Lab 3 – WS2024/25

Legacy Software Engineering with PL/1 and COBOL

Stefan Strobl, Mario Pilz

December 20, 2024

1 Introduction

This lab illustrates a typical batch job programming task. You will develop a simple yet realistic record processing application the mainframe.

Please note that the lab is intended to be solved individually.

2 Scenario

A company has a big data set of tax information records that is regularly updated. The data should be backed up regularly so that the state of the file at certain points in time could be reproduced. However, due to the size of the data set – it contains on the order 100 million records – copying the whole set for the backup every time is not particularly storage efficient. Instead, the differences between the current contents of the file and the state when it was last backed up should be written to a difference data set.

3 Exercise

It is your task to write the program that produces the differences data set for the backup purposes. The program should expect the two versions of the file by the DD names OLDDATA and NEWDATA and write the difference data to the file with DD name DIFFDATA.

The tax record data sets use fixed size records as described in Figure 1. The record size is 1028 bytes. Each record in the difference data set should describe the changes to a record between the two version of the production data set. To be able to use the difference data to restore the state of the data both forward and backward, the difference record should include in what way the tax record was changed (newly created, deleted or updated) and a copy of both the old and new versions of the record. The structure should look like this:

```
DECLARE 1 TAXRECORD UNALIGNED,
          3 META,
            5 TAXID FIXED DECIMAL(9,0),
            5 STATUS CHAR(1),
            5 LAST_UPDATED,
              10 UPDATE_DATE CHAR(6),
              10 UPDATE_TIME CHAR(9),
          3 BALANCE FIXED DECIMAL(13,2),
          3 NOTES(10) CHAR(100);
```

Figure 1: Tax record definition in PL/1 syntax

- One character as the indicator for the kind of change,
- then a 19 character field reserved for future use,
- a copy of the old version of the record (blank if it was newly created),
- and a copy of the new version of the record (blank if it was deleted).

The size of one difference record is thus 2076 bytes. Unchanged records should not be included in the difference data set.

The TAXID field acts as an identifier and should be used to determine if a record in the old and new data set logically refer to the same tax record. The records in the production data sets are also sorted by this field; the same should be the case for the produced difference data set.

In addition to the difference data set, the program should produce a readable listing of the data on the printer (SYSPRINT for PL/1, SYSOUT for COBOL). For newly created and updated records the new version should be written to the printer, for deleted records the old version should be printed. The printed form of the record should be human readable and should include field labels, so that it reads similar to a form. In PL/1, the formatting can be done with *edit-directed I/O*, which provides more control over the format of the output than list-directed I/O. The entries of the NOTES substructure should be numbered; a blank note should not be printed at all. You may assume that any note string, that starts with a null byte (character code 0), is blank.

4 Additional Material and Information

The following additional files are provided to you:

- SAMPDAT.pl1, a PL/1 program that generates sample data for an old and new version of the tax data for testing purposes. Expected by the provided JCL to reside <USER>.LAB3.PL1.

- SAMPCLG.jcl, the JCL to compile and run SAMPDAT. Expects the PL/1 source in an partitioned data set <USER>.LAB3.PL1 and allocates the sample data sets as <USER>.LAB3.OLDDATA and <USER>.LAB3.NEWDATA, where <USER> is the name of the user that executes the script.
- TAXREC.pl1, containing the PL/1 definition of the tax record, except for the top level declaration. Used in SAMPDAT by way of inclusion through the macro preprocessor. Expected by the provided JCL to reside <USER>.LAB3.PL1.
- TAXREC.cobol, containing the COBOL definition of the tax record structure. If you solve this assignment with COBOL, you can copy this into your source file. However, if you need more than one instance of this code in the same program, you will have to change the names once. Alternatively, using a COPY statement—optionally with the REPLACING clause to change the name of the record definiton—can be used, similar to the preprocessor-include in PL/1.

File name extensions are only informational and are meant to be removed before (or while) moving them into the mainframe.

Note For the inclusion of the record definition by the sample data generation PL/1 program to work, a certain limitation of the macro preprocessor of the PL/1 F compiler needs to be kept in mind: it does not support data sets with a block size of more than 5 times the record size. If the PL/1 source partitioned data set is allocated with a block size of more than 400, the macro processor will abort with a cryptic error message and compilation will *not work*. We therefore recommend that you use a block size of 400 for the PL/1 source data set.

5 Submission

The submission is due **on Friday 2025-01-25 at 23:59** via TUWEL.

Please submit the following items in an archive, either Zip or Tar:

- The source code of your implementation (including all complementary information and files, such as JCL)
- A sample execution of your program, consisting of
  - Input parameters (please select non-trivial input)
  - All output. Please separate the output of your program from the remaining job output.
- A document explaining your solution in either English or German language. Document at least:
  - the names and purposes of the required data sets

  - an explanation how your program solves the posed problem
  - the steps you performed to execute the program and obtain the output

  The main focus of Lab 3 is to implement a program for a typical batch processing workload with record-oriented I/O. Therefore, your submission document should also focus on these aspects, in particular:

  - how the access to the files works in your program and what parameters are required;
  - how the JCL passes the datasets to the program and the required JCL statements;
  - how your differencing algorithm works.

- Any known issues or limitations you are aware of.

6 Tips

Do not hesitate to ask questions on the discussion forum or after lectures. If really stuck it is also possible to contact us via mail.

We also want to encourage you to engage in discussion with your colleagues.

If you encounter any restrictions that you think are due to the use of the (very) old compilers (as present in the TurnKey systems), please document as part of your solution.