



Marco Krapf (180934)

Seminar Paper

Can standard database tables
which are potentially suitable for
Process Mining
be detected (semi-)automatically
in the SAP ERP system?

Heilbronn University – Business Informatics
Seminar E-Business Applications – Winter Semester 2013/14
Prof. Dr. Sigurd Schacht

Executive Summary

In this research paper the general possibility of programming a «Time stamp crawler» for finding data base tables in the SAP ERP (Enterprise Resource Planning) system that are potentially suitable for Process Mining is being discussed. A first version of the program in the SAP-owned programming language *ABAP* has been developed which examines tables in search of time stamps, patterns and textual hints. Starting from an initial data base table typed in by the user, the program checks this table for related tables by searching for foreign key tables and then loops at all those tables found one by one. As a result, the crawler displays a detailed analysis of each table as well as a summary of all potentially suitable tables at the end of the searching process.

A direct comparison of the table results identified by the crawler with tables found «manually» is rather difficult since currently finding an initial table for feeding into the crawler still requires prerequisites and concepts of the «manual searching approach».

All in all, the aim of the research work has been achieved with a significantly faster search for time stamps in data base tables. However, further optimization of the crawling process is possible and necessary.

Table of Contents

Executive Summary	I
Table of Contents	II
Table of Figures	III
1 Introduction to Process Mining with SAP ERP	1
1.1 Motivation	1
1.2 Requirements for Process Mining	2
2 Related work	3
3 Searching for tables in SAP ERP	4
3.1 General procedure	4
3.2 Suitable Tables	6
3.3 Manual searching approach	7
3.4 ABAP Report for finding tables	11
4 Limitations	15
5 Gaps for further research	16
6 Conclusions and recommendations	18
References	19
Appendix	20
Tables examined by the ABAP crawler	20
Criteria to be checked by the crawler	23
Date stamp criteria (currently not in use)	25
Source code of the ABAP Process Mining Crawler	26

Table of Figures

Figure 1: SAP Easy Access menu	4
Figure 2: SQL Trace Protocol of a transaction (excerpt)	5
Figure 3: SAP Help / Technical Information	5
Figure 4: SQL Trace Record containing an INSERT command	8
Figure 5: ABAP Dictionary	8
Figure 6: Data Browser - Table entries (excerpt)	9
Figure 7: ABAP Dictionary - Foreign Key Relationships	10
Figure 8: Data Browser - Incoming Foreign Key Relationships	10
Figure 9: SAP Table categories	12
Figure 10: ABAP Process Mining Crawler scheme	14
Figure 11: Crawler exploration depth	16

1 Introduction to Process Mining with SAP ERP

1.1 Motivation

The motivation for doing research in the field of Process Mining in the SAP ERP system has emerged while working on a research project concerning processes in the Human Capital Management (HCM) module for an audit company at Heilbronn University in summer 2013. While searching many days for relevant tables in SAP ERP that are suitable for the extraction of two different processes in the Human Capital Management module, I was thinking about writing a tool to automate or at least to facilitate this monotonous and time-consuming job. As SAP ERP consists of more than 400,000 standard tables of different categories (among them nearly 100,000 so-called *Transparent tables* which are plain data base tables of the SAP system), our main task was to find those few tables – about 10 – which we really needed to feed into a tool that finally visualises the processes.

Process Mining is currently a «hot topic» to be hit by enterprises, and the SAP ERP software is wide-spread especially across big companies, gaining permanently additional market share. An animated visualisation of the processes (*as-is*, i.e. how they *really* proceed) reveals deviations from the standard, unnecessary rework and bottlenecks in business practice. According to van der Aalst (2011), despite the fact that multitudes of events are recorded by today's information systems organizations still have problems extracting value from these data¹.

Even though the SAP ERP system consistently records all process stages that are executed it is rather difficult to backtrack the processes needed for Process Mining. The SAP system has grown over decades and is not designed for process analysis. Information is often spread over many tables, and time stamps that are needed for process analysis can be hidden *anywhere*. The concept of data storage and the thoughts of SAP's data base designers are hard to understand. However, the data base table names have remained identical over all releases of the software. Provided this company policy will be maintained (and in my opinion that can be safe to assume) it is surely worth doing research in Process Mining with SAP.

¹ (van der Aalst, Process Mining - Discovery, Conformance and Enhancement of Business Processes, 2011)

1.2 Requirements for Process Mining

Existing IT records are the basis for Process Mining. For creating an event log which is the «starting point» for the mining process, at least the following attributes of an event need to be available: an activity, a case ID and a time stamp. An activity is a «well-defined step in some process» and is related to a particular case (i.e. a process instance). The events belonging to a case are ordered using the time stamp. Event logs may even store additional information such as resources (e.g. persons or devices)².

Anne Rozinat states that the data needed for Process Mining «is already there» and accumulates as a «by-product» of the increasing automation and digitization of business processes, so it does not have to be collected before starting the mining process³.

² (van der Aalst, Process Mining: Overview and Opportunities, 2012)

³ (Rozinat, 2012)

2 Related work

Martijn van Giessel (2004): «Process Mining in SAP R/3 – A method for applying process mining to SAP R/3»

The first research in this area was done by Martijn van Giessel who wrote a master thesis on Process Mining in SAP R/3. The paper was submitted to the Eindhoven University of Technology in 2004. First supervisor was Wil van der Aalst. The approach is based on the *SAP R/3 Reference Model* which contains the «business blueprint» of an organization. The model consists of four views which together represent business processes. One of them, the *business object model* view, contains all business objects. As these business objects are related to tables, they can be used to find relevant tables. For extracting appropriate data base tables, van Giessel developed a tool coded in Visual Basic for Applications programming language. Using Microsoft Excel, he sorts and links the tables *manually* for constructing *event logs*. Van Giessel recommends finding ways for automating this process. Due to the fact that in newer versions of SAP (e.g. ERP 6.0) reference models are no longer included, new options have to be explored. Nevertheless, this paper serves as a basic for other research work done in Process Mining with SAP.

David Piessens (2011): «Event Log Extraction from SAP ECC 6.0»

This master thesis was also submitted to the Eindhoven University of Technology, however some years later, in 2011. Piessens proposes a method that guides in extracting event logs from SAP ECC 6.0. In this method he identifies two phases: first a repository for each SAP process is being prepared and configured, and in the second stage the event log is being extracted. With a method Piessens calls *table-case mapping* an event log from a set of activities is being computed automatically based on foreign keys that exist between tables in SAP.

As a topic for future work Piessens suggests an automated discovery of events by checking for patterns focussing on time stamps in the SAP database. As a similar method, he mentions performing an SQL trace during the execution of an activity in SAP with an in-depth analysis of the trace protocol.

3 Searching for tables in SAP ERP

3.1 General procedure

First of all, the process to be mined (e.g. order-to-cash) has to be understood. If the implementation of the process in the SAP ERP system is not well-known (which in practise will probably be the standard case) the relevant transactions can either be found in literature or web resources about or from SAP or by navigating through the tree structure in the SAP Easy Access menu. As a rule, the transactions are ordered for the most part along the process chain (see Figure 1).

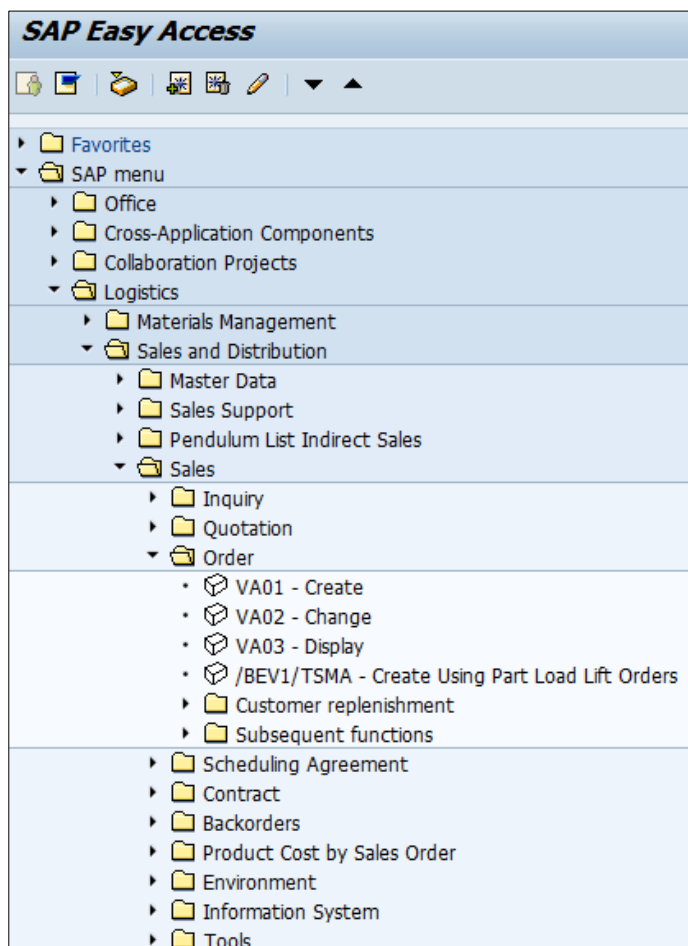


Figure 1: SAP Easy Access menu

After that, the underlying database tables of those transactions need to be detected, which is the most difficult and time-consuming part of the work. There are several different ways to find the relevant SAP ERP database tables, the most promising seem to be the following:

- Searching in published lists and web resources with SAP database table names and their respective descriptions
- Executing a transaction while the System Trace (transaction ST01) is switched on with the field *DB Access (SQL Trace)* checked. The trace protocol shows all objects (i.e. database tables) affected by this transaction (see Figure 2).
- Getting the table name from the *Technical Information* window of the *Performance Assistant* when hitting the *Help* icon or pressing the *F1* key with the cursor placed on an input field or an input description which is connected to a database table (see the example in Figure 3, transaction VA01 (Create Sales Order), input field *Order Type*).

Trace Display				
Client: 904 User: DEVB-41 Transaction VA01				
Work Process 1 PID Date: 30.12.2013				
First Block of Dialog Step Last Block in Dialog Step				
Block Version: 16804 No. of Records: 22 File				
hh:mm:ss:ms	Type	Lasts(us)	Object	Text
16:44:51:796	SQL	3322	TOBJ_OFF	Prog: SAPMV45A Row:
16:44:51:800	SQL	229	TOBJ_OFF	Prog: SAPMV45A Row:
16:44:51:800	SQL	31	TOBJ_OFF	Prog: SAPMV45A Row:
16:44:51:800	SQL	12	TOBJ_OFF	Prog: SAPMV45A Row:
16:44:51:802	SQL	401	T185	Prog: SAPLV00F Row:
16:44:51:803	SQL	32	T185	Prog: SAPLV00F Row:
16:44:51:803	SQL	12	T185	Prog: SAPLV00F Row:
16:44:51:803	SQL	173	T185	Prog: SAPLV00F Row:
16:44:51:803	SQL	14	T185	Prog: SAPLV00F Row:
16:44:51:803	SQL	7	T185	Prog: SAPLV00F Row:
16:44:51:803	SQL	2705	T185V	Prog: SAPLV00F Row:
16:44:51:806	SQL	276	T185V	Prog: SAPLV00F Row:
16:44:51:806	SQL	84	T185V	Prog: SAPLV00F Row:
16:44:51:806	SQL	2	T185V	Prog: SAPLV00F Row:
16:44:51:806	SQL	11	T185V	Prog: SAPLV00F Row:
16:44:51:809	SQL	440	D347T	Prog: SAPLV45C Row:
16:44:51:809	SQL	25	D347T	Prog: SAPLV45C Row:

Figure 2: SQL Trace Protocol of a transaction (excerpt)

Technical Information	
Screen Data	
Program Name	SAPMV45A
Screen Number	0102
GUI Data	
Program Name	SAPMV45B
Status	A02
Field Data	
Table Name	VBAK
Table category	Transparent table
Field Name	VBELN
Search Help	VMVA
Data Element	VBELN_VA
DE Supplement	0
Parameter ID	AUN
Field Description for Batch Input	
Screen Field	VBAK-VBELN
<input checked="" type="checkbox"/> Navigate <input type="checkbox"/>	

Figure 3: SAP Help / Technical Information

3.2 Suitable Tables

The main challenge is to find a table containing a timestamp, either as a single database entry or somehow hidden respectively encoded in a string for *each* of the transactions (i.e. for each process stage) so that the events of the whole process can be completely characterized by IDs, activities and time records.

Hence, a table can only be used for Process Mining when it contains information about the point of time the transaction was executed, in the best case even down to seconds. It is not sufficient to have only a date stamp because some stages in the process could be performed on the same day, so it is not possible to order the activities correctly in time series.

Ideally, the database table includes a column of the *Data Type TIMS* which consists of the predefined elementary ABAP Type⁴ T (Time field) in the format *HHMMSS*, i.e. the field length is six characters with the first two digits representing hours, the following two digits standing for minutes and the last two digits for seconds. The time can be stored either in a human interpretable format or as an inverted time (e.g. 81:77:89) which has technical reasons that are owed to older data base systems. To compute the real time the inverted time has to be subtracted from 999999 (i.e. $999999 - 817789 = 182210$, so the real time is 18:22:10).

Indeed, this data type is being used for different purposes, so the suitability for an event log has to be identified from case to case. Some transactions overwrite the entry, in this case only the last update of the record can be traced back. However, for creating a full event log over time series all entries of the time stamp are needed.

Often a column with a time field is missing, nevertheless the time can be «hidden» in a text field with alphanumeric characters (ABAP type C) or in a string. The entry could look like *A000010520003201308071604550001* for example. When examining this character string it proves that the sequence *160455* is the time (while *20130807* is the date), so this transaction was recorded on August 7th (or maybe July 8th?), 2013 at 16:04 and 55 seconds. This string just serves as an example, the sequence can differ in every table, and often there is no rationale why all this information has been recorded in one single fields and how it is composed.

⁴ (SAP AG (1), 2012)

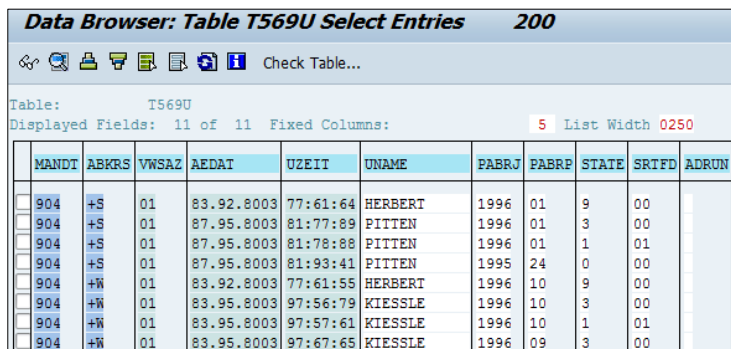
3.3 Manual searching approach

When doing this job for the first time, our research team did not have any clue how to start searching for tables, and on top of it we did not have much understanding of the SAP HCM module and its processes. However, we had some experience in working with SAP in other scenarios.

So, after having understood which transactions are likely to be involved in a standard implementation of the process to be mined, the procedure starts with tracing the execution of one single transaction for only a few seconds just before the moment when data typed into the input fields will be saved. The SAP transaction code for starting the *System trace* program is *ST01*. It seems to be sufficient (or even advisable) to tick only the *DB Access (SQL Trace)* checkbox before switching on the trace, so all other activities are excluded from the trace protocol. Dependent of the transaction and the time span recorded, the protocol (see Figure 2) can easily comprise several ten thousands (!) entries to be scrolled through manually, so the record time should be limited to the absolute maximum that is necessary for including those records which affect a new database entry or a change of an existing entry. There is no way to filtering the protocol in the SAP system; however it can be exported to a spreadsheet analysis like Microsoft Excel for further processing. As a shortcoming of the export, the data of all columns (hh:mm:ss:ms | Type | Lasts(us) | Object | Text) are aggregated into one single cell for each record. Nevertheless, the columns can be split by using the Excel function *Text to columns* with the delimiter «|». Then, a filter or a function can be applied for finding specific objects.

When examining the trace protocol, especially tables with SQL commands containing *INSERT* and *UPDATE* statements are of major interest (see Figure 4). Unfortunately, for opening the detailed record information that displays the SQL statement, each line of the trace record list has to be opened separately by double-clicking the entry. Moreover, it turned out that only very few records of the protocol contain an *INSERT* or an *UPDATE* statement while the utmost part sends *SELECT* statements to getting some information from other tables, or they even have no statement at all that is appropriate for being understood by human beings. Experience shows that detecting the required tables is rather like finding a needle in a haystack. Hits are mostly scored by fortunate coincidence.

For seeing the actual data records, the icon *Contents* or the keystroke combination *Ctrl+Shift+F10* can be used which leads to transaction *SE11_OLD* and has the same effect as transaction *SE16* (Data Browser). The underlying ABAP program of both transactions is equivalent. Usually an execution of this transaction without modifying any selection parameters leads to a proper result displaying a table with the first 200 entries. This table can then be screened manually for time entries (see Figure 6, column *UZEIT* contains inverted time stamps). In case of missing columns the value of the input field *Width of Output List* at the bottom of the selection screen needs to be increased.



	MANDT	ABKRS	VWSAZ	AEDAT	UZEIT	UNAME	FABRJ	FABRP	STATE	SRIFD	ADRUN
<input type="checkbox"/>	904	+S	01	83.92.8003	77:61:64	HERBERT	1996	01	9	00	
<input type="checkbox"/>	904	+S	01	87.95.8003	81:77:89	PITTEN	1996	01	3	00	
<input type="checkbox"/>	904	+S	01	87.95.8003	81:78:88	PITTEN	1996	01	1	01	
<input type="checkbox"/>	904	+S	01	87.95.8003	81:93:41	PITTEN	1995	24	0	00	
<input type="checkbox"/>	904	+W	01	83.92.8003	77:61:55	HERBERT	1996	10	9	00	
<input type="checkbox"/>	904	+W	01	83.95.8003	97:56:79	KIESSLE	1996	10	3	00	
<input type="checkbox"/>	904	+W	01	83.95.8003	97:57:61	KIESSLE	1996	10	1	01	
<input type="checkbox"/>	904	+W	01	83.95.8003	97:67:65	KIESSLE	1996	09	3	00	

Figure 6: Data Browser - Table entries (excerpt)

For verifying the correctness of expected effects in the database concerning the time stamp, a transaction of the process under investigation which inserts a new record into this table can be executed, and the consequences can immediately be observed in a second session, i.e. by parallel having started another SAP instance. Alternatively, the newly created entry can be found by adjusting the selection parameters of transaction *SE16*. Further information about the meaning of the fields can be obtained by placing the cursor on the labels in the selection screen and pressing either the *Help* icon or the function key *F1*.

Due to the fact that information to be stored when executing a transaction is often widely spread in the data base of the SAP system, not only the current data base table should be taken into consideration, but also related tables. The first step could be done by taking a closer look at all tables which are directly connected with foreign keys. There are two ways of displaying the foreign key tables, both graphically and as a table.

In transaction *SE11* (ABAB Dictionary), by hitting the *Graphic* icon or pressing the keystroke combination *Ctrl+Shift+F11* in the *Dictionary: Display Table* screen a window

Both notations do not contain only the table names, but also a short description of the tables. A slight advantage of the graphical notation is that each foreign key table is depicted only once, whereas the list contains duplicates. However, the list is a bit easier to read.

Dictionary: Foreign Key Relationships

Dictionary objects Edit Goto Utilities Settings View Help

224

Display Area

THOCI
Public holiday calendar in

1:1:N
KEY

T553A
Definition Rules for Day T

1:1:N
KEY

1:1:N
KEY

T551A
Period Work Schedules

1:1:N
REF

T000
Clients

T508Z
Assignment of PB Grouping

1:1:N
KEY

T508A
Work Schedule Rules

Check Table	Short Text
KUGKI	
T508A	Work Schedule Rules
MANDT	
T000	Clients
T508A	Work Schedule Rules
T508Z	Assignment of PS Grouping for Work Sched
T551A	Period Work Schedules
T553A	Definition Rules for Day Types
MOFID	
T508A	Work Schedule Rules
THOCI	Public holiday calendar index
MOSID	
T508A	Work Schedule Rules
T508Z	Assignment of PS Grouping for Work Sched
MOTPR	
T551A	Period Work Schedules
MOTTY	
T553A	Definition Rules for Day Types
ZEITY	
T508A	Work Schedule Rules
ZMODN	
T551A	Period Work Schedules

Figure 7: ABAP Dictionary - Foreign Key Relationships Figure 8: Data Browser - Incoming Foreign Key Relationships

10

3.4 ABAP Report for finding tables

Based on the insights gained from the manual searching approach, the author of this paper has developed a tool to partially automate the database table finding process. The program is written in ABAP and consists of one single ABAP Report. Its job is to «crawl» through the SAP standard data base tables in search of tables that are suitable for the Process Mining task.

As described in chapter 3.1 the user as a precondition needs to have a general understanding of the process and furthermore he has to detect a database table that concerns a stage of this process which then serves as a «starting point» for the crawler.

The crawler automatically searches for time stamps in the table that was typed in as well as in its related tables. Fortunately, there are some standard tables in SAP ERP which can be used for getting information about all tables that exist in the system. These tables contain meta data about all data base tables stored in the whole system, so they are ideally suitable for the automatic search. The tables needed for this purpose are *DD02L*, *DD02T* and *DD03L*. They can be accessed in ABAP through SQL commands and contain all relevant information the crawler needs, except for the content of the tables to be examined themselves. For gaining an understanding of these tables they can be looked up manually in the *ABAP Dictionary* (transaction SE11).

Table *DD02L* is called «SAP Tables» and comprises all data base tables of the SAP ERP system. First of all, the crawler checks whether the initial table typed in by the user is a valid SAP database table. If possible, the name of the table name (TABNAME) and the table category (TABCLASS) are read. There are six table categories (see Figure 9), the most relevant of them is *Transparent table*. According to the SAP documentation, «all data of commercial relevance is stored exclusively in transparent tables». Nevertheless, also cluster tables could be of interest as the table cluster by definition has a field *Timestamp*⁵. Concerning the suitability of cluster tables in the context of Process Mining deeper research has to be done (see chapter 5 Gaps for further research). The table category found by *DD02L* is not taken into consideration when assessing the potential suitability. In the current version of the crawler it is only used for displaying the category on the output screen.

⁵ (SAP AG (2), 2012)

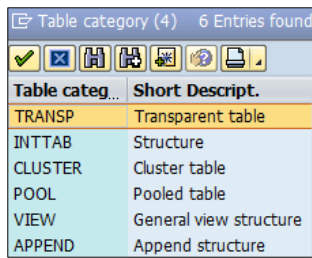
A screenshot of a SAP dialog box titled 'Table category (4) 6 Entries found'. It contains a table with two columns: 'Table categ...' and 'Short Descript.'. The table lists six categories: TRANSP (Transparent table), INTTAB (Structure), CLUSTER (Cluster table), POOL (Pooled table), VIEW (General view structure), and APPEND (Append structure). The first row is highlighted in yellow.

Table categ...	Short Descript.
TRANSP	Transparent table
INTTAB	Structure
CLUSTER	Cluster table
POOL	Pooled table
VIEW	General view structure
APPEND	Append structure

Figure 9: SAP Table categories

The only interesting information taken from table *DD02T* («SAP DD: SAP Table Texts») is a short textual description of the table which is stored in the field *DDTEXT*. This text is language dependant and can be adapted by changing the *DDLLANGUAGE* code in the corresponding SQL SELECT statement of the ABAP report. Just as the table category, this information is currently not used for decision making purposes. In a later version of the crawler, it could be screened for patterns that indicate the relevance of the table for the examined process.

Significantly more information can be extracted from table *DD03L* («Table Fields»). It serves as the central table for getting meta information with regard to time stamps. The most important fields of *DD03L* seem to be *FIELDNAME*, *DOMNAME* and *DATATYPE*. Additionally, the fields *CHECKTABLE* and *PRECFIELD* can be used for feeding the crawler's «table repository», i.e. an internal table that contains all tables waiting to be explored.

First, the program loops at the field names of the current table in quest of patterns like **TIME** and **ZEIT** to find hints of any time information recorded by checking the column labels. This could be useful in case of the column's data type not being «TIMS» but some other type like a character string.

Then, the domain is being checked for both known domain names used in the SAP system (e.g. «UZEIT») and patterns as mentioned before that contain character sequences which indicate a time stamp.

Thirdly, each column of the table is being checked in search of the data type «TIMS». Apparently, this seems to be the most reliable way to finding time stamps. The length of this data type is 6 digits. The value can be stored in regular or inverted time format. An inverted value can easily be converted during the log extraction (which is not part of this paper) as described in chapter 3.2. Unfortunately, by far not every time information is evident and simply stored in an own column of this data type. It could even be hidden

anywhere in the data base, which is addressed later when covering a «deep exploration» of the table content.

Next, the crawler loops at the column which contains the foreign key tables and appends them to a «repository table». All entries of this table serve as input tables for the whole exploration process as described in this chapter. They are processed one by one after the run of the current table has been completed. As all data in the SAP ERP system is stored in a relational database⁶, it needs to be ensured that each table is being processed only once, so the crawler has to check whether the foreign key table has already been stored in the repository or it has even be processed before. If both checks are negative, the table will be appended to the repository.

The last step in table *DD03L* is to checking the table for «Includes»⁷. The table that is currently examined can be composed of its own columns and of other structures that are included. These *Includes* could possibly be of interest, however further research concerning this question has to be done. The current version of the crawler just lists the names of the Includes. They are not used for making any decisions.

After having finished all loops at the relevant fields of *DD03L*, the table itself which is currently under investigation will be checked for suitable content for Process Mining. This can be done by searching for patterns in the data itself like sequences that *could* be a «hidden» time stamp. The ABAP programming language supports the commands *FIND* and *REGEX* for finding occurrences of regular expressions in character sequences⁸. So the content of any data base table can for example be checked for a sequence of 6 numbers composed of the digits 0, 1 or 2 in the first position followed by a digit between 0 and 9 and two 2-digit sequences between 00 and 59 which *could* represent a time between 00:00:00 and 23:59:59. At this point of the ABAP program, any pattern that makes sense can be added to scenting hidden time information. Ideas for regular expressions can be tested by running the built-in tool *DEMO_REGEX_TOY* in transaction *SA38* (ABAP: Program Execution).

Based on the assumption that all data sets of a table have a similar structure, for performance reasons the crawler screens only the very first entry of each data base table.

⁶ (SAP AG (3), 2012)

⁷ (SAP AG (4), 2012)

⁸ (SAP AG (5), 2012)

Each exploration loop results in a detailed report of the table and an assessment whether the table could be suitable for Process Mining or not.

After having processed all tables of the repository or when having reached the third level of foreign key tables (i.e. the initial table's foreign key tables and the foreign key tables of each of them, see Figure 11) the crawler terminates and provides a summary of all potentially suitable tables with the table names and their respective short descriptions.

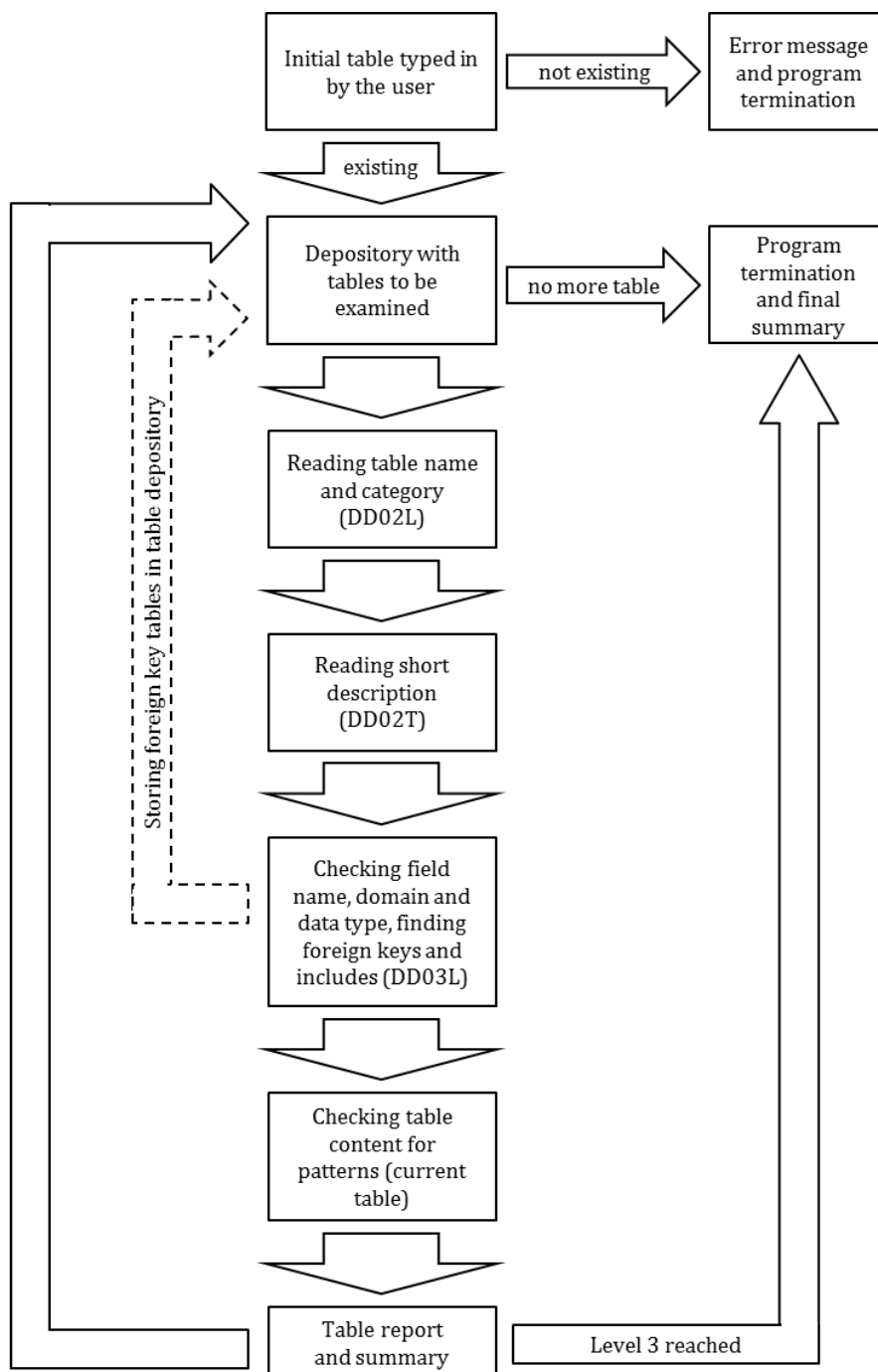


Figure 10: ABAP Process Mining Crawler scheme

4 Limitations

Most of the tables that are identified by the crawler as potentially suitable due to a character sequence fitting to a pattern do actually not contain a time stamp but anything else that «accidentally» looks like time information. In any case, the discovered sequence has to be looked up and assessed manually in the *Data Browser* (SE16) by the user. The tool gives just an indication of a potential hit.

Some important SAP data base tables seem to be only connected to table *T000* (Clients), so it is not possible to catch them by using the crawler.

Concerning the short description of the tables the program is currently restricted to German language. If these descriptions are to be included into the decision making process in multi-language usage of the SAP ERP system, the source code needs to be extended.

The «ABAP Process Mining Crawler» is designed for finding indications of *time* stamps. It is assumed that a *time stamp* is always accompanied by a *date stamp*. For creating event logs both stamps are urgently needed to ordering the extracted activities along the time line.

Moreover, the SAP ERP system does not create a time stamp for each and every transaction, therefore some process stages cannot be reconstructed completely. In that case a «worthwhile» table can only be tracked by activating the *rec/client* parameter⁹ in transaction *RZ11* for table logging and ticking the checkbox *Log data changes* in the data base's *Technical Settings*. In this way, all data base table changes are recorded in the table *DBTABLOG* and can be analysed¹⁰.

Finally, the current version of the ABAP Process Mining Crawler is restricted to three levels of exploration depth (see Figure 11). Later versions can be extended to diving deeper into the data base system up to an unlimited number of levels chosen by the user. In this case, the runtime performance of the program has to be considered.

⁹ (SAP AG (6), 2012)

¹⁰ (Bundesamt für Sicherheit in der Informationstechnik, 2013)

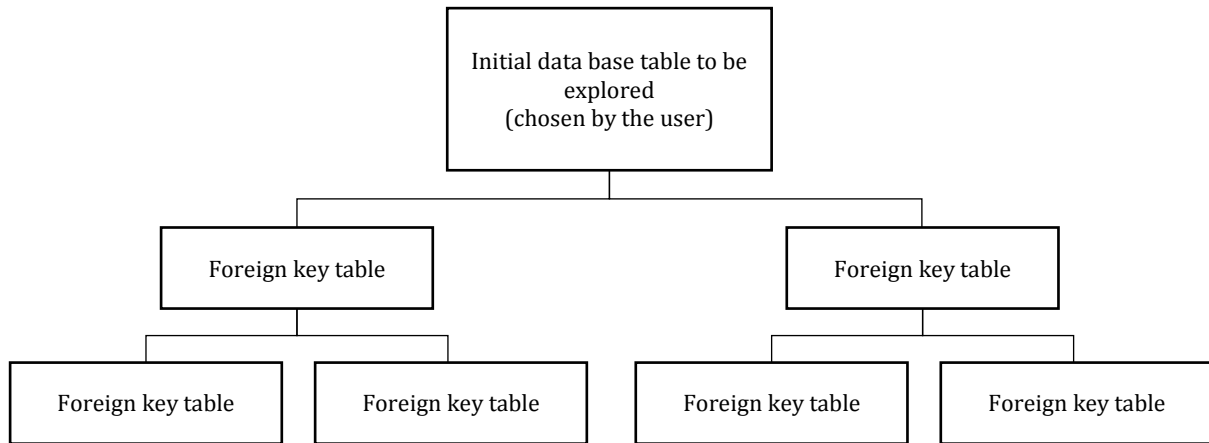


Figure 11: Crawler exploration depth

5 Gaps for further research

While working on this paper to finding an answer to the research question and especially while programming the ABAP Process Mining Crawler, a number of gaps for further research arose.

The program in the current version has been designed rather simple and has been coded in a single ABAP report. It demonstrates the general feasibility to automate a systematic table finding process in the SAP ERP system. The only modularisation technique which was applied is using subroutines called *FORM*. A comprehensive redesign of the software is recommended. More sophisticated input and output screens («Dynpros»), programming in *ABAP Objects* or even building *Web Dynpros* as well as the runtime performance should be taken into consideration.

Table Includes need to be verified for a general applicability in the context of Process Mining. If they are qualified at all, includes which are found while screening a data base table have to be added to the «table repository».

The summary at the end of the crawling process should be optimized. A ranking of the hits could be computed, so that the list of potentially suitable tables can be reported in descending order.

When screening the content of the tables, the code checks each field for appropriate patterns. The number of character sequences to be compared should be enhanced and more meaningful *REGEX* patterns have to be included, so that the number of «false hits» will be reduced.

The feasibility of automatically finding data base tables of a transaction which are affected by SQL *INSERT* and *UPDATE* statements should be explored. Maybe there is a chance for extracting all tables of each stage of a whole process. In this case, the user would just have to choose a process (e.g. order-to-cash) to receive a list of all relevant tables of the standard implementation of this process. These tables could be used as input tables for the crawling tool to verify the correctness and usability in the SAP ERP system he is currently working with.

A great way to getting all data base tables affected by the execution of a transaction is the *System Trace* protocol. There might be a chance for extracting the entries in the column *Object of Type «SQL»*. If this is possible, they can be used as input tables for the crawler.

The tool should be extended to the effect that more than one initial table can be filled in (which is also a precondition for the idea mentioned before).

It is not sure to what extent *Cluster Tables* are captured by the crawler. As there are obviously not that many cluster tables existing in the system, they can easily be examined by the tool one by one. In table *DD02L* (SAP Tables) the selection criterion *TABCLASS* can be limited to *CLUSTER* to receiving all cluster tables.

The table category («*TABCLASS*») is currently not sufficiently factored in and is only used for displaying the category on the output screen. There is also a field *SQLTAB* which contains the name of the corresponding table pool or table cluster for pooled and cluster tables. Maybe considering the entries of those fields can be utilised for diving deeper into the data base table system.

6 Conclusions and recommendations

The «ABAP Process Mining Crawler» was developed and tested with SAP ERP Central Component (ECC) 6.0 Enhancement Package 4 on NetWeaver platform version 7.01 and SAP GUI for Windows.

The research question can be answered insomuch that the objective of a semi-automatic detection of potentially suitable SAP data base tables for Process Mining has been achieved through programming a kind of crawler. If a process in the SAP ERP system is to be examined, the tool considerably facilitates the search for fitting tables.

Best results are reached by tracing the execution of a transaction (i.e. a process stage) and exporting the trace protocol to a Microsoft Excel spreadsheet as described in chapter 3.3. After having filtered out the different tables affected by this transaction, these are used as initial tables of the crawler.

Most of the tables found by the tool prove to be not required or even not suitable for creating an event log for the process to be mined. Furthermore, the selection of the fields to be extracted, the extraction itself, transformation and visualization of the process are not part of this paper. Only the (semi-)automatic detection of tables has been explored. Further research has to be done for integrating these findings in a larger workflow that at the end of the day mines and extracts SAP ERP processes fully automated.

References

- Bundesamt für Sicherheit in der Informationstechnik. (2013).
M 4.270 SAP Protokollierung.
Accessed on January 4, 2014 at the website IT-Grundschutz-Kataloge:
https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m04/m04270.html
- Piessens, D. A. (2011). *Event Log Extraction from SAP ECC 6.0*.
Technische Universiteit Eindhoven, Master Thesis.
- Rozinat, A. (2012). *Data Requirements for Process Mining*.
Accessed on December 29, 2013 at the website Flux Capacitor, the company weblog of Fluxicon: <http://fluxicon.com/blog/2012/02/data-requirements-for-process-mining/>
- SAP AG (1). (2012). *SAP Library*.
Accessed on December 31, 2013 at the website Types and Objects:
http://help.sap.com/saphelp_nw04/helpdata/en/fc/eb2fd9358411d1829f0000e829fbfe/content.htm
- SAP AG (2). (2012). *SAP Library*.
Accessed on January 2, 2014 at the website Pooled and Cluster Tables:
http://help.sap.com/saphelp_45b/helpdata/en/cf/21f083446011d189700000e8322d00/content.htm
- SAP AG (3). (2012). *SAP Library*.
Accessed on January 3, 2014 at the website Relational Databases:
http://help.sap.com/abapdocu_70/en/ABENDATABASE.htm
- SAP AG (4). (2012). *SAP Library*.
Accessed on January 3, 2014 at the website Includes:
http://help.sap.com/saphelp_nw73/helpdata/en/cf/21ea6a446011d189700000e8322d00/content.htm
- SAP AG (5). (2012). *SAP Library*.
Accessed on 3. January 2014 at the website Search Pattern:
http://help.sap.com/abapdocu_702/en/abenregex_search.htm
- SAP AG (6). (2012). *SAP Library*.
Accessed on January 3, 2013 at the website Setting the REC/CLIENT Parameter in the System Profile:
http://help.sap.com/saphelp_45b/helpdata/en/7e/c81ec852c511d182c50000e829fbfe/content.htm
- van der Aalst, W. M. (2011).
Process Mining - Discovery, Conformance and Enhancement of Business Processes.
Berlin, Heidelberg: Springer-Verlag.
- van der Aalst, W. M. (2012). Process Mining: Overview and Opportunities.
ACM Transactions on Management Information Systems, Volume 3, Issue 2, Article No. 7, July 2012.
- van Giessel, M. (2001). *Process Mining in SAP R/3*.
Technische Universiteit Eindhoven, Master Thesis.

Appendix

Tables examined by the ABAP crawler

DD02L (SAP Tables)

Field TABNAME (table name)

Name of a table or structure.

Use in the ABAP report: Input value is the current table to be examined.

Field TABCLASS (table category)

The table category defines whether a physical table exists for the logical table description defined in the ABAP Dictionary and how the table is stored on the database.

The following table types exist:

- TRANSP (Transparent table)
- CLUSTER (Cluster table)
- POOL (Pooled table)
- INTTAB (Structure)
- APPEND (Append structure)
- VIEW (Generated view structure)

Use in the ABAP report: Providing visual information on the output screen.

Field SQLTAB (name of an SQL table or an appended table)

This field contains the name of the corresponding table pool or table cluster for pooled and cluster tables. In an append structure, the field contains the name of the table or structure to which the append is assigned.

Use in the ABAP report: Currently not in use.

DD02T (SAP DD: SAP Table Texts)

Field TABNAME (table name)

Serves as a selection criterion.

Use in the ABAP report: Input value is the current table to be examined.

Field DDTEXT (short description of repository objects)

The short text is used as an explanatory text in documentation (F1 Help), and when lists are generated.

Use in the ABAP report: Providing visual information on the output screen.

Field DDLANGUAGE (language key)

The language key contains the abbreviation for the language in which a text is stored. It controls the language in which texts are output on the printer or on the screen.

Use in the ABAP report: For providing visual information on the output screen in German language (value «D»).

DD03L (Table Fields)

Field TABNAME (table name)

Serves as a selection criterion.

Use in the ABAP report: Input value is the current table to be examined.

Field FIELDNAME

Name of the table field.

The name of a field may only contain letters, digits, and the underscore, and it must begin with a letter. The field name can also begin with a namespace prefix.

Entries in the form .INCLUDE or .APPEND indicate that the field in question is not a regular one but rather an include or an append structure.

The name of the include or append structure is in this case recorded in field *Field type*.

*Use in the ABAP report: Checking for patterns like *ZEIT* and *TIM*.*

Field DOMNAME (domain name)

A domain describes the technical attributes of a field, such as the data type or the number of positions in a field. The domain defines primarily a value range describing the valid data values for the fields referring to this domain.

Different technical fields of the same type can be combined in a domain. Fields referring to the same domain are changed at the same time when a domain is changed. This ensures the consistency of these fields.

*Use in the ABAP report: Checking for patterns like *ZEIT* and *TIM* and comparing with well-known domain names.*

Field DATATYPE (data type in the ABAP dictionary)

The data class describes the data format at the user interface.

If a table field or structure field or a data element is used in an ABAP program, the data class is converted to a format used by the ABAP processor. When a table is created in the database, the data class of a table field is converted to a corresponding data format of the database system used.

Use in the ABAP report: Checking for the data type TIMS.

Field CHECKTABLE (check table name of the foreign key)

Table whose key fields are used to check the foreign key fields. Only entries that are contained in the key fields of the check table can be contained in the foreign key fields. The check table is used to check whether the input values are valid and for the input help (F4 help).

Use in the ABAP report: All check tables found while crawling through the system are stored in an internal «table repository» which only exists during run time. Entries with an asterisk () and table T000 (Clients) are excluded. After having examined all tables of the repository the crawler terminates.*

Field PRECFIELD (name of included table)

This field contains the name of the include if one is used in a table.

Use in the ABAP report: Providing visual information on the output screen.

Criteria to be checked by the crawler

Field names (Table DD03L: FIELDNAME)

- UZEIT
- LOGTIME
- LTIME
- LAETM
- ERSTM
- *TIM*
- *ZEIT*

Data elements (Table DD03L: DOMNAME)

- UZEIT
- SYTIME
- LTIME
- *TIM*
- *ZEIT*

Data types (Table DD03L: DATATYPE)

- TIMS

Short description of the table (Table DD02T: DDTEXT, currently not in use)

- Tageszeit
- Erstellungsuhrzeit
- Uhrzeit der letzten Änderung
- Logische Uhrzeit
- Zeitereignisdaten
- Log
- *zeit*
- *time*

Content of the table (all fields of the current table)

Regular expressions in fields of the ABAP types C (alphanumeric characters), N (numeric characters) and string (character strings) with the following REGEX pattern (to be refined and extended):

- Minimum length of the sequence is 6 digits
- Sequence does not contain only the digits 0 and 1
- Sequenced is composed of
 - 1st digit: 0-2
 - 2nd digit: 0-9
 - 3rd and 4th digits: number 00-59
 - 5th and 6th digits: number 00-59

The following patterns should be detected by the REGEX statement:

- 074114 (regular time format)
- 776164 (inverted time format)
- A000010520003201308071604550001 (the sequence 20130807 represents the time stamp)

Date stamp criteria (currently not in use)

Possible checking criteria for the ABAP Process Mining Crawler (if needed at all) could be as follows.

Field names (Table DD03L: FIELDNAME)

- AEDAT
- AEDTM
- LAEDA
- ERSDA
- LOGDATE
- LDATE

Data elements (Table DD03L: DOMNAME)

- AEDAT
- SYDATS
- LDATE

Data types (Table DD03L: DATATYPE)

- DATS

Short description of the table (Table DD02T: DDTEXT, currently not in use)

- Änderungsdatum
- Datum der letzten Änderung
- Logisches Datum
- Erstellungsdatum
- Log
- *datum*
- *date*

Content of the table fields (all fields of the current table)

- 01012007 and 01.01.2007 (regular date format)
- 83928003 and 83.92.8003 (inverted date format)
- A000010520003201308071604550001 (the sequence 20130807 represents the date stamp)

Source code of the ABAP Process Mining Crawler

```
*&-----*
*& Report  ZZ_DEVB_41_TABELLENFINDER
*&
*&-----*
*& SAP ABAP-Crawler zur Identifizierung potentiell geeigneter Tabellen für Process
Mining
*&
*& Entwickelt im Rahmen des Seminars EBA bei Prof. Sigurd Schacht an der Hochschule
Heilbronn
*& Version 1.1 vom 01. Januar 2014
*& Autor: Marco Krapf
*&
*& Beschreibung des Crawlers siehe zugehörige Seminararbeit
*&
*& Infos zu dieser Version:
*& - Die "Suchtiefe" ist statisch auf 3 Ebenen ab der Starttabelle festgelegt
*& - Gefundene Includes werden nur angezeigt, aber nicht extra durchsucht (siehe
Kommentar im Code)
*& - Auskommentierte WRITE-Befehle können zur Überwachung des Crawlers aktiviert
werden
*&-----*

REPORT  ZZ_DEVB_41_TABELLENFINDER.

* Counter
DATA counterDurchsucht TYPE i. "Counter für die Anzahl der durchsuchten Tabellen
DATA counterFieldname TYPE i.
DATA counterDomname TYPE i.
DATA counterDatatype TYPE i.
DATA counterChecktable TYPE i.
DATA counterPrecfield TYPE i.
DATA counterInhalte TYPE i.
DATA counterAppendChecktable TYPE i.
DATA counterAppendPrecfield TYPE i.
DATA counterFoundChecktable TYPE i.
DATA counterFoundPrecfield TYPE i.
DATA counterFinishedChecktable TYPE i.
DATA counterFinishedPrecfield TYPE i.

* Regelung der Suchtiefe auf 3 Ebenen (das ist verbesserungswürdig!!)
DATA ebeneSuchtiefe TYPE i. "Variable die bestimmt, ob noch eine Ebene durchsucht
wird (Wert 0) oder nicht (Wert 1)
DATA letzteTabelle TYPE string. "Diese Variable markiert die letzte gefundene
Fremdschlüsseltabelle des ersten Durchlaufs

*----- Definitionen
DATA message1(100) TYPE c. "Message, wenn die gesuchte Tabelle nicht existiert

* Tabelle DD02L: Hier sind alle SAP-Tabellen drin, die es gibt (rund 430000 Stück).
Evtl. TABCLASS berücksichtigen.
DATA it_dd02l TYPE TABLE OF DD02L. "Interne Tabelle erstellen
DATA wa_dd02l LIKE LINE OF it_dd02l. "Arbeitsbereich erstellen

* Tabelle DD02T
DATA it_dd02t TYPE TABLE OF DD02T. "Interne Tabelle erstellen
DATA wa_dd02t LIKE LINE OF it_dd02t. "Arbeitsbereich erstellen

* Tabelle DD03L
DATA it_dd03l TYPE TABLE OF DD03L. "Interne Tabelle erstellen
DATA wa_dd03l LIKE LINE OF it_dd03l. "Arbeitsbereich erstellen

* Dynamisch, jeweils die gefundene Tabelle
DATA it_regex TYPE TABLE OF MATCH_RESULT. "Interne Tabelle erstellen für die
Fundstellen des REGEX
```

```
DATA wa_regex LIKE LINE OF it_regex. "Arbeitsbereich erstellen für die Fundstellen
des REGEX
DATA describeTyp TYPE c. "Variable für die DatenTypBestimmung, da FIND nur mit
zeichenartigen Datentypen geht

* Strukturen für Speicherung und Verarbeitung von Checktables
DATA it_checktable TYPE STANDARD TABLE OF string. "Interne Tabelle erstellen
DATA wa_checktable LIKE LINE OF it_checktable. "Arbeitsbereich erstellen

* Strukturen für Speicherung und Verarbeitung von Precfields
DATA it_precfield TYPE STANDARD TABLE OF string. "Interne Tabelle erstellen
DATA wa_precfield LIKE LINE OF it_precfield. "Arbeitsbereich erstellen

* Strukturen für Speicherung und Verarbeitung von Fremdschlüssel- und inkludierten
Tabellen
DATA it_found TYPE STANDARD TABLE OF string. "Interne "Sammel"Tabelle erstellen für
Tabelle, die noch durchsucht werden müssen
DATA wa_found LIKE LINE OF it_found. "Arbeitsbereich erstellen

* Tabelle, in die alle schon durchsuchten Tabellen geschrieben werden
DATA it_finished TYPE STANDARD TABLE OF string.
DATA wa_finished LIKE LINE OF it_finished.
wa_finished = 'HAPPY NEW YEAR'. "Initialer Eintrag
APPEND wa_finished TO it_finished.

* Tabelle, in der alle für Process Mining geeigneten Tabellen gesammelt werden für
die Zusammenfassung
DATA it_positive TYPE STANDARD TABLE OF string.
DATA wa_positive LIKE LINE OF it_positive. "Arbeitsbereich für die Ausgabe der
Tabelleninhalte in der Zusammenfassung

* Aktuelle Tabelle
DATA wa_aktuell TYPE string.

*----- Variablen für die dynamischen Tabellen anlegen
DATA ref_struct TYPE REF TO data.
FIELD-SYMBOLS: <fs_struct> TYPE any, <fs_component> TYPE any.
*-----

*----- Name der Tabelle eingeben, nach der zuerst gesucht werden soll
PARAMETERS suchtab TYPE string OBLIGATORY. "Eingabefeld für die gesuchte Tabelle
APPEND suchtab TO it_found. "An "Sammel"Tabelle anhängen
ebeneSuchtiefe = 1. "Zähler auf erste Ebene stellen

FORMAT COLOR = 7.
WRITE: / 'START DER SUCHE MIT TABELLE', suchtab.
WRITE: / .
FORMAT COLOR = OFF.

LOOP AT it_found INTO wa_aktuell.

*****
*****
*----- Hier startet die Durchsuchung einer Tabelle -----
*-----
*----- Gesuchte Datenbanktabelle auslesen
SELECT * FROM DD02L INTO TABLE it_dd02l WHERE tabname = wa_aktuell. "Alle
Informationen über die gefundene Tabelle in die interne Tabelle schreiben

*----- Gefundene Tabelle untersuchen und interessante Fundstellen ausgeben
IF sy-subrc = 0. "Wenn die Tabelle existiert
    FORMAT COLOR = 1.
    WRITE: / 'TABELLE', wa_aktuell, 'WIRD UNTERSUCHT.'.
    FORMAT COLOR = OFF.
    WRITE: / 'Aktuelle Suche Ebene ausgehend von der Starttabelle:', ebnSuchtiefe.
    PERFORM tabellenartAuslesen.           "Unterprogramm aufrufen -> DD02L
    PERFORM tabellenbeschreibungAuslesen.  "Unterprogramm aufrufen -> DD02T
    PERFORM spalteninfosAuslesen.          "Unterprogramm aufrufen -> DD03L
```

```
PERFORM gefundeneTabelleAuslesen.      "Unterprogramm aufrufen -> Dynamisch,
jeweils die gefundene Tabelle
  counterDurchsucht = counterDurchsucht + 1. "Hochzählen der durchsuchten Tabellen
  DELETE TABLE it_found FROM wa_aktuell. "Tabelle die gerade durchsucht wurde aus
  der "Sammel"Tabelle löschen
  APPEND wa_aktuell TO it_finished. "Tabelle die gerade durchsucht wurde eintragen
  in Tabelle der schon durchsuchten Tabellen eintragen
*----- Fazit der Tabelle ausgeben
  WRITE: / .
  IF counterFieldname = 0 AND counterDomname = 0 AND counterDatatype = 0 AND
  counterInhalte = 0.
    FORMAT COLOR = 6.
    WRITE: / 'FAZIT: Tabelle', wa_aktuell, 'scheint nicht geeignet für Process
Mining.'.
    FORMAT COLOR = OFF.
  ELSE.
    FORMAT COLOR = 5.
    WRITE: / 'FAZIT: Tabelle', wa_aktuell, 'ist potentiell geeignet für Process
Mining.'.
    FORMAT COLOR = OFF.
    APPEND wa_aktuell TO it_positive. "Tabellennamen an die Tabelle mit allen
geeigneten Tabellen anfügen
  ENDIF.

ELSE. "Wenn die Tabelle nicht existiert
  CONCATENATE wa_aktuell ' ist keine SAP-Datenbanktabelle.' INTO message1.
  MESSAGE message1 TYPE 'I'.
ENDIF.

*----- Informationen nach dem Durchsuchen...
WRITE: / .
WRITE: / '***** Bisher wurden', counterDurchsucht, 'Tabellen durchsucht.', / .
*WRITE: / 'Letzte Tabelle der zweiten Ebene:', letzteTabelle, '(auskommentiert, nur
zur Überwachung)'.
IF it_found IS INITIAL. "Beenden wenn alle Tabellen abgearbeitet sind
  WRITE: / .
  WRITE: / '***** Die Suche wurde nach', ebeneSuchtiefe, 'Suchebenen
abgeschlossen.', / .
  WRITE: / .
  WRITE: / '-----'
  -----'.
  WRITE: / .
  FORMAT COLOR = 7.
  WRITE: / 'ZUSAMMENFASSUNG', / .
  FORMAT COLOR = OFF.
  WRITE: / 'Insgesamt wurden', counterDurchsucht, 'Tabellen durchsucht.', / .
  WRITE: / 'Folgende Tabellen wurden ausgehend von Tabelle', suchtab, 'als
potentiell geeignet für Process Mining identifiziert:'.
  LOOP AT it_positive INTO wa_positive. "Alle gefundenen Treffer durchlaufen
    SELECT * FROM DD02T INTO TABLE it_dd02t WHERE tabname = wa_positive AND
ddlanguage = 'D'. "Tabellenbeschreibung auslesen
    LOOP AT it_dd02t INTO wa_dd02t.
      ENDLOOP.
    WRITE: /5 wa_positive, 15 wa_dd02t-ddtext. "Tabellenname und
Tabellenbeschreibung ausgeben
  ENDLOOP.
  WRITE: / .
  WRITE: / '(c) Marco Krapf, Hochschule Heilbronn'.
  EXIT. "SuchSchleife verlassen und Anzeige stehen lassen, Programm ist quasi
fertig
ENDIF.
*WRITE: / .
*WRITE: / 'Diese Tabellen stehen noch aus... (auskommentiert, nur zur
Überwachung)'. "Anzeige der Tabellen, die noch durchsucht werden
*LOOP AT it_found INTO wa_found.
*  WRITE: /5 wa_found.
*ENDLOOP.
WRITE: / .
```



```
WRITE: / '-----'
-----'.
WRITE: / .
IF wa_aktuell = letzteTabelle.
    ebeneSuchtiefe = 3. "Zähler auf dritte Ebene stellen
ENDIF.
*----- Hier endet die Durchsuchung einer Tabelle -----
*****
*****

ENDLOOP.

*&-----*
*&      Form  tabellenartAuslesen
*&-----*
*      DD02L
*-----*
FORM tabellenartAuslesen.

    LOOP AT it_dd02l INTO wa_dd02l. "Tabelle durchlaufen und in Arbeitsbereich
schreiben
        WRITE: /, / '----- GEFUNDEN IN TABELLE DD02L (SAP-Tabellen)', /.
        WRITE: / 'Tabellenname: ', wa_dd02l-tabname, / 'Tabellenart : ', wa_dd02l-
tabclass, /.
    ENDLOOP.

ENDFORM.                "tabellenartAuslesen

*&-----*
*&      Form  tabellenbeschreibungAuslesen
*&-----*
*      DD02T
*-----*
FORM tabellenbeschreibungAuslesen.

    SELECT * FROM DD02T INTO TABLE it_dd02t WHERE tabname = wa_aktuell AND ddlanguage
= 'D'.

    LOOP AT it_dd02t INTO wa_dd02t.
    ENDLOOP.

    WRITE: / '----- GEFUNDEN IN TABELLE DD02T (R/3-DD: Texte zu SAP-Tabellen)', /.
    WRITE: / 'Tabellenbeschreibung: ', wa_dd02t-ddtext, /.

ENDFORM.                "tabellenbeschreibungAuslesen

*&-----*
*&      Form  spalteninfosAuslesen
*&-----*
*      DD03L
*-----*
FORM spalteninfosAuslesen.

    "Counter zurücksetzen
    counterFieldname = 0.
    counterDomname = 0.
    counterDatatype = 0.
    counterChecktable = 0.
    counterPrecfield = 0.
    counterFinishedChecktable = 0.
    counterFinishedPrecfield = 0.
    counterFoundChecktable = 0.
    counterFoundPrecfield = 0.
    counterAppendChecktable = 0.
    counterAppendPrecfield = 0.
```

```
"Interne Tabellen des Unterprogramms zurücksetzen
CLEAR it_checktable.
CLEAR it_precfield.

SELECT * FROM DD03L INTO TABLE it_dd03l WHERE tabname = wa_aktuell.

WRITE: / '----- GEFUNDEN IN TABELLE DD03L (Tabellenfelder)', /.

"1) Check auf Feldname
WRITE: / 'Folgende Feldnamen der Tabelle', wa_aktuell, 'könnten geeignet sein:'.
LOOP AT it_dd03l INTO wa_dd03l.
  "Check auf brauchbare Feldnamen
  IF wa_dd03l-fieldname = 'UZEIT'
    OR wa_dd03l-fieldname = 'LOGTIME'
    OR wa_dd03l-fieldname = 'LTIME'
    OR wa_dd03l-fieldname = 'LAETM'
    OR wa_dd03l-fieldname = 'ERSTM'
    OR wa_dd03l-fieldname CP '*TIM*' "...contains pattern
    OR wa_dd03l-fieldname CP '*ZEIT*'.
  WRITE: /5 wa_dd03l-fieldname.
  counterFieldname = counterFieldname + 1.
ENDIF.
ENDLOOP.
IF counterFieldname = 0.
  WRITE: /5 '(keine)'.
ENDIF.
WRITE: /.

"2) Check auf Domäne
WRITE: / 'Folgende Domänenbezeichnungen der Tabelle', wa_aktuell, 'könnten
geeignet sein:'.
LOOP AT it_dd03l INTO wa_dd03l.
  "Check auf brauchbare Datenelemente
  IF wa_dd03l-domname = 'UZEIT'
    OR wa_dd03l-domname = 'SYTIME'
    OR wa_dd03l-domname = 'LTIME'
    OR wa_dd03l-fieldname CP '*TIM*' "...contains pattern
    OR wa_dd03l-fieldname CP '*ZEIT*'.
  WRITE: /5 wa_dd03l-domname.
  counterDomname = counterDomname + 1.
ENDIF.
ENDLOOP.
IF counterDomname = 0.
  WRITE: /5 '(keine)'.
ENDIF.
WRITE: /.

"Check auf Datentyp
WRITE: / 'Folgende Datentypen des ABAP Dictionaries der Tabelle', wa_aktuell,
'könnten geeignet sein:'.
LOOP AT it_dd03l INTO wa_dd03l.
  "Check auf brauchbare Datentypen
  IF wa_dd03l-datatype = 'TIMS'.
    WRITE: /5 wa_dd03l-datatype.
    counterDatatype = counterDatatype + 1.
  ENDIF.
ENDLOOP.
IF counterDatatype = 0.
  WRITE: /5 '(keine)'.
ENDIF.
WRITE: /.

"3) Check auf Fremdschlüsselbeziehungen
WRITE: / 'Die Tabelle', wa_aktuell, 'hat folgende Fremdschlüsselbeziehungen:'.
LOOP AT it_dd03l INTO wa_dd03l.
  "Tabelle T000 (Mandanten) und Wildcard * ausschließen
```

```
IF wa_dd03l-checktable <> '' AND wa_dd03l-checktable <> '*' AND wa_dd03l-
checktable <> 'T000'.
"Gefundene Tabelle darf nur an die Tabelle it_checktable angefügt werden,
wenn sie nicht schon drin ist (Keine Duplikate!)
IF it_checktable IS INITIAL. "Beim ersten mal auf jeden Fall anfügen, wenn
die Tabelle noch leer ist
APPEND wa_dd03l-checktable TO it_checktable.
ENDIF.
LOOP AT it_checktable INTO wa_checktable. "Prüfen, ob der Eintrag schon drin
ist
IF wa_checktable = wa_dd03l-checktable.
counterAppendChecktable = counterAppendChecktable + 1.
ENDIF.
ENDLOOP.
IF counterAppendChecktable = 0.
APPEND wa_dd03l-checktable TO it_checktable. "An Tabelle anfügen
ENDIF.
counterAppendChecktable = 0. "Zähler zurücksetzen für nächsten Durchlauf
counterChecktable = counterChecktable + 1. "Zähler zeigt an, dass eine
Tabelle gefunden wurde (nur für die optische Ausgabe)
ENDIF.
ENDLOOP. "Tabelle it_checktable ist gefüllt mit allen Funden ohne Duplikate
LOOP AT it_checktable INTO wa_checktable.
WRITE: /5 wa_checktable.
"Gefundene Tabelle darf nur an die Sammeltabelle it_found angefügt werden, wenn
sie nicht schon drin ist und auch noch nicht durchsucht wurde
LOOP AT it_finished INTO wa_finished.
IF wa_finished = wa_checktable.
counterFinishedChecktable = counterFinishedChecktable + 1.
ENDIF.
ENDLOOP.
IF counterFinishedChecktable = 0.
LOOP AT it_found INTO wa_found.
IF wa_found = wa_checktable.
counterFoundChecktable = counterFoundChecktable + 1.
ENDIF.
ENDLOOP.
IF counterFoundChecktable = 0 AND ebeneSuchtiefe < 3. "In dieser Version wird
mit der AND-Bedingung die Suchtiefe auf 3 Ebenen begrenzt !!!!!!!!!!!!!!!
APPEND wa_checktable TO it_found. "Tabellennamen zu Sammeltabelle zufügen
ENDIF.
ENDIF.
counterFoundChecktable = 0. "Zähler zurücksetzen für nächsten Durchlauf
counterFinishedChecktable = 0. "Zähler zurücksetzen für nächsten Durchlauf
ENDLOOP.
"Hier wird das Ende des ersten Suchlaufs nach Fremdschlüsseltabellen markiert
IF ebeneSuchtiefe = 1.
letzteTabelle = wa_checktable. "Zuweisen der letzten gefundenen
Fremdschlüsseltabelle
ebeneSuchtiefe = 2. "Blockieren, damit hier nicht mehr reingesprungen wird und
Ebene auf zwei stellen
ENDIF.
IF counterChecktable = 0.
WRITE: /5 '(keine)'.
ENDIF.
WRITE: /.

"4) Check auf Includes
WRITE: / 'Die Tabelle', wa_aktuell, 'hat folgende Tabellen inkludiert:'.
LOOP AT it_dd03l INTO wa_dd03l.
IF wa_dd03l-precfield <> ''.
APPEND wa_dd03l-precfield TO it_precfield. "An Tabelle anfügen
counterPrecfield = counterPrecfield + 1. "Zähler zeigt an, dass eine Tabelle
gefunden wurde (nur für die optische Ausgabe)
ENDIF.
ENDLOOP.
LOOP AT it_precfield INTO wa_precfield.
WRITE: /5 wa_precfield.
```

```

"
+++++
+++++
" ++++++ Hier sollte genau wie zuvor bei den
CHECKTABLES vorgegangen werden ++++++
" +++++ ist aber noch nicht programmiert, die inkludierten Tabellen werden noch
nicht an die Tabelle it_found angefügt +++++
*   APPEND wa_precfield TO it_found. "Tabellennamen zu Sammeltabelle zufügen
"
+++++
+++++
    ENDLOOP.
    IF counterPrecfield = 0.
        WRITE: /5 '(keine)'.
    ENDIF.
    WRITE: /.

ENDFORM.                                "spalteninfosAuslesen

*&-----*
*&      Form   gefundeneTabelleAuslesen
*&-----*
*      Dynamisch, jeweils die gefundene Tabelle
*-----*
FORM gefundeneTabelleAuslesen.

"Counter zurücksetzen
counterInhalte = 0.

TRY. "Nur ausführen wenn eine "gültige" Tabelle angegeben wurde

    CREATE DATA ref_struct TYPE (wa_aktuell).
    ASSIGN ref_struct->* TO <fs_struct>.

    WRITE: / '----- GEFUNDEN IN TABELLE', wa_aktuell, /.
    WRITE: / 'Möglicherweise interessante Inhalte der Tabelle', wa_aktuell, ':'.

    SELECT SINGLE * FROM (wa_aktuell) INTO <fs_struct>. "Erste Zeile der Tabelle
    auslesen (reicht aus, da alle Zeilen die gleiche Datenstruktur haben

    DO. "Schleife, die alle Felder des Datensatzes durchläuft
        ASSIGN COMPONENT sy-index OF STRUCTURE <fs_struct> TO <fs_component>.
        IF sy-subrc <> 0.
            NEW-LINE.
            EXIT.
        ENDIF.
*      WRITE /5 <fs_component>.                                "/// nur zum Test ///

        DESCRIBE FIELD <fs_component> TYPE describeTyp. "Datentyp ermitteln und in
        Variable speichern

        "----- Check auf Datentyp T (Time) -----
        IF describeTyp = 'T' AND <fs_component> <> 0. "Wenn Feld vom Datentyp T
        (TIME) und Wert nicht 00:00:00
            WRITE: /5 'Feld vom Datentyp T (Time)', '/ Inhalt:', <fs_component>.
            counterInhalte = counterInhalte + 1.
        ENDIF.
        "----- Check auf brauchbares Muster (REGEX) -----
        "Nur Feldinhalte mit den Datentypen Char, Number und String durchsuchen
        IF describeTyp = 'C' OR describeTyp = 'N' OR describeTyp = 'string'.
*      WRITE: '/// Datentyp:', describeTyp.                                "/// nur zum
Test ///

        FIND ALL OCCURRENCES OF REGEX '[0-2]+[0-9]+[00-59]+[00-59]' IN
        <fs_component> IGNORING CASE RESULTS it_regex. " !!! Muster muss eventuell
        überarbeitet werden

```

```
IF sy-subrc = 0 AND <fs_component> CN '0' AND <fs_component> CN '[0-1]'
AND strlen( <fs_component> ) >= 6. "Wenn ein Feld dem REGEX entspricht, mind. 6
Zeichen lang ist und nicht nur 0 und 1 enthält
    WRITE: /5 'Feld vom Datentyp', describeTyp, '/ Inhalt:',
<fs_component>.
    WRITE: /10 'Interessante Fundstellen:'.
    LOOP AT it_regex INTO wa_regex.
        WRITE: /10 '- ', <fs_component>+wa_regex-offset(wa_regex-length).
"Ausgabe ab Fundstelle (offset) mit Länge (length)
    ENDLOOP.
    counterInhalte = counterInhalte + 1.
ENDIF.
ENDIF.
ENDDO.

IF counterInhalte = 0.
    WRITE: /5 '(keine)'.
ENDIF.

CATCH cx_root.
    MESSAGE 'Tabelle irgendwie ungültig... (ohne try/catch fliegen wir hier
raus).' TYPE 'I'.

ENDTRY.
ENDFORM.                "gefundeneneTabelleAuslesen
```