix time can be handled by the procedure automatically.

8. File_level_verification

Finds all the files in DBS for a given DBS block_id and then finds the corresponding file in PhEDEx. If the file is found then compares parameter such as file size and checksums, Adler32 and stores the results via insert_inconsistent_file procedure if an inconsistency is found. If file is not found in PhEDEx then also an inconsistency is recorded using the insert_inconsistent_file procedure.

9. File_level_verification_PhEDEx

Performs the same function as file_level_verification but from PhEDEx's side. It iterates over files in PhEDX's T_DPS_FILE finding files with a given input of a block_id to verify the existence of the same files in DBS's table of FILES. It does not perform parameter checking processes as that is already handled by the file_level_verification and if files of PhEDX are present in DBS then their parameters are assumed to have been checked and handled by the other procedure. This procedure just detects absence of PhEDX files in DBS.

10. checksum_parser_PhEDEx

PhEDEx and DBS store checksums in different ways. While DBS has separate fields for each type of checksum, PhEDEx stores all checksums in a single field as a set of key value pairs.

This procedure takes the string of key value pairs of PhEDEx checksums as input and provides outputs in the form of Adler32 and Checksum. These values then can be used for comparison of files between DBS and PhEDEx.

11. Insert_inconsistent_file

This procedure performs the identical function as the insert_inconsistent_block but instead of inserting inconsistet_blocks, it is used for inserting consistent files to the inconsistent_files table.

12. Create_job

This procedure is used to create the DBMS job that will be run on the database. By default it ise set to run at 03:00 every day. It will execute the verify_consistency_sequence every 24 hours.

CLI using ARGPARSE

After the completion of PL/SQL procedures, the next part of the project was creating the command line interface using python that will be used to generate the CSV files.

I used argparse to create this command line interface due to its simplicity and was able to incorporate many features in it that allows us to create CSV files with a good level of customization.

1. **login**

This file contains a function called return_credentials() and as suggested by its name, it stores and returns the login credentials for the database.

2. **Connect_db**

This file has the connect() function and it is used every time a connection to the DB is required. It is also capable of printing an error message incase invalid credentials are supplied which leads to the connection being refused

3. **Csv_functions**

This file has functions that are used during the processes of creating the CSV files. The rearrange() function is used to rearrange the table's columns into the order that is specified by the required CSV format.

The generate_header() function uses the cursors' description attribute to extract column names and join it into a row that s added to the beginning of every CSV file created.

The assign_sql() function is used in choosing the appropriate sql query to execute in order to produce the get an output of the rows that satisfy the criteria as provided by the user.

4. **Cli_functions**

This file contains functions that are used for parsing arguments by argparse as well as generating appropriate csv files by calling csv_gen() functions.

5. **Csv_gen**

This file has the create_csv() function that performs the actual creation of the CSV file by combining various parameters and functions from the above python files. It will also create CSV files containing all inconsistencies by default if called directly though this approach is not recommended.

6. Verify_consistency

This is the main python file that imports cli_functions module and calls functions which parse the command line arguments and call the appropriate responses to them. This file by default will produce CSV files containing all the inconsistencies if no other parameter is provided.

The code for these files can be seen in the github repository inside the python_files folder[4].

Python versions

The command line interface has been tested with both python 2.6 (used on lxplus) and Python 2.7 (default for CMS) versions and no problems due to the different versions were encountered.

Details about the Bash scripts

There are also two bash scripts that have been included in the github repository. These are install_tables_and_procedures.sh and db_wipe_sequence.sh

Install_tables_and_procedures script is useful as it automates the process of creating all the tables and adding all the procedures to the schema. The user of this script needs to edit the script to add the username and password.It concatenates the sql and plsql code in the sql_files folder in a particular order so that procedures that are dependent on other procedures are not installed first as doing so would cause errors.

Db_wipe_sequence script is useful for deleting all the data and removing all the tables and procedures from the schema.It acts like an uninstall script for our consistency verification tool. It however cannot remove the DBMS job that is created by install_tables_and_procedures script. We also need to edit this script to add the login credentials.

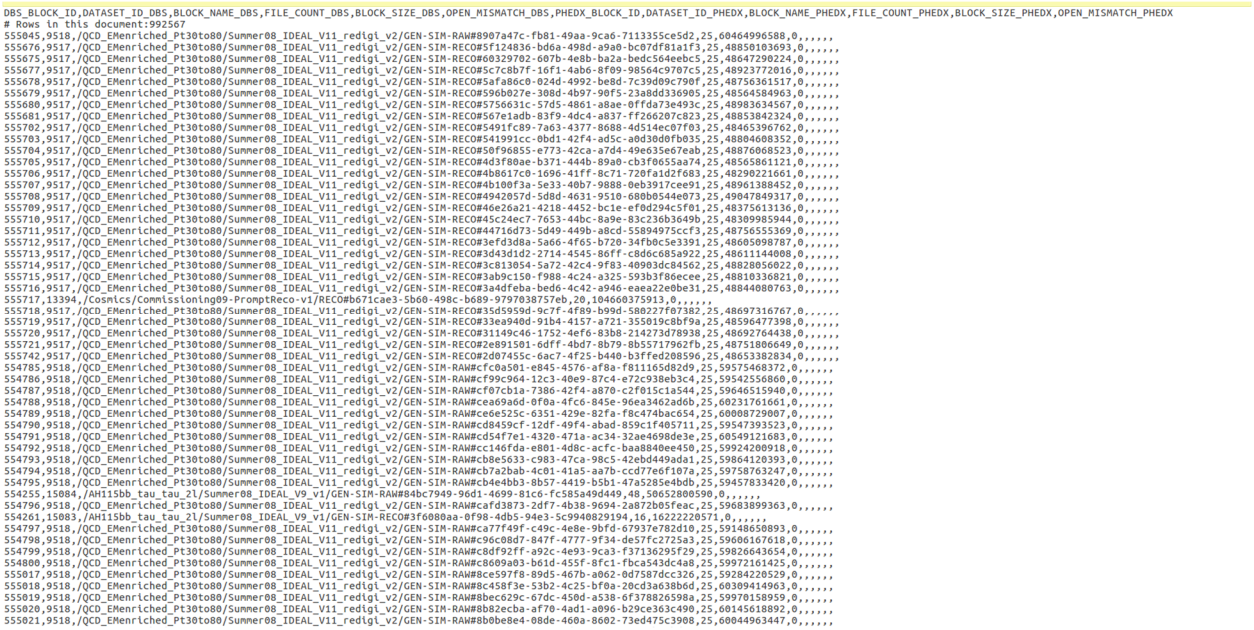
These scripts are designed to function properly only when they are run after navigating into the DBConsistencyCheck folder. If new plsql procedures are created, their names should be added to these scripts also.

A screenshot of the Help message Displayed in the CLI

```
shubham@shubham-Lenovo-Z50-70:~/Desktop/DBConsistencyCheck/python_files$ python
verify_consistency.py -h
usage: verify_consistency.py [-h] [-f] [-b] [-i] [-d] [-p] [-dp] [-pd] [-v]
                             [csv_destination]

Produces CSV document conatining details about inconsistent block and files.

positional arguments:
  csv_destination  Used to specify the destination of the produced CSV files.
                   Default location is the current directory.

optional arguments:
  -h, --help       show this help message and exit
  -f, --files      generates CSV document for inconsistent files
  -b, --blocks     generates CSV document for inconsistent blocks
  -i, --invalid    generates CSV document for invalid blocks which contain
                   only invalid files
  -d, --dbs        generates CSV document contining DBS blocks/files
  -p, --phedx      generates CSV document contining PhEDX blocks/files
  -dp, --dbsphedx  generates CSV document contining both DBS and PhEDX
                   blocks/files
  -pd, --phedxdbs  generates CSV document contining both DBS and PhEDX
                   blocks/files
  -v, --verbose    prints row arrays to console as they are added to the CSV
```

This screenshot shows the message that is displayed when help tags is sent into the python program. As it can be seen various options are available for use by the user and this leads to creation of customized files for the user.

A screenshot of the CSV file



```
DBS_BLOCK_ID,DATASET_ID_DBS,BLOCK_NAME_DBS,FILE_COUNT_DBS,BLOCK_SIZE_DBS,OPEN_MISMATCH_DBS,PHEDX_BLOCK_ID,DATASET_ID_PHEDX,BLOCK_NAME_PHEDX,FILE_COUNT_PHEDX,BLOCK_SIZE_PHEDX,OPEN_MISMATCH_PHEDX
# Rows in this document:992567
555045,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#8907a47c-fb81-49aa-9ca6-7113355ce5d2,25,60464996588,0,,,,,
555676,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#5f124836-bd6a-498d-a9a0-bc07df81a1f3,25,48850103693,0,,,,,
555675,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#60329702-607b-4e8b-ba2a-bedc564eebc5,25,48647290224,0,,,,,
555677,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#5c7c8b7f-16f1-4ab6-8f09-98564c9707c5,25,48923772016,0,,,,,
555678,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#5afa86c0-024d-4992-be8d-7c39d09c790f,25,48756361517,0,,,,,
555679,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#596b027e-308d-4b97-90f5-23a8dd336905,25,48564584963,0,,,,,
555680,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#5756631c-57d5-4861-a8ae-0ffda73e493c,25,48983634567,0,,,,,
555681,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#567e1adb-83f9-4dc4-a837-ff266207c823,25,48853842324,0,,,,,
555702,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#5491fc89-7a63-4377-8688-4d514ec07f03,25,48465396762,0,,,,,
555703,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#541991cc-0bd1-42f4-ad5c-a0d30d0fb035,25,48804608352,0,,,,,
555704,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#50f96855-e773-42ca-a7d4-49e635e67eab,25,48876068523,0,,,,,
555705,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#4d3f80ae-b371-444b-89a0-cb3f0655aa74,25,48565861121,0,,,,,
555706,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#4b8617c0-1696-41ff-8c71-720fa1d2f683,25,48290221661,0,,,,,
555707,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#4b100f3a-5e33-40b7-9888-0eb3917cee91,25,48961388452,0,,,,,
555708,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#4942057d-5d8d-4631-9510-680b0544e073,25,49047849317,0,,,,,
555709,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#46e26a21-4210-4452-bc1e-ef0d294c5f01,25,48375613136,0,,,,,
555710,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#45c24ec7-7653-44bc-8a9e-83c236b3649b,25,48309985944,0,,,,,
555711,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#44716d73-5d49-449b-a8cd-55894975ccf3,25,48756555369,0,,,,,
555712,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#3efd3d8a-5a66-4f65-b720-34fb0c5e3391,25,48605098787,0,,,,,
555713,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#3d43d1d2-2714-4545-86ff-c8d6c685a922,25,48611144008,0,,,,,
555714,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#3cb13054-5a72-42c4-9f83-40903dc84562,25,48828056022,0,,,,,
555715,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#3ab9c150-f988-4c24-a325-593b3f86ecee,25,48810336821,0,,,,,
555716,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#3a4dfeba-bed6-4c42-a946-eaea22e0be31,25,48844080763,0,,,,,
555717,13394,/Cosmics/Commissioning09-PromptReco-v1/RECO#b671cae3-5b60-498c-b689-9797038757eb,20,104660375913,0,,,,,
555718,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#35d5959d-9c7f-4f89-b99d-580227f07382,25,48697316767,0,,,,,
555719,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#33ea940d-91b4-4157-a721-355019c8bf9a,25,48596477398,0,,,,,
555720,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#31149c46-1752-4ef6-83b8-214273d78938,25,48692764438,0,,,,,
555721,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#2e891501-6dff-4bd7-8b79-8b55717962fb,25,48751806649,0,,,,,
555742,9517,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RECO#2d07455c-6ac7-4f25-b440-b3ffed208596,25,48653382834,0,,,,,
554785,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cfc0a501-e845-4576-af8a-f811165d82d9,25,59575468372,0,,,,,
554786,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cf99c964-12c3-40e9-87c4-e72c938eb3c4,25,59542556860,0,,,,,
554787,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cf07cb1a-7386-42f4-a870-c2f015c1a544,25,59646515940,0,,,,,
554788,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cea69a6d-0f0a-4fc6-845e-96ea3462ad6b,25,60231761661,0,,,,,
554789,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#ce6e525c-6351-429e-82fa-f8c474bac654,25,60008729007,0,,,,,
554790,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cd8459cf-12df-49f4-abad-859c1f405711,25,59547393523,0,,,,,
554791,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cd54f7e1-4320-471a-ac34-32ae4698de3e,25,60549121683,0,,,,,
554792,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cc146fda-e801-4d8c-acfc-baa8840ee450,25,59924200918,0,,,,,
554793,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cb8e5633-c983-47ca-98c5-42ebd449ada1,25,59864120393,0,,,,,
554794,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cb7a2bab-4c01-41a5-aa7b-ccd77e6f107a,25,59758763247,0,,,,,
554795,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cb4e4bb3-8b57-4419-b5b1-47a5285e4bdb,25,59457833420,0,,,,,
554255,15084,/AH115bb_tau_tau_2l/Summer08_IDEAL_V9_v1/GEN-SIM-RAW#84bc7949-96d1-4699-81c6-fc585a49d449,48,50652800590,0,,,,,
554796,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#cafd3873-2df7-4b38-9694-2a872b05feac,25,59683899363,0,,,,,
554261,15083,/AH115bb_tau_tau_2l/Summer08_IDEAL_V9_v1/GEN-SIM-RECO#3f6080aa-0f98-4db5-94e3-5c9940829194,16,16222220571,0,,,,,
554797,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#ca77f49f-c49c-4e8e-9bfd-67937e782d10,25,59148650893,0,,,,,
554798,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#c96c08d7-847f-4777-9f34-de57fc2725a3,25,59606167618,0,,,,,
554799,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#c8df92ff-a92c-4e93-9ca3-f37136295f29,25,59826643654,0,,,,,
554800,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#c8609a03-b61d-455f-8fc1-fbca543dc4a8,25,59972161425,0,,,,,
555017,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#8ce597f8-89d5-467b-a062-0d7587dcc326,25,59284220529,0,,,,,
555018,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#8c458f3e-53b2-4c25-bf0a-20cd3a638b6d,25,60309414963,0,,,,,
555019,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#8bec629c-67dc-450d-a538-6f378826598a,25,59970158959,0,,,,,
555020,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#8b82ecba-af70-4ad1-a096-b29ce363c490,25,60145618892,0,,,,,
555021,9518,/QCD_EMenriched_Pt30to80/Summer08_IDEAL_V11_redigi_v2/GEN-SIM-RAW#8b0be8e4-08de-460a-8602-73ed475c3908,25,60044963447,0,,,,,
```

Here a CSV output of the inconsistency data can be seen. This CSV contains inconsistent blocks that are present in DBS but absent in PhEDEx. This is only a single screen shot of the CSV file and does not show the complete file.

Future Work

For the future of this project, I would recommend building the consistency verification tool through apache Spark and then re-implementing a basic version of the consistency verification procedures to get an idea about performance. If the results are found to be favorable, it should be worthwhile to build an implementation of the full-fledged tool in spark. At this point of time, PhEDEx data is already available on HDFS [5] and DBS data can be obtained either through the DBS APIs [5] or may be obtained by dumping its data to HDFS first. In the future, some level of migration of DBS data to HDFS may also be done.

The comparison of the spark tool with the Oracle tool is expected to help us to arrive at a better understanding of speed and performance of both the tools and also help us in the goal of making an optimal choice in the development of a consistency verification tool for CMS metadata which ultimately is our objective.

Acknowledgement

I would like to express my gratitude to my mentors Kate, Nicolò and Valentin for their patience, help and guidance that helped me immensely during the course of my project. I would also like to thank the CERN and the Openlab team for organizing the Openlab summer student program, the visits to Open systems, ETH Zurich and EPFL and for giving me this wonderful opportunity to work and learn at CERN. A special thanks to the IT-DB team and my colleagues in room 8-1/1016 who were always there to help me.

Bibliography

1. http://iopscience.iop.org/article/10.1088/1742-6596/513/4/042052/pdf
2. http://cms.web.cern.ch/news/triggering-and-data-acquisition
3. https://github.com/aerophile/DBConsistencyCheck/blob/master/Direction.md
4. https://github.com/aerophile/DBConsistencyCheck/blob/master/python_files/csv_gen.py
5. https://github.com/aerophile/DBConsistencyCheck/blob/master/README.md