European Organization for Nuclear Research

Project Report:
Development of cleaning tools for BookKeeping database

Nino Pelov
University of Sofia, Bulgaria

Main Supervisor : Joel Closier
Substitute Supervisor : Markus Frank

Abstract

BookKeeping database is used for storing meta-data, describing each step of the processing of data in the LHCb experiment. This data is populated from the production system and sometimes the population fails. These failures cause inconsistencies in the BookKeeping database. In this project, I have analyzed these inconsistencies and I have developed tools for fixing some of them.

Table of Contents

1. BookKeeping Database Model

BookKeeping database is used for storing meta-data, describing each step of the processing of data in the LHCb experiment. The meta-data for each step consists of input files, output files and step related parameters. A step is typically running an application and in the terms of the BookKeeping database is referred to as a job. Because of the asynchronous data population from the production system, there are duplications of jobs and their related files. The goal of my project is analyzing the data duplications and development of tool for fixing them.

The main entities in the BookKeeping database are jobs and files. The relation between them has the following constraints:

- Every file is the output of unique job.
- A file can be input of many jobs.
- A job can have several files as input and several files as output.

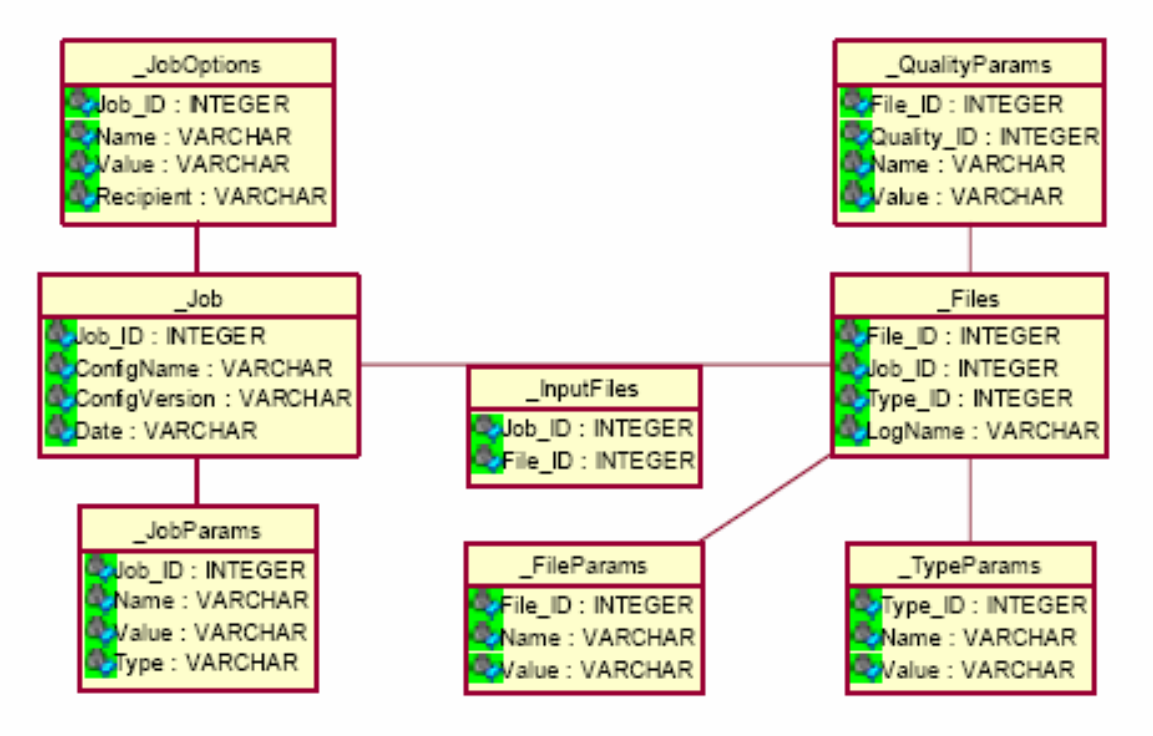


Figure 1: **BookKeeping database model**

Each file and each job has an incremental, system generated unique ID. Each job and each file is described by a set of parameters, stored in the corresponding Params tables. The main parameters, which we use to recognize duplication of a job or a file, are the “name” parameters. The name of each job is stored in the Value column of the JobParams table with parameter name ‘Name’. The name of each file is stored in the LogName column of the Files table.

2. Data duplication patterns

The first task in my project was to find patterns of duplications of jobs and files. I found that all duplicated files (files, with same logical name) are output files of duplicated jobs, which means that if we clean duplicated jobs and their output files, there will not be any duplicated files left. Also I found that there are two major groups of duplications, corresponding to the two chains of the data processing at LHCb – chain A (Figure 2) and chain B (Figure 3).

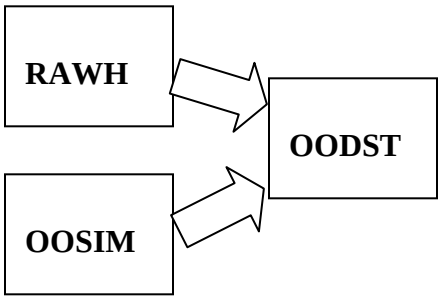


Figure 2: Chain A

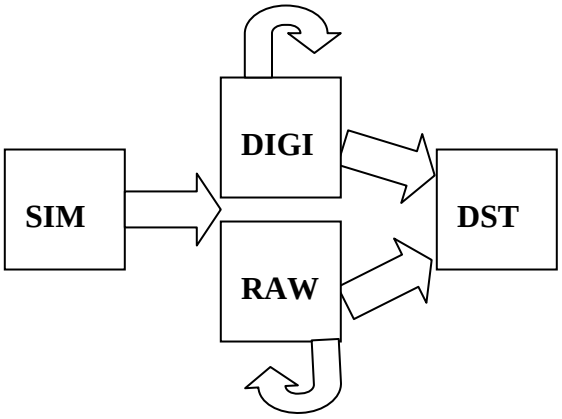


Figure 3: Chain B

2.1. Job duplications of chain A of data processing

The most common pattern of job duplications of chain A of data processing is:

- The jobs are duplicated three times.
- The second and third job of a triplet, have same location parameter value and it is different from the first job location.(the first job is the job with lowest ID)

An example of this pattern is Figure 4.

Further investigation of job duplications of chain A of data processing was stopped, because of the soon deletion of all data related to chain A of data processing and the impossibility to put all job duplications in a pattern.

Figure 4: An example of the most common pattern of job duplication of chain A of data processing

2.2. Job duplications of chain B of data processing

The pattern of job duplications of chain B of data processing is shown on Figure 5. It has the following constraints:

- Jobs are duplicated twice.
- All job parameters of a pair of job duplication are equal.
- Duplicated jobs have the same input files (same file IDs).
- Duplicated jobs have the same output files (same logical names, but different file IDs) or the set of output files of one of the jobs of the pair of duplication is subset of the set of output files of the other of the jobs of the pair of duplication.

Figure 5 describes also the actions for deleting one of the duplicated jobs:

- Objects in green are these, which we are going to save.
- Objects in red are these, which we are going to delete.
- In this case the second job has one more file than the first job and it has to be redirected to the first job – the intersected arrow.

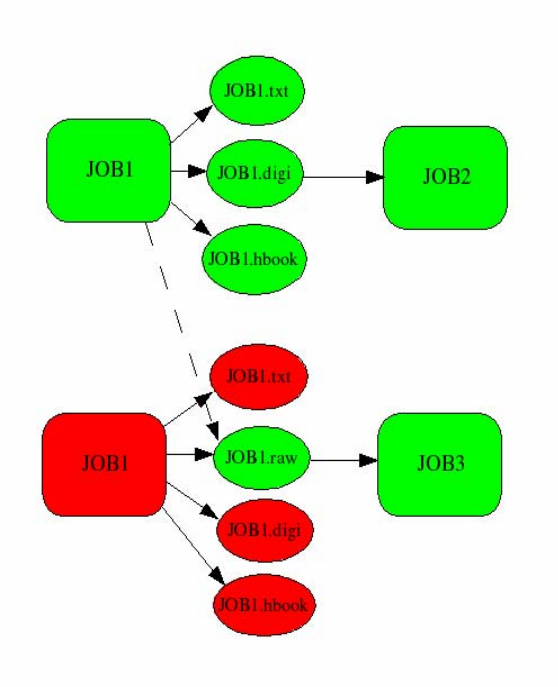


Figure 5

3. Implementation

The tools I have developed are written in Java, using JDBC technology for connection to the BookKeeping database. The development environment I have used is:

- OS – CERN Scientific Linux;
- RDBMS – Oracle 10g;
- Database client – TOra;
- IDE – Eclipse;
- Programming languages – Java, SQL;

The tools are in a single java package **bkcleaner** published on CVS at /LHCb/DataMgmt/Bookkeeping/src/bkcleaner directory.
The JavaDoc documentation is published on CVS at http://lbnts3:8100/doc/index.html .

3.1. BKDumper

BKDumper is utility, which I developed for searching of patterns of duplications. It dumps related information to a specified file in an ASCII file. It is part of the bkcleaner package.

3.2. BKCleaner

BKCleaner is utility for cleaning BookKeeping database of duplications, recognized by the pattern described in point 2.2. It is part of the bkcleaner package.

4. User Guide

Both tools BKDumper and BKCleaner are developed in Java, and you need JRE to execute them. Both tools use Oracle JDBC driver. Before application executing:

- Set CLASSPATH environment variable to include the paths to Oracle JDBC driver jar archive and bkcleaner java package with the compiled java classes.
  Example:
  setenv CLASSPATH /home/nino/BKtools:/home/nino/lib/ojdbc14.jar

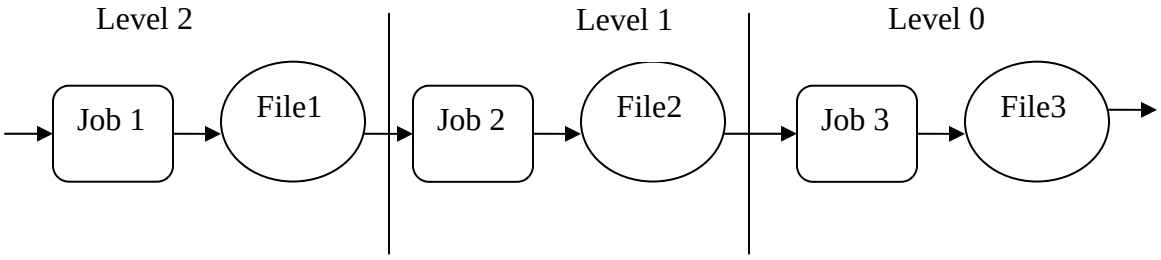
- Check your Oracle Client configuration for the BookKeeping database
  Examples:
  tnsping LHCBR
  sqlplus lhcb_bookeeping/password@LHCBR

4.1. BKDumper

BKDumper is a command line java utility, which dumps data related to specified file in an ASCII text file with name FILEID.dmp in the current directory.It is executed by the following command:

java bkcleaner.BKDumper FILEID LEVEL JDBC_URL

FILEID is the file ID of the file for dumping.
LEVEL is 0 or possitive integer value for the level of dumping. It means the levels of a chain of jobs.



0 - dumps file parameters of the specified file and parameters of the job, which has produced the file.

1 - dumps file parameters of the specified file and parameters of the job, which has produced the file and the same data for all input files of the job
2 – same as 1 + one more level
and so on…

JDBC_URL is a full JDBC URL describing the BookKeeping database connection:
jdbc:oracle:thin:<user>/<password>@//<host>:<port>/<service_name>

4.2. BKCleaner

BKCleaner is a command line java utility, which cleans specified BookKeeping account of duplicated jobs recognized by the pattern described in point 2.2. It needs a repository, which consists of two tables:

BC_DUPL_JOBS – used for temporary storing of duplicated jobs, when clean action is executed. It is truncated every time a new clean action is executed.

BC_REPLICAS_LOG – used for logging deleted replicas, needed for further physical deletion of the files deleted from the BookKeeping database. It is never cleaned by the BKCleaner utility.

The command for executing BKCleaner is:

java bkcleaner.BKCleanerStart ACTION JDBC_URL

ACTION:

init - Creates repository tables, needed for the cleaning action.It has to be done once per bookkeeping account.

clean - Cleans BookKeeping database of duplicated jobs and files.

JDBC_URL is a full JDBC URL describing the BookKeeping database connection:
jdbc:oracle:thin:<user>/<password>@//<host>:<port>/<service_name>

5. Appendix A – created files

The source code files are published on CVS at http://isscvs.cern.ch/cgi-bin/cvsweb.cgi/DataMgmt/Bookkeeping/src/bkcleaner/?cvsroot=lhcb

BKCleaner.java	Contains the basic class of the BKCleaner tool
BKCleanerStart.java	Contains the entry point of the BKCleaner tool
BKDumper.java	Contains the basic class of the BKDumper tool
BKCleanerRepository.sql	Contains SQL statements for the repository creation

The JavaDoc documentation of the bkcleaner package is available at http://lbnts3:8100/doc/index.html